Alibaba Group 阿里巴巴集团 | 淘系技术部 TAO TECHNOLOGY

技术人的百宝黑皮书

阿里淘系技术2020技术年货

本书包含

- 淘系一年实战经验&解决方案
- 阿里技术人社区精彩问答
- 推荐书单、最新 PAPER、开源项目等

前言

在技术进步的巨变时代，价值书写与思考沉淀至关重要。

过去的一年多，新零售业务快速发展，淘宝发布躺平，重塑大家居行业电商模式；云原生浪潮大起，核心交易系统百分之百上云；端侧 AI 实时智能决策，推理引擎 MNN 开源；音视频实时通信引擎将直播延时压至1 秒以内，淘宝直播缔造全球内容电商场的技术顶峰……

新场景、新用户动线和新架构不断提出新挑战，新产品、新技术以及实时自我更新的淘系技术人，通过一次次思考和技术升级不断去逼近能够做到的最好的极限。

交易数字曾一次次俘获人们的注意力，我们认为，那些在背后耕耘与付出，支撑着全球领先的线上新零售技术平台的技术不应被掩盖在历史的备忘录中。

所以，我们相遇了。

这里是阿里巴巴淘系技术 2020一整年的精华技术内容合集，涵盖了大前端、音视频、端智能、用户增长、客户端架构、服务端架构、云原生、技术质量、以及 AI 类等多个技术领域，沉淀了 618、双 11大促、阿里拍卖、淘宝直播等多个业务的技术解决方案，并有淘系工程师推荐书单、开源项目、顶会 paper 等多角度知识与价值的输出。

本书内容页数 1500+，全部内容将近 40w 字，我们怀抱着开放自由的交流心态，输出阿里淘系工程师对技术和商业的理解，并真诚期望能够在更丰富的场景中与大家沟通交流，共同探讨未来技术与商业的形态。

希望你喜欢，并分享给身边的朋友。

新年快乐！

扫一扫，关注公众号【淘系技术】

不错过下一年的技术风景

扫码关注阿里藏经阁

获得更多电子书下载

目录

第一部分 年度精选技术栈内容……9

◉前端篇……10

●技术经典总结……10

混搭 TypeScript + GraphQL + DI + Decorator 风格写 Node.js 应用……10

移动前端开发和 Web 前端开发的区别是什么？……35

如何用 SVG 实现一个优雅的提示框？……45

10 个 VS Code 使用技巧……60

一套代码帮助前端开发了解机器学习……72

闲鱼 Weex 页面优化过程详解……87

智能 UI：面向未来的 UI 开发技术……99

TypeScript 4.0 正式发布：核心概念梳理……113

一张图讲清楚淘宝直播背后的前端技术……134

●前端人精彩问答……146

Q：大公司里怎样开发和部署前端代码？……146

Q：Nodejs 的 websocket 的服务器端是如何实现的？……159

Q：网页布局都有哪种？一般都用什么布局？……178

Q：后端同学是不是比前端同学理解业务更快？……241

Q：如何看待 Web 开发构建工具 Vite？……251

Q：2020 年前端最火的技术是什么？……270

Q：如何看待华为云 Cloud IDE 和 VSCode online 高度相似？……280

◉后端篇……285

●技术经典总结……285

如何正确地实现重试(Retry)……285

面向 5G 的阿里自研标准化协议库 XQUIC……295

设计模式最佳套路 —— 愉快地使用策略模式……313

电商互动消息如何进行架构演进？……323

闲鱼基于 Flutter+FaaS 的业务框架思考与实践……333

阿里技术专家详解DDD系列 第一讲 - Domain Primitive……339

阿里技术专家详解DDD系列 第二讲 - 应用架构……361

阿里技术专家详解DDD系列 第三讲 - Repository 模式……390

阿里技术专家详解DDD系列 第四讲 - 领域层设计规范……416

●后端人精彩问答……450

Q：CDN 是什么？使用 CDN 有什么优势？……450

Q：后端真的有这么难吗？……464

◉算法篇……473

●技术经典总结……473

如何使用深度学习识别 UI 界面组件？ ····· 473
任务调度：时间轮算法经典案例解析及应用实现 ····· 505
详解 Graph Embedding 经典方法：算法原理、代码实现与应用样例 ····· 531
内容推荐算法：异构行为序列建模探索 ····· 571
推荐算法三视角:矩阵，图，时间线 ····· 584

●算法人精彩问答 ····· 595
Q: GAN 的生成器是怎么产生图片的？ ····· 595
Q: 主动学习(Active learning)算法的原理是什么，有哪些比较具体的应用？ ····· 627
Q: 在你做推荐系统的过程中都遇到过什么坑？ ····· 648

◉客户端篇 ····· 652
●技术经典总结 ····· 652
Flutter 和 Web 生态如何对接？ ····· 652
手淘如何突破 native crash 提升稳定性？ ····· 665
手淘引入 Swift :从探索到落地过程实践 ····· 674
客户端稳定性优化实战，Crash 率最高下降 40% ····· 685
IOS 开发体验优化方案 ····· 694
IOS Crash 分析攻略 ····· 702
手淘图片库新特性解析 ····· 717

●客户端精彩问答 ····· 725
Q: Node.js 做 Web 后端优势为什么这么大？ ····· 725
Q: 2020 年了，大家怎么评价 Swift？ ····· 735

◉测试篇 ····· 746
●技术经典总结 ····· 746
智能化测试技术探索与实践 ····· 746
手淘再推新利器 Holmes：一站式智能化异常检测平台 ····· 756
首次面向 B 端展开全链路压测！淘系高难度压测实践方案公开 ····· 766

●测试人精彩问答 ····· 780
Q: 各位都是怎么进行单元测试的？ ····· 780
Q: 如何学习自动化测试？ ····· 799

◉音视频与图像技术篇 ····· 806
●技术经典总结 ····· 806
支撑每秒千万图像访问的鹿班 PICASSO 实时渲染引擎 ····· 806
让机器读懂视频：亿级淘宝视频背后的多模态 AI 算法 ····· 826
AceRay 极速 3D 渲染: 预览速度提升 30 倍 ····· 836
3D 购物时代：实景逛街背后的技术实现 ····· 848
打破传统降噪算法：AliDenoise 的背后技术 ····· 861
5G 时代|淘宝直播高画质低延时技术探索 ····· 869

●音视频精彩问答 ····· 888
Q: WebRTC 有前途吗？ ····· 888

◉端智能与 MNN 篇…… 903
●技术经典总结…… 903
阿里端侧 AI 从探索尝试到逐步展开…… 903
MNN 正式版发布：模型训练、异构性能和 Python 工具链…… 908
端计算在移动设备上的应用探索……918
端智能与前端 AI 探索…… 926
聊一聊 MNN 在端智能生态中的应用…… 937

●端智能精彩问答…… 943
Q: 如何评价阿里开源的移动端深度学习推理框架 mnn？…… 943

◉淘系 Flutter 探索实践篇……954
●技术经典总结…… 954
Flutter 在阿里淘系的体系化建设和业务实践…… 954
Flutter 快速学习和入门方法…… 965
前端工程师学习 flutter 的方法经验…… 983
对比详解 Flutter Widget 和 CSS 布局原理……1006
Flutter 渲染引擎：底层原理解析……1018
Flutter页面卡顿原因详解……1034

●Flutter 精彩问答…… 1048
Q: Flutter 凉了吗？…… 1048

◉淘系实战项目篇……1066
双 11 总结：2020 天猫双 11 前端体系的建设与挑战…… 1066
双 11 总结：淘宝直播“一猜到底”——移动端实时语音识别技术方案及应用……1078
双 11 总结：阿里亿级长连网关的云原生演进之路…… 1088
双 11 总结：手淘双 11 最新实践：PopLayer 弹层领域研发模式升级……1101
向量召回在躺平 APP 的实践…… 1118
阿里拍卖全链路导购策略首次揭秘……1130
淘系娱乐化互动技术经验总结……1155
DinamicX：手淘信息无障碍技术方案全解……1174

◉2020 技术资讯&观点解读…… 1189
2020 双 11 技术总指挥汤兴：今年双 11 是技术和商业的完美共振……1189
WWDC 20 Core Image：iOS 图像处理技术追踪…… 1196
WWDC 20 无线网络优化实践，带来哪些启发？…… 1209
WWDC 20 Apple Widget：下一个顶级流量入口？…… 1226
WWDC 20 Swift 5.3 又更新了什么新奇爽快的语法？…… 1239
WWDC 20 Metal 新特性：大幅度提升 iOS 端性能……1248
WWDC 20 iOS14 隐私适配及部分解决方案……1268
WWDC 20 详解 SwiftUI 的重大改变及核心优势……1285
2020 前端开源领域技术展望…… 1303
阿里专家跨平台思路：Weex、RN 还是 Flutter？……1307

第二部分 职场成长技巧与解惑……1313

◉职场成长技巧&故事……1314
设计转研发，台湾程序员的成长之路……1314
7 年晋升 3 次，他如何摆脱业务前端的职业迷惑……1319
校招实习生4 年升 P7，她想分享的 3 个技能与 2 个反思……1329
关于读书你必须要知道的 3 种方法……1340
极简而高效的沟通管理法……1350

◉一线职场&学习问答实录……1374
Q: 如何写出优雅的代码？……1374
Q: 如何看待 2021 年秋招算法岗灰飞烟灭？……1384
Q: 前端 p6 什么水平？如何衡量？……1391
Q: 做了两年前端开发，平时就是拿 Vue 写写页面和组件，简历的项目经历应该怎么写得好看？……1402
Q: 一份优秀的前端开发工程师简历是怎么样的？……1410
Q: 想学习推荐系统，如何从小白成为高手？……1418
Q: 你写论文时发现了哪些非常神的网站？……1428
Q: 2020 年 WEB 前端学习顺序？……1436

第三部分 年度精选必读好书……1450

◉技术硬核参考……1451
◉商业思维养成……1452
◉文化&科普……1453
◉工具&方法……1454

第四部分 淘系开源项目盘点……1455

第五部分 2020 淘系 A 类顶会论文收录……1463

◉大数据与数据挖掘领域……1464
●ICDE 2020……1464
Two-Level Data Compression using Machine Learning in Time Series Database……1464
Cross-Graph Convolution Learning for Large-Scale Text-Picture Shopping Guide in E-Commerce Search……1476
PoisonRec: An Adaptive Data Poisoning Framework for Attacking Black-box Recommender Systems……1486

●KDD 2020……1498
Prediction and Profiling of Audience Competition for Online Television Series……1498

A Dual Heterogeneous Graph Attention Network to Improve Long-Tail Performance for Shop Search in E-Commerce.....1508
Learning to Generate Personalized Query Auto-Completions via a Multi-View Multi-Task Attentive Approach.....1519

◉信息检索领域.....1529
●SIGIR 2020.....1529
ESAM: Discriminative Domain Adaptation with Non-Displayed Items to Improve Long-Tail Performance.....1529
ATBRG: Adaptive Target-Behavior Relational Graph Network for Effective Recommendation.....1539
GMCM: Graph-based Micro-behavior Conversion Model for Post-click Conversion Rate Estimation.....1549
Entire Space Multi-Task Modeling via Post-Click Behavior Decomposition for Conversion Rate Prediction.....1559

●CIKM 2020.....1569
Deep Time-Aware Item Evolution Network for Click-Through Rate Prediction.....1569

◉移动计算领域.....1579
●MobiCom 2020.....1579
OnRL: Improving Mobile Video Telephony via Online Reinforcement Learning.....1579
Billion-Scale Federated Learning on Mobile Clients: A Submodel Design with Tunable Privacy.....1593

◉数据库领域.....1607
●VLDB 2020.....1607
EdgeDIPN: a Unified Deep Intent Prediction Network Deployed at the Edge.....1607

◉机器学习和计算神经科学领域.....1616
●NIPS 2020.....1616
Hard Example Generation by Texture Synthesis for Cross-domain Shape Similarity Learning.....1616
Neuron-level Structured Pruning using Polarization Regularizer.....1629

◉人工智能综合领域.....1646
●AAAI 2020.....1646
Multi-Question Learning for Visual Question Answering.....1646

◉交叉/综合/新兴领域.....1654
●WWW 2020.....1654
Adversarial Multimodal Representation Learning for Click-Through Rate Prediction....1654
Large-scale Causal Approaches to Debiasing Post-click Conversion Rate Estimation with Multi-task Learning.....1664

第一部分 年度精选技术栈内容

- 前端篇
- 后端篇
- 算法篇
- 客户端篇
- 测试篇
- 音视频与图像技术篇
- 端智能与 MNN 篇
- 淘系 Flutter 探索实践篇
- 淘系实战项目篇
- 2020 技术资讯&观点解读

前端篇

淘系前端在支撑淘宝、天猫这样复杂多变的业务场景的过程中，建立了前端工程、多端架构、Node 架构、多媒体、互动、搭建、智能化等多个基础技术体系，并还在不断的演进和升级；同时也将其中部分通过开源项目和技术产品的方式，回馈到社区中，共同繁荣前端技术生态；本书系列侧重的从前端与客户端结合突破多端体系，与云原生结合面向 Serverless 体系，与 AI 能力结合建立前端智能化应用场景这三个维度出发，向大家介绍一下我们的技术方案以及背后的思考。

——阿里巴巴资深前端技术专家 展炎

技术经典总结

混搭 TypeScript + GraphQL + DI + Decorator 风格写 Node.js 应用

作者 | JSCON
出品 | 淘系技术

阅读本文的知识前提：熟悉 TypeScript + GraphQL + Node.js + Decorator + Dependency Inject 等概念。

前言

恰逢最近需要编写一个简单的后端 Node.js 应用，由于是全新的小应用，没有历史包袱，所以趁着这次机会换了一种全新的开发模式：

- 语言使用 TypeScript，不仅仅是强类型那么简单，它还提供很多高级语法糖，提高编程效率。
- 兼顾 Restful + GraphQL 方式提供数据接口，前两年 GraphQL 特别流行，最近这段时间有些平淡下来（现在比较火热的是 Serverless）；GraphQL 这种查询语言对前端来讲还是很友好的，自己写的话能减少不少的接口开发量。
- 使用 Decorator（装饰器语法） + DI（依赖注入）风格写业务逻辑。因后端 Java 开发服务的模式已经非常成熟，前端在 Node.js 的开发模式基本上是依照 Java 那套开发模子来的，尤其是 DI（依赖注入）设计模式的编程思想。这几年随着 ECMAScript 的标准迭代，以及 TypeScript 的成熟发展，在语言层面提供了很多现代化语法糖的支持，现在也可以利用 Decorator（装饰器）+ DI（依赖注入）风格来写了，个人认为这种风格也将成为书写 Node.js 应用的常用范式之一。
- 选用支持 TS + Decorator + DI 的 Node.js 框架。在集团内使用 Midway，因为 Midway 在集团内部已经是事实标准了，而且发展得很成熟了；如果选非集团内部的话，可以考虑选流行的 Next.js 框架；—— 这类框架功能都很强大，而且提供完善的工具链和生态，就算你不熟，通读他们的官方文档都能收获很多。

前端内部写的后端应用基本上功能并不会太多（太专业的后端服务交给后端开发来做），绝大部分是基础的操作，在这样的情况下会涉及到很多重复工作量要做，基本都是一样的套路：

1. 初始化项目脚手架
2. 数据库的连接操作 + CRUD 操作
3. 创建数据 model 层 + service 层
4. 提供诸如 Restful 接口供多端消费
5. ...

复杂的业务逻辑功能一般是直接调用后端提供的服务，前端很少介入太深的后端功能开发。

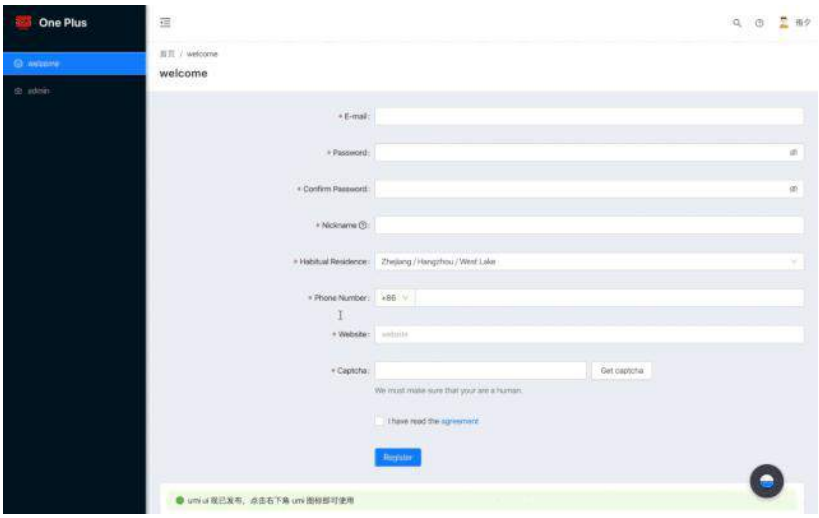
这意味着每次开发新应用都得重新来一遍 —— 这就跟前端平时切页面一样，重复劳动多了之后就内心还是比较烦的，甚至有抗拒心理。繁琐的事大概涉及在工程链路 & 业务代码这么两方面，如果有良好的解决方案，将大大提升开发的幸福感：

1. 第一个方面是结构目录的生成。这个问题比较好解决，市面上成熟的框架（Nest.js，Midway.js，Prisma.io 等）都提供了相应的脚手架工具，直接生成相应的服务端代码结构，写代码既可靠又高效。同时这类成熟框架都能一键搞定部署发布等流程，这样我们就可以将大部分时间用在业务代码上、而不是折腾环境搭建细节上。
2. 第二个方面是业务代码的书写风格。同样是写业务代码，语言风格不一样，代码效率也是不同的，你用 JS 写业务代码，跟 TypeScript + Decorator 来写的效率大相径庭

—— 这也就是技术发展带来的福利。

这里的第一个方面中的目录生成，一键生成支 Midway 6 ts 和 Ant Design Pro4 ts 的版本前后端目录，在这种强强联合下，几行代码下来就能生成非常专业高效的前后端分离的架构体系 —— 包含 client & server 两个文件夹，分别对应标准的 Ant DesignPro 4 目录结构 和 Midway6 TS 目录结构 目录：

然后在最外层根目录执行即可，启动后使用 http://localhost:6001/ 打开提示即可：



点击查看动图演示

该初始化项目后就能可以跑通本地开发调试、构建、aone 部署发布流程，整个很顺畅。

不过这里需要说明的是，这套方案里使用的是 antd 3.x 的版本，建议进行升级到 antd 4.x 版本，升级成本并不算很高，antd 官方提供升级工具，基本一行代码就能搞定：

```
# 通过 npx 直接运行
npx -p @ant-design/codemod-v4 antd4-codemod client/src
```

建议通读一下 官方文档 - 从 v3 到 v4 内容，了解升级的地方在哪些。

本文着重讲解第二部分，即如何使用 TypeScript + Decorator + DI 风格编写 Node.js 应用，让你感受到使用这些技术框架带来的畅快感。本文涉及的知识点比较多，代码尽可能少放，主要是叙述逻辑思路，最后会以实现常见的 分页功能 作为案例来详细展示。

数据库 ORM

首先我们需要解决数据库相关的技术选项，这里说的技术选型是指 ORM 相关的技术选型（数据库固定使用 MySQL），选型的基本原则是能力强大、用法简单。

ORM 选型

1. ORM 实例教程：阮一峰教程，解释 ORM，通俗易懂。
2. 使用 Typeorm 在 Midway 6 中获得灵活一致的数据管理体验 ：一篇很受启发的文章。
3. 数据访问库的选择之 TypeORM：讨论了如何在 Midway 中接入 TypeORM，还讨论了一些高级用法。

除了直接拼 SQL 语句这种略微硬核的方式外，Node.js 应用开发者更多地会选择使用开源的 ORM 库，如 Sequelize。而在 Typescript 面前，工具库层面目前两种可选项，可以使用 sequelize-typescript 或者 TypeORM 来进行数据库的管理。总结原因如下：

- 原生类型声明，与 Typescript 有更好的相容性；
- 支持装饰器写法，用法上简单直观；且足够强的扩展能力，能支持复杂的数据操作；
- 该库足够受欢迎，Github Star 数量高达 20.3k（截止此文撰写 2020.08 时），且官方文档友好。

并非说 Sequelize-typescript 不行，这两个工具库都很强大，都能满足业务技术需求；Sequelize 一方面是 Model 定义方式比较 JS 化在 Typescript 天然的类型环境中显得有些怪异；另一方面也与 Midway 6 整体的编码风格不太统一，所以我个人更加倾向于用 TypeORM 。

两种操作模式

1. 架构模式中的 Active Record 和 Data Mapper
2. 什么是 ActiveRecord 模式

这里简单说明一下，ORM 架构模式中，最流行的实现模式有两种：Active Record 和 Data Mapper。比如 Ruby 的 ORM 采取了 Active Record 的模式是这样的：

```
$user = new User;
$user->username = 'philipbrown';
$user->save();
```

再来看使用 Data Mapper 的 ORM 是这样的：

```
$user = new User;
$user->username = 'philipbrown';
EntityManager::persist($user);
```

现在我们察看到了它们最基本的区别：在 Active Record 中，领域对象有一个 save() 方法，领域对象通常会继承一个 ActiveRecord 的基类来实现。而在 Data Mapper 模式中，领域对象不存在 save() 方法，持久化操作由一个中间类来实现。

这两种模式没有谁比谁好之分，只有适不适合之别：

1. 简单的 CRUD、试水型的 Demo 项目，用 Active Records 模式的 ORM 框架更好。
2. 业务流程和规则较多的、成熟的项目改造用 Data Mapper 型，其允许将业务规则绑定到实体。

Active Records 模式最大优点是简单，直观，一个类就包括了数据访问和业务逻辑，恰好我现在这个小应用基本都是单表操作，所以就用 Active Records 模式了。

TypeORM 的使用

数据库连接

首先，提供数据库初始化 service 类：

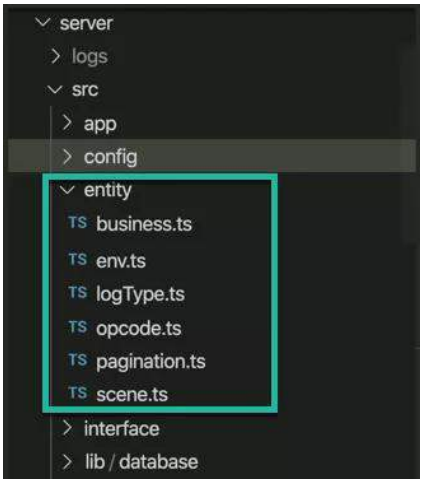
```
// src/lib/database/service.ts

import { config, EggLogger, init, logger, provide, scope, ScopeEnum, Application, ApplicationContext } from '@ali/midway';
import { ConnectionOptions, createConnection, createConnections, getConnection } from 'typeorm';

const defaultOptions: any = {
type: 'mysql',
```

```
    synchronize: false,
    logging: false,
    entities: [
    'src/app/entity/**/*.ts'
    ],
    };

    @scope(ScopeEnum.Singleton)
    @provide()
    export default class DatabaseService {
    static identifier = 'databaseService';
// private connection: Connection;
    /** 初始化数据库服务实例 */
    static async initInstance(app: Application) {
    const applicationContext: ApplicationContext = app.applicationContext;
    const logger: EggLogger = app.getLogger();
    // 手动实例化一次，启动数据库连接
    Const databaseService = await
applicationContext.getAsync<DatabaseService>(DatabaseService.identifier);
    const testResult = await databaseService.getConnection().query('SELECT 1+1');
    logger.info('数据库连接测试：SELECT 1+1 =>', testResult);
    }

    @config('typeorm')
    private ormconfig: ConnectionOptions | ConnectionOptions[];


    @logger()
    logger: EggLogger;

    @init()
    async init() {

    const options = {
    ...defaultOptions,
    ...this.ormconfig
    };
    try {
      if (Array.isArray(options)) {
        await createConnections(options);
        } else {
```

```
      await createConnection(options);
    }
  this.logger.info('[%s] 数据库连接成功~', DatabaseService.name);
  } catch (err) {
  this.logger.error('[%s] 数据库连接失败！', DatabaseService.name);
  this.logger.info('数据库链接信息:', options);
  this.logger.error(err);
    }
  }

    }
  }

  /**
  * 获取数据库链接
  * @param connectionName 数据库链接名称
  */
  getConnection(connectionName?: string) {
  return getConnection(connectionName);
    }
  }
```

说明：

1. 这里一定是单例 @scope(ScopeEnum.Singleton)，因为数据库连接服务只能有一个。但是可以初始化多个连接，比如用于多个数据库连接或读写分离。
2. 默认配置项 defaultOptions 中的 entities 表示数据库实体对象存放的路径，推荐专门创建一个 entity 目录用来存放：



其次，在 Midway 的配置文件中指定数据库连接配置：

```
  // src/config/config.default.ts
export const typeorm = {
  type: 'mysql',
  host: 'xxxx',
  port: 3306,
  username: 'xxx',
  password: 'xxxx',
  database: 'xxxx',
  charset: 'utf8mb4',
  logging: ['error'], // ["query", "error"]
  entities: [`${appInfo.baseDir}/entity/**/!(*.d|base){.js,.ts}`],
};

// server/src/config/config.local.ts
export const typeorm = {
  type: 'mysql',
  host: '127.0.0.1',
  port: 3306,
  username: 'xxxx',
  password: 'xxxx',
  database: 'xxxx',
  charset: 'utf8mb4',
  synchronize: false,
  logging: false,
  entities: [`src/entity/**/!(*.d|base){.js,.ts}`],
}
```

说明：

1. 因为要区分 aone 环境运行和本地开发，所以需要配置两份。
2. entities 的配置项本地和线上配置是不同的，本地直接用 src/entity 就行，而 aone 环境需要使用 ${appInfo.baseDir} 变量。

最后，在应用启动时触发实例化：

```
// src/app.ts

import { Application } from '@ali/midway';
import "reflect-metadata";

import DatabaseService from './lib/database/service';
export default class AppBootHook {
  readonly app: Application;

constructor(app: Application) {
  this.app = app;
  }

// 所有的配置已经加载完毕
// 可以用来加载应用自定义的文件，启动自定义的服务
async didLoad() {
  await DatabaseService.initInstance(this.app);
 }
}
```

说明：

1. 选择在 app 的配置加载完毕之后来启动自定义的数据库服务，具体参考 启动自定义的声明周期参考文档 说明。
2. 为了不侵入 AppBootHook 代码太多，我把初始化数据库服务实例的代码放在了 DatabaseService 类的静态方法中。

数据库操作

- typeorm 数据库 ORM 框架中文文档
- Active Record vs Data Mapper：官方文档对两者的解释

数据库连接上之后，就可以直接使用 ORM 框架进行数据库操作。不同于现有的所有其他 JavaScript ORM 框架，TypeORM 支持 Active Record 和 Data Mapper 模式（在我这次写的项目中，使用的是 Active Record 模式），这意味着你可以根据实际情况选用合适有效的方法编写高质量的、松耦合的、可扩展的应用程序。

首先看一下用 Active Records 模式的写法：

```
import {Entity, PrimaryGeneratedColumn, Column, BaseEntity} from "typeorm";

@Entity()
export class User extends BaseEntity {

@PrimaryGeneratedColumn()
id: number;
@Column()
firstName: string;

@Column()
lastName: string;

@Column()
age: number;

}
```

说明：

1. 类需要用 @Entity() 装饰
2. 需要继承 BaseEntity 这个基类

对应的业务域写法：

```
    const user = new User();
    user.firstName = "Timber";
    user.lastName = "Saw";
    user.age = 25;
    await user.save();
```

其次看一下 Data Mapper 型的写法：

```
// 模型定义
    import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";

    @Entity()
    export class User {

    @PrimaryGeneratedColumn()
    id: number;

    @Column()
    firstName: string;

    @Column()
    lastName: string;

    @Column()
    age: number;
}
```

说明：

1. 类同样需要用 @Entity() 装饰
2. 不需要继承 BaseEntity 这个基类

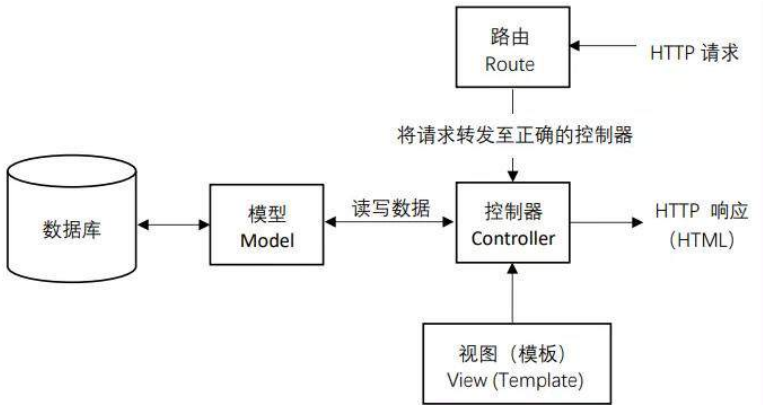
对应的业务域逻辑是这样的：

```
    const user = new User();
    user.firstName = "Timber";
    user.lastName = "Saw";
    user.age = 25;
    await repository.save(user);
```

无论是 Active Record 模式还是 Data Mapper 模式，TypeORM 在 API 上的命名使用上几乎是保持一致，这大大降低了使用者记忆上的压力：比如上方保存操作，都称为 save 方法，只不过前者是放在 Entity 实例上，后者是放在 Repository 示例上而已。

MVC 架构

整个服务器的设计模式，就是经典的 MVC 架构，主要就是通过 Controller、Service、Model 、View 共同作用，形成了一套架构体系；



此图来源于 《Express 教程 4：路由和控制器》

上图是最为基础的 MVC 架构，实际开发过程中还会有更细分的优化，主要体现两方面：

1. 为了方便后期扩展，还会引入 中间件（middleware） 机制，这些概念相信但凡写过 Koa/Express 的都知道 —— 不过这里还是重述一下，因为后面 GraphQL 就是通过中间件方式引入的。
2. 一般不推荐直接让 Controller 调用到 Model 对象，而是要中间添加一层 Service 层来进行解耦（具体的优势详见 Egg.js 官方文档《服务（Service）》，里面有详细的解释）；简单来讲，这样的好处在于解耦 Model 和 Controller，同时保持业务逻辑的独立性（从而带来更好的扩展性、更方便的单元测试等），抽象出来的 Service 可以被多个 Controller 重复调用 —— 比如，GraphQL Resolver 和 Controller 就可以共用同一份 Service。

现代 Node.js 框架初始化的时候都默认帮你做了这事情 —— Midway 也不例外，初始化后去看一下它的目录结构就基本上懂了。

更多关于该架构的实战可参考以下文章：

1. Node Service-oriented Architecture：介绍面向 Service 的 Node.js 架构
2. Designing a better architecture for a Node.js API：初学者教程，从实践中感受面向 Service 架构
3. Bulletproof node.js project architecture：如何打造一个坚固的 Node.js 服务端架构

RESTful API

在 Midway 初始化项目的时候，其实已经具备完整的 RESTful API 的能力，你只要照样去扩展就可以了，而且基于装饰器语法和 DI 风格，编写路由非常的方便直观，正如官方《路由装饰器》里所演示的代码那样，几行代码下来就输出标准的 RESTful 风格的 API：

```
import { provide, controller, inject, get } from 'midway';

@provide()
@controller('/user')
export class UserController {

@inject('userService')
service: IUserService;

@inject()
ctx;

@get('/:id')
async getUser(): Promise<void> {
const id: number = this.ctx.params.id;
const user: IUserResult = await this.service.getUser({id});
this.ctx.body = {success: true, message: 'OK', data: user};
  }
}
```

GraphQL

- 其次需要阅读 type-graph 的教程，比如 Resolvers 章节，具体的代码参考可以前往 recipe-resolver
- TypeScript + GraphQL = TypeGraphQL：阿里 CCO 体验技术部的文章，介绍地比较详细到位，推荐阅读（结合 egg.js 的开发实践）

RESTful API 方式用得比较多，不过我还是想在自己的小项目里使用 GraphQL，具体的优点我就不多说了，可以参考《GraphQL 和 Apollo 为什么能帮助你更快地完成开发需求？》等相关文章。

GraphQL 的理解成本和接入成本还是有一些的，建议直接通读官方文档《GraphQL 入门》去了解 GraphQL 中的概念和使用。

接入 GraphQL 服务中间件

整体的技术选型阵容就是 apollo-server-koa 和 type-graphql：

- apollo-server 是一个在 Node.js 上构建 GraphQL 服务端的 Web 中间件，支持 Koa 也就天然的支持了 Midway。
- TypeGraphQL：它通过一些 TypeScript + Decorator 规范了 Schema 的定义，避免在 GraphQL 中分别写 Schema Type DSL 和数据 Modal 的重复劳动。

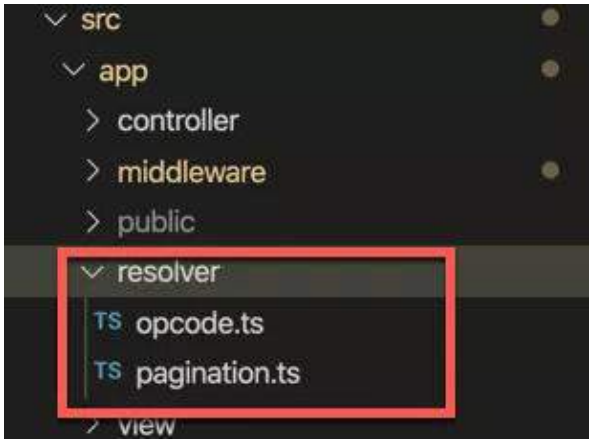
只需要将 Koa 中间件 转 Midway 中间件就行。根据 Midway 项目目录约定，在 /src/app/middleware/ 下新建文件 graphql.ts，将 apollo-server-koa 中间件简单包装一下：

```
import * as path from 'path';
import { Context, async, Middleware } from '@ali/midway';
import { ApolloServer, ServerRegistration } from 'apollo-server-koa';
import { buildSchemaSync } from 'type-graphql';

export default (options: ServerRegistration, ctx: Context) => {
```

```
const server = new ApolloServer({
 schema: buildSchemaSync({

    resolvers: [path.resolve(ctx.baseDir, 'resolver/*.ts')],
    container: ctx.applicationContext
      })
    });
    return server.getMiddleware(options);
    };
```

说明：

1. 利用 apollo-server-koa 暴露的 getMiddleware 方法取得中间件函数，注入 TypeGraphQL 所管理的 schema 并导出该函数。
2. 我们所有的 GraphQL Resolver 都放在 'app/resolver' 目录下：



由于 Midway 默认集成了 CSRF 的安全校验，我们针对 /graphql 路径的这层安全需要忽略掉：

```
export const security = {
csrf: {
// 忽略 graphql 路由下的 csrf 报错
ignore: '/graphql'
  }
}
```

接入的准备工作到这里就算差不多了，接下来就是编写 GraphQL 的 Resolver 相关逻辑。

Resolvers

对于 Resolver 的处理，TypeGraphQL 提供了一些列的 Decorator 来声明和处理数据。通过 Resolver 类的方法来声明 Query 和 Mutation，以及动态字段的处理 FieldResolver。几个主要的 Decorator 说明如下：

- @Resolver：来声明当前类是数据处理的
- @Query：声明改方法是一个 Query 查询操作
- @Mutation：声明改方法是一个 Mutation 修改操作
- @FieldResovler：对 @Resolver(of => Recipe) 返回的对象添加一个字段处理

方法参数相关的 Decorator：

- @Root：获取当前查询对象
- @Ctx：获取当前上下文，这里可以拿到 egg 的 Context （见上面中间件集成中的处理）
- @Arg：定义 input 参数

这里涉及到比较多的知识点，不可能一一罗列完，还是建议先去官网阅读一遍。

接下来我们从接入开始，然后以如何创建一个分页(Pagination) 功能为案例来演示在如何在 Midway 框架里使用 GraphQL，以及如何应用上述这些装饰器。

案例：利用 GraphQL 实现分页功能

分页的数据结构

- Apollo Server: GraphQL 数据分页概述

从使用者角度来，我们希望传递的参数只有两个 pageNo 和 pageSize ，比如我想访问第 2 页、每页返回 10 条内容，入参格式就是：

```
{
pageNo: 2,
pageSize: 10
}
```

而分页返回的数据结构如下：

```
{
articles {
totalCount # 总数
pageNo       # 当前页号
pageSize     # 每页结果数
pages        # 总页数
 list: {      # 分页结果
title,
 author
    }
  }
}
```

Schema 定义

首先利用 TypeGraphQL 提供的 Decorator 来声明入参类型以及返回结果类型：

```
// src/entity/pagination.ts

import { ObjectType, Field, ID, InputType } from 'type-graphql';
import { Article } from './article';

// 查询分页的入参
@InputType()
export class PaginationInput {
@Field({ nullable: true })
pageNo?: number;
```

```
  @Field({ nullable: true })
  pageSize?: number;
  }

  // 查询结果的类型
  @ObjectType()
  export class Pagination {
  // 总共有多少条
  @Field()
  totalCount: number;

   // 总共有多少页
  @Field()
  pages: number;

  // 当前页数
  @Field()
  pageNo: number;

  // 每页包含多少条数据
  @Field()
  pageSize: number;

  // 列表
  @Field(type => [Article]!, { nullable: "items" })
  list: Article[];
  }

  export interface IPaginationInput extends PaginationInput { }
```

说明：

1. 通过这里的 @ObjectType() 、@Field() 装饰注解后，会自动帮你生成 GraphQL 所需的 Schema 文件，可以说非常方便，这样就不用担心自己写的代码跟 Schema 不一致；
2. 对 list 字段，它的类型是 Article[] ，在使用 @Field 注解时需要注意，因为我们想表示数组一定存在但有可能为空数组情况，需要使用 {nullable: "items"}（即 [Item]!），具体查阅 官方文档 - Types and Fields 另外还有两种配置：

a. 基础的 { nullable: true | false } 只能表示整个数组是否存在（即[Item!] 或者 [Item!]!）
b. 如果想表示数组或元素都有可能为空时，需要使用 {nullable: "itemsAndList"}（即 [Item]）

Resolver 方法

基于上述的 Schema 定义，接下来我们要写 Resolver，用来解析用户实际的请求：

```
// src/app/resolver/pagination.ts
import { Context, inject, provide } from '@ali/midway';
import { Resolver, Query, Arg, Root, FieldResolver, Mutation } from 'type-graphql';
import { Pagination, PaginationInput } from '../../entity/pagination';
import { ArticleService } from '../../service/article';

@Resolver(of => Articles)
@provide()
export class PaginationResolver {

@inject('articleService')
articleService: ArticleService;

@Query(returns => Articles)
async articles(@Arg("query") pageInput: PaginationInput) {
  return this.articleService.getArticleList(pageInput);
    }
  }
```

- 实际解析用户请求，调用的是 Service 层中 articleService.getArticleList 方法，只要让返回的结果跟我们想要的 Pagination 类型一致就行。
- 这里的 articleService 对象就是通过容器注入(inject)到当前 Resolver ，该对象的提供来自 Service 层。

Service 层

从上可以看到，请求参数是传到 GraphQL 服务器，而真正进行分页操作的还是 Service 层，内部利用 ORM 提供的方法；在 TypeORM 中的分页功能实现，可以参考一下官方的 find 选项的完整示例：

```
userRepository.find({
  select: ["firstName", "lastName"],
  relations: ["profile", "photos", "videos"],
  where: {
        firstName: "Timber",
        lastName: "Saw"
},
      order: {
          name: "ASC",
           id: "DESC"

           },
           skip: 5,
           take: 10,
           cache: true

           });
```

其中和 分页 相关的就是 skip 和 take 两个参数（ where 参数是跟 过滤 有关，order 参数跟排序有关）。

- How to implement pagination in nestjs with typeorm ：这里给出了使用 Repository API 实现的方式
- Find 选项： 官方 Find API 文档

所以最终我们的 Service 核心层代码如下：

```
// server/src/service/article.ts
import { provide, logger, EggLogger, inject, Context } from '@ali/midway';
import { plainToClass } from 'class-transformer';
import { IPaginationInput, Pagination } from '../../entity/pagination';
...
```

```
@provide('articleService')
export class ArticleService {
  ...

/**
* 获取 list 列表，支持分页
*/
async getArticleList(query: IPaginationInput): Promise<Pagination> {
const {pageNo = 1, pageSize = 10} = query;

const [list, total] = await Article.findAndCount({
order: { create_time: "DESC" },
take: pageSize,
skip: (pageNo - 1) * pageSize
});

return plainToClass(Pagination, {
totalCount: total,
pages: Math.floor(total / pageSize) + 1,
pageNo: pageNo,
pageSize: pageSize,
list: list,
      })
  }
  ...
}
```

- 这里通过 @provide('articleService') 向容器提供 articleService 对象实例，这就上面 Resolver 中的 @inject('articleService') 相对应。
- 由于我们想要返回的是 Pagination 类实例，所以需要调用 plainToClass 方法进行一层转化。

Model 层

Service 层其实也是调用 ORM 中的实体方法 Article.findAndCount（由于我们是用 Active Records 模式的），这个 Article 类就是 ORM 中的实体，其定义也非常简单：

```
// src/entity/article.ts

import { Entity, PrimaryGeneratedColumn, Column, BaseEntity } from "typeorm";
import { InterfaceType, ObjectType, Field, ID } from 'type-graphql';

@Entity()
@InterfaceType()
export class Article extends BaseEntity {

@PrimaryGeneratedColumn()
@Field(type => ID)
id: number;

@Column()
@Field()
title: string;

@Column()
@Field()
author: string;
}
```

仔细观察，这里的 Article 类，同时接受了 TypeORM 和 TypeGraphQL 两个库的装饰器，寥寥几行代码就支持了 GraphQL 类型声明和 ORM 实体映射，非常清晰明了。

到这里一个简单的 GraphQL 分页功能就开发完毕，从流程步骤来看，一路下来几乎都是装饰器语法，整个编写过程干净利落，很利于后期的扩展和维护。

小结

距离上次写 Node.js 后台应用有段时间了，当时的技术栈和现在的没法比，现在尤其得益于使用 Decorator（装饰器语法） + DI（依赖注入）风格写业务逻辑，再搭配使用 typeorm （数据库的连接）、 type-graphql （GraphQL 的处理）工具库来使用，整体代码风格更加简洁，同样的业务功能，代码量减少非常可观且维护性也提升明显。

emm，这种感觉怎么描述合适呢？之前写 Node.js 应用时，能用，但是总觉得哪里很憋屈 —— 就像是白天在交通拥挤的道路上堵车，那种感觉有点糟；而这次混搭了这几种技术，会感受神清气爽 —— 就像是在高速公路上行车，畅通无阻。

前端的技术发展迭代相对来说迭代比较快，这是好事，能让你用新技术做得更少、收获地更多；当然不可否认这对前端同学也是挑战，需要你都保持不断学习的心态，去及时补充这些新的知识。学无止境，与君共勉。

链接：

https://pro.ant.design/docs/getting-started-cn
https://midway.alibaba-inc.com/midway
https://ant.design/docs/react/migration-v4-cn
http://www.ruanyifeng.com/blog/2019/02/orm-tutorial.html
https://github.com/RobinBuschmann/sequelize-typescript
https://github.com/typeorm/typeorm
https://blog.csdn.net/Frankltf/article/details/86626338
https://blog.csdn.net/YamateDD/article/details/6826255
https://eggjs.org/zh-cn/basics/app-start.html
https://juejin.im/post/6844903920578330631
https://typeorm.io/
https://eggjs.org/zh-cn/basics/service.html
https://www.codementor.io/@evanbechtol/node-service-oriented-architecture-12vjt9zs9i
https://dev.to/pacheco/designing-a-better-architecture-for-a-node-js-api-24d
https://softwareontheroad.com/ideal-nodejs-project-structure/
https://midwayjs.org/midway/guide.html
https://typegraphql.com/docs/resolvers.html
https://zhuanlan.zhihu.com/p/56516614
https://segmentfault.com/a/1190000018706816

https://graphql.cn/learn/

https://npm.alibaba-inc.com/package/apollo-server-koa

https://npm.alibaba-inc.com/package/type-graphql

https://segmentfault.com/a/1190000009565131

https://typegraphql.com/docs/types-and-fields.html

https://stackoverflow.com/questions/53922503/how-to-implement-pagination-in-nestjs-with-typeorm

https://github.com/typeorm/typeorm/blob/master/docs/zh_CN/find-options.md

移动前端开发和 Web 前端开发的区别是什么？

作者 | 卓凌
出品 | 淘系技术

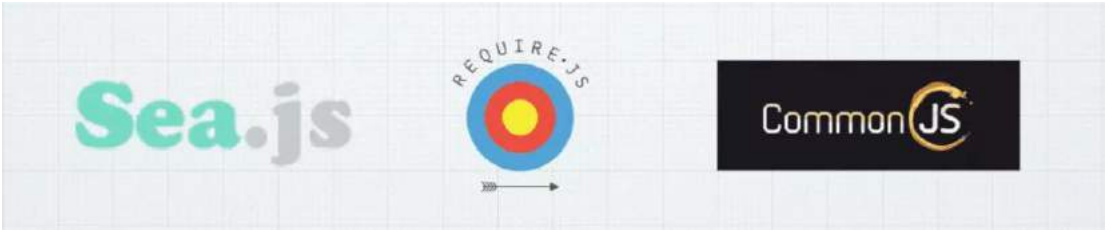
前端这门技术，从诞生发展至今不过寥寥十余年。如果说前十年是 PC 前端的时代，那后十年一定是属于移动前端的时代。特别是随着网络制式的发展，移动设备在全球范围内得到了空前的普及，在前端领域，Hybird Web、React Native、Weex、Flutter 等等一系列新的移动前端技术也如同雨后春笋般冒出来，今天来和大家分享一下我对「移动前端开发和 Web 前端开发」的理解。

回顾：前端发展史

阶段一：刀耕火种

十多年前的前端，开发者还在为兼容 IE6 而头疼，框架上 jQuery 是老大，有追求的前端开发可能会使用 Zepto.js 以减少网页体积。这个时候，前端页面主要还是以 PC 为主，这个时候根本没有移动前端的概念，在小小的手机屏幕上流量的页面则是以纯文本为主。

阶段二：工程化



在 2011 ~ 2014 年之间的历史阶段里，模块化的思路占为主导。当时为了进行 Assets 资源加载器的设计，就制定了模块化的协议规范。当时比较流行的模块化协议就是 AMD（RequireJS）、CMD（Seajs 为代表）、KMD（Kissy 为代表）。在淘宝、天猫，Kissy 应用的很火，所以 KMD 主导天下；在支付宝及外部社区，Seajs 应用的很火，所以 CMD 主导天下，玉伯大大的名气和威望也在前端圈里特别高；而 AMD 在国外比较流行，但渐渐也被后来出现的 CommonJS 规范削弱了气势。

阶段三：移动化

随着 3G、4G 的发展和 iOS 和 Android 手机在市场的普及量大增，PC 业务主战场也逐渐过渡到移动端。前端的思维模式由 PC 转向了移动端，并向 App 的用户体验看齐。移动端的 HTML5 协议支持不完善，前端的生产配套不全，Android 的屏幕碎片化，所以那个时候的前端开发移动端页面适配的痛苦要远远超过 PC 时代。

阶段四：框架化

在前端社区，Angular、React、Vue、RN （React Native） 这样的 MV* 框架一个接着一个出现，让前端接受了数据驱动思想的洗礼之外，还借助 RN 完成了移动端的体验升级，包括后来的 Weex、Flutter。

在这个阶段，前端开始有了终端的底层架构组，开始构思前端页面在移动终端上的加载性能和用户体验表现。在阿里巴巴内部，为了解决多端复用的问题，Rax 借助 VDOM 打通 WebView 和 Weex 两端，一套代码跑天下。

阶段五：垂直化

随着初代 iPhone 的发布，大屏幕手机逐渐变成了主流，移动端的需求开始爆发。在淘宝天猫，每年双十一的移动端成交额逐年攀升，并逐渐占领绝对领先地位。前端的领域也随着这种趋势逐渐细分，按照场景可以简单分为移动(无线)前端开发和中后台前端开发。

移动前端开发面向的是消费者端的 Web 与 轻 App 业务场景，在这个场景下，淘系经过多年的营销活动沉淀，逐渐形成了移动端独特的、轻量级的解决方案，以及模块维度的、面向运营的页面搭建系统。

中后台前端则是面向企业 ERP、CRM 、OA 等偏后的业务场景，如商家后台等系统。在这个场景下，借助飞冰、Fusion Design 等中后台物料，形成可视化的中后台搭建解决方案，为业务的前端、开发或产品角色提供一站式中后台生产解决方案。

移动前端：混合技术的前世今生

曾几何时，移动客户端开发和前端开发是两条没有交集的平行线，但现在我们越来越拥抱两者的合作产物：混合式(Hybird)应用开发，或者用最近比较火的一个概念 -- 大前端技术。

从技术的表现形式思考，本质上客户端开发与前端开发都是终端技术，它的特点是直接面向用户 UI 编程。

同是 UI 编程，我们面对的痛点是什么？

传统 Web 前端技术的技术局限性

1. 资源加载：HTML、JS、CSS、图片等静态资源存放于远端的服务器，需要动态的异步拉取，再拉取数据进行展示，初始化效率上比 Native 慢的多。
2. 渲染机制：在浏览器的设计中，JS 的执行和页面的布局、Paint 都在同一个主线程，无法并行化，再加上 JS 的性能赶不上 AOT 语言，执行复杂逻辑时导致的卡顿通常会阻塞 UI，再加上冗长的渲染管线，导致浏览器的渲染体验在等量对比 Native 时并不占优势。
3. 页面切换：在浏览器中并不存在路由的概念，这导致页面间的切换体验完全依赖于浏览器 Shell 提供的能力，在页面切换的时候会反复加载。当然前端社区中也出现了单页面应用的概念，但是多个页面的资源也显著增加了 JSBundle 的体积，也使页面的开发更加复杂化了。
4. API 能力：浏览器的安全机制是基于同源策略的沙箱机制，这套沙箱机制阻止了前端开发者使用原生系统能力，你只能使用 W3C 标准定义的功能，而且考虑到终端碎片化的问题，这些接口往往不能直接使用。这在 PC 端的场景中是没有什么问题的，但是在移动端则相反，开发者希望有能力调用系统接口实现一些更富交互的场景。
5. 交互性能：浏览器的实时交互性能体验差，在复杂交互场景中大规模的重排限制了 UI 帧率，这种限制在中低端移动设备中尤为严重。
6. 脚本语言，动态解析执行：JS 是一门 JIT 语言，也就是需要动态解析执行，对比预先编译机器码的 AOT 语言的执行性能就差得多了。

传统客户端技术的局限性?

1. 动态性：客户端开发通常是有固定的版本发布计划，而且受制于 Apple 的 App Store 审核规则，版本发布的不确定性还会受到政策影响，Android 在国内的渠道众多，每次发版都要反复检查渠道，一旦发现线上问题需要依赖再次发版，容错成本非常高，这也大大增加了对业务的局限性。
2. 开发成本：客户端的开发成本高，然而生态还不如 Web 丰富，npm 社区的几万开源包，加上更活跃的开发者社区，导致对企业来讲客户端的研发成本是高于 Web 开发的。
3. 跨端一致性：传统客户端开发一套业务，是需要实现 Android + iOS 两套代码的，而且由于 Android 和 iOS 的操作系统能力差异，同样的需求往往会用不同的视觉和交互来实现，这也导致了业务成本居高不下。

混合式前端开发

为什么会出现混合式前端开发?

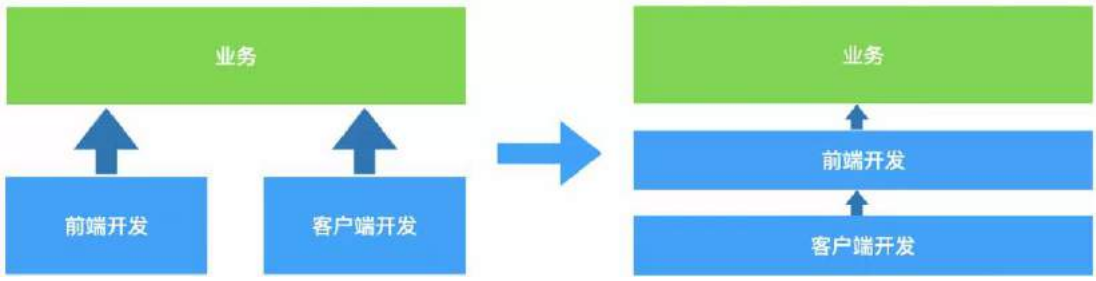
随着 iOS + Android 确立了移动操作系统的统治地位，前端开发者也在寻找使用操作系统提供的能力进行业务开发的模式。Web 的开发方式远比 iOS 和 Android 更加方便和高效，Web 上层出不穷的各种库和框架也远比 Android 和 iOS 的各种库和框架方便的多。Web 上我们可以方便的使用各种前端框架，及时预览效果（想想大型 Android/iOS 工程的编译时间）。

从阿里巴巴的角度来看，随着中台化的理念逐渐被接受：业务需要追求快速发展，前台的 UI 和需求会随着商业决策快速迭代，前端和客户端不同的岗位也形成了分工化的研发模式。

前端向上，前置作为业务方的唯一接口，逐渐演变为大前端的业务层。在这一层，它的职责是负责定义规范，通过框架规范业务的开发过程，同时封装统一的解决方案和工程化能力，将重复的工作抽离。

客户端向下扎，解耦业务需求，转为大前端的架构层，给上层的业务开发者提供能力支持。通过将客户端的系统级 API 以及宿主应用的能力暴露给上层前端，提高前端页面对更多富

交互场景的承载能力。



在这样的大背景下，各种混合开发技术层出不穷，在这里我们简单的把混合式应用的发展定义为三个阶段：

阶段一：JSBridge

在这个阶段，主要还是以 WebView 为主，并配合 JSBridge 提供了 Naive 与 JS 之间的通信链路，基于这个通信基础，Native 可以暴露出一些标准服务 API 提供给 JS 调用，同样的 JS 也可以以此封装一些基础 API 给 Native 调用。前端开发者使用传统 JS + HTML + CSS 进行页面的开发，并且调用 JSBridge API 驱动客户端能力。在这个阶段，Apache Cordova 是业内的佼佼者，大多互联网公司内部也有自己的 JSBridge 框架实现，可以说 JSBridge 第一次给了前端开发者调用 Native 的能力。

但是 JSBridge 方案的主要缺点在于性能方面及高级组件扩展能力缺失，流畅性始终无法与 Native 媲美。

阶段二：原生 UI

虽然 Web 的动态性和高效的开发效率，是原生开发望尘莫及的，但是浏览器技术的瓶颈也非常明显：

1. W3C 作为开放的技术标准，历史悠久，包袱多，显著拖慢了浏览器的性能。
2. WebView 渲染引擎设计的上的缺陷，渲染流水线非常长，导致浏览器对合成器动画和非合成器动画区别对待，非合成动画性能不好。
3. 单线程模型，无法发挥现代硬件架构特别是 ARM 架构多核心的性能。
4. 异步光栅化的设计，在进行长列表滚动时，不可避免出现白屏的现象。

有没有一种两全其美的方式?
React Native/Weex 的出现给前端开发者带来了一道曙光。

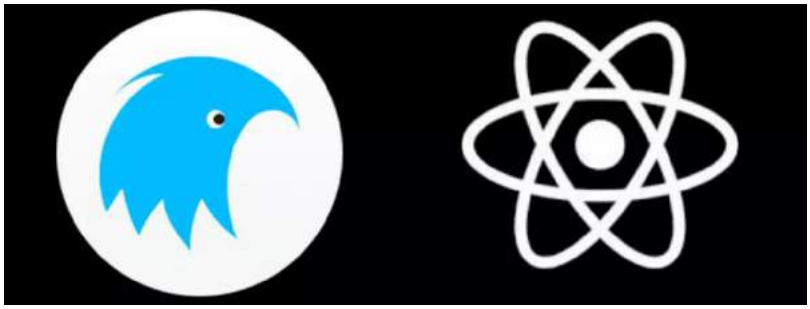
React Native/Weex 利用 JS 引擎来调用 Native 端的组件，从而实现相应的功能。React Native 和 Weex 都允许前端开发者使用 JS 进行业务逻辑开发，使用 VDOM 来描述文档结构，并配合 CSS 的子集来定制样式，样式和模板分离。

Weex 体系中，JS 的执行在 JS Thread，Layout 执行在独立的 Layout Thread，渲染执行在系统的 MainThread ，三个线程相互独立，并行执行。

在 WebView 的体系中 JS 的执行、 Layout 、 Paint 都在 MainThread ，相互影响，在进行复杂任务时会导致界面卡顿。

这种方案的优势在于最大化的复用前端的生态和 Native 的生态体系。

在阿里巴巴，Weex 的大规模应用，甚至支撑起了双十一亿级的流量。



阶段三：自绘引擎

什么是自绘引擎?

所谓自绘引擎，就是不依赖操作系统提供的布局、原生组件能力，直接调用 GPU 或者底层抽象层进行绘制的渲染引擎。

在上一个阶段，前端开发者已经可以使用 JS 引擎驱动原生 UI 了，为什么还需要自绘引擎 React Native/Weex 充分将 Native 的 View 体系输出到前端体系中，在进行 Android/iOS Native View 的封装过程中，存在不少难以逾越的障碍。如：难以抹平的双端一致性问题、复杂样式能力难以实现、Layout 动画需要执行两次（WeexCore Layout 和 Android Native 本身的 Layout ）。组件的封装成本随着复杂度增加也越来越高，难以逾越 Native View 限制提供更细致的 W3C 标准能力。

2018 年 Flutter 诞生，通过 Dart 语言构建一套跨平台的开发组件，所有组件基于 Skia 引擎自绘，在性能上和 Native 平台的 View 相媲美，同时解决了上一代架构难以解决的双端一致性等问题。引起大家广泛关注，充分验证了通过绘制构建组件做到 NativeView 媲美的 UI 渲染引擎的可行性。

但是 Flutter 本身是缺乏动态更新特性的，社区上也有一些项目在探索 Flutter 的动态化特性，我们团队内部也在实现面向前端的动态化 Flutter 引擎：Kraken，与其它方案不同的是 Kraken 并没有基于 Flutter 自带的 Widgets 框架进行扩展，而是从底层扩展了 W3C 标准的 API，这使得它更像一个浏览器，也让 Flutter 面向 Web 开发者的使用门槛大大降低。

未来：回归本源

天下大势，分久必合，合久必分。移动前端开发本质上还是终端开发的一种形态，不管容器、框架、语言怎么变，在前端开发者中只有 W3C 的标准是永远不变的。笔者认为，随着 Web 的发展，在解决一系列性能、体验问题之后，浏览器技术会成为更通用的 UI 编程标准。

PWA

早先年，Google 就已经在这一领域内努力，推出了 PWA（Progress WebApplication）的概念。

PWA 通过在移动端浏览器提供标准化框架，在 Web App 中实现和 Native App 接近的用户体验。它的特性其实是一系列 W3C 标准和私有标准集合，简单的说 PWA 支持：

- 离线加载：通过 Service Worker 等提供的缓存机制，允许用户在断网或者弱网的情况下直接读取离线资源。
- 后台驻留进程：正常情况下，浏览器的页面关闭后它的整个生命周期就结束了，内存也得到了释放。Service Worker 允许页面在关闭的情况下继续运行，这保证了类似于离线缓存、主动推送等。

- 消息通知：允许 Web 开发者实现类似 App 的主动推送机制。
- 其它移动 App 的功能特性，如直接保存图标到桌面，允许用户像正常使用 App 一样打开 PWA 应用；可以隐藏 UI 中的默认浏览器元素，让 Web 内容全屏展示，从视觉上看让 Web 应用更像一个原生应用，有时候你根本无法分辨到底是 Web 应用还是原生应用。

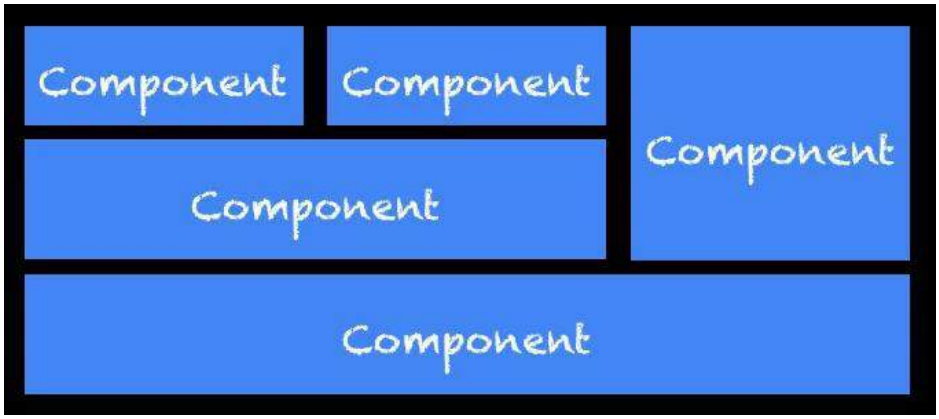
PHA

当然在标准能力不完善，业务又需要定制化能力的当下，混合式应用还会继续发展。PHA（Progress Hybird Application）的概念应用而生，PHA 是一种渐进式的混合应用增强

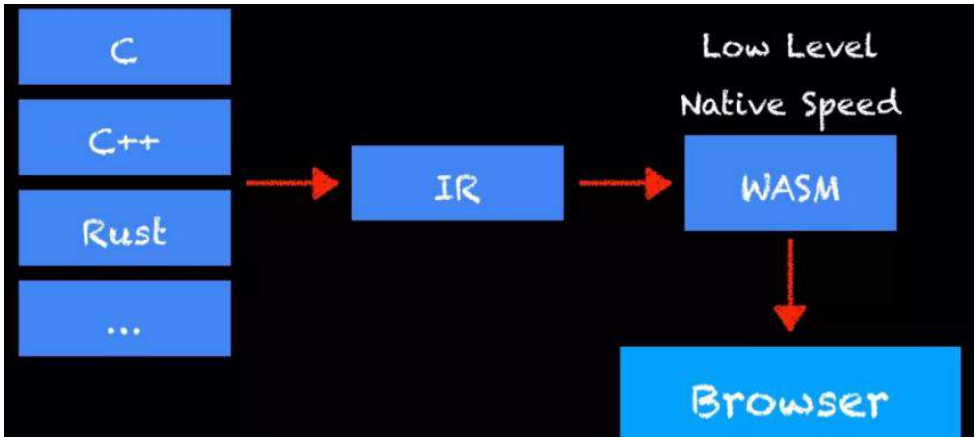
策略， 提供端测的辅助能力，提升 WebView 的渲染性能与体验。广义地说，当下比较火的小程序、快应用都是 PHA 的一种实现。

在淘系内部，我们也在实现一套轻量级的 PHA 方案，并且在大促中也取得了不错的效果，我想后面单独出一篇关于 PHA 的文章来阐述。

关于未来，随着技术方案的多样化、以及端边界的扩展，我们认为最重要的就是效率与性能的问题。



基于大数据的机器学习能力，移动端上会拥有更高效的 UI 编排能力，最终能让 UI 渲染实现实时个性化。



基于最近比较热的 WebAssembly 能力，让浏览器突破 JavaScript 的限制，能拥有更大的想象空间。

如何用 SVG 实现一个优雅的提示框?

作者 | 驭剑
出品 | 淘系技术

Tooltips 常被称为提示框（或信息提示框），提示框能够以较强的交互性、自由度为用户提供相应的提示信息。今天我们要聊的不是如何实现强大的交互行为，而是来看看如何以最好的方式来还原他们的视觉效果，并且能适用于不同的场景。

背景

上图是从平时工作场景碰到的 UI 效果截图过来的。上图中展示的 Tooltips 框基本上覆盖了常见的 UI 风格。简单的来归纳一下：

- 带边框的提示框
- 纯色（或带透明度纯色）的提示框
- 带内阴影（或外阴影）的提示框
- 带边框+渐变的提示框
- 带边框+透明度背景的提示框
- 提示框三角带圆角和阴影的提示框

可能还有我未碰到的提示框 UI 风格。面对这么多的 UI 风格，对于前端实现上来说是具有一定的挑战性，特别是多种效果组合在一起的。比如说，带有边框+内外阴影+渐变（或透明度）+圆角三角等。基本上组合了上图所提到的各种 UI 风格。

clip-path 方案

通常上图的实现是使用 CSS 画个尖角来拼接上去，比较优秀的方案如下图:

点击查看动图演示

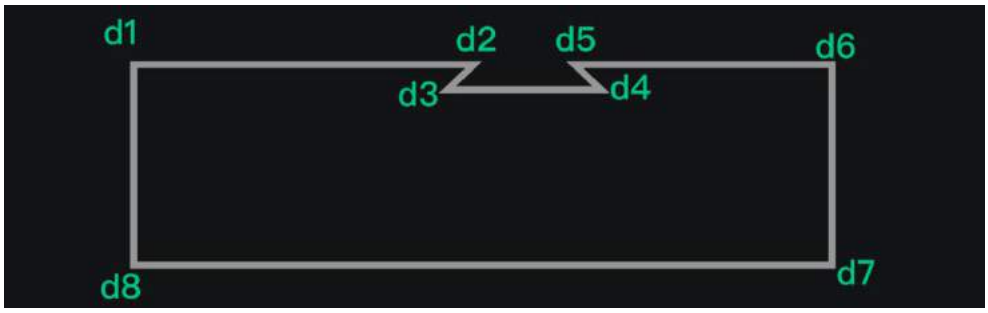
我们简单介绍下 clip-path 方案:

把提示框分成两个部分，一个是四方形，一个是三角形，然后两个拼接在一起组合成一个提示框。这样整个坐标示意图如下:

假设提示框的尺寸是 w x h，边框厚度是 h1，那么绘制带有缺口的时需要以下几个坐标点：

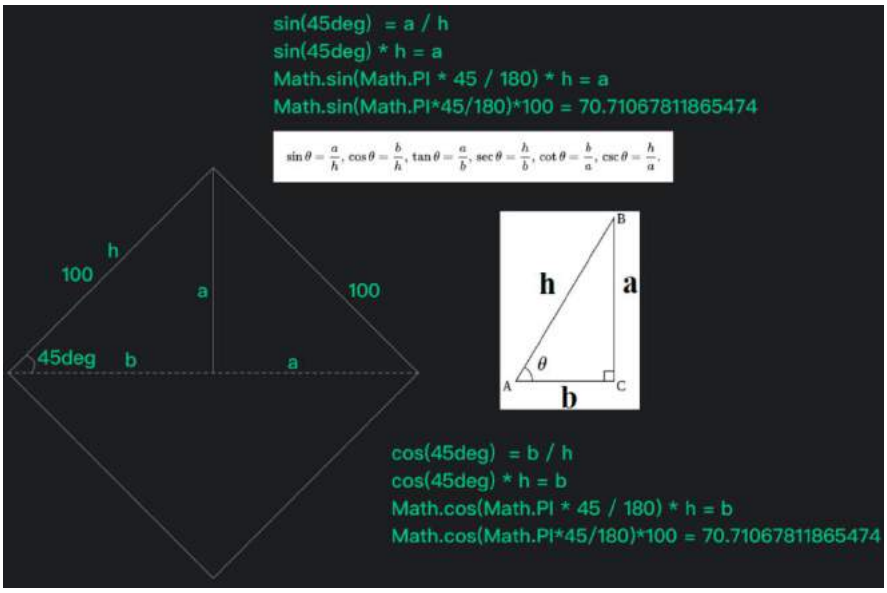
- d1 坐标(0, 0)
- d2 坐标((50% - b), 0)或((w / 2 - b), 0) 其中 b 是三角形对角边长度的一半，后面会介绍到
- d3 坐标((50% - b - h1), h1) 或 ((w / 2 - b - h1), h1)
- d4 坐标((50% + b + h1), h1) 或 ((w / 2 + b + h1), h1)
- d5 坐标((50% + b), 0)或 ((w / 2 + b), 0)
- d6 坐标(100%, 0) 或 (w, 0)
- d7 坐标(100%, 100%) 或 (w, h)
- d8 坐标(0, 100%) 或 (0, h)

坐标点放置到 clip-path 的 polygon()函数中，最终剪切之后的图形看上去像下图：




```
clip-path: polygon(
  0 0,
  calc(50% - 4px) 0,
  calc(50% - 7px) 2px,
  calc(50% + 7px) 2px,
  calc(50% + 4px) 0,
  100% 0,
  100% 100%,
  0 100%,
  0 0);
```

另外就是三角形的部分，如果我们的三角形是一个 10px x 10px 旋转 45deg 得到。根据一些三角函数的公式和已知的正方形边长就可以计算出正方形斜对角的长度：



clip-path 方案碰到问题

这个效果整体看起来还是不错的，但是细看就会发现在接缝处或许会存在有空隙、有重叠的问题，如下图：

采用 vw 方案后这类像素对不齐的问题也算是司空见惯了，同时第一个 Tooltips 由于是背景需要从左到右渐变的，此时尖角的渐变过度要和下方的渐变匹配上就更需要费力气了。

由于先前就遇到过此类 ToolTip 样式问题，告知视觉同学后，体贴的视觉同学修改了一版不带透明度的纯色提示框，然而视觉效果大打折扣。

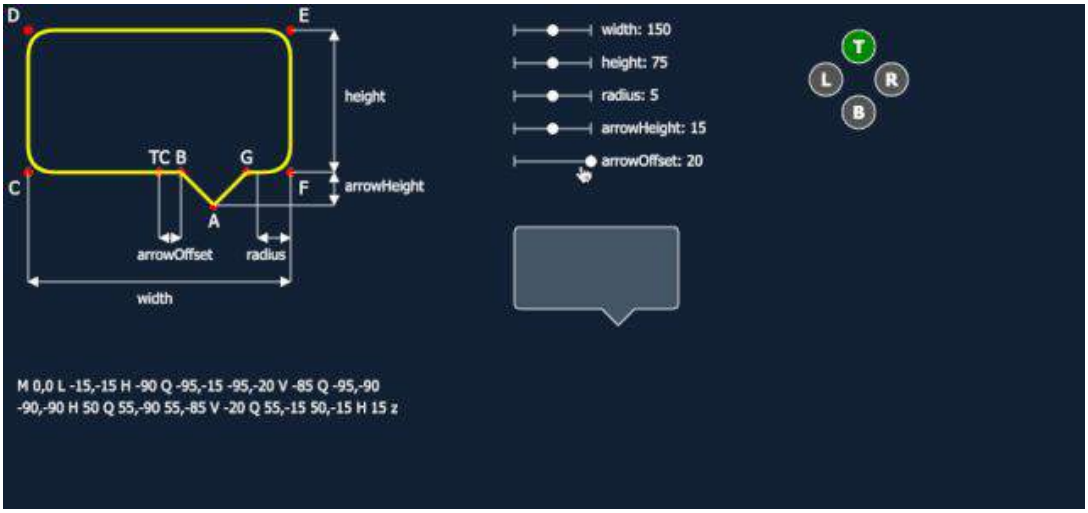
其实我们对于原先采用 CSS clip-path 的方案其实也存在很多的缺陷，它在面对带有阴影、背景透明或者渐变、带边框同时出现时就显出了实现成本高和效果一般的缺点。

SVG 方案

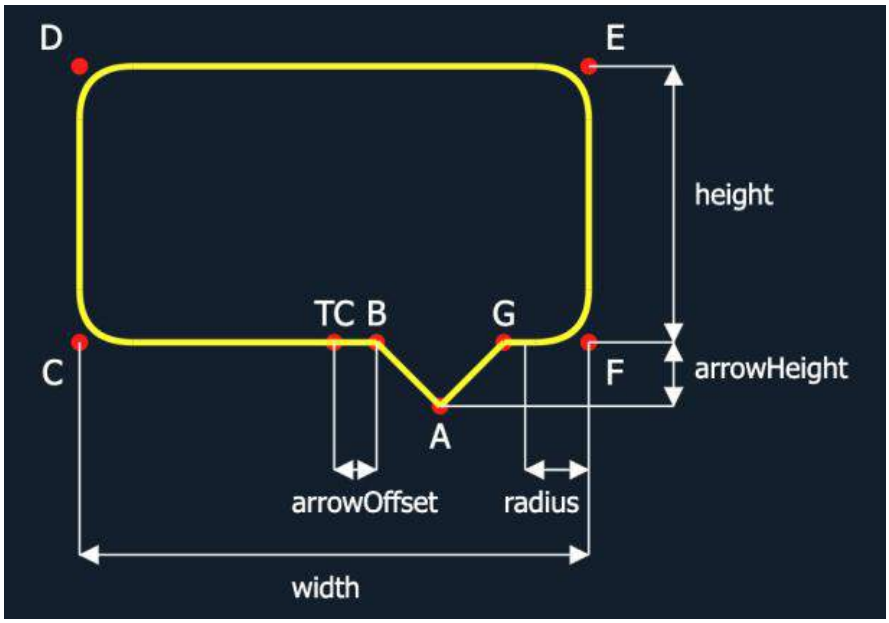
在讨论中我们想到 SVG 的 path 和这个提示框的样式天然的匹配（建议先了解下 path 的相关文档），查阅了相关的文档和资料后我们大致得到了使用 SVG 来实现的如下几个优点：

- 能轻松满足阴影、背景透明或者渐变、带边框的效果，甚至更为复杂多变的场景
- SVG 的 path 实现简单，并且代码量极小
- 可扩展性，可维护性

参考相关文章后，我们完善 Demo 工具如下：



点击查看动图演示



使用 Demo 工具，我们会得到 path 的数据大致如下:

```
M 0,0 L -15,-15 H -79 Q -84,-15 -84,-20 V -85 Q -84,-90-79,-90
H 61 Q 66,-90 66,-85 V -20 Q 66,-15 61,-15 H 15 z
```

通常使用 SVG 画 path 时用到命令如下表:

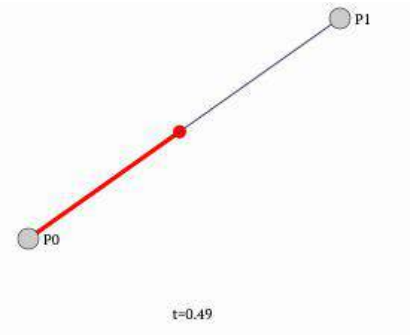
命令	名称	参数
M	moveto(移动到)	(x y)+
Z	closepath（关闭路径）	(none)
L	lineto（画线到）	(x y)+
H	horizontal lineto（水平线到）	x+
V	vertical lineto （垂直线到）	y+
C	curveto（三次贝塞尔曲线到）	(x1 y1 x2 y2 x y)+
S	smooth curveto（光滑三次贝塞尔曲线到）	(x2 y2 x y)+
Q	quadratic Bézier curveto（二次贝塞尔曲线到）	(x1 y1 x y)+
T	smooth quadratic Bézier curveto（光滑二次贝塞尔曲线到）	(x y)+
A	elliptical arc （椭圆弧）	(rx ry x-axis-rotation large-arc-flag sweep-flag x y)+
R	Catmull-Rom curveto* （Catmull-Rom 曲线）	x1 y1 (x y)+

贝塞尔曲线

在 SVG path 命令中我个人认为最精髓的部分是贝塞尔曲线，贝塞尔能画出各种令人愉悦的曲线。

贝塞尔曲线完全由其控制点决定其形状， n 个控制点对应着 n - 1 阶的贝塞尔曲线，并且可以通过递归的方式来绘制。我们先看下一次和二次贝塞尔曲线如何来绘制的：

一次曲线：

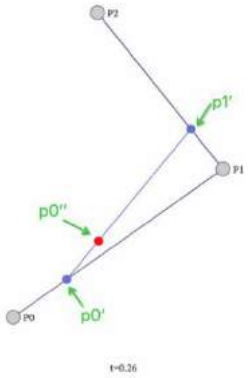


点击查看动图演示

一条直线上，随着时间 t 的变化，红色线段的那个点的坐标公式应该如下：

$$\mathbf{B}(t) = \mathbf{P}_0 + (\mathbf{P}_1 - \mathbf{P}_0)t = (1-t)\mathbf{P}_0 + t\mathbf{P}_1,\ t \in [0,1]$$

二次贝塞尔曲线：

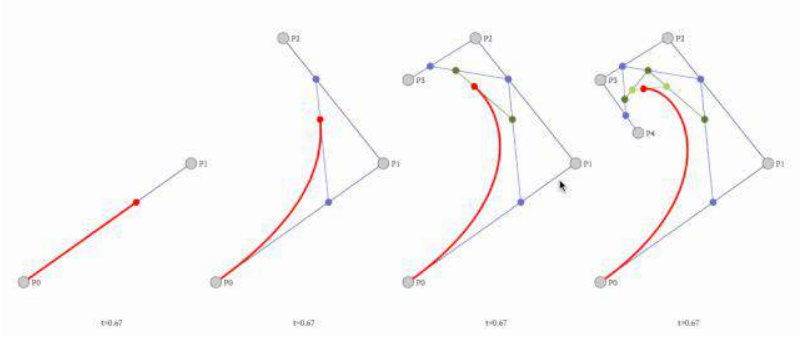


p0、p1、p2 是 3 个不共线的点，依次用线段连接，此时随意取线段 p0p1 上的一个点 p0' ，如上图：我们的 p0'点在 p0p1 线段的 0.26 处（t=0.26）,此刻 p1p2 线段相同比列取 p1'点，此时 p0'和 p1'连接后形成线段 p0'p1'，在按照如上比列进行取值 p0''，这时候就确定了二次贝塞尔曲线的一个点。

通过一番巴拉巴拉牛逼的推导后，二次贝塞尔曲线公式为：

$$\mathbf{B}(t) = (1-t)^2\mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2,\ t \in [0,1]。$$

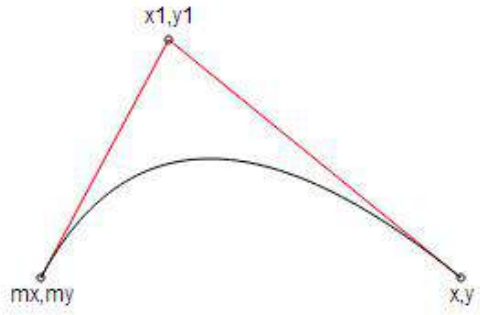
N 次贝塞尔可以认为是如上取值方式的迭代过程，可以通过下图直观的感受到 1~4 次曲线随着时间 t 的变化过程，具体 N 次贝塞尔曲线的公式可以参考下方关于曲线的文章。



点击查看动图演示

SVG 中的 Q 命令

回到我们的 ToolTips 话题，其中的圆角是可以通过二次贝塞尔曲线来实现，SVG 中 Q 命令就是来实现二次贝塞尔曲线的，SVG 中 Q 命令的示例图如下：



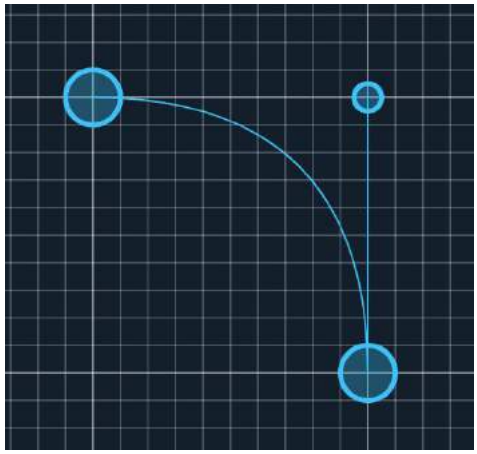
对应的指令,其中 x1,y1 就是我们上面提到的 p1 点:

```
Q x1 y1, x y
```

二次贝塞尔曲线 Q 示例如下:

```
<svg width="190px" height="160px" version="1.1" xmlns="http://www.w3.org/2000/svg">
<path d="M10 80 Q 95 10 180 80" stroke="black" fill="transparent"/>
</svg>
```

通过设置起始点和调整控制点 p1 我们能得到我们想要的圆角，如下图所示，小圆点为我们的 p1 控制点

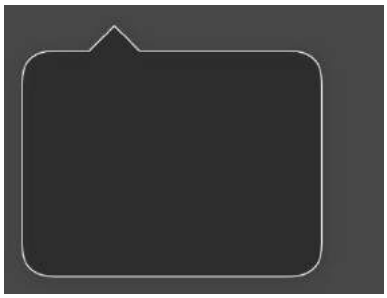


样式设置

实现了上方的 SVG 后接下来的透明、背景渐变、阴影、边框的设置就都不成问题了。

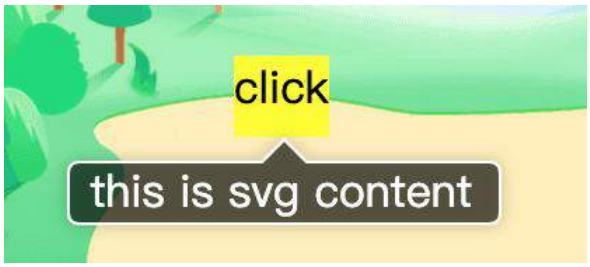
背景透明

```
path {
fill: rgba(0,0,0, .3);
storke: #ffffff;
storke-width: 1px
}
```



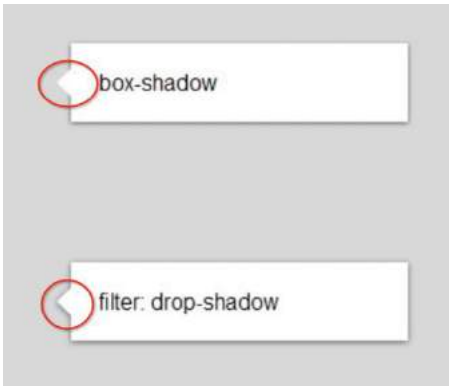
阴影

```
svg {
filter: drop-shadow(2px 4px 6px black)
}
```



关于为何使用 drop-shadow 来实现阴影，可以看下图使用了 box-shadow 和 drop-shadow 效果区别。

使用 box-shadow 的时候我们的尖角部分没有阴影，气泡框部分是有阴影的，就会出现下图所示的情况，而使用 drop-shadow 就能符合我们尖角和气泡框都有阴影的要求。



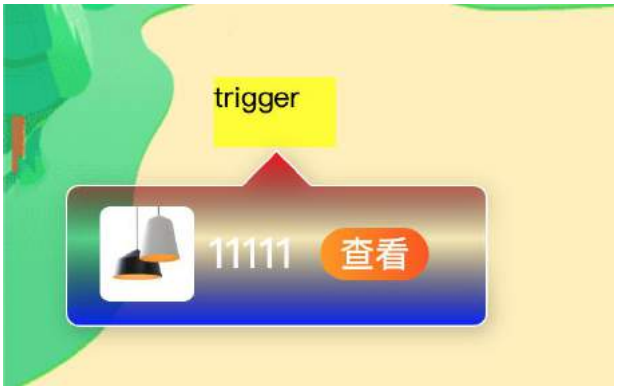
背景渐变

SVG 不仅支持简单的填充，还支持线性渐变和径向渐变以及图形纹理等。为了让渐变能被重复使用，渐变内容需要定义在标签内部。

如下图是径向渐变的演示：

```
<svg width="120" height="240" version="1.1" xmlns="http://www.w3.org/2000/svg">
<defs>
  <linearGradient id="Gradient2" x1="0" x2="0" y1="0" y2="1">
  <stop offset="0%" stop-color="red"/>
  <stop offset="50%" stop-color="black" stop-opacity="0"/>
  <stop offset="100%" stop-color="blue"/>
    </linearGradient>
</defs>
<rect x="10" y="120" rx="15" ry="15" width="100" height="100" fill="url(#Gradient2)"/>
</svg>
```

将这个渐变作用到我们提示框后可以看到如下图的效果，终于不用辛辛苦苦的处理尖角的渐变衔接问题了。

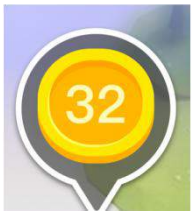


更多

SVG 同时也支持纹理叠加效果，具体感兴趣的可以自行去研究下。

需求还没完

上面方案落地到项目中后，可能是我们不经意感动了设计师，最近的需求视觉稿中我们发现其中涉及到的 Tooltips 样式已经越发令人惊艳。简单列举如下两个样式：

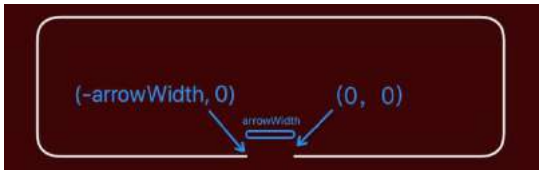


第一版方案我们基于 Demo 工具演示我们已经产出了 ToolTips 的 SDK，我们使用的单个参数 arrowHeight 传入来生成尖角。在应付上方两个样式是不可能的，尖角样式多变，如何来扩展性和易用性成为了一个问题，不可能多变的尖角样式都开发一个 SDK。

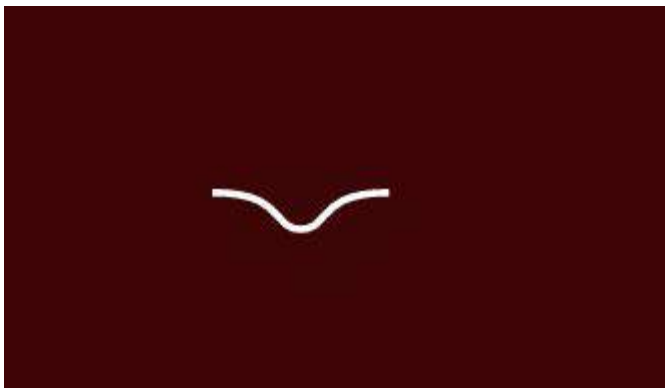
方案改进

要应付多变的气泡尖角一定要想办法把尖角抽离出原先的气泡外层路径，生成尖角路径后在整合到气泡上形成一个完整的闭合路径。

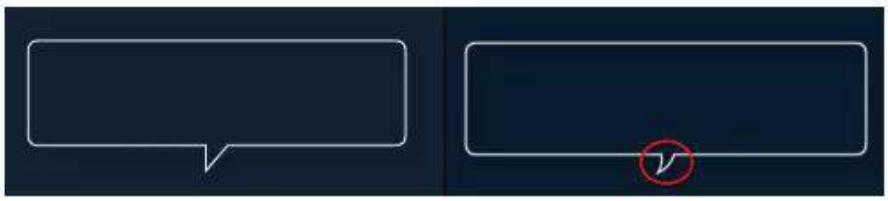
为了简单处理数值，我将原先的尖角 (0，0) 坐标定义更换到下方图示点：



所以接下来尖角可以自由设计了，只要保证从(0，0)出发最后回到（-arrowWidth,0）就行了，如下是一个尖角的路径：（M 0 0 C -10 0 -8 5 -12 5 S -14 0 -24 0）



通过设计不同的尖角路径我们就能组合成不同的气泡样式：



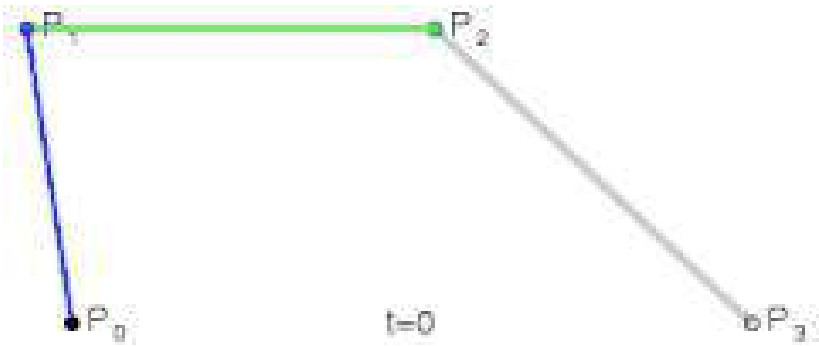
上方右侧的尖角气泡最终给出的路径字符串如下，其中 Q -2 7 -9 10 Q -6 5 -7 0 这一段即为我们的尖角路径：

```
M 0 0
Q -2 7 -9 10 Q -6 5 -7 0
H -110
Q -116,0 -116,-6
V -56
Q -116,-62 -110,-62
H 101
Q 107,-62 107,-56
V -6Q 107,0 101,0
H 0 z
```

从上方简短的路径能看出，我们的尖角路径是完整的整合在整个 SVG 气泡路径中的，所以就不会担心会出现 CSS 的 clip-path 方案的问题。

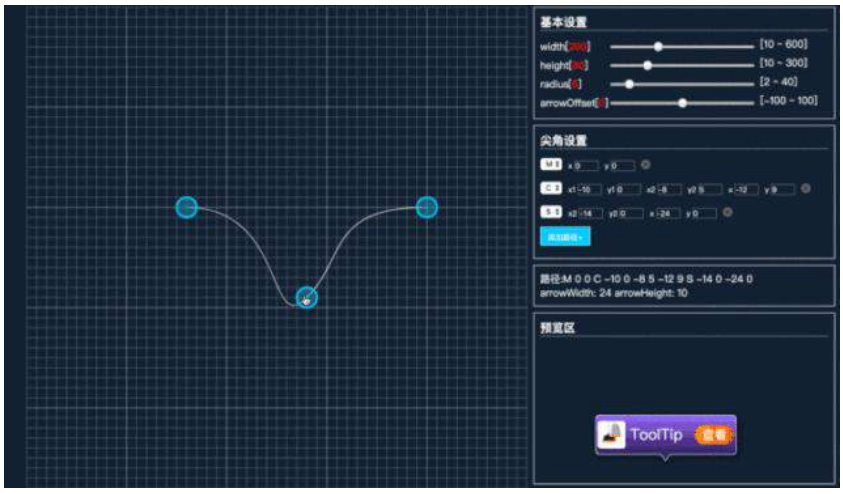
可视化工具

方案看起来好像已经搞定了需求中的尖角样式，然而你能会说这尖角路径是如何产生，难道需要通过强大的数学能力推导出来？如下三次贝塞尔曲线就已经不敢直视了，更何况四次、五次...



点击查看动图演示

所以想配合的我们一定要产出可视化工具来实现这路径生成过程，得益于 D3.js 工具库操作 SVG 方面的强大功能，我们开发完的 生成工具地址 (https://market.m.taobao.com/app/fdilab/svg-tool-demo/pages/index/index.html) 如下，



点击查看动图演示

对于熟悉 SVG 的 path 命令的同学来说这个操作不难，如果不熟悉的推荐看下下方的参考文章，了解了曲线命令后就能画出圆滑的曲线。

总结

至此在 ToolTips 这块基本已经满足了设计的需求，同时也沉淀了 SVG 路径生成工具。使用 SVG 来实现 ToolTips 能覆盖 CSS clip-path 不能完美解决的几个场景。在此特别感谢大漠老师的指导。

参考文章：

D3 官网
曲线篇：贝塞尔曲线
Tooltips using SVG Path
SVG 渐变
深度掌握 SVG 路径 path 的贝塞尔曲线指令
css drop-shadow

10 个 VS Code 使用技巧

作者 | 梧忌
出品 | 淘系技术

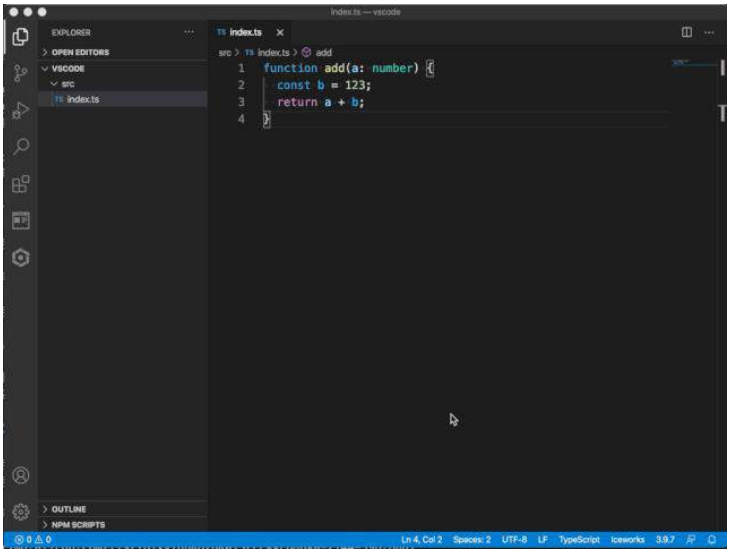
VS Code 作为广受好评的开发工具，已经被越来越多的开发者使用。然而，你真的了解 VS Code 了吗。

笔者经常帮一些同学 One-on-One 地解决问题，在看部分同学使用 VS Code 的时候，有些蹩脚，实际上一些有用的技巧能够提高我们的日常工作效率，本文整理了 10 个 VS Code 使用技巧，希望可以帮助到有需要的同学。

重构代码

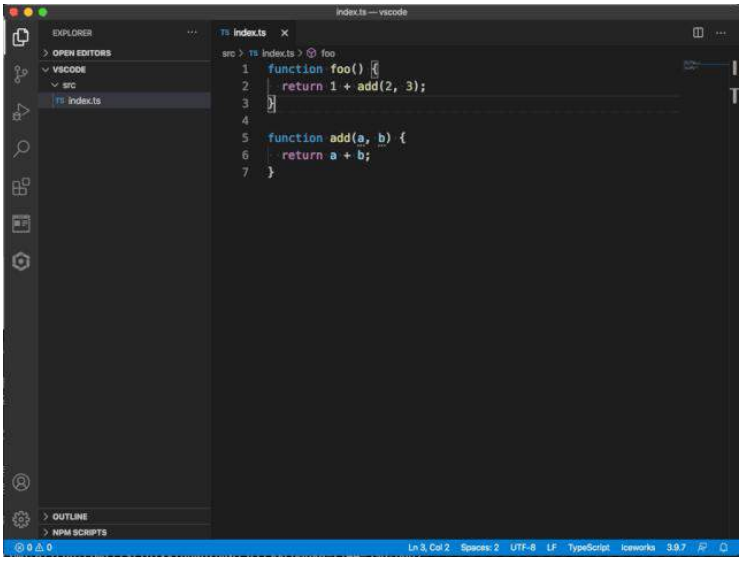
VS Code 提供了一些快速重构代码的操作，例如：

将一整段代码提取为函数：选择要提取的源代码片段，然后单击做成槽中的灯泡查看可用的重构操作。代码片段可以被提取到一个新方法中，或者在不同的范围内（当前闭包、当前函数内、当前类中、当前文件内）提取到一个新函数中。在提取重构期间，VS Code 会引导为该函数进行命名。



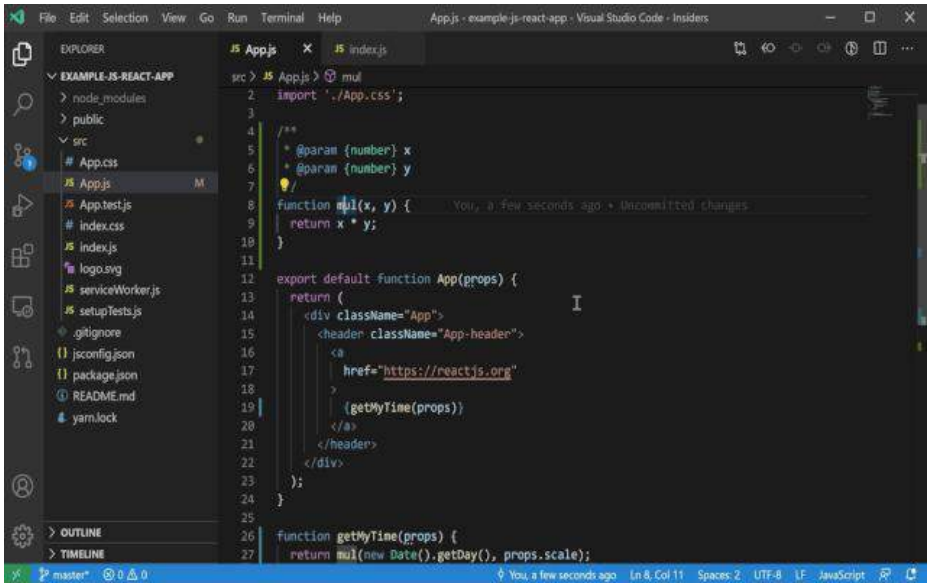
点击查看动图演示

将表达式提取到常量：为当前选定的表达式创建新的常量。



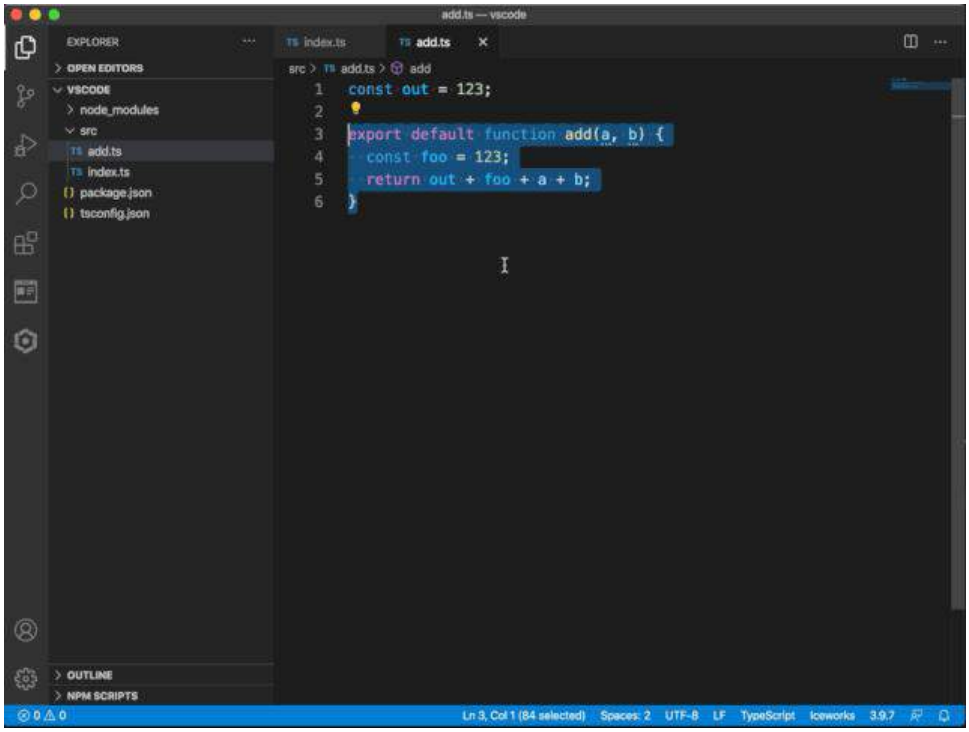
点击查看动图演示

移动到新的文件：将指定的函数移动到新的文件，VS Code 将自动命名并创建文件，且在当前文件内引入新的文件。



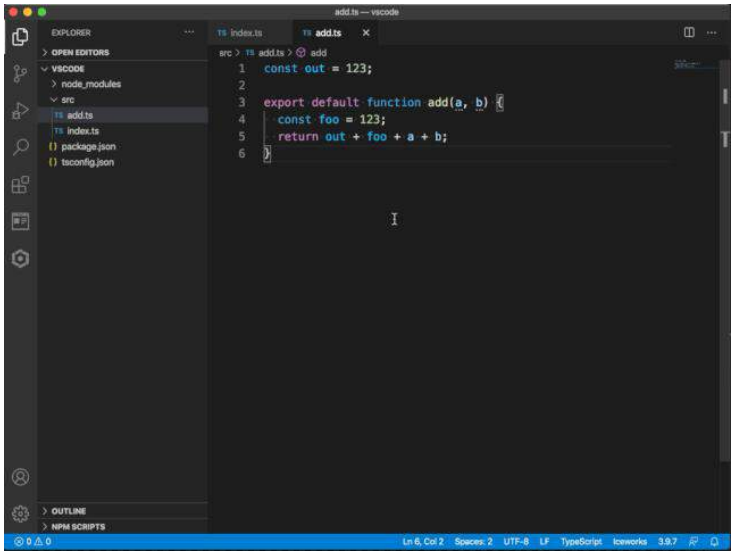
点击查看动图演示

转换导出方式：export const name 或者 export default。



点击查看动图演示

合并参数：将函数的多个参数合并为单个对象参数。

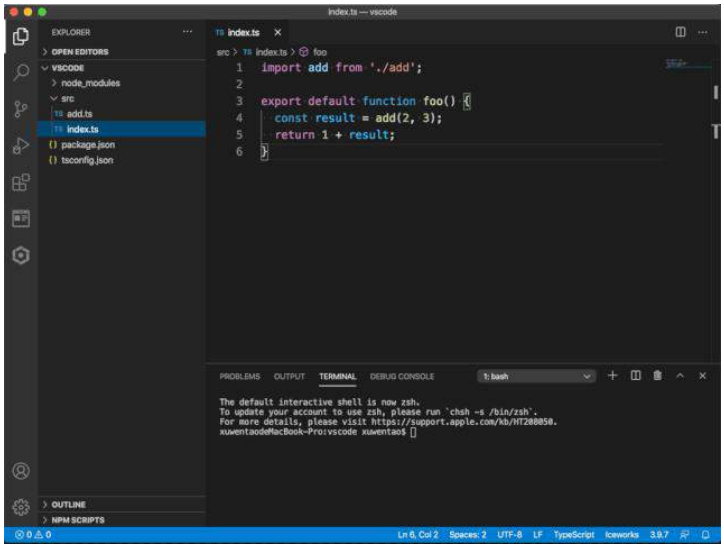


点击查看动图演示

参考： 重构操作(https://code.visualstudio.com/docs/editor/refactoring)、JS/TS 重构操作(https://code.visualstudio.com/Docs/languages/typescript#_refactoring)

自定义视图布局

VS Code 的布局系统非常灵活，可以在工作台上的活动栏、面板中移动视图。

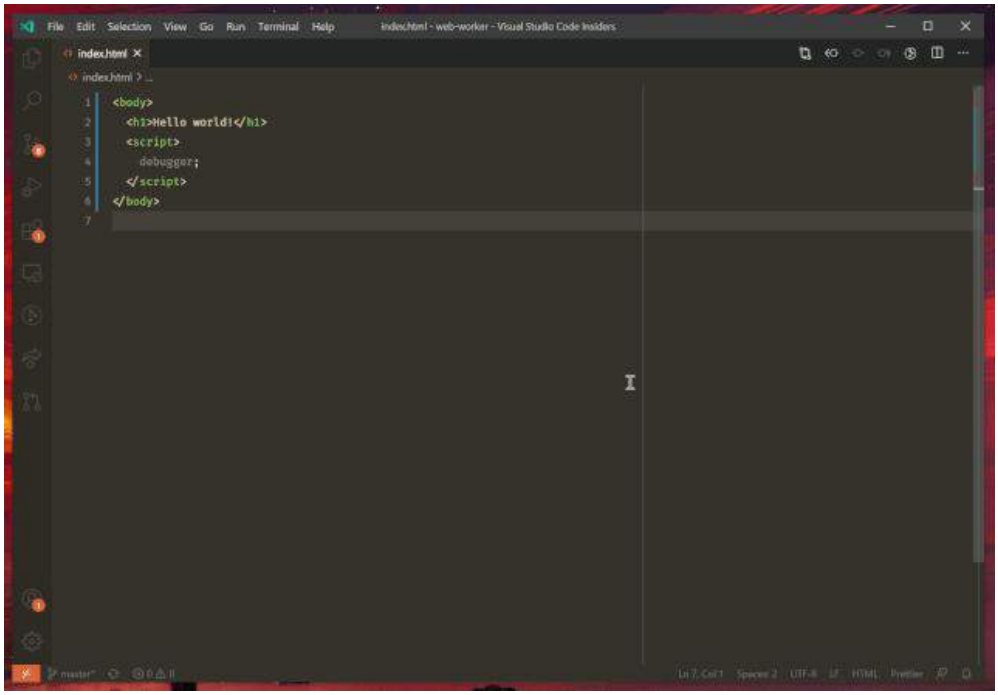


点击查看动图演示

参考：重新排列视图(https://code.visualstudio.com/updates/v1_45?ref=codebldr#_dynamic-view-icons-and-titles)

快速调试代码

在 VS Code 内调试 JS/TS 代码非常简单，只需要使用 Debug: Open Link 命令即可。这在调试前端或 Node 项目时非常有用，这类型的项目通常会启动一个本地服务，这时候只需要将本地服务地址填写到 Debug: Open Link 输入框中即可。

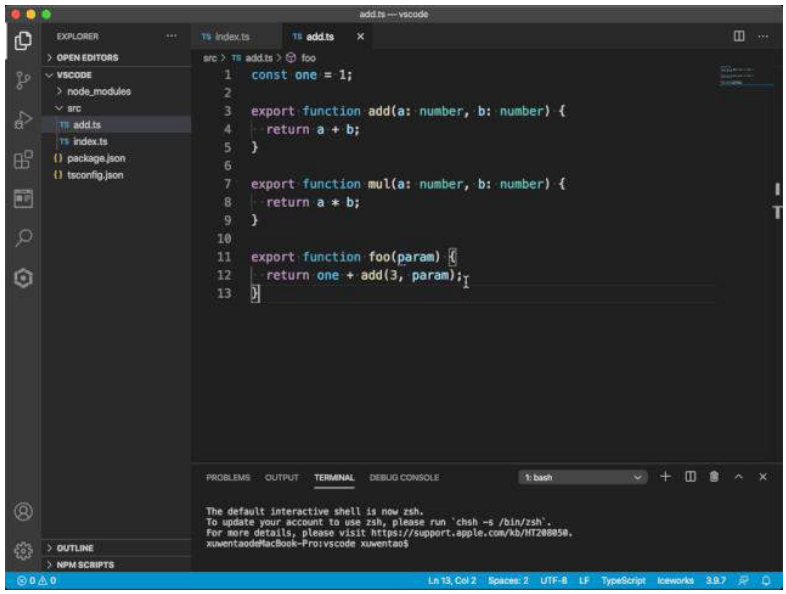


点击查看动图演示

参考：Debug(https://code.visualstudio.com/docs/editor/debugging)

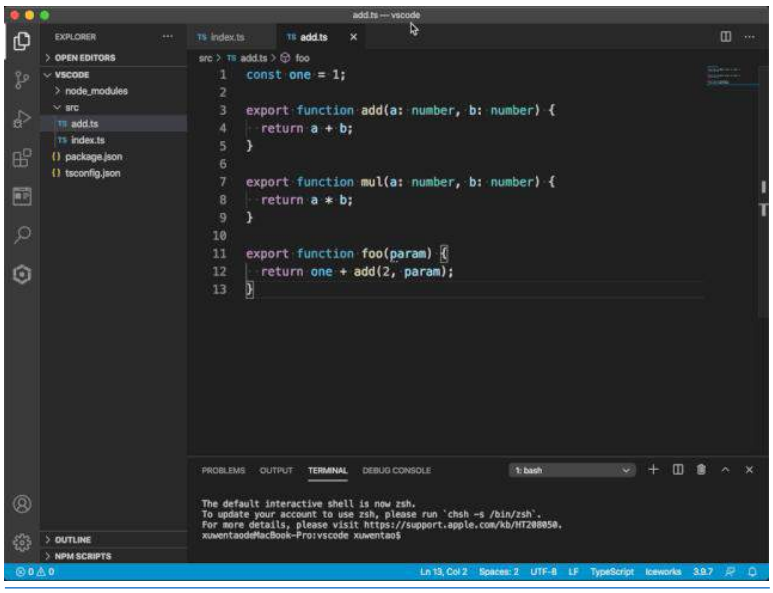
查看和更新符号的引用

查看符号的引用、快速修改引用的上下文：例如，快速预览某个函数在哪些地方被调用了及其调用时上下文，还可以在预览视图中更新调用上下文的代码。



点击查看动图演示

重命名符号及其引用：接着上面的例子，如果想更新函数名以及所有调用，怎么实现？按 F2 键，然后键入所需的新名称，再按 Enter 键进行提交。符号的所有引用都将被重命名，该操作还是跨文件的。

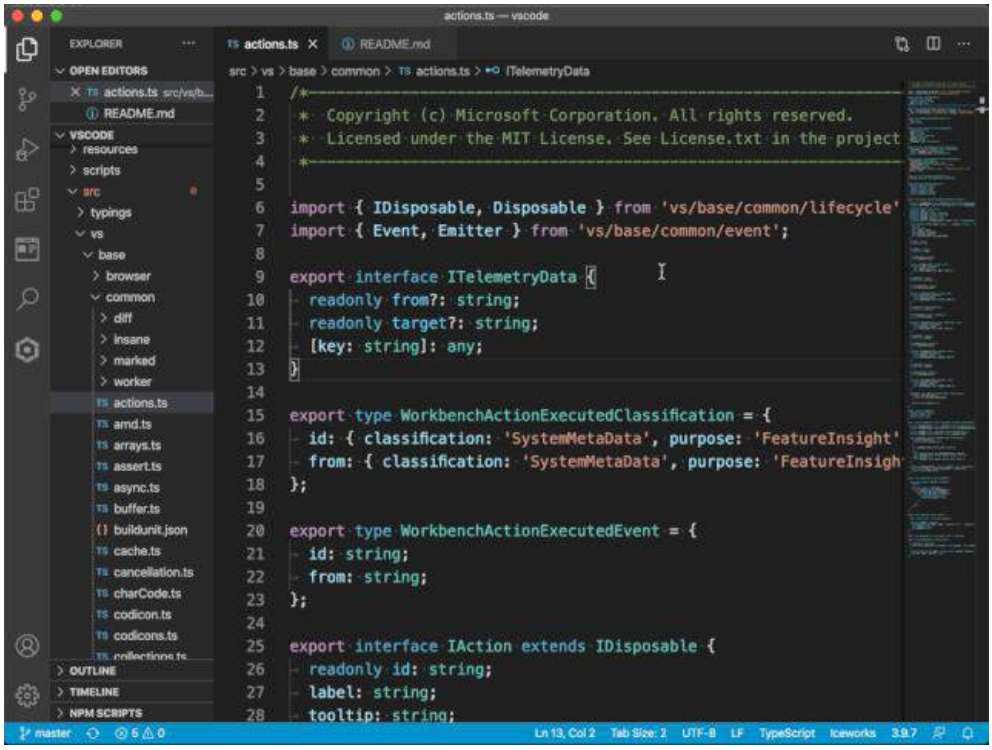


点击查看动图演示

参考：Peek(https://code.visualstudio.com/docs/editor/editingevolved#_peek)、Rename Symbol(https://code.visualstudio.com/docs/editor/editingevolved#_rename-symbol)

符号导航

在查看一个长文件的时候，代码定位会是非常痛苦的事情。一些开发者会使用 VS Code 的小地图，但其实还有更便捷的方法：可以使用 ⇧ ⌘O 快捷键唤起符号导航面板，在当前编辑的文件中通过符号快速定位代码。在输入框中键入字符可以进行筛选，在列表中通过箭头来进行上下导航。这种方式对于 Markdown 文件也非常友好，可以通过标题来快速导航。

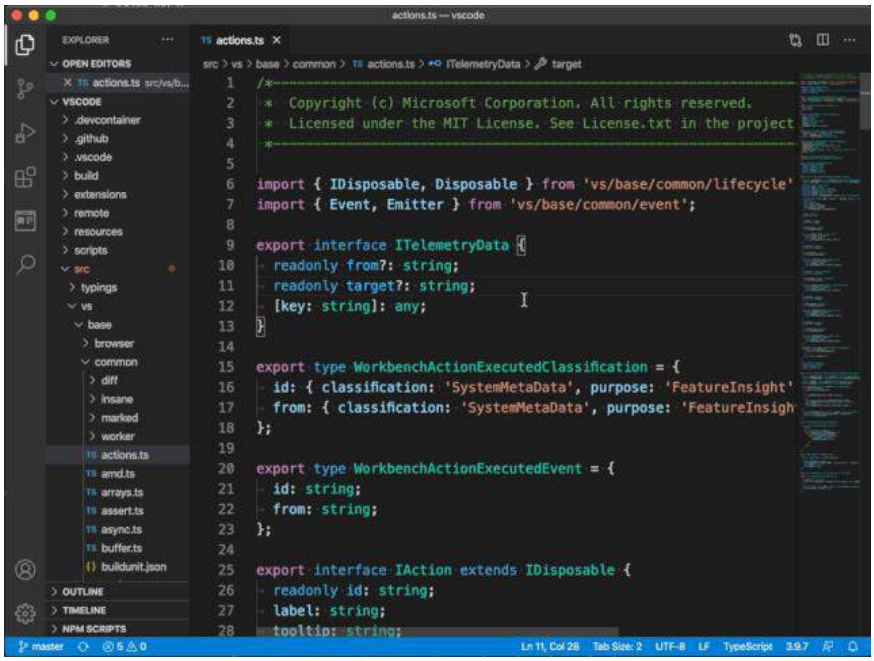


点击查看动图演示

参考：Go to Symbol(https://code.visualstudio.com/docs/editor/editingevolved#_go-to-symbol)

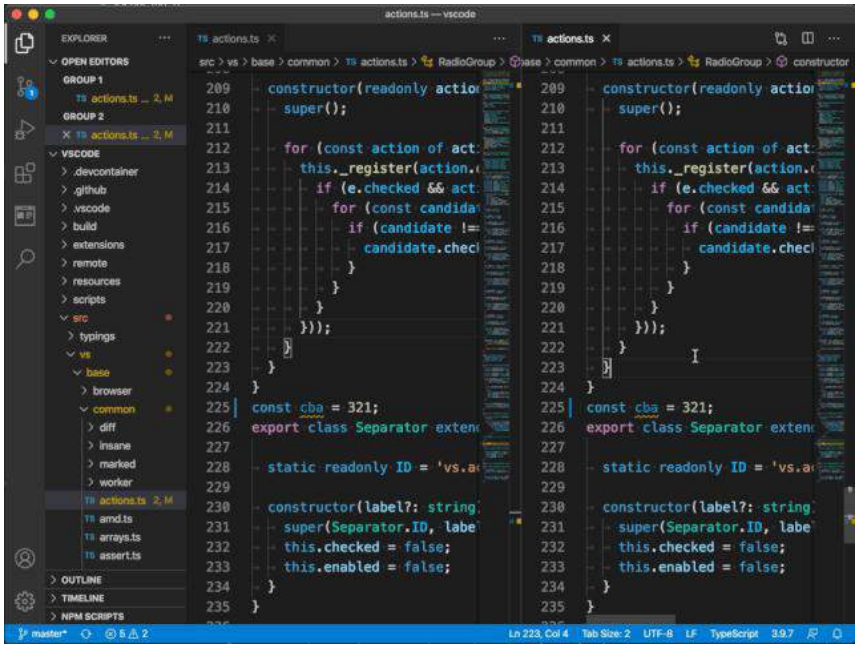
拆分编辑器

当对内容特别多的文件进行编辑的时候，经常需要在上下文中进行切换，这时候可以通过拆分编辑器来使用两个编辑器更新同一个文件：按下快捷键 ⌘\ 将活动编辑器拆分为两个。



点击查看动图演示

可以继续无尽地拆分编辑器，通过拖拽编辑器组的方式排列编辑器视图。

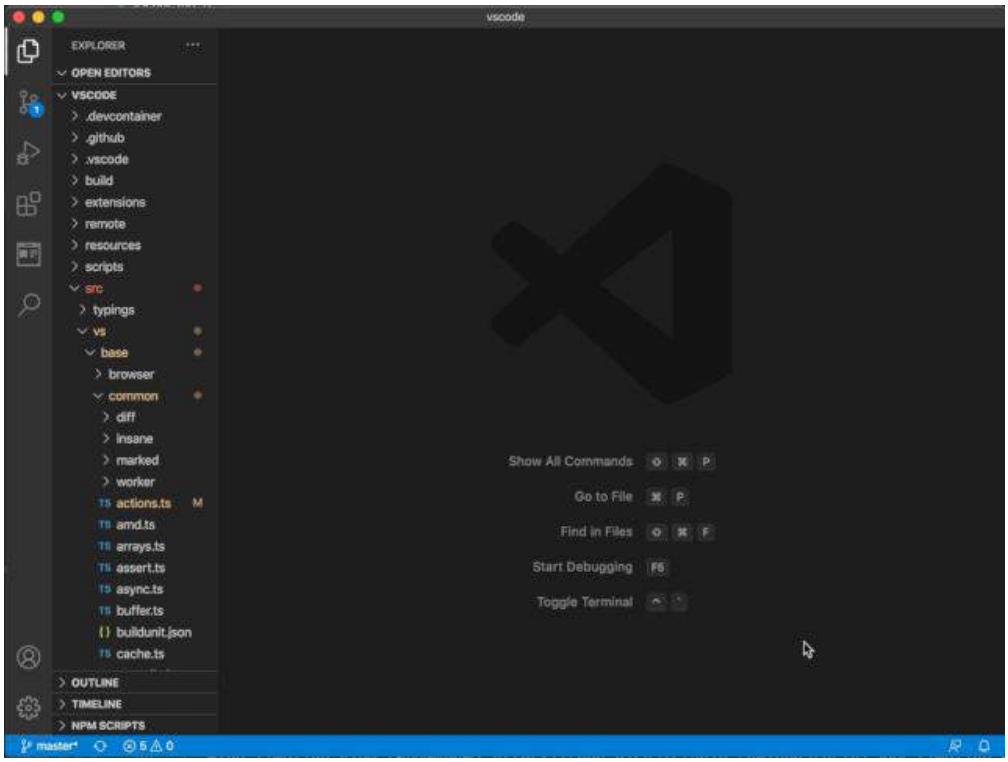


点击查看动图演示

参考：Side by side editing(https://code.visualstudio.com/docs/getstarted/userinterface#_side-by-side-editing)

重命名终端

VS Code 提供了集成终端，可以很方便地快速执行命令行任务。用得多了经常会打开多个终端，这时候给终端命名可以提高终端定位的效率。



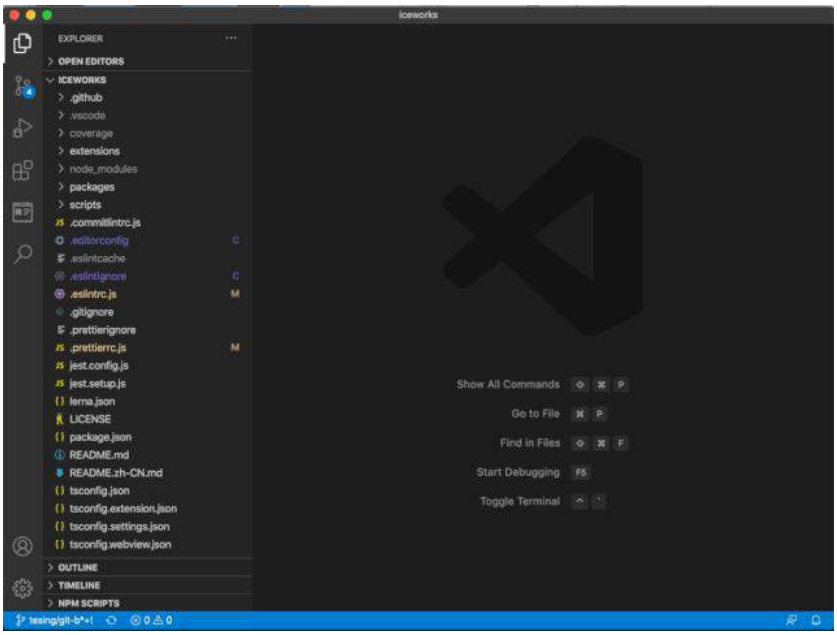
点击查看动图演示

参考：Rename terminal sessions(https://code.visualstudio.com/docs/editor/integrated-terminal#_rename-terminal-sessions)

Git 操作

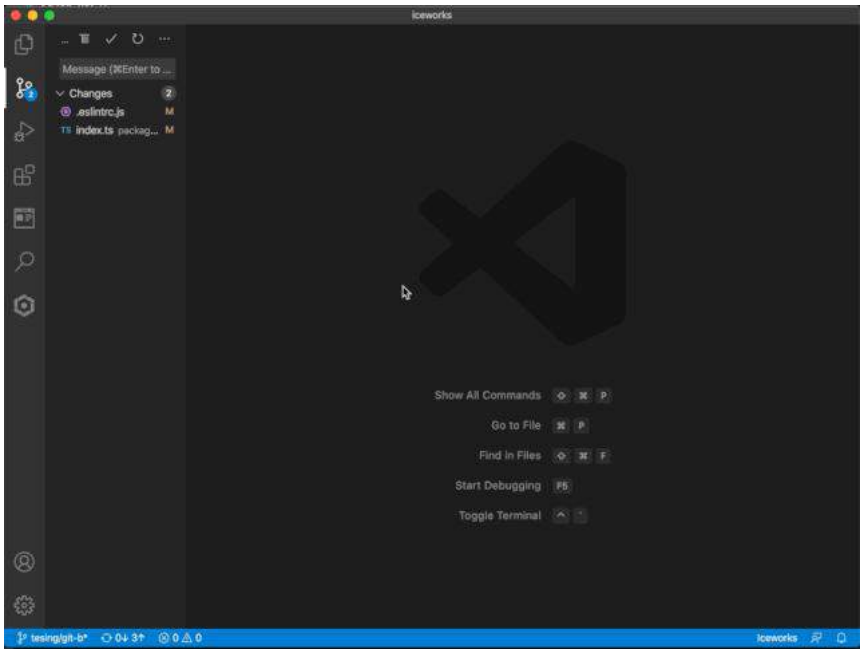
VS Code 内置了 Git 源代码管理功能，提供了一些便捷的 Git 操作方式。例如：

解决冲突：VS Code 会识别合并冲突，冲突的差异会被突出显示，并且提供了内联的操作来解决冲突。



点击查看动图演示

暂存或撤销选择的代码行：在编辑器内可以针对选择的行来撤销修改、暂存修改、撤销暂存。

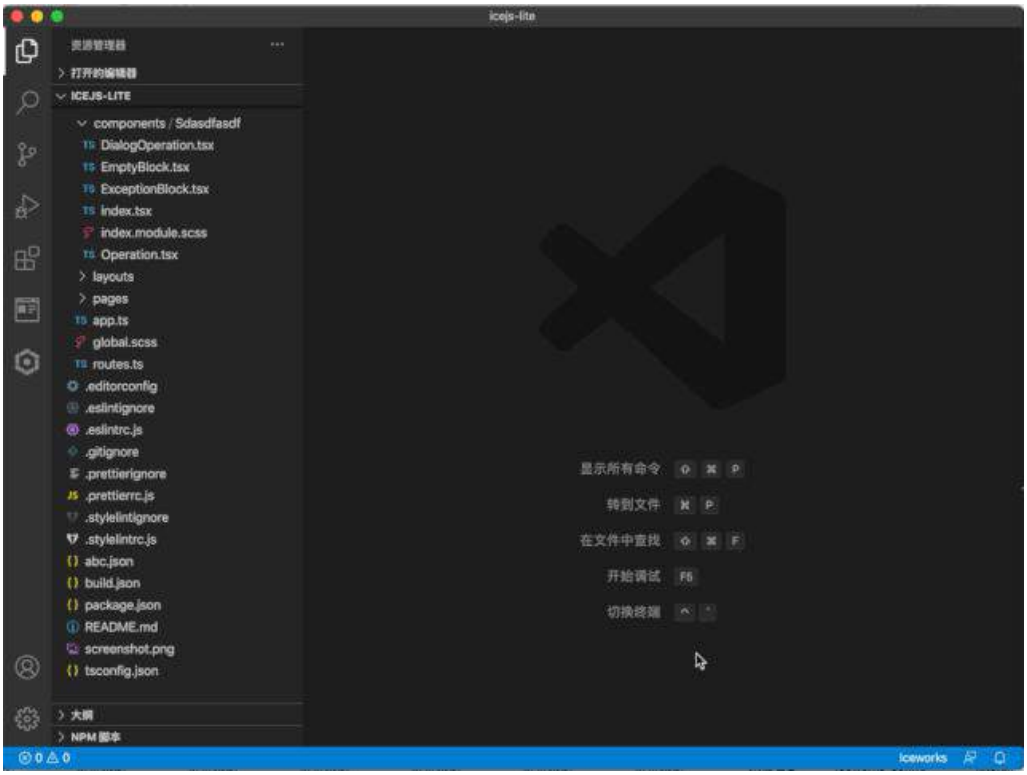


点击查看动图演示

参考：Using Version Control in VS Code(https://code.visualstudio.com/docs/editor/versioncontrol)

搜索结果快照

VS Code 提供了跨文件搜索功能，搜索结果快照可以提供更多的搜索结果的信息，例如代码所在行码、搜索关键字的上下文，并且可以对搜索结果进行编辑和保存。

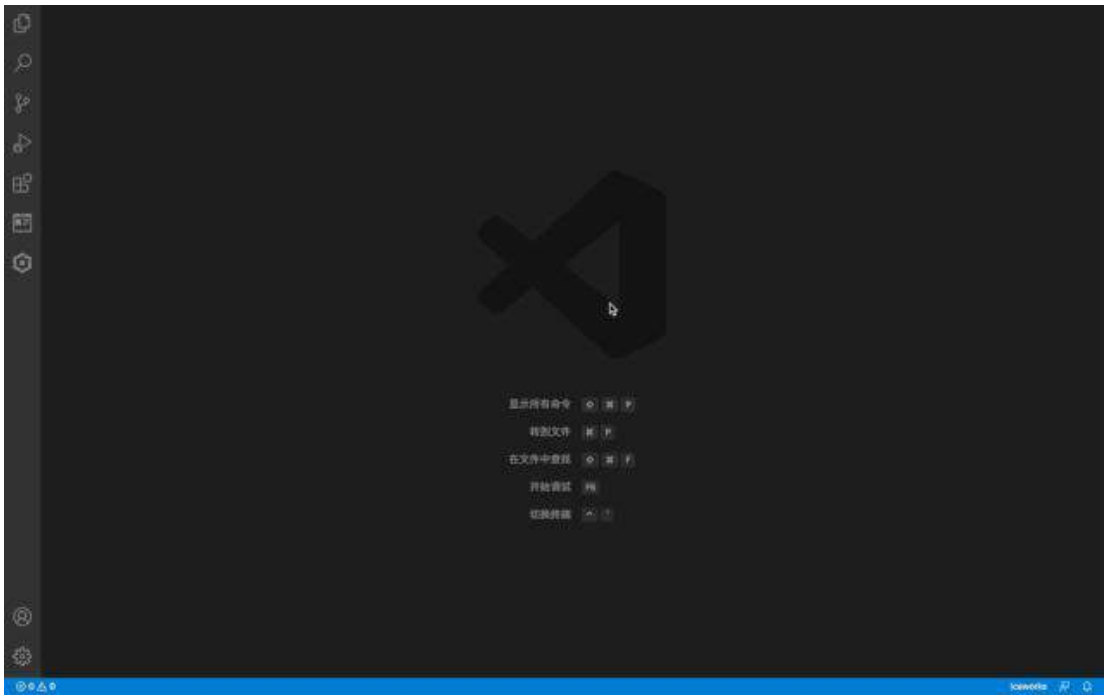


点击查看动图演示

参考：Search Editors(https://code.visualstudio.com/updates/v143#search-editors)

可视化搭建页面

在 VS Code 中可以通过可视化搭建的方式生成 Web 页面，这是通过安装 VS Code 的 Iceworks(https://marketplace.visualstudio.com/items?itemName=iceworks-team.iceworks) 插件实现的。安装插件后，通过 ⇧ ⌘P 唤起命名面板，在命令面板中输入『可视化搭建』即可唤起可视化搭建界面，在界面内通过选择网页元素、进行拖拽布局、设置元素样式和属性来搭建页面，最后点击『生成代码』就可以生成 React 代码。



点击查看动图演示

参考：Iceworks 可视化搭建(https://ice.work/docs/iceworks/guide/visual-construction)

一套代码帮助前端开发了解机器学习

作者 | 苏川
出品 | 淘系技术

可能有些同学学习机器学习的时候比较迷茫，不知道该怎么上手，看了很多经典书籍介绍的各种算法，但还是不知道怎么用它来解决问题，就算知道了，又发现需要准备环境、准备训练和部署的机器，啊，好麻烦。

今天，我来给大家介绍一种容易上手的方法，给你现成的样本和代码，按照步骤操作，就可以在自己的 Mac 上体验运用机器学习的全流程啦~~~

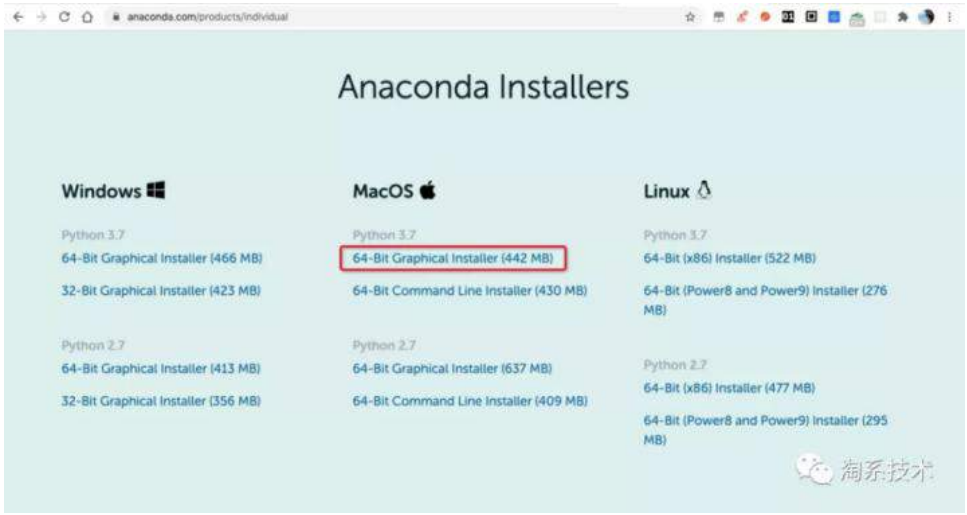
下面的 Demo， 最终的效果是给定一张图片，可以预测图片的类别。比如我们训练模型用的样本是猫啊狗啊，那模型能学到的认识的就是猫啊狗啊， 如果用的训练样本是按钮啊搜索框啊，那模型能学到的认识的就是这个按钮啊搜索框啊。

如果想了解用机器学习是怎么解决实际问题的，可以看这篇：如何使用深度学习识别 UI 界面组件？从问题定义、算法选型、样本准备、模型训练、模型评估、模型服务部署、到模型应用都有介绍。

环境准备

安装 Anaconda

下载地址：https://www.anaconda.com/products/individual



安装成功后，在终端命令行执行以下命令，使环境变量立即生效：

```
$ source ~/.bashrc
```

可以执行以下命令，查看环境变量：

```
$ cat ~/.bashrc
```

可以看到 anaconda 的环境变量已经自动添加到 .bashrc 文件了。

```
# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/Applications/anaconda3/bin/conda' 'shell.bash' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/Applications/anaconda3/etc/profile.d/conda.sh" ]; then
        . "/Applications/anaconda3/etc/profile.d/conda.sh"
    else
        export PATH="/Applications/anaconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<
```


执行以下命令：

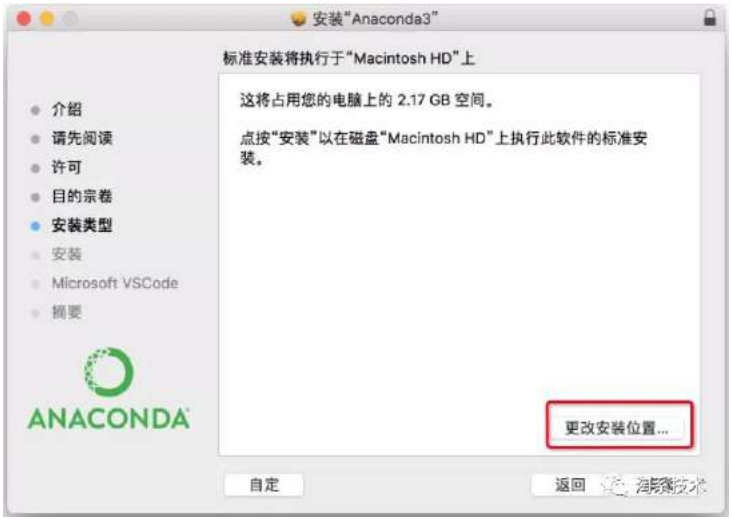
```
$ conda list
```

可以看到 Anaconda 中有很多已经安装好的包，如果有使用到这些包的就不需要再安装了，python 环境也装好了。

```
(base) chang:Desktop chang$ conda list
# packages in environment at /Applications/anaconda3:
#
# Name                    Version                   Build  Channel
_ipyw_jlab_nb_ext_conf    0.1.0                     py37_0
alabaster                 0.7.12                    py37_0
anaconda                  2020.02                   py37_0
anaconda-client           1.7.2                     py37_0
anaconda-navigator        1.9.12                    py37_0
anaconda-project          0.8.4                       py_0
applaunchservices         0.2.1                       py_0
appnope                   0.1.0                     py37_0
appscript                 1.1.0            py37h1de35cc_0
argh                      0.26.2                    py37_0
asn1crypto                1.3.0                     py37_0
astroid                   2.3.3                     py37_0
astropy                   4.0              py37h1de35cc_0
atomicwrites              1.3.0                     py37_1
attrs                     19.3.0                      py_0
autopep8                  1.4.4                       py_0
babel                     2.8.0                       py_0
backcall                  0.1.0                     py37_0
backports                 1.0                         py_2
backports.functools_lru_cache 1.6.1                       py_0
backports.shutil_get_terminal_size 1.0.0                       py37_2
backports.tempfile        1.0                         py_1
backports.weakref         1.0.post1                   py_1
beautifulsoup4            4.8.2                     py37_0
bitarray                  1.2.1            py37h1de35cc_0
bkcharts                  0.2                       py37_0
blas                      1.0                         mkl
bleach                    3.1.0                     py37_0
blosc                     1.16.3               hd9629dc_0
bokeh                     1.4.0                     py37_0
boto                      2.49.0                    py37_0
```

淘系技术

注意：如果安装失败，重新安装，在提示安装在哪里时，选择「更改安装位置」，安装位置选择其他地方不是用默认的，安装在哪里自己选择，可以放在「应用程序」下。



安装相关依赖

anaconda 中没有 keras、tensorflow 和 opencv-python， 需要单独安装。

```
$ pip install keras
$ pip install tensorflow
$ pip install opencv-python
```

样本准备

这里只准备了 4 个分类：button、keyboard、searchbar、switch，每个分类 200 个左右的样本。

模型训练

开发训练逻辑

新建一个项目 train-project，文件结构如下：

```
.├── CNN_net.py
├── dataset
├── nn_train.py
└── utils_paths.py
```

入口文件代码如下，这里的逻辑是将准备好的样本输入给图像分类算法SimpleVGGNet，并设置一些训练参数，例如学习率、Epoch、Batch Size，然后执行这段训练逻辑，最终得到一个模型文件。

```
# nn_train.py
from CNN_net import SimpleVGGNet
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from keras.optimizers import SGD
from keras.preprocessing.image import ImageDataGenerator
import utils_paths
import matplotlib.pyplot as plt
from cv2 import cv2
import numpy as np
import argparse
import random
import pickleimport os
# 读取数据和标签
print("------开始读取数据------")
data = [ ]
labels = [ ]
# 拿到图像数据路径，方便后续读取
imagePaths = sorted(list(utils_paths.list_images('./dataset')))
random.seed(42)
random.shuffle(imagePaths)
image_size = 256
# 遍历读取数据
for imagePath in imagePaths:
    # 读取图像数据
    image = cv2.imread(imagePath)
    image = cv2.resize(image, (image_size, image_size))
    data.append(image)
        # 读取标签
    label = imagePath.split(os.path.sep)[-2]
    labels.append(label)
data = np.array(data, dtype="float") / 255.0
labels = np.array(labels)
# 数据集切分
(trainX, testX, trainY, testY) = train_test_split(data,labels, test_size=0.25, random_state=42)
# 转换标签为 one-hot encoding 格式
```

```
    lb = LabelBinarizer()
    trainY = lb.fit_transform(trainY)
    testY = lb.transform(testY)

    # 数据增强处理
    aug = ImageDataGenerator(
        rotation_range=30,

        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode="nearest")
    # 建立卷积神经网络
    model = SimpleVGGNet.build(width=256, height=256, depth=3,classes=len(lb.classes_))
    # 设置初始化超参数
    # 学习率
    INIT_LR = 0.01
    # Epoch# 这里设置 5 是为了能尽快训练完毕，可以设置高一点，比如 30
    EPOCHS = 5
    # Batch Size
    BS = 32
    # 损失函数，编译模型
    print("------开始训练网络------")
    opt = SGD(lr=INIT_LR, decay=INIT_LR / EPOCHS)
    model.compile(loss="categorical_crossentropy", optimizer=opt,metrics=["accuracy"])
    # 训练网络模型
    H = model.fit_generator(
        aug.flow(trainX, trainY, batch_size=BS),
        validation_data=(testX, testY),
        steps_per_epoch=len(trainX) // BS,
        epochs=EPOCHS
        )
        # 测试
        print("------测试网络------")
        predictions = model.predict(testX, batch_size=32)
        print(classification_report(testY.argmax(axis=1), predictions.argmax(axis=1),
target_names=lb.classes_))
        # 绘制结果曲线
        N = np.arange(0, EPOCHS)
        plt.style.use("ggplot")plt.figure()
```

```
plt.plot(N, H.history["loss"], label="train_loss")
plt.plot(N, H.history["val_loss"], label="val_loss")
plt.plot(N, H.history["accuracy"], label="train_acc")
plt.plot(N, H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend()plt.savefig('./output/cnn_plot.png')
# 保存模型
print("------保存模型------")
model.save('./cnn.model.h5')
f = open('./cnn_lb.pickle', "wb")
f.write(pickle.dumps(lb))
f.close()
```

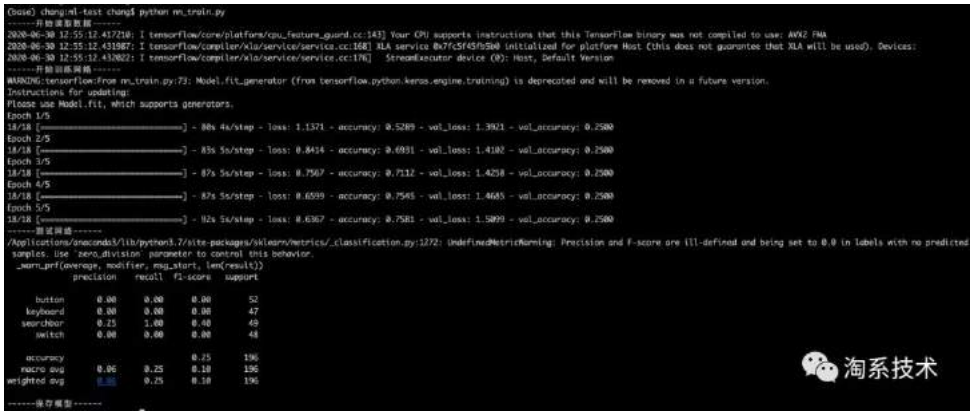
对于实际应用场景下，数据集很大，epoch 也会设置比较大，并在高性能的机器上训练。现在要在本机 Mac 上完成训练任务，我们只给了很少的样本来训练模型，epoch 也很小（为 5），当然这样模型的识别准确率也会很差，但我们此篇文章的目的是为了在本机完成一个机器学习的任务。

开始训练

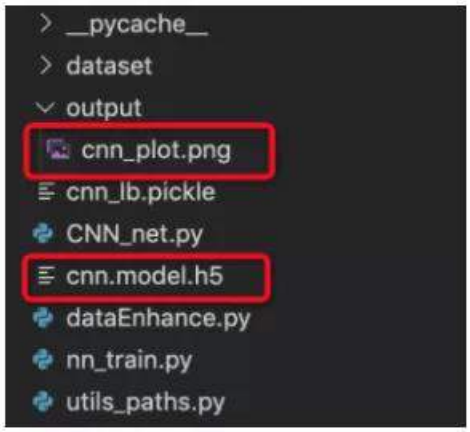
执行以下命令开始训练：

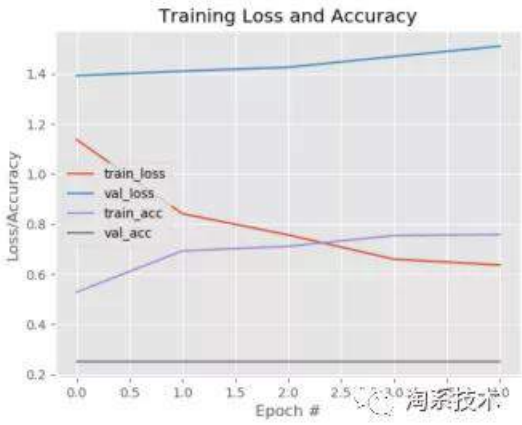
```
$ python nn_train.py
```

训练过程日志如下：



训练结束后，在当前目录下会生成两个文件：模型文件 cnn.model.h5 和 损失函数曲线 output/cnn_plot.png





模型评估

现在，我们拿到了模型文件 cnn.model.h5， 可以写一个预测脚本，本地执行脚本预测一张图片的分类。

```
$ python predict.py
```

```
# predict.py
import allspark
import io
import numpy as np
import json
from PIL
import Image
import requests
import threading
import cv2
import os
import tensorflow as tf
from tensorflow.keras.models import load_model
import time
model = load_model('./train/cnn.model.h5')
# pred 的输入应该是一个 images 的数组，而且图片都已经转为 numpy 数组的形式
```

```
# pred = model.predict(['./validation/button/button-demoplus-20200216-16615.png'])
#这个顺序一定要与 label.json 顺序相同，模型输出是一个数组，取最大值索引为预测值
Label = [
  "button",
  "keyboard",
  "searchbar",
  "switch"
  ]
  testPath = "./test/button.png"
  images = [ ]
  image = cv2.imread(testPath)
  image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
  image = cv2.resize(image,(256,256))
  images.append(image)
  images = np.asarray(images)
  pred = model.predict(images)
  print(pred)max_ = np.arg
  max(pred)
  print('预测结果为：',Label[max_])
```

如果想要知道这个模型的准确率，也可以给模型输入一批带有已知分类的数据，通过模型预测后，将模型预测的分类与真实的分类比较，计算出准确率和召回率。

模型服务部署

开发模型服务

但在实际应用中，我们预测一张图片的类别， 是通过给定一张图片，请求一个 API 来拿到返回结果的。我们需要编写一个模型服务，然后部署到远端，拿到一个部署之后的模型服务 API。

现在，我们可以编写一个模型服务，然后在本地部署。

```
# 模型服务 app.py
import allspark
import io
import numpy as np
```

```
import json
from PIL
import Image
import requests
import threading
import cv2
import tensorflow as tf
from tensorflow.keras.models import load_model
with open('label.json') as f:
    mp = json.load(f)
labels = {value:key for key,value in mp.items()}
def create_opencv_image_from_stringio(img_stream, cv2_img_flag=-1):
img_stream.seek(0)
img_array = np.asarray(bytearray(img_stream.read()), dtype=np.uint8)
image_temp = cv2.imdecode(img_array, cv2_img_flag)
if image_temp.shape[2] == 4:
    image_channel3 = cv2.cvtColor(image_temp, cv2.COLOR_BGRA2BGR)
    image_mask = image_temp[:,:,3] #.reshape(image_temp.shape[0],image_temp.shape[1], 1)
    image_mask = np.stack((image_mask,
    image_mask, image_mask), axis = 2)
    index_mask = np.where(image_mask == 0)
    image_channel3[index_mask[0], index_mask[1], index_mask[2]] = 255
    return image_channel3
else:
    return image_temp
def get_string_io(origin_path):
    r = requests.get(origin_path, timeout=2)
    stringIo_content = io.BytesIO(r.content)
    return stringIo_content
def handleReturn(pred, percent, msg_length):
    result = {
        "content":[]
}
argm = np.argsort(-pred, axis = 1)
for i in range(msg_length):
    label = labels[argm[i, 0]]
    index = argm[i, 0]
    if(pred[i, index] > percent):
        confident = True
    else:
        confident = False
```

```
    result['content'].append({'isConfident': confident, 'label': label})
    return result
def process(msg, model):
    msg_dict = json.loads(msg)
    percent = msg_dict['threshold']
    msg_dict = msg_dict['images']
    msg_length = len(msg_dict)
    desire_size = 256
images = [ ]
for i in range(msg_length):
  image_temp = create_opencv_image_from_stringio(get_string_io(msg_dict[i]))
    image_temp = cv2.cvtColor(image_temp, cv2.COLOR_BGR2RGB)
    image = cv2.resize(image_temp, (256, 256))
    images.append(image)
    images = np.asarray(images)
    pred = model.predict(images)
    return bytes(json.dumps(handleReturn(pred, percent, msg_length)) ,'utf-8')
def worker(srv, thread_id, model):
while True:
    msg = srv.read()
    try:
        rsp = process(msg, model)
        srv.write(rsp)
    except Exception as e:
        srv.error(500,bytes('invalid data format', 'utf-8'))
if __name__ == '__main__':
    desire_size = 256
    model = load_model('./cnn.model.h5')

    context = allspark.Context(4)
    queued = context.queued_service()
    workers = [ ]       for i in range(10):
        t = threading.Thread(target=worker, args=(queued, i, model))
        t.setDaemon(True)
        t.start()
        workers.append(t)
for t in workers:
        t.join()
```

部署模型服务

模型服务编写完成后，在本地部署，需要安装环境。首先创建一个模型服务项目：deploy-project，将 cnn.model.h5 拷贝到此项目中， 并在此项目下安装环境。

```
├── app.py
├── cnn.model.h5
└── label.json
```

安装环境

可以看下阿里云的模型服务部署文档：3、Python 语言-3.2 构建开发环境-3.2.3 使用预构建的开发镜像(推荐)

安装 Docker

可以直接查看 Mac Docker 安装文档

```
# 用 Homebrew 安装 需要先现状 Homebrew: https://brew.sh
$ brew cask install docker
```

安装完之后，桌面上会出现 Docker 的图标。

创建 anaconda 的虚拟环境

```
# 使用 conda 创建 python 环境，目录需指定固定名字：ENV
$ conda create -p ENV python=3.7
# 安装 EAS python sdk
$ ENV/bin/pip install
http://eas-data.oss-cn-shanghai.aliyuncs.com/sdk/allspark-0.9-py2.py3-none-any.whl
# 安装其它依赖包
$ ENV/bin/pip install tensorflow keras opencv-python
# 激活虚拟环境
$ conda activate ./ENV
# 退出虚拟环境（不使用时）
$ conda deactivate
```

运行 Docker 环境

/Users/chang/Desktop/ml-test/deploy-project 换成自己的项目路径:

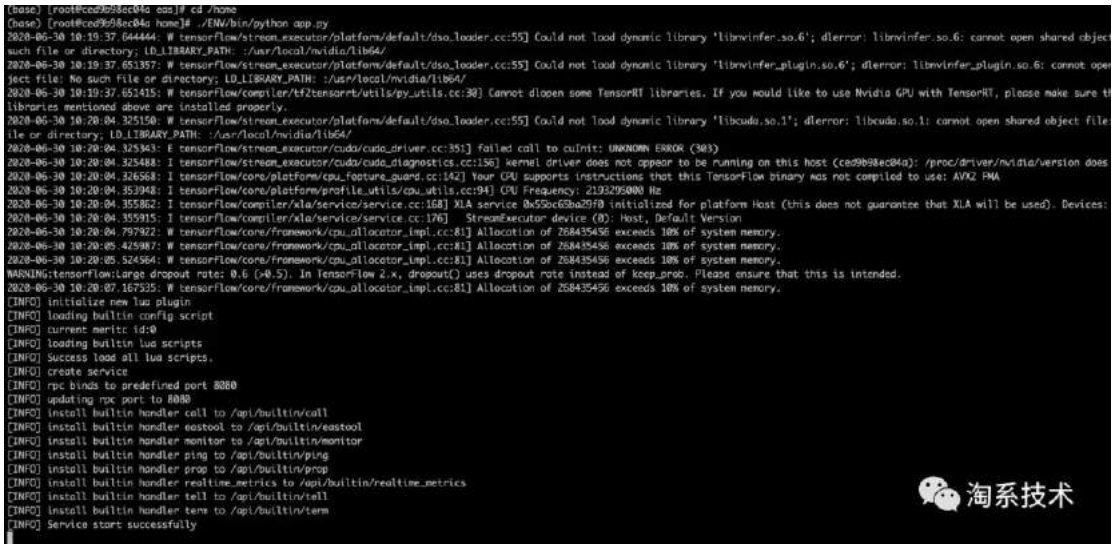
```
sudo docker run -ti -v  /Users/chang/Desktop/ml-test/deploy-project:/home -p 8080:8080
registry.cn-shanghai.aliyuncs.com/eas/eas-python-base-image:py3.6-allspark-0.8
```

本地部署

现在可以本地部署了，执行以下命令:

```
cd /home
./ENV/bin/python app.py
```

下面的日志可以看到部署成功。



部署成功后，可以通过 localhost:8080/predict 访问模型服务了。

我们用 curl 命令来发一个 post 请求， 预测图片分类:

```
curl -X POST 'localhost:8080/predict' \
-H 'Content-Type: application/json' \
-d '{
"images":["https://img.alicdn.com/tfs/TB1W8K2MeH2gK0jSZJnXXaT1FXa-638-430.png"],
"threshold": 0.5
}'
```

得到预测结果：

```
{"content": [{"isConfident": true, "label": "keyboard"}]}
```

完整代码

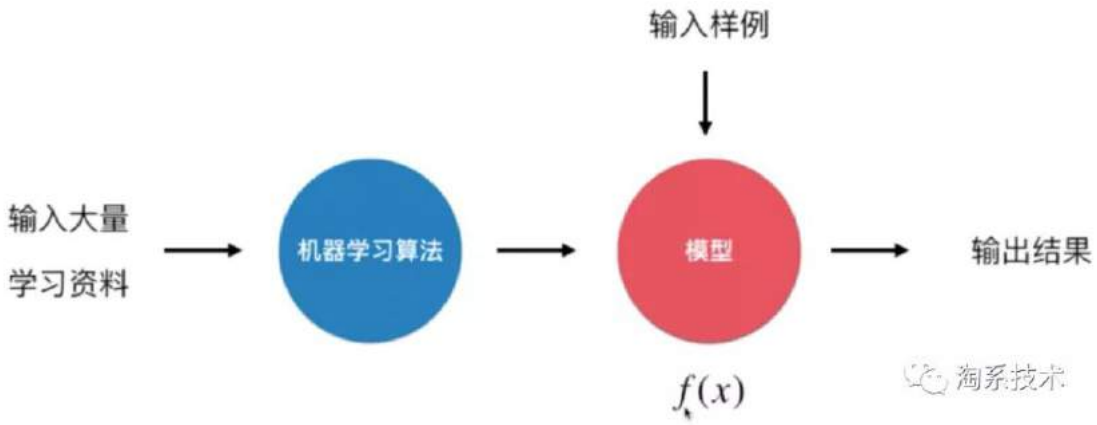
可以直接 clone 代码仓库：https://github.com/imgcook/ml-mac-classify

在安装好环境后，直接按以下命令运行。

```
# 1、训练模型
$ cd train-project
$ python nn_train.py
# 生成模型文件：cnn.model.h5
# 2、将模型文件拷贝到 deploy-project 中，部署模型服务
# 先安装模型服务运行环境
$ conda activate ./ENV
$ sudo docker run -ti -v  /Users/chang/Desktop/ml-test/deploy-project:/home -p 8080:8080
registry.cn-shanghai.aliyuncs.com/eas/eas-python-base-image:py3.6-allspark-0.8
$ cd /home
$ ./ENV/bin/python app.py
# 得到模型服务 API：localhost:8080/predict
# 3、访问模型服务
curl -X POST 'localhost:8080/predict' \
-H 'Content-Type: application/json' \
-d '{
"images":["https://img.alicdn.com/tfs/TB1W8K2MeH2gK0jSZJnXXaT1FXa-638-430.png"],
"threshold": 0.5}'
```

最后

好啦，总结一下这里使用深度学习的流程。我们选用了 SimpleVGGNet 作为图像分类算法（相当于一个函数），将准备好的数据传给这个函数，运行这个函数（学习数据集的特征和标签）得到一个输出，就是模型文件 model.h5。



这个模型文件可以接收一张图片作为输入，并预测这张图片是什么，输出预测结果。但如果想要让模型可以在线上跑，需要写一个模型服务(API)并部署到线上以得到一个 HTTPAPI，我们可以在生产环境直接调用。

闲鱼 Weex 页面优化过程详解

作者 | 颂晨
出品 | 淘系技术

闲鱼前端页面的性能常常被人念叨，凡跳转、必跳鱼的印象深入人心，部分页面甚至需要跳四五下才能打开，最近我们对闲鱼前端页面系统性的做了些优化，由于闲鱼前端技术栈相对多元，不同栈技术原理各不相同，优化方案也有所差异，本文主要介绍目前闲鱼占比较重的 Weex 页面的优化过程。

背景

闲鱼 Weex 页面多以前端渲染为主，其打开过程与 Web 页面略微相近，大致分为以下几个阶段：

我们将「从开始加载（navigationStart）到屏幕首次 paint（绘制）像素内容」的这段时间称为白屏时间（FP），将「从开始加载（navigationStart）到首屏内容全部渲染完成」的这段时间称为 首屏时间（FSP），受限于统计口径，目前 Weex 下的首屏时间是不包含图片下载及后续过程的。

优化前后

我们拿闲鱼的直播频道页和玩家频道页作为参考，通过录屏的方式看下优化前后的对比：

点击查看动图演示　　点击查看动图演示

通过录屏分帧的方式我们统计了下这两个频道页在不同系统不同机型下的首屏时长：

	直播频道页（优化前）	直播频道页（优化后）	玩家频道页（优化前）	玩家频道页（优化后）
Phone 11 / iOS 高端机	2500ms	550ms ↓ 78%	2000ms	500ms ↓ 75%
iPhone 6 / iOS 低端机	2650ms	850ms ↓ 68%	3900ms	1600ms ↓ 59%
华为荣耀 V20 / Android 主流机	3650ms	600ms ↓ 83%	2600ms	1100ms ↓ 58%
VIVO Y67 / Android 低端机	7400ms	1550ms ↓ 79%	4800ms	1900ms ↓ 60%

可以看到，优化前 iOS、Android 主流机型上的首屏时间都要超过 2s，低端机甚至要 3-5s，优化后各机型的首屏时间均大幅下降，低端机首屏时长控制到了 2s 内，中高端机近乎直开。

拆解分析

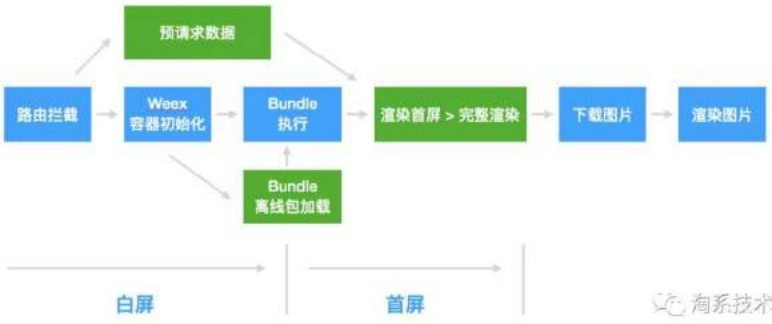
确定优化方案前我们对现有的 Weex 页面做了拆解分析，从结果来看，以下几个因素对首屏时间的影响较大：

- Bundle 体积：不仅影响 Bundle 加载时长，同时也影响 Bundle 的解析执行耗时（低端机尤为明显）。
- 首屏数据请求：页面渲染必须在首屏数据请求返回后，接口耗时直接影响首屏时间。
- 首屏渲染范围：首屏渲染量直接影响渲染时长（低端机尤为明显）。

优化方案

基于上面的分析调研，我们初步把优化方案定为四层：

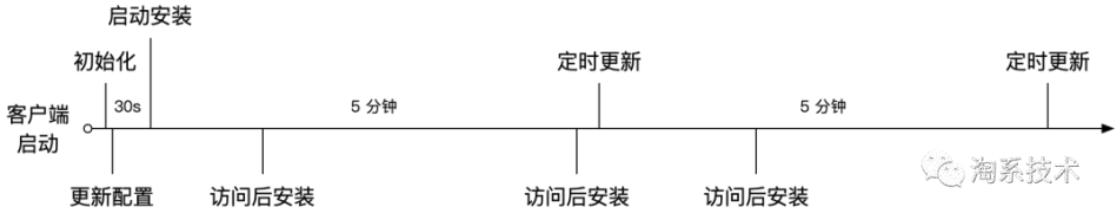
按照预期优化效果，Weex 页面的打开过程是这样的：



体现在上述的四层结构中，主要包含以下几个优化点：

Bundle 离线

具体实现是将 Weex Bundle 以资源包为单位、以 URL 前缀为索引，通过一定的更新策略离线到客户端本地，之前的更新策略主要有访问后安装、启动安装 两种，对应的更新时机如下：



这套机制在容器层有统一的方案支撑，但是包命中率一直不高（25% - 55%），导致最终效果差强人意，分析后发现默认的更新策略（访问后安装）与页面回访率强相关，闲鱼的前端页面大都是频道导购型的页面，回访率天然不高，所以包命中率相对应也不会高。

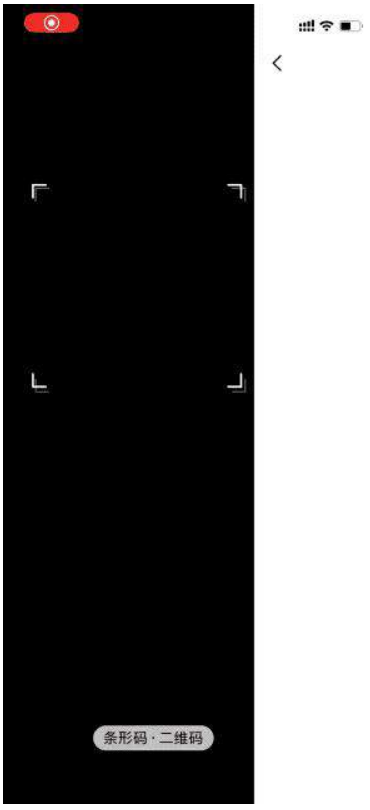
本次优化主要是对更新策略进行了扩展，增加了 “闲时安装” 的更新策略：会在定时更新期间主动安装，如果安装后未使用，则会在一周之后淘汰；如果一周内使用过，则进入常规的更新淘汰机制（一个月未使用淘汰）。

在“闲时安装” 的更新策略上线后，包命中率大幅提升（稳定后 90% 左右），页面性能也得到了显著提升：

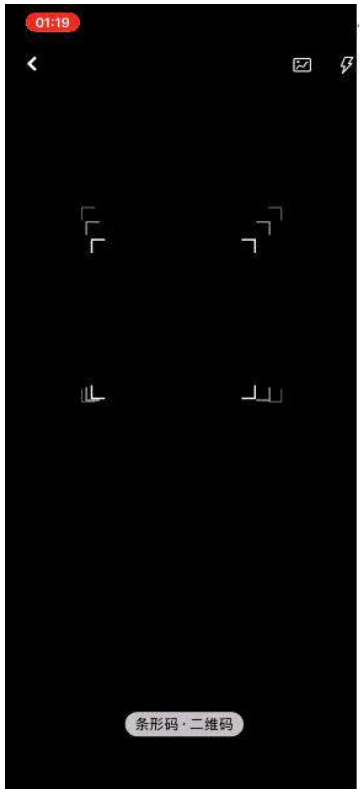
	优化前	优化后
直播频道页	白屏时间 Android：456.85 ms iOS：410.91 ms 首屏时间 Android：2489.35 ms iOS：1797.02 ms ZCache Android：9.78 % iOS：10.77 %	白屏时间 Android：188.16 ms（↓58.8%） iOS：132.71 ms（↓67.7%） 首屏时间 Android：2152.15 ms（↓13.5%） iOS：1497.53 ms（↓16.6%） ZCache Android：80.74 % iOS：79.18 %
玩家频道页	白屏时间 Android：390.7 ms iOS：342.25 ms 首屏时间 Android：2144.14 ms iOS：1275.97 ms ZCache Android：33.26 % iOS：35.32 %	白屏时间 Android：184.44 ms（↓52.8%） iOS：147.25 ms（↓57%） 首屏时间 Android：1881.62 ms（↓12.2%） iOS：1057.31 ms（↓17.1%） ZCache Android：78.83 % iOS：77.52 %

淘系技术

不依赖首屏接口渲染的页面甚至可以达到「直开」：



点击查看动图演示



点击查看动图演示

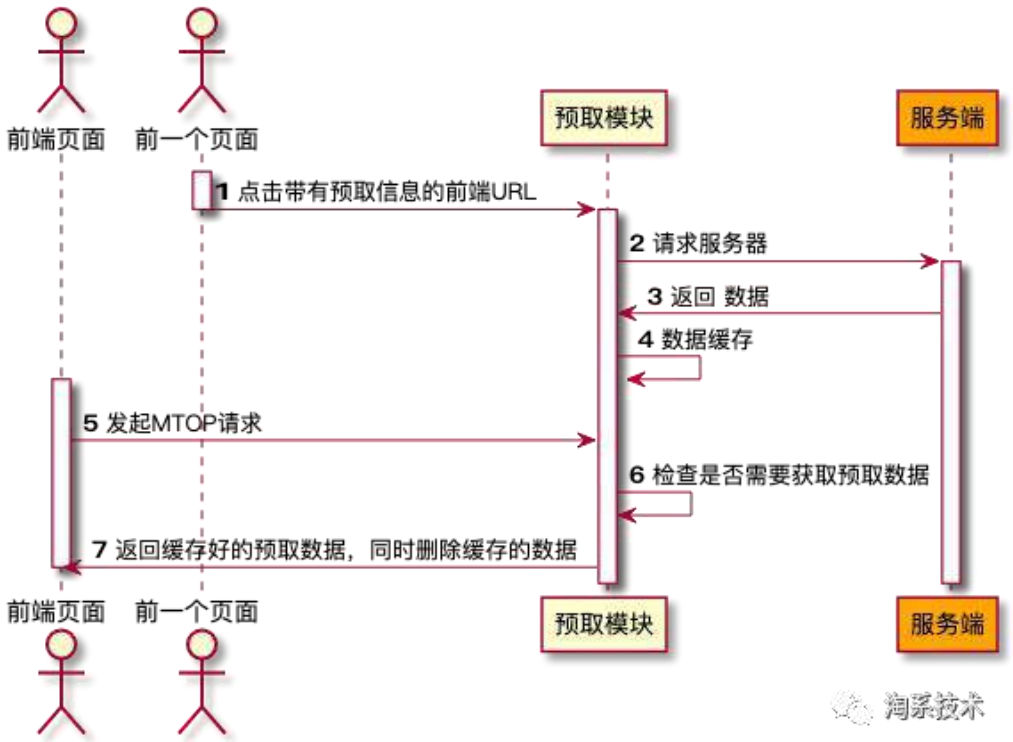
数据预取

传统的首屏数据请求都是在 Bundle 解析完以后发起的，首屏数据返回后渲染页面，是个典型的串行过程。

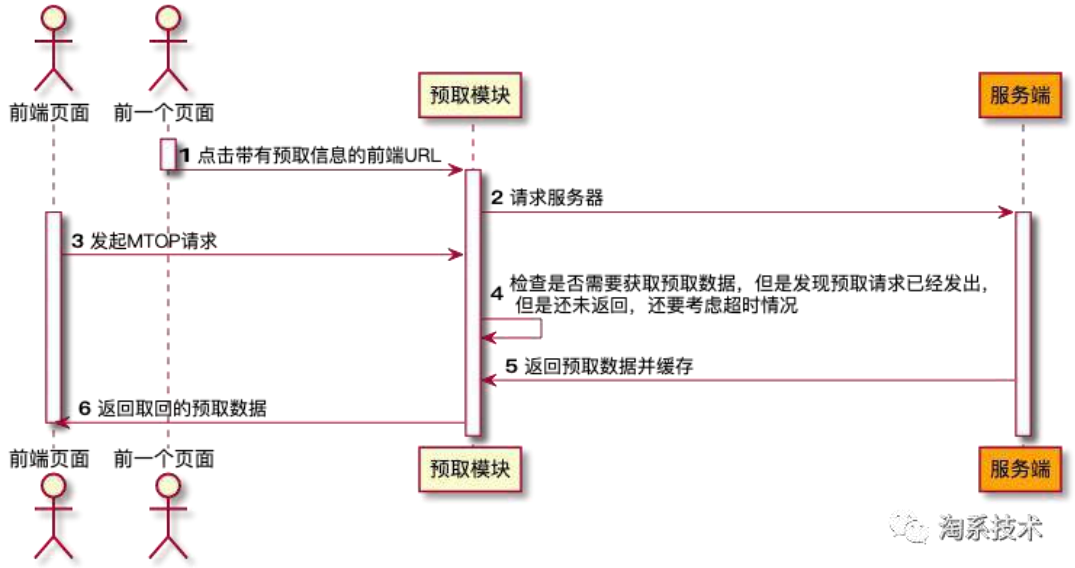
本次优化中我们把这个串行的过程并行化了：

1. 将首屏请求的配置序列化以后作为参数配置到了 URL 上，同时支持一些动态替换的参数（譬如经纬度、城市等参数）；
2. 在 navigationStart 的时候由客户端提取首屏请求配置，然后发起请求，并将结果以特定的 Hash Key（通过首屏配置生成的）作为索引存储到本地；
3. 在业务层真正发起首屏请求的时候会通过 Hash Key 进行比对，命中后将数据取出来返回给业务层；

时序图如下：



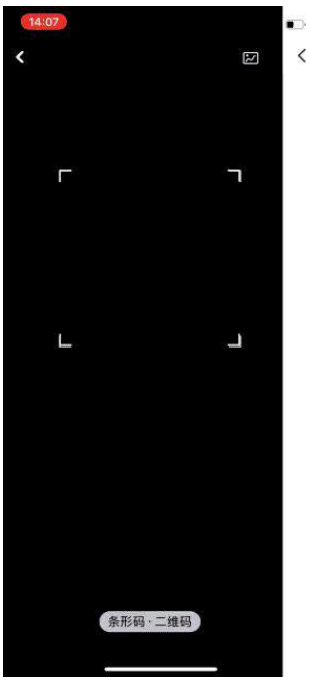
特殊情况下的时序图：

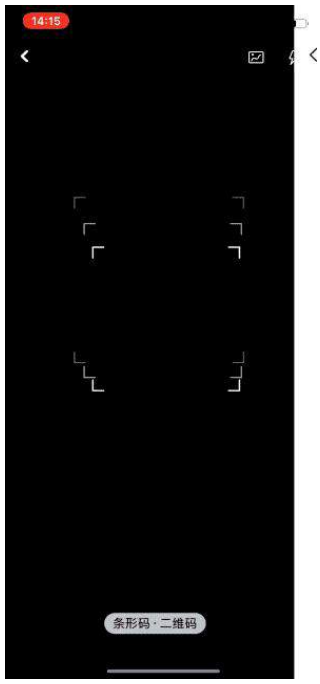


具体的技术细节本文不再赘述，数据预取的优化策略上线后，首屏时间也得到了一定程度的提升，如下（iOS 侧由于各优化策略并行上线，没能做到单一变量采集性能数据，暂以 Android 侧为参考）：

	优化前	优化后
直播频道页	白屏时间 Android：132.89 ms 首屏时间 Android：2186.74 ms ZCache Android：93.57 %	白屏时间 Android：128.4 ms 首屏时间 Android：1921.09 ms（↓12.1%） ZCache Android：89.47 %
玩家频道页	白屏时间 Android：131.2 ms 首屏时间 Android：2052.21 ms ZCache Android：93.66 %	白屏时间 Android：131.11 ms 首屏时间 Android：1685.38 ms（↓17.9%） ZCache Android：88.27 %

Bundle 离线、数据预取的优化策略上线后，部分页面在中高端机型上逼近「直开」：



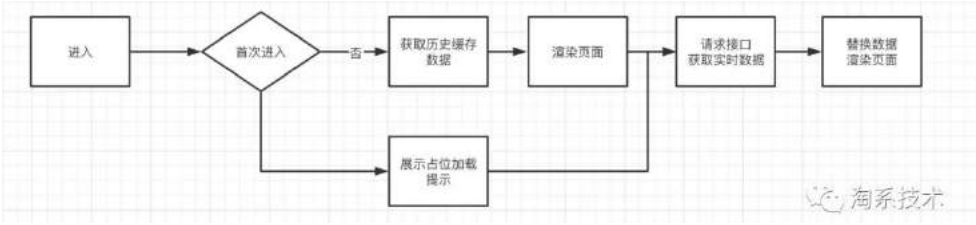


点击查看动图演示　　点击查看动图演示

渐进式首屏

渐进式首屏解决的是「最后一公里」的问题，因为在上了「离线包」和「数据预取」的方案后，我们发现：页面首屏时间一定程度上还是受限于首屏接口请求耗时，该方案就是为了降低用户侧的白屏等待时长，具体从以下三个方面着手：

- 以接口请求配置生成的索引对接口数据进行缓存；
- 当用户首次进入时，以骨架屏占位来等待业务数据加载；
- 当用户非首次进入时，会根据接口请求配置生成的索引在本地缓存中查找缓存数据，并完成首屏渲染，同时并行发送接口请求，待新数据返回后，触发页面更新，完成最终渲染；



低端机降级方案

为了用户体验能够更好，在此我们尝试了低端机降级优化方案。以直播频道为例：

- 只对首屏 Tab 做缓存数据占位优化
- 减少了低端机上首屏渲染展示数据量

图片渲染效果优化

渐进式首屏带来的一个问题是界面更新时的闪动（特别是图片占大篇幅的时候），为了优化此问题，我们将图片从加载到出现的过程改为了渐显过渡，一定程度上消除了图片闪动的生硬感。

点击查看动图演示

点击查看动图演示

优化前	优化后
iOS: 开始加载到首屏数据渲染间隔约 700ms Android: 开始加载到首屏数据渲染间隔约 1000ms	iOS: 开始加载到首屏数据渲染间隔约 60ms（↓91.4%） Android: 开始加载到首屏数据渲染间隔约 200ms（↓80%）

淘系技术

按需渲染

渲染页面作为首屏链路中的一环，不同技术栈、不同设备环境下，在页面首屏时间中也会有不同的占比。类 Weex、RN 通过前端脚本映射原生组件的技术方案，渲染路径总结起来是：渲染前端 Virtual DOM -> 映射为 Native 指令 -> 将指令传输到 Native 侧 -> Native 执行指令完成渲染。在前三个步骤上，较重的业务逻辑或不合理的代码通常会带来较长的计算、通信耗时，中低端机器上尤为明显。通过按需渲染可以有效解决这一问题。

按需渲染主要思路是通过只渲染首屏可见视图来最小化首屏渲染耗时。本次优化中，主要针对以下几个场景做了按需渲染：

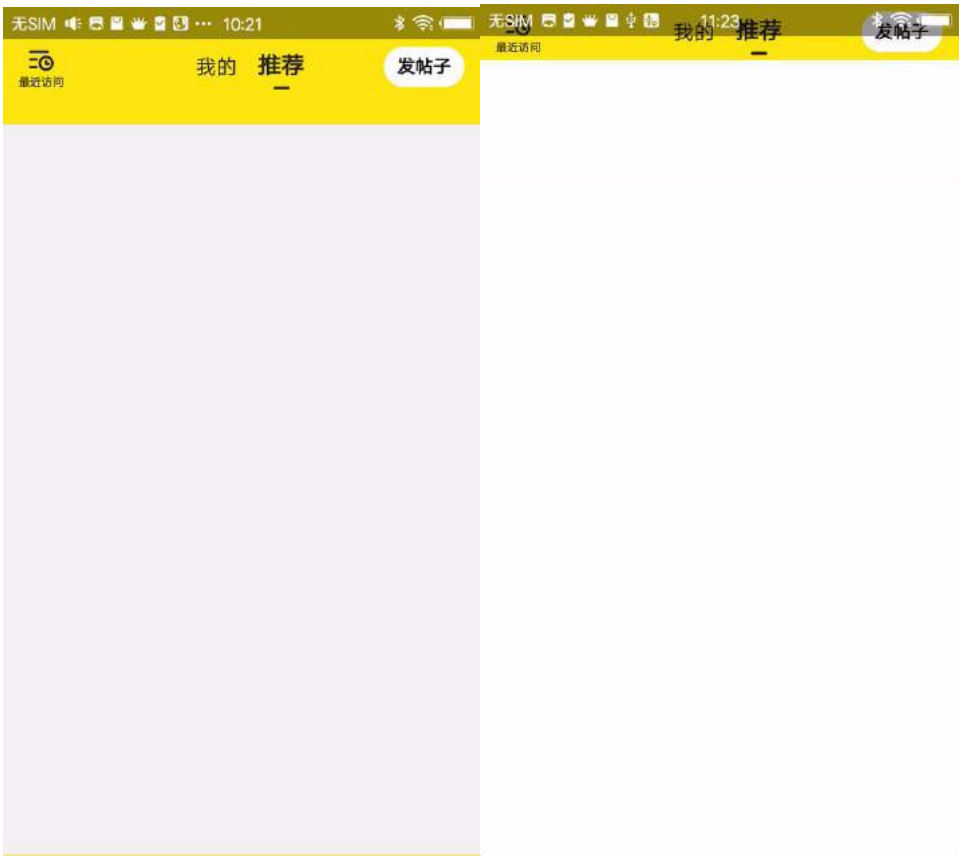
- 多 Tab 情况下，对于有性能要求的非首屏 tab 页，做数据预加载、页面懒渲染处理。
- 对带/不带回收机制的长列表做首屏只渲染可见条目，剩余懒渲染处理。可减少带回收机制列表的脚本计算、通信耗时，减少不带回收机制列表的全链路渲染耗时。
- 自建或使用轻量级组件替换非必要的重量级组件，如：xSlider。

优化上线后，鱼塘广场页中低端机型的首屏性能有了部分数据上的提升：

	优化前	优化后
鱼塘广场页首屏时间	Android 中端机型：3138 ms 低端机型：4526 ms iOS 中端机型：2269 ms 低端机型：4166 ms	Android 中端机型：2589 ms（↓18%） 低端机型：3678 ms（↓19%） iOS 中端机型：1788 ms（↓22%） 低端机型：2911ms（↓3[illegible]）

淘系技术

低端机上优化前端渲染阶段对比：



点击查看动图演示　　点击查看动图演示

Bundle 瘦身

Bundle 体积一方面直接影响 Bundle 下载时间，另一方面也会影响 Android 端的渲染性能（耗时随 Bundle 体积增加 1-2ms/KB），我们在 Bundle 体积上的优化方案较为常规，包括：

1. 通过 Webpack Bundle Analyzer 分析依赖，减少同 npm 包不同版本依赖。
2. 抽象公共模块，提高代码复用率。
3. 重构基础工具类库，支持按需加载打包。

优化前	优化后
• 一个库引入了多个版本； • 依赖了较多三方库，部分库体积较大，但实际使用到的功能较简单； • 基础工具类存在重复实现的冗余代码，且不支持按需引入；	• 移除了冗余库； • 重新梳理了常用的三方工具库，将体积过大的简化实现； • 升级了组件，重构了工具库，同时支持按需引入； Bundle 体积整体减小了 40KB 左右，为 Android 首屏时间整体带来了 40-80ms 的提升

总结

闲鱼前端的性能优化暂时告一段落，优化过程中沉淀了较多的通用能力，像 Bundle 离线、数据预取、渐进式首屏等等，这些能力在后续会有更大的发挥空间，一些能力也会变得更加智能，譬如目前的数据预取是在 navigationStart 的时候发起的，这个时机已经比传统的页面加载时的时机提前了许多，但其实还可以更加提前，譬如可以在闲鱼客户端中常驻个 TaskSchedule，专门用来处理数据预取的 Task，同时可以结合用户的访问习惯做智能数据预取。

在前端性能要求越来越高的背景下，传统的 Web 加载流程已无法再满足性能优化的需要，所以出现了各种新兴容器 + 配套能力，所以下一代容器的标准形态应该是什么样的？

智能 UI：面向未来的 UI 开发技术

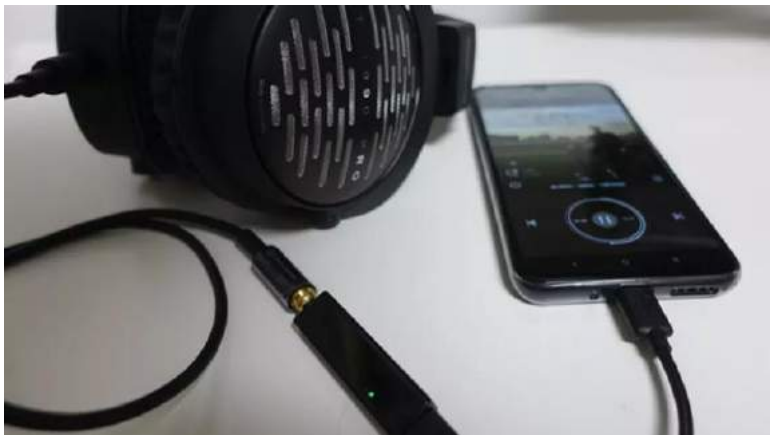
作者 | 甄子
出品 | 淘系技术

随着机器学习和人工智能渗入到各行各业，随着你画我猜、智能推荐、以图搜图、尬舞……渗入生活的方方面面，前端作为编程领域的一支，也必将迎来更多变化和挑战。

前言

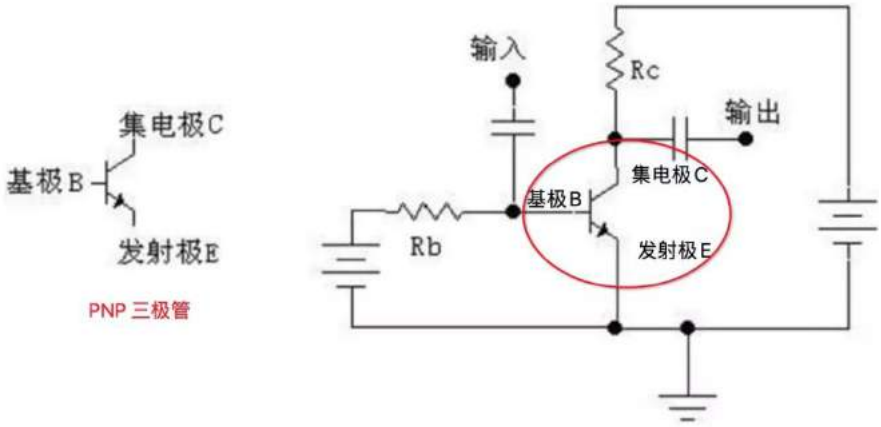
近几年来，UI 和交互最重要的变化在于如何满足不同人群的需求，下面笔者以简单电路图为例，谈谈如何把 UI 和交互从“功能”中释放出来，放大至更广阔的领域中，再来看看淘系在前端智能化领域做了哪些实践？

举例来说，虽然我是木耳，但是，偶尔还是会拿有线耳机听一下无损音乐。同事跟我说：你那个耳机手机带不动，要配个耳机放大器。耳机放大器（简称“耳放”）的原理，笼统概括就是把微弱信号变成强劲信号，从而能带动（专业术语是“推”）xx 欧姆的耳机。如果把整个耳机看做一个功率耗散元件——电阻，这里 xx 欧姆就是电阻的阻值。满足耳机功率要求，耳机才能够播放出强劲的低音、丰富的细节、广阔的音场……带给我们身临其境的震撼。

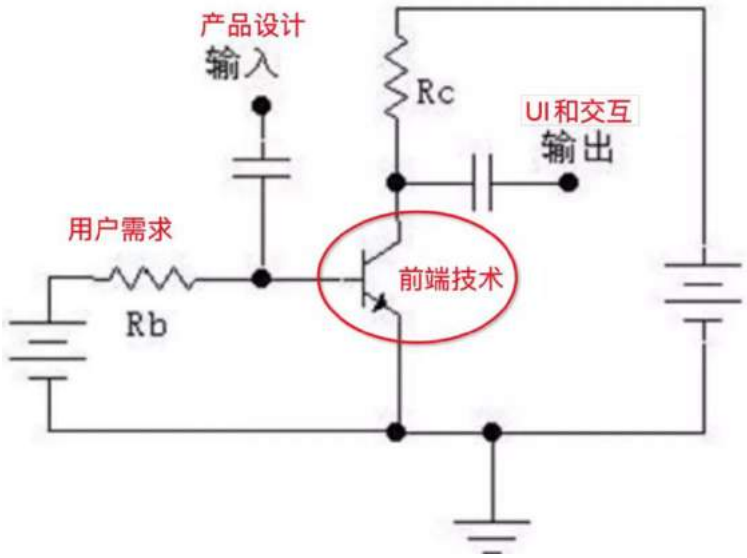


手机上的耳放

要带动上图这个大家伙，手机上插的就是耳放。耳放很复杂，不仅需要提升电流电压，还有音色、降噪、还原……等 DSP（数字信号处理）能力。不过，我们可以像把耳机近似成电阻一样（耳机的功率 xx 欧姆），把整个耳放近似成一个三极管。



上图左边是三极管的元件符号，右边是功放（耳机功放的"功放"）电路图。如果我们事先在三极管的基极上加上一个合适的电流（叫做偏置电流，上图中那个电阻 Rb 就是用来提供这个电流的，所以它被叫做基极偏置电阻），那么当一个小信号跟这个偏置电流叠加在一起时，小信号就会导致基极电流的变化，而基极电流的变化，就会被放大并在集电极上输出。如果这里的偏执电流 Rb 是手机输出的音频信号，在集电极上输出的就是放大后的信号了。



如果换个角度看这个放大电路，我们可以把偏执电流类比为用户需求，把输入电源类比为产品设计（包含视觉和交互设计及技术方案等），把前端技术类比为三极管，把集电极输出类比为 UI 和交互（最终的交付物）。

首先，不论用户需求是用户提出的、产品 YY 的还是老板拍的，在最初提出的那一刻都模糊而粗糙。根据这些原始粗糙的想法，通过产品设计和前端开发，最终才能在用户面前呈现强劲的低音、丰富的细节、广阔的音场……带给用户身临其境的震撼。

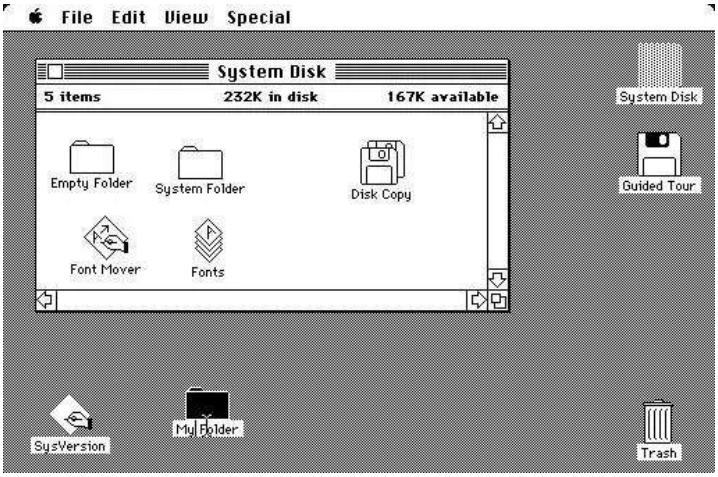
其次，三极管——前端开发是放大电路的心脏，三极管从截止区（未进入工作状态）末端到饱和区（超负荷）开端的范围：工作区，直接决定了三极管的放大倍数。就像用不同的前端技术方案去实现需求，技术方案的先进性和合理性，直接决定了最终交付的 UI 和交互质量，进而影响用户体验。

最后，具体如何理解放大需求呢？放大需求是相对于满足需求而言的。中国有句古话叫：饱暖思淫欲。“饱暖”是人的基本生理需求，相当于满足需求。“思淫欲”是人的高级需求，有人可能会问：这不是生理需求么？那叫“繁殖”。

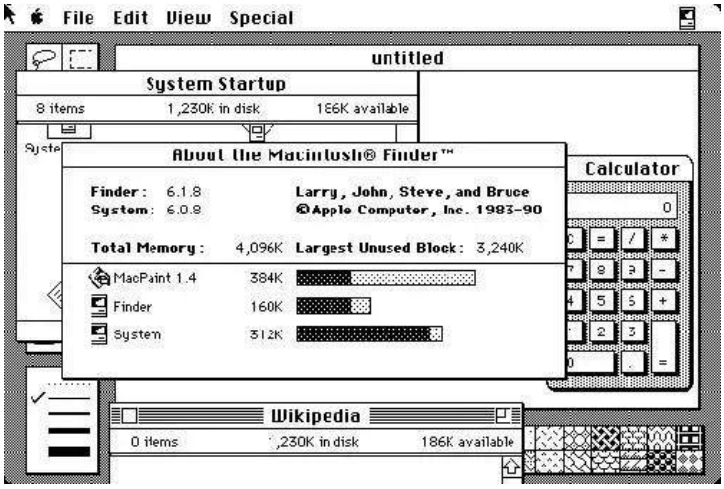
举个例子，在相机出现之前，绘画的目的主要是记录，记录人物、事件……等等。相机的出现，让记录这件事变得简单高效，绘画得以从繁重的记录工作中释放出来，从而追求艺术性。这就相当于，照相技术本质上放大了绘画的价值——艺术性。

今天，耳放技术可以推 xx 欧姆的高级耳机，把手机从音频播放的听个响、打电话等基本需求中释放出来，放大至“唯美的音乐”、“震撼的视频临场感”……那么，前端技术应该如何把 UI 和交互从“功能”中释放出来，放大至更广阔的领域呢？接下来谈谈我的一些愚见。

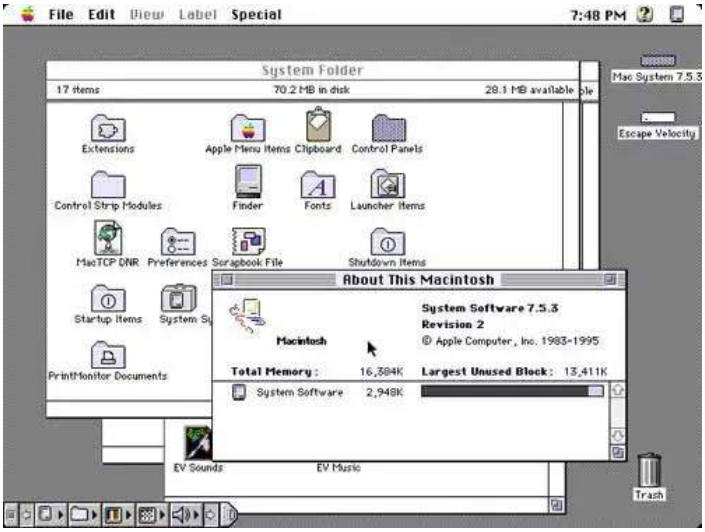
基础 UI 和交互



Macintosh 桌面



System 6, 1988



System 7.5.3, 1991

MacOS 8.1, 1997

MacOS 9, 1999

Mac OS 一直深受其用户的赞誉，它的很多设计都成为了其它软件和系统所模仿和借鉴的对象，对全世界人机界面及其相关行业的影响极其深远。在诸多版本的 Mac OS 中，MacOS 9 及以前的 Mac 操作系统通常被称为古典系统（Classic OS），从 2000 年开始，Mac OS 进入了全新的 Mac OS X 时代。

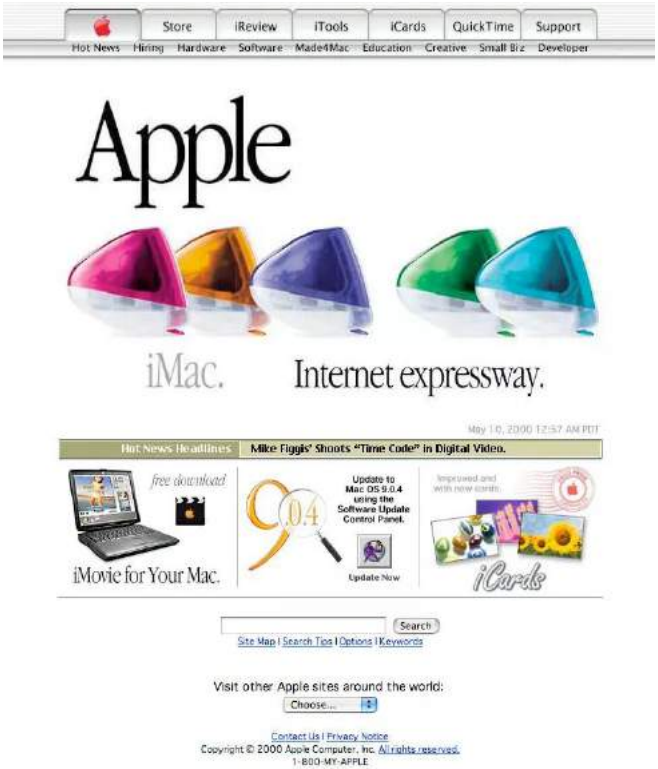
我认为，在进入 Mac OS X 之前，Apple 着重在打造一套完善的基础 UI 和交互系统。这就像照相技术出现前的绘画一样，先去满足人们的基本需求：记录。

搞 UI 的都应该听说过 Fitts' Law - 鼠标光标指向某一物体的时间，受物体大小和光标距离其远近这两个因素所决定。位于顶部的菜单栏其实是对 Fitts' Law 的灵活运用：由于光标不会跑到屏幕外面去，因此你尽可以随便把鼠标往上甩，光标肯定位于菜单栏上，这降低了光标指向的难度。

然而时过境迁，在大屏幕、多显示器越来越普及的今天，这个设计逐渐受到了挑战：在某些极端的情况下（比如主窗口位于最右面的显示器上），用户不得不移动鼠标横跨数台显示器、几千个像素去操作菜单。

被技术放大的 UI 和交互

Apple Quartz Extreme 放大器

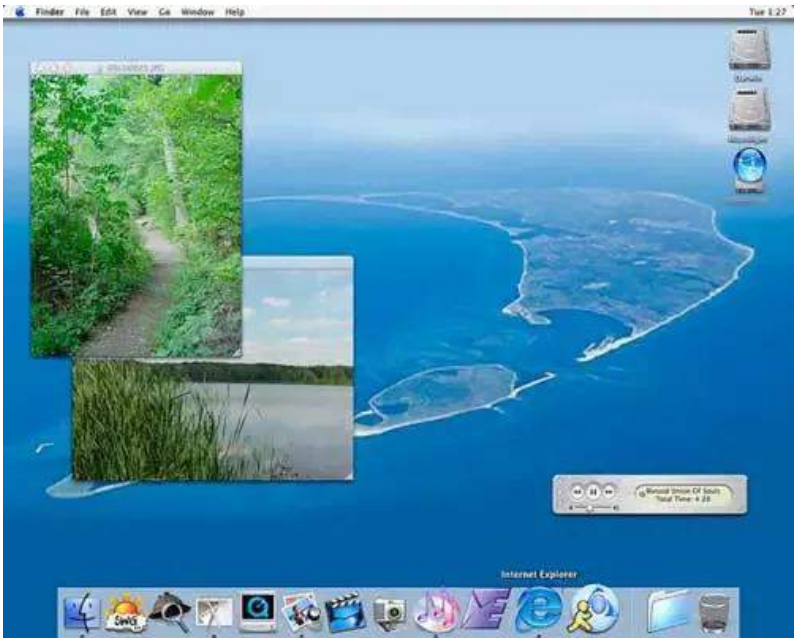


2000 年 5 月 10 日这天的 apple.com

在 Jobs 第一次公开介绍 Mac OS X 的用户界面时，他宣称不仅要做出一个高可用性的用户界面，更重要的是，Mac OS X 将具备一个名为 AQUA 的"Dream User Interface"。AQUA 一方面保留有古典系统高度易用的特点；另一方面，通过各种视觉特效的引入，AQUA 将带来前所未有的用户体验。

Mac OS X Public Beta, 2000

Mac OS X 10.0 Cheetah, 2001



Mac OS X 10.1 Puma, 2001

Mac OS X 10.2 Jaguar, 2002

从上图可见，从 Jaguar 开始拉丝金属已经得到广泛的应用。由于 Quartz Extreme 这项被简称为 QE 的新技术给 UI 设计带来了新的可能性。简单地说，它就是利用了显卡的强劲性能，为 UI 提供硬件加速。这样进一步解除了机器性能对 UI 设计的限制，设计师可以充分发挥自己的创造力。由此可见，想要产出或保持一个高水平的 UI 设计，先进的底层技术是重要基础。

Apple Core Animation 放大器

Core Animation 是一个面向程序员的动画开发框架，它旨在统一、规范并简化程序员在 UI 设计上的编码工作，并通过提供一系列预定义好的 API，让开发者可以轻松地实现各种视觉特效。

Core Animation 把 UI 设计向前推进一大步，是它加速了直接操纵（Direct Manipulation）这种交互方式的普及。在 10.5 中，它作为一个重要的基础框架被提出，奠定了未来几年 Apple 在其各个产品上 UI 设计的基础。我们目前常见的最典型的应用就是 iPhone，可以说，Core Animation 是 iPhone 易用且绚烂多姿的 UI 的最大功臣。

下面是此技术在 iPhone 上的几个直接操纵的例子：

被技术放大的 UI 和交互，基本实现了 Jobs：不仅要作出一个高可用性的用户界面，更重要的是，Mac OS X 将具备一个名为 AQUA 的“Dream User Interface”之愿景。AQUA 一方面保留有古典系统高度易用的特点；另一方面，通过各种视觉特效的引入，AQUA 将带来前所未有的用户体验。

在近几年的拟物、扁平、分层……等设计风格的小幅变化外，UI 和交互最重要的变化就在于：如何满足不同人群的需求。过去，对于一个视力不好的人，会外挂辅助功能来解决：放大镜。如今，对于一个视力不好的人，手机淘宝利用智能 UI 这个牛逼的放大器，直接从 UI 和交互上来解决。

智能 UI：一枚牛逼的放大器

理论依据

先看一个 UGD（User Growth Design）的例子：

UGD 这个思想，试图回答：什么样的设计能够驱动增长？就像当初 UCD（User Center Design），试图回答：什么样的设计能够改善用户体验？然而，他们都忽略了一个问题：一万个人眼里有一万个哈姆雷特。假如把上面这些问题变成：什么样的设计能够驱动什么类型用户增长？什么样的设计能够改善什么类型用户的体验？我们就给一万个类型的人，每类提供一万种类型的 UI 和交互，再根据用户的行为数据判断每类用户喜欢那种类型的 UI 和交互，就能解决一万个人眼里有一万个哈姆雷特的问题。

实现方法

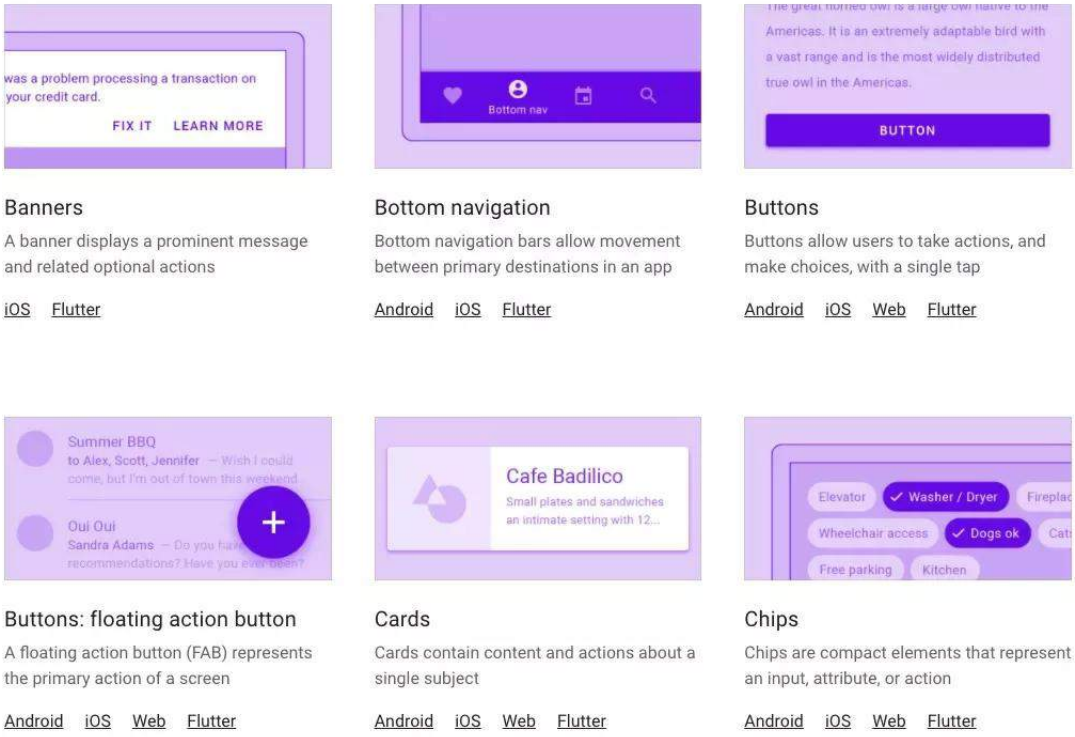
业务可变性方案

业务可变性方案需要和业务确认允许变化的范围和选项。比如，业务上有一个弹窗，如果业务需要中断用户当前操作流程，可变性就从“用弹窗还是浮层”问题降级到“必须用弹窗”，这和产品设计、UI 和交互设计无关。

假如业务允许在“用弹窗还是浮层”问题上做选择呢？一般来说，设计会介入和产品一起判断“用弹窗还是用浮层”，最后，还是拿着一个弹窗一个浮层两套方案，去找老板拍板。我们智能 UI 的做法却不同，两套方案都实现，根据不同类型用户在两套方案上的数据表现，分别给用户呈现不同的方案。

设计可变性方案

设计可变性方案需要和设计确认：不同类型用户可能在 UI 和交互上的差异（如视弱、冷淡、二次元……），并根据这些差异输出不同的设计方案。这里的设计方案主要由三部分构成：页面、模块、组件，很熟悉是不？《前端智能化》里有介绍我们针对页面构成的原理，如何设计智能生成页面代码的体系，如今 imgcook.com 已经在组件识别上取得了较大的进步。我们可以用组件去对应设计中 Symbol，例如 material design 的 Components：



通过不同的页面布局、模块布局、模块内布局，并组合不同样式的组件，最终给每类用户呈现出他们喜爱的 UI、提供他们最舒服的交互方式。

技术可变性方案

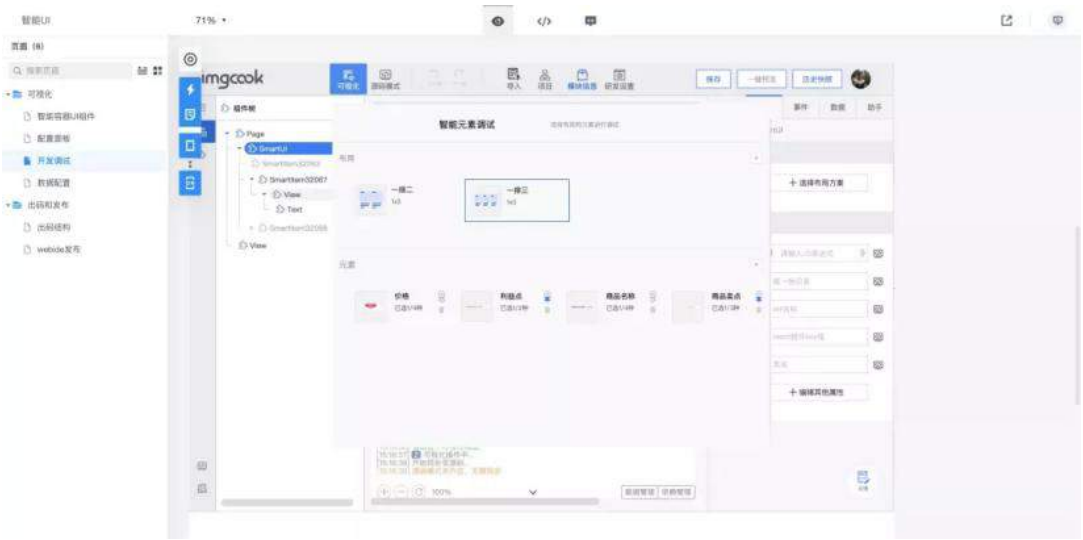
在网页出现的第一天，模板技术就几乎同时出现了，她的核心任务是静态的框架呈现动态的内容。模板像一个杯子，倒进去水，喝完了，再倒进去果汁，杯子还是那个杯子。而智能 UI 要求的是，一个茶杯，倒进去水，喝完了，拿个玻璃杯，倒进去果汁。模板自身也在变化，就是第一代智能 UI 的技术可变性方案。

在经历痛苦的模板配置后，第二代智能 UI 会根据设计系统的约束、设计美学算法模型打分、分人群优秀模板的机器学习等手段，用组件、模块自动化且智能的进行模板的生成。

事实上，如果粗略算一下：100 人群风格 x 30 组件元素 （平均每个页面用到的组件个数），即便模块和模板页面都可以自动生成，也有 3000 个组件需要开发，这庞大的开发工作量需要全新的开发方式来解决。

imgcook：一个大功率电源

为了能够给智能 UI 提供充分的组件生成和供给能力，我们对 imgcook.com 进行了升级，支持多态的组件生成能力：



当然，对于 imgcook 上生成交互逻辑、业务逻辑代码，以及数据字段的自动化绑定能力，都是保留和继承的。我认为简单的 UI 和交互，未来都可以业务级完整交付，辅之以机器视觉加持的自动化测试能力，甚至可以做到自动化上线。

写在最后

Alan Kay 说：用技术去放大人们的需求。就像照相技术放大了绘画的艺术性一样，智能 UI 技术放大了 UI 和交互的个性化，imgcook 放大了什么？对于接触编程 30 年、做前端 16 年的我来说，一直对前端被称为“切图仔”这件事儿深恶痛绝。在整个技术领域，待过腾讯和阿里的我，深感前端缺乏技术含量而被歧视的痛苦。imgcook 不仅能把前端“施工队”这个称号甩给机器学习和人工智能，我提出并坚持了三年多的“前端智能化”方向，也逐渐在众多领域里开花结果。如果你使用 VSCode 的 imgcook 插件，我们的 Code2Code 前端智能化方向代码纠错能力，能够帮助你智能提醒、自动推荐前端代码，还在 BFF（Backend For Front-end）胶水层代码生成上积极探索：Service2Code。

TypeScript 4.0 正式发布：核心概念梳理

作者 | 刘江涛（江墨）
出品 | 淘系技术

8 月 20 日，TypeScript 4.0 正式发布了（ Announcing TypeScript 4.0 ），虽然没有重大的变更和特性，可以看做是 3.9 版本正常迭代，不过 Daniel 也在公告中说了：对于初学者而言，现在是最好的上手时机。

In fact, if you're new to the language, now is the best time to start using it.

确实 TS 在经过了几年的发展后，使用 TS 的团队也越来越多，更重要的是 TS 的生态越来越完备，非常多的库、框架等都支持了类型系统甚至直接用 TS 重写，现在开始使用 TS 就能够直接享受整个技术生态带来的开发效率提升。回归到业务，我们团队最近也确实在开始用 TS 来进行类库的开发，所以结合官方文档和社区文档并从新人学习的角度梳理了一份包含大部分 TS 核心概念的学习手册。本文主要是做减法，梳理出核心。

背景

因为日常工作会使用页面搭建系统来生成很多前端页面，所以会开发很多的楼层模块配合搭建系统使用，最近模块是基于 Rax 开发的，后续会支持越来越多的投放渠道：web、weex、淘宝小程序、支付宝小程序等等，为了兼容约来越多的渠道，很多功能被抽象成了一个个小的类库，在类库中去兼容各个渠道，从而让模块中的业务代码保持尽量的只有清晰的业务逻辑。

但是随着支持渠道的增多，不可避免的导致类库在不同的渠道支持的特性不一致，比如 web 中，类库 A 支持三个参数甲、乙、丙，而在小程序中类库 A 仅支持两个参数甲、乙，所以丙这个参数要设计成可选参数，在类似的场景变多之后，这些类库的文档说明成了很重要的工作，同时在 IDE 中写代码时如果有类型系统能自动告诉开发人员这个函数支持哪些

参数就更好了，所以我们准备对类库进 TS 的重写来提高我们的生产效率，也为后续 TS 在团队内的落地打一个好基础。

什么是 TypeScript

- 简单的说 TypeScript 是 JavaScript 一个超集，能够编译成 JavaScript 代码。
- 其核心能力是在代码编写过程中提供了类型支持，以及在编译过程中进行类型校验。

先说一下 JS 的现状：

1. 在 JS 中的变量本身是没有类型，变量可以接受任意不同类型的值，同时可以访问任意属性，属性不存在无非是返回 undefined。
2. JS 也是有类型的，但是 JS 的类型是和值绑定的，是值的类型，用 typeof 判断变量类型其实是判断当前值的类型。

```
// JavaScript

var a = 123t
ypeof a // "number"a = 'sdf'
typeof a // "string"
a = { name: 'Tom' }
a = function () {
return true
}
a.xxx // undefined
```

TS 做的事情就是给变量加上类型限制：

1. 限制在变量赋值的时候必须提供类型匹配的值。
2. 限制变量只能访问所绑定的类型中存在的属性和方法。

举个简单的例子，如下是一段能够正常执行的 JS 代码：

```
let a = 100
if (a.length !== undefined) {
  console.log(a.length)
} else {
  console.log('no length')
  }
```

直接用 TS 来重写上面的代码，把变量 a 的类型设置为 number。

在 TS 中给变量设置类型的语法是 【 : Type 】 类型注解：

```
let a: number = 100
if (a.length !== undefined) { // error TS2339: Property 'length' does not exist on type 'number'.
  console.log(a.length)
} else {
  console.log('no length')
  }
```

但是如果直接对这个 TS 代码进行编译会报错，因为当变量被限制了类型之后，就无法访问该类型中不存在的属性或方法。

那再来写一段能正常执行的 TS：

```
let a: string = 'hello'
console.log(a.length)
```

编译成 JS 后的代码为：

```
var a = 'hello'
console.log(a.length)
```

可以发现 : string 这个类型限制编译之后是不存在的，只在编译时进行类型校验。

当 TS 源码最终被编译成 JS 后，是不会产生任何类型代码的，所以在运行时自然也不存在类型校验。

也就是说，假设一个项目，用 TS 来写，哼哧哼哧加上各种类型检验，项目测试通过部署到线上之后。

最后运行在客户端的代码和我直接用 JS 来写的代码是一样的，写了很多额外的类型代码，竟然是为了保证能顺利编译成原来的代码。

TypeScript 的作用

那 TS 的作用究竟是什么呢，主要是以下三点：

1. 将类型系统看作为文档，在代码结构相对复杂的场景中比较适用，本质上就是良好的注释。
2. 配合 IDE，有更好的代码自动补全功能。
3. 配合 IDE，在代码编写的过程中就能进行一些代码校验。例如在一些 if 内部的类型错误，JS 需要执行到了对应代码才能发现错误，而 TS 在写代码的过程中就能发现部分错误，代码交付质量相对高一些，不过对于逻辑错误，TS 当然也是无法识别的。

TypeScript 类型梳理

分两类来介绍 TS 的类型系统：

1. JS 中现有的值类型在 TS 中对应如何去限制变量。
2. TS 中拓展的类型，这些类型同样只在编译时存在，编译之后运行时所赋的值其实也是 JS 现有的值类型。

下文中会穿插一些类似 [xx] 这样的标题，这是在列举介绍 TS 类型的过程中插入介绍的 TS 概念。

JS 中现有的值类型如何绑定到变量

使用语法：类型注解【 : Type 】。

布尔值：

```
let isDone: boolean = false
```

数值：

```
let age: number = 18
```

字符串：

```
let name: string = 'jiangmo'
```

空值：

```
function alertName(): void { // 用 : void 来表示函数没有返回值
alert('My name is Tom')
}
```

Null 和 Undefined：

```
let u: undefined = undefined
let n: null = null// 注意：和所有静态类型的语言一样，TS 中不同类型的变量也无法相互赋值
age = isDone // error TS2322: Type 'false' is not assignable to type 'number'.// 但是因为 undefined 和 null 是所有类型的子类型，所以可以赋值给任意类型的变量
age = n // ok
```

[类型推论]

如果没有明确的指定类型，那么 TypeScript 会依照类型推论的规则推断出一个类型。

例如：定义变量的时候同时进行赋值，那么 TS 会自动推断出变量类型，无需类型注解。

```
let age = 18/
/ 等价于
let age: number = 18
// 所以上面代码中的类型声明其实都可以省略
// 但是如果定义的时候没有赋值，不管之后有没有赋值，则这个变量完全不会被类型检查(被推断成了any 类型)
let xx = 'seven
'x = 7
// 所以这个时候应该显示的声明类型
let x: number
x = 7
```

继续列举类型。

数组的类型

语法是 【 Type[] 】：

```
let nameList: string[] = ['Tom', 'Jerry']
let ageList: number[] = [5, 6, 20]
```

对象的类型

接口(interface)用于描述对象的类型：

```
interface Person { // 自定义的类型名称，一般首字母大写
   name: string
   age: number
}
let tom: Person = {
   name: 'Tom',
   age: 25,}
```

函数的类型

以函数表达式为例(函数声明定义的函数也是用类似的参数注解 语法来进行类型约束)。

```
// JavaScript

const sum = function (x, y) {
    return x + y
}
```

TS 中有多种语法来定义函数类型。

直接约束出入参类型：

```
const sum = function (x: number, y: number): number {
  return x + y
}
```

单独给 sum 变量设置类型：

```
const sum: (x: number, y: number) => number = function (x, y) {   return x + y}
```

这里如果把函数类型直接提取出来用并起一个自定义的类型名，代码会更美观，也易复用。

利用类型别名可以给 TS 类型重命名。

[类型别名]

类型别名的语法是【 type 自定义的类型名称 = Type 】：

```
type MySum = (x: number, y: number) => number
const sum: MySum = function (x, y) {
  return x + y
}
```

回到函数类型

用接口定义函数的类型：

```
interface MySum {  (a: number, b: number): number}
const sum: MySum = function (x, y) {
  return x + y
}
```

函数类型介绍完了，最后额外补充一下函数类型怎么定义剩余参数的类型以及如何设置默认参数。

```
const sum = function (x: number = 1, y: number = 2, ...args: number[]): number {
  return x + y
}
```

类的类型

和函数类型的语法相似，直接在 ES6 语法中用【 : Type 】类型注解和参数注解语法给类的属性和方法设置类型：

```
class Animal {
  name: string // 这一行表示声明实例属性 name
constructor(name: string) {
    this.name = name
  }
  sayHi(): string {
  return `My name is ${this.name}`
  }
}
let a: Animal = new Animal('Jack') // : Animal 约束了变量 a 必须是 Animal 类的实例
console.log(a.sayHi()) // My name is Jack
```

顺便值得一提的是，除了类型支持以外，TS 也拓展了 class 的语法特性。

新增了三种访问修饰符 public、 private、 protected 和只读属性关键字 readonly 以及 abstract 抽象类。

这里就不展开了，有需要的再去查阅一下官方文档即可。

内置对象和内置方法

JavaScript 中有很多内置对象和工具函数，TS 自带其对应的类型定义。

很多内置对象可以直接在 TypeScript 中当做定义好了的类型来使用：

```
let e: Error = new Error('Error occurred')
let d: Date = new Date()
let r: RegExp = /[a-z]/
let body: HTMLElement = document.body
```

一些内置的方法，TS 也补充了类型定义，配合 IDE 在编写代码的时候也能得到 TS 的参数提示：

```
Math.pow(2, '3') // error TS2345: Argument of type '"3"' is not assignable to parameter of type 'number'.
```

TS 中拓展的类型

任意值 any

与其说 any 是 JS 中不存在的类型，不如说原本 JS 中的变量只有一个类型就是 any

任意值 any 的特点：

any 类型的变量可以赋值给任何别的类型，这一点和 null 与 undefined 相同任何类型都可以赋值给 any 类型的变量，在任意值上访问任何属性都是允许的。

```
let a: any = 123
a = '123' // ok
let n: number[] = a // ok
a.foo && a.foo() // ok
```

所以 any 是万金油，也是和 TS 进行类型约束的目的是相违背的，要尽量避免使用 any。

联合类型

类型中的或操作，在列出的类型里满足其中一个即可：

```
let x: string | number = 1x = '1'
```

不过联合类型有一个额外约束：

当 TypeScript 不确定一个联合类型的变量到底是哪个类型的时候，我们只能访问此联合类型的所有类型里共有的属性或方法。

```
let x: string | number = 1
x = '1'
x.length // 这里能访问到 length ，因为 TS 能确定此时 x 是 string 类型
// 下面这个例子就会报错
function getLength(something: string | number): number {
  return something.length // error TS2339: Property 'length' does not exist on type 'string | number'.
  }
```

两种解决思路：

让 TS 能够自行推断出具体类型：

```
function getLength(something: string | number): number {
  if (typeof something === 'string') { // TS 能识别 typeof 语句
    return something.length // 所以在这个 if 分支里， something 的类型被推断为 string
} else {
    return 0
  }
}
```

利用类型断言，手动强制修改现有类型：

```
function getLength(something: string | number): number {
 return (something as string).length // 不过这样做实际上代码是有问题的，所以用断言的时候要小心
}
```

[类型断言]

用来手动指定一个值的类型，语法 【 value as Type 】。

用类型断言修改类型时的限制：

1. 联合类型可以被断言为其中一个类型
2. 父类可以被断言为子类
3. 任何类型都可以被断言为 any
4. any 可以被断言为任何类型

总结成一条规律就是：要使得 A 能够被断言为 B，只需要 A 兼容 B 或 B 兼容 A 即可。

双重断言

利用利用上述 3 和 4 两条规则，可以强制把一个值改为任意其他类型：

```
let a = 3
(a as any) as string).split // ok
```

如果说断言有风险，那双重断言就是在反复横跳了。

字符串字面量类型

用来约束取值只能是某几个字符串中的一个：

```
type EventNames = 'click' | 'scroll' | 'mousemove'
function handleEvent(ele: Element, event: EventNames) {   /
  / do something
  }
```

注意，只有一个字符串也是字符串字面量类型：

```
type MyType = 'hello'
```

虽然一般不会手动设置这样的类型，不过类型推论经常会推断出这种类型。

比如某次编译报错提示为：Argument of type '"foo"' is not assignable to parameter of type 'number'.

提示中的 type '"foo"' 一般就是根据字符串 'foo' 推断出来的字符串字面量类型。

元组

类似 Python 中的元组，可以看做是固定长度和元素类型的数组：

```
let man: [string, number] = ['Tom', 25]

// 不过 TS 中的元组支持越界
// 当添加越界的元素时，它的类型会被限制为元组中每个类型的联合类型
man.push('male')
```

枚举

用于取值被限定在一定范围内的场景，可以替代 JS 中用字面量来定义一个对象作为字典的场景：

```
enum Directions {
  Up,
  Down,
  Left,
```

```
    Right,
    }
let d: Directions = Directions.Left
```

这里看到 Directions.Left 直接把类型当做一个值来用了。

不是说类型是用于【 : Type 】类型注解 语法来约束变量，编译之后类型代码都会被删除吗?

为了解释这个问题，我们先来来看看单纯的类型代码会被编译成什么。

首先以一个联合类型举例:

```
type MyType = string | number | boolean
```

编译结果:

```
// 不会产生任何 JS 代码
```

再来看看枚举类型会被编译成什么:

```
enum Directions {
    Up,
    Down,
    Left,
    Right,
}
console.log(Directions)
```

编译结果:

```
var Directions
;(function (Directions) {
    Directions[(Directions['Up'] = 0)] = 'Up'
```

```
    Directions[(Directions['Down'] = 1)] = 'Down'
    Directions[(Directions['Left'] = 2)] = 'Left'
    Directions[(Directions['Right'] = 3)] = 'Right'
})(Directions || (Directions = {}))
console.log(Directions)
/*
  运行时 log 出来的 Directions 变量如下
  {
    '0': 'Up',
    '1': 'Down',
    '2': 'Left',
    '3': 'Right',
    Up: 0,
    Down: 1,
    Left: 2,
    Right: 3
  }
*/
```

这怎么理解呢？

```
let d: Directions = Directions.Left
```

其实这一行代码中，前一个 Directions 表示类型，后一个 Directions 表示值。

即 Directions 是一个值和类型的“复合体”，在不同的语法中具象化为值或者类型。

其实有办法可以把类型部分从 Directions 中抽离出来。

```
enum Directions {
  Up,
  Down,
  Left,
  Right,
}
type MyDirections = Directions
console.log(MyDirections) // error TS2693: 'MyDirections' only refers to a type, but is being used as a value here.
```

此时 MyDirections 就是一个纯粹的类型，不能当做一个值来使用。

其实之前介绍的函数类型、类类型等声明中，也存在这样的值与类型的“复合体”。

```
const sum = function (x: number, y: number = 5): number {
  return x + y
}
console.log(sum) // [Function: sum]
type MySum = typeof sum // 注意，剥离出来的函数类型是不会带有默认参数的，因为默认参数其实是函数的特性，和类型系统无关
const f: MySum = (a, b) => 3 // ok
console.log(MySum) // error TS2693: 'MySum' only refers to a type, but is being used as a value here.
```

然后再回到枚举。

字符串枚举

用字符串字面量初始化枚举成员，在实际使用过程中很常见：

```
enum Directions {
  Up = 'UP',
  Down = 'DOWN',
  Left = 'LEFT',
  Right = 'RIGHT',
}
console.log(Directions.Up === 'UP') // true
```

常数枚举

用 const enum 定义的枚举类型。

和普通枚举的区别就是对应的值也会在编译阶段被删除，只会留下枚举成员的值:

```
const enum Directions {
  Up = 'UP',
  Down = 'DOWN',
  Left = 'LEFT',
  Right = 'RIGHT',
}
let d = Directions.Left

// 如果取消注释下面这行代码，编译会报错
// console.log(Directions) // error TS2475: 'const' enums can only be used in property or index access expressions or the right hand side of an import declaration or export assignment or type query.
```

编译结果:

```
var d = 'LEFT' /* Left */
```

泛型

其实泛型并不是一种具体的类型，而是在定义函数、接口或类的类型时的的拓展特性。

泛型是类型系统里的“函数”，通过传入具体的 类型参数 来得到一个具体的类型，从而达到复用类型代码的目的。

假设一个场景，某个函数的入参类型为 number | string ，并且出参类型和入参相同。

先尝试用联合类型来约束出入参:

```
type MyFunc = (x: number | string) => number | string
```

但是 MyFunc 无法表示出参类型和入参相同，即入参是 number 的时候出参也是 number。

在这个场景下，可以利用泛型来定义出多个类似的函数类型。

泛型函数

<T> 表示声明了一个类型参数，在定义类型的时候 T 就可以作为一个类型来使用。

类型参数也可以定义多个，比如 <A，B，C>。

```
function GenericFunc<T>(arg: T): T {
  return arg
}
// 这里的 GenericFunc<number> 是表示的是一个函数值，同时将类型参数 T 赋值为 number
let n = GenericFunc<number>(1) // n 可以通过类型推论得出类型为 :number
// 进一步，利用 泛型约束 ，限制出入参为 number | string
type MyType = number | string
function GenericFunc<T extends MyType>(arg: T): T { // extends MyType 表示类型参数 T 符合 MyType 类型定义的形状
  return arg
}
let s = GenericFunc<string>('qq')
let b = GenericFunc<boolean>(false) // error TS2344: Type 'boolean' does not satisfy the constraint 'string | number'.
```

泛型接口

用泛型接口来定义函数类型：

```
interface GenericFn<T> {
  (arg: T): T
}
// 定义一个泛型函数作为函数实现
function identity<T>(arg: T): T {
  return arg
}
// 使用泛型时传入一个类型来使 类型参数 变成具体的类型
// <number> 表示 T 此时就是 number 类型，GenericFn<number> 类似是“函数调用”并返回了一个具体的类型（这里是一个函数类型）
const myNumberFn: GenericFn<number> = identity
const myStringFn: GenericFn<string> = identity
```

```
let n = myNumberFn(1) // n 可以通过类型推论得出类型为 :number
let s = myStringFn('string') // s 可以通过类型推论得出类型为 :string
```

对比上述的泛型函 和泛型接口，有一个区别，给泛型函数传参之后得到的是一个函数值，而不是类型：

```
// GenericFunc 是上面定义的泛型函数
type G = GenericFunc<string> // error TS2749: 'GenericFunc' refers to a value, but is being used as a type here.
```

而泛型接口传参之后得到的是一个类型，而不是函数值：

```
// GenericFn 是上面定义的泛型接口
type G = GenericFn<number> // ok
GenericFn<number>() // error TS2693: 'GenericFn' only refers to a type, but is being used as a value here.
```

泛型类

用泛型类来定义类的类型：

```
class GenericClass<T> {
    zeroValue: T
    constructor(a: T) {
        this.zeroValue = a
        }
}
let instance = new GenericClass<number>(1)
// 等价于
let instance: GenericClass<number> = new GenericClass(1)
// 因为有类型推论，所以可以简写成
let instance = new GenericClass(1)
```

内置的数组泛型

TS 中内置类一个数组泛型 Array，传入类型参数后会返回对应的数组类型：

```
// 数组的类型之前是用 【 Type[] 】 语法来表示的
let list: number[] = [1, 2, 3]
// 现在也可以这么表示
let list: Array<number> = [1, 2, 3]
```

[声明合并]

上面那个场景，某个函数的入参类型为 number | string ，并且出参类型和入参相同，其实不用泛型也可以用函数重载来实现。

函数的合并

即函数声明的合并，即函数重载。

TS 中的重载并不是真正意义上的重载，只是在根据不同的实参类型，从上而下挑选出一个具体的函数类型来使用：

```
function func(x: number): number
function func(x: string): string
function func(x: any): any {
  // 这里定义的函数类型 (x: any): any 会被覆盖失效
  return x.length
}
let n = func(1) // n 可以通过类型推论得出类型为 :number
let s = func('1') // s 可以通过类型推论得出类型为 :string
// 需要注意的是，如上重载之后只剩下两种函数类型，调用时的入参要么是 number 要么是 string，无法传入其他类型的值
let b = func(true) // error
/*
- error TS2769: No overload matches this call.
Overload 1 of 2, '(x: number): number', gave the following error.
  Argument of type 'true' is not assignable to parameter of type 'number'.
Overload 2 of 2, '(x: string): string', gave the following error.
  Argument of type 'true' is not assignable to parameter of type 'string'.*/
```

接口的合并

接口中方法的合并和函数的合并相同，但是属性的合并要求类型必须唯一。

```
interface Alarm {
  price: number
  alert(s: string): string
}
interface Alarm {
  weight: number
  alert(s: string, n: number): string
}
// 相当于
interface Alarm {
  price: number
  weight: number
  alert(s: string): string
  alert(s: string, n: number): string
}
```

声明文件

- 以 .d.ts 结尾的文件。
- 声明文件里面 100% 全部都是纯类型的声明，不会编译出任何 JS 代码。

一般来说，TS 会解析项目中所有的 *.ts 以及 .d.ts 结尾的文件，从而获取其中这些类型声明。

使用场景：作为一个第三方类库的开发者。

假如你的库是用 TypeScript 写的，但是最终你的库分发出去的时候要编译成 JS。

否则这个类库就只能给 TS 项目来使用了，因为没有使用 TS 的项目没法直接引用 .ts 文件。

但是编译成 JS 有一个问题就是类型代码都被删除了之后，对于 TS 项目的使用方来说就没法继承 TS 的三大优势（文档、自动补全、类型校验）。

所以需要在类库的编译产出文件中保留一些 .d.ts 类型声明文件，和编译出来的 JS 文件中导出的函数、类等进行一一匹配。

这样 JS 文件和类型文件分离之后，你的类库就可以同时被 JS 项目和 TS 项目引用了。

假如你的库是并没有使用 TypeScript 来编写，那就需要额外有人给这个库写单独的声明文件。

比如 jQuery 并不是用 TS 来写的，但是你可以安装单独的 TS 类型包来实现补全这个包的类型系统。

```
npm install @types/jquery --save-dev
```

感谢&全文引用：

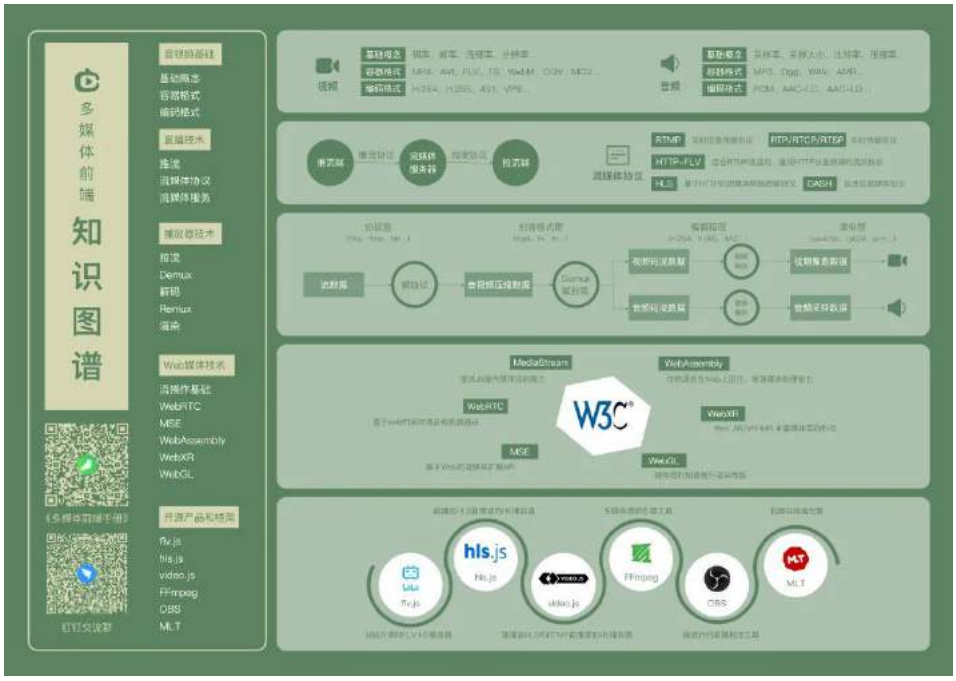
TypeScript 入门教程
TypeScript 官方手册
https://www.typescriptlang.org/docs/handbook/basic-types.html
TypeScript Handbook
https://zhongsp.gitbooks.io/typescript-handbook/content

一张图讲清楚淘宝直播背后的前端技术

作者 | 淘系多媒体前端
出品 | 淘系技术

2020 年，直播带货火爆全网。想一探淘宝直播背后的前端技术？本文将带你进入淘宝直播前端技术的世界。

对于大多数前端工程师来说，音视频技术是一个比较少涉足的领域，本文涵盖了流媒体技术中的文本、图形、图像、音频和视频多种理论知识，涉及到播放器、web 媒体技术、主流框架等介绍，只需要花上一点点时间，你将进入前端多媒体的领域。



音视频基础

视频

基础概念

码率	单位时间内取样率越大，精度就越高，处理出来的文件就越接近原始文件。
帧率	对视频来说，帧率对应这观看卡顿。帧率越高，流畅度越高，低帧率会造成视觉卡顿。
压缩率	经过压缩后文件的大小 / 原始文件的大小 * 100% = 压缩率。编码压缩越小越好，但压得越小，解压时间。
分辨率	用于度量图像内数据量多少的一个参数，和视频清晰度息息相关。

视频容器格式

容器格式相信大家经常见到：MP4、AVI、FLV、TS/M3U8、WebM、OGV、MOV...

H.264	目前最流行的编码格式。
H.265	新型的编码格式，高效的视频编码。用来以替代 H.264/AVC 编码标准。
VP9	VP9 是 WebM Project 开发的下一代视频编码格式 。VP9 支持从低比特率压缩到高质量超高清的所有 Web 和移动用例，并额外支持 10/12 位编码和 HDR。
AV1	AOM（Alliance for Open Media，开放媒体联盟）制定的一个开源、免版权费的视频编码格式。AV1 是 google 制定的 VP9 标准的继任者，也是 H265 强有力的竞争者。

音频

基础概念

采样率	音频采样率是指录音设备在一秒钟内对声音信号的采样次数，采样频率越高声音的还原就越真实越自然。
采样大小	一秒钟所采的样本数为比特率，每个样本中信息的比特数就是位深，即采样精度，单位为 Bit。
比特率	指每秒传送的比特(bit)数，又称数据信号速率。单位为比特/秒、千比特/秒或兆比特/秒。比特率越高，表示单位时间传送的数据就越多。
压缩率	原始音频数据与通过 PCM 等压缩编码技术压缩后的数据大小的比率。

音频容器格式

音频格式也比较常见：WAV、AIFF、AMR、MP3、Ogg...

音频编码格式

PCM	脉冲编码调制(Pulse Code Modulation,PCM)，PCM 是数字通信的编码方式之一。
AAC-LC(MPEG AAC Low Complexity)	低复杂度编码解码器（AAC-LC — 低复杂度高级音频编码）是低比特率、优质音频 的高性能音频编码解码器。
AAC-LD	（又名 AAC 低延迟或 MPEG-4 低延迟音频编码器），为电话会议和 OTT 服务量身打造的低延迟音频编解码器。
LAC（Free Lossless Audio Codec）	免费无损音频编解码器。是一套著名的自由音频压缩编码，其特点是无损压缩。2012 年以来它已被很多软件及硬件音频产品（如 CD 等）所支持。

直播技术

流媒体协议

每一个你在网络上观看的视频或音频媒体都是依靠特定的网络协议进行数据传输，基本分布在会话层（Session Layer）、表示层（Presentation Layer）、应用层（Application Layer）。

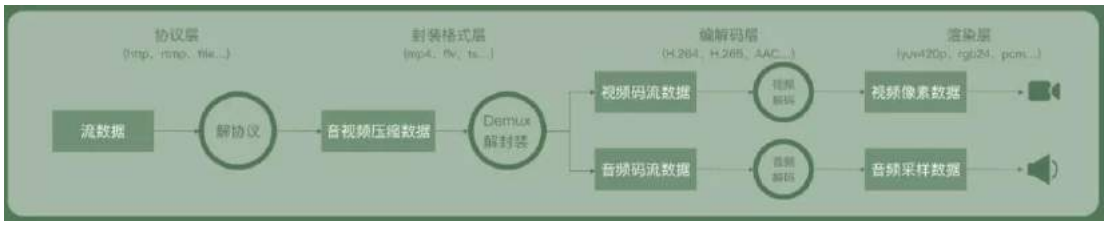
常用协议：RTMP、RTP/RTCP/RTSP、HTTP-FLV、HLS、DASH。各个协议都有自己优势与劣势。

推拉流过程

主播在设备上开启直播，采集设备将主播声音及画面采集后通过对应协议推流到「流媒体服务器」上。此时观看端(即拉流端)通过拉流协议即可从「流媒体服务器」上拉取到流数据进行播放。

播放器

本节主要讲述播放器相关技术，在本节中会简要讲述播放器在拿到相关流之后如何运作。



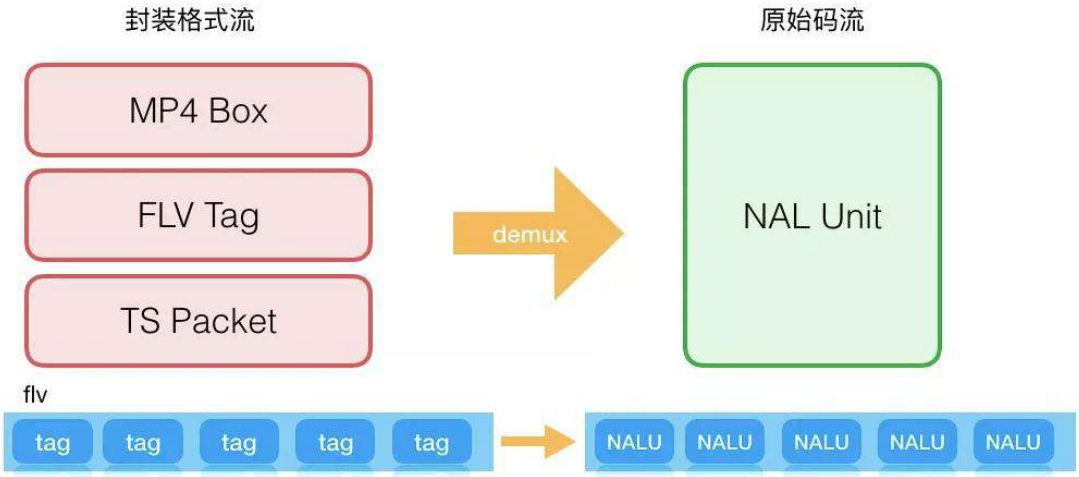
拉流

第一步是拉流，在播放之前率先需要拿到视频流才可能执行播放。

举个例子，flv 格式的视频流数据，我们可以通过浏览器提供的：Fetch API、Stream API 进行拉流。

解封装

拿到流数据之后，紧接着需要执行解封装操作。在开始播放的之前，需要把图像、声音、字幕(可能不存在)等从拉取的流数据中分离出来，这个分离的行为和过程就是解封装(demux)。



在解封装之后获得图像、声音、字幕等基本流，而后基本流可以通过解码器进行解码。

demux(解码)

从上层解封装中，我们了解到，在解封装之后，需要对分离出来的原始码流进行解码，生成音、视频播放器可播放的数据。

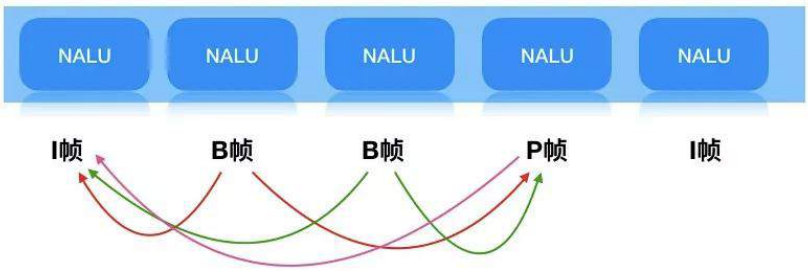
在解码过程中，我们会得到各式各样的数据，我们挑选几个重要的来讲：

SPS 和 PPS

这俩哥们儿决定了最大视频分辨率、帧率等以及还有一系列视频播放当中的参数。PPS 通常与 SPS 一起，保存在码流的起始位置。

- SPS、PPS 中保存了一组编码视频序列的全局参数，如果丢掉，解码过程很可能 GG。

IBP 帧



- I 帧，关键帧。I 帧进行帧内预测，可以单独解码本帧的数据，I 帧通常是每个 GOP
- （MPEG 所使用的一种视频压缩技术）的第一帧，经过适度地压缩，作为随机访问的参考点可以当成静态图像。
- B 帧，向前预编码帧。它要使用一个前面的 I 帧或 P 帧和一个后面的 I 帧或 P 帧进行预测。不仅要取得之前的缓存画面，还要解码之后的画面，通过前后画面的与本帧数据的叠加取得最终的画面。

- P 帧，前向预测编码在帧(predictive-frame),通过将图像序列中前面已编码帧的时间冗余信息去充分去除压缩传输数据量的编码图像，也成为预测帧。

SEI(补充增强信息)

视频编码器在输出视频码流的时候，可以没有 SEI。举个简单例子，之前特别火的直播答题，通过 SEI 传递较多和答题业务相关的信息，并通过 SEI 承载的信息，优化题目显示和观众音视频观看的同步性。

PTS 和 DTS

- DTS（Decoding Time Stamp）：即解码时间戳，这个时间戳的意义在于告诉播放器该在什么时候解码这一帧的数据。
- PTS（Presentation Time Stamp）：即显示时间戳，这个时间戳用来告诉播放器该在什么时候显示这一帧的数据。
  简而言之，这俩哥们儿很可能直接决定了你音视频播放是不是同步的。

解码还会生成各式各样的产物，这里就不展开介绍了，有兴趣的同学可以在本篇最后查看。

remux(复用)

有 demux，自然就有 remux。把基本的音频 ES、视频 ES、字幕 ES 等组合成一个完整的多媒体就是 Remux(复用)。

对一个视频来说，改变封装格式，改变视频编码，需要 remux 和 demux 的配合。这里不展开叙述。

渲染

渲染，指的是将解码后的数据，在 pc 硬件上（显示器、扬声器）进行播放。负责渲染的模块我们称之为渲染器(Render)，视频渲染器主流有 EVR（Enhanced Video Render）以及 madVR (madshi Video Render)，Web 播放器一般使用 video 标签来嵌入。

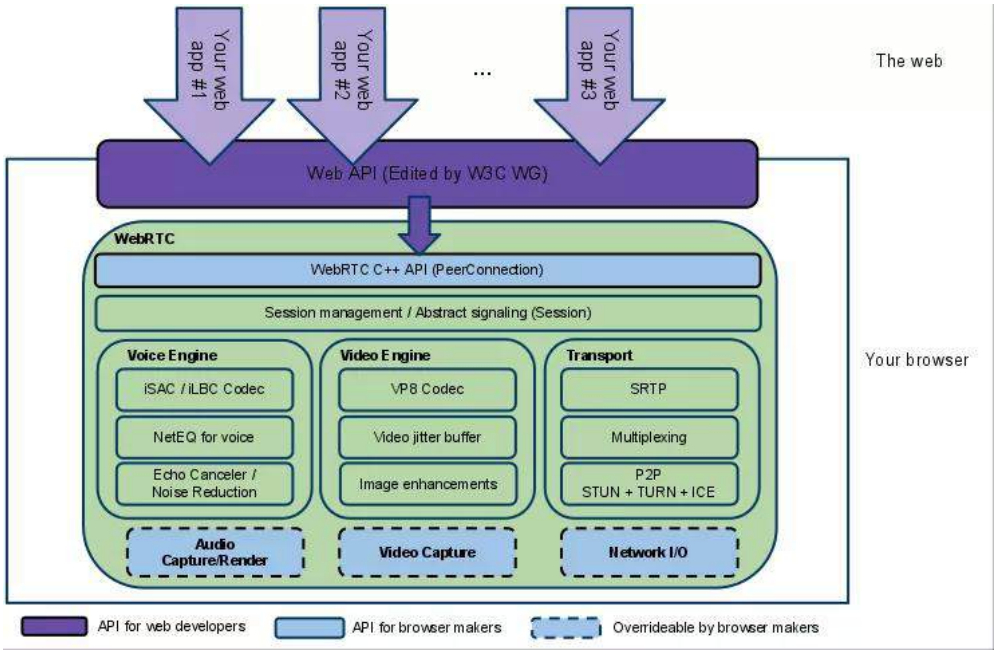
自定义渲染：以我们的 H.265 播放器为例，利用浏览器提供的接口来实现一个模拟的 video 标签，通过 canvas 和 audio 来实现渲染。

web 媒体技术

webRTC

网页即时通信（英语：Web Real-Time Communication），它允许网络应用或者站点，在不借助中间媒介的情况下，建立浏览器之间点对点（Peer-to-Peer）的连接，实现视频流和（或）音频流或者其他任意数据的快速传输。

组成形式：视频引擎（VideoEngine）、音效引擎（VoiceEngine）、会议管理（Session Management）、iSAC(音效压缩)、VP8（Google 自家的 WebM 项目的视频编解码器）、APIs（Native C++ API, Web API）



MSE

用过播放器的同学对于 MSE 肯定不会陌生。媒体源扩展 API（MSE） 提供了实现无插件且基于 Web 的流媒体的功能。使用 MSE，媒体串流能够通过 JavaScript 创建，并且能通过使用 audio 和 video 元素进行播放。

MSE 大大地扩展了浏览器的媒体播放功能，提供允许 JavaScript 生成媒体流。这可以用于自适应流（adaptive streaming）及随时间变化的视频直播流（live streaming）等应用场景。

这里不展开叙述 MSE 的使用，感兴趣的同学可以去搜索一下 MSE，相信能帮助到你们。

WebXR

XR 是 Extended Reality (扩展现实) 的简写，包括了 VR (虚拟现实)，AR (增强现实)，MR (Mixed reality，混合现实)，WebXR 支持各种 XX 现实的设备。WebXR 允许开发人员创建在所有 VR/AR 设备都可运行的沉浸式内容，以实现基于 Web 的 VR/AR 体验。

WebGL

WebGL（全写 Web Graphics Library）是一种 3D 绘图标准，并允许用户与之交互。这种绘图技术标准允许把 JavaScript 和 OpenGL ES 2.0 结合在一起，通过增加 OpenGL ES 2.0 的一个 JavaScript 绑定，WebGL 可以为 HTML5 Canvas 提供硬件 3D 加速渲染，这样 Web 开发人员就可以借助系统显卡来在浏览器里更流畅地展示 3D 场景和模型了，还能创建复杂的导航和数据视觉化。

WebGL 基于 canvas 画布来进行渲染。在「播放器」章节，我们了解到播放器可以通过 canvas 实现播放器图像渲染，通过 WebGL，播放器播放流畅性能等能力得到增强。

WebAssembly

WebAssembly 或者 wasm 是一个可移植、体积小、加载快并且兼容 Web 的全新格式，是由主流浏览器厂商组成的 W3C 社区团体制定的一个新的规范。

感兴趣的同学可以去 https://webassembly.org/ 了解学习。

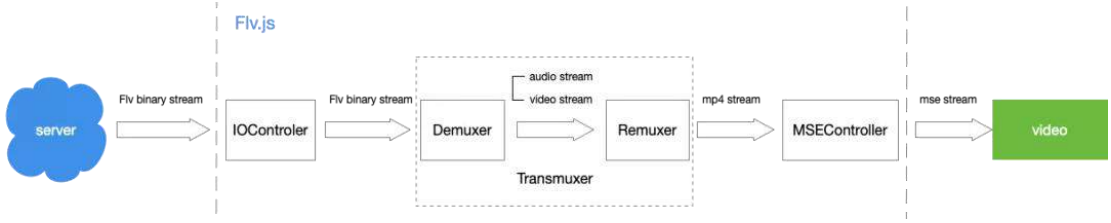
基于 wasm，播放器可以与 FFmpeg 结合，对目前浏览器器不能够识别的 H.265 视频进行解码。

使用的开源产品和框架

市面上目前流行着许多开源产品和框架，我们为各位同学收集了一些优秀的主流框架。

flv.js

flv.js 是 Bilibili 网站开源的 HTML5 flv 播放器，基于 HTTP-FLV 流媒体协议，通过纯 js 实现 FLV 转封装，使 flv 格式文件能在 web 上进行播放。

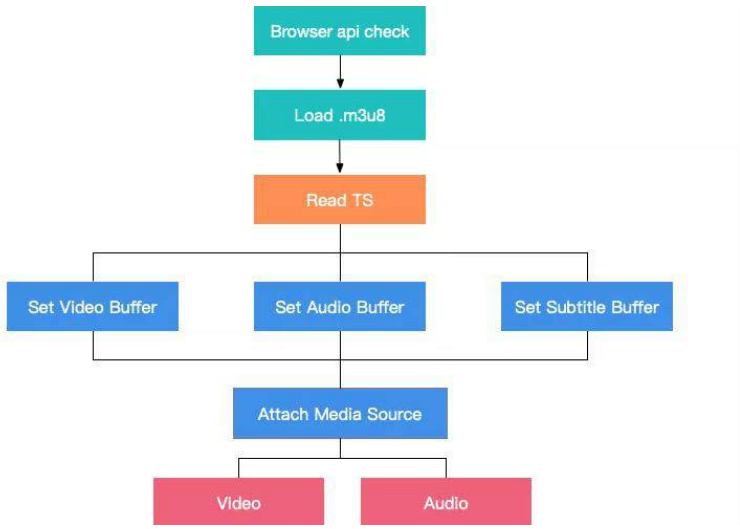


官方 Github：https://github.com/bilibili/flv.js

hls.js

hls.js 是基于 Http Live Stream 协议开发，利用 Media Source Extension，用于实现 HLS 在 web 上播放的一款 js 播放库。

值得一提的是由于 HLS 协议由苹果提出，并且在移动端设备上广泛支持，因此可以被广泛应用与直播场景。



官方 GitHub：https://github.com/video-dev/hls.js/

video.js

video.js 是一款基于 html5 的播放器，同时支持 h5 和 flash 播放，并且拥有超过 100 个插件可进行使用，可满足 hls、dash 格式播放，支持定制主题，字幕扩展等不同层次的诉求，在世界范围拥有大量的应用场景。

官方 GitHub：https://github.com/videojs/video.js
官方文档：https://videojs.com/

FFmpeg

FFmpeg 是一套领先的多媒体框架，是一套开源且跨平台的多媒体解决方案，提供了音视频的编码、解码、转码、封装、解封装、流媒体、滤镜、播放等功能，官网地址。

对于前端来说 FFmpeg 可以用来：

- JS 播放器：可以基于 FFmpeg 和 WebAssembly 实现浏览器端的 JS 播放器，或扩展浏览器端其他的音视频能力。
- Node 模块 fluent-ffmpeg：node.js 中非常实用的模块，该模块简化了 ffmpeg 复杂的命令操作，且配合文件上传以及视频流的处理等非常实用，更多详情可参考 fluent-ffmpeg。

OBS

OBS（Open Broadcaster Software）是一个用于录制和进行网络直播的自由开源软件包。OBS 使用 C 和 C++语音编写，提供实时源和设备捕获、场景组成、编码、录制和广播。数据传输主要通过实时消息协议（RTMP）完成，可以发送到任何支持 RTMP 的目的地，包括 YouTube、Twitch.tv、Instagram 和 Facebook 等流媒体网站。

在视频编码方面，OBS 可以使用 X264 自由软件程序库、Intel Quick Sync Video、Nvidia NVENC 和 AMD 视频编码引擎将视频流编码为 H.264/MPEG-4 AVC 和 H.265/HEVC 格式。音频可以使用 MP3 或 AAC 编解码器进行编码。进阶用户可以选择使用 Libavcodec/libavformat 中的任何编解码器和容器，也可以将流输出到自定义 FFmpeg URL。

MLT

MLT 是一个够用于多种类型 app 非线性视频编辑器引擎，且不局限于桌面领域(同样适用于 Android、iOS 等平台，功能十分强大。

官网地址：https://www.mltframework.org/
Github：https://github.com/mltframework/mlt/

由于篇幅问题，本文仅介绍目前淘宝直播背后的技术，更多技术细节可参考淘宝直播知识文档，或点击下方「阅读原文」。

https://www.yuque.com/webmedia/handbook/knowledge-map

前端人精彩问答

Q: 大公司里怎样开发和部署前端代码?

作者 | 上坡

出品 | 淘系技术

主要有以下问题:

开发时的和部署时类库的引用和存放是一致还是不同?

模块放在项目中还是放在 CDN 之类服务器?

渲染网页用 Nginx 还是其他动态语言的 Web 服务器?

制作网页的流程，是先有设计师的稿，还是先看模块?

会选择用自己写的模块还是从社区寻找模块?

看了看，问题和答案都有些年头了。要更新。

从 2013 年到 2020 年，阿里是怎么开发和部署前端代码的，我们打算敞开聊一聊。

在阿里的淘系前端团队，开发与部署的模式随着技术的发展，仍然正在处于不断变化的过程中。一方面体系内外纷繁复杂的能力模块不断地向前发展，另一方面 LSP、DAP 等底层技术基础也逐渐成熟。

在当下，我们正在通过集成研发环境 IDE 的方式，将上一个阶段孵化沉淀的研发生态进行再一次的整合升华，将原有的链路再次重组，从当下用户的痛点中找到突破口，发掘当下各个研发场景中的最佳能力组合，搭建通用底层平台，升级变革现有模式。

作为从 14 年实习到 15 年正式进入公司的前端开发，在前身为淘宝前端团队的 淘系前端团队 经历了大大小小的开发和部署方式的更迭，同时也有幸在整个的变革潮流中参与过其中的一些能力的建设。今天从一个亲历者的角度，通过自身经历与向同事考究，以分阶段的方式来进行回忆和描述。

回答将整个故事分成四个阶段，首先是 13 年左右以代码发布存储、部署改造成 Gitlab 技术体系为主旋律的 "石器时代"； 14 年左右伴随着 NodeJS 技术成熟，用前端 JS 语言建设工程化工具的 "白银时代"；从 15 年开始，在借助 NodeJS 技术完成工程化工具尝试之后，更系统化建设线上线下前端工程化体系的 "黄金时代"；以及在当下我们正在通过客户端、容器、算法等多元化技术打造未来研发模式的"未来时代"。

石器时代

在 13 年前，前端的研发模式其实与后端研发的方式没有太大的差异，大部分是基于 SVN 进行 SCM 代码管理。在完成每天的代码研发工作后，通过命令行或者 小乌龟 等工具将代码上传到 SVN 服务器中完成一天的开发流程。在部署阶段，通过手动拷贝或者 FTP 上传的方式，将测试代码进行上传到测试服务器上，在完成测试之后，人工检查下代码版本及内容，然后进一步上传到生产环境，完整整个研发部署流程。

在那个时期，代码管理工具除了 SVN 之外，也出现了基于 GIT 协议实现的代码管理工具 Gitlab。面对 SHA-1 算法带来的版本检测便利性、本地分布式的代码版本控制的灵活性等优势，我们逐步将部门内的 SVN 研发工作流程迁移到了 Gitlab 上。同时这个变化也算是淘系前端研发变革的起源。

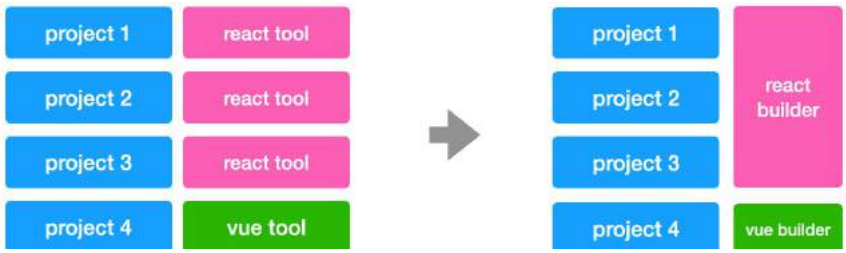
在经历代码版本管理工具切换带来的便利同时，我们思考当时较为操作繁琐、需要人工保证的部署流程是否也能进一步的在新的体系上得到改善。在新版发布系统的背景下，我们在新的 Gitlab 系统所提供的各种能力中，发现可以将 webhook 机制进行一定的上层封装，借助规则化的操作，来触发与发布系统的关连，自动完成发布上线流程。我们基于 publish/版本信息 这样固定规则的 Git Tag 来触发 webhook 事件通知，触发发布系统的调用上线流程，从而完成了第一个实际意义的前端研发全自动化发布流程。

这个阶段就像 "石器时代" 般原始，不过在完成了研发基础设施这块如釜底抽薪般的改造之后，为后续的发展提供了坚实基础。同时随着前端技术发展的契机，我们也基于这套底层体系进入了新的阶段。

白银时代

在 14 年左右的时候，已经诞生 5 年多的 NodeJS 技术也逐渐变得成熟起来，当时团队基于 NodeJS，产出了一套名为 DEF 的研发本地端 CLI 终端工具，工具的核心是一套 Node 模块的安装、调用管理机制。工具开发者通过将研发功能模块封装成 DEF 体系下的 插件 形式，通过不同 插件 模块的调用组合，来实现对前端项目的编译、调试、上线等操作。

在当时 KISSY 框架研发体系下，我们借助 NodeJS 的能力，通过正则匹配、 UglifyJS 解析 AST 等方式进行脚本编译处理，通过 JS 技术开始构建起前端领域的工程化工具，逐渐替代掉了以 java 体系中借助 Ant 平台实现的工具。



在当时我们除了借助使用 yeoman 等工具来完成初始化流程，也随着业务编译构建逻辑的不断抽象产出 构建器 这样的基础概念。

当时在传统的项目组织中，构建编译的配置逻辑与项目的代码文件是放置到同一个代码目录中。从团队视角下，这块构建编译逻辑零散自由，谈不上有统一的更新管理逻辑。如果团队内某一类的研发场景构建编译工具本身有任何的更新变化，在团队范围内工具的更新覆盖将需要消耗非常大的成本；与此同时，在用户的角度，在相同类型的不同项目中，也存在着大量重复安装的构建依赖，每一个项目在构建前也都需要安装一遍构建依赖，在空间和时间上都有一定占比的浪费。

通过 构建器 概念将编译构建依赖进行收敛、抽象，将项目的编译依赖维护在一个 npm 包中。这样编译逻辑就从多对多变成了一对多，在大幅降低构建逻辑的空间占用的同时，也能较好的复用已经安装的构建依赖，简化构建环节，提升构建效率。进一步也为后续的线上构建体系埋下了伏笔。

在这个阶段，NodeJS 技术也逐渐变成了我们除日常页面开发之外的又一个基本技能。伴随着 NodeJS 工具的诞生与规模应用，前端开发与部署从 "石器时代" 进入到 "白银时代"。随着大家对 NodeJS 体系的基本运用变得愈发成熟，我们在基础体系的使用之上，进一步开始进行更深入的思考发散和抽象设计，开始建设更成熟的前端工程体系。

黄金时代

在团队完成基于 NodeJS 技术的工程工具以及 Web 服务框架等基础设施建设之后，在随后的几年时间内，越来越多的更为成熟的线上线下的工程化体系发展起来。在那个阶段设计、建设的服务、工具后续也逐步形成了如今淘系前端研发、部署流程中的基础设施。

研发套件

随着本地 DEF 工具的发展，越来越多工具插件雨后春笋般地涌现出来。一方面用户有着非常多的工具选择，基本上本地研发过程中的功能都能在插件生态中找到；但是另一方面，在实际用户视角下，在某一个项目开发的时候，用户需要对这个项目使用到的插件以及插件用法要有一定的熟悉程度、知道最佳的组合使用方法。随着项目逐渐变多，对于工具组合的记忆成本变得高起来，同时也需要注意到不同项目之间插件组合的切换。

在百花齐放之后，面对延伸出来的问题，在原有的本地 DEF 工具之上，我们提出了 研发套件 的概念。通过对本地研发过程中进行总结收敛，抽象出 init 初始化、dev 编译预览、

build 构建、test 测试、publish 发布 这五个本地工具的功能节点。将原有的插件能力根据项目研发类型进行分类后，总结沉淀出每一类研发类型的标准本地工具 --研发套件。

借助套件概念的封装，用户在不同项目之间研发的时候直接通过 统一的命令 ，就能完成项目研发对应工具服务的启动使用。同时在新的本地套件体系，也能针对每一位用户有着更细粒度、灵活的版本指定方式。配备更完善的日志监控体系，在使用工具的用户遇到问题时能实时感知、解决，最大程度保证一线同学本地研发工具的使用体验。

研发部署平台

在借助 Gitlab 的 webhook 能力发布前端资源的流程成熟之后，一方面在发布流程上研发用户期望有更优化的发布体验，另一方面团队对于前端研发流程更体系化、结构化的流程治理、数据统计也有了更高的要求。

在这样的背景下，我们通过打通原有链路上发布能力，将发布链路中各个环节进行更友好的串联，产出了前端体系下的研发部署平台。在体验上，用户可以借助一个简单的命令行命令直接开始发布任务的启动，简化了发布时 git 发布 tag 的操作流程。在整个发布流程中，通过长连接的方式将各个环节的运行信息、日志进行返回展示，提供了全新操作的一条发布链路。用户第一次开始对发布流程有着更全面的掌控，从开始进行代码的提交检测到最后资源 cdn 上线都有着完善的信息披露，极大的提高了发布体验。

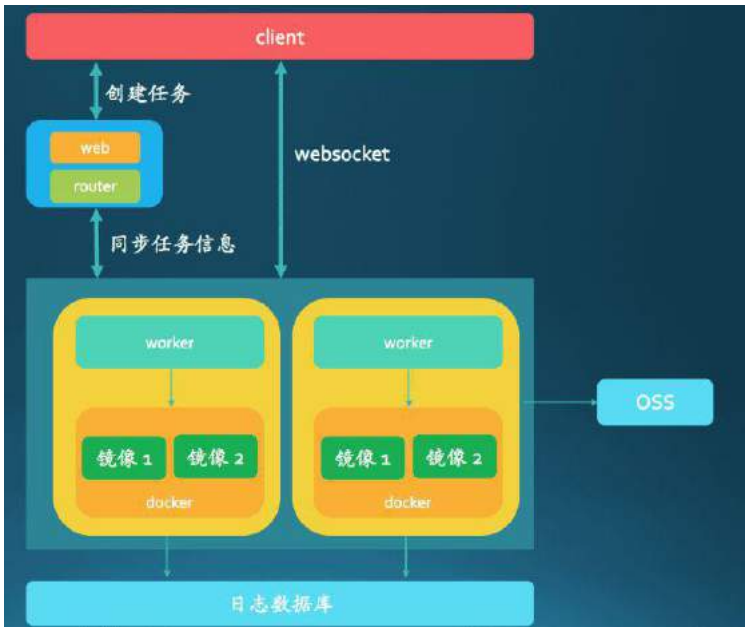
基于内部 NodeJS 实现的研发部署平台，在完成了基础链路的打通之后，在更体系化的角度上也逐步地开始作为工程研发上线流程的基础，对原有的上线发布链路的进行重塑。将原有流程中的环节进行抽象，逐步分散到 构建、检测、发布 三个主要步骤。在随着业务研发类型从 PC 到无线的切换过程中，基于三个基础步骤，在执行上层也逐步孵化出前端领域下 发布类型 的概念。例如在当时存在着 Web 应用类型、前端资源类型、Weex 应用 等

类型，底层系统针对每个细分的 发布类型，在整体的发布流程的细节对接上采取不同的发布部署方案，本地研发结合套件工具，形成线上、线下链路关联统一的研发组合方式，更精细化地为每一种研发模式都提供最佳的研发部署体验。

用户从此之后就不再需要通过 Gitlab 的 webhook 触发发布流程，"一键上线" 代表着前端工程体系又进入新的一个阶段。

云构建

在上一个阶段中，我们通过将编译逻辑经过中心化封装的方式，借助 npm 的形式，形成了在研发编译构建环节的规范描述。但是在部署上线环节，构建器并未与线上部署流程进行串联，而不同的本地系统环境，同样也会导致构建结果的不稳定。



在当时伴随着 docker 技术的发展，我们思考通过 docker 快速启停统一环境的能力来实现一个构建器运行的环境，通过 docker 启动的 container 容器来模拟执行本地的编译构建流程。同时利用统一的容器环境，结合前文介绍的迁移之后的 Gitlab 提供的 commit 等版本标识信息，能在最大程度上保证构建内容的稳定统一。

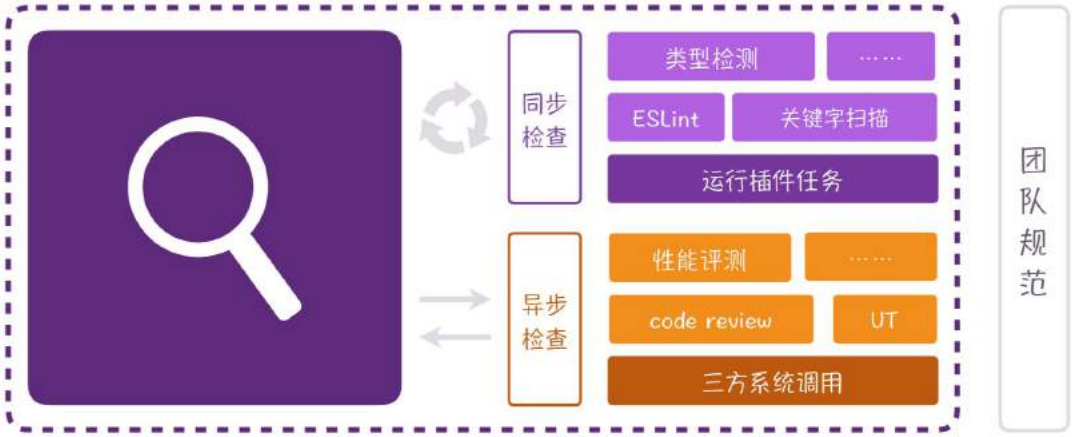
经过了一段时间的摸索，通过在体系内经历了例如 建立前端 docker 容器集群、串联打通容器与应用的网路通道、建设构建任务调度策略逻辑、借助 redis 打通 docker 容器运行日志等技术环节的验证探索之后，我们基于容器技术搭建起来了在前端持续集成领域的 云构建 系统。

在通过淘系的 NodeJS 应用框架 midwayjs 完成线上构建能力搭建的同时，在业务侧我们也在逐步建立更友好、完善的线上构建业务逻辑。借助在线上发布流程时调用 云构建 服务，在整个构建任务的执行流程中，串联起来了 构建器、仓库、用户 三者的业务关系。在这个关系网下，我们建立了完善的任务执行情况的调度管理，能完整记录构建器任务执行的具体情况。

在构建器视角下，可以看到构建器针对不同仓库的执行情况，有着完整的运行日志与构建时间、构建错误记录，方便用户针对构建器进行错误排查以及性能优化。而针对构建器的迭代更新，构建器开发者可以针对用户进行灰度范围划分设置，在版本更迭时通过完整的灰度流程保证上线的稳定覆盖。在构建器发布过程中如有问题，立即取消灰度进行处理。通过这样的模式，最大化地降低发布版本的风险。在针对横向覆盖较大的场景下，完整的灰度发布机制起着举足轻重的作用。

用户在研发过程中需要将本地构建代码推送到仓库的这个环节也从此变成了历史，在源码完成开发之后，直接提交源码到仓库中，执行发布任务的同时，云构建系统会自动构建出稳定、统一的构建产出结果。

门神



在完成在完成项目的编译、压缩过程之后，看似已经走到部署环节的前端资源，在面对不断复杂起来的业务场景下，实际仍然存在着许多隐蔽的问题，而以往通过人工检查的方式已经逐渐无法保障上线资源的可靠性。在这样的背景下，也需要一个流程化、规范化的方式进行系统干预、检测，做好发布资源的最后一道防线。

与 云构建 类似，基于 NodeJS 搭建线上资源检查运行环境，通过将检查逻辑也进一步抽象化，同样以 npm 的形式进行承载抽象出 检查器 的概念。在用户资源完成所有的前置处理后，再进入到门神系统并行检查任务的执行，在完成例如 资源地址、敏感词、代码注释 等等检查之后，通过不同分级的检查结果，反馈用户相应的处理方式。作为上线前资源的最后一道关卡。系统取名 门神 寓意保障每一次的发布资源的上线基本质量和安全。

通过门神自动化的方式，用户不再需要在上线前"火眼金睛"检验各种无法通过常规编译工具发现的问题，轻松上线。

能力开放

一方面随着体系内前端工程领域不断的发展，不断有更多的研发解决方案随着业务场景的不断发展涌现出来，另一方面随着体系内 研发部署平台、云构建、门神 逐步成为集团前端研发场景的事实规范和基础设施。在这个背景下，底层能力得到了进一步抽象开放的契机，上

层的业务体系可以根据自身的业务情况，通过借助成熟的底层工程中台能力搭建属于自身业务体系的前端研发流程。

例如从阿里体系视角下，不同的投放客户端，不同的资源组织形式，借助通用抽象化的底层工程能力，最终让每一个业务场景的开发者都能拥有在项目业务场景下的最佳研发体验。

搭建平台

如何设计阿里经济体都在用的搭建服务—天马

除了常规的项目研发，在电商体系下，特别是在营销场景下存在着需要快速生产页面的需求。在这样的业务场景背景下，孵化出来了通过模块生成页面的搭建平台。通过将页面进行重新划分，分割成 页面骨架 和 页面模块 的方式，通过前端同学开发这些页面元素，完成开发后进行组合搭配，同时打通业务数据流，实现快速搭建页面的能力。

前端研发同学在完成本地模块开发后，将模块资源进行发布到线上，在搭建系统中进行页面组装和上下文注入预览后，完成页面级别发布。在底层同样借助 NodeJS 实现的源站系统来承载 CDN 内容的分发回源，提供亿万用户的访问预览。这套体系经过几代的发展更迭，已经成为当下前端研发的最核心场景之一。

大致从 15 年开始的几年时间里，伴随着技术的发展，淘系前端的研发体系从最初的本地工具、研发部署平台作为排头兵，逐步发展成线上线下的体系化专业解决方案。在保证前端角色的研发基础体验不断突破升级的同时，企业化规模化的前端工程建设将日常零散的研发形式不断推向更体系化的、高效率的、可扩展的 研发模式，同时也为后续更上层的研发部署模式的突破和升级打下坚实的基础。

未来时代

在逐步成熟的工程体系之上，在最近的几年时间内，我们也在思考面对 研发提效 这件事情上，是否还能有更突破的、面向未来的解法。随着许多内部产品不断地改造和迭代，也不断

浮现出了星星之火可以燎原的方向和模式。

D2C

imgcook 项目最初起源于可视化搭建，到目前外部看到的研发模式也是经历了不断的改进、变化。在整个的研发模式变化中，从外部的视角大致经历几个阶段

第一阶段：当时的用户的使用方式是通过页面元素的拖拽进行页面搭建，产出源代码之后进行发布上线。

第二阶段：通过建立图像扫描引擎进行像素扫描布局，同时接入 sketch 等视觉设计工具，自动的将设计稿进行转换，这个阶段前端用户的研发方式式就开始变化成了将视觉稿进行上传到平台处理进行代码转换。

第三阶段：在当下的第三个阶段，平台正通过深度学习等人工智能技术来提效图像代码转换的能力，与此同时在平台端完善基础研发链路的打通。内部的研发模式也正逐步切换到了在设计、代码的工作台中进行操作，实现一站式研发、上线的操作体验。

目前 D2C 能力逐步在智能化还原组件、表单、模块、页面的研发场景中逐步得到验证并收获一定的成果。

IDE

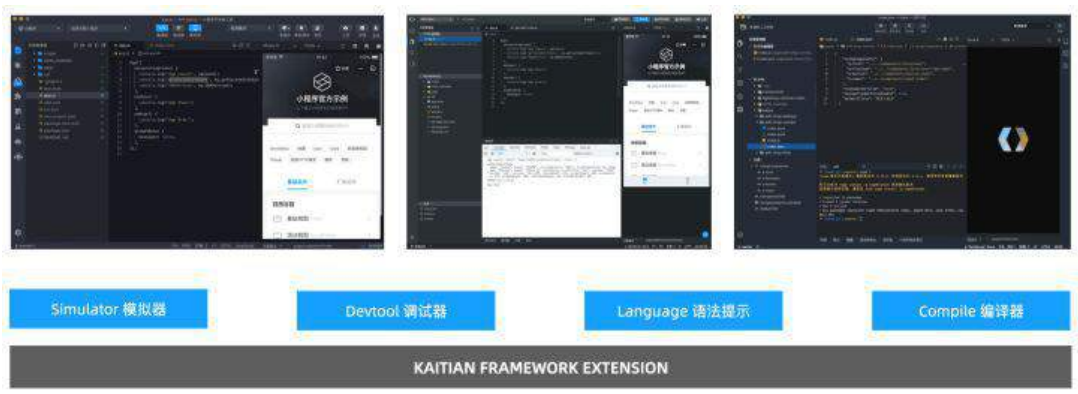
在最近的一年时间，我们也逐步投入到了新的 IDE 方向建设中，期望通过 IDE 的平台能力来孵化新的更高效的研发模式。

外部趋势

从外部视角来看，可以看到有两个趋势浮出水面。一个是 IDE 领域相关的创业公司逐渐浮现出来，出现了许多相关的创业明星公司，例如有 Eclipse 体系下的 theia ，通过兼容 VSCode 接口实现的 coder，以及研发编辑领域的行业新秀 codesandbox；另一个是云厂商的加入，AWS 收购 Cloud9、腾讯收购 Coding、Azure 提供 Codespace 服务。

在编辑器、Docker 等技术的不断发展成熟之下，各个 IDE 相关服务厂商都希望能借助集成研发环境的能力来找到研发效率、体验提升的机会，从而抓住用户的痛点，扩张产品市场。

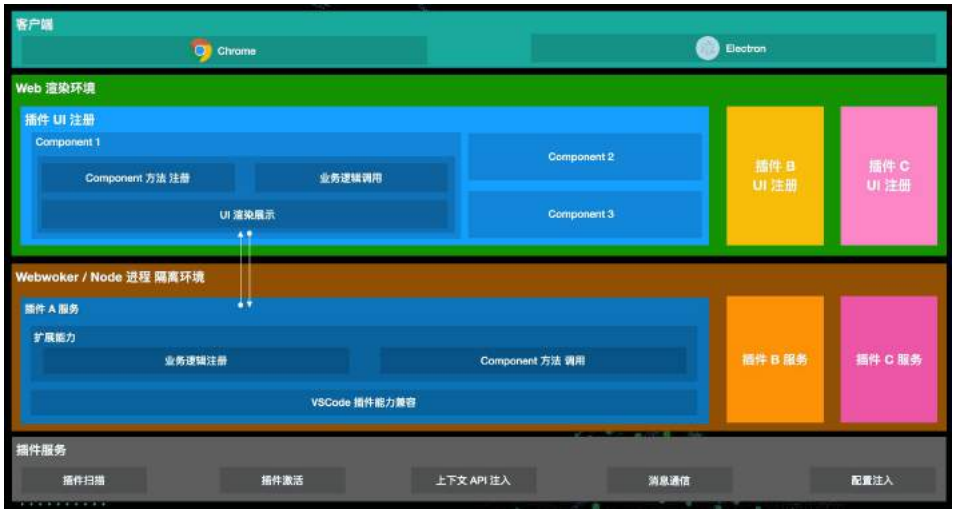
内部体系



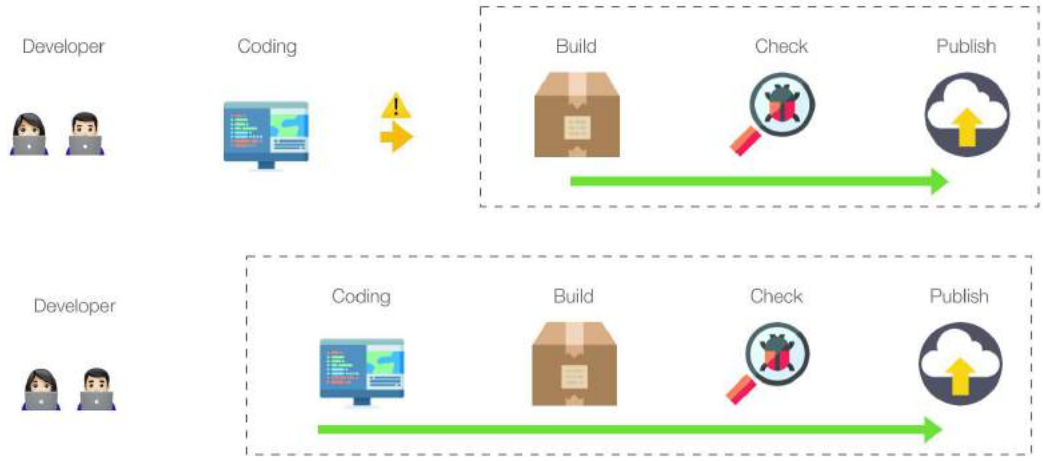
前端研发模式随着不断的分化、发展，涌现出越来越多的工具、服务。当下工具、服务的形式也不单单只是前文提到的命令行终端工具，或者说终端工具也逐渐成为了富交互的工具、

服务的启动入口。同时一种业务研发模式往往也需要串联到不同体系、不同团队提供的研发工具、服务。举一个例子，例如当下支付宝小程序的研发，除了基础的编译服务、预览服务之外，需要用到模拟器、调试器、真机调试等等服务。

支付宝小程序其实是内部很多场景的一个缩影，支付宝小程序当下的解法其实是借助 Electron 平台打造了一款本地的 IDE 研发工具。在这样的背景之下，体系内去年也开始了 IDE 底层体系 的搭建。期望能通过一套 IDE 底层实现线上、线下的基础 IDE 解决方案。



借助 IDE 底层插件机制，淘系也孵化出自己的 IDE 集成研发工具，例如前文提到的 D2C 、支付宝小程序、serverless 等场景都在紧锣密鼓的进行内部拓展和使用改进。我们通过 IDE 及 IDE 的插件体系，一方面借助 VSCode 的生态完善基本研发体验，另一方面通过插件在 VSCode 能力范围之上扩展的插件 UI 能力来串联起单个项目链路上所涉及到的所有的工具、服务。



与以往的流程的不同的是，除了将线上的发布部署流程进行串联整合，研发同学在当下所有的在研发过程的操作包括 研发编码、调试预览、部署流程 都在统一的研发 IDE 中有机的衔接起来，直接通过研发面板完成所有操作。这是我们为未来埋下的一颗种子，正在慢慢发芽。

整篇答案其实从一个在体系内的员工角度向大家介绍了关于淘系前端研发部署相关的一些历程，当下的研发模式我们也在日赴夜继的探索突破，最后介绍下团队做的事情。
我们是 淘系前端团队-工程中台团队 ，作为淘系前端及阿里集团前端的前端研发的基础设施，目前在做前端工程相关的体系建设，主要有几块事情:

1. 下一代前端研发模式: 通过自建阿里 IDE 核心底层 KAITIAN，借助底层两端一致、可视化插件体系等基础能力深度整合集成现有研发服务资产，搭建当下及未来研发模式，新的领域等你来架构。
2. 前端工程基础服务：基于 NodeJS、Docker 等能力建设前端研发持续集成流程，百万级别任务量等你来优化、挑战。
3. 大规模 CDN 源站: 基于 NodeJS 实现集团核心 CDN 回源核心服务，大促基础设施，亿级流量等你来"开飞机换引擎"。

如果对做的事情有兴趣或者对内容有想法的同学可以微信 tagzw1128 或者邮箱 shangpo.zw@alibaba-inc.com 欢迎交流。

Q: Nodejs 的 websocket 的服务器端是如何实现的?

作者 | 玄农
出品 | 淘系技术

网上一直找不到相关的资料，全都是使用的现成的框架。但是框架都太笨重了，代码量太大。我现在需要写一个 nodejs 的 websocket 的后端小框架，请问怎么开始呢?

最近正在研究 Websocket 相关的知识，想着如何能自己实现 Websocket 协议。到网上搜罗了一番资料后用 Node.js 实现该协议，倒也没有想象中那么复杂，除去注释语句和 console 语句后，大约 200 行代码左右。本文记录了实现过程中的经验和总结。

前言

如果你想要写一个 WebSocket 服务器，首先需要读懂对应的网络协议 RFC6455，不过这对于一般人来说有些“晦涩”，英文且不说，还得咬文嚼字理解 网络编程 含义。

好在 WebSocket 技术出现比较早，所以可以搜到 RFC6455 中文版，网上也有很多针对该协议的剖析文章，很多文章里还有现成的实现代码可以参考，所以说实现一个简单的 Websocket 服务并非难事。

本文更偏向实战（in action），会从知识储备、具体代码分析以及注意事项角度去讲解如何用 Node.js 实现一个简单的 Websocket 服务，至于 Websocket 概念、定义、解释和用途等基础知识不会涉及，因为这些知识在本文所列的参考文章中轻松找到。（也可以自行网上随便一搜，就能找到很多）

知识储备

如果要自己写一个 Websocket 服务，主要有两个难点：

1. 熟练掌握 Websocket 的协议，这个需要多读现有的解读类文章；（下面会给出参考文章）。
2. 操作二进制数据流，在 Node.js 中需要对 Buffer 这个类稍微熟悉些。

同时还需要具备两个基础知识点：

1. 网络编程中使用 大端次序（Big endian）表示大于一字节的数据，称之为 网络字节序（不晓得大小端的，推荐阅读 什么是大小端？）。
2. 了解最高有效位（MSB， Most Significant Bit），不太清楚的，可以参考 LSB 最低有效位和 MSB 最高有效位。

具体的做法如下，推荐先阅读以下几篇参考文章：

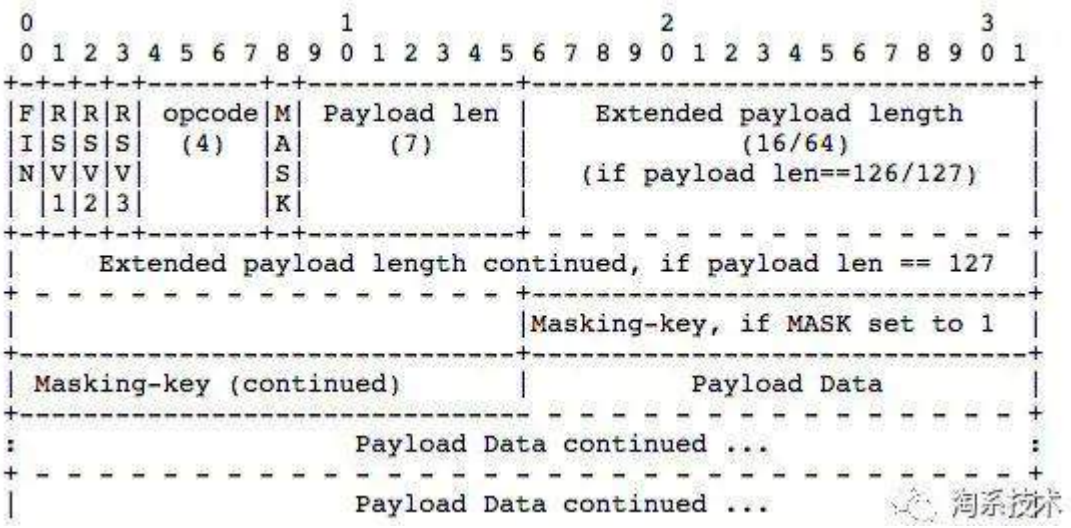
- 学习 WebSocket 协议—从顶层到底层的实现原理（修订版）：作者本身自己就用
- Node.js 实现过一遍，知识点讲解挺透彻的，适合前端同学优先阅读。
- WebSocket 详解（一）：初步认识 WebSocket 技术：是一系列的文章，从浅入深，配有丰富的图文。
- WebSocket：5 分钟从入门到精通：全文以 Q&A 的方式组织而成，协议的要点都解读到了，除此之外还很全面，涉及了 WebSocket 如何建立连接、交换数据的细节、数据帧的格式以及网络安全等。
- MDN - Writing WebSocket servers：MDN 官方教程，读一遍没啥坏处。

然后开始写代码，在实现过程中的大部分代码可以从下面 3 篇文章中找到并借鉴（copy）：

- nodejs 实现：简化版本的从这儿借鉴过来的。
- 学习 WebSocket 协议—从顶层到底层的实现原理（修订版）。
- WebSocket 协议解析：虽然是 C++ 写的，但不影响代码逻辑的理解。

阅读完上面的文章，你会有发现一个共同点，就是在实现 WebSockets 过程中，最最核心的部分就是 解析 或者 生成 Frame（帧），就是下面这结构：

分就是 解析 或者 生成 Frame（帧），就是下面这结构：



帧结构标准

截图来自规范 Base Framing Protocol

想要理解 frame 各个字段的含义，可参考 WebSocket 详解（三）：深入 WebSocket 通信协议细节，文中作者绘制了一副图来解释这个 frame 结构；

而在代码层面，frame 的解析或生成可以在 RocketEngine - parser 或者 _processBuffer 中找到。

在完成上面几个方面的知识储备之后，而且大多有现成的代码，所以自己边抄边写一个 Websocket 服务器并不算太难。

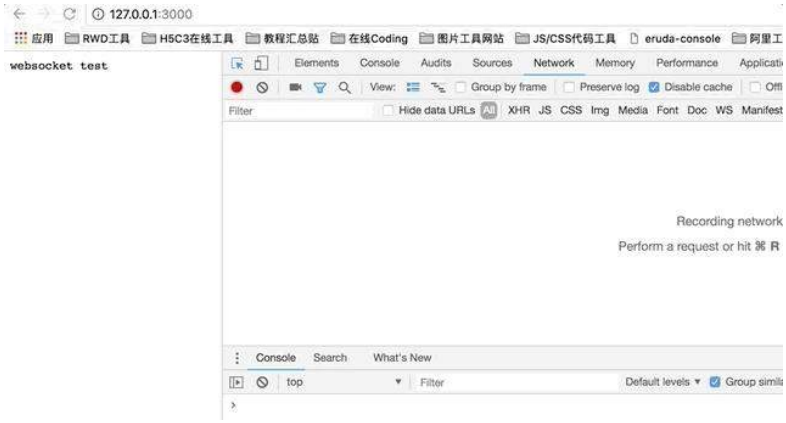
对于 Websocket 初学者，请务必阅读以上参考文章，对 Websocket 协议有大概的了解之后再继续本文剩下部分的阅读，否则很有可能会觉得我写得云里雾里，不知所云。

实战

实现代码放在自己的 demos 仓库的 micro-ws 的目录了，git clone 后本地运行，执行:

```
node index.js
```

将会在 http://127.0.0.1:3000 创建服务。运行服务之后，打开控制台就能看到效果:



动图中浏览器 console 所执行的 js 代码步骤如下:

先建立连接:

```
var ws = new WebSocket("ws://127.0.0.1:3000");
ws.onmessage = function(evt) {
  console.log( "Received Message: " + evt.data);
};
```

然后发送消息: （注意一定要在建立连接之后再执行该语句，否则发不出消息的）

```
ws.send('hello world');
```

从效果可见，我们已经实现 Websocket 最基本的通讯功能了。接下来我们详细看一下具体实现的细节。

调用所写的 Websocket 类

站在使用者的角度，假设我们已经完成 Websocket 类了，那么应该怎么使用？客户端通过 HTTP Upgrade 请求，即 101 Switching Protocol 到 HTTP 服务器，然后由服务器进行协议转换。

在 Node.js 中我们通过 http.createServer 获取 http.server 实例，然后监听 upgrade 事件，在处理这个事件：

```
// HTTP 服务器部分
var server = http.createServer(function(req, res) {
  res.end('websocket test\r\n');
});

// Upgrade 请求处理
server.on('upgrade', function(req, socket, upgradeHead){
  // 初始化 ws
  var ws = new WebSocket(req, socket, upgradeHead);

  // ... ws 监听 data、error 的逻辑等

});
```

这里监听 upgrade 事件的回调函数中第二个参数 socket 是 net.Socket 实例，这个类是 TCP 或 UNIX Socket 的抽象，同时一个 net.Socket 也是一个 duplex stream，所以它能被读或写，并且它也是一个 EventEmitter。

我们就利用这个 socket 对象上进行 Websocket 类实例的初始化工作；

构造函数

```
class WebSocket extends EventEmitter {
  constructor(req, socket, upgradeHead){
    super(); // 调用 EventEmitter 构造函数

    // 1. 构造响应头 resHeaders 部分

    // 2. 监听 socket 的 data 事件，以及 error 事件

    // 3. 初始化成员属性
  }
}
```

注意，我们需要继承内置的 EventEmitter ，这样生成的实例才能监听、绑定事件；

Node.js 采用事件驱动、异步编程，天生就是为了网络服务而设计的，继承 EventEmitter 就能享受到非阻塞模式的 IO 处理；

讲一下其中 响应头的构造 和 事件监听 部分。

返回响应头（Response Header）

根据协议规范，我们能写出响应头的内容：

1. 将 Sec-WebSocket-Key 跟 258EAFA5-E914-47DA-95CA-C5AB0DC85B11 拼接。
2. 通过 SHA1 计算出摘要，并转成 base64 字符串。

具体代码如下：

```
var resKey = hashWebSocketKey(req.headers['sec-websocket-key']);

    // 构造响应头
    var resHeaders = [
```

```
      'HTTP/1.1 101 Switching Protocols',
      'Upgrade: websocket',
      'Connection: Upgrade',
      'Sec-WebSocket-Accept: ' + resKey
    ]
      .concat('', '')
      .join('\r\n');
    socket.write(resHeaders);
```

当执行 socket.write(resHeaders); 到后就和客户端建立起 WebSocket 连接了，剩下去就是数据的处理。

监听事件

socket 就是 TCP 协议的抽象，直接在上面监听已有的 data 事件和 close 事件这两个事件。

还有其他事件，比如 error、end 等，详细参考 net.Socket 文档

```
socket.on('data', data => {
      this.buffer = Buffer.concat([this.buffer, data]);
      while (this._processBuffer()) {} // 循环处理返回的 data 数据
    });

    socket.on('close', had_error => {
      if (!this.closed) {
        this.emit('close', 1006);
        this.closed = true;
      }
    });
```

close 的事件逻辑比较简单，比较重要的是 data 的事件监听部分。核心就是 this._processBuffer() 这个方法，用于处理客户端传送过来的数据（即 Frame 数据）

注意该方法是放在 while 循环语句里，处理好边界情况，防止死循环。

Frame 帧数据的处理

WebSocket 客户端、服务端通信的最小单位是帧（frame），由 1 个或多个帧组成一条完整的消息（message）。

这 this._processBuffer() 部分代码逻辑就是用来解析帧数据的，所以它是实现 Websocket 代码的关键；（该方法里面用到了大量的位操作符以及 Buffer 类的操作）

帧数据结构详细定义可参考 RFC6455 5.2 节，上面罗列的参考文章都有详细的解读，我在这儿也不啰嗦讲细节了，直接看代码比听我用文字讲要好。

这里就其中两个细节需要铺垫一下，方便更好地理解代码。

操作码（Opcode）

Opcode 即 操作代码，Opcode 的值决定了应该如何解析后续的数据载荷（data payload）根据 Opcode 我们可以大致将数据帧分成两大类：数据帧 和 控制帧。

数据帧：目前只有 3 种，对应的 opcode 是：

- 0x0：数据延续帧
- 0x1：utf-8 文本
- 0x2：二进制数据；
- 0x3 - 0x7：目前保留，用于后续定义的非控制帧

控制帧：除了上述 3 种数据帧之外，剩下的都是控制帧

- 0x8：表示连接断开

- 0x9：表示 ping 操作
- 0xA：表示 pong 操作
- 0xB - 0xF：目前保留，用于后续定义的控制帧

在代码里，我们会先从帧数据中提取操作码：

```
var opcode = byte1 & 0x0f; //截取第一个字节的后 4 位，即 opcode 码
```

然后根据协议获取到真正的数据载荷（data payload），然后将这两部分传给 _handleFrame 方法：

```
this._handleFrame(opcode, payload); // 处理操作码
```

该方法会根据不同的 opcode 做出不同的操作：

```
_handleFrame(opcode, buffer) {
    var payload;
    switch (opcode) {
      case OPCODES.TEXT:
        payload = buffer.toString('utf8'); //如果是文本需要转化为 utf8 的编码
        this.emit('data', opcode, payload); //Buffer.toString()默认 utf8 这里是故意指示的
        break;
      case OPCODES.BINARY: //二进制文件直接交付
        payload = buffer;
        this.emit('data', opcode, payload);
        break;
      case OPCODES.PING: // 发送 pong 做响应
        this._doSend(OPCODES.PONG, buffer);
        break;
      case OPCODES.PONG: //不做处理
        console.log('server receive pong');
        break;
      case OPCODES.CLOSE: // close 有很多关闭码
```

```
        let code, reason; // 用于获取关闭码和关闭原因
        if (buffer.length >= 2) {
          code = buffer.readUInt16BE(0);
          reason = buffer.toString('utf8', 2);
        }
        this.close(code, reason);
        this.emit('close', code, reason);
        break;
      default:
        this.close(1002, 'unhandle opcode:' + opcode);
    }
  }
```

分片（Fragment）

规范文档：5.4 - Fragmentation

一旦 WebSocket 客户端、服务端建立连接后，后续的操作都是基于数据帧的传递。理论上来说，每个帧（Frame）的大小是没有限制的。

对于大块的数据，Websocket 协议建议对数据进行分片（Fragment）操作。分片的意义主要是两方面：

- 主要目的是允许当消息开始但不必缓冲该消息时发送一个未知大小的消息。如果消息不能被分片，那么端点将不得不缓冲整个消息以便在首字节发生之前统计出它的长度。对于分片，服务器或中间件可以选择一个合适大小的缓冲，当缓冲满时，再写一个片段到网络。
- 另一方面分片传输也能更高效地利用多路复用提高带宽利用率，一个逻辑通道上的一个大消息独占输出通道是不可取的，因此多路复用需要可以分割消息为更小的分段来更好的共享输出通道。参考文档 I/O 多路复用技术（multiplexing）是什么？

WebSocket 协议提供的分片方法，是将原本一个大的帧拆分成数个小的帧。下面是把一个

大的 Frame 分片的图示：

分片编号	0	1	...	n-2	n-1
FIN	0	0	...	0	1
opcode	!0	0	0	0	0

分片图示根据 FIN 的值来判断，是否已经收到消息的最后一个数据帧。由图可知，第一个分片的 FIN 为 0，Opcode 为非 0 值（0x1 或 0x2），最后一个分片的 FIN 为 1，Opcode 为 0。中间分片的 FIN 和 opcode 二者均为 0。

- `FIN=1` 表示当前数据帧为消息的最后一个数据帧，此时接收方已经收到完整的消息，可以对消息进行处理。
- `FIN=0`，则接收方还需要继续监听接收其余的数据帧。
- opcode 在数据交换的场景下，表示的是数据的类型。
- `0x01` 表示文本，永远是 `utf8` 编码的。
- `0x02` 表示二进制。
- 而 `0x00` 比较特殊，表示 延续帧（continuation frame），顾名思义，就是完整消息对应的数据帧还没接收完。

代码里，我们需要检测 FIN 的值，如果为 0 说明有分片，需要记录第一个 FIN 为 0 时的 opcode 值，缓存到 this.frameOpcode 属性中，将载荷缓存到 this.frames 属性中：

```
var FIN = byte1 & 0x80; // 如果为 0x80，则标志传输结束，获取高位 bit
    // 如果是 0 的话，说明是延续帧，需要保存好 opCode
    if (!FIN) {
      this.frameOpcode = opcode || this.frameOpcode; // 确保不为 0;
    }

    //....
    // 有可能是分帧，需要拼接数据
    this.frames = Buffer.concat([this.frames, payload]); // 保存到 frames 中
```

当接收到最后一个 FIN 帧的时候，就可以组装后给 _handleFrame 方法：

```
if (FIN) {
        payload = this.frames.slice(0); // 获取所有拼接完整的数据
        opcode = opcode || this.frameOpcode; // 如果是 0 ，则保持获取之前保存的 code
        this.frames = Buffer.alloc(0); // 清空 frames
        this.frameOpcode = 0; // 清空 opcode
        this._handleFrame(opcode, payload); // 处理操作码
    }
```

发送数据帧

上面讲的都是接收并解析来自客户端的数据帧，当我们想给客户端发送数据帧的时候，也得按协议来。

这部分操作相当于是上述 _processBuffer 方法的逆向操作，在代码里我们使用 encodeMessage 方法（为了简单起见，我们发送给客户端的数据没有经过掩码处理）将发送的数据分装成数据帧的格式，然后调用 socket.write 方法发送给客户端；

```
_doSend(opcode, payload) {
    // 1. 考虑数据分片
    this.socket.write(
      encodeMessage(count > 0 ? OPCODES.CONTINUE : opcode, payload)
    ); //编码后直接通过 socket 发送
```

为了考虑分片场景，特意设置 MAX_FRAME_SIZE 来对每次发送的数据长度做截断做分片：

```
// ...
    var len = Buffer.byteLength(payload);
    // 分片的距离逻辑
    var count = 0;
    // 这里可以针对 payload 的长度做分片
    while (len > MAX_FRAME_SIZE) {
```

```
      var framePayload = payload.slice(0, MAX_FRAME_SIZE);
      payload = payload.slice(MAX_FRAME_SIZE);
      this.socket.write(
        encodeMessage(
          count > 0 ? OPCODES.CONTINUE : opcode,
          framePayload,
          false
        )
      ); //编码后直接通过 socket 发送
      count++;
      len = Buffer.byteLength(payload);
    }
  // ...
```

至此已经实现 Websocket 协议的关键部分，所组装起来的代码就能和客户端建立 Websocket 连接并进行数据交互了。

Q&A

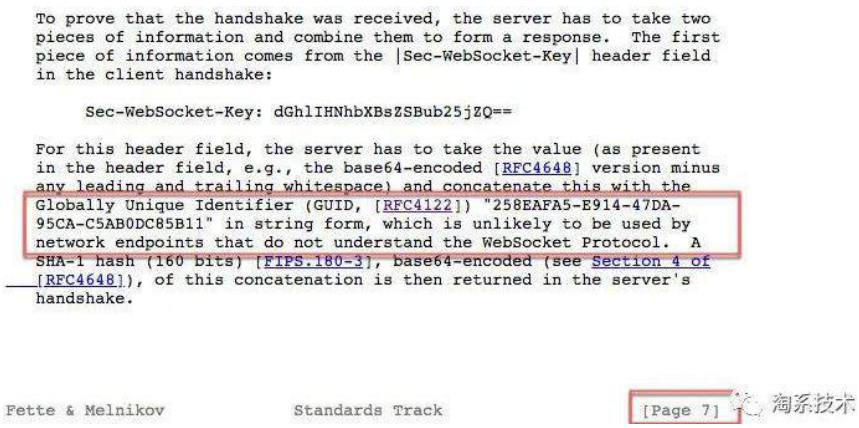
字符串“258EAFA5-E914-47DA-95CA-C5AB0DC85B11”怎么来的?

这个标志性字符串是专门标示 Websocket 协议的 UUID；UUID 是长度为 16-byte（128-bit）的 ID，一般以形如 f81d4fae-7dec-11d0-a765-00a0c91e6bf6 的字符串作为 URN（Uniform Resource Name，统一资源名称）

UUID 可以移步到 UUID 原理 和 RFC 4122 获取更多知识

为啥选择这个字符串?

在规范的第七页已经有明确的说明了：



To prove that the handshake was received, the server has to take two pieces of information and combine them to form a response. The first piece of information comes from the |Sec-WebSocket-Key| header field in the client handshake:

Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==

For this header field, the server has to take the value (as present in the header field, e.g., the base64-encoded [RFC4648] version minus any leading and trailing whitespace) and concatenate this with the Globally Unique Identifier (GUID, [RFC4122]) "258EAFA5-E914-47DA-95CA-C5AB0DC85B11" in string form, which is unlikely to be used by network endpoints that do not understand the WebSocket Protocol. A SHA-1 hash (160 bits) [FIPS.180-3], base64-encoded (see Section 4 of [RFC4648]), of this concatenation is then returned in the server's handshake.

Fette & Melnikov Standards Track [Page 7]

淘系技术

专用于 websocket 的 uuid 标识符

之所以选用这个 UUID ，主要该 ID 极大不太可能被其他不了解 Websocket 协议的网络终端所使用。

我也不晓得该怎么翻译。。。总之就说这个 ID 就相当于 Websocket 协议的“身份证号”了。

Websocket 和 HTTP 什么关系?

HTTP、WebSocket 等应用层协议，都是基于 TCP 协议来传输数据的，我们可以把这些高级协议理解成对 TCP 的封装。

既然大家都使用 TCP 协议，那么大家的连接和断开，都要遵循 TCP 协议中的三次握手和四次握手 ，只是在连接之后发送的内容不同，或者是断开的时间不同。

对于 WebSocket 来说，它必须依赖 HTTP 协议进行一次握手 ，握手成功后，数据就直接从 TCP 通道传输，与 HTTP 无关了。

浏览器中 Websocket 会自动分片么?

答案是：看具体浏览器的实现。

WebSocket 是一个 message based 的协议，它可以自动将数据分片，并且自动将分片的数据组装。

每个 message 可以是一个或多个分片。message 不记录长度，分片才记录长度；

根据协议 websocket 协议中帧长度上限为 2^63 byte（为 8388608 TB），可以认为没有限制，很明显按协议的最大上限来传输数据是不靠谱的。所以在实际使用中 websocket 消息长度限制取决于具体的实现。关于哲方面，找了两篇参考文章：

- Websocket 需要像 TCP Socket 那样进行逻辑数据包的分包与合包吗?：WebSocket 是一个 message-based 的协议，它可以自动将数据分片，并且自动将分片的数据组装；
- websocket 长文本问题？：这里给出了长文本 ws 传输实践总结。

在文章 WebSocket 探秘 中，作者就做了一个实验，作者发送 27378 个字节，结果被迫分包了；如果是大数据量，就会被 socket 自动分包发送。

而经过我本人试验，发现 Chrome 浏览器（版本 68.0.3440.106 - 64bit）会针对 131072（=2^17）bytes 大小进行自动分包。我是通过以下测试代码验证：

```
var ws = new WebSocket("ws://127.0.0.1:3000");
ws.onmessage = function(evt) {
  console.log( "Received Message: " + evt.data);
};
var myArray = new ArrayBuffer(131072 * 2 + 1);
ws.send(myArray);
```

服务端日志：

```
server detect fragment, sizeof payload: 131072
server detect fragment, sizeof payload: 131072
receive data: 2 262145
```

客户端日志：

```
Received Message: good job
```

截图如下：

chrome 浏览器会自动分片

而以同样的方式去测试一些自己机器上的浏览器：

- Firefox（62.0，64bit）
- safari（11.1.2 - 13605.3.8）
- IE 11

这些客户端上的 Websocket 几乎没有大小的分片（随着数据量增大，发送会减缓，但并没有发现分片现象）。

总结

从刚开始决定阅读 Websocket 协议，到自己使用 Node.js 实现一套简单的 Websocket 协议，到这篇文章的产出，前后耗费大约 1 个月时间（拖延症。。。）。

感谢文中所提及的参考文献所给予的帮助，让我实现过程中事半功倍。

之所以能够使用较少的代码实现 Websocket，是因为 Node.js 体系本身了很好的基础，比

如其所提供的 EventEmitter 类自带事件循环，http 模块让你直接使用封装好的 socket 对象，我们只要按照 Websocket 协议实现 Frame（帧）的解析和组装即可。

在使用 Node.js 实现一遍 Websocket 协议后，就能较为深刻地理解以下知识点（理解起来一切都是那么自然而然）：

- Websocket 是一种应用层协议，是为了提供 Web 应用程序和服务端全双工通信而专门制定的。
- WebSocket 和 HTTP 都是基于 TCP 协议实现的。
- WebSocket 和 HTTP 的唯一关联就是 HTTP 服务器需要发送一个 "Upgrade" 请求，即 101 Switching Protocol 到 HTTP 服务器，然后由服务器进行协议转换。
- WebSocket 使用 HTTP 来建立连接，但是定义了一系列新的 header 域，这些域在 HTTP 中并不会使用。
- WebSocket 可以和 HTTP Server 共享同一 port。
- WebSocket 的 数据帧有序。
- ...

本文仅仅是协议的简单实现，对于 Websocket 的其实还有很多事情可以做（比如支持 命名空间、流式 API 等），有兴趣的可以参考业界流行的 Websocket 仓库，去练习锻造一个健壮的 Websocket 工具库轮子：

- socketio/http://socket.io：43.5k star，不多说，业界权威龙头老大。（不过这实际上不是一个 WebSocket 库，而是个实时 pub/sub 框架。简单地说，http://Socket.IO 只是包含 WebSocket 功能的一个框架，如果要使用该库作为 server 端的服务，则 client 也必须使用该库，因为它不是标准的 WebSocket 协议，而是基于 WebSocket 再包装的消息通信协议）。
- websockets/ws：9k star，强大易用的 websocket 服务端、客户端实现，还有提供很多强大的特性。
- uNetworking/uWebSockets：9.5k star，小巧高性能的 websocket 实现，C++ 写的，想更多了解 Websocket 的底层实现，该库是不错的案例。

- theturtle32/WebSocket-Node：2.3k star，大部分使用 JavaScript，性能关键部分使用 C++ node-gyp 实现的库。其所列的 测试用例 有挺好的参考价值。

链接：

https://tools.ietf.org/html/rfc6455?spm=ata.13261165.0.0.b404272cuT4Xuz
https://wenku.baidu.com/view/2dc93465844769eae009ed86.html?spm=ata.13261165.0.0.b404272cuT4Xuz
http://nodejs.cn/api/buffer.html?spm=ata.13261165.0.0.b404272cuT4Xuz
https://www.bysocket.com/?spm=ata.13261165.0.0.b404272cuT4Xuz&p=615
https://blog.csdn.net/qq_29350001/article/details/51177404?spm=ata.13261165.0.0.b404272cuT4Xuz
https://github.com/abbshr/abbshr.github.io/issues/22?spm=ata.13261165.0.0.b404272cuT4Xuz
https://github.com/abbshr/abbshr.github.io/issues/22?spm=ata.13261165.0.0.b404272cuT4Xuz
https://segmentfault.com/a/1190000012709475?spm=ata.13261165.0.0.b404272cuT4Xuz
https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_servers?spm=ata.13261165.0.0.b404272cuT4Xuz
https://blog.csdn.net/Newpidian/article/details/50850670?spm=ata.13261165.0.0.b404272cuT4Xuz
https://github.com/abbshr/abbshr.github.io/issues/22?spm=ata.13261165.0.0.b404272cuT4Xuz
https://tools.ietf.org/html/rfc6455?spm=ata.13261165.0.0.b404272cuT4Xuz#section-5.2
http://www.52im.net/thread-332-1-1.html?spm=ata.13261165.0.0.b404272cuT4Xuz
https://github.com/abbshr/RocketEngine/blob/master/lib/wsframe/parser.js?spm=ata.13261165.0.0.b404272cuT4Xuz&file=parser.js
http://gitlab.alibaba-inc.com/lastep/micro-ws?spm=ata.13261165.0.0.b404272cuT4Xuz
http://nodejs.cn/api/http.html?spm=ata.13261165.0.0.b404272cuT4Xuz#http_http_createserver_options_requestlistener

http://nodejs.cn/api/http.html?spm=ata.13261165.0.0.b404272cuT4Xuz#http_event_upgrade

http://nodejs.cn/api/net.html?spm=ata.13261165.0.0.b404272cuT4Xuz#net_class_net_socket

http://nodejs.cn/

http://nodejs.cn/api/net.html?spm=ata.13261165.0.0.b404272cuT4Xuz#net_class_net_socket

http://nodejs.cn/api/buffer.html?spm=ata.13261165.0.0.b404272cuT4Xuz

https://tools.ietf.org/html/rfc6455?spm=ata.13261165.0.0.b404272cuT4Xuz#section-5.2

https://tools.ietf.org/html/rfc6455?spm=ata.13261165.0.0.b404272cuT4Xuz#section-5.4

https://www.zhihu.com/question/28594409?spm=ata.13261165.0.0.b404272cuT4Xuz

https://blog.csdn.net/ScrappyShiyu/article/details/79936897?spm=ata.13261165.0.0.b404272cuT4Xuz

https://tools.ietf.org/html/rfc4122?spm=ata.13261165.0.0.b404272cuT4Xuz

https://www.zhihu.com/question/29916578?spm=ata.13261165.0.0.b404272cuT4Xuz

https://segmentfault.com/q/1010000010620489?spm=ata.13261165.0.0.b404272cuT4Xuz

https://segmentfault.com/a/1190000012319848?spm=ata.13261165.0.0.b404272cuT4Xuz#articleHeader8

https://github.com/socketio/socket.io?spm=ata.13261165.0.0.b404272cuT4Xuz&file=socket.io

https://github.com/websockets/ws?spm=ata.13261165.0.0.b404272cuT4Xuz

https://github.com/uNetworking/uWebSockets?spm=ata.13261165.0.0.b404272cuT4Xuz

http://theturtle32.github.io/WebSocket-Node/test-report/servers/?spm=ata.13261165.0.0.b404272cuT4Xuz

Q: 网页布局都有哪种？一般都用什么布局？

作者 | 大漠
出品 | 淘系技术

> 像优酷、知乎这样的网站在 pc 端用什么布局？在移动端又用什么布局？

超干货长文预警。这次把网页布局方案讲得通通透透的，等高布局，水平垂直居中，经典的圣杯布局等等全都有了。

随着 Web 技术不断的革新，CSS 近几年也变得更强大。

在 Web 开发中，CSS 是不可或缺的一部分，对于很多 Web 开发者来说，有很多 CSS 属性不知道，或者说他们知道，但忘记在最恰当的时候使用最适合的 CSS 属性。

而且时至今日，其中有一些 CSS 的属性可以让开发者能节约更多的时间。比如说，在 Web 布局中，现代 CSS 特性就可以更好的帮助我们快速实现，例如等高布局，水平垂直居中，经典的圣杯布局、宽高比例、页脚保持在底部等。

在本文中，我将会介绍一些不同的 CSS 属性来实现这些效果，希望大家会感兴趣。更希望对大家今后的工作有所帮助。

水平垂直居中

如何实现水平垂直居中 可以说是 CSS 面试题中的经典面试题，在多年前这个面试题给很多同学都带来了困惑，但 Flexbxo 布局模块 和 CSS Grid 布局模块 的到来，可以说实现水平垂直居中已是非常的容易。

Flexbox 中实现水平垂直居中

在 Flexbox 布局模块中，不管是单行还是多行，要让它们在容器中水平垂直居中都是件易事，而且方法也有多种。最常见的是在 Flex 容器上设置对齐方式，在 Flex 项目上设置 margin: auto。

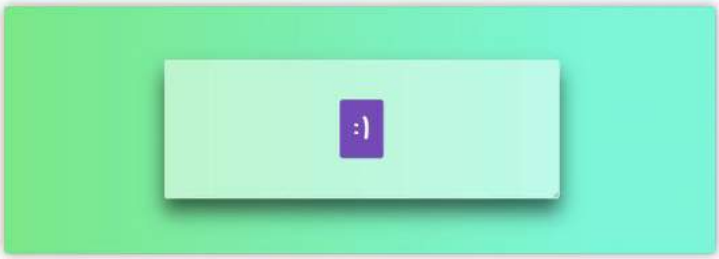
先来看在 Flex 容器上设置对齐方式。

Flex 容器和 Flex 项目上设置对齐方式

你可能已经知道在 Flex 容器上设置 justify-content、align-items 的值为 center 时，可以让元素在 Flex 容器中达到水平垂直居中的效果。来看一个示例：

```
<!-- HTML -->
<div class="flex__container">
    <div class="flex__item"></div>
</div>

/* CSS */
.flex__container {
    display: flex;
    justify-content: center;
    align-items: center;
}
```

效果如下：

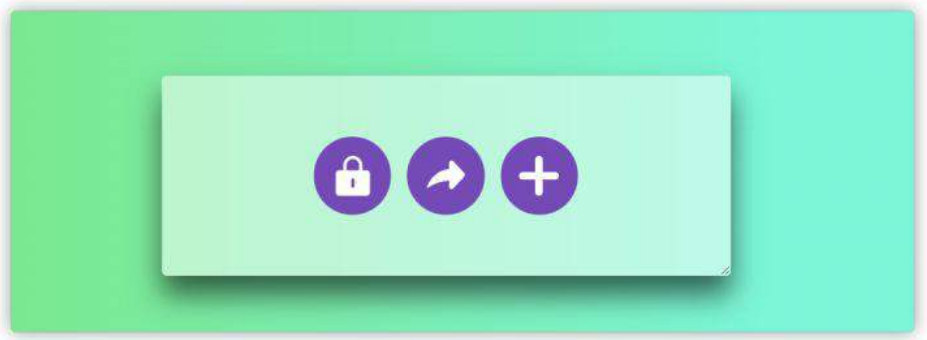


Demo(https://codepen.io/airen/embed/YzwYRRy)

这种方式特别适应于让 Icon 图标在容器中水平垂直居中，不同的是在 Icon 图标容器上显示设置 display: inline-flex。比如下面这个示例：

```
<!-- HTML -->
<div class="flex__container">
    <svg> </svg>
</div>

/* CSS */
.flex__container {
    display: inline-flex;
    align-items: center;
    justify-content: center;
}
```

效果如下：

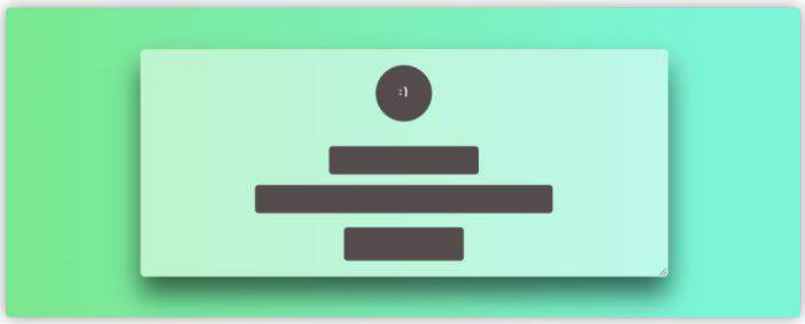


Demo(https://codepen.io/airen/embed/xxZpQNv)

在这种模式之下，如果要让多个元素实现水平垂直居中的效果，那还需要加上 flex-direction: column，比如：

```
<!-- HTML -->
<div class="flex__container">
    <div class="avatar">:)</div>
    <div class="media__heading"></div>
    <div class="media__content"></div>
    <div class="action"></div>
</div>

/* CSS */
.flex__container {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}
```

效果如下：



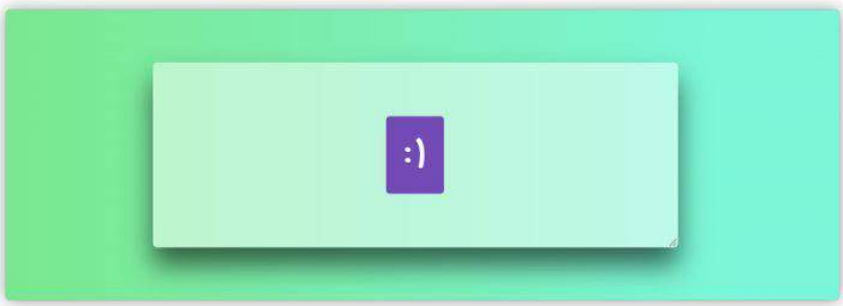
Demo(https://codepen.io/airen/embed/QWyazpZ)

在 Flexbox 布局中，还可以像下面这样让 Flex 项目在 Flex 容器中达到水平垂直居中的效果：

```
<!-- HTML -->
<div class="flex__container">
    <div class="flex__item"></div>
</div>
```

```
/* CSS */
.flex__container {
    display: flex; // 或 inline-flex
    justify-content: center;
}

.flex__item {
    align-self: center;
}
```

效果如下：

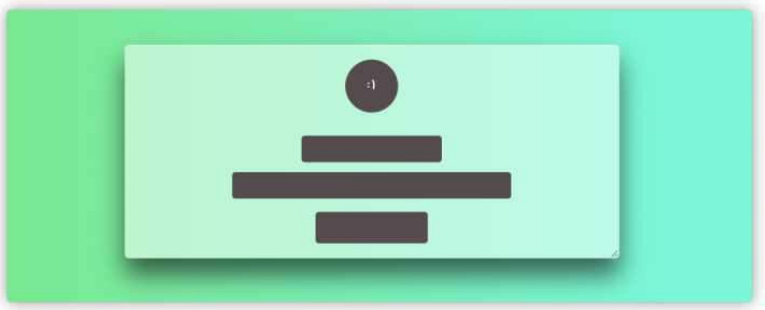


Demo(https://codepen.io/airen/embed/yLepGKW)

如果在 Flex 容器中有多个 Flex 项目时，该方法同样有效：

```
.flex__container {
    display: flex; // 或 inline-flex
    justify-content: center;
}

.flex__container > * {
    align-self: center;
}
```

比如下面这个效果：

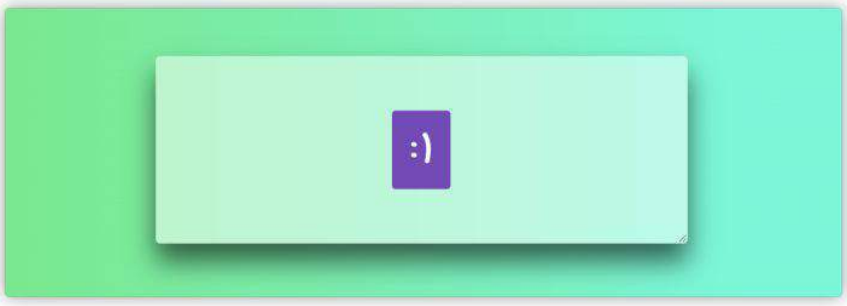


Demo(https://codepen.io/airen/embed/bGEaOjm)

除此之外，还可以使用 `place-content: center` 让 Flex 项目实现水平垂直居中：

```
.flex__container {
    display: flex;
    place-content: center;
}

.flex__item {
    align-self: center;
}
```

效果如下：

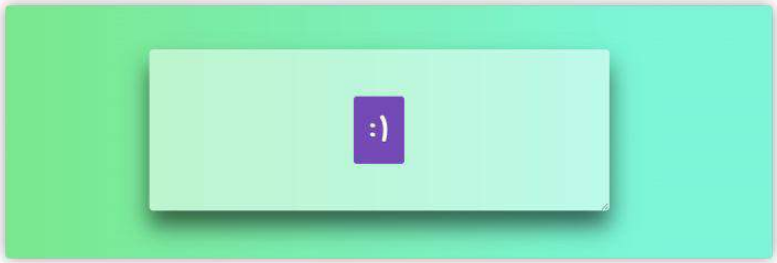


Demo(https://codepen.io/airen/embed/gOPoZQz)

或者换：

```
.flex__container {
    display: flex;
    place-content: center;
    place-items: center;
}
```

效果如下：



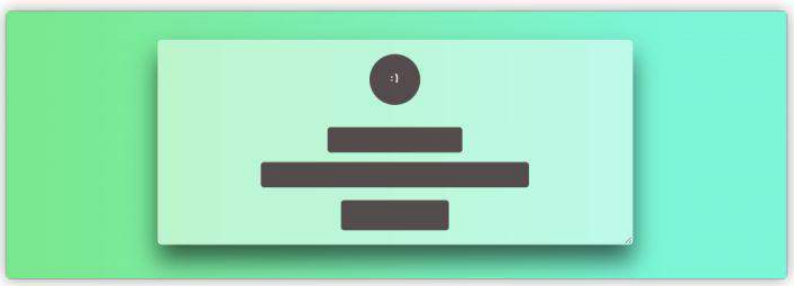
Demo(https://codepen.io/airen/embed/JjGMwzE)

这两种方式同样适用于 Flex 容器中有多个 Flex 项目的情景：

```
.flex__container {
    display: flex;
    flex-direction: column;
    place-content: center;
}

.flex__container > * {
    align-self: center;
}

// 或

.flex__container {
```

```
    display: flex;
    flex-direction: column;
    place-content: center;
    place-items: center;
}
```

效果如下：



Demo(https://codepen.io/airen/embed/XWXVoLd)

可能很多同学对于 place-content 和 place-items 会感到陌生。其实 place-content 是 align-content 和 justify-content 的简写属性；而 place-items 是 align-items 和 justify-items 的简写属性。即：

```
.flex__container {
    place-content: center;
    place-items: center;
}
等效于：
.flex__container {
    align-content: center;
    justify-content: center;

    align-items: center;
    justify-items: center;
}
```

虽然扩展出来有四个属性，但最终等效于：

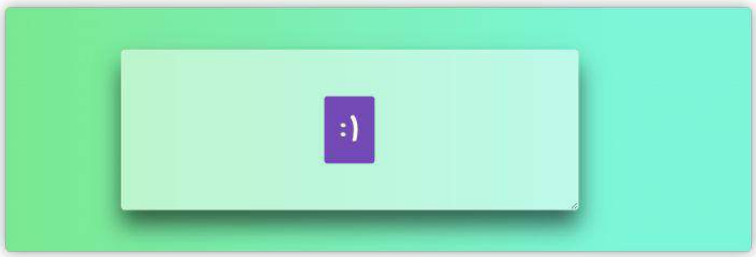
```
.flex__container {
    display: flex;
    align-items: center;
    justify-content: center;
}

// 多行
.flex__container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
}
```

在 Flex 项目上设置`margin: auto`

如果在 Flex 容器中只有一个 Flex 项目，还可以显式在 Flex 项目中显式设置 `margin` 的值为 `auto`，这样也可以让 Flex 项目在 Flex 容器中水平垂直居中。例如：

```
.flex__container {
    display: flex; // 或 inline-flex
}

.flex__item {
    margin: auto;
}
```

效果如下：



Demo(https://codepen.io/airen/embed/KKVZJNp)

整个过程，你可以通过下面这个示例来体验。尝试着选中不同方向的 margin 值：

Demo(https://codepen.io/airen/embed/gOPoqRq)

Grid 中实现水平垂直居中

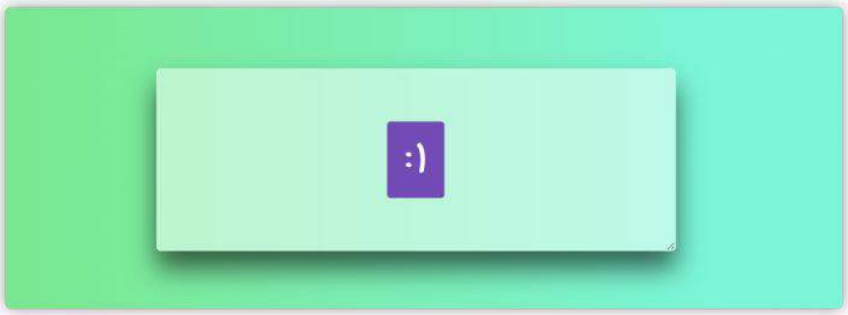
CSS Grid 布局可以说是现代 Web 布局中的银弹。它也是到目前为止布局系统中唯一一个二维布局系统。

在 CSS Grid 布局中，只需要仅仅的几行代码也可以快速的帮助我们实现水平垂直居中的效果。比如下面这个示例：

```
<!-- HTML -->
<div class="grid__container">
    <div class="grid__item"></div>
</div>
```

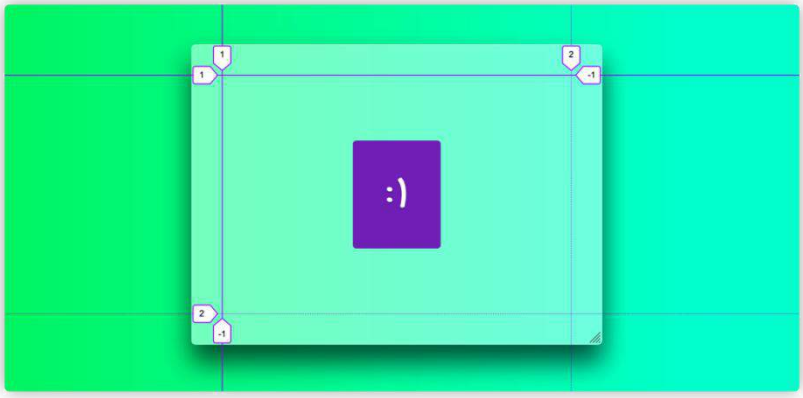
```
/* CSS */
.grid {
    display: grid; // 或 inline-grid
    place-items: center
}
```

效果如下：

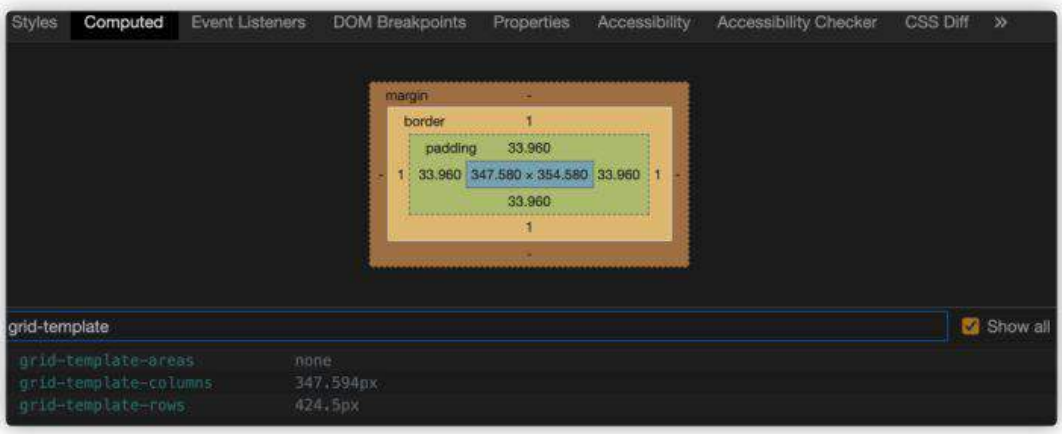


Demo(https://codepen.io/airen/embed/zYrRYxW)

在 CSS Grid 布局模块中，只要显式设置了 display: grid（或 inline-grid）就会创建 Grid 容器和 Grid 项目，也会自动生成网格线，即行和列（默认为一行一列）。



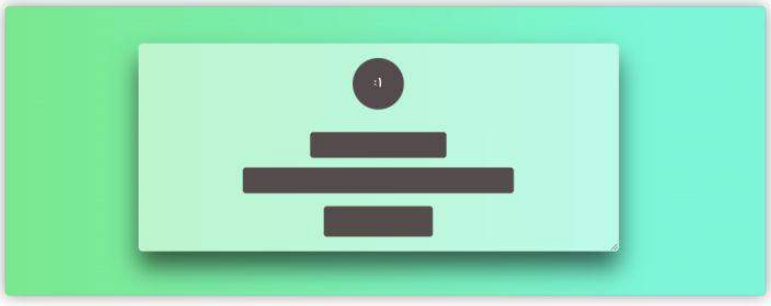
在没有显式地在 Grid 容器上设置 grid-template-columns 和 grid-template-rows，浏览器会将 Grid 容器默认设置为 Grid 内容大小：



这种方法也适用于 CSS Grid 容器中有多个子元素（Grid 项目），比如：

```
<!-- HTML -->
<div class="grid__container">
    <div class="avatar">:)</div>
    <div class="media__heading"></div>
    <div class="media__content"></div>
    <div class="action"></div>
</div>
```

这个时候你看到的效果如下：

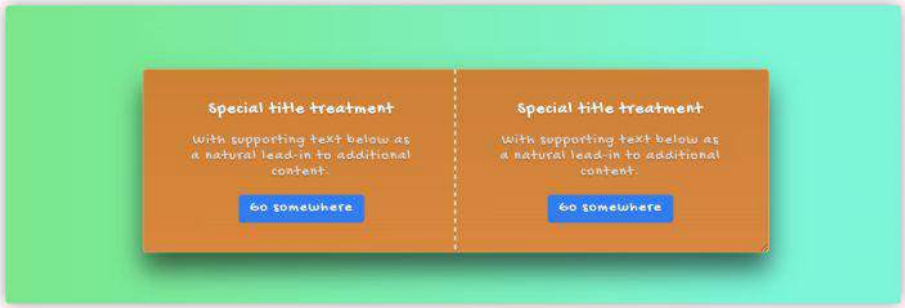


Demo(https://codepen.io/airen/embed/PoZQoGP)

而且 palce-items 适用于每个单元格。这意味着它将居中单元格的内容。比如下面这个示例：

```
<!-- HTML -->
<div class="grid__container">
    <div class="grid__item">
        <h3>Special title treatment</h3>
        <p>With supporting text below as a natural lead-in to additional content.</p>
        <div class="action">Go somewhere</div>
    </div>
</div>

/* CSS */
.grid__container {
    display: grid;
    place-items: center;
    grid-template-columns: repeat(2, 1fr);
    gap: 2vh;
}


.grid__item {
    display: grid;
    place-items: center;
}
```

效果如下：



Demo(https://codepen.io/airen/embed/mdVXdpe)

等高布局

等高布局也是 Web 中非常常见的一种布局方式，而且实现等高布局的方案也有很多种。这里我们主要来看 Flexbox 布局模块和 Grid 布局模块给我们带来了什么样的变化。

在 Flexbox 和 Grid 布局模块中，让我们实现等高布局已经是非常的简单了，比如：

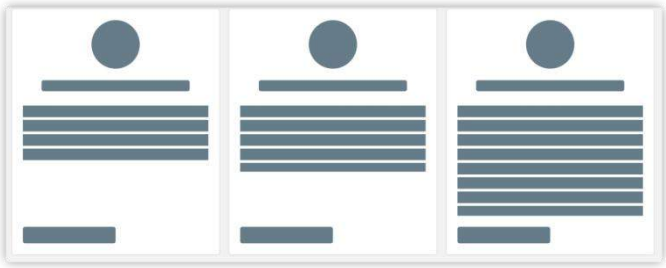
```
<!-- Flexbox -->
<flex__container>
    <flex__item></flex__item>
    <flex__item></flex__item>
    <flex__item></flex__item>
</flex__container>

/* CSS */
.flex__container {
    display: flex; // 或 inline-flex
}
```

简单地说，在容器上显式设置了 display 的值为 flex 或 inline-flex，该容器的所有子元素的高度都相等，因为容器的 align-items 的默认值为 stretch。

这个时候你看到的效果如下：

Demo(https://codepen.io/airen/embed/NWxyOQq)

这种方式特别适用于卡片组件中：



Demo(https://codepen.io/airen/embed/OJMQoOO)

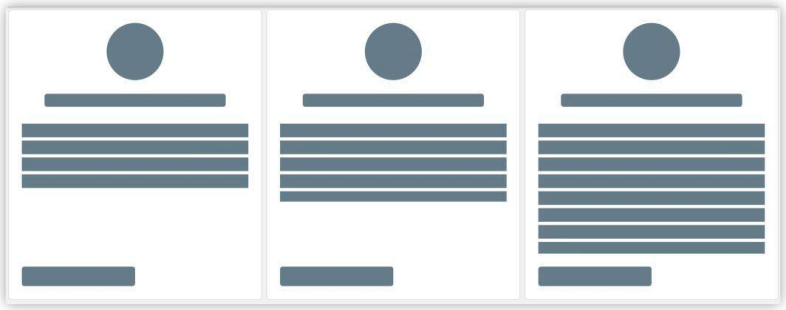
在 Grid 布局模块中类似：

```
<!-- HTML -->
<grid__container>
    <grid__item></grid__item>
    <grid__item></grid__item>
    <grid__item></grid__item>
</grid__container>

/* CSS */
.grid__container {
    display: grid;
    grid-template-columns: 20vw 1fr 20vw; /* 根据需求调整值*/
}
```

效果如下：

Demo(https://codepen.io/airen/embed/mdVXQON)

同样在一些卡片类布局中运用：



Demo(https://codepen.io/airen/embed/MWKQzmE)

如果需求有所调整，比如在 Flex 项目 或 Grid 项目的子元素高度和容器高度相同。

```
<!-- HTML -->
<flex__container>
    <flex__item>
        <content></content>
    </flex__item>
</flex__container>

/* CSS */
.flex__container {
    display: flex;
}

.content {
    height: 100%
}

// 或
.grid__container {
```

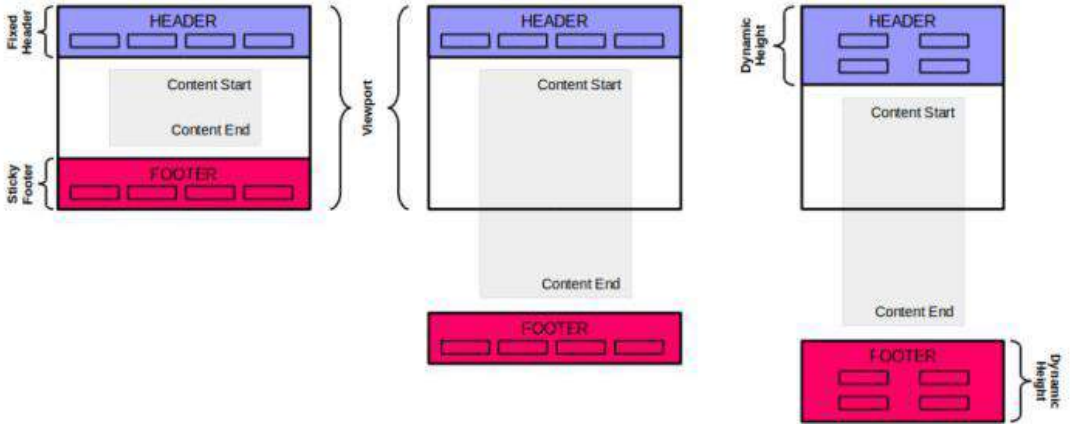
```
    display: grid;
    grid-auto-flow: column;
}

.content {
    height: 100%;
}
```

效果如下：

Demo(https://codepen.io/airen/embed/jOWZdbo)

Sticky Footer

首先用下图来描述什么是 Sticky Footer 布局效果：



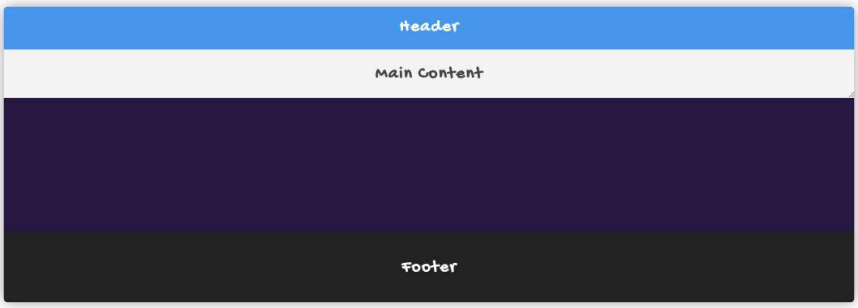
Sticky Footer 实现方案和等高、垂直居中一样，同样有很多种方案可以实现(//http://css-tricks.com/couple-takes-sticky-footer/)。

比如像下面这样的结构：

```
<!-- HTML -->
<header></header>
<main></main>
<footer></footer>
```

先来看 Flexbox 布局模块中的实现方案：

```
body {
    display: flex;
    flex-direction: column;
}

footer {
    margin-top: auto;
}
```



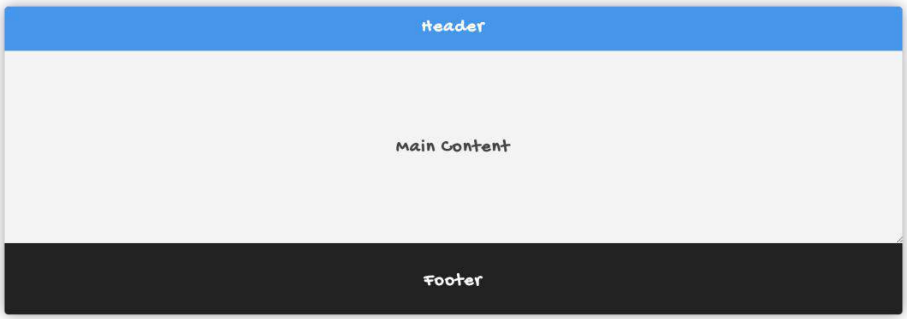
Demo(https://codepen.io/airen/embed/bGELzYy)

可以尝试着在 main 区域右下角向下拖动，改变主内容区域的高度，你会发现“当内容不足一屏时，<footer>会在页面的最底部，当内容超出一屏时，<footer>会自动往后延后”。

在 Flexbox 布局中，还可以在<main>区域上设置下面的样式，达到相等的效果：

```
body {
    display: flex;
    flex-direction: column;
}

main {
    flex: 1 0 auto;
}
```

效果如下：



Demo(https://codepen.io/airen/embed/zYrRXmY)

<main>中的 flex: 1 0 auto 相当于是：

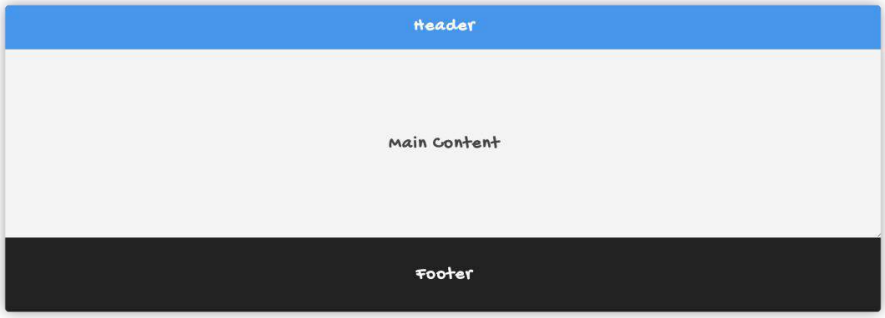
```
main {
    flex-grow: 1; /*容器有剩余空间时，main 区域会扩展*/
    flex-shrink: 0; /*容器有不足空间时，main 区域不会收缩*/
    flex-basis: auto; /*main 区域高度的基准值为 main 内容自动高度*/
}
```

如果你想省事的话，可以在 main 上显式设置 flex-grow:1，因为 flex-shrink 和 flex-basis 的默认值为 1 和 auto。

在 CSS Grid 布局中我们可以借助 1fr 让<main>区域根据 Grid 容器剩余空间来做计算。

```
.grid__container {
    display: grid;
    grid-template-rows: auto 1fr auto;
}
```

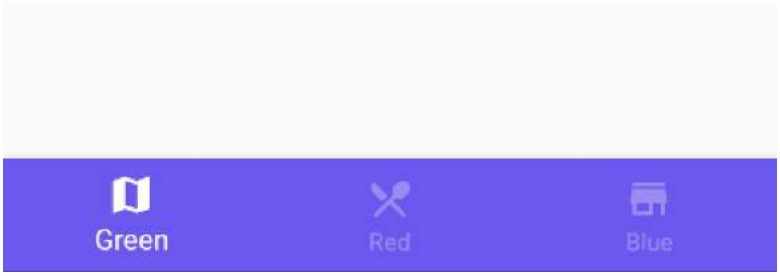
效果如下：



Demo(https://codepen.io/airen/embed/MWKQRxd)

均分列

在 Web 布局中，很多时候会对列做均分布局，最为常见的就是在移动端的底部 Bar，比如下图这样的一个效果：



在 Flexbox 和 Grid 还没出现之前，如果希望真正的做到均分效果，可以用 100%（或 100vw）除以具体的列数。比如：

```
<!-- HTML -->
<container>
    <column></column>
    <column></column>
    <column></column>
</container>

/* CCSS */
.container {
    inline-size: 50vw;
    min-inline-size: 320px;
    display: flex-row;
}

.column {
    float: left;
    width: calc(100% / 3);
}
```

效果如下：

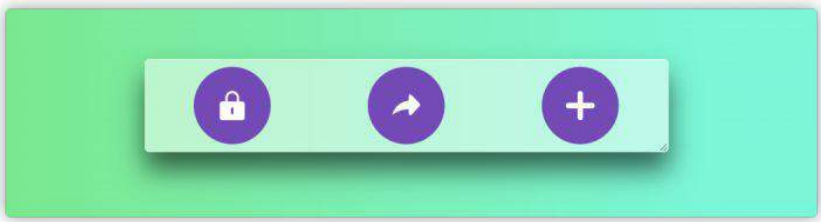
Demo(https://codepen.io/airen/embed/LYGQoxL)

通过浏览器调试器中可以发现，现个列的宽度都是相等的：

在 Flexbox 和 Grid 布局中，实现上面的效果会变得更容易地多。先来看 Flexbox 中的布局：

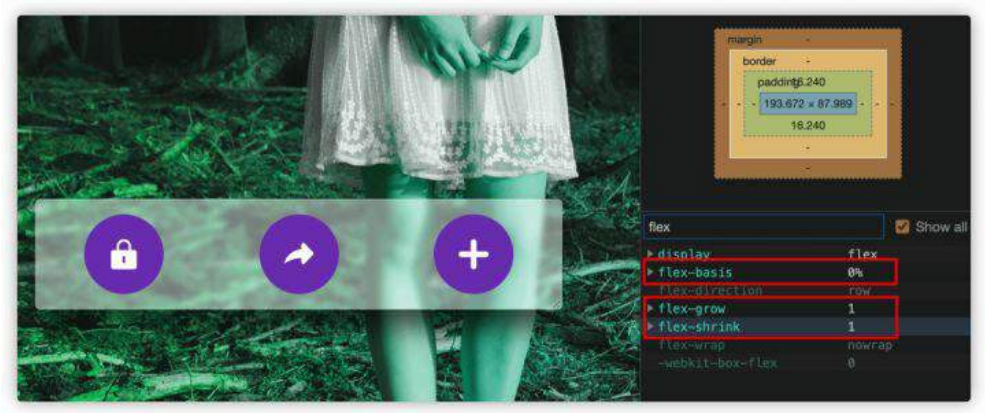
```
<!-- HTML -->
<flex__container>
    <flex__item></flex__item>
    <flex__item></flex__item>
    <flex__item></flex__item>
</flex__container>

/* CSS */
.flex__container {
    inline-size: 50vw;
    display: flex;
}

.flex__item {
    flex: 1;
}
```

效果如下：



Demo(https://codepen.io/airen/embed/yLevWEe)

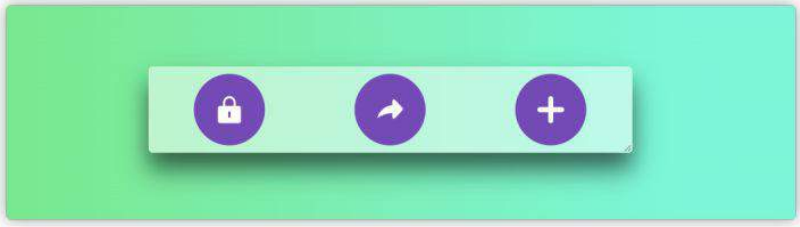
在 Flexbox 布局模块中，当 flex 取的值是一个单值（无单位的数），比如示例中的 flex:1，它会当作显式的设置了 flex-grow: 1。浏览器计算出来的 flex：



接下来看 Grid 中如何实现上例的效果：

```
<!-- HTML -->
<grid__container>
    <grid__item></grid__item>
    <grid__item></grid__item>
    <grid__item></grid__item>
</grid__container>

/* CSS */
.grid__container {
    display: grid;
    grid-template-columns: repeat(3, 1fr); /*这里的 3 表示具体的列数*/
}
```

最终的效果是相同的：



Demo(https://codepen.io/airen/embed/NWxyVQP)

这样的布局方式也适用于其他的布局中。但不管是 Flexbox 还是 Grid 布局中，都存在一定的缺陷，当容器没有足够的空间容纳 Flex 项目（或 Grid 项目）时，Flex 项目或 Grid 项目会溢出（或隐藏，如果 Flex 容器或 Grid 容器显式设置了 overflow:hidden）：

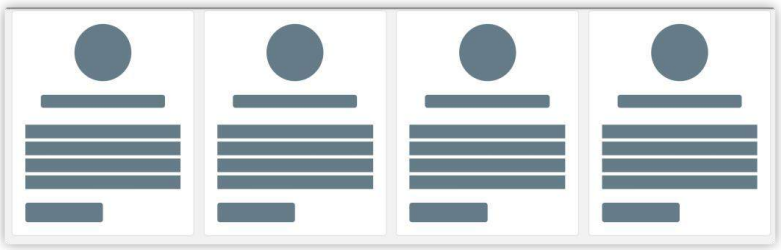
修复这种现象最简单的方式是在 Flex 容器或 Grid 容器显式设置一个 min-width（或 min-inline-size）：

```
.flex__container {
    min-inline-size: 300px;
}
```

不过话又说回来，比如我们的 Flex 项目（或 Grid 项目）是一个卡片，每张卡片宽度是相等之外，更希望容器没有足够空间时，Flex 项目（或 Grid 项目）会自动断行排列。

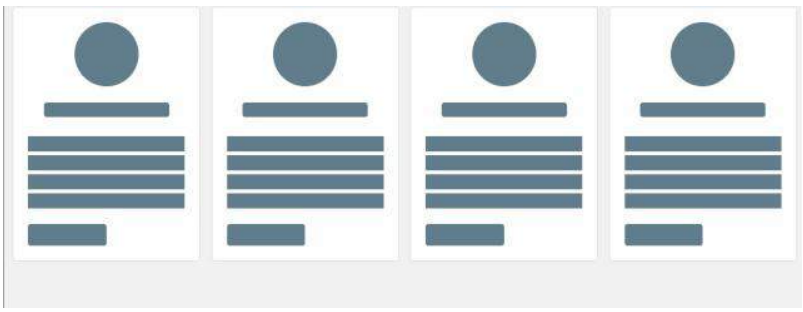
我们继续通过示例向大家展示。先来看 Flexbox 实现方案：

```
.flex__container {
    display: flex;
    flex-wrap: wrap;
}

.flex__item {
    flex: 0 1 calc((100vw - 18vh) / 4); /* calc(100vw -18vh) / 4 是 flex-basis 的基准值 */
}
```



Demo(https://codepen.io/airen/embed/dyGdBpw)

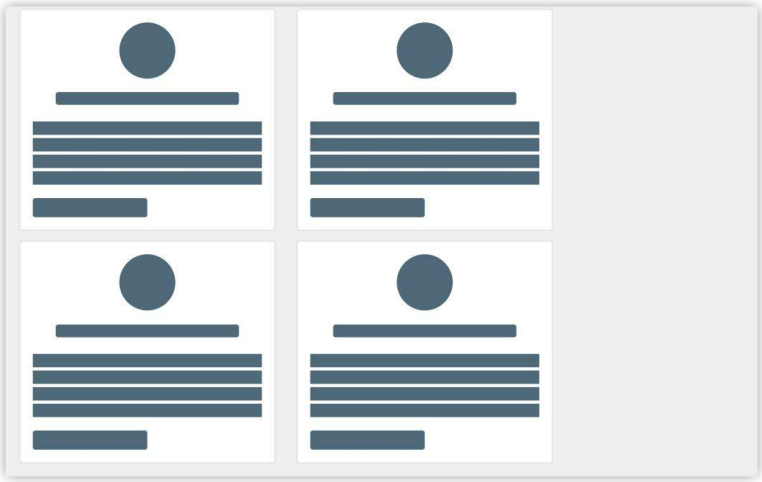
你可以尝试着调整浏览器的视窗宽度，当浏览器的视窗越来越小时，Flex 容器宽度也就会越来越小，当 Flex 容器小到没有足够的空间容纳四个 Flex 项目（就此例而言），那么 Flex 项目就会断行排列：



基于该例，如果把 Flex 项目的 `flex` 值改成：

```
.flex__item {
    flex: 0 0 400px;
}
```

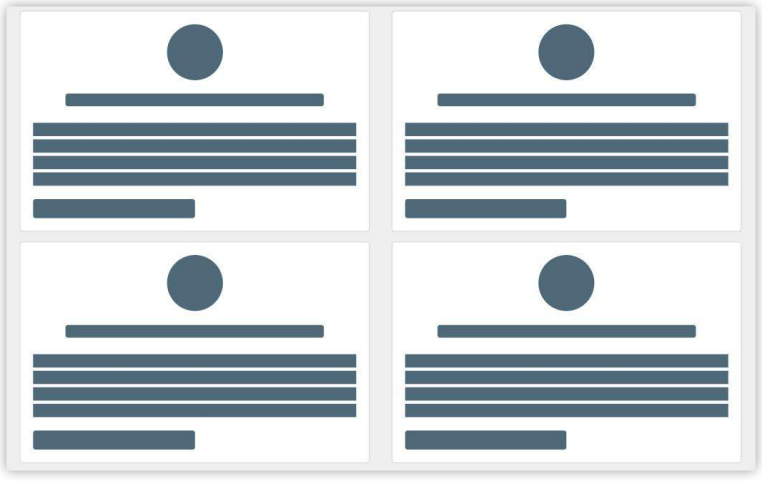
这个时候，当 Flex 容器没有足够空间时，Flex 项目会按 `flex-basis: 400px` 计算其宽度，Flex 容器没有足够空间时，Flex 就会断行：



反过来，如果 Flex 项目的值 `flex` 改成：

```
.flex__item {
    flex: 1 0 400px;
}
```

当 Flex 容器没有足够空间排列 Flex 项目时，Flex 项目会按 `flex-basis: 400px` 计算其宽度，Flex 会断行，并且同一行出现剩余空间时，Flex 项目会扩展，占满整个 Flex 容器：

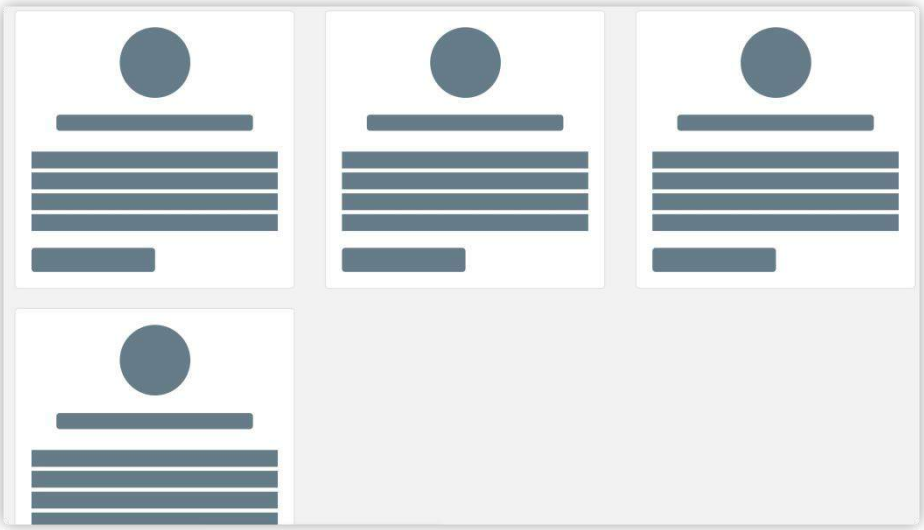


在 Grid 中实现类似的效果要更复杂一点。可以使用 repeat()函数，1fr 以及 auto-fit 等特性：

.

```
grid__container {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
    gap: 2vh;
}
```

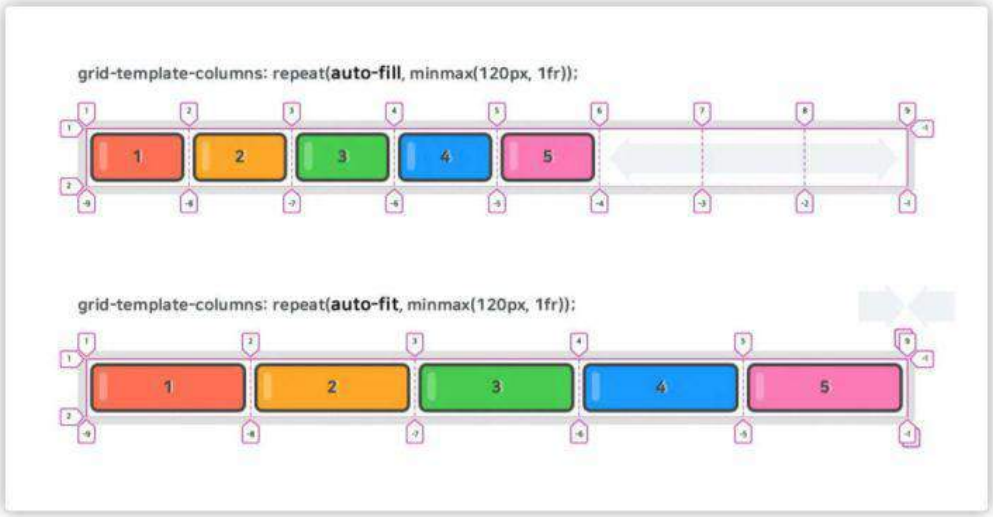
效果如下：



Demo(https://codepen.io/airen/embed/RwrQzeN)

如果你对这方面知识感兴趣的话，还可移步阅读《Container Query Solutions with CSS Grid and Flexbox(//moderncss.dev/container-query-solutions-with-css-grid-and-flexbox/)》一文。

其实在 Grid 中与 auto-fit 对比的值还有一个叫 auto-fill。但两者的差异是非常地大，用下图来描述 auto-fit 和 auto-fill 的差异：



另外这种方式也是到目前为止一种不需要借助 CSS 媒体查询就可以实现响应式布局效果。

圣杯布局

圣杯布局（Holy Grail Layout）(//http://en.wikipedia.org/wiki/Holygrail(web_design))是 Web 中典型的布局模式(//http://alistapart.com/article/holygrail/)。看上去像下图这样：

对于圣杯布局而言，HTML 结构是有一定的要求，那就是内容为先：

```
<!-- HTML -->
<header></header>
<main>
    <article></article> <!-- 主内容 -->
    <nav></nav>
    <aside></aside>
</main>
<footer></footer>
```

在这里主要还是和大家一起探讨，如何使用 Flexbox 和 Grid 布局模块来实现圣杯布局。先来看 Flexbox 实现方案：

```
body {
    width: 100vw;
    display: flex;
    flex-direction: column;
}

main {
    flex: 1;
    min-height: 0;

    display: flex;
    align-items: stretch;
    width: 100%;
}

footer {
    margin-top: auto;
}

nav {
    width: 220px;
    order: -1;
}
```

```
article {
    flex: 1;
}

aside {
    width: 220px;
}
```

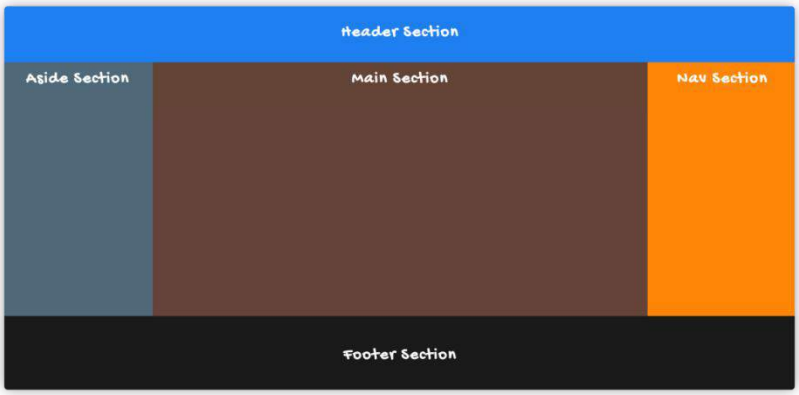
效果如下：

Demo(https://codepen.io/airen/embed/rNxJXYb)

通过在 nav、aside 和 article 上显式设置 order 的值，可以很好的控制这三个区域的布局顺序。比如说，希望<aside>在<article>之前排列，只需要在上面的示例基础上做一点点调整：

```
nav {
    order: 0;
}

aside {
    order: -1;
}
```

效果如下：



注意，order 的默认值为 0，值越大越排在后面！

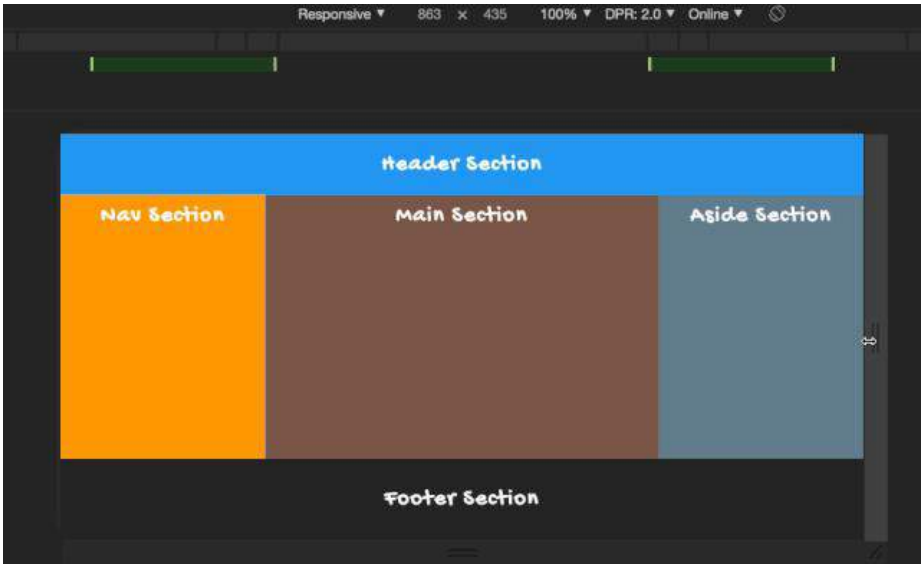
在上例的基础上，借助 CSS 媒体对象的特性，可以很容易实现响应式的圣杯布局效果：

```
@media screen and (max-width: 800px) {
    main {
        flex-direction: column;
    }

    nav, aside {
        width: 100%;
    }
}
```

效果如下：

Demo(https://codepen.io/airen/embed/gOPvVZX)

尝试着拖动浏览器来改变视窗大小，你可以看到如下图的效果：



在 Grid 布局模块中，实现圣杯布局要比 Flexbox 布局模块中更容易，而且更灵活。在 CSS Grid 布局模块中，HTML 结构可以更简洁：

```
<!-- HTML -->
<body>
    <header></header>
    <main></main>
    <nav></nav>
    <aside></aside>
    <footer></footer>
</body>
```

在 CSS 方面有很多种方案可以实现圣杯布局效果。我们先来看第一种：

```
body {
    display: grid;
    grid-template: auto 1fr auto / 220px 1fr 220px;
}
```

```
header {
    grid-column: 1 / 4;
}

main {
    grid-column: 2 / 3;
    grid-row: 2 / 3;
}

nav {
    grid-column: 1 / 2;
    grid-row: 2 / 3;
}

aside {
    grid-column: 3 / 4;
    grid-row: 2 / 3;
}

footer {
    grid-column: 1 / 4;
}
```

效果如下：

Demo(https://codepen.io/airen/embed/PoZRYPa)

上面示例采用的是网格线来给每个区域进行定位的：

和 Flexbox 布局类似，在媒体查询中可以改变每个网格区域的位置：

```
@media screen and (max-width: 800px) {
    body {
        grid-template-rows: auto;
        grid-template-columns: auto;
    }

    header,
    main,
    nav,
    aside,
    footer {
        grid-column: 1 / 2;
        min-height: auto;
    }

    main {
```

```
        grid-row: 3 / 4;
        margin: 0;
    }

    nav {
        grid-row: 2 / 3;
    }

    aside {
        grid-row: 4 / 5;
    }

    footer {
        grid-row: 5 / 6;
    }
}
```

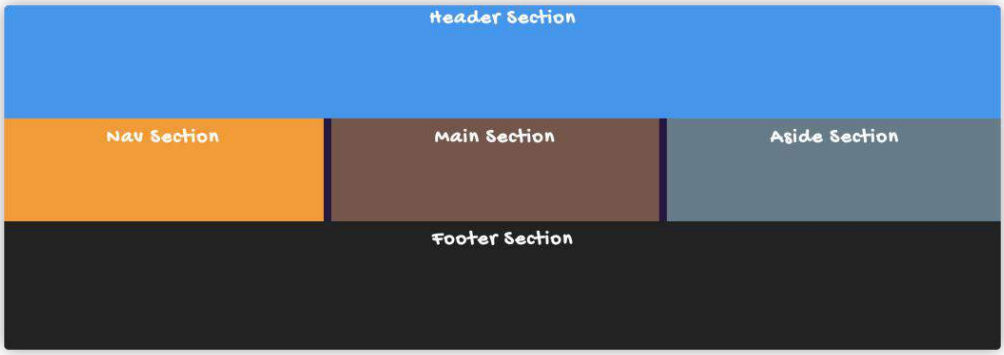
Demo(https://codepen.io/airen/embed/vYLRBaa)

除了 grid-template（即 grid-template-columns 和 grid-template-rows）之外，在 Grid 布局中还可以使用 grid-area 和 grid-template-areas 属性的结合，也能很方便的实现 CSS 圣杯布局。基于上面的示例上，只需要把你的 CSS 调整为：

```
body {
    display: grid;
    grid-template-areas:
```

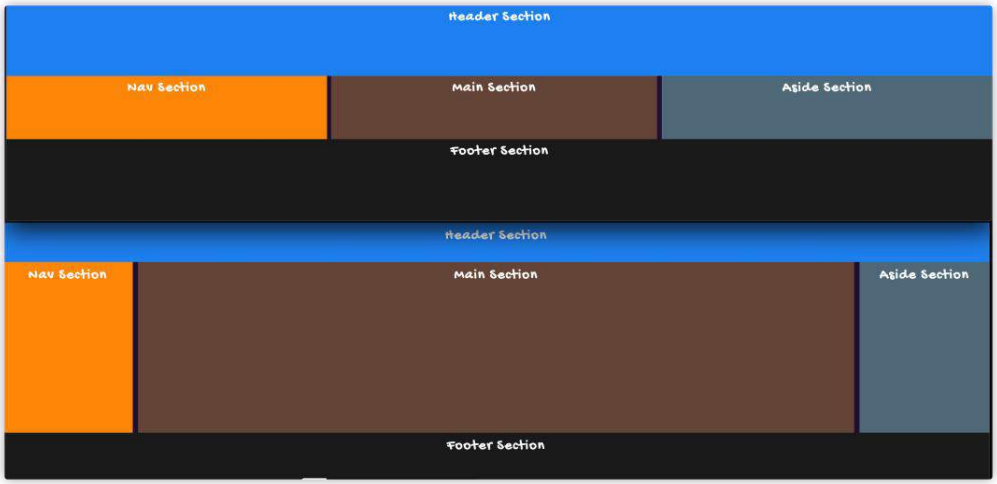
```
            "header header header"
            "nav main aside"
            "footer footer footer";
}

header {
    grid-area: header;
}

main {
    grid-area: main;
}

nav {
    grid-area: nav;
}

aside {
    grid-area: aside;
}

footer {
    grid-area: footer;
}

@media screen and (max-width: 800px) {
    body {
        grid-template-areas:
            "header"
            "nav"
            "main"
            "aside"
            "footer";
    }
}
```

效果如下：

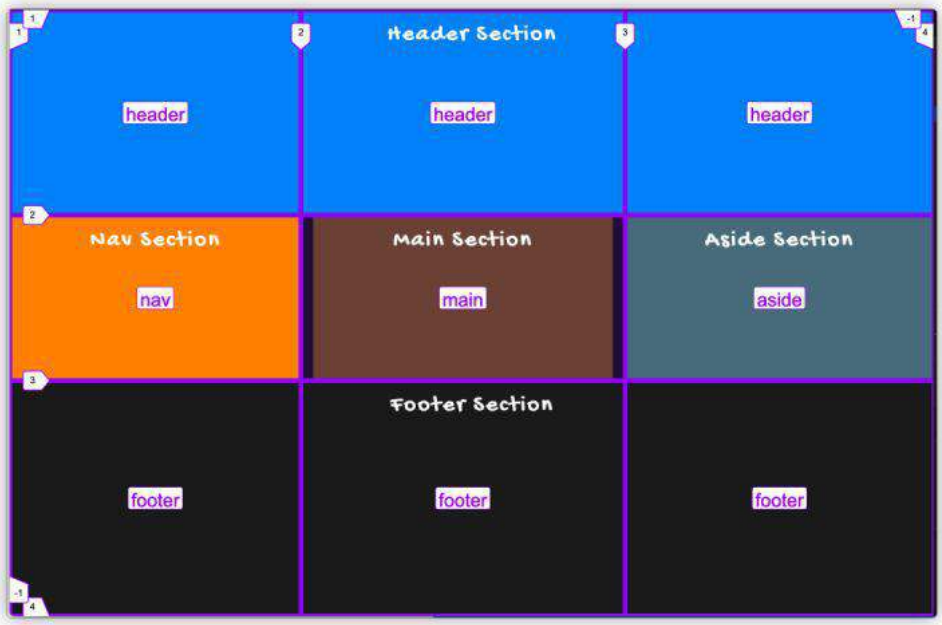


Demo(https://codepen.io/airen/embed/LYGdRrG)

你可能发现了它们之间的差异性：



后面这个示例中，`<nav>`、`<main>`和`<aside>`区域宽度相等。这是因为我们示例中通过 grid-template-areas 来声明网格，在使用 grid-template-areas 创建网格时，其实也隐式的创建了网格线，只不过他和 grid-template 不同的是 grid-template 可以显式的指定网格轨道大小，而 grid-template-areas 在该示例中相当于网格轨道大小都是 1fr。



如果我们希望<main>的区域变得更大，那么可以在 grid-template-areas 上做个调整：

```
body {
    display: grid;
    grid-template-areas:
        "header header header header header"
        "nav main main main aside"
        "footer footer footer footer footer";
}
```

效果如下：

Demo(https://codepen.io/airen/embed/QWymKYZ)

这个时候网格区域的划分像下图这样：

虽然在效果有所调整了，但还是均分状态。更好的解决方案是，将 grid-template-areas 和 grid-template 结合起来使用：

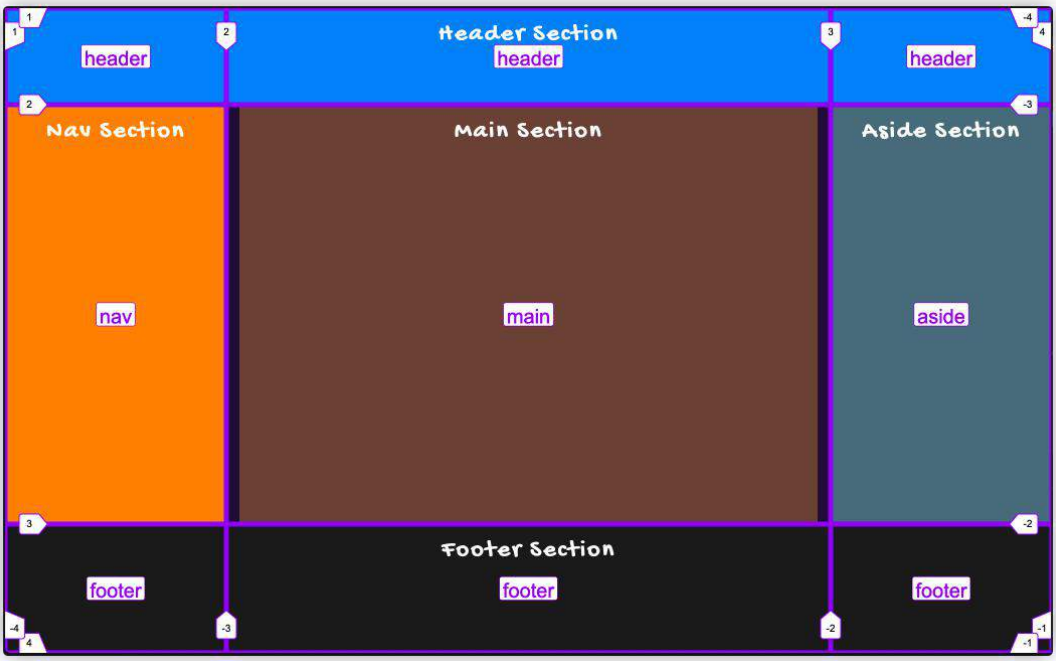
```
body {
    display: grid;
    grid-template-areas:
        "header header header"
        "nav main aside"
        "footer footer footer";
    grid-template-columns: 220px 1fr 220px;
    grid-template-rows: auto 1fr auto;
}

header {
    grid-area: header;
}
```

```
main {
    grid-area: main;
}

nav {
    grid-area: nav;
}

aside {
    grid-area: aside;
}

footer {
    grid-area: footer;
}

@media screen and (max-width: 800px) {
    body {
        grid-template-areas:
            "header"
            "nav"
            "main"
            "aside"
            "footer";
        grid-template-columns: 1fr;
        grid-template-rows: auto auto 1fr auto auto;
    }

    main {
        margin-left: 0;
        margin-right: 0;
    }
}
```

效果如下：

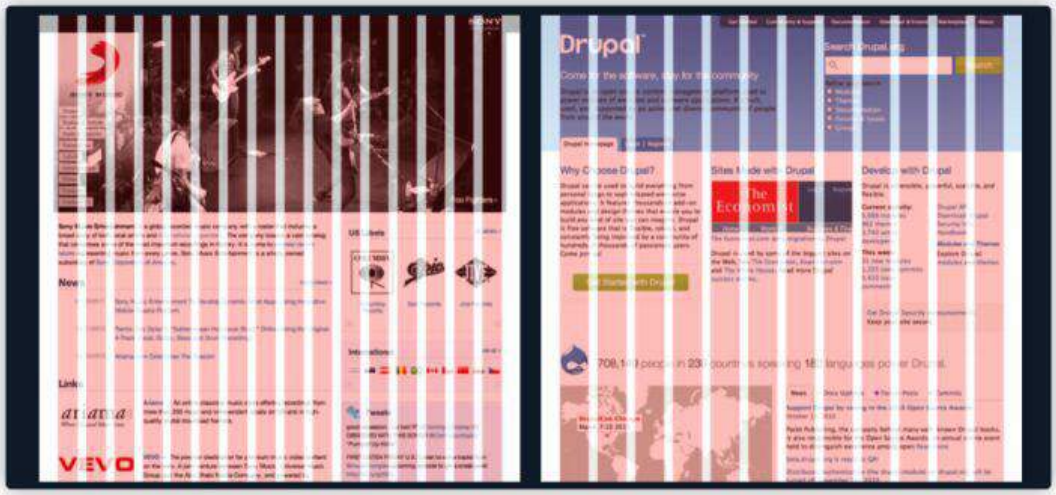
Demo(https://codepen.io/airen/embed/OJMvRev)

你可以发现，这个时候，网格线的区域的命名像下图这样：



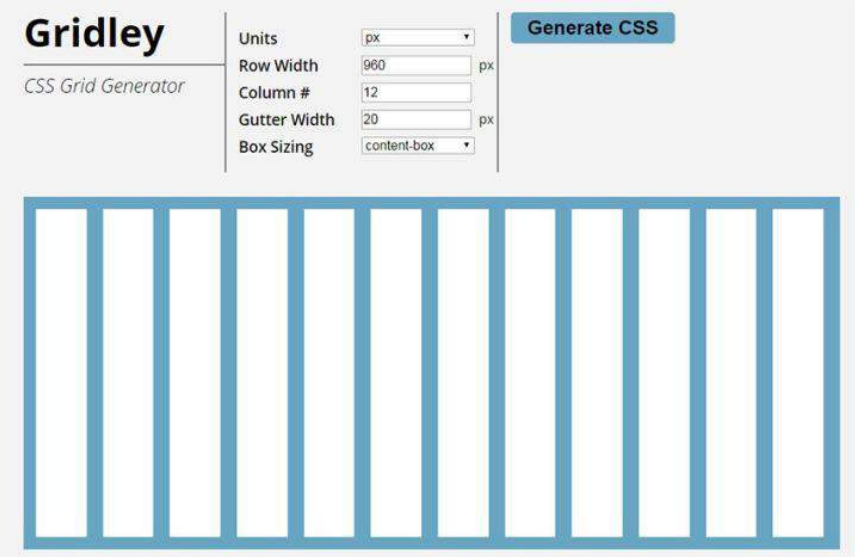
12 列网格布局

12 列网格布局最早是由 960.gs 提出的网格布局系统(//960.gs/)：



12 列网格布局在设计系统和 CSS Framework 中经常使用，比如业内经典的 Bootstrap(//http://getbootstrap.com/)就采用了 12 列网格布局系统：

在社区中也有很多在线工具，帮助我们快速构建 12 列网格系统，比如 Free CSS Grid Tools & Resources For Developers(//http://1stwebdesigner.com/free-css-grid-tools-resources/) 一文中罗列的工具。



(http://paulhebertdesigns.com/gridley/)

不过这里主要是想和大家一起看看在 Flexbox 和 Grid 布局模块中是如何实现 12 列的网格布局系统。

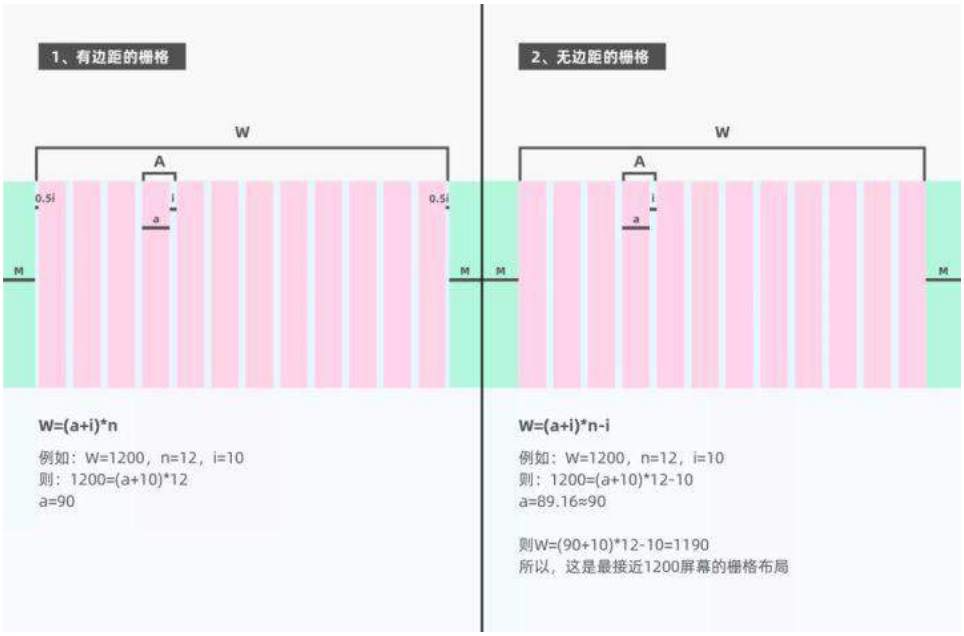
先来看 Flexbox 布局模块。12 列网格布局的 HTMl 结构一般类似于下面这样：

```
<!-- HTML -->
<flex__grid>
    <flex__row>
        <flex__item col4></flex__item col4>
        <flex__item col4></flex__item col4>
        <flex__item col4></flex__item col4>
    </flex__row>
</flex__grid>
```

注意，12 列网格中，一般同一行的列数值和刚好等于 12。比如上面的 HTML 结构，行中有三列，每列的宽度刚好四个网格宽度加两个列间距。并且在计算的时候有一套成熟的计算公式：

固定网格计算	流体网格计算
scw = (容器宽度 - (m * (mc - 1))) / mc cw = (scw * cs) + (m * (cs - 1)) scw：指的是单列宽度 m：指的是列间距 mc：最大列数（一般是12） cw：列宽度 cs：列数(1~12)	scw = (100% - (m * (mc - 1))) / mc cw = (scw * cs) + (m * (cs - 1)) scw：指的是单列宽度 m：指的是列间距(1.6%) mc：最大列数（一般是12） cw：列宽度 cs：列数(1~12)

而且还设计上也会有所差异，比如说距离容器两侧有没有间距等：



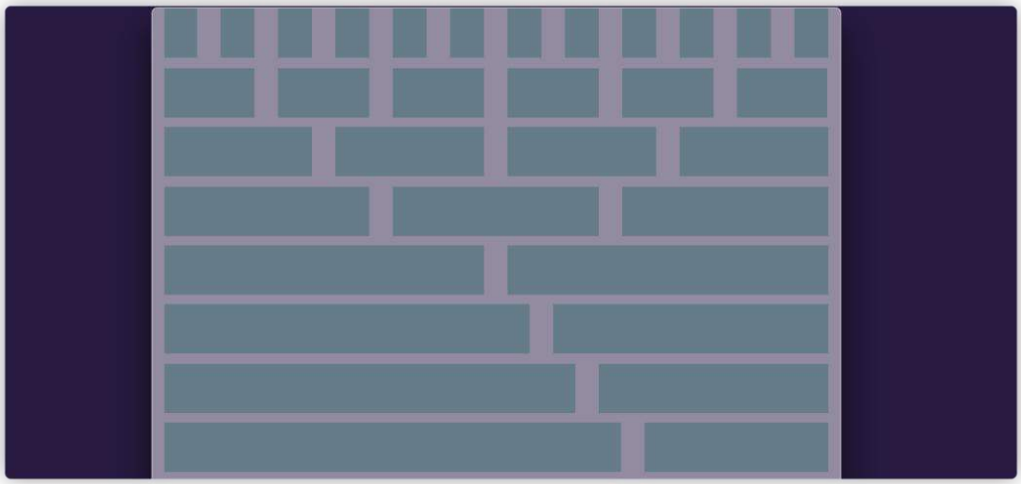
这些的差异对于计算公式和样式代码的设计都略有差异。我们用其中一个为例：

```
:root {
    --gutter: 10px;
    --columns: 12;
    --span: 1;
}

.flex__container {
    display: flex;
```

```
    flex-direction: column;
    padding-left: var(--gutter);
    padding-right: var(--gutter);
}

.flex__row {
    display: flex;
    margin-left: calc(var(--gutter) * -1);
    margin-right: calc(var(--gutter) * -1);
}

.flex__row + .flex__row {
    margin-top: 2vh;
}

.flex__item {
    flex: 1 1
        calc((100% / var(--columns) - var(--gutter)) * var(--span));
    margin: 0 var(--gutter);
}

.flex__item1 {
    --span: 1;
}

.flex__item2 {
    --span: 2;
}

.flex__item3 {
    --span: 3;
}

.flex__item4 {
    --span: 4;
```

```
}

.flex__item5 {
    --span: 5;
}

.flex__item6 {
    --span: 6;
}

.flex__item7 {
    --span: 7;
}

.flex__item8 {
    --span: 8;
}

.flex__item9 {
    --span: 9;
}

.flex__item10 {
    --span: 10;
}

.flex__item11 {
    --span: 11;
}

.flex__item12 {
    --span: 12;
}
```

你会看到的效果如下：



Demo(https://codepen.io/airen/embed/YzwaxwX)

在该示例中采用了 CSS 自定义属性相关的特性，让整个计算变得更容易一些。

对于使用 CSS Grid 布局模块来实现 12 列网格布局，相对而言，不管是 HTML 结构还是 CSS 代码都会更简易一些。在使用 CSS Grid 布局模块实现 12 列网格布局，将会运用到 repeat()、minmax()、gap 和 fr 等特性。具体的来看一个示例吧。

```
<!-- HTML -->
<grid__container>
    <grid__item></grid__item>
</grid__container>
```

我们来看 CSS 代码：

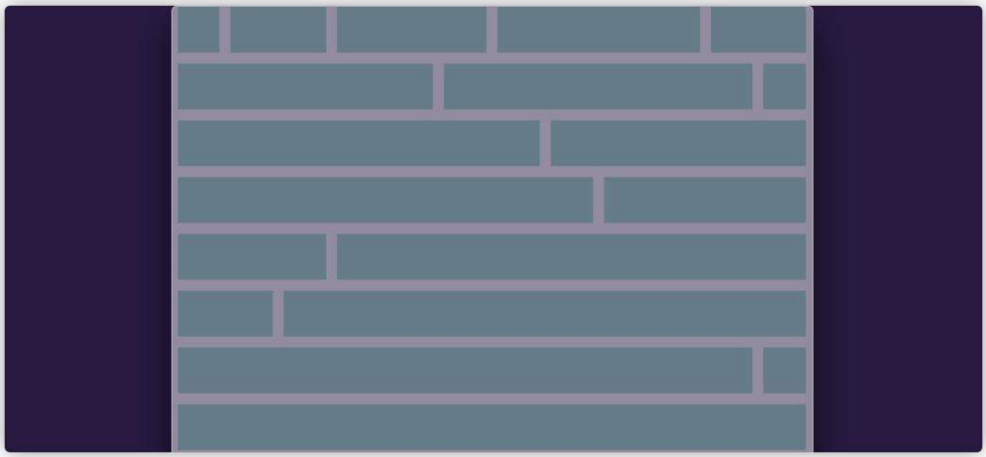
- 使用 fr 将网格均分为相等的值，即每列宽度都是 1 个 fr；配合 repeat()函数，即 repeat(12, 1fr)创建了 12 列网格。
- 使用 gap 可以用来控制网格之间的间距。
- 配合 minmax()还可以设置网格最小值。

具体的代码如下：

```
:root {
    --columns: 12;
    --gap: 10px;
    --span: 1;
}

.grid__container {
    display: grid;
    grid-template-columns: repeat(var(--columns), 1fr);
    grid-template-rows: 1fr;
    gap: var(--gap);
    padding-left: calc(var(--gap) / 2);
    padding-right: calc(var(--gap) / 2);
}

.grid__item {
    min-block-size: 10vh;
    grid-column: span var(--span);
}

.col1 {
    --span: 1;
}

.col2 {
    --span: 2;
}

.col3 {
    --span: 3;
}

.col4 {
```

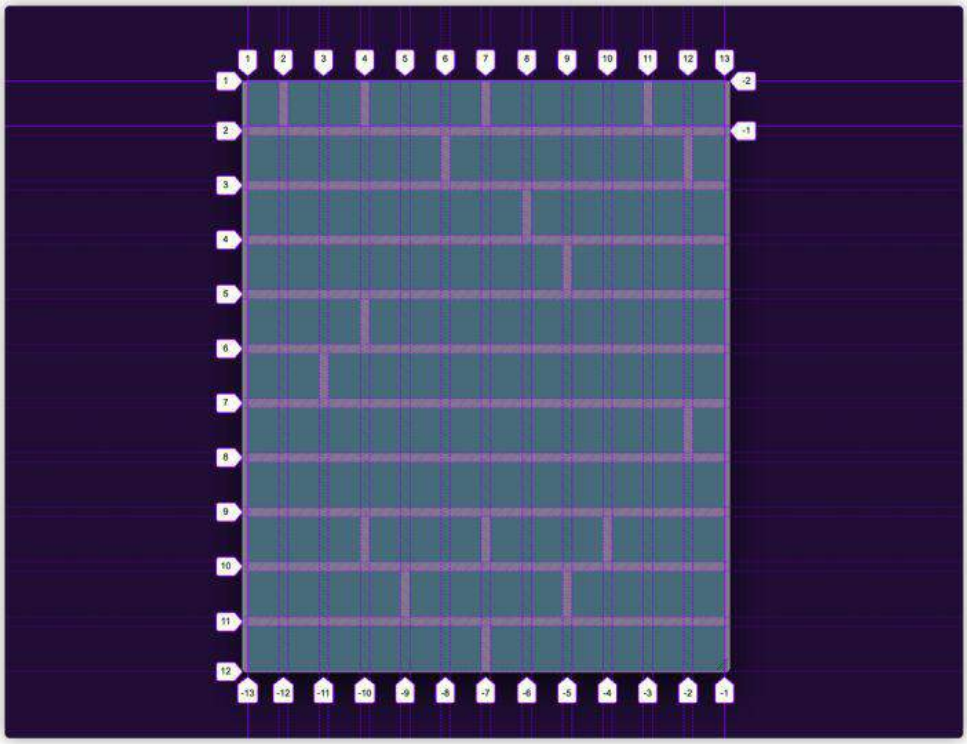
```
    --span: 4;
}

.col5 {
    --span: 5;
}

.col6 {
    --span: 6;
}

.col7 {
    --span: 7;
}

.col8 {
    --span: 8;
}

.col9 {
    --span: 9;
}

.col10 {
    --span: 10;
}

.col11 {
    --span: 11;
}

.col12 {
    --span: 12;
}
```

你将看到的效果如下：



Demo(https://codepen.io/airen/embed/yLeKPPb)

就该示例而言，`grid-template-columns: repeat(12, 1fr)`创建网格如下图所示：



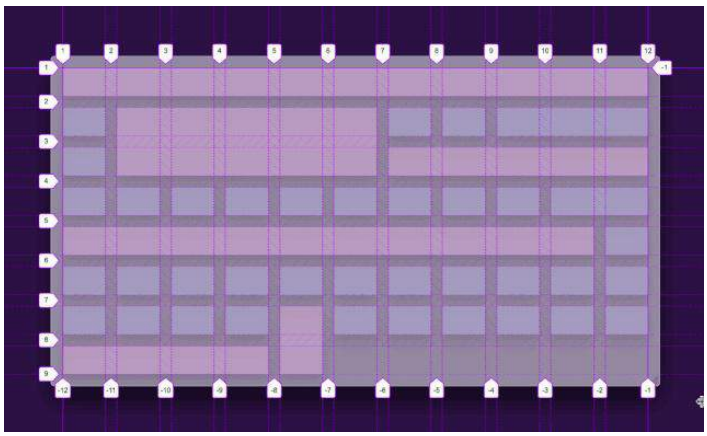
除了上述这种粗暴的方式，还可以更灵活一些，将 auto-fit、minmax()以及 grid-auto-flow: dense 等来创建：

```
.grid__container {
    padding: 1em;
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(60px, 1fr));
    gap: 1em;
    grid-auto-flow: dense;
}
```

对于.grid__item 可以通过 grid-column、grid-row 来控制网格项目的位置：

Demo(https://codepen.io/airen/embed/QWymabq)

加上 grid-auto-flow: dense 会根据 Grid 容器空间，Grid 项目会自动流到合适的位置：



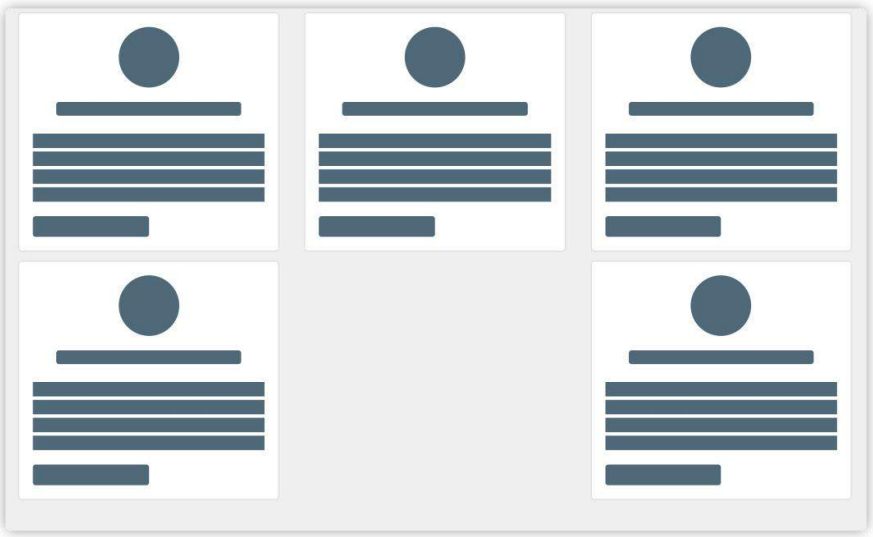
这种布局对于杂志类的布局非常的适用。有关于这方面更详细的介绍可以阅读@Keir Watson 的《Responsive Grid Magazine Layout in Just 20 Lines of CSS(//http://css-tricks.com/responsive-grid-magazine-layout-in-just-20-lines-of-css/)》一文。

两端对齐

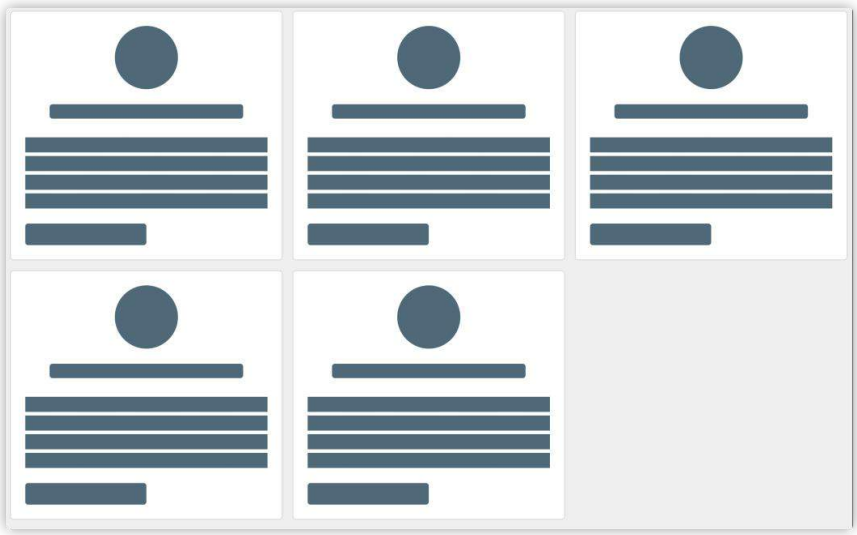
在 Web 布局中时常碰到两端对齐的需求。在 Flexbox 布局中，时常在 Flex 容器中显式设置 justify-content 的值：

```
.flex__container {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;

    width: 100%;
}
```

但在末尾行，如果和前面行的个数不相同（Flex 项目）就会出现下图这样的效果：



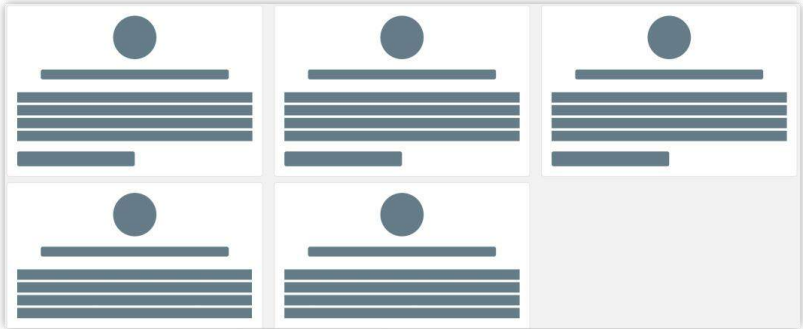
像上图这样的效果，并不是我们所需要的，因为我们希望在最后一行的 Flex 项目不足够排列满一行时，希望 Flex 项目一个紧挨一个的排列：



在 Flexbox 要实现上图这样的效果，只需要在 Flex 容器中添加一个伪元素：

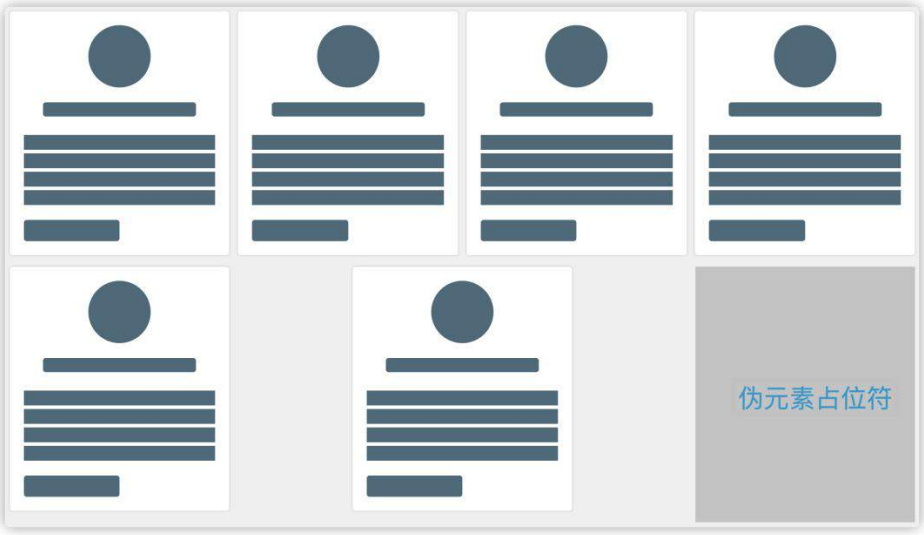
```
.flex__container::after {
    content: "";
    display: flex;
    flex: 0 1 32vw;
}
```

注意，伪元素的 flex-basis 建议设置的和卡片的 flex-basis（或宽度）等同。这个时候你将看到像下面这样的示例：



Demo(https://codepen.io/airen/embed/QWymaam)

不过这种方式也不是最佳的方式，当末尾行的个数不只少一个时，就会出现下图这样的效果：



面对这样的场景，我们需要给 Flex 容器添加额外的空标签元素：

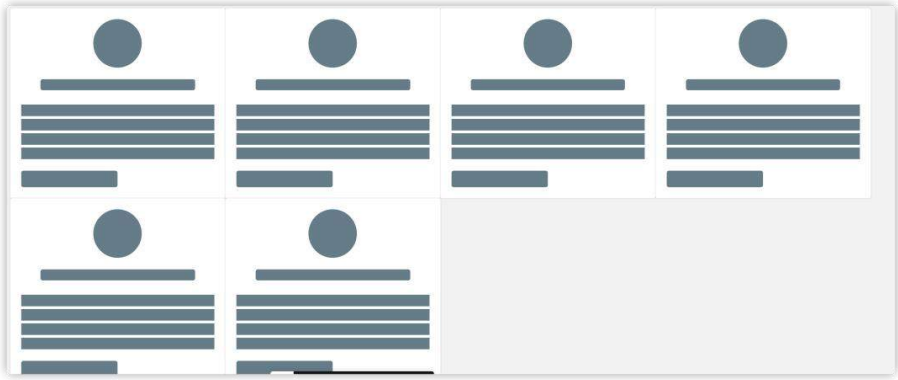
```
占位符元素数量 = 每行最大的列数 - 2
```

但是 gap 属性出现之后，要实现这样的效果就不难了：

```
body {
    padding: 1vh;
}

.flex__container {
    display: flex;
    flex-wrap: wrap;
    gap: 2vh;

    width: 100%;
}
```

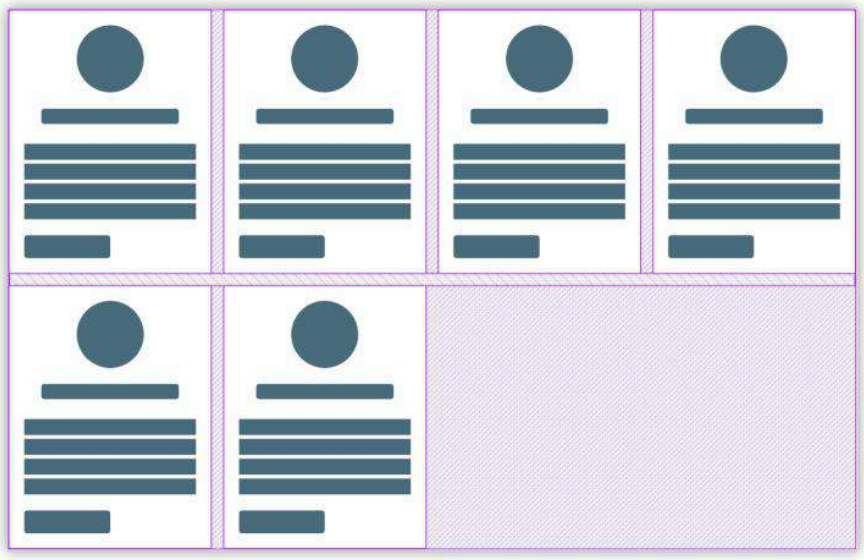
```
.flex__item {
    flex: 0 1 calc((100vw - 8vh) / 4);
}
```

效果如下：



Demo(https://codepen.io/airen/embed/YzwaYBN)

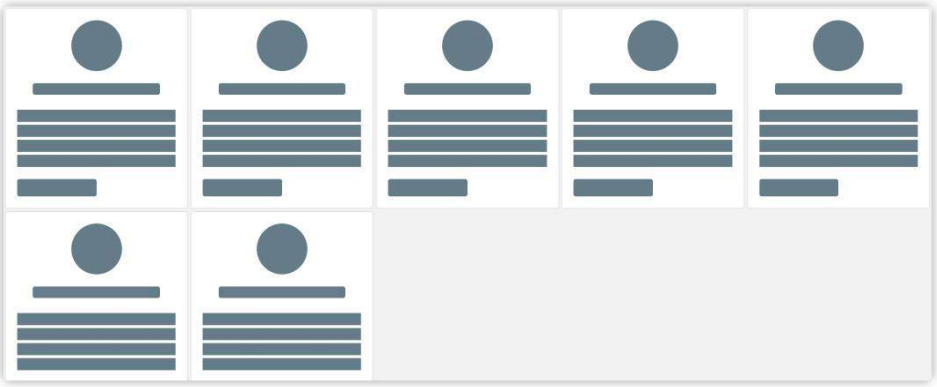
注意，gap 运用在 Flexbox 中到目前为止，仅得到了 Firefox 浏览器的支持。上面的示例，使用 Firefox 浏览器，你看到的效果如下：



在 CSS Grid 布局中，就可以直接使用 gap：

```
body {
    padding: 1vh;
}

.grid__container {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
    gap: 1vh;
}
```

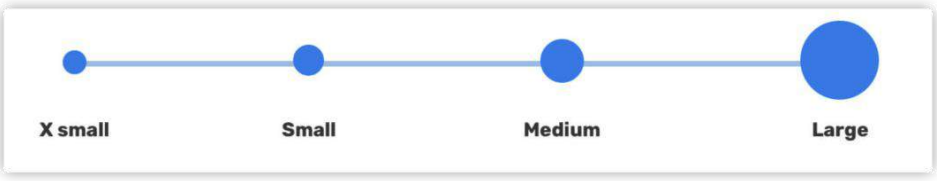
效果如下：



Demo(https://codepen.io/airen/embed/eYJMVbz)

选择最佳的值

很多时候，针对不同的场景，设计师会为我们提供不同的设计风格，比如元素大小：



随着 clam()函数的到来，这一切都变得容易地多。

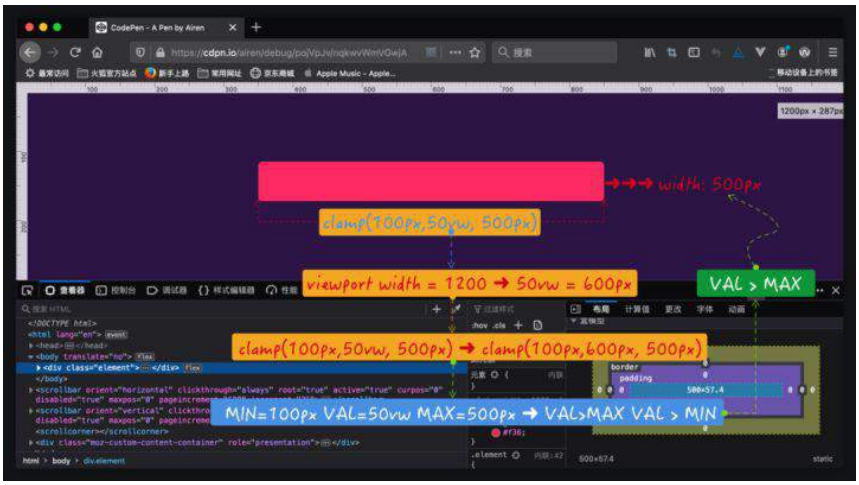
clam()函数接受三个参数，即 clam(MIN，VAL，MAX)，其中 MIN 表示最小值，VAL 表示首选值，MAX 表示最大值。它们之间：

- 如果 VAL 在 MIN 和 MAX 之间，则使用 VAL 作为函数的返回值；
- 如果 VAL 大于 MAX，则使用 MAX 作为函数的返回值；
- 如果 VAL 小于 MIN，则使用 MIN 作为函数的返回值。

我们来看一个示例：

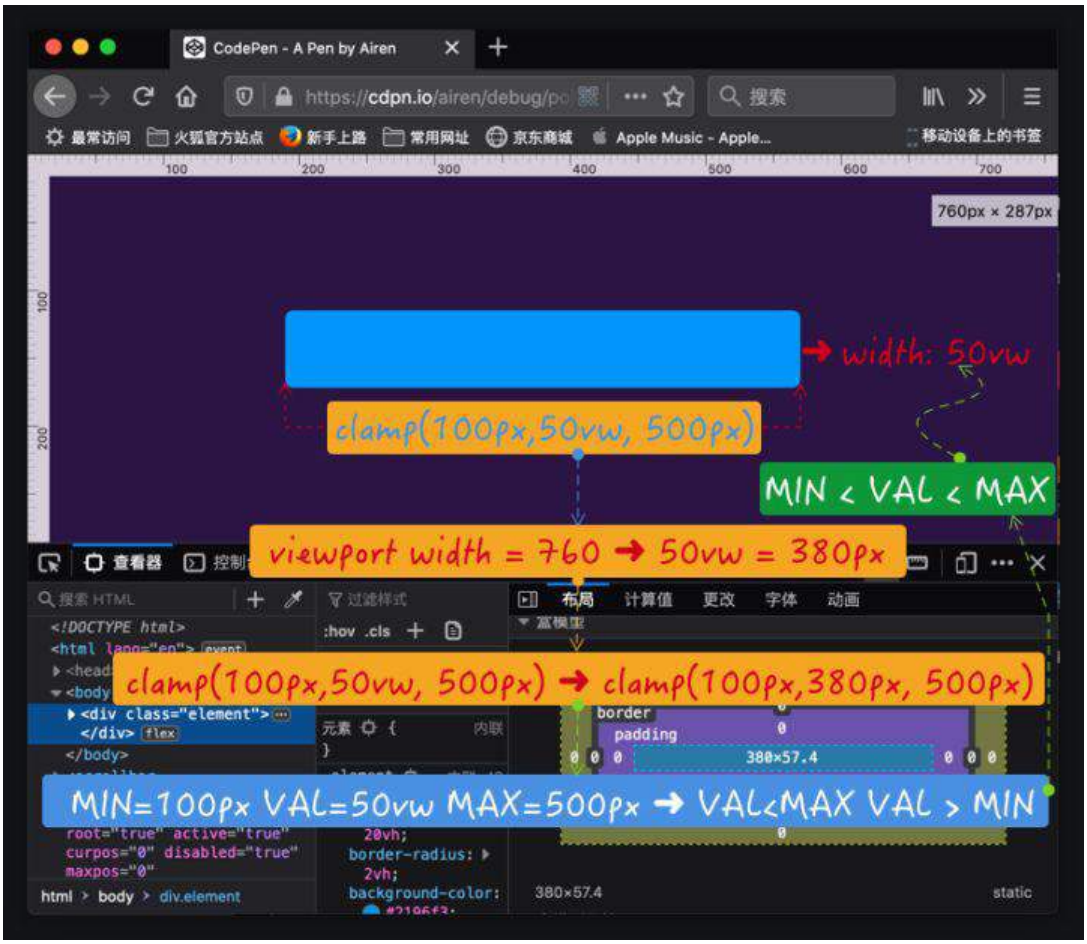
```
.element {
    /**
    * MIN = 100px
    * VAL = 50vw ➜ 根据视窗的宽度计算
    * MAX = 500px
    **/

    width: clamp(100px, 50vw, 500px);
}
```

比如浏览器视窗现在所处的位置是 1200px 的宽度，那么.element 渲染的结果如下：



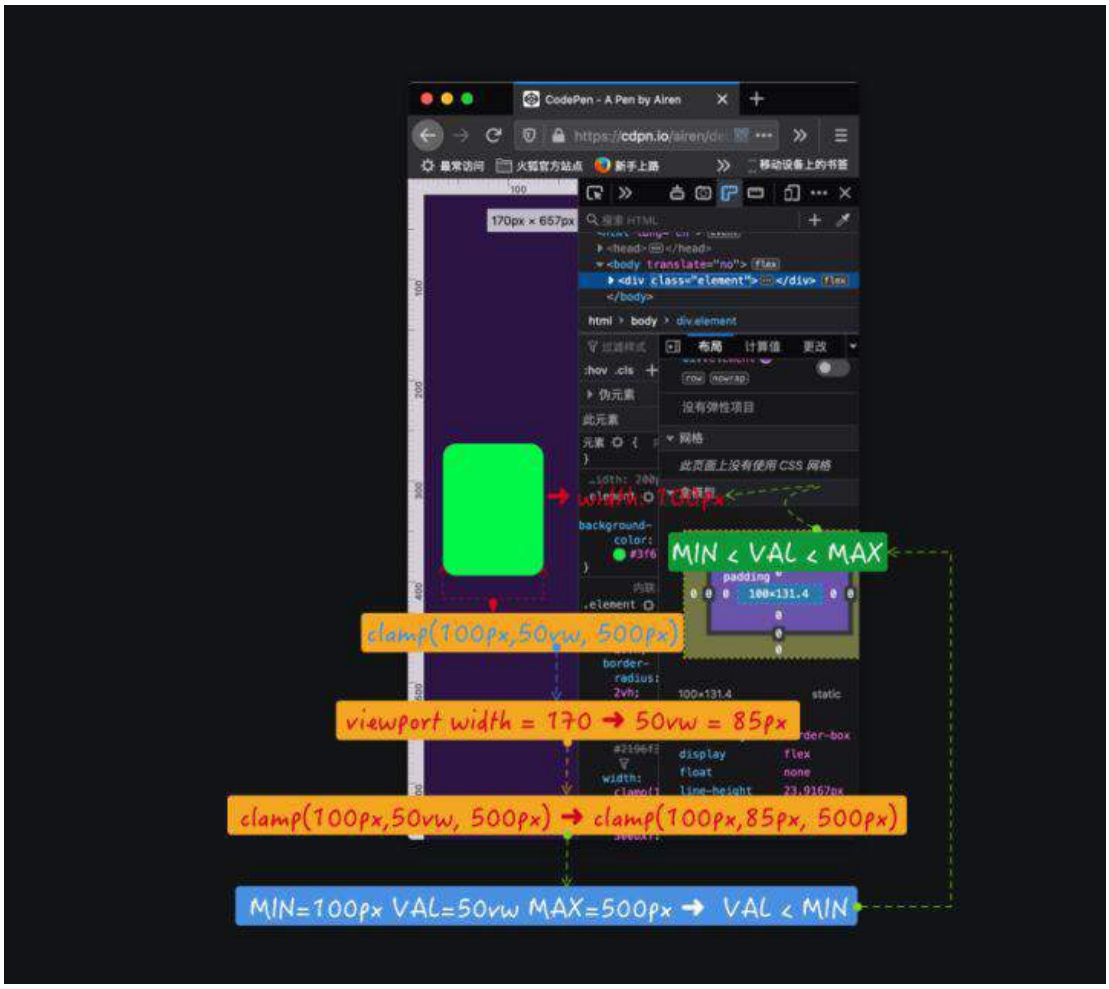
这个时候.element 元素的 width 是 500px。此时，clamp(100px，50vw，500px)相当于 clamp(100px，600px，500px)，对应的 VAL 值是 600px，大于 MAX 值，那么这个时候 clamp()函数返回的值是 MAX，即 500px，这个时候.element 的 width 值就是 500px（即 MAX 的值）。

如果我们把浏览器视窗缩小至 760px:



这个时候.element 元素的 width 是 50vw。此时，clamp(100px，50vw，500px)相当于 clamp(100px，380px，500px)，对应的 VAL 值是 380px，该值大于 MIN 值（100px），小于 MAX 值（500px），那么这个时候 clamp()函数返回的值是 VAL，即 50vw，这个时候 .element 的 width 值就是 50vw（即 VAL 的值）。

如果继续将浏览器的视窗缩小至 170px:



这个时候.element 元素的 width 是 100px。此时，clamp(100px，50vw，500px)相当于 clamp(100px，85px，500px)，对应的 VAL 值是 85px，该值小于 MIN 值（100px），那么这个时候 clamp()函数返回的值是 MIN，即 100px，这个时候.element 的 width 值就是 100px（即 MIN 的值）。

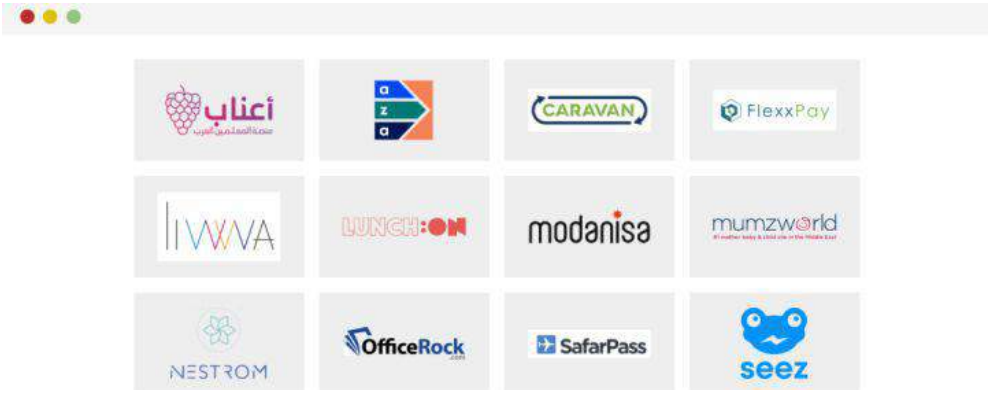
就该示例而言，clamp(100px，50vw，500px)还可以这样来理解：

- 元素.element 的宽度不会小于 100px（有点类似于元素设置了 min-width：100px）
- 元素.element 的宽度不会大于 500px（有点类似于元素设置了 max-width：500px）
- 首选值 VAL 为 50vw，只有当视窗的宽度大于 200px 且小于 1000px 时才会有效，即元素.element 的宽度为 50vw（有点类似于元素设置了 width：50vw）

具体效果如下占击这里查看(https://codepen.io/airen/embed/pojVpJv)。

Logo 图标的对齐

我想你在 Web 开发中可能碰到过类似下图的这样的场景：



正像上图所示，Logo 图像的有大有小（宽度和高度都不一样）。面对这样的业务场景，很多时候都希望设计师能提供相同尺寸的图像。但这样势必会影响 Logo 图像的外观。

前段时间看到@Ahmad Shadeed 专门写了一篇博文《Aligning Logo Images in CSS(//http://ishadeed.com/article/aligning-logos-css/)》，就是介绍如何实现上图这样的布局效果。

其实实现这样的布局效果，主要运用到的就是 CSS 的 `object-fit` 属性，而这个属性早在多年前就得到了各大主流浏览器的支持。

这里我们用一个简单的示例，来看看具体实现过程。先来看 HTML 结构：

```
<!-- HTML -->
<ul class="brands">
    <li class="brands__item">
        <a href="#">
            <img src="img/logo.png" alt="">
```

```
        </a>
    </li>
    <li> <!-- ... --> </li>
</ul>
```

居中对齐前面已经介绍过了，这里主要是看图像大小方面的处理：

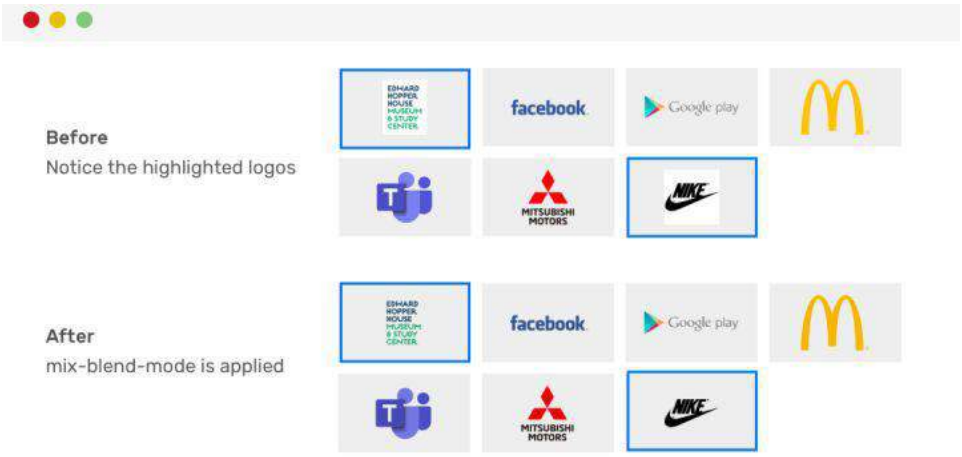
```
.brands {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));
    grid-gap: 1rem;
}

.brands__item {
    background: #eee;
}

.brands__item a {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100%;
}

.brands__item img {
    width: 130px;
    height: 75px;
    object-fit: contain;
}
```

这样就能实现上图的效果。你可能发现了，有些 Logo 图像带有背景颜色，如果让效果更好一些，可以把 CSS 混合模式相关的特性运用进来：

.

```
brands__item img {
    width: 130px;
    height: 75px;
    object-fit: contain;
    mix-blend-mode: multiply;
}
```

这个时候，你看到的效果如下：



object-fit 除了取值 contain 之外，还有其他几个值，具体的可以看这个示例(https://codepen.io/airen/embed/VweXXoo)：

其实这个方案也适用于产品图片，人物头像等布局。

小结

文章中主要介绍了 Web 中一些布局的实现思路和具体方案。其实文章提到的效果，比如水平垂直居中、等高布局、平均分布列和 Sticky Footer 等，在 CSS 中一直有多种解决方案，只不过随着 CSS Flexbox 布局模块和 CSS Grid 布局模块的到来，实现这些效果变得更为灵活和简洁。

当然，文章中提到的只是一些最为常见一些效果，其实在 Web 布局中，特别是 Flexbox 布局和 Grid 布局中还存在很多有意思的东西，只不过因为篇幅的时间没有一一罗列。如果你感兴趣可以再挖掘一些出来，如果你在这方面有更好的经验或方案，欢迎在下面的评论中分享。最后希望这篇文章对你平时的工作有所帮助。

Q: 后端同学是不是比前端同学理解业务更快?

作者 | 淘系技术
出品 | 淘系技术

像我所在的部门因为业务逻辑主要在后端，前端同学与后端同学的比例大概为 1:4。因为业务的扩张，新的后端同学在不断的加入，发现一个现象，新入职的后端同学可以在 1.5 周左右的时间里，对部门的业务线基本可以达到 70% 左右的了解（新人会做分享）。然后有做了三年的前端同学对业务都不是很了解，同组的几个前端同学都是，因此有这个疑问。我在上家公司的时候，也是做了几个月，对业务模型都不是很了解。发现大多数情况下，工作中都是拧螺丝、CRUD 类型的页面功能。请问是不是普遍后端比前端更容易理解业务模型？或者说，前端如何快速的搞懂公司/部门的业务模型？

在阿里巴巴淘系技术部，最近有位参加工作不久的前端同学总结了自己理解业务的一整套心法，从困境心情到突破方法，瞬间成为我们“网红”。(啪啪鼓掌

这里我们就不做前后端比较了，主要回答题主的最后一个问题：前端如何快速的搞懂公司/部门的业务模型？

作为一个业务前端，完成业务需求的同时，还要处理各种线上问题，加班辛苦忙碌了一年，还要被老板说“思考是不够的”、“没有业务 sence”，出去面试，被问项目，也说不出什么有亮点或者有挑战的东西，想做点牛逼的东西，也没有发现什么有价值的方向，好不容易找到一些方向，还要被老板一顿质问，业务价值是什么？ROI 怎样？最终可能就只是做了一点性能优化工作，抽离了一些可复用的组件……不禁让人感叹，业务难、前端难、做业务的前端更难！

如果你也有这样的感受和困境，我想告诉你，这真的是太正常了，在阿里内部的技术论坛就有多篇关于这个问题的思考，我根据根据自己理解和调研，同时参考了多位不同前端领域专

家的总结，整理成这篇文章，希望能对大家有所帮助。

业务前端的困境

业务前端"好忙"

业务前端，顾名思义，做业务的前端，直接与业务的 PD、运营接触，对产品的用户直接负责。在实际的工作中，业务前端经常忙于业务的各种会议、项目和答疑，即便一条业务线上有多个前端同学支持，面对成山的需求，可能依然感到吃力，这其中的原因可能有：

1. 用户侧产品往往需要快速上线，大部分需求都需要倒排工期，开发时间尤其紧张。
2. 对业务不熟悉，在项目需求已确定的时候才去参加视觉评审，没有办法判断需求背后的业务逻辑跟业务大节奏是否匹配、需求本身是否能够达成业务目标、有没有更好的实现方式，只能接下需求，然后排期。
3. 维护成本高，每天还要忙于解决各种线上问题，比如这里样式有点问题，那里怎么没有显示……各种琐碎问题让你过的非常"充实"。
4. 需求响应速度较慢，比如业务的技术栈较老，或者定制逻辑过多，边写代码还要边查文档，查不到可能还要查源码，效率大幅降低。又或者跟别的业务技术体系不同，难以复用和沉淀，如果要用，可能还要重写一遍……

业务前端是"资源"？

前端岗位的特点就是有视觉稿就可以完成工作，不需要理解业务全貌，所以在繁忙期很容易让前端忽视了业务思考，加上之前描述的各种原因，业务前端经常沦落为"资源"，当你沦落为"资源"的时候，其实就已经失去了和业务平等对话的资格，他们只会把你当成莫得感情的开发机器，跟你输入需求，让你吐出页面，而你在这样的关系中，本来写着还算工整的代码，为了快速实现业务需求，也开始写起乱糟糟的代码，对于你所创造的产品也没有话语权，久而久之也失去了激情和耐心。

失去激情，写的不开心也就算了，因为你没有做出什么特别的东西，老板也不会特别认可你的辛苦，还会觉得你思考不够、没有业务 sence，对业务没有助力，没有让业务因为你的存在而有所不同……

业务前端想突破

好吧，那我决定做点什么改变一下，于是跟老板提出了一系列想法：

1. 这里技术体系太老了，为了进一步提升开发效率，我们想要搞技术重构。
2. 前后端联调有点费劲，我们想搞个联调数据中台，提升联调效率。
3. 那里展现速度太慢了，我们要搞性能优化。
4. ……

老板往往会来一系列灵魂提问：

1. 为什么要做？（有什么业务价值？有什么技术价值？）
2. 为什么是现在做？
3. 为什么是你做？
4. ROI（投入产出比）怎么样？

还没有开始，躁动的心就被老板的一系列"质疑"浇了一盆冷水。

如果没有回答好这些问题、说服老板，自然也争取不到什么资源，只能一个人搞搞，一个人搞的往往质量不行、也没有人用，久而久之自己也不维护了，只能又开始埋头在需求中。

干的不开心，也没有成长，最后只能暗淡离职，但换了一个公司就会好吗，很可能又是类似的过程……

这真的堪称是业务前端的"困境"，那么如何突破这种困境呢？首先我们就要摆正心态，从了解业务开始。

了解业务

业务和需求

在了解业务之前，首先我们要知道，业务跟需求是不一样的。理解需求并不等于理解业务，需求是业务经过产品消化后的产物，可能已经经过演绎或者拆解，因此需求并不是业务本身，当然了解的需求越多，对业务的全貌也会更加了解。

那么什么是业务呢？业界对"业务"有多种定义，但是其主要思想基本不变，业务就是一系列人通过一系列活动完成某一任务的过程，因此，业务可大可小，可以无限拆分。

我们本文涉及的业务泛指商业业务，就是与该 BU 或者公司商业模式直接关联的业务或其组成部分。

前端为什么要学习业务

前端即使不学习业务，其实也不影响做需求，毕竟你只要告诉我交互是什么样的，前端就可以帮你实现，而且已经有产品经理的角色了，大家各司其职不就好了，为什么一个做技术的，要狗拿耗子、或者是越俎代庖呢？这就要说到：

1. 只有了解业务，才能从技术的角度想到业务方不曾想到的地方；不了解业务，你可能听不懂业务方要什么，甚至连需求的业务逻辑都搞不清，这种情况的合作模式只有一种，需求下来了，你接住，然后给排期。也许，这个需求的设计不合理，你不知道；这个需求有更好的实现方案，你不知道；这个需求可以通过现成的关联产品方案解决，省时省人力，你也不知道。
2. 只有了解到业务背后的原因，才能从全局的视角去规划技术的未来。不了解业务，会让你离用户的真实需求很远，你越难发现其中的一些痛点和挑战，没法真正提出你的思考和解决方案，去解决用户的难题。
3. 作为一名产品研发工程师，自然是希望亲手打磨一款解决用户问题、体验友好的产品，如果产品能得到用户认可，产生影响力、自然会特别有成就感。

4. 阿里作为一家商业科技公司，对技术人的要求就是技术与业务相结合，在满足业务需求的基础上，成为技术与业务的桥梁，主动走进业务，思考如何通过技术手段帮助业务做赢、满足市场和用户需求，先一步技术规划、人才储备、技术架构和技术预研。

你了解业务吗?

那么目前你了解你对接的业务吗？不妨尝试回答下以下问题：

1. 业务做的是什么？产品大图有吗？
2. 业务的核心指标是什么？KPI 目标是什么，这些数字背后的含义是什么？要达成这些目标，业务策略是什么？
3. 业务的用户是谁？流量怎么分层？占比多少？分别在业务中是怎样的定位？
4. 业务的商业模式？靠什么吸引流量，盈利模式是怎样的？
5. 我们做的页面是什么东西？为业务带来什么价值？要创造更多的价值，我们可以做什么？

如何学习业务?

业务领域知识的阅读

找到该领域相关的评分较好的书籍集中阅读，快速形成知识框架。

了解业务背景和规划。

1. 刚刚接手新的业务，可以邀请业务方老板或者资深的运营/产品同学，给你讲讲这块业务的过去、现在、未来、愿景、财年规划，以及对技术同学的期望。
2. 花时间读合作方（运营、产品、研发）的周报，了解现在在发生什么，是不是离目标越来越近了。
3. 了解业务目标、落地策略、衡量目标的数据口径，关注数据，关注目前做的项目是否为了达成目标而战，如果不是，提出你的想法和建议。

4. 多参会，建立产品 sense。收集信息最好的方式就是参加所处业务老大的 KO 会，各种 KO 会会把战略上的拆解和背后的思考整体梳理之后宣讲传达给 BU 或部门的同学。

多交流

与服务端同学聊天，与 PM 聊天，与用户聊天，多角度看业务，但要注意的是，针对专业型比较强的业务，需要先做功课，至少一些英文的缩写要清楚的明白意思。

谨记数字

如果前面还需要花比较长的时间，那这一个可以现在就做起来，那就是把业务相关的数字记得越精细约好，越具体越好，越全面越多越好。这样做有两个好处：

1. 所记的数字指标本身，很大程度已经涵盖了这个业务价值方向，你便知道了这个业务重点关注的是哪个维度的东西。
2. 这些数字可作为和业务方以及产品"平等对话"的源头，否则连最基本对话基础都没有。

从日常需求入手

对于项目中的需求，我们要尝试分析背后的目的和价值，做了之后有什么预期的收益，为什么这么做就可以达到这个收益，跟总体目标是否契合，还要判断业务方提到的点是不是有效的方案或者说成本太大的方案，看能不能给出替代方案，用现有的方案或者小成本的方式来满足业务方。

而在项目提测上线后，还要仔细分析以及多关注上线之后的业务数据和效果，会有如下好处：

1. 提高自己对业务的理解能力，你在关注业务数据的同时，也就会更多的从业务的角度来看到这个功能所带来的价值是否符合预期，当出现不符合预期的时候，可以和业务方一起进行数据漏斗的分析从而找到问题所在，避免我们的劳动成果成为一次性的工作。

2. 总结的同时可以帮助自己梳理这个项目中自己哪些地方做的不足，或者相关推进中存在什么问题，以及后面怎么改进，提高了下次项目中的迭代效率和质量。比如这个项目是否存在需求理解不到位存在返工，或者沟通 & 联调低效，环境不稳定，自己设计的方案是否合理等问题，后续要怎么解决。
3. 也可以从数据和总结中判断出什么样的需求是靠谱的 & 什么的样业务方是靠谱的，频繁争取资源上线效果又不好的业务方，下次再有需求过来则需要多增加一个心眼和思考的过程。

坚持

业务思考力，没有个至少半年是不会见效的。

助力业务

思考

尽管平时的业务很忙，但再忙，也要抽时间思考，那么思考哪些内容呢？以下举一些例子：

1. 养成每天记工作内容的习惯，分析一下自己的时间到底耗在哪了？
2. 在业务开发中，有遇到让你特别想吐槽的点吗？想下问题背后的原因，有什么方法可以避免下次不犯，能不能提炼为更加通用的解决方案，其他同学怎么解决的，我可以怎么解决？
3. 不断地输入、观察，业务的真实需求是什么？站在业务方的角度思考，业务遇到的痛点、挑战在哪里？

沟通

和老板、团队同学、业务方对焦，确认“我想做的”是不是“大家想要的”？你可能会提出很多意见，但一般会遭到老板或者业务方无情的拒绝，而且问得你一脸懵逼，就比如：

1. 当前业务背景下，为什么要做？（有什么业务价值？有什么技术价值？）
2. 现在必须做么？
3. 为什么是你做？
4. 怎么做？（体系化、全链路、单点技术挑战）
5. 有什么业务和技术结果？能否被复用？
6. 未来规划（能否跟 BU 或集团的方案联动、共建）

而这往往是因为你提出要做的事情，有价值但不是必须做的，没有结合目前业务需要什么。也就是说，你想做的技术是个人和纯技术角度思考的，没有基于业务的现状和痛点去考虑技术方案，不接地气，投入产出比不高。

所以给技术产出先找好业务的阵地，看看有没有可以借力的地方，不要重复造轮子。快速验证这个方向的正确性后，再逐渐多加投入、丰满技术设计。不要自己 YY、默默地做完，这样做出来的东西没有业务场景埋单。

技术规划

业务赋能其实是需要我们紧贴业务规划，制定技术规划和方案。在了解业务方今年的 KPI 重点是什么，预计的拆解和实现路径是什么后，再结合自己的和团队情况，想想自己能做哪些事情来帮助业务实现其 KPI，这里有两点需要注意下：

抓住本质从点及面，通盘考虑：很多时候，我们收到的痛点和业务需求都是单点的，这时我们不能着眼于眼前的单点问题，而需要通盘来考虑，比如 SEO 的页面对性能非常敏感，经常可能会收到一些业务方来反馈，说目前我们的 SEO 有这个地方，那个地方需要优化下，而单点解决这些问题可能对业务带来的收益并不大，对自己的技能也没有什么成长。这时候如果通盘考虑这个命题，其实会发现做 SEO 页面的优化，其实目的是为了提升 SEO 页面的收录和排名。而提升 SEO 页面的收录和排名其实不仅有前端性能优化这一个路径，而是还有一些其他的路径：比如优化关键词&长尾词，采用 Google 的 AMP 技术改造 SEO 页面，优化爬虫爬取页面的耗时提升爬取率等等。这样就能吧点的问题转化为面的问题，才能制定更有效和全面的抓手来赋能业务。

既要解决眼前痛点，也要长远谋划：很多时候我们不能仅满足于眼前的 KPI，还需要了解业务方长远的想法和可以预见的规划。就比如试点的新业务，一层规划是保证业务项目的按时上线，考虑到未来，另一层规划可能就是如何做到技术方案的可以复制性。

站在巨人的肩膀上

当你需要制定一个产品化的方案或者工具和框架的时候，最好先放眼集团内部和行业进行一番调研，看看业界和其他同事是怎么解决这个问题的。尽量站在别人的肩膀上做出创新或者参与共建，避免小团队内造出重复和质量低的轮子

技术深度

技术知识与技术能力

“技术”不能是一个笼统的词汇，我想它至少可以分为“技术知识”和“技术能力”两大部分。

什么是“技术知识”？知识就是 I KNOW

- 《TypeScript 从入门到放弃》
- 《React 从入门到放弃》
- 《Webpack 从入门到放弃》
- ……

什么是“技术能力”？能力就是 I CAN

- 我用 TypeScript 重构了一个大型系统，代码健壮性及研发效率大幅提升。
- 我用 React Hooks 给全栈同学进行前端培训，培训效果大幅提升。
- 我深入研究了 Webpack，优化配置，使得系统构建速度大幅提升。
- …..

培养技术视野

1. 关注日常业界新技术。不一定要深入了解，但对新技术保持好奇心，大概了解它是做什么的，如果在工作中遇到匹配的落地环境，可以考虑写个 demo 看看是不是有价值。
2. 关注集团和业界的解决方案。在业务中发现问题，做解决方案的时候，我们很容易陷入自己的设计中，一脑子地想把所有东西都自己做出来，但投入会非常大，产出的价值是否一样大呢？不知道。大部分情况下，你想做的，在 ATA 能搜到，前人踩的坑，或者已有的成熟的解决方案，只要你去沟通去接触，就可以轻松地接进来，为什么要花大量的时间去造轮子呢？可以借力的地方，就去借力吧，把时间剩下来，做你的解决方案中更核心更有价值的事情。

技术深度

一聊到"技术深度"，可能很自然地会认为是在某项技术上挖得很深，或者解决了一个业界公认难度很高的技术难题，但这只是"技术深度"的其中一部分：

1. 体系化 / 系统化

   体系化思维是认识事物的一种方式，在面对问题的时候，能够针对复杂的问题，列出关键的要素和解决方法，将散乱无序的问题，变得逻辑清晰，有章可循。

   在问题的定位和解决的体现，从表象到本质，拆解出造成问题背后的原因，针对性地去解决本质的原因，而非治标不治本，有解决方案有节奏地解决。
2. 全链路

   除了前端的部分，向前向后的技术栈，还能挖多深。
3. 单点技术挑战

   在某个技术挑战上，你的思考和解决方案是怎样的。

技术与业务共赢

真正有突破性的、带来重大价值的业务成果必然伴随着技术上的深入乃至创新，所以在做业务成果的时候，一定会有让我们增加技术深度的场景。

Q: 如何看待 Web 开发构建工具 Vite?

作者 | 木偶
出品 | 淘系技术

> Vite 使用 native ES module imports 实现组件、样式、模板加载以及热更新，这会不会成为后续工具链的发展方向?
> https://github.com/vuejs/vite github.com
> 类似工具：open-wc/open-wc github.com

vite 是什么

vite —— 一个由 vue 作者尤雨溪开发的 web 开发工具，它具有以下特点：

1. 快速的冷启动
2. 即时的模块热更新
3. 真正的按需编译

从作者在微博上的发言：

Vite，一个基于浏览器原生 ES imports 的开发服务器。利用浏览器去解析 imports，在服务器端按需编译返回，完全跳过了打包这个概念，服务器随起随用。同时不仅有 Vue 文件支持，还搞定了热更新，而且热更新的速度不会随着模块增多而变慢。针对生产环境则可以把同一份代码用 rollup 打。虽然现在还比较粗糙，但这个方向我觉得是有潜力的，做得好可以彻底解决改一行代码等半天热更新的问题。

从中可以看出 vite 主要特点是基于浏览器 native 的 ES module (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import) 来开发，省略打包这

个步骤，因为需要什么资源直接在浏览器里引入即可。

基于浏览器 ES module 来开发 web 应用也不是什么新鲜事，snowpack 也基于此，不过目前此项目社区中并没有流行起来，vite 的出现也许会让这种开发方式再火一阵子。

有趣的是 vite 算是革了 webpack 的命了（生产环境用 rollup），所以 webpack 的开发者直接喊大哥了...

（作者注：本文完成于 vite 早期时候，vite 部分功能和原理有更新。）

vite 的使用方式

同常见的开发工具一样，vite 提供了用 npm 或者 yarn 一建生成项目结构的方式，使用 yarn 在终端执行：

```
$ yarn create vite-app <project-name>
$ cd <project-name>
$ yarn
$ yarn dev
```

即可初始化一个 vite 项目（默认应用模板为 vue3.x），生成的项目结构十分简洁：

```
|___node_modules
|___App.vue // 应用入口
|___index.html // 页面入口
|___vite.config.js // 配置文件
|___package.json
```

执行 `yarn dev` 即可启动应用 。

vite 启动链路

命令解析

这部分代码在 src/node/cli.ts 里，主要内容是借助 minimist —— 一个轻量级的命令解析工具解析 npm scripts，解析的函数是 `resolveOptions` ，精简后的代码片段如下。

```
function resolveOptions() {
    // command 可以是 dev/build/optimize
    if (argv._[0]) {
        argv.command = argv._[0];
    }
    return argv;
}
```

拿到 options 后，会根据 `options.command` 的值判断是执行在开发环境需要的 runServe 命令或生产环境需要的 runBuild 命令。

```
if (!options.command || options.command === 'serve') {
    runServe(options)
 } else if (options.command === 'build') {
    runBuild(options)
```

```
} else if (options.command === 'optimize') {
    runOptimize(options)
}
```

在 runServe 方法中，执行 server 模块的创建开发服务器方法，同样在 runBuild 中执行 build 模块的构建方法。

最新的版本中还增加了 optimize 命令的支持，关于 optimize 做了什么，我们下文再说。

server

这部分代码在 src/node/server/index.ts 里，主要暴露一个 createServer 方法。

vite 使用 koa 作 web server，使用 clmloader 创建了一个监听文件改动的 watcher，同时实现了一个插件机制，将 koa-app 和 watcher 以及其他必要工具组合成一个 context 对象注入到每个 plugin 中。

context 组成如下：

plugin 依次从 context 里获取上面这些组成部分，有的 plugin 在 koa 实例添加了几个 middleware，有的借助 watcher 实现对文件的改动监听，这种插件机制带来的好处是整个应用结构清晰，同时每个插件处理不同的事情，职责更分明，

plugin

上文我们说到 plugin，那么有哪些 plugin 呢？它们分别是：

- 用户注入的 plugins —— 自定义 plugin
- hmrPlugin —— 处理 hmr
- htmlRewritePlugin —— 重写 html 内的 script 内容
- moduleRewritePlugin —— 重写模块中的 import 导入
- moduleResolvePlugin ——获取模块内容
- vuePlugin —— 处理 vue 单文件组件
- esbuildPlugin —— 使用 esbuild 处理资源
- assetPathPlugin —— 处理静态资源
- serveStaticPlugin —— 托管静态资源
- cssPlugin —— 处理 css/less/sass 等引用
- ...

我们来看 plugin 的实现方式，开发一个用来拦截 json 文件 plugin 可以这么实现：

```
interface ServerPluginContext {
  root: string
  app: Koa
  server: Server
  watcher: HMRWatcher
  resolver: InternalResolver
  config: ServerConfig
}

type ServerPlugin = (ctx:ServerPluginContext) => void;

const JsonInterceptPlugin:ServerPlugin = ({app})=>{
    app.use(async (ctx, next) => {
      await next()
      if (ctx.path.endsWith('.json') && ctx.body) {
        ctx.type = 'js'
        ctx.body = `export default json`
      }
  })}
```

vite 背后的原理都在 plugin 里，这里不再一一解释每个 plugin 的作用，会放在下文背后的原理中一并讨论。

build

这部分代码在 node/build/index.ts 中，build 目录的结构虽然与 server 相似，同样导出一个 build 方法，同样也有许多 plugin，不过这些 plugin 与 server 中的用途不一样，因为 build 使用了 rollup ，所以这些 plugin 也是为 rollup 打包的 plugin ，本文就不再多提。

vite 运行原理

ES module

要了解 vite 的运行原理，首先要知道什么是 ES module，目前流览器对其的支持如下：

主流的浏览器（IE11 除外）均已经支持，其最大的特点是在浏览器端使用 exportimport 的方式导入和导出模块，在 script 标签里设置 type="module"，然后使用模块内容。

```
<script type="module">
  import { bar } from './bar.js'
</script>
```

当 html 里嵌入上面的 script 标签时候，浏览器会发起 http 请求，请求 htttp server 托管的 bar.js ，在 bar.js 里，我们用 named export 导出 bar 变量，在上面的 script 中能获取到 bar 的定义。

```
// bar.js
export const bar = 'bar';
```

在 vite 中的作用

打开运行中的 vite 项目，访问 view-source 可以发现 html 里有段这样的代码：

```
<script type="module">
    import { createApp } from '/@modules/vue'
    import App from '/App.vue'
    createApp(App).mount('#app')
</script>
```

从这段代码中，我们能 get 到以下几点信息：

- 从 `http://localhost:3000/@modules/vue` 中获取 `createApp` 这个方法。
- 从 `http://localhost:3000/App.vue` 中获取应用入口。
- 使用 `createApp` 创建应用并挂载节点。

createApp 是 vue3.X 的 api，只需知道这是创建了 vue 应用即可，vite 利用 ES module，把“构建 vue 应用”这个本来需要通过 webpack 打包后才能执行的代码直接放在浏览器里执行，这么做是为了：

1. 去掉打包步骤
2. 实现按需加载

去掉打包步骤

打包的概念是开发者利用打包工具将应用各个模块集合在一起形成 bundle，以一定规则读取模块的代码——以便在不支持模块化的浏览器里使用。

为了在浏览器里加载各模块，打包工具会借助胶水代码用来组装各模块，比如 webpack 使用 map 存放模块 id 和路径，使用 `__webpack_require__` 方法获取模块导出。

vite 利用浏览器原生支持模块化导入这一特性，省略了对模块的组装，也就不需要生成 bundle，所以打包这一步就可以省略了。

实现按需打包

前面说到，webpack 之类的打包工具会将各模块提前打包进 bundle 里，但打包的过程是静态的——不管某个模块的代码是否执行到，这个模块都要打包到 bundle 里，这样的坏处就是随着项目越来越大打包后的 bundle 也越来越大。

开发者为了减少 bundle 大小，会使用动态引入 import() 的方式异步的加载模块（被引入模块依然需要提前打包），又或者使用 tree shaking 等方式尽力的去掉未引用的模块，然而这些方式都不如 vite 的优雅，vite 可以只在需要某个模块的时候动态（借助 import()）的引入它，而不需要提前打包，虽然只能用在开发环境，不过这就够了。

vite 如何处理 ESM

既然 vite 使用 ESM 在浏览器里使用模块，那么这一步究竟是怎么做的？

上文提到过，在浏览器里使用 ES module 是使用 http 请求拿到模块，所以 vite 必须提供一个 web server 去代理这些模块，上文中提到的 koa 就是负责这个事情，vite 通过对请求路径的劫持获取资源的内容返回给浏览器，不过 vite 对于模块导入做了特殊处理。

@modules 是什么?

通过工程下的 index.html 和开发环境下的 html 源文件对比，发现 script 标签里的内容发生了改变，由

```
<script type="module">
    import { createApp } from 'vue'
    import App from '/App.vue'
    createApp(App).mount('#app')
</script>
```

变成了:

```
<script type="module">
    import { createApp } from '/@modules/vue'
    import App from '/App.vue'
    createApp(App).mount('#app')
</script>
import
```

1. 在 koa 中间件里获取请求 body
2. 通过 es-module-lexer 解析资源 ast 拿到 import 的内容
3. 判断 import 的资源是否是绝对路径，绝对视为 npm 模块
4. 返回处理后的资源路径: `"vue" => "/@modules/vue"`

这部分代码在 serverPluginModuleRewrite 这个 plugin 中，

为什么需要 @modules?

如果我们在模块里写下以下代码的时候，浏览器中的 esm 是不可能获取到导入的模块内容的:

```
import vue from 'vue'
```

因为 vue 这个模块安装在 node_modules 里，以往使用 webpack，webpack 遇到上面的代码，会帮我们做以下几件事：

- 获取这段代码的内容
- 解析成 AST
- 遍历 AST 拿到 import 语句中的包的名称
- 使用 enhanced-resolve 拿到包的实际地址进行打包

但是浏览器中 ESM 无法直接访问项目下的 node_modules，所以 vite 对所有 import 都做了处理，用带有 @modules 的前缀重写它们。

从另外一个角度来看这是非常比较巧妙的做法，把文件路径的 rewrite 都写在同一个 plugin 里，这样后续如果加入更多逻辑，改动起来不会影响其他 plugin，其他 plugin 拿到资源路径都是 @modules，比如说后续可能加入 alias 的配置：就像 webpack alias 一样：可以将项目里的本地文件配置成绝对路径的引用。

怎么返回模块内容

在下一个 koa middleware 中，用正则匹配到路径上带有 @modules 的资源，再通过 `require('xxx')` 拿到 包的导出返回给浏览器。

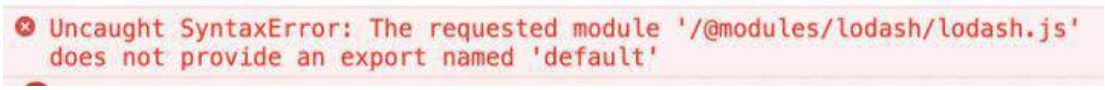
以往使用 webpack 之类的打包工具，它们除了将模块组装到一起形成 bundle，还可以让使用了不同模块规范的包互相引用，比如：

- ES module (esm) 导入 cjs
- CommonJS (cjs) 导入 esm
- dynamic import 导入 esm
- dynamic import 导入 cjs

关于 es module 的坑可以看这篇文章(https://zhuanlan.zhihu.com/p/40733281)。

起初在 vite 还只是为 vue3.x 设计的时候，对 vue esm 包是经过特殊处理的，比如：需要 @vue/runtime-dom 这个包的内容，不能直接通过 require('@vue/runtime-dom') 得到，而需要通过 require('@vue/runtime-dom/dist/runtime-dom.esm-bundler.js' 的方式，这样可以使得 vite 拿到符合 esm 模块标准的 vue 包。

目前社区中大部分模块都没有设置默认导出 esm，而是导出了 cjs 的包，既然 vue3.0 需要额外处理才能拿到 esm 的包内容，那么其他日常使用的 npm 包是不是也同样需要支持？答案是肯定的，目前在 vite 项目里直接使用 lodash 还是会报错的。



不过 vite 在最近的更新中，加入了 optimize 命令，这个命令专门为解决模块引用的坑而开发，例如我们要在 vite 中使用 lodash，只需要在 vite.config.js（vite 配置文件）中，配置 optimizeDeps 对象，在 include 数组中添加 lodash。

```
// vite.config.js
module.exports = {
  optimizeDeps: {
    include: ["lodash"]
  }
}
```

这样 vite 在执行 runOptimize 的时候中会使用 roolup 对 lodash 包重新编译，将编译成符合 esm 模块规范的新的包放入 node_modules 下的 .vite_opt_cache 中，然后配合 resolver 对 lodash 的导入进行处理：使用编译后的包内容代替原来 lodash 的包的内容，这样就解决了 vite 中不能使用 cjs 包的问题，这部分代码在 depOptimizer.ts 里。

不过这里还有个问题，由于在 depOptimizer.ts 中，vite 只会处理在项目下 package.json 里的 dependencies 里声明好的包进行处理，所以无法在项目里使用：

```
import pick from 'lodash/pick'
```

的方式单使用 pick 方法，而要使用：

```
import lodash from 'lodash'
lodash.pick()
```

的方式，这可能在生产环境下使用某些包的时候对 bundle 的体积有影响。

返回模块的内容的代码在：serverPluginModuleResolve.ts 这个 plugin 中。

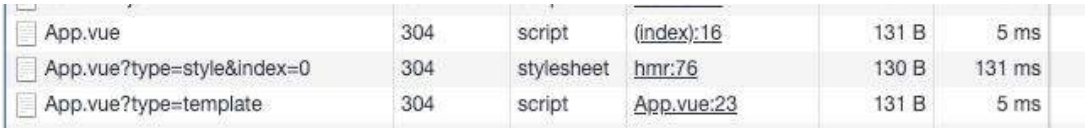
vite 如何编译模块

最初 vite 为 vue3.x 开发，所以这里的编译指的是编译 vue 单文件组件，其他 es 模块可以直接导入内容。

SFC

vue 单文件组件（简称 SFC） 是 vue 的一个亮点，前端届对 SFC 褒贬不一，个人看来，SFC 是利大于弊的，虽然 SFC 带来了额外的开发工作量，比如为了解析 template 要写模板解析器，还要在 SFC 中解析出逻辑和样式，在 vscode 里要写 vscode 插件，在 webpack 里要写 vue-loader，但是对于使用方来说可在一个文件里可以同时写 template、js、style，省了各文件互相跳转。

与 vue-loader 相似，vite 在解析 vue 文件的时候也要分别处理多次，我们打开浏览器的 network，可以看到：



App.vue	304	script	(index):16	131 B	5 ms
App.vue?type=style&index=0	304	stylesheet	hmr:76	130 B	131 ms
App.vue?type=template	304	script	App.vue:23	131 B	5 ms

1 个请求的 query 中什么都没有，另 2 个请求分别通过在 query 里指定了 type 为 style 和 template。

先来看看如何将一个 SFC 变成多个请求，我们从第一次请求开始分析，简化后的代码如下：

```
function vuePlugin({app}){
  app.use(async (ctx, next) => {
    if (!ctx.path.endsWith('.vue') && !ctx.vue) {
      return next()
    }

    const query = ctx.query
    // 获取文件名称
    let filename = resolver.requestToFile(publicPath)

    // 解析器解析 SFC
    const descriptor = await parseSFC(root, filename, ctx.body)
    if (!descriptor) {
      ctx.status = 404
      return
    }
    // 第一次请求 .vue
    if (!query.type) {
      if (descriptor.script && descriptor.script.src) {
        filename = await resolveSrcImport(descriptor.script, ctx, resolver)
      }
      ctx.type = 'js'
      // body 返回解析后的代码
      ctx.body = await compileSFCMain(descriptor, filename, publicPath)
    }
```

```
    // ...
}
```

在 compileSFCMain 中是一段长长的 generate 代码：

```
function compileSFCMain(descriptor, filePath: string, publicPath: string) {
  let code = ''
  if (descriptor.script) {
    let content = descriptor.script.content
    code += content.replace(`export default`, 'const __script =')
  } else {
    code += `const __script = {}`
  }

  if (descriptor.styles) {
    code += `\nimport { updateStyle } from "${hmrClientId}"\n`
    descriptor.styles.forEach((s, i) => {
      const styleRequest = publicPath + `?type=style&index=${i}`
      code += `\nupdateStyle("${id}-${i}", ${JSON.stringify(styleRequest)})`
    })
    if (hasScoped) {
      code += `\n__script.__scopeId = "data-v-${id}"`
    }
  }

  if (descriptor.template) {
    code += `\nimport { render as __render } from ${JSON.stringify(
      publicPath + `?type=template`
    )}`
    code += `\n__script.render = __render`
  }
  code += `\n__script.__hmrId = ${JSON.stringify(publicPath)}`
  code += `\n__script.__file = ${JSON.stringify(filePath)}`
  code += `\nexport default __script`
```

```
  return code
}
```

直接看 generate 后的代码：

```
import { updateStyle } from "/vite/hmr"
updateStyle("c44b8200-0", "/App.vue?type=style&index=0")
__script.__scopeId = "data-v-c44b8200"
import { render as __render } from "/App.vue?type=template"
__script.render = __render
__script.__hmrId = "/App.vue"
__script.__file = "/Users/muou/work/playground/vite-app/App.vue"
export default __script
```

出现了 `vite/hmr` 的导入，`vite/hmr` 具体内容我们下文再分析，从这段代码中可以看到，对于 style vite 使用 `updateStyle` 这个方法处理，`updateStyle` 内容非常简单，这里就不贴代码了，就做了 1 件事：通过创建 style 元素，设置了它的 innerHtml 为 css 内容。

这两种方式都会使得浏览器发起 http 请求，这样就能被 koa 中间件捕获到了，所以就形成了上文我们看到的：对一个 .vue 文件处理三次的情景。

这部分代码在：serverPluginVue 这个 plugin 里。

CSS

如果在 vite 项目里引入一个 sass 文件会怎么样？

最初 vite 只是为 vue 项目开发，所以并没有对 css 预编译的支持，不过随着后续的几次大更新，在 vite 项目里使用 sass/less 等也可以跟使用 webpack 的时候一样优雅了，只需要安装对应的 css 预处理器即可。

在 cssPlugin 中，通过正则：`/(.+).(less|sass|scss|styl|stylus)$/` 判断路径是否需要 css 预编译，如果命中正则，就借助 cssUtils 里的方法借助 postcss 对要导入的 css 文件编译。

vite 热更新的实现

上文中出现了 vite/hmr ，这就是 vite 处理热更新的关键，在 serverPluginHmr plugin 中，对于 path 等于 vite/hmr 做了一次判断：

```
app.use(async (ctx, next) => {
  if (ctx.path === '/vite/hmr') {
      ctx.type = 'js'
      ctx.status = 200
      ctx.body = hmrClient
  }
}
```

hmrClient 是 vite 热更新的客户端代码，需要在浏览器里执行，这里先来说说通用的热更新实现，热更新一般需要四个部分：

1. 首先需要 web 框架支持模块的 rerender/reload。
2. 通过 watcher 监听文件改动。
3. 通过 server 端编译资源，并推送新模块内容给 client。
4. client 收到新的模块内容，执行 rerender/reload。

vite 也不例外同样有这四个部分，其中客户端代码在：client.ts 里，服务端代码在 serverPluginHmr 里，对于 vue 组件的更新，通过 vue3.x 中的 `HMRRuntime` 处理的。

client 端

在 client 端， WebSocket 监听了一些更新的类型，然后分别处理，它们是：

- vue-reload —— vue 组件更新：通过 import 导入新的 vue 组件，然后执行 HMR Runtime.reload
- vue-rerender —— vue template 更新：通过 import 导入新的 template ，然后执行 HMRRuntime.rerender
- vue-style-update —— vue style 更新：直接插入新的 stylesheet
- tyle-update —— css 更新：document 插入新的 stylesheet
- style-remove —— css 移除：document 删除 stylesheet
- js-update —— js 更新：直接执行
- full-reload —— 页面 roload：使用 window.reload 刷新页面

server 端

在 server 端，通过 watcher 监听页面改动，根据文件类型判断是 js Reload 还是 Vue Reload：

```
watcher.on('change', async (file) => {
    const timestamp = Date.now()
    if (file.endsWith('.vue')) {
      handleVueReload(file, timestamp)
    } else if (
      file.endsWith('.module.css') ||
      !(file.endsWith('.css') || cssTransforms.some((t) => t.test(file, {})))
    ) {
      // everything except plain .css are considered HMR dependencies.
      // plain css has its own HMR logic in ./serverPluginCss.ts.
      handleJSReload(file, timestamp)
    }
  })
```

在 handleVueReload 方法里，会使用解析器拿到当前文件的 template/script/style ，并且与缓存里的上一次解析的结果进行比较，如果 template 发生改变就执行 vue-rerender，如果 style 发生改变就执行 vue-style-update，简化后的逻辑如下：

```
async function handleVueReload(
    file
    timestamp,
    content
  ) {
    // 获取缓存
    const cacheEntry = vueCache.get(file)

    // 解析 vue 文件
    const descriptor = await parseSFC(root, file, content)
    if (!descriptor) {
      // read failed
      return
    }
    // 拿到上一次解析结果
    const prevDescriptor = cacheEntry && cacheEntry.descriptor

    // 设置刷新变量
    let needReload = false // script 改变标记
    let needCssModuleReload = false // css 改变标记
    let needRerender = false // template 改变标记

    // 判断 script 是否相同
    if (!isEqual(descriptor.script, prevDescriptor.script)) {
      needReload = true
    }

     // 判断 template 是否相同
    if (!isEqual(descriptor.template, prevDescriptor.template)) {
      needRerender = true
    }

    // 通过 send 发送 socket
    if (needRerender){
      send({
```

```
          type: 'vue-rerender',
          path: publicPath,
          timestamp
        })
      }
    }
```

handleJSReload 方法则是根据文件路径引用，判断被哪个 vue 组件所依赖，如果未找到 vue 组件依赖，则判断页面需要刷新，否则走组件更新逻辑，这里就不贴代码了。

整体代码在 client.ts 和 serverPluginHmr.ts 里。

结语

本文分析了 vite 的启动链路以及背后的部分原理，虽然在短时间内 vite 不会替代 webpack，但是能够看到社区中多了一种方案还是很兴奋的，这也是我写下这篇文章的原因。

vite 更新的实在太快了，佩服尤大的勤奋和开源精神，短短一个月就加入了诸如 css 预编译/react 支持/通用 hmr 的支持，由于篇幅有限本文不再一一介绍这些新特性，这些新的特性等待读者朋友们自行去探讨了。

Q: 2020 年前端最火的技术是什么?

作者 | 卓风
出品 | 淘系技术

作为一个程序员，你觉得最火的前端技术将是什么?

我们知道，即便前端工程化能力已经成熟，但提高"生产效率"的问题永远是前进路上的重要历程。

作为一个严谨的程序员，不能说这个"最"字（毕竟也违反广告法)。我们结合阿里淘系前端的一些实践，聊聊对未来的技术展望。（个人视角）

展望：前端发展的未来 10 年（深水区）

解决业务问题不说，那么前端为什么要关注生产效率问题呢?

因为这直接与阿里的业务体量相关，阿里每一年的业务体量都是相比去年翻番的(比如出海、下沉、创新业务.....），所以如果生产力效率跟不上业务的发展节奏，那么在市场竞争上就不占优势，以 2019 年三四线下沉市场高度竞对的场景为例，如果前端撑不住业务发展的节奏，还是慢慢悠悠地搞生产，那么企业就很难占据市场了。

所以，每个前端身上背负的都是业务体量的成倍增加，如何能快速支撑住业务发展以及如何帮助业务突围和增长（2017 年手机出货量触顶下滑，移动端的自然用户增长红利达到顶峰，可以从《用户流量红利消退的下半场，淘宝如何保持高速增长?》（https://blog.csdn.net/Taobaojishu/article/details/101488512）便可感知到）是我们每时每刻都在思考的问题。

前端已到瓶颈!

我们知道，即便工程化能力已经成熟，但还是解决不了的问题就是"生产效率"的问题，试想：

假设 1 个中等水平的前端产出一张功能齐全的页面需要 1 周时间，1 个牛逼的前端可能只需 2 天时间；而即便都雇佣牛逼的前端，1 个前端单打独斗一周之内最多也就 4 张页面产出，如果仅是生产 10 张页面，那么雇佣 1~2 个牛逼的前端一周之内就搞定了，但如果是生产 100 张、1000 张页面呢？这个时候雇佣多少前端比较合适呢？高端人才的紧俏和招人成本的控制，都会导致厂内的前端的业务压力倍增。

解铃还须系铃人，所以业内开始不断地涌现 hardcode 向 lowcode 方向转变的提效热潮。不说外界，单以阿里为例，面向中后台、C 端、数据可视化方向的 lowcode 平台就层出不穷，虽说上手复杂度很高（毕竟解决问题的复杂度摆在那里，就像 Photoshop 一样），但也都在趋于成熟。

可这样就高枕无忧了嘛？其实并没有，因为业务的迭代速度太快了，即便有这些平台存在了，依然还是解决不了业务上的燃眉之急、前端效率问题依然是业内的瓶颈。

以我带的团队为例，我们服务的每一条线下的业务量和复杂度都是居高不下（每条线承接的是千万级流量，所以业务复杂度自然会高），除了日常产品迭代，每月至少有 1~2 次的营销活动同时进行，即便用了上述的 lowcode 产品，但还是解不了业务方频繁上诉要人的困局，甚至排期、砍需求这种传统小伎俩如今也对业务方没有药效了。

怎么办？一人难敌四手，更何况是一堆数都数不过来的手了。

前端如何突围?

要讲清楚这一块，我们换个视角看看。众所周知，市场是有清浊、淘汰机制的，任何一个行业都不是一成不变的，只要有先进的出现，那么就势必会将落伍的清理淘汰掉，而这个过程自变量仅是时间。

就像电商互联网兴起的那一刻，有多少实体从业者会意识到自己的饭碗会跟不上时代？就像移动端来临的那一刻，有多少公司及个人还在沉淫在 PC 的时代产物上，再后知后觉地意识到落后时已经被竞争者甩了好几条街了。

就像当 iOS、Android App 生态刚开始兴起阶段，不断地有客户端的人才在向市场输入，而今当 App 在市场饱和、用户分配在终端上安装的 App 数量有限，以及移动端 Web 和轻 App 技术的飞速发展等客观因素冲击下，客户端的从业者发现保住自己的饭碗越发的困难了。

就像 AI、区块链兴起，有一大批的算法从业者和新技术的创业公司输出到市场，而经过市场竞争的洗涤下，又发现算法人才饱和过剩、创业公司也死了一大片。

所以要看一个行业的未来发展怎样，就看这个行业的人才目前和未来在市场上被密集需要的地方在哪、规则最混浊或混乱的地方在哪。如果说这个行业的规则出奇地清晰、人才的供给又出奇的冷静，那么基本上来说，这个行业在市场的发展已经达到平衡状态，而能打破这种平衡重新建立平衡的也肯定是另外的行业的发展渗入。

所以回归到我们所处的前端行业，如今前端人才被需要的肯定是在互联网公司，尤其是大厂，因为业务发展需要，且被需要的很密集（劳动密集型产业），而且这个行业恰巧也是发展规则相对混乱的。为何混乱呢？一方面是因终端多元化趋势严重，比如智能穿戴设备和 IOT 智能家居、智慧医疗、智能建筑等新兴产业的市场冲击，另一方面是因业务的发展形态、发展规模、发展距离（国内到国外）等因素的影响，都导致着过去的终端的技术规则无法适应到新兴终端领域内，所以规则在变、技术在变、框架在变、从业者的领域也在变。

所以从这个角度看前端的职能领域只会越来越宽，人才的需求量只会越来越大，供给的能力要求只会越来越高。可以说这是市场对前端这个行业利好的信号，但同样也是对前端这个行业压力提醒的信号，如果在这个市场内的前端不能很好的解决市场压力问题，一旦有新兴技术手段形成的新生产力出现，那么前端的这个香饽饽的行业饭碗也就不保了。市场就是这样冷静残酷的，当市场出清淘汰一个行业的时候或许连一声招呼都不会打，没有为什么，这是发展的必须。

如今我们看清形势，再反观我们的生产力手段，可以说还是人肉劳动密集型的，就算招再牛逼的人才进来，如还是以这种的生产手段生产，那么早晚都会被淘汰，不管有多资深，哪怕是专家、研究员。所以前端发展到这个档口下，看似成熟，实则危机四伏。

我们需要反思，更需要一种全景视角的突破和自我革命。与其让别人革我们的命，那么真不如我们自己革自己命。所以接下来前端的发展势必会面临着一个最习以为常却又最为关键的挑战 —— 前端生产效率该如何翻倍的提升？

历史的经验告诉我们，一个行业的生产供给能力翻倍，那么一定跟这个行业的工艺手段脱不了关系。比如传统制造业制造一款鞋子、织一块布，都是人手工的，当这种供给达到瓶颈之后，就开始出现机械化来辅助人来生产，机械化达到一定程度就是自动化，自动化就可以完全脱离人工进行生产。

同样的道理，前端目前的生产工艺还是人肉的，即便有一定的程度的 lowcode 产品手段来辅助前端释放生产压力，但还是解决不了供不应求的问题，所以没有别的办法，只有一条路就是去人肉，改成完全自动化的生产手段，只有让供给能力远远超越需求的市场增长指数，那么才能彻底解决供不应求的问题。

那么，前端该如何将生产手段提升到自动化阶段呢？

首先，我们能想到的生产手段上肯定不能重度依赖人，那么剩下的也仅有机器，对于我们而言，肯定就是计算机了。

其次，我们要想的问题是该如何用计算机来解决我们所面临的生产问题，想到第一步不难，而最难的恰巧就是这一步。该怎么解呢（how）？

调研发现，市面上就 2 种形态的解决思路，一种就是堆人的 hardcode 方式，包括传统的组件化生态，也都停留在这个阶段上；再有一种解法是 lowcode 的方式，或者辅助自己或者辅助其他角色来做生产（换一句话来说就是生产关系转移到其他角色身上），这种方式在特定领域内能一定程度上提效，但一旦领域拓宽或稍有移植，就会面临着不适应，用它工作

量反而比 hardcode 增加很多。目前我们就在第 2 阶段，但生产效率问题还是非常突出。所以我选择的解法是 nocode，虽然这个词也不是我新创的，但这个词的涵义足以表达我对生产力供给能力提升下一个阶段的看法。而能帮助前端实现 nocode 解法的技术，一定就是 AI（准确来说是机器学习）。Why?

互联网的发展就是带来了海量的数据，依靠人脑已经无法去分析清楚一个行业的特征了，至少我们都是凡人大众，那种类似爱因斯坦的天才毕竟还是少见，不可能哪个行业都要等着爱因斯坦出现才能找到解决方案。所以凡人大脑做不到的事情我们就交给计算机来做，如今的云计算发展和 AI 发展，已经降低了我们应用 AI 来解决我们问题的门槛，所以入行 AI 也是迟早的事，不可能每天都蒙着眼睛装看不到，而且也一定程度上得承认 AI 比我们更聪明，所以逃避不了的事实我们干脆一些接受好了。AI 就是为海量数据和复杂问题而生的。

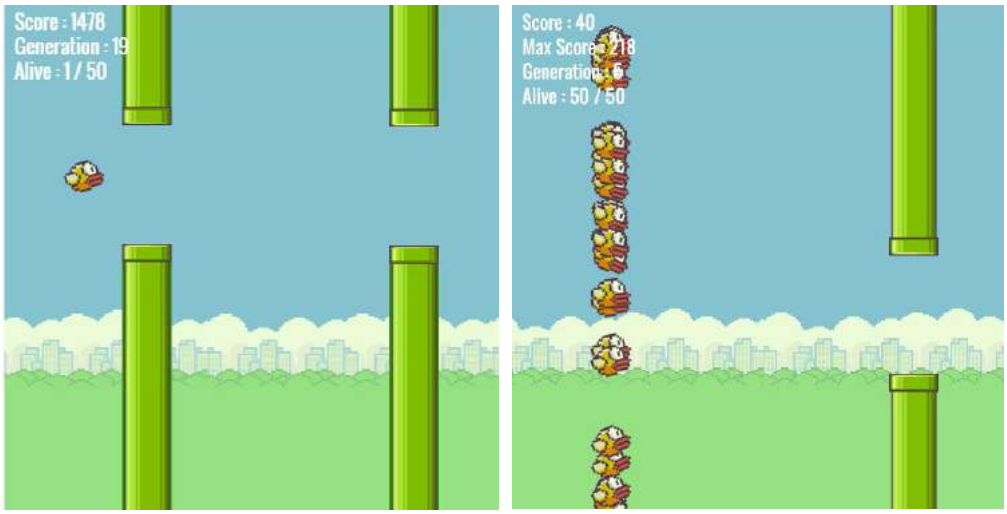
思维方式转变

那么前端究竟该怎样加持 AI 的能力呢?

这个问题对纯前端从业者来说很难，对算法从业者来说也很难，但对既懂前端又会算法的从业者来说就不难了。为了讲清楚这个问题，那么我首先来讲解下这两者解题思维的惯性差异是什么，帮助大家先从思想上进行转变，这样大家也就更易接受一些。

我们以一个具体的案例为例：当你的产品经理让你做一款类似下图这样过障的小游戏，管道洞口高度固定，且是匀速向左移动的，小鸟只会上下，同时受重力影响会跳动，你如何让这只小鸟自己会躲避障碍成功过关呢?

demo 地址：https://xviniette.github.io/FlappyLearning/



如果前端看到上面需求时，他的思维惯式一般是这样的：首先，要有一张画布，上面有小鸟、管道这两种对象（Object），小鸟对象中有 x、y、width、height、alive 等属性，x、y 代表它的水平和垂直位移，width、height 代表小鸟自身宽高，alive 代表小鸟的生死；管道对象中至少有 x、y、width、height、speed 5 个属性，x、y 代表管道的水平和垂直坐标位置，width、height 代表管道的宽高，speed 代表的是管道向左的移动速度。其次，要根据管道的移动速度给小鸟建立一个雷达预警机制，通过轮询的方式不断地探测正前方是否有障碍物，一旦有了就不断地在垂直方向上上下位移来做避障。最后，依次类推的方式达到终点。

如果算法看到上面需求时，他的思维惯式一般这样的：首先，要找一款模型，拿 network 为例，可以利用遗传算法来解决上面的问题，具体就是通过 50 代小鸟不断地尝试碰撞，将每一代失败的小鸟的基因记录下来，然后遗传给下一代，形成遗传记忆，这样小鸟就不会以失败的方式过障，以此类推，直到没有失败的小鸟出现，那么成功的基因就训练完毕，这样的一代小鸟就可以完全过障了。

大家可以看到，前端的给的解题思路的代码里是有具体交代小鸟应该怎样判断过障的；而看算法解题思路的代码里其实并没有具体教小鸟过障的代码逻辑，有的只是将一些特征和反馈抽取传递给到 network，而真正的过障判断过程是模型去做的。而这就是这两种思路的关键差别，准确说是 程序员 和 AI 算法工程师的思维惯式差别，程序员的脑海里有着"我能用代码定义世界"的思维，而 AI 算法工程师脑海里有着"我该用什么样的数据训练模式让模

型自己尽快掌握对错”的思维。前者是一种由自己来解问题的主观视角，所以写的代码纯粹是翻译给计算机要怎么去解这个问题；后者是一种由机器来替我解决问题的客观视角，所以写的代码纯粹是怎样把问题抛给计算机，并告知输入及结果对错，至于计算机是怎么解决这个问题的过程和规律算法工程师都不关心，只关心结果。

看到这里，想必大家对 2 种思维模式有了一个切身的体感，如果还有不太理解的，也可以看 甄子 老师的《前端智能化—思维转变之路》（https://juejin.im/post/5e7d8b87f265da79861b7e0e）文章，这更是对这个思维差异做了更深入的介绍。

重新审视求解

那么既然知道了 2 者的差异，我们就可以将前端领域内遇到的生产效率问题以最新的视角重新进行审视了。

前端，里面的关键字是“端”，所谓的“前”就是交代离用户最近的地方。所以用户接触到的终端（包括各种异形屏的、没屏幕的仅有传感器的终端等等）上面所呈现的任何人机交互内容（可视觉传达、听觉传达、肢体传达、甚至可能嗅觉传达等等）都可以认为是前端职责范围内的工作。面对这种形式多样的终端，要想快速产出人机交互的内容，我们用 AI 该怎么做呢？

鉴于话题有点大，我们还是聚焦在 Web 页面上（其他的以此类推），如何借助 AI 实现高效地生产呢？

一种思路是，首先，聚焦在网页上能呈现的内容形态看看到底有哪几种（空间轴上的语言），比如文字、图片、视频（视频可以理解为图片的逐帧动画，加上音频）、音频；然后，我们再看下网页上什么样的内容是经常变化的（时间轴上无序状态），什么样的内容是通过交互方式产生变化的（时间轴上的有序状态）。最后，我们的生产策略是，优先考虑将一组时间轴上的训练数据喂给一个模型让它识别出时间轴上的变化内容，然后再借助 CV 或 NLP 模型针对变化的内容进行实体识别（实体识别可能具体到一系列的模型存在，比如细化到商品图识别模型），再然后借助另外的 CV 或 NLP 模型来识别时间轴上不变的内容（往往

这部分内容就是页面布局和容器框架），再通过一系列实体识别模型来做页面结构代码上的映射（高维空间向量余弦值相等）。理论上来说，如有大量的训练样本数据，那么模型针对时间轴上的有序状态（即事件响应）也是可以慢慢自己学习出规律出来的。

上面这种思路是纯算法的思路，其中没有借助前端的任何思维模式来影响，但具体效果怎样和实施难度上有多大呢，目前还不好说，至少我们自己也还没开始这种尝试。

也许上面思维未来是对的，但今天来说，前端还没有准备好，还在一步步进行思维上的转变和迭代，这的确是需要一个过程。而且机器学习也不是万能的，它受模型的制约因素很大，而模型往往也是一种算力的象征。我们可以把机器学习比作是一个拥有高复杂度并行密集计算能力（高维空间上的矩阵计算）的统计学计算器，而模型就是这款计算器的内核。也许它能在背后计算出这样的宇宙规律，但至少也是进行了深度计算的，而这种深度的计算需要的就是海量的样本。而样本就是这款计算器内核塑造成型的灵魂，但这种海量样本的制造工作也绝非是一朝一夕依靠一个软件工程出身的技术人员搞得定的。样本本身就是数据，所以一定要有存量的数据才会有往深度学习方向上发展的可能性。否则人肉制造的样本，要不质量太差（不够客观）、要么就是量的规模不够。当然，也可以先把计算器搭起来，至于样本可以随着时间进行积累，这样的办法也不是不可以，就是等待的时日可能比较长，没法立竿见影收到奇效。

所以，针对商业行为来说，我们至少得有 2 套方案，一套是长远的（如上的方案）的准备，一套是短期眼前的方案。如果做短期的，就借助规则系统 + 机器学习的混合方式来做方案。但不管哪种，样本问题都是要解的。2 套方案也是 2 种选择，也许你还有第 3 种选择，都是选择，所以多与少没有什么差别，只是看能在选择之后投入多少和坚持多久。这种投入就涉及到知识和技能的储备了，所以前端想解决问题，还是得尽快上手机器学习。至于具体怎么上手在此就不做过多介绍了，网上的课程有很多，也可以看西瓜书上手，但关键是动手。可以先从 CV 领域入手，NLP 工程对个人来说单机部署有点难，得借助云（比如谷歌的 TPU 平台）。

前端智能化技术大图

长远来看，前端 + AI 的这种前端智能化方向肯定是持续存在的，前端也会因为 AI 能力的加入，会产生诸多不一样的生产力变化。这种变化可能是阶段性的，也可能是终极的，总之生产力会慢慢向计算机身上过渡，前端做的工作是驱动这一切的更深层工作。这个方向没有退路，也绕不过去（专家系统不可能无敌），所以要解的问题直到彻底解决为止。

智能化的 10 年

最后，我还是带着大家一起展望一下前端智能化的未来 10 年（以如今的互联网发展节奏看，或许下面的 5 年内就达到也是有可能的）：

- 2~3 年内，前端智能化从业者数量翻倍，AI 在前端领域内或多或少有一些产品形态上的应用，终端开始浮现各种前端机器学习框架，用户产品在智能化体验方面的设计也有对应的倾斜，社区上也开始浮现出各种前端智能化的工程框架、训练框架和 AI 平台；

- 3~5 年内，前端智能化从业者数量继续增长，传统前端已经被淘汰，前端领域内智能化在特定领域内小有成绩，可以解决特定领域内的一些业务或人力生产效率问题，终端智能体验会趋渐成熟，给用户带来的沉浸式体验增强，线上线下无屏化无差异体验趋近相同，社区上开始开源一些前端的智能化产品；
- 5~10 年内，前端智能化从业市场已趋近饱和，端智能体验设计者应该在市场上比较吃香（如沉浸式人机对抗游戏等），nocode 问题应该能彻底解决，但同时带来其他的消费诉求，可能会引入新的领域难题。

结语

如今危机四伏，未来无限可期，所以大家还是要未雨绸缪，提前做好准备。如今前端智能化是新赛道，如果想直接参加前端智能化第一线的，也可以通过下面这种方式加入我们。

团队产品

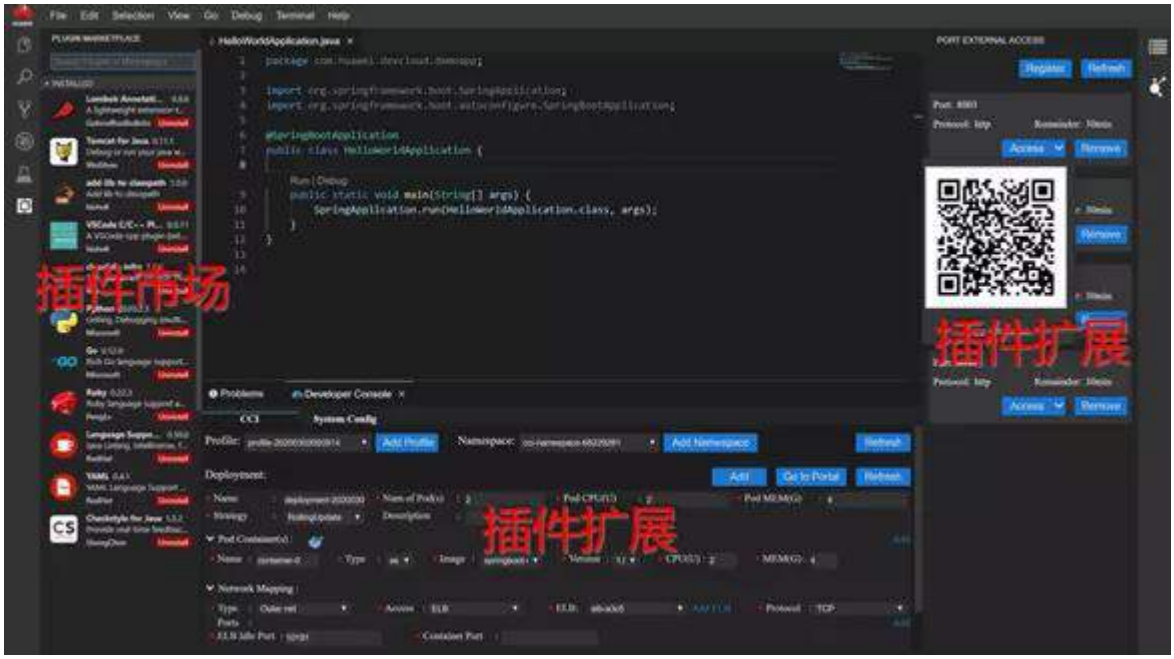
- imgcook —— 由设计稿一键智能生成代码（https://www.imgcook.com/）
- pipcook —— 前端算法工程框架，前端应用机器学习的最后一公里（https://github.com/alibaba/pipcook）

Q: 如何看待华为云 Cloud IDE 和 VSCode online 高度相似?

作者 | 柳千

出品 | 淘系技术

为什么大家做 Cloud IDE 的时候都会基于 VSCode/类 VSCode 这一套体系
在此之上各家可以做出什么样的亮点?



来源: https://mp.weixin.qq.com/s/K1XEWZFASspMpNppaM2g_A

为了更严谨一些，我将原本的问题从“如何看待华为云 Cloud IDE 是换 LOGO 的 VSCode Online”更正为“如何看待华为云 Cloud IDE 和 VSCode online 高度相似?”

参见 : https://support.huaweicloud.com/usermanual-cloudide/start_ide_instance.html

首先关于题目中「华为云 Cloud IDE 和 VSCode online 高度相似」这一点，事实上华为云 Cloud IDE 是基于 eclipse-theia 项目搭建的在线 IDE 服务。与 VS Code online 高度相似是因为 Theia 自身就是基于 VS Code 及其插件生态的。Theia 定位是一个 Web/桌面端的 IDE 底层框架，任何人都可以基于 Theia 搭建出自己的 IDE，社区较为流行的 gitpod 就是最早基于 Theia 搭建的在线 IDE 服务。

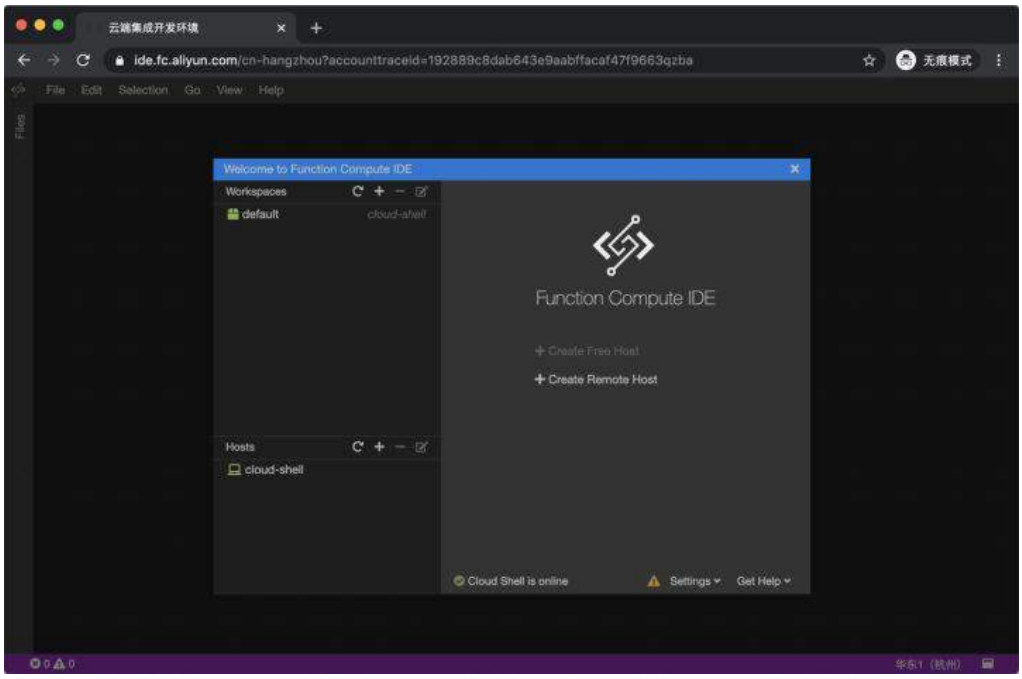
所以严格来说，华为云 Cloud IDE 真正高度相似的是 gitpod。

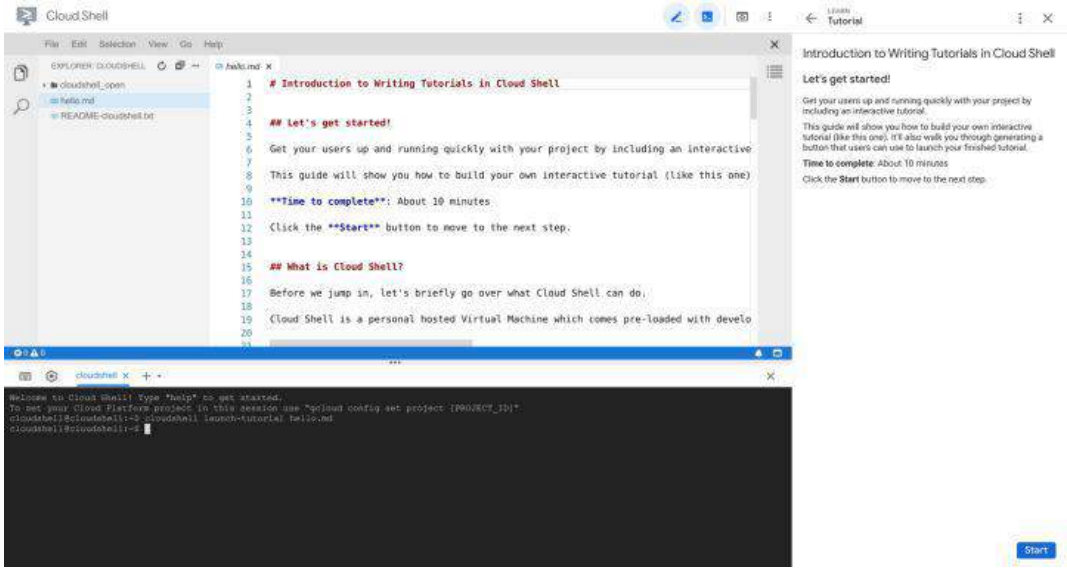
可以说 Theia 使用了 VS Code 众多核心技术，包括 Monaco Editor、Language Server Protocol、Debug Adapter Protocol、以及 VS Code 扩展协议。

这篇 Multi-Language IDE Implemented in JS 是 Theia 的架构文档，其中有提到核心的 Language support 以及 Debugging suport 都是基于 VS Code 生态的 LSP/DAP 协议实现的。早期 Theia 内核集成了 Node.js、Java 等主流语言的语言服务及调试器实现。但在 Theia 架构支持 VS Code 插件后已经可以无缝继承大部分 VS Code 语言相关的插件了。

同时由于 Theia 本身模块化的架构对于二次开发非常友好，熟悉一下 Theia 的 DI 体系以及基本的模块划分即可在 Theia 上层做定制化开发。 阿里云 Function Compute IDE 、Google Cloud Shell 以及题目中的华为云 Cloud IDE 等都是基于 Theia 的在线 IDE 服务。

值得注意的是 Theia 目前已经成为了 Eclipse 基金会项目，另一款老牌在线 IDE 服务 Eclipse Che 最新版本 Eclipse Che 7 也同样使用 Theia 作为其底层框架。



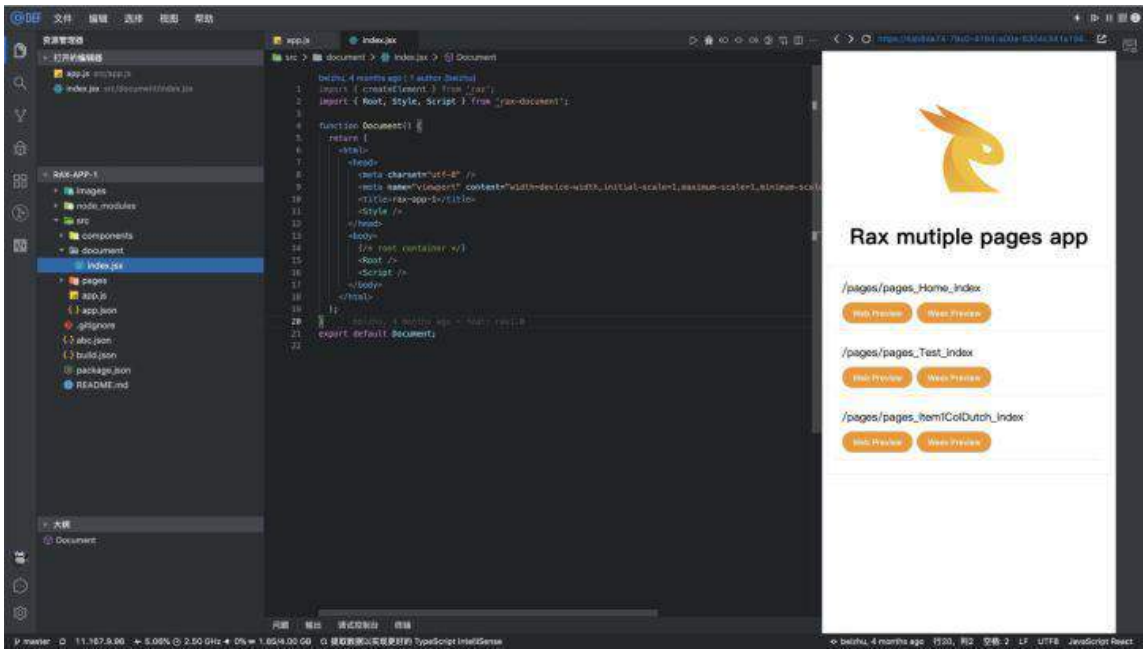


一点私货

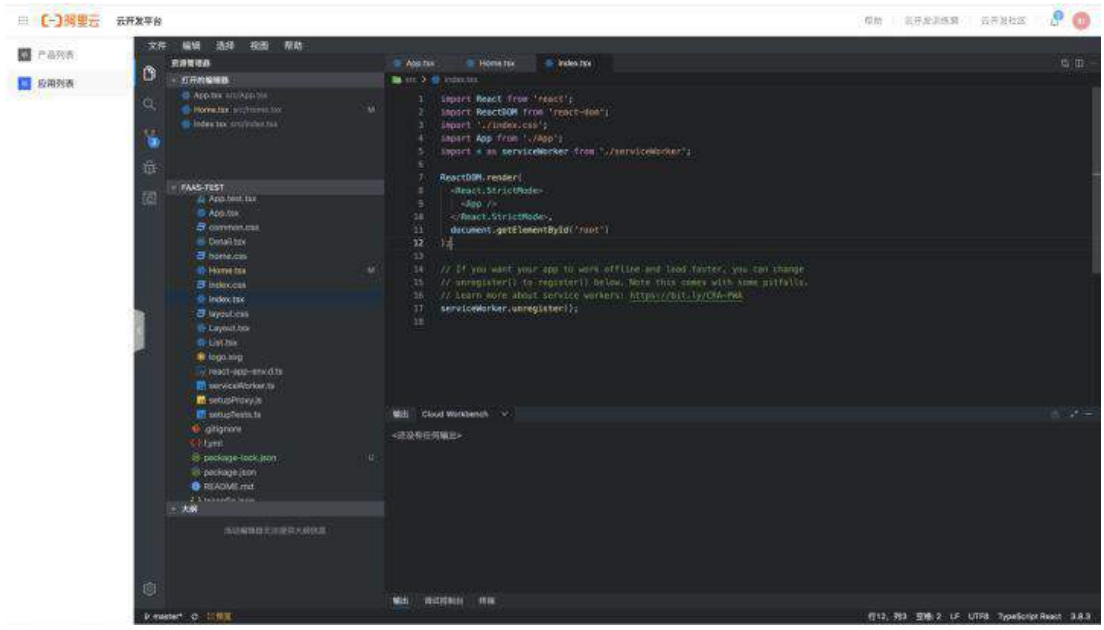
随着 2019 年 VS Online 重磅发布，月初更名为 VS Codespace，再到前几天 GitHub 推出 GitHub Codespace。坐拥前端第一编辑器 Visual Studio Code (以及宇宙第一 IDE Visual Studio)的微软终于在收购 GitHub 2 年以后的今天将这两大开发者最青睐的产品双剑合璧。

前者是当下最热门的编辑器(没有之一)，而后者是最大的同性交友平台(雾)。如果说 5 年前提到 IDE 还是一个玩具级别的产品的话，那么今年则是 WebIDE 大爆发的一年。

KAITIAN IDE 作为阿里前端统一的 IDE 框架，目前已经在包括淘系、蚂蚁、阿里云等众多、BU 的业务中落地。淘系的 DEF WebIDE 也承接了模块开发、FaaS 以及中后台研发等场景。



同样基于 KAITIAN 的阿里云云开发平台 CloudIDE 也在上个月正式上线，结合阿里 FaaS 体系提供全链路的云端 FaaS 一体化研发体验。



与 Theia 类似的是，KAITIAN 定位也是一个底层框架，基于模块化的架构，支持 Web/Electron 端集成。同时也采用了与 VS Code 相同的插件架构，这使得 KAITIAN 可以运行主流的 VS Code 插件，目前 KAITIAN 的 VS Code 插件 API 覆盖率已经达到 96% 以上。

作为服务于集团内十多个 BU 的框架产品，KAITIAN 还在不断打磨与高速迭代中，预计今年会正式开源跟大家见面。

后端篇

在过去的 2020 年，淘系的服务端系统随着业务的深入发展，技术进入深水区，更为灵活的业务系统架构支撑业务的高速变化，驱动了技术团队在架构方面的深入思考，形成了一系列业务平台，推进领域架构设计的进步；基础性技术更为深入的研究，寻本溯源，自底向上更高效的解决业务问题。服务端的几篇文章从一个侧面呈现了这一年来在架构和基础技术上的探索，希望与行业里的同伴共同探讨。

——阿里巴巴资深技术专家 孤星

技术经典总结

如何正确地实现重试(Retry)

作者 | 虎鸣

出品 | 淘系技术

在日常的编码过程中，无论是和本地服务相关的本机资源交互，还是和本地服务相关的远程资源甚至是远程服务进行交付，都可能会遇到失败（异常），这时候，我们最常见的做法就是重试，本文将和大家介绍一下如何正确实现重试。

什么是重试

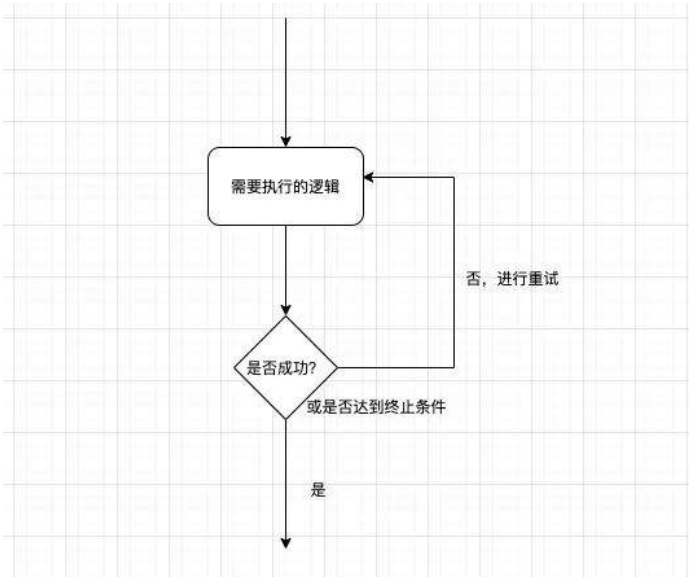
重试：即从新尝试，以观察结果是否符合预期。

to try (something) again to see if it is successful, working, or satisfactory。

在生活中，以买彩票为例，再次尝试购买彩票有以下几种情况：

- 彩票没中（结果不符合预期）
- 上次没带钱（条件不符合）
- 彩票门店没开门（结果异常）

图形化的表述，可以简化为：



什么是正确的重试

和任何的锲而不舍都需要向着现实低头一样，“重试”也需要有终止条件（即有条件的重试），想象一样买彩票的场景，如果屡次不中，一直尝试不停歇，那不是得破产吗？

在日常的编码中，我们最常见的做法也是如此，即指定一个重试次数的上限，然后单次请求达到上限后返回。但是这样做了就没有问题了吗？答案当然是否定的。

固定循环次数方式

这是最常见的版本，样板方法为：

```
public static <T> T retryNoDelay(final Callable<T> callable, final int maxAttempts) {
    for (int i = 0; i < maxAttempts; i++) {
        try {
            final T t = callable.call();
            if (isExpected(t)) {
                return t;
            }
        } catch (Exception e) {
            //logging
        }
    }
    //return default t or error
    return null;
}
```

比如：

```
for (int i = 0; i < configBean.getInsertMessageInboxRetry(); i++) {
    try {
        insertMessageInboxResult = messageInboxManager.insertMessage(messageInbox);
        if (insertMessageInboxResult) {
            break;
        }
    } catch (Exception e) {
        log.error("insertMessageInbox exception retry {}, messageInbox={}", i, messageInbox)
    }
}
```

这种方式的问题在于：不带 backoff 的重试，对于下游来说会在失败发生时进一步遇到更多的请求压力，继而进一步恶化。

带固定 delay 的方式

在失败之后，进行固定间隔的 delay， delay 的方式按照是方法本身是异步还是同步的，可以通过定时器或则简单的 Thread.sleep 实现，样板方法为：

```java
public static <T> T retryWithFixedDelay(final Callable<T> callable, final int maxAttempts,
                                        final int fixedBackOff) {
    for (int i = 0; i < maxAttempts; i++) {
        try {
            final T t = callable.call();
            if (isExpected(t)) {
                return t;
            }
        } catch (Exception e) {
            //logging
            try {
                Thread.sleep(fixedBackOff);
            } catch (Exception ee){}
        }
    }
    //return default t or error
    return null;
}
```

比如：

```java
private T retry(Supplier<T> function) throws RuntimeException {
    log.error("first Command failed, will be retried " + maxRetries + " times.");
    invokeCnt = 0;
    Exception exception = null;
    while (invokeCnt < maxRetries) {
        try {
            if (sleepTime > MIN_SLEEP_TIME && sleepTime <= MAX_SLEEP_TIME) {
                Thread.sleep(sleepTime);
            }
            return function.get();
        } catch (InterruptedException ex) {...} catch (Exception ex) {
            exception = ex;
            invokeCnt++;
            log.error(invokeCnt + " times retry failed " + " of " + maxRetries + " error: " + ex);
            if (invokeCnt >= maxRetries) {...}
        }
    }
    throw new RuntimeException(maxRetries + " retries all failed",exception);
}
```

这种方式的问题在于：虽然这次带了固定间隔的 backoff，但是每次重试的间隔固定，此时对于下游资源的冲击将会变成间歇性的脉冲；特别是当集群都遇到类似的问题时，步调一致的脉冲，将会最终对资源造成很大的冲击，并陷入失败的循环中。

想想一下，一群鼓手，协调一致地击鼓时所产生的效果。

带随机 delay 的方式：

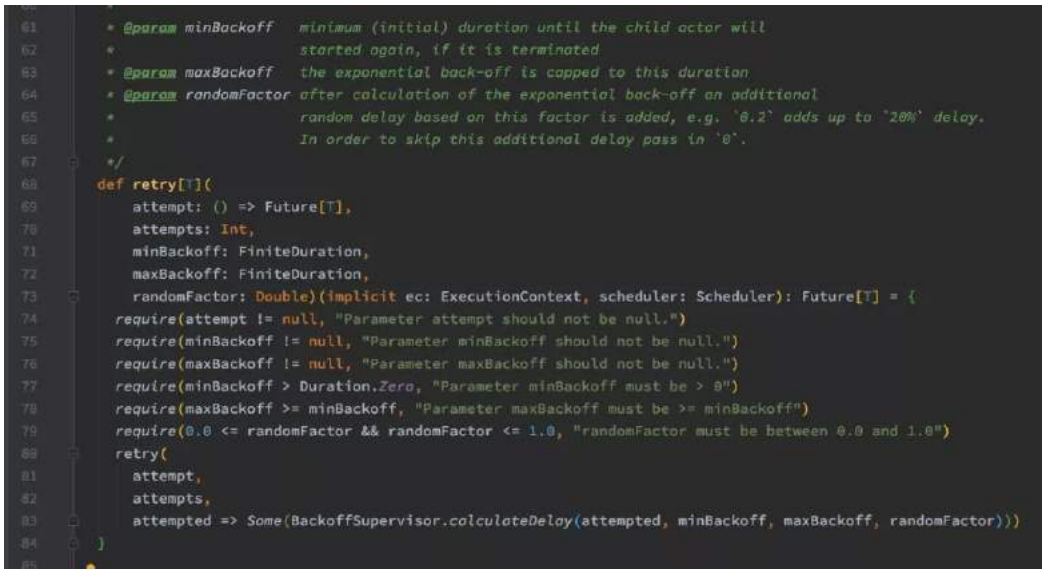
和 2 中固定间隔的 delay 不一样，现在采用随机 backoff 的方式，即具体的 delay 时间，在一个最小值和最大值之间浮动，样板代码如下：

```java
public static <T> T retryWithRandomDelay(final Callable<T> callable, final int maxAttempts,
                                         final int minBackoff,
                                         final int maxBackoff,
                                         //randomFactor, 0.0 ~ 1.0
                                         final double randomFactor) {
    for (int i = 0; i < maxAttempts; i++) {
        try {
            final T t = callable.call();
            if (isExpected(t)) {
                return t;
            }
        } catch (Exception e) {
            //logging
            try {
                final double rnd = 1.0 + ThreadLocalRandom.current().nextDouble() * randomFactor;
                long backOffTime;
                try {
                    backOffTime = (long)(Math.min(maxBackoff, minBackoff * Math.pow(2, i)) * rnd) ;
                } catch (Exception ee) {
                    backOffTime = maxBackoff;
                }
                Thread.sleep(backOffTime);
            } catch (Exception ee){}
        }
    }
    //return default t or error
    return null;
}
```

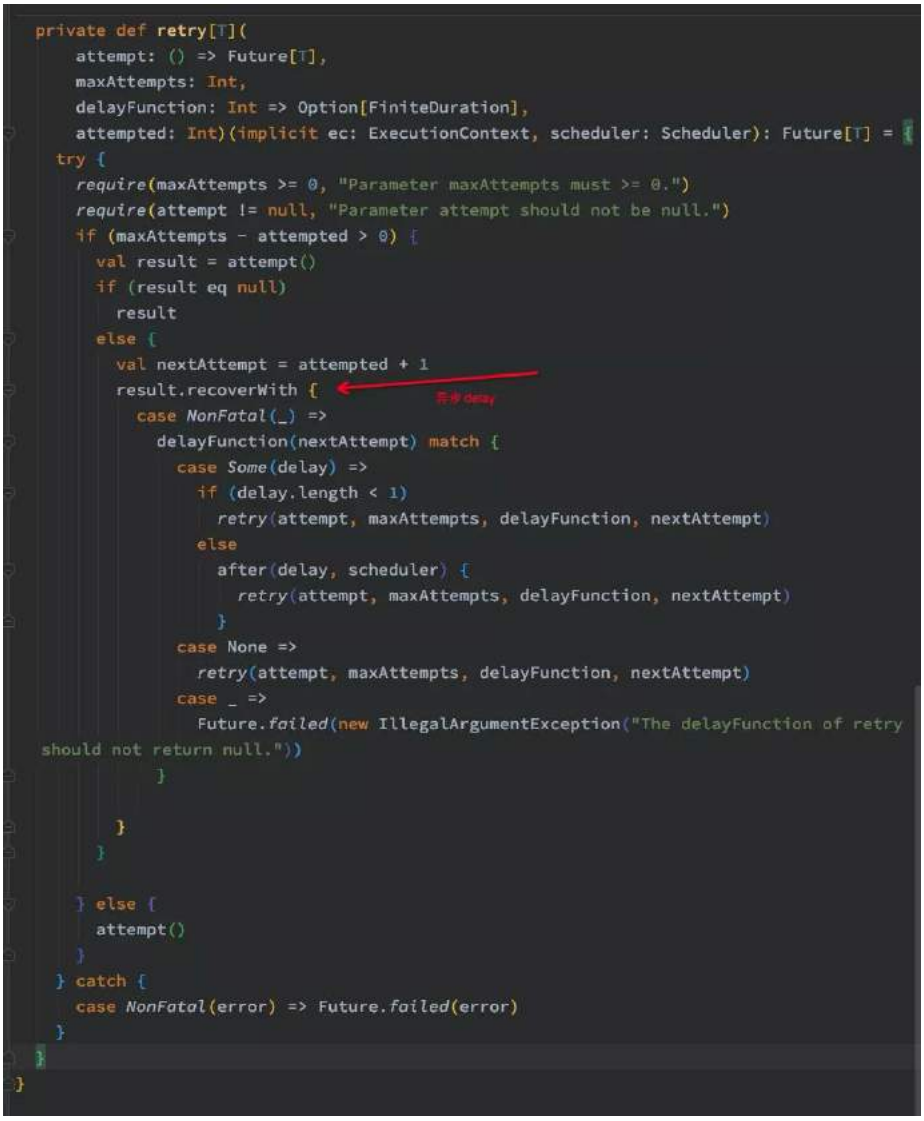
比如：

```java
public static <V> V retryWithBackoff(
  final Callable<V> callable,
  final int maxRetryTime,
  final int sleepInMills) throws Exception {
  if (maxRetryTime >= 10){...}
  if (sleepInMills <= 0){...}
  try {
    return callable.call();
  } catch (Throwable e) {
    if (maxRetryTime <= 0) {...} else {
      try {
        LOGGER.error("retry with maxRetryTime:{},sleepInMills:{}",maxRetryTime,sleepInMills,e);
        Thread.sleep(sleepInMills);
        return retryWithBackoff(callable,
                                maxRetryTime: maxRetryTime - 1,
                                //随机1-3倍，集群的行为不要步调一致
                                sleepInMills: sleepInMills * ThreadLocalRandom.current().nextInt( origin: 1, bound: 3));
      } catch (InterruptedException e1) {
        //ignore
        Thread.currentThread()
          .interrupt();
        throw e;
      }
    }
  }
}
```

或则一个类似的异步版本：



```
   * @param minBackoff   minimum (initial) duration until the child actor will
   *                     started again, if it is terminated
   * @param maxBackoff   the exponential back-off is capped to this duration
   * @param randomFactor after calculation of the exponential back-off an additional
   *                     random delay based on this factor is added, e.g. `0.2` adds up to `20%` delay.
   *                     In order to skip this additional delay pass in `0`.
   */
  def retry[T](
      attempt: () => Future[T],
      attempts: Int,
      minBackoff: FiniteDuration,
      maxBackoff: FiniteDuration,
      randomFactor: Double)(implicit ec: ExecutionContext, scheduler: Scheduler): Future[T] = {
    require(attempt != null, "Parameter attempt should not be null.")
    require(minBackoff != null, "Parameter minBackoff should not be null.")
    require(maxBackoff != null, "Parameter maxBackoff should not be null.")
    require(minBackoff > Duration.Zero, "Parameter minBackoff must be > 0")
    require(maxBackoff >= minBackoff, "Parameter maxBackoff must be >= minBackoff")
    require(0.0 <= randomFactor && randomFactor <= 1.0, "randomFactor must be between 0.0 and 1.0")
    retry(
      attempt,
      attempts,
      attempted => Some(BackoffSupervisor.calculateDelay(attempted, minBackoff, maxBackoff, randomFactor)))
  }
```



```
  private def retry[T](
      attempt: () => Future[T],
      maxAttempts: Int,
      delayFunction: Int => Option[FiniteDuration],
      attempted: Int)(implicit ec: ExecutionContext, scheduler: Scheduler): Future[T] = {
    try {
      require(maxAttempts >= 0, "Parameter maxAttempts must >= 0.")
      require(attempt != null, "Parameter attempt should not be null.")
      if (maxAttempts - attempted > 0) {
        val result = attempt()
        if (result eq null)
          result
        else {
          val nextAttempt = attempted + 1
          result.recoverWith {
            case NonFatal(_) =>
              delayFunction(nextAttempt) match {
                case Some(delay) =>
                  if (delay.length < 1)
                    retry(attempt, maxAttempts, delayFunction, nextAttempt)
                  else
                    after(delay, scheduler) {
                      retry(attempt, maxAttempts, delayFunction, nextAttempt)
                    }
                case None =>
                  retry(attempt, maxAttempts, delayFunction, nextAttempt)
                case _ =>
                  Future.failed(new IllegalArgumentException("The delayFunction of retry
  should not return null."))
              }

          }
        }

      } else {
        attempt()
      }
    } catch {
      case NonFatal(error) => Future.failed(error)
    }
  }
}
```

这种方式的问题在于：虽然现在解决了 backoff 的时间集中的问题，对时间进行了随机打散，但是依然存在下面的问题：

- 如果依赖的底层服务持续地失败，改方法依然会进行固定次数的尝试，并不能起到很好的保护作用。
- 对结果是否符合预期，是否需要进行重试依赖于异常。
- 无法针对异常进行精细化的控制，如只针部分异常进行重试。

可进行细粒度控制的重试

比如可以针对特定的异常来说，其样板代码为：

```
public static <T> T retryWithRandomDelayAndCondition(final Callable<T> callable, final int maxAttempts,
                                                     final int minBackoff,
                                                     final int maxBackoff,
                                                     //randomFactor, 0.0 - 1.0
                                                     final double randomFactor,
                                                     final Predicate<Throwable> retryPredicate) {
    for (int i = 0; i < maxAttempts; i++) {
        try {
            final T t = callable.call();
            if (isExpected(t)) {
                return t;
            }
        } catch (Exception e) {
            //logging
            if (!retryPredicate.test(e)) {
                break;
            }
            try {
                final double rnd = 1.0 + ThreadLocalRandom.current().nextDouble() * randomFactor;
                long backOffTime;
                try {
                    backOffTime = (long)(Math.min(maxBackoff, minBackoff * Math.pow(2, i)) * rnd) ;
                } catch (Exception ee) {
                    backOffTime = maxBackoff;
                }
                Thread.sleep(backOffTime);
            } catch (Exception ee){}
        }
    }
    //return default t or error
    return null;
}
```

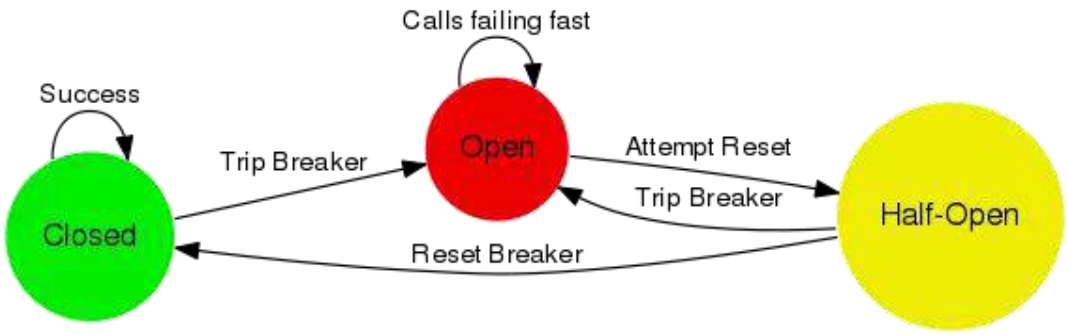
一般这个时候，代码已经相对来说比较复杂了，个人推荐使用 resilience4j-retry 或则 spring-retry 等库来进行组合，减少自己编写时维护成本，比如以 resilience4j-retry 为例，其可以使用配置代码对重试策略进行细粒度的控制，比如：

```
RetryConfig config = RetryConfig.custom()
.maxAttempts(2)
.waitDuration(Duration.ofMillis(1000))
.retryOnResult(response -> response.getStatus() == 500)
.retryOnException(e -> e instanceof WebServiceException)
.retryExceptions(IOException.class, TimeoutException.class)
.ignoreExceptions(BunsinessException.class, OtherBunsinessException.class)
.build();
RetryRegistry registry = RetryRegistry.of(config);
Retry retryWithDefaultConfig = registry.retry("name1");
CheckedFunction0<String> retryableSupplier = Retry
.decorateCheckedSupplier(retry, helloWorldService::sayHelloWorld);
```

这种方式的问题在于：虽然可以比较好的控制重试策略，但是对于下游资源持续性的失败，依然没有很好的解决。当持续的失败时，对下游也会造成持续性的压力。一般这种问题的解法，我们日常工作中都是通过一个开关来进行人工断路，另一个比较好的解法是和断路器结合。

和断路器结合

断路器 在每个家庭中都有，但是在软件工程上，看到大家应用的并不多。断路器模式 一般用在当下游资源失败后，但是失败恢复的时间不固定时，自动地进行探索式地恢复尝试，并且在遇到较多失败时，能够快速自动地断开，从而避免失败蔓延的一种模式。



有人将这种模式叫做『熔断器模式』，其实是错误的，能够「熔断」的，那是保险丝，而不是断路器，断路器来自于电气工程，如下图示：

在应用断路器时，需要对下游资源的每次调用都通过断路器，对代码具备一定的结构侵入性。常见的有 Hystrix 或 resilience4j.：

```
    // Given
    CircuitBreaker circuitBreaker = CircuitBreaker.ofDefaults("testName");
/
    // When I decorate my function
    CheckedFunction0<String> decoratedSupplier = CircuitBreaker
    .decorateCheckedSupplier(circuitBreaker, () -> "This can be any method which returns: 'Hello");
```

又或者：

```
    def callWithCircuitBreakerCS[T](body: Callable[CompletionStage[T]]): CompletionStage[T]
```

当断路器处于开断状态时，所有的请求都会直接失败，不再会对下游资源造成冲击，并能够在一段时间后，进行探索式的尝试，如果没有达到条件，可以自动地恢复到之前的闭合状态。

重试的一些其他实现

目前 重试 在 RxJava 、Reactor、Akka-Stream 等中也都有实现，不过所实现的组合子（operator/操作）实现的相对简单，在实践中，如果需要得到很好的效果，还需要配合断路器来进行，从而最大限度地进行保护下游。

对失败做出反应

在反应式宣言中，也有提到，对对失败做出反应，系统在遇到失败时，可以恢复，并隔离失败的组件，而不是不受控的失败。系统是否具备回弹性，对于线上正常安全生产有很大的影响。正确地实现"重试"，只是整个大图中非常小的一环，实际生产中还需要从架构、生产流程、编码细节处理，监控报警等多种手段入手。

失败（和"错误"相对照）

失败是一种服务内部的意外事件， 会阻止服务继续正常地运行。失败通常会阻止对于当前的、 并可能所有接下来的客户端请求的响应。和错误相对照， 错误是意料之中的，并且针各种情况进行了处理（ 例如， 在输入验证的过程中所发现的错误）， 将会作为该消息的正常处理过程的一部分返回给客户端。而失败是意料之外的， 并且在系统能够恢复至（和之前）相同的服务水平之前，需要进行干预。这并不意味着失败总是致命的（fatal）， 虽然在失败发生之后， 系统的某些服务能力可能会被降低。错误是正常操作流程预期的一部分， 在错误发生之后， 系统将会立即地对其进行处理， 并将继续以相同的服务能力继续运行。失败的例子有：硬件故障、 由于致命的资源耗尽而引起的进程意外终止，以及导致系统内部状态损坏的程序缺陷。

回弹性

系统在出现失败时依然保持即时响应性。这不仅适用于高可用的、 任务关键型系统——任何不具备回弹性的系统都将会在发生失败之后丢失即时响应性。回弹性是通过复制、 遏制、 隔离以及委托来实现的。失败的扩散被遏制在了每个组件内部， 与其他组件相互隔离，从而确保系统某部分的失败不会危及整个系统，并能独立恢复。每个组件的恢复都被委托给了另一个（外部的）组件， 此外，在必要时可以通过复制来保证高可用性。（因此）组件的客户端不再承担组件失败的处理。

小结

写这篇文章和大家分享，抛砖引玉，大家感兴趣也可以看看自己负责的应用中目前对于重试的处理，以及一些主流的开源框架或者库中的处理。

面向 5G 的阿里自研标准化协议库 XQUIC

作者 | 刘彦梅(喵吉)
出品 | 淘系技术

XQUIC 是阿里巴巴淘系架构团队自研的 IETF QUIC 标准化协议库实现，在手机淘宝上进行了广泛的应用，并在多个不同类型的业务场景下取得明显的效果提升，为手机淘宝 APP 的用户带来丝般顺滑的网络体验：

- 在 RPC 请求场景，网络耗时降低 15%
- 在直播高峰期场景，卡顿率降低 30%、秒开率提升 2%
- 在短视频场景，卡顿率降低 20%

从以上提升效果可以看出，对 QUIC 的一个常见认知谬误："QUIC 只对弱网场景有优化提升"是不准确的。实际上 QUIC 对于整体网络体验有普遍提升，弱网场景由于基线较低、提升空间更显著。此外，在 5G 推广初期，基站部署不够密集的情况下，如何保证稳定有效带宽速率，是未来 2-3 年内手机视频应用将面临的重大挑战，而我们研发的 MPQUIC 将为这些挑战提供有效的解决方案。

本文将会重点介绍 XQUIC 的设计原理，面向业务场景的网络传输优化，以及面向 5G 的 Multipath QUIC 技术（多路径 QUIC）。

QUIC

网络分层模型及 QUIC 进化史

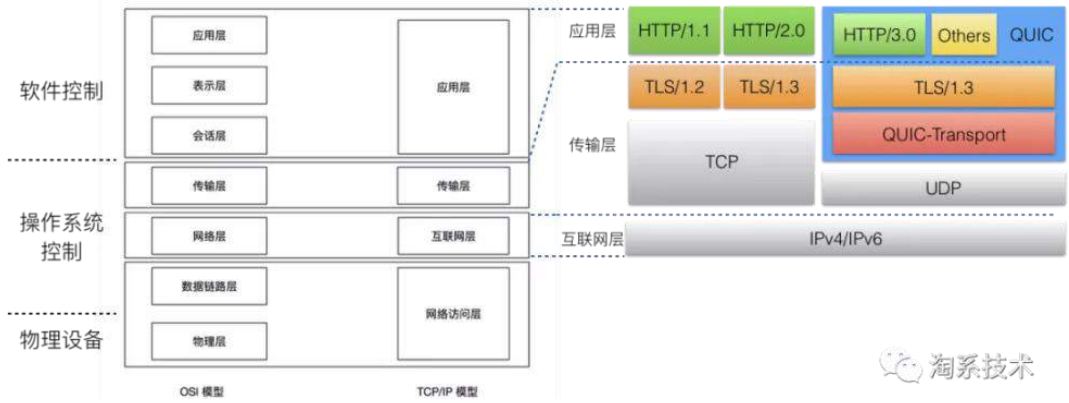


图 1. 网络七层/四层模型 和 QUIC 分层设计

为了方便说明 QUIC 在网络通信协议栈中所处的位置及职能，我们简单回顾一下网络 OSI 模型（七层模型）和 TCP/IP 模型（四层模型）。从两套网络模型中可以看出，网络传输行为和策略主要由传输层来控制，而 TCP 作为过去 30 年最为流行和广泛使用的传输层协议，是由操作系统控制和实现的。

QUIC 是由 Google 从 2013 年开始研究的基于 UDP 的可靠传输协议，它最早的原型是 SPDY + QUIC-Crypto + Reliable UDP，后来经历了 SPDY[1]转型为 2015 年 5 月 IETF 正式发布的 HTTP/2.0[2]，以及 2016 年 TLS/1.3[3]的正式发布。QUIC 在 IETF 的标准化工作组自 2016 年成立，考虑到 HTTP/2.0 和 TLS/1.3 的发布，它的核心协议族逐步进化为现在的 HTTP/3.0 + TLS/1.3 + QUIC-Transport 的组合。

QUIC 带来的核心收益是什么

众所周知，QUIC 具备多路复用/0-RTT 握手/连接迁移等多种优点，然而在这些优势中，最关键的核心收益，当属 QUIC 将四/七层网络模型中控制传输行为的传输层，从内核态实现迁移到了用户态实现，由应用软件控制。这将带来 2 个巨大的优势：

迭代优化效率大大提升。以服务端角度而言，大型在线系统的内核升级成本往往是非常高的，考虑到稳定性等因素，升级周期从月到年为单位不等。以客户端角度而言，手机操作系统版本升级同样由厂商控制，升级周期同样难以把控。调整为用户态实现后，端到端的升级都非常方便，版本迭代周期以周为计（甚至更快）。

灵活适应不同业务场景的网络需求。在过去 4G 的飞速发展中，短视频、直播等新的业务场景随着基建提供的下行带宽增长开始出现，在流媒体传输对于稳定高带宽和低延迟的诉求下，TCP 纷纷被各类标准/私有 UDP 解决方案逐步替代，难以争得一席之地。背后的原因是，实现在内核态的 TCP，难以用一套拥塞控制算法/参数适应快速发展的各类业务场景。这一缺陷将在 5G 下变得更加显著。QUIC 则可将拥塞控制算法/参数调控到连接的粒度，针对同一个 APP 内的不同业务场景（例如 RPC/短视频/直播等）具备灵活适配/升级的能力。

在众多增强型 UDP 的选择中，QUIC 相较于其他的方案最为通用，不仅具备对于 HTTP 系列的良好兼容性，同时其优秀的的分层设计，也使得它可以将传输层单独剥离作为 TCP 的替代方案，为其他应用层协议提供可靠/非可靠传输能力（是的，QUIC 也有非可靠传输草案设计）。

XQUIC 是什么、为什么选择自研 + 标准

XQUIC 是阿里自研的 IETF QUIC 标准化实现，这个项目由淘系架构网关与基础网络团队发起和主导，当前有阿里云 CDN、达摩院 XG 实验室与 AIS 网络研究团队等多个团队参与其中。

现今 QUIC 有多家开源实现，为什么选择标准协议 + 自研实现的道路？我们从 14 年开始关注 Google 在 QUIC 上的实践（手机淘宝在 16 年全面应用 HTTP/2），从 17 年底开始跟进并尝试在电商场景落地 GQUIC[4]，在 18 年底在手淘图片、短视频等场景落地 GQUIC 并拿到了一定的网络体验收益。然而在使用开源方案的过程中或多或少碰到了一些问题，Google 的实现是所有开源实现中最为成熟优秀的，然而由于 Chromium 复杂的运行环境和 C++实现的缘故，GQUIC 包大小在优化后仍然有 2.4M 左右，这使得我们在集成手淘时面临困难。在不影响互通性的前提下，我们进行了大量裁剪才勉强能够达到手淘集成的包大小要求，然而在版本升级的情况下难以持续迭代。其他的开源实现也有类似或其他的问题（例如依赖过多、无服务端实现、无稳定性保障等）。最终促使我们走上自研实现的道路。

为什么要选择 IETF QUIC[5]标准化草案的协议版本？过去我们也尝试过自研私有协议，在端到端都由内部控制的场景下，私有协议的确是很方便的，但私有协议方案很难走出去建立

一个生态圈 / 或者与其他的应用生态圈结合（遵循相同的标准化协议实现互联互通）；从阿里作为云厂商的角度，私有协议也很难与外部客户打通；同时由于 IETF 开放讨论的工作模式，协议在安全性、扩展性上会有更全面充分的考量。

因此我们选择 IETF QUIC 标准化草案版本来落地。截止目前，IETF 工作组草案已经演化到 draft-29 版本（2020.6.10 发布），XQUIC 已经支持该版本，并能够与其他开源实现基于 draft-29 互通。

XQUIC 整体架构和传输框架设计

XQUIC 是 IETF QUIC 草案版本的一个 C 协议库实现，端到端的整体链路架构设计如下图所示。XQUIC 内部包含了 QUIC-Transport（传输层）、QUIC-TLS（加密层、与 TLS/1.3 对接）和 HTTP/3.0（应用层）的实现。在外部依赖方面，TLS/1.3 依赖了开源 boringssl 或 openssl 实现（两者 XQUIC 都做了支持、可用编译选项控制），除此之外无其他外部依赖。

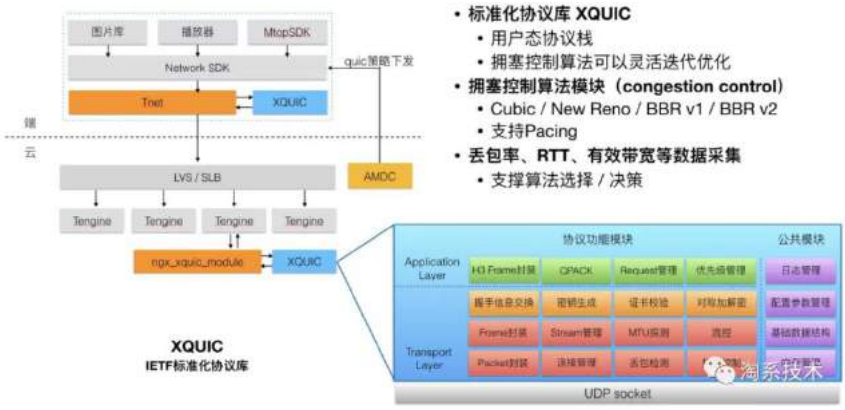


图 2. XQUIC 端到端架构设计 和 内部分层模块

XQUIC 整体包大小在 900KB 左右（包含 boringssl 的情况下），对于客户端集成是较为轻量的（支持 Android/iOS）。服务端方面，由于阿里内部网关体系广泛使用 Tengine（Nginx 开源分支），我们开发了一个 ngx_xquic_module 用于适配 Tengine 服务端。协议的调度方面，由客户端网络库与调度服务 AMDC 配合完成，可以根据版本/地域/运营商/设备百分比进行协议调度。

XQUIC 传输层内部流程设计如下图，可以看到 XQUIC 内部的读写事件主流程。考虑到跨平台兼容性，UDP 收发接口由外部实现并注册回调接口。XQUIC 内部维护了每条连接的状态机、Stream 状态机，在 Stream 级别实现可靠传输（这也是根本上解决 TCP 头部阻塞的关键），并通过读事件通知的方式将数据投递给应用层。传输层 Stream 与应用层 HTTP/3 的 Request Stream 有一一映射关系，通过这样的方式解决 HTTP/2 over TCP 的头部阻塞问题。

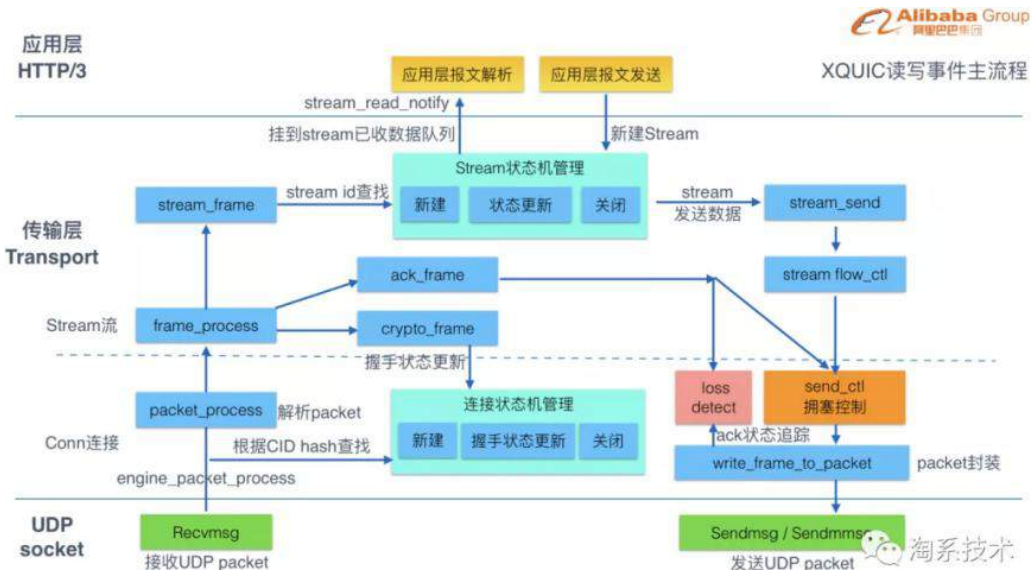


图 3. XQUIC 读写事件主流程设计

考虑到 IETF QUIC 传输层的设计可以独立剥离，并作为 TCP 的替代方案对接其他应用层协议，XQUIC 内部实现同样基于这样的分层方式，并对外提供两套原生接口：HTTP/3 请求应答接口 和 传输层独立接口（类似 TCP），使得例如 RTMP、上传协议等可以较为轻松地接入。

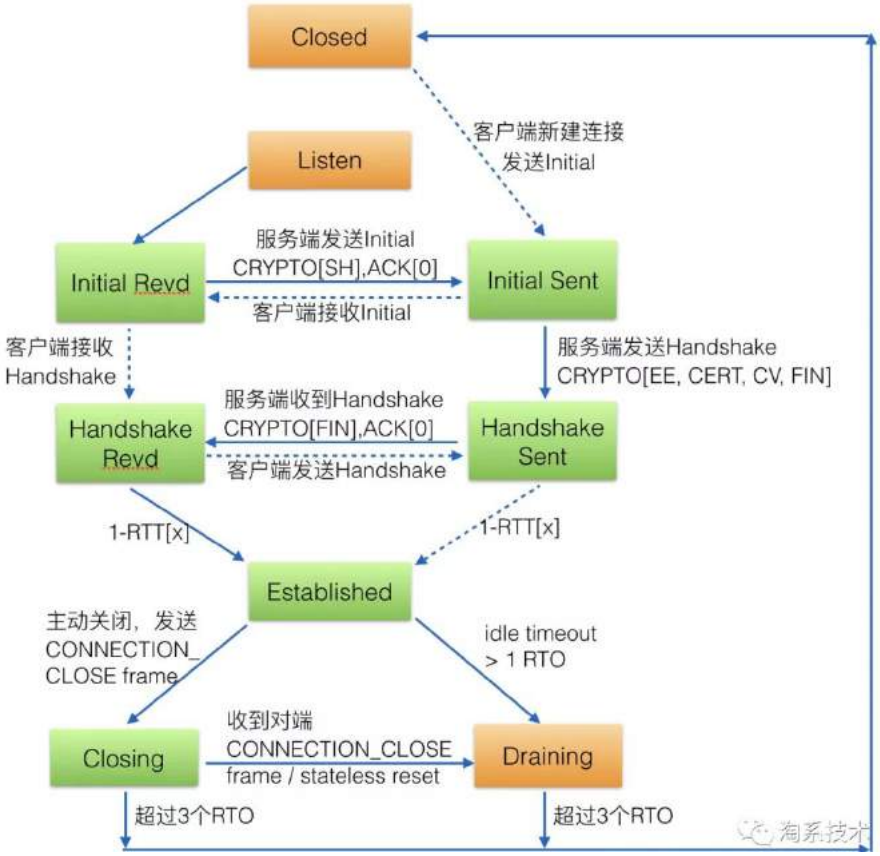


图 4. XQUIC 内部的连接状态机设计（参考 TCP）

XQUIC 拥塞控制算法模块

我们将 XQUIC 传输层的内部设计放大，其中拥塞控制算法模块，是决定传输行为和效率的核心模块之一。

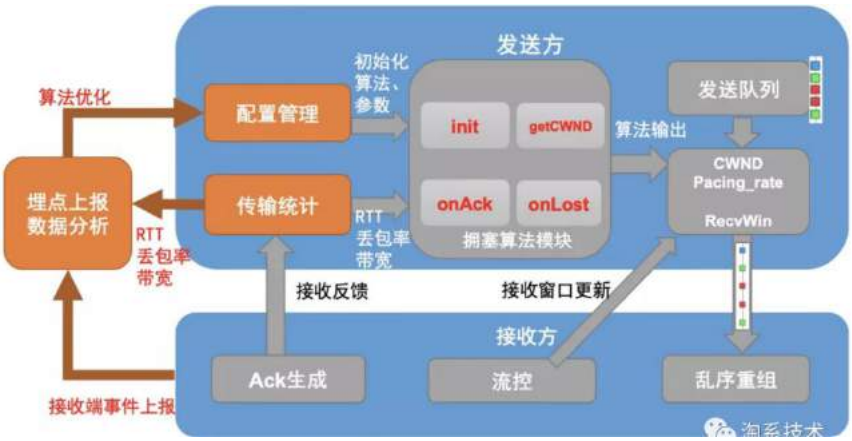


图 5. XQUIC 拥塞控制算法模块设计

为了能够方便地实现多套拥塞控制算法，我们将拥塞控制算法流程抽象成 7 个回调接口，其中最核心的两个接口 onAck 和 onLost 用于让算法实现收到报文 ack 和检测到丢包时的处理逻辑。XQUIC 内部实现了多套拥塞控制算法，包括最常见的 Cubic、New Reno，以及音视频场景下比较流行的 BBR v1 和 v2，每种算法都只需要实现这 7 个回调接口即可实现完整算法逻辑。

为了方便用数据驱动网络体验优化，我们将连接的丢包率、RTT、带宽等信息通过埋点数据采样和分析的方式，结合每个版本的算法调整进行效果分析。同时在实验环境下模拟真实用户的网络环境分布，更好地预先评估算法调整对于网络体验的改进效果。

面向业务场景的传输优化

XQUIC 在 RPC 请求场景降低网络耗时 15%，在短视频场景下降低 20%卡顿率，在直播场景高峰期降低 30%卡顿率、提升 2%秒开率（相对于 TCP）。以下基于当下非常火热的直播场景，介绍 XQUIC 如何面向业务场景优化网络体验。

优化背景

部分用户网络环境比较差，存在直播拉流打开慢、卡顿问题。

丢包率	连接比例
> 5%	20%
> 10%	10%
> 20%	5%

RTT	连接比例
> 100ms	5%
> 250ms	1.5%
> 500ms	0.5%

淘系技术

图 6. 某节点丢包率和 RTT 统计分布

这是 CDN 某节点上统计的丢包率和 RTT 分布数据，可以看到，有 5%的连接丢包率超过 20%，0.5%的连接 RTT 超过 500ms，如何优化网络较差用户的流媒体观看体验成为关键。

秒开卡顿模型

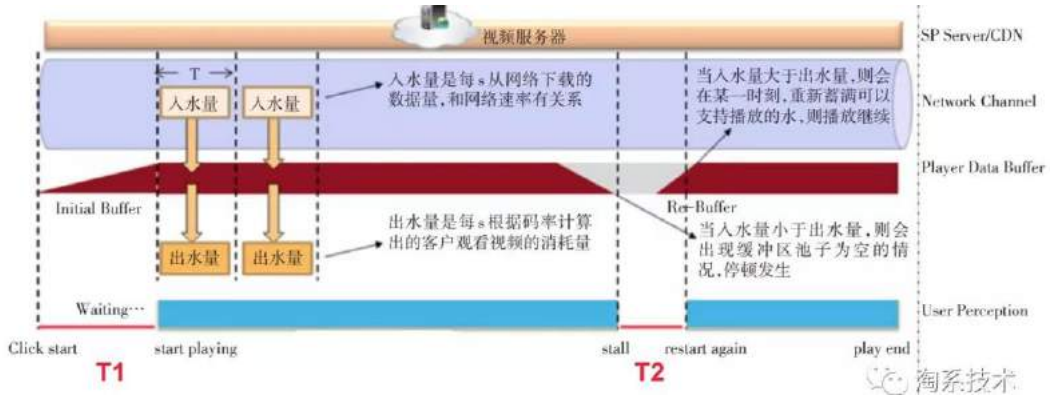


图 7. 直播拉流模型

直播拉流可以理解为一个注水模型，上面是 CDN 服务器，中间是播放器缓冲区，可以理解成一个管道，下面是用户的体感，用户点击播放时，CDN 不断向管道里注水，当水量达到播放器初始 buffer 时，首帧画面出现，然后播放器以一定速率排水，当水被排完时，播放器画面出现停顿，当重新蓄满支持播放的水后，继续播放。

我们假设 Initial Buffer（首帧）为 100K（实际调整以真实情况为准），起播时间 T1 < 1s 记为秒开，停顿时间 T2 > 100ms 记为卡顿。

- 优化目标
  - 提升秒开：1s 内下载完 100K
  - 降低卡顿：保持下载速率稳定，从而保持管道内始终有水

- 核心思路
  - 提升秒开核心--快

    高丢包率用户：加快重传

    高延迟用户：减少往返次数
  - 降低卡顿核心--稳

    优化拥塞算法机制，稳定高效地利用带宽

拥塞算法选型

常见的拥塞算法可分为三类：

- 基于路径时延（如 Vegas、Westwood）

将路径时延上升作为发生拥塞的信号，在单一的网络环境下（所有连接都使用基于路径时延的拥塞算法）是可行的，但是在复杂的网络环境下，带宽容易被其他算法抢占，带宽利用率最低。

- 基于丢包（如 Cubic、NewReno）

将丢包作为发生拥塞的信号，其背后的逻辑是路由器、交换机的缓存都是有限的，拥塞会导致缓存用尽，进而队列中的一些报文会被丢弃。

拥塞会导致丢包，但是丢包却不一定拥塞导致的。事实上，丢包可以分为两类，一类是拥塞丢包，另一类是噪声丢包，特别是在无线网络环境中，数据以无线电的方式进行传递，无线路由器信号干扰、蜂窝信号不稳定等都会导致信号失真，最终数据链路层 CRC 校验失败将报文丢弃。

基于丢包的拥塞算法容易被噪声丢包干扰，在高丢包率高延迟的环境中带宽利用率较低。

- 基于带宽时延探测（如 BBR）

既然无法区分拥塞丢包和噪声丢包，那么就不以丢包作为拥塞信号，而是通过探测最大带宽和最小路径时延来确定路径的容量。抗丢包能力强，带宽利用率高。

三种类型的拥塞算法没有谁好谁坏，都是顺应当时的网络环境的产物，随着路由器、交换机缓存越来越大，无线网络的比例越来越高，基于路径时延和基于丢包的的拥塞算法就显得不合时宜了。对于流媒体、文件上传等对带宽需求比较大的场景，BBR 成为更优的选择。

秒开率优化

加快握手包重传

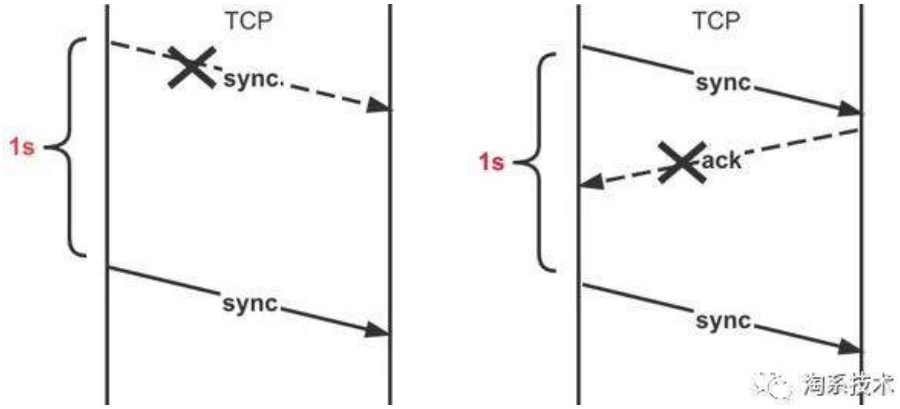


图 8. TCP 握手阶段出现丢包

如图，TCP 在握手时，由于尚未收到对端的 ACK，无法计算路径 RTT，因此，RFC 定义了初始重传超时，当超过这个超时时间还未收到对端 ACK，就重发 sync 报文。

TCP 秒开率上限：Linux 内核中的初始重传超时为 1s（RFC6298，June 2011），3%的丢包率意味着 TCP 秒开率理论上限为 97%，调低初始重传时间可以有效提升秒开率。

同理，如果你有一个 RPC 接口超时时间为 1s，那么在 3%丢包率的环境下，接口成功率不会超过 97%。

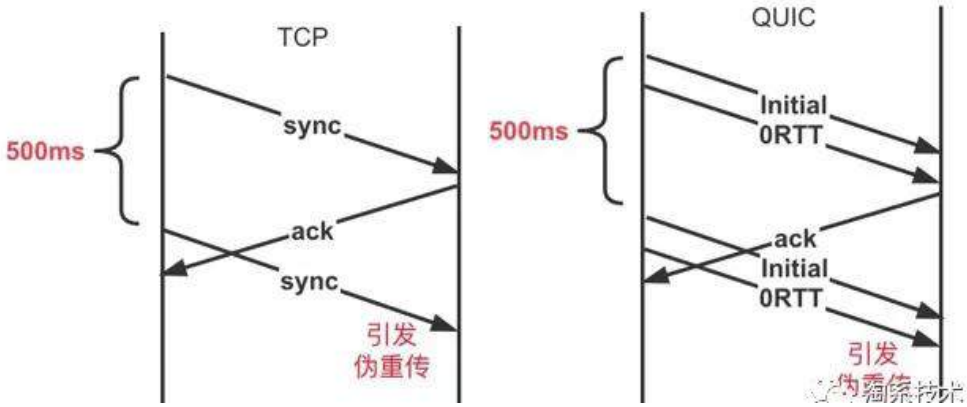


图 9. TCP 和 QUIC 握手阶段 - 伪重传模拟

另一方面，调低初始重传超时会引发伪重传，需要根据用户 RTT 分布进行取舍，比如初始重传超时调低到 500ms，那么 RTT 大于 500ms 的用户在握手期间将会多发一个 sync 报文。

减少往返次数

慢启动阶段 N 个 RTT 内的吞吐量（不考虑丢包）：

T = init_cwnd * (2^N-1) * MSS, N = L t / RTT 」
Linux 内核初始拥塞窗口=10（RFC 6928，April 2013）

首帧 100KB，需要 4 个 RTT，如果 RTT>250ms，意味着必然无法秒开。在我们举的这个例子中，如果调整为 32，那么只需要 2 个 RTT。

中国宽带速率状况报告

	2015 Q2	2019 Q2
固定宽带	6.11 Mbit/s	35.46 Mbit/s
	2016 Q3	2019 Q2
移动宽带	3.75 Mbit/s(3G)	23.58 Mbit/s(4G)

淘系技术

图 10. 宽带速率报告

从 2015 到 2019，固定带宽翻了 4.8 倍。从 2016 到 2019，移动宽带翻了 5.3 倍。初始拥塞窗口从 10 调整为 32 在合理范围内。

卡顿率优化

BBR RTT 探测优化

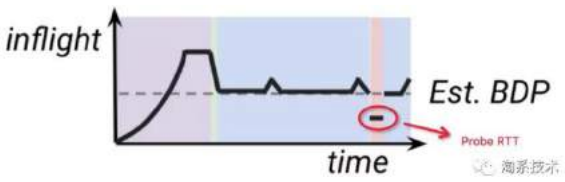


图 11. BBR v1 示意图：ProbeRTT 阶段

问题

- BBR v1 的 ProbeRTT 阶段会把 inflight 降到 4*packet 并保持至少 200ms。会导致传输速率断崖式下跌，引起卡顿。
- 10s 进入一次 ProbeRTT，无法适应 RTT 频繁变化的场景。

优化方案

- 减少带宽突降：inflight 降到 4*packet 改为降到 0.75 * Estimated_BDP。
- 加快探测频率：ProbeRTT 进入频率 10s 改为 2.5s。

推导过程

- 为什么是 0.75x?

  Max Estimated_BDP = 1.25*realBDP

  0.75 * Estimated_BDP = 0.75 * 1.25 * realBDP = 0.9375* realBDP
- 保证 inflight < realBDP，确保 RTT 准确性。

为什么是 2.5s?

- 优化后 BBR 带宽利用率：(0.2s * 75% + 2.5s * 100%) / (0.2s+ 2.5s) = 98.1%
- 原生 BBR 带宽利用率：(0.2s * 0% + 10s * 100%) / (0.2s + 10s) = 98.0%
- 在整体带宽利用率不降低的情况下，调整到 2.5s 能达到更快感知网络变化的效果。

优化效果

- 保证带宽利用率不低于原生 BBR 的前提下，使得发送更平滑，更快探测到 RTT 的变化。

BBR 带宽探测优化

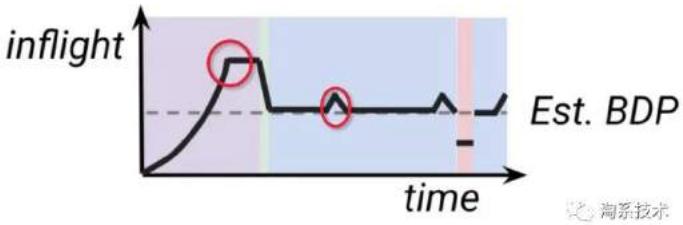


图 12. BBR v1 示意图：StartUp 和 ProbeBW 阶段

问题分析

StartUp 阶段 2.89 和 ProbeBW 阶段 1.25 的增益系数导致拥塞丢包，引发卡顿和重传率升高（Cubic 重传率 3%， BBR 重传率 4%）重传导致带宽成本增加 1%。

优化策略

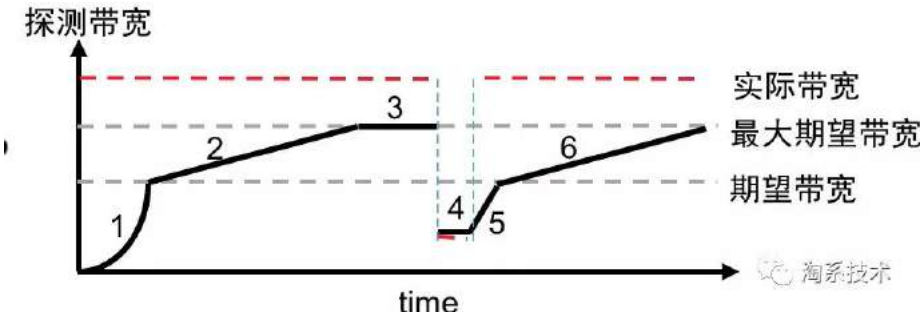


图 13. 带宽变化示意图

定义两个参数：

- 期望带宽：满足业务需要的最小带宽
- 最大期望带宽：能跑到的最大带宽

未达到期望带宽时采用较大的增益系数，较激进探测带宽。

达到期望带宽后采用较小的增益系数，较保守探测带宽。

优化效果

成本角度：重传率由 4%降到 3%，与 Cubic 一致，在不增加成本的前提下降低卡顿率。

另外，该策略可以推广到限速场景，如 5G 视频下载限速，避免浪费过多用户流量。

卡顿、秒开优化效果

在直播拉流场景下与 TCP 相比较，高峰期卡顿率降低 30%+，秒开率提升 2%。

写在优化最后的思考

最后，我们看看 TCP 初始重传超时和初始拥塞窗口的发展历程：

- 初始重传时间
  - RFC1122 (October 1989) 为 3s
  - RFC6298 (June 2011) 改为 1s，理由为 97.5%的连接 RTT<1s

- 初始拥塞窗口
  - RFC 3390 (October 2002) 为 min(4 * MSS, max(2 * MSS, 4380 bytes)) (MSS=1460 时为 3)
  - RFC 6928 (April 2013) 改为 min (10 * MSS, max (2 * MSS, 14600)) (MSS=1460 时为 10) —— 由 google 提出，理由为 90% of Google's search responses can fit in 10 segments (15KB).

首先 IETF RFC 是国际标准，需要考虑各个国家的网络情况，总会有一些网络较慢的地区。在 TCP 的 RFC 标准中，由于内核态实现不得不面临一刀切的参数选取方案，需要在考虑大盘分布的情况下兼顾长尾地区。

对比来看，QUIC 作为用户态协议栈，其灵活性相比内核态实现的 TCP 有很大优势。未来我们甚至有机会为每个用户训练出所在网络环境最合适的一套最优算法和参数，也许可以称之为千人千面的网络体验优化：）

Multipath QUIC 技术

Multipath QUIC（多路径 QUIC）是当前 XQUIC 内部正在研究和尝试落地的一项新技术。

MPQUIC 可以同时利用 cellular 和 wifi 双通道进行数据传输，不仅提升了数据的下载和上传速度，同时也加强了应用对抗弱网的能力，从而进一步提高用户的端到端体验。此外，由于 5G 比 4G 的无线信号频率更高，5G 的信道衰落问题也会更严重。所以在 5G 部署初期，基站不够密集的情况下如何保证良好信号覆盖是未来 2-3 年内手机视频应用的重大挑战，而我们研发的 MPQUIC 将为这些挑战提供有效的解决方案。

Multipath QUIC 的前身是 MPTCP[6]。MPTCP 在 IETF 有相对成熟的一整套 RFC 标准，但同样由于其实现在内核态，导致落地成本高，规模化推广相对困难。业界也有对 MPTCP 的应用先例，例如苹果在 iOS 内核态实现了 MPTCP，并将其应用在 Siri、Apple Push Notification Service 和 Apple Music 中，用来保障消息的送达率、降低音乐播放的卡顿次数和卡顿时间。

我们在 18 年曾与手机厂商合作（MPTCP 同样需要厂商支持），并尝试搭建 demo 服务器验证端到端的优化效果，实验环境下测试对「收藏夹商品展示耗时」与「直播间首帧播放耗时」降低 12-50%不等，然而最终由于落地成本太高并未规模化使用。由于 XQUIC 的用户态协议栈能够大大降低规模化落地的成本，现在我们重新尝试在 QUIC 的传输层实现多路径技术。

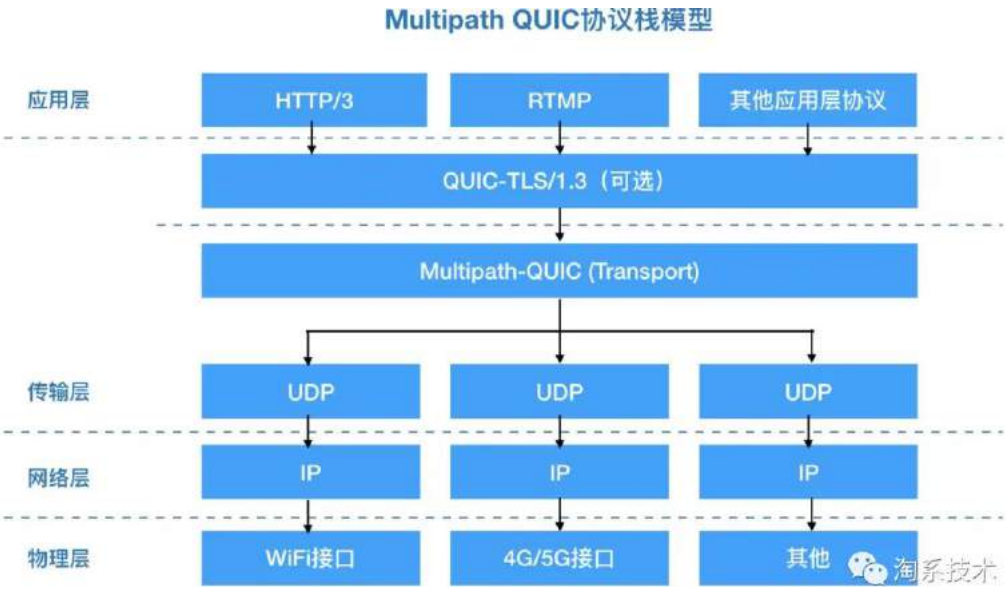


图 15. Multipath QUIC 协议栈模型

Multipath QUIC 对于移动端用户最核心的提升在于，通过在协议栈层面实现多通道技术，能够在移动端同时复用 Wi-Fi 和蜂窝移动网络，来达到突破单条物理链路带宽上限的效果；在单边网络信号强度弱的情况下，可以通过另一条通道补偿。适用的业务场景包括上传、短视频点播和直播，可以提升网络传输速率、降低文件传输耗时、提升视频的秒开和卡顿率。在 3GPP Release 17 标准中也有可能将 MPQUIC 引入作为 5G 标准的一部分[7]。

技术层面上，和 MPTCP 不同，我们自研的 MPQUIC 采取了全新的算法设计，这使得 MPQUIC 相比于 MPTCP 性能更加优化，解决了 slow path blocking 问题。在弱网中的性能比以往提升 30%以上。

Single Path	MPTCP	MPQUIC
0	51%~60%	51%~78%

图 16. 弱网下相对单路径的文件平均下载时间降低比例（实验数据）

Single Path	MPTCP	MPQUIC
0	> 60%	> 90%

图 17. 弱网下相对单路径的视频播放卡顿率降低比例（实验数据）

在推进 MPQUIC 技术落地的过程中，我们将会尝试在 IETF 工作组推进我们的方案作为 MPQUIC 草案的部分内容[8]。期望能够为 MPQUIC 的 RFC 标准制定和落地贡献一份力量。

下一步的未来

我们论证了 QUIC 的核心优势在于用户态的传输层实现（面向业务场景具备灵活调优的能力），而非单一针对弱网的优化。在业务场景的扩展方面，除了 RPC、短视频、直播等场景外，XQUIC 还会对其他场景例如上传链路等进行优化。

在 5G 逐步开始普及的时代背景下，IETF QUIC 工作组预计也将在 2020 年底左右将 QUIC 草案发布为 RFC 标准，我们推测在 5G 大背景下 QUIC 的重要性将会进一步凸显。

QUIC/MPQUIC 对于 5G

对于 5G 下 eMBB（Enhanced Mobile Broadband）和 URLLC（Ultra Reliable Low Latency）带来的不同高带宽/低延迟业务场景，QUIC 将能够更好地发挥优势，贴合场景需求调整传输策略（拥塞控制算法、ACK/重传策略）。对于 5G 运营商提供的切片能力，QUIC 同样可以针对不同的切片适配合适的算法组合，使得基础设施提供的传输能力能够尽量达到最大化利用的效果。在 XQUIC 的传输层实现设计中，同样预留了所需的适配能力。

在 5G 推广期间，在基站部署不够密集的情况下，保障稳定的有效带宽将会是音视频类的应用场景面临的巨大挑战。Multipath QUIC 技术能够在用户态协议栈提供有效的解决方案，然而这项新技术仍然有很多难点需要攻克，同时 3GPP 标准化组织也在关注这一技术的发展情况。

XQUIC 开源计划

阿里淘系技术架构团队计划在 2020 年底开源 XQUIC，期望能够帮助加速 IETF 标准化 QUIC 的推广，并期待更多的开源社区开发者参与到这个项目中来。

附录：参考文献

SPDY - HTTP/2.0 原型，由 Google 主导的支持双工通信的应用层协议

HTTP/2.0 - https://tools.ietf.org/html/rfc7540

TLS/1.3 - https://tools.ietf.org/html/rfc8446

GQUIC - 指 Google QUIC 版本，与 IETF QUIC 草案版本有一定差异

IETF QUIC - 指 IETF QUIC 工作组正在推进的 QUIC 系列草案：

https://datatracker.ietf.org/wg/quic/documents/，包括 QUIC-Transport、QUIC-TLS、QUIC-recovery、HTTP/3.0、QPACK 等一系列草案内容

MPTCP - https://datatracker.ietf.org/wg/mptcp/documents/

3GPP 向 IETF 提出的需求说明

https://tools.ietf.org/html/draft-bonaventure-quic-atsss-overview-00

MPQUIC - 我们正在尝试推进的草案

https://datatracker.ietf.org/doc/draft-an-multipath-quic-application-policy/

设计模式最佳套路 —— 愉快地使用策略模式

作者 | 周密（之叶）
出品 | 淘系技术

策略模式（Strategy Pattern）定义了一组策略，分别在不同类中封装起来，每种策略都可以根据当前场景相互替换，从而使策略的变化可以独立于操作者。比如我们要去某个地方，会根据距离的不同（或者是根据手头经济状况）来选择不同的出行方式（共享单车、坐公交、滴滴打车等等），这些出行方式即不同的策略。

何时使用策略模式

阿里开发规约-编程规约-控制语句-第六条：超过 3 层的 if-else 的逻辑判断代码可以使用卫语句、策略模式、状态模式等来实现。相信大家都见过这种代码：

```
if (conditionA) {
    逻辑 1
} else if (conditionB) {
    逻辑 2
} else if (conditionC) {
    逻辑 3
} else {
    逻辑 4
}
```

这种代码虽然写起来简单，但是很明显违反了面向对象的 2 个基本原则：

- 单一职责原则（一个类应该只有一个发生变化的原因）：因为之后修改任何一个逻辑，当前类都会被修改。
- 开闭原则（对扩展开放，对修改关闭）：如果此时需要添加（删除）某个逻辑，那么不可避免的要修改原来的代码。

因为违反了以上两个原则，尤其是当 if-else 块中的代码量比较大时，后续代码的扩展和维护就会逐渐变得非常困难且容易出错，使用卫语句也同样避免不了以上两个问题。因此根据我的经验，得出一个我个人认为比较好的实践：

- if-else 不超过 2 层，块中代码 1~5 行，直接写到块中，否则封装为方法。
- if-else 超过 2 层，但块中的代码不超过 3 行，尽量使用卫语句。
- if-else 超过 2 层，且块中代码超过 3 行，尽量使用策略模式。

愉快地使用策略模式

在 Spring 中，实现策略模式的方法多种多样，下面我分享一下我目前实现策略模式的“最佳套路”（如果你有更好的套路，欢迎赐教，一起讨论哦）。

需求背景

我们平台的动态表单，之前专门用于模型输入的提交。现在业务方希望对表单能力进行开放，除了可用于模型提交，还可以用于业务方指定功能的提交（方式设计为绑定一个 HSF 泛化服务，HSF 即淘系内部的 RPC 框架）。加上我们在配置表单时的“预览模式”下的提交，那么表单目前便有以下三种提交类型：

- 预览表单时的提交
- 模型输入时的提交
- 绑定 HSF 时的提交

现在，有请我的“最佳套路”上场。

第一步，定义策略接口

首先定义策略的接口，包括两个方法：

1. 获取策略类型的方法
2. 处理策略逻辑的方法

```
/**
 * 表单提交处理器
 */
public interface FormSubmitHandler<R extends Serializable> {


    /**
     * 获得提交类型（返回值也可以使用已经存在的枚举类）
     *
     * @return 提交类型
     */
    String getSubmitType();
    /**
     * 处理表单提交请求
     *
     * @param request 请求
     * @return 响应，left 为返回给前端的提示信息，right 为业务值
     */
    CommonPairResponse<String, R> handleSubmit(FormSubmitRequest request);
}
```

```
/**
 * 表单提交的请求
 */
@Getter
@Setter
public class FormSubmitRequest {

    /**
     * 提交类型
     *
     *
     * @see FormSubmitHandler#getSubmitType()
     */
    private String submitType;

    /**
```

```
    * 用户 id
    */
    private Long userId;

    /**
    * 表单提交的值
    */
    private Map<String, Object> formInput;

    // 其他属性
    }
```

其中，FormSubmitHandler 的 getSubmitType 方法用来获取表单的提交类型（即策略类型），用于根据客户端传递的参数直接获取到对应的策略实现；客户端传递的相关参数都被封装为 FormSubmitRequest，传递给 handleSubmit 进行处理。

第二步，相关策略实现

预览表单时的提交：

```
    @Component
    public class FormPreviewSubmitHandler implements FormSubmitHandler<Serializable> {

        private final Logger logger = LoggerFactory.getLogger(this.getClass());

        @Override
        public String getSubmitType() { return "preview"; }

        @Override
        public CommonPairResponse<String, Serializable> handleSubmit(FormSubmitRequest request)
{
        logger.info("预览模式提交：userId={}, formInput={}", request.getUserId(), request.getFormInput());

         return CommonPairResponse.success("预览模式提交数据成功！", null);
           }
        }
```

模型输入时的提交：

```
@Component
public class FormModelSubmitHandler implements FormSubmitHandler<Long> {

  private final Logger logger = LoggerFactory.getLogger(this.getClass());

  @Override
  public String getSubmitType() { return "model"; }

  @Override
  public CommonPairResponse<String, Long> handleSubmit(FormSubmitRequest request) {
   logger.info("模型提交：userId={}, formInput={}", request.getUserId(), request.getFormInput());

  // 模型创建成功后获得模型的 id
  Long modelId = createModel(request);

   return CommonPairResponse.success("模型提交成功！", modelId);
    }

   private Long createModel(FormSubmitRequest request) {
  // 创建模型的逻辑
  return 123L;
    }
  }
```

HSF 模式的提交：

```
@Component
public class FormHsfSubmitHandler implements FormSubmitHandler<Serializable> {

private final Logger logger = LoggerFactory.getLogger(this.getClass());

@Override
public String getSubmitType() { return "hsf"; }

@Override
public CommonPairResponse<String, Serializable> handleSubmit(FormSubmitRequest request) {
logger.info("HSF 模式提交：userId={}, formInput={}", request.getUserId(), request.getFormInput());
```

```
    // 进行 HSF 泛化调用，获得业务方返回的提示信息和业务数据
    CommonPairResponse<String, Serializable> response = hsfSubmitData(request);

    return response;
        }
      ...
    }
```

第三步，建立策略的简单工厂

```
    @Component
    public class FormSubmitHandlerFactory implements InitializingBean, ApplicationContextAware {

     private static final
    Map<String, FormSubmitHandler<Serializable>> FORM_SUBMIT_HANDLER_MAP = new
HashMap<>(8);

    private ApplicationContext appContext;

    /**
    * 根据提交类型获取对应的处理器
    *
    * @param submitType 提交类型
    * @return 提交类型对应的处理器
    */
    public FormSubmitHandler<Serializable> getHandler(String submitType) {
      return FORM_SUBMIT_HANDLER_MAP.get(submitType);
        }

      @Override
      public void afterPropertiesSet() {
    // 将 Spring 容器中所有的 FormSubmitHandler 注册到 FORM_SUBMIT_HANDLER_MAP
      appContext.getBeansOfType(FormSubmitHandler.class)
      .values()
      .forEach(handler -> FORM_SUBMIT_HANDLER_MAP.put(handler.getSubmitType(), handler));
        }

       @Override
      public void setApplicationContext(@NonNull ApplicationContext applicationContext) {
      appContext = applicationContext;
        }}
```

我们让 FormSubmitHandlerFactory 实现 InitializingBean 接口，在 afterPropertiesSet 方法中，基于 Spring 容器将所有 FormSubmitHandler 自动注册到 FORM_SUBMIT_HANDLER_MAP，从而 Spring 容器启动完成后， getHandler 方法可以直接通过 submitType 来获取对应的表单提交处理器。

第四步，使用 & 测试

在表单服务中，我们通过 FormSubmitHandlerFactory 来获取对应的表单提交处理器，从而处理不同类型的提交：

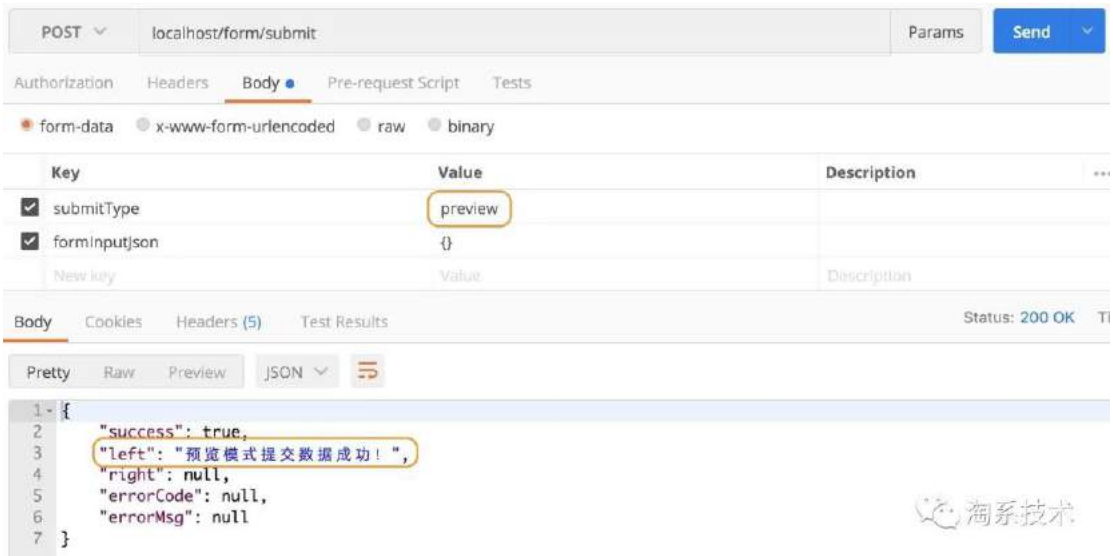
```
    @Service
    public class FormServiceImpl implements FormService {

    @Autowired
    private FormSubmitHandlerFactory submitHandlerFactory;

    public CommonPairResponse<String, Serializable> submitForm(@NonNull FormSubmitRequest
request) {
    String submitType = request.getSubmitType();

    // 根据 submitType 找到对应的提交处理器
    FormSubmitHandler<Serializable> submitHandler =
submitHandlerFactory.getHandler(submitType);

    // 判断 submitType 对应的 handler 是否存在
    if (submitHandler == null) {
    return CommonPairResponse.failure("非法的提交类型: " + submitType);
      }

    // 处理提交
     return submitHandler.handleSubmit(request);
      }
    }
```

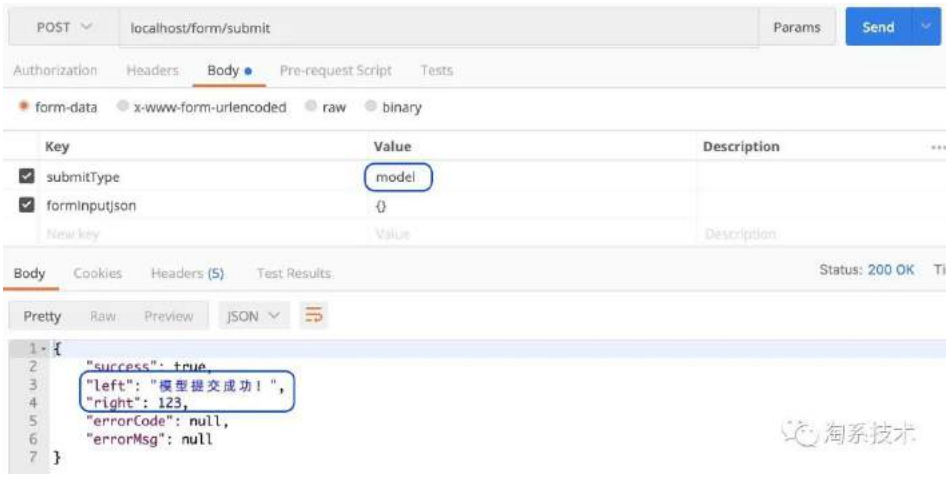
Factory 只负责获取 Handler，Handler 只负责处理具体的提交，Service 只负责逻辑编排，从而达到功能上的“低耦合高内聚”。

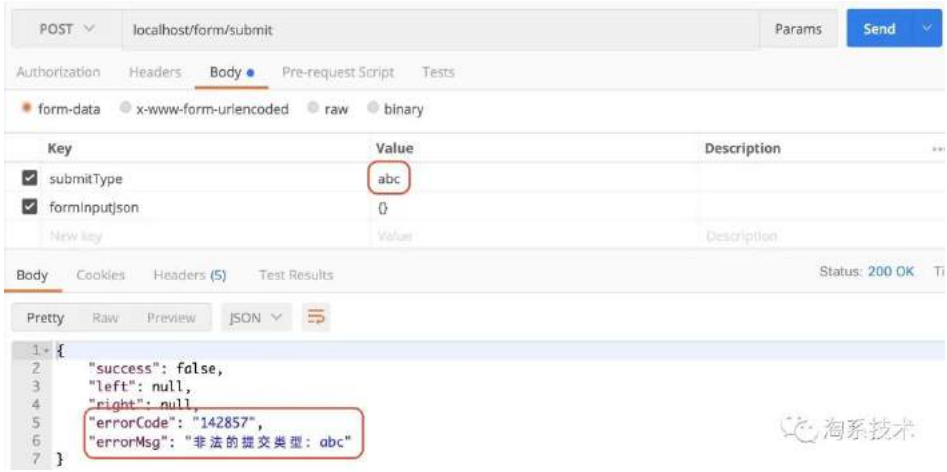
写一个简单的 Controller：

```
@RestController
public class SimpleController {

@Autowired
private FormService formService;

@PostMapping("/form/submit")
public CommonPairResponse<String, Serializable> submitForm(@RequestParam String
submitType,                                                    @RequestParam
String formInputJson) {
    JSONObject formInput = JSON.parseObject(formInputJson);

    FormSubmitRequest request = new FormSubmitRequest();
    request.setUserId(123456L);
    request.setSubmitType(submitType);
    request.setFormInput(formInput);

return formService.submitForm(request);
    }
}
```

最后来个简单的测试：







我感觉到了，这就是非常流畅的感觉~

设想一次扩展

如果我们需要加入一个新的策略，比如绑定 FaaS 函数的提交，我们只需要添加一个新的策略实现即可：

```
@Component
public class FormFaasSubmitHandler implements FormSubmitHandler<Serializable> {

    private final Logger logger = LoggerFactory.getLogger(this.getClass());

    @Override
```

```
        public String getSubmitType() { return "faas"; }

        @Override
        public CommonPairResponse<String, Serializable> handleSubmit(FormSubmitRequest request)
{
        logger.info("FaaS 模式的提交：userId={}, formInput={}", request.getUserId(),
request.getFormInput());

        // 进行 FaaS 函数调用，并获得业务方返回的提示信息和业务数据
        CommonPairResponse<String, Serializable> response = faasSubmitData(request);

        return response;
            }
          ...
        }
```

此时不需要修改任何代码，因为 Spring 容器重启时会自动将 FormFaasSubmitHandler 注册到 FormSubmitHandlerFactory 中 —— 面向 Spring 编程，太香惹~

电商互动消息如何进行架构演进?

作者 | 李历岷(骨来)
出品 | 淘系技术

策 2020 年双 11，第一次改变节奏，从光棍节变成双节棍，从一个峰变成了两个峰，在新的挑战下，如何做好技术保障，做好技术支撑，所有技术人都进入了一个新的学习过程。

新的形势下，显著特点是时间跨度长、活动周期长以及用户互动玩法更多。从单用户 CC 分享到群内分享，以及直播间互动消息等，作为电商的消息 IM 承接着整个互动的场地，用户在整个"互动消息"场景下，可以进行实时分享、聊天、客服沟通、商家优惠、商家优惠活动和红包以及商品秒杀等。"消息"作为用户和用户之间架起"连接"的重要桥梁，消息连接了用户和用户、用户和商家、用户和平台等。

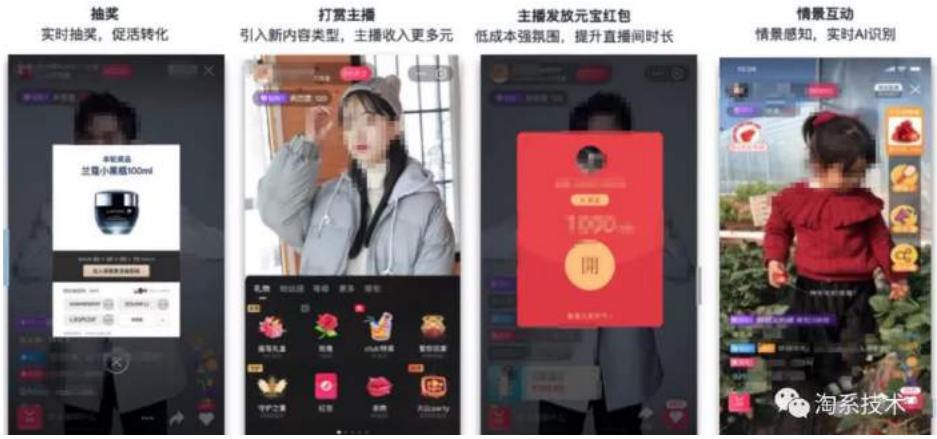
如何做到高并发强互动下消息有序地呈现在用户面前，是消息团队亘古不变的主题，尤其是在用户的互动行为不确定性情况下，在面对不确定性的流量和规模时，系统应该怎样做到"丝般顺滑"的体验呢？本文将带着读者一起来进行深入探讨。

推荐阅读：读者对传统的 IM 系统架构有所了解

互动场景挑战

直播间：淘宝直播带来新的流量变化，百万在线已经成为常态化，大量的直播间 7X24 小时直播带货，用户在直播频道页频繁地进出不同的直播间，有时候被某个直播间吸引，停在那里等着优惠、红包和宝贝，随着主播在直播间喊一嗓子，推送一张宝贝卡片，几十万人蜂拥而至秒杀。这样的场景每天都在上演...

354 < 电商互动消息如何进行架构演进?



互动 PK 群：主互动玩法"超级星秀猫瓜分 20 亿红包"开启的时候，意味着大量互动拉赞、分享到好友、分享到群成为流量的入口，无论淘口令/图片等，大量的消息卡片被转发、二次转发带来的分享裂变效应，每天 09:00 和晚上 22:00 成为活跃的峰值，每个组队的用户拉群互赞，这样的场景从 10 月 20 日到 11 月 11 日持续每天固定时段上演...

不确定性流量：分享面板入口一次列表排布改变，营销活动的一次优惠推送，直播频道页一次运营活动都可能导致直播间/群的流量瞬间波动，纷涌而至的用户带来了大量的互动行为，各种点击/分享/消息发送纷至沓来，如何应对这样的流量和规模，作为平台系统，必须能够做到应对不确定性流量能力和机制，必须从流量入口到流量出口端到端考虑整体的全链路架构，是否存在单点瓶颈和缺口，尤其在大促前期，系统架构梳理、强弱依赖梳理、上下游链路串联、破坏性压测、脉冲流量压测和全链路压测保障和优化，以及配合相关的流量控制、预案保护、业务流量隔离机制、资源调控等手段，才得以做到"不确定性流量"向"确定性流量"转变，才可以更大程度上保障系统高可用和极致用户体验。

问题定义之群聊

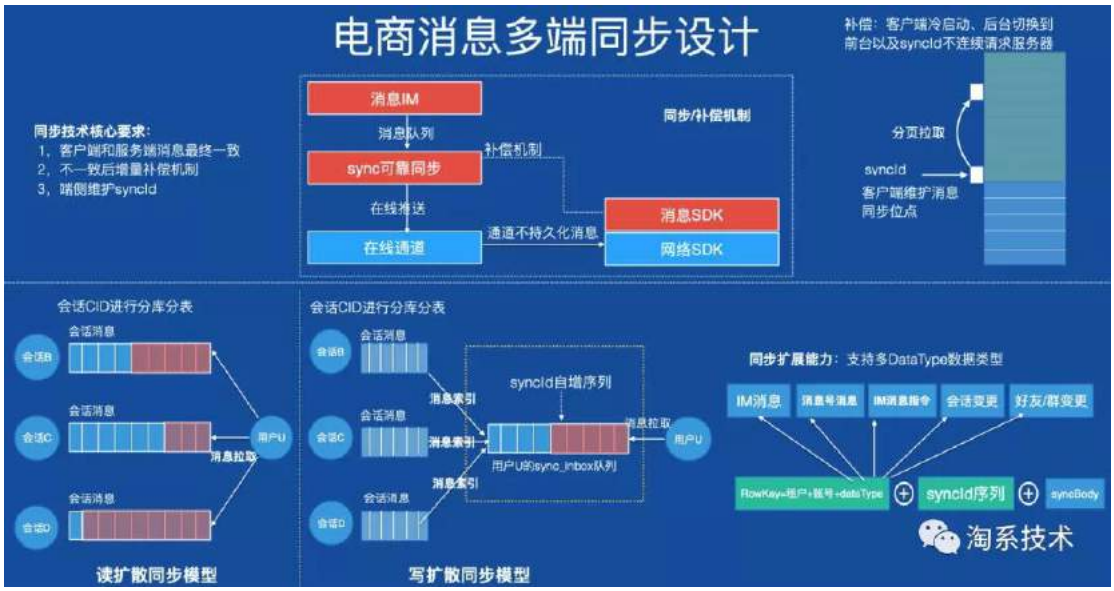
随着 2018 年淘系电商首推"双 11 合伙人计划"，更简单直接的双 11 玩法，给大众带来更多期待和惊喜。尤其是盖楼互赞活动更是把"群聊"推上风口浪尖，在手淘 APP 内部分享比在微信和钉钉等社交/企业软件分享更加便捷和高效，用户不停地在群内互动分享拉赞,通过好友助力提升盖楼积分。

基于传统的 IM 架构技术，尤其在群内聊天或者分享，每条消息按照群内人数进行写扩散，按照主互动 500 人群规模来计算，平均群大小 320+，1:N 的写入必然导致写入 DB 的 RT 以及存储压力，按照 DB 库承接 120w/s 写入速度，导致消息上行 3K/s 的极限，而实际参与互动分享的用户在峰值的时候远大于这部分互动分享和聊天消息流量，其次集群的写入不可能完全给 IM 聊天消息，还有其它的营销活动、交易、物流等通知类型的消息。基于"写扩散"架构，在高并发互动场景下遇到了瓶颈，导致消息大量的延迟下推，影响最终用户体验。

基于写扩散的扩散比 1:N，能否做到 1:1 写入？答案是肯定，针对群内的消息可以分为"非个性化消息"和"个性化消息"，所谓"非个性化消息"就是大家看到都是一样的，应该只需要写一条数据，群内成员可以共享取这条数据，所谓"个性化消息"，指定某个成员发送的单边消息，譬如进群欢迎语等。

在群内，99.99%都是"非个性化消息"，也就是可以从 1:N 压缩到 1:1 写入，基于这个理论假设，需要考虑"读扩散"让每个用户的消息从对应的"群会话消息队列同步"数据，而不是从"

用户队列同步"数据，其中同步队列围绕队列 offset 偏移量进行，通过队列的自增 syncId 保证有序，每个客户端维护相应的队列的同步位点，采取"客户端存储位点的去中心化"方案，实现"下行消息的推拉"结合，通过队列位点 syncId 进行比对，如果服务端消息队列 syncId-客户端队列 syncId=1，表示云端消息无空洞，否则携带客户端的队列和对应的 syncId 到云端重新同步区间数据，实现最终一致性。



业界 IM：腾讯 QQ、腾讯微信、网易云通讯、抖音 IM、钉钉 IM、脉脉 IM、支付宝 IM。

备注：市面上 APP80%都具备 IM 聊天能力，均采取写扩散简单模式进行云端消息同步。

读写扩散技术方案对比

1、写扩散技术

优点：

- 整体架构简洁，方案简单，维护用户同步队列实现数据同步机制。
- 无论是单聊还是群聊等会话消息，写入用户维度同步队列，集中获取同步数据。
- 推和拉情况下，存储模型和数据处理简单，且天然支持个性化数据。

缺点：

- 群会话消息，天然存在 1:N 写入扩散比，存储压力 N 倍压力，在线用户收到消息延迟增大。
- 多个群内消息队列混合在同步队列，无优先级处理能力，无法针对互动群等做隔离。

2、读扩散技术

优点：

- 降低写扩散 N 倍的 DB 存储压力，减少下行在线用户端到端扩散的 RT 时间。
- 提升消息上行集群整体的吞吐量，用户体验更流畅。
- 端到端实现会话级别的同步优先级队列，实现差异化服务。

缺点：

- 整体架构偏复杂，需要维护每个动态会话消息同步队列，端侧需要实时感知新增的动态同步队列。
- 客户端冷启动需要分散读取多个会话消息同步队列数据，对于存储会带来读 QPS 压力。

3、读写扩散混合模式

核心在于架构的演进推进"读写扩散混合模式"落地，结合两者的优点进行云端一体化数据同步技术，覆盖几千万互动群用户，保证整体峰值上行消息以及用户在群内端到端延迟体验，做到一条上行消息 500ms 以内到达群内其他用户端侧，让整体用户体验提升明显，群内强互动成为可能。

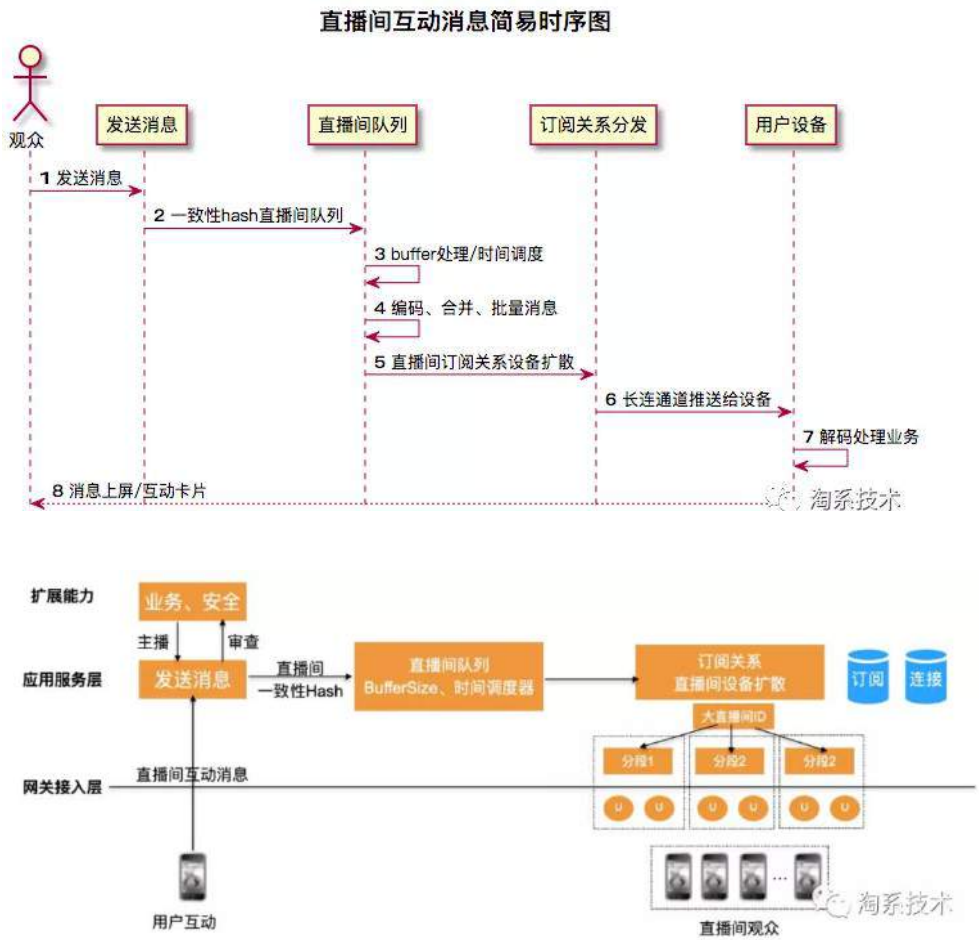
问题定义之直播互动消息

电商直播呈现大直播若干个(大于 30W 同时在线）、中直播间几百个、小直播几万个这样分布，尤其是晚会和主播带来的热点直播间效应，对系统整体的架构存在热点带来严重挑战。

热点问题的产生需要从不确定性的流量来源说起。直播间人员规模天然和群聊不一样(单个群<=500 人），一个直播间可以突破 40 万-200 万之间设备同时在线，直播间在另一个层面是"特殊的群"，群维护了群和群成员强关系，直播间维护直播和设备之间的订阅弱关系，在扩散维度群按照 500 人进行分库分表切片模式分发下行消息，直播间需要以直播间几十万设备作为分段切片进行分发。同时直播间的指令消息如果不加以干预，会形成超大的流量热点和扩散问题，那么针对这个问题如何应对?

直播间互动全链路和核心处理节点

简易时序图如下：观众互动消息->网关接入->应用系统 Buffer(秒级直播间维度消息)->编码、合并、批量消息流->直播间维度设备扩散->设备长连通道推送->设备接收->解码消息->业务处理



充分利用直播间在线人数

针对大直播间和小直播间区分对待，充分利用在线人数这个关键性指标，譬如小直播间不做 buffer 队列处理，所谓 buffer，就是维护 topic 维度的缓冲队列，支持 N 秒或消息条数的异步 flush 机制，实现削峰过程，其中针对"可靠消息"进行持久化缓存存储，如果当前消息推送失败，重试 3 次或者下一条消息再次携带失败的可靠消息，整体按照推拉结合的方式，端侧做重复消息去重控制。

用户进出房间关系维护

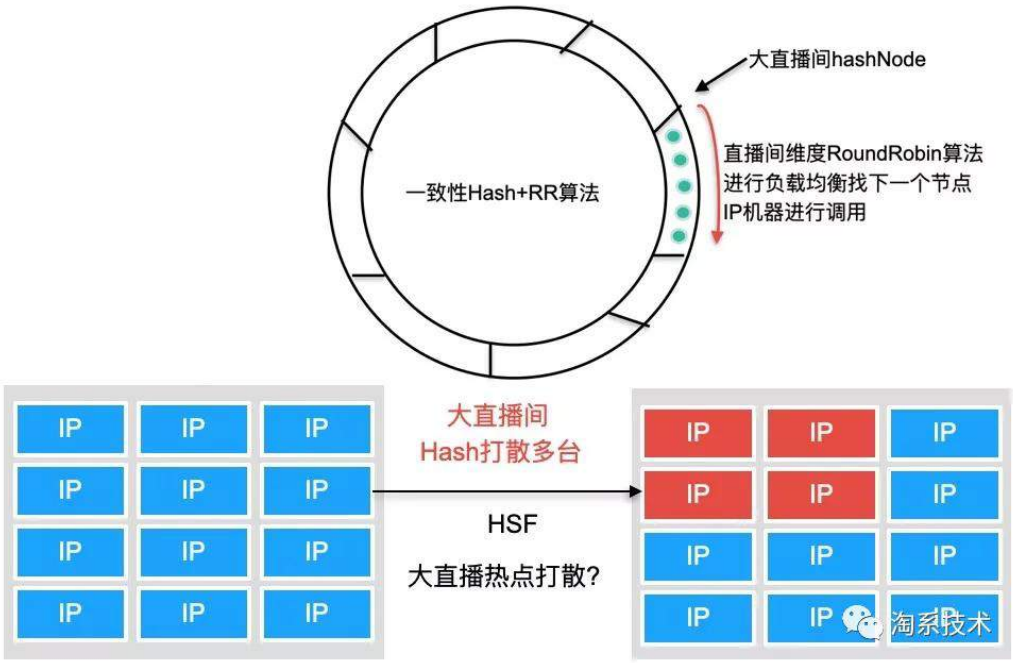
从用户第一次进直播间，发起订阅请求，最终通过大小直播间分片进行路由到指定的直播间，针对特殊大直播间提前做好集群分片，或者通过每天的离线数据分析大直播间的账号，主播开播创建直播间的时候，提前做好分片确定性，在脉冲绑定关系的时候，流量分散到 N 台订阅集群进行绑定关系建立。

流控策略考虑

流控不等于一刀切，而是针对不同的 subType 指令进行控制，针对可靠消息（红包、优惠、宝贝卡片）等进行持久化存储，利用多次消息下推携带机制，同时兼顾端侧拉取机制，保证消息最终一致性且在 3 秒以内到达端侧。

一致性 hash 问题-热点直播间

一致性 Hash 的架构必然存在热点问题，如果在直播间进行分散架构，必然存在指令风暴问题，基于此需要进行 Hash 热点二次打散处理，针对这样的热点问题技术上是可行的，热点问题散列到多台 IP 机器上进行流量承接，另外需要考虑直播间维度的统计数据分布式合并等问题，需要用到分布式锁并发写入统计数据，且直播维度数据合并计算，类似 JDKStriped64 原理。



对比两套架构思考

规模差异化架构思考

因为早期两套系统各自演进应对不同的流量和产品需求，用户规模化和产品要求带来的差异化架构，对比“群聊”和“直播间互动”两类互动场景，群聊基于消息、会话、会话视图以及附加消息存储和消息漫游能力进行整体设计，而对于直播间来说，围绕直播间大规模实时互动进行抽象设计，充分实现 buffer/批量/订阅关系切片等机制。

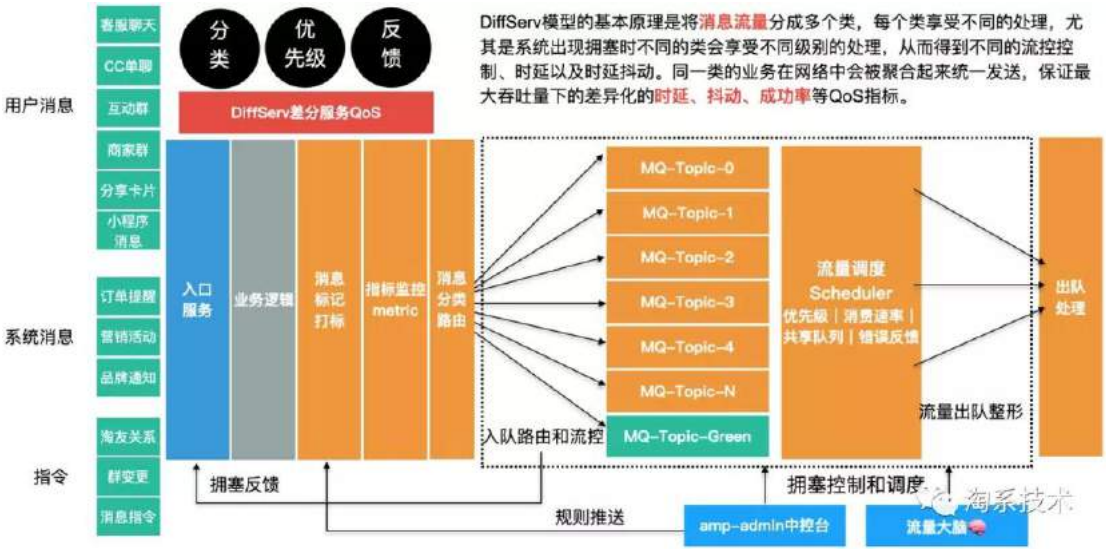
对于关系来说，群关系是强关系，需要用户确认加群，才可以进群、退群、查看群好友等；对于直播间来说是弱关系，用户进直播、退出直播都是自动完成关系建立和取消，当然用户可以后续进入回放直播间。

因此，在功能上和模型设计上需要进一步思考，尤其现有回放直播间需要一套录制/回放指令机制，保存互动指令等关键性指令数据，而这点在消息 IM 场景完全支持了，包括用户可以漫游历史消息等。

技术的发展不会一尘不变，针对合适的场景进行差异化架构和设计才是最重要的，同样在应对不确定性流量这个场景下，解决问题的方式是不完全相同的，这个过程需要反复迭代和优化，围绕端到端用户体验这个核心目标进行技术演进才是最重要的价值选择和方向判断。

不确定性流量的 QoS 思考

不确定性的流量如何转为确定性流量，是技术必须要解决的"确定性"问题。尤其面对流量需要确定性进行分解，分优先级保障不同的消息 QoS 能力是"高可用架构永恒不变"的主题，就像应用系统的限流，需要分场景进行流控一样。基于这样的思考推演，QoS 分级机制来承诺消息服务 SLA，最终才可以做到隔离/优先级/差异化处理，完成整体的消息顺滑体验。



小结

面对不确定性的流量，需要进行流量再细分以及面向不同的场景采取不同的处理方案去应对，惯用的手段是指令合并处理、链路隔离和共享、租户存储隔离、流量打标机制区分场景来源等，以及有相对确定性的流量大脑进行观测，在峰值流量来临之前，需要具备流量感知和预警机制，做好流量的优先级划分，应急预案是快速失败限流还是慢速等待消费以及优先级控制等机制。

同时在系统层面，需要做到水平扩展能力，也就是能否通过弹性扩容的方式应对高流量的峰值，另外需要做到整体全链路合理的架构设计，避免"写入放大"以及"读取放大"的单点问题产生，需要针对入口流量和出口流量进行比对，是否可以做到上下游 1:1 的情况下，尽可能地避免单点瓶颈带来集群瓶颈乃至整体不可用。

"高可用架构"是技术人员永恒不变的主题之一，随着用户规模增长以及集中流量爆发的情形下，更需要敏锐地找到全链路单点问题，提前预判定义出面临的问题，然后给出合理的优化方案，最终灰度切流以及全链路压测验证保证整体方案有序推进，尤其面对的在线系统优化，好比开着飞机换轮胎一样。具备优化效果数据和监控是衡量每一步的预期收益，只有这样才可以做好"架构演进"，才可以持续交付更好的用户产品体验，赢得用户的喜爱，以及互动效果和产品流畅性才可以得到用户满意度提升。架构演进是个动态化的过程，需要持续投入和改进，推动全链路整体落地。

闲鱼基于 Flutter+FaaS 的业务框架思考与实践

作者 | 熊华丽(匠修)
出品 | 淘系技术

前言

闲鱼将使用 Flutter 和 FaaS 来建设未来的技术开发体系，这是一项长期的规划，新的技术在现在看来犹如雾里看花，需要我们不断的思考，探索，实践才能渐渐描绘出它的轮廓。本文对此提供一种思考角度，对未来基于 FaaS+Flutter 之上的编程形态做思考，并介绍自己的初步实践。

Flutter，Faas，与闲鱼的一体化

闲鱼长期在做技术一体化的探索与实践：我们希望使用一门语言，一套技术栈，能让开发工程师在任何场景完成业务开发，实现开发模式和技术栈的统一。这是对开发效率的极致追求，也是对开发人员的深度赋能，更好的释放人员价值，驱动业务成长。

闲鱼已经借助 Flutter 良好的跨栈能力来对 App 上的技术栈做统一，并取得了初步的成果。因此想更近一步的整合前后端，结合 Flutter 来建立统一的技术栈。 FaaS 的兴起给我们带来了新的视角和机会，在后端开发场景中，FaaS 将运行环境和部署运维从日常开发中剥离出来，让开发者更聚焦于业务，降低了后端开发准入门槛，闲鱼基于此已经在做 Flutter+FaaS 的一体化开发体系建设。

技术在发展中会对当前的解决方案不断的抽象，总结和提炼，逐渐分离出其中的变化的部分与不变的部分，让原来的问题变得收敛，开发的关注点会变的更聚焦，开发效率得以提升。这样会出现分层，进而沉淀出基础设施，这也是技术体系演进的一般规律，如图 1-1。

图 1-1

Flutter+FaaS 的技术方案也是如此，我们在当前的中台的基础设施之上建设用于一体化开发的新的基础设施层，让业务开发更加聚焦。由此需要思考两个基本的问题：

- Flutter+FaaS 的技术体系作为支撑一体化开发的基础设施该提供什么样的能力？
- 基于新的一体化基础设施的业务开发是什么样的形态？

图 2-1

这是一体两面的两个问题，只要回答其中一个问题，另一个问题的答案也会变得清晰起来。本文是思考和探索第二个问题。

Flutter+Faas 一体化下的开发形态思考

也许只有到了最后一刻我们才能最终回答这个问题，但是技术总是带着疑问前行，我们先从尝试做顶层做抽象定义开始，然后落地实践，在实践后总结提炼，完善顶层抽象，迭代往复，最终将概念变清晰。

先来看看当前的业务开发形态，如图 2-1，当前在业务开发中几个主要的关注点我大概的归纳为：数据处理，网络通信，状态管理和界面绘制。从这 4 个点出发，逐一思考，在新的一体化的场景下会有什么变化：

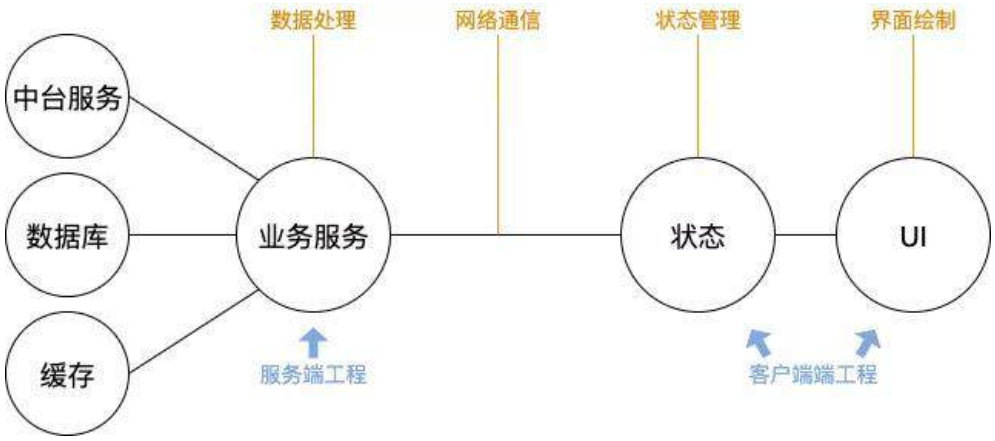


图 2-1

数据处理：泛指传统的服务端 REST API 这部分，在一体化的场景下，这部分会由 FaaS 函数来实现，其实这一部分的职责和定位并没有改变，只是开发的组织沟通形式变了。传统开发页面会由服务端和客户端同学合力完成，后面可能由一位同学前后一体化开发完成。康威定律指出，软件的设计和开发人员的组织沟通结构是息息相关的，所以这一部分可能有较大的变化，但首先是他与客户端的交互方式上的变化，即网络通信。

网络通信：在一体化场景下，前后端都由一人实现，代码也会是一个工程中的同种语言，网络通信会加轻量安全，使用起来也会更加自然，接近于普通的函数调用。一个可能的变化是，通信模式可能会突破 CS 模式，在一次通信"会话"中，Client 端和 Server 端能更加自然的相互调用，实现"对等对话"，而不是传统的客户端请求，服务端响应。随着网络硬件升级，网速在变快，通信成本在下降，创新的空间很大。

状态管理：应用状态很大程度上是缓存在客户端上的数据，受技术体系隔离，开发沟通成本，网络通信成本的影响，在客户端上缓存状态是有必要的。但在一体化的场景中，这些因素的影响在减小和消失，所以我们想进一步的消灭状态，将状态管理降至最小，尽可能的由底层管理。

界面绘制：也许是受 Flutter 的影响，响应式风格结合数据驱动的 UI 理念非常契合我们的思路，这也是业界 UI 开发框架的潮流。

Flutter+FaaS 一体化技术体系下，应用开发应该会更加简单，前后端之间的差异变小，通信更加轻量自然，职责更加聚焦,如图 2-2。

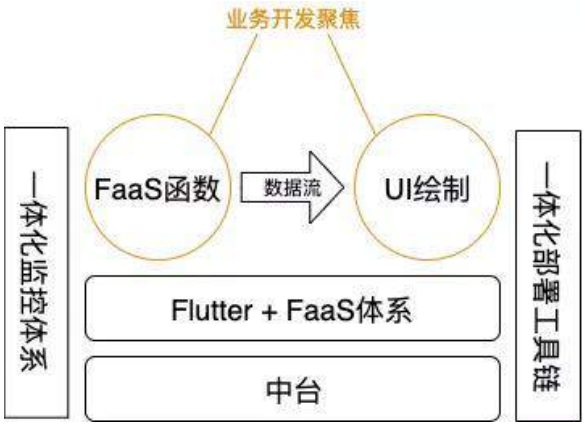


图 2-2

一体化框架设计实践

在一体化的场景下,业务可以由一个同学负责前后端的开发,最大限度的减小沟通协作成本。当然，大体量的业务必然需要协作的，但协作的方式有所改变，工作的划分可以由传统的前后端的横向划分，变成前后一体的垂直划分，如图 3-1，协作方式的改变也会影响到我们设计的思路。

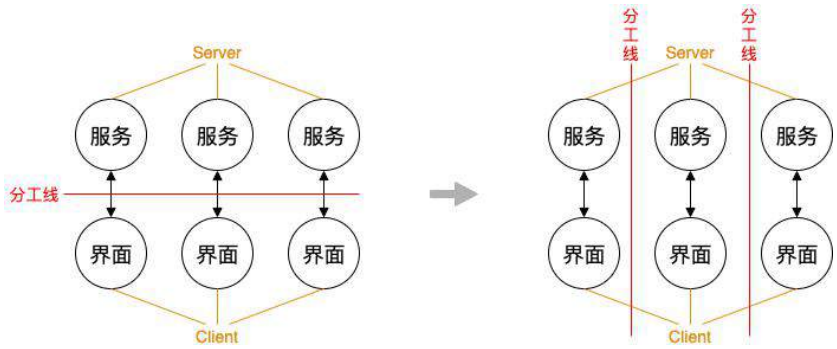


图 3-1

基于前面的讨论，我们尝试做了框架设计：首先我们将要开发的业务命名为一个 Story ，即一个 Story 代表一个产品业务，Story 会按照上面垂直划分的方式，将业务划分为一系列的 Scene。Scene 对应于传统意义上的"页面"的概念，但和产品设计上的页面不强求一一对应，如图 3-2-(1)。Scene 是个前后一体的虚拟概念，运行时并没有实体。它由 3 部分组成，如图 3-2-(2)，Model 部分处理业务数据（RawData）,Converter 将业务数据转换成渲染所使用的数据(State)，最后 Render 使用渲染数据绘制界面。Model 和 Converter 部署在后端，在 FaaS 函数中运行，Render 运行在客户端，他们之间的数据流是单向的。在实现逻辑处理的时候，统一使用事件，事件优先在本端处理，本端不处理则路由到另一端，如果另一端也不处理，则丢弃，如图 3-2-(3)。

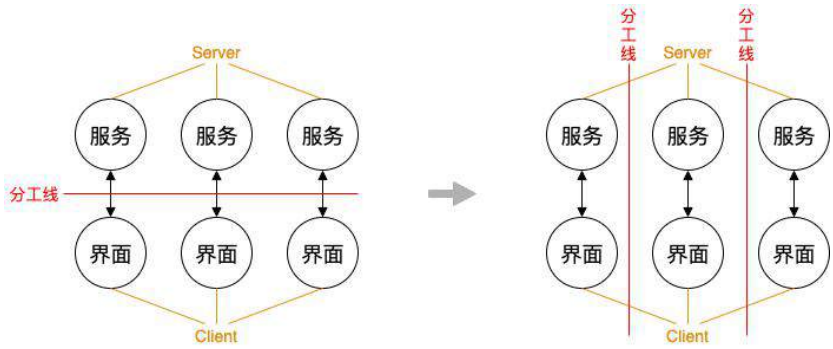


图 3-2

Story 已经开始在闲鱼的业务中落地实践，后期带来实践效果的分享。

实践中的挑战与借鉴

要建设完善一体化技术体系不是容易的事情，实践中肯定会有诸多挑战。好消息是 FaaS 和它背后的 Serverless 理念已经是业内潮流了，实践也在如火如荼的进行中。其中，阿里的前端同学集中力量，对 Serverless 已经实践的很深入了，虽然并不是一体化的理念，但实践上很多的思路可以做相互印证和借鉴：

开发部署工具链：闲鱼一体化工程依赖于底层开发部署工具链的支持，是基础设施的一部分，部署也是 Serverless 体系中关键一环，从前端同学的实践中看，工具，插件和平台都有。闲鱼有直接使用的平台，也会有自建的工具插件，长远会做一步的体系标准化建设。

全链路追踪监控：这是工程化必须的系统，闲鱼直接受益于集团的平台，但也会建设自己特殊场景的工具，比如借鉴 Flutter 的函数热更新，本地一体化调试等。

当然，闲鱼也有自己感兴趣探索的方向：

网络通信：我们的很多设想对网络通信有着很高的要求，诚然，网络质量会越来越好，成本也会也来越低，但弱网场景总会存在，如何保证服务的稳定可用依然是挑战。一体化会弱化开发对网络的感知，提供新的网络使用方式。

组件分治：在闲鱼的业务中，复杂页面是始终要考虑的问题。在前面的设计中，我们对此做了预留(Converter 部分)，但前后一体化的组件该怎么抽象，又如何组合复用，我们还在探索中。

阿里技术专家详解 DDD 系列 第一讲 - Domain Primitive

作者 | 殷浩
出品 | 淘系技术

导读：对于一个架构师来说，在软件开发中如何降低系统复杂度是一个永恒的挑战，无论是 94 年 GoF 的 Design Patterns ， 99 年的 Martin Fowler 的 Refactoring ，02 年的 P of EAA ，还是 03 年的 Enterprise Integration Patterns ，都是通过一系列的设计模式或范例来降低一些常见的复杂度。但是问题在于，这些书的理念是通过技术手段解决技术问题，但并没有从根本上解决业务的问题。所以 03 年 Eric Evans 的 Domain Driven Design 一书，以及后续 Vaughn Vernon 的 Implementing DDD ，Uncle Bob 的 Clean Architecture 等书，真正的从业务的角度出发，为全世界绝大部分做纯业务的开发提供了一整套的架构思路。

前言

由于 DDD 不是一套框架，而是一种架构思想，所以在代码层面缺乏了足够的约束，导致 DDD 在实际应用中上手门槛很高，甚至可以说绝大部分人都对 DDD 的理解有所偏差。举个例子， Martin Fowler 在他个人博客里描述的一个 Anti-pattern，Anemic Domain Model ①（贫血域模型）在实际应用当中层出不穷，而一些仍然火热的 ORM 工具比如 Hibernate，Entity Framework 实际上助长了贫血模型的扩散。同样的，传统的基于数据库技术以及 MVC 的四层应用架构（UI、Business、Data Access、Database），在一定程度上和 DDD 的一些概念混淆，导致绝大部分人在实际应用当中仅仅用到了 DDD 的建模的思想，而其对于整个架构体系的思想无法落地。

我第一次接触 DDD 应该是 2012 年，当时除了大型互联网公司，基本上商业应用都还处于单机的时代，服务化的架构还局限于单机 +LB 用 MVC 提供 Rest 接口供外部调用，或

者用 SOAP 或 WebServices 做 RPC 调用，但其实更多局限于对外部依赖的协议。让我关注到 DDD 思想的是一个叫 Anti-Corruption Layer（防腐层）的概念，特别是其在解决外部依赖频繁变更的情况下，如何将核心业务逻辑和外部依赖隔离的机制。到了 2014 年，SOA 开始大行其道，微服务的概念开始冒头，而如何将一个 Monolith 应用合理的拆分为多个微服务成为了各大论坛的热门话题，而 DDD 里面的 Bounded Context（限界上下文）的思想为微服务拆分提供了一套合理的框架。而在今天，在一个所有的东西都能被称之为"服务"的时代（XAAS）， DDD 的思想让我们能冷静下来，去思考到底哪些东西可以被服务化拆分，哪些逻辑需要聚合，才能带来最小的维护成本，而不是简单的去追求开发效率。

所以今天，我开始这个关于 DDD 的一系列文章，希望能继续在总结前人的基础上发扬光大 DDD 的思想，但是通过一套我认为合理的代码结构、框架和约束，来降低 DDD 的实践门槛，提升代码质量、可测试性、安全性、健壮性。
未来会覆盖的内容包括：

- 最佳架构实践：六边形应用架构 / Clean 架构的核心思想和落地方案
- 持续发现和交付：Event Storming > Context Map > Design Heuristics > Modelling
- 降低架构腐败速度：通过 Anti-Corruption Layer 集成第三方库的模块化方案
- 标准组件的规范和边界：Entity, Aggregate, Repository, Domain Service, Application Service, Event, DTO Assembler 等
- 基于 Use Case 重定义应用服务的边界
- 基于 DDD 的微服务化改造及颗粒度控制
- CQRS 架构的改造和挑战
- 基于事件驱动的架构的挑战
- 等等

今天先给大家带来一篇最基础，但极其有价值的 Domain Primitive 的概念。

Domain Primitive

就好像在学任何语言时首先需要了解的是基础数据类型一样，在全面了解 DDD 之前，首先给大家介绍一个最基础的概念：Domain Primitive（DP）。

Primitive 的定义是：

- 不从任何其他事物发展而来
- 初级的形成或生长的早期阶段

就好像 Integer、String 是所有编程语言的 Primitive 一样，在 DDD 里， DP 可以说是一切模型、方法、架构的基础，而就像 Integer、String 一样， DP 又是无所不在的。所以，第一讲会对 DP 做一个全面的介绍和分析，但我们先不去讲概念，而是从案例入手，看看为什么 DP 是一个强大的概念。

案例分析

我们先看一个简单的例子，这个 case 的业务逻辑如下：
一个新应用在全国通过 地推业务员 做推广，需要做一个用户注册系统，同时希望在用户注册后能够通过用户电话（先假设仅限座机）的地域（区号）对业务员发奖金。

先不要去纠结这个根据用户电话去发奖金的业务逻辑是否合理，也先不要去管用户是否应该在注册时和业务员做绑定，这里我们看的主要还是如何更加合理的去实现这个逻辑。一个简单的用户和用户注册的代码实现如下：

```
public class User {
    Long userId;
    String name;
    String phone;
    String address;
    Long repId;
}

public class RegistrationServiceImpl implements RegistrationService {
```

```
    private SalesRepRepository salesRepRepo;
    private UserRepository userRepo;

    public User register(String name, String phone, String address)
      throws ValidationException {
        // 校验逻辑
        if (name == null || name.length() == 0) {
            throw new ValidationException("name");
        }
        if (phone == null || !isValidPhoneNumber(phone)) {
            throw new ValidationException("phone");
        }
        // 此处省略 address 的校验逻辑

        // 取电话号里的区号，然后通过区号找到区域内的 SalesRep
        String areaCode = null;
        String[] areas = new String[]{"0571", "021", "010"};
        for (int i = 0; i < phone.length(); i++) {
            String prefix = phone.substring(0, i);
            if (Arrays.asList(areas).contains(prefix)) {
                areaCode = prefix;
                break;
            }
        }
        SalesRep rep = salesRepRepo.findRep(areaCode);

        // 最后创建用户，落盘，然后返回
        User user = new User();
        user.name = name;
        user.phone = phone;
        user.address = address;
        if (rep != null) {
            user.repId = rep.repId;
        }

        return userRepo.save(user);
    }

    private boolean isValidPhoneNumber(String phone) {
        String pattern = "^0[1-9]{2,3}-?\\d{8}$";
        return phone.matches(pattern);
    }
}
```

我们日常绝大部分代码和模型其实都跟这个是类似的，乍一看貌似没啥问题，但我们再深入一步，从以下四个维度去分析一下：接口的清晰度（可阅读性）、数据验证和错误处理、业务逻辑代码的清晰度、和可测试性。

问题 1 - 接口的清晰度

在 Java 代码中，对于一个方法来说所有的参数名在编译时丢失，留下的仅仅是一个参数类型的列表，所以我们重新看一下以上的接口定义，其实在运行时仅仅是：

```
User register(String, String, String);
```

所以以下的代码是一段编译器完全不会报错的，很难通过看代码就能发现的 bug ：

```
service.register("殷浩", "浙江省杭州市余杭区文三西路 969 号", "0571-12345678");
```

当然，在真实代码中运行时会报错，但这种 bug 是在运行时被发现的，而不是在编译时。普通的 Code Review 也很难发现这种问题，很有可能是代码上线后才会被暴露出来。这里的思考是，有没有办法在编码时就避免这种可能会出现的问题？
另外一种常见的，特别是在查询服务中容易出现的例子如下：

```
User findByName(String name);
User findByPhone(String phone);
User findByNameAndPhone(String name, String phone);
```

在这个场景下，由于入参都是 String 类型，不得不在方法名上面加上 `ByXXX` 来区分，而 `findByNameAndPhone` 同样也会陷入前面的入参顺序错误的问题，而且和前面的入参不同，这里参数顺序如果输错了，方法不会报错只会返回 `null`，而这种 bug 更加难被发现。这里的思考是，有没有办法让方法入参一目了然，避免入参错误导致的 bug ？

问题 2 - 数据验证和错误处理

在前面这段数据校验代码：

```
if (phone == null || !isValidPhoneNumber(phone)) {
    throw new ValidationException("phone");
}
```

在日常编码中经常会出现，一般来说这种代码需要出现在方法的最前端，确保能够 fail-fast。但是假设你有多个类似的接口和类似的入参，在每个方法里这段逻辑会被重复。而更严重的是如果未来我们要拓展电话号去包含手机时，很可能需要加入以下代码：

```
if (phone == null || !isValidPhoneNumber(phone) || !isValidCellNumber(phone)) {
    throw new ValidationException("phone");
}
```

如果你有很多个地方用到了 phone 这个入参，但是有个地方忘记修改了，会造成 bug 。这是一个 DRY 原则被违背时经常会发生的问题。

如果有个新的需求，需要把入参错误的原因返回，那么这段代码就变得更加复杂：

```
if (phone == null) {
    throw new ValidationException("phone 不能为空");
} else if (!isValidPhoneNumber(phone)) {
    throw new ValidationException("phone 格式错误");
}
```

可以想像得到，代码里充斥着大量的类似代码块时，维护成本要有多高。

最后，在这个业务方法里，会（隐性或显性的）抛 ValidationException，所以需要外部调用方去 try/catch，而业务逻辑异常和数据校验异常被混在了一起，是否是合理的？

在传统 Java 架构里有几个办法能够去解决一部分问题，常见的如 BeanValidation 注解或 ValidationUtils 类，比如：

```
// Use Bean Validation
User registerWithBeanValidation(
  @NotNull @NotBlank String name,
  @NotNull @Pattern(regexp = "^0?[1-9]{2,3}-?\\d{8}$") String phone,
  @NotNull String address
);

// Use ValidationUtils:
public User registerWithUtils(String name, String phone, String address) {
    ValidationUtils.validateName(name); // throws ValidationException
    ValidationUtils.validatePhone(phone);
    ValidationUtils.validateAddress(address);
    ...
}
```

但这几个传统的方法同样有问题，

BeanValidation：

- 通常只能解决简单的校验逻辑，复杂的校验逻辑一样要写代码实现定制校验器。
- 在添加了新校验逻辑时，同样会出现在某些地方忘记添加一个注解的情况，DRY 原则还是会被违背。

ValidationUtils 类：

- 当大量的校验逻辑集中在一个类里之后，违背了 Single Responsibility 单一性原则，导致代码混乱和不可维护。
- 业务异常和校验异常还是会混杂。

所以，有没有一种方法，能够一劳永逸的解决所有校验的问题以及降低后续的维护成本和异常处理成本呢？

问题 3 - 业务代码的清晰度

在这段代码里：

```
String areaCode = null;
String[] areas = new String[]{"0571", "021", "010"};
for (int i = 0; i < phone.length(); i++) {
    String prefix = phone.substring(0, i);
    if (Arrays.asList(areas).contains(prefix)) {
        areaCode = prefix;
        break;
    }
}
SalesRep rep = salesRepRepo.findRep(areaCode);
```

实际上出现了另外一种常见的情况，那就是从一些入参里抽取一部分数据，然后调用一个外部依赖获取更多的数据，然后通常从新的数据中再抽取部分数据用作其他的作用。这种代码通常被称作"胶水代码"，其本质是由于外部依赖的服务的入参并不符合我们原始的入参导致的。比如，如果 SalesRepRepository 包含一个 findRepByPhone 的方法，则上面大部分的代码都不必要了。

所以，一个常见的办法是将这段代码抽离出来，变成独立的一个或多个方法：

```
private static String findAreaCode(String phone) {
    for (int i = 0; i < phone.length(); i++) {
        String prefix = phone.substring(0, i);
        if (isAreaCode(prefix)) {
            return prefix;
        }
    }
    return null;
}

private static boolean isAreaCode(String prefix) {
    String[] areas = new String[]{"0571", "021"};
    return Arrays.asList(areas).contains(prefix);
}
```

然后原始代码变为：

```
String areaCode = findAreaCode(phone);
SalesRep rep = salesRepRepo.findRep(areaCode);
```

而为了复用以上的方法，可能会抽离出一个静态工具类 PhoneUtils 。但是这里要思考的是，静态工具类是否是最好的实现方式呢？当你的项目里充斥着大量的静态工具类，业务代码散在多个文件当中时，你是否还能找到核心的业务逻辑呢？

问题 4 - 可测试性

为了保证代码质量，每个方法里的每个入参的每个可能出现的条件都要有 TC 覆盖（假设我们先不去测试内部业务逻辑），所以在我们这个方法里需要以下的 TC ：

条件 \ 入参	name	phone	address
入参为null	✓	✓	✓
入参为空	✓	✓	✓
入参不符合要求（可能多个）	✓	✓	✓

假如一个方法有 N 个参数，每个参数有 M 个校验逻辑，至少要有 N * M 个 TC 。

如果这时候在该方法中加入一个新的入参字段 `fax` ，即使 `fax` 和 `phone` 的校验逻辑完全一致，为了保证 TC 覆盖率，也一样需要 M 个新的 TC 。

而假设有 P 个方法中都用到了 `phone` 这个字段，这 P 个方法都需要对该字段进行测试，也就是说整体需要：

P * N * M

个测试用例才能完全覆盖所有数据验证的问题，在日常项目中，这个测试的成本非常之高，导致大量的代码没被覆盖到。而没被测试覆盖到的代码才是最有可能出现问题的地方。

在这个情况下，降低测试成本 == 提升代码质量，如何能够降低测试的成本呢？

解决方案

我们回头先重新看一下原始的 use case，并且标注其中可能重要的概念：

一个新应用在全国通过 地推业务员 做推广，需要做一个用户的注册系统，在用户注册后能够通过用户电话号的区号对业务员发奖金。

在分析了 use case 后，发现其中地推业务员、用户本身自带 ID 属性，属于 Entity（实体），而注册系统属于 Application Service（应用服务），这几个概念已经有存在。但是发现电话号这个概念却完全被隐藏到了代码之中。我们可以问一下自己，取电话号的区号的逻辑是否属于用户（用户的区号？）？是否属于注册服务（注册的区号？）？如果都不是很贴切，那就说明这个逻辑应该属于一个独立的概念。所以这里引入我们第一个原则：

Make Implicit Concepts Explicit
将隐性的概念显性化

在这里，我们可以看到，原来电话号仅仅是用户的一个参数，属于隐形概念，但实际上电话号的区号才是真正的业务逻辑，而我们需要将电话号的概念显性化，通过写一个 Value Object：

```
public class PhoneNumber {

    private final String number;
    public String getNumber() {
        return number;
    }

    public PhoneNumber(String number) {
        if (number == null) {
            throw new ValidationException("number 不能为空");
        } else if (isValid(number)) {
```

```
            throw new ValidationException("number 格式错误");
        }
        this.number = number;
    }

    public String getAreaCode() {
        for (int i = 0; i < number.length(); i++) {
            String prefix = number.substring(0, i);
            if (isAreaCode(prefix)) {
                return prefix;
            }
        }
        return null;
    }

    private static boolean isAreaCode(String prefix) {
        String[] areas = new String[]{"0571", "021", "010"};
        return Arrays.asList(areas).contains(prefix);
    }

    public static boolean isValid(String number) {
        String pattern = "^0?[1-9]{2,3}-?\\d{8}$";
        return number.matches(pattern);
    }

}
```

这里面有几个很重要的元素：

1. 通过 `private final String number` 确保 `PhoneNumber` 是一个（Immutable）Value Object。（一般来说 VO 都是 Immutable 的，这里只是重点强调一下）
2. 校验逻辑都放在了 constructor 里面，确保只要 `PhoneNumber` 类被创建出来后，一定是校验通过的。
3. 之前的 `findAreaCode` 方法变成了 `PhoneNumber` 类里的 `getAreaCode` ，突出了 `areaCode` 是 `PhoneNumber` 的一个计算属性。

这样做完之后，我们发现把 `PhoneNumber` 显性化之后，其实是生成了一个 Type（数据类型）和一个 Class（类）：

- Type 指我们在今后的代码里可以通过 PhoneNumber 去显性的标识电话号这个概念。
- Class 指我们可以把所有跟电话号相关的逻辑完整的收集到一个文件里。

这两个概念加起来，构造成了本文标题的 Domain Primitive（DP）。

我们看一下全面使用了 DP 之后效果：

```
public class User {
    UserId userId;
    Name name;
    PhoneNumber phone;
    Address address;
    RepId repId;
}

public User register(
  @NotNull Name name,
  @NotNull PhoneNumber phone,
  @NotNull Address address
) {
    // 找到区域内的 SalesRep
    SalesRep rep = salesRepRepo.findRep(phone.getAreaCode());

    // 最后创建用户，落盘，然后返回，这部分代码实际上也能用 Builder 解决
    User user = new User();
    user.name = name;
    user.phone = phone;
    user.address = address;
    if (rep != null) {
        user.repId = rep.repId;
    }

    return userRepo.saveUser(user);
}
```

我们可以看到在使用了 DP 之后，所有的数据验证逻辑和非业务流程的逻辑都消失了，剩下都是核心业务逻辑，可以一目了然。我们重新用上面的四个维度评估一下：

评估 1 - 接口的清晰度

重构后的方法签名变成了很清晰的：

```
public User register(Name, PhoneNumber, Address)
```

而之前容易出现的 bug，如果按照现在的写法

```
service.register(new Name("殷浩"), new Address("浙江省杭州市余杭区文三西路 969 号"), new PhoneNumber("0571-12345678"));
```

让接口 API 变得很干净，易拓展。

评估 2 - 数据验证和错误处理

```
public User register(
  @NotNull Name name,
  @NotNull PhoneNumber phone,
  @NotNull Address address
) // no throws
```

如前文代码展示的，重构后的方法里，完全没有了任何数据验证的逻辑，也不会抛 ValidationException 。原因是因为 DP 的特性，只要是能够带到入参里的一定是正确的或 null（Bean Validation 或 lombok 的注解能解决 null 的问题）。所以我们把数据验证的工作量前置到了调用方，而调用方本来就是应该提供合法数据的，所以更加合适。

再展开来看，使用 DP 的另一个好处就是代码遵循了 DRY 原则和单一性原则，如果未来需要修改 `PhoneNumber` 的校验逻辑，只需要在一个文件里修改即可，所有使用到了 `PhoneNumber` 的地方都会生效。

评估 3 - 业务代码的清晰度

```
SalesRep rep = salesRepRepo.findRep(phone.getAreaCode());
User user = xxx;
return userRepo.save(user);
```

除了在业务方法里不需要校验数据之外，原来的一段胶水代码 findAreaCode 被改为了 PhoneNumber 类的一个计算属性 getAreaCode ，让代码清晰度大大提升。而且胶水代码通常都不可复用，但是使用了 DP 后，变成了可复用、可测试的代码。我们能看到，在刨除了数据验证代码、胶水代码之后，剩下的都是核心业务逻辑。（ Entity 相关的重构在后面文章会谈到，这次先忽略）

评估 4 - 可测试性

条件 \ 入参	PhoneNumber	phone	fax
入参为null	✓	✓	✓
入参为空	✓		
入参不符合要求（可能多个）	✓		

当我们将 PhoneNumber 抽取出来之后，在来看测试的 TC ：

- 首先 PhoneNumber 本身还是需要 M 个测试用例，但是由于我们只需要测试单一对象，每个用例的代码量会大大降低，维护成本降低。
- 每个方法里的每个参数，现在只需要覆盖为 null 的情况就可以了，其他的 case 不可能发生（因为只要不是 null 就一定是合法的）

所以，单个方法的 TC 从原来的 N * M 变成了今天的 N + M 。同样的，多个方法的 TC 数量变成了

N + M + P

这个数量一般来说要远低于原来的数量 N* M * P ，让测试成本极大的降低。

评估总结

进阶使用

在上文我介绍了 DP 的第一个原则：将隐性的概念显性化。在这里我将介绍 DP 的另外两个原则，用一个新的案例。

案例 1 - 转账

假设现在要实现一个功能，让 A 用户可以支付 x 元给用户 B ，可能的实现如下：

```
public void pay(BigDecimal money, Long recipientId) {
    BankService.transfer(money, "CNY", recipientId);
}
```

如果这个是境内转账，并且境内的货币永远不变，该方法貌似没啥问题，但如果有一天货币变更了（比如欧元区曾经出现的问题），或者我们需要做跨境转账，该方法是明显的 bug ，因为 `money` 对应的货币不一定是 CNY 。

在这个 case 里，当我们说"支付 x 元"时，除了 x 本身的数字之外，实际上是有一个隐含的概念那就是货币"元"。但是在原始的入参里，之所以只用了 `BigDecimal` 的原因是我们认为 CNY 货币是默认的，是一个隐含的条件，但是在我们写代码时，需要把所有隐性的条件显性化，而这些条件整体组成当前的上下文。所以 DP 的第二个原则是：

Make Implicit Context Explicit

将 隐性的 上下文 显性化

所以当我们做这个支付功能时，实际上需要的一个入参是支付金额 + 支付货币。我们可以把这两个概念组合成为一个独立的完整概念：Money。

```
@Value
public class Money {
    private BigDecimal amount;
    private Currency currency;
    public Money(BigDecimal amount, Currency currency) {
        this.amount = amount;
        this.currency = currency;
    }
}
```

而原有的代码则变为：

```
public void pay(Money money, Long recipientId) {
    BankService.transfer(money, recipientId);
}
```

通过将默认货币这个隐性的上下文概念显性化，并且和金额合并为 Money ，我们可以避免很多当前看不出来，但未来可能会暴雷的 bug。

案例 2 - 跨境转账

前面的案例升级一下，假设用户可能要做跨境转账从 CNY 到 USD ，并且货币汇率随时在波动：

```
public void pay(Money money, Currency targetCurrency, Long recipientId) {
    if (money.getCurrency().equals(targetCurrency)) {
        BankService.transfer(money, recipientId);
    } else {
        BigDecimal rate = ExchangeService.getRate(money.getCurrency(), targetCurrency);
        BigDecimal targetAmount = money.getAmount().multiply(new BigDecimal(rate));
        Money targetMoney = new Money(targetAmount, targetCurrency);
        BankService.transfer(targetMoney, recipientId);
    }
}
```

在这个 case 里，由于 targetCurrency 不一定和 money 的 Curreny 一致，需要调用一个服务去取汇率，然后做计算。最后用计算后的结果做转账。

这个 case 最大的问题在于，金额的计算被包含在了支付的服务中，涉及到的对象也有 2 个 Currency ， 2 个 Money ， 1 个 BigDecimal ， 总共 5 个对象。这种涉及到多个对象的业务逻辑，需要用 DP 包装掉，所以这里引出 DP 的第三个原则：

Encapsulate Multi-Object Behavior

封装 多对象 行为

在这个 case 里，可以将转换汇率的功能，封装到一个叫做 ExchangeRate 的 DP 里：

```
@Value
public class ExchangeRate {
    private BigDecimal rate;
    private Currency from;
    private Currency to;

    public ExchangeRate(BigDecimal rate, Currency from, Currency to) {
        this.rate = rate;
        this.from = from;
        this.to = to;
    }

    public Money exchange(Money fromMoney) {
        notNull(fromMoney);
        isTrue(this.from.equals(fromMoney.getCurrency()));
        BigDecimal targetAmount = fromMoney.getAmount().multiply(rate);
        return new Money(targetAmount, to);
    }
}
```

ExchangeRate 汇率对象，通过封装金额计算逻辑以及各种校验逻辑，让原始代码变得极其简单：

```
public void pay(Money money, Currency targetCurrency, Long recipientId) {
    ExchangeRate rate = ExchangeService.getRate(money.getCurrency(), targetCurrency);
    Money targetMoney = rate.exchange(money);
    BankService.transfer(targetMoney, recipientId);
}
```

讨论和总结

Domain Primitive 的定义

让我们重新来定义一下 Domain Primitive：Domain Primitive 是一个在特定领域里，拥有精准定义的、可自我验证的、拥有行为的 Value Object。

- DP 是一个传统意义上的 Value Object，拥有 Immutable 的特性
- DP 是一个完整的概念整体，拥有精准定义
- DP 使用业务域中的原生语言
- DP 可以是业务域的最小组成部分、也可以构建复杂组合

注：Domain Primitive 的概念和命名来自于 Dan Bergh Johnsson & Daniel Deogun 的书 Secure by Design。

使用 Domain Primitive 的三原则

- 让隐性的概念显性化
- 让隐性的上下文显性化
- 封装多对象行为

Domain Primitive 和 DDD 里 Value Object 的区别

在 DDD 中， Value Object 这个概念其实已经存在：

- 在 Evans 的 DDD 蓝皮书中，Value Object 更多的是一个非 Entity 的值对象
- 在 Vernon 的 IDDD 红皮书中，作者更多的关注了 Value Object 的 Immutability、Equals 方法、Factory 方法等

Domain Primitive 是 Value Object 的进阶版，在原始 VO 的基础上要求每个 DP 拥有概念的整体，而不仅仅是值对象。在 VO 的 Immutable 基础上增加了 Validity 和行为。当然同样的要求无副作用（side-effect free）。

Domain Primitive 和 Data Transfer Object (DTO) 的区别

在日常开发中经常会碰到的另一个数据结构是 DTO ，比如方法的入参和出参。DP 和 DTO 的区别如下：

	DTO	DP
功能	数据传输 属于技术细节	代表业务域中的概念
数据的关联	只是一堆数据放在一起 不一定有关联度	数据之间的高相关性
行为	无行为	丰富的行为和业务逻辑

什么情况下应该用 Domain Primitive

常见的 DP 的使用场景包括：

- 有格式限制的 String：比如 Name，PhoneNumber，OrderNumber，ZipCode，Address 等。
- 有限制的 Integer：比如 OrderId（>0），Percentage（0-100%），Quantity（>=0）等。
- 可枚举的 int ：比如 Status（一般不用 Enum 因为反序列化问题）。

- Double 或 BigDecimal：一般用到的 Double 或 BigDecimal 都是有业务含义的，比如 Temperature、Money、Amount、ExchangeRate、Rating 等。
- 复杂的数据结构：比如 Map<String，List<Integer>> 等，尽量能把 Map 的所有操作包装掉，仅暴露必要行为。

实战 - 老应用重构的流程

在新应用中使用 DP 是比较简单的，但在老应用中使用 DP 是可以遵循以下流程按部就班的升级。在此用本文的第一个 case 为例。

第一步 - 创建 Domain Primitive，收集所有 DP 行为

在前文中，我们发现取电话号的区号这是一个可以独立出来的、可以放入 PhoneNumber 这个 Class 的逻辑。类似的，在真实的项目中，以前散落在各个服务或工具类里面的代码，可以都抽出来放在 DP 里，成为 DP 自己的行为或属性。这里面的原则是：所有抽离出来的方法要做到无状态，比如原来是 static 的方法。如果原来的方法有状态变更，需要将改变状态的部分和不改状态的部分分离，然后将无状态的部分融入 DP 。因为 DP 本身不能带状态，所以一切需要改变状态的代码都不属于 DP 的范畴。

(代码参考 PhoneNumber 的代码，这里不再重复)

第二步 - 替换数据校验和无状态逻辑

为了保障现有方法的兼容性，在第二步不会去修改接口的签名，而是通过代码替换原有的校验逻辑和根 DP 相关的业务逻辑。比如：

```
public User register(String name, String phone, String address)
        throws ValidationException {
    if (name == null || name.length() == 0) {
        throw new ValidationException("name");
    }
    if (phone == null || !isValidPhoneNumber(phone)) {
```

```
            throw new ValidationException("phone");
        }

        String areaCode = null;
        String[] areas = new String[]{"0571", "021", "010"};
        for (int i = 0; i < phone.length(); i++) {
            String prefix = phone.substring(0, i);
            if (Arrays.asList(areas).contains(prefix)) {
                areaCode = prefix;
                break;
            }
        }
        SalesRep rep = salesRepRepo.findRep(areaCode);
        // 其他代码...
}
```

通过 DP 替换代码后：

```
public User register(String name, String phone, String address)
            throws ValidationException {

        Name _name = new Name(name);
        PhoneNumber _phone = new PhoneNumber(phone);
        Address _address = new Address(address);

        SalesRep rep = salesRepRepo.findRep(_phone.getAreaCode());
        // 其他代码...
}
```

通过 new PhoneNumber(phone) 这种代码，替代了原有的校验代码。

通过 _phone.getAreaCode() 替换了原有的无状态的业务逻辑。

第三步 - 创建新接口

创建新接口，将 DP 的代码提升到接口参数层：

```
public User register(Name name, PhoneNumber phone, Address address) {
    SalesRep rep = salesRepRepo.findRep(phone.getAreaCode());
}
```

第四步 - 修改外部调用

外部调用方需要修改调用链路，比如：

```
service.register("殷浩", "0571-12345678", "浙江省杭州市余杭区文三西路 969 号");
```

改为：

```
service.register(new Name("殷浩"), new PhoneNumber("0571-12345678"), new Address("浙江省杭州市余杭区文三西路 969 号"));
```

通过以上 4 步，就能让你的代码变得更加简洁、优雅、健壮、安全。你还在等什么？今天就去尝试吧！

链接：

https://martinfowler.com/bliki/AnemicDomainModel.html

阿里技术专家详解 DDD 系列 第二讲 - 应用架构

作者 | 殷浩
出品 | 淘系技术

架构这个词源于英文里的"Architecture"，源头是土木工程里的"建筑"和"结构"，而架构里的"架"同时又包含了"架子"（scaffolding）的含义，意指能快速搭建起来的固定结构。而今天的应用架构，意指软件系统中固定不变的代码结构、设计模式、规范和组件间的通信方式。在应用开发中架构之所以是最重要的第一步，因为一个好的架构能让系统安全、稳定、快速迭代。在一个团队内通过规定一个固定的架构设计，可以让团队内能力参差不齐的同学们都能有一个统一的开发规范，降低沟通成本，提升效率和代码质量。

在做架构设计时，一个好的架构应该需要实现以下几个目标：

- 独立于框架：架构不应该依赖某个外部的库或框架，不应该被框架的结构所束缚。
- 独立于 UI：前台展示的样式可能会随时发生变化（今天可能是网页、明天可能变成 console、后天是独立 app），但是底层架构不应该随之而变化。
- 独立于底层数据源：无论今天你用 MySQL、Oracle 还是 MongoDB、CouchDB，甚至使用文件系统，软件架构不应该因为不同的底层数据储存方式而产生巨大改变。
- 独立于外部依赖：无论外部依赖如何变更、升级，业务的核心逻辑不应该随之而大幅变化。
- 可测试：无论外部依赖了什么数据库、硬件、UI 或者服务，业务的逻辑应该都能够快速被验证正确性。

这就好像是建筑中的楼宇，一个好的楼宇，无论内部承载了什么人、有什么样的活动、还是外部有什么风雨，一栋楼都应该屹立不倒，而且可以确保它不会倒。但是今天我们在做业务研发时，更多的会去关注一些宏观的架构，比如 SOA 架构、微服务架构，而忽略了应用内部的架构设计，很容易导致代码逻辑混乱，很难维护，容易产生 bug 而且很难发现。今天，我希望能够通过案例的分析和重构，来推演出一套高质量的 DDD 架构。

案例分析

我们先看一个简单的案例需求如下：

用户可以通过银行网页转账给另一个账号，支持跨币种转账。
同时因为监管和对账需求，需要记录本次转账活动。
拿到这个需求之后，一个开发可能会经历一些技术选型，最终可能拆解需求如下：

1. 从 MySql 数据库中找到转出和转入的账户，选择用 MyBatis 的 mapper 实现 DAO；
2. 从 Yahoo（或其他渠道）提供的汇率服务获取转账的汇率信息（底层是 http 开放接口）；
3. 计算需要转出的金额，确保账户有足够余额，并且没超出每日转账上限；
4. 实现转入和转出操作，扣除手续费，保存数据库；
5. 发送 Kafka 审计消息，以便审计和对账用；

而一个简单的代码实现如下：

```
    public class TransferController {

    private TransferService transferService;

    public Result<Boolean> transfer(String targetAccountNumber, BigDecimal amount,
HttpSession session) {
        Long userId = (Long) session.getAttribute("userId");
        return transferService.transfer(userId, targetAccountNumber, amount, "CNY");
    }
}

public class TransferServiceImpl implements TransferService {

    private static final String TOPIC_AUDIT_LOG = "TOPIC_AUDIT_LOG";
    private AccountMapper accountDAO;
    private KafkaTemplate<String, String> kafkaTemplate;
    private YahooForexService yahooForex;

    @Override
```

```
    public Result<Boolean> transfer(Long sourceUserId, String targetAccountNumber,
BigDecimal targetAmount, String targetCurrency) {
        // 1. 从数据库读取数据，忽略所有校验逻辑如账号是否存在等
        AccountDO sourceAccountDO = accountDAO.selectByUserId(sourceUserId);
        AccountDO targetAccountDO =
accountDAO.selectByAccountNumber(targetAccountNumber);

        // 2. 业务参数校验
        if (!targetAccountDO.getCurrency().equals(targetCurrency)) {
            throw new InvalidCurrencyException();
        }

        // 3. 获取外部数据，并且包含一定的业务逻辑
        // exchange rate = 1 source currency = X target currency
        BigDecimal exchangeRate = BigDecimal.ONE;
        if (sourceAccountDO.getCurrency().equals(targetCurrency)) {
            exchangeRate = yahooForex.getExchangeRate(sourceAccountDO.getCurrency(),
targetCurrency);
        }
        BigDecimal sourceAmount = targetAmount.divide(exchangeRate,
RoundingMode.DOWN);

        // 4. 业务参数校验
        if (sourceAccountDO.getAvailable().compareTo(sourceAmount) < 0) {
            throw new InsufficientFundsException();
        }

        if (sourceAccountDO.getDailyLimit().compareTo(sourceAmount) < 0) {
            throw new DailyLimitExceededException();
        }

        // 5. 计算新值，并且更新字段
        BigDecimal newSource = sourceAccountDO.getAvailable().subtract(sourceAmount);
        BigDecimal newTarget = targetAccountDO.getAvailable().add(targetAmount);
        sourceAccountDO.setAvailable(newSource);
        targetAccountDO.setAvailable(newTarget);

        // 6. 更新到数据库
        accountDAO.update(sourceAccountDO);
        accountDAO.update(targetAccountDO);
```

```
            // 7. 发送审计消息
            String message = sourceUserId + "," + targetAccountNumber + "," + targetAmount +
"," + targetCurrency;
            kafkaTemplate.send(TOPIC_AUDIT_LOG, message);

            return Result.success(true);
        }

}
```

我们可以看到，一段业务代码里经常包含了参数校验、数据读取存储、业务计算、调用外部服务、发送消息等多种逻辑。在这个案例里虽然是写在了同一个方法里，在真实代码中经常会被拆分成多个子方法，但实际效果是一样的，而在我们日常的工作中，绝大部分代码都或多或少的接近于此类结构。在 Martin Fowler 的 P of EAA 书中，这种很常见的代码样式被叫做 Transaction Script（事务脚本）。虽然这种类似于脚本的写法在功能上没有什么问题，但是长久来看，他有以下几个很大的问题：可维护性差、可扩展性差、可测试性差。

问题 1-可维护性能差

一个应用最大的成本一般都不是来自于开发阶段，而是应用整个生命周期的总维护成本，所以代码的可维护性代表了最终成本。

可维护性 = 当依赖变化时，有多少代码需要随之改变

参考以上的案例代码，事务脚本类的代码很难维护因为以下几点：

- 数据结构的不稳定性：AccountDO 类是一个纯数据结构，映射了数据库中的一个表。这里的问题是数据库的表结构和设计是应用的外部依赖，长远来看都有可能会改变，比如数据库要做 Sharding，或者换一个表设计，或者改变字段名。
- 依赖库的升级：AccountMapper 依赖 MyBatis 的实现，如果 MyBatis 未来升级版本，可能会造成用法的不同（可以参考 iBatis 升级到基于注解的 MyBatis 的迁移成本）。同样的，如果未来换一个 ORM 体系，迁移成本也是巨大的。

- 第三方服务依赖的不确定性：第三方服务，比如 Yahoo 的汇率服务未来很有可能会有变化：轻则 API 签名变化，重则服务不可用需要寻找其他可替代的服务。在这些情况下改造和迁移成本都是巨大的。同时，外部依赖的兜底、限流、熔断等方案都需要随之改变。
- 第三方服务 API 的接口变化：YahooForexService.getExchangeRate 返回的结果是小数点还是百分比？入参是（source，target）还是（target，source）？谁能保证未来接口不会改变？如果改变了，核心的金额计算逻辑必须跟着改，否则会造成资损。
- 中间件更换：今天我们用 Kafka 发消息，明天如果要上阿里云用 RocketMQ 该怎么办？后天如果消息的序列化方式从 String 改为 Binary 该怎么办？如果需要消息分片该怎么改？

我们发现案例里的代码对于任何外部依赖的改变都会有比较大的影响。如果你的应用里有大量的此类代码，你每一天的时间基本上会被各种库升级、依赖服务升级、中间件升级、jar 包冲突占满，最终这个应用变成了一个不敢升级、不敢部署、不敢写新功能、并且随时会爆发的炸弹，终有一天会给你带来惊喜。

问题 2-可拓展性差

事务脚本式代码的第二大缺陷是：虽然写单个用例的代码非常高效简单，但是当用例多起来时，其扩展性会变得越来越差。

可扩展性 = 做新需求或改逻辑时，需要新增/修改多少代码

参考以上的代码，如果今天需要增加一个跨行转账的能力，你会发现基本上需要重新开发，基本上没有任何的可复用性：

- 数据来源被固定、数据格式不兼容：原有的 AccountDO 是从本地获取的，而跨行转账的数据可能需要从一个第三方服务获取，而服务之间数据格式不太可能是兼容的，导致从数据校验、数据读写、到异常处理、金额计算等逻辑都要重写。
- 业务逻辑无法复用：数据格式不兼容的问题会导致核心业务逻辑无法复用。每个用例都

是特殊逻辑的后果是最终会造成大量的 if-else 语句，而这种分支多的逻辑会让分析代码非常困难，容易错过边界情况，造成 bug。

- 逻辑和数据存储的相互依赖：当业务逻辑增加变得越来越复杂时，新加入的逻辑很有可能需要对数据库 schema 或消息格式做变更。而变更了数据格式后会导致原有的其他逻辑需要一起跟着动。在最极端的场景下，一个新功能的增加会导致所有原有功能的重构，成本巨大。

在事务脚本式的架构下，一般做第一个需求都非常的快，但是做第 N 个需求时需要的时间很有可能是呈指数级上升的，绝大部分时间花费在老功能的重构和兼容上，最终你的创新速度会跌为 0，促使老应用被推翻重构。

问题 3-可测试性能差

除了部分工具类、框架类和中间件类的代码有比较高的测试覆盖之外，我们在日常工作中很难看到业务代码有比较好的测试覆盖，而绝大部分的上线前的测试属于人肉的"集成测试"。低测试率导致我们对代码质量很难有把控，容易错过边界条件，异常 case 只有线上爆发了才被动发现。而低测试覆盖率的主要原因是业务代码的可测试性比较差。

可测试性 = 运行每个测试用例所花费的时间 * 每个需求所需要增加的测试用例数量

参考以上的一段代码，这种代码有极低的可测试性：

- 设施搭建困难：当代码中强依赖了数据库、第三方服务、中间件等外部依赖之后，想要完整跑通一个测试用例需要确保所有依赖都能跑起来，这个在项目早期是及其困难的。在项目后期也会由于各种系统的不稳定性而导致测试无法通过。
- 运行耗时长：大多数的外部依赖调用都是 I/O 密集型，如跨网络调用、磁盘调用等，而这种 I/O 调用在测试时需要耗时很久。另一个经常依赖的是笨重的框架如 Spring，启动 Spring 容器通常需要很久。当一个测试用例需要花超过 10 秒钟才能跑通时，绝大部分开发都不会很频繁的测试。
- 耦合度高：假如一段脚本中有 A、B、C 三个子步骤，而每个步骤有 N 个可能的状态，

当多个子步骤耦合度高时，为了完整覆盖所有用例，最多需要有 N * N * N 个测试用例。当耦合的子步骤越多时，需要的测试用例呈指数级增长。

在事务脚本模式下，当测试用例复杂度远大于真实代码复杂度，当运行测试用例的耗时超出人肉测试时，绝大部分人会选择不写完整的测试覆盖，而这种情况通常就是 bug 很难被早点发现的原因。

总结分析

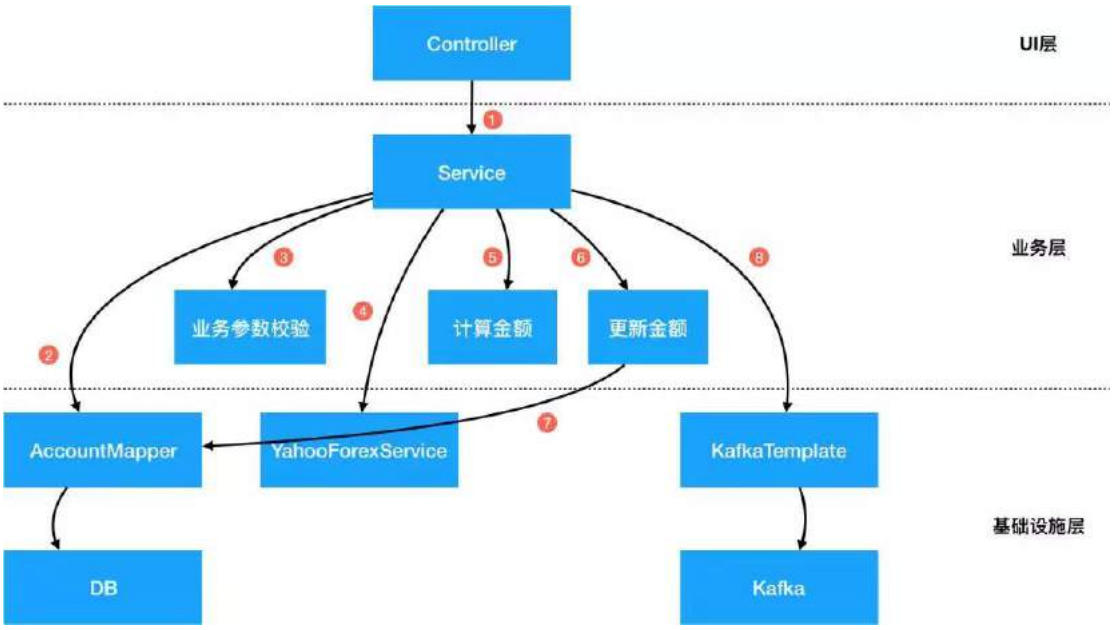
我们重新来分析一下为什么以上的问题会出现？因为以上的代码违背了至少以下几个软件设计的原则：

- 单一性原则（Single Responsibility Principle）：单一性原则要求一个对象/类应该只有一个变更的原因。但是在这个案例里，代码可能会因为任意一个外部依赖或计算逻辑的改变而改变。
- 依赖反转原则（Dependency Inversion Principle）：依赖反转原则要求在代码中依赖抽象，而不是具体的实现。在这个案例里外部依赖都是具体的实现，比如 YahooForexService 虽然是一个接口类，但是它对应的是依赖了 Yahoo 提供的具体服务，所以也算是依赖了实现。同样的 KafkaTemplate、MyBatis 的 DAO 实现都属于具体实现。
- 开放封闭原则（Open Closed Principle）：开放封闭原则指开放扩展，但是封闭修改。在这个案例里的金额计算属于可能会被修改的代码，这个时候该逻辑应该需要被包装成为不可修改的计算类，新功能通过计算类的拓展实现。

我们需要对代码重构才能解决这些问题。

重构方案

在重构之前，我们先画一张流程图，描述当前代码在做的每个步骤：



这是一个传统的三层分层结构：UI 层、业务层、和基础设施层。上层对于下层有直接的依赖关系，导致耦合度过高。在业务层中对于下层的基础设施有强依赖，耦合度高。我们需要对这张图上的每个节点做抽象和整理，来降低对外部依赖的耦合度。

抽象数据存储层

第一步常见的操作是将 Data Access 层做抽象，降低系统对数据库的直接依赖。具体的方法如下：

- 新建 Account 实体对象：一个实体（Entity）是拥有 ID 的域对象，除了拥有数据之外，同时拥有行为。Entity 和数据库储存格式无关，在设计中要以该领域的通用严谨语言（Ubiquitous Language）为依据。
- 新建对象储存接口类 AccountRepository：Repository 只负责 Entity 对象的存储和读取，而 Repository 的实现类完成数据库存储的细节。通过加入 Repository 接口，底层的数据库连接可以通过不同的实现类而替换。

具体的简单代码实现如下：

Account 实体类：

```
    @Data
public class Account {
    private AccountId id;
    private AccountNumber accountNumber;
    private UserId userId;
    private Money available;
    private Money dailyLimit;

    public void withdraw(Money money) {
        // 转出
    }

    public void deposit(Money money) {
        // 转入
    }
}
```

和 AccountRepository 及 MyBatis 实现类：

```
    public interface AccountRepository {
    Account find(AccountId id);
    Account find(AccountNumber accountNumber);
    Account find(UserId userId);
    Account save(Account account);
}
public class AccountRepositoryImpl implements AccountRepository {

    @Autowired
    private AccountMapper accountDAO;

    @Autowired
    private AccountBuilder accountBuilder;

    @Override
    public Account find(AccountId id) {
        AccountDO accountDO = accountDAO.selectById(id.getValue());
```

```
        return accountBuilder.toAccount(accountDO);
    }

    @Override
    public Account find(AccountNumber accountNumber) {
        AccountDO accountDO =
accountDAO.selectByAccountNumber(accountNumber.getValue());
        return accountBuilder.toAccount(accountDO);
    }

    @Override
    public Account find(UserId userId) {
        AccountDO accountDO = accountDAO.selectByUserId(userId.getId());
        return accountBuilder.toAccount(accountDO);
    }

    @Override
    public Account save(Account account) {
        AccountDO accountDO = accountBuilder.fromAccount(account);
        if (accountDO.getId() == null) {
            accountDAO.insert(accountDO);
        } else {
            accountDAO.update(accountDO);
        }
        return accountBuilder.toAccount(accountDO);
    }

}
```

Account 实体类和 AccountDO 数据类的对比如下：

- Data Object 数据类：AccountDO 是单纯的和数据库表的映射关系，每个字段对应数据库表的一个 column，这种对象叫 Data Object。DO 只有数据，没有行为。AccountDO 的作用是对数据库做快速映射，避免直接在代码里写 SQL。无论你用的是 MyBatis 还是 Hibernate 这种 ORM，从数据库来的都应该先直接映射到 DO 上，但是代码里应该完全避免直接操作 DO。

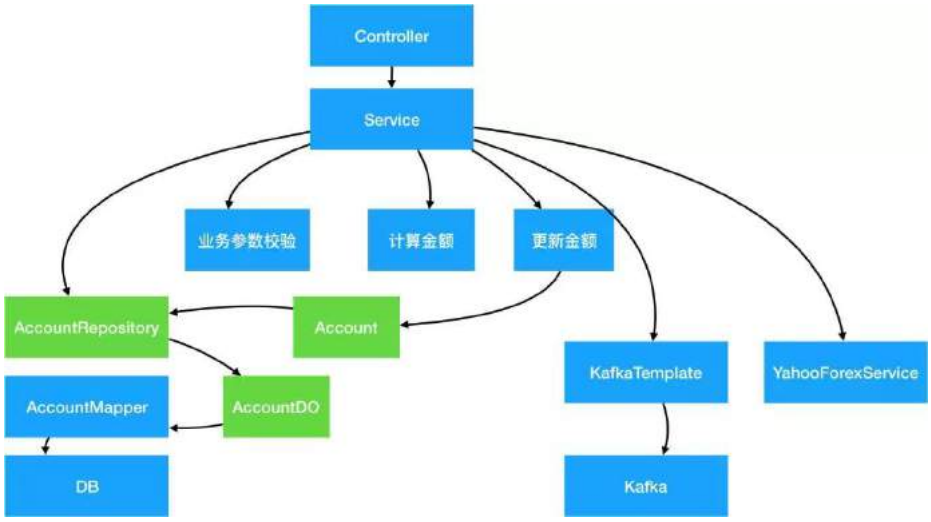
- Entity 实体类：Account 是基于领域逻辑的实体类，它的字段和数据库储存不需要有必然的联系。Entity 包含数据，同时也应该包含行为。在 Account 里，字段也不仅仅是 String 等基础类型，而应该尽可能用上一讲的 Domain Primitive 代替，可以避免大量的校验代码。

DAO 和 Repository 类的对比如下：

- DAO 对应的是一个特定的数据库类型的操作，相当于 SQL 的封装。所有操作的对象都是 DO 类，所有接口都可以根据数据库实现的不同而改变。比如 insert 和 update 属于数据库专属的操作。
- Repository 对应的是 Entity 对象读取储存的抽象，在接口层面做统一，不关注底层实现。比如，通过 save 保存一个 Entity 对象，但至于具体是 insert 还是 update 并不关心。Repository 的具体实现类通过调用 DAO 来实现各种操作，通过 Builder/Factory 对象实现 AccountDO 到 Account 之间的转化

Repository 和 Entity

- 通过 Account 对象，避免了其他业务逻辑代码和数据库的直接耦合，避免了当数据库字段变化时，大量业务逻辑也跟着变的问题。
- 通过 Repository，改变业务代码的思维方式，让业务逻辑不再面向数据库编程，而是面向领域模型编程。
- Account 属于一个完整的内存中对象，可以比较容易的做完整的测试覆盖，包含其行为。
- Repository 作为一个接口类，可以比较容易的实现 Mock 或 Stub，可以很容易测试。
- AccountRepositoryImpl 实现类，由于其职责被单一出来，只需要关注 Account 到 AccountDO 的映射关系和 Repository 方法到 DAO 方法之间的映射关系，相对于来说更容易测试。



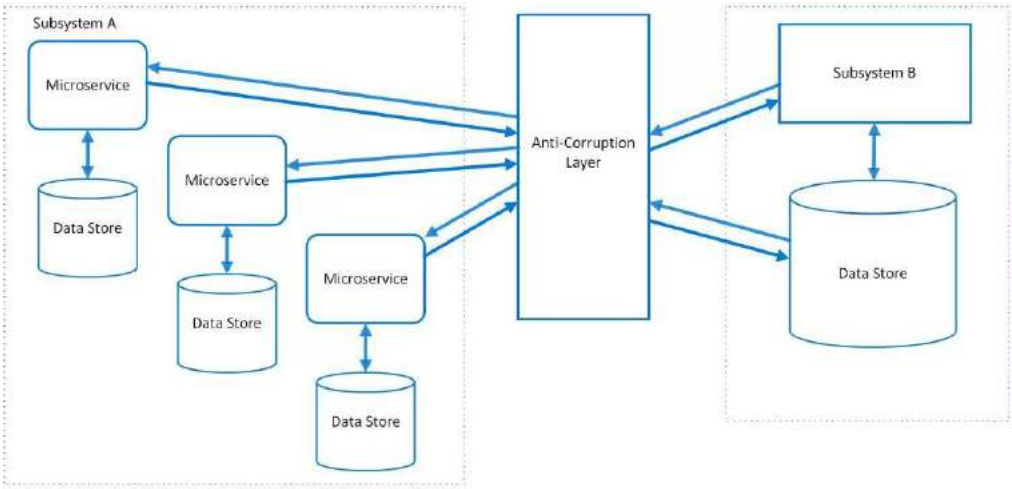
抽象第三方服务

类似对于数据库的抽象，所有第三方服务也需要通过抽象解决第三方服务不可控，入参出参强耦合的问题。在这个例子里我们抽象出 ExchangeRateService 的服务，和一个 ExchangeRate 的 Domain Primitive 类：

```
    public interface ExchangeRateService {
    ExchangeRate getExchangeRate(Currency source, Currency target);
}

public class ExchangeRateServiceImpl implements ExchangeRateService {

    @Autowired
    private YahooForexService yahooForexService;

    @Override
    public ExchangeRate getExchangeRate(Currency source, Currency target) {
        if (source.equals(target)) {
            return new ExchangeRate(BigDecimal.ONE, source, target);
        }
  BigDecimal forex = yahooForexService.getExchangeRate(source.getValue(),target.getValue())
        return new ExchangeRate(forex, source, target);
    }
```

防腐层（ACL）

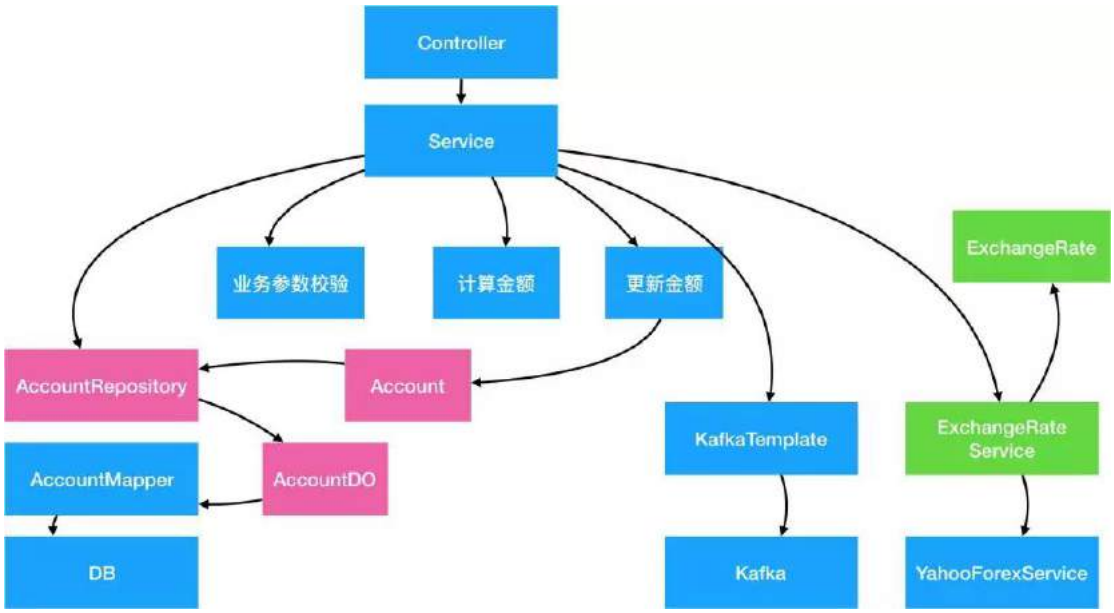
这种常见的设计模式叫做 Anti-Corruption Layer（防腐层或 ACL）。很多时候我们的系统会去依赖其他的系统，而被依赖的系统可能包含不合理的数据结构、API、协议或技术实现，如果对外部系统强依赖，会导致我们的系统被"腐蚀"。这个时候，通过在系统间加入一个防腐层，能够有效的隔离外部依赖和内部逻辑，无论外部如何变更，内部代码可以尽可能的保持不变。



ACL 不仅仅只是多了一层调用，在实际开发中 ACL 能够提供更多强大的功能：

- 适配器：很多时候外部依赖的数据、接口和协议并不符合内部规范，通过适配器模式，可以将数据转化逻辑封装到 ACL 内部，降低对业务代码的侵入。在这个案例里，我们通过封装了 ExchangeRate 和 Currency 对象，转化了对方的入参和出参，让入参出参更符合我们的标准。
- 缓存：对于频繁调用且数据变更不频繁的外部依赖，通过在 ACL 里嵌入缓存逻辑，能够有效的降低对于外部依赖的请求压力。同时，很多时候缓存逻辑是写在业务代码里的，通过将缓存逻辑嵌入 ACL，能够降低业务代码的复杂度。
- 兜底：如果外部依赖的稳定性较差，一个能够有效提升我们系统稳定性的策略是通过 ACL 起到兜底的作用，比如当外部依赖出问题后，返回最近一次成功的缓存或业务兜底数据。这种兜底逻辑一般都比较复杂，如果散落在核心业务代码中会很难维护，通过集中在 ACL 中，更加容易被测试和修改。

- 易于测试：类似于之前的 Repository，ACL 的接口类能够很容易的实现 Mock 或 Stub，以便于单元测试。
- 功能开关：有些时候我们希望能在某些场景下开放或关闭某个接口的功能，或者让某个接口返回一个特定的值，我们可以在 ACL 配置功能开关来实现，而不会对真实业务代码造成影响。同时，使用功能开关也能让我们容易的实现 Monkey 测试，而不需要真正物理性的关闭外部依赖。



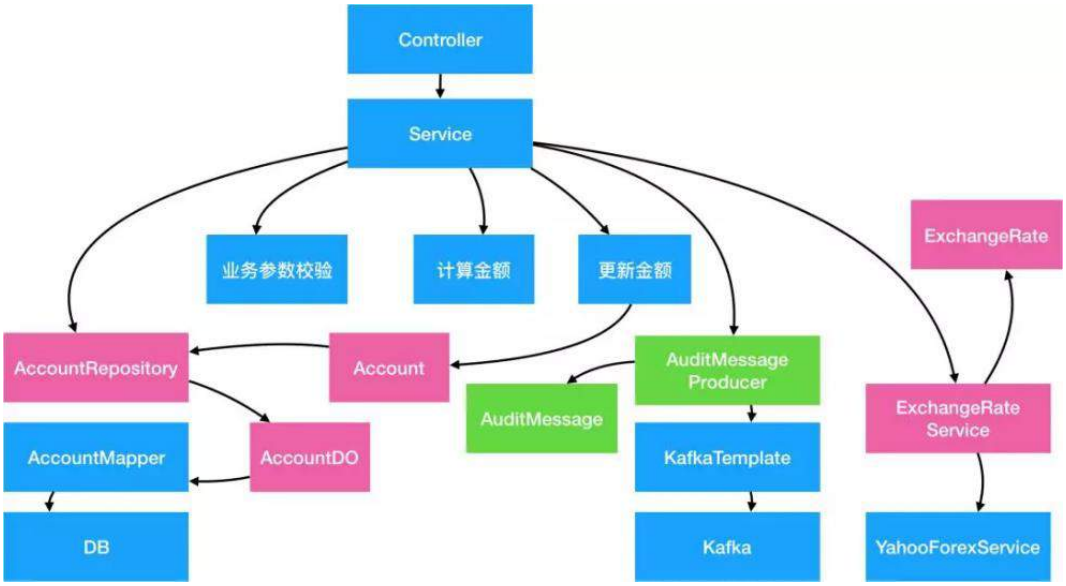
抽象中间件

类似于 2.2 的第三方服务的抽象，对各种中间件的抽象的目的是让业务代码不再依赖中间件的实现逻辑。因为中间件通常需要有通用型，中间件的接口通常是 String 或 Byte[] 类型的，导致序列化/反序列化逻辑通常和业务逻辑混杂在一起，造成胶水代码。通过中间件的 ACL 抽象，减少重复胶水代码。

在这个案例里，我们通过封装一个抽象的 AuditMessageProducer 和 AuditMessage DP 对象，实现对底层 kafka 实现的隔离：

```
    @Value
@AllArgsConstructor
public class AuditMessage {

    private UserId userId;
    private AccountNumber source;
    private AccountNumber target;
    private Money money;
    private Date date;

    public String serialize() {
        return userId + "," + source + "," + target + "," + money + "," + date;

    }

    public static AuditMessage deserialize(String value) {
        // todo
        return null;
    }
}

public interface AuditMessageProducer {
    SendResult send(AuditMessage message);
}

public class AuditMessageProducerImpl implements AuditMessageProducer {

    private static final String TOPIC_AUDIT_LOG = "TOPIC_AUDIT_LOG";

    @Autowired
    private KafkaTemplate<String, String> kafkaTemplate;

    @Override
    public SendResult send(AuditMessage message) {
        String messageBody = message.serialize();
        kafkaTemplate.send(TOPIC_AUDIT_LOG, messageBody);
        return SendResult.success();
    }
}
```

具体的分析和 2.2 类似，在此略过。



封装业务逻辑

在这个案例里，有很多业务逻辑是跟外部依赖的代码混合的，包括金额计算、账户余额的校验、转账限制、金额增减等。这种逻辑混淆导致了核心计算逻辑无法被有效的测试和复用。在这里，我们的解法是通过 Entity、Domain Primitive 和 Domain Service 封装所有的业务逻辑：

用 Domain Primitive 封装跟实体无关的无状态计算逻辑

在这个案例里使用 ExchangeRate 来封装汇率计算逻辑：

```
    BigDecimal exchangeRate = BigDecimal.ONE;
if (sourceAccountDO.getCurrency().equals(targetCurrency)) {
    exchangeRate = yahooForex.getExchangeRate(sourceAccountDO.getCurrency(),
targetCurrency);
}
BigDecimal sourceAmount = targetAmount.divide(exchangeRate, RoundingMode.DOWN);
```

变为：

```
    ExchangeRate exchangeRate =
exchangeRateService.getExchangeRate(sourceAccount.getCurrency(),
targetMoney.getCurrency());
Money sourceMoney = exchangeRate.exchangeTo(targetMoney);
```

用 Entity 封装单对象的有状态的行为，包括业务校验

用 Account 实体类封装所有 Account 的行为，包括业务校验如下：

```
    @Data
public class Account {

    private AccountId id;
    private AccountNumber accountNumber;
    private UserId userId;
    private Money available;

    private Money dailyLimit;

    public Currency getCurrency() {
        return this.available.getCurrency();
    }

    // 转入
    public void deposit(Money money) {
        if (!this.getCurrency().equals(money.getCurrency())) {
            throw new InvalidCurrencyException();
        }
        this.available = this.available.add(money);
    }

    // 转出
    public void withdraw(Money money) {
        if (this.available.compareTo(money) < 0) {
            throw new InsufficientFundsException();
        }
        if (this.dailyLimit.compareTo(money) < 0) {
            throw new DailyLimitExceededException();
        }
        this.available = this.available.subtract(money);
```

```
    }
}
```

原有的业务代码则可以简化为：

```
    sourceAccount.deposit(sourceMoney);
    targetAccount.withdraw(targetMoney);
```

用 Domain Service 封装多对象逻辑

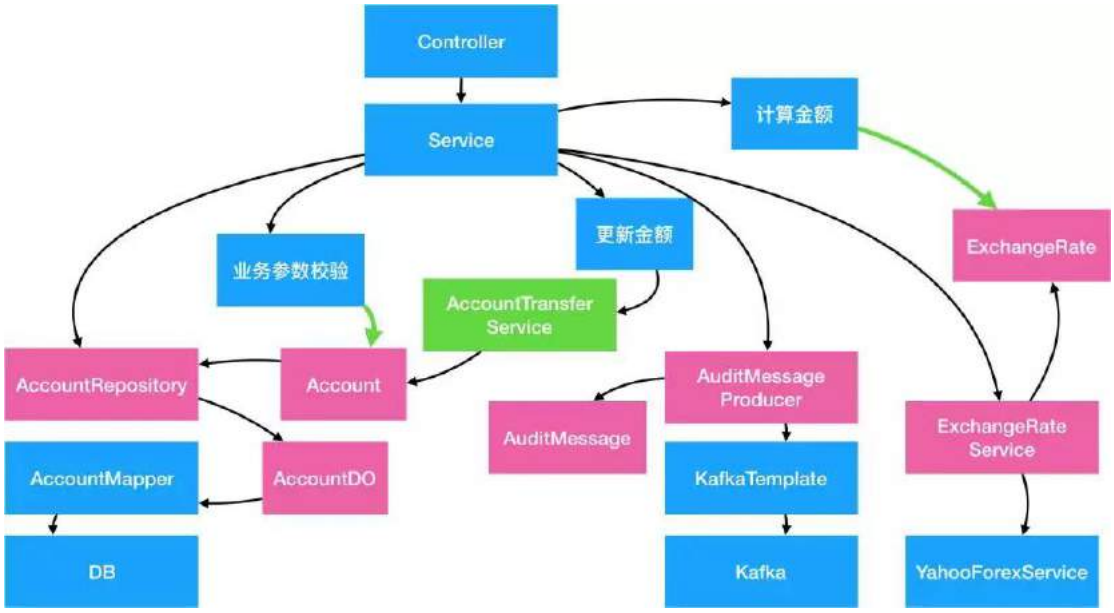
在这个案例里，我们发现这两个账号的转出和转入实际上是一体的，也就是说这种行为应该被封装到一个对象中去。特别是考虑到未来这个逻辑可能会产生变化：比如增加一个扣手续费的逻辑。这个时候在原有的 TransferService 中做并不合适，在任何一个 Entity 或者 Domain Primitive 里也不合适，需要有一个新的类去包含跨域对象的行为。这种对象叫做 Domain Service。

我们创建一个 AccountTransferService 的类：

```
    public interface AccountTransferService {
    void transfer(Account sourceAccount, Account targetAccount, Money targetMoney,
ExchangeRate exchangeRate);
}

public class AccountTransferServiceImpl implements AccountTransferService {
    private ExchangeRateService exchangeRateService;

    @Override
    public void transfer(Account sourceAccount, Account targetAccount, Money targetMoney,
ExchangeRate exchangeRate) {
        Money sourceMoney = exchangeRate.exchangeTo(targetMoney);
        sourceAccount.deposit(sourceMoney);
        targetAccount.withdraw(targetMoney);
    }
}
```

而原始代码则简化为一行：

```
    accountTransferService.transfer(sourceAccount, targetAccount, targetMoney,
exchangeRate);
```



重构后结果分析

这个案例重构后的代码如下：

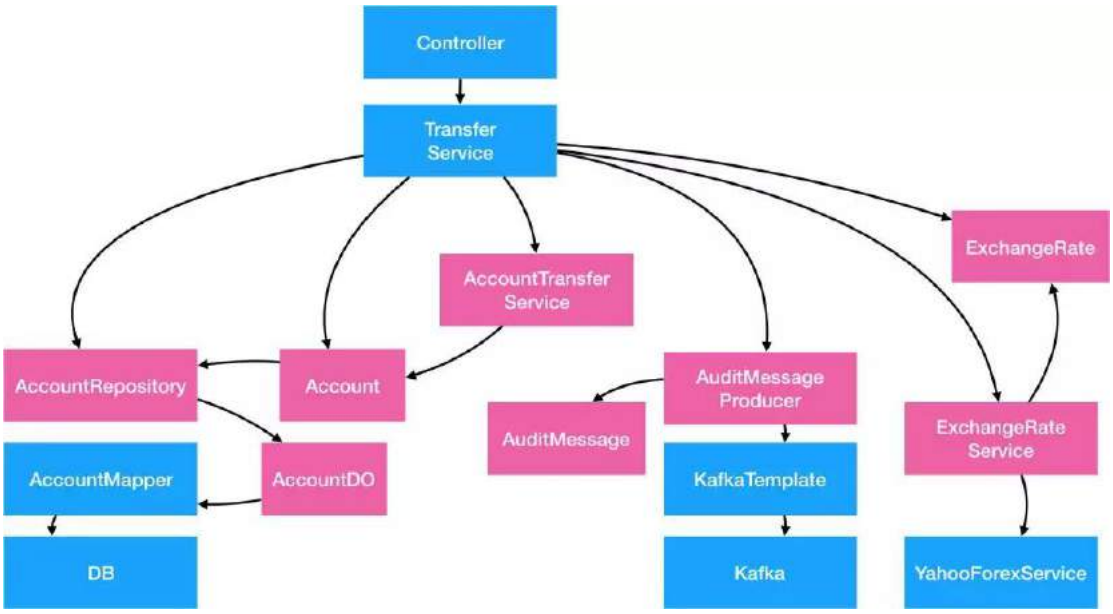
```
    public class TransferServiceImplNew implements TransferService {

    private AccountRepository accountRepository;
    private AuditMessageProducer auditMessageProducer;
    private ExchangeRateService exchangeRateService;
    private AccountTransferService accountTransferService;

    @Override
    public Result<Boolean> transfer(Long sourceUserId, String targetAccountNumber,
BigDecimal targetAmount, String targetCurrency) {

        // 参数校验
        Money targetMoney = new Money(targetAmount, new Currency(targetCurrency));
```

```
        // 读数据
        Account sourceAccount = accountRepository.find(new UserId(sourceUserId));
        Account targetAccount = accountRepository.find(new
AccountNumber(targetAccountNumber));
        ExchangeRate exchangeRate =
exchangeRateService.getExchangeRate(sourceAccount.getCurrency(),
targetMoney.getCurrency());

        // 业务逻辑
        accountTransferService.transfer(sourceAccount, targetAccount, targetMoney,
exchangeRate);

        // 保存数据
        accountRepository.save(sourceAccount);
        accountRepository.save(targetAccount);

        // 发送审计消息
        AuditMessage message = new AuditMessage(sourceAccount, targetAccount,
targetMoney);
        auditMessageProducer.send(message);

        return Result.success(true);

    }
}
```

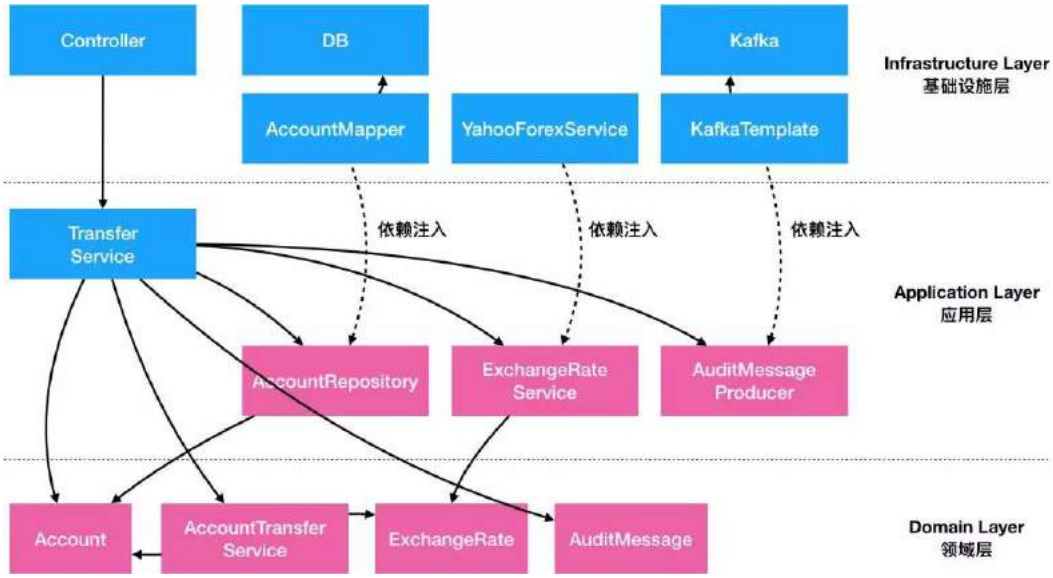
可以看出来，经过重构后的代码有以下几个特征：

- 业务逻辑清晰，数据存储和业务逻辑完全分隔。
- Entity、Domain Primitive、Domain Service 都是独立的对象，没有任何外部依赖，但是却包含了所有核心业务逻辑，可以单独完整测试。
- 原有的 TransferService 不再包括任何计算逻辑，仅仅作为组件编排，所有逻辑均 delegate 到其他组件。这种仅包含 Orchestration（编排）的服务叫做 Application Service（应用服务）。

我们可以根据新的结构重新画一张图：



然后通过重新编排后该图变为：



我们可以发现，通过对外部依赖的抽象和内部逻辑的封装重构，应用整体的依赖关系变了：

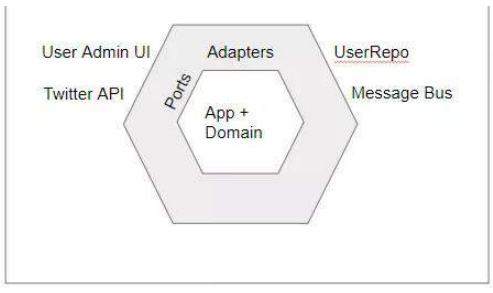
- 最底层不再是数据库，而是 Entity、Domain Primitive 和 Domain Service。这些对象不依赖任何外部服务和框架，而是纯内存中的数据和操作。这些对象我们打包为 Domain Layer（领域层）。领域层没有任何外部依赖关系。

- 再其次的是负责组件编排的 Application Service，但是这些服务仅仅依赖了一些抽象出来的 ACL 类和 Repository 类，而其具体实现类是通过依赖注入注进来的。Application Service、Repository、ACL 等我们统称为 Application Layer（应用层）。应用层 依赖 领域层，但不依赖具体实现。
- 最后是 ACL，Repository 等的具体实现，这些实现通常依赖外部具体的技术实现和框架，所以统称为 Infrastructure Layer（基础设施层）。Web 框架里的对象如 Controller 之类的通常也属于基础设施层。

如果今天能够重新写这段代码，考虑到最终的依赖关系，我们可能先写 Domain 层的业务逻辑，然后再写 Application 层的组件编排，最后才写每个外部依赖的具体实现。这种架构思路和代码组织结构就叫做 Domain-Driven Design（领域驱动设计，或 DDD）。所以 DDD 不是一个特殊的架构设计，而是所有 Transction Script 代码经过合理重构后一定会抵达的终点。

DDD 的六边形架构

在我们传统的代码里，我们一般都很注重每个外部依赖的实现细节和规范，但是今天我们需要敢于抛弃掉原有的理念，重新审视代码结构。在上面重构的代码里，如果抛弃掉所有 Repository、ACL、Producer 等的具体实现细节，我们会发现每一个对外部的抽象类其实就是输入或输出，类似于计算机系统中的 I/O 节点。这个观点在 CQRS 架构中也同样适用，将所有接口分为 Command（输入）和 Query（输出）两种。除了 I/O 之外其他的内部逻辑，就是应用业务的核心逻辑。基于这个基础，Alistair Cockburn 在 2005 年提出了 Hexagonal Architecture（六边形架构），又被称之为 Ports and Adapters（端口和适配器架构）。

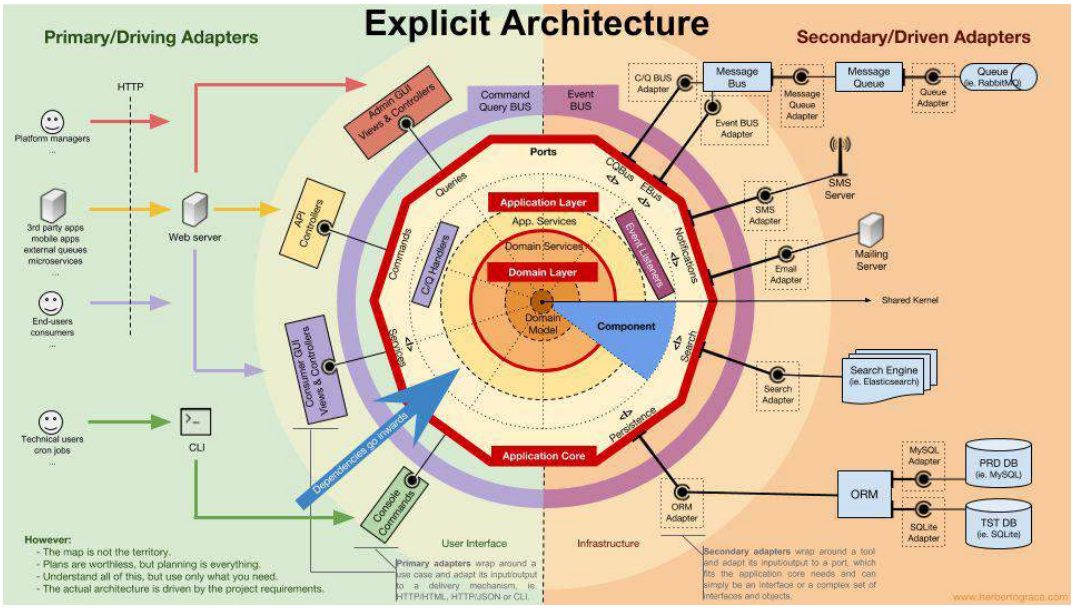


在这张图中：

- I/O 的具体实现在模型的最外层
- 每个 I/O 的适配器在灰色地带
- 每个 Hex 的边是一个端口
- Hex 的中央是应用的核心领域模型

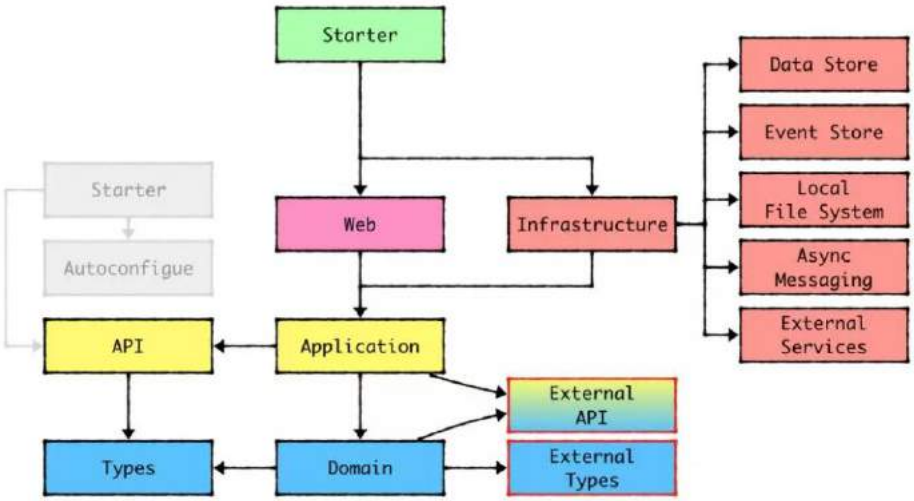
在 Hex 中，架构的组织关系第一次变成了一个二维的内外关系，而不是传统一维的上下关系。同时在 Hex 架构中我们第一次发现 UI 层、DB 层、和各种中间件层实际上是没有本质上区别的，都只是数据的输入和输出，而不是在传统架构中的最上层和最下层。

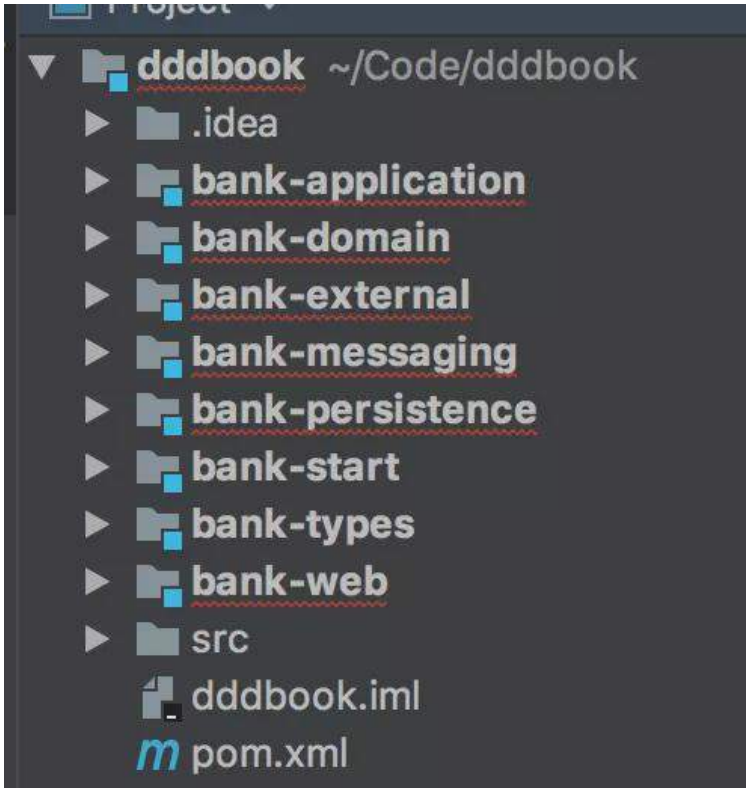
除了 2005 年的 Hex 架构，2008 年 Jeffery Palermo 的 Onion Architecture（洋葱架构）和 2017 年 Robert Martin 的 Clean Architecture（干净架构），都是极为类似的思想。除了命名不一样、切入点不一样之外，其他的整体架构都是基于一个二维的内外关系。这也说明了基于 DDD 的架构最终的形态都是类似的。Herberto Graca 有一个很全面的图包含了绝大部分现实中的端口类，值得借鉴。



代码组织结构

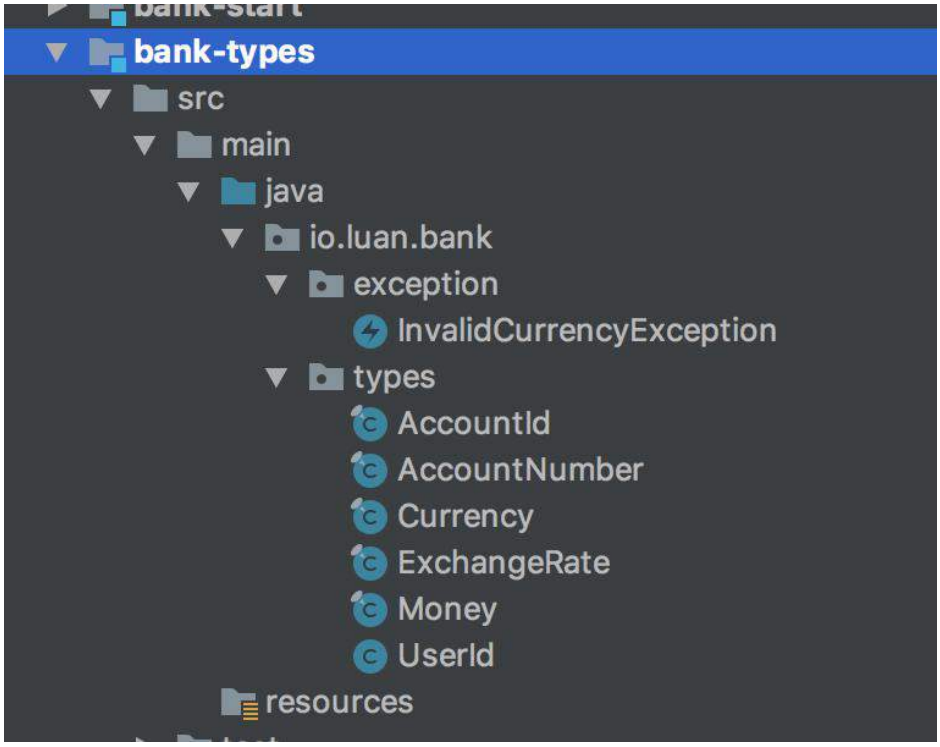
为了有效的组织代码结构，避免下层代码依赖到上层实现的情况，在 Java 中我们可以通过 POM Module 和 POM 依赖来处理相互的关系。通过 Spring/SpringBoot 的容器来解决运行时动态注入具体实现的依赖的问题。一个简单的依赖关系图如下：





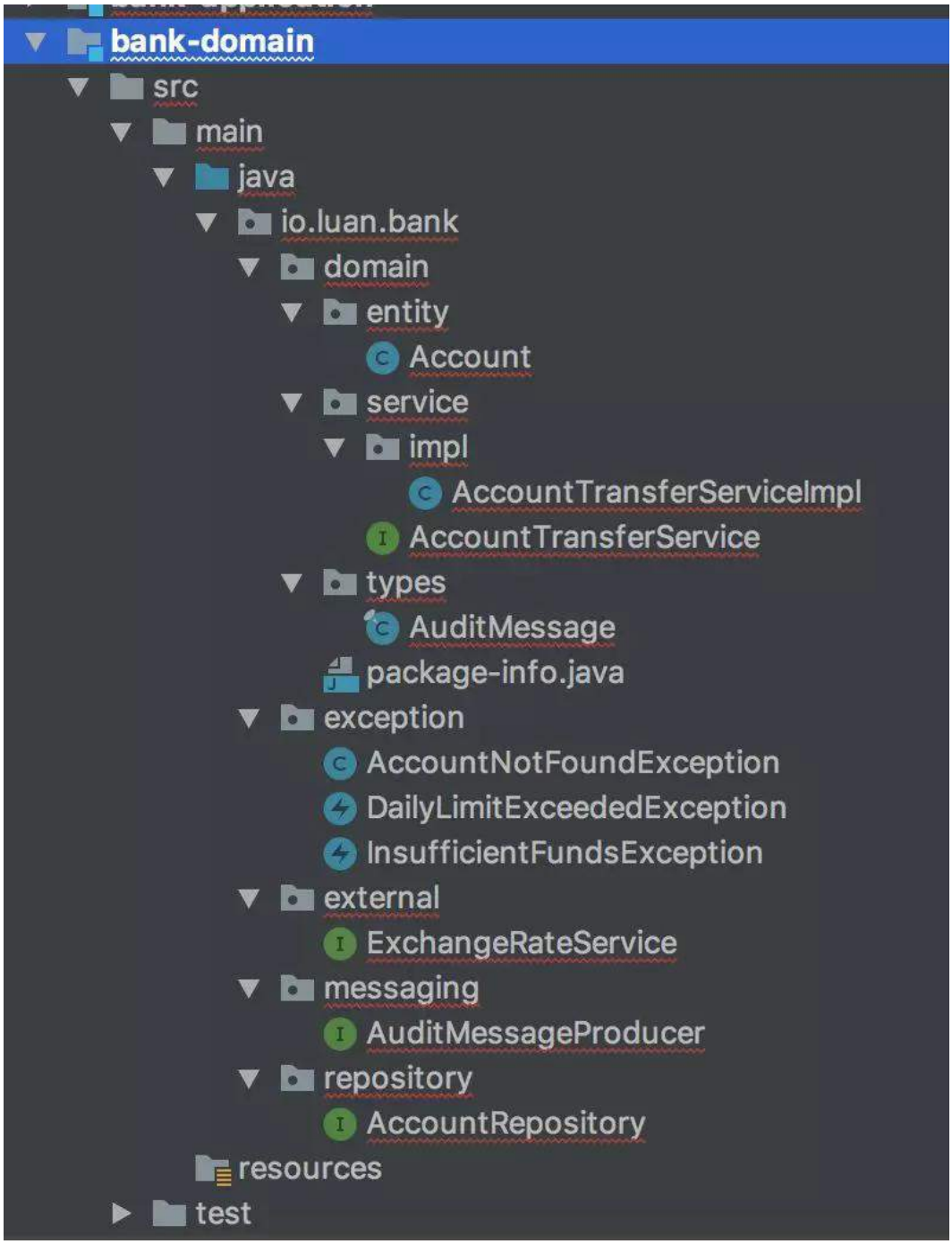
Types 模块

Types 模块是保存可以对外暴露的 Domain Primitives 的地方。Domain Primitives 因为是无状态的逻辑，可以对外暴露，所以经常被包含在对外的 API 接口中，需要单独成为模块。Types 模块不依赖任何类库，纯 POJO 。



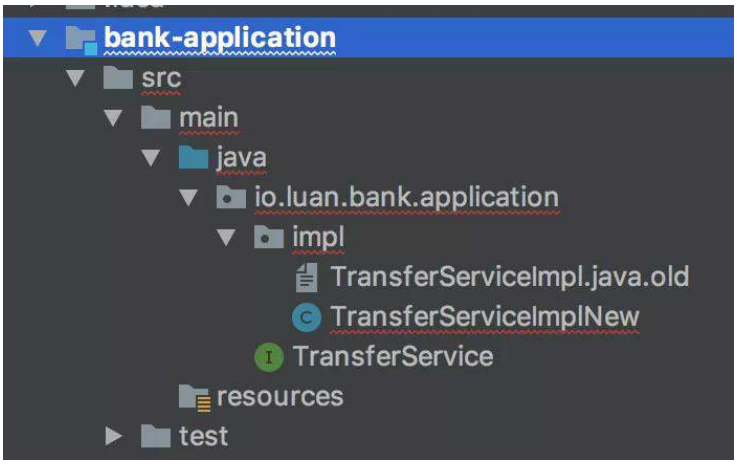
Domain 模块

Domain 模块是核心业务逻辑的集中地，包含有状态的 Entity、领域服务 Domain Service、以及各种外部依赖的接口类（如 Repository、ACL、中间件等。Domain 模块仅依赖 Types 模块，也是纯 POJO 。



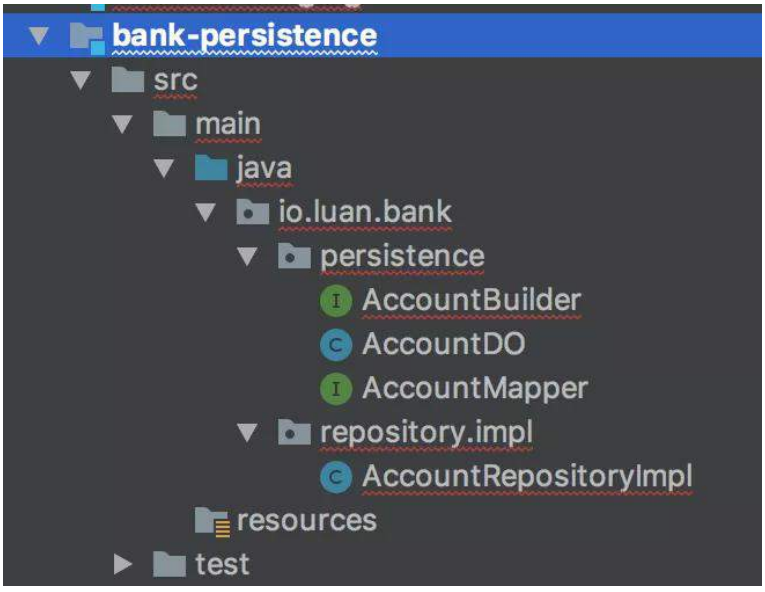
Application 模块

Application 模块主要包含 Application Service 和一些相关的类。Application 模块依赖 Domain 模块。还是不依赖任何框架，纯 POJO。



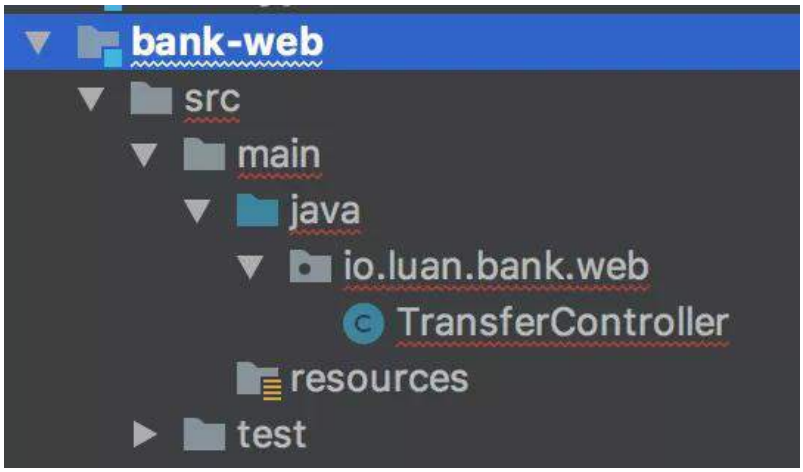
Infrastructure 模块

Infrastructure 模块包含了 Persistence、Messaging、External 等模块。比如：Persistence 模块包含数据库 DAO 的实现，包含 Data Object、ORM Mapper、Entity 到 DO 的转化类等。Persistence 模块要依赖具体的 ORM 类库，比如 MyBatis。如果要用 Spring-Mybatis 提供的注解方案，则需要依赖 Spring。



Web 模块

Web 模块包含 Controller 等相关代码。如果用 SpringMVC 则需要依赖 Spring。



Start 模块

Start 模块是 SpringBoot 的启动类。

测试

- Types，Domain 模块都属于无外部依赖的纯 POJO，基本上都可以 100%的被单元测试覆盖。
- Application 模块的代码依赖外部抽象类，需要通过测试框架去 Mock 所有外部依赖，但仍然可以 100%被单元测试。
- Infrastructure 的每个模块的代码相对独立，接口数量比较少，相对比较容易写单测。但是由于依赖了外部 I/O，速度上不可能很快，但好在模块的变动不会很频繁，属于一劳永逸。
- Web 模块有两种测试方法：通过 Spring 的 MockMVC 测试，或者通过 HttpClient 调用接口测试。但是在测试时最好把 Controller 依赖的服务类都 Mock 掉。一般来说当你把 Controller 的逻辑都后置到 Application Service 中时，Controller 的逻辑变得极为简单，很容易 100%覆盖。
- Start 模块：通常应用的集成测试写在 start 里。当其他模块的单元测试都能 100%覆盖后，集成测试用来验证整体链路的真实性。

代码的演进/变化速度

在传统架构中，代码从上到下的变化速度基本上是一致的，改个需求需要从接口、到业务逻辑、到数据库全量变更，而第三方变更可能会导致整个代码的重写。但是在 DDD 中不同模块的代码的演进速度是不一样的：

- Domain 层属于核心业务逻辑，属于经常被修改的地方。比如：原来不需要扣手续费，现在需要了之类的。通过 Entity 能够解决基于单个对象的逻辑变更，通过 Domain Service 解决多个对象间的业务逻辑变更。
- Application 层属于 Use Case（业务用例）。业务用例一般都是描述比较大方向的需求，接口相对稳定，特别是对外的接口一般不会频繁变更。添加业务用例可以通过新增 Application Service 或者新增接口实现功能的扩展。
- Infrastructure 层属于最低频变更的。一般这个层的模块只有在外部依赖变更了之后才会跟着升级，而外部依赖的变更频率一般远低于业务逻辑的变更频率。

所以在 DDD 架构中，能明显看出越外层的代码越稳定，越内层的代码演进越快，真正体现了领域"驱动"的核心思想。

总结

DDD 不是一个什么特殊的架构，而是任何传统代码经过合理的重构之后最终一定会抵达的终点。DDD 的架构能够有效的解决传统架构中的问题：

- 高可维护性：当外部依赖变更时，内部代码只用变更跟外部对接的模块，其他业务逻辑不变。
- 高可扩展性：做新功能时，绝大部分的代码都能复用，仅需要增加核心业务逻辑即可。
- 高可测试性：每个拆分出来的模块都符合单一性原则，绝大部分不依赖框架，可以快速的单元测试，做到 100%覆盖。
- 代码结构清晰：通过 POM module 可以解决模块间的依赖关系， 所有外接模块都可以单独独立成 Jar 包被复用。当团队形成规范后，可以快速的定位到相关代码。

阿里技术专家详解 DDD 系列 第三讲 - Repository 模式

作者 | 殷浩
出品 | 淘系技术

写在前面：

这篇文章和《阿里技术专家详解 DDD 系列 第二弹 - 应用架构》隔了比较久，一方面是工作比较忙，另一方面是在讲 Repository 之前其实应该先讲 Entity（实体）、Aggregate Root（聚合根）、Bounded Context（限界上下文）等概念。但在实际写的过程中，发现单纯讲 Entity 相关的东西会比较抽象，很难落地。所以本文被推倒重来，从 Repository 开始入手，先把可以落地的、能形成规范的东西先确定下来，最后再尝试落地 Entity。这个当然也是我们可以在日常按照 DDD 重构时尝试的路径。

提前预告，接下来的一篇文章将覆盖 Anti-Corruption Layer（防腐层）的逻辑，但是你会发现跟 Repository 模式的理念非常接近。等所有周边的东西都覆盖之后，再详细讲 Entity 也许会变得不那么抽象。

DDD 的宏观理念其实并不难懂，但是如同 REST 一样，DDD 也只是一个设计思想，缺少一套完整的规范，导致 DDD 新手落地困难。

我之前的架构篇主要从顶层设计往下看，从这一篇开始我希望能填补上一些 DDD 的代码落地规范，帮助同学在日常工作中落地 DDD 思想，并且希望能通过一整套规范，让不同的业务之间的同学能够更快的看懂、掌握对方的代码。但是规则是死的、人是活的，各位同学需要根据自己业务的实际情况去有选择的去落地规范，DDD 的规范不可能覆盖所有场景，但我希望能通过解释，让同学们了解 DDD 背后的一些思考和取舍。

为什么要用 Repository

实体模型 vs. 贫血模型

Entity（实体）这个词在计算机领域的最初应用可能是来自于 Peter Chen 在 1976 年的 “The Entity-Relationship Model - Toward a Unified View of Data"（ER 模型），用来描述实体之间的关系，而 ER 模型后来逐渐的演变成为一个数据模型，在关系型数据库中代表了数据的储存方式。

而 2006 年的 JPA 标准，通过@Entity 等注解，以及 Hibernate 等 ORM 框架的实现，让很多 Java 开发对 Entity 的理解停留在了数据映射层面，忽略了 Entity 实体的本身行为，造成今天很多的模型仅包含了实体的数据和属性，而所有的业务逻辑都被分散在多个服务、Controller、Utils 工具类中，这个就是 Martin Fowler 所说的的 Anemic Domain Model（贫血领域模型）。

如何知道你的模型是贫血的呢？可以看一下你代码中是否有以下的几个特征：

1. 有大量的 XxxDO 对象：这里 DO 虽然有时候代表了 Domain Object，但实际上仅仅是数据库表结构的映射，里面没有包含（或包含了很少的）业务逻辑；
2. 服务和 Controller 里有大量的业务逻辑：比如校验逻辑、计算逻辑、格式转化逻辑、对象关系逻辑、数据存储逻辑等；
3. 大量的 Utils 工具类等。

而贫血模型的缺陷是非常明显的：

1. 无法保护模型对象的完整性和一致性：因为对象的所有属性都是公开的，只能由调用方来维护模型的一致性，而这个是没有保障的；之前曾经出现的案例就是调用方没有能维护模型数据的一致性，导致脏数据使用时出现 bug，这一类的 bug 还特别隐蔽，很难排查到。

2. 对象操作的可发现性极差：单纯从对象的属性上很难看出来都有哪些业务逻辑，什么时候可以被调用，以及可以赋值的边界是什么；比如说，Long 类型的值是否可以是 0 或者负数？
3. 代码逻辑重复：比如校验逻辑、计算逻辑，都很容易出现在多个服务、多个代码块里，提升维护成本和 bug 出现的概率；一类常见的 bug 就是当贫血模型变更后，校验逻辑由于出现在多个地方，没有能跟着变，导致校验失败或失效。
4. 代码的健壮性差：比如一个数据模型的变化可能导致从上到下的所有代码的变更。
5. 强依赖底层实现：业务代码里强依赖了底层数据库、网络/中间件协议、第三方服务等，造成核心逻辑代码的僵化且维护成本高。

虽然贫血模型有很大的缺陷，但是在我们日常的代码中，我见过的 99%的代码都是基于贫血模型，为什么呢？我总结了以下几点：

1. 数据库思维：从有了数据库的那一天起，开发人员的思考方式就逐渐从"写业务逻辑"转变为了"写数据库逻辑"，也就是我们经常说的在写 CRUD 代码。
2. 贫血模型"简单"：贫血模型的优势在于"简单"，仅仅是对数据库表的字段映射，所以可以从前到后用统一格式串通。这里简单打了引号，是因为它只是表面上的简单，实际上当未来有模型变更时，你会发现其实并不简单，每次变更都是非常复杂的事情。
3. 脚本思维：很多常见的代码都属于"脚本"或"胶水代码"，也就是流程式代码。脚本代码的好处就是比较容易理解，但长久来看缺乏健壮性，维护成本会越来越高。

但是可能最核心的原因在于，实际上我们在日常开发中，混淆了两个概念：

- 数据模型（Data Model）：指业务数据该如何持久化，以及数据之间的关系，也就是传统的 ER 模型；
- 业务模型/领域模型（Domain Model）：指业务逻辑中，相关联的数据该如何联动。

所以，解决这个问题的根本方案，就是要在代码里严格区分 Data Model 和 Domain Model，具体的规范会在后文详细描述。在真实代码结构中，Data Model 和 Domain Model 实际上会分别在不同的层里，Data Model 只存在于数据层，而 Domain Model 在

领域层，而链接了这两层的关键对象，就是 Repository。

Repository 的价值

在传统的数据库驱动开发中，我们会对数据库操作做一个封装，一般叫做 Data Access Object（DAO）。DAO 的核心价值是封装了拼接 SQL、维护数据库连接、事务等琐碎的底层逻辑，让业务开发可以专注于写代码。但是在本质上，DAO 的操作还是数据库操作，DAO 的某个方法还是在直接操作数据库和数据模型，只是少写了部分代码。在 Uncle Bob 的《代码整洁之道》一书里，作者用了一个非常形象的描述：

- 硬件（Hardware）：指创造了之后不可（或者很难）变更的东西。数据库对于开发来说，就属于"硬件"，数据库选型后基本上后面不会再变，比如：用了 MySQL 就很难再改为 MongoDB，改造成本过高。
- 软件（Software）：指创造了之后可以随时修改的东西。对于开发来说，业务代码应该追求做"软件"，因为业务流程、规则在不停的变化，我们的代码也应该能随时变化。
- 固件（Firmware）：即那些强烈依赖了硬件的软件。我们常见的是路由器里的固件或安卓的固件等等。固件的特点是对硬件做了抽象，但仅能适配某款硬件，不能通用。所以今天不存在所谓的通用安卓固件，而是每个手机都需要有自己的固件。

从上面的描述我们能看出来，数据库在本质上属于"硬件"，DAO 在本质上属于"固件"，而我们自己的代码希望是属于"软件"。但是，固件有个非常不好的特性，那就是会传播，也就是说当一个软件强依赖了固件时，由于固件的限制，会导致软件也变得难以变更，最终让软件变得跟固件一样难以变更。

举个软件很容易被"固化"的例子：

```
private OrderDAO orderDAO;

public Long addOrder(RequestDTO request) {
  // 此处省略很多拼装逻辑
  OrderDO orderDO = new OrderDO();
  orderDAO.insertOrder(orderDO);
```

```
return orderDO.getId();
}

public void updateOrder(OrderDO orderDO, RequestDTO updateRequest) {
orderDO.setXXX(XXX); // 省略很多
orderDAO.updateOrder(orderDO);}

public void doSomeBusiness(Long id) {
OrderDO orderDO = orderDAO.getOrderById(id);
// 此处省略很多业务逻辑
}
```

在上面的这段简单代码里，该对象依赖了 DAO，也就是依赖了 DB。虽然乍一看感觉并没什么毛病，但是假设未来要加一个缓存逻辑，代码则需要改为如下：

```
private OrderDAO orderDAO;
private Cache cache;

public Long addOrder(RequestDTO request) {
// 此处省略很多拼装逻辑
OrderDO orderDO = new OrderDO();
orderDAO.insertOrder(orderDO);
cache.put(orderDO.getId(), orderDO);
return orderDO.getId();}

public void updateOrder(OrderDO orderDO, RequestDTO updateRequest) {
orderDO.setXXX(XXX); // 省略很多
orderDAO.updateOrder(orderDO);
cache.put(orderDO.getId(), orderDO);}

public void doSomeBusiness(Long id) {
OrderDO orderDO = cache.get(id);
if (orderDO == null) {
orderDO = orderDAO.getOrderById(id);
}
// 此处省略很多业务逻辑
}
```

这时，你会发现因为插入的逻辑变化了，导致在所有的使用数据的地方，都需要从 1 行代码改为至少 3 行。而当你的代码量变得比较大，然后如果在某个地方你忘记了查缓存，或者在某个地方忘记了更新缓存，轻则需要查数据库，重则是缓存和数据库不一致，导致 bug。当你的代码量变得越来越多，直接调用 DAO、缓存的地方越来越多时，每次底层变更都会变得越来越难，越来越容易导致 bug。这就是软件被"固化"的后果。

所以，我们需要一个模式，能够隔离我们的软件（业务逻辑）和固件/硬件（DAO、DB），让我们的软件变得更加健壮，而这个就是 Repository 的核心价值。

模型对象代码规范

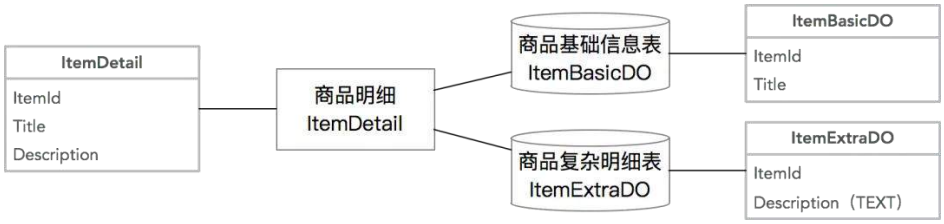
对象类型

在讲 Repository 规范之前，我们需要先讲清楚 3 种模型的区别，Entity、Data Object（DO)和 Data Transfer Object（DTO)：

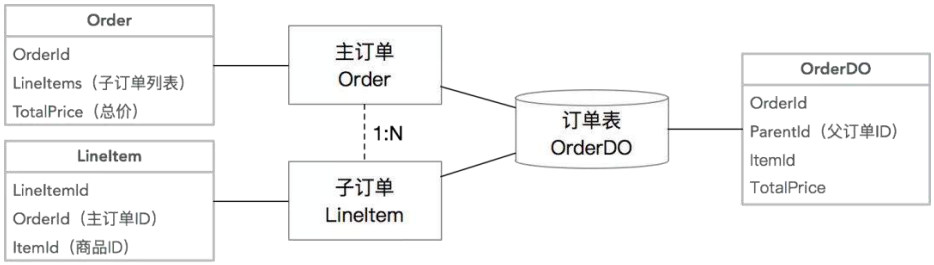
- Data Object （DO、数据对象）：实际上是我们在日常工作中最常见的数据模型。但是在 DDD 的规范里，DO 应该仅仅作为数据库物理表格的映射，不能参与到业务逻辑中。为了简单明了，DO 的字段类型和名称应该和数据库物理表格的字段类型和名称一一对应，这样我们不需要去跑到数据库上去查一个字段的类型和名称。（当然，实际上也没必要一摸一样，只要你在 Mapper 那一层做到字段映射）。
- Entity（实体对象）：实体对象是我们正常业务应该用的业务模型，它的字段和方法应该和业务语言保持一致，和持久化方式无关。也就是说，Entity 和 DO 很可能有着完全不一样的字段命名和字段类型，甚至嵌套关系。Entity 的生命周期应该仅存在于内存中，不需要可序列化和可持久化。
- DTO（传输对象）：主要作为 Application 层的入参和出参，比如 CQRS 里的 Command、Query、Event，以及 Request、Response 等都属于 DTO 的范畴。DTO 的价值在于适配不同的业务场景的入参和出参，避免让业务对象变成一个万能大对象。

模型对象之间的关系

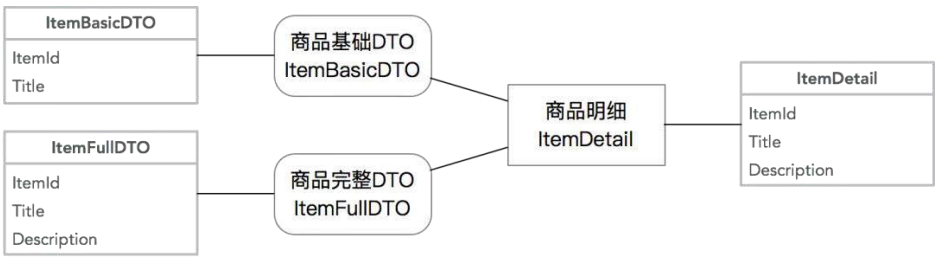
在实际开发中 DO、Entity 和 DTO 不一定是 1:1:1 的关系。一些常见的非 1:1 关系如下：复杂的 Entity 拆分多张数据库表：常见的原因在于字段过多，导致查询性能降低，需要将非检索、大字段等单独存为一张表，提升基础信息表的检索效率。常见的案例如商品模型，将商品详细描述等大字段单独保存，提升查询性能：



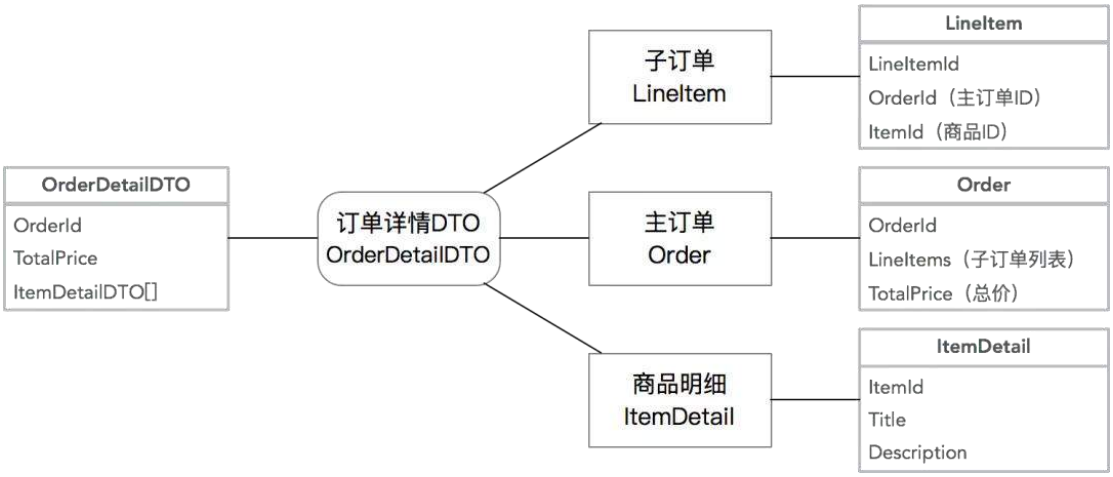
多个关联的 Entity 合并一张数据库表：这种情况通常出现在拥有复杂的 Aggregate Root - Entity 关系的情况下，且需要分库分表，为了避免多次查询和分库分表带来的不一致性，牺牲了单表的简洁性，提升查询和插入性能。常见的案例如主子订单模型：



从复杂 Entity 里抽取部分信息形成多个 DTO：这种情况通常在 Entity 复杂，但是调用方只需要部分核心信息的情况下，通过一个小的 DTO 降低信息传输成本。同样拿商品模型举例，基础 DTO 可能出现在商品列表里，这个时候不需要复杂详情：

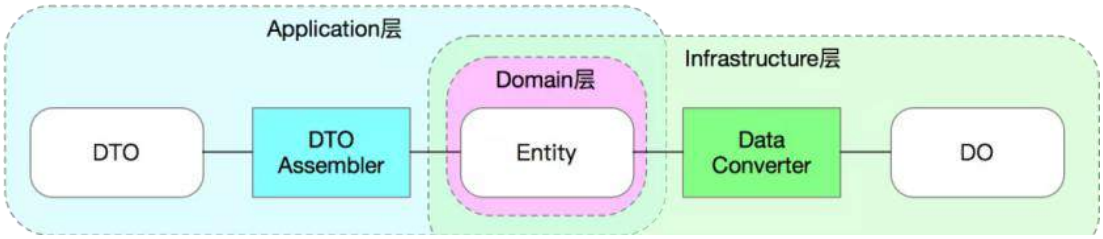


合并多个 Entity 为一个 DTO：这种情况通常为了降低网络传输成本，降低服务端请求次数，将多个 Entity、DP 等对象合并序列化，并且让 DTO 可以嵌套其他 DTO。同样常见的案例是在订单详情里需要展示商品信息：



模型所在模块和转化器

由于现在从一个对象变为 3+个对象，对象间需要通过转化器（Converter/Mapper）来互相转化。而这三种对象在代码中所在的位置也不一样，简单总结如下：



DTO Assembler：在 Application 层，Entity 到 DTO 的转化器有一个标准的名称叫 DTO Assembler。Martin Fowler 在 P of EAA 一书里对于 DTO 和 Assembler 的描述：Data Transfer Object。DTO Assembler 的核心作用就是将 1 个或多个相关联的 Entity 转化为 1 个或多个 DTO。

Data Converter：在 Infrastructure 层，Entity 到 DO 的转化器没有一个标准名称，但是为了区分 Data Mapper，我们叫这种转化器 DataConverter。这里要注意 Data

Maper 通常情况下指的是 DAO，比如 Mybatis 的 Mapper。Data Mapper 的出处也在 P of EAA 一书里：Data Mapper。

如果是手写一个 Assembler，通常我们会去实现 2 种类型的方法，如下；Data Converter 的逻辑和此类似，略过。

```
public class DtoAssembler {
// 通过各种实体，生成 DTO
  public OrderDTO toDTO(Order order, Item item) {
    OrderDTO dto = new OrderDTO();
    dto.setId(order.getId());
    dto.setItemTitle(item.getTitle()); // 从多个对象里取值，且字段名称不一样
    dto.setDetailAddress(order.getAddress.getDetail()); // 可以读取复杂嵌套字段
    // 省略 N 行
     return dto;
    }

     // 通过 DTO，生成实体
    public Item toEntity(ItemDTO itemDTO) {
    Item entity = new Item();
    entity.setId(itemDTO.getId());
    // 省略 N 行
    return entity;
    }
    }
```

我们能看出来通过抽象出一个 Assembler/Converter 对象，我们能把复杂的转化逻辑都收敛到一个对象中，并且可以很好的单元测试。这个也很好的收敛了常见代码里的转化逻辑。

在调用方使用时是非常方便的（请忽略各种异常逻辑）：

```
public class Application {
  private DtoAssembler assembler;
  private OrderRepository orderRepository;
  private ItemRepository itemRepository;

  public OrderDTO getOrderDetail(Long orderId) {
```

```
    Order order = orderRepository.find(orderId);
    Item item = itemRepository.find(order.getItemId());
    return assembler.toDTO(order, item); // 原来的很多复杂转化逻辑都收敛到一行代码了
      }
    }
```

虽然 Assembler/Converter 是非常好用的对象，但是当业务复杂时，手写 Assembler/Converter 是一件耗时且容易出 bug 的事情，所以业界会有多种 Bean Mapping 的解决方案，从本质上分为动态和静态映射。

动态映射方案包括比较原始的 BeanUtils.copyProperties、能通过 xml 配置的 Dozer 等，其核心是在运行时根据反射动态赋值。动态方案的缺陷在于大量的反射调用，性能比较差，内存占用多，不适合特别高并发的应用场景。

所以在这里我给用 Java 的同学推荐一个库叫 MapStruct（MapStruct 官网）。MapStruct 通过注解，在编译时静态生成映射代码，其最终编译出来的代码和手写的代码在性能上完全一致，且有强大的注解等能力。如果你的 IDE 支持，甚至可以在编译后看到编译出来的映射代码，用来做 check。在这里我就不细讲 MapStruct 的用法了，具体细节请见官网。

用了 MapStruct 之后，会节省大量的成本，让代码变得简洁如下：

```
@org.mapstruct.Mapper
  public interface DtoAssembler { // 注意这里变成了一个接口，MapStruct 会生成实现类
  DtoAssembler INSTANCE = Mappers.getMapper(DtoAssembler.class);

  // 在这里只需要指出字段不一致的情况，支持复杂嵌套
  @Mapping(target = "itemTitle", source = "item.title")
  @Mapping(target = "detailAddress", source = "order.address.detail")
  OrderDTO toDTO(Order order, Item item);

  // 如果字段没有不一致，一行注解都不需要
  Item toEntity(ItemDTO itemDTO);
  }
```

在使用了 MapStruct 后，你只需要标注出字段不一致的情况，其他的情况都通过 Convention over Configuration 帮你解决了。还有很多复杂的用法我就不一一指出了。

模型规范总结

	DO	Entity	DTO
目的	数据库表映射	业务逻辑	适配业务场景
代码层级	Infrastructure	Domain	Application
命名规范	XxxDO	Xxx	XxxDTO XxxCommand XxxRequest 等
字段名称标准	数据库表字段名	业务语言	和调用方商定
字段数据类型	数据库字段类型	尽量是有业务含义的类型，比如 DP	和调用方商定
是否需要序列化	不需要	不需要	需要
转化器	Data Converter	Data Converter DTO Assembler	DTO Assembler

从使用复杂度角度来看，区分了 DO、Entity、DTO 带来了代码量的膨胀（从 1 个变成了 3+2+N 个）。但是在实际复杂业务场景下，通过功能来区分模型带来的价值是功能性的单一和可测试、可预期，最终反而是逻辑复杂性的降低。

Repository 代码规范

接口规范

上文曾经讲过，传统 Data Mapper（DAO）属于“固件”，和底层实现（DB、Cache、文件系统等）强绑定，如果直接使用会导致代码“固化”。所以为了在 Repository 的设计上体现出“软件”的特性，主要需要注意以下三点：

1. 接口名称不应该使用底层实现的语法：我们常见的 insert、select、update、delete 都属于 SQL 语法，使用这几个词相当于和 DB 底层实现做了绑定。相反，我们应该把 Repository 当成一个中性的类 似 Collection 的接口，使用语法如 find、save、remove。在这里特别需要指出的是区分 insert/add 和 update 本身也是一种和底层强绑定的逻辑，一些储存如缓存实际上不存在 insert 和 update 的差异，在这个 case 里，使用中性的 save 接口，然后在具体实现上根据情况调用 DAO 的 insert 或 update 接口。
2. 出参入参不应该使用底层数据格式：需要记得的是 Repository 操作的是 Entity 对象（实际上应该是 Aggregate Root），而不应该直接操作底层的 DO 。更近一步，Repository 接口实际上应该存在于 Domain 层，根本看不到 DO 的实现。这个也是为了避免底层实现逻辑渗透到业务代码中的强保障。
3. 应该避免所谓的“通用”Repository 模式：很多 ORM 框架都提供一个“通用”的 Repository 接口，然后框架通过注解自动实现接口，比较典型的例子是 Spring Data、Entity Framework 等，这种框架的好处是在简单场景下很容易通过配置实现，但是坏处是基本上无扩展的可能性（比如加定制缓存逻辑），在未来有可能还是会被推翻重做。当然，这里避免通用不代表不能有基础接口和通用的帮助类，具体如下。

我们先定义一个基础的 Repository 基础接口类，以及一些 Marker 接口类：

```
public interface Repository<T extends Aggregate<ID>, ID extends Identifier> {

 /**
* 将一个 Aggregate 附属到一个 Repository，让它变为可追踪。
* Change-Tracking 在下文会讲，非必须
*/       void attach(@NotNull T aggregate);

 /**
* 解除一个 Aggregate 的追踪
* Change-Tracking 在下文会讲，非必须
*/
void detach(@NotNull T aggregate);

 /**
* 通过 ID 寻找 Aggregate。
```

```
 * 找到的 Aggregate 自动是可追踪的
 */
 T find(@NotNull ID id);

  /**
 * 将一个 Aggregate 从 Repository 移除
 * 操作后的 aggregate 对象自动取消追踪
 */
 void remove(@NotNull T aggregate);

  /**
 * 保存一个 Aggregate
 * 保存后自动重置追踪条件
 */
 void save(@NotNull T aggregate);
 }

 // 聚合根的 Marker 接口
 public interface Aggregate<ID extends Identifier> extends Entity<ID> {

 }

 // 实体类的 Marker 接口
 public interface Entity<ID extends Identifier> extends Identifiable<ID> {

 }

 public interface Identifiable<ID extends Identifier> {
 ID getId();
 }

 // ID 类型 DP 的 Marker 接口
 public interface Identifier extends Serializable {

 }
```

业务自己的接口只需要在基础接口上进行扩展，举个订单的例子：

```
// 代码在 Domain 层
public interface OrderRepository extends Repository<Order, OrderId> {

// 自定义 Count 接口，在这里 OrderQuery 是一个自定义的 DTO
Long count(OrderQuery query);

// 自定义分页查询接口
Page<Order> query(OrderQuery query);

// 自定义有多个条件的查询接口
Order findInStore(OrderId id, StoreId storeId);
}
```

每个业务需要根据自己的业务场景来定义各种查询逻辑。

这里需要再次强调的是 Repository 的接口是在 Domain 层，但是实现类是在 Infrastructure 层。

Repository 基础实现

先举个 Repository 的最简单实现的例子。注意 OrderRepositoryImpl 在 Infrastructure 层：

```
// 代码在 Infrastructure 层
@Repository // Spring 的注解
public class OrderRepositoryImpl implements OrderRepository {
  private final OrderDAO dao; // 具体的 DAO 接口
  private final OrderDataConverter converter; // 转化器

  public OrderRepositoryImpl(OrderDAO dao) {
    this.dao = dao;
    this.converter = OrderDataConverter.INSTANCE;     }

  @Override
  public Order find(OrderId orderId) {
    OrderDO orderDO = dao.findById(orderId.getValue());
    return converter.fromData(orderDO);
```

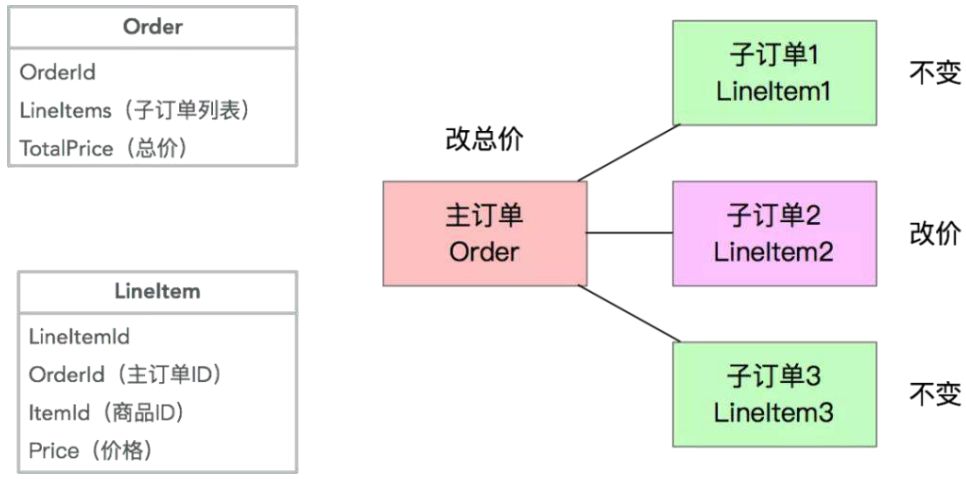
```
        }

        @Override
        public void remove(Order aggregate) {
        OrderDO orderDO = converter.toData(aggregate);
        dao.delete(orderDO);
        }

        @Override
        public void save(Order aggregate) {
        if (aggregate.getId() != null && aggregate.getId().getValue() > 0) {
          // update
          OrderDO orderDO = converter.toData(aggregate);
          dao.update(orderDO);
        } else {
          // insert
          OrderDO orderDO = converter.toData(aggregate);
          dao.insert(orderDO);
          aggregate.setId(converter.fromData(orderDO).getId());
          }
    }

        @Override
          public Page<Order> query(OrderQuery query) {
            List<OrderDO> orderDOS = dao.queryPaged(query);
            long count = dao.count(query);
            List<Order> result =
orderDOS.stream().map(converter::fromData).collect(Collectors.toList());
            return Page.with(result, query, count);       }

        @Override
          public Order findInStore(OrderId id, StoreId storeId) {
            OrderDO orderDO = dao.findInStore(id.getValue(), storeId.getValue());
            return converter.fromData(orderDO);
          }
    }
```

从上面的实现能看出来一些套路：所有的 Entity/Aggregate 会被转化为 DO，然后根据业务场景，调用相应的 DAO 方法进行操作，事后如果需要则把 DO 转换回 Entity。代码基本很简单，唯一需要注意的是 save 方法，需要根据 Aggregate 的 ID 是否存在且大于 0 来判断

一个 Aggregate 是否需要更新还是插入。

Repository 复杂实现

针对单一 Entity 的 Repository 实现一般比较简单，但当涉及到多 Entity 的 Aggregate Root 时，就会比较麻烦，最主要的原因是在一次操作中，并不是所有 Aggregate 里的 Entity 都需要变更，但是如果用简单的写法，会导致大量的无用 DB 操作。举一个常见的例子，在主子订单的场景下，一个主订单 Order 会包含多个子订单 LineItem，假设有个改某个子订单价格的操作，会同时改变主订单价格，但是对其他子订单无影响：



如果用一个非常 naive 的实现来完成，会导致多出来两个无用的更新操作，如下：

```
public class OrderRepositoryImpl extends implements OrderRepository {
  private OrderDAO orderDAO;
  private LineItemDAO lineItemDAO;
  private OrderDataConverter orderConverter;
  private LineItemDataConverter lineItemConverter;

  // 其他逻辑省略

  @Override
    public void save(Order aggregate) {
      if (aggregate.getId() != null && aggregate.getId().getValue() > 0) {
      // 每次都将 Order 和所有 LineItem 全量更新
```

```
        OrderDO orderDO = orderConverter.toData(aggregate);
        orderDAO.update(orderDO);
         for (LineItem lineItem: aggregate.getLineItems()) {
        save(lineItem);
          }
        } else {
        // 插入逻辑省略
          }      }

         private void save(LineItem lineItem) {
          if (lineItem.getId() != null && lineItem.getId().getValue() > 0) {
             LineItemDO lineItemDO = lineItemConverter.toData(lineItem);
             lineItemDAO.update(lineItemDO);
        } else {
          LineItemDO lineItemDO = lineItemConverter.toData(lineItem);
          lineItemDAO.insert(lineItemDO);
          lineItem.setId(lineItemConverter.fromData(lineItemDO).getId());
             }
          }
        }
```

在这个情况下，会导致 4 个 UPDATE 操作，但实际上只需要 2 个。在绝大部分情况下，这个成本不高，可以接受，但是在极端情况下（当非 Aggregate Root 的 Entity 非常多时），会导致大量的无用写操作。

Change-Tracking 变更追踪

在上面那个案例里，核心的问题是由于 Repository 接口规范的限制，让调用方仅能操作 Aggregate Root，而无法单独针对某个非 Aggregate Root 的 Entity 直接操作。这个和直接调用 DAO 的方式很不一样。

这个的解决方案是需要能识别到底哪些 Entity 有变更，并且只针对那些变更过的 Entity 做操作，就需要加上变更追踪的能力。换一句话说就是原来很多人为判断的代码逻辑，现在可以通过变更追踪来自动实现，让使用方真正只关心 Aggregate 的操作。在上一个案例里，通过变更追踪，系统可以判断出来只有 LineItem2 和 Order 有变更，所以只需要生成两个 UPDATE 即可。

业界有两个主流的变更追踪方案：

- 基于 Snapshot 的方案：当数据从 DB 里取出来后，在内存中保存一份 snapshot，然后在数据写入时和 snapshot 比较。常见的实现如 Hibernate。
- 基于 Proxy 的方案：当数据从 DB 里取出来后，通过 weaving 的方式将所有 setter 都增加一个切面来判断 setter 是否被调用以及值是否变更，如果变更则标记为 Dirty。在保存时根据 Dirty 判断是否需要更新。常见的实现如 Entity Framework。

Snapshot 方案的好处是比较简单，成本在于每次保存时全量 Diff 的操作（一般用 Reflection），以及保存 Snapshot 的内存消耗。

Proxy 方案的好处是性能很高，几乎没有增加的成本，但是坏处是实现起来比较困难，且当有嵌套关系存在时不容易发现嵌套对象的变化（比如子 List 的增加和删除等），有可能导致 bug。

由于 Proxy 方案的复杂度，业界主流（包括 EF Core）都在使用 Snapshot 方案。这里面还有另一个好处就是通过 Diff 可以发现哪些字段有变更，然后只更新变更过的字段，再一次降低 UPDATE 的成本。

在这里我简单贴一下我们自己 Snapshot 的实现，代码并不复杂，每个团队自己实现起来也很简单，部分代码仅供参考：

DbRepositorySupport：

```
    // 这个类是一个通用的支撑类，为了减少开发者的重复劳动。在用的时候需要继承这个类
    public abstract class DbRepositorySupport<T extends Aggregate<ID>, ID extends
Identifier> implements Repository<T, ID> {

    @Getter
    private final Class<T> targetClass;

     // 让 AggregateManager 去维护 Snapshot
```

```
@Getter(AccessLevel.PROTECTED)
private AggregateManager<T, ID> aggregateManager;

protected DbRepositorySupport(Class<T> targetClass) {
this.targetClass = targetClass;
this.aggregateManager = AggregateManager.newInstance(targetClass);
}

/**
* 这几个方法是继承的子类应该去实现的
*/
   protected abstract void onInsert(T aggregate);
   protected abstract T onSelect(ID id);
   protected abstract void onUpdate(T aggregate, EntityDiff diff);
   protected abstract void onDelete(T aggregate);

/**
* Attach 的操作就是让 Aggregate 可以被追踪
*/
   @Override
   public void attach(@NotNull T aggregate) {
   this.aggregateManager.attach(aggregate);
}

/**
* Detach 的操作就是让 Aggregate 停止追踪
*/
   @Override
   public void detach(@NotNull T aggregate) {
   this.aggregateManager.detach(aggregate);
}

   @Override
   public T find(@NotNull ID id) {
   T aggregate = this.onSelect(id);
   if (aggregate != null) {
  // 这里的就是让查询出来的对象能够被追踪。
// 如果自己实现了一个定制查询接口，要记得单独调用 attach。
   this.attach(aggregate);              }
   return aggregate;
}
```

```
    @Override
    public void remove(@NotNull T aggregate) {              t
    his.onDelete(aggregate);
  // 删除停止追踪
    this.detach(aggregate);
  }

    @Override
    public void save(@NotNull T aggregate) {
  // 如果没有 ID，直接插入
    if (aggregate.getId() == null) {
    this.onInsert(aggregate);
    this.attach(aggregate);
    return;
  }

  // 做 Diff
    EntityDiff diff = aggregateManager.detectChanges(aggregate);
    if (diff.isEmpty()) {
    return;
    }

  // 调用 UPDATE
  this.onUpdate(aggregate, diff);

  // 最终将 DB 带来的变化更新回 AggregateManager
  aggregateManager.merge(aggregate);
  }
}
```

使用方只需要继承 DbRepositorySupport：

```
  public class OrderRepositoryImpl extends DbRepositorySupport<Order, OrderId>
implements OrderRepository {
    private OrderDAO orderDAO;
    private LineItemDAO lineItemDAO;
    private OrderDataConverter orderConverter;
    private LineItemDataConverter lineItemConverter;

    // 部分代码省略，见上文
```

```
        @Override
        protected void onUpdate(Order aggregate, EntityDiff diff) {
          if (diff.isSelfModified()) {
            OrderDO orderDO = converter.toData(aggregate);
            orderDAO.update(orderDO);
          }

            Diff lineItemDiffs = diff.getDiff("lineItems");
          if (lineItemDiffs instanceof ListDiff) {
            ListDiff diffList = (ListDiff) lineItemDiffs;
            for (Diff itemDiff : diffList) {
          if (itemDiff.getType() == DiffType.Removed) {
            LineItem line = (LineItem) itemDiff.getOldValue();
            LineItemDO lineDO = lineItemConverter.toData(line);
            lineItemDAO.delete(lineDO);
          }
          if (itemDiff.getType() == DiffType.Added) {
            LineItem line = (LineItem) itemDiff.getNewValue();
            LineItemDO lineDO = lineItemConverter.toData(line);
            lineItemDAO.insert(lineDO);
          }
          if (itemDiff.getType() == DiffType.Modified) {
            LineItem line = (LineItem) itemDiff.getNewValue();
            LineItemDO lineDO = lineItemConverter.toData(line);
            lineItemDAO.update(lineDO);
                      }
                    }
                  }
                }
          }
```

AggregateManager 实现，主要是通过 ThreadLocal 避免多线程公用同一个 Entity 的情况：

```
    class ThreadLocalAggregateManager<T extends Aggregate<ID>, ID extends Identifier>
implements AggregateManager<T, ID> {

        private ThreadLocal<DbContext<T, ID>> context;
        private Class<? extends T> targetClass;

        public ThreadLocalAggregateManager(Class<? extends T> targetClass) {
          this.targetClass = targetClass;
```

```
    this.context = ThreadLocal.withInitial(() -> new DbContext<>(targetClass));
    }

  public void attach(T aggregate) {
    context.get().attach(aggregate);
    }

  @Override
  public void attach(T aggregate, ID id) {
    context.get().setId(aggregate, id);
    context.get().attach(aggregate);
    }

  @Override
    public void detach(T aggregate) {
    context.get().detach(aggregate);
    }

  @Override
  public T find(ID id) {
    return context.get().find(id);
    }

  @Override
  public EntityDiff detectChanges(T aggregate) {
    return context.get().detectChanges(aggregate);
    }

   public void merge(T aggregate) {
    context.get().merge(aggregate);
    }
  }
```

```
class DbContext<T extends Aggregate<ID>, ID extends Identifier> {

  @Getter
  private Class<? extends T> aggregateClass;

    private Map<ID, T> aggregateMap = new HashMap<>();

    public DbContext(Class<? extends T> aggregateClass) {
     this.aggregateClass = aggregateClass;
```

```
        }

    public void attach(T aggregate) {
        if (aggregate.getId() != null) {
        if (!aggregateMap.containsKey(aggregate.getId())) {
        this.merge(aggregate);
        }
    }
        }

    public void detach(T aggregate) {
        if (aggregate.getId() != null) {
        aggregateMap.remove(aggregate.getId());
        }
    }

    public EntityDiff detectChanges(T aggregate) {
        if (aggregate.getId() == null) {
        return EntityDiff.EMPTY;
        }
    T snapshot = aggregateMap.get(aggregate.getId());
        if (snapshot == null) {
            attach(aggregate);
}
    return DiffUtils.diff(snapshot, aggregate);
        }

    public T find(ID id) {
        return aggregateMap.get(id);
        }

    public void merge(T aggregate) {
        if (aggregate.getId() != null) {
        T snapshot = SnapshotUtils.snapshot(aggregate);
            aggregateMap.put(aggregate.getId(), snapshot);
        }
    }

    public void setId(T aggregate, ID id) {
        ReflectionUtils.writeField("id", aggregate, id);
        }
    }
```

跑个单测（注意在这个 case 里我把 Order 和 LineItem 合并单表了）：

```
    @Test
      public void multiInsert() {
        OrderDAO dao = new MockOrderDAO();

        OrderRepository repo = new OrderRepositoryImpl(dao);

        Order order = new Order();
        order.setUserId(new UserId(11L));
        order.setStatus(OrderState.ENABLED);
        order.addLineItem(new ItemId(13L), new Quantity(5), new Money(4));
        order.addLineItem(new ItemId(14L), new Quantity(2), new Money(3));

        System.out.println("第一次保存前");
        System.out.println(order);

        repo.save(order);
        System.out.println("第一次保存后");
        System.out.println(order);

        order.getLineItems().get(0).setQuantity(new Quantity(3));
        order.pay();
        repo.save(order);

        System.out.println("第二次保存后");
        System.out.println(order);
          }
```

单测结果：

```
    第一次保存前
    Order(id=null, userId=11, lineItems=[LineItem(id=null, itemId=13, quantity=5, price=4),
LineItem(id=null, itemId=14, quantity=2, price=3)], status=ENABLED)

    INSERT OrderDO: OrderDO(id=null, parentId=null, itemId=0, userId=11, quantity=0,
price=0, status=2)
    UPDATE OrderDO: OrderDO(id=1001, parentId=1001, itemId=0, userId=11, quantity=0,
price=0, status=2)
    INSERT OrderDO: OrderDO(id=null, parentId=1001, itemId=13, userId=11, quantity=5,
price=4, status=2)
    INSERT OrderDO: OrderDO(id=null, parentId=1001, itemId=14, userId=11, quantity=2,
price=3, status=2)

    第一次保存后
```

```
    Order(id=1001, userId=11, lineItems=[LineItem(id=1002, itemId=13, quantity=5, price=4),
LineItem(id=1003, itemId=14, quantity=2, price=3)], status=ENABLED)

    UPDATE OrderDO: OrderDO(id=1001, parentId=1001, itemId=0, userId=11, quantity=0,
price=0, status=3)
    UPDATE OrderDO: OrderDO(id=1002, parentId=1001, itemId=13, userId=11, quantity=3,
price=4, status=3)

    第二次保存后
    Order(id=1001, userId=11, lineItems=[LineItem(id=1002, itemId=13, quantity=3, price=4),
LineItem(id=1003, itemId=14, quantity=2, price=3)], status=PAID)
```

其他注意事项

并发乐观锁

在高并发情况下，如果使用上面的 Change-Tracking 方法，由于 Snapshot 在本地内存的数据*有可能* 和 DB 数据不一致，会导致并发冲突的问题，这个时候需要在更新时加入乐观锁。当然，正常数据库操作的 Best Practice 应该也要有乐观锁，只不过在这个 case 里，需要在乐观锁冲突后，记得更新本地 Snapshot 里的值。

一个可能的 BUG

这个其实算不上 bug，但是单独指出来希望大家能注意一下，使用 Snapshot 的一个副作用就是如果没更新 Entity 然后调用了 save 方法，这时候实际上是不会去更新 DB 的。这个逻辑跟 Hibernate 的逻辑一致，是 Snapshot 方法的天生特性。如果要强制更新到 DB，建议手动更改一个字段如 gmtModified，然后再调用 save。

Repository 迁移路径

在我们日常的代码中，使用 Repository 模式是一个很简单，但是又能得到很多收益的事情。最大的收益就是可以彻底和底层实现解耦，让上层业务可以快速自发展。

我们假设现有的传统代码包含了以下几个类（还是用订单举例）：

- OrderDO
- OrderDAO

可以通过以下几个步骤逐渐的实现 Repository 模式：

1. 生成 Order 实体类，初期字段可以和 OrderDO 保持一致。
2. 生成 OrderDataConverter，通过 MapStruct 基本上 2 行代码就能完成。
3. 写单元测试，确保 Order 和 OrderDO 之间的转化 100%正确。
4. 生成 OrderRepository 接口和实现，通过单测确保 OrderRepository 的正确性。
5. 将原有代码里使用了 OrderDO 的地方改为 Order。
6. 将原有代码里使用了 OrderDAO 的地方都改为用 OrderRepository。
7. 通过单测确保业务逻辑的一致性。

恭喜你！从现在开始 Order 实体类和其业务逻辑可以随意更改，每次修改你唯一需要做的就是变更一下 Converter，已经和底层实现完全解藕了。

写在后面

感谢你，能有耐心看到这里的都是 DDD 真爱。一个问题，你是否在日常工作中能大量的利用 DDD 的架构来推进你的业务？你是否有一个环境能把你的所学用到真正实战中去？

链接：

https://martinfowler.com/eaaCatalog/dataTransferObject.html?spm=ata.13261165.0.0.590a62fcaM6bCk

https://martinfowler.com/eaaCatalog/dataMapper.html?spm=ata.13261165.0.0.590a62fcaM6bCk

https://mapstruct.org/?spm=ata.13261165.0.0.590a62fcaM6bCk

阿里技术专家详解 DDD 系列 第四讲 - 领域层设计规范

作者 | 殷浩
出品 | 淘系技术

在一个 DDD 架构设计中，领域层的设计合理性会直接影响整个架构的代码结构以及应用层、基础设施层的设计。但是领域层设计又是有挑战的任务，特别是在一个业务逻辑相对复杂应用中，每个业务规则是应该放在 Entity、ValueObject 还是 DomainService 是值得用心思考的，既要避免未来的扩展性差，又要确保不会过度设计导致复杂性。今天我用一个相对轻松易懂的领域做一个案例演示，但在实际业务应用中，无论是交易、营销还是互动，都可以用类似的逻辑来实现。

初探龙与魔法的世界架构

背景和规则

平日里看了好多严肃的业务代码，今天找一个轻松的话题，如何用代码实现一个龙与魔法的游戏世界的（极简）规则?

基础配置如下:

- 玩家（Player）可以是战士（Fighter）、法师（Mage）、龙骑（Dragoon）
- 怪物（Monster）可以是兽人（Orc）、精灵（Elf）、龙（Dragon），怪物有血量
- 武器（Weapon）可以是剑（Sword）、法杖（Staff），武器有攻击力
- 玩家可以装备一个武器，武器攻击可以是物理类型（0），火（1），冰（2）等，武器类型决定伤害类型

攻击规则如下：

- 兽人对物理攻击伤害减半
- 精灵对魔法攻击伤害减半
- 龙对物理和魔法攻击免疫，除非玩家是龙骑，则伤害加倍

OOP 实现

对于熟悉 Object-Oriented Programming 的同学，一个比较简单的实现是通过类的继承关系（此处省略部分非核心代码）：

```
public abstract class Player {
Weapon weapon
}
public class Fighter extends Player {}
public class Mage extends Player {}
public class Dragoon extends Player {}

public abstract class Monster {
Long health;
}
public Orc extends Monster {}
public Elf extends Monster {}
public Dragoon extends Monster {}

public abstract class Weapon {
int damage;
int damageType; // 0 - physical, 1 - fire, 2 - ice etc.
}
public Sword extends Weapon {}
public Staff extends Weapon {}
```

而实现规则代码如下：

```
public class Player {
  public void attack(Monster monster) {
  monster.receiveDamageBy(weapon, this);
```

```
    }
}

public class Monster {
    public void receiveDamageBy(Weapon weapon, Player player) {
    this.health -= weapon.getDamage(); // 基础规则
        }
    }

    public class Orc extends Monster {
    @Override
    public void receiveDamageBy(Weapon weapon, Player player) {
    if (weapon.getDamageType() == 0) {
    this.setHealth(this.getHealth() - weapon.getDamage() / 2); // Orc 的物理防御规则
    } else {
    super.receiveDamageBy(weapon, player);
          }
        }
    }

    public class Dragon extends Monster {
    @Override
        public void receiveDamageBy(Weapon weapon, Player player) {
        if (player instanceof Dragoon) {
        this.setHealth(this.getHealth() - weapon.getDamage() * 2); // 龙骑伤害规则
        }
        // else no damage, 龙免疫力规则

         }

        }
```

然后跑几个单测：

```
public class BattleTest {


    @Test
    @DisplayName("Dragon is immune to attacks")
    public void testDragonImmunity() {
    // Given
    Fighter fighter = new Fighter("Hero");
    Sword sword = new Sword("Excalibur", 10);
```

```
fighter.setWeapon(sword);
Dragon dragon = new Dragon("Dragon", 100L);

// When
 fighter.attack(dragon);

// Then
assertThat(dragon.getHealth()).isEqualTo(100);
}

@Test
@DisplayName("Dragoon attack dragon doubles damage")
public void testDragoonSpecial() {
// Given
Dragoon dragoon = new Dragoon("Dragoon");
Sword sword = new Sword("Excalibur", 10);
dragoon.setWeapon(sword);
Dragon dragon = new Dragon("Dragon", 100L);

 // When
dragoon.attack(dragon);

// Then
assertThat(dragon.getHealth()).isEqualTo(100 - 10 * 2);
}

@Test
@DisplayName("Orc should receive half damage from physical weapons")
public void testFighterOrc() {
// Given
Fighter fighter = new Fighter("Hero");
Sword sword = new Sword("Excalibur", 10);
fighter.setWeapon(sword);
Orc orc = new Orc("Orc", 100L);

// When
 fighter.attack(orc);

// Then
assertThat(orc.getHealth()).isEqualTo(100 - 10 / 2);
}

@Test
@DisplayName("Orc receive full damage from magic attacks")
```

```
    public void testMageOrc() {
    // Given
    Mage mage = new Mage("Mage");
    Staff staff = new Staff("Fire Staff", 10);
    mage.setWeapon(staff);
    Orc orc = new Orc("Orc", 100L);

    // When
    mage.attack(orc);

     // Then
    assertThat(orc.getHealth()).isEqualTo(100 - 10);
      }
    }
```

以上代码和单测都比较简单，不做多余的解释了。

分析 OOP 代码的设计缺陷

编程语言的强类型无法承载业务规则

以上的 OOP 代码可以跑得通，直到我们加一个限制条件：

- 战士只能装备剑
- 法师只能装备法杖

这个规则在 Java 语言里无法通过强类型来实现，虽然 Java 有 Variable Hiding（或者 C#的 new class variable），但实际上只是在子类上加了一个新变量，所以会导致以下的问题：

```
@Data
public class Fighter extends Player {
 private Sword weapon;
}

@Test
public void testEquip() {
Fighter fighter = new Fighter("Hero");
```

```
Sword sword = new Sword("Sword", 10);
fighter.setWeapon(sword);

Staff staff = new Staff("Staff", 10);
fighter.setWeapon(staff);

assertThat(fighter.getWeapon()).isInstanceOf(Staff.class); // 错误了
  }
```

在最后，虽然代码感觉是 setWeapon(Staff)，但实际上只修改了父类的变量，并没有修改子类的变量，所以实际不生效，也不抛异常，但结果是错的。

当然，可以在父类限制 setter 为 protected，但这样就限制了父类的 API，极大的降低了灵活性，同时也违背了 Liskov substitution principle，即一个父类必须要 cast 成子类才能使用：

```
@Data
public abstract class Player {
@Setter(AccessLevel.PROTECTED)
private Weapon weapon;
}

@Test
public void testCastEquip() {
Fighter fighter = new Fighter("Hero");

Sword sword = new Sword("Sword", 10);
fighter.setWeapon(sword);

Player player = fighter;
Staff staff = new Staff("Staff", 10);
player.setWeapon(staff); // 编译不过，但从 API 层面上应该开放可用
  }
```

最后，如果规则增加一条：

- 战士和法师都能装备匕首（dagger）

BOOM，之前写的强类型代码都废了，需要重构。

对象继承导致代码强依赖父类逻辑，违反开闭原则 Open-Closed Principle（OCP）

开闭原则（OCP）规定"对象应该对于扩展开放，对于修改封闭"，继承虽然可以通过子类扩展新的行为，但因为子类可能直接依赖父类的实现，导致一个变更可能会影响所有对象。在这个例子里，如果增加任意一种类型的玩家、怪物或武器，或增加一种规则，都有可能需要修改从父类到子类的所有方法。

比如，如果要增加一个武器类型：狙击枪，能够无视所有防御一击必杀，需要修改的代码包括：

- Weapon
- Player 和所有的子类（是否能装备某个武器的判断）
- Monster 和所有的子类（伤害计算逻辑）

```
public class Monster {
  public void receiveDamageBy(Weapon weapon, Player player) {
   this.health -= weapon.getDamage(); // 老的基础规则
  if (Weapon instanceof Gun) { // 新的逻辑
 this.setHealth(0);
    }
  }
}

public class Dragon extends Monster {
  public void receiveDamageBy(Weapon weapon, Player player) {
    if (Weapon instanceof Gun) { // 新的逻辑
    super.receiveDamageBy(weapon, player);
    }
            // 老的逻辑省略
      }
    }
```

在一个复杂的软件中为什么会建议"尽量"不要违背 OCP？最核心的原因就是一个现有逻辑的变更可能会影响一些原有的代码，导致一些无法预见的影响。这个风险只能通过完整的单元测试覆盖来保障，但在实际开发中很难保障单测的覆盖率。OCP 的原则能尽可能的规避这种风险，当新的行为只能通过新的字段/方法来实现时，老代码的行为自然不会变。

继承虽然能 Open for extension，但很难做到 Closed for modification。所以今天解决 OCP 的主要方法是通过 Composition-over-inheritance，即通过组合来做到扩展性，而不是通过继承。

Player.attack(monster) 还是 Monster.receiveDamage(Weapon, Player)?

在这个例子里，其实业务规则的逻辑到底应该写在哪里是有异议的：当我们去看一个对象和另一个对象之间的交互时，到底是 Player 去攻击 Monster，还是 Monster 被 Player 攻击？目前的代码主要将逻辑写在 Monster 的类中，主要考虑是 Monster 会受伤降低 Health，但如果是 Player 拿着一把双刃剑会同时伤害自己呢？是不是发现写在 Monster 类里也有问题？代码写在哪里的原则是什么？

多对象行为类似，导致代码重复

当我们有不同的对象，但又有相同或类似的行为时，OOP 会不可避免的导致代码的重复。在这个例子里，如果我们去增加一个"可移动"的行为，需要在 Player 和 Monster 类中都增加类似的逻辑：

```
public abstract class Player {
  int x;
  int y;
  void move(int targetX, int targetY) {
  // logic
    }
  }

    public abstract class Monster {
    int x;
```

```
    int y;
    void move(int targetX, int targetY) {
    // logic
      }
    }
```

一个可能的解法是有个通用的父类：

```
public abstract class Movable {
  int x;
  int y;
  void move(int targetX, int targetY) {
  // logic
 }
}

public abstract class Player extends Movable;
public abstract class Monster extends Movable;
```

但如果再增加一个跳跃能力 Jumpable 呢？一个跑步能力 Runnable 呢？如果 Player 可以 Move 和 Jump，Monster 可以 Move 和 Run，怎么处理继承关系？要知道 Java（以及绝大部分语言）是不支持多父类继承的，所以只能通过重复代码来实现。

问题总结

在这个案例里虽然从直觉来看 OOP 的逻辑很简单，但如果你的业务比较复杂，未来会有大量的业务规则变更时，简单的 OOP 代码会在后期变成复杂的一团浆糊，逻辑分散在各地，缺少全局视角，各种规则的叠加会触发 bug。有没有感觉似曾相识？对的，电商体系里的优惠、交易等链路经常会碰到类似的坑。而这类问题的核心本质在于：

- 业务规则的归属到底是对象的“行为”还是独立的“规则对象”？
- 业务规则之间的关系如何处理？
- 通用“行为”应该如何复用和维护？

在讲 DDD 的解法前，我们先去看一套游戏里最近比较火的架构设计，Entity-Component-System（ECS）是如何实现的。

Entity-Component-System（ECS）架构简介

ECS 介绍

ECS 架构模式是其实是一个很老的游戏架构设计，最早应该能追溯到《地牢围攻》的组件化设计，但最近因为 Unity 的加入而开始变得流行（比如《守望先锋》就是用的 ECS）。要很快的理解 ECS 架构的价值，我们需要理解一个游戏代码的核心问题：

- 性能：游戏必须要实现一个高的渲染率（60FPS），也就是说整个游戏世界需要在 1/60s（大概 16ms）内完整更新一次（包括物理引擎、游戏状态、渲染、AI 等）。而在一个游戏中，通常有大量的（万级、十万级）游戏对象需要更新状态，除了渲染可以依赖 GPU 之外，其他的逻辑都需要由 CPU 完成，甚至绝大部分只能由单线程完成，导致绝大部分时间复杂场景下 CPU（主要是内存到 CPU 的带宽）会成为瓶颈。在 CPU 单核速度几乎不再增加的时代，如何能让 CPU 处理的效率提升，是提升游戏性能的核心。
- 代码组织：如同第一章讲的案例一样，当我们用传统 OOP 的模式进行游戏开发时，很容易就会陷入代码组织上的问题，最终导致代码难以阅读，维护和优化。
- 可扩展性：这个跟上一条类似，但更多的是游戏的特性导致：需要快速更新，加入新的元素。一个游戏的架构需要能通过低代码、甚至 0 代码的方式增加游戏元素，从而通过快速更新而留住用户。如果每次变更都需要开发新的代码，测试，然后让用户重新下载客户端，可想而知这种游戏很难在现在的竞争环境下活下来。

而 ECS 架构能很好的解决上面的几个问题，ECS 架构主要分为：

- Entity：用来代表任何一个游戏对象，但是在 ECS 里一个 Entity 最重要的仅仅是他的 EntityID，一个 Entity 里包含多个 Component

- Component：是真正的数据，ECS 架构把一个个的实体对象拆分为更加细化的组件，比如位置、素材、状态等，也就是说一个 Entity 实际上只是一个 Bag of Components。
- System（或者 ComponentSystem，组件系统）：是真正的行为，一个游戏里可以有很多个不同的组件系统，每个组件系统都只负责一件事，可以依次处理大量的相同组件，而不需要去理解具体的 Entity。所以一个 ComponentSystem 理论上可以有更加高效的组件处理效率，甚至可以实现并行处理，从而提升 CPU 利用率。

ECS 的一些核心性能优化包括将同类型组件放在同一个 Array 中，然后 Entity 仅保留到各自组件的 pointer，这样能更好的利用 CPU 的缓存，减少数据的加载成本，以及 SIMD 的优化等。

一个 ECS 案例的伪代码如下：

```
public class Entity {
  public Vector position; // 此处 Vector 是一个 Component，指向的是 MovementSystem.list 里的一个
    }

    public class MovementSystem {
    List<Vector> list;

    // System 的行为  public void update(float delta) {
      for(Vector pos : list) { // 这个 loop 直接走了 CPU 缓存，性能很高，同时可以用 SIMD 优化
      pos.x = pos.x + delta;
      pos.y = pos.y + delta;
        }
       }
      }

      @Test
      public void test() {
        MovementSystem system = new MovementSystem();
        system.list = new List<>() { new Vector(0, 0) };
        Entity entity = new Entity(list.get(0));
        system.update(0.1);
        assertTrue(entity.position.x == 0.1);
        }
```

由于本文不是讲解 ECS 架构的，感兴趣的同学可以搜索 Entity-Component-System 或者看看 Unity 的 ECS 文档等。

ECS 架构分析

重新回来分析 ECS，其实它的本源还是几个很老的概念：

组件化

在软件系统里，我们通常将复杂的大系统拆分为独立的组件，来降低复杂度。比如网页里通过前端组件化降低重复开发成本，微服务架构通过服务和数据库的拆分降低服务复杂度和系统影响面等。但是 ECS 架构把这个走到了极致，即每个对象内部都实现了组件化。通过将一个游戏对象的数据和行为拆分为多个组件和组件系统，能实现组件的高度复用性，降低重复开发成本。

行为抽离

这个在游戏系统里有个比较明显的优势。如果按照 OOP 的方式，一个游戏对象里可能会包括移动代码、战斗代码、渲染代码、AI 代码等，如果都放在一个类里会很长，且很难去维护。通过将通用逻辑抽离出来为单独的 System 类，可以明显提升代码的可读性。另一个好处则是抽离了一些和对象代码无关的依赖，比如上文的 delta，这个 delta 如果是放在 Entity 的 update 方法，则需要作为入参注入，而放在 System 里则可以统一管理。在第一章的有个问题，到底是应该 Player.attack(monster) 还是 Monster.receiveDamage(Weapon, Player)。在 ECS 里这个问题就变的很简单，放在 CombatSystem 里就可以了。

数据驱动

即一个对象的行为不是写死的而是通过其参数决定，通过参数的动态修改，就可以快速改变一个对象的具体行为。在 ECS 的游戏架构里，通过给 Entity 注册相应的 Component，以及改变 Component 的具体参数的组合，就可以改变一个对象的行为和玩法，比如创建一个水

壶+爆炸属性就变成了“爆炸水壶”、给一个自行车加上风魔法就变成了飞车等。在有些 Rougelike 游戏中，可能有超过 1 万件不同类型、不同功能的物品，如果这些不同功能的物品都去单独写代码，可能永远都写不完，但是通过数据驱动+组件化架构，所有物品的配置最终就是一张表，修改也极其简单。这个也是组合胜于继承原则的一次体现。

ECS 的缺陷

虽然 ECS 在游戏界已经开始崭露头角，我发现 ECS 架构目前还没有在哪个大型商业应用中被使用过。原因可能很多，包括 ECS 比较新大家还不了解、缺少商业成熟可用的框架、程序员们还不够能适应从写逻辑脚本到写组件的思维转变等，但我认为其最大的一个问题是 ECS 为了提升性能，强调了数据/状态（State）和行为（Behaivor）分离，并且为了降低 GC 成本，直接操作数据，走到了一个极端。而在商业应用中，数据的正确性、一致性和健壮性应该是最高的优先级，而性能只是锦上添花的东西，所以 ECS 很难在商业场景里带来特别大的好处。但这不代表我们不能借鉴一些 ECS 的突破性思维，包括组件化、跨对象行为的抽离、以及数据驱动模式，而这些在 DDD 里也能很好的用起来。

基于 DDD 架构的一种解法

领域对象

回到我们原来的问题域上面，我们从领域层拆分一下各种对象：

实体类

在 DDD 里，实体类包含 ID 和内部状态，在这个案例里实体类包含 Player、Monster 和 Weapon。Weapon 被设计成实体类是因为两把同名的 Weapon 应该可以同时存在，所以必须要有 ID 来区分，同时未来也可以预期 Weapon 会包含一些状态，比如升级、临时的 buff、耐久等。

```
public class Player implements Movable {
  private PlayerId id;      private String name;
  private PlayerClass playerClass; // enum
  private WeaponId weaponId; //  (Note 1)
  private Transform position = Transform.ORIGIN;
  private Vector velocity = Vector.ZERO;
    }

    public class Monster implements Movable {
      private MonsterId id;
      private MonsterClass monsterClass; // enum
      private Health health;
      private Transform position = Transform.ORIGIN;
      private Vector velocity = Vector.ZERO;}

      public class Weapon {
      private WeaponId id;
      private String name;
      private WeaponType weaponType; // enum
      private int damage;
      private int damageType; // 0 - physical, 1 - fire, 2 - ice
        }
```

在这个简单的案例里，我们可以利用 enum 的 PlayerClass、MonsterClass 来代替继承关系，后续也可以利用 Type Object 设计模式来做到数据驱动。

Note 1: 因为 Weapon 是实体类，但是 Weapon 能独立存在，Player 不是聚合根，所以 Player 只能保存 WeaponId，而不能直接指向 Weapon。

值对象的组件化

在前面的 ECS 架构里，有个 MovementSystem 的概念是可以复用的，虽然不应该直接去操作 Component 或者继承通用的父类，但是可以通过接口的方式对领域对象做组件化处理：

```
public interface Movable {
// 相当于组件
Transform getPosition();
Vector getVelocity();
```

```
// 行为
void moveTo(long x, long y);
void startMove(long velX, long velY);
void stopMove();
boolean isMoving();
  }

  // 具体实现
  public class Player implements Movable {
     public void moveTo(long x, long y) {
      this.position = new Transform(x, y);
     }

     public void startMove(long velocityX, long velocityY) {
     this.velocity = new Vector(velocityX, velocityY);
     }

     public void stopMove() {
     this.velocity = Vector.ZERO;
     }

     @Override
     public boolean isMoving() {
     return this.velocity.getX() != 0 || this.velocity.getY() != 0;
        }
     }

     @Valuepublic class Transform {
     public static final Transform ORIGIN = new Transform(0, 0);
     long x;
     long y;}

     @Value
     public class Vector {
     public static final Vector ZERO = new Vector(0, 0);
     long x;
     long y;
        }
```

注意两点：

- Moveable 的接口没有 Setter。一个 Entity 的规则是不能直接变更其属性，必须通过 Entity 的方法去对内部状态做变更。这样能保证数据的一致性。
- 抽象 Movable 的好处是如同 ECS 一样，一些特别通用的行为（如在大地图里移动）可以通过统一的 System 代码去处理，避免了重复劳动。

装备行为

因为我们已经不会用 Player 的子类来决定什么样的 Weapon 可以装备，所以这段逻辑应该被拆分到一个单独的类里。这种类在 DDD 里被叫做领域服务（Domain Service）。

```
public interface EquipmentService {
  boolean canEquip(Player player, Weapon weapon);
}
```

在 DDD 里，一个 Entity 不应该直接参考另一个 Entity 或服务，也就是说以下的代码是错误的：

```
public class Player {
  @Autowired
  EquipmentService equipmentService; // BAD: 不可以直接依赖

  public void equip(Weapon weapon) {
      // ...
    }
  }
```

这里的问题是 Entity 只能保留自己的状态（或非聚合根的对象）。任何其他的对象，无论是否通过依赖注入的方式弄进来，都会破坏 Entity 的 Invariance，并且还难以单测。

正确的引用方式是通过方法参数引入（Double Dispatch）：

```
public class Player {

public void equip(Weapon weapon, EquipmentService equipmentService) {
 if (equipmentService.canEquip(this, weapon)) {
```

```
this.weaponId = weapon.getId();
  } else {
throw new IllegalArgumentException("Cannot Equip: " + weapon);
    }
  }
}
```

在这里，无论是 Weapon 还是 EquipmentService 都是通过方法参数传入，确保不会污染 Player 的自有状态。

Double Dispatch 是一个使用 Domain Service 经常会用到的方法，类似于调用反转。

然后在 EquipmentService 里实现相关的逻辑判断，这里我们用了另一个常用的 Strategy（或者叫 Policy）设计模式：

```
public class EquipmentServiceImpl implements EquipmentService {
private EquipmentManager equipmentManager;

  @Override
  public boolean canEquip(Player player, Weapon weapon) {
  return equipmentManager.canEquip(player, weapon);
    }
  }

  // 策略优先级管理
  public class EquipmentManager {
  private static final List<EquipmentPolicy> POLICIES = new ArrayList<>();
  static {
  POLICIES.add(new FighterEquipmentPolicy());
  POLICIES.add(new MageEquipmentPolicy());
  POLICIES.add(new DragoonEquipmentPolicy());
  POLICIES.add(new DefaultEquipmentPolicy());
   }

  public boolean canEquip(Player player, Weapon weapon) {
  for (EquipmentPolicy policy : POLICIES) {
  if (!policy.canApply(player, weapon)) {
  continue;
  }
  return policy.canEquip(player, weapon);
```

```
    }
  return false;
    }
}

// 策略案例
public class FighterEquipmentPolicy implements EquipmentPolicy {

@Override
public boolean canApply(Player player, Weapon weapon) {
return player.getPlayerClass() == PlayerClass.Fighter;
}

/**
* Fighter 能装备 Sword 和 Dagger
*/
@Override
public boolean canEquip(Player player, Weapon weapon) {
return weapon.getWeaponType() == WeaponType.Sword
|| weapon.getWeaponType() == WeaponType.Dagger;
    }
}

// 其他策略省略，见源码
```

这样设计的最大好处是未来的规则增加只需要添加新的 Policy 类，而不需要去改变原有的类。

攻击行为

在上文中曾经有提起过，到底应该是 Player.attack(Monster)还是 Monster.receiveDamage(Weapon, Player)？在 DDD 里，因为这个行为可能会影响到 Player、Monster 和 Weapon，所以属于跨实体的业务逻辑。在这种情况下需要通过一个第三方的领域服务（Domain Service）来完成。

```
public interface CombatService {
void performAttack(Player player, Monster monster);
}
```

```
public class CombatServiceImpl implements CombatService {
  private WeaponRepository weaponRepository;
  private DamageManager damageManager;

  @Override
  public void performAttack(Player player, Monster monster) {
  Weapon weapon = weaponRepository.find(player.getWeaponId());
  int damage = damageManager.calculateDamage(player, weapon, monster);
  if (damage > 0) {
  monster.takeDamage(damage); //  (Note 1) 在领域服务里变更 Monster
    }
   // 省略掉 Player 和 Weapon 可能受到的影响
    }
  }
```

同样的在这个案例里，可以通过 Strategy 设计模式来解决 damage 的计算问题：

```
// 策略优先级管理
    public class DamageManager {
    private static final List<DamagePolicy> POLICIES = new ArrayList<>();
    static {
    POLICIES.add(new DragoonPolicy());
    POLICIES.add(new DragonImmunityPolicy());
    POLICIES.add(new OrcResistancePolicy());
    POLICIES.add(new ElfResistancePolicy());
    POLICIES.add(new PhysicalDamagePolicy());
    POLICIES.add(new DefaultDamagePolicy());
    }

    public int calculateDamage(Player player, Weapon weapon, Monster monster) {
    for (DamagePolicy policy : POLICIES) {
    if (!policy.canApply(player, weapon, monster)) {
    continue;
        }
    return policy.calculateDamage(player, weapon, monster);
      }
    return 0;
        }
    }

    // 策略案例
    public class DragoonPolicy implements DamagePolicy {
    public int calculateDamage(Player player, Weapon weapon, Monster monster) {
```

```
    return weapon.getDamage() * 2;
    }
    @Override
    public boolean canApply(Player player, Weapon weapon, Monster monster) {
     return player.getPlayerClass() == PlayerClass.Dragoon &&
    monster.getMonsterClass() == MonsterClass.Dragon;
     }
    }
```

特别需要注意的是这里的 CombatService 领域服务和 3.2 的 EquipmentService 领域服务，虽然都是领域服务，但实质上有很大的差异。上文的 EquipmentService 更多的是提供只读策略，且只会影响单个对象，所以可以在 Player.equip 方法上通过参数注入。但是 CombatService 有可能会影响多个对象，所以不能直接通过参数注入的方式调用。

单元测试

```
@Test
  @DisplayName("Dragoon attack dragon doubles damage")
    public void testDragoonSpecial() {
    // Given
    Player dragoon = playerFactory.createPlayer(PlayerClass.Dragoon, "Dart");
    Weapon sword = weaponFactory.createWeaponFromPrototype(swordProto, "Soul Eater", 60);
 ((WeaponRepositoryMock)weaponRepository).cache(sword);
    dragoon.equip(sword, equipmentService);
    Monster dragon = monsterFactory.createMonster(MonsterClass.Dragon, 100);

    // When
    combatService.performAttack(dragoon, dragon);

    // Then
    assertThat(dragon.getHealth()).isEqualTo(Health.ZERO);
    assertThat(dragon.isAlive()).isFalse();}

    @Test
    @DisplayName("Orc should receive half damage from physical weapons")
    public void testFighterOrc() {
    // Given
    Player fighter = playerFactory.createPlayer(PlayerClass.Fighter, "MyFighter");
    Weapon sword = weaponFactory.createWeaponFromPrototype(swordProto, "My Sword");
((WeaponRepositoryMock)weaponRepository).cache(sword);
```

```
    fighter.equip(sword, equipmentService);
    Monster orc = monsterFactory.createMonster(MonsterClass.Orc, 100);

    // When
    combatService.performAttack(fighter, orc);

    // Then
    assertThat(orc.getHealth()).isEqualTo(Health.of(100 - 10 / 2));
      }
```

具体的代码比较简单，解释省略

移动系统

最后还有一种 Domain Service，通过组件化，我们其实可以实现 ECS 一样的 System，来降低一些重复性的代码：

```
public class MovementSystem {

  private static final long X_FENCE_MIN = -100;
  private static final long X_FENCE_MAX = 100;
  private static final long Y_FENCE_MIN = -100;
  private static final long Y_FENCE_MAX = 100;

  private List<Movable> entities = new ArrayList<>();

  public void register(Movable movable) {
  entities.add(movable);      }

  public void update() {
  for (Movable entity : entities) {
   if (!entity.isMoving()) {
  continue;
  }

  Transform old = entity.getPosition();
  Vector vel = entity.getVelocity();
  long newX = Math.max(Math.min(old.getX() + vel.getX(), X_FENCE_MAX), X_FENCE_MIN);
  long newY = Math.max(Math.min(old.getY() + vel.getY(), Y_FENCE_MAX), Y_FENCE_MIN);
      entity.moveTo(newX, newY);    }}}
```

单测：

```
@Test
@DisplayName("Moving player and monster at the same time")
public void testMovement() {
  // Given
  Player fighter = playerFactory.createPlayer(PlayerClass.Fighter, "MyFighter");
  fighter.moveTo(2, 5);
  fighter.startMove(1, 0);

    Monster orc = monsterFactory.createMonster(MonsterClass.Orc, 100);
    orc.moveTo(10, 5);
    orc.startMove(-1, 0);

    movementSystem.register(fighter);
    movementSystem.register(orc);

    // When
    movementSystem.update();

    // Then
    assertThat(fighter.getPosition().getX()).isEqualTo(2 + 1);
    assertThat(orc.getPosition().getX()).isEqualTo(10 - 1);}
```

在这里 MovementSystem 就是一个相对独立的 Domain Service，通过对 Movable 的组件化，实现了类似代码的集中化、以及一些通用依赖/配置的中心化（如 X、Y 边界等）。

DDD 领域层的一些设计规范

上面我主要针对同一个例子对比了 OOP、ECS 和 DDD 的 3 种实现，比较如下：

- 基于继承关系的 OOP 代码：OOP 的代码最好写，也最容易理解，所有的规则代码都写在对象里，但是当领域规则变得越来越复杂时，其结构会限制它的发展。新的规则有可能会导致代码的整体重构。

- 基于组件化的 ECS 代码：ECS 代码有最高的灵活性、可复用性、及性能，但极具弱化了实体类的内聚，所有的业务逻辑都写在了服务里，会导致业务的一致性无法保障，对商业系统会有较大的影响。
- 基于领域对象 + 领域服务的 DDD 架构：DDD 的规则其实最复杂，同时要考虑到实体类的内聚和保证不变性（Invariants），也要考虑跨对象规则代码的归属，甚至要考虑到具体领域服务的调用方式，理解成本比较高。

所以下面，我会尽量通过一些设计规范，来降低 DDD 领域层的设计成本。领域层里的 Value Object（Domain Primitive）设计规范请参考我之前的文章。

实体类（Entity）

大多数 DDD 架构的核心都是实体类，实体类包含了一个领域里的状态、以及对状态的直接操作。Entity 最重要的设计原则是保证实体的不变性（Invariants），也就是说要确保无论外部怎么操作，一个实体内部的属性都不能出现相互冲突，状态不一致的情况。所以几个设计原则如下：

创建即一致

在贫血模型里，通常见到的代码是一个模型通过手动 new 出来之后，由调用方一个参数一个参数的赋值，这就很容易产生遗漏，导致实体状态不一致。所以 DDD 里实体创建的方法有两种：

constructor 参数要包含所有必要属性，或者在 constructor 里有合理的默认值。

比如，账号的创建：

```
public class Account {
  private String accountNumber;
  private Long amount;}
```

```
@Test
public void test() {
    Account account = new Account();
    account.setAmount(100L);
    TransferService.transfer(account); // 报错了，因为 Account 缺少必要的 AccountNumber
      }
```

如果缺少一个强校验的 constructor，就无法保障创建的实体的一致性。所以需要增加一个强校验的 constructor：

```
public class Account {
  public Account(String accountNumber, Long amount) {
    assert StringUtils.isNotBlank(accountNumber);
    assert amount >= 0;
    this.accountNumber = accountNumber;
    this.amount = amount;
     }
    }

    @Test
    public void test() {
    Account account = new Account("123", 100L); // 确保对象的有效性
    }
```

使用 Factory 模式来降低调用方复杂度

另一种方法是通过 Factory 模式来创建对象，降低一些重复性的入参。比如：

```
public class WeaponFactory {
public Weapon createWeaponFromPrototype(WeaponPrototype proto, String newName) {

Weapon weapon = new Weapon(null, newName, proto.getWeaponType(), proto.getDamage(), pro
to.getDamageType());
return weapon;
  }
}
```

通过传入一个已经存在的 Prototype，可以快速的创建新的实体。还有一些其他的如 Builder 等设计模式就不一一指出了。

尽量避免 public setter

一个最容易导致不一致性的原因是实体暴露了 public 的 setter 方法，特别是 set 单一参数会导致状态不一致的情况。比如，一个订单可能包含订单状态（下单、已支付、已发货、已收货）、支付单、物流单等子实体，如果一个调用方能随意去 set 订单状态，就有可能导致订单状态和子实体匹配不上，导致业务流程走不通的情况。所以在实体里，需要通过行为方法来修改内部状态：

```
@Data @Setter(AccessLevel.PRIVATE) // 确保不生成
  public setterpublic class Order {
      private int status; // 0 - 创建，1 - 支付，2 - 发货，3 - 收货
      private Payment payment; // 支付单      private Shipping shipping; // 物流单

    public void pay(Long userId, Long amount) {
    if (status != 0) {
    throw new IllegalStateException();
    }
    this.status = 1;
    this.payment = new Payment(userId, amount);
      }

    public void ship(String trackingNumber) {
    if (status != 1) {
    throw new IllegalStateException();
      }
    this.status = 2;
    this.shipping = new Shipping(trackingNumber);
      }
    }
```

【建议】在有些简单场景里，有时候确实可以比较随意的设置一个值而不会导致不一致性，也建议将方法名重新写为比较"行为化"的命名，会增强其语意。比如 setPosition(x, y)可以叫做 moveTo(x, y)，setAddress 可以叫做 assignAddress 等。

通过聚合根保证主子实体的一致性

在稍微复杂一点的领域里，通常主实体会包含子实体，这时候主实体就需要起到聚合根的作用，即：

- 子实体不能单独存在，只能通过聚合根的方法获取到。任何外部的对象都不能直接保留子实体的引用。
- 子实体没有独立的 Repository，不可以单独保存和取出，必须要通过聚合根的 Repository 实例化。
- 子实体可以单独修改自身状态，但是多个子实体之间的状态一致性需要聚合根来保障。

常见的电商域中聚合的案例如主子订单模型、商品/SKU 模型、跨子订单优惠、跨店优惠模型等。很多聚合根和 Repository 的设计规范在我前面一篇关于 Repository 的文章中已经详细解释过，可以拿来参考。

不可以强依赖其他聚合根实体或领域服务

一个实体的原则是高内聚、低耦合，即一个实体类不能直接在内部直接依赖一个外部的实体或服务。这个原则和绝大多数 ORM 框架都有比较严重的冲突，所以是一个在开发过程中需要特别注意的。这个原则的必要原因包括：对外部对象的依赖性会直接导致实体无法被单测；以及一个实体无法保证外部实体变更后不会影响本实体的一致性和正确性。

所以，正确的对外部依赖的方法有两种：

1. 只保存外部实体的 ID：这里我再次强烈建议使用强类型的 ID 对象，而不是 Long 型 ID。强类型的 ID 对象不单单能自我包含验证代码，保证 ID 值的正确性，同时还能确保各种入参不会因为参数顺序变化而出 bug。具体可以参考我的 Domain Primitive 文章。
2. 针对于"无副作用"的外部依赖，通过方法入参的方式传入。比如上文中的 equip(Weapon，EquipmentService）方法。

如果方法对外部依赖有副作用，不能通过方法入参的方式，只能通过 Domain Service 解决，见下文。

任何实体的行为只能直接影响到本实体（和其子实体）

这个原则更多是一个确保代码可读性、可理解的原则，即任何实体的行为不能有“直接”的”副作用”，即直接修改其他的实体类。这么做的好处是代码读下来不会产生意外。

另一个遵守的原因是可以降低未知的变更的风险。在一个系统里一个实体对象的所有变更操作应该都是预期内的，如果一个实体能随意被外部直接修改的话，会增加代码 bug 的风险。

领域服务（Domain Service）

在上文讲到，领域服务其实也分很多种，在这里根据上文总结出来三种常见的：

单对象策略型

这种领域对象主要面向的是单个实体对象的变更，但涉及到多个领域对象或外部依赖的一些规则。在上文中，EquipmentService 即为此类：

- 变更的对象是 Player 的参数
- 读取的是 Player 和 Weapon 的数据，可能还包括从外部读取一些数据

在这种类型下，实体应该通过方法入参的方式传入这种领域服务，然后通过 Double Dispatch 来反转调用领域服务的方法，比如：

```
Player.equip(Weapon, EquipmentService) {
  EquipmentService.canEquip(this, Weapon);
}
```

为什么这种情况下不能先调用领域服务，再调用实体对象的方法，从而减少实体对领域服务的入参型依赖呢？比如，下面这个方法是错误的：

```
boolean canEquip = EquipmentService.canEquip(Player, Weapon);
if (canEquip) {
    Player.equip(Weapon); // ✕，这种方法不可行，因为这个方法有不一致的可能性
  }
```

其错误的主要原因是缺少了领域服务入参会导致方法有可能产生不一致的情况。

跨对象事务型

当一个行为会直接修改多个实体时，不能再通过单一实体的方法作处理，而必须直接使用领域服务的方法来做操作。在这里，领域服务更多的起到了跨对象事务的作用，确保多个实体的变更之间是有一致性的。

在上文里，虽然以下的代码虽然可以跑到通，但是是不建议的：

```
public class Player {
  void attack(Monster, CombatService) {
    CombatService.performAttack(this, Monster); // ✕，不要这么写，会导致副作用
  }
}
```

而我们真实调用应该直接调用 CombatService 的方法：

```
public void test() {
  //...
combatService.performAttack(mage, orc);
  }
```

这个原则也映射了 4.1.5 的原则，即 Player.attack 会直接影响到 Monster，但这个调用 Monster 又没有感知。

通用组件型

这种类型的领域服务更像 ECS 里的 System，提供了组件化的行为，但本身又不直接绑死在一种实体类上。具体案例可以参考上文中的 MovementSystem 实现。

策略对象（Domain Policy）

Policy 或者 Strategy 设计模式是一个通用的设计模式，但是在 DDD 架构中会经常出现，其核心就是封装领域规则。

一个 Policy 是一个无状态的单例对象，通常需要至少 2 个方法：canApply 和 一个业务方法。其中，canApply 方法用来判断一个 Policy 是否适用于当前的上下文，如果适用则调用方会去触发业务方法。通常，为了降低一个 Policy 的可测试性和复杂度，Policy 不应该直接操作对象，而是通过返回计算后的值，在 Domain Service 里对对象进行操作。

在上文案例里，DamagePolicy 只负责计算应该受到的伤害，而不是直接对 Monster 造成伤害。这样除了可测试外，还为未来的多 Policy 叠加计算做了准备。

除了本文里静态注入多个 Policy 以及手动排优先级之外，在日常开发中经常能见到通过 Java 的 SPI 机制或类 SPI 机制注册 Policy，以及通过不同的 Priority 方案对 Policy 进行排序，在这里就不作太多的展开了。

加餐 - 副作用的处理方法 - 领域事件

在上文中，有一种类型的领域规则被我刻意忽略了，那就是"副作用"。一般的副作用发生在核心领域模型状态变更后，同步或者异步对另一个对象的影响或行为。在这个案例里，我们可以增加一个副作用规则：

- 当 Monster 的生命值降为 0 后，给 Player 奖励经验值

这种问题有很多种解法，比如直接把副作用写在 CombatService 里：

```
public class CombatService {
  public void performAttack(Player player, Monster monster) {
    // ...
  monster.takeDamage(damage);
  if (!monster.isAlive()) {
  player.receiveExp(10); // 收到经验
      }
    }
  }
```

但是这样写的问题是：很快 CombatService 的代码就会变得很复杂，比如我们再加一个副作用：

- 当 Player 的 exp 达到 100 时，升一级

这时我们的代码就会变成：

```
public class CombatService {
  public void performAttack(Player player, Monster monster) {
     // ...
    monster.takeDamage(damage);
    if (!monster.isAlive()) {
    player.receiveExp(10); // 收到经验
    if (player.canLevelUp()) {
    player.levelUp(); // 升级
          }
        }
      }
    }
```

如果再加上"升级后奖励 XXX"呢？"更新 XXX 排行"呢？依此类推，后续这种代码将无法维护。所以我们需要介绍一下领域层最后一个概念：领域事件（Domain Event）。

领域事件介绍

领域事件是一个在领域里发生了某些事后，希望领域里其他对象能够感知到的通知机制。在上面的案例里，代码之所以会越来越复杂，其根本的原因是反应代码（比如升级）直接和上

面的事件触发条件（比如收到经验）直接耦合，而且这种耦合性是隐性的。领域事件的好处就是将这种隐性的副作用"显性化"，通过一个显性的事件，将事件触发和事件处理解耦，最终起到代码更清晰、扩展性更好的目的。

所以，领域事件是在 DDD 里，比较推荐使用的跨实体"副作用"传播机制。

领域事件实现

和消息队列中间件不同的是，领域事件通常是立即执行的、在同一个进程内、可能是同步或异步。我们可以通过一个 EventBus 来实现进程内的通知机制，简单实现如下：

```
// 实现者：瑜进 2019/11/28
  public class EventBus {

  // 注册器
  @Getter
  private final EventRegistry invokerRegistry = new EventRegistry(this);

  // 事件分发器
  private final EventDispatcher dispatcher = new EventDispatcher(ExecutorFactory.getDirectExecuto
r());

  // 异步事件分发器
  private final EventDispatcher asyncDispatcher = new EventDispatcher(ExecutorFactory.getThreadP
oolExecutor());

  // 事件分发
  public boolean dispatch(Event event) {
  return dispatch(event, dispatcher);
    }

  // 异步事件分发
  public boolean dispatchAsync(Event event) {
  return dispatch(event, asyncDispatcher);
    }

  // 内部事件分发
  private boolean dispatch(Event event, EventDispatcher dispatcher) {
  checkEvent(event);
```

```
// 1.获取事件数组
Set<Invoker> invokers = invokerRegistry.getInvokers(event);
// 2.一个事件可以被监听 N 次，不关心调用结果
dispatcher.dispatch(event, invokers);
return true;
  }

// 事件总线注册
public void register(Object listener) {
 if (listener == null) {
throw new IllegalArgumentException("listener can not be null!");
    }
invokerRegistry.register(listener);
  }

private void checkEvent(Event event) {
if (event == null) {
throw new IllegalArgumentException("event");
  }
if (!(event instanceof Event)) {
throw new IllegalArgumentException("Event type must by " + Event.class);
    }
  }
}
```

调用方式：

```
public class LevelUpEvent implements Event {
  private Player player;
  }

    public class LevelUpHandler {
    public void handle(Player player);
    }

    public class Player {
    public void receiveExp(int value) {
    this.exp += value;
    if (this.exp >= 100) {
    LevelUpEvent event = new LevelUpEvent(this);
    EventBus.dispatch(event);
    this.exp = 0;
      }
```

```
    }
  }
  @Test
  public void test() {
  EventBus.register(new LevelUpHandler());
  player.setLevel(1);     player.receiveExp(100);
  assertThat(player.getLevel()).equals(2);
  }
```

目前领域事件的缺陷和展望

从上面代码可以看出来，领域事件的很好的实施依赖 EventBus、Dispatcher、Invoker 这些属于框架级别的支持。同时另一个问题是因为 Entity 不能直接依赖外部对象，所以 EventBus 目前只能是一个全局的 Singleton，而大家都应该知道全局 Singleton 对象很难被单测。这就容易导致 Entity 对象无法被很容易的被完整单测覆盖全。

另一种解法是侵入 Entity，对每个 Entity 增加一个 List:

```
public class Player {
  List<Event> events;

  public void receiveExp(int value) {
  this.exp += value;
  if (this.exp >= 100) {
    LevelUpEvent event = new LevelUpEvent(this);
    events.add(event); // 把 event 加进去
    this.exp = 0;
        }
      }
    }

    @Test
    public void test() {
    EventBus.register(new LevelUpHandler());
    player.setLevel(1);
    player.receiveExp(100);

    for(Event event: player.getEvents()) { // 在这里显性的 dispatch 事件
    EventBus.dispatch(event);
```

```
    }
assertThat(player.getLevel()).equals(2);
}
```

但是能看出来这种解法不但会侵入实体本身，同时也需要比较啰嗦的显性在调用方 dispatch 事件，也不是一个好的解决方案。

也许未来会有一个框架能让我们既不依赖全局 Singleton，也不需要显性去处理事件，但目前的方案基本都有或多或少的缺陷，大家在使用中可以注意。

总结

在真实的业务逻辑里，我们的领域模型或多或少的都有一定的"特殊性"，如果 100%的要符合 DDD 规范可能会比较累，所以最主要的是梳理一个对象行为的影响面，然后作出设计决策，即：

- 是仅影响单一对象还是多个对象
- 规则未来的拓展性、灵活性
- 性能要求
- 副作用的处理，等等

当然，很多时候一个好的设计是多种因素的取舍，需要大家有一定的积累，真正理解每个架构背后的逻辑和优缺点。一个好的架构师不是有一个正确答案，而是能从多个方案中选出一个最平衡的方案。

后端人精彩问答

Q: CDN 是什么？使用 CDN 有什么优势？

作者 | 淘系技术
出品 | 淘系技术

> 刚刚开始学习 Bootstrap，发现一些文章中总是提起使用 CDN 服务。
> 请问这个 CDN 服务是必须的吗？
> 我的系统都是在纯内网里面使用，与 Internet 不联通，还需要 CDN 吗？

淘宝的图片访问，有 98%的流量都走了 CDN 缓存。只有 2%会回源到源站，节省了大量的服务器资源。

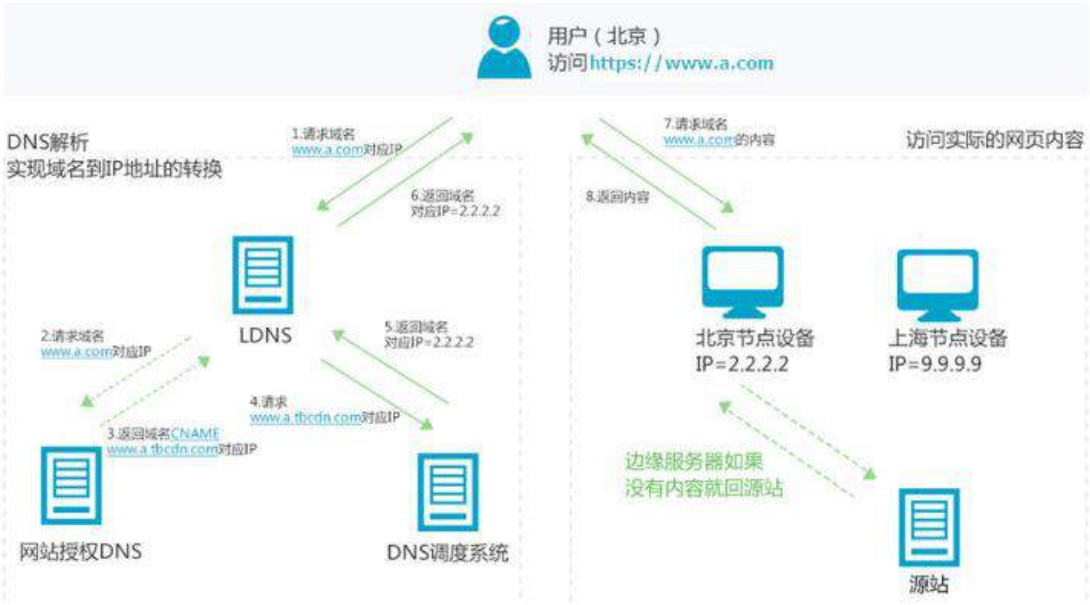
但是，如果在用户访问高峰期，图片内容大批量发生变化，大量用户的访问就会穿透 cdn，对源站造成巨大的压力。

结合阿里淘系 2020 年双 11 的淘宝实践，给大家分享下在图片业务里，我们是如何使用 CDN 以及如何解决挑战和困难。

CDN 工作原理

内容分发网络（Content Delivery Network，简称 CDN）是建立并覆盖在承载网之上，由分布在不同区域的边缘节点服务器群组成的分布式网络。

CDN 应用广泛，支持多种行业、多种场景内容加速，例如：图片小文件、大文件下载、视音频点播、直播流媒体、全站加速、安全加速。



借用阿里云官网的例子，来简单介绍 CDN 的工作原理。

假设通过 CDN 加速的域名为 www.a.com，接入 CDN 网络，开始使用加速服务后，当终端用户（北京）发起 HTTP 请求时，处理流程如下：

1. 当终端用户（北京）向 www.a.com 下的指定资源发起请求时，首先向 LDNS（本地 DNS）发起域名解析请求。
2. LDNS 检查缓存中是否有 www.a.com 的 IP 地址记录。如果有，则直接返回给终端用户；如果没有，则向授权 DNS 查询。
3. 当授权 DNS 解析 www.a.com 时，返回域名 CNAME www.a.tbcdn.com 对应 IP 地址。
4. 域名解析请求发送至阿里云 DNS 调度系统，并为请求分配最佳节点 IP 地址。
5. LDNS 获取 DNS 返回的解析 IP 地址。
6. 用户获取解析 IP 地址。
7. 用户向获取的 IP 地址发起对该资源的访问请求。

- 如果该 IP 地址对应的节点已缓存该资源，则会将数据直接返回给用户，例如，图中步骤 7 和 8，请求结束。

- 如果该 IP 地址对应的节点未缓存该资源，则节点向源站发起对该资源的请求。获取资源后，结合用户自定义配置的缓存策略，将资源缓存至节点，例如，图中的北京节点，并返回给用户，请求结束。

从这个例子可以了解到：

1. CDN 的加速资源是跟域名绑定的。
2. 通过域名访问资源，首先是通过 DNS 分查找离用户最近的 CDN 节点（边缘服务器）的 IP。
3. 通过 IP 访问实际资源时，如果 CDN 上并没有缓存资源，则会到源站请求资源，并缓存到 CDN 节点上，这样，用户下一次访问时，该 CDN 节点就会有对应资源的缓存了。

淘宝鹿班图片业务背景

商品的主图贯穿整个导购和交易链路，相比文字，图片更能吸引眼球，主图对消费者的购物决策有很大的影响。主图上表达的内容各式各样，但其中一定少不了的一定是价格的表达。

长期以来，主图上的价格表达都是商家自己维护，商品价格发生变化后，手动去换图。这样做，会带来 3 个问题：

1. 价格的准确性：商家手动填写的图片价格，跟实际的购买价可能不一致，造成不好的用户体验。
2. 价格更新的及时性：有时候，由于优惠券/品类券的生效失效，会导致商品的价格变化会很频繁，商家根本来不及换图。
3. 商家的操作成本：手动修改图片的价格，成本还是很高的，需要通过 ps 等软件修改图片，重新上传，编辑商品。

今年双 11，我们淘宝鹿班团队，试图通过技术手段来解决这些问题。当商品价格发生变化后，系统自动计算新的价格，自动合成图片，然后更新商品主图。

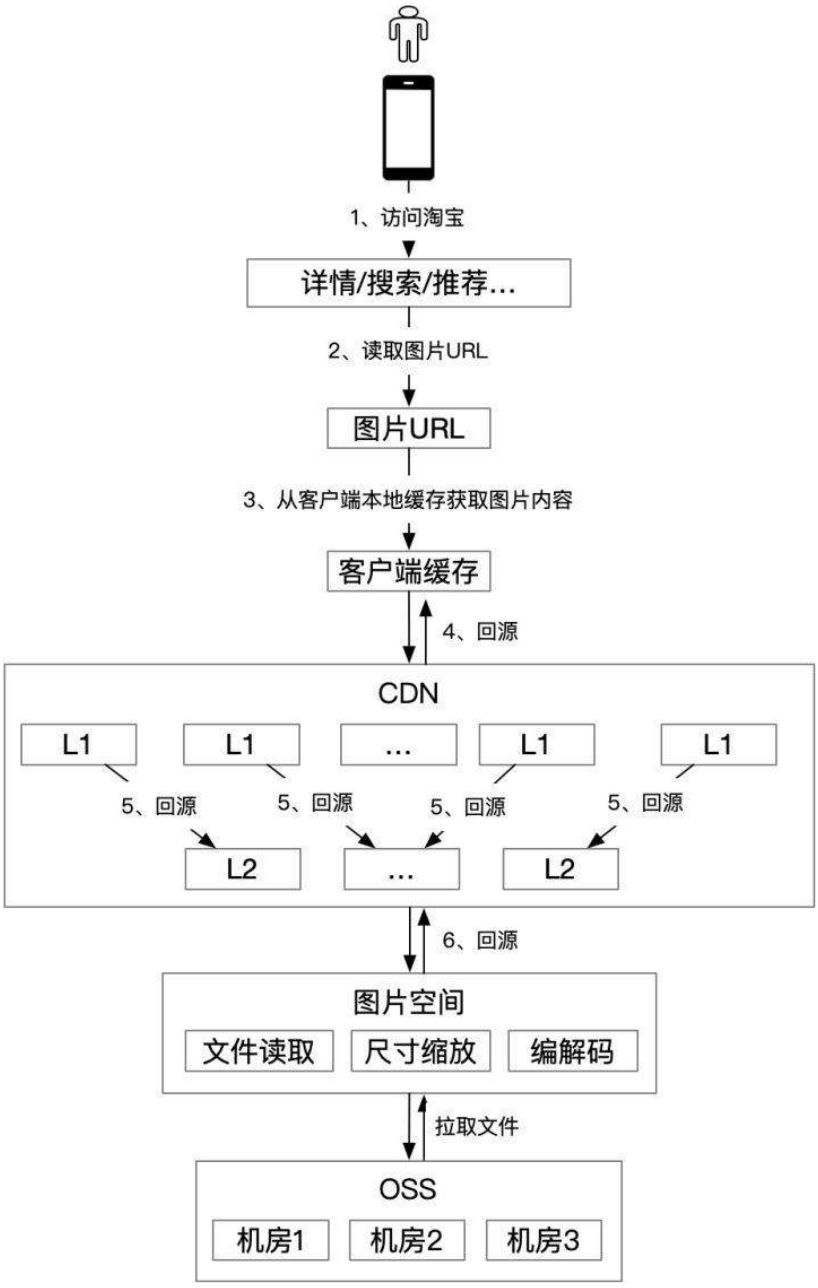
我们知道，淘宝网有上亿的商品，光大促商品就有几千万，因此，价格变化导致的图片变化频率非常高。最高的就是在双 11 的 0 点，全部大促商品的价格都会由日常价格变成大促价格。

这就意味着，大促高峰期，有上千万的图片刚生成就会被用户访问。那这个情况会产生什么问题呢，让我们先了解下淘宝的图片空间和 CDN 的架构，就清楚了。

淘宝图片空间和 CDN 的架构

淘宝整个图片的访问链路有三级缓存（客户端本地、CDN L1、CDN L2），所有图片都持久化的存储到 OSS 中。真正处理图片的是 img-picasso 系统，它的功能比较复杂，包括从 OSS 读取文件，对图片尺寸进行缩放，编解码，所以机器成本比较高。

CDN 的缓存分成 2 级，合理的分配 L1 和 L2 的比例，一方面，可以通过一致性 hash 的手段，在同等资源的情况下，缓存更多内容，提升整体缓存命中率；另一方面，可以平衡计算和 O，IO，充分利用不同配置的机器的能力。

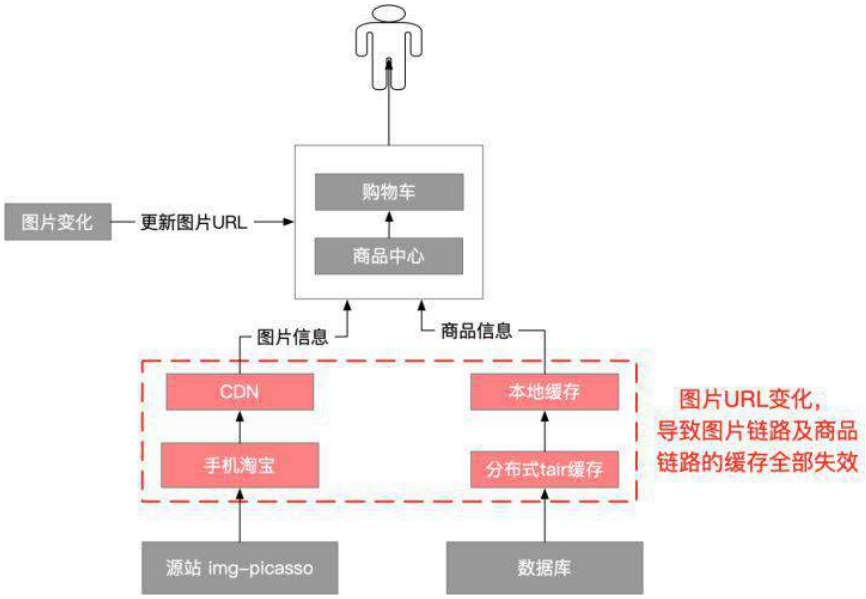


用户访问图片的过程如下：

1. 用户通过手机淘宝来搜索商品或者查看宝贝详情。
2. 详情/搜索/推荐通过调用商品中心返回商品的图片 URL。
3. 客户端本地如果有该图片的缓存，则直接渲染图片，否则执行下一步。
4. 从 CDN L1 回源图片，如果 L1 有该图片的缓存，则客户端渲染图片，同时缓存到本地，如果 L1 没有缓存，则执行下一步。

5. 从 CDN L2 回源图片，如果 L2 有该图片的缓存，则客户端渲染图片，同时 CDN L1 及客户端缓存图片内容，如果 CDN L2 没有缓存该图片，则执行下一步。
6. 从图片空间回源图片，图片空间会从 OSS 拉取图片源文件，按要求进行尺寸缩放，然后执行编解码，返回客户端能够支持的图片内容，之后客户端就可以渲染图片，同时 CDN 的 L1、L2 以及客户端都会缓存图片内容。

频繁换图带来的技术挑战



当商品的价格发生变化时，我们会使用新的价格重新合成图片，更新商品中心中存储的图片 URL。

这样会带来 2 个问题：

1. CDN 及手机淘宝原本缓存的图片内容失效，用户访问图片会全部回源到 img-picasso。
2. 由于更改了商品的字段，交易的核心应用（购物车和商品中心）的缓存也失效了，用户浏览及购物时，对商品的访问会走到 db。

源站 img-picasso 处理图片，以及查询商品 DB，都是非常消耗资源的。CDN 及商品的缓存命中率降低后，对源站 img-picsasso 以及 db 会产生巨大的压力。

拿 CDN 缓存为例，简单计算一下，CDN 平时的命中率是 98%，假设命中率降低 1 个点，对源站的压力就会增加 1/3（原本承担 2%的流量，现在需要承担 3%的流量），意味着 img-picasso 需要扩容 1/3。如果全网一半的图片都同时变化，cdn 的命中率降到 50%，对 img-picasso 的访问量就会增加 25 倍，这个扩容成本肯定没法接受。

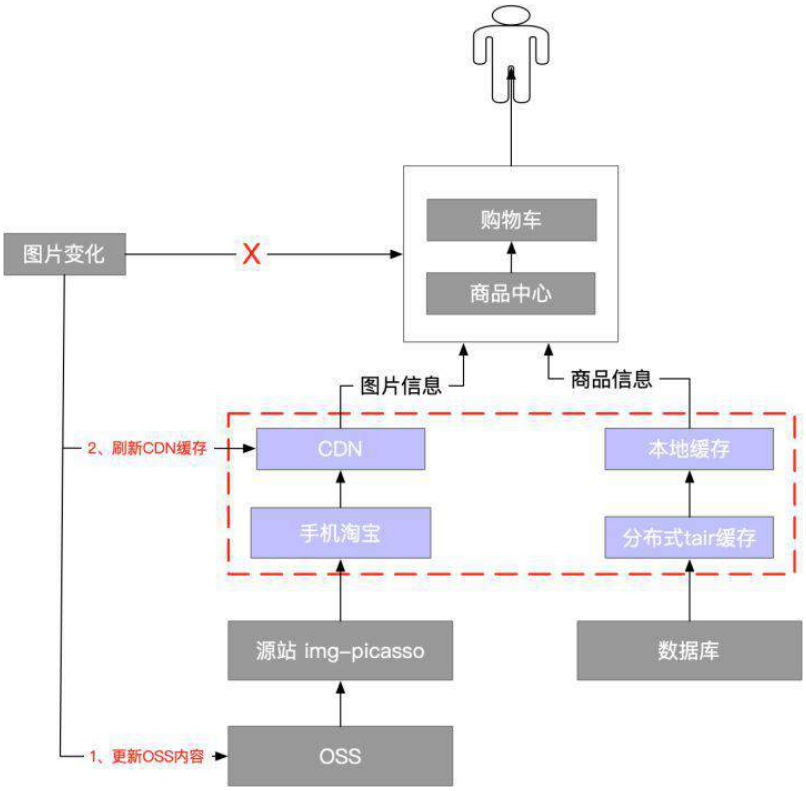
解决这 2 个问题，对应的有 2 个办法：

1. 改图保持图片 URL 不变，可以避免商品链路的缓存失效。
2. 在访问高峰到来之前，提前预热图片到 CDN，可以避免 CDN 缓存失效对源站的压力。

下面，介绍下我们具体是怎么做到这 2 点的。

频繁换图的应对方案

改图保持图片 URL 不变



图片内容发生变化时，执行下面 2 个操作：

1. 更新 OSS 内容：使用新的图片内容替换 OSS 中老的图片内容
2. 刷新 CDN 缓存：清除 CDN 之前缓存的图片内容

这样，用户再次访问图片时，发现 CDN 没有缓存，就会回源到 img-picasso，从 OSS 拉取新的图片内容。

由于图片 URL 没有变化，就不必去更新商品中心的图片链接，这样商品链路的缓存可以保持不变。

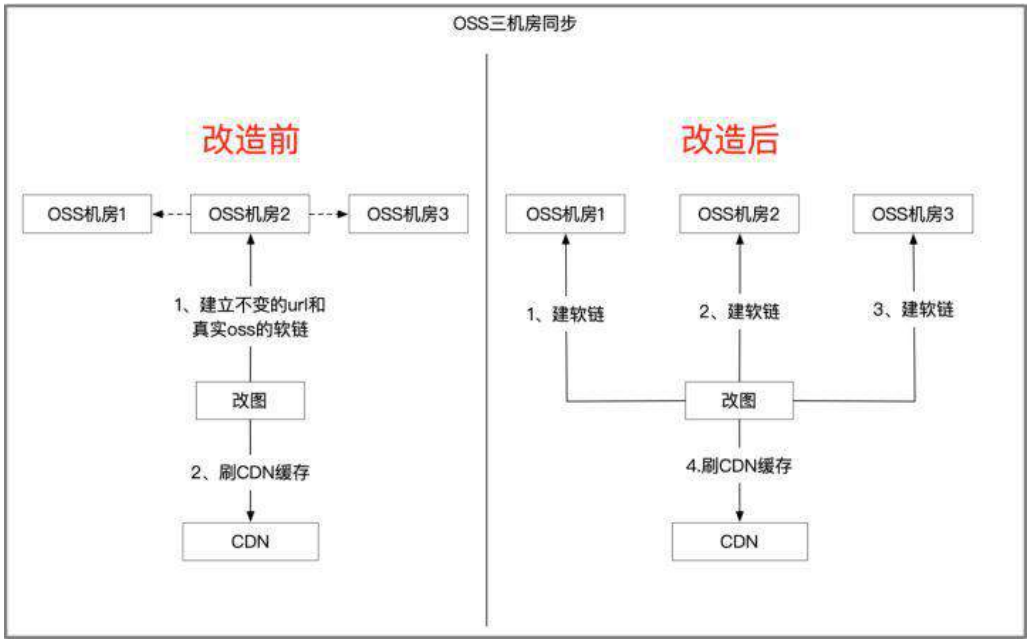
在真正实施这个方案的过程中，遇到了几个问题，简单跟大家分享下：

OSS 三地同步

淘宝的图片空间，承载了淘系所有图片的上下行稳定性保障，为了保障高可用，一份资源会存储到三地 OSS。图片上传时，默认只上传一地，利用 OSS 的能力，自动同步到另外两地。

但是使用 URL 不变方案，CDN 缓存已经清除完成后，如果另外 2 地的 OSS 还未同步完成，用户访问后，就会回源到旧的图片内容，发现图片内容没有变化。

针对该问题，我们将异步同步 OSS 软链的模式，改成三地同步建软链，三地都返回成功后，再去清除 CDN 缓存，这就保证了用户访问的图片一定是最新的内容。



图片尺寸收敛

同一张商品图片会用于不同的场景坑位展现，不同的坑位对图片的尺寸有不同的要求。为此，图片空间提供了一项功能，可以方便的生成不同尺寸的缩率图。只需要访问图片时，给图片增加不同的后缀，img-picasso 源站就可以按要求进行图片进行缩放。

由于历史原因，之前对缩放的尺寸种类没有限制，导致 CDN 上的图片后缀格式多达 2400 种+，TOP6 格式覆盖率 46%，TOP15 格式覆盖率 64%。这意味着，一张图片，在 cdn 上最多可能有 2400+个不同的 url，当图片内容变化后，要把这些缓存全部清掉，才能保证所有用户看到的图片都是新内容。

为了解决这个问题，我们对域名格式进行了收敛。

图片空间对于图片质量压缩参数的规则如下：

* 图片质量参数常见有一下 8 种形式：Q90、Q75、Q50、Q30、q90、q75、q50、q30
* 图片锐化参数常见有一下 3 种形式：s100，s150，s200

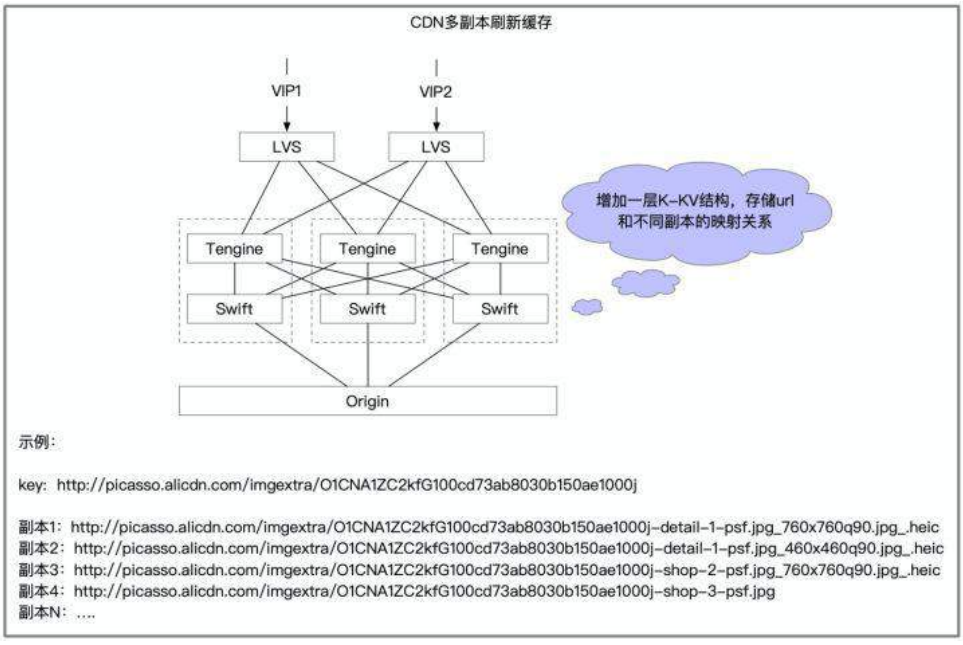
我们重新将图片质量定义为高质量图片和低质量图片，收敛格式为 q90 和 p50s150。

这样，就可以把 2000 多种格式收敛到 6 种主要格式，CDN 清除缓存才变得可行。

多副本清除 CDN 缓存

通过图片尺寸收敛，每张图片只需要清除 6 个不同的 url 就可以了，那能不能进一步提升刷新效率呢？

为此，阿里云 CDN 为我们提供了多副本刷新的解决方案：每种不同后缀的图片，作为图片的一个副本，在 CDN 的 swift 层增加一层 KV 结构，存储 url 和不同副本的映射关系，清除缓存时，可以通过该结构找到所有副本，实现快速清除所有副本。这样，每张图片，我们只需要调用一次 CDN 清除缓存接口就可以了，极大提升了 CDN 缓存刷新效率。



图片域名收敛

淘系的图片域名有 300 多种，主要有下面 2 个原因：

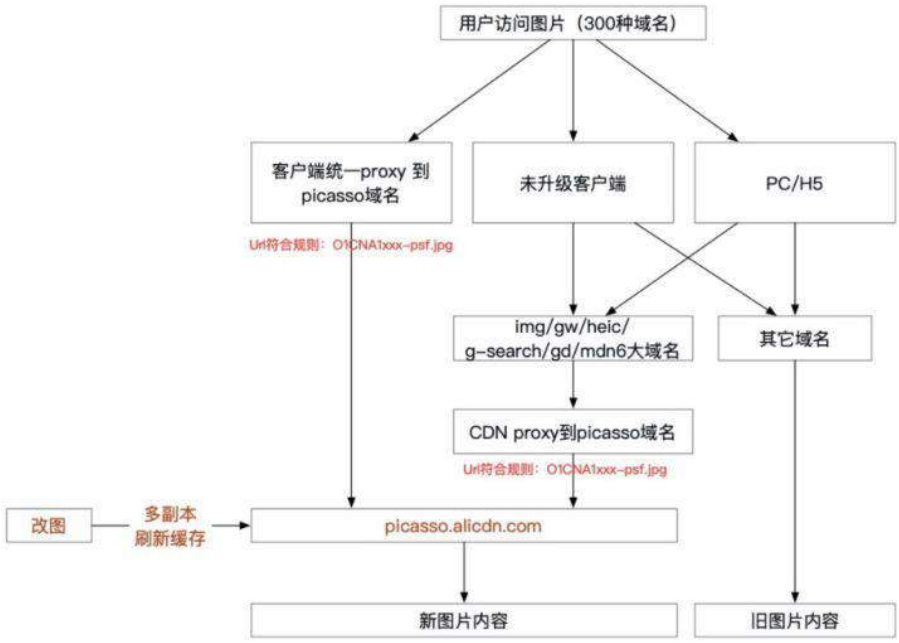
1. 图片完整的链接太长，所以存储时经常只存最后一段，业务自己来拼域名，很多业务就自己申请了一个图片域名来拼。
2. PC 时代，浏览器对同一域名下的并发请求数是有限制的，不同浏览器不一样，一般 6 个左右。

为了突破该限制，一些业务就会申请多个域名，随机的拼不同的域名。

前面我们讲过，CDN 的缓存是跟域名绑定的，不管是缓存命中还是缓存清除，都只能针对一个域名。

我们显然不可能改一张图，就去对 300 个域名调用 CDN 刷新。于是我们考虑对图片域名进行收敛，使得用户对图片的访问都路由到同一个域名，我们希望将所有的图片访问统一收敛到 http://picasso.alicdn.com，具体实现方式如下：

1. 对于手淘和猫客客户端，图片访问都收口在图片库，我们推进图片库进行改造，符合一定规则的 url，统一收敛到 http://picasso.alicdn.com，实现了域名的一刀切。
2. 对于 PC 浏览器端，就比较麻烦了，没有统一收口的地方。我们只能退而求其次，针对访问最多的 6 大域名，在 cdn 上配置域名转发规则，重定向到 picasso 域名。



通过这种方式，我们实现了全网 99%以上的图片访问流量都路由到 picasso 域名，图片内容发生变化时，通过清除 picasso 域名的 cdn 缓存，就能保证基本所有的场景都能看到新的图片内容。

客户端及浏览器缓存

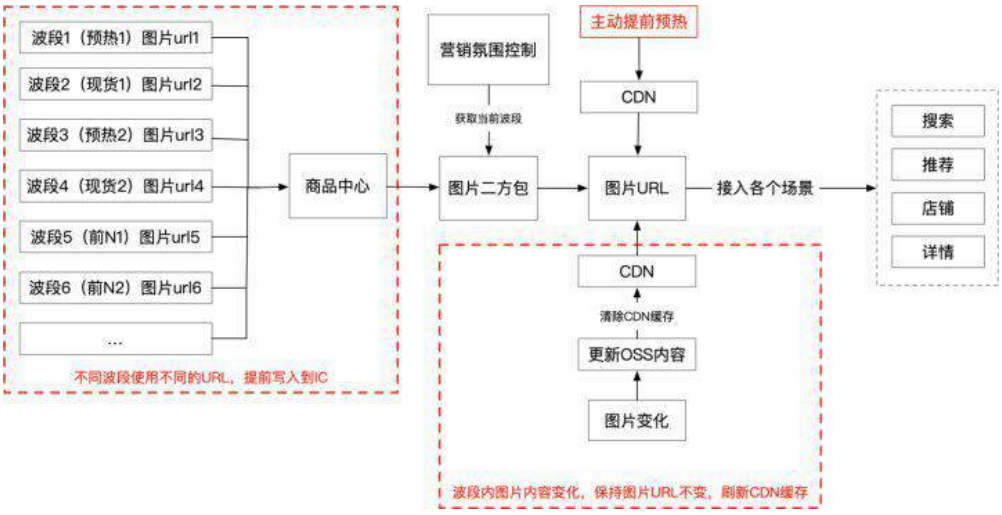
通过多副本和图片域名收敛，cdn 的缓存问题得到了解决。但在 cdn 之上，用户图片访问首先是来自客户端或者浏览器，这里也会有一层缓存。

大家知道，浏览器的缓存都遵循标准的 http max-age 协议，指定该 header 后，到了时间图片就会失效，访问到新的图片。所以我们可以在源站 img-picasso 回源给 cdn 时，添加 max-age 协议头，值为 1 分钟，cdn 会原封不动的透给浏览器，这样浏览器就可以实现 1 分钟内图片缓存失效，重新到 cdn 拉新的图片资源。

对于手机淘宝客户端，我们在原有的 LRU 缓存机制之上，另外支持标准的 http 协议。这样，手机淘宝也实现了 1 分钟内图片缓存失效。

提前预热 CDN 图片

通过改图保持图片 URL 不变，我们解决了改图对商品链路缓存的影响。但是，图片变化时，虽然 URL 没有变，但我们清除了 CDN 缓存，导致用户访问时还是会回源到 img-picasso 源站，所以对图片源站的压力依然存在。



我们发现，商品的价格变化大部分发生在大促节奏变化的时刻，基于这个特点，我们通过提前合成图片，提前预热到 CDN，可以实现图片切换瞬间生效，同时对源站没有压力。
具体方案如下：

1. 提前合成多波段图片：我们知道大促期间商家集中换图的时间点后，按这些时间点把图片的展示分成多个波段，每个波段图片提前合成，并提前将图片 URL 写入到商品中心扩展结构中。
2. 图片访问路由：营销系统根据配置的大促氛围切换计划，告诉鹿班图片二方包，当前是哪个波段，鹿班根据当前波段及场景，返回正确的图片 URL 给各个场景。
3. 图片渲染：各个场景拿到图片 URL 后，结合自身的业务逻辑，决定是否要展现该图片。
4. CDN 图片预热：为了避免图片集中切换时，把源站击垮，我们会在集中切换前把这些冷图片内容预热到 CDN。
5. 波段内图片变化：提前合成各个波段图片后，商家可能会临时发券/改价，导致商品价格再次变化，对于这类换图需求，为了避免更新商品中心的图片 URL，我们通过本文上一章节刷 CDN 缓存的方式实现。

总结和展望

CDN 技术广泛应用于互联网的各个场景，如今的 CDN 服务商，都提供了非常简单的业务接入方式，而且 CDN 的费用每年都在降低，这一切使得 CDN 的接入和使用成本越来越低。

本文通过淘宝图片业务的例子，为大家阐述了使用 CDN 过程中可能遇到的问题和解决思路。

淘宝的图片业务除了访问量大，还会面临更新频繁的问题。图片的频繁更新，一方面会由于商品上的图片 url 变化，导致商品缓存失效，另一方面会大幅降低 CDN 的图片访问缓存命中率。

针对图片 url 变化导致商品缓存失效的问题，我们通过刷新 cdn 缓存，用户访问时重新回源的方式，实现了改图保持图片 url 不变，这个过程中了，我们解决了一些列的问题，包括：OSS 三地同步更新、图片尺寸收敛、图片域名收敛、客户端及浏览器本地缓存。

针对改图降低 CDN 图片缓存命中率的问题，我们根据业务的特点，提前合成不同波段的图片，并预热到 CDN，保障了源站的安全。

目前，淘宝上用户看到的图片，都是提前合成好的。未来，我们考虑在用户访问图片时，实时合成图片。通过这项技术，可以实时感知业务更多的实时信息，可以根据这些信息，在图片上合成当前用户或者环境更匹配的文案/元素等内容，给用户带来更多的惊喜。

当然，实时合图也会面临更多的挑战，如：计算能力、合图性能。此外，对于 CDN 而言，由于每次用户访问的内容是临时合成的，CDN 的缓存策略也是一个很大的挑战。

Q: 后端真的有这么难吗?

作者 | 骨来
出品 | 淘系技术

目前大三，在学校学 JAVA 前端知识，数据库 数据结构 操作系统 计算机网络 ，想以后从事后端工作而身边一位好朋友说后端难学，想问是真的吗?

有的同学说后端难学，后端的技术发展越来越精细化和分层拆分，原理也晦涩难懂。
有的同学说后端除了增删改查，就没有其他亮点和价值了。

到底后端开发是什么样的技术，阿里巴巴淘系技术部高级技术专家-骨来，和大家分享一下自己对于后端技术的看法。

浅谈 CRUD

从 CRUD 的定义进行解读，CRUD 是指在做计算处理时的增加(Create)、读取(Retrieve)、更新(Update)和删除(Delete)几个单词的首字母简写，CRUD 主要被用在描述软件系统中数据库或者持久层的基本操作功能。

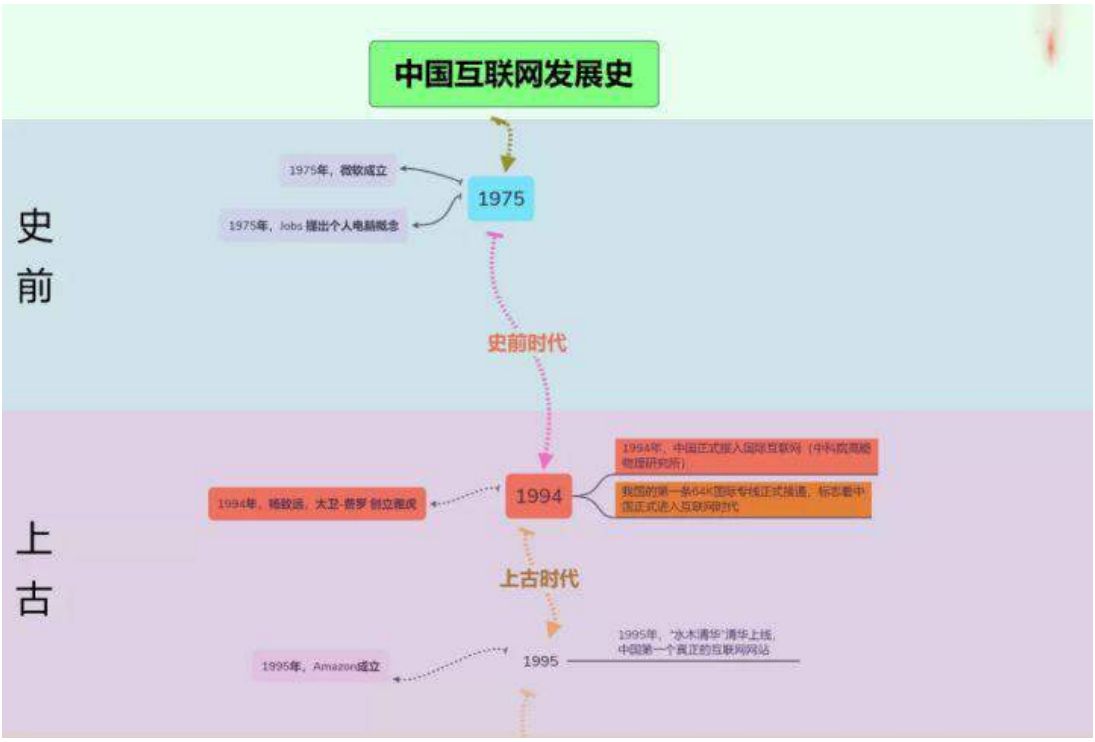
从软件分层架构的角度来看，这一层在业界被定义为 DAO(DataAccessObjects 数据存取对象)是指位于业务逻辑和持久化数据之间实现对持久化数据的访问。

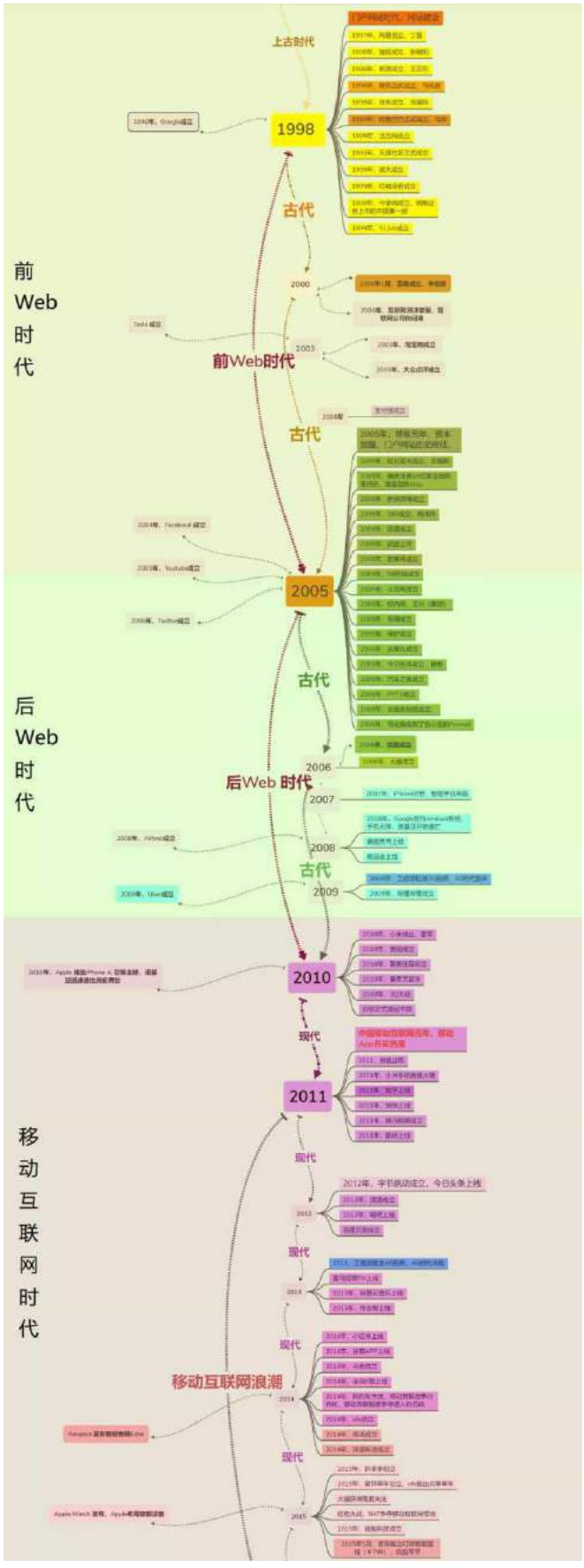
顾名思义，如果把 CRUD 等同于为后端开发技术栈内容，那么就是偷换概念和错误总结，从大部分后端工程师日常口头禅来看：在产品需求不明确或者不是特别有价值的情况下，常常对自己所做的事情或者需求，简单的定义为 CRUD，这是一种"自我调侃"和"对产品需求的轻视"。

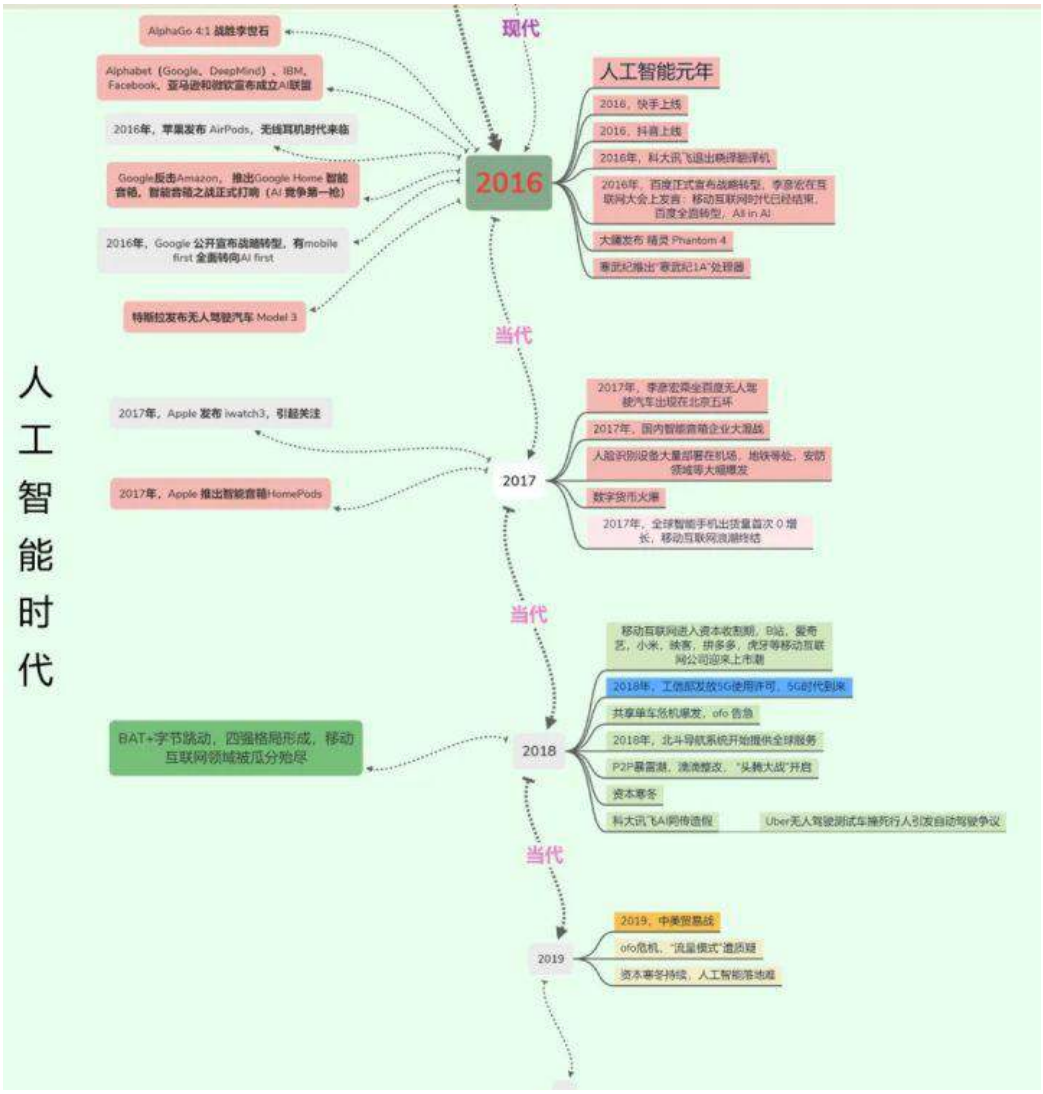
技术持续演进

回看计算机的发展史，从单机到网络计算机，从单体应用到分布式应用，从这 2 篇文章可以看到技术的飞速发展。

从 Web 时代到移动互联网时代再到人工智能时代，硬件的加速发展带来软件的飞速发展，无论是传统软件和互联网软件，软件岗位角色被一步步的细化，譬如前端、客户端、后端、算法、大数据等细分角色出现，每个角色面临的技术挑战和解决的问题域被重新定义和划分，职责更加清晰，能力范围更加专注和聚焦，技术栈更加清晰和明确，从这个划分来看后端除了 CRUD，还需要具备什么？




前Web时代
后Web时代
移动互联网时代
上古时代
古代
前Web时代
后Web时代
现代
移动互联网浪潮
1998
2000
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015



冰山下的后端领域

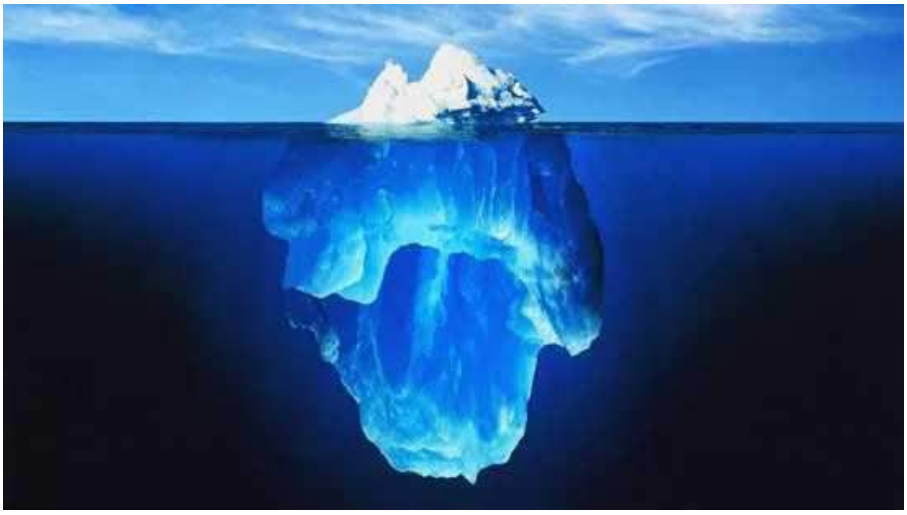
软件应用程序就像冰山一样，用户看到的只是应用程序的极小的部分，应用程序的最大部分是看不到的，这就是令人难以捉摸又神秘的"后端领域"。

作为后端人员，不仅仅停留在用户端侧体验以及交互形式，更重要的是从用户的数据模型、业务场景、业务模型、系统架构设计、领域建模、业务流程以及高并发、高可靠、规模化等稳定性角度，去提升产品整体的体验和价值，解决用户的诉求。

后端开发即"应用服务器后端"开发，主要涉及软件系统"后端"支撑。

比如，用于支撑 Web 网站和 App 客户端的服务器，同样介于后端服务器与浏览器以及 App 之间的应用中间件，它们都属于后端范畴。

简单地说，那些你在屏幕上看不到但又被用来为前端提供支持的数据和服务等基础服务以及对应的业务系统就是后端开发核心需要思考和解决的范畴。



后端开发职责

作为后端技术同学，需要针对技术栈进行完备的技能储备，并且需要从宏观技术广度和微观技术深度进行拓展，基于业务的场景和价值映射技术的方案和策略，从业务扩展性的角度进行架构设计，更好的保证可持续产品交付能力，从需求、产品、研发、质量、运维、运营等进行过程中精细化考量，站在用户的角度提升用户的体验，提升用户对产品的忠诚度，最终实现产品业务价值。

当下，后端技术发展日新月异，一句话："永远不要停止学习"。

以产品的生命周期作为研发的目标，从一而终直到产品下线，过程中各种产品的需求和问题，均需要后端开发持续性的投入和解决，以及通过引进新技术或技术创新，给产品带来进一步的增量价值。

举个例子，譬如现在非常热门的电商直播，涉及高清直播能力、人脸识别能力、直播间实物捕获能力、大规模实时互动能力、大型秒杀购物能力等，都需要前后端深度技术以此带来情景丰富性和实时互动性。

对于产品来说，需要保障 7x24 小时高可靠不停机服务，一切以满足用户的需求且高质量提供服务为遵旨，技术带来业务价值。

后端基础拆分

纵观近 10 年互联网发展，尤其是移动互联网开始兴起以后，亿级 DAU 用户产品大量涌现，一个单应用很难再垂直压榨硬件资源来提升性能，并且单点故障不能满足"停机不停服务

SLA"要求，需要做水平架构扩展支持集群部署以及单元化异地多活部署。

前台流量入口需要基于 DNS 服务做 VIP 路由负载均衡，并且基于 nginx 服务器做反向代理；不仅如此，还得考虑登陆状态 session 问题和 session 的一致性。

随着业务场景和业务流程越来越复杂，我们不得不把某些业务逻辑放到一个集群上，把另外一部分业务放到另外一个机集群上，"微服务架构理念"就此产出。

显然系统的计算能力和处理能力大大增强，但是这些系统之间的通信就变成了头疼的问题，消息队列和 RPC 框架因此出现，为了提高通信效率，各种序列化的工具等解决数据压缩和效率问题。

当海量大数据出现的时候，单个数据库也撑不住了，开始做数据库的读写分离，如果还不行，就做分库和分表，因为数据要做跨库跨表查询和分布式事务，为了解决这个问题，又有各种各样存储方案和存储技术出现。

另外，随着 VM 技术和 Docker/k8s 技术出现,大规模部署成为可能，同样伴随单机内存大小限制以及集群共享内存诉求，缓存从应用集群剥离出去单独进行集群部署提供高质量的分布式缓存服务，同样伴随 servicemesh 等应用技术飞速发展。

与此同时多媒体技术越来越互动性和丰富性，基于强大的手机终端技术，随时随地拍摄照片和视频上传到云端，亿级用户在终端需要更快的下载和观看体验，基于 CDN 边缘节点缓存加速和 OSS 云存储能力诞生。

随着互联网尤其是移动互联网的快速增长以及海量信息获取的诉求，搜索引擎高速发展和应用。

近来，人工智能的飞速发展和丰富的应用场景，端到端的智能化进一步降低用户获取信息的成本和效率，转变用户主动搜索到被动接收获取有价值的信息，精准推荐帮助用户做决策。跟随 3G/4G 的飞速发展，端侧即时性和实时交互的要求，其中以直播互动以及即时消息

IM 等应用场景出现，用户实时交互和实时感知。基础设施尤其是网络通信的发展和终端设备处理能力增强，让海量的移动设备接入和端边缘计算接入成为现实。

从传统以太网，到无线局域网的 WIFI（IEEE 802.11 b/g/n/ac），到不断发展的广域网蜂窝网络（3GPP 的 GSM、GPRS、UMTS、LTE、LTE-Advanced……），甚至整个广域网在 IoT 领域兴起以及终端技术的演进（蓝牙、NFC），标志着设备的泛化连接和网络接入的丰富性爆发式增长，尤其是 IoT/5G 的时代的到来，技术带来的新局面和新场景伴随而生，未来可期。

后端业务拆分

聊完后端的基础中间件和移动中间件系列，后端面对业务系统应该怎么建设呢？

从技术视角去看，基础中间件已经被更多细分化和深度化，从业务视角看，首先需要掌握中间件的基本原理和应用场景，以及各个中间件的特性，针对业务场景进行选择和应用。
除了技术选型之外，作为业务后端角色需要重点聚焦在需求分析、系统架构设计、领域模型构建、产品交付质量等，包括近年来比较火的"大中台，小前台"架构理念，作为前台的一线业务会更敏捷，更快速适应瞬息万变的市场，中台将集合整个集团的运营数据能力、产品技术能力，对各前台业务形成强力支撑。

诚然，基于拆分后的每个子系统，后端同学同样需要聚焦到应用上下游依赖关系、接口 API /SPI 的设计、核心性能指标、业务指标和相应的流控策略、安全防控等，基于业务的需求设计出对应的领域模型和业务逻辑以及 SOP 操作流程等；

包含对设计模式的采用和框架的选型考量，以及对中间件（缓存、存储、文件、网络）等依赖选择，对整个的发布过程和代码质量的把控，最终完成发布上线和后期的运维过程，产品的持续交付和迭代需要进一步作出技术的判断和演进思考，推动技术演进和产品持续迭代，充分考虑技术 ROI 投入产出比，在业务需求下作出正确的技术判断，技术发展日新月异，将来更需要密切把握技术脉搏、又要兼具全局视野，做出关键判断和取舍，给用户带来价值。

加油后浪

互联网技术一直在变化，技术越来越更聚焦，后端的技术发展越来越精细化和分层拆分。

作为后端的同学，更需要关注业界的动态和方向，包括 QCon 技术峰会、架构峰会以及 github 和技术社区。

需要能够有敏锐的嗅觉和判断力，通过技术栈的组合和应用，作出正确判断和技术选型，提升产品能力和推动产品和技术演进，在业务飞速发展下持续交付高质量产品和缩减产品交付迭代周期，快速落地业务和解决用户需求。

后端开发需要具备全情投入，能够做到真正地掌握背后技术原理同时做到触类旁通，能够把各个技术栈转化为自己的理解，再也不会以一句"后端 CRUD"作为简单定论了。

未来在 5G 时代，更需要技术的持续革新，新的业务场景和产品必将出现，需要我们做好技术储备和技术创新，应对新的变化和挑战，加油，奔涌中的后浪们！

最后，愿我们能与时代同行，具备技术前瞻性和创新变革，携手共进！

算法篇

淘系技术有大量的算法应用场景，从这些场景中来，本书覆盖了内容和商品推荐、多模态内容理解、用户交互和工程结合的算法应用。淘系技术的工程师们正在用崭新的人工智能方法，重新定义和解决问题，突破原来的问题领域，实现更大的技术目标。这些解决问题的新方法、新思路，结合解决问题过程中一手的工作经验，理论结合实践，值得算法从业者和对算法感兴趣的读者参考。

——阿里巴巴资深算法专家 乐田

技术经典总结

如何使用深度学习识别 UI 界面组件？

作者 | 常艳芳（苏川）

出品 | 淘系技术

导读：智能生成代码平台 imgcook 以 Sketch、PSD、静态图片等形式的视觉稿作为输入，可以一键生成可维护的前端代码，但从设计稿中获取的都是 div、img、span 等元件，而前端大多是组件化开发，我们希望能从设计稿直接生成组件化的代码，这就需要能够将设计稿中的组件化元素，例如 Searchbar、Button、Tab 等识别出来。识别网页中的 UI 元素在人工智能领域是一个典型的的目标检测问题，我们可以尝试使用深度学

应用背景

imgcook 以 Sketch、PSD、静态图片等形式的视觉稿作为输入，通过智能化技术一键生成可维护的前端代码，Sketch/Photoshop 设计稿的代码生成需要安装插件，在设计稿中通过 imgcook 插件导出视觉稿的 JSON 描述信息（D2C Schema）粘贴到 imgcook 可视化编辑器，在编辑器中可以进行视图编辑、逻辑编辑等来改变 JSON 描述信息。

我们可以选择 DSL 规范来生成对应的代码。例如生成 React 规范的代码，需要实现从 JSON 树转换成 React 代码 （自定义 DSL）。

如下图，左侧为 Sketch 中的视觉稿，右侧为使用 React 开发规范生成的按钮部分的代码。

从 Sketch 视觉稿「导出数据」生成「React 开发规范」的代码，图为按钮部分代码片段。

生成的代码都是由 div、img、span 这些标签组成，但实际应用开发有这样的问题：

- web 页面开发为提升可复用性，页面组件化，例如：Searchbar、Button、Tab、Switch、Stepper。
- 一些原生组件不需要生成代码，例如状态栏 Statusbar、Navbar、Keyboard。

我们的需求是，如果想要使用组件库，例如 Ant Design，我们希望生成的代码能像这样：

```
// Antd Mobile React 规范

import { Button } from "antd-mobile";

<div style={styles.ft}>
<Button style={styles.col1}>进店抢红包</Button>
<Button style={styles.col2}>加购物车</Button>
</div>
"smart":
{ "layerProtocol": {
 "component": {
"type": "Button"
}
}
  }
```

为此我们在 JSON 描述中添加了 smart 字段，用来描述节点的类型。

我们需要做的，就是找到视觉稿中需要组件化的元素，用这样的 JSON 信息来描述它，以便在 DSL 转换代码时， 通过获取 JSON 信息中的 smart 字段来生成组件化代码。

现在问题转化为：如何找到视觉稿中需要组件化的元素，它是什么组件，它在 DOM 树中的位置或者在设计稿中的位置。

解决方案

约定生成规则

通过指定设计稿规范来干预生成的 JSON 描述，从而控制生成的代码结构。例如在我们的 设计稿高级干预规范 中关于组件的图层命名规范：将图层中的组件、组件属性等显性标记出来。

```
#component:组件名?属性=值#
#component:Button?id=btn#
```

在使用 imgcook 的插件导出 JSON 描述数据时就通过规范解析拿到图层中的约定信息。

学习识别组件

人工约定规则的方式需要按照我们制定的协议规范来修改设计稿，一个页面上的组件可能会有很多，这种人工约定方式让开发者多了很多额外工作，不符合使用 imgcook 提高开发效率的宗旨，我们期望通过智能化手段自动识别视觉稿中的可组件化元素，识别的结果最终会转换并填充在 smart 字段中，与手动约定组件协议所生成的 json 中的 smart 字段内容相同。

这里需要完成两件事情：

- 找到组件信息：类别、位置、尺寸等信息。
- 找到组件中的属性， 例如 button 中的文字为"提交"。

第二个事情我们可以根据 json 树来解析组件的子元素。第一个事情我们可以通过智能化来自动化的完成，这是一个在人工智能领域典型的的目标检测问题，我们可以尝试使用深度学习目标检测手段来自动化解决这个手动约定的流程。

学习识别 UI 组件

业界现状

目前业界也有一些使用深度学习来识别网页中的 UI 元素的研究和应用，对此有一些讨论：

- Use R-CNN to detect UI elements in a webpage?
- Is machine learning suitable for detecting screen elements?
- Any thoughts on a good way to detect UI elements in a webpage?
- How can I detect elements of GUI using opencv?
- How to recognize UI elements in image?

讨论中的诉求主要有两种：

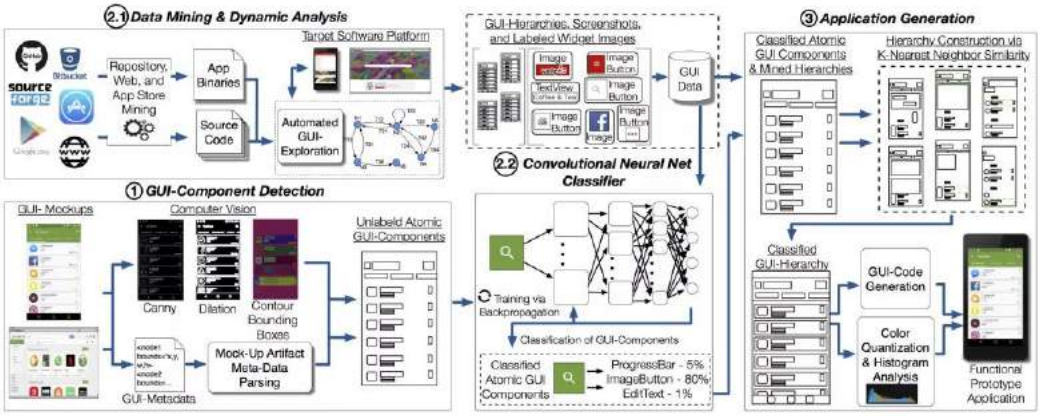
- 期望通过识别 UI 界面元素来做 Web 页面自动化测试的应用场景。
- 期望通过识别 UI 界面元素来自动生成代码。

既然是使用深度学习来解决 UI 界面元素识别的问题，带有元素信息的 UI 界面数据集则是必须的。目前业界开放且使用较多的数据集有 Rico 和 ReDraw。

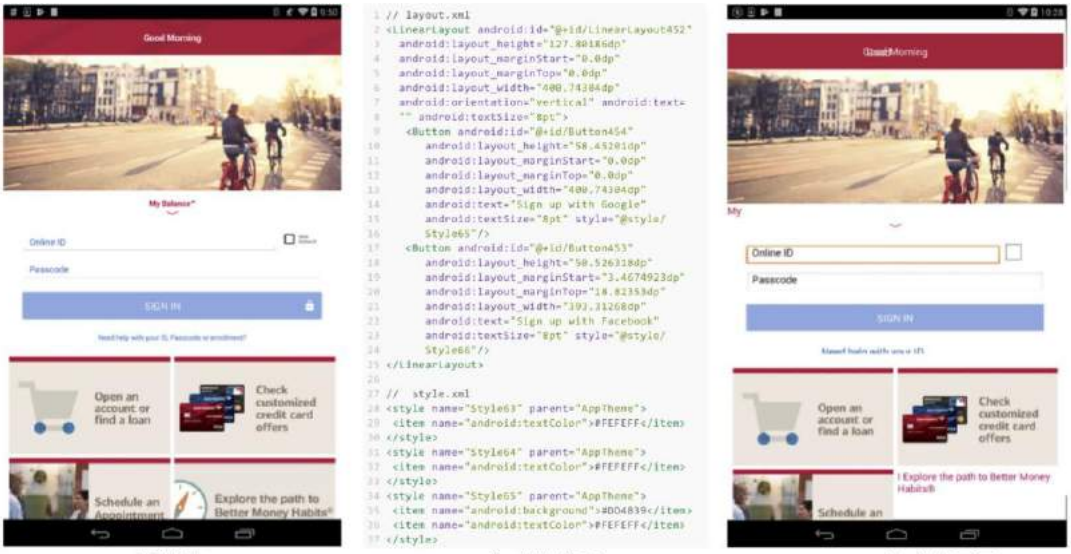
ReDraw

一组 Android 屏幕截图，GUI 元数据和标注了 GUI 组件图像，包含 RadioButton、ProgressBar、Switch、Button、CheckBox 等 15 个分类，14,382 个 UI 界面图片和 191,300 个带有标签的 GUI 组件，该数据集经过处理之后使每个组件的数量达到 5000 个。关于该数据集的详细介绍可查看 The ReDraw Dataset。这是用于训练和评估 ReDraw 论文中提到的 CNN 和 KNN 机器学习技术的数据集，该论文于 2018 年在 IEEE Transactions on Software Engineering 上发布。该论文提出了一种通过三个步骤来实现从 UI 转换为代码自动化完成的方法：

- **检测 Detection**
  先从设计稿中提取或使用 CV 技术提取 UI 界面元信息，例如边界框（位置、尺寸）。
- **分类 Classification**
  再使用大型软件仓库挖掘、自动动态分析得到 UI 界面中出现的组件，并用此数据作为 CNN 技术的数据集学习将提取出的元素分类为特定类型，例如 Radio、ProgressBar、Button 等。
- **组装 Assembly**
  最后使用 KNN 推导 UI 层次结构，例如纵向列表、横向 Slider。



在 ReDraw 系统中使用这种方法生成了 Android 代码。评估表明，ReDraw 的 GUI 组件分类平均精度达到 91%，并组装了原型应用程序，这些应用程序在视觉亲和力上紧密地反映了目标模型，同时展现了合理的代码结构。



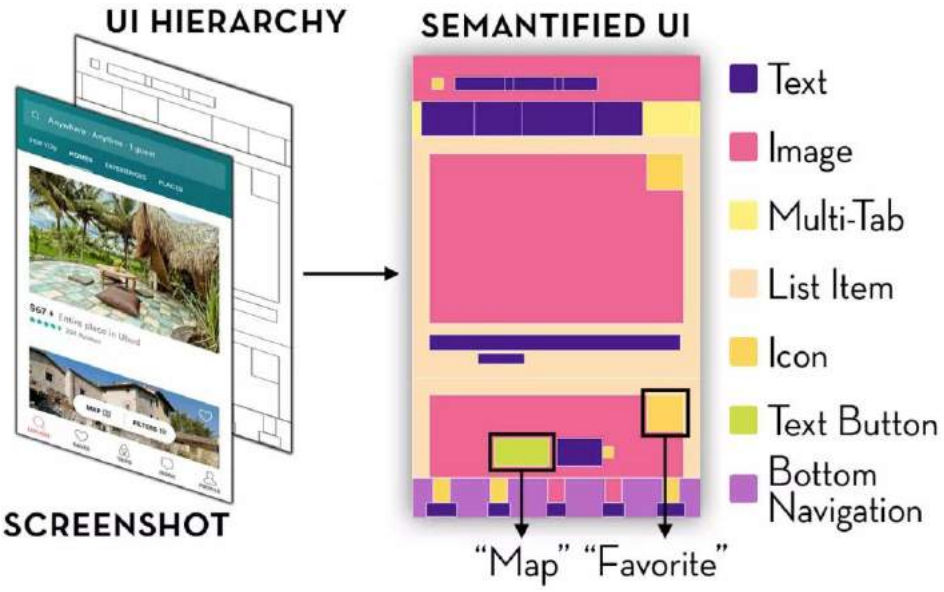
视觉稿　　生成的代码　　生成的页面

Rico

迄今为止最大的移动 UI 数据集，创建目的是支持五类数据驱动的应用程序：设计搜索，UI 布局生成，UI 代码生成，用户交互建模和用户感知预测。 Rico 数据集包含 27 个类别、1 万多个应用程序和大约 7 万个屏幕截图。该数据集在 2017 年第 30 届 ACM 年度用户界面软件和技术研讨会上对外开放（RICO: A Mobile App Dataset for Building Data-Driven Design Applications）。

此后有一些基于 Rico 数据集的研究和应用。例如： Learning Design Semantics for Mobile Apps， 该论文介绍了一种基于代码和视觉的方法给移动 UI 元素添加语义注释。根据 UI 屏幕截图和视图层次结构可自动识别出 25

UI 组件类别，197 个文本按钮概念和 99 个图标类。



应用场景

这里列举一些基于以上数据集的研究和应用场景。

智能生成代码

Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps | ReDraw Dataset

智能生成布局

Neural Design Network: Graphic Layout Generation with Constraints | Rico Dataset

用户感知预测

Modeling Mobile Interface Tappability Using Crowdsourcing andDeep Learning | Rico Dataset

UI 自动化测试

A Deep Learning based Approach to Automated Android App Testing | Rico Dataset

问题定义

在上述介绍的基于 Redraw 数据集生成 Android 代码的应用中，我们了解了它的实现方案， 对于第 2 步需要使用大型软件仓库挖掘和自动动态分析技术来获取大量组件样本作为 CNN 算法的训练样本，以此来得到 UI 界面中存在的特定类型组件，例如 ProgressBar、Switch 等。

对于我们 imgcook 的应用场景，其本质问题也是需要找到 UI 界面中这种特定类型的组件信息：类别和边界框，我们可以把这个问题定义为一个目标检测的问题，使用深度学习对 UI 界面进行目标检测。那么我们的目标是什么？

检测目标就是 Progress Bar、Switch、Tab Bar 这些可在代码中组件化的页面元素。

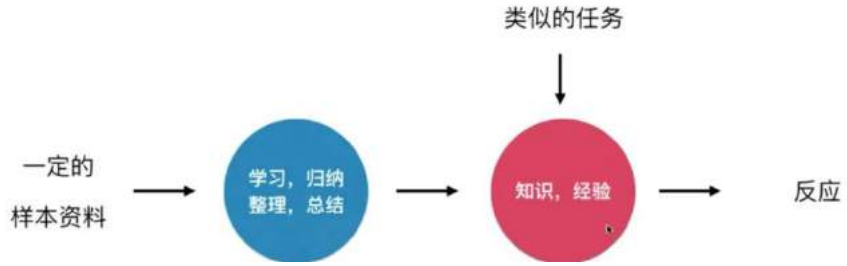
UI 界面目标检测

基础知识

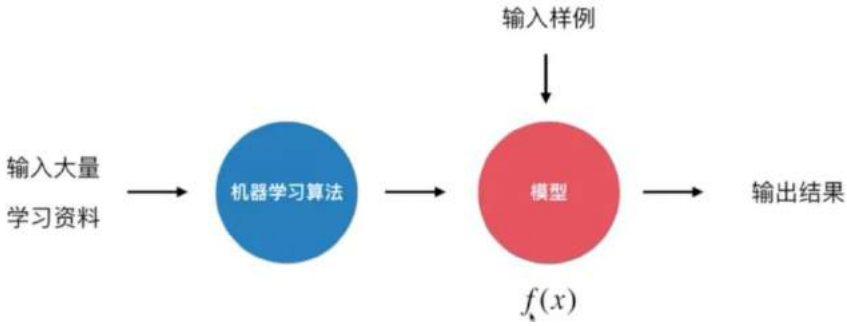
机器学习

人类是怎么学习的？通过给大脑输入一定的资料，经过学习总结得到知识和经验，有当类似的任务时可以根据已有的经验做出决定或行动。

机器学习(Machine Learning)的过程与人类学习的过程是很相似的。机器学习算法本质上就是获得一个 f(x) 函数表示的模型，如果输入一个样本 x 给 f(x) 得到的结果是一个类别，解决的就是一个分类问题，如果得到的是一个具体的数值那么解决的就是回归问题。



机器学习与人类学习的整体机制是一致的，有一点区别是人类的大脑只需要非常少的一些资料就可以归纳总结出适用性非常强的知识或者经验，例如我们只要见过几只猫或几只狗就能正确的分辨出猫和狗，但对于机器来说我们需要大量的学习资料，但机器能做到的是智能化不需要人类参与。



深度学习

深度学习（Deep Learning）是机器学习的分支，是一种试图使用包含复杂结构或由多重非线性变换构成的多个处理层对数据进行高层抽象的算法。

深度学习与传统机器学习的区别可以看这篇 Deep Learning vs. Machine Learning，有数据依赖性、硬件依赖、特征处理、问题解决方式、执行时间和可解释性这几个方面。

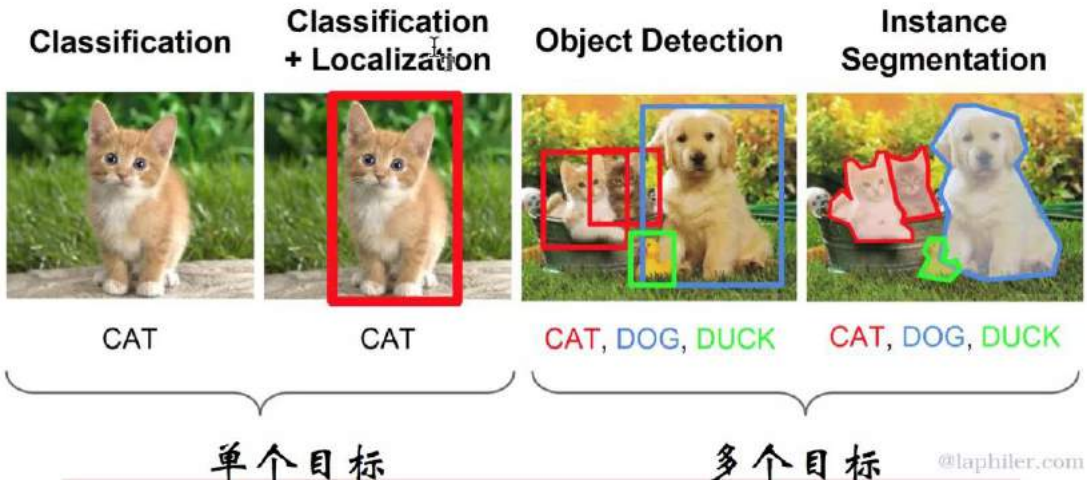
深度学习对数据量和硬件的要求很高且执行时间很长，深度学习与传统机器学习算法的主要不同在于对特征处理的方式。在传统机器学习用于现实任务时，描述样本的特征通常需要由人类专家来设计，这称为"特征工程"（Feature Engineering），而特征的好坏对泛化性能有至关重要的影响，要设计出好特征并非易事。深度学习可以通过特征学习技术分析数据自动产生好特征。

目标检测

机器学习有很多应用，例如：

- 计算机视觉（Computer Vision，CV） 用于车牌识别和面部识别等的应用。
- 信息检索 用于诸如搜索引擎的应用 - 包括文本搜索和图像搜索。
- 市场营销 针对自动电子邮件营销和目标群体识别等的应用。
- 医疗诊断 诸如癌症识别和异常检测等的应用。
- 自然语言处理（Natural Language Processing， NLP） 如情绪分析和照片标记等的应用。

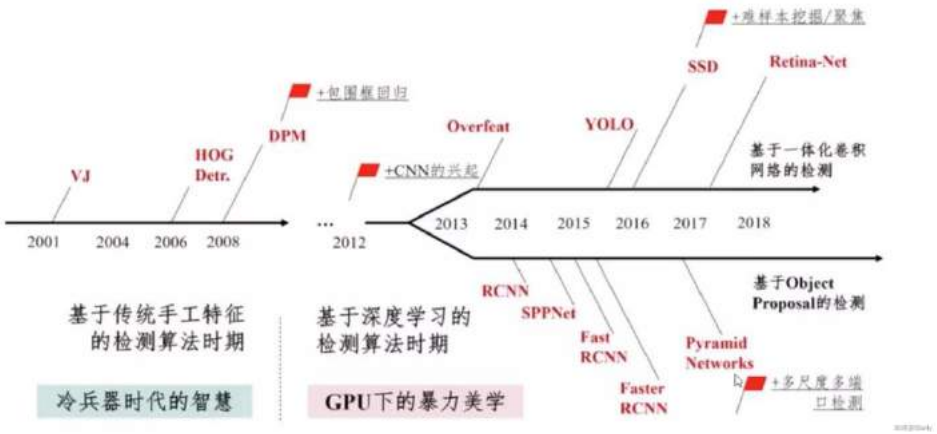
目标检测（Object Detection）就是一种与计算机视觉和图像处理有关的计算机技术，用于检测数字图像和视频中特定类别的语义对象（例如人，动物物或汽车）。



而我们在 UI 界面上的目标是一些设计元素， 可以是原子粒度的 Icon、Image、Text、或是可组件化的 Searchbar、Tabbar 等。

算法选型

用于目标检测的方法通常分为基于机器学习的方法（传统目标检测方法）或基于深度学习的方法（深度学习目标检测方法），目标检测方法经历了从传统目标检测方法到深度学习目标检测方法的变迁：



传统目标检测方法

对于基于机器学习的方法 ， 需要先使用以下方法之一来定义特征，然后使用诸如支持向量机（SVM）的技术进行分类。

- 基于 Haar 功能的 Viola–Jones 目标检测框架
- 尺度不变特征变换（SIFT）
- 定向梯度直方图（HOG）特征

深度学习目标检测方法

对于基于深度学习的方法，无需定义特征即可进行端到端目标检测，通常基于卷积神经网络（CNN）。基于深度学习的目标检测方法可分为 One-stage 和 Two-stage 两类，还有继承这两种类方法优点的 RefineDet 算法。

One-stage

基于 One-stage 的目标检测算法直接通过主干网络给出类别和位置信息，没有使用 RPN 网路。这样的算法速度更快，但是精度相对 Two-stage 目标检测网络了略低。典型算法有：

- SSD（Single Shot MultiBox Detector）系列
- YOLO (You Only Look Once)系列（YOLOv1、YOLOv2、YOLOv3）
- RetinaNet

Two-stage

基于 Two-stage 的目标检测算法主要通过一个卷积神经网络来完成目标检测过程，其提取的是 CNN 卷积特征，在训练网络时，其主要训练两个部分，第一步是训练 RPN 网络，第二步是训练目标区域检测的网络。即先由算法生成一系列作为样本的候选框，再通过卷积神经网络进行样本分类。网络的准确度高、速度相对 One-stage 慢。典型算法有：

- R-CNN，Fast R-CNN，Faster R-CNN

其他（RefineDet）

RefineDet (Single-Shot Refinement Neural Network for Object Detection) 是基于 SSD 算法的改进。继承了两种方法（例如，单一阶段设计方法，两阶段设计方法）的优点，并克服了它们的缺点。

目标检测方法对比

传统方法 VS 深度学习

基于机器学习的方法和基于深度学习的方法的算法流程如图所示，传统目标检测方法需要手动设计特征，通过滑动窗口获取候选框，再使用传统分类器进行目标区域判定，整个训练过程分成多个步骤。而深度学习目标检测方法则通过机器学习特征，通过更高效的 Proposal 或直接回归的方式获取候选目标，它的准确度和实时性更好。

关于目标检测算法的研究现在基本都是基于深度学习的，传统的目标检测算法已经很少用到了，深度学习目标检测方法更适合工程化， 具体对比如下：

比较项	传统目标检测方法	深度学习目标检测方法
获取特征的手段	手动设计特征	深度网络学习特征
获得候选目标手段	滑动窗口	Proposal 或者直接回归
目标区域判定(分类)手段	传统分类器	深度网络
训练过程	多步骤	端到端
效果	准确度和实时性差	准确度和实时性好

One-stage VS Two-stage

方法	优点	缺点
One-stage	速度快 避免背景错误产生 false positives 学到物体的泛化特征	精度低（定位、检出率） 小物体检测效果不好
Two-stage	精度高（定位、检出率） Anchor 机制 共享计算量	速度慢 训练时间长 误报高

算法优缺点

这里就不写各个算法的原理了，直接看下优缺点。

目标检测算法	优点	缺点
RCNN	最早使用 CNN 特征	多次使用 CNN 提取特征、耗时高
Fast RCNN	引入 ROI Pooling; bbox 回归 和 cls 回归	selective search 产生 region proposals
Faster RCNN	RPN 网络，学习产生 region proposals, 精度高	速度慢，训练时间长
SSD	Anchor 机制+回归; 边框定位准; 目标检出率高;误报比 YOLO 多	耗时相比 YOLO 高， 相比 Faster RCNN 低
YOLO	误报少，耗时少	直接回归，边框定位不够精准;目标检出率低，小目标难

总结

由于对 UI 界面元素检测的精度要求比较高， 所以最终选择 Faster RCNN 算法。

框架选择

机器学习框架

这里简要列举下几个机器学习框架：Scikit Learn、TensorFlow、Pytorch、Keras。

Scikit Learn 是通用的机器学习框架，内部实现了各种分类,回归和聚类算法（包括支持向量机，随机森林，梯度增强，k-means 等）；还包括数据降维、模型选择和数据预处理等工具库，容易安装和使用，样例丰富，而且教程和文档也非常详细。

TensorFlow、Keras 和 Pytorch 是目前深度学习的主要框架，提供各种深度学习算法调用。这里推荐个学习资源： 强烈推荐的 TensorFlow、Pytorch 和 Keras 的样例资源，特别同意这篇作者说的: 只要把以上资源运行一次，不懂的地方查官方文档，很快就能理解和运用这三大框架。

在后面的模型训练代码中可以看到实际任务中是怎么使用这些框架的。

目标检测框架

目标检测框架可以理解成是把目标检测算法整合在一起的一个库，例如深度学习算法框架 TensorFlow 不是一个目标检测框架，但它提供目标检测的 API：Object Detection API。

目标检测框架主要有：Detecn-benchmark、mmdetection、Detectron2。目前使用较广的是 Facebook AI 研究院于 2019 年 10 月 10 日开源的 Detectron2 目标检测框架。我们做 UI 界面组件识别也是用的 Detectron2， 后面会有使用示例代码。tron、maskrcn 可参考：如何评价 FAIR 于 2019 年 10 月 10 日开源的 Detectron2 目标检测框架?

前端机器学习框架 Pipcook

作为一个前端开发者，我们还可以选择 Pipcook，这是由阿里巴巴前端委员会智能化小组开源的一个帮助前端工程师使用机器学习的前端算法工程框架。

Pipcook 使用对前端友好的 JS 环境，基于 Tensorflow.js 框架作为底层算法能力并且针对前端业务场景包装了相应算法，从而让前端工程师可以快速简单的运用起机器学习的能力。

Pipcook 是基于管道的框架，为前端开发者封装了数据收集、数据接入、数据处理、模型配置、模型训练、模型服务部署、在线训练七个部分的机器学习工程链路。

关于 Pipcook 的原理和使用可查看：

- Pipcook Github | Docs
- pipcook - 让前端拥抱智能化的一站式算法框架
- 怎样基于 tfjs-node 构建一个高阶前端机器学习框架

样本准备

环境和模型准备好了，机器学习的大头还是数据集的收集和处理。我们的样本来源有两种：

- 阿里系应用的 UI 界面图片。目前移动端 UI 界面 25647 张图片，人工标注了 10 个分类共计 49120 个组件。
- 代码自动生成的图片。支持 10 个分类的样本生成，生成图片时自动标注。

组件类型定义

目前圈定的组件类型有 Statusbar、Navbar、Searchbar、Tabbar 等，不管是人工标注还是自动标注目标组件，都需要有明确的组件类型定义。

- 人工标注时需要根据明确定义的特征来标注组件。
- 自动生成时需要根据明确定义的特征来编写样式代码。

例如，对手机状态栏 Status Bar 的定义：

例如，对选项卡 Tab Bar 的定义：

阿里系应用 UI 界面样本

阿里系 APP 和产品业务有很多，这个业务的视觉稿都有平台集中管理，我们可以拿到这些视觉稿作为样本来源。目前只选取了 Sketch 视觉稿， 因为 PSD 文件难以导出以页面为维度的图片。

收集样本

这里讲的比较细，因为我觉得有些地方可以还是给人启示的。

下载 Sketch 文件

从阿里内部平台下载 Sketch 文件是第一步，后面的每个步骤脚本都以序号开头，例如 1-download-sketch.ts。因为样本处理脚本很多，友好的命名更易理解。

```
/**
* 【用途】下载 Sketch 文件
* 【命令】ts-node 1-download-sketch.ts
*/
```

使用 sketchtool 批量导出为图片

Sketch 自带一个命令行工具 sketchtool，我们可以用 sketchtool 来批量导出为图片。点此查看更多 sketchtool 的用法。

```
# 【用途】使用 sketchtool 导出 Sketch 中的 Artboards 保存为 1x 的 png 图片
# 【命令】sh 2-export-image-from-sketch.sh $inputDir $outputDir

for file in $1"/*"
do
sketchtool export artboards $file --output=$2 --formats='png' --scales=1.0
done
```

样本预处理

设计师出的设计稿你懂得，淘宝直播 4.0_v1、淘宝直播 4.0_v2、淘宝直播 4.0_v3...

每个小版本改动可能不太大，每个 Sketch 文件中的页面可能还有：详情页初版、详情页 2 版、详情页终版，这在导出图片之后会有大量重复的图片。

另外还有一些不规范的设计稿，一个画板画一个 ICON 这种的，还有交互稿、PC 端视觉稿等都不是我们需要的。所以需要做一些处理。

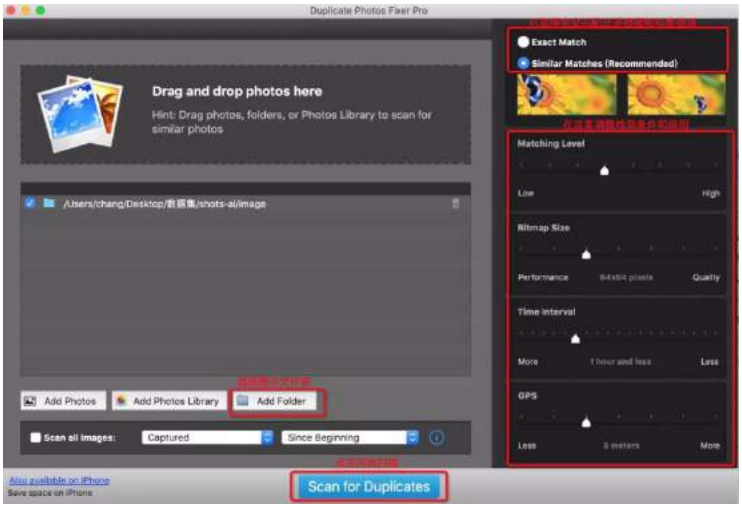
按尺寸分类过滤

将图片分为移动端 和 PC 端，按尺寸去除无效图片。

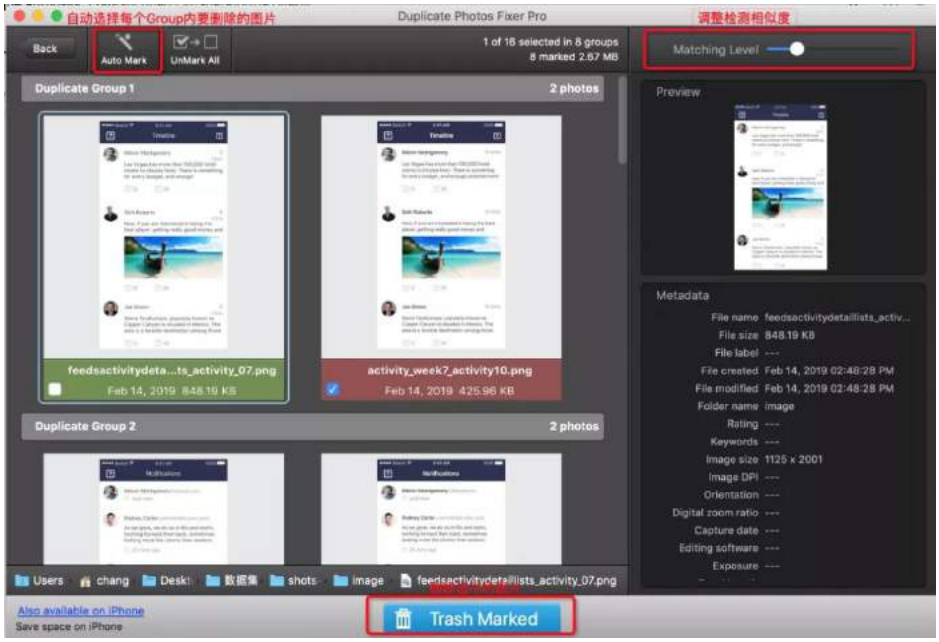
```
# 【用途】将图片按尺寸分类剔除
# 【命令】python3 3-classify-by-size.py $inputDir $outputDir

# 删除尺寸不规范的图片，width_list 中是数量大于 100 的尺寸
if width not in width_list:
  print('move {}'.format(img_name))
move_file(img_dir, other_img_dir, img_name)
# 删除 高度小于 30 的图片
elif height < 30:
print('move {}'.format(img_name))
move_file(img_dir, other_img_dir, img_name)
# 按尺寸归档
else:
width_dir = os.path.join(img_dir, str(width))
if not os.path.exists(width_dir):
print('mkdir:{}'.format(width))
os.mkdir(width_dir)
print('move {}'.format(img_name))
move_file(img_dir, width_dir, img_name)
```

图片去重

如果自己写图片去重的逻辑也可以，做一个图片相似度比较。这里偷个懒，直接用了现成的图片相似度检测工具 duplicate Photos Fixer Pro。这里简单说一下使用方法， 如图红框提示，支持调整检测条件和相似程度。



每张图片的 Hash 值计算完成之后，还可以再调整相似度来筛选。



图片重命名

这里主要是想说一下怎么做样本管理，因为数据集是会逐步丰富的，可能会存在很多个版本的数据集。友好的命名便于管理，比如 generator-mobile-sample-10cate-20200101-1.png，表明这是 2020.01.01 自动生成的第 1 个移动端样本，这一批数据集包含 10 个分类。

样本标注

半自动标注

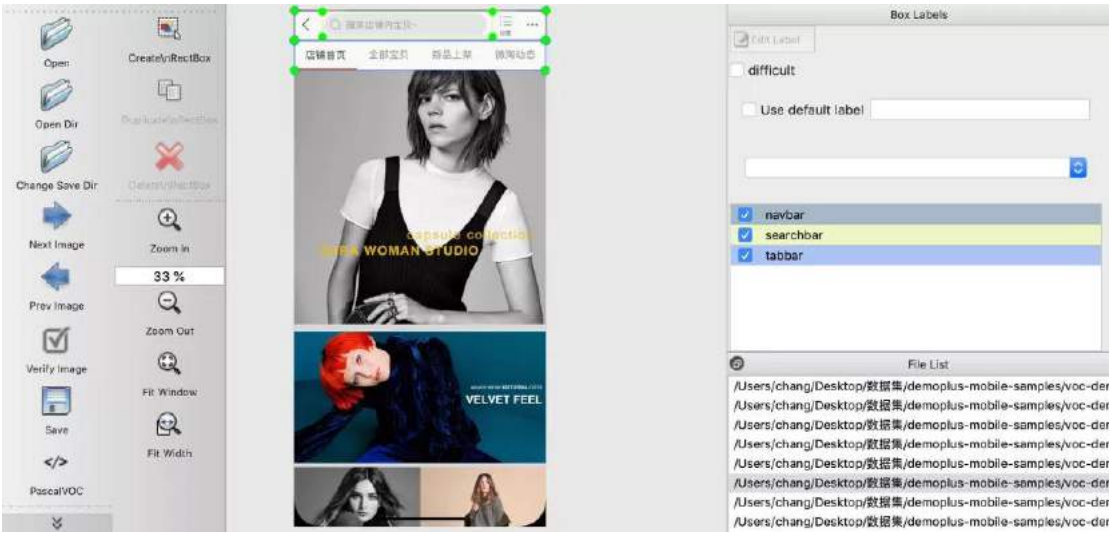
有一些组件是可以自动标注的，比如 statusbar 和 navbar， 因为几乎每张图片都会有且位置和尺寸基本相同，所以可以为每张图片自动生成一个 VOC 格式的 xml， 包含两个目标组件分类。然后在人工标注其他组件时只有少部分需要调整，可以节省很多人力。
目前也在探索更多的半自动化标注方式以减少人工标注的成本。

人工标注

使用 labelImg 工具进行人工标注，按照链接中提供的安装步骤安装，这里简单介绍下用法。

```
// 下载 labelImggit clone https://
github.com/tzutalin/labelImg.git

// 进入 labelImg
cd labelImg-master

// 之后按照 github 中的提示安装环境

// 执行一下命令就会打开可视化界面
python3 labelImg.py
```

可视化界面如下， 支持 Pascal VOC 和 Yolo 两种保存标注的格式。



界面怎么用就不说了，这里推荐一些提高标注效率的快捷键，另外 选中 View > Auto Save mode 可以自动保存。

```
w 新建立一个矩形框

d 下个图片
```

a 上个图片

del/fn + del 删除选中的矩形框，我的电脑需要 fn + del

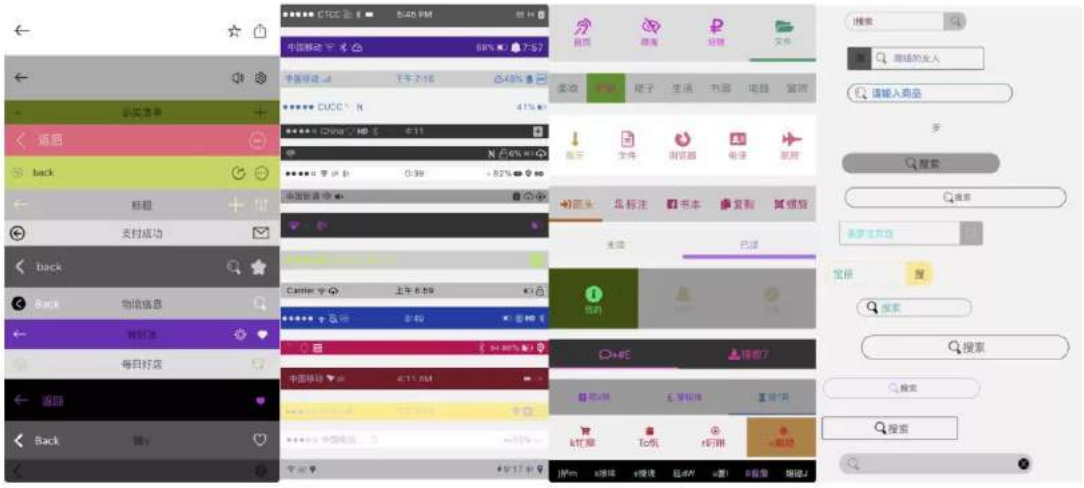
Ctrl/Command++ 放大

Ctrl/Command-- 缩小

↑→↓← 移动矩形框

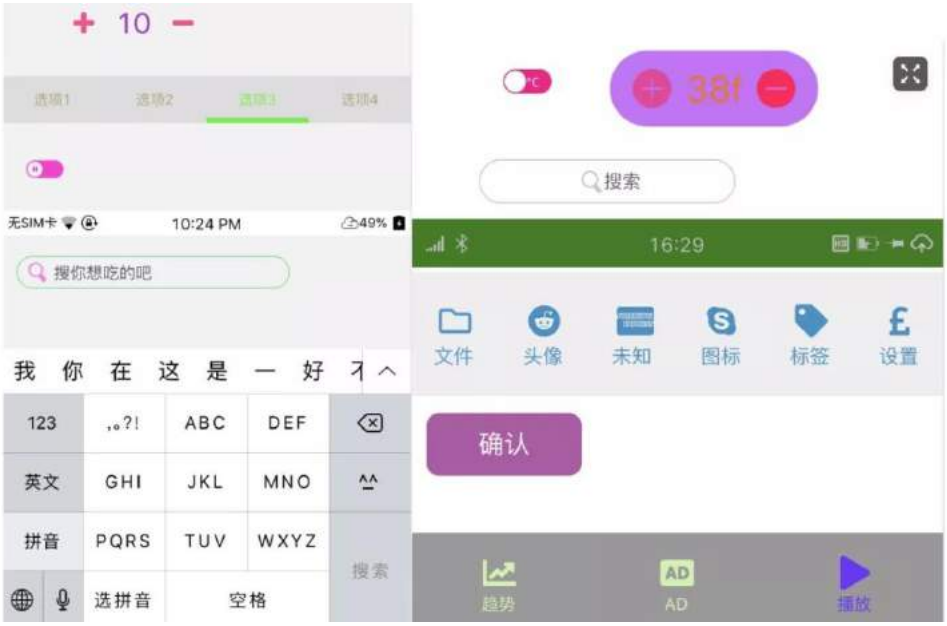
Puppeteer 自动生成样本

根据组件类型定义随机生成组件，这个组件的样式也是随机的。示例如下， 每个正样本都有一个 class 为 element-开头的选择器，例如 element-button 便于后续获取组件的类别信息。

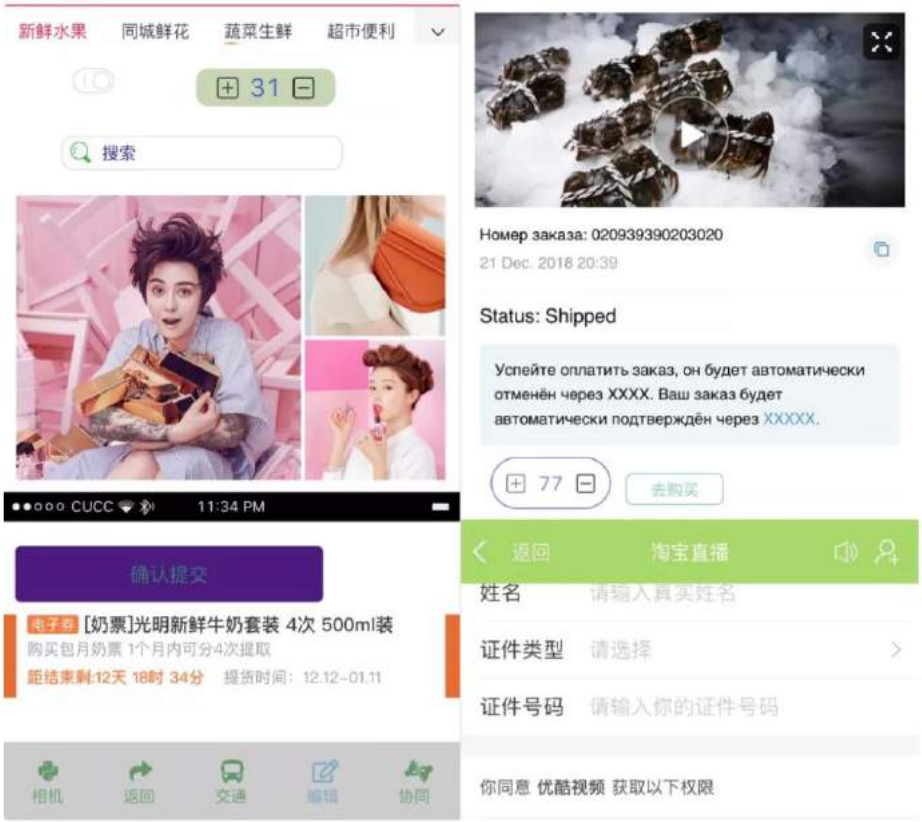


随机生成页面

编写一个页面，随机选取一些组件展示。本地启动服务，打开页面，例如 http://127.0.0.1:3333/#/generator，示例页面如下：



但是这个的页面与实际的 UI 界面相差太大，只有正样本，背景太简单。这里通过裁剪出真实的 UI 界面中的片段与自动生成的目标组件组合的方式，来提高生成样本的质量， 示例样本如下，应该能看出来页面中自动生成的组件吧?



Puppeter 截图生成样本

确定随机页面（http://127.0.0.1:3333/#/generator ）可访问后，使用 Puppeteer 编写脚本自动打开页面、截图保存、并获取组件的类别和边界框。主要逻辑如下：

```
const pptr = require('puppeteer')
// 存放 COCO 格式的样本数据
const mdObj = {};
const browser = await pptr.launch();
const page = await browser.newPage();
await page.goto(`http://127.0.0.1:3333/#/generator/${Date.now()}`)
await page.evaluate(() => {
  const container: HTMLElement | null = document.querySelector('.container');
  const elements = document.querySelectorAll('.element');
  const msg: any = {bbox: []};
  // 获取页面中所有带 .element 选择器的元素
  elements.forEach((element) => {
  const classList = Array.from(element.classList).join(',')
  if (classList.match('element-')) {
  // 获取类别
  const type = classList.split('element-')[1].split(',')[0];
  // 计算边界框并保存至 msg
  pushBbox(element, type);
  }
  });
  });
  // 保存 COCO 格式样本数据
  logToFile(mdObj);
  // 保存 UI 截图
  await page.screenshot({path: 'xxx.png'});
  // 关闭浏览器
  await browser.close();
```

样本评估

阿里系应用 UI 界面样本中，组件数量和丰富度不平衡，可通过自动生成样本来平衡每种组件的数量。对于自动生成的样本，如何来评估样本质量？如何自动生成目标组件的逻辑可枚举为 1 万种，自动生成 2 万个此类组件就没有意义。如何评估自动生成的样本的丰富度和

数量是否合理？这是目前我们在探索的问题。

模型训练

Detectron 2

使用 Facebook 开源的目标检测框架 Detectron 2，通过 merge_from_file 指定使用 Faster R-CNN。

```
    from detectron2.data import MetadataCatalog
    from detectron2.evaluation import PascalVOCDetectionEvaluator

    from detectron2.engine import DefaultTrainer,hooks
    from detectron2.config import get_cfg

    cfg = get_cfg()

    cfg.merge_from_file("./lib/detectron2/configs/COCO-Detection/faster_rcnn_R_50_C4_3x.yaml")
    cfg.DATASETS.TRAIN = ("train_dataset",)
    cfg.DATASETS.TEST = ('val_dataset',) # no metrics implemented for this dataset
    cfg.DATALOADER.NUM_WORKERS = 4 # 多开几个 worker 同时给 GPU 喂数据防止 GPU 闲置
    cfg.MODEL.WEIGHTS = "detectron2://ImageNetPretrained/MSRA/R-50.pkl" # initialize from model
zoo
    cfg.SOLVER.IMS_PER_BATCH = 4
    cfg.SOLVER.BASE_LR = 0.000025
    cfg.SOLVER.NUM_GPUS = 2
    cfg.SOLVER.MAX_ITER = 100000 # 300 iterations seems good enough, but you can certainly train
longer
    cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128 # faster, and good enough for this toy
dataset
    cfg.MODEL.ROI_HEADS.NUM_CLASSES = 29 # only has one class (ballon)

    # 训练集
    register_coco_instances("train_dataset", {}, "data/train.json", "data/img")
    # 测试集
    register_coco_instances("val_dataset", {}, "data/val.json", "data/img")

    import os
```

```
os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)

class Trainer(DefaultTrainer):
@classmethod def build_evaluator(cls, cfg, dataset_name, output_folder=None):
### 按需求重写

@classmethod
def test_with_TTA(cls, cfg, model):
### 按需求重写

trainer = Trainer(cfg)
trainer.resume_or_load(resume=True)
trainer.train()
```

使用 Detectron 2 模型训练的产物是一个.pth 格式的模型文件。这个.pth 格式的模型文件长啥样可以看下这篇 Pytorch 中保存的模型文件.pth 深入解析。

Pipcook

Pipcook 已经帮我们封装好了从数据收集、数据接入、模型训练、模型评估的代码，我们无需写这些工程链路的 Python 脚本。目前 Pipcook 中的目标检测链路使用的是 Detectron 2 中的 Faster RCNN 算法，大家可以去 Pipcook Plugins 看下实现就明白了。

以下是使用 Pipcook 进行目标检测的示例代码。

```
const {DataCollect, DataAccess, ModelLoad, ModelTrain, ModelEvaluate, PipcookRunner} =
require('@pipcook/pipcook-core');

const imageCocoDataCollect
=require('@pipcook/pipcook-plugins-image-coco-data-collect').default;
const imageDetectronAccess =
require('@pipcook/pipcook-plugins-detection-detectron-data-access').default;
const detectronModelLoad =
require('@pipcook/pipcook-plugins-detection-detectron-model-load').default;
const detectronModelTrain =
require('@pipcook/pipcook-plugins-detection-detectron-model-train').default;
```

```
    const detectronModelEvaluate =
require('@pipcook/pipcook-plugins-detection-detectron-model-evaluate').default;

    async function startPipeline() {
    // collect detection data
    const dataCollect = DataCollect(imageCocoDataCollect, {
    url:'http://ai-sample.oss-cn-hangzhou.aliyuncs.com/image_classification/datasets/autoLayoutGrou
pRecognition.zip',
    testSplit: 0.1,
    annotationFileName: 'annotation.json'
    });
    const dataAccess = DataAccess(imageDetectronAccess);
    const modelLoad = ModelLoad(detectronModelLoad, {
    device: 'cpu'
     });
    const modelTrain = ModelTrain(detectronModelTrain);
    const modelEvaluate = ModelEvaluate(detectronModelEvaluate);

    const runner = new PipcookRunner( {
    predictServer: true
    });
    runner.run([dataCollect, dataAccess, modelLoad, modelTrain, modelEvaluate])}
    startPipeline();
```

模型评估

评估指标

发现文章太长了，这里扔个学习资源吧， 强烈推荐这个视频课程--Python3 入门机器学习经典算法与应用[慕课网] ，没有时间看的话直接看人家写的笔记吧，第 10 章详细讲了使用精准率和召回率评估分类结果。

啊， 还是简单解释下吧。

精准率可以理解成查准率，比如预测了 100 个是 Button，其中 80 个是预测正确的，精准率是 80 / 100。

召回率可以理解成查全率，比如实际有 60 个是 Button， 成功预测了 40 个，召回率是 40 / 60。

在目标检测的性能评价指标是 mAP 和 FPS，mAP 计算的是所有类别的平均准确率， 但由于目标检测结果除了类别还有个边界框，如何评价这个边界框的预测准确性，又涉及到 IoU （Intersection over Union）交并比的概念，用来表示预测的边界框与真实的边界框的交并比。

在后面的评估结果中可以看到这样的结果，当 IoU=0.50:0.95 即预测的边界框与真实的边界框的交并比在 0.5 到 0.95 之间时都算边界框预测正确，此时的精准率 AP 为 0.772。

```
Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.772
Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.951
Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.915
```

评估代码

评估代码如下：

```
from pycocotools.cocoeval import COCOeval
from pycocotools.coco import COCO
annType = 'bbox
'# 测试集 ground truth
gt_path = '/Users/chang/coco-test-sample/data.json
'# 测试集 预测结果
dt_path = '/Users/chang/coco-test-sample/predict.json'

gt = COCO(gt_path)
gt.loadCats(gt.getCatIds())

dt = COCO(dt_path)
imgIds=sorted(gt.getImgIds())
cocoEval = COCOeval(gt,dt,annType)

for cat in gt.loadCats(gt.getCatIds()):
cocoEval.params.imgIds = imgIds
```

```
    cocoEval.params.catIds = [cat['id']]
    print '------------------------------ ' cat['name'] ' --------------------------------'
    cocoEval.evaluate()
    cocoEval.accumulate()
    cocoEval.summarize()
```

评估结果

目前用阿里系 UI 界面与自动生成的样本组合训练， mAP 基本在 75% 左右。

```
------------------------------ searchbar --------------------------
    Running per image evaluation...
    Evaluate annotation type *bbox*
    DONE (t=2.60s).
    Accumulating evaluation results...
    DONE (t=0.89s).
    Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.772
    Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.951
    Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.915
    Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
    Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.795
    Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.756
    Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.816
    Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.830
    Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.830
    Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
    Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.838
    Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.823
```

我们用非阿里系的 UI 界面看下预测结果， 可以看出一些容易被误识别的情况。

模型服务部署

我们期望的是当输入一张图片时能返回给模型预测的结果。所以拿到模型文件之后，需要写一个模型服务，接收一个样本，并返回模型预测的结果。

```
from detectron2.config import get_cfg
from detectron2.engine.defaults import DefaultPredictor

with open('label.json') as f:
mp = json.load(f)

cfg = get_cfg()
cfg.merge_from_file("./config/faster_rcnn_R_50_C4_3x.yaml")
cfg.MODEL.WEIGHTS = "./output/model_final.pth" # initialize from model zoo
cfg.MODEL.ROI_HEADS.NUM_CLASSES = len(mp) # only has one class (ballon)
cfg.MODEL.DEVICE='cpu'

model = DefaultPredictor(cfg)

def predict(image):
im2 = cv2.imread(image)
out = model(x)
data = {'status': 200}
data['content'] = trans_data(out)

# EAS python sdk
import spark
```

```
num_io_threads = 8
endpoint = '' # '127.0.0.1:8080'

spark.default_properties().put('rpc.keepalive', '60000')
context = spark.Context(num_io_threads)
queued = context.queued_service(endpoint)

while True:
receive_data = srv.read()
try:
msg = json.loads(receive_data.decode())
ret = predict(msg["image"])
srv.write(json.dumps(ret).encode())
except Exception as e:
srv.error(500, str(e))
```

模型部署后拿到访问链接即可直接调用例如 http://example.com/api/predict/detect 来预测。

模型应用

模型部署之后就可以在我们的应用中调用获取预测结果(视觉稿中的组件类别和边界框)，与从视觉稿中导出的 JSON 树比对，从而获得一个带有组件信息(D2C Schema 中的 smart.layerProtocol.component 字段)的 JSON 描述（处理后的最终 JSON 作为 DSL 的输入生成代码）。

```
const detectUrl = 'http://example.com/api/predict/detect';
const res = await request(detectUrl, {
method: 'post',
dataType: 'json',
timeout: 1000 * 10,
content: JSON.stringify({
image: image,
}),
});
const json = res.content;
```

关于模型服务的部署和调用可以去 PAI 的阿里云文档查看。

未来展望

由于选择的是深度学习算法， 需要大量的训练集样本，所以样本数量和质量是亟需解决的问题。

目前我们已经拥有 2.5 万+ 的 UI 界面样本，包含 10 个分类，自动生成的样本支持 10 个分类。但人工标注的 UI 界面样本均为阿里系产品，虽然样本图片不同但设计风格会有相似且设计规范较统一，使得组件样式的丰富度不够，对阿里系之外的设计稿泛化能力较阿里系的要差。另外遵循一定的随机化规则自动生成的样本还存在版式和样式与实际样本有差异的地方，自动生成的样本质量无法评估。

未来在数据集方面，会考虑加入业界样本数量较大的数据集，并优化样本自动生成逻辑，同时探索评估自动生成的样本质量的方法。

任务调度：时间轮算法经典案例解析及应用实现

作者 | 平勇（润辰）
出品 | 淘系技术

平时大家的工作中应该会遇到较多需要在某个时间点执行某个任务，比如对运维来说，定时数据库的备份，日志和监控信息的抓取；比如业务系统，某个时间点给某个人群用户发放优惠券，甚至从操作系统角度，人机交互进程、视频播放的实时进程、批处理的后台进程等进程间的调度。。。
所以如何将这些任务高效、精准的调度？是任务调度系统中最重要的命题，当然在业务系统中一个完善的任务调度系统是很复杂的，需要具备能调度、可视化管理、过程可追溯、结果可分析、持久化、高可用等特性，这篇文章主要讨论任务调度逻辑，其余的内

背景

最近的一次招聘面试中，候选人介绍了他基于时间轮算法实现的一套任务调度系统，在我的个人历史工作中简单的场景会直接使用 java 自带的工具类，如 Timer、ScheduledThreadPool、DelayQueue 配合线程池等方法，复杂的一些场景用开源工具包 Quartz、集团的 SchedulerX 等，但一直没有对这些工具的底层实现逻辑做过深入的学习，此次的沟通引起了我的兴趣。

我们可以将业务系统中需要使用定时任务调度的总结成以下三种场景：

1. 时间驱动处理场景：如整点发送优惠券，每天定时更新收益，每天定时刷新标签数据和人群数据。
2. 批量处理数据：如按月批量统计报表数据，批量更新某些数据状态，实时性要求不高。
3. 异步执行解耦：如先反馈用户操作状态，后台异步执行较耗时的数据操作，以实现异步逻辑解耦。

所以从时间维度可以分为定时或者延迟一定时间两个维度，而时间轮就是一种高效的利用线程资源来进行批量化调度的一种调度模型，把大批量的调度任务全部都绑定到同一个的调度器上面，使用这一个调度器来进行所有任务的管理，触发以及运行，能够高效的管理各种延时任务，周期任务。

这篇文章主要有三块内容：

1. JAVA 自带的工具类的内部实现逻辑是什么？
2. 时间轮算法原理？
3. 时间轮算法有哪些应用案例？（Netty、Kafka 案例）

Timer、ScheduledThreadPool、DelayQueue 实现逻辑

Timer

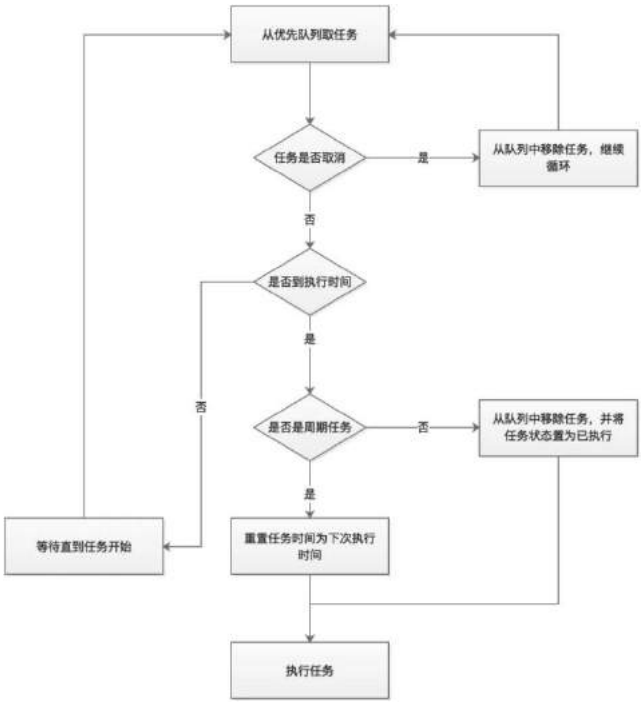
Timer 可以实现延时任务，也可以实现周期性任务。我们先来看看 Timer 核心属性和构造器。

```
private final TaskQueue queue = new TaskQueue();//基于数组实现的优先队列，用作存放任务

private final TimerThread thread = new TimerThread(queue);//执行任务线程

public Timer(String name) {
    thread.setName(name);
    thread.start(); //构造时默认启动线程
}
```

Timer 核心就是一个优先队列和封装的执行任务的线程，从这我们也可以看到一个 Timer 只有一个线程执行任务，再来看看是如何实现延时和周期性任务的。

我先简单的概括一下：首先维持一个小顶堆，即最快需要执行的任务排在优先队列的第一个，根据堆的特性我们知道插入和删除的时间复杂度都是 O(logn)，然后 TimerThread 不断地

拿排在第一个任务的执行时间和当前时间做对比，如果时间到了先看看这个任务是不是周期性执行的任务？如果是则修改当前任务时间为下次执行的时间，如果不是周期性任务则将任务从优先队列中移除，最后执行任务。如果时间还未到则调用 wait() 等待（参考以下流程图）。



再看下代码逻辑，先来看下 TaskQueue，主要看下插入任务的过程，特别是插入时间复杂度部分。

```
class TaskQueue {
private TimerTask[] queue = new TimerTask[128];
void add(TimerTask task) {
  // Grow backing store if necessary
  if (size + 1 == queue.length)
    queue = Arrays.copyOf(queue, 2*queue.length); //扩容

    queue[++size] = task; //先将任务添加到数组最后面
    fixUp(size); //调整堆
    }
    private void fixUp(int k) { //时间复杂度为 O(logn)
    while (k > 1) {
```

```
            int j = k >> 1;
            if (queue[j].nextExecutionTime <= queue[k].nextExecutionTime)//通过任务执行时间对比，调整
顺序
            break;
            TimerTask tmp = queue[j];   queue[j] = queue[k]; queue[k] = tmp;
            k = j;
            }
    }
    /**
     * Return the "head task" of the priority queue.  (The head task is an
     * task with the lowest nextExecutionTime.)
     */
    TimerTask getMin() {
    return queue[1]; //返回最接近执行时间的任务
    }
    //.....
}
```

再来看看 TimerThread 的 run 操作。

```
    public void run() {
      try {
      mainLoop();//无异常捕获
      } finally {
      // Someone killed this Thread, behave as if Timer cancelled
      synchronized(queue) {
        newTasksMayBeScheduled = false;
        queue.clear();  // Eliminate obsolete references
        }
      }
    }

    /**
     * The main timer loop.  (See class comment.)
     */
    private void mainLoop() {
      while (true) {
      try {
        TimerTask task;
        boolean taskFired;
        synchronized(queue) {
```

```
    // Wait for queue to become non-empty
    while (queue.isEmpty() && newTasksMayBeScheduled)
    queue.wait();

    if (queue.isEmpty())
    break;

    // Queue is empty and will forever remain; die// Queue nonempty; look at first evt and do the right
thing
    long currentTime, executionTime;
    task = queue.getMin(); //获取任务
    synchronized(task.lock) {
    if (task.state == TimerTask.CANCELLED) { //取消泽移除并继续循环
    queue.removeMin();
    continue;   // No action required, poll queue again
    }
    currentTime = System.currentTimeMillis();
    executionTime = task.nextExecutionTime;
    if (taskFired = (executionTime<=currentTime)) { //执行时间到了

      if (task.period == 0) { // 不是周期任务
      queue.removeMin(); //移除任务
      task.state = TimerTask.EXECUTED;//变更任务状态为已执行
      } else { // 周期任务，更新时间为下次执行时间
      queue.rescheduleMin(
      task.period<0 ? currentTime     -task.period
      : executionTime + task.period);
            }
        }
      }
      if (!taskFired) // 还未到达执行时间等待
      queue.wait(executionTime - currentTime);
      }
      if (taskFired)   // 执行任务，无异常捕获
      task.run();
      } catch(InterruptedException e) {
            }
        }
      }
```

小结：

可以看出 Timer 实际就是根据任务的执行时间维护了一个优先队列，并且起了一个线程不断地拉取任务执行，根据代码可以看到有三个问题：

1. 优先队列的插入和删除的时间复杂度是 O(logn)，当任务量大的时候，频繁的入堆出堆性能有待考虑。
2. 单线程执行，如果一个任务执行的时间过久则会影响下一个任务的执行时间(当然你任务的 run 要是异步执行也行)。
3. 从代码中可以看到对异常没有做什么处理，那么一个任务出错的时候会导致之后的任务都无法执行。

ScheduledThreadPoolExecutor

在看 Timer 源码时，看到了以下一段注释：

```
/** *
 ..... *
  <p>Java 5.0 introduced the {@code java.util.concurrent} package and
 * one of the concurrency utilities therein is the {@link
 * java.util.concurrent.ScheduledThreadPoolExecutor
 * ScheduledThreadPoolExecutor} which is a thread pool for repeatedly
 * executing tasks at a given rate or delay.  It is effectively a more
 * versatile replacement for the {@code Timer}/{@code TimerTask}
 * combination, as it allows multiple service threads, accepts various
 * time units, and doesn't require subclassing {@code TimerTask} (just
 * implement {@code Runnable}).  Configuring {@code
 * ScheduledThreadPoolExecutor} with one thread makes it equivalent to
 * {@code Timer}.
 * .....
 */public class Timer {
//。。。。
}
```

简单翻译下：ScheduledThreadPoolExecutor 是一个具有更多功能的 Timer 的替代品，允许

多个服务线程，如果设置一个服务线程和 Timer 没啥差别。其实从这段注释中已经能看出 ScheduledThreadPoolExecutor 的定位，通过注释看出相对于 Timer，可能就是单线程跑任务和多线程跑任务的区别，我们从代码详细看下。

```
public class ScheduledThreadPoolExecutor
extends ThreadPoolExecutor
implements ScheduledExecutorService {
// ...
public ScheduledThreadPoolExecutor(int corePoolSize) {
super(corePoolSize, Integer.MAX_VALUE, 0, NANOSECONDS,
new DelayedWorkQueue());
  }
private class ScheduledFutureTask<V>
extends FutureTask<V> implements RunnableScheduledFuture<V> {
//...
}
}
```

从核心属性和构造器看，它继承了 ThreadPoolExecutor，实现了 ScheduledExecutorService，基本确认是基于线程池实现了延迟任务操作，我们知道线程池流程中有两个重要角色：任务，阻塞队列，ScheduledThreadPoolExecutor 实现方案也是将这两者进行了重新设计：一个是 ScheduledFutureTask ，一个是 DelayedWorkQueue。

```
private class ScheduledFutureTask<V>
  extends FutureTask<V> implements RunnableScheduledFuture<V> {
  //...
  public void run() {
    boolean periodic = isPeriodic();
    if (!canRunInCurrentRunState(periodic))
    cancel(false);
    else if (!periodic)
    ScheduledFutureTask.super.run(); //如果不是周期任务，直接 run
    else if (ScheduledFutureTask.super.runAndReset()) {//周期性任务,执行并重置任务状态为 NEW
    setNextRunTime();//先设置下次任务执行时间
    reExecutePeriodic(outerTask); //任务重新入队列
        }
      }
    //...}
```

ScheduledFutureTask 继承自 FutureTask 重写了 run 方法，实现了周期性任务的场景。

```
    static class DelayedWorkQueue extends AbstractQueue<Runnable>
    implements BlockingQueue<Runnable> {

    public RunnableScheduledFuture<?> take() throws InterruptedException {
      final ReentrantLock lock = this.lock;
      lock.lockInterruptibly();
      try {
        for (;;) {
      RunnableScheduledFuture<?> first = queue[0];
      if (first == null)
        available.await(); //如果队列中没有任务，等待
      else {
      long delay = first.getDelay(NANOSECONDS);
      if (delay <= 0)//有任务并且到执行时间了
        return finishPoll(first);
      first = null; // don't retain ref while waiting
      if (leader != null)
        available.await();
      else {
        Thread thisThread = Thread.currentThread();
        leader = thisThread;
        try {
        available.awaitNanos(delay);//有任务，没有到执行时间
        } finally {
        if (leader == thisThread)
        leader = null;
                }
              }
            }
          }
      } finally {
      if (leader == null && queue[0] != null)
      available.signal();
      lock.unlock();
          }
      }
}
```

DelayedWorkQueue 继承阻塞队列，实现了优先队列逻辑，本质也是利用数组实现的小顶堆，同时重写了 take 等方法，做了延迟阻塞处理。

现在我们来看下 ScheduledThreadPoolExecutor 提交一个任务后，整体的执行过程：

- 提交一个任务后，为了满足 ScheduledThreadPoolExecutor 能够延时执行任务和能周期执行任务的特性，会先将实现 Runnable 接口的类转换成 ScheduledFutureTask。
- 然后会调用 delayedExecute 方法进行执行任务:先将任务放入到队列中，然后调用 ensurePrestart 方法，新建 Worker 类（此逻辑为线程池 ThreadPoolExecutor 实现）。
- 当执行任务时，就会调用被 Worker 所重写的 run 方法，进而会继续执行 runWorker 方法。在 runWorker 方法中会调用 getTask 方法从阻塞队列中不断的去获取任务进行执行，直到从阻塞队列中获取的任务为 null 的话，线程结束终止。(此处逻辑都是线程池 ThreadPoolExecutor 的实现)。
  - getTask 方法会调用队列的 poll 和 take 方法，此处就调用到 DelayedWorkQueue 重写的 poll 和 take 逻辑，实现了延迟任务的阻塞。
  - 执行任务时，将调用 ScheduledFutureTask 重载的 run 方法，实现周期性任务的场景。

小结：

- ScheduledThreadPoolExecutor 继承了 ThreadPoolExecutor，通过重写任务、阻塞队列实现了延迟任务调度的实现。
- ScheduledThreadPoolExecutor 大致的流程和 Timer 差不多，都是通过一个阻塞队列维护任务，能实现单次任务、周期性任务的执行，主要差别在于能多线程运行任务，不会单线程阻塞，并且 Java 线程池的底层 runworker 实现了异常的捕获，不会因为一个任务的出错而影响之后的任。
- 在任务队列的维护上，与 Timer 一样，也是优先队列，插入和删除的时间复杂度是 O(logn)。

DelayQueue

我们还可以通过 DelayQueue 配合线程池的方式实现延迟任务，看了下 DelayQueue 的源码，实现比较简单，实现逻辑和 ScheduledThreadPoolExecutor 中 DelayedWorkQueue 基本差不多，内部使用优先队列，队列元素为 Delayed，Delayed

接口继承了 Comparable,优先队列通过 Comparable 进行排序。

```
//元素必须实现 Delayed 接口，也实现了阻塞队列
public class DelayQueue<E extends Delayed> extends AbstractQueue<E>
    implements BlockingQueue<E> {

    private final transient ReentrantLock lock = new ReentrantLock();
    private final PriorityQueue<E> q = new PriorityQueue<E>();//优先队列，

public E take() throws InterruptedException {
    final ReentrantLock lock = this.lock;
    lock.lockInterruptibly();
    try {
    for (;;) {
    E first = q.peek();
    if (first == null)
    available.await();
    else {
    long delay = first.getDelay(NANOSECONDS);
    if (delay <= 0) //小于等于 0，时间到了
    return q.poll();
    first = null; // don't retain ref while waiting
    if (leader != null)
    available.await();//没有抢到 leader 的线程进入等待，避免大量唤醒操作
    else {
    Thread thisThread = Thread.currentThread();
    leader = thisThread;
    try {
    available.awaitNanos(delay);//leader 线程，在等待一定时间后再次尝试获取
    } finally {
    if (leader == thisThread)//重置 leader
    leader = null;
            }
```

```
            }
          }
        }
      } finally {
      if (leader == null && q.peek() != null)
      available.signal();
      lock.unlock();
      }
  }
      //...
      }/
      /继承了 Comparable
      public interface Delayed extends Comparable<Delayed> {

      long getDelay(TimeUnit unit);
      }
```

小结：

延迟队列就是个容器，需要调用方获取任务和执行任务。整体实现比较简单，利用优先队列实现的，元素通过实现 Delayed 接口来返回延迟的时间。

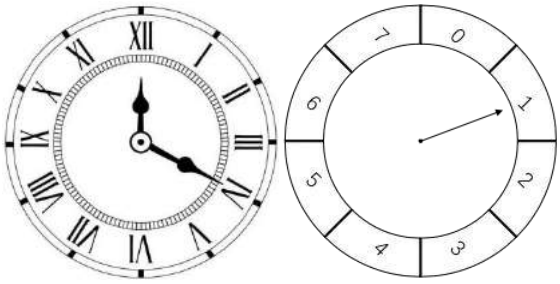
总结

三者实现思想基本类似，都围绕三个要素：任务、任务的组织者（队列），执行者调度执行者。

Timer、ScheduledThreadPoolExecutor 完整的实现了这三个要素，DelayQueue 只实现了任务组织者这个要素，需要与线程配合使用。其中任务组织者这个要素，它们都是通过优先队列来实现，因此插入和删除任务的时间复杂度都为 O(logn)，并且 Timer 、ScheduledThreadPool 的周期性任务是通过重置任务的下一次执行时间来完成的。

个人觉得在时间复杂度上存在较大的性能风险，插入删除时间复杂度是 O(logn)，如果面对大数量级的任务频繁的插入、删除，这种实现方式性能问题会比较严重。

时间轮算法

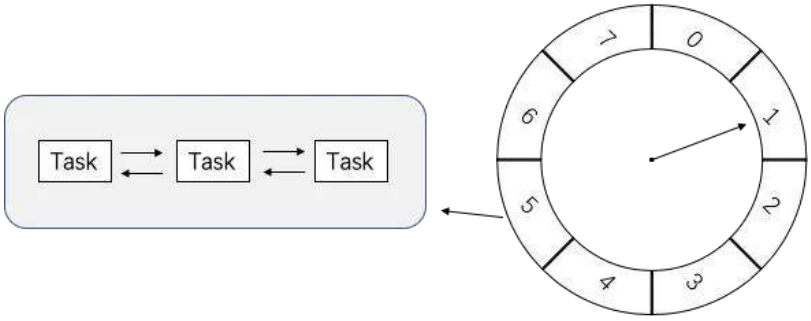


前面学习中基本可以总结出延迟任务有这几种调度：

1. 约定一段时间后执行
2. 约定某个时间点执行
3. 周期性执行

1、2，这两者之间可以相互转换，比如给你个任务，要求 12 点钟执行，你看了一眼时间，发现现在是 9 点钟，那么你可以认为这个任务过三个小时执行。同样的，给你个任务让你 3 个小时后执行，你看了一眼现在是 9 点钟，那么你当然可以认为这个任务 12 点钟执行。

对于 3，需重复执行的任务，其实我们需要关心的只是下次执行时间，并不关心这个任务需要循环多少次，比如每天上午 9 点执行生成报表的任务，在当天 9 点任务执行时发现是周期性任务，只需要将该任务下次执行时间设置成明天 9 点即可，以此重复，直至到了结束时间，将任务移除。我们可以将这些场景抽象用时间轮管理，时间轮就是和钟表很相似的存在！



时间轮用环形数组实现，数组的每个元素可以称为槽，槽的内部用双向链表存着待执行的任务，添加和删除的链表操作时间复杂度都是 O(1)，槽位本身也指代时间精度，比如一秒扫一个槽，那么这个时间轮的最高精度就是 1 秒。也就是说延迟 1.2 秒的任务和 1.5 秒的任务会被加入到同一个槽中，然后在 1 秒的时候遍历这个槽中的链表执行任务。

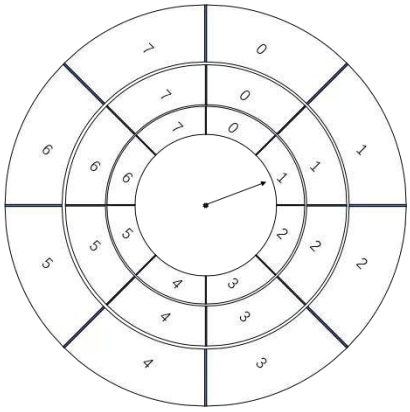
案例如图，从图中可以看到此时指针指向的是第 2 个槽（下标 1），一共有八个槽 0~7，假设槽的时间单位为 1 秒，现在要加入一个延时 4 秒的任务，计算方式就是 4 % 8 + 2 = 6，即放在槽位为 6，下标为 5 的那个槽中，就是拼到槽的双向链表的尾部。然后每秒指针顺时针移动一格，这样就扫到了下一格，遍历这格中的双向链表执行任务。然后再循环继续。可以看到插入任务从计算槽位到插入链表，时间复杂度都是 O(1)。

那假设现在要加入一个 50 秒后执行的任务怎么办？这槽好像不够啊？难道要加槽嘛？和 HashMap 一样扩容？

常见有两种方式：

一种是通过增加轮次的概念,先计算槽位：50 % 8 + 2 = 4，即应该放在槽位是 4，下标是 3 的位置。然后计算轮次：(50 - 1) / 8 = 6，即轮数记为 6。也就是说当循环 6 轮之后扫到下标的 3 的这个槽位会触发这个任务。Netty 中的 HashedWheelTimer 使用的就是这种方式。

还有一种是通过多层次的时间轮,这个和我们的手表就更像了，像我们秒针走一圈，分针走一格，分针走一圈，时针走一格，多层次时间轮就是这样实现的。假设上图就是第一层，那么第一层走了一圈，第二层就走一格，可以得知第二层的一格就是 8 秒，假设第二层也是 8 个槽，那么第二层走一圈，第三层走一格，可以得知第三层一格就是 64 秒。那么一个三层，每层 8 个槽，一共 24 个槽时间轮就可以处理最多延迟 512 秒的任务。



而多层次时间轮还会有降级的操作，假设一个任务延迟 500 秒执行，那么刚开始加进来肯定是放在第三层的，当时间过了 436 秒后，此时还需要 64 秒就会触发任务的执行，而此时相对而言它就是个延迟 64 秒后的任务，因此它会被降低放在第二层中，第一层还放不下它。再过个 56 秒，相对而言它就是个延迟 8 秒后执行的任务，因此它会再被降级放在第一层中，等待执行。

降级是为了保证时间精度一致性,Kafka 内部用的就是多层次的时间轮算法。

时间轮算法应用案例

很有需要延迟操作的应用场景中都有时间轮的身影，比如 Netty、Kafka、Akka、Zookeeper 等组件，我们挑选上面聊到的时间轮的两种实现方式的典型代表学习一下，轮次时间轮算法实现-Netty，多层次时间轮算法实现-Kafka。

轮次时间轮算法-Netty 案例

在 Netty 中时间轮的实现类是 HashedWheelTimer，代码中的 wheel 就是时间轮循环数组，mask 就是取余找槽位的逻辑，不过这里的设计通过限制槽位数组的大小为 2 的次方，然后利用位运算来替代取模运算，提高性能。

tickDuration 就是每格的时间精度，可以看到配备了一个工作线程来处理任务的执行。以及双向链路的槽 HashedWheelBucket。

```
    public class HashedWheelTimer implements Timer {
    //...
        public HashedWheelTimer(ThreadFactory threadFactory, long tickDuration, TimeUnit unit) {
       this(threadFactory, tickDuration, unit, 512);
       }
       public HashedWheelTimer(ThreadFactory threadFactory, long tickDuration, TimeUnit unit, int
ticksPerWheel, boolean leakDetection, long maxPendingTimeouts) {
          this.worker = new HashedWheelTimer.Worker();
          this.workerState = 0;
          this.startTimeInitialized = new CountDownLatch(1);
          this.timeouts = PlatformDependent.newMpscQueue();
          this.cancelledTimeouts = PlatformDependent.newMpscQueue();
          this.pendingTimeouts = new AtomicLong(0L);
          if (threadFactory == null) {
             throw new NullPointerException("threadFactory");
          } else if (unit == null) {
          throw new NullPointerException("unit");
          } else if (tickDuration <= 0L) {
          throw new IllegalArgumentException("tickDuration must be greater than 0: " + tickDuration);
          } else if (ticksPerWheel <= 0) {

          throw new IllegalArgumentException("ticksPerWheel must be greater than 0: " +
ticksPerWheel);
          } else {
          this.wheel = createWheel(ticksPerWheel);//默认 512 的槽数量
           this.mask = this.wheel.length - 1;//mask 设计提高了"确定槽位下标"的性能，通过限制
wheel.length 为 2 的 N 次方，
          this.tickDuration = unit.toNanos(tickDuration);//时间精度，即每个槽位的时间跨度
          if (this.tickDuration >= 9223372036854775807L / (long)this.wheel.length) {
          throw new IllegalArgumentException(String.format("tickDuration: %d (expected: 0 <
tickDuration in nanos < %d", tickDuration, 9223372036854775807L / (long)this.wheel.length));
          } else {
          this.workerThread = threadFactory.newThread(this.worker);//工作线程
          this.leak = !leakDetection && this.workerThread.isDaemon() ? null : leakDetector.track(this);
          this.maxPendingTimeouts = maxPendingTimeouts;
          if (INSTANCE_COUNTER.incrementAndGet() > 64 &&
WARNED_TOO_MANY_INSTANCES.compareAndSet(false, true)) {
          reportTooManyInstances();
                }
             }
           }
          }
```

```
    //...

    //双向链表的 bucket
        private static final class HashedWheelBucket {
            private HashedWheelTimer.HashedWheelTimeout head;
            private HashedWheelTimer.HashedWheelTimeout tail;
    //...
    }
    //链表元素，任务
        private static final class HashedWheelTimeout implements Timeout {
        HashedWheelTimer.HashedWheelTimeout next;
        HashedWheelTimer.HashedWheelTimeout prev;
        HashedWheelTimer.HashedWheelBucket bucket;

    //...
        }
        }
```

我们从任务的添加作为切入点，看下整体的逻辑。

```
    public Timeout newTimeout(TimerTask task, long delay, TimeUnit unit) {
        if (task == null) {
        throw new NullPointerException("task");
        } else if (unit == null) {
        throw new NullPointerException("unit");
        } else {
        long pendingTimeoutsCount = this.pendingTimeouts.incrementAndGet();
        if (this.maxPendingTimeouts > 0L && pendingTimeoutsCount >
        this.maxPendingTimeouts) {
        this.pendingTimeouts.decrementAndGet();
    throw new RejectedExecutionException("Number of pending timeouts (" +pendingTimeoutsCount
+ ") is greater than or equal to maximum allowed pending timeouts (" + this.maxPendingTimeouts +
")");
        } else {
        this.start();//启动工作线程，内部有判断状态
    long deadline = System.nanoTime() + unit.toNanos(delay) - this.startTime;//计算延迟时间
    HashedWheelTimer.HashedWheelTimeout timeout = new
    HashedWheelTimer.HashedWheelTimeout(this, task, deadline);//创建任务对象
this.timeouts.add(timeout);//添加到队列中，这边标记问题，没看到添加到时间轮中?
    return timeout;
            }       } }
```

我们可以看到，整体逻辑比较清晰：启动工作线程-》创建任务对象-》放入到队列中。此处有个疑问，任务没有直接添加到时间轮中，而是被添加到队列中？唯一的线索是启动了工作线程，我们看一下工作线程的逻辑。

```
    private final class Worker implements Runnable {//工作线程
    //...
    public void run() {
    HashedWheelTimer.this.startTime = System.nanoTime();//启动时间
    if (HashedWheelTimer.this.startTime == 0L) {
    HashedWheelTimer.this.startTime = 1L;
    }
    HashedWheelTimer.this.startTimeInitialized.countDown();

    int idx;
    HashedWheelTimer.HashedWheelBucket bucket;
    do {
    long deadline = this.waitForNextTick();//等待执行任务时间到来
    if (deadline > 0L) {
    idx = (int)(this.tick & (long)HashedWheelTimer.this.mask);//或者槽位下标
    this.processCancelledTasks();//先处理取消了的任务
    bucket = HashedWheelTimer.this.wheel[idx];//获取对应的槽
    this.transferTimeoutsToBuckets();//将添加到任务队列中的任务添加到时间轮中的槽中
    bucket.expireTimeouts(deadline);//处理时间到了的任务
    ++this.tick;//移动槽位
    }
    } while(HashedWheelTimer.WORKER_STATE_UPDATER.get(HashedWheelTimer.this) == 1);
//...             }
```

通过代码查阅，整体逻辑简单清晰：等待时间-》处理取消的任务-》队列中的任务入槽-》处理执行的任务。

主要看下这三个问题：

1. 等待时间是如何计算的，这个跟时间精度相关
2. 队列中的任务如何入槽的（对应上面的疑问）
3. 任务如何执行的

等待时间是如何计算的？worker.waitForNextTick

```
    private long waitForNextTick() {
    long deadline = HashedWheelTimer.this.tickDuration * (this.tick + 1L);//根据时间精度，算出需要下一次的检查时间
    while(true) {
    long currentTime = System.nanoTime() - HashedWheelTimer.this.startTime;
    long sleepTimeMs = (deadline - currentTime + 999999L) / 1000000L;
    if (sleepTimeMs <= 0L) {//不用睡了，时间已经到了，直接执行
    if (currentTime == -9223372036854775808L) {//溢出了兜底？
    return -9223372036854775807L;
    }

    return currentTime;
    }
    if (PlatformDependent.isWindows()) {//windows 下的 bug，sleep 的时间需是 10 的整数倍
    sleepTimeMs = sleepTimeMs / 10L * 10L;
    }
    try {
    Thread.sleep(sleepTimeMs);//等待时间的到来
    } catch (InterruptedException var8) {
    if (HashedWheelTimer.WORKER_STATE_UPDATER.get(HashedWheelTimer.this) == 2) {
    return -9223372036854775808L;
            }
        }
      }
    }
```

就是通过 tickDuration 和此时已经移动的 tick 算出下一次需要检查的时间，如果时间未到就 sleep。

队列中的任务如何入槽的？worker.transferTimeoutsToBuckets

```
     private void transferTimeoutsToBuckets() {
    for(int i = 0; i < 100000; ++i) {//设置了一次性处理 10w 个任务入槽，可能考虑这块处理太多会影响后续任务的处理？
    HashedWheelTimer.HashedWheelTimeout timeout = (HashedWheelTimer.HashedWheelTimeout)HashedWheelTimer.this.timeouts.poll();//从队列中拿出任务
    if (timeout == null) {
```

```
        break;
        }
      if (timeout.state() != 1) {
        long calculated = timeout.deadline / HashedWheelTimer.this.tickDuration;
        timeout.remainingRounds = (calculated - this.tick) /
(long)HashedWheelTimer.this.wheel.length;//计算轮数
        long ticks = Math.max(calculated, this.tick);//如果时间已经过了，放到当前即将执行的槽位中
        int stopIndex = (int)(ticks & (long)HashedWheelTimer.this.mask);//计算槽位下标
        HashedWheelTimer.HashedWheelBucket bucket = HashedWheelTimer.this.wheel[stopIndex];
        bucket.addTimeout(timeout);//入槽
            }
          }
        }
```

任务如何执行的？hashedWheelBucket.expireTimeouts

```
      public void expireTimeouts(long deadline) {
    HashedWheelTimer.HashedWheelTimeout next;
    for(HashedWheelTimer.HashedWheelTimeout timeout = this.head; timeout != null; timeout = next)
{
    //循环链表
        next = timeout.next;
        if (timeout.remainingRounds <= 0L) {//处理轮数为 0 的任务
        next = this.remove(timeout);
        if (timeout.deadline > deadline) {
          throw new IllegalStateException(String.format("timeout.deadline (%d) > deadline (%d)",
timeout.deadline, deadline));
          }
          timeout.expire();//过期任务，就是执行任务的 run 防范
          } else if (timeout.isCancelled()) {
          next = this.remove(timeout);//任务取消了
          } else {
            --timeout.remainingRounds;//轮数减一
                }
              }
            }
```

就是通过轮数和时间双重判断，执行任务。

小结：

Netty 中时间轮算法是基于轮次的时间轮算法实现，通过启动一个工作线程，根据时间精度 TickDuration，移动指针找到槽位，根据轮次+时间来判断是否是需要处理的任务。

不足之处：

1. 时间轮的推进是根据时间精度 TickDuration 来固定推进的，如果槽位中无任务，也需要移动指针，会造成无效的时间轮推进，比如 TickDuration 为 1 秒，此时就一个延迟 500 秒的任务，那就是有 499 次无用的推进。
2. 任务的执行都是同一个工作线程处理的，并且工作线程的除了处理执行到时的任务还做了其他操作，因此任务不一定会被精准的执行，而且任务的执行如果不是新起一个线程执行，那么耗时的任务会阻塞下个任务的执行。

优势就是时间精度可控，并且增删任务的时间复杂度都是 O(1)

多层次时间轮算法-Kafka 案例

去 kafka 官网下载源码(http://kafka.apache.org/downloads)，在 kafka.utils.timer 中找到基于 scala 实现的时间轮。

主要有 Timer 接口--SystemTimer 实现了该接口，是核心的调度逻辑；TimingWheel--时间轮数据结构的实现；TimerTaskList--槽位中的链表；TimerTaskEntry--链表元素，TimerTask 的包装类，在 TimerTaskList 中定义；TimerTask--继承 Runnable,实际的任务执行线程。

DOWNLOAD

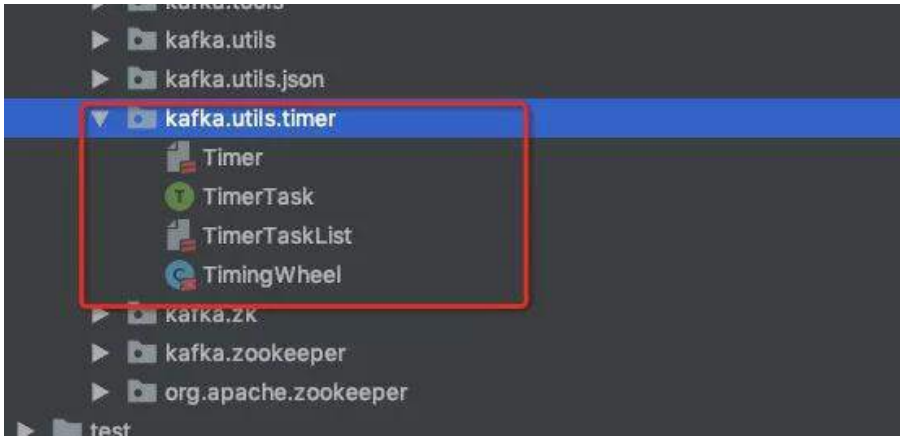
2.6.0 is the latest release. The current stable version is 2.6.0.

You can verify your download by following these procedures and using these KEYS.

2.6.0

- Released Aug 3, 2020
- Release Notes
- Source download: kafka-2.6.0-src.tgz (asc, sha512)
- Binary downloads:
  - Scala 2.12 - kafka_2.12-2.6.0.tgz (asc, sha512)
  - Scala 2.13 - kafka_2.13-2.6.0.tgz (asc, sha512)

We build for multiple versions of Scala. This only matters if you are using Scala and you want a
Otherwise any version should work (2.13 is recommended).



老规矩，我们从添加任务、任务调度（工作线程）两个切入点看一下整体逻辑，添加任务逻辑在 TimingWheel 中，工作线程在 SystemTimer 中。

TimingWheel--时间轮数据结构的核心实现

```
    @nonthreadsafe
    private[timer] class TimingWheel(tickMs: Long, wheelSize: Int, startMs: Long, taskCounter:
AtomicInteger, queue: DelayQueue[TimerTaskList]) {
    //queue 是一个 TimerTaskList 的延迟队列,每个槽的 TimerTaskList 都被加到这个延迟队列中,
expireTime 最小的槽会排在队列的最前面,此处要注意，这是 kafka 实现的一个特殊地方
```

```
    private[this] val interval = tickMs * wheelSize //该层级时间轮的时间跨度
    private[this] val buckets = Array.tabulate[TimerTaskList](wheelSize) { _ => new
TimerTaskList(taskCounter) }

    private[this] var currentTime = startMs - (startMs % tickMs) //起始时间(startMs)都设置为创建此层时
间轮时前面一轮的 currentTime，每一层的 currentTime 都必须是 tickMs 的整数倍
    @volatile private[this] var overflowWheel: TimingWheel = null

  //当时间溢出时，需要新增上一级时间轮，通过 overflowWheel 引用上一级侧时间轮
    private[this] def addOverflowWheel(): Unit = {
    synchronized {
     if (overflowWheel == null) {
          overflowWheel = new TimingWheel(
          tickMs = interval,
          wheelSize = wheelSize,
          startMs = currentTime,
          taskCounter = taskCounter,
          queue
       )
       }
    }
    }

    //添加任务
    def add(timerTaskEntry: TimerTaskEntry): Boolean = {
    val expiration = timerTaskEntry.expirationMs//获取任务延迟时间
    if (timerTaskEntry.cancelled) {
    false//取消
    } else if (expiration < currentTime + tickMs) {
    false         // 过期
    } else if (expiration < currentTime + interval) {         /

    /判断当前时间轮所能表示的时间范围是否可以容纳该任务
    //计算槽位
    val virtualId = expiration / tickMs
    val bucket = buckets((virtualId % wheelSize.toLong).toInt)
    bucket.add(timerTaskEntry)//将任务添加到槽位链表中
    if (bucket.setExpiration(virtualId * tickMs)) {//设置 expiretime
    //将槽位的 TimerTaskList 添加到延迟队列中
    queue.offer(bucket)
    }
```

```
    true
    } else {
    //到此处，说明时间溢出了，应该讲任务丢给父时间轮处理
    if (overflowWheel == null) addOverflowWheel()//判断上一级时间轮是否存在，不存在创建
    overflowWheel.add(timerTaskEntry)//调用上一级时间轮的添加任务方法，递归处理，直至添加成功
      }
    }

    //时间轮的推进
    def advanceClock(timeMs: Long): Unit = {
    if (timeMs >= currentTime + tickMs) {          currentTime = timeMs - (timeMs % tickMs)
    //调整当前时间
    if (overflowWheel != null) overflowWheel.advanceClock(currentTime)//递归推进上级时间轮
      }
    }
```

这是 kafka 对于时间轮最核心的实现部分，包含时间轮的数据结构、添加任务、时间溢出(添加上一级时间轮)、时间轮推进四个核心部分。大的逻辑是添加任务-》是否时间溢出？-》溢出时添加上一级时间轮，并调用上一级时间轮的添加任务方法 -》未溢出，直接添加到槽位 -》递归处理。所以时间轮的数据结构、时间溢出都通过添加任务的逻辑串联了起来。而时间轮推进方法主要由工作线程 SystemTimer 调用。

SystemTimer--核心的调度逻辑

```
    trait Timer {
    def add(timerTask: TimerTask): Unit
    def advanceClock(timeoutMs: Long): Boolean
    def size: Int
    def shutdown(): Unit
    }
    @threadsafe
    class SystemTimer(executorName: String,
      tickMs: Long = 1,
      wheelSize: Int = 20,
      startMs: Long = Time.SYSTEM.hiResClockMs) extends Timer {

      // timeout timer
      private[this] val taskExecutor = Executors.newFixedThreadPool(1,
```

```
        (runnable: Runnable) => KafkaThread.nonDaemon("executor-" + executorName, runnable))
        private[this] val delayQueue = new DelayQueue[TimerTaskList]()//延迟队列，会传递到每个层级的
时间轮中
        private[this] val taskCounter = new AtomicInteger(0)
        private[this] val timingWheel = new TimingWheel(
          tickMs = tickMs,
          wheelSize = wheelSize,
          startMs = startMs,
          taskCounter = taskCounter,
          delayQueue
          )
          //...
          //时间轮的推进
    def advanceClock(timeoutMs: Long): Boolean = {
    var bucket = delayQueue.poll(timeoutMs, TimeUnit.MILLISECONDS)//从延迟队列获取槽
    if (bucket != null) {
    writeLock.lock()
    try {
    while (bucket != null) {
    timingWheel.advanceClock(bucket.getExpiration())//时间轮的推进，具体看上面 timingwheel 中的
advanceClock 方法
    bucket.flush(reinsert)//把槽中所有任务都取出来，并重新 add 一遍，此时因为任务到期，并不会真正
add 进去，而是调用线程池运行任务，具体看下面的 reinsert 和 TimingWheel.add 方法
    bucket = delayQueue.poll()//获取下一个槽
    }
    } finally {
    writeLock.unlock()
    }
    true
    } else {
    false
    }
    }
    private[this] val reinsert = (timerTaskEntry: TimerTaskEntry) => addTimerTaskEntry(timerTaskEntry)
    private def addTimerTaskEntry(timerTaskEntry: TimerTaskEntry): Unit = {
    if (!timingWheel.add(timerTaskEntry)) {//
    //如果到执行时间或者取消会走到这
    if (!timerTaskEntry.cancelled)//到执行时间，执行任务
    taskExecutor.submit(timerTaskEntry.timerTask)
            }
    }//...}
```

kafka 会启动一个线程去推动时间轮的转动，实现原理就是通过 queue.poll()取出放在最前面的槽的 TimerTaskList，在时间轮添加任务时，所有有任务元素的 TimerTaskList 都会被添加到 queue 中。由于 queue 是一个延迟队列，如果队列中的 expireTime 没有到达，该操作会阻塞，等待 expireTime 时间到达。如果 poll 取到了 TimerTaskList，说明该槽里面的任务时间到达，会先推进时间轮的转动变更为当前时间，同时将到期的槽的所有任务都取出来，并通过 TimingWheel 重新 add 一遍，此时因为任务到期，并不会真正 add 进去，而是调用线程池运行任务，具体看下面的 SystemTimer.reinsert 和 TimingWheel.add 方法。

在前面分析 DelayQueue 时，DelayQueue 明显不适合 Kafka 这种高性能要求的定时任务，为何这里还要引入 DelayQueue 呢？注意对于定时任务项 TimerTaskEntry 插入和删除操作而言，TimingWheel 时间复杂度为 O(1)，性能高出 DelayQueue 很多，如果直接将 TimerTaskEntry 插入 DelayQueue 中，那么性能显然难以支撑，就算我们根据一定的规则将若干 TimerTaskEntry 划分到 TimerTaskList 这个组中，然后再将 TimerTaskList 插入到 DelayQueue 中，试想下如果这个 TimerTaskList 中又要多添加一个 TimerTaskEntry 该如何处理？对于 DelayQueue 而言，这类操作显然变得力不从心。

我们分析 Netty 中时间轮的推进，主要是通过固定的时间间隔扫描槽位，有可能槽位上没有任务，所以会有空推进的情况。Kafka 用 DelayQueue 保存每个槽，通过每个槽的过期时间排序，这样拥有最早需要执行任务的槽会被优先获取。如果时候未到，那么 delayQueue.poll 就会阻塞着，这样就不会有空推进的情况发生,同时呢，任务组织结构仍由时间轮组织，也兼顾了任务插入、删除操作的高性能。

小结：

Kafka 中时间轮算法是基于多层次的时间轮算法实现，并且是按需创建时间轮，采用任务的绝对时间来判断延期，空间换时间的思想，用 DelayQueue 存放每个槽，并以每个槽的过期时间排序，通过 delayQueue.poll 阻塞式进行时间轮的推进，杜绝了空推进的问题。

总结

Timer、DelayQueue 和 ScheduledThreadPool，它们都是基于优先队列实现的，O(logn) 的时间复杂度在任务数多的情况下频繁的插入、删除操作有性能问题，因此适合于任务数不多的情况。

1. Timer 是单线程的会有任务阻塞的风险,并且对异常没有做处理,一个任务出错 Timer 就挂了。
2. ScheduledThreadPool 相比于 Timer 引入了线程池，并且线程池对异常做了处理，使得任务之间不会有影响。
3. Timer 和 ScheduledThreadPool 可以周期性执行任务，DelayQueue 就是个具有优先级的阻塞队列，需要配合外部的工作线程使用。

时间轮的实现类似钟表的运作方式，它的任务插入和删除时间复杂度都为 O(1)，相对而言时间轮更适合任务数很大的延时场景。

1. 对于延迟超过时间轮所能表示的范围有两种处理方式，一是通过增加一个字段-轮数（Netty），二是多层次时间轮（Kakfa）。
2. 相比而言 Netty 的实现会有空推进的问题，而 Kafka 采用 DelayQueue 来保存有任务数据的槽，利用空间换时间的思想解决了空推进的问题。

PS:对时间轮算法有了初步的学习、理解，后面继续学习实现"基于时间轮算法的分布式任务调度系统"!

详解 Graph Embedding 经典方法：算法原理、代码实现与应用样例

作者 | 浅梦
出品 | 淘系技术

导读

我们都知道在数据结构中，图是一种基础且常用的结构。现实世界中许多场景可以抽象为一种图结构，如社交网络，交通网络，电商网站中用户与物品的关系等。以躺平 APP 社区为例，它是"躺平"这个大生态中生活方式分享社区，分享生活分享家，努力打造社区交流、好物推荐与居家指南。用户在社区的所有行为：发布、点击、点赞、评论收藏等都可以抽象为网络关系图。因此 Graph Embedding 技术非常自然地成为学习社区中用户与内容的 embedding 的一项关键技术。

目前落地的模型大致两类：直接优化节点的浅层网络模型和基于 GNN 的深层网络模型。前者包括基于用户行为理解内容，学习内容向量表征的 item2vec,用于扩充 i2i 召回；同时学习用户与内容向量表征的异构网络表示学习模型 metapath2vec，用于提高内容召回的多样性；以群体行为代替个体行为的 userCluster2vec，缓解新用户冷启动问题。后者包括采用邻域聚合的思想，同时融入节点属性信息，通过多层节点聚合使得网络中的拓扑结构能够有效捕捉获的 graphSage，以及将 attention 机制运用邻域信息聚合阶段的 GAT，对用户与内容向量表征进行更加细致化学习。

这里先看下 Graph Embedding 的相关内容。

Graph Embedding 技术将图中的节点以低维稠密向量的形式进行表达，要求在原始图中相似（ 不同的方法对相似的定义不同 ）的节点其在低维表达空间也接近。得到的表达向量可以用来进行下游任务，如节点分类，链接预测，可视化或重构原始图等。

DeepWalk

虽然 DeepWalk 是 KDD 2014 的工作，但却是我们了解 Graph Embedding 无法绕过的一个方法。

我们都知道在 NLP 任务中，word2vec 是一种常用的 word embedding 方法，word2vec 通过语料库中的句子序列来描述词与词的共现关系，进而学习到词语的向量表示。

DeepWalk 的思想类似 word2vec，使用图中节点与节点的共现关系来学习节点的向量表示。那么关键的问题就是如何来描述节点与节点的共现关系，DeepWalk 给出的方法是使用随机游走 (RandomWalk) 的方式在图中进行节点采样。

RandomWalk 是一种可重复访问已访问节点的深度优先遍历算法。给定当前访问起始节点，从其邻居中随机采样节点作为下一个访问节点，重复此过程，直到访问序列长度满足预设条件。

获取足够数量的节点访问序列后，使用 skip-gram model 进行向量学习。

Algorithm 1 DeepWalk(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
  $\mathcal{O} = \text{Shuffle}(V)$
  for each $v_i \in \mathcal{O}$ **do**
    $\mathcal{W}_{v_i} = RandomWalk(G, v_i, t)$
    SkipGram$(\Phi, \mathcal{W}_{v_i}, w)$
  end for
end for

DeepWalk 核心代码

DeepWalk 算法主要包括两个步骤，第一步为随机游走采样节点序列，第二步为使用 skip-gram modelword2vec 学习表达向量。

- 构建同构网络，从网络中的每个节点开始分别进行 Random Walk 采样，得到局部相关联的训练数据。
- 对采样数据进行 SkipGram 训练，将离散的网络节点表示成向量化，最大化节点共现，使用 Hierarchical Softmax 来做超大规模分类的分类器。

Random Walk

我们可以通过并行的方式加速路径采样，在采用多进程进行加速时，相比于开一个进程池让每次外层循环启动一个进程，我们采用固定为每个进程分配指定数量的 num_walks 的方式，这样可以最大限度减少进程频繁创建与销毁的时间开销。

deepwalk_walk 方法对应上一节伪代码中第 6 行，_simulate_walks 对应伪代码中第 3 行开始的外层循环。最后的 Parallel 为多进程并行时的任务分配操作。

```
def deepwalk_walk(self, walk_length, start_node):

walk = [start_node]

while len(walk) < walk_length:
  cur = walk[-1]
  cur_nbrs = list(self.G.neighbors(cur))
  if len(cur_nbrs) > 0:
    walk.append(random.choice(cur_nbrs))
    else:
      break
    return walk

    def _simulate_walks(self, nodes, num_walks, walk_length,):
      walks = []
      for _ in range(num_walks):
      random.shuffle(nodes)
      for v in nodes:
      walks.append(self.deepwalk_walk(alk_length=walk_length, start_node=v))
      return walks

      results = Parallel(n_jobs=workers, verbose=verbose, )(
```

```
        delayed(self._simulate_walks)(nodes, num, walk_length) for num in
        partition_num(num_walks, workers))

        walks = list(itertools.chain(*results))
```

Word2vec

这里就偷个懒直接用 gensim 里的 Word2Vec 了。

```
    from gensim.models import Word2Vec
    w2v_model = Word2Vec(walks,sg=1,hs=1)
```

DeepWalk 应用

这里简单的用 DeepWalk 算法在 wiki 数据集上进行节点分类任务和可视化任务。wiki 数据集包含 2,405 个网页和 17,981 条网页之间的链接关系，以及每个网页的所属类别。

本例中的训练，评测和可视化的完整代码在 git 仓库中。

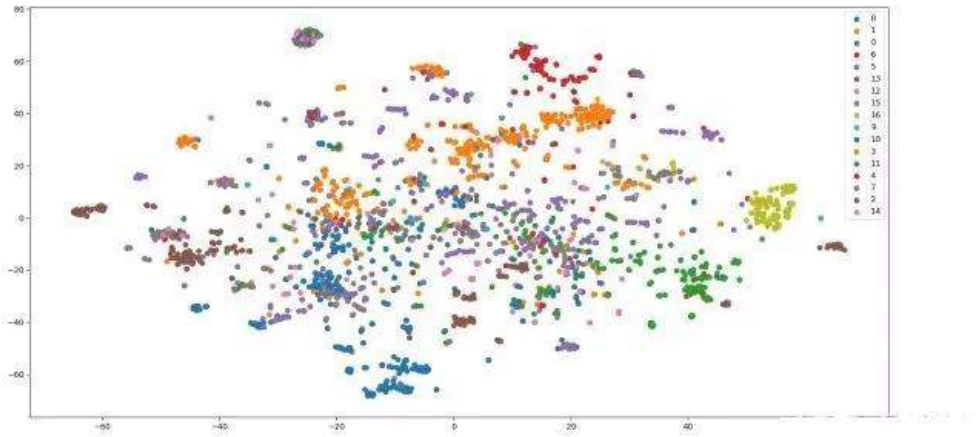
```
    G =
nx.read_edgelist('../data/wiki/Wiki_edgelist.txt',create_using=nx.DiGraph(),nodetype=None,data=[('weight',int)])

    model = DeepWalk(G,walk_length=10,num_walks=80,workers=1)
    model.train(window_size=5,iter=3)embeddings =
    model.get_embeddings()

    evaluate_embeddings(embeddings)
    plot_embeddings(embeddings)
```

分类任务结果

micro-F1 : 0.6674

macro-F1 : 0.5768

可视化结果



LINE

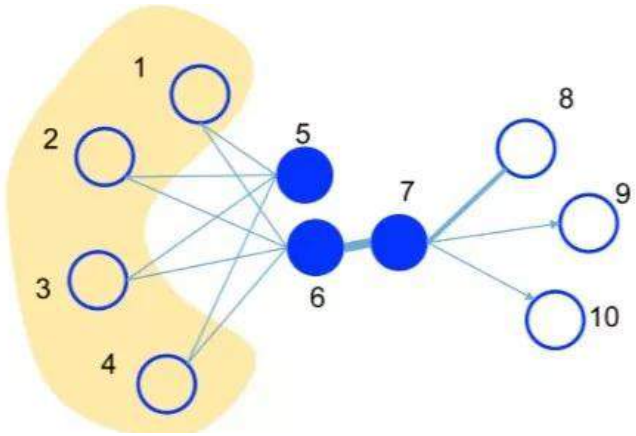
之前介绍过 DeepWalk，DeepWalk 使用 DFS 随机游走在图中进行节点采样，使用 word2vec 在采样的序列学习图中节点的向量表示。

LINE 也是一种基于邻域相似假设的方法，只不过与 DeepWalk 使用 DFS 构造邻域不同的是，LINE 可以看作是一种使用 BFS 构造邻域的算法。此外，LINE 还可以应用在带权图中(DeepWalk 仅能用于无权图)。

之前还提到不同的 graph embedding 方法的一个主要区别是对图中顶点之间的相似度的定义不同，所以先看一下 LINE 对于相似度的定义。

LINE 算法原理

1. 一种新的相似度定义



first-order proximity

1 阶相似度用于描述图中成对顶点之间的局部相似度，形式化描述为若 u，v 之间存在直连边，则边权 w_{uv} 即为两个顶点的相似度，若不存在直连边，则 1 阶相似度为 0。如上图，6 和 7 之间存在直连边，且边权较大，则认为两者相似且 1 阶相似度较高，而 5 和 6 之间不存在直连边，则两者间 1 阶相似度为 0。

second-order proximity

仅有 1 阶相似度就够了吗？显然不够，如上图，虽然 5 和 6 之间不存在直连边，但是他们有很多相同的邻居顶点(1,2,3,4)，这其实也可以表明 5 和 6 是相似的，而 2 阶相似度就是用来描述这种关系的。形式化定义为，令 $p_u = (w_{u,1}, \ldots, w_{u,|V|})$ 表示顶点 u 与所有其他顶点间的 1 阶相似度，则 u 与 v 的 2 阶相似度可以通过 p_u 和 p_v 的相似度表示。若 u 与 v 之间不存在相同的邻居顶点，则 2 阶相似度为 0。

2. 优化目标

1st-order

对于每一条无向边 (i,j)，定义顶点 v_i 和 v_j 之间的联合概率为：

$p_1(v_i, v_j) = \frac{1}{1+\exp(-\vec{u}_i^T \cdot \vec{u}_j)}$，$\vec{u_i}$ 为顶点 v_i 的低维向量表示。（可以看作一个内积模型，计算两个 item 之间的匹配程度）

同时定义经验分布：

$$\hat{p}_1 = \frac{w_{ij}}{W}, W = \sum_{(i,j)\in E} w_{ij},$$

优化目标为最小化：

$$O_1 = d(\hat{p}_1(\cdot,\cdot), p_1(\cdot,\cdot))$$

$d(\cdot,\cdot)$是两个分布的距离，常用的衡量两个概率分布差异的指标为 KL 散度，使用 KL 散度并忽略常数项后有：

$$O_1 = -\sum_{(i,j)\in E} w_{ij} \log p_1(v_i, v_j)$$

1st order 相似度只能用于无向图当中。

2nd-order

这里对于每个顶点维护两个 embedding 向量，一个是该顶点本身的表示向量，一个是该点作为其他顶点的上下文顶点时的表示向量。

对于有向边(i,j)，定义给定顶点v_i条件下，产生上下文(邻居)顶点v_j的概率为：

$p_2(v_j|v_i) = \frac{\exp(\vec{c}_j^T \cdot \vec{u}_i)}{\sum_{k=1}^{|V|} \exp(\vec{c}_k^T \cdot \vec{u}_i)}$，其中$|V|$为上下文顶点的个数。

优化目标为：

$O_2 = \sum_{i \in V} \lambda_i d(\hat{p}_2(\cdot|v_i), p_2(\cdot|v_i))$，其中$\lambda_i$为控制节点重要性的因子，可以通过顶点的度数或者 PageRank 等方法估计得到。

经验分布定义为：$\hat{p}_2(v_j|v_i) = \frac{w_{ij}}{d_i}$，$w_{ij}$是边$(i,j)$的边权，$d_i$是顶点$v_i$的出度，对于带权图，$d_i = \sum_{k \in N(I)} W_{ik}$使用 KL 散度并设$\lambda_i = d_i$，忽略常数项，有$O_2 = -\sum_{(i,j) \in E} w_{ij} \log p_2(v_j|v_i)$

3. 优化技巧

Negative sampling

由于计算 2 阶相似度时，softmax 函数的分母计算需要遍历所有顶点，这是非常低效的，论文采用了负采样优化的技巧，目标函数变为：

$\log \sigma(\vec{c}_j^T \cdot \vec{u}_i) + \sum_{i=1}^{K} E_{v_n \sim P_n(v)} [-\log \sigma(\vec{c}_n^T \cdot \vec{u}_i)]$，$K$是负边的个数。论文使用$P_n(v) \propto d_v^{3/4}$，$d_v$是顶点$v$的出度。

Edge Sampling

注意到我们的目标函数在 log 之前还有一个权重系数w_{ij}，在使用梯度下降方法优化参数时，w_{ij}会直接乘在梯度上。如果图中的边权方差很大，则很难选择一个合适的学习率。若使用较大的学习率那么对于较大的边权可能会引起梯度爆炸，较小的学习率对于较小的边权则会导致梯度过小。

对于上述问题，如果所有边权相同，那么选择一个合适的学习率会变得容易。这里采用了将带权边拆分为等权边的一种方法，假如一个权重为w的边，则拆分后为w个权重为 1 的边。

这样可以解决学习率选择的问题，但是由于边数的增长，存储的需求也会增加。

另一种方法则是从原始的带权边中进行采样，每条边被采样的概率正比于原始图中边的权重，这样既解决了学习率的问题，又没有带来过多的存储开销。这里的采样算法使用的是 Alias 算法，Alias 是一种$O(1)$时间复杂度的离散事件抽样算法。具体内容可以参考：

Alias Method:时间复杂度 O(1)的离散采样方法

4. 其他问题

低度数顶点

对于一些顶点由于其邻接点非常少会导致 embedding 向量的学习不充分，论文提到可以利用邻居的邻居构造样本进行学习，这里也暴露出 LINE 方法仅考虑一阶和二阶相似性，对高阶信息的利用不足。

新加入顶点

对于新加入图的顶点v_i，若该顶点与图中顶点存在边相连，我们只需要固定模型的其他参数，优化如下两个目标之一即可：

$$-\sum_{j\in N(i)} w_{ji}\log p_1(v_j,v_i) \quad 或 \quad -\sum_{j\in N(i)} w_{ji}\log p_1(v_j|v_i)$$

若不存在边相连，则需要利用一些 side info，留到后续工作研究。

LINE 核心代码

1. 模型和损失函数定义

LINE 使用梯度下降的方法进行优化，直接使用 tensorflow 进行实现，就可以不用人工写参数更新的逻辑了~

这里的 实现中把 1 阶和 2 阶的方法融合到一起了，可以通过超参数 order 控制是分开优化还是联合优化，论文推荐分开优化。

首先输入就是两个顶点的编号，然后分别拿到各自对应的 embedding 向量，最后输出内积的结果。真实 label 定义为 1 或者-1，通过模型输出的内积和 line_loss 就可以优化使用了负采样技巧的目标函数了~

```
    def line_loss(y_true, y_pred):
        return -K.mean(K.log(K.sigmoid(y_true*y_pred)))def create_model(numNodes, embedding_size,
order='second'):

        v_i = Input(shape=(1,))
        v_j = Input(shape=(1,))

        first_emb = Embedding(numNodes, embedding_size, name='first_emb')
        second_emb = Embedding(numNodes, embedding_size, name='second_emb')
        context_emb = Embedding(numNodes, embedding_size, name='context_emb')

        v_i_emb = first_emb(v_i)
        v_j_emb = first_emb(v_j)

        v_i_emb_second = second_emb(v_i)
        v_j_context_emb = context_emb(v_j)

        first = Lambda(lambda x: tf.reduce_sum(
        x[0]*x[1], axis=-1, keep_dims=False), name='first_order')([v_i_emb, v_j_emb])
        second = Lambda(lambda x: tf.reduce_sum(

        x[0]*x[1], axis=-1, keep_dims=False), name='second_order')([v_i_emb_second, v_j_context_emb])

        if order == 'first':

            output_list = [first]
        elif order == 'second':
            output_list = [second]
```

```
else:
        output_list = [first, second]

        model = Model(inputs=[v_i, v_j], outputs=output_list)
```

2. 顶点负采样和边采样

下面的函数功能是创建顶点负采样和边采样需要的采样表。中规中矩，主要就是做一些预处理，然后创建 alias 算法需要的两个表。

```
def _gen_sampling_table(self):

# create sampling table for vertex
power = 0.75
numNodes = self.node_size
node_degree = np.zeros(numNodes)    # out degree
node2idx = self.node2idx

   for edge in self.graph.edges():
node_degree[node2idx[edge[0]]
] += self.graph[edge[0]][edge[1]].get('weight', 1.0)

 total_sum = sum([math.pow(node_degree[i], power)
   for i in range(numNodes)])
norm_prob = [float(math.pow(node_degree[j], power)) /
total_sum for j in range(numNodes)]

self.node_accept, self.node_alias = create_alias_table(norm_prob)

# create sampling table for edge
numEdges = self.graph.number_of_edges()
total_sum = sum([self.graph[edge[0]][edge[1]].get('weight', 1.0)
   for edge in self.graph.edges()])
norm_prob = [self.graph[edge[0]][edge[1]].get('weight', 1.0) *
numEdges / total_sum for edge in self.graph.edges()]
self.edge_accept, self.edge_alias = create_alias_table(norm_prob)
```

LINE 应用

和之前一样，还是用 LINE 在 wiki 数据集上进行节点分类任务和可视化任务。wiki 数据集包含 2,405 个网页和 17,981 条网页之间的链接关系，以及每个网页的所属类别。由于 1 阶相似度仅能应用于无向图中，所以本例中仅使用 2 阶相似度。

本例中的训练，评测和可视化的完整代码在 git 仓库中。

```
    G =
nx.read_edgelist('../data/wiki/Wiki_edgelist.txt',create_using=nx.DiGraph(),nodetype=None,data=[('weight',int)])

    model = LINE(G,embedding_size=128,order='second')
    model.train(batch_size=1024,epochs=50,verbose=2)
    embeddings = model.get_embeddings()

    evaluate_embeddings(embeddings)
    plot_embeddings(embeddings)
```

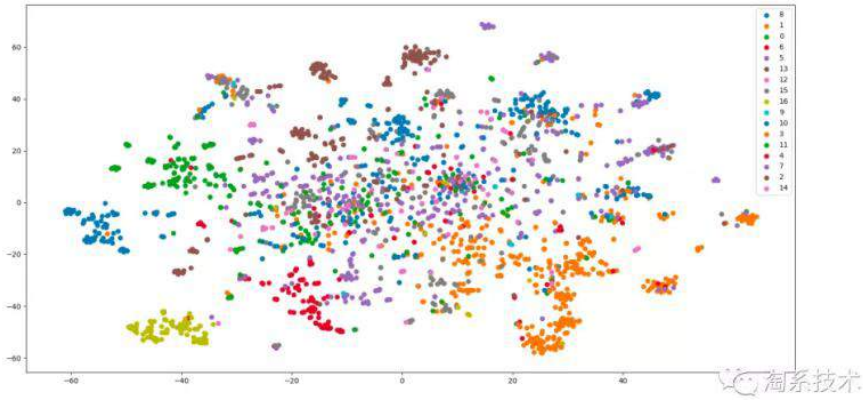
分类任务结果

micro-F1: 0.6403

macro-F1:0.5286

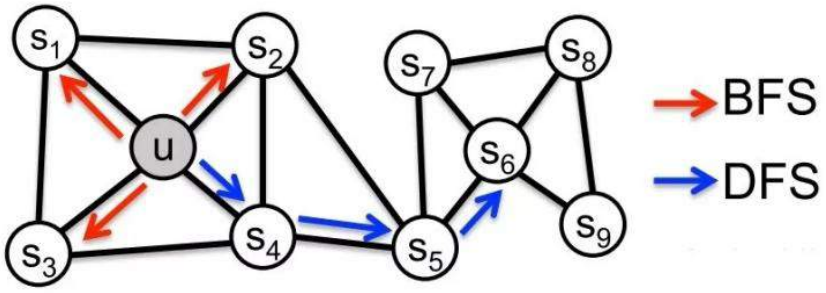
结果有一定随机性，可以多运行几次，或者稍微调整 epoch 个数。

可视化结果



node2vec

前面介绍过基于 DFS 邻域的 DeepWalk 和基于 BFS 邻域的 LINE。



node2vec 是一种综合考虑 DFS 邻域和 BFS 邻域的 graph embedding 方法。简单来说，可以看作是 deepwalk 的一种扩展，是结合了 DFS 和 BFS 随机游走的 deepwalk。

node2vec 算法原理

1. 优化目标

设 $f(u)$是将顶点 u 映射为 embedding 向量的映射函数,对于图中每个顶点u，定义 $N_S(u)$为通过采样策略 S采样出的顶点 u 的近邻顶点集合。

node2vec 优化的目标是给定每个顶点条件下，令其近邻顶点（如何定义近邻顶点很重要）出现的概率最大。

$$max_f \sum_{u \in V} \log Pr(N_S(U)|f(u))$$

为了将上述最优化问题可解，文章提出两个假设：

- 条件独立性假设

假设给定源顶点下，其近邻顶点出现的概率与近邻集合中其余顶点无关。

$$Pr(N_s(u)|f(u)) = \prod_{n_i \in N_s(u)} Pr(n_i|f(u))$$

- 特征空间对称性假设

这里是说一个顶点作为源顶点和作为近邻顶点的时候共享同一套 embedding 向量。(对比 LINE 中的 2 阶相似度，一个顶点作为源点和近邻点的时候是拥有不同的 embedding 向量的) 在这个假设下，上述条件概率公式可表示为：

$$Pr(n_i|f(u)) = \frac{\exp f(n_i) \cdot f(u)}{\sum_{v \in V} \exp f(v) \cdot f(u)}$$

根据以上两个假设条件，最终的目标函数表示为：

$$max_f \sum_{u \in V} [-\log Z_u + \sum_{n_i \in N_s(u)} f(n_i) \cdot f(u)]$$

由于归一化因子：

$Z_u = \sum_{n_i \in N_s(u)} \exp(f(n_i) \cdot f(u))$ 的计算代价高，所以采用负采样技术优化。

2. 顶点序列采样策略

node2vec 依然采用随机游走的方式获取顶点的近邻序列，不同的是 node2vec 采用的是一种有偏的随机游走。

给定当前顶点 v，访问下一个顶点 x 的概率为：

$$P(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases}$$

π_{vx} 是顶点 v 和顶点 x 之间的未归一化转移概率，Z 是归一化常数。

node2vec 引入两个超参数 p 和 q 来控制随机游走的策略，假设当前随机游走经过边 (t, v) 到达顶点 v 设 $\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$，$w_{vx}$ 是顶点 v 和 x 之间的边权。

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} = & \text{if } d_{tx} = 0 \\ 1 = & \text{if } d_{tx} = 1 \\ \frac{1}{q} = & \text{if } d_{tx} = 2 \end{cases}$$

d_{tx} 为顶点 t 和顶点 x 之间的最短路径距离。

下面讨论超参数 p 和 q 对游走策略的影响：

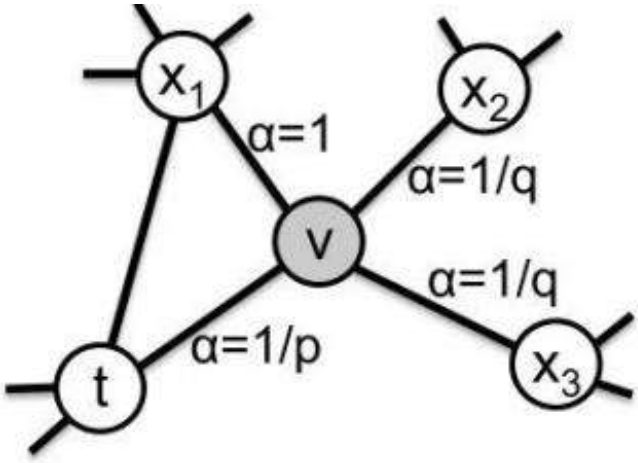
- Return parameter,p

参数 p 控制重复访问刚刚访问过的顶点的概率。注意到 p 仅作用于$d_{tx} = 0$的情况，而 $d_{tx} = 0$表示顶点 x 就是访问当前顶点 v之前刚刚访问过的顶点。那么若 p 较高，则访问刚刚访问过的顶点的概率会变低，反之变高。

- In-out papameter,q

q 控制着游走是向外还是向内，若$q > 1$ ，随机游走倾向于访问和 t 接近的顶点(偏向 BFS)。若$q > 1$ ，倾向于访问远离 t 的顶点(偏向 DFS)。

下面的图描述的是当从 t 访问到v 时，决定下一个访问顶点时每个顶点对应的$q > 1$ 。



3. 学习算法

采样完顶点序列后，剩下的步骤就和 deepwalk 一样了，用 word2vec 去学习顶点的 embedding 向量。值得注意的是 node2vecWalk 中不再是随机抽取邻接点，而是按概率抽取，node2vec 采用了 Alias 算法进行顶点采样。

Alias Method:时间复杂度 O(1)的离散采样方法

node2vec 核心代码

Algorithm 1 The *node2vec* algorithm.

LearnFeatures (Graph $G = (V, E, W)$, Dimensions d, Walks per node r, Walk length l, Context size k, Return p, In-out q)
  π = PreprocessModifiedWeights(G, p, q)
  $G' = (V, E, \pi)$
  Initialize $walks$ to Empty
  for $iter = 1$ **to** r **do**
    for all nodes $u \in V$ **do**
      $walk$ = node2vecWalk(G', u, l)
      Append $walk$ to $walks$
  f = StochasticGradientDescent(k, d, $walks$)
  return f

node2vecWalk (Graph $G' = (V, E, \pi)$, Start node u, Length l)
  Inititalize $walk$ to $[u]$
  for $walk_iter = 1$ **to** l **do**
    $curr = walk[-1]$
    V_{curr} = GetNeighbors($curr, G'$)
    s = AliasSample(V_{curr}, π)
    Append s to $walk$
  return $walk$

1. node2vecWalk

通过上面的伪代码可以看到，node2vec 和 deepwalk 非常类似，主要区别在于顶点序列的采样策略不同，所以这里我们主要关注 node2vecWalk 的实现。

由于采样时需要考虑前面 2 步访问过的顶点，所以当访问序列中只有 1 个顶点时，直接使用当前顶点和邻居顶点之间的边权作为采样依据。当序列多余 2 个顶点时，使用文章提到的有偏采样。

```
def node2vec_walk(self, walk_length, start_node):
G = self.G
alias_nodes = self.alias_nodes
alias_edges = self.alias_edges
walk = [start_node]
while len(walk) < walk_length:
  cur = walk[-1]
  cur_nbrs = list(G.neighbors(cur))
  if len(cur_nbrs) > 0:
  if len(walk) == 1:
  walk.append(cur_nbrs[alias_sample(alias_nodes[cur][0], alias_nodes[cur][1])])
  else:
```

```
    prev = walk[-2]
    edge = (prev, cur)
    next_node = cur_nbrs[alias_sample(alias_edges[edge][0],alias_edges[edge][1])]
    walk.append(next_node)
    else:
    break
     return walk
```

2. 构造采样表

preprocess_transition_probs 分别生成 alias_nodes 和 alias_edges，alias_nodes 存储着在每个顶点时决定下一次访问其邻接点时需要的 alias 表（不考虑当前顶点之前访问的顶点）。alias_edges 存储着在前一个访问顶点为 t，当前顶点为 v 时决定下一次访问哪个邻接点时需要的 alias 表。

get_alias_edge 方法返回的是在上一次访问顶点 t，当前访问顶点为v时到下一个顶点x的未归一化转移概率:

$$\pi_{vx} = \alpha_{pq}(t,x) \cdot w_{vx}$$

```
def get_alias_edge(self, t, v):
    G = self.G
    p = self.p
    q = self.q
    unnormalized_probs = []
    for x in G.neighbors(v):
        weight = G[v][x].get('weight', 1.0)# w_vx
        if x == t:# d_tx == 0
            unnormalized_probs.append(weight/p)
        elif G.has_edge(x, t):# d_tx == 1
            unnormalized_probs.append(weight)
        else:# d_tx == 2
            unnormalized_probs.append(weight/q)
    norm_const = sum(unnormalized_probs)
    normalized_probs = [float(u_prob)/norm_const for u_prob in unnormalized_probs]
    return create_alias_table(normalized_probs)def preprocess_transition_probs(self):
```

```
    G = self.G
    alias_nodes = {}
    for node in G.nodes():
    unnormalized_probs = [G[node][nbr].get('weight', 1.0) for nbr in G.neighbors(node)]
    norm_const = sum(unnormalized_probs)
    normalized_probs = [float(u_prob)/norm_const for u_prob in unnormalized_probs]
    alias_nodes[node] = create_alias_table(normalized_probs)
    alias_edges = {}
    for edge in G.edges():
    alias_edges[edge] = self.get_alias_edge(edge[0], edge[1])
    self.alias_nodes = alias_nodes
    self.alias_edges = alias_edges
    return
```

node2vec 应用

使用 node2vec 在 wiki 数据集上进行节点分类任务和可视化任务。wiki 数据集包含 2,405 个网页和 17,981 条网页之间的链接关系，以及每个网页的所属类别。通过简单的超参搜索，这里使用 p=0.25,q=4 的设置。

本例中的训练，评测和可视化的完整代码在 git 仓库中：

```
    G =
nx.read_edgelist('../data/wiki/Wiki_edgelist.txt',create_using=nx.DiGraph(),nodetype=None,data=[('weight',int)])

    model = Node2Vec(G,walk_length=10,num_walks=80,p=0.25,q=4,workers=1)
    model.train(window_size=5,iter=3)      embeddings = model.get_embeddings()

    evaluate_embeddings(embeddings)
    plot_embeddings(embeddings)
```

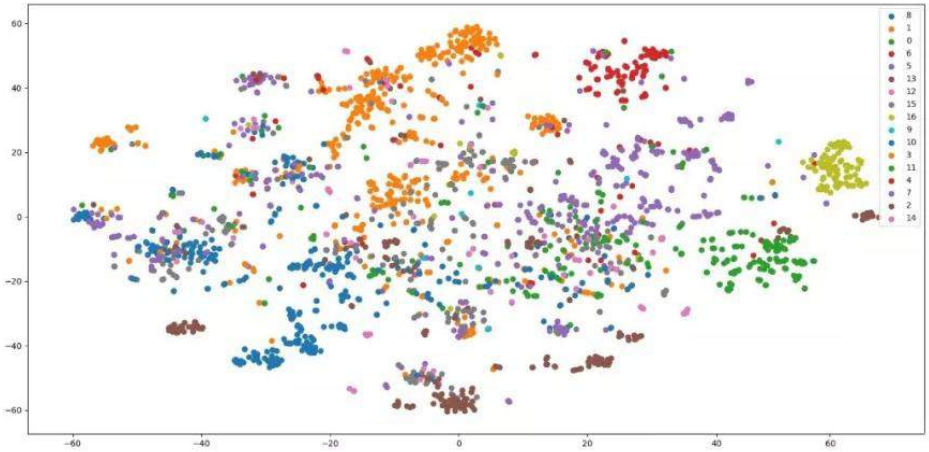
分类任务

micro-F1: 0.6757 macro-F1: 0.5917

这个结果相比于 DeepWalk 和 LINE 是有提升的。

可视化

这个结果相比于 DeepWalk 和 LINE 可以看到不同类别的分布更加分散了。



最后补充一个 node2vec 在业界的应用介绍：

学习遇上复杂网络：解析微信朋友圈 Lookalike 算法当机器

SDNE

SDNE（Structural Deep Network Embedding ）是和 node2vec 并列的工作，均发表在 2016 年的 KDD 会议中。可以看作是基于 LINE 的扩展，同时也是第一个将深度学习应用于网络表示学习中的方法。

SDNE 使用一个自动编码器结构来同时优化 1 阶和 2 阶相似度(LINE 是分别优化的)，学习得到的向量表示能够保留局部和全局结构，并且对稀疏网络具有鲁棒性。

SDNE 算法原理

相似度定义

SDNE 中的相似度定义和 LINE 是一样的。简单来说，1 阶相似度衡量的是相邻的两个顶点对之间相似性。2 阶相似度衡量的是，两个顶点他们的邻居集合的相似程度。

2 阶相似度优化目标

$$L_{2nd} = \sum_{i=1}^{n} ||\hat{x}_i - x_i||_2^2$$

这里我们使用图的邻接矩阵进行输入，对于第 i 个顶点，有 $x_i = s_i$，每一个 s_i 都包含了顶点 i 的邻居结构信息，所以这样的重构过程能够使得结构相似的顶点具有相似的 embedding 表示向量。

这里存在的一个问题是由于图的稀疏性，邻接矩阵 S 中的非零元素是远远少于零元素的，那么对于神经网络来说只要全部输出 0 也能取得一个不错的效果，这不是我们想要的。
文章给出的一个方法是使用带权损失函数，对于非零元素具有更高的惩罚系数。修正后的损失函数为:

$$L_{2nd} = \sum_{i=1}^{n} ||(\hat{x}_i - x_i) \odot \mathbf{b_i}||_2^2 = ||(\hat{X} - X) \odot B||_F^2$$

其中 $\odot$ 为逐元素积，$\mathbf{b_i} = \{b_{i,j}\}_{j=1}^{n}$，若 $s_{i,j} = 0$，则 $b_{i,j} = 1$，否则 $b_{i,j} = \beta > 1$

1 阶相似度优化目标

对于 1 阶相似度，损失函数定义如下:

$$L_{1st} = \sum_{i,j=1}^{n} s_{i,j} ||\mathbf{y_i}^{(K)} - \mathbf{y_j}^{(K)}||_2^2 = \sum_{i,j=1}^{n} s_{i,j} ||\mathbf{y_i} - \mathbf{y_j}||_2^2$$

该损失函数可以让图中的相邻的两个顶点对应的 embedding vector 在隐藏空间接近。
L_{1st} 还可以表示为:

$$L_{1st} = \sum_{i,j=1}^{n} ||\mathbf{y}_i - \mathbf{y}_j||_2^2 = 2tr(Y^T LY)$$

其中 L 是图对应的拉普拉斯矩阵，$L = D - S$，D 是图中顶点的度矩阵，S 是邻接矩阵，$D_{i,i} = \sum_j s_{i,j}$。

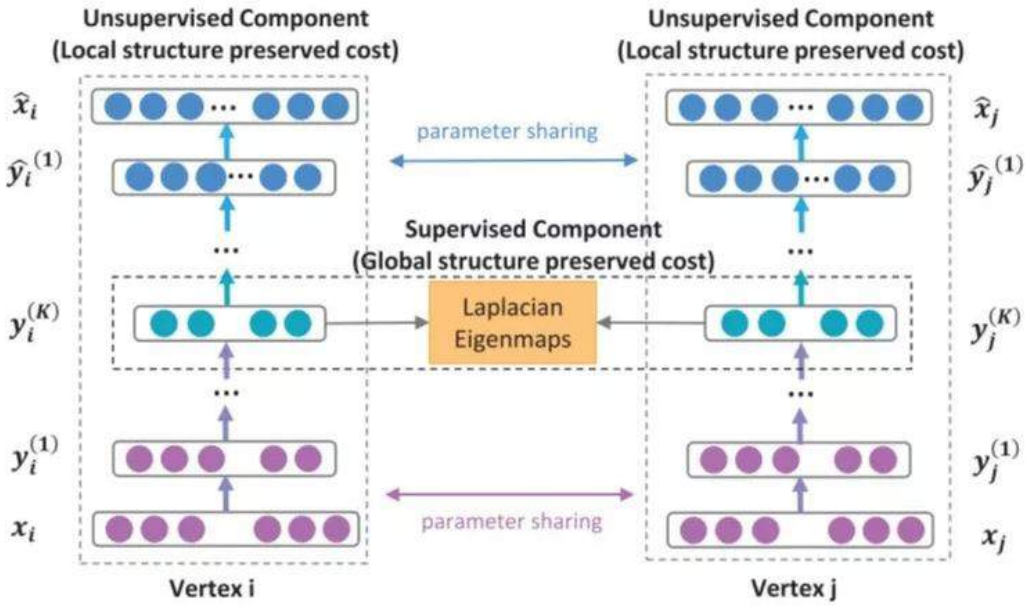
整体优化目标

联合优化的损失函数为:

$$L_{mix} = L_{2nd} + \alpha L_{1st} + \nu L_{reg}$$

L_{reg} 是正则化项，α 为控制 1 阶损失的参数，ν 为控制正则化项的参数。

模型结构



先看左边，是一个自动编码器的结构，输入输出分别是邻接矩阵和重构后的邻接矩阵。通过优化重构损失可以保留顶点的全局结构特性(论文的图画错了，上面应该是 Global structure preserved cost)。

再看中间一排，$y_i^{(K)}$ 就是我们需要的 embedding 向量，模型通过 1 阶损失函数使得邻接的顶点对应的 embedding 向量接近，从而保留顶点的局部结构特性(中间应该是 Local structure preserved cost)

实现

文章提出使用深度信念网络进行预训练来获得较好的参数初始化，这里我就偷个懒省略这个步骤了~

损失函数定义

l_2nd 是 2 阶相似度对应的损失函数，参数 beta 控制着非零元素的惩罚项系数。y_true 和 y_pred 分别是输入的邻接矩阵和网络重构出的邻接矩阵。

l_1st 是 1 阶相似度对应的损失函数，参数 alpha 控制着其在整体损失函数中的占比。

```
def l_2nd(beta):
def loss_2nd(y_true, y_pred):
b_ = np.ones_like(y_true)
b_[y_true != 0] = beta
x = K.square((y_true - y_pred) * b_)
t = K.sum(x, axis=-1, )
return K.mean(t) return loss_2nd

def l_1st(alpha):
def loss_1st(y_true, y_pred):
L = y_true
Y = y_pred
batch_size = tf.to_float(K.shape(L)[0])
return alpha * 2 * tf.linalg.trace(tf.matmul(tf.matmul(Y, L, transpose_a=True), Y)) / batch_size
```

```
return loss_1st
```

模型定义

create_model 函数创建 SDNE 模型，l1 和 l2 分别为模型的正则化项系数，模型的输入 A 为邻接矩阵，L 为拉普拉斯矩阵。输出 A_为重构后的邻接矩阵，Y 为顶点的 embedding 向量。

函数中两个 for 循环分别对应 encoder 和 decoder 结构。

```
def create_model(node_size, hidden_size=[256, 128], l1=1e-5, l2=1e-4):
A = Input(shape=(node_size,))
L = Input(shape=(None,))
fc = A
for i in range(len(hidden_size)):
if i == len(hidden_size) - 1:
fc = Dense(hidden_size[i], activation='relu',kernel_regularizer=l1_l2(l1, l2),name='1st')(fc)
else:
fc = Dense(hidden_size[i], activation='relu',kernel_regularizer=l1_l2(l1, l2))(fc)
Y = fc
for i in reversed(range(len(hidden_size) - 1)):
fc = Dense(hidden_size[i], activation='relu',kernel_regularizer=l1_l2(l1, l2))(fc)
A_ = Dense(node_size, 'relu', name='2nd')(fc)
model = Model(inputs=[A, L], outputs=[A_, Y])
return model
```

应用

使用 SDNE 在 wiki 数据集上进行节点分类任务和可视化任务(感兴趣的同学可以试试别的数据集，我比较懒就选了个很小的数据集)。wiki 数据集包含 2,405 个网页和 17,981 条网页之间的链接关系，以及每个网页的所属类别。

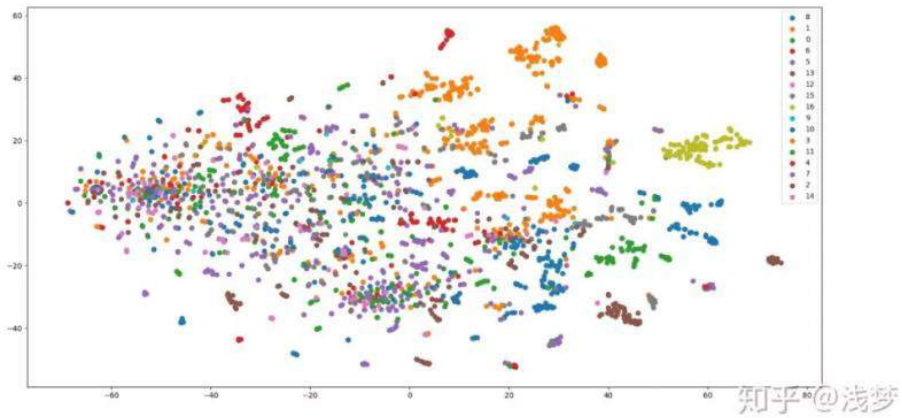
本例中的训练，评测和可视化的完整代码在 git 仓库中。

分类任务

micro-F1: 0.6341 macro-F1: 0.4962

可视化

这个貌似某些类别的点的向量都聚集在一起了，可能和超参的设置还有网络权重的初始化有关，我懒得调了~



这里还有一个 SDNE 在业界的应用的介绍：

阿里凑单算法首次公开！基于 Graph Embedding 的打包购商品挖掘系统解析

Struc2Vec

前面介绍过 DeepWalk,LINE,Node2Vec,SDNE 几个 graph embedding 方法。这些方法都是基于近邻相似的假设的。其中 DeepWalk,Node2Vec 通过随机游走在图中采样顶点序列来构造顶点的近邻集合。LINE 显式的构造邻接点对和顶点的距离为 1 的近邻集合。SDNE 使用邻接矩阵描述顶点的近邻结构。

事实上，在一些场景中，两个不是近邻的顶点也可能拥有很高的相似性，对于这类相似性，上述方法是无法捕捉到的。Struc2Vec 就是针对这类场景提出的。Struc2Vec 的论文发表在 2017 年的 KDD 会议中。

Struc2Vec 算法原理

相似度定义

Struc2Vec 是从空间结构相似性的角度定义顶点相似度的。

用下面的图简单解释下，如果在基于近邻相似的模型中，顶点 u 和顶点 v 是不相似的，第一他们不直接相连，第二他们不共享任何邻居顶点。

而在 struc2vec 的假设中，顶点 u 和顶点 v 是具有空间结构相似的。他们的度数分别为 5 和 4，分别连接 3 个和 2 个三角形结构，通过 2 个顶点(d,e;x,w)和网络的其他部分相连。

直观来看，具有相同度数的顶点是结构相似的，若各自邻接顶点仍然具有相同度数，那么他们的相似度就更高。

顶点对距离定义

令 $R_k(u)$ 表示到顶点 u 距离为 k 的顶点集合，则 $R_1(u)$表示是 u 的直接相连近邻集合。
令 $s(S)$ 表示顶点集合 S 的有序度序列。

通过比较两个顶点之间距离为 k 的环路上的有序度序列可以推出一种层次化衡量结构相似度的方法。

令 $f_k(u,v)$ 表示顶点 u 和 v 之间距离为 k（这里的距离 k 实际上是指距离小于等于 k 的节点集合）的环路上的结构距离(注意是距离，不是相似度)。

$$f_k(u,v) = f_{k-1}(u,v) + g(s(R_k(u)), s(R_k(v))), k \geq 0 \text{ and } |R_k(u)|, |R_k(v)| > 0$$

其中 $g(D_1, D_2) \geq 0$ 是衡量有序度序列 D_1 和 D_2 的距离的函数，并且 .
下面就是如何定义有序度序列之间的比较函数了，由于 $s(R_k(u))$ 和 $s(R_k(u))$ 的长度不同，并且可能含有重复元素。所以文章采用了 Dynamic Time Warping(DTW)来衡量两个有序度序列。

一句话，DTW 可以用来衡量两个不同长度且含有重复元素的的序列的距离（距离的定义可以自己设置）。

基于 DTW，定义元素之间的距离函数 $d(a,b) = \frac{max(a,b)}{min(a,b)} - 1$

至于为什么使用这样的距离函数，这个距离函数实际上惩罚了当两个顶点的度数都比较小的时候两者的差异。举例来说，$a = 1, b = 2$ 情况下的距离为 1，$a = 101, b = 102$ 情况下的距离差异为 0.0099。这个特性正是我们想要的。

构建层次带权图

根据上一节的距离定义，对于每一个 k 我们都可以计算出两个顶点之间的一个距离，现在要做的是通过上一节得到的顶点之间的有序度序列距离来构建一个层次化的带权图（用于后续的随机游走）。

我们定义在某一层 k 中两个顶点的边权为 $w_k(u,v) = e^{-f_k(u,v)}, k = 0, \ldots, k^*$
这样定义的边权都是小于 1 的，当且仅当距离为 0 的是时候，边权为 1。

通过有向边将属于不同层次的同一顶点连接起来，具体来说，对每个顶点，都会和其对应的上层顶点还有下层顶点相连。边权定义为:

$$w(u_k, u_{k+1}) = \log\left(\Gamma_k(u) + e\right), k = 0, \ldots, k^* - 1$$

其中 $\Gamma_k(u)$ 是第 k 层与 u 相连的边的边权大于平均边权的边的数量。

$\Gamma_k(u) = \sum_{v \in V} 1(w_k(u, v) > \bar{w_k})$ ， $\bar{w_k}$ 就是第 k 层所有边权的平均值。

采样获取顶点序列

使用有偏随机游走在构造出的图 M 中进行顶点序列采样。每次采样时，首先决定是在当前层游走，还是切换到上下层的层游走。

若决定在当前层游走，设当前处于第 k 层，则从顶点 u 到顶点 v 的概率为：

$p_k(u, v) = \frac{e^{-f_k(u,v)}}{Z_k(u)}$ 其中 $p_k(u, v) = \frac{e^{-f_k(u,v)}}{Z_k(u)}$ 是第 k 层中关于顶点 u 的归一化因子。

通过在图 M 中进行随机游走，每次采样的顶点更倾向于选择与当前顶点结构相似的顶点。因此，采样生成的上下文顶点很可能是结构相似的顶点，这与顶点在图中的位置无关。

若决定切换不同的层，则以如下的概率选择 $k + 1$ 层或 $k - 1$ 层：

$$p_k(u_k, u_{k+1}) = \frac{w(u_k, u_{k+1})}{w(u_k, u_{k+1}) + w(u_k, u_{k-1})}$$

$$p_k(u_k, u_{k-1}) = 1 - p_k(u_k, u_k + 1)$$

三个时空复杂度优化技巧：

- OPT1 有序度序列长度优化

前面提到过对于每个顶点在每一层都有一个有序度序列，而每一个度序列的空间复杂度为 O(n)。

文章提出一种压缩表示方法，对于序列中出现的每一个度，计算该度在序列里出现的次数。压缩后的有序度序列存储的是(度数，出现次数)这样的二元组。

同时修改距离函数为：

$$dist(a,b) = (\frac{max(a_0,b_0)}{min(a_0,b_0)} - 1)max(a_1,b_1) \quad a_0,b_0$$

为度数， a_1,b_1 为度的出现次数。

- OPT2 相似度计算优化

在原始的方法中，我们需要计算每一层 k 中，任意两个顶点之间的相似度。事实上，这是不必要的。因为两个度数相差很大的顶点，即使在$k=0$的时候他们的距离也已经非常大了，那么在随机游走时这样的边就几乎不可能被采样到，所以我们也没必要计算这两个顶点之间的距离。

文章给出的方法是在计算顶点 u 和其他顶点之间的距离时，只计算那些与顶点 u 的度数接近的顶点的距离。具体来说，在顶点 u 对应的有序度序列中进行二分查找，查找的过程就是不断逼近顶点 u 的度数的过程，只计算查找路径上的顶点与 u 的距离。这样每一次需要计算的边的数量从 n^2 数量级缩减到 $n \log n$ 。

- OPT3 限制层次带权图层数

层次带权图 M 中的层数是由图的直径 k^* 决定的。但是对很多图来说，图的直径会远远大于顶点之间的平均距离。

当 k 接近 k^* 时，环上的度序列 $s(R_k(u))$ 长度也会变得很短， $f_k(u,v)$ 和 $f_{k-1}(u,v)$会变得接近。

因此将图中的层数限制为 $f_{k-1}(u,v)$，使用最重要的一些层来评估结构相似度。
这样的限制显著降低构造 M 时的计算和存储开销。

Struc2Vec 核心代码

Struc2Vec 的实现相比于前面的几个算法稍微复杂一些，这里我主要说下大体思路，对一些细节有疑问的同学可以邮件或者私信我~

根据前面的算法原理介绍，首先确定一下我们要做哪些事情 ：

、

1. 获取每一层的顶点对距离
2. 根据顶点对距离构建带权层次图
3. 在带权层次图中随机游走采样顶点序列

顶点对距离计算

每一层的顶点对距离计算涉及到 4 个函数，我们一个一个看。。。

首先看第一个函数_get_order_degreelist_node，这个函数的作用是计算顶点 root 对应的有序度序列，也就是前面提到的 $s(R_k(u))$。

这里我们采用层序遍历的方式从 root 开始进行遍历，遍历的过程计算当前访问的层次 level，就是对应文章中的 k。每次进行节点访问时只做一件事情，就是记录该顶点的度数。当 level 增加时，将当前 level 中的度序列(如果使用优化技巧就是压缩度序列)排序，得到有序度序列。函数的返回值是一个字典，该字典存储着 root 在每一层对应的有序度序列。

第 2 个函数_compute_ordered_degreelist 函数就很简单了，用一个循环去计算每个顶点对应的有序度序列。

```
def _get_order_degreelist_node(self, root, max_num_layers=None):
if max_num_layers is None:
max_num_layers = float('inf')

ordered_degree_sequence_dict = {}
visited = [False] * len(self.graph.nodes())
queue = deque()
level = 0
queue.append(root)
visited[root] = True

 while (len(queue) > 0 and level <= max_num_layers):

count = len(queue)
if self.opt1_reduce_len:
degree_list = {}
else:
degree_list = []
while (count > 0):

top = queue.popleft()
node = self.idx2node[top]
degree = len(self.graph[node])

 if self.opt1_reduce_len:
degree_list[degree] = degree_list.get(degree, 0) + 1
else:
degree_list.append(degree)

for nei in self.graph[node]:
nei_idx = self.node2idx[nei]
if not visited[nei_idx]:
visited[nei_idx] = True
queue.append(nei_idx)
count -= 1
if self.opt1_reduce_len:
orderd_degree_list = [(degree, freq)
for degree, freq in degree_list.items()]
orderd_degree_list.sort(key=lambda x: x[0])
else:
orderd_degree_list = sorted(degree_list)
ordered_degree_sequence_dict[level] = orderd_degree_list
```

```
    level += 1

    return ordered_degree_sequence_dict

    def _compute_ordered_degreelist(self, max_num_layers):

    degreeList = {}
    vertices = self.idx # self.g.nodes()
    for v in vertices:
    degreeList[v] = self._get_order_degreelist_node(v, max_num_layers)
    return degreeList
```

有了所有顶点对应的$s(R_k(u))$后，我们要做的就是计算顶点对之间的距离 $s(R_k(u))$， 然后再利用公式 ：

$$f_k(u,v) = f_{k-1}(u,v) + g(s(R_k(u)), s(R_k(v)))$$

得到顶点对之间的结构距离 $f_k(u,v)$。

这里先看第 3 个函数 compute_dtw_dist,该函数实现的功能是计算顶点对之间的距离 $g(s(R_k(u)), s(R_k(v)))$, 参数 degreeList 就是前面一步我们得到的存储每个顶点在每一层的有序度序列的字典。

第 4 个函数 convert_dtw_struc_dist 的功能是根据 compute_dtw_dist 得到的顶点对距离完成关于 $f_k(u,v)$ 的迭代计算。

最后说明一下根据我们是否使用优化技巧 self.opt2_reduce_sim_calc 函数会选择计算所有顶点对间的距离，还是只计算度数接近的顶点对之间的距离。

```
    def compute_dtw_dist(part_list, degreeList, dist_func):
    dtw_dist = {}
    for v1, nbs in part_list:
    lists_v1 = degreeList[v1] # lists_v1 :orderd degree list of v1
    for v2 in nbs:
```

```
lists_v2 = degreeList[v2] # lists_v1 :orderd degree list of v2
max_layer = min(len(lists_v1), len(lists_v2)) # valid layer
dtw_dist[v1, v2] = {}
for layer in range(0, max_layer):
dist, path = fastdtw(
lists_v1[layer], lists_v2[layer], radius=1, dist=dist_func)
dtw_dist[v1, v2][layer] = dist
return dtw_dist

def _compute_structural_distance(self, max_num_layers, workers=1, verbose=0,):

if os.path.exists(self.temp_path+'structural_dist.pkl'):
structural_dist = pd.read_pickle(
self.temp_path+'structural_dist.pkl')
else:
if self.opt1_reduce_len:
dist_func = cost_max
else:
 dist_func = cost

if os.path.exists(self.temp_path + 'degreelist.pkl'):
degreeList = pd.read_pickle(self.temp_path + 'degreelist.pkl')
else:
degreeList = self._compute_ordered_degreelist(max_num_layers)
pd.to_pickle(degreeList, self.temp_path + 'degreelist.pkl')

if self.opt2_reduce_sim_calc:
degrees = self._create_vectors()
degreeListsSelected = {} vertices = {}
n_nodes = len(self.idx)
for v in self.idx: # c:list of vertex
nbs = get_vertices(
v, len(self.graph[self.idx2node[v]]), degrees, n_nodes)
vertices[v] = nbs # store nbs
degreeListsSelected[v] = degreeList[v] # store dist
 for n in nbs:
 # store dist of nbs
degreeListsSelected[n] = degreeList[n]
 else:
 vertices = {} f
or v in degreeList:
```

```
    vertices[v] = [vd for vd in degreeList.keys() if vd > v]



    results = Parallel(n_jobs=workers, verbose=verbose,)(
     delayed(compute_dtw_dist)(part_list, degreeList, dist_func) for part_list in partition_dict(vertices,
workers))
    dtw_dist = dict(ChainMap(*results))

    structural_dist = convert_dtw_struc_dist(dtw_dist)
    pd.to_pickle(structural_dist, self.temp_path +
    'structural_dist.pkl')

     return structural_dist
```

构建带权层次图

构建带权层次图的一个主要操作就是根据前面计算得到的每一层中顶点之间的结构化距离来计算同一层中顶点之间和同一顶点在不同层之间的转移概率，通过函数_get_transition_probs 实现。

layers_adj 存储着每一层中每个顶点的邻接点，layers_distances 存储着每一层每个顶点对的结构化距离。_get_transition_probs 只做了一件事情，就是逐层的计算顶点之间的边权 $w_k(u,v) = e^{-f_k(u,v)}$，并生成后续采样需要的 alias 表。

```
    def _get_transition_probs(self, layers_adj, layers_distances):
    layers_alias = {}
    layers_accept = {}

     for layer in layers_adj:
    neighbors = layers_adj[layer]
    layer_distances = layers_distances[layer]
    node_alias_dict = {}
    node_accept_dict = {}
    norm_weights = {}

     for v, neighbors in neighbors.items():
    e_list = []
```

```
sum_w = 0.0

for n in neighbors:
 if (v, n) in layer_distances:
wd = layer_distances[v, n]
else:
wd = layer_distances[n, v]
 w = np.exp(-float(wd))
e_list.append(w)
sum_w += w

e_list = [x / sum_w for x in e_list]
norm_weights[v] = e_list
accept, alias = create_alias_table(e_list)
node_alias_dict[v] = alias
node_accept_dict[v] = accept

pd.to_pickle(
norm_weights, self.temp_path + 'norm_weights_distance-layer-' + str(layer)+'.pkl')

 layers_alias[layer] = node_alias_dict
layers_accept[layer] = node_accept_dict

return layers_accept, layers_alias
```

前面的部分仅仅得到了在同一层内，顶点之间的转移概率，那么同一个顶点在不同层之间的转移概率如何得到呢?

下面的 prepare_biased_walk 就是计算当随机游走需要跨层时，决定向上还是向下所用到的 $\Gamma_k(u)$ 。

$$w(u_k, u_{k+1}) = \log\left(\Gamma_k(u) + e\right), k = 0, \ldots, k^* - 1$$

```
def prepare_biased_walk(self,):

sum_weights = {}
sum_edges = {}
```

```
average_weight = {}
gamma = {}
layer = 0
while (os.path.exists(self.temp_path+'norm_weights_distance-layer-' + str(layer))):
probs = pd.read_pickle(
self.temp_path+'norm_weights_distance-layer-' + str(layer))
for v, list_weights in probs.items():
sum_weights.setdefault(layer, 0)
sum_edges.setdefault(layer, 0)
sum_weights[layer] +=
sum(list_weights) sum_edges[layer] += len(list_weights)

average_weight[layer] = sum_weights[layer] / sum_edges[layer]

 gamma.setdefault(layer, {})

for v, list_weights in probs.items():
num_neighbours = 0
for w in list_weights: if (w > average_weight[layer]):
num_neighbours += 1
gamma[layer][v] = num_neighbours

layer += 1

 pd.to_pickle(average_weight, self.temp_path + 'average_weight')
pd.to_pickle(gamma, self.temp_path + 'gamma.pkl')
```

随机游走采样

采样的主体框架和前面的 DeepWalk,Node2Vec 差不多，这里就说下不同的地方。由于 Struc2Vec 是在一个多层图中进行采样，游走可能发生在同一层中，也可能发生跨层，所以要添加一些跨层处理的逻辑。

$$p_k(u_k, u_{k+1}) = \frac{w(u_k, u_{k+1})}{w(u_k, u_{k+1}) + w(u_k, u_{k-1})}$$

$$p_k(u_k, u_{k-1}) = 1 - p_k(u_k, u_k + 1)$$

```
    def _exec_random_walk(self, graphs, layers_accept,layers_alias, v, walk_length, gamma,
stay_prob=0.3):
    initialLayer = 0
    layer = initialLayer

    path = []
    path.append(self.idx2node[v])

    while len(path) < walk_length:
    r = random.random()
    if(r < stay_prob): # same layer
    v = chooseNeighbor(v, graphs,
    layers_alias, layers_accept, layer)
    path.append(self.idx2node[v])
    else: # different layer
    r = random.random()
    try:
     x = math.log(gamma[layer][v] + math.e)
    p_moveup = (x / (x + 1))
    except:
    print(layer, v)
    raise ValueError()

    if(r > p_moveup):
    if(layer > initialLayer):
    layer = layer - 1
    else:
    if((layer + 1) in graphs and v in graphs[layer + 1]):
    layer = layer + 1

    return path
```

Struc2Vec 应用

Struc2Vec 应用于无权无向图(带权图的权重不会用到，有向图会当成无向图处理)，主要关注的是图中顶点的空间结构相似性，这里我们采用论文中使用的一个数据集。该数据集是一个机场流量的数据集，顶点表示机场，边表示两个机场之间存在航班。机场会被打上活跃等级的标签。

这里我们用基于空间结构相似的 Struc2Vec 和基于近邻相似的 Node2Vec 做一个对比实验。

本例中的训练，评测和可视化的完整代码在 git 仓库中。

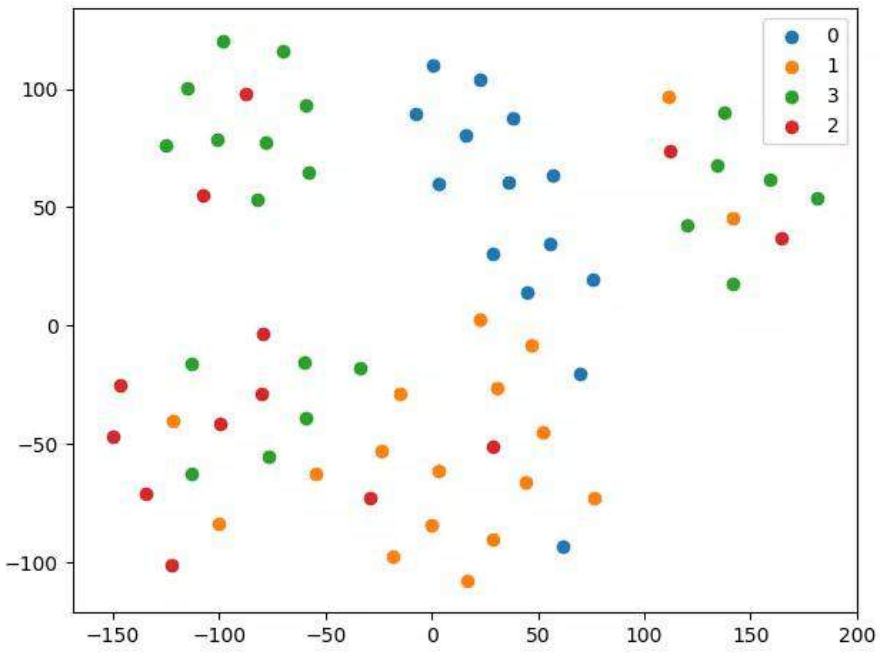
分类

- Struc2Vec 结果 micro-F1: 0.7143, macro-F1: 0.7357
- Node2Vec 结果 micro-F1: 0.3571, macro-F1: 0.3445

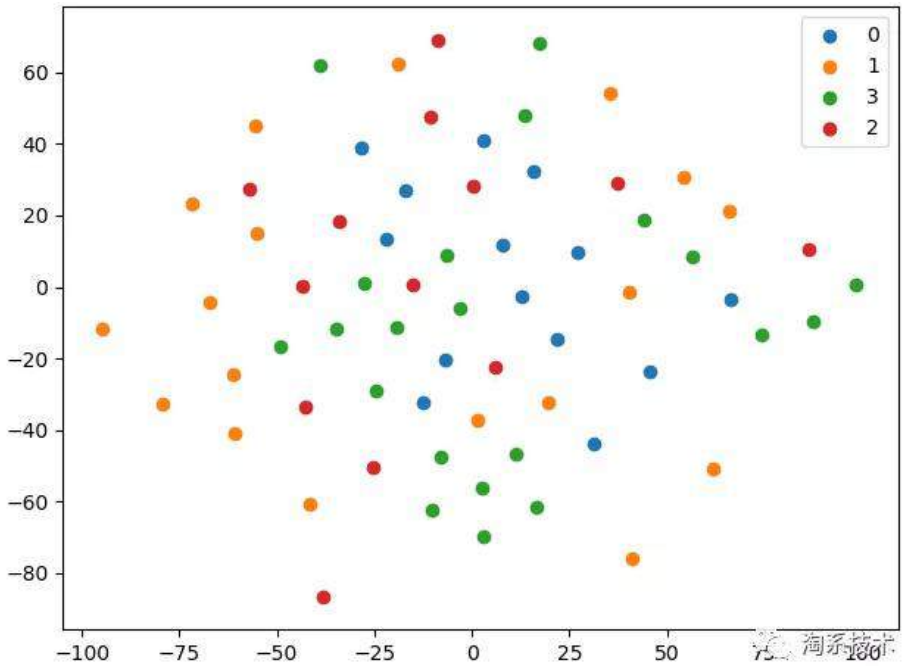
差距还是蛮大的，说明 Struc2Vec 确实能够更好的捕获空间结构性。

可视化

Struc2Vec 结果



Node2Vec 结果



关注淘系技术微信公众号，后台回复：ge 即可获取本文涉及的所有论文和代码汇总 ~

参考资料

Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations [C]//Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014: 701-710.http://www.perozzi.net/publications/14_kdd_deepwalk.pdf

Graph Neural Network Reviewhttps://zhuanlan.zhihu.com/p/43972372

Tang J, Qu M, Wang M, et al. Line: Large-scale information network embedding[C]//Proceedings of the 24th International Conference on WorldWide Web. International World Wide Web Conferences Steering Committee, 2015: 1067-1077.

Grover A, Leskovec J. node2vec: Scalable Feature Learning for Networks[C]// Acm Sigkdd International Conference on Knowledge Discovery &Data Mining. 2016.https://www.kdd.org/kdd2016/papers/files/rfp0218-groverA.pdf

Wang D, Cui P, Zhu W. Structural deep network embedding[C]//Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2016: 1225-1234.https://www.kdd.org/kdd2016/papers/files/rfp0191-wangAemb.pdf

struc2vec: Learning Node Representations from Structural Identityhttps://arxiv.org/pdf/1704.03165.pdf

内容推荐算法：异构行为序列建模探索

作者 | 杜易初（逸初）
出品 | 淘系技术

随很多秃头小可爱沉迷熬夜逛淘宝，以下是来自宇宙洪荒之力的最新线报：

- 男生比女生多
- 天蝎座多
- 夜间小裙子搜索量位居第一
- ……

看完视频和下方文章，你就知道为什么上头了。

点击可跳转原文查看视频详情

前言

来洋淘，更有 young！洋淘轻应用的内容均是手淘用户的真实购后分享，在这里，你可以个性化地浏览他人的美好生活，并鼓励自我表达与分享。

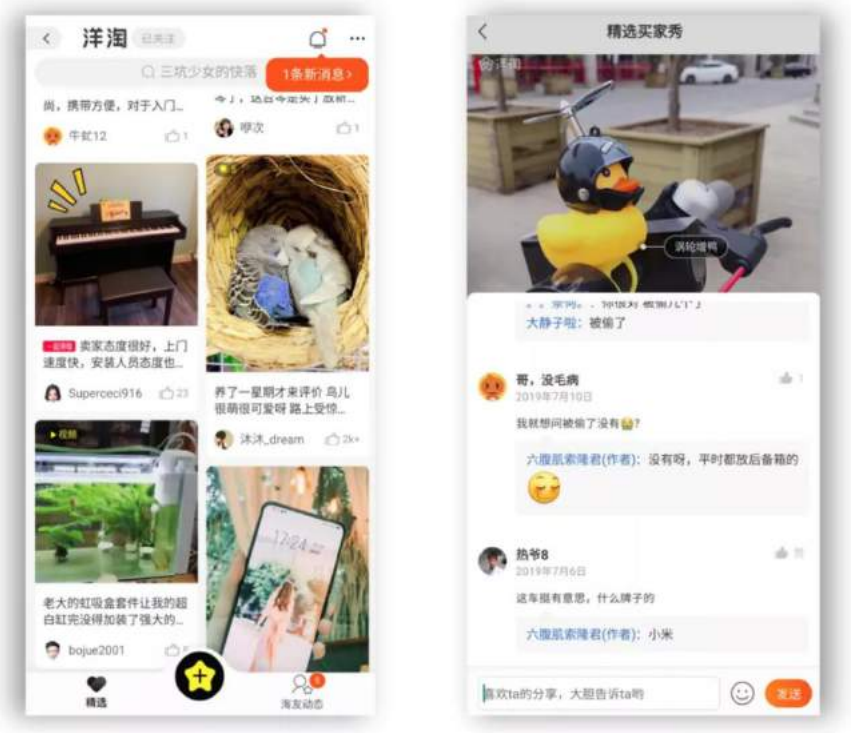


图 1. 洋淘轻应用截图， 左）为双列流， 右）为详情页

在推荐任务上，同导购产品相比，洋淘场景有着自身的特性：

- 新用户冷启严峻，六成以上的用户无洋淘场景内的历史内容点击；
- 老用户内容兴趣也稀少，历史内容点击数不超过 10，反观商品点击数却高达数倍；
- 用户在电商与内容两个 domain 的兴趣差异较大。统计日志发现，电商兴趣对洋淘场景下内容点击样本的覆盖度仅为 30% 。

于是，如何把跨域的异构行为用好，提升用户的浏览深度就显得弥足珍贵。我们从召回，ctr 预估两大关键环节均作了探索与创新，大幅提升了推荐效果，取得了显著的业务收益。

向量召回模型 In_Match

前文提到内容兴趣不够用，我们首先基于 i2i 尝试了商品与内容的混合协同，取得了一定的收益。然后基于电商兴趣的标题作相关内容的严格相似召回，却收益为负，这说明跨域的兴趣不可生搬硬套，也与前文的兴趣差异统计相吻合。

所以一个能胜任的召回模型，应满足以下期望：

- {电商，内容}两个 domain 的历史行为都能用上；
- 交互序列是刻画用户精确意图的关键，要能对序列建模而非孤立的交互单元；
- 既然两个 domain 兴趣差异存在，就要能学到电商兴趣到内容兴趣的迁移规律。

基于以上理念，我们设计了兴趣迁移 Interest Transfer Matching Model （In_Match for short），取得了以下显著收益（业务指标为内部数据，此处只列 ab 实验的相对提升）

- 推荐综合指标上，人均点击提升+11.17%，pv_ctr 提升+12.16%;
- 大幅缓解推荐过于集中的状况，top 内容的曝光占比下降 25%;
- 该路向量召回大幅超越 content-based feature 对应的几路召回，甚至高于内容协同召回。

模型架构及主要贡献

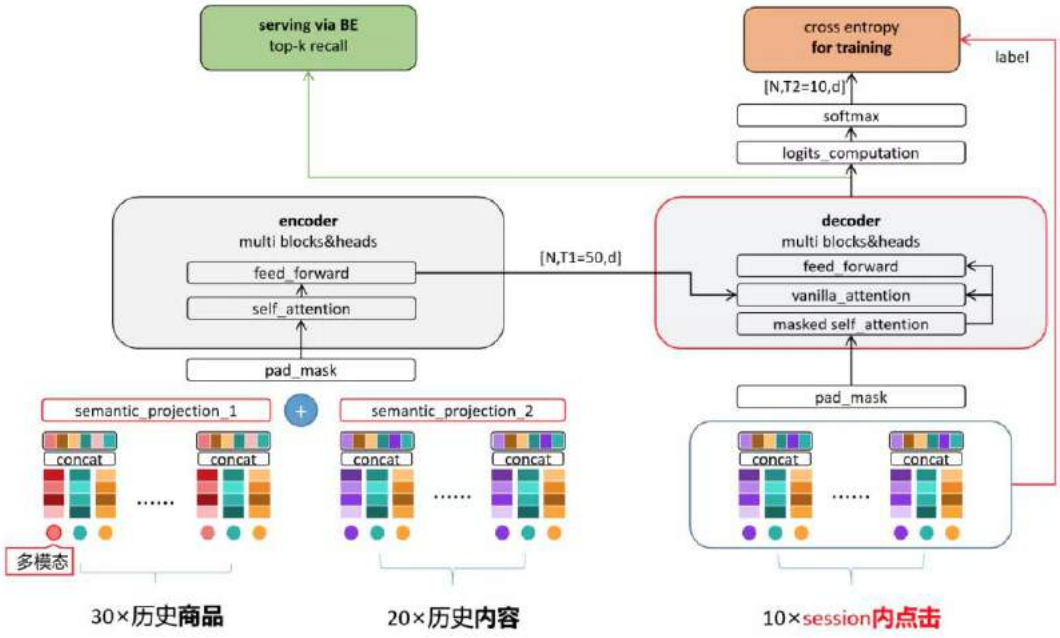


图 1-1.In_Match 的模型架构

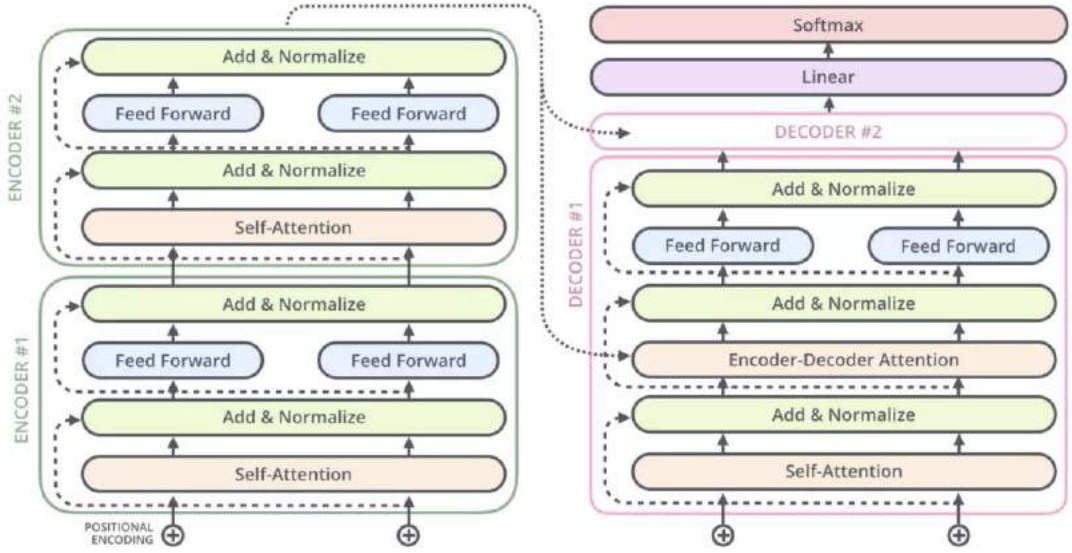


图 1-2.encoder-decoder 内部细节示例

主要贡献有：

- multi modality
  - 考虑到稀疏性，引入多模态性能提升+5%
- semantic projection
  - 填补 商品/内容 语义空间上的 gap
- 基于 decoder 来学习[整个 session]点击
  - 学习 session 点击，相较 next one ，SDM 有对比收益
  - decoder 架构更适合序列生成任务， GPT vs. BERT
  - beam-search 进一步提升多样性， 在 page 推荐下更友好

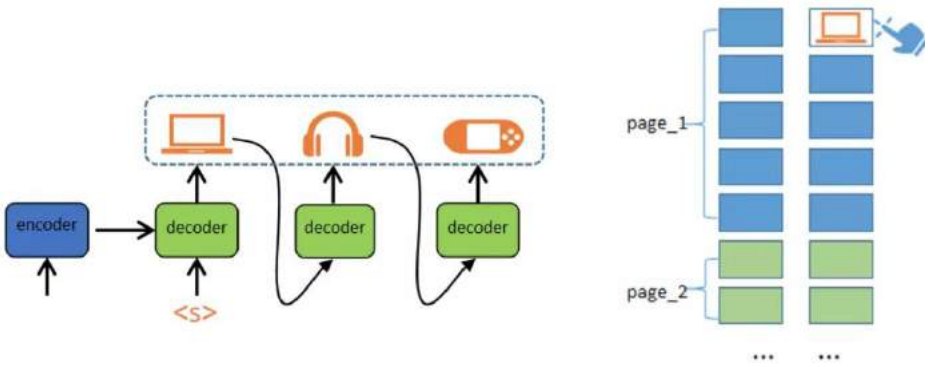


图 1-3.（beam-search）序列预测工作图示，多样性与兴趣演化

下文会分章节阐述我们的设计思想。

序列建模的必要性

推荐系统的首要任务是精准的捕捉用户意图，早期的人工抽取 content-based feature（如商品的叶子类目）作 trigger 的方式过于粗浅，有兴趣粒度不足及 trigger 间无法有效排序两处弊端，分别讨论。

1. content feature 粒度不够精细，譬如 {方便粉丝，米线，螺蛳粉} 三个品类共用一个叶子类目；又如连衣裙直接作为叶子类目，{A 字裙，荷叶边裙，百褶裙} 等款式差异无从体现，而不同的女孩子多半有着差异化的兴趣指向。
2. feature 庞杂难以取舍。洋淘用户多从首猜轻应用跳转而来，他们的商品点击行为丰富且多样，并且，把时间窗口扩大到七天，一个月，这些叶子类目兴趣 trigger 会更加膨胀，能召回成百上千个符合要求的洋淘内容，但推荐系统一次只返回一页有限个内容，如此多的兴趣作何取舍呢？典型的做法是， 综合考虑这些叶子类目对应商品的 {交互行为，交互次数，时间新旧}，给不同的叶子类目计算权重然后倒排，但这么计算兴趣分布也很粗糙，指标归因中可以看到叶子类目兴趣召回的效果是低于大盘指标的。所以我们不能孤立地看待这一次次点击行为，分别抽取各自的 content-based feature 作 trigger。完整的交互序列直接体现了用户过往来到淘宝逛了啥点了啥，且包含了时间的演进，直观上以及业界论文经验都能证明，日志中的交互序列对接下来的点击预测具有十分重要的指导意义，且价值远胜于{年龄，性别，地域}等用户画像信息。

注意力机制

Attention Mechanism 是近来大放异彩，业界青睐的建模手段。它允许模型在做当下的决策时，只把注意力放在最应该关注的那一部分，不再眉毛胡子一把抓。

scaled dot-product attention

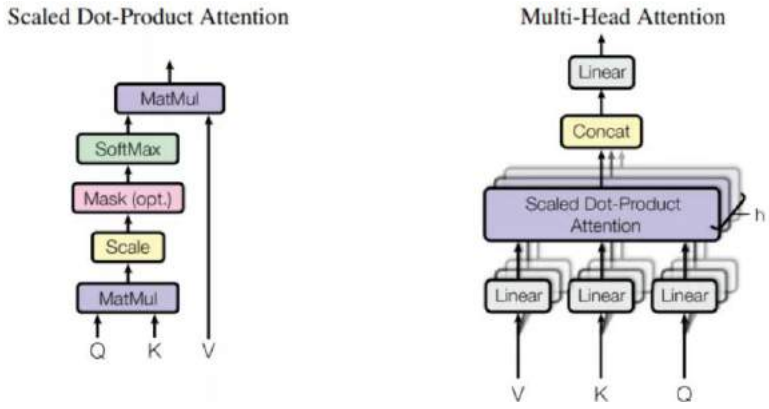


图 1-4.multi-head attention 图示

Dot-product attention 相较于传统的 RNN/CNN layer 的类 attention 处理，无论是训练效率上还是 long-range dependency 情形下的表现均有显著优势。它可以描述为这么一个映射，output=f(query，{key_value})，它们全都是向量。先计算 query 与若干个 key 的紧密程度得到 affinity matrix，再归一化后把这些 weight 分配到相应的 value 上，最终求和，就得到了一个 query 在 attention 后的最终表达。

self attention

自注意力，即在 encoder 与 decoder 两侧， 各自对同一 sequence 内的各个 item 间的关联做计算。它有啥好处呢，以图 1-5 为例，孤立看用户的某一次点击，我们很难知道背后的意图，就像人类在做阅读理解时碰到多义词，需要联系上下文才能判断具体的含义。这里也一样，通过 self attention 可得到一个交互商品（或内容）的上下文表达，即 contextual representation。

图 1-5.孤立看用户的某一次点击， 我们很难知道背后的意图

vanilla attention

它指的是 decoder 对 encoder，一个序列对另一个序列的 attention。这个更直观一点，相当于把下面的问题抛给了模型，让它自己去学自己去学——即用户当下的一次点击，应归因到又长又丰富的前序点击序列中的哪几次点击所体现出来的用户兴趣上呢？

多模态特征

我们的推荐数据集是非常稀疏的。为此分别通过 BERT in NLP 及 ResNet in CV 这些流行预训练模型做二次开发，产出期望的多模态特征，提升泛化性。消融实验表明此项子工作带来了额外+5%的性能提升。

图 1-6.BERT additional pre-train 指标：masked_lm_accuracy=73.52%，nsp_accuracy=99.33%，框选商品作 query，基于此作标题 k 近邻验证

图 1-7.框选图片作 query，基于此作图片 k 近邻验证

case study

人们常说深度网络像个黑盒，可解释性差。但在本网络中，可以观察 attention 权重，尝试作为对本次推荐结果的解释。attention 权重来自 block 3 of 3 的 vanilla attention，对四个 multi_head 的 attention 作 avg_pooling。

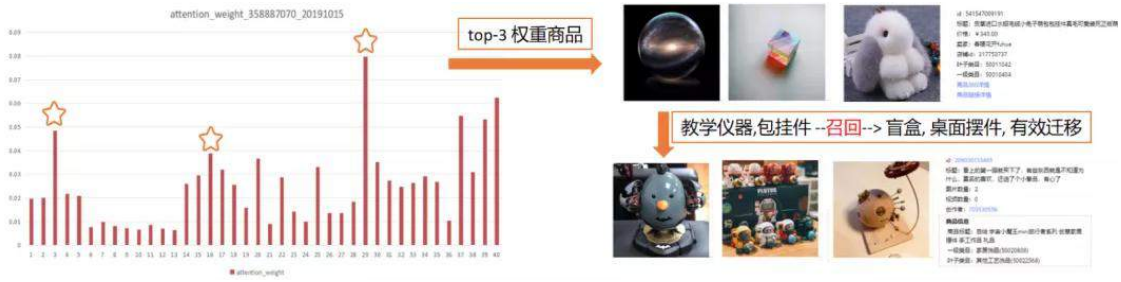


图 1-8.case 中， decoder 对历史商品序列的权重分配

CTR 预估模型 In_CTR

base 回顾

洋淘场景的 base 模型由 Wide&Deep FM&DIN 三部分融合而来，其理念见下:

- Wide

关于特征的学习，在神经网络 generalization 特性的帮助外，Wide and Deep 认为必要的 memorization 也是不能少的， 而退化成线性模型的 wide 子模块就被期望做这样的贡献。

- Deep Fm

CTR 任务某种程度上就是判断 target(如待打分的 doc)与用户意图间的匹配关系， 所以离不开用户特征与 target 特征的重度交叉。传统的 LR 模型中，只能依靠人工的高阶特征组合，这样的特征工程低效且低质。DeepFM 在 LR 基础上为每一个特征组中的每一个 field 学习一个 latent vector，依靠彼此点乘作二阶交叉。

$$Y_{FM} = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} < V_i, V_j > x_i x_j$$

它要求不同的特征向量维度固定，输出为一个 scalar， 使用时通常作为网络最后 logits 多项加和中的一项。

- DIN

DIN 认为用户的兴趣是多样的，对于确定的一个 target，通常只能匹配到历史行为序列中的若干商品而非全部，所以要有针对性地激活这些匹配到的商品，同时抑制之外的噪声。以这种有的放矢的思想建模序列表达， 相较 历史序列 mean pooling 能够带来额外的收益。

这些模块都是当下的一些主流实现，能力并不弱，但其缺少对异构序列的差异化建模，导致对 In_Match 该路召回内容的 pred_ctr 长期偏低，成为了推荐系统的新短板。

模型架构及主要贡献

针对 base 短板，我们提出并设计了 Interest Transfer CTR model（named In_CTR briefly），侧重 CTR 任务下异构序列的兴趣迁移挑战。既然商品兴趣不等于内容兴趣，我们就在深度网络中对兴趣的匹配程度 而不是 兴趣的直接表达 进行建模，并利用 匹配程度会直接影响点击意愿 这一先验设想设计了辅助 loss， 鼓励模型有所侧重的去学习，最终取得了 Offline GAUC+8.72%，Online uv_ctr +6.16% 的显著收益。

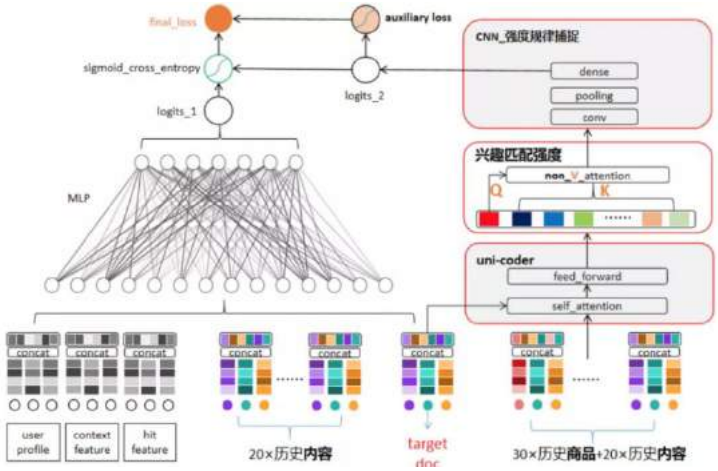


图 2-1.In_CTR 模型架构

模型左侧是常规的设计范式，我们的创新为右侧的红色框选模块，后续三个章节逐个介绍。

uni-coder 有效建模 异构序列

图 2-2.uni-coder 有效建模异构序列

我们参照文献 unicoder-VL ，对送入 encoder 之前的异构序列作如图 2-2 所示的预处理。entity 即异构序列中的一个元素，可以是商品，此时 side_info={cate，brand}；也可以是内容，此时 side_info={id，cate，brand，封图宽高比，etc.}；还可以是 special token，如[sep]作分隔，此时没有 side_info。

接着往上，是三层 emb 的 element-wise 加和。1) segment 标识电商兴趣/内容兴趣两个不同的 domain;2) context 标识用户对商品的交互行为，如点击/收藏/加购物车/购买，内容序列都是点击，出于对齐考虑，加一个 mock 即可； 3) position 标识交互次序，我们按时间排序。

提取兴趣的 匹配强度 而不是直接表达

我们对 self-att 模块的期望就是用它来捕捉序列内的相互依赖关系，可通过 tensorboard 可视化来验证它的表现是否符合预期。可视化 attention 方阵可以得到一个灰度图，元素(i，j)表示 i 作 query，j 作 key 时的 attention_score(soft_max 处理)，颜色越白表示权重越大。

在搜索团队的工作中，明确给出了他们学习收敛后得到的 self-att 可视化分布图，以对角线为主， 兼有 adaptive weight 观感。在我们的 In_Match 模型中同样观测到了这样的对角线(见 2-3 左)，同 In_Match 取得的成功可以互相印证。但在 ctr 任务中，异构序列加入后，one-hot 现象严重，表现不理想(见 2-3 右)。

图 2-3.异构序列的 self-attention 对比， 左): In_Match 模型， 对角线为主， 兼有 adaptive weight，表现理想; 右): ctr 模型，one-hot 现象严重， 表现不理想（without 兴趣匹配强度模块）

失败之后我们转变方向，不再执着于商品序列这一兴趣的直接表达，转向兴趣的匹配强度建模。我们的破局思考是: 预估 ctr，某种程度上就是在预估待打分内容与用户意图(历史异构序列) 的匹配程度的高低与否，也即紧密与否。既然商品兴趣不等于内容兴趣，且二者不在同一语义空间，我们就在深度网络中对 兴趣的匹配程度而不是兴趣的直接表达进行建模，体现出商品序列应有的贡献。改良后，获得了期望的 self-att 对角线观感。

兴趣强度的分布可以用直方图作可视化(图 2-4，中间部分)，那么怎么样的分布会倾向于较高的 ctr 呢？是整体偏高但平缓，还是少数几个柱子鹤立鸡群的观感呢？类比于图像特征的捕捉，我们用 CNN 作后续的特征提取。

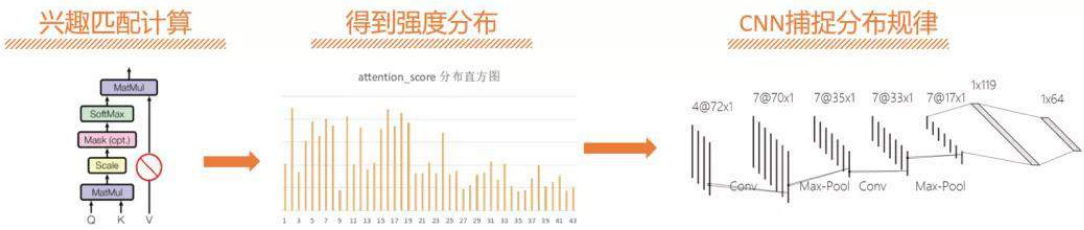


图 2-4.兴趣匹配强度 完整设计

auxiliary loss 设计

验证了兴趣匹配强度建模的有效性后， 将其作为子模块迁回原有网络参与训练， 再次遗憾地发现 attention 可视化分布中的对角线再次消失， 离线指标也涨幅有限。

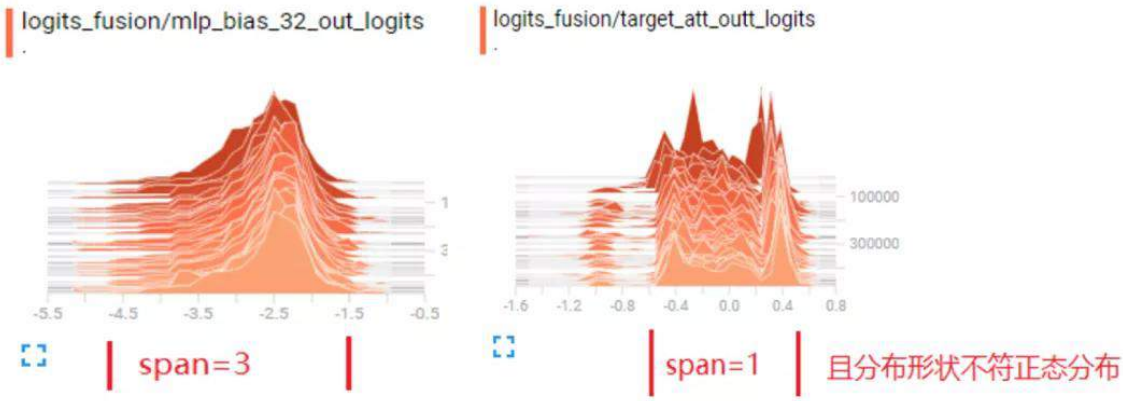


图 2-5.添加辅助 loss 前，兴趣匹配强度参与 logits 加和，其值域跨度和分布形状皆不符预期，体现不出异构序列的应有贡献

模型可能也会偷懒 ，bias 及 prefer 类特征规律性强，容易学习，就放弃了兴趣匹配子模块的学习。此时就要用 图 2-1 中 In_CTR 架构的 loss 处理方法，在其参与 logits 加和的同时，也为其独立出一个 loss，并分配合适的权重(实验中为 0.65，较为激进)。再次试验并观察 logits 分布，符合预期。

文末小总与展望

用户行为序列是其兴趣与意图的原始表达，重要性等同于搜索场景下的 query 输入。

商品行为作为异构序列，如何在跨 domain 的洋淘推荐中建模？围绕这一挑战，我们创新性地提出了：

1. In_Match，用序列生成的 task 训练 decoder，抽取其顶层输出向量作内容召回，召回效果超越内容协同;
2. In_CTR，将 motivation 由常规的序列兴趣的直接表达转换为序列兴趣的匹配程度，并根据这一先验设想设计了辅助 loss，鼓励模型有所侧重地去学习，最终离在线指标上都有斩获。

洋淘的发展方向是内容社区，所谓社区就要以人为本。未来我们会基于用户关注关系，兴趣下的用户分组等切面作推荐，也可以直接作"关注关系"的引导，诸如给出"你和 ta 都喜欢萌萌的治愈系挂件"这样的推荐理由，提升同好相吸的社区氛围。

推荐算法三视角:矩阵，图，时间线

作者 | 九胜
出品 | 淘系技术

关于推荐系统，如果忘掉所有的公式和代码，忘记所有的语言描述，脑海里就剩下几张图景，会是什么？一张二维表格，一个拓扑图，一条时间线。这三幅图景，是我看待推荐算法的三种视角。

视角一：矩阵视角

在脑中想象一个二维的表格，每一行代表一个用户，每一列代表一个物品，表格里的每一个点代表用户对物品的操作，这个操作可以是评分，点击，点赞。其中，有些格子记录了行为，有些格子是空的。到这里，我们就建立了基本的矩阵视角，推荐问题转化成了如何补上那些空格子。

用户对物品的评分等于相似用户对该物品评分的加权平均值，这就是 user-base 的协同过滤了。换一个方向，用户对物品的评分等于该用户对其他物品的评分按物品相似加权平均值，

这就是 item-base 的协同过滤。度量用户之间的相似度，把矩阵的一行——对物品的评分向量作为该用户的表示向量，那么用户之间可以计算向量的距离，可以选择任何距离公式，如余弦距离，皮尔森距离。对于物品之间的相似度，换一个方向即可。

对于任何两个物品，可以计算它们的评分差值。具体来说，两个物品有一批共同的历史评分用户，也就是矩阵里两列有交集的行，每一行可以计算一个差值，将差值平均起来，作为两个物品的距离。和上面的距离不同的，这个差值可以想象成物理中的位移，带着符号的。推荐时，某用户对于某个物品的评分，等于某用户对其他物品评分加上这个位移，再进行平均得到的平均评分。和上面的 item-base 一样的，都是列向量计算相似度，只不过相似度由距离变成了位移。这就是著名的 Slope-One 算法。

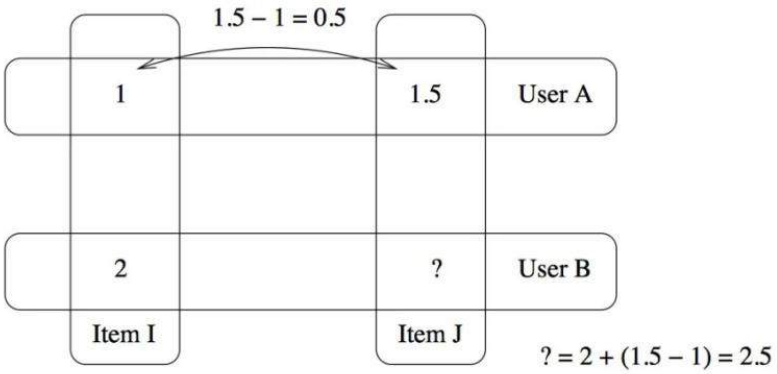
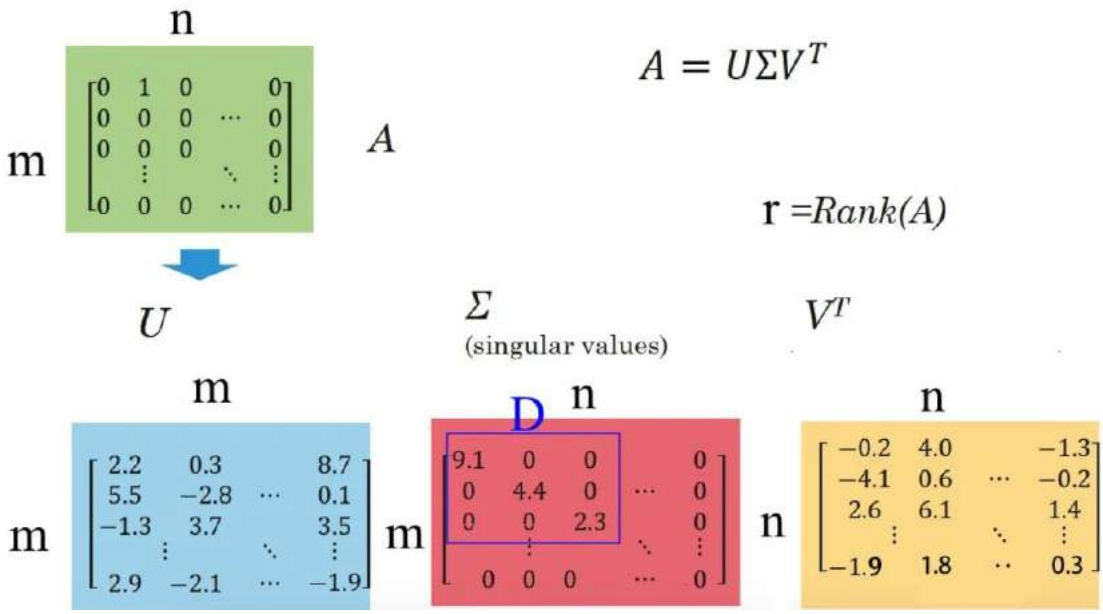


Figure 1: Basis of SLOPE ONE schemes: User A's ratings of two items and User B's rating of a common item is used to predict User B's unknown rating.

物品直接的相似度，除了上面的启发式算法，能不能通过数据本身学习所得？这就诞生了 SLIM(Sparse Linear Methods)方法。矩阵A是$n \times m$评分矩阵，要学习一个$m \times m$维的物品相似的矩阵W。A的每一行是用户的历史评分，W的每一列是每一个物品和该列对应物品的相似度，计算内积即为该用户对该列物品的评分，通过梯度下降训练来拟合真实评分。其中，W非负体现了相似度的物理意义；对角线限制为 0 避免对角线全都学习到 1 完美过拟合；添加 L1 正则产生稀疏的W，使得结果在大规模物品集上可用; W的每一列的学习都可以看作一个线性回归模型，训练时可以彼此相互独立，因而可以分布式学习。

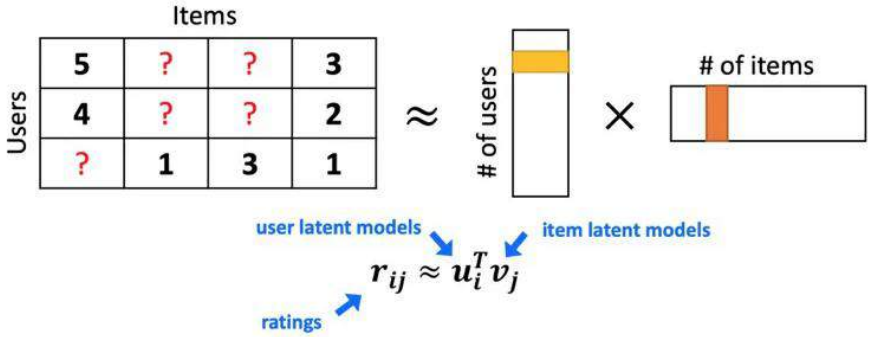
$$\begin{aligned} \underset{W}{\text{minimize}} \quad & \frac{1}{2}\|A - AW\|_F^2 + \frac{\beta}{2}\|W\|_F^2 + \lambda\|W\|_1 \\ \text{subject to} \quad & W \geq 0 \\ & \text{diag}(W) = 0, \end{aligned}$$

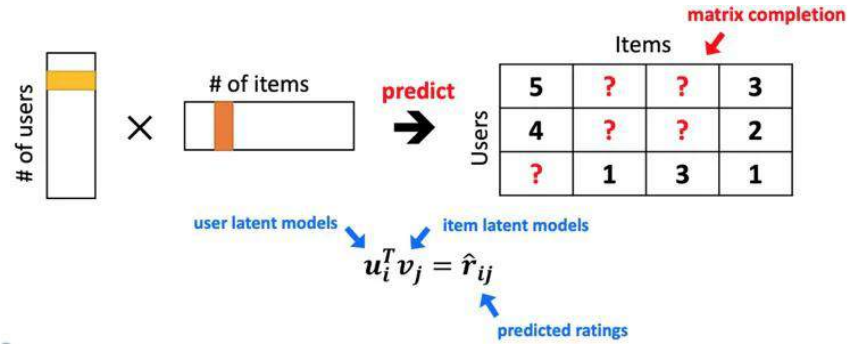
在矩阵视角下，很自然可以进行矩阵分解。SVD 矩阵分解将 n 个用户 m 个物品的大矩阵分解成三个矩阵相乘，中间的矩阵越靠近左上角的特征值越大，代表矩阵分解的主要成分，也就是说保留左上角的$k \times k$维矩阵 D，其余的都置为零，将原来的等于变为约等于。将蓝色和红色的矩阵合并，得到一个$m \times k$维的矩阵，每一个行代表一个k维的用户向量，对于黄色矩阵保留其前k行（后面的不影响计算了），每一列代表一个物品向量，用户和物品向量的内积也就是矩阵相乘后对应矩阵的值，也就是空缺处的评分，将向量索引起来就可以推荐了。



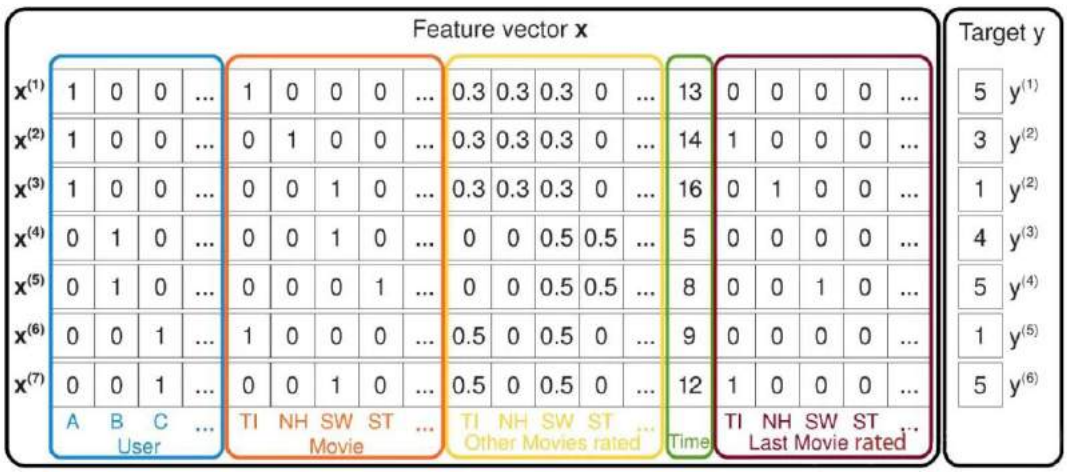
要使用 SVD 分解，待分解矩阵要是稠密的，稀疏的评分矩阵要按照统计学方法填充，如填充均值。另外，SVD 过拟合现象严重，泛化误差太大。在 2006 年 Netflix Prize 的百万推荐大奖赛上，Simon Funk 在博客公开 FunkSVD 算法。直接将评分矩阵分解成两个矩阵相乘，$n \times k$维度的用户矩阵，每一行是用户的隐式向量表示，$k \times m$维的物品矩阵，每一列是物品的隐式向量表示，用户和物品向量的内积即为预估的评分。那如何进行分解呢？随机初

始化矩阵，使用均方误差作为 loss，梯度下降进行学习。这个过程中还可以加入正则项，降低泛化误差。由 FunkSVD 开始，基于 Matrix factor（MF）的方法大放异彩。

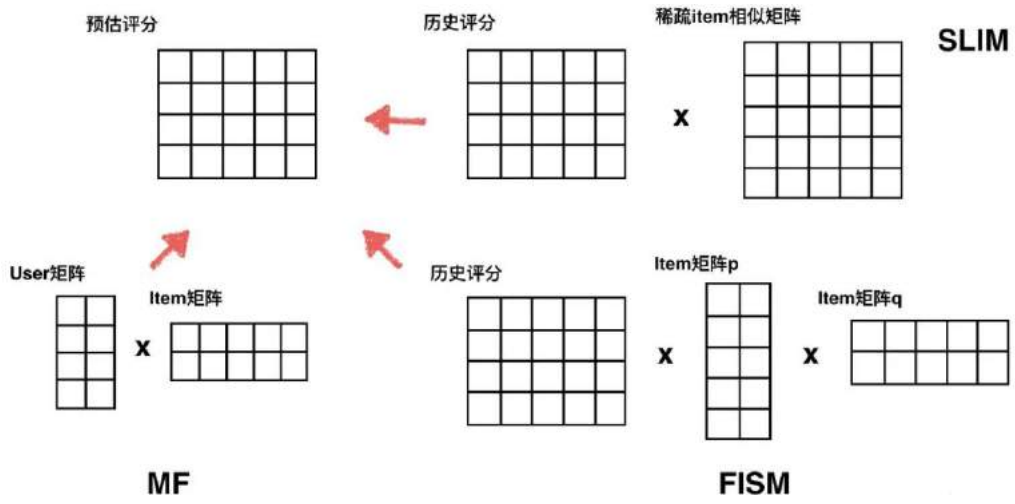




在 MF 的基础上，考虑推荐中的 side information，如用户的年龄性别，物品的类目价格。用户和物品自身或属性称作一个 field，field 之间可以两两进行矩阵分解，这个被称作二阶项，类似 BiasSVD 考虑每一个 field 都有一个 bias，这个被称作一阶项，再加上一个全局的 bias 项。这就是著名的 Factorization Machines（FM）。

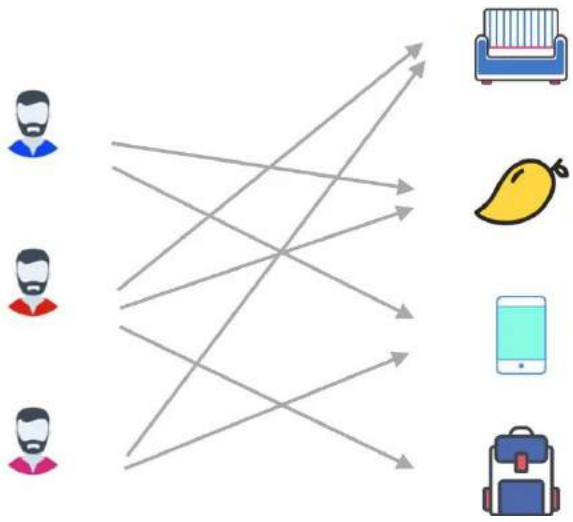


如果把上面介绍的 SLIM 和 MF 解结合起来，将物品的相似度矩阵 W 分解成 $P \times Q$ 两个低维矩阵，用户对某物品的评分，等于他过去评分过的物品在 P 中对应的向量和 Q 中该物品向量内积的和，这就是 FISM 算法。相比 SLIM 的稀疏处理，变为分解降维。最后再附上一张图，说明 MF，SLIM 和 FISM 之间的关系。



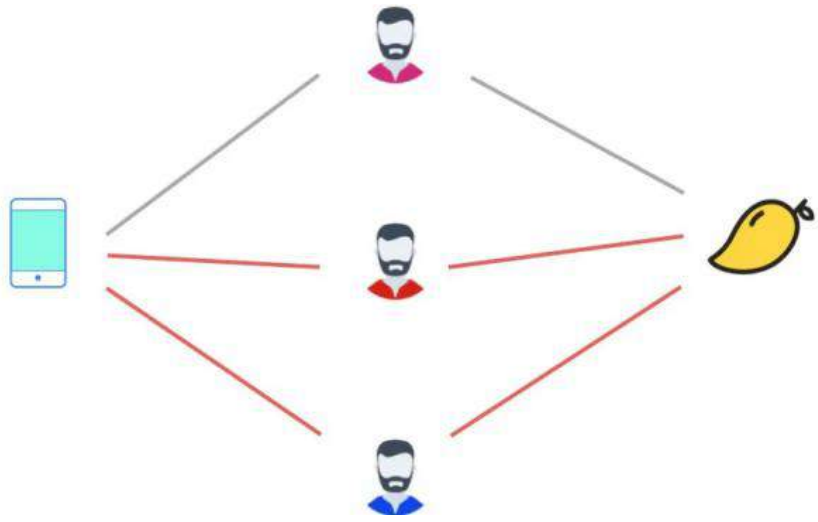
视角二：图视角

把用户和物品看作顶点，用户的评分在用户和物品之间建立起边，就得到了一个二部图；在二部图的基础上添加更多的顶点和边，形成一个更为复杂的图，辅助二部图的计算。在图的视角下，推荐问题转化成了在图上寻找高效的链接模式。

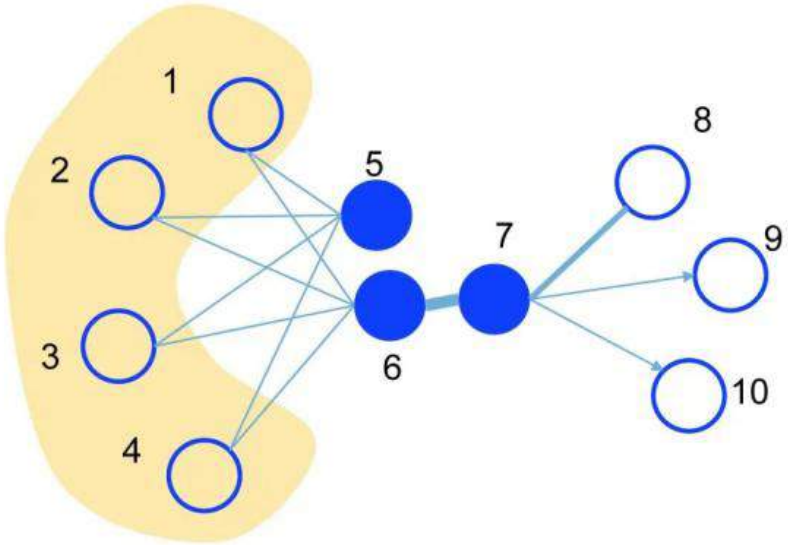


我们认为在同一个用户的历史行为中，那么两个物品之间有一条边，现在要计算两个物品之间的相似度，最朴素的思想就是数一数他们之间有多少条边。考虑每一条边权重不一样，边是通过用户建立的，用户的点击的物品越多，对应边的权重就越小。这就是 Adamic/Adar 算法的思想。

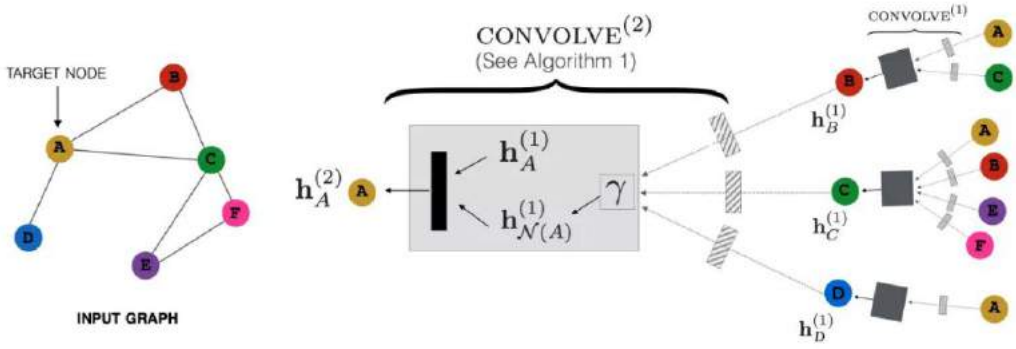
阿里著名的协同过滤推荐算法 swing，寻找图中更加稳固的形状，共同评分过两个物品的用户集合中，每两个用户和这个两个物品形成了一个四边形(下图红边为一个 swing 结构)，统计有多少个这样的结构，每一个结构的权重是不同的，这个结构里两个用户共同评分过的物品的数量越多权重就越小。



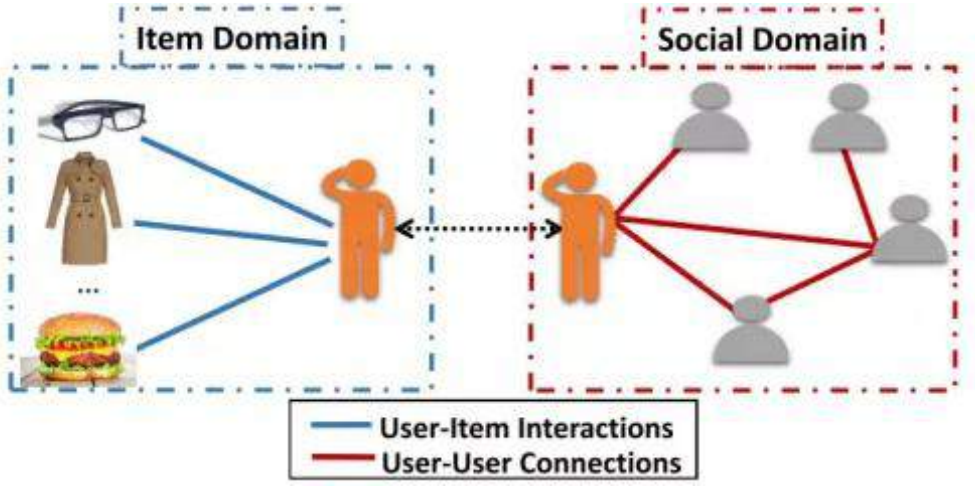
从用户和物品的二部图出发进行构图，再结合隐因子模型(Latent Factor Model)，就进入了 Graph-Embedding 的领域。DeepWalk 算法在图上随机游走深度优先遍历得到序列，然后和 word2vec 类似地使用 Skip-Gram（A 和 B 序列中相邻，用 A 的 embedding 作为特征最大化 B 的选中概率）进行训练。Node2Vec 算法在 DeepWalk 的基础上，考虑随机游走的方式，引入深度优先和广度优先的权衡，能够得到更好的更灵活的顶点隐式表示。LINE 算法考虑顶点的二阶相似，两个顶点有边为一阶相似，两个顶点有共同的邻居顶点为二阶相似，它虽不做随机游走，但可以看作是广度优先的采样。Graph-Embedding 取得了顶点的 embedding，计算相似度可以得到用户物品距离，物品物品距离，用于推荐。



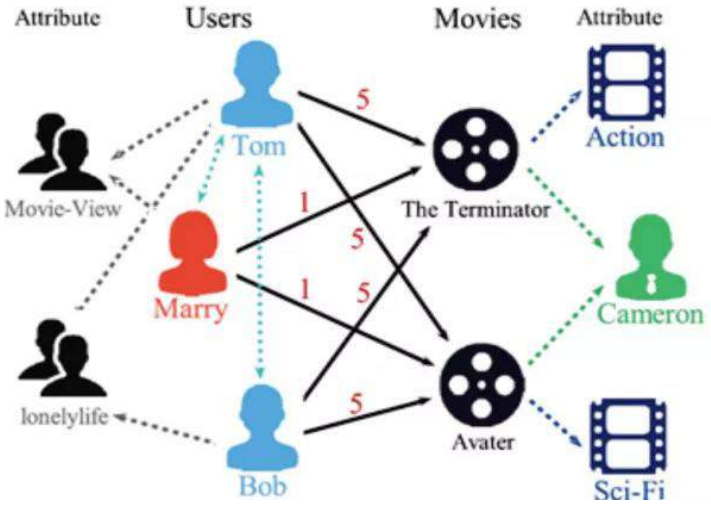
GCN（图卷积）接收拓扑图作为网络输入，可以计算每一个顶点更好的表示，相比 graph-embedding 可以有监督地为推荐目标而训练。但 GCN 在运算时，每一层都要输入整个图，在推荐系统里，物品和用户都可以是百万级别以上，实际中无法使用。GraphSAGE 通过 RandomWalk 采样，解决了这个问题，用在推荐领域就是 PinSage 算法。从某顶点出发，深度优先走 k 步，得到多个子图，组成一个 batch 进行训练，。然后按照采样的反方向做前向传播，这就是一个 k 层的图网络，下图是一个 k 为 2 的例子。



在用户和物品的二部图基础上，用户和用户根据社会关系建立起边来，这就是社会化推荐。

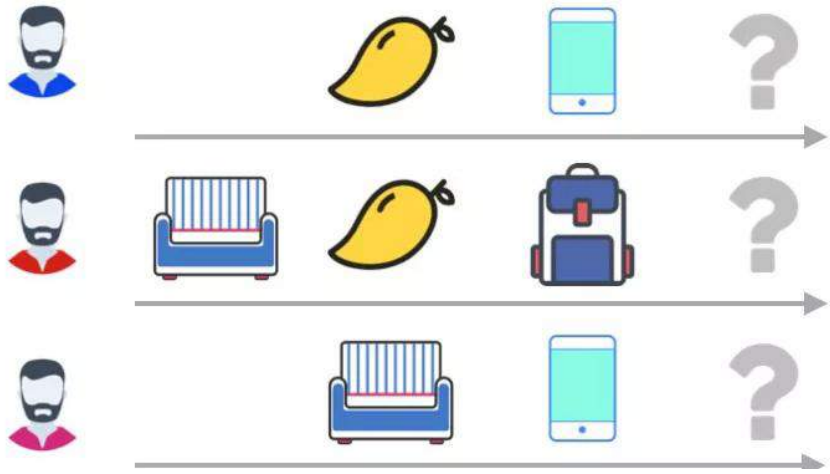


在用户和物品的二部图基础上，增加物品的属性作为顶点，建立新的边，就得到了一个异质信息网络。比如一个电影推荐系统，除了用户和电影外，还有导演，演员，电影类型，导演拍摄电影，电影属于某种类型，演员出演电影，导演与演员合作，诸如此类就能建立很多边。其中一类推荐算法叫做 meta-path，通过专家经验人工挑选出一些图中路径，如用户->演员->电影，用户->导演->电影，这样的路径称之为 meta-path，计算每一条 meta-path 的权重，将用户和物品间的所有 meta-path 联合计算评分。



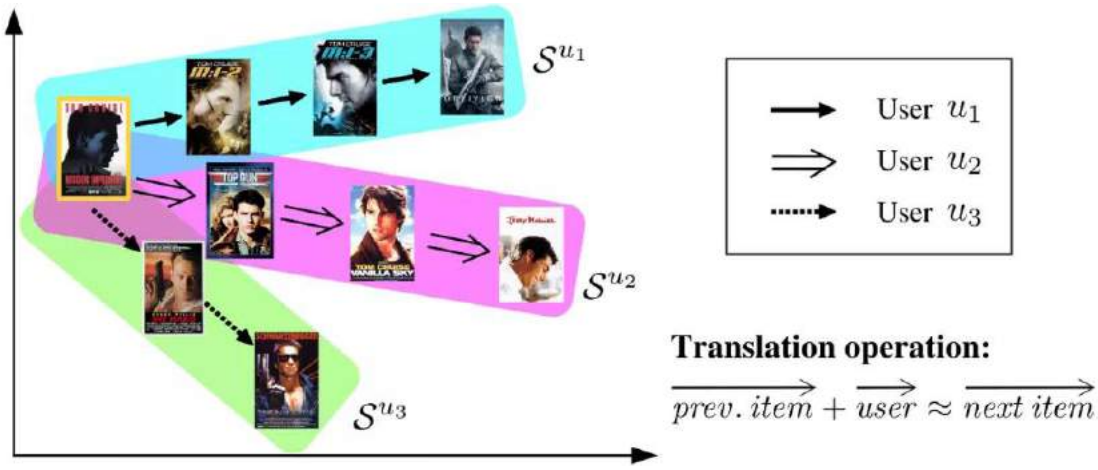
视角三：时间线

把用户对物品的行为想象成一条时间线，我们已知当前时刻前用户的物品行为序列，推荐问题被转化成了预测下一个时刻用户发生行为的物品。



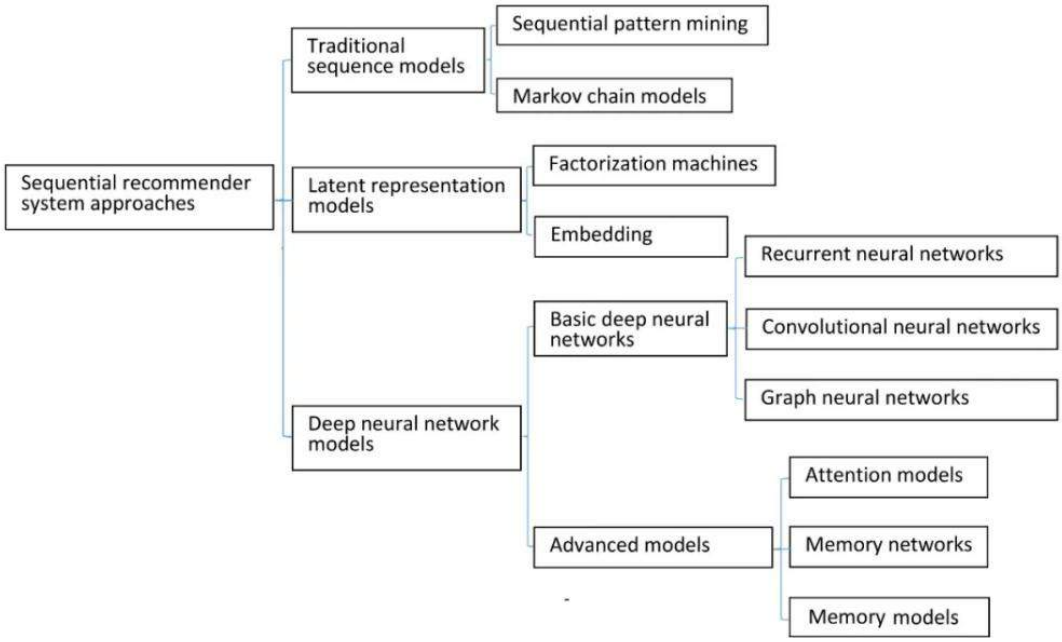
假设序列中下一个物品只与上一个物品有关，可以使用马尔科夫模型 MC（Markov Chains），序列中相邻的物品间进行矩阵分解。结合上文提到的用户和物品间矩阵分解 MF，用户，当前行为物品和下一个物品三者之间两两进行矩阵分解，将三个值加起来拟合评分，就得到了 FPMC（Factorizing Personalized Markov Chains）算法。

Translation-based 推荐在序列建模中引入 Metric Learning（把行为关系和高维空间距离映射起来），用户u，当前行为物品 i，下一个物品 j 三者向量化表示，训练使得它们满足了 $u + i \approx j$，推荐时只需拿到用户历史行为的物品向量加上用户向量得到下一个物品向量，然后在推荐集合中 KNN 寻找即可完成推荐。



以前模型的输入形式有限，人们通过特征处理将数据组织成模型可以接受的形式；随着深度

学习的发展，数据越来越倾向于保存其原有的形式，人们通过模型设计来学习有效的模式。在时间线的视角下，直接用深度模型结构建模序列，预测下一物品，形成了一个可以发挥想象力和燃烧算力的领域——Sequential/Session-base 推荐。在 2016 年的时候，RNN 是处理序列问题的标配，它从 NLP 领域走来，诞生了 GRU4Rec 算法。受到 NLP 领域 Char-CNN 启发，CNN 的结构也逐渐用于建模序列结构，Attention 机制大火之后，RNN+Attention，CNN+Attention，RNN+CNN+Attention 被枚举了一遍。随着 google 老师的 BERT 取得 NLP 领域巨大成就，Self-Attention 以及多层的 Transformer 结构开始成为序列建模的常规配置。最近的文章里，图神经网络(GNN), Memory networks,变分自编码器(VAE)也成为了序列推荐领域的深度乐高积木。



在 CTR 预估领域，越来越多的模型直接将用户历史行为序列按照时间顺序排列，使用深度模型结构直接建模。

总结

其实如果要细数，还有一个视角叫做高维空间视角。用户和物品都是一个高维度空间里的点，空间里点之间的距离越近，代表着物品和物品越相关，用户对物品越偏好，推荐问题转化成了如何将用户和物品嵌入到高维空间里。典型的主题如 Metric Learning。不过这个视角的

正交性不好，深度学习席卷推荐系统后，embedding 是个太常见的思路，前面很多的方法也都是最终把问题转化成了高维空间嵌入，如 graph-embedding，Transition-base 推荐。为了避免归类上的纠结；再加上任何一个深度网络作为 Encoder 把用户和物品 embedding，都可以归在这个视角，没有那么多令人印象深刻的典型方法，就不做单独梳理了。

To My Best Knowledge，我把自己认为推荐系统里经典且令人印象深刻的方法归在三种视角中——矩阵，图，时间线。本来想谈谈认识的，写着写着写多了，变成了一篇梳理文章。希望能对大家从偏算法的角度理解推荐系统有所助益。

引用

https://link.zhihu.com/?target=https%3A//blog.csdn.net/baimafujinji/article/details/6474273

http://xtf615.com/2018/05/03/recommender-system-survey/
Sequential Recommender Systems: Challenges, Progress and Prospects
Translation-based Recommendation

算法人精彩问答

Q: GAN 的生成器是怎么产生图片的?

作者 | 宝澜
出品 | 淘系技术

> 生成器接收一个噪声，然后产品图片，这个然后是个怎么样的过程，这个图片是怎么产生的呢?

分享一个我们做过的案例全过程。

通过 DCGAN 实现动漫人物图像的自动生成，我们也总结了这套方法的问题和后续解决方案。

背景

基于生成对抗网络（GAN）的动漫人物生成近年来兴起的动漫产业新技术。传统的 GAN 模型利用反向传播算法，通过生成器和判别器动态对抗，得到一个目标生成模型。由于训练过程不稳定，网络难以收敛，导致生成的图像缺乏多样性和准确性，甚至会产生模式崩溃。本文基于深度学习，参考相关实战项目 pytorch-book，学习网络的训练方法，采用经过标准化处理和分类的动漫人物面部图像知乎用户何之源分享的素材，训练 DCGAN，实现动漫人物图像自动生成。在训练过程中，控制实验参数，进行定量分析和优化，得到可自动生成动漫人物图像的生成器模型。主要工作如下：

- 设计 DCGAN 网络模型。对课题中所涉及到 GAN 原理、优化器等技术进行分析，并根据 DCGAN 的基本原则设计面向动漫人物生成的神经网络。

- 建立图像样本库。采集 40000 张动漫人物图像，按照比例随机分为训练集和测试集，并利用数据集对 DCGAN 进行无监督训练以及测试。
- 实现并训练 DCGAN 网络。使用 Pytorch 作为 DCGAN 的框架，利用 Vidsom 将图像生成过程的可视化，完成目标网络的构造以及训练。在训练过程中，控制实验参数，进行定量分析和优化，得到一组高质量网络参数。

背景及意义

随着二次元文化逐渐走进大众视野，各种动漫作品所塑造的角色已经成为一种特殊的文化符号。但是，由于动漫本身复杂的性质，其对于制作成本，质量，创意都有较高的要求，导致动漫行业频频出现投入高，收益低的现象。由于生成对抗网络（GAN）在图像生成领域和视频生成领域具有巨大的发展潜力，许多研究者尝试从 GAN 入手，实现动漫图像的自动生成，为创作者带来了灵感，还节省了巨额创作开支。

目前已有的动漫图像生成方式主要分为两种，第一种，是基于已有的真实人脸图片，进行人脸风格迁移，U-GAT-IT 项目利用图像处理将原始图像内容与动漫风格相融合，实现动漫风格转换，如图 1.1 所示。虽然风格迁移技术已经有丰富研究成果，但有一些缺陷依然无法避免，例如：生成的卡通图像表情单一、面部特征畸变、网络结构复杂等问题。

图 1.1（a）(b) 热力图 （c）动漫图像

第二种方法，是将已有的动漫人物图像，输入到训练好的生成对抗神经网络模型中，对动漫图像的面部进行特征提取，最后训练出一组网络模型，实现自动生成新的动漫人物图像，如图 1.2 所示。DCGAN 模型一般为五层，相较于传统的 GAN 模型更加稳定，生成的图像也更具有多样性，更能满足创作者的需求。

图 1.2 自动生成的动漫人物图像

基于 GAN 的图像处理研究现状

深度学习领域在近几年取得了重大突破，其中大部分研究成果都基于感知技术，计算机通过模仿人类的思维方式，感知物体、识别内容。生成对抗网络的理念由 Goodfellow 于 2014 年提出的，它的发展历程只有六年，却对人工智领域带来了极大的冲击。

GAN 的博弈过程，就是使用生成器制造的数据分布来拟合真实的数据分布。设置是一个生成动漫图片的网络，接收一个随机的噪声，生成动漫图片并输出。设置为判别网络，输入一张动漫图片，判别器可以计算出该图为生成的或真实图片的概率。两者分别根据返回的结果反向更新网络，相互抗衡，动态变化最后达到纳什均衡。

基于 GAN 性能的优越性，该模型逐渐被运用到图像处理领域的各个方向，其中包括图片转换、图像修复、风格迁移、图像生成等。例如：2016 年 Phillip Isola 等人设计的 pix2pixGAN[4]模型，可以将语义图片转换为街景和建筑的照片、素描图转彩色图片等，在图像转换领域实现了重大突破。

原始的 GAN 网络虽然在 2014 年才首次提出，但其扩展速度迅猛，产生了大量衍生网络，如：DCGAN、SGAN、ACGAN 等。2015 年 Mattya 首次提出了 chainerGAN,通过 DCGAN 实现动漫人物生成，但 Chainer 框架并未得到大范围应用。随后，Jie Lei 等人在 2017 年发表了 Animegan，该项目使用 GAN 框架，提出了三种新的损失函数：灰度风格 loss、灰度对抗 loss、色彩重构 loss，将各类现实场景下的图像转化为动漫风格的图像，例如自然风景、道路街景的转换。但遗憾的是，这些项目对于非计算机领域的爱好者使用具有一定难度，产生了局限性。

2017 年，复旦大学和 CMU 的学生共同发布了一项名为 MakeGirlsMore 的动漫人物生成项目，发布后在 Github 受到广泛关注。该项目使用 React.js 作为页面基本框架，为了提高项目的实用性，开发者将 Chainer 模型转化为基于 WebAssembly 的 Java 模型。用户可以在浏览器的界面上设置动漫人物的相关参数，例如：眼睛颜色、面部表情、发型等信息，实现动漫人物的自动生成。

训练方案

首先，建立图像样本库需要大量动漫图像，可使用 Python 在动漫素材相关网站爬取或使用网络上已有的数据集，按照固定比例划分训练集和测试集，并对样本进行标准化处理，使图像大小保持，每一张图片拥有唯一 ID。

其次，选择 CNN 作为 DCGAN 的基础网络，对 CNN 的池化层和全连接层进行调整。DCGAN 是一种深度卷积网络，可以实现目标特征提取以及图像分类。在定义 DCGAN 中的生成网络和判别网络时，通过增加 ReLU、Sigmoid 等激活函数用于参数处理、图像分类等工作。

在训练网络之前，使用 Pytorch 作为框架，Python 作为编程语言完成代码编写，利用 Adam 优化器最小化损失函数，优化程序调整超参数。训练网络时，使用 Visdom 实现训练过程的可视化，通过控制学习率、迭代次数、训练比例等参数，设置多组对照实验，观察实验结果并利用测试集进行多次测试。

将多组实验结果进行对比分析，可得到一组最优网络参数，利用训练好的目标网络模型，便可输出多张不同的动漫图像，技术路线如图 1.3 所示：

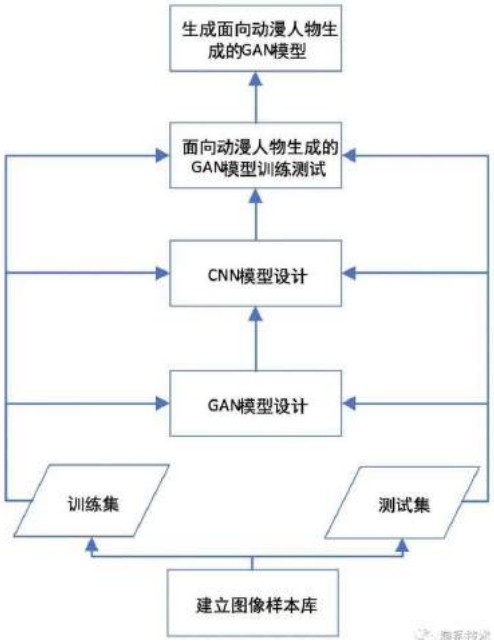


图 1.3 技术路线图

神经网络及 GAN 相关技术介绍

神经网络相关概念

计算机网络中的神经元是神经网络基本组成单位，它的结构参考了生物神经元。1943 年 McCulloch 等人提出了神经元模型 M-P。对于神经网络的发展有着重要的影响。M-P 抽象模型如图 2.1 所示：

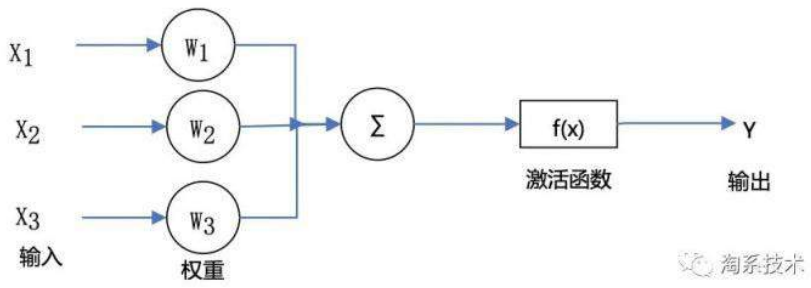


图 2.1 神经元结构模型

结合图 2.1 来看，神经元的输入输出关系用有向箭头来表示，输入信号可以用 Xi 表示，输出用 Y 来表示， Wi 表示权重。多个神经元可以通过设定的连接关系组合成为神经网络。

目前，神经网络根据工作原理可以分为人工神经网络和生物神经网络，在计算机领域中，人工神经网络的原理是利用计算机模拟大脑的工作方式。下图 2.2 展示了神经网络主要类别，本文的 DCGAN 模型就是基于多层神经网络 CNN 实现的。

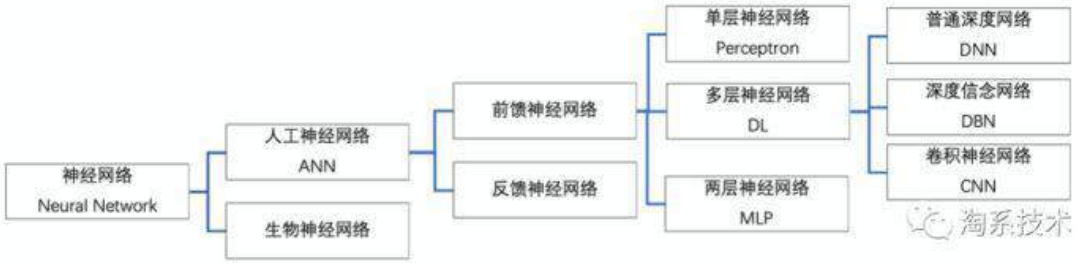


图 2.2 神经网络主要类别

单层感知器模型前馈神经网络最经典的模型，该模型与 1957 年由 Frank Rosenblatt 提出，可实现二类线性分类。将神经元节点添加在网络的输入、输出位置，作为网络的输入单元和输出单元。输入单元的功能是传输数据，输出单元实现对上一层的输入进行计算功能。下图 2.3 为第一代神经网络感知模型：

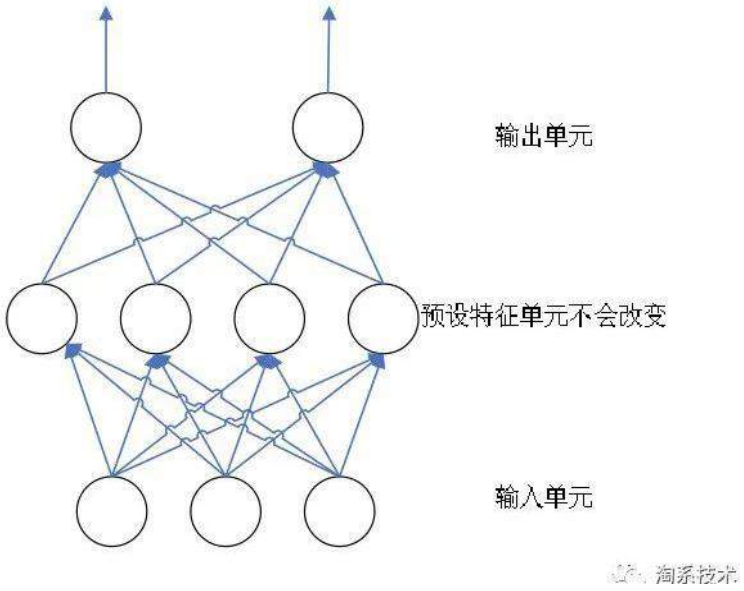


图 2.3 第一代神经网络模型

由于第一代神经网络模型不能适用于处理异或问题，仅局限于线性可分问题。所以直至 20 世纪 80 年代中期，分布式并行处理（Parallel Distributed Pro-cessing ，PDP）模型开始受到广泛关注。反向传播算法也逐渐成为 PDP 模型的主要学习算法这时，神经网络才又开始引起人们的注意，并重新成为新的研究热点[7]。第二代神经网络结构也应运而生，如图 2.4 所示：

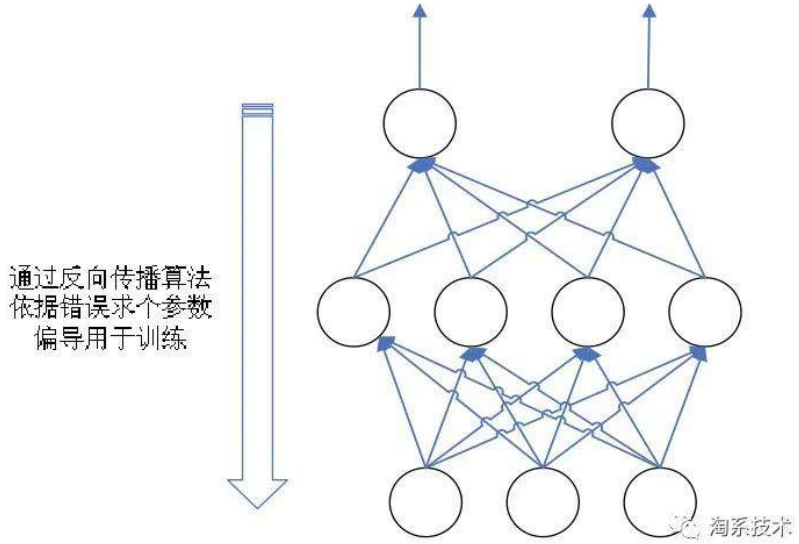


图 2.4 第二代神经网络结构图

2006 年 Geoffrey Hinton 提出了深度网络，通过大量数据训练神经元间的权重，让整个网络按照最大概率来生成数据，是一种概率生成模型，能够通过学习表示高阶抽象的复杂函数，可以使用它来识别特征、分类数据、生成数据等。

卷积神经网络

卷积神经网络（convolutional neural network，CNN）是指至少在网络的一层中使用卷积运算来代替一般的矩阵乘法运算的神经网络[18]。第一个卷积神经网络是 Alexander Waibel 在 1987 年提出的时延神经网络（TDNN），但是直到 2012 年 ImageNet 竞赛之前，CNN 才重新走进人们的视野。目前常用的 CNN 结构是由多个卷积层、池化层、全连接层组合构成的。如图 2.5 所示:

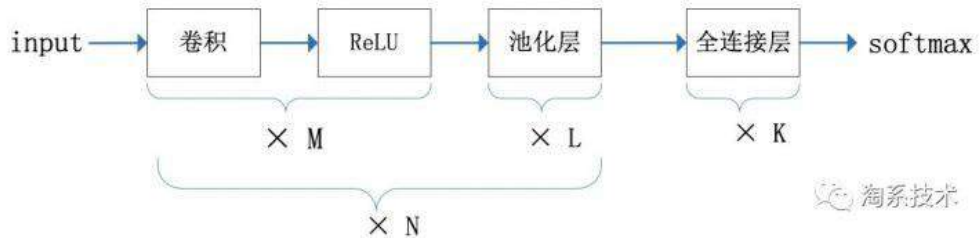


图 2.5 神经网络结构图

卷积层主要通过卷积核进行图像局部特征提取，其神经元按照宽度，高度和深度的排列方式，适用于彩色图像的 RGB 三色通道结构，因此它的排列方式又称为三维排列。避免了传统的感知器模型纬度较低的问题，可以更好地生成于高分辨率的图像，卷积层的各个参数设置影响卷积神经网络的性能，网络的精度和卷积层数呈正相关。

激活层一般常使用 ReLU 激活函数，它可以缓解梯度消失问题，通过将特征图映射到新的特征空间，提高模型鲁棒性、非线性表达能力。

池化层的主要作用是进行特征选择。池化层可以通过下采样过程，对于特征图进行融合和降维，相较于卷积层降低了计算量，避免了信息冗余。常用最大池化法和平均池化法。
全连接层用于将前一层输出的局部特征，重新连接成完整的图，可以将卷积神经网络的尾部重新拟合，减少特征信息的损失。

卷积运算

卷积(Convolution)的过程是在图像每个位置进行线性变换映射成新值的过程。本节主要讨论卷积运算在图像处中的运用。卷积运算主要是通过特征提取器(滤波器)对图片进行特征提取，例如图像的线条、结构等，降低深层网络模型的复杂度。其下列公式 2.1、2.2 可表示多层神经，上下层之间的关系：

$$W_2 = (W_1 - F + 2P) \div S + 1 \quad (2.1)$$

$$H_2 = (H_1 - F + 2P) \div S + 1 \quad (2.2)$$

淘系技术

上式 2.1、2.2 中有几个重要参数，W1：卷积前图像的宽度；W2：卷积后特征图的宽度；H1：卷积前图像的宽度；H2：卷积后特征图高度 ；S：步长（Stride），卷积核在滑动时的间隔；F：filter 的宽度；P：零补充（Zero Padiding），在原始图像两端补零的圈数。

下图 2.6 是一张大小的原始图片，设置经过卷积操作后，得到一张大小的特征图像的部分过程，图 2.6 为原始输入图像，绿色部分为卷积核，蓝色图像为目标图像,为卷积后图像长度，图 2.7 中的值可以通过绿色部分的运算。

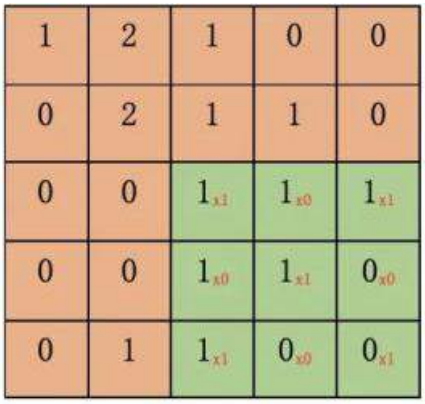


图 2.6 输入图像

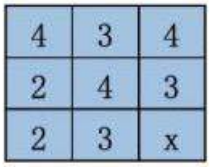


图 2.7 特征向量图

淘系技术

卷积后图像特征图的长度为：

$$W_2 = (5-3+2*0) \div 1+1=3 \quad (2.3)$$

淘系技术

运算后的取值为：

$$x = 1*1+1*0+1*1+0*0+1*1+1*0+0*1+0*0+1*1=4 \quad (2.4)$$

淘系技术

通过以上运算过程，可以总结出，卷积运算就是将高维数据映射到低维数据，而逆卷积运算就是卷积运算的相反操作，将低维数据映射到高维数据。

动漫图像生成网络设计

DCGAN 设计原则

DCGAN 目前是 GAN 在实际工程实践中被采用最多的衍生网络，为了提高图像生成质量，增强其稳定性，许多研究学者尝试进行优化，并提出了四点设计原则，本课题中 DCGAN 的生成器和判别器的设计是基于该原则实现，下面将进一步阐述它们之间的关系。

1. 卷积层代替池化层池化操作会使卷积核在缩小的特征图上覆盖了更大的图像视野，但是对网络性能的优化效果较小，使用卷积层代替池化层，让网络自动选择筛去不必要信息，学习上采样和下采样过程，提高计算机运算能力。
2. 去掉全连接层全连接层一般添加在网络的末层，用于将图像特征进行连接，可以减少特征信息的损失，但是由于其参数过多，会产生过拟合、计算速度降低等问题。由于面部图像特征提取的感受野范围较小，不需要提取全图特征，所以为了避免上述问题，本项目中网络模型去掉了全连接层。
3. 批量归一化本课题中的生成器和判别器都是五层神经网络，每一层输入的数据的复杂度都会逐层递增，使输出数据的分布发生变化，对网络参数的初始化和 BP 算法的性能产生影响。将数据进行批量归一化（Bach Normalization,BN），可以使输出的数据服从某个固定数据的分布，把数据特征转换为相同尺度，从而加速神经网络的收敛速度。
4. 激活函数激活函数（Activation Function）具有连续可导的特性，可以使神经网络进行非线性变化，通过对数值优化来学习网络参数，提升网络的扩展性。本课题的生成器和判别器均为五层网络模型，计算量较大，每一层的激活函数选择需要满足高计算效率和训练稳定两点，其导函数的值域分布合理。

基于以上原则，在 DCGAN 的生成器添加了 ReLu 函数、Tanh 函数，判别器中添加了 LeakyReLu 函数和 Sigmoid 函数，如下内容将对这四个激活函数进行的简单介绍。

Sigmoid 函数是一种两端饱和型函数，取值范围在 0~ 1 之间，Sigmoid 函数定义如公式 3.1 所示：

$$\sigma(x)=\frac{1}{1+e^{-x}} \tag{3.1}$$

淘系技术

下图 3.1 是 Sigmoid 的函数图像，它以坐标轴原点 0 为分界，输入值变大，输出结果接近于 1，输入值减小，输出结果接近 0，所以一般用它做输出端，解决二分类问题。这种特性也存在一定弊端，例如，神经网络训练结果输出恒大于零，并且当输入数据为极大，极小值时，Sigmoid 函数梯度无限趋近于 0，不利于神经网络的反向传播。

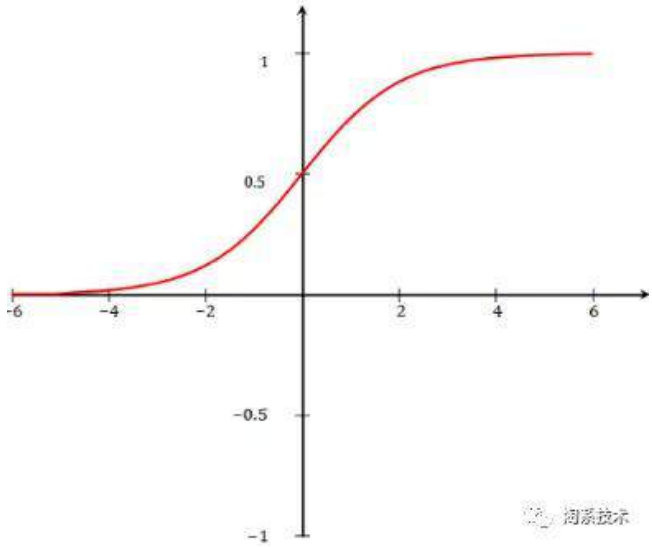


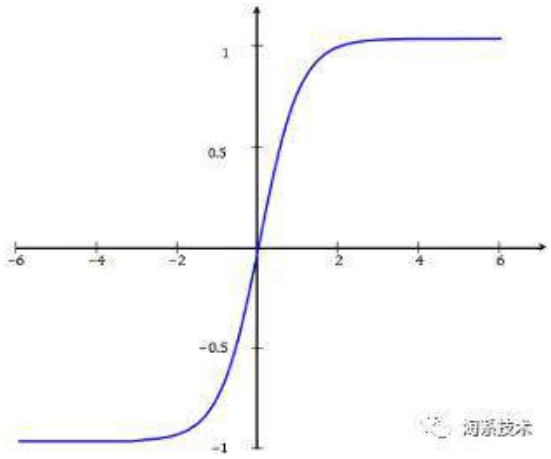
图 3.1 Sigmoid 函数图像

Tanh 函数的图像也是 S 型，一般情况下 Tanh 的收敛速度都优于 Sigmoid 函数，它可以避免出现均值不为零的情况，公式如 3.2 所示：

$$f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

淘系技术

由函数图像 3.2 可得，它的值域为(0,1)，DCGAN 网络在生成器的输出层添加 Tanh 函数，有利于图像色彩覆盖。



Relu 激活函数是一种左饱和的激活函数。它的计算效率较高，一般常用于隐层神经元输出，在近几年被广泛用于训练多层神经网络模型，其函数公式 3.3 所示：

$$\mathrm{Re}LU = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} = \max(0, x)$$

由 ReLU 函数图 3.3 可知，导数在 x>0 时为 1，x<0 时为 0，可以提高神经网络的收敛速度。但是 ReLU 也存在一些问题，例如"死亡 ReLU",在训练时，如果有一个数据较大的梯度流通过 ReLu 神经元，导致参数更新出现问题，数据分布被打乱，这个神经元的梯度保持为 0，且无法对任一数据进行激活操作。

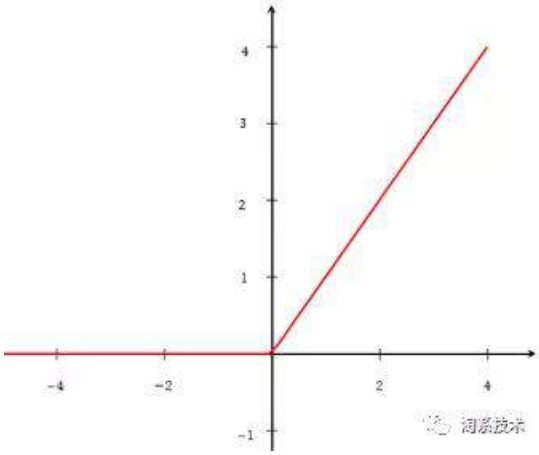


图 3.3 ReLU 函数图像

LeakyReLU 函数可以解决"死亡 ReLU"现象，是基于 ReLU 函数的基础上提出的，函数公式如 3.4 所示：

$$Leaky\,\mathrm{Re}\,LU(x)=\begin{cases}x & if \quad x>0\\ \gamma x & if \quad x\le 0\end{cases}$$
$$=\max(0,x)+\gamma\min(0,x)$$

如图 3.4 所示，当小于 0 时，给增加一个斜率，取值一般较小，避免了 ReLU 在负半轴训练时停滞的问题。

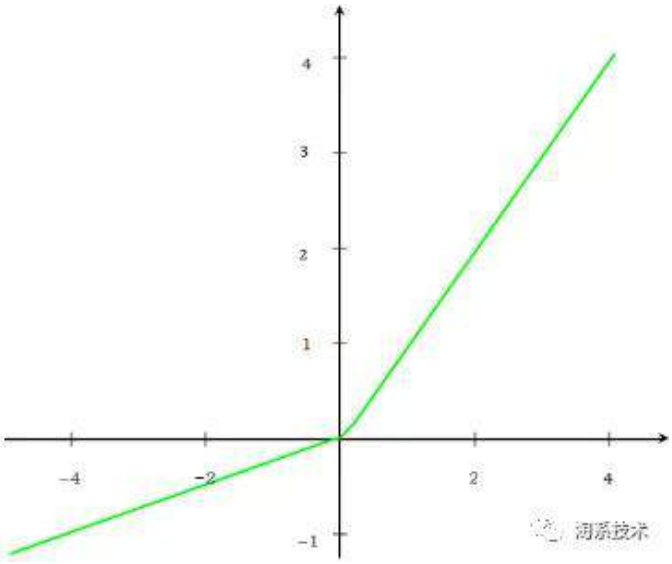


图 3.4 LeakyReLU 函数图像

DCGAN 网络结构

DCGAN 模型由生成网络 G 与判别网络 D 组成，网络结构均为五层，生成网络的接受一个随机输入的噪声，通过该噪声生成一张目标图像；判别网络接受一张图像，对该图像做出判断后，计算出一个概率值，若该图像来源为生成网络，则输出 0，若该图像来自真实的数据分布，则输出为 1。两者根据反馈的结果，不断更新自己的参数，直至达到纳什均衡。

生成网络和判别网络详细设计

生成器网络结构

生成模型的网络结构一共有五层，通过上采样方法生成大小的图像，它的主要结构如图 3.6 所示：

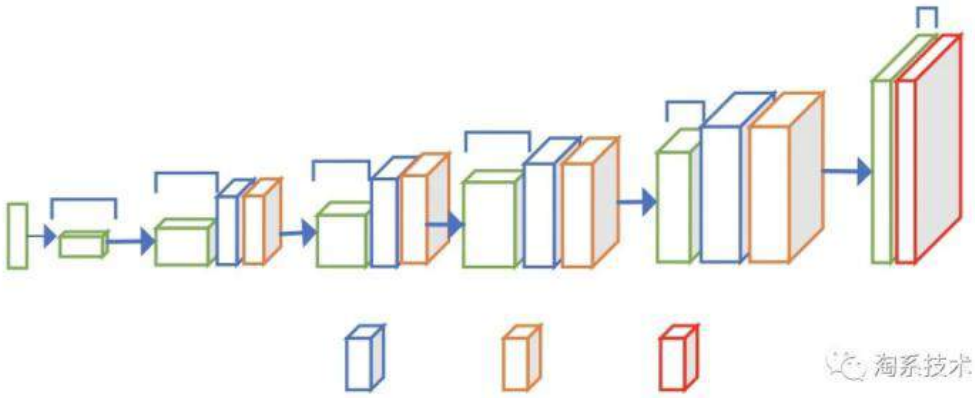


图 3.6 DCGAN 的生成器网络结构图

生成器整个网络结构没有池化层，输入一个服从均匀分布的 nz 维度的噪声，神经网络根据输入的向量信息，分步获取输入图像的特征信息，例如：线条、风格等，随后根据网络的深度，不断优化图片的细节。

输入一张的图片，根据需要的特征图大小设置相应的卷积核，卷积核的大小影响了生成网络的可学习空间特征值的大小。在代码中使用 nn.Sequential()函数可以根据传入的次序，将卷积、激活、池化等按照顺序加入该模块中，在构建完网络层后，自动调用 forward 方法。在除了输出层之外的每一层，加上批归一化（BN）处理，缓解模型崩溃问题。根据每层网络结构的需求，使用对应的激活函数，最后，输出一个像素的 3 通道 RGB 图像。具体的层次结构如下表 3.1 所示：

Layer	Kernel size	Stride	Padding	Activa-tion Funct ion	BN
Layer1	4×4	1	0	ReLU	Yes
Layer2	4×4	2	1	ReLU	Yes
Layer3	4×4	2	1	ReLU	Yes
Layer4	4×4	2	1	ReLU	Yes
Layer5	5×5	3	1	Tanh	No

淘系技术

表 3.1 DCGAN 生成模型层次结构

Layer1 是输入层，输入随机噪声，将 Stride 设置为 1，将 Padding 设置为 0。接下来，通过 ConvTranspose2d（）函数求逆卷积,并在 BN 层对数据进行批量归一化处理，使用 ReLU 激活函数加速收敛，输出对应的特征映射图；

Layer2：将 Stride 设置为 2，将 Padding 设置为 1，添加 BN 层将数据批量归一化处理，使用 ReLU 激活函数加速收敛，输出对应的特征映射图；

Layer3：将 Stride 设置为 2，将 Padding 设置为 1，添加 BN 层将数据批量归一化处理，使用 ReLU 激活函数加速收敛，输出对应的特征映射图；

Layer4：将 Stride 设置为 2，将 Padding 设置为 1，添加 BN 层将数据批量归一化处理，使用 ReLU 激活函数加速收敛，输出对应的特征映射图；

Layer5：输出层，将 Stride 设置为 3，将 Padding 设置为 1，使用 Tanh 作为激活函数，输出对应的图像。

生成器代码结构图如图像 3.7 所示：

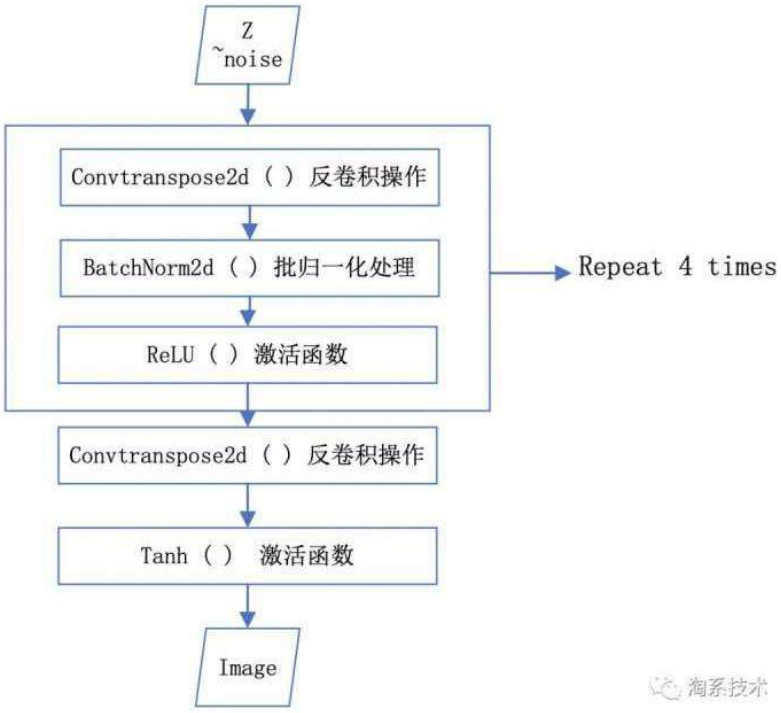


图 3.7 生成器代码结构图

判别器网络结构

判别模型的网络结构一共有五层，采用下采样方法。整个网络结构没有池化层，采用 LeakyReLU 作为激活函数，最后，通过一个全连接层输出判别结果，范围为 0~ 1，表示输入图像属于真实样本还是由生成器所生成的图像，它的结构如图 3.8 所示：

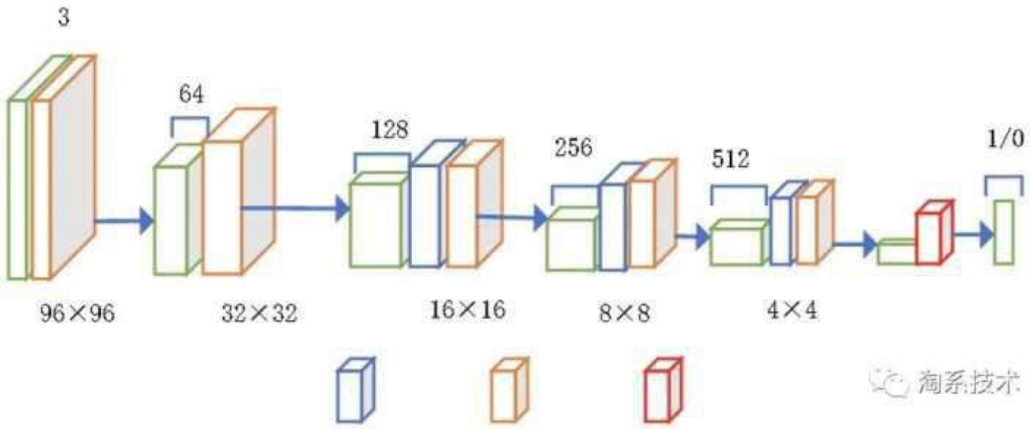


图 3.8 DCGAN 的判别器网络结构图

判别器中也添加了 BN 层，并且每一层都使用激活函数进行非线性处理，具体的层次结构如下表 3.2 所示：

Layer	Kernel size	Stride	Padding	Activation Function	BN
Layer1	5×5	3	1	LeakyReLU	No
Layer2	4×4	2	1	LeakyReLU	Yes
Layer3	4×4	2	1	LeakyReLU	Yes
Layer4	4×4	2	1	LeakyReLU	Yes
Layer5	4×4	1	0	Sigmoid	No

淘系技术

表 3.2 DCGAN 判别模型层次结构

Layer1：输入层，首先输入图片，Stride 设置为 1，将 Padding 设置为 0。通过 Conv2d（）函数求卷积，接下来使用 LeakyReLU()激活函数，优化网络收敛速度；

Layer2：输入图像，Stride 设置为 2，将 Padding 设置为 1，添加 BN 层将数据批量归一化处理，使用 LeakyReLU()激活函数加速收敛；

Layer3：输入图像，Stride 设置为 2，将 Padding 设置为 1，添加 BN 层将数据批量归一化处理，使用 LeakyReLU()激活函数加速收敛；

Layer4：输入图像，Stride 设置为 2，将 Padding 设置为 1，添加 BN 层将数据批量归一化处理，使用 LeakyReLU()激活函数加速收敛；

Layer5：输出层，Stride 设置为 3，将 Padding 设置为 1，使用 sigmoid（）函数输出概率，表示图像的真实性。

综上，可以看出这两个网络模型的 Padding、Stride 等设置几乎相同，整体的网络结构是对称，只是各层使用的激活函数有差异。

损失函数

损失函数是一种非负实数函数，可以评估模型的预测值与真实值不一致的程度。两者差距减少，概率分布越接近，差距增加，概率差异越高，在 Pytorch 中可以通过导入 torch.nn 包来使用。Pytorch 中提供许多损失函数，每一种函数都有其特性，例如：MSELoss 是一种均方误差损失函数，使用梯度下降算法，一般常用于解决股票预测、房价预测等回归类问题；SmoothL1Loss 是一种稳定的损失函数，也被用于解决回归问题，它的函数曲线光滑可以避免梯度爆炸的问题；BCELoss 是 CrossEntropyLoss 的一个特例，常用于解决分类问题。在本课题中我们需要，判断样本的输出是真实图片还是生成图片，所以本课题选择 BCELoss 作为损失函数，它在 PyTorch 中的定义如公式 2.1 所示:

$$\ell(x,y)=L=\{l1,\ldots,lN\} \quad ,ln=-[yn\cdot logxn+(1-yn)\cdot log(1-xn)] \tag{3.5}$$

淘系技术

由于生成网络和判别网络的输出层的激活函数分别为 Than 函数和 Sigmoid 函数，两者都是 S 型函数，其函数特性会导致反向传播算法收敛速度降低，使用 BCELoss 函数后，解决了因 sigmoid 函数导致的梯度消失问题。

Adam 优化器

在 DCGAN 的训练过程中，可以通过优化器最小化损失函数，一般分为一阶优化算法和二阶优化算法。本课题选用 Adam 优化程序调整超参数，它结合了 AdaGrad 和 RMSProp 算法最优的性能，不仅可以计算每个参数的自适应学习率，还可以通过训练数据的不断迭代使网络权重自动更新，相较于其他几种算法而言 Adam 算法实现简单、对计算机资源占用率较低，收敛速度也更快。

Adam 算法有一些重要参数，其中 params 表示用于优化的可以迭代参数或定义参数组；lr 表示学习率，可以调节权重的更新比例，影响网络的收敛速度，在 Pytorch 源码中定义如下：

torch.optim.Adam(params,lr=a,betas=(b,c),eps=d, weight_decay=e)

本文基于 Adam 优化器的默认参数做出改进，采用一种动态调整学习率的方法，改善训练过程中产生的图像模糊，模式崩塌，损失函数震荡明显等问题，详细步骤在网络训练部分阐述。

动漫图像生成实战

本章首先将几种常用的开源框架进行对比分析，选择了最适用于本实验需要的开源框架 Pytorch，并对可视化工具 Visdom 的功能进行说明。接下来，详细描述了网络训练流程，对各组实验生成的图像和 Loss 函数进行了分析，对传统的 DCGAN 网络训练参数提出了改进，针对相关问题提出优化方案。

实验环境搭建

深度学习框架 Pytorch

随着各类开源深度学习框架的不断更新迭代，大规模的深度学习模型得到了极大的简化，促进了图像处理、NLP、语音识别等技术的发展。本课题的核心功能是自动生成动漫图像，近几年大火的 TensorFlow、Pytorch 等框架都可用于处理图像类问题。

TensorFlow 框架属于静态计算图，生成图像后无法改变，在初始化时需要考虑到所有的可能性，对计算机配置要求高。在代码中，静态图不支持基本的 python 语法，缺乏灵活性，不适用于研究性学习。

由于 Pytorch 框架的设计思想基于动态计算图，它的 API 的设计具有极高的通用性，源码直观简洁，易于理解，支持 python 的条件判断、循环等基本语法，可以动态执行并且调试

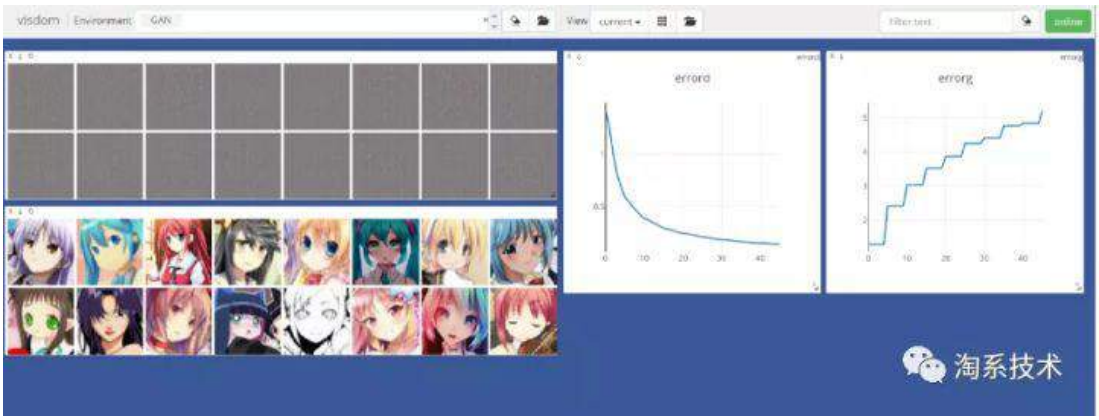
方便。基于它的高效性、易用性，本课题选 Pytorch 作为深度学习框架。

可视化工具 Visdom

可视化工具可以为我们实时提供程序运行过程中各类数据变化情况和训练效果。Visdom 是一款轻量级的可视化工具。适用于 Pytorch 框架下的图像展示，支持 Python 语言，可绘制多种图形，例如：热力图、地理图、自定义图像。

Visdom 的 UI 可视化框架主要由环境和窗格组成。一般默认使用 main 环境，不同环境之间相互独立。窗格相当于一个容器，支持多种图像类型，一个环境中可以创建多个窗格。用户通过创建一个自己的绘图环境，启动浏览器窗口，便可以实时查看图像数据，这些图像可以缩放、保存、动态更新。

基于 Visdom 具有轻量、实用性、灵活性的特点，本课题选择它作为可视化工具，绘制 Loss 函数和生成图形变化过程，便于分析数据，调整优化器和网络训练参数。下图 4.1 展示了在 Visdom 中网络训练过程和 Loss 函数。



综上，选用第三方框架为 Pytorch，使用 Python 完成代码编写，通过 Visdom 实现图像生成过程的可视化，使用样本库对网络进行训练，实验所需的环境和软件的详细信息在下表 4.1 所示：

系统	Windows	10
第三方框架	Pytorch	0.4.1
编程软件	Pycharm	2019.3
编程语言	Python	3.6.10
运行环境	Conda	4.8.2

淘系技术

表 4.1 实验环境

训练过程

建立图像样本库

图像样本库内容为动漫人物面部特征图片，可通过爬虫获取大量原始动漫人物形象素材后，使用 opencv 工具将图片进行剪裁，仅留出动漫人物面部图像，本项目使用知乎用户何之源提供的已处理好的动漫图像样本库，按照 1：1 随机划分为训练集和测试集。

- 训练集包含 20000 张动漫人物图像，图像格式为 JPEG，分辨率为 3 96 96，每张图像的文件名是一个唯一的 id；
- 测试集包含 20000 张动漫人物图像，图像格式为 JPEG，分辨率为 3 96 96，每张图像的文件名是一个唯一的 id，样本参数图 4.1:

图 4.1 图像样本参数

图 4.2 图像样本库（部分）

超参数设置

为了证明各参数设置对网络训练效果的影响，在不同的对照组中按照比例设置参数。其中，G 代表生成器在单位周期的训练次数，D 代表判别器单位周期的训练次数，LrG 表示生成器学习率，LrD 表示判别器学习率，Epcho 表示总训练周期，BatchSize 为一次迭代时输入的样本数量，经过多次实验与测试，选取了以下 4 组具有代表性的实验参数用于实验结果分析，如下表 4.2 所示：

实验组	G	D	Epcho	BatchSize	LrG	LrD	Beat1
实验一	5	1	200	128	0.0002	0.0002	0.999
实验二	1	5	200	128	0.0002	0.0002	0.999
实验三	1	1	100	128	0.0002	0.0002	0.999
实验四	1	1	100	128	0.0002	0.00005	0.999

淘系技术

表 4.2 实验参数

训练网络

训练 DCGAN 时，我们需要分别固定生成器和判别器。将生成器输出的动漫图片，与样本库所提供的动漫图片，输入到判别器中，并利用 sigmoid 函数对这两种图像进行分类。生成器可以根据返回的结果，不断反向调整和优化参数，从而使生成的动漫图像更加逼真，接近真实图像。

判别器的输入是一张动漫人物图片，计算出这张图片是真实图片的概率后，用 Sigmoid 函数将结果进行处理，使其输出值范围为 0~ 1 之间。我们可以根据判别器的计算结果，来量化它的辨别能力，根据反馈的结果，不断优化，从而使它对真实的动漫图像和生成的动漫图像的判别结果更加准确。

这两个过程交替重复，当达到指定的训练次数后，便可保存最后的网络模型，并检验训练效果，不断改进，基本的训练流程如下图所示：

图 4.3 DCGAN 动态博弈过程

训练结果展示与分析

由于此次 DCGAN 网络模型的主要功能是生成动漫人物图像，主要根据生成图形的色彩、清晰度、动漫人物面部完整性作为评价标准，用人眼主观对图像效果进行评价，Loss 函数用于对网络模型收敛程度的判定指标。

1. 当训练次数比例 G:D=5:1，G&D Learning rate=0.0002，batchsize=128，训练 200 个周期时，DCGAN 的第 1、49、99、199 个周期的图像效果如下图所示：

图 4.4 实验一生成图像对比

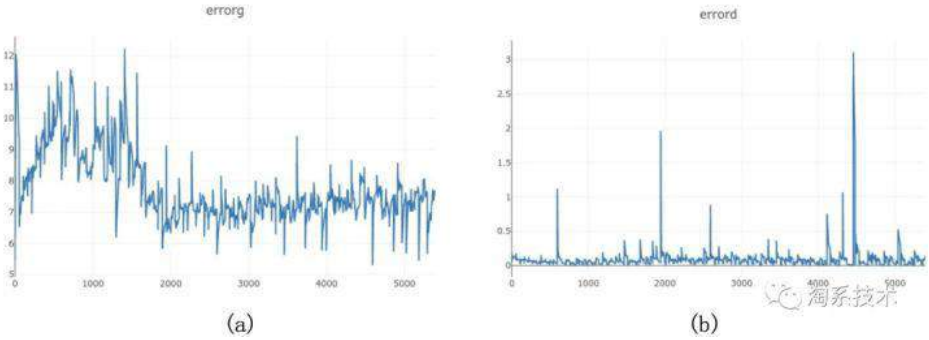


图 4.5 实验一生成器和判别器损失函数图像

以上实验结果分析可知，随着训练周期的提升，DCGAN 生成的图像质量不断提高，但动漫人物面部存在畸变，图像整体色彩发灰的问题。生成器得 Loss 函数数值振荡明显，网络收敛效果较差，而判别器的 Loss 函数数值整体低于生成器，表面判别器的网络收敛更快，能力更强。

2. 当训练次数比例 G:D=1:5，G&D Learning rate=0.0002，batchsize=128，训练 200 周期，DCGAN 的第 1、49、99、199 个周期的图像生成效果如下图所示：

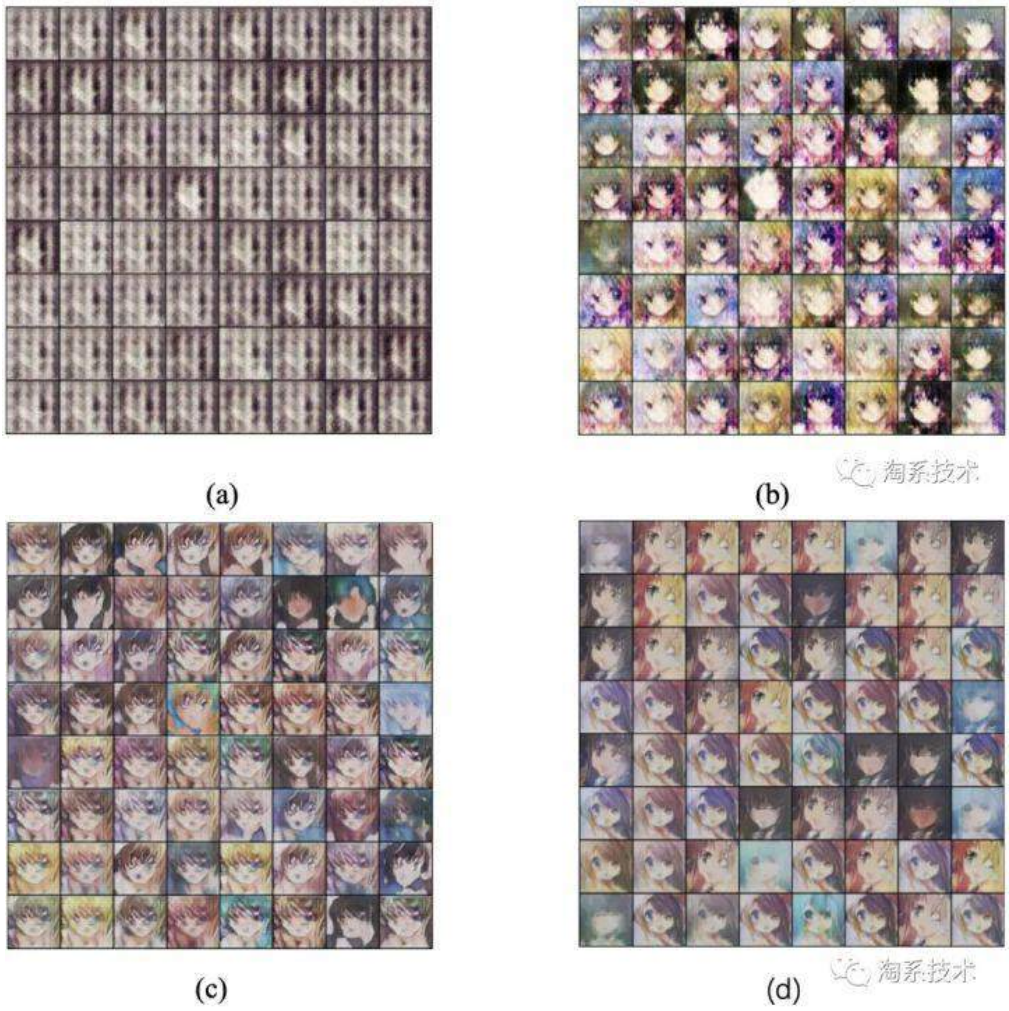


图 4.6 实验二生成图像对比

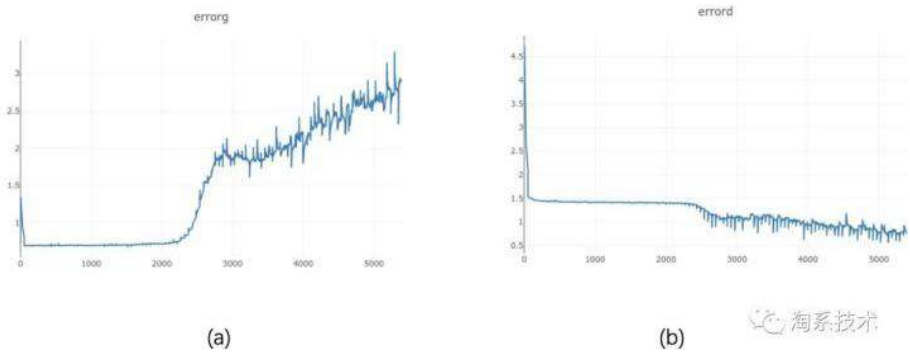


图 4.7 实验二生成器和判别器损失函数图像

以上实验结果分析可知，随着训练周期的提升，DCGAN 生成的动漫人物图像趋于同化，产生了模式坍塌，生成器输出一些不易被识别的图像，大部分结果都与我们预期的图像不符合。生成器的 Loss 函数值虽然比较低，但从训练中期开始上升。本实验生成的图像与实验一的图像进行对比可得，生成器训练次数小于判别器训练次数时，网络训练不稳定。

3. 当训练次数比例 G:D=1:1，G&D Learning rate=0.0002，batchsize = 128，训练第 1、49、79、99 个周期时，动漫人物图像的生成效果如下图所示：

图 4.8 实验三生成图像对比

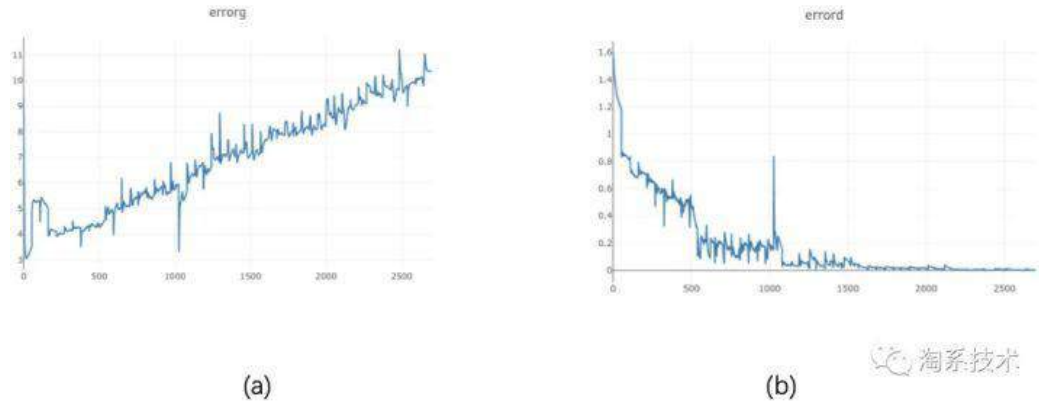


图 4.9 实验三生成器和判别器损失函数图像

由以上实验图像可知，当训练次数比例 G:D=1:1 时，在第 49 个训练周期时已经产生较为完整的图像，且图像细节优于前面两组实验，但是图像依然存在色调偏灰的问题，生成图像无法满足需求。生成器的损失函数一直处于增长状态，网络不断反向更新参数，收敛效果较差。

4. 当训练次数比例 G:D=1:1,G Learning rate=0.0002,D Learning rate = 0.00005,Batchsize=128，训练第 1、49、79、99 个周期时，动漫人物图像的生成效果如下图所示：

图 4.10 实验四生成图像对比

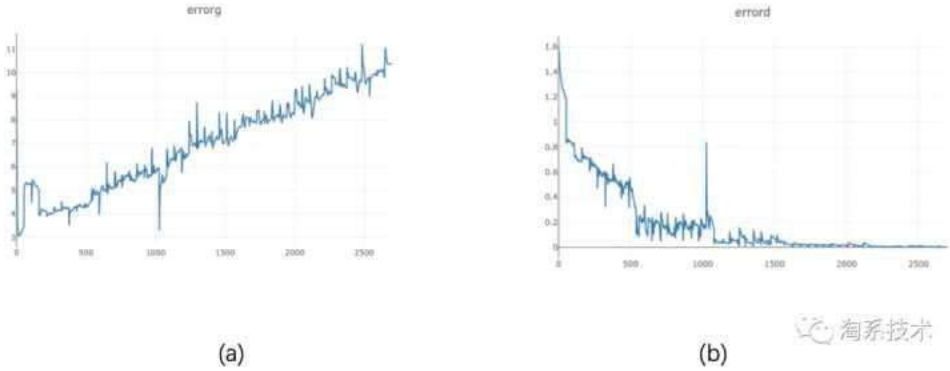


图 4.9 实验三生成器和判别器损失函数图像

由以上实验图像可知，当训练次数比例 G:D=1:1 时，在第 49 个训练周期时已经产生较为完整的图像，且图像细节优于前面两组实验，但是图像依然存在色调偏灰的问题，生成图像无法满足需求。生成器的损失函数一直处于增长状态，网络不断反向更新参数，收敛效果较差。

5. 当训练次数比例 G:D=1:1,G Learning rate=0.0002,D Learning rate = 0.00005,Batchsize=128，训练第 1、49、79、99 个周期时，动漫人物图像的生成效果如下图所示：

由以上实验图像可知，当训练次数比例 G:D=1:1 时，调整生成网络和判别网络学习率的比值，可以 使 DCGAN 生成的图像细节、色彩都更好。虽然图像在第 99 个训练周期出现了模式坍塌，但前 98 个周期的图片质量都较好，并且它们的损失函数在训练中期"此消彼长"，生成器的学习率在 5 以下，判别器的学习率基本从 40 个训练周期后变降到了 0.5 以下。后期可以通过动态调整判别器的学习率，减缓模式坍塌的发生。

通过上述实验结果对比可以得出，实验四设置的训练参数可以获得较好的动漫人物图形，其色彩、细节都强于其他组实验，并且损失函的数值变化状态较为均衡，因此选用第四次实验的第 97 个训练周期输出的网络作为目标网络。

DCGAN 模型测试

使用同一组超参数对测试集进行训练，图像生成效果与训练集基本一致，使用第四次实验的第 97 个训练周期 DCGAN 模型进行测试，选取 4 组图像，如下图所示：

图 4-12 生成器输出图像结果

观察以上实验结果可以得出，生成器所输出的图像风格符合动漫人物风格的要求，但细节上还存在缺陷。从动漫人物面部图像的完整性来看，人物的发型、面部轮廓的特征全都覆盖；约 11%的图像存在眼睛不清晰、不完整的问题；约 38%的图像存在嘴巴不完整的问题。

从动漫人物图像的色彩性来看，图像的“发灰”问题已经解决，生成的图像色彩与真实图像已经非常接近。由于动漫人物图像的特殊性，样本的面部细节与真实世界人类的面部特征存在较大差异，动漫人物的鼻子、嘴巴等特征都是由简单线条构成，具体的形状、颜色信息不够丰富，导致训练效果低于其它面部特征，可以通过收集面部细节更精细、面部特征更明显的图像样本改善这一问题。

通过 GAN 模型实现动漫图像自动生成的研究工作，在本文之前已经取得一定的研究成果。一种是通过风格迁移，将真实的人物图像转换为动漫风格；另一种是将已有的动漫人物图像，输入 DCGAN 模型中，对动漫图像的面部进行特征提取，最后训练出一组网络模型，输出动漫人物图像。本课题基于第二种方案，不同之处在于优化了默认的参数设置和传统的训练方式，在更短的训练周期内获得细节更完善、色彩更饱满的图像。

问题与解决方案

从上述训练过程可以得出，随着实验参数的调整、样本的优化，DCGAN 的效果不断提高，基于以上实验内容，总结出两个主要问题，并根据实践提出了对应的解决方案。

1. 判别网络收敛速度太快，导致生成器 Loss 数值持续上升。

解决方案 1：调整生成器和判别器单位周期内的训练次数。就目前实验数据来看，当生成器和判别器学习率相同的情况下，提高生成器的训练次数，DCGAN 训练效果更好。

解决方案 2：调整生成器和判别器学习率。一般情况下，进行 DCGAN 训练时两者在 Adam 优化器的学习率都使用参考值 0.0002。当出现其中一方收敛速度过快时，可以通过观察 Loss 函数的变化，按照一定比例调整学习率，在本课题中，判别器的网络收敛速度更快，导致虚假图像很快被识别。生成器反向更新参数后，导致后续每一层输入的数据分布也发生变化，Loss 函数值持续增加。可以通过降低判别器的学习率，间接影响生成器的网络收敛状况。

2. 生成图像质量较差，有畸变、模糊、模式崩溃等现象。

解决方案 1：检查图像样本库，在样本数量足够的情况下，检查样本中是否存在非动漫图像，动漫风格是否类似，样本的表情、发色等面部属性是否足够丰富。

解决方案 2：将训练次数比例和学习率结合，动态调整。判别器训练效果太好，会导致生成器反向调整参数，生成一些已经被识别为“真”的样本，特殊情况下，还输出许多面部特征畸

变的图像，导致样本缺乏多样性和准确性。根据 Loss 函数，查看损失值突变的训练周期，可以按照周期来动态调整生成器或判别器的学习率或训练次数，例如，每经过 10 个训练周期，将生成网络或判别网络的学习率递减为原来的 1/2。

总结

此次网络训练过程在前人研究的基础上，基本完成了深度神经网络的训练，得到一组高质量网络参数。但是受制于能力和硬件等方面的因素，还有许多不足之处，在以下几个方面需要进行更加深入的学习和探索：

1. 精细化样本库。由于图像样本库的素材由网络图片剪裁而成，动漫人物形象风格、色彩、背景都有较大差异，对网络训练产生干扰，影响生成图像的质量。采集白色或统一背景的动漫人物面部图像，可以提高生成图片质量。
2. 增加网络模型深度。可以使生成的图片细节更加完善，图片质量清晰，虽然目前实验环境有限制，但是生成更加清晰的动漫图像可以扩展其使用范围，更加满足动漫创作者的需求。
3. 改善生成图像质量。随着训练次数的增多，部分会产生图像色彩失真现象，图片看起来色调偏灰，可以在图片输出之前对生成图片质量判断，根据色彩饱和度等信息，使用 Torch 中的图像增强函数，对图片色彩进行调整。
4. 优化工程界面。软件的普及离不开其易用、美观的特点。目前本课题还需要利用命令行完成运行，通过一键点击自动生成动漫图像，并用简洁的前端页面展示是接下来的工作目标。

链接：

https://github.com/chenyuntc/pytorch-book?spm=ata.13261165.0.0.4061525dFyDth4https://zhuanlan.zhihu.com/p/24767059?spm=ata.13261165.0.0.4061525dK3i1w7https://make.girls.moe/#/

Q: 主动学习(Active learning)算法的原理是什么，有哪些比较具体的应用？

作者 | 初类
出品 | 淘系技术

搞风控的同学们可能都遇到过坏样本太少的问题，最近同事提示可以研究一下主动学习（active learning），据说这个算法可以减少人工打标的数量、成本？

本篇可以作为主动学习入门篇，介绍基本原理和知识，并以阿里巴巴淘系的应用场景作为案例分享。

前言

在大数据和算力的助力下，深度学习掀起了一波浪潮，在许多领域取得了显著的成绩。以监督学习为主的深度学习方法，往往期望能够拥有大量的标注样本进行训练，模型能够学到更多有价值的知识（如下左图展示了 3 组常见的图像分类数据集，拥有上万的标注样本）。

然而，实际应用场景的标注样本严重稀缺。并且，标注大量样本将产生昂贵的标注成本（如下右图所示，标注一张 X 射线图需要 5 分钟和 30 元左右的成本，一张 CT 图需要 20 分钟和 70 元的成本）。

数据集	训练集样本数量	验证集样本数量
MNIST	50,000+	10,000
CIFAR10	50,000	10,000
ImageNet	15 million	

（模型的性能依赖于大量的标注样本）

	X-Ray	CT
时间成本	4-5 min/image	15-20 min/image
金钱成本	20-30 RMB/image	50-70 RMB/person

（标注样本很贵！）

在庞大而复杂的淘系电商场景中，类似的需求比比皆是：例如，咸鱼&躺平和洋淘等社区内容的治理，拍立淘的以图搜图，服饰分类（例如，iFashion）等场景都存在标注样本严重稀缺的问题。

综上，在实际应用场景中，如何"在模型达到目标性能的前提下，尽可能地减少标注成本"是一项亟需解决的挑战。

主动学习作为机器学习的一个子领域，旨在以尽可能少的标注样本达到模型的目标性能，广泛应用于实际需求中。本文的定位是主动学习方法的入门篇，主要介绍的内容包括：1）详细地介绍主动学习的基础知识；2）简要地介绍主动学习在学术界的研究现状；3）主动学习实践部分将简单介绍几个图像分类的案例；4）文末将给出本文的参考文献和相关资料。

主动学习的基本知识：

主动学习的概念和基本流程：

主动学习是什么

Burr Settles[1] 的文章《Active Learning Literature Survey》详细地介绍了主动学习："主动学习是机器学习的一个子领域，在统计学领域也叫查询学习或最优实验设计"。

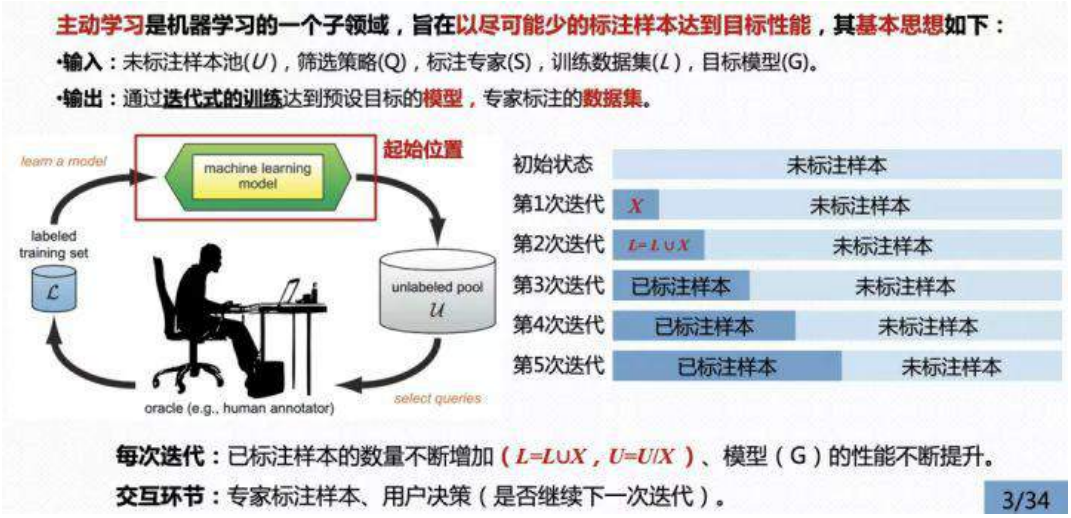
主动学习方法尝试解决样本的标注瓶颈，通过主动优先选择最有价值的未标注样本进行标注，以尽可能少的标注样本达到模型的预期性能。

主动学习的基本流程

如下图所示，主动学习方法是一个迭代式的交互训练过程，主要由五个核心部分组成，包括：未标注样本池（unlabeled pool，记为 U）、筛选策略（select queries，记为 Q）、标注者（human annotator，记为 S），标注数据集（labeled training set，记为 L），目标模型（machine learning model，记为 G）。

主动学习将上述五个部分组合到同一个流程中，并通过如下图所示的顺序，以不断迭代的训练方式更新模型性能、未标注样本池和标注数据集，直到目标模型达到预设的性能或者不再提供标注数据为止。

其中，在每次迭代过程中，已标注样本的数量不断增加，模型的性能也随之提升（理想情况）。在实际应用中，应尽可能保证标注者的准确率，缓解模型在训练初期学偏（此处特指错误标注的样本导致）的情况。

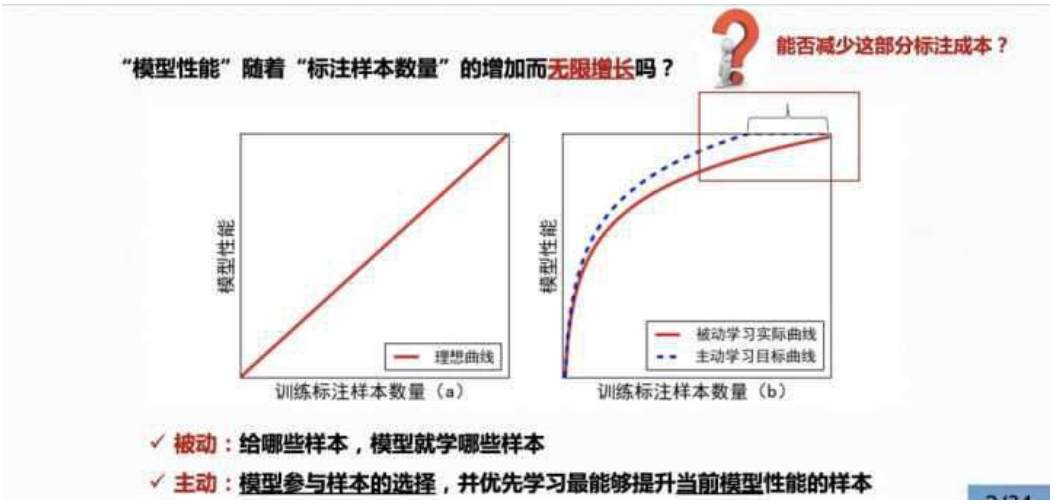


主动学习和被动学习、半监督学习的关系

主动学习和被动学习

如下图(a)所示，红色实线表示理想情况下模型性能随着训练标注样本数量的增多而无限地提升。实际情况下往往是如下图(b)的红色实线所示，模型的性能不是随着标注数据量的增多而无限地提升。

此外，每个模型都会有与之对应的瓶颈性能(peak performance)，研究者通过增加训练数据以及调参使之不断逼近瓶颈性能。主动学习核心解决的问题正是如何使用尽可能少的标注数据达到模型的瓶颈性能，从而减少不必要的标注成本。如下图(b)的蓝色虚线所示，主动学习根据合适的策略筛选出最具有价值的样本优先标注并给模型训练，从而以更少的标注样本达到模型的瓶颈性能。



主动学习和半监督学习

在机器学习领域中，根据是否需要样本的标签信息可分为“监督学习”和“无监督学习”。此外，同时利用未标注样本和标注样本进行机器学习的算法可进一步归纳为三类：半监督学习、直推式学习和主动学习 。

文献[21]简要介绍了主动学习与半监督学习的异同点：“半监督学习和主动学习都是从未标记样例中挑选部分价值量高的样例标注后补充到已标记样例集中来提高分类器精度，降低领域专家的工作量。”

但二者的学习方式不同：半监督学习一般不需要人工参与，是通过具有一定分类精度的基准分类器实现对未标注样例的自动标注；而主动学习有别于半监督学习的特点之一就是需要将挑选出的高价值样例进行人工准确标注。

半监督学习通过用计算机进行自动或半自动标注代替人工标注，虽然有效降低了标注代价，但其标注结果依赖于用部分已标注样例训练出的基准分类器的分类精度，因此并不能保证标注结果完全正确。

相比而言，主动学习挑选的样本是人工标注，尽可能引入最少的错误类标。值得一提的是，目前已有许多研究者尝试将主动学习和半监督学习进行结合，并取得了不错的效果。

主动学习的基本策略

样本的筛选策略直接关系到模型能够节约标注成本的程度。例如，使用不确定性策略比随机采样策略能够节约更多的标注样本[4,5] 。因为随机采样策略既没有利用到模型的预测信息，也没有利用到大量未标注样本池的结构信息，仅凭随机采样决定优先标注的样本。而不确定性策略通过与模型的预测信息进行交互，优先筛选出相对当前模型最有价值的样本。本节将围绕部分经典的筛选策略展开讨论。

1. 随机采样策略(Random Sampling，RS): RS 不需要跟模型的预测结果做任何交互，直接通过随机数从未标注样本池筛选出一批样本给专家标注，常作为主动学习算法中最基础的对比实验。
2. 不确定性策略(Uncertainty Strategy，US):US 假设最靠近分类超平面的样本相对分类器具有较丰富的信息量，根据当前模型对样本的预测值筛选出最不确定的样本。

US 包含了一些基础的衡量指标:

- 最不确定指标(Least Confidence，LC)将预测概率的最大值的相反数作为样本的不确定性分数。
- 边缘采样(Margin Sampling，MS)认为距离分类超平面越近的样本具有越高的不确定性，常与 SVM 结合并用于解决二分类任务，但在多分类任务上的表现不佳。
- 多类别不确定采样(Multi-Class Level Uncertainty，MCLU)是 MS 在多分类问题上的扩展，MCLU 选择离分类界面最远的两个样本，并将它们的距离差值作为评判标准。MCLU 能够在混合类别区域中筛选出最不确信度的样本，如式(2.3)所示。其中，xj 表示被选中的样本，C 表示样本 xi 所属的类别集合，c+ 表示最大预测概率对应的类别，f (xi, c) 表示样本 xi 到分类超平面的距离。
- 熵值最大化(Maximize Entropy，ME)优先筛选具有更大熵值的样本，熵值可以通过计算

$$EntropyScore = -\sum_{i=1}^{C} pi \times log(pi)$$

得到，其中 pi 表示第 i 个类别的预测值。

样本最优次优类别(Best vs Second Best, BvSB)[79]主要是针对多分类问题的一种衡量指标，并且能够缓解 ME 在多分类问题上效果不佳的情况。BvSB 只考虑样本预测值最大的两个类别，忽略了其他预测类别的影响，从而在多分类问题上的效果更佳。

$$x_j = \arg\min_{x_i \in U} \left\{ \max_{c \in C} f(x_i, c) - \max_{c \in C \setminus c^+} f(x_i, c) \right\} \tag{2.3}$$

3. 委员会投票(Query by Committee，QBC):QBC[31]是一种基于版本空间缩减的采样策略，核心思想是优先选择能够最大程度缩减版本空间的未标记样本。

QBC 包括两个基本步骤:

- 使用多个模型构成委员会;
- 委员会中所有的模型依次对未标注样本进行预测并优先筛选出投票最不一致的样本进行标注。

由于 QBC 在实际应用的过程中需要训练若干个模型，导致具有较高的计算复杂度。基于此，熵值装袋算法(Entropy Query-By-Bagging，EQB)[80]和自适应不一致最大化(Adaptive Maximize Disagree，AMD)被提出并缓解了计算复杂度问题。其中，EQB 同时引入了 bagging 继承方法以及 bootstrap 采样;AMD 主要针对高维数据，将特征空间划分为一定数量的子集并构造委员会。

4. 其他经典的策略:梯度长度期望(Expected Gradient Length，EGL) 策略根据未标注样本对当前模型的影响程度优先筛选出对模型影响最大的样本;EGL [4] 是代表性方法之一，能够应用在任意基于梯度下降方法的模型中。方差最小(Variance Reduction，VR)策略通过减少输出方差能够降低模型的泛化误差[81,82];Ji 等[82]提出了一种基于图的 VR 衡量指标的主动学习方法，通过将所有未标注样本构建在同一个图中，每个样本分布在图中每个结点上。紧接着，通过调和高斯随机场分类器直接预测未标注样本所属的

标签;在优化的过程中，通过挑选一组未标注样本进行预测并获得对应的预测类别，使得未标注样本的预测类别方差最小。

主动学习的扩展方法

近年来，主动学习策略在很多实际应用场景中取得显著的效果。但同时也存在一些亟需解决的挑战。例如，不确定性策略只关注样本的不确定性，在 BMAL（批量式主动学习方法，每次迭代筛选出 N>1 的样本数量）场景下会产生大量具有冗余信息的样本。因此，仅使用单一的策略尚未能最大程度地节约标注成本。

本节将围绕本文的核心工作简要地介绍几种主动学习的扩展方法。

1. 组合多种基本策略的主动学习方法:组合策略将多个基本策略以互补的方式进行融合，广泛应用于图像分类任务中[36,37,38,83]。其中，Li 等[36]基于概率分类模型提出一种自适应的组合策略框架。Li 等[36]通过信息密度指标(Information Density Measure)将未标注样本的信息考虑在内，弥补了不确定性策略的不足。如算法 2-2 所示，该算法能够自然地扩展到更多的组合策略。

算法 2 - 2 组合多种基本策略的主动学习方法

输入： 初始模型 G，未标注样本池 U，标注专家 S，策略权重 B，基本策略 Q_1 和 Q_2
输出： 最终模型 G 及其性能，标注数据集 L

初始化： $B = [0.1, 0.2, ..., 1]$，假设 Q_1 和 Q_2 分别表示不确定性策略和信息密度策略
repeat:
  if $L \neq \varnothing$ **then**
    使用标注数据集 L 对模型 G 进行训练
  end if
  for $x \in U$ **do**
    根据策略 Q_1 计算得分并记为 $f(x)$
    根据策略 Q_2 计算得分并记为 $d(x)$
    根据公式 $h_\beta(x) = f(x)^\beta d(x)^{1-\beta}$ 计算组合策略的最终得分
  end for
  $X = \varnothing$
  for $\beta \in B$ **do**
    选出一个样本 x，使其满足 $x = \arg\min_{x \in U} h_\beta(x)$
    将样本 x 加入候选集合 X 中：$X = X \cup x$
  end for
  从集合 X 中筛选出最佳样本 x^* 并给专家标注
  更新相关的数据集：$L = L \cup x^*$，$U = U / x^*$
until 模型达到预定的准确率或者标注成本达到上限

2. 结合半监督学习(Semi-Supervised Learning)的主动学习方法:自训练(Self-training)算法作为半监督学习的一种基础方法，其核心步骤如算法 2-3 所示。由于自训练算法在训练过程中会根据模型的预测信息，挑选合适的样本及其对应的预测标签加入训练集，而且初始化少量的标注样本能够保证模型的初始性能，因此初始化训练环节对其后续的学习过程至关重要。

半监督学习算法需要解决的挑战之一是:在训练的过程中容易引入大量的噪声样本，导致模型学习不到正确的信息。部分研究员们通过构建多个分类器的协同训练算法缓解噪声样本，如 Co-Training[84] 和 Tri-Training[85]。

算法 2 - 3 自训练算法的基本步骤

输入: 初始模型 G，未标注样本池 U，阈值 *threshold*
输出: 最终模型 G 及其性能
1: **初始化:** 标注少量的训练样本 L
2: **repeat**:
3: 使用标注数据集 L 训练模型 G
4: 使用模型 G 对未标注样本池 U 中的每个样本进行预测，并记录对应的预测值
5: 根据样本的预测值计算对应的置信度 *confidence*
6: 挑选满足 *threshold* 的样本及其对应的预测标签构成集合 X
7: 更新相关的数据集: $L = L \cup X$, $U = U/X$
8: **until** 满足预先设定的结束条件

3. 结合生成对抗网络的主动学习方法:生成对抗网络(Generative Adversarial Networks，GAN)模型以无监督的训练方式对大量未标注样本进行训练，并通过生成器产生新的样本。

经典的 GAN[15] 主要包括生成器和判别器等两个核心部分，两者以互相博弈的方式进行对抗训练，直到两者达到一个动态均衡的状态。GAN 的目标函数如式(2.4)所示，其中，V(G,D)=Ex~ Pdata[logD(x)]+Ex~ PG [log(1−D(x))] 表示数据真实分布 x ~ Pdata 与生成模型得到的分布 x ~ PG 之间的差异。文献[19,50]将生成器和主动学习策略进行融合并构建目标函数，通过解决优化问题控制生成器产生的样本。

$$G^* = \arg\min_G \max_D V(G, D) \tag{2.4}$$

主动学习方法的基本评价指标

本文侧重介绍主动学习方法在保证不损失模型准确率的情况下，节约标注成本的性能，评价指标如下式所示。其中，SavedRate 表示主动学习方法相对于全样本训练减少的标注成本;ExpertAnnotated 表示当模型达到预定的目标性能时专家标注的样本数量;Full Samples 表示当前数据集提供的未标注样本数量，即全样本训练时所使用的标注样本数量。本文涉及的实验会先进行全样本训练，并分别记录最佳验证集准确率作为主动学习相关算法的目标准确率。

例如，在某组数据集中使用 AlexNet 模型对 Full Samples 张标注图像进行训练，记录训练过程中最佳的验证准确率(Best accuracy)并将其作为主动学习的目标准确率(Targetaccuracy);随后，模型通过迭代过程不断提升性能，当达到目标准确率时，记录专家所标注的样本数量 ExpertAnnotated;此时，就可以算出 SavedRate 的值，即该方法能够节约多少标注成本。此外，我们也会将主动学习方法与一些常见的方法进行比较，比如 RS 策略常用于基准对比实验(baseline)。

$$SavedRate = 1 - \frac{ExpertAnnotated}{FullSamples} \tag{2.5}$$

早期的主动学习面临的挑战及其解决方案

多类分类问题

在处理多类分类问题时，基于 Margin Sampling 的样例选择标准忽略了样例可能属于其他类别的信息，因此所选样例质量较差。基于熵的方法"基于不确定性的主动学习算法研究（2011）"虽考虑了样例从属于每个类别的概率，但在多类分类问题中，样例的熵也会受到那些不重要类别的干扰。

文献"Multi-class active learning for image classification（2009）"提出了基于最优标号和次优标号的准则（BvSB），考虑样例所属概率最高的前 2 个类别，忽略剩余类别对样例选择标准产生的干扰。

文献"基于主动学习和半监督学习的多类图像分类（2011）"将 BvSB 和带约束的自学习（Constrained self-training，CST）引入到基于 SVM 的图像分类中，显著提高了分类精度。

样本中的孤立点

若选择样例时能综合考虑样其代表性和不确定性，通常可避免采集到孤立点。

文献"Active Learning by querying informative and representative examples（2010）"中提出了一种综合利用聚类信息和分类间隔的样例选择方法；

文献"Active Learning using a Variational Dirichlet Processing model for pre-clustering and classification of underwater stereo imagery（2011）"提出了一种利用预聚类协助选择代表性样例的主动学习方法；

文献"Dual strategy active learning（2007）"利用样例的不确定性及其先验分布密度进行样例选择以获取优质样例；

文献"基于样本不确定性和代表性相结合的可控主动学习算法研究 （2009）"将样例的分布密度作为度量样例代表性的指标，结合以熵作为不确定性指标，提出了一种基于密度熵的样例选择策略，有效解决了孤立点问题给样例选择质量造成的影响。

训练集样本冗余

如下图所示，蓝色圆圈所表示的新训练样本中，样例 1 与分类超平面的距离比样例 2 近，根据 BvSB 准则应当挑选样例 1 进行标注并补充到训练集中；但紧挨着样例 1 的绿色样例 a 已经在训练集中，此时若再加入样例 1 则对分类界面影响甚微。

相比而言，将样例 2 补充到训练集中，对当前分类模型的训练贡献度更大。通过上述分析可知，主动学习中的样例选择度量主要分为 2 种：

- 不确定性度量；
- 差异性度量或代表性度量；

样例的不确定性一般可通过计算其信息熵获得，样例的代表性通常可根据其是否在聚类中心判断，而样例的差异性则可通过计算余弦相似度（基于采样策略的主动学习算法研究进展，2012）或用高斯核函数（基于多特征融合的中文评论情感分类算法，2015）获得。

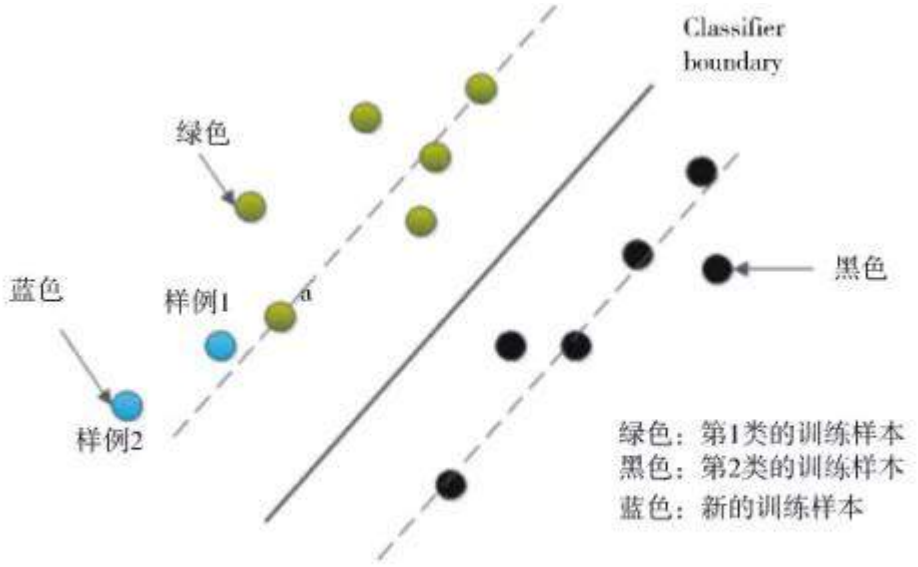


图 4　样例冗余问题举例

Fig.4　An example of sample redundancy problem

不平衡数据集

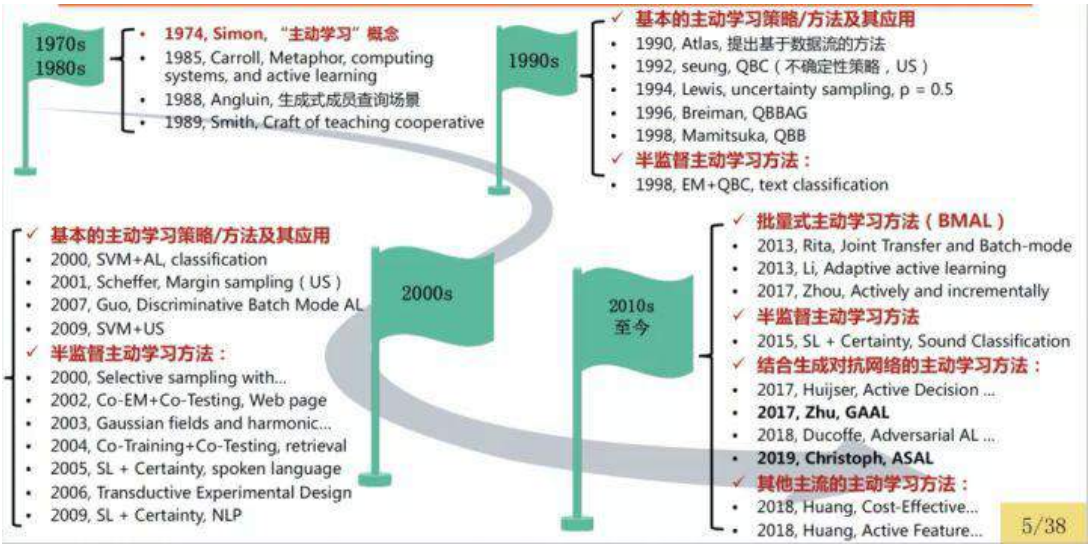
文献"一种新的 SVM 主动学习算法及其在障碍物检测中的应用（2009）"提出 KSVMactive 主动学习算法；文献"基于主动学习的加权支持向量机的分类（2009）"提出了改进的加权支持向量机模型；文献"基于专家委员会的主动学习算法研究（2010）"提出了基于 SVM 超平面位置校正的主动学习算法。

主动学习的研究现状

本节将围绕如下要点对主动学习方法的研究现状展开讨论，包括：

- 基于未标注样本池的主动学习策略；
- 批量式主动学习方法，侧重于组合式策略以及引入聚类算法的主动学习方法；
- 半监督主动学习方法；
- 结合生成对抗网络的主动学习方法。

此外，主动学习方法在近几年的进展不仅局限于上述归类的方法，本节将其总结在"其他主流的主动学习方法"（本文涉及的参考文献，都可以通过文末的参考文献提供的链接中获取）。



主动学习方法概述

主动学习作为机器学习的一个子领域，核心思想是通过一些启发式策略找到相对最具有"价值"的训练样本，使得模型能够以尽可能少的标注样本达到甚至超过预期的效果。

主动学习的概念是 Simon[23]在 1974 年提出。随后，主动学习方法在许多领域中层出不穷，并进一步被归纳为生成式成员查询(Membership Query Synthesis)、流式主动学习方法(Stream-Based Selective Sampling)和基于未标注样本池的主动学习方法(Pool-Based Sampling)等经典的场景[4]。

Angluin 等[24]于 1988 年提出了生成式成员查询场景，模型通过预设的条件控制生成新的样本并向标注专家询问标签；由于当时生成模型的能力有限，并且无法较好的控制生成所需的样本，因此这类方法的应用范围未被推广。

Atlas 等[25]在 1990 提出了基于数据流的方法，模型按照顺序依次判断是否需要对样本进行标记。由于基于数据流的方法不需要将所有样本统一放在池中，因此适用于存储空间较小以及处理能力有限的情况(如，应用到移动设备)，但存在最大的缺陷是无法获取样本的结构分布。

相较之下，基于未标注样本池的主动学习方法[26]将大量未标注样本构成未标注样本池，通过设计样本筛选策略从未标注样本池中筛选出最有"价值"的样本优先进行标注。此外，伴随着互联网的热潮以及数据采集技术的不断提升，很多领域能够以廉价的成本获取大量的未标注数据。因此，基于未标注样本池的主动学习方法最流行并且广泛应用于不同的领域中，在机器学习和数据挖掘的应用中处于非常重要的地位。

基于未标注样本池的主动学习方法

样本筛选策略的质量直接影响到基于未标注样本池的主动学习方法的效果。目前，一些手工设计策略不断被提出并应用到主动学习方法中，如不确定性策略和代表性策略。

文献[27,28]通过计算信息熵(entropy)表示最不确定的样本。文献[12,29,30]使用 SVM 作为目标分类器，通过选择距离支持向量最近的样本作为最不确定的样本。Seung 等[31]首次提出了基于委员会的筛选算法(Query-by-Committee，QBC)，首先训练了一组分类器组成委员会。紧接着，以委员投票的方式决定筛选哪个样本作为最不确定的样本。

随后，一些基于 QBC 的改进方法不断被提出:例如，Breiman 等[32]基于 Bagging 提出的 Query-by-Bagging(QBBAG)以及 Mamitsuka 等[33]基于 Boosting 提出的 Query-by-Boosting(QBB)。

对于样本的代表性策略，文献[34,35]通过使用未标注样本的先验密度(PriorDensity)作为不确定性指标的权重，从而达到利用未标注样本的目的。Settles 等[28]提出一种相似的框架，使用 cosine 距离衡量信息密度(InformationDensity)。

批量式主动学习(BatchModeActiveLearning，BMAL)方法

目前，大多数主动学习方法存在一个共同的问题:串行地筛选样本，即每次迭代选择一个样本进行标注，这种方式非常低效且无法满足大多数实际需求。在实际应用中，往往需要以分布式的形式并行处理，多名标注专家同时在不同的环境下标注样本。

BMAL 旨在每次迭代中能够产生一批未标注样本，同时提供给多名标注者，从而极大地提升了应用效率。BMAL 的发展历程中，起初，有研究尝试将很多不同的预测模型应用到不同的策略中。但他们在筛选样本时，只使用了单一的不确定性指标或者多样性指标的主动选择策略，导致所挑选的样本中存在大量的冗余信息，从而造成了额外的标注成本。

基于此，Li 等[36]提出一种新颖的自适应组合式的样本筛选策略，将不确定性策略和信息密度指标进行结合。在每次迭代中，通过自适应地调整两种策略的权重，从而选择最具有"价值"的样本给专家标注，并在三组图像分类数据集上验证了所提出方法的有效性。

Gu 等[37]提出了一种面向多分类的 BMAL，通过组合不确定性策略和多样性策略，并在两组图像分类的数据集上进行验证，实验结果表明该方法能够挑选出同时满足最不确定性和最具多样性的样本。Zhou 等[38]通过组合不确定性指标和多样性指标，同时引入了迁移学习和数据增强等技术，提出了 AIFT 方法并将其应用到医疗图像领域，验证了该方法至少能够减少一半的标注成本。Cardoso 等[39]在传统 BMAL 的基础上提出了一种排序批量式主动学习方法(RBMAL)，通过生成一个优化过的排序表决定样本被标注的优先级。RBMAL 避免了标注专家频繁等待被选中的未标注样本，实验结果表明 RBMAL 能够在保证甚至提升模型性能的条件下显著地减少标注成本。此外，为了更加充分利用大量未标注样本的信息，有研究员[40,41,42]尝试将聚类算法引入主动学习中。

然而，目前大多数聚类方法都是先通过手工提取特征再聚类，在很大一定程度上局限于特征的质量。我们尝试将卷积自编码聚类算法[43]应用到 BMAL 中，通过将特征提取和聚类算法以端到端的形式整合到同一个模型里（本文暂不展开介绍）。从而既能够提升聚类性能，又能够利用卷积神经网络的优势处理更复杂的图像。

半监督主动学习方法

半监督学习能够在少量标注成本的情况下训练模型，通过挑选出预测结果较明确的样本并由模型直接给标签，但是容易产生噪声标签。

而主动学习则是挑选预测结果最不确定的样本给专家标注，能够保证标签质量。因此，半监督学习方法和主动学习方法的结合能够在一定程度上互补优缺。
1998 年，McCallumzy 等[44]首次组合了 QBC 和期望最大化(EM)算法，使用朴素贝叶斯方法作为分类器并在文本分类任务上进行实验。

随后，Muslea 等[45]提出了一种 QBC 的改进方法，联合测试方法(Co-Testing)，通过分别在不同视角训练的两个分类器共同筛选样本给专家标注，并将其与联合期望最大化(Co-EM)算法结合。Zhou 等[46]尝试将 Co-Testing 和 Co-Training 方法进行结合并在图像检索任务中验证了算法的优势。

此外，文献[47,48,49]组合了不确定性策略和自学习方法(Self-Training)。上述方法将半监督学习和主动学习巧妙地结合，充分利用各自的优势并弥补不足，取得了显著的成绩。然而，目前的半监督主动学习方法尚未对噪声样本进行有效地处理，因此仍会对模型造成不小的影响。

结合生成对抗网络的主动学习方法

GANs 对提升主动学习方法的样本筛选效率具有重要的意义。

文献[19,50]将主动学习策略结合生成器构建目标函数，通过解决优化问题使得生成器直接生成目标样本，提升了筛选样本的效率。Huijser 等[20]首先使用 GAN 沿着与当前分类器决策边界垂直的方向生成一批样本。

紧接着，通过可视化从生成的样本中找出类别发生改变的位置，并将其加入待标注样本集。最后，通过大量的图像分类实验验证了该方法的有效性。此外，除了图像分类任务以外，主动学习方法与 GAN 的结合也广泛应用到其他领域中，例如离群点检测[21]。

其他主流的主动学习方法

Huang 等[51]提出一种针对深度神经网络的主动学习方法，能够用更少的标记样本将预训练好的深度模型迁移到不同的任务上，从而降低深度神经网络的学习代价。

Huang 等[52]提出一种结合主动学习和矩阵补全技术的方法，能够在特征缺失严重的情况下有效利用标记信息，节省特征提取代价。

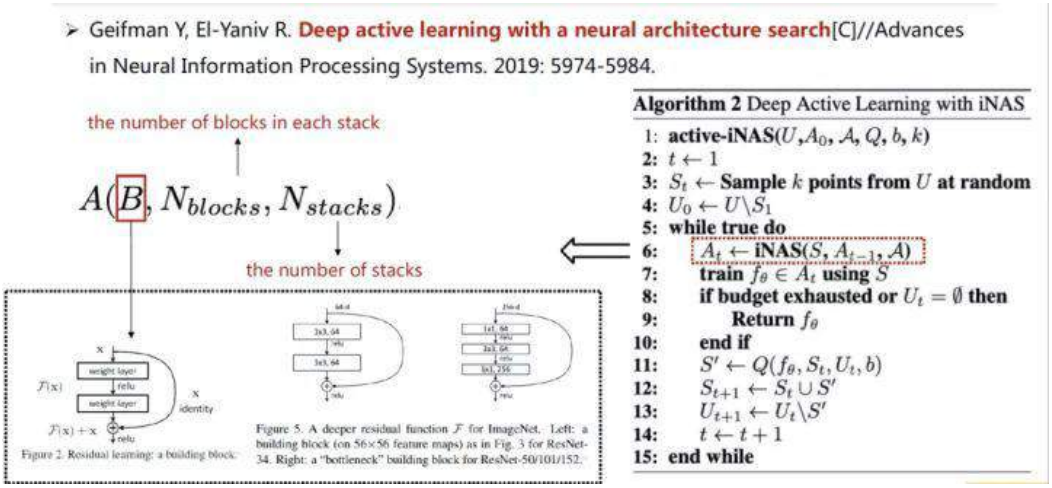
Chu 等[53]认为应用在不同数据集上的主动学习策略存在有效的经验，并且这些经验可以被迁移到其他数据集中进而提升模型或者策略的性能。作者尝试将模型迁移到不同的数据集中，实验部分证明了当前大多数策略不仅存在有效的经验，而且经验能够被迁移到不同的数据集中，并提升特征学习任务的性能。

NAS + Active Learning

最后，值得一提的是，考虑到上述归纳的主动学习方法中，任务模型是根据先验知识从现成的模型中筛选，即模型的网络结构是固定的。存在如下缺陷：

- 很多领域没有现成的模型可用，例如医疗图像领域；
- 在前期的迭代过程中，标注样本量较少，固定网络结构（通常会比较复杂一点）的模型可能会陷入过拟合。

如下图所示，Geifman 等人首次尝试将 NAS 应用到主动学习方法中，使得模型的网络结构能够自适应新增的标注数据。实验结果表明，加入 NAS 后的主动学习方法的效率显著地优于固定网络结构的主动学习方法。



主动学习实践：牛刀小试

主动学习如何减少标注样本的简单案例

如下图所示，文献《Active Learning Literature Survey》提供了一个基于 pool-based 的主动学习案例。

其中，数据集（toy data）是从高斯分布产生的 400 个样本，任务是 2 分类问题（每个类有 200 个样本），如(a)图所示将这些数据映射在 2D 特征空间上；图(b)使用了逻辑回归模型，通过训练随机选择的 30 个标注样本，得到 70%的验证精度，蓝色线表示决策边界（decision boundary）；图(c)同样使用逻辑回归模型，但训练的 30 个标注样本是通过主动学习策略（uncertain strategy）选择而来，达到 90%的验证精度。

这个简单的案例体现了引入主动学习策略所带来的效果，使用 30 个标注样本能够提升 20%的精度。值得注意的是，上述 2 分类的样本分别 200 个，样本数据非常平衡。但是在实际应用中，分类样本数据比例往往不能达到 1:1，相关领域的研究者正在尝试解决这类问题。

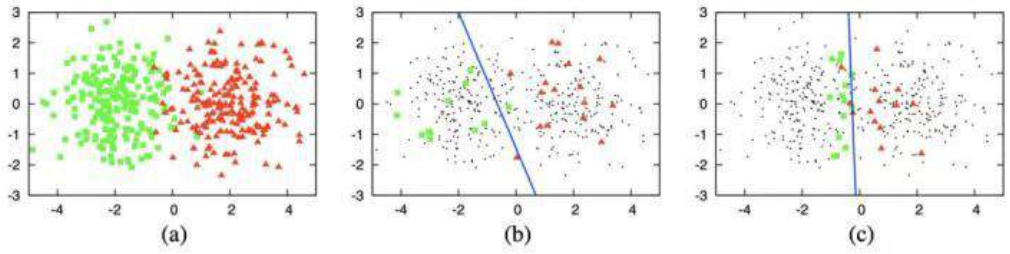


Figure 2: An illustrative example of pool-based active learning. (a) A toy data set of 400 instances, evenly sampled from two class Gaussians. The instances are represented as points in a 2D feature space. (b) A logistic regression model trained with 30 labeled instances randomly drawn from the problem domain. The line represents the decision boundary of the classifier (accuracy = 0.7). (c) A logistic regression model trained with 30 actively queried instances using uncertainty sampling (accuracy = 0.9).

图像分类数据集的实践

如算法 2-1 所示给出了"基于为标注样本池的主动学习方法"，本文也在第一部分详细地介绍了主动学习的基本流程，此处不再赘述。

算法 2 - 1 基于未标注样本池的主动学习基本算法

输入： 初始模型 G，未标注样本池 U，标注专家 S，批模式数量 k，筛选策略 Q
输出： 最终模型 G 及其性能，标注数据集 L

repeat:
  if $L \neq \varnothing$ **then**
    使用标注数据集 L 对模型 G 进行训练
  end if
  使用模型 G 对未标注样本池 U 中的每个样本进行预测，并记录对应的预测值
  结合样本的预测值并根据筛选策略 Q 计算每个未标注样本的得分情况 *score*
  根据 *score* 进行降序排序并挑选前 k 个样本给专家 S 标注，构成集合 X
  更新相关的数据集：$L = L \cup X$，$U = U/X$
until 模型达到预定的准确率或者标注成本达到上限

本文分享的实践部分，按照算法 2-1 分别对 MNIST、Cifar-10 和 Dog-Cat 三个数据集进行实验（分类模型使用了 AlexNet，深度学习框架使用了 PyTorch）。

如下表所示，在 MNIST 数据集的实验中（train_num=55000，val_num = 10000）：

- 使用全部 5.5 万的训练数据直接训练模型，在 1 万个验证集得到的准确率为 98.99%；
- 使用主动学习的不确定性策略（Uncertainty Strategy），只需要 5000 张标注样本，在相同的 1 万个验证集得到的准确率就达到 99.14%。

此外，将训练好的模型对剩余的 50000（55000-5000）张样本进行预测，得到 99.70% 的效果。由此可见，仅仅使用不确定性策略在 MNIST 数据集上，就能够显著地减少大量的标注成本。值得注意的是，表中所示的三组图像分类数据集 acc_left_active_samples 的准确率都很高。这部分样本表示未被主动学习策略筛选中的样本，即当前模型已经具备识别这部分样本的能力。

因此，模型在训练数据集的准确率达到 99.4% 时，使用当前模型对 acc_left_active_samples 这部分样本进行预测的精度也同样在 99.378% 左右，甚至更高。

Method	dataset	train set	val set	model	val-acc	acc_left_active_samples
Just Train	Dog-Cat	25000	10000	AlexNet	92.76%	\
active learning	Dog-Cat	6000	10000	AlexNet	92.76%	97.85% (13700/14001)
Just Train	cifar10	50000	10000	AlexNet	90.00%	\
random select	cifar10	44000	10000	AlexNet	90.06%	\
active learning	cifar10	17750	10000	AlexNet	90.16%	99.39%
Just Train	MNIST	55000	10000	AlexNet	98.99%	\
random select	MNIST	55000	10000	AlexNet	98.98%	\
active learning	MNIST	5000	10000	AlexNet	99.14%	99.70%

问题 1：主动学习为什么有时还能提升分类模型的准确率？

杨文柱等人发表的"主动学习算法研究进展"给出的解释是：标注样本可能存在低质量的样本，会降低模型的鲁棒性（模型过渡拟合噪声点）。如何高效地筛选出具有高分类贡献度的无类标样例进行标注，并补充到已有训练集中逐步提高分类器精度与鲁棒性是主动学习亟待解决的关键问题。

问题 2：不确定性策略具体怎么实现?

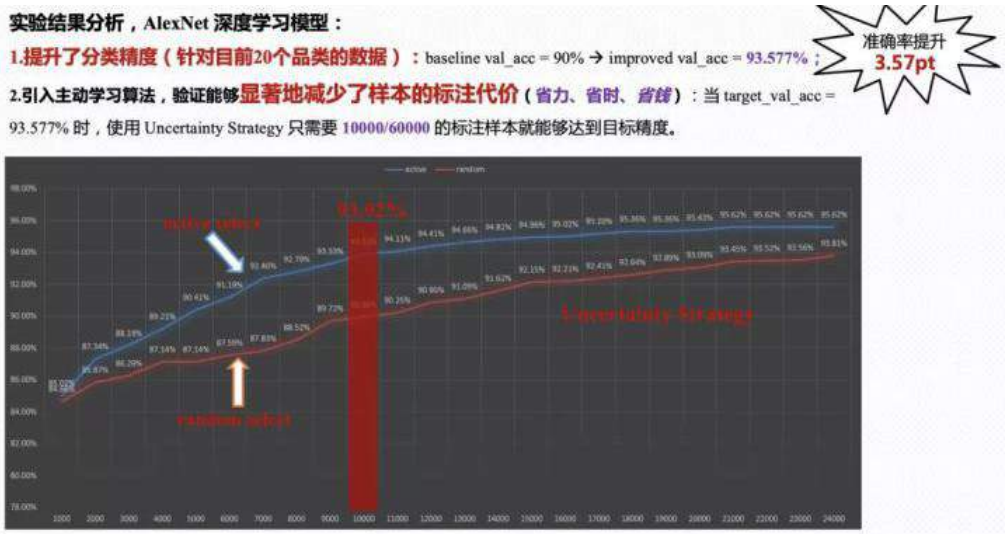
重点关注每个样本预测结果的最大概率值：p_pred_max。我们初步认为 p_pred_max>0.5 的情况表示当前模型对该样本有个确定的分类结果（此处分类结果的正确与否不重要）；反之，当前模型对该样本的判断结果模棱两可，标记为 hard sample；比如：模型进行第一次预测，得到 10 个概率值，取其最大的概率 p_pred_max；对 P(real lable) < p_threshold（此处的 10 分类任务取 p_threshold=0.5）的样本进行排序，取前 N 个样本加入集合 train_samples 中；

应用：淘系商品的二分类问题

背景：商品的单包装和多包装属性影响着客户对商品价格的认知。比如：有些多包装属性的标价较高，但实际单价可能已经很划算了，而客户误将多包装的价格认为是单价，导致购买意向降低。因此区分出商品的包装属性对提高客户购买意向和优化商品价格分布具有较大的实际意义。对于此问题，有多种不同的解决方案。

其中，基于图像的分类方法能够直接的区分出商品的单/多包装属性。然而，监督学习需要大量的标注样本，众多品类将产生大量的标注需求，如何能够显著地减少标注代价也同样具有重大的意义。

因此，我们尝试将主动学习方法应用图像分类中，解决单包装和多包装的二分类问题。如下图所示，我们分别对比了随机筛选策略和不确定策略。实验结果表明，引入不确定性策略主动筛选样本显著地减少了标注成本。



此外，我们尝试了更加复杂的模型（DesNet121），提高模型学习能力的同时，也带来了更多训练时长的弊端。但总体的分类精度提升了 3pt。同时，我们也分别在 AlexNet 和 DenseNet121 等模型上验证了模型预训练带来的效率。

Tab. 1 Transfer Learning 在模型精度上的作用

AlexNet（模型-1）				DenseNet-121（模型-2）			
ImageNet	train_acc	validation_acc	test_acc	ImageNet	train_acc	validation_acc	test_acc
without-pretrained	90.87%	88.68%	88.16%	without-pretrained	91.21%	92.74%	93.01%
pretrained	98.16%	93.20%	92.44%	pretrained	98.41%	96.74%	96.92%

说明：Transfer Learning（迁移学习）分别在 AlexNet 和 DenseNet-121 模型上提升 4pt 左右的准确率；
Without-pretrained 表示模型未使用迁移学习，参数随机初始化；pretrained 表示模型迁移 ImageNet 的参数作为初始化。

参考文献：

本文涉及的参考文献较多，由于篇幅问题，参考文献详见：

【Active Learning - 13】总结与展望 & 参考文献的整理与分享（The End...） blog.csdn.net

【Active Learning - 08】主动学习（Active Learning）资料汇总与分享 blog.csdn.net

Q: 在你做推荐系统的过程中都遇到过什么坑?

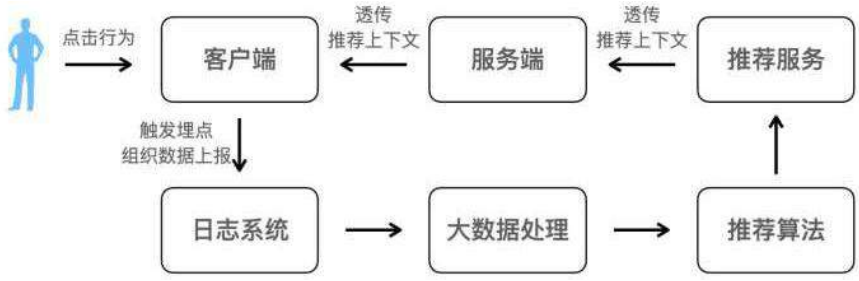
作者 | 九胜
出品 | 淘系技术

不管项目大小，也不论推荐什么东西，只要你做过和推荐系统相关的项目，希望能简单说一下你们遇到过的坑，如果能再简单说一下解决方案就最好了。

数据错误、离在线不一致、可以锦上添花不能雪中送炭，做推荐系统过程中最大的 3 个坑。

1. 数据错误

推荐算法是最容易发生数据错误的算法领域，很多机器学习应用的数据来自于标注，数据的错误可以很容易的发现，而推荐算法依赖的数据来自于用户的行为反馈，这个极易出错又极难发现。



用户在屏幕上的一个点击，行为发生的上下文和一个推荐物品的上下文，需要客户端发送给服务端，服务端把信息上报给日志系统，日志系统把数据导入到大数据平台，算法才能开始他的工作，而其中的物品推荐上下文，通过推荐接口透传给服务端，服务端带给前端，前端要解析好才能使用。

这样的一整条链路，有太多的出错可能，而出错时又很难意识到错误的存在，更别说错误的排查。

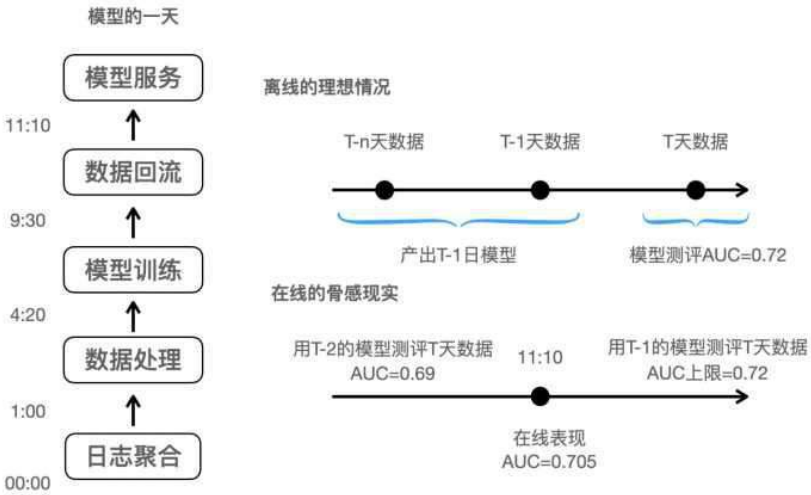
讲一个愚蠢的错误。在一个初上线的业务上，依靠几天的日志搭建了 Rank 模型，也有收益，看起来没有任何问题。当时唯一的怪现象，这个场景的 pv 是 uv 的 100 倍，也就是人均的浏览数是 100，按照经验这个数字高处之前同类业务的几倍，这个数字呗暴露出来一周，大家只是奇怪，没人真的怀疑。

我大概从觉得不对劲，到动手查，确认原因，到解决问题，花了一周的时间。服务端有一个环节把创作者的 id 和用户的 id 弄混了，所以才有人均 100 的浏览，实际的 uv 是创作者的数量。问题修复后，推荐效果涨了一倍！

2.离在线不一致

做排序模型的时候，以某一个离线指标为依据，疯狂地优化，然后幻想上线后指标能够大涨，3.75 向你招手。实际上这时候距离成功还有一个巨大的坑——离在线一致性。

离在线一致性可以来自于人的失误，离线的特征处理和在线的特征处理天然是两个流程，出现问题的可能性非常大，但开始可以靠细致的检查来避免，甚至可以将离在线的关键流程用一套组件抽象和处理，降低认为出错的可能。而另一种离在线不一致性来自于系统本身，在系统限定下无可避免，当然可以通过系统的升级来缓解，但成本的曲线也会变得非常陡峭，下面就重点聊聊这个。

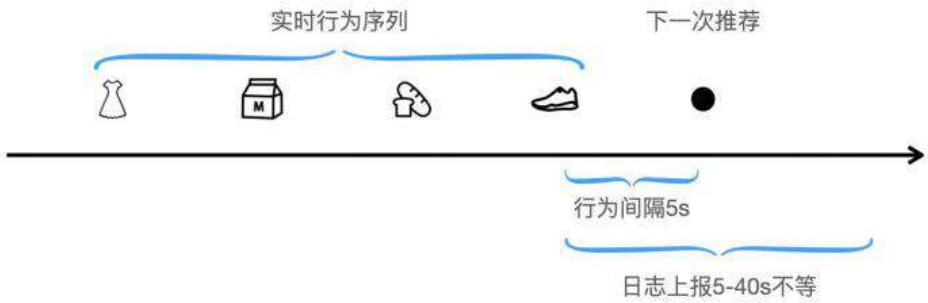


很多推荐系统的排序模型，在系统升级前或业务初期都是 T+1 更新的。

在离线，使用 T-n 到 T-1 的 n 天数据训练模型，用 T 天的数据进行测评，拿到了很好的离线指标，比如 AUC 为 0.72。但是在线服务的模型，并不是这样的理想情况，一个模型每天重新迭代训练，需要新一天（T-1 天）的日志，日志从数据队列传输到大数据平台，进行日志的处理，新一天各种特征的计算，组织训练样本，进行模型训练，之后还要把模型从大数据平台更新到在线服务器，整个流程走下来个把小时总是要的（图中例子为 11 点 10 分）。那么在新模型上线前，也就是 11：10 分之前，在线服务的是 T-2 的模型，相当于在离线用 T-2 的模型去测评 T 天的样本，效果会大打折扣。因而线上一整天的平均测评指标，是低于离线测评指标的。

有的业务和模型下，有可能 T-2 和 T-1 的差距不大，但推荐系统非常依赖 ID 型记忆型特征，T-2 可能会导致非常差的结果。这个时候，让你的模型能越早上线，效果就越好。有可能你增加了离线特征的处理流程，使用了复杂的特征和模型，让 AUC 提升了一个点，但是模型晚上线了 3 个小时，在线效果可能就掉到了坑里。

升级到小时级模型更新，甚至流式更新，能避免这个系统性的问题。当然成本也会变大，也会要爬更多的新的坑。



再聊聊特征上的不一致，如果是天级别计算的离线特征，它有着和上面讨论的模型不一致的类似的问题。而使用实时特征，也有隐秘的不一致问题。

实时特征在线使用的时候，经过客户端上报，流式计算处理日志数据进入在线数据源或特征服务器，需要几秒到几十秒不等的时间。也就是说，如果你刚刚点击了鞋子，紧接着下滑翻

页，系统是拿不到鞋子这个行为的。如果离线模型训练中有用到了带有鞋子的特征，由于近期行为的影响非常大，那么离在线的不一致会非常严重。

离线获取实时特征，一个是通过离线日志复现实时特征，依靠是日志中记录的时间戳，如果仅考虑样本的时间戳大于行为的时间戳，那么离在线不一致就发生了，相对明智的做法是流出一个时间 GAP，让离线和在线尽可能一致。另外要注意，样本的时间戳，不是用户曝光行为的时间戳，而是推荐系统的时间戳，否则也会用到在线无法拿到的穿越特征。

另一种方法，是将在线使用的特征直接依靠日志系统记录下来，在线的时间 GAP 假如是 5 到 40 秒飘忽不定的，整体上虽然离线在线一致了，但对于某个样本，有可能记录了最近的一个行为，有可能丢掉了最近的行为。这样做比离线复现实时特征的方法好多了，唯一的问题可能是要积攒一定天数的实时特征日志，才能完成初次模型上线，每次特征的修改也要等一段时间进行日志积累。

3.可以锦上添花，不能雪中送炭

很多的产品经理把推荐算法当成产品最后的希望，把用户增长的希望压到推荐算法身上。但是，推荐系统只能锦上添花，不能雪中送炭。

没有或有一个弱的推荐系统，能挣 10 个亿，通过优化多挣 3 个亿，推荐系统就非常有意义。一个产品，没有多少用户，靠输血勉强维持体面，想几倍增长，单纯靠推荐算法真的是无能为力，一个不愚蠢的推荐系统和一个优质的推荐系统之间的差距已经没有几倍增长的可能了。

那些增长案例，如谷歌邮箱推出 5G 空间突破 outlook 实现增长，airbnb 发现用户在意的核心因素是房间的照片，扛着专业设备一个一个去拍照实现增长。这都是满足了用户未被满足的需求而聚集起大量的用户。一个不愚蠢的推荐系统没有什么门槛，升级到优质的推荐系统，并没有满足任何新的需求。靠推荐系统实现大规模的用户增长和产品成功，是错误的努力方向。

客户端篇

面向未来，我们发现客户端技术当前正处在一个新的轮回，随着新一轮技术升级来临，这必将催化移动技术更深层次的变革，技术和商业将更进一步融合，未来的移动技术将发生翻天覆地的变化。淘宝客户端，这一艘超级航母 App 也身处其中，如何在用户体验和开发者体验上做进一步研究创新是对客户端提出的一个新的命题，体验，效率，质量永恒不变的话题，客户端的几篇文章从一个侧面反映了目前我们在客户端技术上的一些探索和思考。

——阿里巴巴资深技术专家 白衣

技术经典总结

Flutter 和 Web 生态如何对接?

作者 | 门柳

出品 | 淘系技术

导读：Flutter 设计之初是不考虑 Web 生态的，原因很简单：两种技术设计理念不同，强行融合很可能让彼此都丧失了优势。但是业界又有很多团队在做这种尝试，说明需求是存在的。今天，阿里无线开发专家门柳就来手把手教如何实现 Flutter 和 Web 生态的对接?

先说结论：

不要对接！不要对接！不要对接！

开个玩笑，以上仅代表个人观点，大家也知道这种"三体式警告"根本没有用的，我自己也研究如何对接，说不定做完后就觉得"真香"了。

为什么要对接?

首先讨论一下为什么要把 Flutter 对接到 Web 生态。

Flutter 现在是一个炙手可热的跨平台技术，能够一套代码运行在 Android、iOS、PC、IoT 以及浏览器上，被认为是下一代跨平台技术。相比于 Weex 和 React Native 可以很好地解决多平台一致性问题，原生渲染性能相近，上层没有 JS 那么厚的封装层次，整体性能会略好一些。

但是大部分兴冲冲去学 Flutter 的人疑惑的第一个问题就是：为什么 Flutter 要用 Dart? 一个全新的语言意味着新的学习成本，难道 JS 不香吗? JS 不香不是还有 TypeScript 吗! 事实上 Flutter 抛弃的岂止是 JS 这门语言，也抛弃了 HTML 和 CSS，设计了一

套解耦得更好的 Widget 体系，Flutter 抛弃的是整个 Web，致力于打造一个新的生态，但是这个生态无法复用 Web 生态的代码和解决方案。尤其是之前所有跨平台方案 Hybrid、React Native、Weex 都是对接 Web 生态的，这让 Flutter 显得有些格格不入，也让

大部分前端开发者望而却步。

下面是我整理出来的，前端开发者使用 Flutter 的各方面成本：

	改变	成本分析	成本评估
框架成本	React/Vue/小程序 --> Flutter	学习新框架 API	很低
语言成本	JS --> Dart	学习新语言	比较低
UI 成本	HTML/CSS --> Widget	转变写 UI 的思路	一般
		组件库要迁移成 Widget	
工程成本	前端工具链 --> Flutter/Native 工具链	研发/调试/测试/发布/搭建 都需要做改造	比较高
生态成本	npm --> pub	前端积累的库不可复用	很高
	Web 解决方案 --> Flutter 解决方案	前端常用的技术方案和性能优化手段在 Flutter 上未必有效	

因为 Flutter 的开发模式和前端框架比较像（可以说就是抄的 React），所以框架的学习成本并不高，稍微高一些的是 Dart 语言的学习成本，另外还要学习如何用 Widget 组装 UI，虽然很多布局 Widget 设计得和 CSS 很像，灵活度还是差了很多。要想在真实项目中用起来，还要改造整个工具链，以"Native First"的视角做开发，开发 Flutter 和开发原生应用的链路是比较像的，和开发前端页面有较大差异。最高的还是生态成本，前端生态的积累无论是代码还是技术方案都很难复用，这是最痛的一点，生态也是 Flutter 最弱的一环。

无论是为了先进的技术理念还是出于商业私心，先不管 Flutter 为什么抛弃 Web 生态，现实问题是最大的 UI 开发者群体是前端，最丰富的生态是 Web 生态，我觉得 Web 技术也是开发 UI 最高效的方式。如果能在上层使用 Web 技术栈开发，在底层使用 Flutter 实现跨平台渲染，不是可以很好的兼顾开发效率、性能和跨平台一致性吗？还能复用 Web 技术栈大量的技术积累。

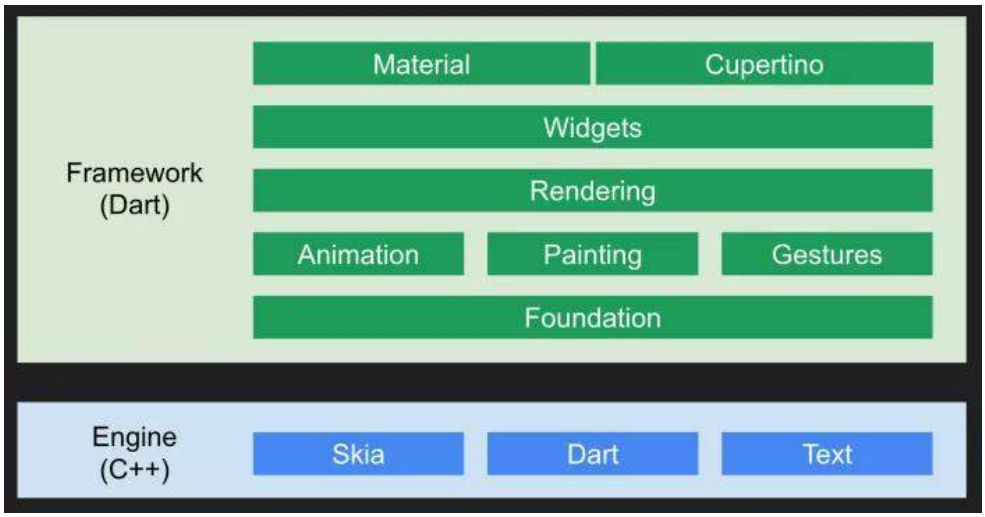
可能这些理由也不够充分，暂且先照着这个假设继续分析，最后再重新讨论到底该不该对接。

关于 Flutter 和 Web 生态的对接涉及两个方面：

- 从 Web 到 Flutter。就是使用 Web 技术栈来开发，然后对接到 Flutter 上实现跨平台渲染。对 Web 来说是解决性能和跨平台一致性问题，对 Flutter 来说是解决生态复用问题。
- 从 Flutter 到 Web。就是官方已经实现的 Web support for Flutter，把已经用 Dart 开发好的 App 编译成 HTML/JS/CSS 然后运行在浏览器上，可以用于降级和外投场景。

如何实现"从 Web 到 Flutter"？

首先分析一下 Flutter 的架构图，看看可以从哪里下手。



Flutter 可以分为 Framework 和 Engine 两部分，Engine 部分比较底层也比较稳定了，最好不要动，需要改的是用 Dart 实现的 Framework。要想对接 Web 生态的话，JS 引擎肯定是要引入的，至于是否保留 Dart VM 有待讨论。图中最上面 Material 和 Cupertino 两个 UI 库前端是不需要的，前端有自己的。关键是 Widget 这部分，是替换成 HTML/CSS 的方式写 UI，还是继续保留 Widget 但是把语言换成 JS，不同方案给出的解法也不一样。

有不少方案可以实现对接，业界有挺多尝试的，我总结了下面三种方式：

- TS 魔改：用 JS 引擎替换掉 Dart VM，用 JS/TS 重新实现 Flutter Framework（或者直接 dart2js 编译过来）。
- JS 对接：引入 JS 引擎同时保留 Dart VM，用前端框架对接 Flutter Framework。
- C++ 魔改：用 JS 引擎替换掉 Dart VM，用 C++ 重新实现 Flutter Framework。

TS 魔改

TS 魔改就是完全抛弃掉 Dart VM，用 TypeScript 重新实现一遍用 Dart 写的 FlutterFramework。

为啥是 TS 而不是 JS？这不是因为 TS 是个大热门嘛，而且向下兼容 JS，现在几乎所有时髦的框架都要用 TS 重写了。

这种方案的出发点是"如果能把 Flutter 的 Dart 换成 JS 就好了"，最容易想到的路就是把 Dart 翻译成 TS，或者直接用 dart2js 把代码编译成 js，但是编译出来的代码包含很多 dart:ui 之类的库的封装，生成的包也挺大的，也比较难定制需要导出的接口，不如干脆用 TS 重写一遍，工具链更熟悉一些，还可以加一些定制。

理论上讲翻译之后 Flutter 绝大部分功能都依然支持，可以复用各种 npm 包，还可以动态化，但是丧失了 AOT 能力，JS 语言的执行性能应该是不如 Dart 的。而且所有节点的布局运算都发生在 JS，底层只需要提供基础的图形能力就好了，就好像是基于 CanvasAPI 写了一套 UI 框架，性能未必有现存前端框架的性能高。

此外最大的问题是如何与官方 Flutter 保持一致，假如现在是从 v1.13 版本翻译过来的，以后官方升级到了 v1.15 要不要同步更新？这个过程没啥技术含量，而且需要持续投入，做起来比较恶心。

另外还需要考虑上层是用 Widget 的方式写 UI，还是用前端熟悉的 HTML+CSS。如果依然用 Widget 的话，那大部分前端组件还是用不了的，UI 还是得重写一遍。反正要重写的话，成本也没降下来，那就用 Dart 重写呗…… 直接用官方原版 Flutter 也避免每次更新都要翻译一遍 Dart 代码。所以既然选择了对接前端生态，那就要对接 CSS，不然就没有足够的价值。然而 CSS 和 Widget 的对接也是很繁琐的过程，而且存在完备性问题。

JS 对接

翻译代码的方式不够优雅，那就保留 Dart，把 JS/CSS 对接到 Widget 上面不就好了？当然可以，这种方式是仅把 Flutter 当做了底层的渲染引擎，上层保持前端框架的写法，仅把渲染部分对接到 Flutter。现存的很多前端框架都把底层渲染能力做了抽象，可以对接到不同渲染引擎上，如 Vue/Rax 同时支持浏览器和 Weex，用同样的方式，可以再支持一个 Flutter。

这种方式对前端框架的兼容性比较好，但是链路太长了，业务代码调用前端框架接口做渲染，一顿操作之后发出了渲染指令，这个渲染指令要基于通信的方式传给 Flutter Framework，

这中间涉及一次 JS 到 C++ 再到 Dart 的跨语言转换，然后再接收到渲染指令之后还要转成相应的 Widget 树，从 CSS 到 Widget 的转换依然很繁琐。而且 Widget 本身是可以带有状态的，本身就是响应式更新的，在更新时会重新生成 widget 并 diff，如果在前端更新 UI 的话，前端框架在 js 里 diff 一次 vdom，传到 Flutter 之后又 diff 一次 widget。

如果要绕过 Widget 直接对接图中的 Rendering 这一层，可以绕过 widget diff 但是得改 Flutter Framework 的渲染链路，既然要改 Flutter Framework 那为什么不直接用 TS 魔改呢，还绕过了 JS 到 Dart 的通信，又回到了第一种方案。

总结来说，这个方案的优点是：实现简单、能最大化保留前端开发体验，缺点是：渲染链路长、通信成本高、响应式逻辑冲突、CSS 转 Widget 不完备等。

C++ 魔改

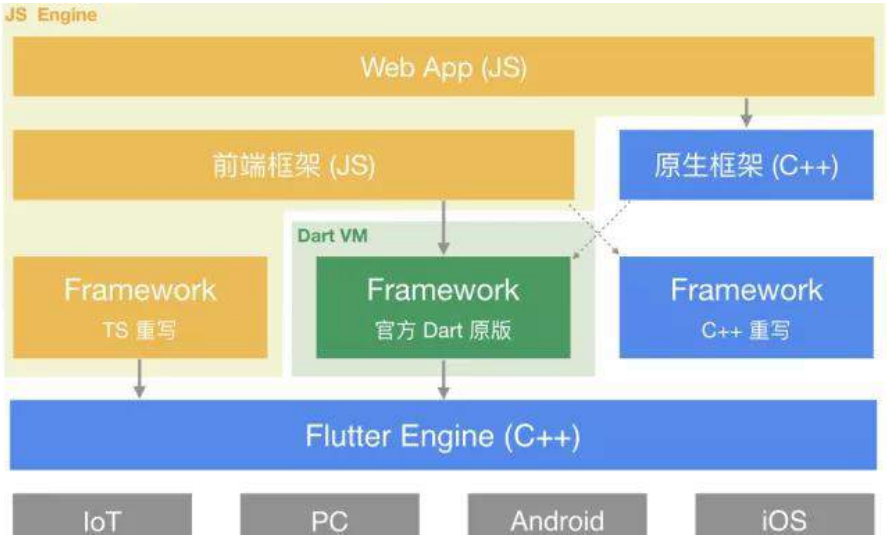
想要干掉 Dart VM，就需要用其他语言重新实现用 Dart 开发的 Framework，用 JS/TS 可以，用 C++ 当然可以，最硬核的方式就是用 C++ 重新实现 Flutter 的 Framework，然后接入 JS 引擎，通过 binding 把 C++ 接口透出到 JS 环境，上层应用还是用 JS 做开发。

把 Framework 层下沉到 C++ 之后，不仅会有更好的性能，也能支持更多语言。原本 Flutter Framework 是在 Dart VM 之上的，必须依赖 Dart VM 才能运行，所以对 Dart 有强依赖；用 C++ 重新实现之后，JS 引擎是在 C++ 版 Framework 之上的，框架本身并不依赖 JS 引擎，还可以对接其他各种语言，如对接了 JVM 之后可以支持 Java 和 Kotlin，对接回 Dart VM 可以继续支持 Dart。

这个方案可以增强性能，也能保持和 Flutter 的一致性，但是改造成本和维护成本都相当高。C++ 的开发效率肯定不如 Dart，当 Flutter 快速迭代之后如何跟进是很大的问题，如果跟进不及时或者实现不一致那很可能就分化了。从 CSS 到 Widget 的转换也是不得不面对的问题。

几种方案对比

把上面几种方案画在同一张图里是这个样子的：



图中实线部分表示了跨语言的通信，太过频繁会影响性能，虚线部分表示了其他对接可能性。

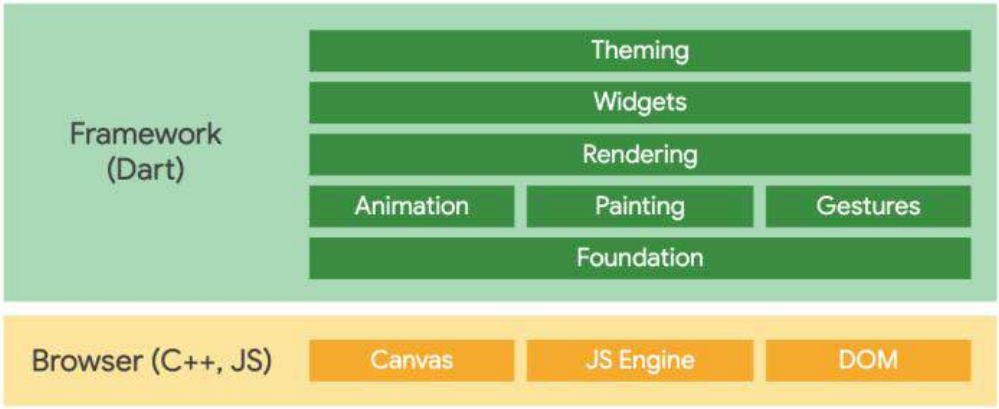
从下到上，Flutter Engine 是不需要动的，这一层是跨平台的关键。Framework 则有三种语言版本，JS/TS、Dart、C++，性能是 C++ 版本最好，成本是 Dart 版本最低。然后还需要向上处理 HTML/CSS 和 Widget 的问题，可以直接对接一个前端框架，也可以直接在 C++ 层实现（不然需要透出的 binding 接口就太多了，用通信的方式也太过频繁了）。

如何实现“从 Flutter 到 Web”?

这个功能官方已经实现了，可以把使用 Dart 开发的 App 编译成 Web App 运行在浏览器上，官方文档以介绍用法和 API 为主，我这里简单分析一下内部具体的实现方案。

实现原理

结合 Flutter 的架构图来看，要实现 Web 到 Flutter 需要改造的是上层 Framework，要实现 Flutter 到 Web 需要改造的则是底层 Engine。



Framework 对 Engine 的核心依赖是 dart:ui，这是库是在 Engine 里实现的，抽象出了绘制 UI 图层的接口，底层对接 skia 的实现，向上透出 Dart 语言的接口。这样来看，对接方式就比较简单了：

- 使用 dart2js 把 Framework 编译成 JS 代码。
- 基于浏览器的 API 重新实现 dart:ui，即 dart:web_ui。

把 Dart 编译成 JS 没什么问题，性能可能会有一点影响，功能都是可以完全保留的，关键是 dart:web_ui 的实现。在原生 Engine 中，dart:ui 依赖 skia 透出的 SkCanvas 实现绘制，这是一套很底层的图形接口，只定义了画线、画多边形、贴图之类的底层能力，用浏览器接口实现这一套接口还是很有挑战的。上图可以看到 Web 版 Engine 是基于 DOM 和 Canvas 实现的，底层定义了 DomCanvas 和 BitmapCanvas 两种图形接口，会把传来的 layer tree 渲染成浏览器的 Element tree，但是节点上仅包含了 position，transform，opacity 之类的样式，只用到 CSS 很小的一个子集，一些更复杂的绘制直接用 2D <canvas> 实现。

存在的问题

我编译了一个还算复杂的 demo 试了一下，性能很不理想，滑动不流畅，有时候图片还会闪动。生成出来的 js 代码有 1.1MB (minify 之后，未 gzip)，节点层次也比较深，我评估这个页面用前端写不会超过 300KB，节点数可以少一半以上。

另外再看一下 Flutter 仓库的 issue，过滤出 platfrom-web 相关的，可以看到大量：文字编辑失效、找不到光标、ListView 在 ios 上不可滚动、checkbox/button 行为不正常、安卓滚动卡顿图片闪烁、字体失效、某些机型视频无法播放、文字选中后无法复制、无法调试……感觉 flutter for web 已经陷入泥潭，让人回想起前端当年处理各种浏览器兼容性的噩梦。

这些性能和兼容性问题，核心原因是浏览器未暴露足够的底层能力，以及浏览器处理手势、用户输入和方式和 Flutter 差异巨大。

实现 Flutter Engine 需要的是底层的图形接口和系统能力，虽然 <canvas> 提供了相似的图形接口，如果全部用 canvas 实现的话很难处理可访问性、文本选择、手势、表单等问题，也会存在很多兼容性问题。所以真实方案里用的是 Canvas + DOM 混合的方式，封装层次太高了，渲染链路太长。就好像 Flutter Framework 里进行了一顿猛如虎的操作之后，节点生成好了、布局算好了、绘制属性也处理好了，就差一个画布画出来了，然后交到浏览器手里，又生成一遍 Element，再算一遍布局，在处理一遍绘制，最终才交给了底层的图形库画出来。

再比如长页面的滚动，浏览器里只要一条 CSS (overflow:scroll) 就可以让元素可滚动，手势的监听以及页面的滚动以及滚动动画都是浏览器原生实现的，不需要与 JS 交互，甚至不需要重新 layout 和 paint，只需要 compositing。如上图所示，在 Flutter 中 Animation 和 Gesture 是用 Dart 实现的，编译过来就是 JS 实现的，浏览器本身并不知道这个元素是否可滚，只是不断派发 touchmove 事件，JS 根据事件属性计算节点偏移，然后运算动画，然后把 transform 或者新的 position 作用到节点上，然后浏览器再来一遍完整的渲染流程……

优化方案

性能和兼容性的问题还是要解决的，短期内先把 issue 解掉，长线的优化方案，官方有两种尝试：

1. 使用 CSS Painting API 做绘制。

a. 这是还处于提案状态的新标准，可以用 JS 实现一些绘制功能，自定义 CSS 属性。
b. 目前还未实现，需要等浏览器先把 CSS Houdini 支持好。
2. 使用 WebAssembly 版本的 Skia 做绘制 https://skia.org/user/modules/canvaskit。
a. 这样可以发挥 wasm 的性能优势，并且保持 skia 功能的一致。但是目前 wasm 在浏览器环境里未必有性能优势，这里不展开讨论了。
b. 已经部分实现，参考这里的配置启用功能： https://github.com/flutter/flutter/issues/41062#issuecomment-533952994

这两个方案都是想更多的利用到浏览器的底层能力，只有浏览器暴露了更多底层能力，才能更好的实现 Flutter 的 Web Engine。不过这个要等挺久的时间，我们也参与不了，现阶段想要使用 flutter for web，还是得保持现有架构，一起参与进去把 issue 解决掉，优先保障功能，其次优化性能。

一种适应性更好的架构

如果理想化一点，能不能从架构角度让 Flutter 和 Web 生态融合的更好一些呢?

回顾文章最开始的官方架构图，上面是 Framework(Dart)，下面是 Engine(C++)，切分在 Foundation 这一层，双方之间的交互是几何图形信息。如果还保持这个架构，把切分层次划分的更靠上一些，如下图所示，划分在 Widgets 和 Rendering 这一层，理论上讲对 Flutter 的开发者来说是无感知的，因为上层的开发语言和 Widget 接口都是不变的。

切分在这一层，Framework 和 Engine 之间的交互就不再是几何图形而是节点信息，Widget 的组合、setState 响应式更新、Widget diff 都还在 Dart 中，展开后的 RenderObject 的布局、绘制、裁剪、动画全都在 C++ 中，不仅有更好的性能，还可以与 Engine 有更好的结合。

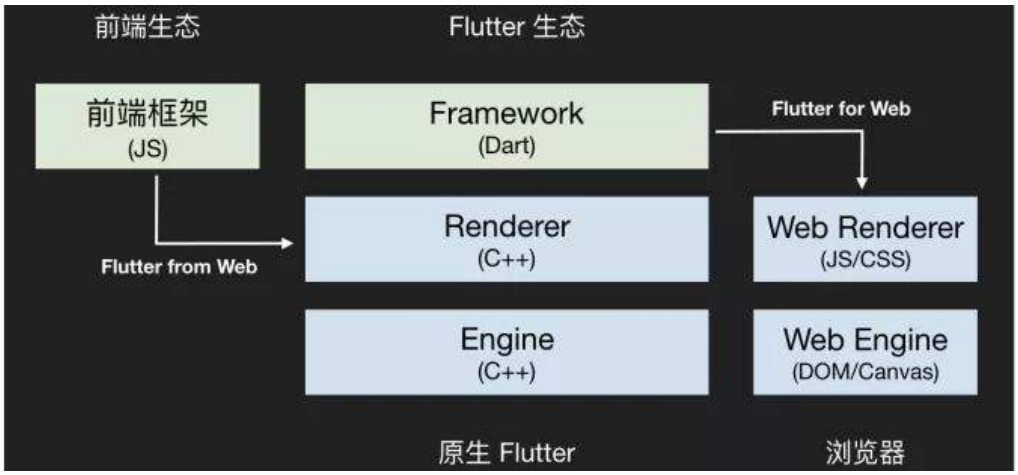
或者说，还原本保留 Engine 的设计，把下沉的这部分逻辑上划分成 Renderer，就有了如下三层的结构：

这样划分出来的每一层都有明确的定位：

- Framework：开发框架。为开发者提供可编程 API，实现响应式的开发模式，提供细粒度 Widget 供开发者自由封装和组合。
- Renderer：渲染引擎。专门实现布局、绘制、动画、手势的的处理，这部分功能相对独立，是可以与开发框架解耦的，也不必与特定语言绑定。
- Engine：图形引擎。实现跨平台一致的图形接口，合成输入的层并绘制到屏幕上，处理好平台力的接入和适配。

这样切分除了有性能优势以外，也使得渲染引擎摆脱了对 Dart 的依赖，能够支持多种语言，也能支持多种开发模式。对接到 Dart VM 就可以用 Dart 写代码，对接到 JS 引擎就可以用 JS 写代码，对接到 JVM 还可以写 Java，但是无论怎么写，底层的渲染能力是一样的，一套统一的布局算法，动画和手势的处理行为也是一致的。

在这样的架构下，对接 Web 生态就更容易了。Dart 和 Widget 是前端不想要的，希望能换成 JS 和 CSS，但是又想要底层的跨平台一致渲染引擎，那从 Renderer 层开始对接就好了，绕过了所有不想要的，也保留了所有想要的。



要实现 Flutter for Web 也更简单了一些。在 Engine 层做对接，一直苦于浏览器透出的底层能力不够，如果是在 Renderer 之上做对接就更容易一些，基于 JS/CSS/DOM/Canvas 的能力封装出一套 Rendering 接口，供 Widget 调用就好了，这样可以使渲染链路更短一些，但是依然要处理 Widget 和 DOM/CSS 之间的兼容性问题。

再讨论一遍：为什么要对接?

技术上已经分析完了，要想搞定 Flutter 生态和 Web 生态的对接，需要投入很大的成本，所以真正决定做之前，要先讨论清楚为什么要做对接？到底要不要做对接?

首先 Google 官方对 Flutter 的定位就是个问题。Flutter 设计之初就是不考虑 Web 生态的，甚至在刻意回避，倡导的是更贴近原生的开发方式。我之所以在开头说不要对接，原因也很简单：两种技术设计理念不同，不是朝着一个方向发展的，生态不通，技术方案不通，

强行融合很可能让彼此都丧失了优势。但是业界又有很多团队在做这种尝试，说明需求是存在的，如果 Google 抵制这个方向，那就不好做了。不过现在官方已经支持了 Flutter forWeb，已经向 Web 生态迈了一步，未来是否进一步与 Web 融合，也是有可能的。

另外就是跨平台技术本身的问题，浏览器发展了二三十年，已经是个很强大的跨平台产品了，几乎是 Web 的代名词了，这一点无人能敌。但是也臃肿不堪，有大量历史包袱，性能和体验不够好，和 Native 的结合度差，尤其在移动和 IoT 平台。虽然硬件性能在不断提升，但这是所有软件共享的，浏览器的性能和体验总会比 Native 差一些，差的这一些很可能就是新业务和新场景的发挥空间。观察一下近几年新诞生的业务场景，很多都是利用到了 Native 新提供的能力才火爆起来的，如 AI/AR/视频/直播 等，有因为新的 Web API 而孵化生出来的商业模式吗？

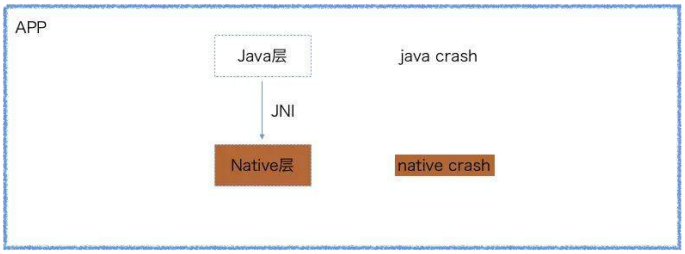
手淘如何突破 native crash 提升稳定性?

作者 | 潘文超(开风)
出品 | 淘系技术

导读：本文是作者在 「Top100 全球软件案例研究峰会」上分享的——手淘 Native 治理，结合分享后的反馈焦点，从 native 问题线下发现和快速定位、新 so 集成标准、线上历史 native 问题治理等几个方面为大家介绍，特产此文。

扫描下方二维码，关注「淘系技术」微信公众号，回复"native"即可获取本次分享完整版 PPT 资料!

背景



无线客户端 crash 一般分为 java crash 和 native crash ，无线应用往往为了追求更好的性能，把一些复杂的计算放到 native 层去实现，java 层通过 jni 调用 native 层实现，满足功能场景；在 native 层产生的崩溃就产生了 native crash ；无线客户端衡量稳定性的最重要的指标就是 crash 率，稳定性一直是各系统治理的重中之重，也是一直绕不开的话

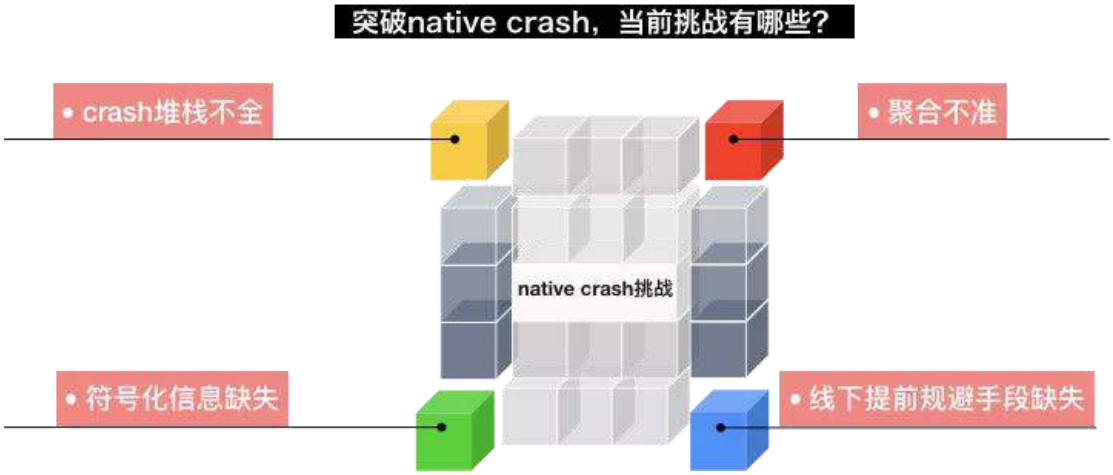
题，而 native 是目前业界认为治理最有难度，也是要重点突破的方向。接下来分析一下为什么 native 治理值得去做以及如何做好。

目前手淘 Android 的 crash 现状——找方向

从图中我们可以看出，java crash 正常可以维持在较好的水平，手淘 native crash 一般比 java crash 要高，大促期间，由于手淘内存瓶颈， native crash 率会涨到日常水准的 2-3 倍。

从数据可见，native crash 是 java crash 的 6 倍之多，如果要想进一步突破，native 是有很大空间的，但 native 问题一般都很难定位，堆栈不全，或者堆栈都集中在系统 so 上，无法直接定位问题，所以想要突破难度很大。手淘稳定性再上升一个台阶，nativecrash 是瓶颈，需要突破。

native crash 治理挑战——难



难点一：crash 堆栈目前绝大部分只有系统 so，缺乏问题关联的业务模块 so，定位问题难度大，可以看一下如下的一个 native crash 堆栈，堆栈中的 so 都是系统的，无法直接定位业务问题。

难点二：线上 so 都是去符号化的，即使堆栈中有业务 so，也需要记录该 APP 版本对应的

符号化 so 做反解才能拿到能看得懂的堆栈。这点已经通过第一阶段的 native 工程标准化解决，打出来的 SDK 包里面必须要有对应的符号化 so 才可以集成。

难点三：目前线下有效提前发现 native 问题的手段缺乏，想要提前发现，需要平台工具和有效手段。

native crash 治理核心问题是什么？——找抓手

要治理、要解决问题，首先得理清目前导致 native crash 的问题，在做之前，做了一下 crash 数据分析，于是手动捞取了当时近 5 个版本的 top native crash 数据，占比最多的就是 sig 6 和 sig 11。

那能说明什么问题呢？signal 6 这种崩溃信号要看具体场景，但根据具体数据分析，手淘里面一般都是堆栈溢出、OOM 等导致的。signal 11 这种崩溃基本就锁定为内存问题了。

根据实际数据，大部分 crash 原因是因为内存，可以初步下的结论是，目前手淘 native crash 治理的关键是内存，解决手淘 native 内存相关的问题即可解决掉不部分问题。

Native 问题治理平台工具调研

对集团内及业界的一些产品做了一些调研，详细如下：

维度\产品	malloc debug	AddressSanitizer	memhook（UC）
是否需要root	需要	需要	需要
功能支撑广度	野指针、内存泄漏、内存崩溃等	野指针、堆栈缓冲区溢出、 全局缓冲区溢出、内存泄漏等	检测内存泄漏
堆栈能力	有	有	有
性能	差	好	不详
使用成本	高	高	一般

开发了Native Finder，希望能彻底治理好手淘Native问题

分别从使用成本、功能支撑、是否有堆栈能力，性能如何等维度进行了比较，其实我们的诉求是希望不需要 root 就能 run 起来，因为我们要持续集成、线上能灰度验证，线下可以大规模任务执行、并且可以做手淘 native 问题沉淀，能把问题沉淀做成检查项，可以涵盖解决主要的内存问题，不止是内存泄漏。因此开发了 Native Finder，希望能彻底治理好手淘 Native 问题。

Native 治理——Native Finder 整体技术方案

Native Finder 是利用 elf hook 的原理，对 malloc、free、realloc、alloc、mmap、munmap 等操作进行 hook，在自己的 hook 函数中，做一些智能分析和统计，并最后会调用系统内存操作函数，不破坏 so 原本发起的 malloc 等操作。

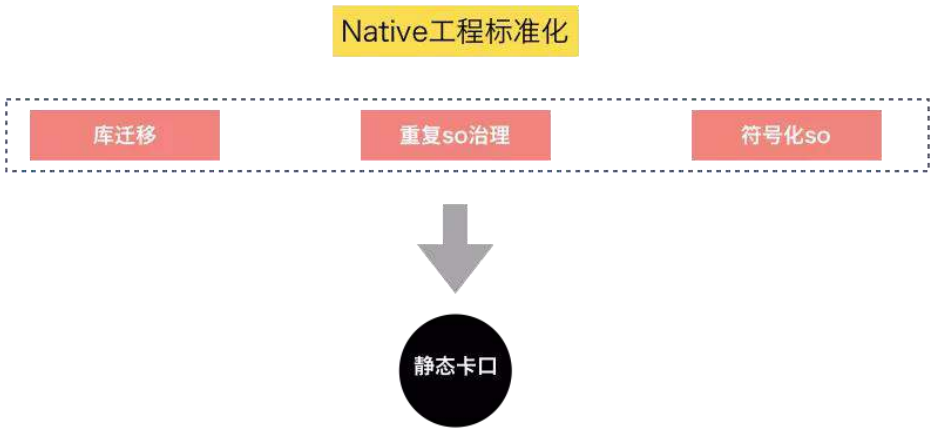
这个方案的缺点是会有一定的性能损耗，毕竟在 malloc 等操作中增加了一些分析和统计操作，但性能影响还好，在开了堆栈能力之后，APP 性能会受影响，这个也是后面要优化的地方。整体技术方案如下：

native crash 治理过程

首先在前期，花了比较多时间去研究历史数据及问题，认真分析 crash 的关键问题和痛点是什么，才找准了方向，分析出来内存问题是最痛的点。治理过程总结如上图，分为 5 个阶段。接一下讲详细介绍结合 Native Finder 工具平台的治理过程和心得。

native 工程标准化

治理总结如下：

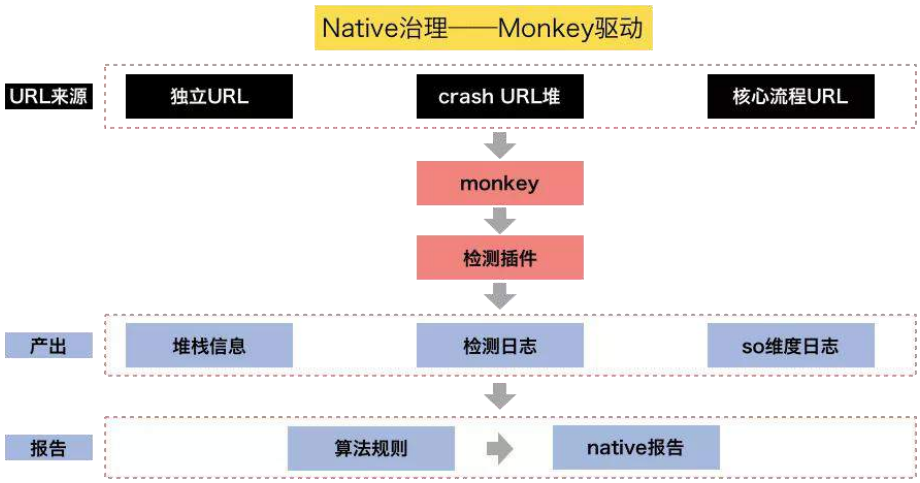


总结起来这个阶段我们做了 3 件事情：

- 第一，库迁移，为什么要做库迁移呢？我们把老的 gnu C++基础库迁移成 libc++_shared 库，所有 so 都做统一，归一化，方便归一 native 层问题。

- 第二，我们还做了重复 so 治理，因为不同 so 可能依赖了相同的 so 基础库，举个例子，例如 A so 依赖了 libopenSsl 基础 so，B so 也依赖了 libopenSsl 库，但他们依赖的版本不同，这样带来的坏处是，首先会增加包大小，其次会有相同 so 的不同版本存在应用中，这给定位问题带来了麻烦，所以需要对基础 so 去重。
- 第三，每个版本包包括灰度版本都需要存储下对应的符号化 so，便于在 crash 发生后，我们对堆栈做符号化处理，这样堆栈我们就能看懂了，加快和精准定位问题。这个阶段是后面做的所有事情的基础，非常重要。接下来我们看看下一个阶段治理。

Native Finder 开发完成，线下 monkey 跑 native 问题

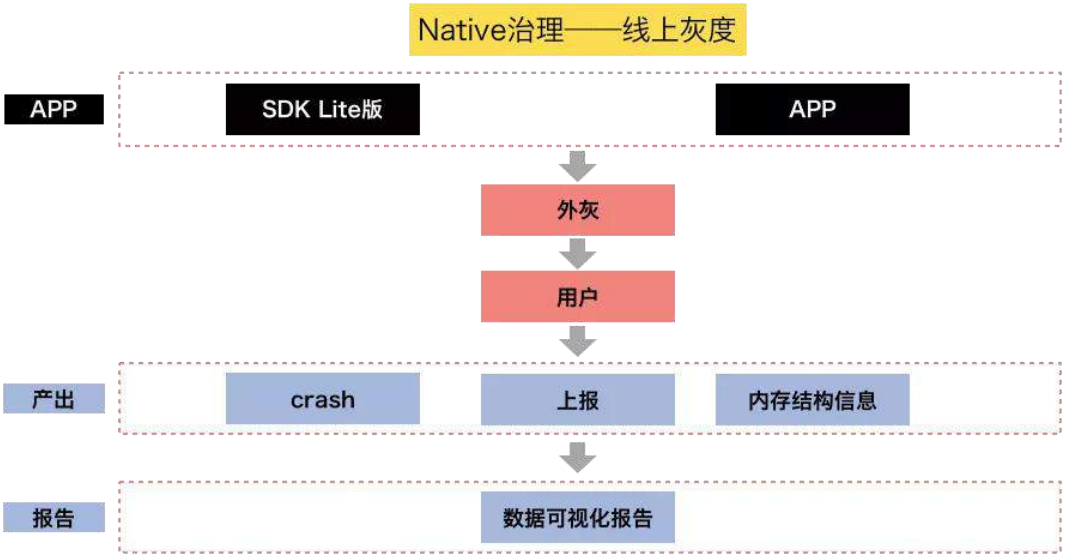


此阶段发现了几个堆破坏的问题，但是经过好几天反复线下执行 monkey，并未有任何进展。

后续调整思路，开始逐个分析线上存在的 native crash，并根据这些 crash 特征和根因开始沉淀经典问题，并把这些问题做成检查项，跟同学交流和对焦后。

通过进一步的数据，发现内存 OOM 是目前优先级较高，且比较严重的问题，所以开始做这方面的技术建设，跟 crashSDK 打通，Native Finder 中统计和分析 so 维度占用内存未释放的数据，在 crash 的时候做内存信息 dump，并输出辅助信息，例如 malloc、mmap 次数、大内存（大于 2M，属于大内存申请，可配置，可动态调整）的申请等信息。接下来看一下线下 monkey 驱动阶段

线上灰度，结合用户真实操作场景 crash



线下 monkey 并不能都能复现问题，借助工具平台拿到关键信息去做问题解决和定位，我们希望场景更加丰富和多样，所以我们把 Native Finder 放到线上，做了线上灰度。

随即把 Native Finder 放到线上做外灰，在用户真实操作场景下的 crash，拿到 crash 时的内存 dump，但是经过一段时间的线上 crash 内存信息采集，然后分析之后，没发现明显问题，从现在回头看这个阶段，其实当时的数据是能够体现出问题的，只是当时的想法不对。

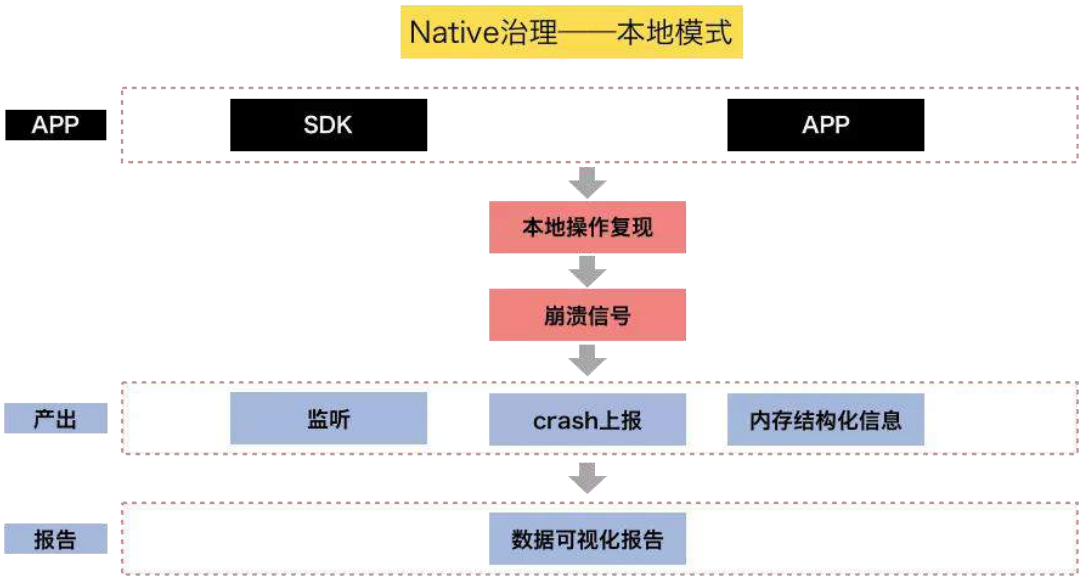
虚拟内存不足，是目前 OOM 主要原因

我们对线上 crash 做了分析，同时也在线下做了可疑场景的尝试复现，复现过程中，我们也做了大量数据的对比分析；

分析发现：正常 crash 跟 native oom crash，在内存详细数据上做对比，发现 OOM crash 在 native vmsize 上有较大差异，又看了很多手淘 OOM crash，发现都是这个原因，vmsize 暴涨。

大家都知道，32 位系统下，系统 vmszie 只有 4G，还要抛去一些系统内核占用、以及共享内存占用，vmsize 比较有限，手淘又是一个体量很大的航空母舰，各个业务都想有最佳的业务体验，都想用空间换时间，每个业务泄漏那么一点，那手淘就被撑爆了，是累加泄漏的结果。

所以手淘 Android OOM 要一个一个解决，逐个挖出来，才能根治。我们整个过程沉淀了如下检查项：一共沉淀了 8 项内存检查，内存检查项已经能覆盖 80% 以上的内存问题了；fd 文件句柄检查项一共沉淀了 6 项，fd 的检查项已经能覆盖几乎 95%以上的 fd 问题了。我们同时也研发了本地调试模式，方便开发和测试同学，能快速在本地复现和定位问题，具体的技术方案如下：



开始陆续发现各种 native 问题

朝着这个方向，对 Native Finder 做了逐步优化，开始陆续发现各种问题，治理初步阶段，我们通过 Native Finder 工具平台一共治理发现 20+ 问题，其中包括了多种问题类型，例如内存堆破坏、内存泄漏、OOM、内存操作越界、多次 free、内存操作错误等。

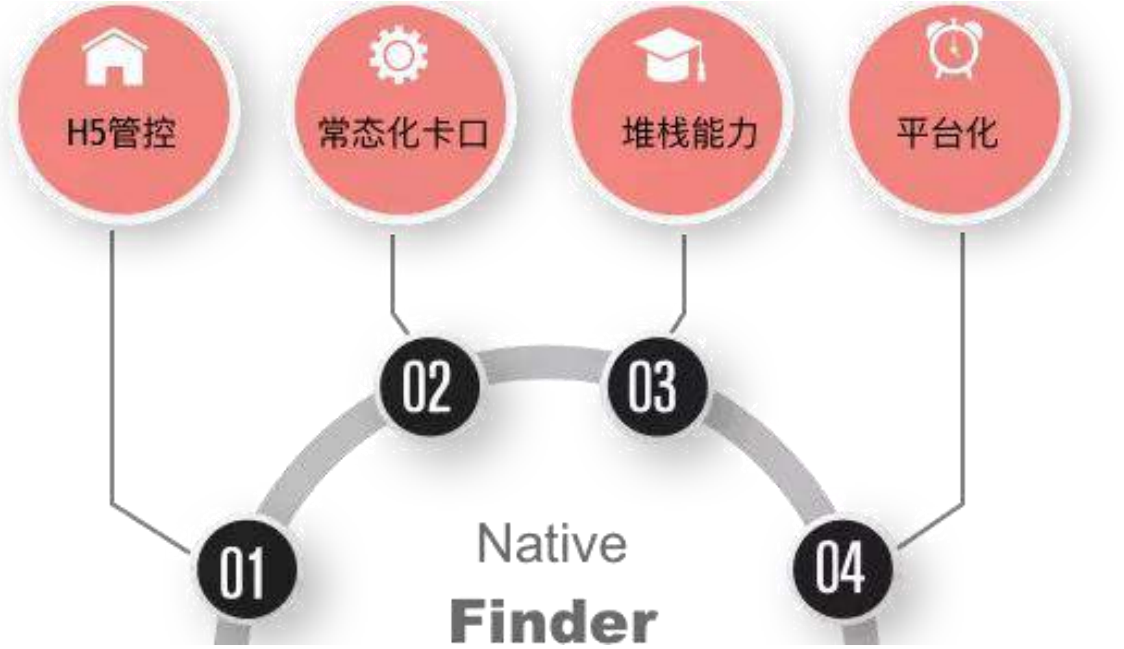
同时手淘日常的 native crash 率也有明显降低。到这里，我们 native crash 已经初见成效。虽然通过治理，陆续发现了不少问题，但是还远远不够，手淘内存问题依然严峻，特别

是双十一场景下，互动、活动以及各种场景链路互拉的场景，内存问题还是很严峻，后续还需更加努力。

展望

接下来我们希望还能做的更多，现在才刚刚开始；手淘内存问题依然严峻，要彻底治理，需要建立卡口，发现问题自动加入必改问题池，形成良性循环，旧问题不断发现解决的同时，还需要杜绝新问题的引入。

除了 native so 导致的内存问题，当前 H5 场景的内存问题也比较严峻，去手淘随便拿几个 H5 页面，看一下内存增量，都超过内存标准，H5 缺乏管控，下半年需要对 H5 内存做卡口严控；目前针对前端的内存泄漏，还没有有效的手段去检测发现，从 native 层看，都体现在内核上，内核应该是没有问题的，如果有效发现前端代码导致的内存泄漏，也是一个值得研究的点，不过先做卡口再做进一步突破。



手淘引入 Swift :从探索到落地过程实践

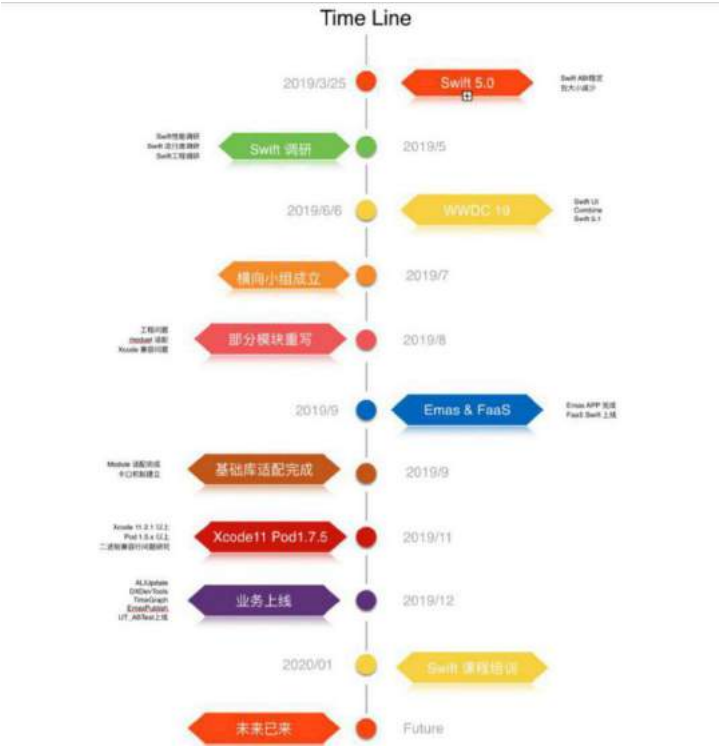
作者 | 姜沂(倾寒)
出品 | 淘系技术

背景

手淘 iOS APP 在 2019 年经过了约一年的时间，完成了 Swift 语言从调研到基础设施建设再到顺利落地业务。

手淘作为一个航母级别的 APP， 组织结构，工程结构，都是普通 APP 难以企及的，在手淘中实践犹如在沼泽地艰难探索，期间和集团内众多 Swift 爱好者，中间件负责人，一起努力探索出一条较为明朗的 Swift 落地指南。

时间轴



Swift 预研

Swift 语言在 2018 年就已经宣布 ABI 稳定是最重要的目标，虽然早在 Swift 4.x 时代，语法就已基本不变，但受限于手淘是一个航母级 APP， 工程交付以二进制组件化为主，ABI 未稳定的 Swift 只能固定 Swift Toolchain 的版本，加之会有将近 3MB 的包大小压力，彼时都不适合引入 Swift。苹果与 2019 年 3 月 25 日发布了 Swift 5.0 版本，宣布了 ABI 稳定，彼时又让团队的 Swift 燃起了信心，架构组作为新技术的把控团队，决定在百忙之余让团队成员各自分担一些对 Swift 的调研工作。

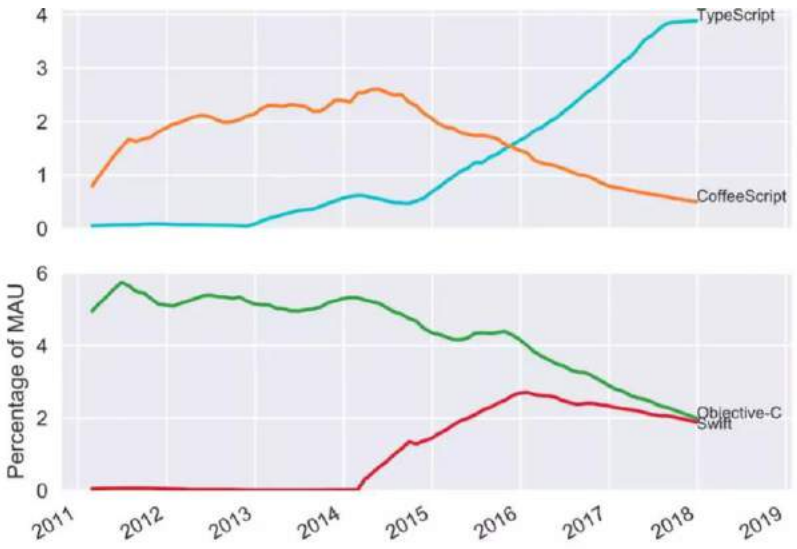
在工作之余，我们完成了调研工作，从中得到两个比较重要的结论。

Swift 流行度

我们通过爬虫分析国内外 APP Store 排行榜 Top1000 的 APP，通过文件扫描分析得到结论。

1. 国内使用 Swift 的 APP 约占比 22%，美区使用 Swift 的 APP 约占比 78%，其中美区剩余没有使用 Swift 的 APP 大部分来自中国地区本地化的产品，如抖音，快手等，可以得出一个结论，国内还是小众的 Swift，在国外已经是现状。
2. Github/Stack Over Flow 社区等 Objective-C 开源库和问题提问已经基本停滞，未来我们在落地新技术，Objective-C 可能已经是最坏的打算，加之 WWDC 17 年以来，苹果不再提供 Objective-C 的示例，组内同学也多次遇见 Objective-C Bug 去社区提问，毫无热度的情况。
3. 苹果在 WWDC19 年发布了 4 个 Pure Swift 框架，无法简单的被 Objective-C 混编。

未来我们极有可能因为苹果的强制推进风格和社区文化的落后产生技术踏空，无法迅速响应业务，甚至无法招聘到会使用 Objective-C 的工程师。



Swift 研发效率

Swift 从诞生之初就是一门先进的语言，主要有三个特性。

1. Safe Swift 从语法层面避免了很多未定义的行为，空值访问，值类型，对工程的稳定性有非常正面的好处，曾经面试一个候选人，简单重写为 Swift 让项目的崩溃率降低30%。
2. Fast，Swift 的语言设计让更多的方法调度通过静态调度完成，语言运行时效率优于 Objective - C。

3. Expressive， 富有表现力的语法特性，让代码量清晰已理解，减少的代码量约有 30%-50%不等，简洁的代码也让一些 Bug 无处藏身，也让工程师能更高效的支撑业务发展，大大提升了研发效率。

优势项	Swift
新特性	元组、闭包、泛型、命名空间
先进性	函数改进、高阶函数，struct、enum更强大灵活，控制流更先进（如defer、区间操作符、switch case任意类型）
安全性	类型安全、禁止隐式转换，编译期容易发现问题，guard、optional保护
包大小	编译后减少10%（strip后）
编码效率	代码更简短，可减少编码量20%~30%
性能	没有动态派发，函数调用更快
跨平台	支持
编程范式	面向协议编程、函数式编程、SwiftUI、Combine

下图是一个业务模块重写后的代码量对比：

其他不利因素

Swift 虽然在 5.0 版本完成了 ABI 稳定，但是在低版本操作系统中 （iOS 12.2 以下）仍旧会有一些不够完美的地方。

1. 在低于 iOS12.2 以下的操作系统会带来约有 3MB 的包大小问题，但幸运的是苹果放开了蜂窝数据网络下 200M 的下载大小。在 iOS13 上甚至可以超过 200MB。
2. 在低于 iOS12.2 以下的操作系统会有 100-200ms 不等的启动耗时增加，但在团队同学的努力下上线了二进制重排，启动性能大幅度上升 具体分析请看 手淘架构组最新实践 | iOS 基于静态库插桩的二进制重排启动优化。

集团横向组织成立

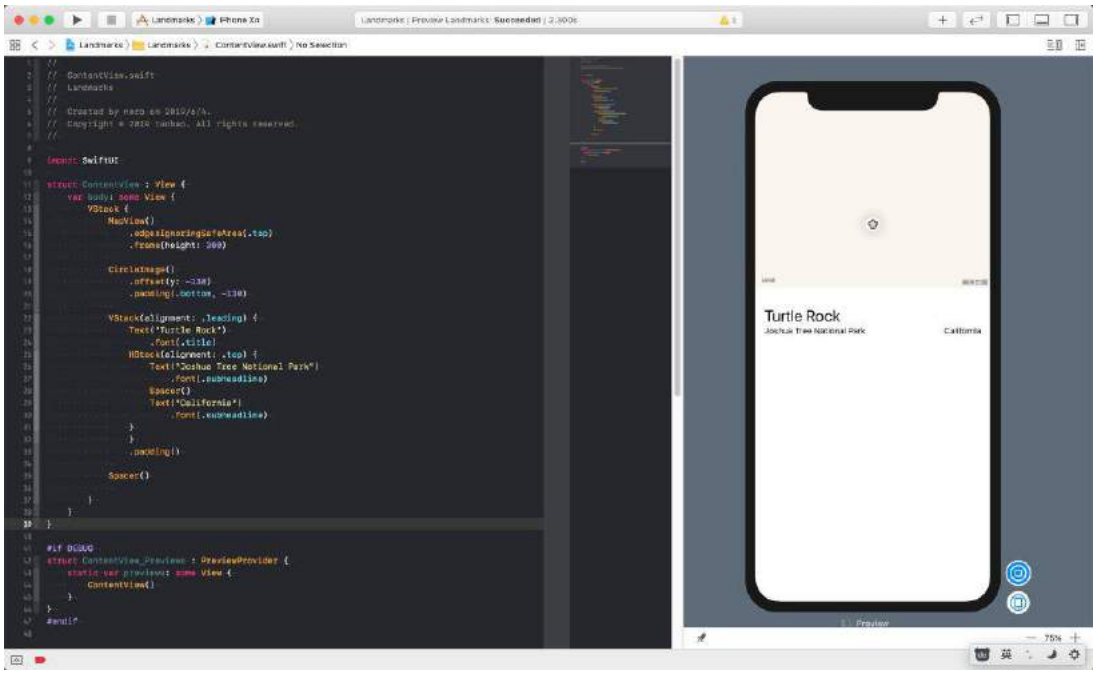
在充分的说明了 Swift 的优势和未来的情况下，我们决定不再停留与各种了解，准备下决心落地 Swift，毕竟所有的 iOSer 都知道，Swift 是未来，但是喊了这么多年的口号，总需要有组织的向前推进，只有自己做了并且落地了才有未来。

在基础会议上讨论了一些 关于 Swift 落地的计划和核心任务。如 Swift 基础建设，基础库适配，Xcode 升级， 以及新框架 SwiftUI， Combine， 容器化架构等。
并逐步拆解任务，落地下去。

SwiftUI & GAIA FaaS

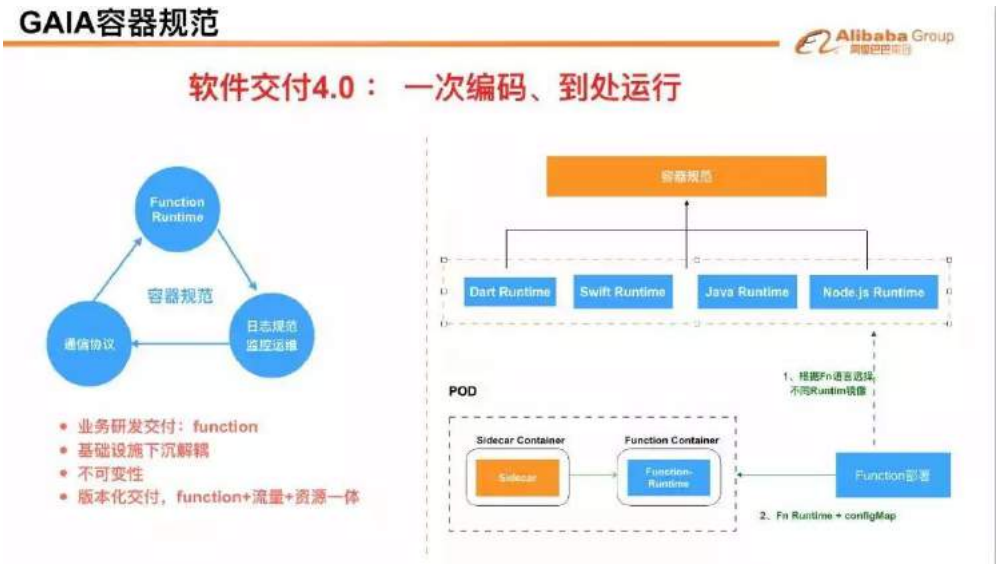
在了解 Swift 的同时恰逢 WWDC19， 苹果与 WWDC 推出了跨 Apple 平台下一代声明式的 UI 布局框架 “SwiftUI”。作者第一时间了解了 SwiftUI 的方方面面的特性，简洁清晰的 DSL， 表现力十足的数据流管理，能减少传统 Cocoa 命令式编程 80% 的代码，对研发效率有极大提升。

以下是一个简单的代码展示：



同时我们日常工作中一直有值班看数据大盘的诉求，有时甚至需要在外出的时候查看，之前我们不得不携带工作电脑，迫切需要一款移动值班的 APP，刚好出来的 SwiftUI 新技术可以帮助我们快速落地原型。

但此时没有后端支持，我们想到了 GAIA 平台， GAIA 平台是一款 FaaS 平台的风格，且 Swift 是一门跨平台语言，早在很早 IBM 就曾有 Serverless 服务，于是我们使用 Swift 遵守 GAIA 规范实现了一套 FaaS 服务，作为我们的数据大盘后端支撑。



之前已经编写了多篇文章分析了 SwiftUI 和 FaaS 的核心实现，这里就不再赘述，更多信息详见。（点击标题即可阅读）

1. 新晋网红 SwiftUI——淘宝带你初体验
2. 系列文章深度解读|SwiftUI 背后那些事儿
3. 历时五天用 SwiftUI 做了一款 APP，阿里工程师如何做的？
4. Swift 在 GAIA 平台云端一体化的探索

SwiftUI 的未来

随着前面的调研，使用分析等，虽然 SwiftUI 在初期使用 UIKit/APPKit 在性能上略显不足于 Flutter，但是 SwiftUI 的 DSL 和其他特性比 Flutter 简介不少，相信苹果在未来的版本种会持续优化，现在的技术储备让我们可以轻松应对未来 SwiftUI 的落地。

业务模块重写

手淘在多年的历史迭代中，或多或少遗留了一些历史包袱，如 Cocoapods 不规范使用，头文件管理不规范，工程模版老旧。为了发现是否存在工程问题会导致 Swift 无法落地业务，我们尝试编写一个代码量适中的 SDK，来验证 Swift 的研发效率等提升。完成 SDK 初步重写后，我们发现了现有工程中遗留了大量的问题，导致现有的项目完全无法使用 Swift。

主要存在以下问题：

1. 头文件不规范无法混编。
2. 不支持 Module 无法被 Swift 导入。
3. 工具链老旧支持 Swift 源码开发调试。
4. Swift 5.0 不是 Module Stability，对 Xcode 版本有强制限制。

适配基础库

为了解决 Swift 落地工程中的问题，我们结合集团 Swift 横向组织的力量，号召 Swift 爱好者，调研适配技术，编写适配文档，提供自动化工具，梳理出大量的 Swift 适配文档，推动中间件负责的同学，在不影响业务业务迭代的同时，顺手改掉历史包袱，满足 Swift 业务。

截至目前阶段，我们适配了约 100+ 基础库模块， 开启了强制打包卡口，长期有效的治理历史包袱，部分模块在治理历史包袱时，对自己的打包时常，和研发效率都有不小的提升。

适配 Xcode + Cocoapods

手淘的开发模式不同于其他中小型 APP，我们的开发团队复杂，交付方式主要由打包 SDK 二进制集成为主，但 Swift 5.1 才满足了 Module Stability，苹果于 9 月份才正式推出 Swift 5.1，我们开始着手升级 Xcode11。

对于手淘这样的 APP， 每年升级 Xcode 也会带来大量的 Break Change Future，我们通过查看官方文档，提供适配方案，推动业务方修改了大量 Bug 适应了大量 Futuer，我们与 11 月初完成了 Xcode11 的适配工作。

同时由于 Cocoapods 1.5.x 才支持 Swift 静态库，我们将 Cocoapods 升级到了 1.7.5，升级还带来不少 SDK 打包效率巨大的提升。

-	Xcode 10 + pod 1.2.0	Xcode 11 + pod 1.7.5
Swift 5.1	无法支持 Swift 环境	落地多个业务，且向前兼容
DarkMode	Xcode10 无法使用 DarkModeAPI	支撑业务适配 DarkMode 已上线
打包时间最短时间	8min 左右	5 分 30s
打包时间平均时间 (Debug)	14.5	8 min
打包时间平均时间 (Release)	14.2	9.7 min
源码环境构建 （初次 pod install)	3 分钟	1 分钟
源码环境（增量构建）	1 分钟	20s

Swift 二进制不兼容研究

Swift 是一门相对静态的语言，不同于 Objective-C 大部分通过 msg_send 动态调度，静态语言需要访问到其他依赖 SDK 的数据结构的静态布局信息，且手淘的研发方式是二进制依赖交付，容易造成下层 SDK 升级未通知上层 SDK 升级重新编译，带来的二进制不兼容问题很有可能留到线上在爆发，Swift 5.1 同时带来的 Library evoluation 功能帮助我们避免二进制不兼容问题发生。

以下是一份会打破二进制不兼容的 API 变动列表:

Change	Normal struct	@frozen struct
Adding fields	Allowed	Affects ABI
Reordering fields	Allowed	Affects ABI
Removing ABI-public fields	Affects ABI	Affects ABI
Removing non-ABI-public fields	Allowed	Affects ABI
Changing the type of an ABI-public field	Affects ABI	Affects ABI
Changing the type of a non-ABI-public field	Allowed	Affects ABI
Changing a stored instance property to computed	Allowed	Affects ABI
Changing a computed instance property to stored	Allowed	Affects ABI
Changing the access of a non-ABI-public field	Allowed	Allowed
Marking an internal field as @usableFromInline	Allowed	Allowed
Changing an internal ABI-public field to be public	Allowed	Allowed

业务完成

完成以上基本工作后，我们把之前重写的 SDK 通过回归测试后在手淘新版本正式上线，上线后的业务调用量等同于手淘 UV，且 0 Crash， 对开发同学造成了极大的鼓舞，多位同学主动尝试使用 Swift 落地业务。

培训

在初期的业务落地中，我 Review 了大量的代码，发现各位同学写的代码还带有浓浓的 Objective-C 风格，甚至由于不习惯可选类型，引入了强制解包等语法，这些语法会在安全性造成隐患，极易 Crash，我们了解到大量的同学 “看的懂 Swift 语法，但应对不了 Swift 风格和混编环境” 为了解决这些问题，我们组织了集团一些 Swift 大牛，指定规划了一列课程帮助各位开发者轻松的过渡到 Swift 开发环境中 未来我们会考虑向业界开发者提供，敬请期待。

未来已来

手淘在完成 Swift 落地实践中产出了大量的适配文档，工具，教程，方法论，在 2019 年初我们指定的目标和方向已经基本完成，我们做到了！！！在 Swift 诞生这 6 年时间里，各位开发者一直在声称 Swift 是未来，但是在我们手淘架构组的努力下，Swift 不再是未来，而是此刻。

在接下来的计划里，我们会提供一种渐进式的 SDK 迁移方案，逐步将老旧 SDK 迁移到 Swift 中去，Swift 的高效研发效率和安全性将更易支撑业务的快速发展。

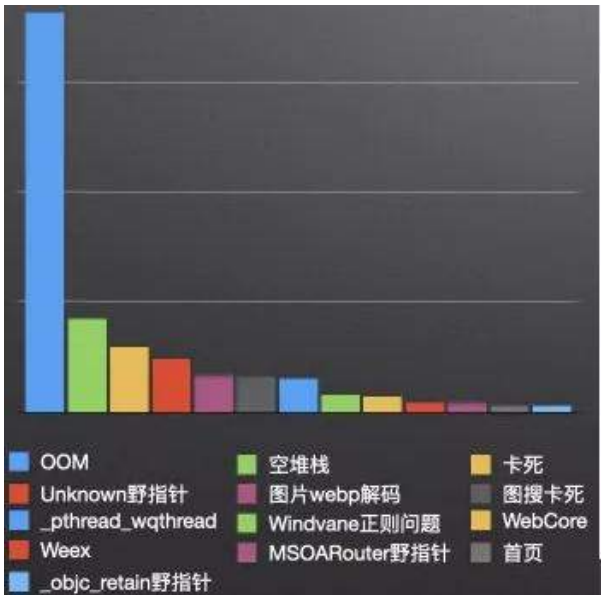
参考

手淘架构组最新实践 | iOS 基于静态库插桩的二进制重排启动优化
新晋网红 SwiftUI——淘宝带你初体验
系列文章深度解读|SwiftUI 背后那些事儿
历时五天用 SwiftUI 做了一款 APP，阿里工程师如何做的？
Swift 在 GAIA 平台云端一体化的探索
ranking-programming-languages-by-github-users
Library-Evoluation
Developer SwiftUI

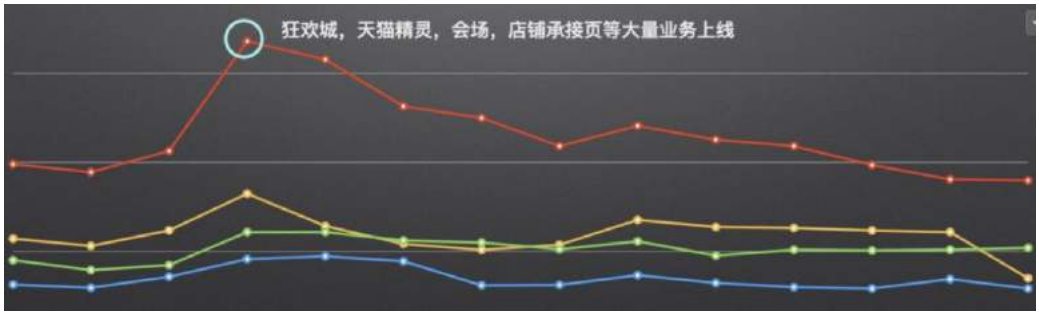
客户端稳定性优化实战，Crash 率最高下降 40%

作者 | 邹迪飞（之羲）
出品 | 淘系技术

大促一直是技术和产品的练兵场，每到大促，各种丰富的富媒体，如直播，视频，3D，游戏互动，AR 等竞相上线，在淘宝的大航母战略下，都集中在万千宠爱于一身的淘宝 App 上，在这样的大促场景下，开始触碰到端侧系统资源上限的天花板。在 17 年双 11 大促期间，端侧的内存问题尤为突显，OOM 的高居所有问题的榜首。内存问题成为了这几年大促端侧稳定性最大的挑战。



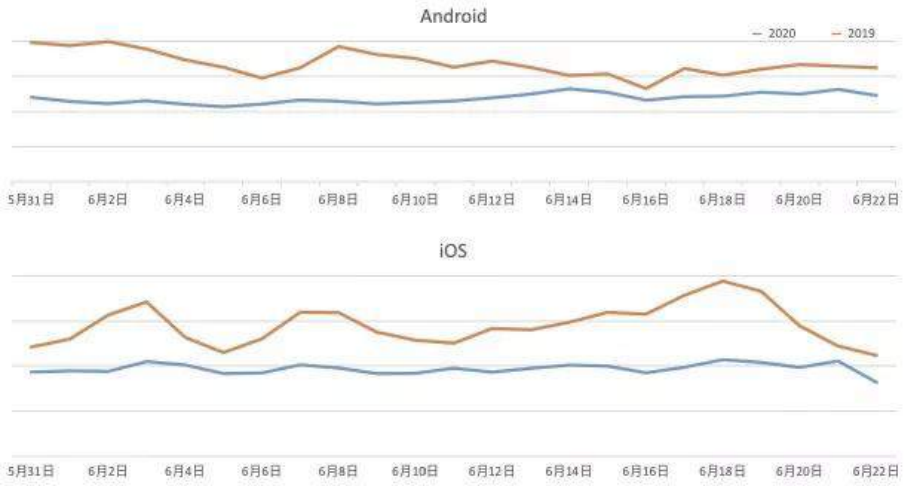
17 年双 11 Crash 问题分类



17 年双 11 Crash 走势与业务上线关系

放飞业务

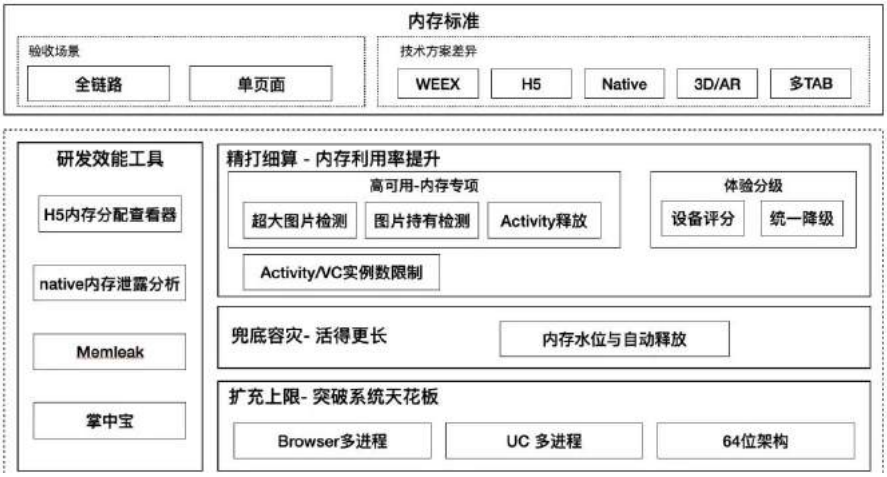
两年后的今天，通过我们持续的技术挖掘与治理，内存问题不再是影响大促稳定性的最主要的因素，本次 618 大促前所未有的支持了猜你喜欢无限坑位，支持了丰富的直播和小视频玩法，支持了会场上百个运营坑位，支持了互动业务的不降级策略，各种业务花式上线的同时，我们的端侧稳定性还进一步提升，crash 率远好于去年同期。



618 期间今年与去年对比 crash 走势

迎难而上，各显神通

面对内存带来的挑战，我们 2 年以来，一直在摸索中前行，沉淀一套内存治理的经验。



面向大促，当出现了问题后，我们要去思考当前的机制与规范，于是我们制定了内存标准与业务的上线验收。同时，提供了内存分析的一套工具，方便很快很准找到问题。同时，我们制定了三套内存优化策略：

1. 精打细算，提升内存的使用率
2. 兜底容灾，尽量让应用延长生命
3. 提升内存上限，突破系统的天花板

验收标准-----一夫当关

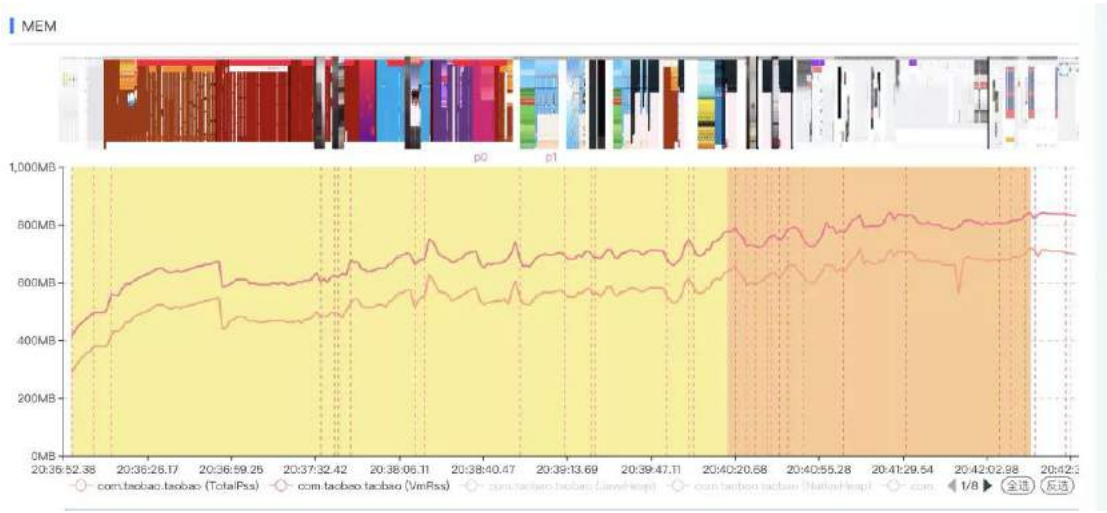
由于内存天花板的存在，从稳定性角度综合考虑，引入了大促的验收标准。标准的制定过程中，我们统计了发生 OOM 时的水位内位，分析出了高危，危险，正常水位线，以此为内存标准的制定指引。

内存问题之所以复杂，是因为内存是一个全局共享池子，当出现溢出问题时，在没有明显问题时，很难去界定哪个业务存在问题，因此，在考虑标准的时候，我们定义了两种场景。单页面及链路。

单页面场景主要是为了减少单个业务过多的占用内存引发的风险。前面提到内存池子是全局且有限的，当单页面占据内存过多，就会导致系统整体可用的内存大幅减少，在浏览相同页面次数的情况，增加整体内存风险。

链路场景是对常见浏览链路的内存检测，比如从首页-会场-互动-店铺-详情-下单这样的常规玩法进行多页面叠加检测，判断用户正常场景下的内存风险。

同时，在制度内存标准时，也考虑了不同技术栈之间的差异。比如 H5,weex,native，包括多 tab 的会场形式及直播，3D 等。



测试同学研发的 TMQ 自动化测试工具

内存优化三板斧

前面提到内存优化主要有三种策略，这里分开详述。

精打细算-提升内存的利用率

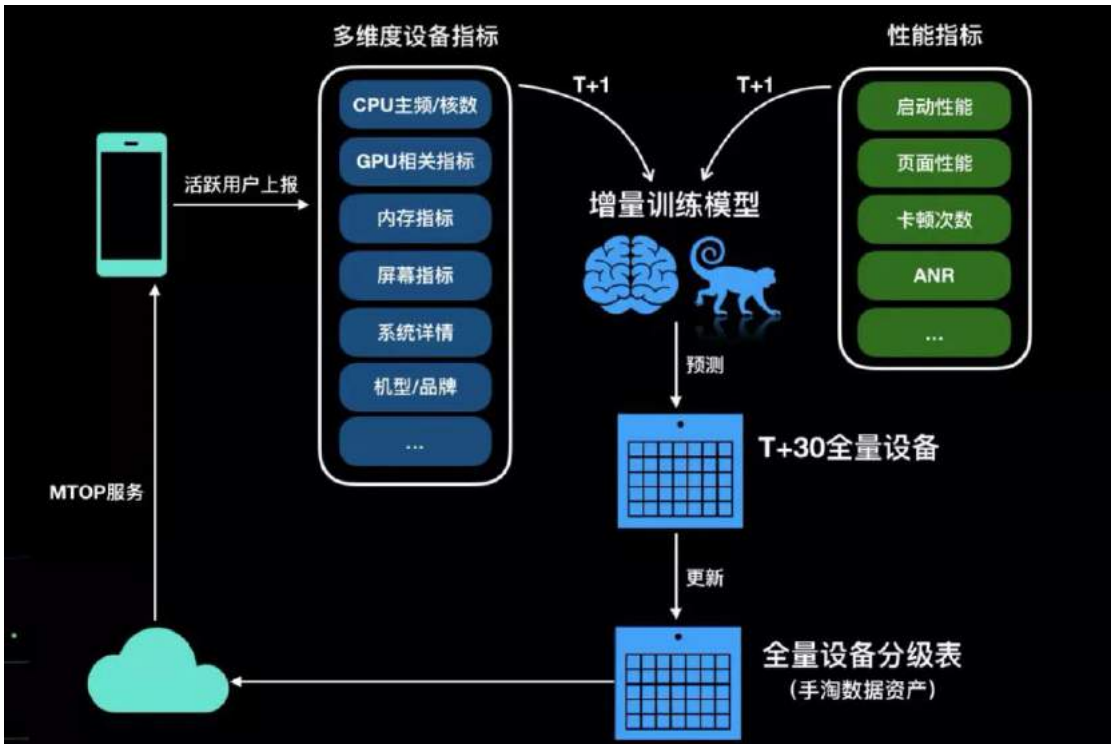
在业务屡屡触及内存天花板的情况下，每 1KB 的内存，都显得非常珍贵。

在实际对内存占用的分析中，我们偶尔会发现有些场景加载的图片远大于视图的大小，造成内存的浪费。或者在某些场景下，图片在内存中持有过久，比如在后台或是压栈很久后，图片所持有的空间仍不能释放出来给当前界面使用，面对这样的场景，我们在高可用体系中引用了对应的功能，能够检测出这些 case，以便把内存交给用户正在使用的组件，以此来提升内存的利用率。

从图片库的数据流转以及 View 生命周期出发，来设计图片自动回收和恢复的实现，即当 View 不可见的时候，自动释放图片到图片库缓存，只保留图片的 key 值；当 View 可见的时候，又通过 key 恢复图片。图片片自带三级缓存策略，回收后的图片如果还在缓存，能立马恢复，对体验几乎无损。

同时，针对一些内存大户，也和各业务方约定一些实例数限制，比如详情页，有大图，还带视频，webview 等，内存使用相对较大，这种情况下会对实例数做要求。目前有限制包括详情页，播放器实例等。

为了更好的体验，在降级策略上我们也做了一些优化，不再一刀切，而是根据各设备自身的能力，有选择的进行降级。要更好的达成目标，我们首先对设备进行分级，依赖于创建的智能分级。



统一降级在设备评分的基础上，提供默认的高中低端机型的设备分级能力，增加了配置化能力，为每个核心业务分配一个 orange，支持业务对系统、品牌、机型、设备分、应用版本、生效时间等多个维度进行配置化降级。

依赖于统一降级，可以做到精准的体验分级，高端机型，可以采用各种特效和高清图片，能保障最优体验。中端机型，降级掉一部分特效，可以获得较好效果，低端机型，保障稳定性和基础体验。实现“高端设备最炫体验，低端设备流畅优先，紧急问题快速降级”

统一降级后的效果

兜底容灾--尽量延长生命周期

在应用内存最危险的时候，也许下一次的内存申请即面临崩溃，在这最危险的时候，我们是否有能力缓解一下，让用户多下一单呢，为此我们设计了内存容灾 SDK。

具体原理是基于 gc 和 lowmemorykiller 原理实现（Android 的 OOM 要区分 jvm heap 内存不足和 native 的内存不足），通过监听系统的 gc 和 lowmemorykiller，去计算系统当前所处的内存状态，当内存不足的时候，销毁掉优先级较低的 Activity，从而保障用户可见面 Activity 能尽可能多的使用内存而不出现稳定性问题。

内存容灾基本原理

扩充上限-突破系统天花板

手淘的战略一直是航母战略，前面的打法只能缓解当前的稳定性问题，只能治标，不能治本。业务技术对内存的需求有增无减，无限坑位，会场上的直播视频等，都带来进一步的压力。只有提升端内的内存容量，才是解决内存问题的最终解法。

多进程

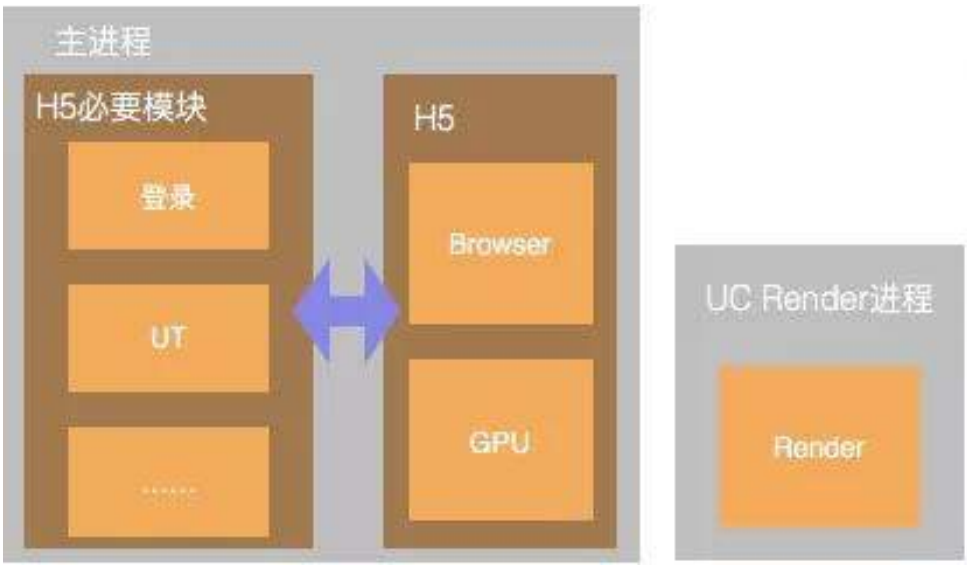
多进程的使用是突破系统天花板的方法之一。由于大促态的变化新增以 H5 的页面居多，所以我们重点希望在 webview 上能有一些突破。这时苹果的 WKWebivew 被纳入到研究范围，关于 WKWebview 在内存的优势，经过我们的分析结论如下：

WKWebView 的内存并不计算在主应用的内存中，而是作为独立进程单独进行计算，因此对于应用来说使用 WKWebView 相比 UIWebView，应用整体可以使用更多的内存，因为 Web 的内存都在 WKWewbView 的 Web 进程中，并不影响主应用的内存上限。

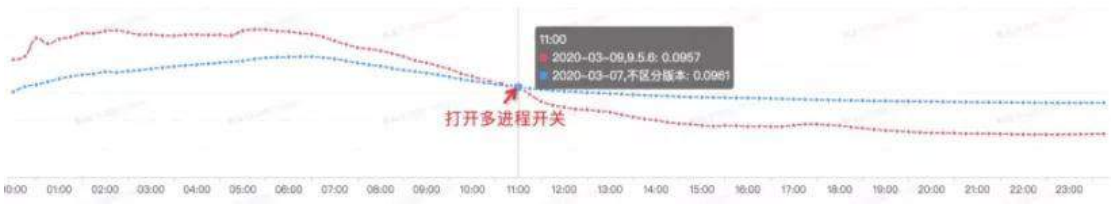
对于 Android 来说，平台本身则支持的多进程方式，因此，我们最初的设计中，是依赖于 Activity 的独立进程方式，即使 BrowserActivity 独立出来。

在 99 大促的 AB 实验中，对比对照组，在访问过淘金币/互动的用户中，主进程 nativecrash 率下降 15%-18%，Crash 计数(主+子) 下降 1 万次以上。在所有用户中，下降 3%-5%，对内存的优化效果还是比较明显。

但是考虑到很多基础 SDK 在设计之初并没有考虑到多进程的方式，且多进程下应用的生命周期也有一些变化，整体方案不确认的风险较大。最终采用的是 UC 内核的多进程方案，它将整个 H5 页面的解析、排版、js 执行等实现抽离封装到一个独立的进程中，分担了主进程一部分内存压力，从而实现突破单进程内存容量天花板的目标。



UC 多进程示意图



uc 多进程对 crash 率的影响

根据严格 AB 实验的结果评估，手淘开启 UC 多进程之后，Crash 率能有 30-40%的下降收益。

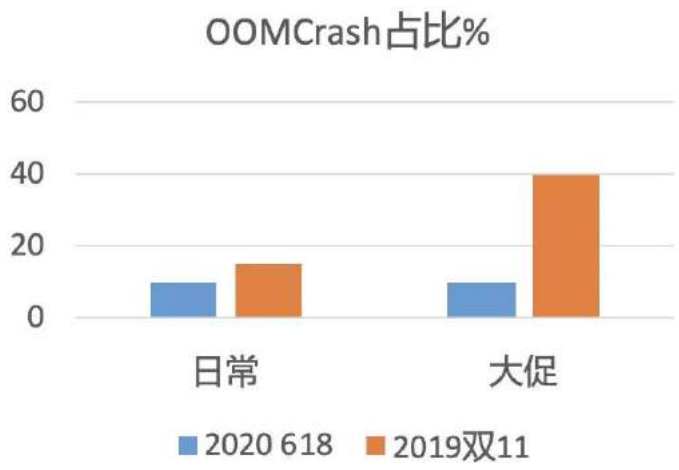
64 位升级

一般说来，现在使用的程序，都是在 32 位的指令集下编译出来的结果，在 32 位子系统下，内存地址的大小只有 4 个字节，理论上的最大寻址空间只有 4G。前面提到，在当前手淘的业务容量下，4G 的内存地址已经不能满足，在今年开始力推手淘 andorid 架构从 32 位升级到 64 位。

说到 64 位，在硬件上，arm v8 及以后的 cpu 都升级到了 64 位的架构，在软件上，android 5.0 以后的系统开始支持了 64 位子系统。我们做过比较准确统计埋点，在市面上的手机，约有 95%是支持 64 位运算的，也就是说 64 位升级带来的收益可以覆盖到绝大多数的用户。另一方面，我们也要看到 64 位升级带来的风险，所有的 C/C++代码都需要重新编译到 64 位指令集，可能的风险点包括：

- 指针长度是从 32 位升到 64 位，对一些 HardCode 的写法可能计算出错，产生稳定性问题。
- 自定义 so 的加载逻辑（如服务端远程下载）可能没有考虑多 cpu abi 的情况，加载错误 so，产生稳定性问题。
- 储存的数据，看看有没有因为 64 位和 32 位不同导致不兼容，特别是一些二进制数据，导致覆盖安装或升级后原数据不可用

为了应对这些风险，自 3 月份起就开始针对手淘中的 120 多个 so 进行回归，包括看 32 位与 64 位相互覆盖的升级场景，另一方面，针对 so 的加载逻辑，进行全手淘代码扫描，分析和查看自定义加载 so 的场景确认是否支持多 cpuabi。经过几个月的灰度和迭代，最终在 618 版本前，完成了 64 位的上线。



在本次的 618 大促中，可以明显看到，以往大促中，OOM 的占比，最高的时候，可以占到近 40%，经过 64 位升级与多进程手段叠加后，可以看到看大促态的 OOM 占比，已经降到了 10%左右。这里面还包括了 5%的 32 位用户，对 OOM 的治理效果非常显著。

展望

新技术形态的挑战

内存问题一直是大促端侧稳定性最大的挑战，在今天已经得到比较好的解决，当然，系统资源终归是有限的，我们仍然需要有效合理的使用系统资源。更重要的是，面向未来，新的技术形态像 flutter 等入淘，多个虚似机的并存，对系统资源仍然会有较大的挑战。

从稳定性到流畅体验

对用户来说，稳定性只是最基础要求，后续我们会在体验上持续优化，带给手淘用户真正的如丝般顺滑的体验。

IOS 开发体验优化方案

作者 | 柬超
出品 | 淘系技术

随着 Flutter 等跨端框架的出现，业务开发同学经常需要在 Android/IOS 上跨端进行业务开发，问题定位等。新的不熟悉的环境的搭建总会遇到各种各样的问题，导致搭建失败，特别是 IOS 开发环境，是最复杂的，不仅环境搭建繁琐，而且切分支后的打包速度很慢，所以我们设计实现了两个工具，用于优化闲鱼 IOS 开发体验。

IOS 开发体验存在的问题

开发环境搭建难

开发环境依赖特定软件版本，配置复杂

闲鱼 IOS 工程不仅依赖 XCode，还依赖了 taobaoenv 1.2.0 和 cocoapods 1.2.0 这两个包管理工具。根据大家的经验，这两个工具在 ruby2.3.x 时，问题较少。特定软件版本，系统自带软件版本冲突，环境变量设置等等一些列的操作步骤导致环境搭建复杂，需要求助 IOS 开发同学才能搞定。

维护难

Mac 系统升级后，cocoapod 容易出现问题，不得不重新搭建开发环境。具体原因也是多种多样：系统环境变量变了，导致找不到特定版本 ruby；ruby 随系统升级导致 cocoapod 不能用，需要重新安装；Gem 版本问题；Ruby 源问题等等。这也导致许多开发同学不敢轻易的去升级系统，无法及时体验到新系统的特性。

Pod 依赖下载量大

由于 cocoapod 本身的工作原理，pod 更新下载工程依赖时，会下载各个版本的文件信息，总量特别大。以闲鱼 IOS 工程为例，总共需要下载近 20G 的缓存文件，而且大部分都是几 K 的小文件，下载时间可能会持续十几个小时，导致新环境搭建到初次体验时间跨度非常久。

切分后 APP 打包慢

当开发同学在多个分支/版本开发的时候，时常需要切换分支开发调试和 bugfix。但是切换分支之后，整个 IOS 工程打包时间在 30-40 分钟左右。有时候为了修复一个版本的 bug，不得不切换分支，然后重新打包调试。修复和验证 bug 可能只需要五分钟，打包却用 30 多分钟，投入产出不成比例。

为了解决这些存在的问题，我们进行了一些列的探索，跟大家一起分享下，也欢迎有更好的解决方案出现。

IOS 环境搭建

虚拟化技术的不断发展，为我们统一端侧开发环境提供了新思路，我们设想如果 IOS 开发环境能够跟 Mac 解耦，且可以移植，大家可以轻松复用，那么第一二个问题就迎刃而解了。为此我们做了几个尝试：

虚拟机方案

在 MacOS 本地搭建虚拟机，内装 MacOS 系统。在虚拟机内搭建 IOS 开发环境，然后通过虚拟机镜像 copy 实现 IOS 开发环境移植，解决环境搭建难题。

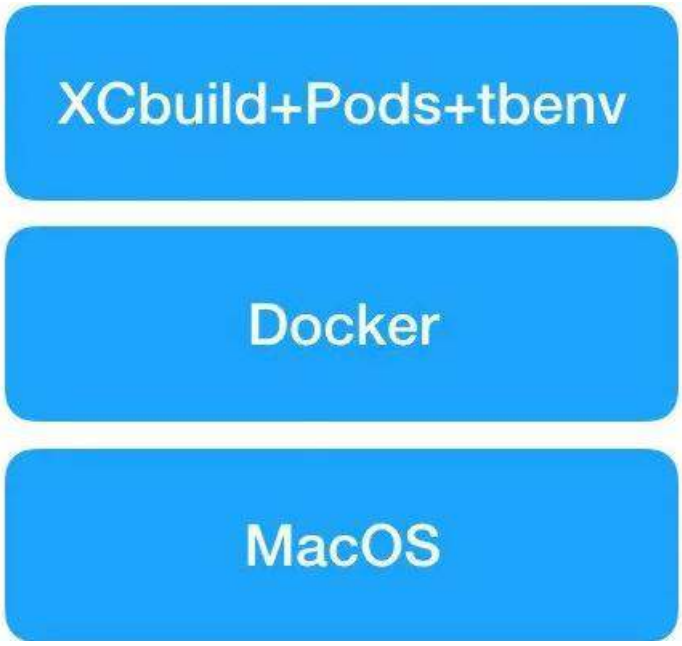
这个方案存在以下几个问题：

1. 性能问题：IOS 的编译过程是一个 IO 密集型和 CPU 密集型操作，虚拟机通过虚拟 HOST 系统的磁盘和 CPU，性能会大打折扣，导致编译时间变长，影响开发体验。
2. 安全问题：在 Mac 工作机安装虚拟机，需要通过公司安全审核。
3. 黑苹果问题：虚拟机内的 Mac 系统是没有经过授权的，会带来盗版侵权风险。

虚拟机是比较重的虚拟化技术，因此我们转向了更加轻量化的 docker 技术。

完全 Docker 化

将 IOS 开发依赖的软件和环境变量全部 docker 化。通过 docker 镜像实现 IOS 开发环境的移植。对于 cocoapod，taobaoenv 等 ruby 类工具，鉴于 ruby 的跨平台特性，可以很方便的迁移到 docker 内。但是对于强依赖 MacOS 的 XCode，我们尝试用 Facebook 出品的 xcbuild 替代。这是一款兼容 xcodebuild 的编译工具，网上也确实有网友用这个软件搭建 IOS 编译环境。



这个方案存在以下几个问题：

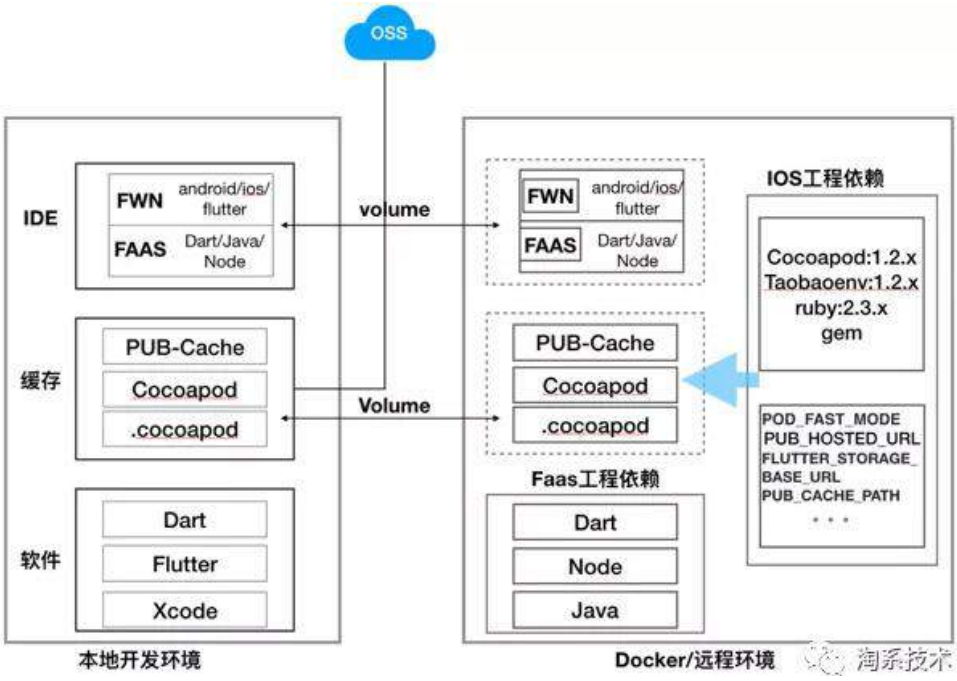
1. Xcbuild 跟 Xcodebuild 的兼容性无法评估。
2. xcode 升级后，xcbuild 跟进升级兼容的一段时间内，只能回退到原来的开发方案，来回切换开发环境，体验差。

上面两个方案都没有很好的解决 IOS 开发环境移植和解耦的问题，但是在完全 docker 化的尝试中，我们发现最复杂的 cocoapod 和 ruby 安装配置部分是能够 docker 化的，xcode 安装后并不需要特殊的配置，因此我们设计实现了一个折中方案：Host 内开发（部分 docker 化）。

Host 内开发（部分 docker 化）

本方案中：开发编译调试工作仍然在 MacOS 本地，使用 xcode；而将 cocoapod 和 taobaoenv 相关的软件和环境变量配置等 docker 化。这样既遵循了开发同学一贯的开发体验，又兼顾了开发环境的可移植问题。

为了能够让 Docker 内 cocoapod 拉取的依赖文件和生成的 pod 工程能被本地的 XCode 识别，我们将本地 pod 缓存目录挂载到 docker，这样 Pod 拉取的依赖既能在 docker 内更新，也能在 MacOS 中被 XCode 访问，具体如下图（统一编程平面端+Faas 软件架构图）：



这样既做到了简化开发环境搭建的复杂度，方便了想尝试 IOS 开发的同学快速搭建环境，还能给开发同学无差别体验。而且通过这个方案，我们的 IOS 开发环境可以方便的在各个同学的开发环境中迁移，而且也可以统一进行升级改造。

本方案将 Pod 相关的依赖迁移到了 Docker 中，与 MacOS 解耦，因此 IOS 开发同学可以自由升级 Mac 系统，不用担心开发环境被破坏,解决了维护难的问题。为了解决新搭建的环境需要大量拉取 pod 依赖的问题，我们将 pod 的本地中间文件上传到 OSS 云盘（上图蓝色 OSS 云盘），开发同学只需要一次性下载压缩包并解压到本地，然后增量更新就可以了。

切分支后 APP 打包速度问题

客户端开发同学经常需要在多个分支（版本）上面开发业务，且时常需要来回切换进行业务开发和问题定位。这带来的一个问题是：当开发同学从 A 分支切换到 B 分支的时候，需要重新打包 APP，整个过程大概需要 30-40 分钟左右。

在分析了闲鱼 IOS 工程打包过程后，我们将耗时锁定在两个阶段：Pod 操作和 XCode 编译。打包速度优化也将分为两个阶段进行：

Pod 操作加速

Pod install/update 主要的工作是读取 Podfile，进行依赖版本控制和冲突解决，并生成 Pod 工程。生成的相关文件存储在 Pods 目录和 Pods.xcodeproj 中。当切换回之前分支时，Podfile 经常是不会发生变化的，因此重新生成 pod 工程实属浪费。

经过测试，如果我们将这些中间文件保存起来，多次切换分支后，这些中间文件仍然能够还原之前的 Pod 工程，从而避免切分支后重新生成 Pod 工程的步骤，省去 10 分钟左右的开销。

XCode 编译速度优化

对于 XCode 编译速度优化，网上有很多方案，大致可以分为三类：

- Cocopods 依赖编译加速
  比如 cocoapods-packager，它可以将 pod 依赖打包成 static library,IOS 工程以静态库的形式引入 pod 依赖，省去重复编译的时间。

但是这个方案也存在一些问题；私有库和第三方库更新很麻烦，每次都需要重新打包静态库，并上传到代码仓库；且很难调试源码

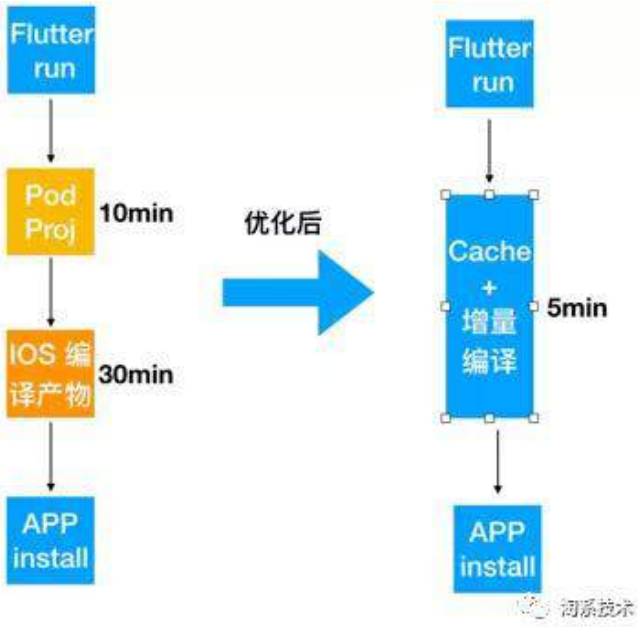
- **分布式编译：比如 distcc**

  分布式编译的原理是将需要编译的文件分散到编译集群的其他机器上编译，然后将编译好的二进制文件传回。本地编译器再将这些二进制文件链接在一起。分布式编译对于大工程提速明显，但是对于小工程，反而会拖累编译速度。

- **缓存编译的中间结果**

  CCache，BUCK 更为广泛的加速方案是缓存编译的中间结果，比如 CCache，Buck 等，这些方案，网上有详细的资料，不再一一赘述。但是引入这些方案，都需要对目前的 IOS 工程进行改造，甚至需要改变用户的开发习惯，因此不符合我们的要求。但是缓存中间编译结果的方案给我们提供了一些启发：我们知道 XCode 是具有增量编译能力的，这其实也是利用了上一次编译的中间产物，本地再次编译的时候，如果发现文件没有变化，则忽略这个文件，如果源码文件时间戳更新了，那么就重新编译这个文件，因为每次变化的源码都是少量的，这样就可以达到加快编译速度的目的。

对于闲鱼 IOS 工程，如果我们在切分支之前保存当前 IOS 分支编译的中间产物，然后在切换回当前分支的时候，恢复之前保存的中间产物，那么是不是就可以触发 XCode 增量编译了呢？事实确实如此。



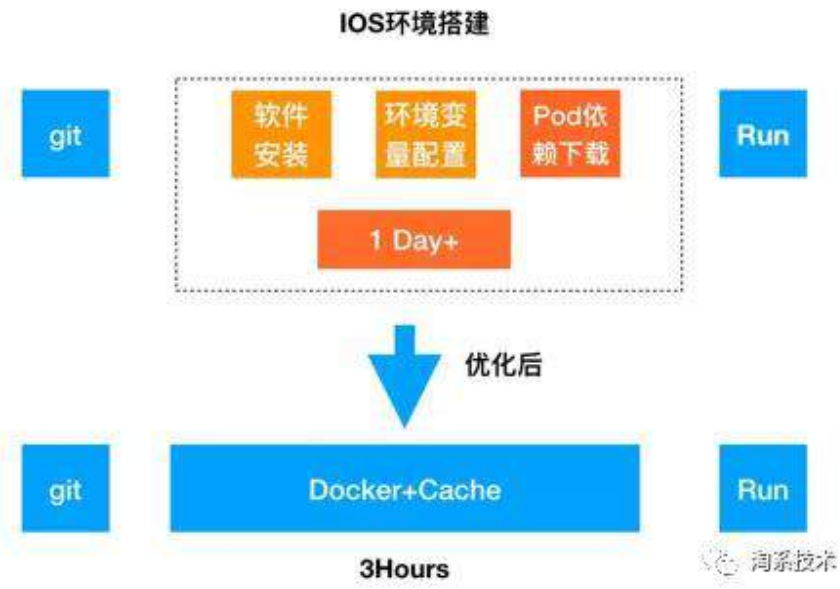
具体方案：

1. 在切分之前缓存当前分支的 Pods Project，Flutter Project 以及编译的中间产物，Podfile.lock，linkmap 等等相关文件。
2. 切换分支
3. 恢复新分支之前缓存的中间产物
4. 重新打包 IOS APP。

通过这几步优化，我们将闲鱼 IOS 工程切分支后的打包时间由原来的 30-40 分钟降低到五分钟以内，效率提升近六倍。

总结

IOS 环境搭建中复杂和耗时的步骤，通过 docker 镜像和缓存优化后，搭建的难度大大降低，IOS 新手也基本可以在三小时内搞定。



同时，通过缓存和复用打包过程产生的中间产物，切换分支后的打包耗时控制在五分钟内，降低为原来的六分之一，提升了开发效率。

IOS Crash 分析攻略

作者 | 刘坤（念纪）
出品 | 淘系技术

应用崩溃是影响 APP 体验的重要一环， 而崩溃定位也常常让开发者头疼。本文就讲讲关于 Crash 分析的那些事。

Crash 日志的渠道

Crash 日志从哪来？一般有 2 个渠道:

苹果收集的 Crash 日志

- 用户手机上 设置 -> 隐私 -> 分析 里面的，可以连接电脑 Xcode 导出。
- 在 Xcode -> Window -> Organizer -> Crashes 里面可以查看。

自己应用内收集的

- 接入一些 APM 产品， 如 EMAS、mPaaS、phabricator 等。
- 接入 PLCrashReporter 、 KSCrash 等 SDK 进行收集，上报到自建平台统计。

两者各有利弊，但是二者的捕获原理是差不多的。

苹果的日志

- 优点：理论上捕获类型最全，因为是 launchd 进程捕获的日志。
- 缺点：不是全量日志，因为需要用户隐私授权才会上报，没有数据化支撑。

自己收集的

- 优点：可以自建数据化支撑，获取 Crash 率等指标。
- 缺点：存在无法捕获的 Crash 的类型。

Crash 捕获的原理

要了解 Crash 捕获的原理，要先清楚几个基本概念，和它们之间的关系：

软件异常

软件异常主要来源于两个 API 的调用 kill()、 pthread_kill()，而 iOS 中我们常常遇到的 NSException 未捕获、 abort() 函数调用等，都属于这种情况。比如我们常看到 Crash 堆栈中有 pthead_kill 方法的调用。

硬件异常

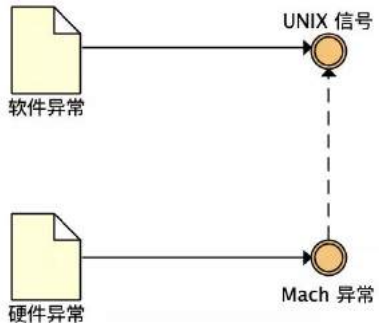
硬件产生的信号始于处理器 trap，处理器 trap 是平台相关的。比如我们遇到的野指针崩溃大部分是硬件异常。

Mach 异常

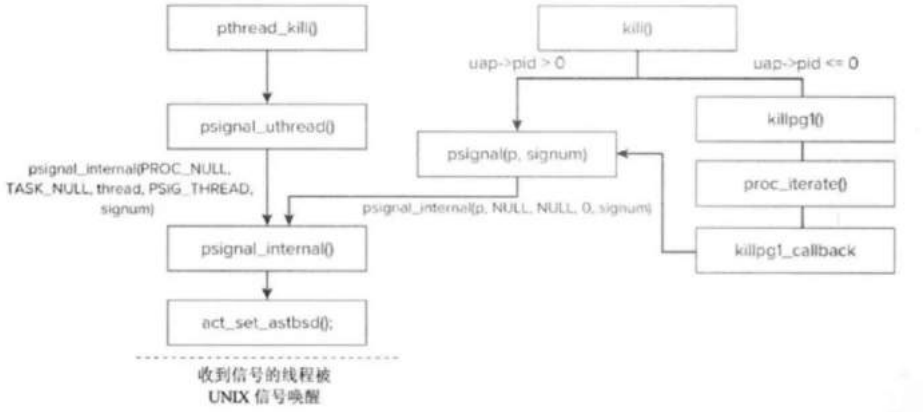
这里虽然叫异常，但是要和上面的两种分开来看。我们了解到苹果的内核 xnu 的核心是 Mach，在 Mach 之上建立了 BSD 层。“Mach 异常”是“Mach 异常处理流程”的简称。

UNIX 信号

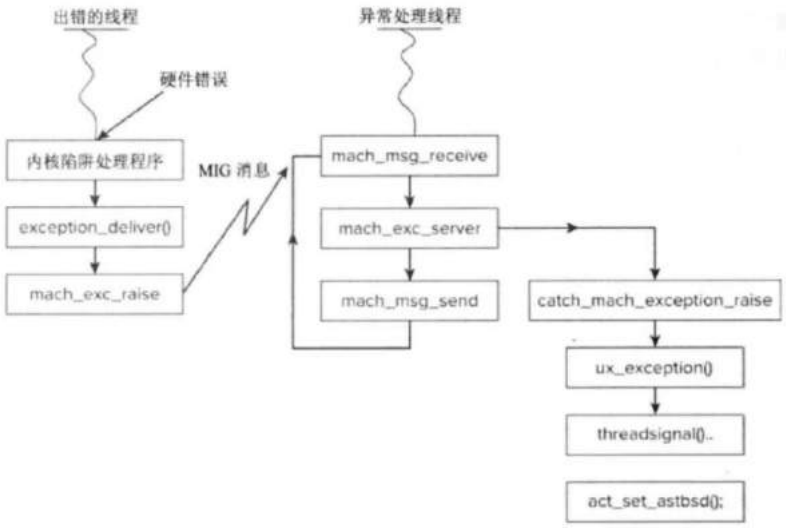
这就是我们常说的信号了，如 SIGBUS SIGSEGV SIGABRT SIGKILL 等。



软件产生的信号的处理流程如下图， 就是用户态的“UNIX 信号”处理流程：



硬件产生的信号的处理流程如下图：硬件错误被 Mach 层捕获，然后转换为对应的“UNIX 信号”。为了维护一个统一的机制，操作系统和用户产生的信号首先被转换为 "Mach 异常"，然后再转换为信号。如下图所示:



上面两图来自“深入解析 Mac OS X & iOS 操作系统”一书。

由上图可以看到，无论是硬件产生的信号，还是软件产生的信号，都会走到 act_set_astbsd() 进而唤醒收到信号的进程的某一个线程。这个机制就给我们在“自身进程内捕获 Crash”提供了可能性。就是可以通过拦截“UNIX 信号”或“Mach 异常”来捕获崩溃。

而且这里还有个小知识，当我们拦截信号处理之后是可以让程序不崩溃而继续运行的，但是不建议这样做，因为程序已经处于异常不可知状态。

PLCrashReporter 和 KSCrash 两个开源库都提供了 2 种方式拦截异常，包括“Mach 异常拦截”和“UNIX 信号拦截”。说到这里有人可能困惑了，看上图里面最终都会转换为“UNIX 信号”， 是不是代表我们只用监听“UNIX 信号”就够了呢？为什么还要拦截 Mach 异常呢？

有两个原因：

- 不是所有的 "Mach 异常”类型都映射到了“UNIX 信号”。 如 EXC_GUARD 。在苹果开源的 xnu 源码中可以看到这点。
- “UNIX 信号”在崩溃线程回调，如果遇到 Stackoverflow 问题，已经没有条件(栈空间)再执行回调代码了。

那么，是不是我们只拦截“Mach 异常”就够了呢？也不是，用户态的软件异常是直接走信号流程的，如果不拦截信号可能导致这部分 Crash 丢失。

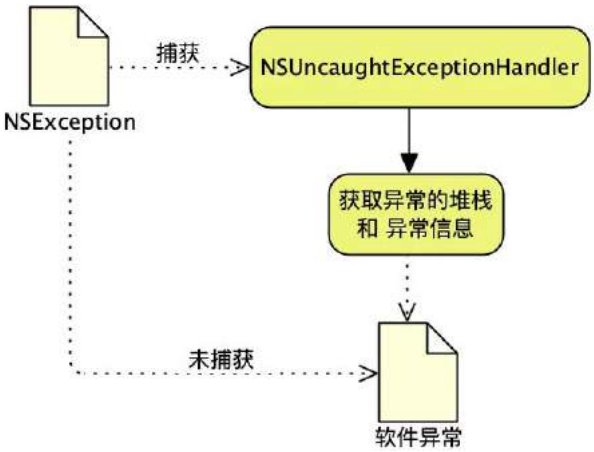
附“Mach 异常”与“UNIX 信号”的转换关系代码，来自 xnu 中的 bsd/uxkern/ux_exception.c ：

```
    switch(exception) {
    case EXC_BAD_ACCESS:
    if (code == KERN_INVALID_ADDRESS)
      *ux_signal = SIGSEGV;
```

```
else
  *ux_signal = SIGBUS;
break;

case EXC_BAD_INSTRUCTION:
  *ux_signal = SIGILL;
       break;

  case EXC_ARITHMETIC:
    *ux_signal = SIGFPE;
   break;

case EXC_EMULATION:
     *ux_signal = SIGEMT;
    break;

  case EXC_SOFTWARE:
    switch (code) {
    case EXC_UNIX_BAD_SYSCALL:
      *ux_signal = SIGSYS;
    break;
    case EXC_UNIX_BAD_PIPE:
      *ux_signal = SIGPIPE;
    break;
    case EXC_UNIX_ABORT:
      *ux_signal = SIGABRT;
    break;
    case EXC_SOFT_SIGNAL:
      *ux_signal = SIGKILL;
    break;
    }
    break;

    case EXC_BREAKPOINT:
      *ux_signal = SIGTRAP;
    break;
    }
```

看到这里，有同学可能会说，还有 NSException 呢？我们都用 NSUncaughtExceptionHandler 来捕获异常 Crash 的。在前面就将 c++/ObjC 异常归类到了“软件异常”类型，那是

不是"捕获信号"就行了呢？为什么还要注册 NSUncaughtExceptionHandler 呢？是因为 CrashReporter 需要通过这个 handler 来获取异常相关信息和堆栈。



看懂 Crash 日志

我们这里说的 Crash 日志，是指 Apple Format 格式的 全堆栈 Crash 日志。

Crash 头部信息

- Incident Identifier: 每个 Crash 生成的唯一的 uuid。
- CrashReporter Key: CrashReporter 的 uuid，如果自己捕获日志，这个可以忽略。
- Hardware Model: 机型。
- Process: 进程名和进程 ID。
- Path: 进程的可执行文件路径。
- Identifier: Info.plist 中配置的 CFBundleIdentifier 值。
- Version: CFBundleVersion (CFBundleVersionShort) 即应用的 Build 号+版本号。
- Code Type: 机型的 CPU 架构，但是不是详细的架构名。比如 arm64e 在这里也是 ARM-64。
- Parent Process: 父进程和进程 ID。
- Data/Time: Crash 发生的具体时间。
- Launch Time： 进程启动时间。

- OS Version: iOS 系统版本和 build 号。
- Report Version: Crash 日志格式的版本号，一般是 104。如果这个 version 偏高，用系统的 symbolicatecrash 命令不能符号化日志，一般如果看到是 204， 改成 104 之后用 symbolicatecrash 就可以符号化了。

Crash 异常码

在 Crash 头部信息之下，会有个段记录了 Crash 异常码。类似下图：

```
Exception Type:  EXC_BAD_ACCESS (SIGSEGV)
Exception Subtype: KERN_INVALID_ADDRESS at 0x00000000164ab280
Triggered by Thread:  0
```

这里我们应该关注：

- Exception Type: 异常码，一般格式是 Mach 异常码（ UNIX 信号类型 ）。
- Exception Subtype: 一般情况里面带的是 Mach 异常的 subcode，还有 Crash 相关地址信息。
- Triggered by Thread: 发生 Crash 的线程，大部分情况到这个线程的堆栈里面去看 Crash 堆栈。
- Application Specific Information: 如果是 Objc/c++ Exception 异常，这里是异常的信息，这个是定位异常的关键信息。
- Last Exception Backtrace: 抛出异常的代码堆栈，如果是 Objc/c++ Exception 异常造成的 Crash，就看这个堆栈，Crashed Thread: 里的堆栈是 abort()，没有意义。

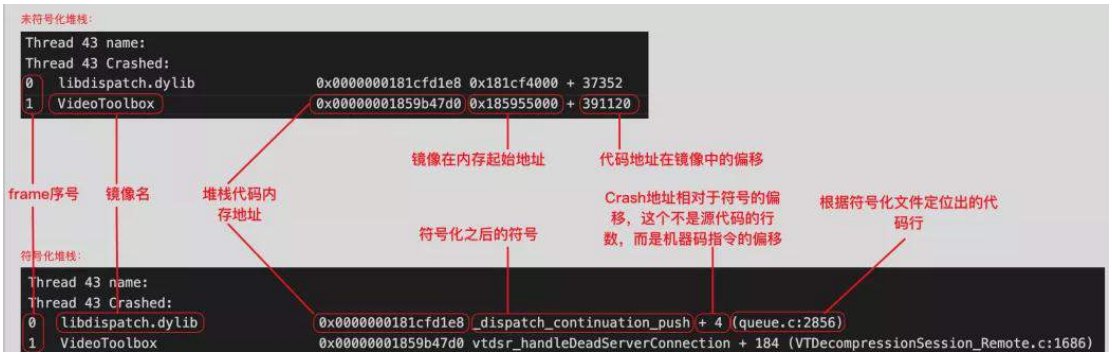
附异常码的解释，非所有异常码，只是我们 Crash 中可能会看到的 ：

Mach 异常	UNIX 信号	简介	案例
EXC_BAD_ACCESS	SIGBUS	总线错误	1、内存地址对齐出错 2、试图执行没有执行权限的代码地址
	SIGSEGV	段错误	1、访问未申请的虚拟内存地址 2、没有写权限的内存写入
EXC_BAD_INSTRUCTION	SIGILL	非法指令，即机器码指令不正确	1，iOS 上偶现的问题，遇到之后用户会连续闪退，直到应用二进制的缓存重新加载 或重启手机。此问题挺影响体验，但是报给苹果不认，因为苹果那边没有收集到，目前没有太好办法。因为 iOS 应用内无法对一篇内存同时获取 w+x 权限的，因此应用无法造成此类问题，所以判断是苹果的问题。
EXC_ARITHMETIC	SIGFPE	算术运算出错，比如除 0 错误	iOS 默认是不启用的，所以我们一般不会遇到
EXC_SOFTWARE (我们在 Crash 日志中一般不会看到这个类型，苹果的日志里会是 EXC_CRASH)	SIGSYS	系统调用异常	无
	SIGPIPE	管道破裂	1，Socket 通信是可能遇到，如读进程以及终止时，写进程继续写入数据。2，根据苹果的文档，我们可以忽略这个信号: https://developer.apple.com/library/archive/documentation/NetworkingInternetWeb/Conceptual/NetworkingOverview/CommonPitfalls/CommonPitfalls.html
	SIGABRT	abort() 发生的信号	典型的软件信号，通过 pthread_kill() 发送
	SIGKILL	9毫米子弹,大杀特杀。进程内无法拦截	1，exit()，kill(9) 等函数调用 2，iOS 系统杀进程用的，比如 watchDog 杀进程

Mach 异常	UNIX 信号	简介	案例
EXC_BREAKPOINT	SIGTRAP	由断点指令或其它 trap 指令产生	部分系统框架里面会用 __builtin_trap() 来产生一个 SIGTRAP 类型的 Crash
EXC_GUARD	无	文件句柄错误	试图 close 一个内核的 fd.
EXC_RESOURCE	无	资源受限	线程调度太频繁，子线程每秒被唤醒次数超过 150: https://stackoverflow.com/questions/25848441/app-shutdown-with-exc-resource-wakeups-exception-on-ios-8-gm

Crash 堆栈

下面一张图介绍了 Crash 堆栈中每个段的含义:



Thread State

Thread State 中记录了 Crash 线程的寄存器的值，对于一些问题的定位是有一定帮助的。

```
Thread 43 crashed with ARM Thread State (64-bit):
    x0: 0x00000001c46f0200   x1: 0x00000001c52699c0   x2: 0x0000000000000000   x3: 0x00000001c0637c60
    x4: 0x0000000000000000   x5: 0x00000001859b4718   x6: 0x0000000000000000   x7: 0x0000000000000000
    x8: 0x000021a1b2f2c0b1   x9: 0x7fffffffffffffff  x10: 0x0000000000000004  x11: 0x0000000000000000
   x12: 0x0000000000000000  x13: 0x001ffe2000000000  x14: 0x0000000000000000  x15: 0x04eebd0104eebdc0
   x16: 0x0000000181f91ab0  x17: 0x000000018225d4fc  x18: 0x0000000000000000  x19: 0x00000001c4387b80
   x20: 0x0000000170dba310  x21: 0x0000000170dba2f0  x22: 0x00000001b2f19e88  x23: 0x00000001b2f29000
   x24: 0x00000001b2f29000  x25: 0x0000000000000000  x26: 0x0000000116ee0040  x27: 0x00000001b2f23650
   x28: 0x0000000000000001   fp: 0x0000000170dba360   lr: 0x00000001859b47d0
    sp: 0x0000000170dba280   pc: 0x0000000181cfd1e8 cpsr: 0xa0000000
```

比如通过 fp, sp 的值配合 异常码 中的地址判断是否是栈溢出问题。但是大部分情况下，这块内容可以提供的帮助都比较小，因为都是单纯的数值，而不是其代表的：意义。在这一点上， KSCrash 提供了分析功能，可以对寄存器内值的意义进一步分析，比如分析是不是一个 ObjC 的对象，分析 string 的内容，可以帮助我们对 Crash 进一步分析，因为 Crash 现场如果能拿到更多的信息，对于定位 Crash 的帮助可能是很大的，这个功能是很赞的。

一个简单的 Case: 当 ObjC 对象野指针时，调用它的任何方法都会 Crash， 而往往野指针问题不太容易快速定位野的对象是什么， 但是我们可以通过分析 x1 的值，也就是最后调用它的 @selector 再结合代码就很容易定位出野掉的对象是谁了。

Binary Images

Binary Images 中记录了进程加载的所有镜像列表， 这块内容是符号化 Crash 日志的关键，符号化的原理就是通过这里的镜像 UUID 来找到对应镜像的符号化文件从而进行对堆栈的符号化工作的。

```
    镜像起始地址     镜像结束地址     镜像名        架构      镜像 uuid          镜像完整路径
    0x1815b8000 - 0x181c73fff libobjc.A.dylib arm64    <5f420cdc6f593721a9cf0464bd87e1a2>
/usr/lib/libobjc.A.dylib
```

读这段内容在 Crash 定位时也是有帮助的，比如：

- 我们可以根据是否有加载越狱的动态库来判断设备是否越狱。
- 有些越狱设备会更改 iOS 系统版本号，也可以通过 uuid 来判断是否是有此种行为。

Crash 分析方法

在了解了上面基本知识后，我们已经具备一定的定位 Crash 的能力了。然而光有理论知识，也还要有实战工具。

符号化

关于堆栈符号化的文章可以说是很多了，但是既然叫攻略，这里还是讲一下吧。

- **Xcode 自动符号化**

  苹果收集的日志，Xcode 会自动帮我们符号化，如果你没有发布包，比如是别人电脑打包的发布包，或者是一些平台上打的包，只需要你把 xcarchive 拷贝到 $HOME/Library/Developer/Xcode/Archives 目录下之后，Xcode 就可以自动帮你符号化了。

- **手动符号化之 symbolicatecrash**

  Xcode 自带了一个命令行工具 symbolicatecrash ，在 /Applications/Xcode.app/Contents/SharedFrameworks/DVTFoundation.framework/Versions/A/Resources/symbolicatecrash ，这个工具可以帮助我们将整个 Crash 日志符号化：

```
    # 这两行可以写到 ~/.bash_profile 里面，这样不用每次敲了
    export DEVELOPER_DIR=/Applications/Xcode.app/Contents/Developer/
    alias
symbolicatecrash="/Applications/Xcode.app/Contents/SharedFrameworks/DVTFoundation.framework/Versions/A/Resources/symbolicatecrash"

    # 执行符号化
    symbolicatecrash [原始 CrashLog 路径] [dSYM 文件路径或待符号的.app 路径] > [导出符号化的文件路径]
```

使用 symbolicatecrash ，如果要符号化系统库的堆栈的话，需要有对应系统库的符号表放到 $HOME/Library/Developer/Xcode/iOS DeviceSupport 目录下面。符号表的获取可以通过插入对应系统版本的手机，或者从别人那拷贝获取。

- **atos 单行符号化**

  在了解了 3.3 中的堆栈的信息后，我们也可以使用 atos 命令来对 单行或多行 的堆栈进行符号化操作。

使用方法：

```
atos -o [镜像的 DWAF 文件地址] -l [镜像的起始地址] [堆栈内存地址 1] [堆栈内存地址 2] ...
```

汇编定位法

因为 iOS 上的 Native 代码都会被编译为机器码，且 Crash 堆栈中的很多信息其实是二进制上的内容，哪怕我们符号化了，但是经过编译器的翻译，有时候也无法通过符号化之后匹配到的代码行来定位最精确的原因。因为一行源代码可能包含很多逻辑而被编译为大段汇编，或者编译优化将多行代码合并优化等操作。

所以，有时候 Crash 定位就需要我们进阶版的 汇编定位法 。当然要学习汇编定位法可能需要一定的基础知识，比如看懂一些基础的汇编指令，可以通过学习和练习来提高，推荐一下一篇旧文: https://blog.cnbluebox.com/blog/2017/07/24/arm64-start/

汇编定位法，还需要一个反汇编的工具。

- 推荐使用 Hopper Disassembler 工具，这是个收费工具。
- 将二进制文件拖进工具就可以了。
- 也可以用 mac 自带的 otool 命令。
- otool -tV 二进制文件。
- Xcode 本身自带汇编调试。
- Xcode -> Debug -> Debug Workflow -> Always Show Disassmbly。

Tips: 二进制就是我们工程编译的可执行文件、动态库。系统库的二进制可以在 $HOME/Library/Developer/Xcode/iOS DeviceSupport/ 中找到。dylib 在 Symbols/usr/lib 目录 ，动态库在 Symbols/System/Library/Frameworks 目录。

工具准备好了之后，怎么定位到具体位置呢，还是回到 3.3 节中的图，“代码在镜像中的偏移”我们就根据这个信息进行查找就可以找到对应的汇编行了。

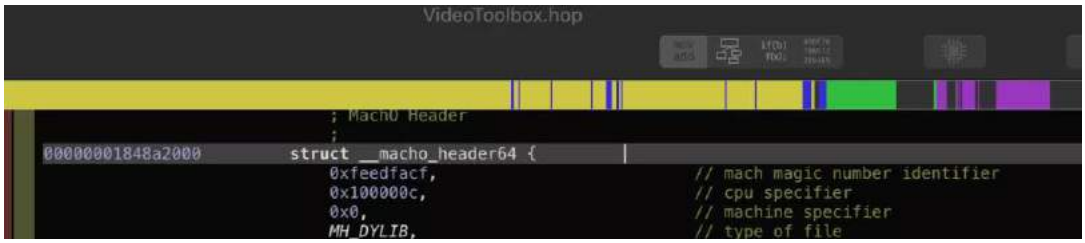
符号化的堆栈，无法直接看到"代码在镜像中的偏移"，我们可以自己减一下就行了。

一个例子：

- Crash 堆栈

- 找到 VideoToolbox 的镜像起始地址

- 找到 VideoToolbox 的镜像，拖到 Hopper



- 算出"代码偏移"

因为 VideoToolbox 是系统库，放在 dyld_shared_cache 里面的，所以它镜像的文件地址也不是 0x0， 所以这里计算时候要加上 0x1848a2000，我们自己打的动态库一般不存在这个。

```
0x1b1700618 - 0x1b16a2000 + 0x1848a2000 =

0x184900618
```

- 找到对应的代码行

注意，这里 0x184900618 虽然是 orr 这一行，但其实是 0x184900614 这一行，因为这是 iOS 堆栈采集的原理所决定的（取 lr），除了 frame 0 的堆栈地址是最后崩溃的地址，frame 序号 大于 0 的地址都是实际地址的下一行。

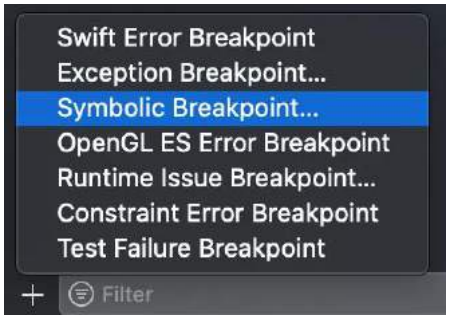
现场勘探法

这是基于汇编法的基础上的一种分析方法。会了汇编分析之后，可能发现定位到了具体的汇编行，有时候问题还是较难分析，因为我们只拿到了代码信息，而运行时的各种状态都是丢失的。

现场勘探法，就是我们使用 Xcode 调试应用，断点到 Crash 地方附近，哪怕是我们线下无法复现的 Crash， 我们也可以到现场去看下正常情况的 栈、寄存器是怎么样的，再对比 Crash 日志中的信息，就可以推断出哪里出了问题。

打断点的方法：

- **符号断点**



- **地址断点**

1. 通过 image list 读取加载的镜像，获取镜像的起始地址，然后就可以算出代码地址。

2. 使用 br set -a 0x184900618 这样的指令来设置地址断点。

小结

本文主要从 Crash 日志渠道、Crash 捕获的原理、看懂 Crash 日志、Crash 分析方法等角度向大家普及下 Crash 原理及分析的思路。

其中讲到的知识点有：Mach 异常、UNIX 信号、常见 Crash 错误码、Mach-O、汇编等。

Crash 定位的过程是不断追溯 Crash 现场发生了什么的过程，通过对 Crash 日志的原理和内容的深入了解，可以帮助我们更快更好的定位应用崩溃问题。

然而现实中可能还是会有些疑难的 Crash 不易定位，是因为 Crash 日志其实也并未保存了 Crash 现场最全的信息，更进一步的优化就是丰富 Crash 日志的信息，让我们获取更多的 Crash 现场信息， 比如 KSCrash 这个框架在这块做了一些努力。这些开源项目的代码是值得学习的。

参考&引用&拓展阅读

总线错误: https://zh.wikipedia.org/wiki/%E6%80%BB%E7%BA%BF%E9%94%99%E8%AF%AF
书籍:《深入解析 Mach OS X & iOS 操作系统》
崩溃捕获系统的原理: https://junyixie.github.io/2019/09/28/CrashMonitorSystem/
分析 iOS Crash 文件: https://developer.aliyun.com/article/8854
iOS 开发同学的 arm64 入门: https://blog.cnbluebox.com/blog/2017/07/24/arm64-start/
SIGSEGV 和 SIGBUS: https://www.cnblogs.com/charlesblc/p/6262783.html?spm=ata.13261165.0.0.26cd7529Cecpmj
iOS 调试进阶： https://zhuanlan.zhihu.com/c_142064221
KSCrash： https://github.com/kstenerud/KSCrash
PLCrashReporter: https://github.com/microsoft/plcrashreporter

手淘图片库新特性解析

作者 | 云移
出品 | 淘系技术

随着手淘拉新和用户体量的增加，CDN 图片资源的访问量也随之增加。我们知道访问量增加，会带来带宽的增加，服务器成本也随之增加。同时新增的用户，喜好各不相同，为了满足不同用户的喜好，商家会修改主图样式来吸引消费者。如何缓解用户增长带来的带宽压力和满足商家和用户的多样化需求呢？针对这两个问题，图片库和图片空间同学对客户端和服务端做了相应的改造来解决这些问题，主要的做法是 H5HEIC 化以及图片库磁盘缓存标准化。

HEIC 的应用

HEIC 是什么

HEIF 是一种图像容器格式，它所生成的图像文件相对较小，且图像质量也高于较早的 JPEG 标准。HEIF 这种新的图像格式基于高效视频压缩格式（也称为 HEVC 或 H.265），它通过使用更先进的压缩算法来实现图片的压缩存储。

- 感兴趣的同学可以看下 https://nokiatech.github.io/heif/examples.html。
- H.265 一般是用在视频上的一种编码标准，这个标准会有不同的实现。
- 引用 https://www.sysgeek.cn/heif/。

手淘 NativeHEIC 化

从 2018 年开始手淘的 native 页面已经全面支持 HEIC,为 CDN 侧节省了大量流量，优化了用户的图片体验。native 页面可以 HEIC 化需要满足两个条件：

1. 业务的图片是存放在手淘图片空间，才能够使用图片空间的图片处理能力。
2. 业务需要使用图片库的适配库来对原始 url 进行适配，然后再借助图片库的能力完成图片的加载。

目前从统计来看淘内的图片 40%已经使用了 HEIC，为什么只有 40%的量？一方面是因为手淘部分业务的图片不是放在图片空间做处理，比如有的业务使用 TFS 来做存储，另一方面是手淘使用的 H5 内核目前无法解码 HEIC 图片。说到这里需要简单介绍下手淘的图片的业务流程：

以上只是简单的示意，大体展现图片如何从商家流动到用户的，主要的中转便是图片空间它可以对图片进行处理和转化。用户使用手淘客户端访问页面时，业务会将商品图片 url 传递给图片库，图片库会对传入的 url 做处理，按照图片空间的规则拼接上 HEIC 后缀，这样当 url 到达 CDN 时，如果有 HEIC 图片缓存直接返回，如果没有会请求图片空间的服务将原始图片转码成 heic 图片。以上是 native 的流程，我们还忽略了另一个流量来源 H5，目前手淘 H5 容器无法使用 HEIC 解码器，所以没有进行 HEIC 化。

H5 页面 HEIC 化

为什么今年可以做 H5 页面 HEIC 化了呢？主要是以下两个方面：

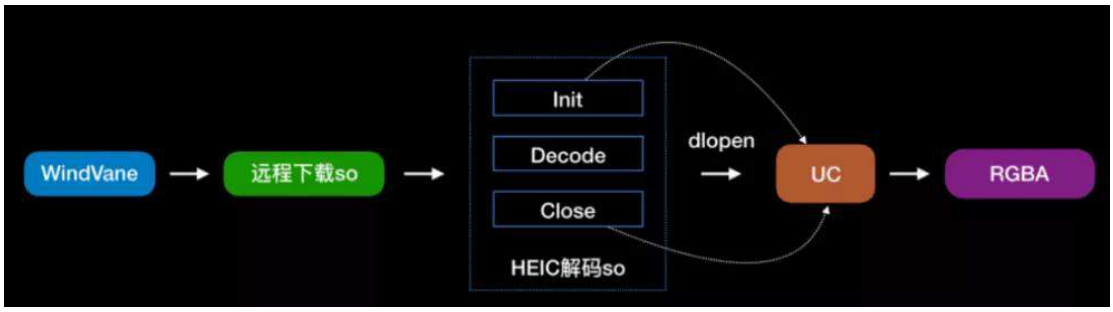
1. 内核提供了外接三方解码器的能力
2. 手淘支持远程下载 so

有同学可能会好奇，之前提到的图片空间是什么？图片空间可以看作是提供强大图片处理能力的服务。从上一小节的流程图可以看出 CDN 后面是图片空间，这里简单介绍下图片空间和 CDN 之间的关系。当有图片请求访问到 CDN 时，CDN 会查找当前缓存中有没有这个请

求需要的图片，如果有直接从 CDN 中返回，如果没有 CDN 会去请求图片空间处理原图给出符合要求的图片，然后再返回给 CDN，同时 CDN 会对这张图片做缓存，当下次有相同的请求过来时就直接返回缓存的图片，不再回源到图片空间，缩短请求响应时间。

首先介绍下 UC 内核外接三方解码器能力，这个能力是如何实现的呢？简单来说就是提供实现 UC 规定接口的 so 文件(保证规定的函数符号会被正确导出)，UC 内核会使用 dlopen 打开 so，同时获取接口函数的函数指针，当需要解码时直接使用函数指针直接调用即可。下面是大体的调用流程以及相关的接口定义：

简短介绍下上图蓝框的流程：



1. Init:初始化图片解码器，并将相应的数据结构透传到 UC。
2. Decode:UC 会透传相应的数据到解码器，解码器解码完成后将相应的 RGBA 数据给到 UC，UC 最终将解码后的数据给到渲染层。
3. Close：回收初始化过程中创建的系统资源。

之前从事过 PC 开发的同学可能会熟悉这这种做法，PC 插件化大多是利用 dll(可以理解为 Linux 下的 so)来实现，这在 PC 时代是常用做法，Python 调用 C++/C 代码也可以通过封装为 so 进行调用。针对 UC 提供的这种能力，图片库对原有的 HEIC 解码器做了封装，实现了上面的接口规范，提供给 UC 调用，这样 H5 解码 HEIC 的问题就解决掉了。

真的这么顺利么？答案是 No。手淘目前对 so 集成时的大小有限制，刚好 HEIC 的 32 位和 64 位 so 加起来将近 4.5MB，这已经是严重超标了，是不可以接受的。

那该如何解决呢？架构组提供了远程加载 so 组件（上图中的远程下载），可以将 so 放在远程，不需要打进 apk 包，当需要的时候从远程下载，同时 HEIC 解码 so 是独立的没有依赖，直接 dlopen 打开就可以使用。这样我们的解码功能就顺利完成了，剩下的工作就需要 Windvane 同学添加相应的降级逻辑和监控逻辑提供必要的稳定性保障，到这里这个 H5 支持 HEIC 功能就完成了。

说了这么多从 native 支持 HEIC 到 H5 支持 HEIC，收益是什么呢？有两个方面，一个是端上的收益，一个是服务端收益：

客户端

目前端上的计算能力已不是瓶颈，主要是网络的通信时间，图片体积越小，网络耗时越少，同时网络耗时的减少可以抵消端上解码耗时的增加，总体而言图片的加载性能会提高。同理对于 H5 页面也一样，网络耗时越少，页面性能也会有所提高。

服务端

对于服务端而言，图片的体积更小，占用的带宽也更小，单位时间内处理的请求增加，简单来说就是提高了 QPS。

H5 支持 HEIC 这个功能已经灰度了几个版本，后续会逐步全量，对于 H5 页而而言，页面性能也会得到提升。

我们可以先参考下 native 页面下 HEIC 图片和 WEBP 图片的下载时间和解码时间的对比数据：

Format	网络时间（ms）	解码时间（msHE）
HEIC	141	27.5
WEBP	645	30

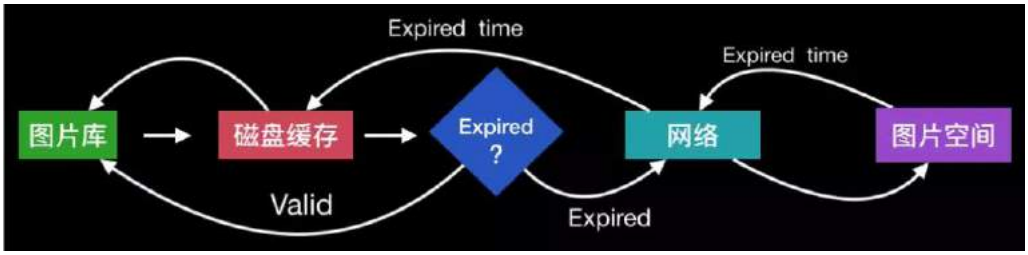
数据说明：样本数在几十万级别，基本可以抹平图片尺寸带来的统计差异。

从上面的数据也可以表明，网络时间的优化可以抵消部分解码时间的增加，有的同学看到数据会有疑问，解码时间并没有增加反而有所优化，这主要是多媒体算法团队对 HEIC 解码做了优化，同时图片空间空间同学对 HEIC 的封装做了优化(优化的后的封装，仍然是符合 HEIF 文件标准的)。

图片库缓存标准化

图片库磁盘缓存标准化

用过 Phenix 图片库或者三方开源图片库的同学都知道，图片库会有三层缓存（内存、磁盘、网络），磁盘缓存一般是持久化的，除非超过磁盘缓存大小被 LRU 淘汰掉，为了满足图片空间的需求，图片库对磁盘缓存做了改造。大体流程如下：



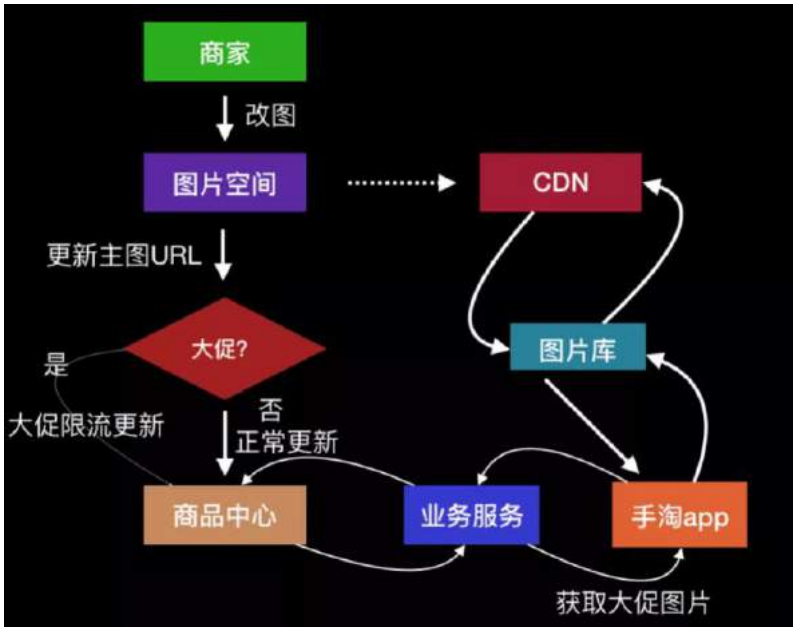
简单来说图片库的磁盘缓存可以指定过期时间，过期后图片请求会打到后端。图片空间会在图片响应头里添加过期字段来控制本次请求到的图片在磁盘缓存中存在的时长。目前实现的缓存控制是在原先 LRU 基础上实现的，即首先会判断这张图片有没有被 LRU 淘汰掉，如果没有被淘汰掉会走到缓存控制逻辑，如果已经被 LRU 淘汰，保持原有逻辑。

有同学可能会疑惑这么做有什么意义的呢？之前 LRU 策略图片更新时间是不确定的，有可能很快就过期了也有可能要等几十小时才能过期，完全取决于用户访问的频次以及用户的操作习惯，过期时间字段可以精确到小时级别，让图片及时过期，请求到最新的图片。

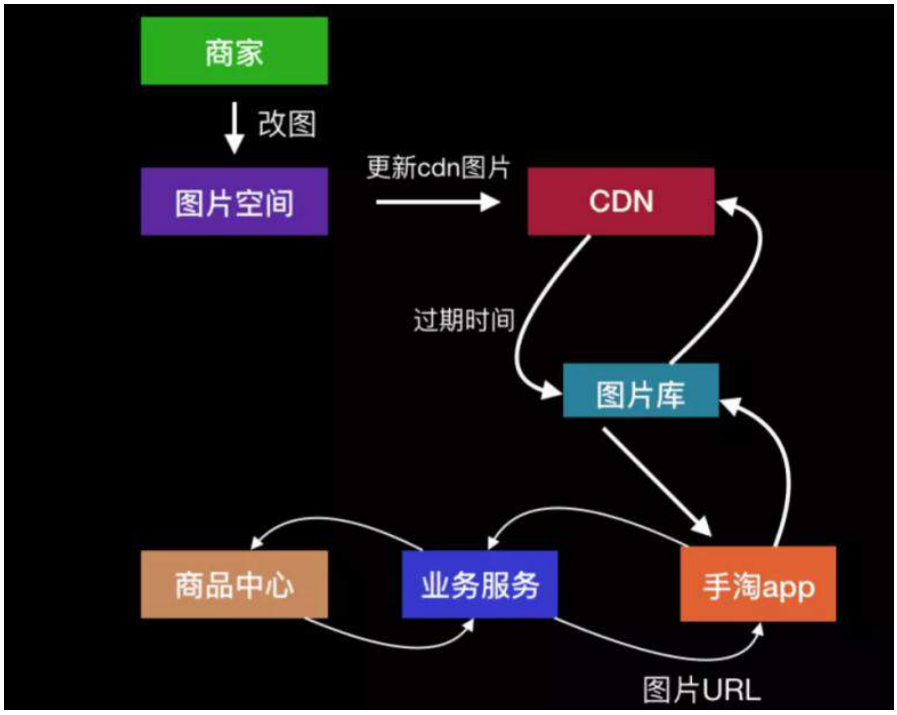
URL 不变更新图片

看到这个标题很多同学可能会有疑惑，URL 不变更新图片是什么意思？简单来说就是访问相同的图片 url 图片的内容可能不同，经过这么一解释有的同学可能更加疑惑，为什么要这么做？

带着这个疑问我们来解释为什么会有这样的功能，促销对手淘来说比较常见，商家也会抓住这个机会尽可能吸引消费者去下单，提高成交量。图片作为手淘商品的主要信息载体，有活动的时候商家为了能够传递更多的信息给消费者，商家会频繁修改商品主图的促销样式和信息来吸引消费者去点击，更多的点击意味这更高的流量，从而带来更高的成交量。商家通过图片空间修改商品主图，图片空间会将修改后的主图 URL 更新到商品服务，但是这给商品服务带来了挑战，手淘承载着成千上万商家，如果促销期间大量商家去更新商品主图 URL，对商品服务带来的压力不可小觑(不止主图更新会调用商品服务，其他业务也会去调用)，因此在大促期间为了保证商品服务的稳定性，会进行必要的限流。大体流程如下图：



这样商家在特定的时间是不能及时修改商品图的，这会导商家和消费者的体验下降，因为主图的促销信息不能及时更新，导致主图显示的优惠信息和实际购买的优惠信息有可能会有偏差。这该如何解决呢？活动期间商家改图后商品主图 URL 不变，图片空间会将修改的图片更新到 CDN，不再去频繁刷商品服务，商品服务的压力也会减小，消费者的体验也得到了保障。修改后的流程如下图：



有同学可能会有疑问，为什么不是提升商品服务的性能呢？以下是我简单的分析，不一定准确：数据库的写性能是有限的，无论再怎么优化再怎么加机器，大量的并发过来总会达到瓶颈。有的同学可能会说，分布式、分库、一致性哈希，或许可以做也可以解决问题，可是大促是短期行为，做这么大的改造耗费大量的人力成本，但是对非大促，这样的改造没有收益，同时也会增加系统的复杂度。

通过这样小成本的改造，将一部分流量转移到 CDN 侧，减轻了商品服务的压力同时也能提升商家和消费者的体验，商家也可以随时更新商品主图。

说了这么多图片库做了什么？一开始介绍的图片库磁盘缓存标准化，大促期间图片空间会在下发图片的 Response 中带上过期时间字段，设置一个合理的过期时间，便于大促期间商品主图的磁盘缓存隔一段时间过期一次，过期后请求新的图片，这样用户看到的图片始终是最新的，虽做不到实时但也可以达到小时级，能够满足业务的诉求。

总结

今年图片库的主要改动有两个方面：H5 支持 HEIC 化以及图片库磁盘缓存标准化，这两个改造从本质上来说是将服务端的优化通过端上整合触达到用户。HEIC 图片 native 端在 2018 年就已经开始支持，但是由于 H5 容器 UC 内核当时没有 HEIC 解码能力，所以 H5 页面没有使用 HEIC，今年随着 UC 对外提供了三方解码插件的功能，利用这个契机将 H5 页面进行 HEIC 化。图片库磁盘缓存标准化。

客户端精彩问答

Q: Node.js 做 Web 后端优势为什么这么大?

作者 | 聚海
出品 | 淘系技术

前几天听别人说 nodejs 在未来的微服务 还有 serverless 上都有很大的优势，除此之外，作为 web 后端优势同样很大，虽然国内 nodjes 在后端并未很火，但是在国外，nodejs 却非常火（看的一篇文章，硅谷很多都用 nodejs 做后端）。
nodejs 这么大的优势，难道就是因为 单线程、高并发、事件循环，前后端语言统一吗?
（我这说的是 nodejs 本质，并不是前端心中被神话的 nodejs，请勿对号入座）

谢邀。我们今年淘宝天猫双十一，就是用 Node.js + Serverless 支撑的。

业务研发提效 38.89%，并保障了双 11 大流量下稳定可靠。

和大家分享下我们的 Node.js 最新实践方案。

依托阿里云 Serverless 函数计算等基础设施，今年（2020）淘系前端在行业导购、会场、营销 v 榜等业务场景中规模化使用了 Node.js Serverless 技术，请求相比去年（2 千 QPS）今年达到新的水平（峰值总计 5 万 QPS），同时将 CPU 水位，相比去年 FaaS 平均水位（20%）提升到（40%）。在业务价值方面，Node.js Serverless 支撑的云端一体研发模式交付平均提效 38.89%。

今年更是依托 Serverless 的便利性和可靠性，帮助双十一特色的淘宝、天猫、飞猪等会场页面快捷地落地 SSR 技术提高页面体验。在运维方面，今年优化大促态和日常态流程，让

开发者在应对更高 QPS 规模时，精力花费降低约 50%。在研发体验方面，打造解决方案体系，降低研发门槛，支持外包快速入场。

本文将介绍我们在淘系双十一，从日常开发到作战的方案与经验，为大家提供一些 Node.js Serverless 的经验参考，也欢迎大家提出宝贵的意见。

探索之路

Midway Serverless Framework

相比传统 Node.js 应用，由于部署模式以及运行环境的不同，我们在原有 Node 框架之上进行了方案升级，提出了 Midway Serverless。

Midway Serverless 是一套面向 Serverless 的解决方案，它包括框架、运行时、工具链、配置规范等几个部分。

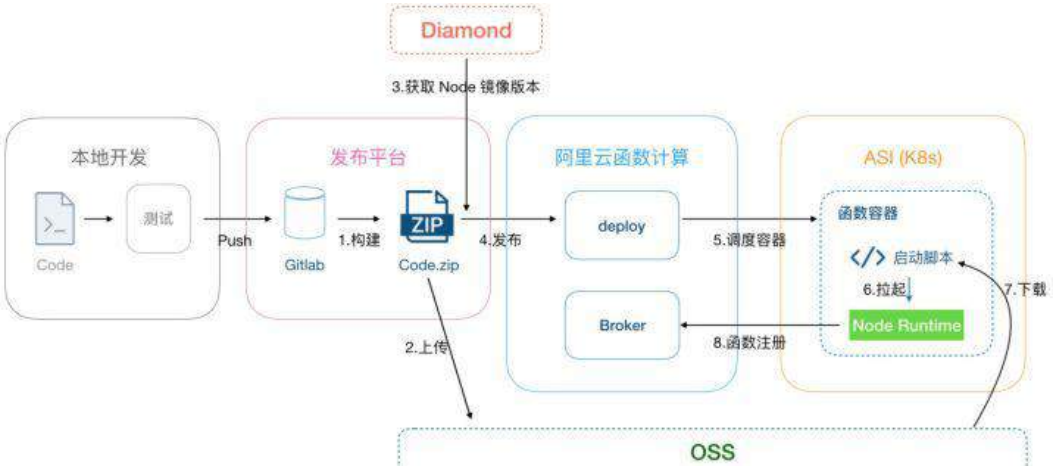
在框架设计过程中，我们主要把握了这几个特点：

- 多平台适配能力,社区不同云平台底座能力适配，提供 Serverless 底座迁移替换能力，防止云平台锁定。
- 传统应用和 Serverless 模式，相互迁移的能力，做功能降级和能力升级。
- 让应用层更易维护和扩展。
- 使用了 Typescript 作为基础语言，定义基础类，便于上层应用开发者定制化。
- 提供应用开发过程中常用的插件以及提供了插件体系。
- 框架默认结合基建能力，提供问题排查和性能优化等基础功能。
- 面向不同业务场景的开放性
- 提供基础上层接口，满足不同开发者基于不同业务场景进行定制化开发。

打造快速试错、高可用的研发模式

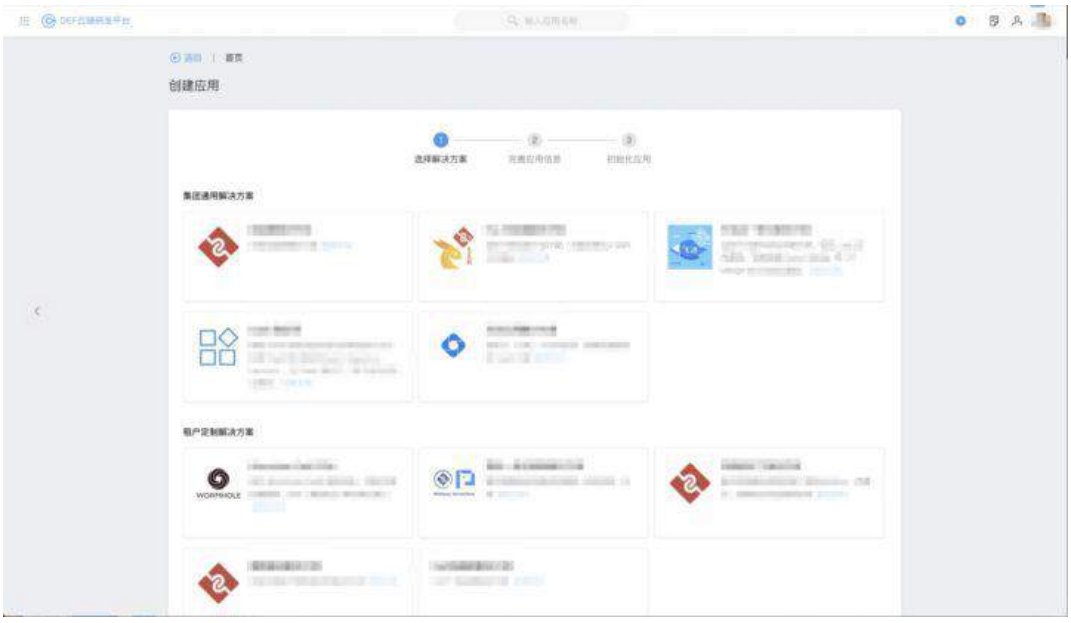
统一云研发平台，从研发到交付构建更敏捷的前台应用开发。

为了提升开发者本地的开发者体验，我们做了 Vscode 插件、本地脚手架来提升开发者的开发体验。发布平台，我们通过将发布链路进行流程化，做代码云构建，最终将热加载的代码打成压缩包。最终函数运行时，我们通过热调度的模式，加速函数启动时间。



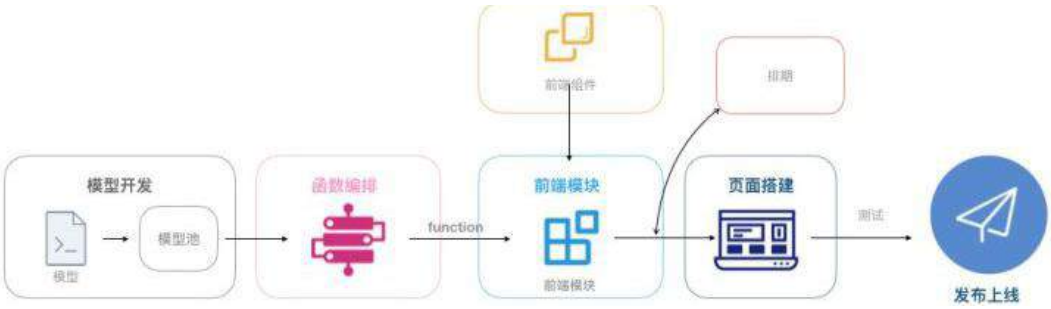
解决方案沉淀

同时我们和业务开发者在过去实践中，沉淀各类解决方案，各部门一方可以定制化自己的租户，同时各部门也可以选择通用解决方案，并选择自己部门的网关进行组合。



模型驱动、函数编排

借助模型驱动，函数编排产出 FaaS 函数。同时结合我们原有的搭建系统，业务同学进行模块的排期，页面的搭建，快速响应淘宝双十一相关的业务需求。



技术实现层面主要分为：

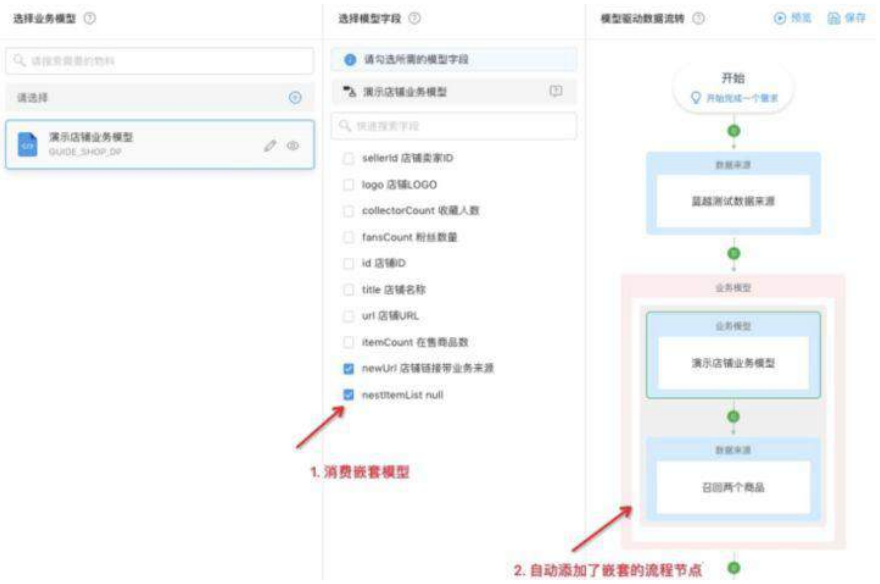
- 数据链路设计
- 操作面板设计
- 代码生成方案

数据链路设计

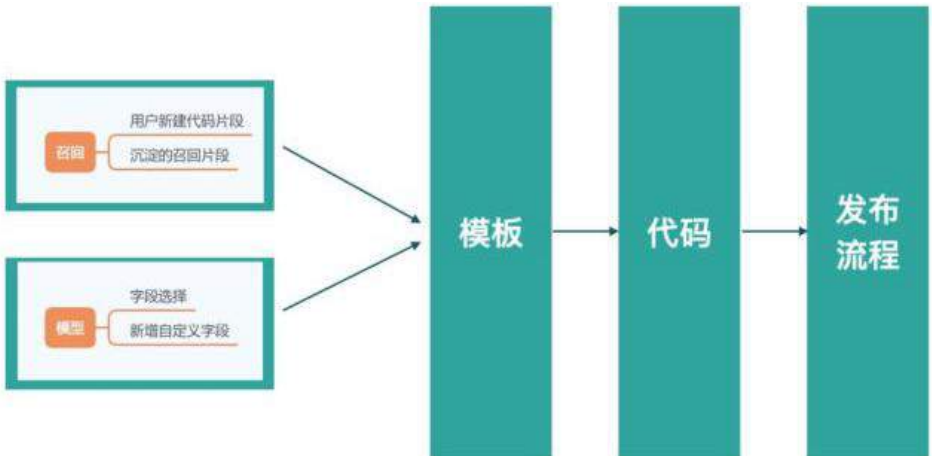
主要设计定义 DSL，对数据来源、模型、字段、出入参、代码片段进行了结构化存储。例如代码片段定义：

```
export enum FragmentTargetType {
  UNKNOWN = 0,
  FIELD = 1,
  DATASOURCE = 2,
  COMPLETE = 3,
  PLUGIN = 4,
}
export interface IFragment {
  id?: number;
  name?: string;
  description?: string;
  code?: string;
  creator?: string;
  targetType?: number;
  codeDetail?: string;
}
```

操作面板设计，主要是函数逻辑编排：



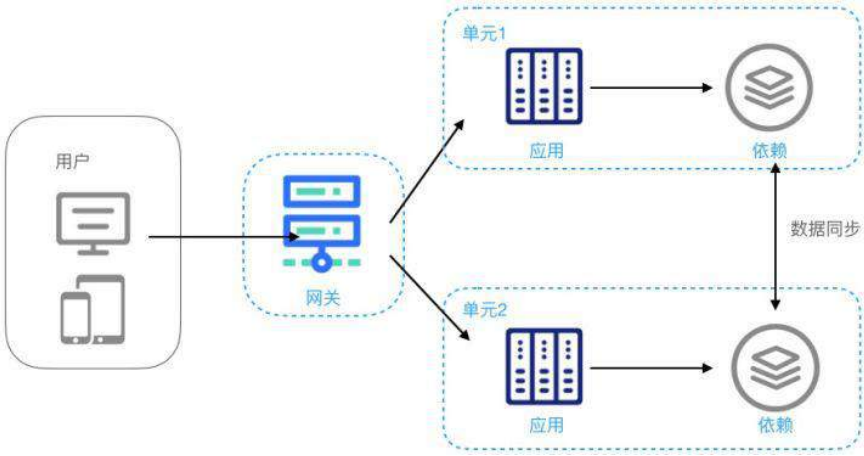
FaaS 代码生成逻辑：



稳定性建设

单元化保障

通过异地容灾，对业务进行多地多机房部署，实现真正意义上的异地多活。



单元化优点：

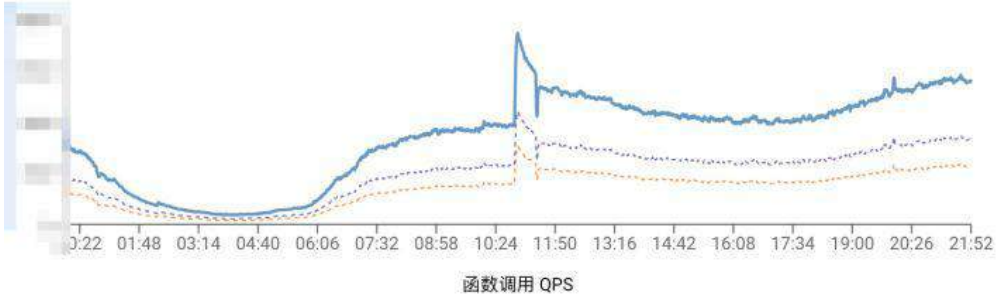
1. 在车辆高速行驶过程中，进行更换轮胎。框架升级，可以在某个单元进行灰度。
2. 当单个单元出现问题时，切换到另外一个单元提供服务。
3. 性能评测，可以做单个单元的性能评估。

监控保障

我们通过在框架 Runtime 层面，接入全链路跟踪系统，方便开发者查看错误的链路情况。

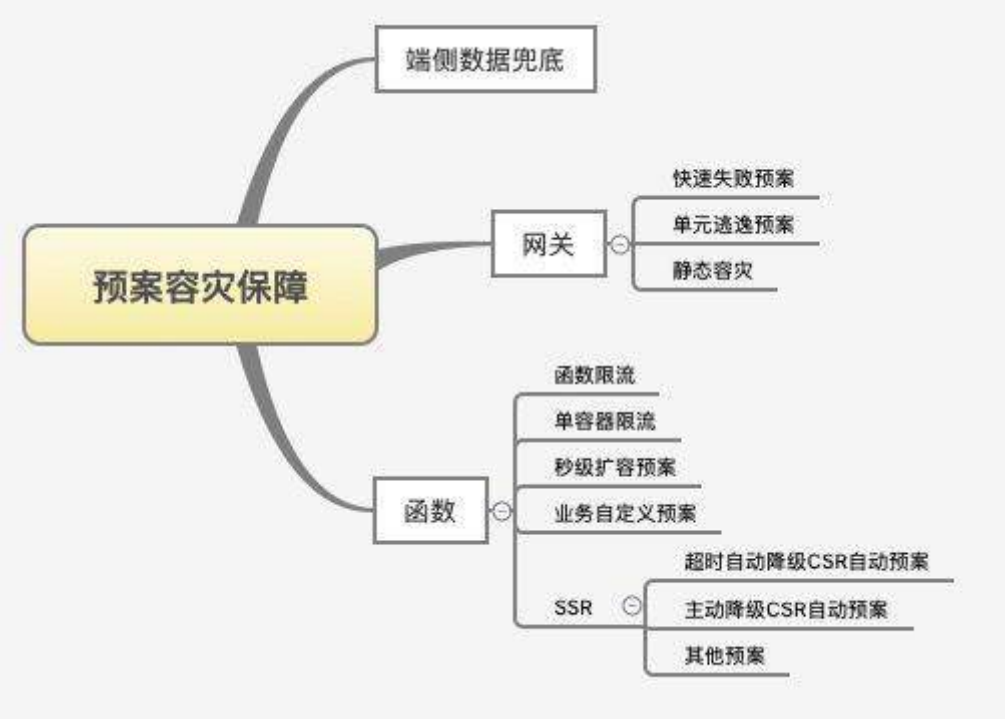
业务同学只需关注自己业务函数的 QPS 和成功率，而且对于成功率的下跌，或者 QPS 的环比下跌都将通过 IM、短信或者电话的方式进行及时报警。

例如下图我们一个函数的日常情况的 QPS 变化情况：



预案保障

为应对各个环节可能存在的未知风险，架构层面，我们通过设置了多种预案来进行面对。



当任何一个环节出现监控报警时，都能作出合理的应对措施。

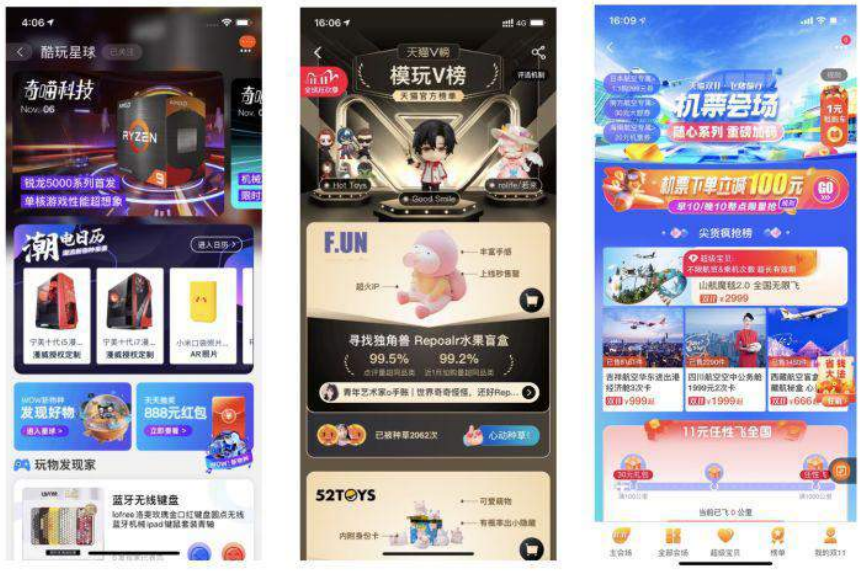
限流保障

根据 2019 年的流量情况，预估今年双十一的规模，我们进行了建站压测、单链路压测、全链路压测等多个环节。根据压测表现，对于流量链路的各个组件，进行限流配置，进行服务的保护。

今年在保障相同 QPS 的情况下，我们对于容器层面的 CPU 消耗情况进行更多的压榨，相比去年提升 1 倍。

业务实践

导购营销云端一体研发



通过 2.2 中的研发模式，让开发者能快速选择通用解决方案，快速响应业务需求，制作出满足业务场景的前端功能。最终运营借助我们前端开发者提供的功能，能玩出花。

会场 SSR - 实现双 11 会场秒开

借助 SSR 服务端渲染 + Serverless 模式，前端开发者通过前后端一起优化的模式，进一步优化用户体验。帮助双 11 分会场大幅提高了页面打开速度，业务转换上提高的受益也有不错体现。

中后台高效研发

中后台一体化研发模式具备特性：

- 前后端一体化开发、上线、运维，高效开发部署上线。
- 无须关注服务器、域名申请等事情，让业务同学只需关注业务。
- 解决长尾应用服务器成本问题，合理把控中后台生命周期。

总结与展望

Serverless 是未来开发的基座，前端利用 Serverless 快速实现业务交付，以往，前端的开发会受到资源的限制，而如今，在创造力的驱动下，Serverless 使得前端不再受到资源成本的制约，同时又能使用 Node.js 快速的迭代业务，其效率远高于传统开发，而成本又相对较低。如今，这不仅仅是未来，也是当下前端在持续投入的事情，Serverless 和多端技术，整个前端都将有一次大的变更和飞越。

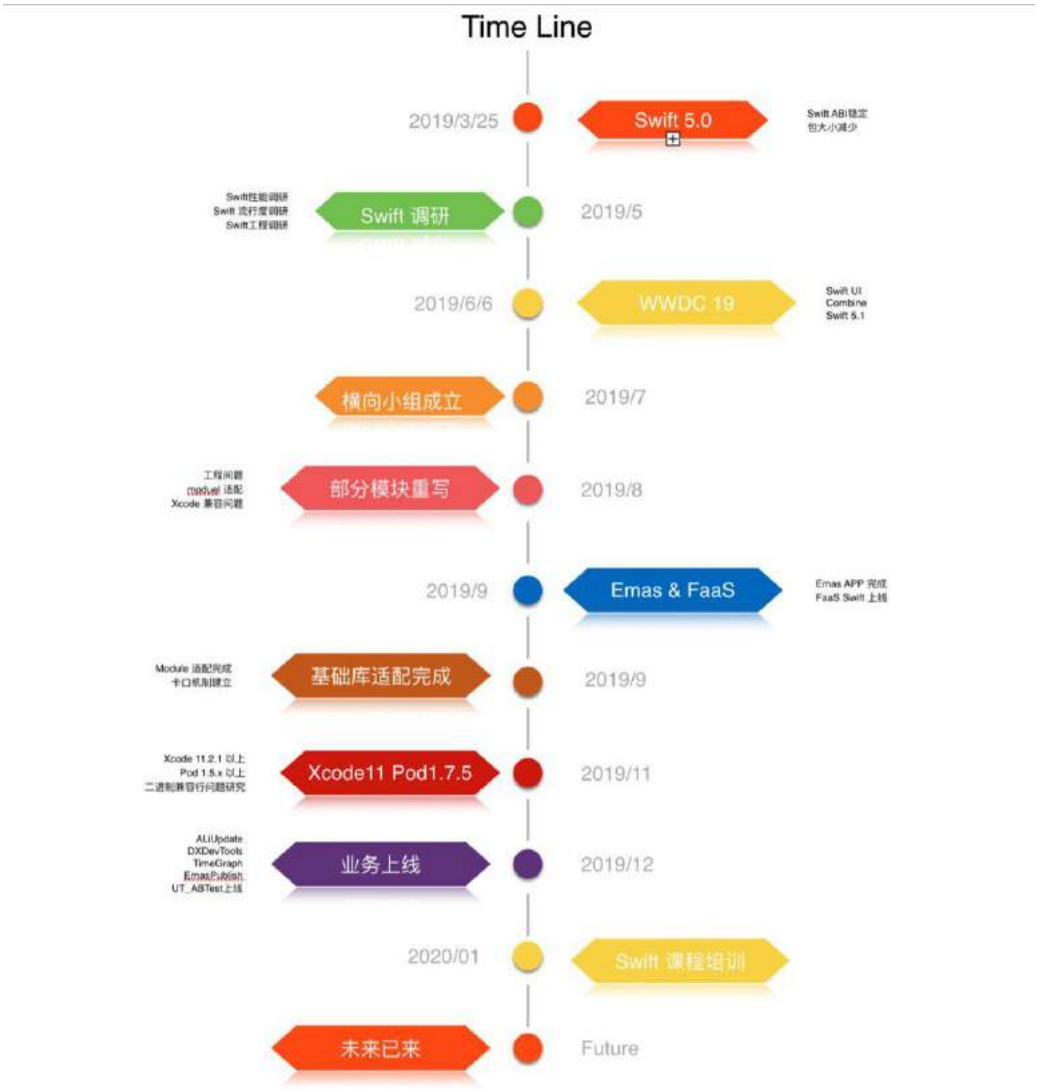
Q: 2020 年了，大家怎么评价 Swift?

作者 | 淘系技术
出品 | 淘系技术

2020 年了，知乎上有关 Swift 的问题不可否认的过时了（还在 14、15、16、17，那时连 Apple 自家 ABI 都还没弄好），我想听听大家现在对 Swift 有怎样的评价。
能否对一下几个方面分享一点想法呢?
1、Swift 在 Linux 上
2、对 Swift4TF 怎么看 和 Swift（及其分支）可能（部分）取代 python 吗?
3、Server-Side Swift，把它投入生产可行吗？将来呢?
4、Swift on Windows 有希望吗?
5、你还总是把 Swift 和 iOS 连在一块想吗？（我反正是用 Swift 写脚本和 cli 比 写 iOS app 都多的那种）。

2020 年了，基于手淘 iOS APP 在 2019 年完成 Swift 语言从调研到基础设施建设再到顺利落地业务，我们可以聊聊在阿里淘系 Swift 语言的应用。

手淘作为一个航母级别的 APP， 组织结构，工程结构，都是普通 APP 难以企及的，在手淘中实践犹如在沼泽地艰难探索，期间和集团内众多 Swift 爱好者，中间件负责人，一起努力探索出一条较为明朗的 Swift 落地指南。



Swift 预研

Swift 语言在 2018 年就已经宣布 ABI 稳定是最重要的目标，虽然早在 Swift 4.x 时代，语法就已基本不变，但受限于手淘是一个航母级 APP， 工程交付以二进制组件化为主，ABI 未稳定的 Swift 只能固定 Swift Toolchain 的版本，加之会有将近 3MB 的包大小压力，彼时都不适合引入 Swift。苹果与 2019 年 3 月 25 日发布了 Swift 5.0 版本，宣布了 ABI 稳定，彼时又让团队的 Swift 燃起了信心，架构组作为新技术的把控团队，决定在百忙之余让团队成员各自分担一些对 Swift 的调研工作。

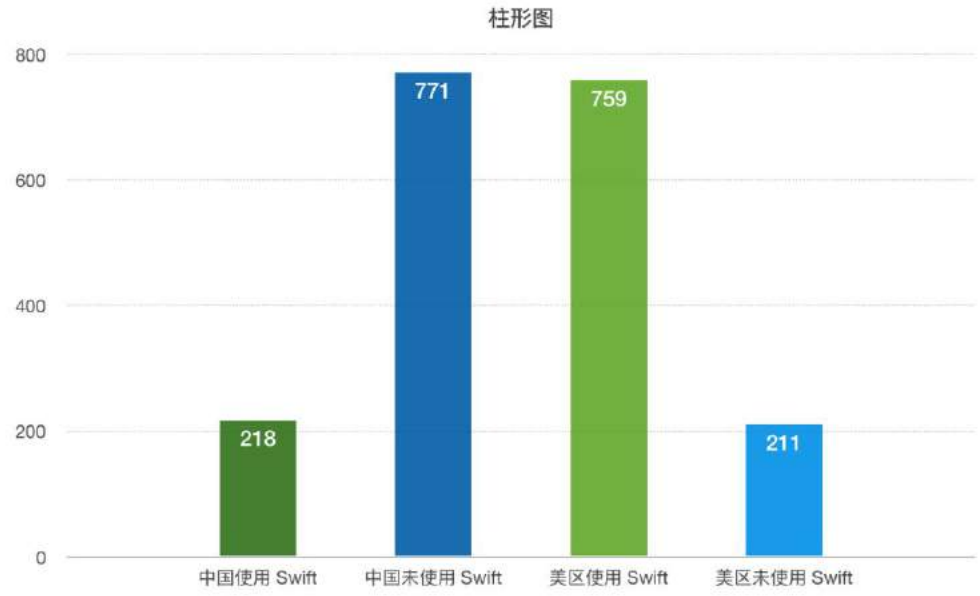
在工作之余，我们完成了调研工作，从中得到两个比较重要的结论。

Swift 流行度

我们通过爬虫分析国内外 APP Store 排行榜 Top1000 的 APP，通过文件扫描分析得到结论。

1. 国内使用 Swift 的 APP 约占比 22%，美区使用 Swift 的 APP 约占比 78%，其中美区剩余没有使用 Swift 的 APP 大部分来自中国地区本地化的产品，如抖音，快手等，可以得出一个结论，国内还是小众的 Swift，在国外已经是现状。
2. Github/Stack Over Flow 社区等 Objective-C 开源库和问题提问已经基本停滞，未来我们在落地新技术，Objective-C 可能已经是最坏的打算，加之 WWDC 17 年以来，苹果不再提供 Objective-C 的示例，组内同学也多次遇见 Objective-C Bug 去社区提问，毫无热度的情况。
3. 苹果在 WWDC19 年发布了 4 个 Pure Swift 框架，无法简单的被 Objective-C 混编。

未来我们极有可能因为苹果的强制推进风格和社区文化的落后产生技术踏空，无法迅速响应业务，甚至无法招聘到会使用 Objective-C 的工程师。



Swift 研发效率

Swift 从诞生之初就是一门先进的语言，主要有三个特性。

1. Safe Swift 从语法层面避免了很多未定义的行为，空值访问，值类型，对工程的稳定性有非常正面的好处，曾经面试一个候选人，简单重写为 Swift 让项目的崩溃率降低30%。
2. Fast，Swift 的语言设计让更多的方法调度通过静态调度完成，语言运行时效率优于 Objective - C。
3. Expressive， 富有表现力的语法特性，让代码量清晰已理解，减少的代码量约有 30%-50%不等，简洁的代码也让一些 Bug 无处藏身，也让工程师能更高效的支撑业务发展，大大提升了研发效率。

优势项	Swift
新特性	元组、闭包、泛型、命名空间
先进性	函数改进、高阶函数，struct、enum更强大灵活，控制流更先进（如defer、区间操作符、switch case任意类型）
安全性	类型安全、禁止隐式转换，编译期容易发现问题，guard、optional保护
包大小	编译后减少10%（strip后）
编码效率	代码更简短，可减少编码量20%~30%
性能	没有动态派发，函数调用更快
跨平台	支持
编程范式	面向协议编程、函数式编程、SwiftUI、Combine

下图是一个业务模块重写后的代码量对比：

其他不利因素

Swift 虽然在 5.0 版本完成了 ABI 稳定，但是在低版本操作系统中 （iOS 12.2 以下）仍旧会有一些不够完美的地方。

1. 在低于 iOS12.2 以下的操作系统会带来约有 3MB 的包大小问题，但幸运的是苹果放开了蜂窝数据网络下 200M 的下载大小。在 iOS13 上甚至可以超过 200MB。
2. 在低于 iOS12.2 以下的操作系统会有 100-200ms 不等的启动耗时增加，但在团队同学的努力下上线了二进制重排，启动性能大幅度上升 具体分析请看 手淘架构组最新实践 | iOS 基于静态库插桩的二进制重排启动优化。

集团横向组织成立

在充分的说明了 Swift 的优势和未来的情况下，我们决定不再停留与各种了解，准备下决心落地 Swift，毕竟所有的 iOSer 都知道，Swift 是未来，但是喊了这么多年的口号，总需要有组织的向前推进，只有自己做了并且落地了才有未来。

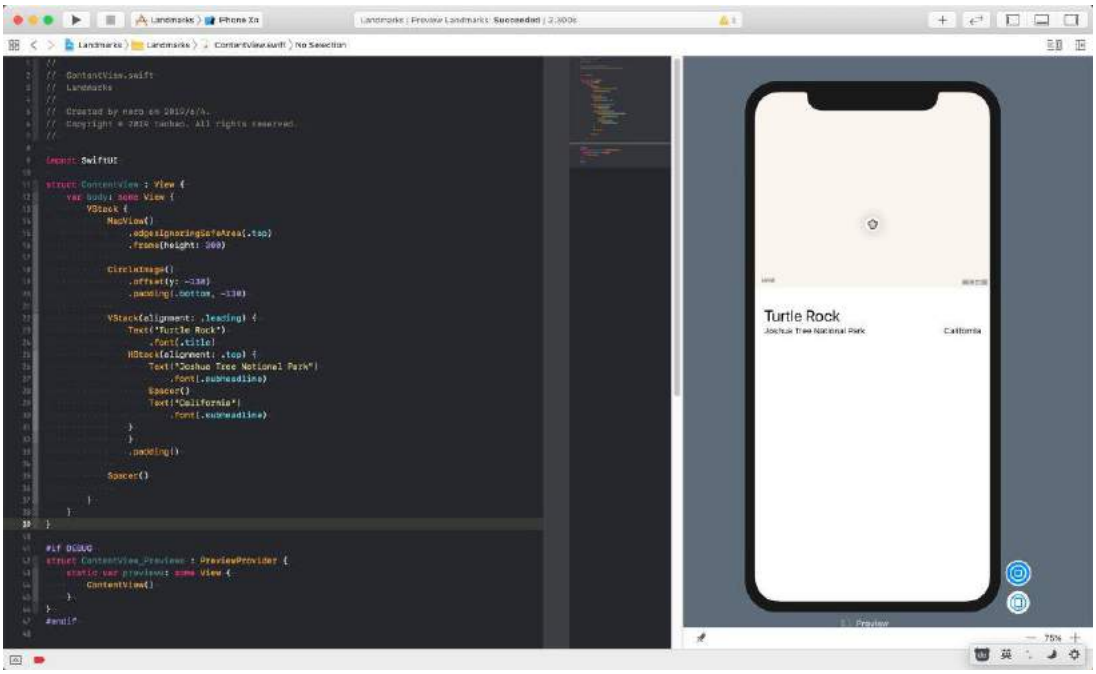
在基础会议上讨论了一些 关于 Swift 落地的计划和核心任务。如 Swift 基础建设，基础库适配，Xcode 升级， 以及新框架 SwiftUI， Combine， 容器化架构等。

并逐步拆解任务，落地下去。

SwiftUI & GAIA FaaS

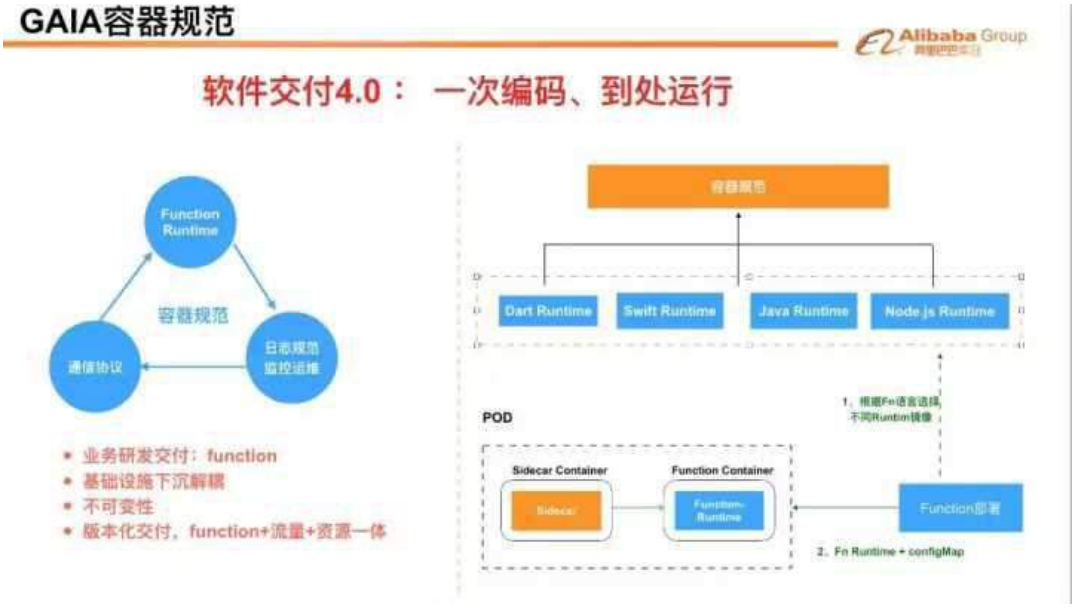
在了解 Swift 的同时恰逢 WWDC19， 苹果与 WWDC 推出了跨 Apple 平台下一代声明式的 UI 布局框架 “SwiftUI”。作者第一时间了解了 SwiftUI 的方方面面的特性，简洁清晰的 DSL， 表现力十足的数据流管理，能减少传统 Cocoa 命令式编程 80% 的代码，对研发效率有极大提升。

以下是一个简单的代码展示



同时我们日常工作中一直有值班看数据大盘的诉求，有时甚至需要在外出的时候查看，之前我们不得不携带工作电脑，迫切需要一款移动值班的 APP，刚好出来的 SwiftUI 新技术可以帮助我们快速落地原型。

但此时没有后端支持，我们想到了 GAIA 平台， GAIA 平台是一款 FaaS 平台的风格，且 Swift 是一门跨平台语言，早在很早 IBM 就曾有 Serverless 服务，于是我们使用 Swift 遵守 GAIA 规范实现了一套 FaaS 服务，作为我们的数据大盘后端支撑。



之前已经编写了多篇文章分析了 SwiftUI 和 FaaS 的核心实现，这里就不再赘述，更多信息详见。（点击标题即可阅读）

新晋网红 SwiftUI——淘宝带你初体验

系列文章深度解读|SwiftUI 背后那些事儿

历时五天用 SwiftUI 做了一款 APP，阿里工程师如何做的?

Swift 在 GAIA 平台云端一体化的探索

SwiftUI 的未来

随着前面的调研，使用分析等，虽然 SwiftUI 在初期使用 UIKit/APPKit 在性能上略显不足于 Flutter，但是 SwiftUI 的 DSL 和其他特性比 Flutter 简介不少，相信苹果在未来的版本种会持续优化，现在的技术储备让我们可以轻松应对未来 SwiftUI 的落地。

业务模块重写

手淘在多年的历史迭代中，或多或少遗留了一些历史包袱，如 Cocoapods 不规范使用，头文件管理不规范，工程模版老旧。为了发现是否存在工程问题会导致 Swift 无法落地业务，我们尝试编写一个代码量适中的 SDK，来验证 Swift 的研发效率等提升。完成 SDK 初步重写后，我们发现了现有工程中遗留了大量的问题，导致现有的项目完全无法使用 Swift。

主要存在以下问题：

1. 头文件不规范无法混编。
2. 不支持 Module 无法被 Swift 导入。
3. 工具链老旧支持 Swift 源码开发调试。
4. Swift 5.0 不是 Module Stability，对 Xcode 版本有强制限制。

适配基础库

为了解决 Swift 落地工程中的问题，我们结合集团 Swift 横向组织的力量，号召 Swift 爱好者，调研适配技术，编写适配文档，提供自动化工具，梳理出大量的 Swift 适配文档，推动中间件负责的同学，在不影响业务业务迭代的同时，顺手改掉历史包袱，满足 Swift 业务。

截至目前阶段，我们适配了约 100+ 基础库模块， 开启了强制打包卡口，长期有效的治理历史包袱，部分模块在治理历史包袱时，对自己的打包时常，和研发效率都有不小的提升。

适配 Xcode + Cocoapods

手淘的开发模式不同于其他中小型 APP，我们的开发团队复杂，交付方式主要由打包 SDK 二进制集成为主，但 Swift 5.1 才满足了 Module Stability，苹果于 9 月份才正式推出 Swift 5.1，我们开始着手升级 Xcode11。

对于手淘这样的 APP， 每年升级 Xcode 也会带来大量的 Break Change Future，我们通过查看官方文档，提供适配方案，推动业务方修改了大量 Bug 适应了大量 Futuer，我们与 11 月初完成了 Xcode11 的适配工作。

同时由于 Cocoapods 1.5.x 才支持 Swift 静态库，我们将 Cocoapods 升级到了 1.7.5，升级还带来不少 SDK 打包效率巨大的提升。

-	Xcode 10 + pod 1.2.0	Xcode 11 + pod 1.7.5
Swift 5.1	无法支持Swift环境	落地多个业务，且向前兼容
DarkMode	Xcode10 无法使用DarkModeAPI	支撑业务适配 DarkMode 已上线
打包时间最短时间	8min左右	5 分 30s
打包时间平均时间 (Debug)	14.5	8 min
打包时间平均时间 (Release)	14.2	9.7 min
源码环境构建 （初次 pod install）	3分钟	1分钟
源码环境（增量构建）	1分钟	20s

Swift 二进制不兼容研究

Swift 是一门相对静态的语言，不同于 Objective-C 大部分通过 msg_send 动态调度，静态语言需要访问到其他依赖 SDK 的数据结构的静态布局信息，且手淘的研发方式是二进制依赖交付，容易造成下层 SDK 升级未通知上层 SDK 升级重新编译，带来的二进制不兼容问题很有可能留到线上在爆发，Swift 5.1 同时带来的 Library evoluation 功能帮助我们避免二进制不兼容问题发生。

以下是一份会打破二进制不兼容的 API 变动列表:

Change	Normal struct	@frozen struct
Adding fields	Allowed	Affects ABI
Reordering fields	Allowed	Affects ABI
Removing ABI-public fields	Affects ABI	Affects ABI
Removing non-ABI-public fields	Allowed	Affects ABI
Changing the type of an ABI-public field	Affects ABI	Affects ABI
Changing the type of a non-ABI-public field	Allowed	Affects ABI
Changing a stored instance property to computed	Allowed	Affects ABI
Changing a computed instance property to stored	Allowed	Affects ABI
Changing the access of a non-ABI-public field	Allowed	Allowed
Marking an internal field as @usableFromInline	Allowed	Allowed
Changing an internal ABI-public field to be public	Allowed	Allowed

业务完成

完成以上基本工作后，我们把之前重写的 SDK 通过回归测试后在手淘新版本正式上线，上线后的业务调用量等同于手淘 UV，且 0 Crash， 对开发同学造成了极大的鼓舞，多位同学主动尝试使用 Swift 落地业务。

未来已来

手淘在完成 Swift 落地实践中产出了大量的适配文档，工具，教程，方法论，在 2019 年初我们指定的目标和方向已经基本完成，我们做到了！！！在 Swift 诞生这 6 年时间里，各位开发者一直在声称 Swift 是未来，但是在我们手淘架构组的努力下，Swift 不再是未来，而是此刻。

在接下来的计划里，我们会提供一种渐进式的 SDK 迁移方案，逐步将老旧 SDK 迁移到 Swift 中去，Swift 的高效研发效率和安全性将更易支撑业务的快速发展。

本文参考:

手淘架构组最新实践 | iOS 基于静态库插桩的二进制重排启动优化
新晋网红 SwiftUI——淘宝带你初体验
系列文章深度解读|SwiftUI 背后那些事儿
历时五天用 SwiftUI 做了一款 APP，阿里工程师如何做的?
Swift 在 GAIA 平台云端一体化的探索
ranking-programming-languages-by-github-users
Library-Evoluation
Developer SwiftUI

测试篇

这一年对所有技术人意义非凡，砥砺前行、突破进取，我们看到了挑战中的机遇，依托大数据、智能化的背景，质量边界进一步拓展。每一位用户的极致体验都是我们追求的目标，感恩所有的用户，我们还可以做的更多、更好。未来质量体系建设一如既往关注体验，这也是淘系每一位技术人的使命！

——阿里巴巴资深测试技术专家 青灵

技术经典总结

智能化测试技术探索与实践

作者 | 金晖(定源)
出品 | 淘系技术

谢涛教授简介

谢涛教授长期从事软件工程的研究，在软件测试、软件解析学、软件安全、智能软件工程、教育软件工程等方面做出了很多开创性的工作及突出的研究成果，因此先后当选美国计算机协会杰出科学家（ACM Distinguished Scientist），电气电子工程师学会会士（IEEE Fellow）和美国科学促进会会士（AAAS Fellow）。

谢涛教授对软件工程界做出杰出贡献和服务，获 2020 年度 IEEE 计算机协会软件工程技术委员会杰出服务奖（为其年度四大奖项之一）。

他于 2007 年与微软合作研发自动测试数据产生工具 Pex，他设计和实现的 Fitnex 路径搜索技术集成到 Pex 里，成为其三大核心技术之一。自 2015 起，Pex 被微软发布为 Visual Studio 企业版的 IntelliTest 主打功能，使广大 Visual Studio 用户受益。基于 Pex，他还设计了一款新颖的编程教育游戏 Coding Duels，拥有几百万的注册用户，并被微软研究院主办的编程之美大赛和 Imagine Cup 大赛采用作为其主要竞赛形式之一。

他于 2010 年与微软亚洲研究院合作，开创全新领域-软件解析学(Software Analytics)，引领软件工程研究者在该领域开展研究。

他于 2017 年与复旦大学合作，研发应对产业界广泛采用微服务架构后所遇到的开发和运维挑战的解决方案，并自主研发了第一个中大型规模的开源微服务系统 TrainTicket，赋能学术界和产业界联合开展微服务相关的技术攻关。

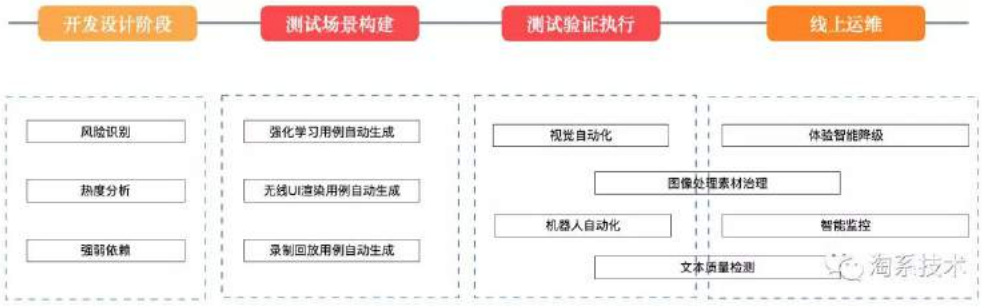
近期谢涛教授将围绕如何通过 AI 的方式提升端到端功能自动化测试的能力和价值与淘系技术部技术质量团队开展访问研究，共同探索手工探索式测试、Monkey 等方式，成本和范围无法保障；目前端到端功能自动化测试工具"逐步执行、单点验证"为主的传统模式成本高、难以快速应用和推广等难题。

本次淘系技术部邀请到谢涛教授就**"软件质量的下一个五年：智能化测试技术探索与实践"**为主题，面向阿里经济体技术同学进行分享。

智能探索，极测 we 来

淘系技术质量掌门人崔婧（青灵），首先介绍了淘系技术质量（极测）团队的发展历程：**淘系从服务化—平台化—智能化的质量体系演进。**

以及淘系在智能化测试的诸多实践和探索模式变化：



淘系探索模式 → 淘系探索+软工领域知名学者、实验室团队共研模式

最后对淘系智能化测试未来探索做了展望：

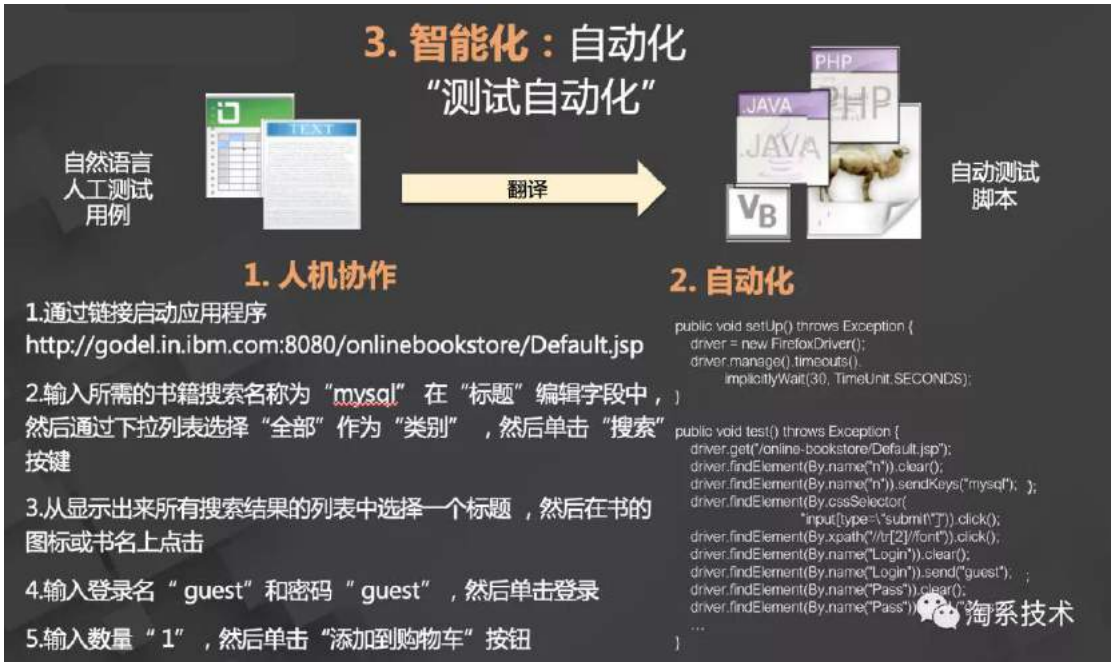
智能化软件质量创新：机会与挑战

在崔婧（青灵）的隆重介绍下，谢涛教授登场，从软件质量的全流程出发，分享了研发测试流程各个环节的智能化创新成果。

谢涛教授首先介绍了智能测试的分层：

并重点从测试用例执行、测试数据生成、测试预言生成三个方面介绍了智能化测试层次的实践：

首先是测试用例执行的智能化：



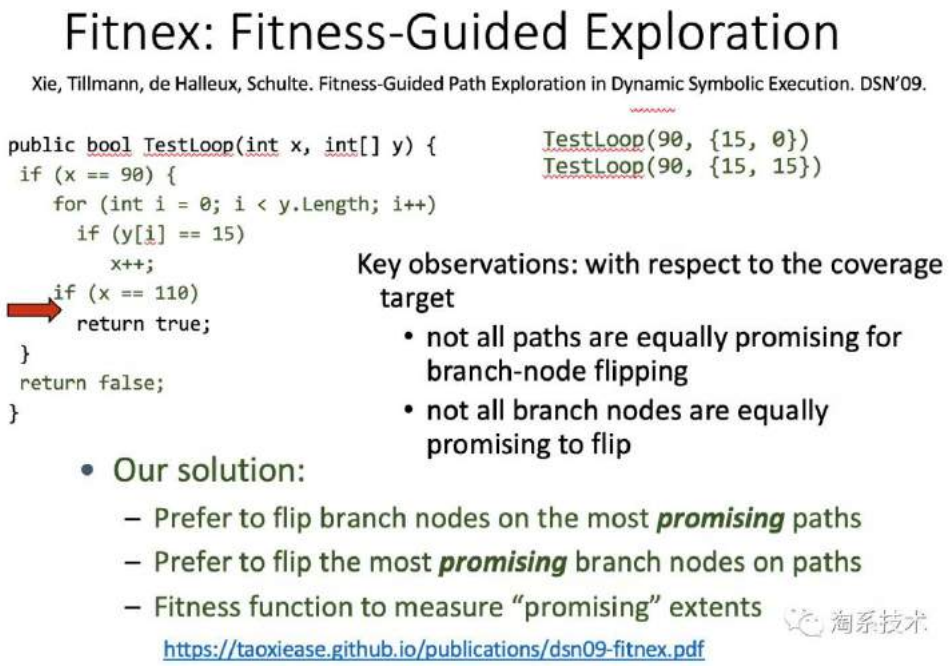
Test Step	Preliminary Segments	Candidate Tuples
Enter the intended book search name as MySQL at the Title Edit field **/and/** select Category as All **/by/** Drop down list **/and/** then Click Search Button	**Enter** the intended book search name as MySQL at the Title Edit field	1 <Enter, Name, Book> 2 <Enter, Name, MySQL> 3 <Enter, Name, Title> 4 < . . . >
	select Category as All	1 <Select, Category, All>
	Drop down list	1 <Drop, Down, List> 2 <Drop, List, Down>
	then **Click** Search Button	1 <Click, Search , Button> 2 <Click, Search, –> 3 <Click, Button, Search> 4 <Click, Button, –>

Thummalapenta, Sinha, Singhania, Chandra. Automating Test Automation. ICSE 2012.

其次在测试数据生成、测试预言生成方面：

介绍了智能编程机器人在研发、测试代码编写中提供的诸多智能化能力以及实践效果。

与微软合作研发自动测试数据产生工具 Pex，其中的 Fitnex 路径搜索技术，在测试数据生成领域的智能化实践：



介绍了自然语言测试、持续学习能力、带参数的测试用例等更多智能化测试实战分享：

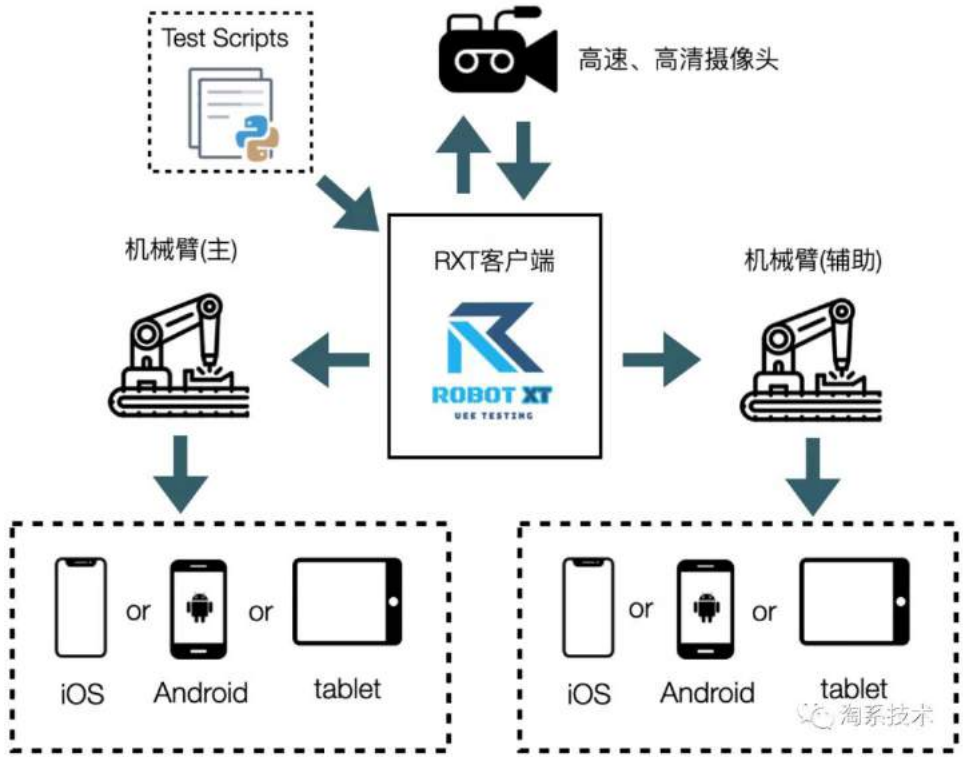
UEE 测试新模式 - 极测机器人 RXT 端到端测试

最后，本次谢涛老师访问学者在淘系技术质量合作方代表金晖（定源），分享了淘系在 UEE 端到端智能化测试新模式下的探索和实践：

UEE 自动化测试包括两大核心能力：

- 用户交互体验评测：通过 IoT（如摄像头、传感器）等外部设备采集并将真实用户感受数字化，来获取最接近真实用户体感的性能及用户体验指标；
- 多设备端到端功能自动化：通过机械手实现同时多设备的跨终端的自动化操作执行；通过 AI 算法实现 IoT 设备的调度控制、场景理解、测试步骤执行、体验指标计算、异常处理等能力。

通过 RXT 机器人测试平台来开展 UEE 自动化测试：



UEE 自动化和 UI 自动化的差异

通过脱离被测 APP 本身的一些限制实现真正的端到端测试。并进一步屏蔽被测 APP 在不同平台（Android/iOS/PC/IoT）之间自动化操作层面的差异、避免软件层进行性能体验度量的干扰和差异、以及增加了多设备间交互式自动化的能力。

- 测试范围的差异：基于高速摄像头的外部视角+机器视觉算法的识别处理，UEE 测试同时包含了端到端功能自动化+用户体验评测的能力。
- 精度的差异：高速摄像头摆脱了手机录屏软件对被测 APP 的性能干扰和帧率限制，将用户体验指标精度提升至 10ms。同时可以统一衡量标准，以外部摄像头作为统一的标准来衡量不同手机机型、不同终端设备的性能指标，减少软件层面造成的衡量标准差异。
- 支持多设备跨终端的操作：基于机械手封装的执行引擎 RXT，可以同时操作多部手机。将以外部用户的模式操作被测手机和 APP，并可以同时操作多个设备。且用一套 DSL 脚本支持，屏蔽底层设备和操作系统的差异（Android/iOS/PC Touch/IoT 系统等）。

未来展望

和谢涛教授团队一起在智能探索测试、智能验证、新一代智能化移动实验室等方向打造更加智能的测试机器人。

活动现场

手淘再推新利器 Holmes：一站式智能化异常检测平台

作者 | 吾铭、豆豆
出品 | 淘系技术

指标监控关乎稳定性，但随着数据量的增加、指标的复杂周期性和模式变化的动态性，基于阈值/同比环比的规则难以适用，而且复杂的领域知识导致为每条指标配置相应的规则费时费力，无法应用在大规模数据监控上。在监控的有效性方面，传统的规则报警无法智能识别季节性，也经常受到噪声/抖动数据的干扰而导致误报，固定的规则以及阈值更无法进行提前预警。

前言

在上面的背景下，淘系技术质量团队打造了一款基于 AI 算法的异常检测平台，Holmes 是一款智能化、轻量级、易接入、可扩展的异常检测平台，使用基于 AI 的异常检测算法，替代传统的规则监控方案。解决规则告警系统准确率低、时效性低、规则配置复杂与耗费人力等诸多问题。

Holmes 目前已经在阿里内进行开放，覆盖淘宝、千牛、优酷、钉钉、淘宝直播等多个应用，核心指标量 200+，整体准确率在 90%以上，算法调用量高达 1000 万余次，有效进行故障的提前预警 20 余次。

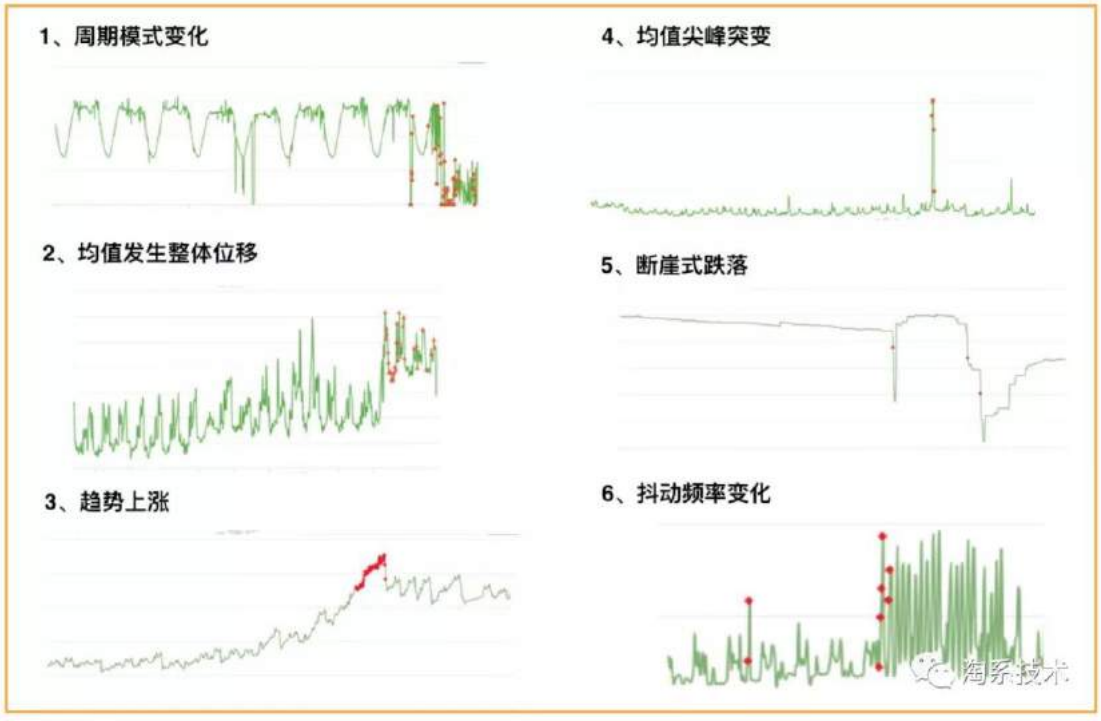
平台理念

Holmes 通过 AI 算法对业务指标进行检测和预测，从而实现智能化报警，弥补了规则监控的不足，减少误报和漏报情况。提供一站式的异常检测服务，满足时序数据的实时检测，提供多种接入方式，利用机器学习和深度学习方式，定制化学习模型，有效提升报警覆盖率，以智能化方式进行稳定性的监控和保障。

- 特点：
  - 学习历史数据，分析当前指标曲线趋势是否异常
  - 基于以往数据，进行预测未来指标走势

- 优势：
  - 算法检测代替规则检测
  - 告警准确率高
  - 更早发现异常情况
  - 可适应业务发展带来的趋势变化

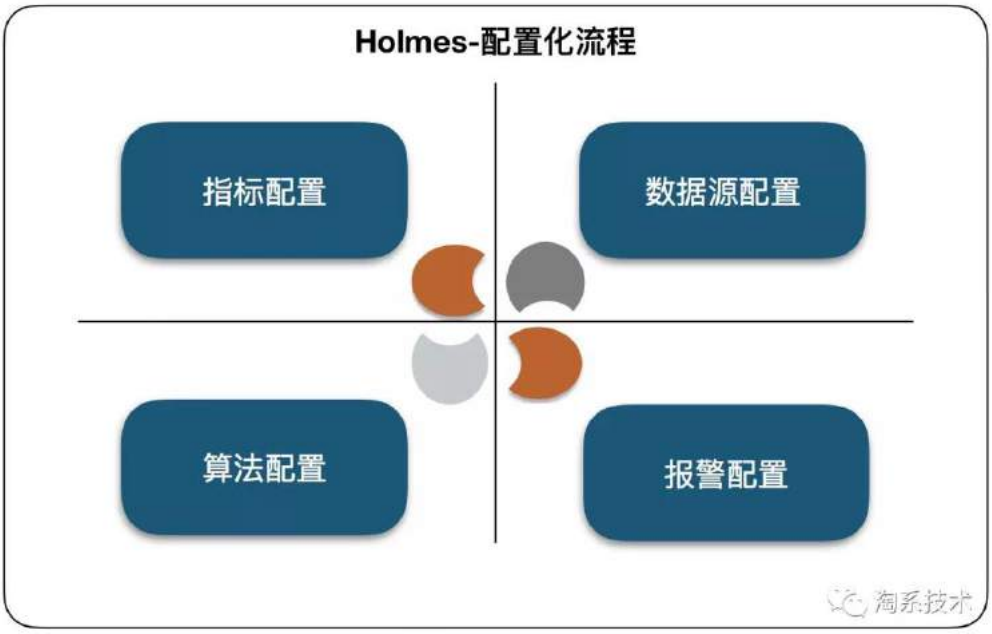
解决的异常场景：



系统架构

配置化流程

通过 4 步简单配置进行指标的接入和算法选择，轻松开启智能异常检测。



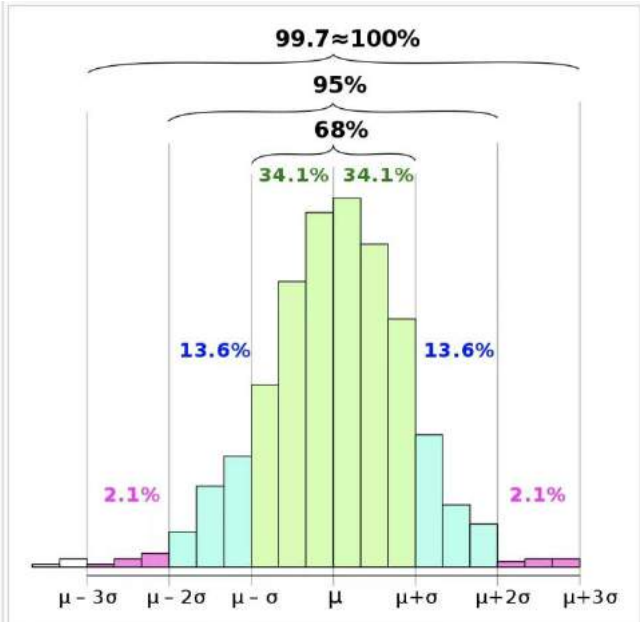
算法概览

Holmes 融合了多种检测、预测的时序算法，检测异常响应速度快，预测数据走势准，二者的完美结合，奠定了异常检测的基础能力，同时平台也支持扩展算法的能力，以适应新增场景的特殊需求。

检测算法：

Statistic

In statistics, the 68-95-99.7 rule is a shorthand used to remember the percentage of valuesthat lie within a band around the mean in a normal distribution with a width of two, four andsix standard deviations, respectively; more accurately, 68.27%, 95.45% and 99.73% of the valueslie within one, two and three standard deviations of the mean, respectively.



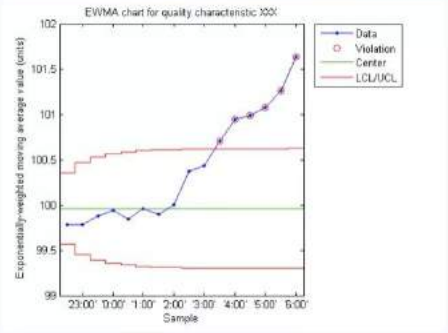
For an approximately normal data set, the values within one standard deviation of the mean account for about 68% of the set; while within two standard deviations account for about 95%; and within three standard deviations account for about 99.7%. Shown percentages are rounded theoretical probabilities intended only to approximate the empirical data derived from a normal population. This rule is generally taught within common core mathematics.

淘系技术

Ewma

In statistical quality control, the EWMA chart (or exponentially weighted moving average chart)is a type of control chart used to monitor either variables or attribu tes-type data using the monitored businessor industrial process's entire history of output. While other control charts treat rational subgroups of samplesindividually, theEWMA chart tracks the exponentially-weighted moving average of all prior sample means.

Process observations	
Rational subgroup size	n = 1
Measurement type	Moving average of the quality characteristic
Quality characteristic type	Variables data
Underlying distribution	Normal distribution
Performance	
Size of shift to detect	⩽ 1.5σ
Process variation chart	
Not applicable	
Process mean chart	

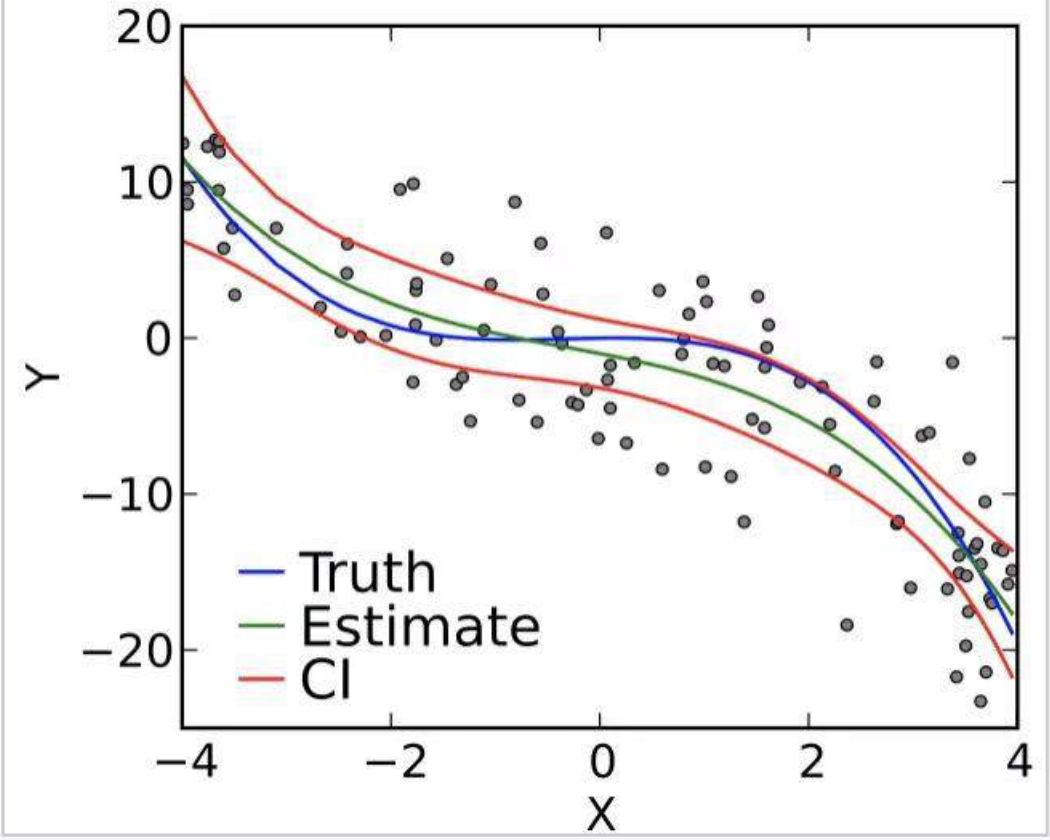


Center line	The target value, T, of the quality characteristic
Control limits	$T \pm L\frac{S}{\sqrt{n}}\sqrt{\frac{\lambda}{2-\lambda}[1-(1-\lambda)^{2i}]}$
Plotted statistic	$z_i = \lambda\bar{x}_i + (1-\lambda)\,z_{i-1}$

淘系技术

Polynomial

In statistics, polynomial regression is a form of regression analysis in which the relationshipbetween the independent variable x and the dependent variable y is modelled as an nth degree polynomial in x.



A cubic polynomial regression fit to a simulated data set. The confidence band is a 95% simultaneous confidence band constructed using the Scheffé approach.

淘系技术

IsolationForest

The IsolationForest 'isolates' observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.

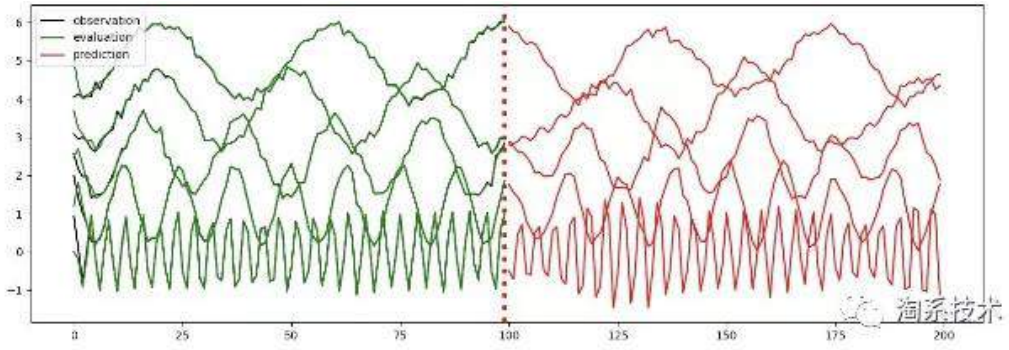
XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under theGradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

预测算法:

Tensorflow-LSTM

Long Short-Term Memory layer - Hochreiter 1997.预测效果图:



Facebook-prophet

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

实践效果

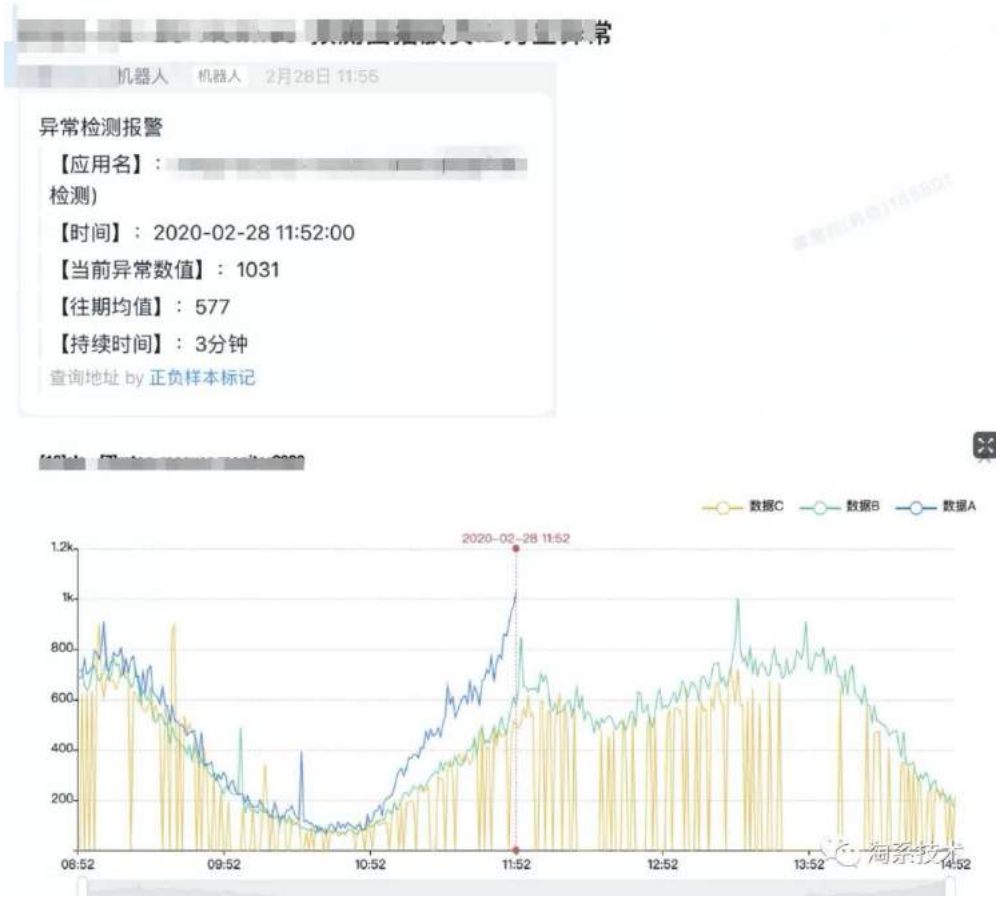
目前 Holmes 异常检测平台已经在集团内部开放接入和运行，支持集团内常用数据源。平台运行几个月以来，帮助接入业务方的开发测试同学构建智能监控体系，减少繁琐的规则配置，有效提高了线上质量监控的覆盖率。今年 618 大促期间，Holmes 的准确性方面也进一步得到验证，有效保障了大促的稳定性质量。

覆盖应用：淘宝、千牛、优酷、钉钉、淘宝直播、咸鱼等接入指标：核心业务指标 200+提前预警：有效提前预警线上问题 20+算法调用量：累计 1000 万+

报警示例

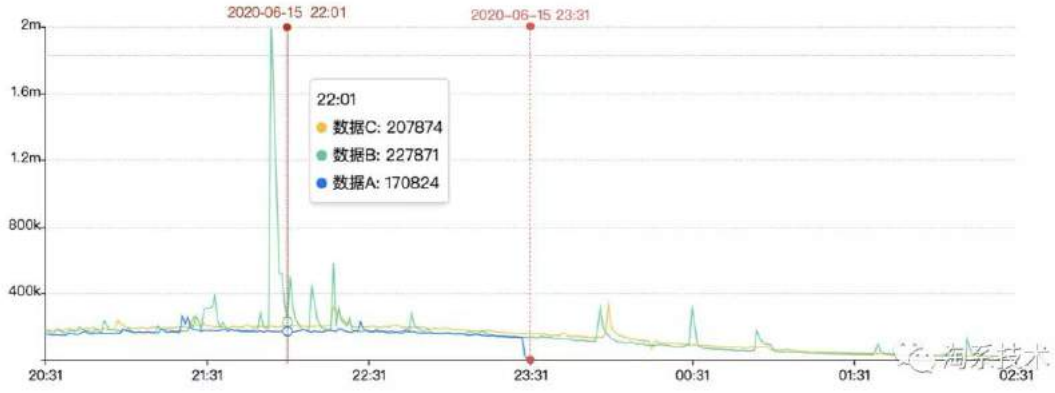
实践案例 1:

A 客户端新版本放量期间，由于客户端请求传参问题导致服务端返回大量空返回错误，Holmes 进行了有效检测到异常，相关同学收到报警及时中止放量并修复问题。然而传统规则监控因未达到阈值没有预警。

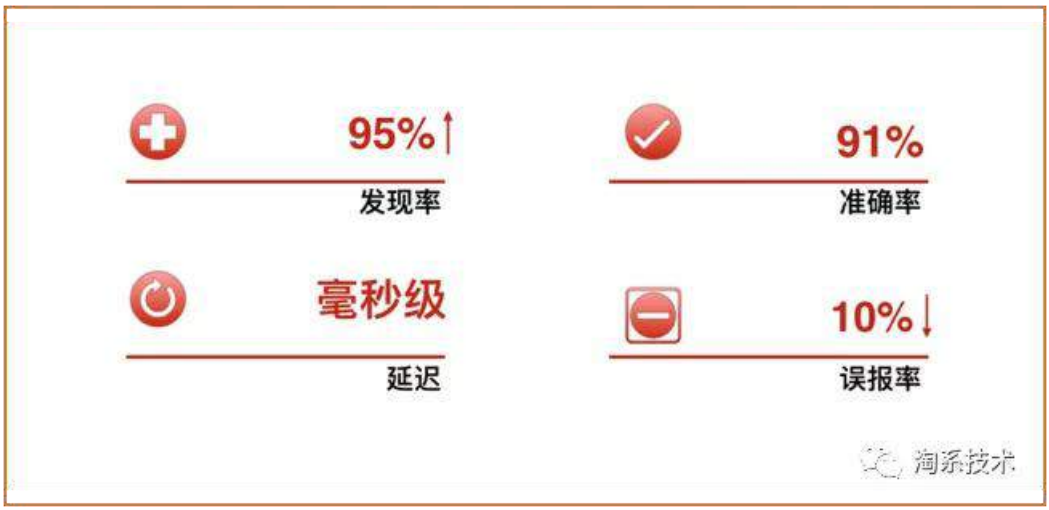


实践案例 2：

在 618 大促压测期间，由于 B 业务订单数据量持续下降（缓慢下跌），Holmes 检测到持续异常信息，预测数据下跌风险，早于传统规则监控 90 分钟提前报警，有效避免了线上故障。



算法检测效果



未来展望

Holmes 异常检测平台是淘系技术质量团队打造，在智能化测试领域的一次实践，未来我们希望利用 AI 算法实现业务全方位智能化监控和问题定位。覆盖更多的数据类型、打造通用的算法模型、更优越的算法指标。同时我们也在全链路监控排查、智能舆情处理等多方面进行探索，期待后续跟大家分享。

首次面向 B 端展开全链路压测！淘系高难度压测实践方案公开

作者 | 达海、追溯
出品 | 淘系技术

背景

“今年的双 11 是全球极大内容电商场的超级爆发，消费者、技术、内容与商业生态之间每一秒都在产生激烈共振，实时性、复杂性和持续峰值的叠加令其成为全球技术顶峰。2020 年双 11，阿里巴巴峰值交易达到了 58.3 万笔/秒，其背后的商家链路也承受着史无前例的压力”阿里巴巴副总裁汤兴如此描述今年的双 11。

阿里商家业务域涉及集团近 20 条业务线，100 多个场景，400+链路，部分业务深度融入在导购、交易链路中。商家链路业务涵盖淘系的上千万商家及数百家核心三方服务商的伙伴。以往由于平台和商家 IT 基础设施存在“水位差”，平台很难帮助商家进行系统的改造升级和全链路验收。随着阿里面向商家与生态伙伴推出了新一代数字化基建体系，借助云原生技术引擎与云 IT 治理能力，帮助商家与生态伙伴重构系统的高可用基线，使得越来越多的商家具备了和阿里同等量级的超大规模数据处理能力。

常规备战中各个业务依靠单链路压测，缺少互通联动，缺少全局把控，容易有质量盲点，隐患容易被忽视，风险较高。商家链路出现问题，直接影响商家生产经营能力，对商家和消费者造成重大的体验伤害。业内的全链路压测普遍面向 C 端场景，B 端场景往往重视度不够。商家端的场景结构极其庞杂，涉及内部系统还涉及众多三方系统，开展全链路压测面临着诸多挑战。为有效预防风险，提升各个业务以及三方合作伙伴系统稳定性，提升用户体验，今年我们克服了重重困难首次面向商家链路开展了全链路压测。

挑战

核心难点

商家业务涵盖商家工具、消息、多媒体、算法等多种业务形态，各应用间调用错综复杂，如何理清核心链路的依赖，保障核心链路不遗漏是开展压测的首要挑战。

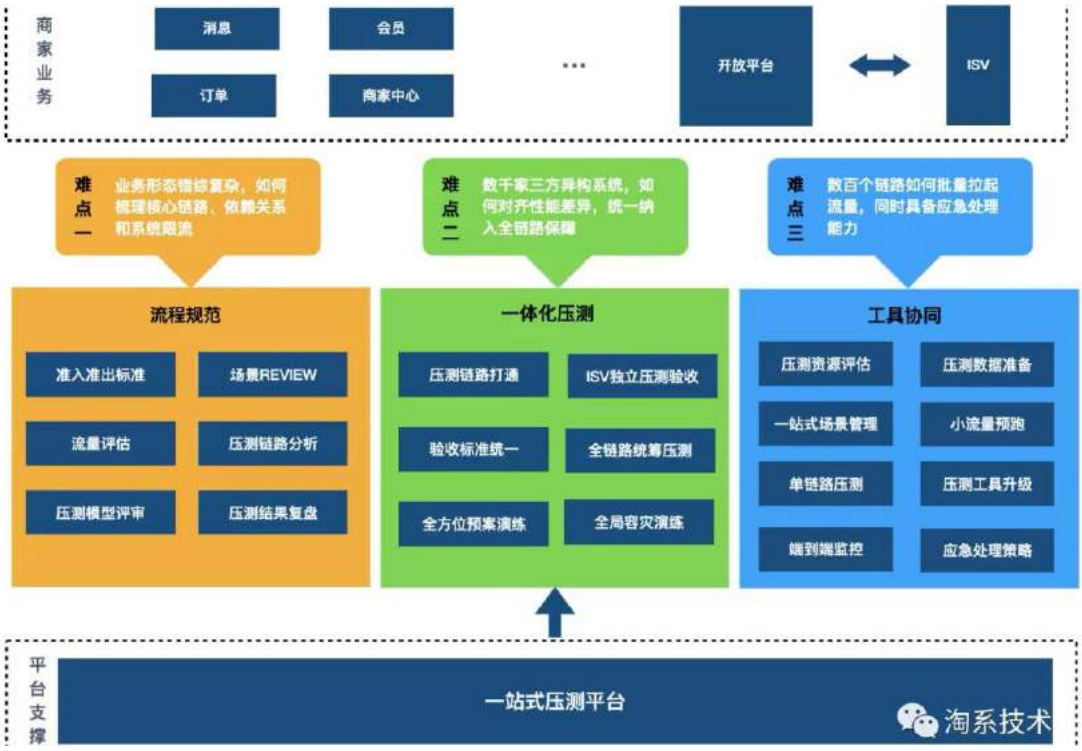
压测实施上，如何模拟数百个场景流量的叠加耦合，并留好应对突发情况的操作空间，且在同一时间批量将流量拉到峰值是压测执行的一大挑战。

商家业务涉及众多合作伙伴服务商的系统，阿里生态的复杂性，使得接入阿里平台的服务商系统架构存在异构化、多样化、性能差异大等多方面技术挑战，每个服务商对于稳定性的认知存在差异。

解决方案

针对上述挑战，我们的核心解决方案为：

- 统一流程规范标准、压测组织协同、统一验收复盘。
- 工具支撑压测一体化。
- 压测涵盖预案、限流、演练、监控等内容，模拟大促真实情况。



全链路压测方案

1. 统一流程规范

各个业务参与全链路压测明确准入准出标准。全链路压测的基本准入原则：每个场景单链路压测通过后才能进入全链路压测。准出标准包括：系统水位、流量比例、响应时间、缓存命中率、限流、jvm 指标等。服务商系统与阿里系统标准一致。

流程上统一进行场景 review、链路评审、全链路压测以及复盘，复盘中的问题同步优化解决。

2. 工具支撑一体化

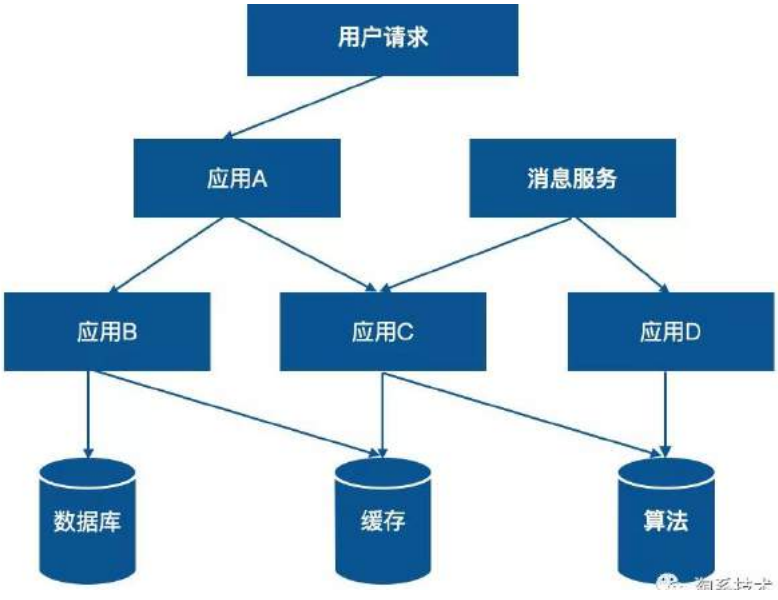
在集团已有的压测能力基础上，我们重点在商家链路分析，流量评估，压测模型、结果校验自动化等方面建设一站式压测工具，解决商家全链路压测的核心问题。

链路分析

压测场景选取的基本原则：

- 大流量场景
- 核心链路场景
- 复杂业务链路场景
- 流量扩散场景

参照以上原则，在我们在集团中间件支持的基础上，开发了链路分析场景推荐工具，用人工+工具 check 双保险的方式进行链路模型生成。示意原理如下图，应用 A 某接口的一次调用，我们可以获取其下游应用的 trace，访问的数据库或者缓存，并且分析出 1 次调用对下游流量的放大。



链路示意图

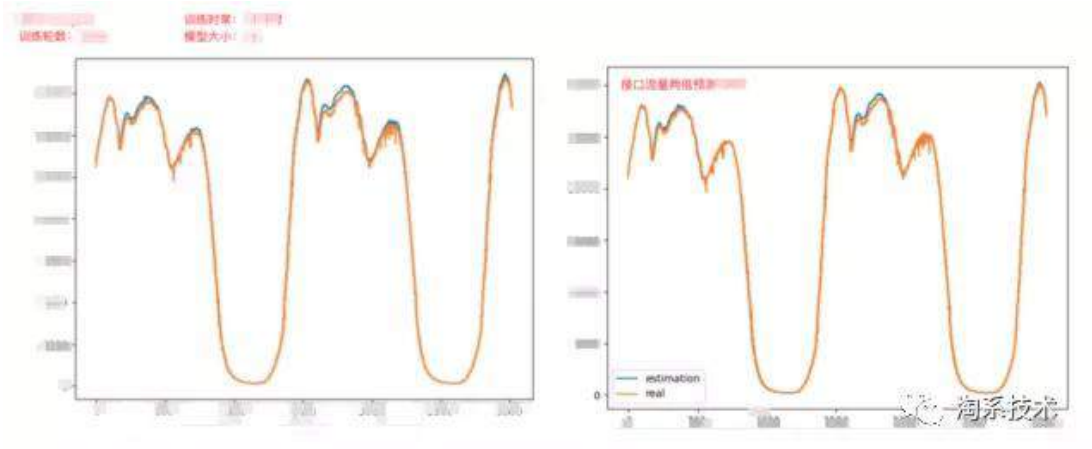
其工作流程如下：

- 统计流量调用的服务，存储等指标，获取对下游的调用次数，得到对下游请求的放大。
- 获取流量大于某阈值的入口链路。
- 获取调用链路深度或者依赖调用大于某阈值的链路。
- 获取翻倍调用大于某阈值的链路。

流量评估

常规流量评估通常是根据监控和上游调用来评估接口流量，评估比较理想化，容易产生误差。如果前期流量预估不充分，会导致压测无法达到目标或远超目标值，靠压测来发现这些问题成本太高。对此我们引入了深度机器学习，利用算法能力来做智能预测，更加精确的进行流量评估。其主要的工作原理如下:

- 抓取应用的出入口流量、RT、QPS、错误率、应用系统水位等信息。
- 应用深度机器学习，对基础数据进行分析学习，产生该应用的算法模型。该模型可以根据入口流量，预估出口流量、应用系统水位等信息。效果如图所示。



接口流量模拟对比图

其核心思想为:

- 算法每天根据线上真实数据持续学习，持续优化，生成可靠模型。
- 用户输入接口预估流量，平台输出预估的下游接口流量、存储 QPS、机器数和系统水位等信息。

模型构建

结合业务场景，我们建立了两种压测模型：0 点模型和非 0 点模型。0 点模型涉及场景在大

促 0 点流量达到峰值，非 0 点模型场景峰值则在其他时间达到峰值，压测模型的设计不是简单的覆盖业务，还需要考虑各个业务之间的流量耦合。有上下游关系的链路，下游的压测流量也是来自上游。除了考虑商家域内的流量耦合，还要考虑集团其他域的流量耦合。因此，与之配合的还有非常多的压测预案，用以控制压测流量的准入准出。

有流量的耦合就要考虑耦合流量不足的情况，不是每次压测上游都会有充足流量打到下游。在保障能够耦合上游流量的同时，下游业务自身还要具备流量补充能力，当来自上游的流量不足时，自身可以补充足够的流量，满足自身系统的验证。

3. 压测执行

压测数据准备采用淘系技术质量部自研凤凰平台[见附 1]录制线上流量做“平移”，场景数据更加真实有效。录制化也提升了数据准备的效率，一次录制多次使用。场景管理上做到了 1 人 1 键执行所有的场景，极大的节约人力。每次全链路压测商家和交易、导购都会同步开展，0 点流量全部打到 100%，以验证跨业务域的流量依赖和叠加的情况。从压测开始到流量拉到 100%时间非常有限，在压测初期，这个过程中通常会伴随着非常多的问题。这些问题在前期准备过程中很难发现，主要问题有以下几类：

- 压力分布不均。
- 模型紧急调整。
- 数据文件紧急修正。
- 个别压测机异常。
- 受压测管控影响，服务调用受限。
- 压测任务紧急管控。

执行人员需要一边控制不受影响的业务继续按计划加压，同时需要迅速对出现的问题作出判断，给出解决方案。对于能够快速修正的场景，首先将其从压测活动中单独摘掉，调整完成后重新关联到压测任务中。对于无法快速修正的场景，要迅速协调业务同学进行单链路操作。受上述问题影响，现场需要制定不同的应对策略，主要包括以下几种情况：

- 压测过程中某个场景需要紧急卸载压力，其他场景保持。
- 压测过程中某个场景没有达到目标，但系统资源已经出现瓶颈，需要保持压力水位排查问题，其余业务继续加压。
- 压力拉到 100%个别业务没达到预期需要继续增加压力。

这几类问题在前期压测过程中出现频率非常高，如果前期准备不到位会阻塞压测节奏。

此外全链路压测具备从客户端（成功率、错误量、舆情等）到服务器（接口成功率、中间件成功率、水位等）的完整监控体系，避免出现只关注到服务可用性而忽视客户端及用户实际体验受损的盲区。

在数百个商家全链路压测场景中，消息、开放、小程序是几个典型的非常规压测场景，压测实施难度大，接下来重点介绍一下这几个场景的全链路压测实践。

核心场景 1: IM 消息体系的端到端全链路压测

传统服务端压测通常是 http 或者 tcp 类型的短链接压测，也就是客户端请求一次，拿到返回后，连接就关闭了。但是这种方式并不适用于 IM 消息系统的压测，IM 的系统与服务端建立的是长连接，用户 A 发送消息给用户 B 的时候，用户 B 是被动收到服务器的推送，而不是主动拉取数据，这个模式对压测的实施非常有挑战。

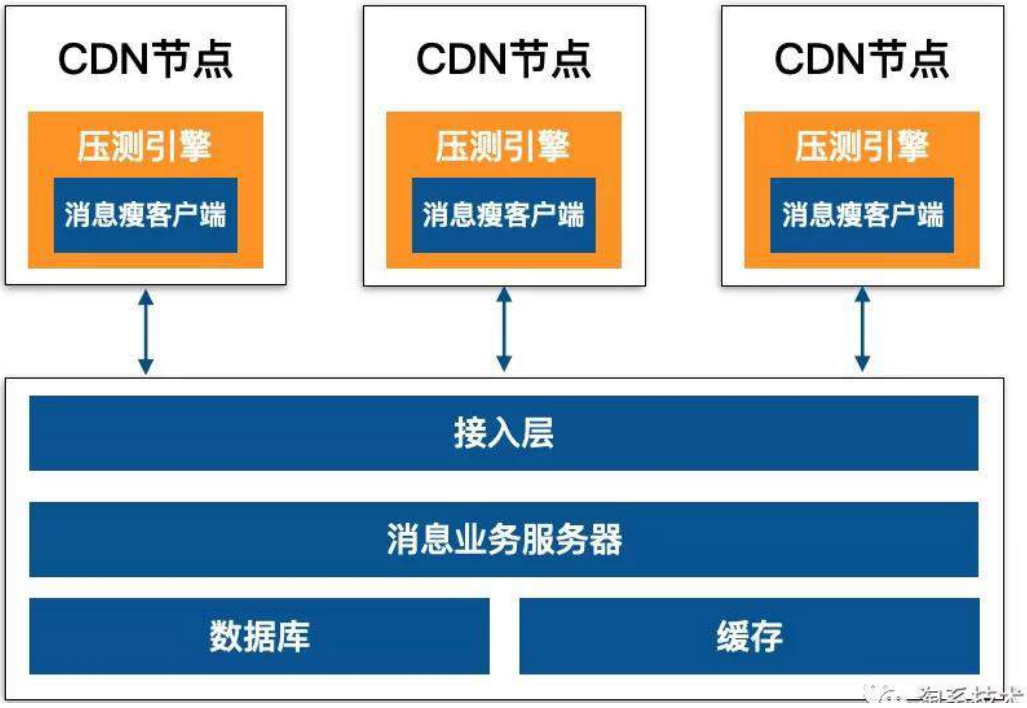
为了解决这个问题，我们基于 NIO 开发了手淘和千牛的瘦客户端，模拟真实用户与服务端保持长连接在服务端，并将部分业务逻辑做了参数化，集成到了瘦客户端中。比如瘦客户端在收到消息推送后，回复 ACK，表示收到了消息，并且根据已读比例对该消息进行已读设置。

长连接瘦客户端

消息链路压测需要模拟客户端登录后保持长连接，目前业内常规压测工具对消息场景都不适用，需要独立开发工具。核心方案如下：

- 开发瘦客户端集成到压测引擎中模拟端到端的场景
  - 模拟消息回复
  - 模拟消息已阅读
  - 模拟消息漫游

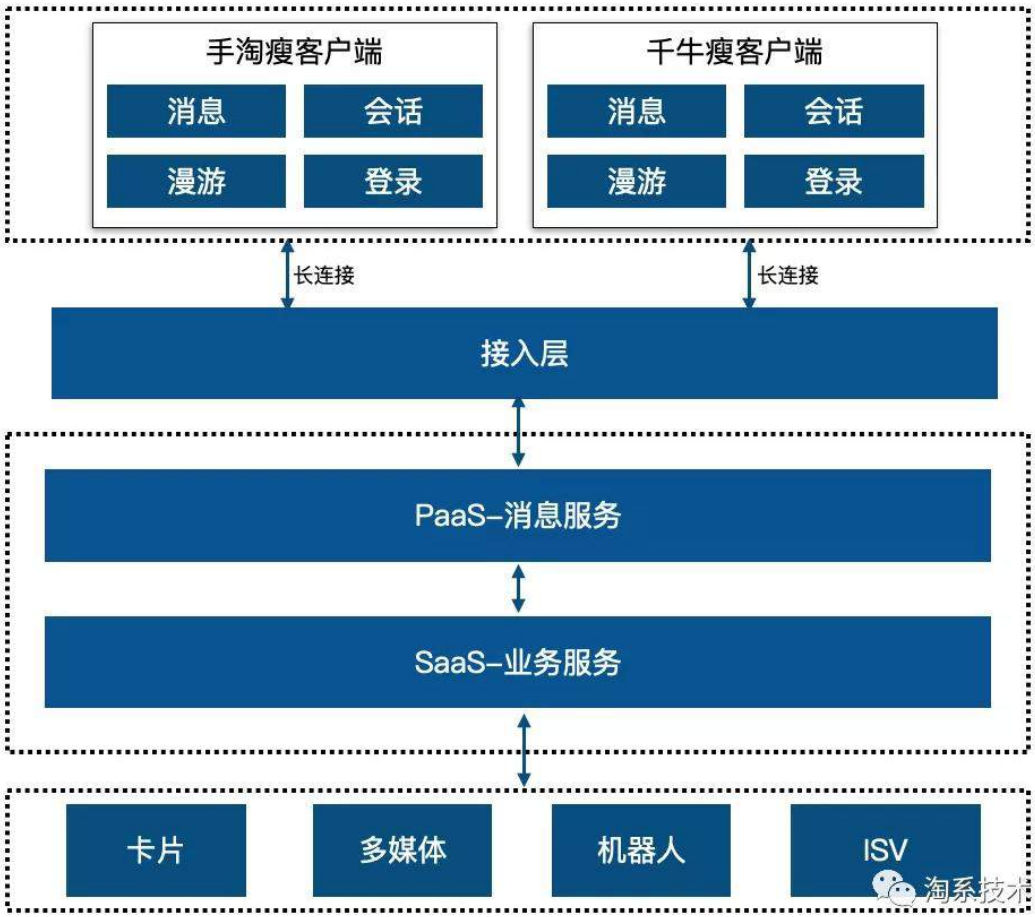
- 压测引擎部署到 CDN 节点上模拟真实用户收发消息
- 创建长连接模拟用户真实保持长连接在线



长连接压测架构图

全链路消息业务打通

消息全链路除了包含消息上下行，消息已读未读，消息内容模型（文本、卡片、多媒体消息），消息漫游等基础消息业务，也在集团二方和合作伙伴三方中衍生出了如机器人智能辅助、订单核对，修改地址，智能客服等消息业务。

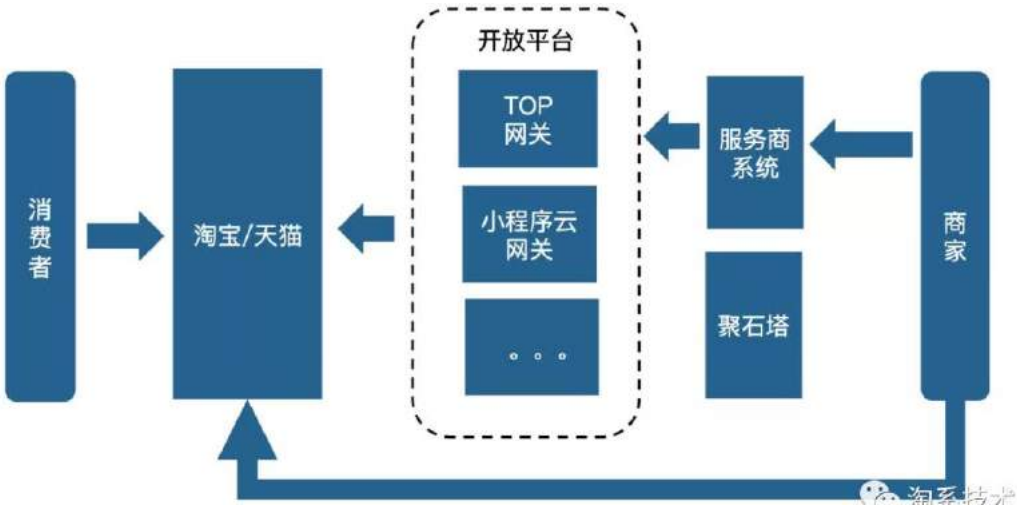


消息业务图

由于消息业务的复杂性，导致我们在压测的时候需要协调各个业务方的压测数据，来调整实际压测时各业务方的压力。例如某个账号开通了 A 业务也开通了 B 业务，发送给该账号的消息走到了两个业务，导致流量叠加，最后的压测结果不准确。为了解决这一问题，我们统一收口了消息业务压测脚本，通过统一分配账号，通过统一开通业务，统一发起流量等方法，解决了流量叠加互相干扰以及各业务线脚本无法复用等一系列问题。

核心场景 2：三方服务商全链路压测

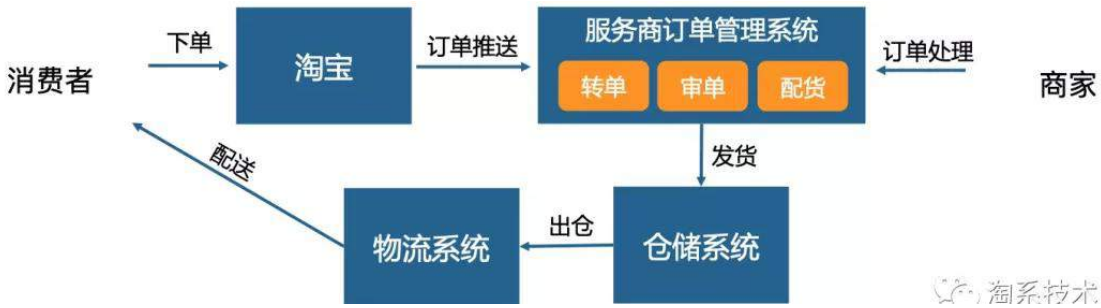
淘系电商业务是一个覆盖消费者、平台、商家、三方服务商的生态链路。借助平台开放的接口，三方服务商可以开发出客户管理、订单管理、互动营销等方向的电商支持工具，帮助商家提升运营效率。要实现商家全链路保障，就需要驱动并赋能三方服务商一起参与到全链路压测中来。



淘系电商生态交互图

订单推送全链路压测

在涉及三方的众多场景中最核心的是订单场景，订单推送全链路覆盖了消费者在淘宝下单后到收到货物的整个过程。链路涵盖多个内外部系统，直接关系到消费者的购物体验，压测重要性不言而喻。



订单推送全链路交互图

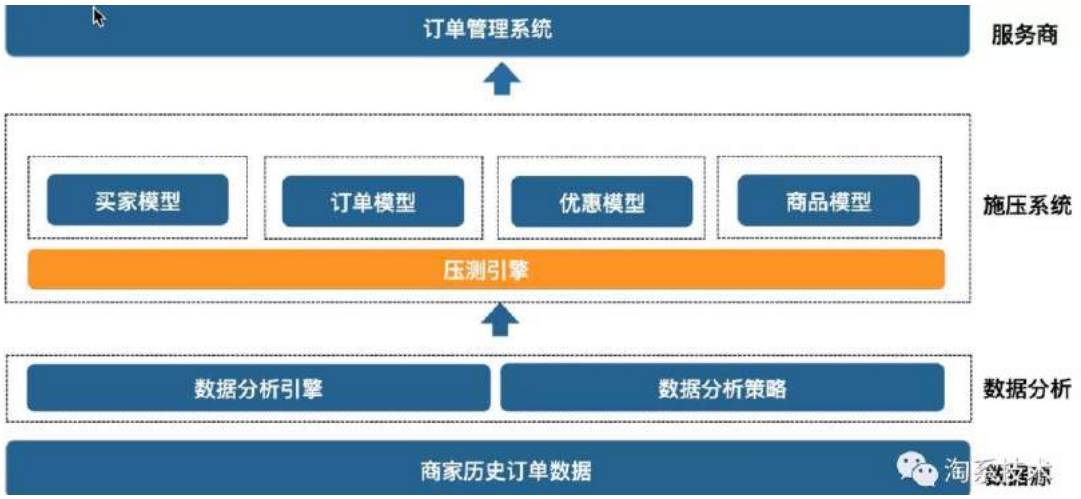
订单推送全链路压测面临以下痛点：

- 服务商数量众多，没有统一的压测工具。
- 压测订单模型构造难度大、准确性差。
- 压测结果收集困难，数据分析工作量大、标准不统一。

为此我们开发了订单推送全链路压测平台赋能服务商，平台具有以下特点：

基于历史订单数据一键生成压测模型，方便、精准

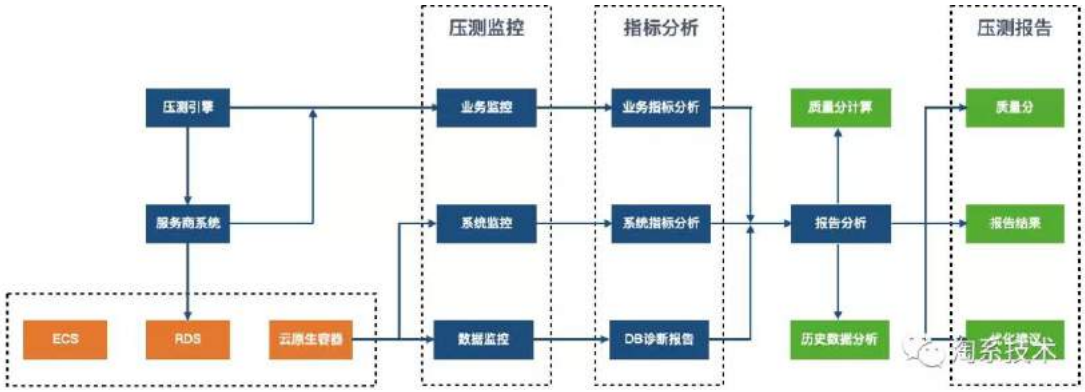
用户只需指定历史订单的起止时间，系统会自动分析该时间段内的线上订单数据，并根据订单的买家、优惠、商品等信息模拟生成压测订单模型。



订单推送压测模型

压测结果数据自动收集、分析，全面、直观

工具会自动从监控数据中收集相关压测指标，在报告分析阶段跟历史数据进行比对、根据压测标准计算出压测质量分、产出压测结论、给出优化建议。



订单推送压测报告生成示意图

支持 Service Mesh 技术，生产环境压测更简单、安全

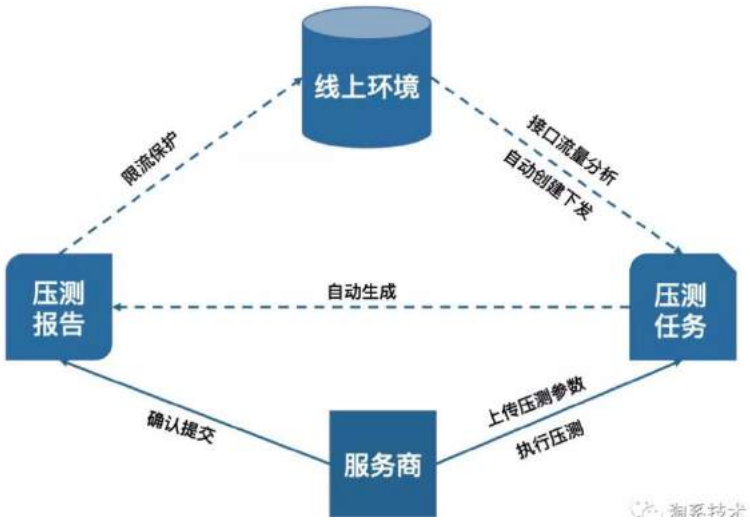
压测平台支持云原生的 Service Mesh 技术，对于生产环境压测改造只需修改 POD 节点的相关配置，无需对系统业务代码进行改造，大大降低生产环境压测成本，同时减少代码改造可能带来的安全风险

核心场景 3:三方小程序压测

小程序是淘系在 APP 端开放的重要形式，由于三方小程序集成在手淘、千牛等淘系 APP 中，其稳定性关系到淘系 APP 的稳定和用户体验。受技术和安全上的限制，三方服务商无法自主对小程序接口进行线上压测，为此我们开发了小程序压测平台为服务商伙伴们提供高效、易用的官方压测工具。

三方小程序压测平台使用简单、易上手、自动化程度高，大大降低了技术门槛和服务商的压测成本，其核心能力如下：

- 系统根据线上接口流量自动分析、创建、下发压测任务。
- 服务商只需编辑压测参数，执行压测任务即可。
- 压测通过并提交报告后，系统会根据压测流量值设定限流保护值。



三方小程序压测流程图

此外，我们还组织了头部核心服务商联合预演。通过全链路压测，模拟业务峰值流量，演练了各种应急预案和沟通机制，以应对大促中可能出现的异常和突发状况，形成了压测、封网、预演的完整保障链路：

- 在全链路压测的同时对核心三方小程序进行压测，模拟更真实的双十一环境。
- 在压测的同时进行异常情况演练：人工模拟引入问题、故障。
- 对问题、故障进行紧急处理，演练沟通、协同机制和应急处理流程。

总结

全链路压测期间总计发现 130+内部系统问题，大促前所有问题都得到了有效解决。整个大促期间商家业务系统 0 故障、商家问题反馈较往年降低了 50%，商家以及消费者感受到了如丝般顺滑，捍卫了商家利益与体验。

一百多家核心的 ISV 参与了全链路压测验收，全链路压测帮助商家与生态伙伴重构了系统的高可用基线，使商家在信息处理和数据处理的水平上，和淘宝拉到同一个水位，使得商家具备了和阿里一样的超大规模数据应用能力，帮助商家系统发现并修复上千个性能问题，在效率和能力上大幅度提升商家系统稳定性。

未来我们计划将在三个方面持续优化，为商家和合作伙伴提供更优质的服务：

- 基于算法能力的智能化压测。
- 场景、预案、限流、压测结果等全方位的自动化分析。
- 多维度监控、海量告警信息的智能分析定位处理。

附 1:

凤凰生态利用 JVM-Sandbox 提供的统一底层能力，大家奇思妙想产生模块原子能力生态，提供开放、快速的模块开发、管理、鉴权、部署能力(魔方平台)，通过对模块能力的组合，衍生出录制回放、故障注入、强弱依赖梳理、系统 Mock、快速问题定位、测试质量评估等

上层产品模块(凤凰平台 2.0)，并对各个环节产生的数据、标准能力都开放 API 和能力，在业务回归、攻防演练、架构治理等领域输出方案，致力于快速、高效的提升系统整体稳定性。

开源地址：
https://github.com/alibaba/jvm-sandbox-repeater?spm=ata.13261165.0.0.11ee30bfW4qEoF

JVM-Sandbox 属于基于 Instrumentation 的动态编织类的 AOP 框架，通过精心构造了字节码增强逻辑，使得沙箱的模块能在不违反 JDK 约束情况下实现对目标应用方法的无侵入运行时 AOP 拦截。

开源地址：
https://github.com/alibaba/JVM-Sandbox?spm=ata.13261165.0.0.5a094b01By8WRH

测试人精彩问答

Q: 各位都是怎么进行单元测试的?

作者 | 志春

出品 | 淘系技术

1. 大牛们都是怎么测的，要用什么工具吗?
2. 对于 GUI 的部分一般怎么测?
3. 想系统的从零开始学学 UnitTest 应该怎么做?

在阿里淘系，有很多复杂的巨型项目，包罗多个业务，而且往往服务依赖比较复杂；再加上一些特殊环境变量的设置，想要在本地运行、debug 自测这种大型应用的难度越来越高；尤其是对环境不太熟悉的新人而言成本会更高。

这类应用的单元测试不能像微服务化的应用一样，可以方便的将整个 service 在本地 Run Test，但是依靠于日常开发部署环境的远程 debug、日志、Arthas 等工具定位项目自测联调中的问题又会显得格外的笨重，问题修复几秒钟，发布一次 10min 会成为严重的效率瓶颈。

如何高效的自测代码逻辑，如何不启动整个服务就能验证我的目标方法呢？那就是我今天要介绍的单元测试三板斧：Mockito + PowerMock + AssertJ

上手

Mock 框架能帮助我们 mock 待测试的类中使用到的外部服务依赖，分布式缓存，DB 查询等复杂逻辑，让我们轻松验证待测试类的目标方法的逻辑，当遇到外部依赖时可通过存根

mock 对应的返回结果，从而专注于验证本方法的逻辑正确性，而且跑单元测试不用把整个项目在本地跑起来，只会把当前测试所用到的类加载出来。换言之，Mock 能让代码对外部系统(或复杂依赖)隔离，不需要进行各种初始化操作。在假设外部依赖都能如预期返回的情况下验证自身逻辑的自洽性。

talk is cheap，show me your code. 开始盘它~

配置 Maven 依赖

```
<dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.11</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-core</artifactId>
        <version>3.5.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.powermock</groupId>
        <artifactId>powermock-module-junit4</artifactId>
        <version>2.0.5</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.powermock</groupId>
        <artifactId>powermock-api-mockito2</artifactId>
        <version>2.0.5</version>
        <scope>test</scope>
    </dependency>
```

```
    <dependency>
      <groupId>org.assertj</groupId>
      <artifactId>assertj-core</artifactId>
      <!-- use 2.9.1 for Java 7 projects -->
      <version>3.17.1</version>
      <scope>test</scope>
    </dependency>
```

Mockito

Mockito 可以 mock 类的 public 方法或接口的方法。它是通过 cglib 动态生成一个 Proxy，因此在未指定某个方法行为的情况下，会默认返回空值，当然，一个完善的框架肯定会支持直接访问被代理的对象的真实方法的，下文会有介绍，一共会有 3 种方式哦，我们继续吧。

这里我们使用的 mock 类定义如下：

```
import java.util.concurrent.TimeUnit;

public class MockTarget {

    public void soSth() {
        System.out.println("do sth.");
    }

    public String sayHello() {
        return "Hello";
    }

    public String sayHello(String greetings) {
        return "Hello " + greetings;
    }

    public String callMethod(Object p) {
```

```
        return "callMethod " + p.toString();
    }

    public String callMethodWait(long million) {
        try {
            TimeUnit.MILLISECONDS.sleep(million);
        } catch (InterruptedException ignored) {
        }
        return "callMethod sleep " + million;
    }

    public Object callMethodWithException(Object p) {
        throw new IllegalStateException("测试异常");
    }
}
```

when..then

用于 mock 方法调用的各种返回情况。

- 通过 doCallRealMethod 指定 mock 对象的方法调用它的真实逻辑，也可通过 thenAnswer(Answers.CALLS_REAL_METHODS) 实现。
- 通过 when..thenThrow 或者 doThrow..when 的方式 mock 目标方法返回对应的异常。
- 通过 AssertJ 的句法 assertThatExceptionOfType..isThrownBy..withXxx 断言某个方法的执行会抛出预期异常。
- anyXxx() 可用于表示任意类型的任意参数。
  - anyString() 代表任意字符串。
  - anyInt() 代表任意 int 数值。
  - anyObject() 代表任意类型对象。

```
@Test
    public void testWhenAndThen() {
        MockTarget mock = Mockito.mock(MockTarget.class);
```

```
        when(mock.sayHello()).thenReturn("mock hello");
        assertEquals(mock.sayHello(), "mock hello");

        doCallRealMethod().when(mock).sayHello();
        assertEquals(mock.sayHello(), "Hello");

        when(mock.sayHello(anyString())).thenAnswer(Answers.CALLS_REAL_METHODS);
        assertEquals(mock.sayHello("testRun"), "Hello testRun");

        when(mock.callMethod(any())).thenReturn("mock return");
        assertEquals(mock.callMethod(new Object()), "mock return");

        when(mock.callMethodWithException(any())).thenThrow(new RuntimeException("mock throw
exception"), new IllegalArgumentException("test illegal argument"));

        Assertions.assertThatExceptionOfType(RuntimeException.class)
                .isThrownBy(() -> mock.callMethodWithException("first invoke"))
                .withMessage("mock throw exception");
        Assertions.assertThatExceptionOfType(IllegalArgumentException.class)
                .isThrownBy(() -> mock.callMethodWithException("second invoke"))
                .withMessage("test illegal argument")
                .withNoCause();

        doAnswer((Answer<String>) invocation -> {
            Object[] args = invocation.getArguments();
            MockTarget mock1 = (MockTarget) invocation.getMock();
            return "mock sayHello " + args[0];
        }).when(mock).sayHello("doAnswer");
        assertEquals(mock.sayHello("doAnswer"), "mock sayHello doAnswer");

        // 1.doNothing, 2. throw RuntimeException
        doNothing().doThrow(RuntimeException.class).when(mock).soSth();
        mock.soSth();
        Assertions.assertThatExceptionOfType(RuntimeException.class).isThrownBy(mock::soSth);
    }
```

verify

用于验证某个方法是否被调用，包括可以验证该方法被调用的次数，以及等待异步方法调用完成等特性。

常用句式 verify(mockObject [, times(n)]).targetMethod

```
@Test
    public void testVerifyInteractions() {
        // mock creation
        List mockedList = mock(List.class);
        mockedList.clear();
        // only clear() invoked
        verify(mockedList, only()).clear();
        verifyNoMoreInteractions(mockedList);

        // 此处不会抛异常，因为是 mock 的 list 对象，非实际 list 对象
        when(mockedList.get(1)).thenReturn("two");
        assertEquals(mockedList.get(1), "two");

        // using mock object - it does not throw any "unexpected interaction" exception
        mockedList.add("one");

        // selective, explicit, highly readable verification
        verify(mockedList).add("one");
        verify(mockedList, times(1)).clear();
        verify(mockedList, atLeastOnce()).add("one");
        verify(mockedList, atMostOnce()).add("one");
        verify(mockedList, atMost(1)).add("one");
        verify(mockedList, atLeast(1)).add("one");
        verify(mockedList, never()).add("never");
    }
```

verify 之 after 与 timeout

针对异步调用，我们可以通过 after 或 timeout 等待一定时间，来校验目标方法是否有调用，以及在此之后获取目标方法的返回值，作进一步逻辑校验。

- after 会阻塞等满时间之后再往下执行，是固定等待多长时间的语义。
- timeout 在等待期内，拿到结果后立即向下执行，不做多余等待；是最多等待多长时间的语义。

```
@Test
    public void testAfterAndTimeout() throws Exception {
        MockTarget mock = mockTarget;
        doCallRealMethod().when(mock).callMethodWait(anyLong());

        final long timeout = 500L;
        final long delta = 100L;
        // 异步调用
        CompletableFuture<Void> async = CompletableFuture.runAsync(() -> {
            try {
                TimeUnit.MILLISECONDS.sleep(timeout);
            } catch (InterruptedException ignored) {
            }
            mock.sayHello();
            mock.callMethod("test");
            mock.callMethod("test");
        });

        // timeout() exits immediately with success when verification passes
        // verify(mock, description("invoke not yet, This will print on failure")).callMethod("test");
        verify(mock, timeout(timeout + delta).times(2)).callMethod("test");
        // immediately success
        verify(mock, timeout(10)).sayHello();
        async.get();
```

```
        // after() awaits full duration to check if verification passes
        verify(mock, after(10).times(2)).callMethod("test");
        verify(mock, after(10)).sayHello();
    }
```

spy

spy 的官方定义是：partial mocking, real methods are invoked but still can be verified and stubbed

会调用被 spy 的真实对象的方法，但仍能被 Mockiton 所直接用于 mock 和 verify，也就是说在没有配置 mock 行为的情况下默认是调用被 mock 对象的真实方法。

- 句式 doXxx..when 当同一目标方法上定义了多个 mock 行为，后序 mock 可以覆盖前序 mock。
- clearInvocations 仅清理之前的调用。
- reset 会重置为初始状态(所有中途的赋值都会被清理掉)。

```
@Test
    public void testDoReturn() {
        // real creation
        List list = new LinkedList();
        List spy = spy(list);

        //optionally, you can stub out some methods:
        int mockSize = 100;
        when(spy.size()).thenReturn(mockSize);
        //size() method was stubbed - 100 is printed
        assertEquals(spy.size(), mockSize);

        // Overriding a previous exception-stubbing:
        when(spy.size()).thenThrow(new IllegalStateException("not init"));
```

```
        doReturn(mockSize).when(spy).size();
        assertEquals(spy.size(), mockSize);
        //Impossible: real method is called so spy.get(0) throws IndexOutOfBoundsException (the list is yet empty)
        Assertions.assertThatExceptionOfType(IndexOutOfBoundsException.class).isThrownBy(() -> spy.get(0));
        doReturn("mock data").when(spy).get(1);

        //using the spy calls real methods
        spy.add("one");
        assertEquals(spy.get(0), "one");

        /*
        Use this method in order to only clear invocations, when stubbing is non-trivial. Use-cases can be:
        You are using a dependency injection framework to inject your mocks.
        The mock is used in a stateful scenario. For example a class is Singleton which depends on your mock.
        Try to avoid this method at all costs. Only clear invocations if you are unable to efficiently test your program.
         */
        clearInvocations(spy);
        verify(spy, times(0)).add("two");
        reset(spy);
        when(spy.size()).thenReturn(0);
        assertEquals(spy.size(), 0);
    }
```

PowerMock

以上介绍的是 Mockiton 中常用的 API，而 PowerMock 则更强大，可以 mock static 方法，private 方法，final 方法，enum，构造函数调用等。

示例代码中用到的测试类如下：

```
public enum TypeEnum {
    Y("TRUE"),
    N("FALSE");

    private final String title;

    TypeEnum(String title) {
        this.title = title;
    }

    public String getTitle() {
        return title;
    }
}

public final class FinalTarget {

    public FinalTarget() { }

    public final String finalMethod() {
        return "Hello final!";
    }

}

public class StaticTarget {

    public static String firstMethod(String name) {
        return "Hello " + name + " !";
    }

    public static String secondMethod() {
        return "Hello no one!";
    }
```

```
}

public class PartialTarget {
    private String arg;

    public PartialTarget(String arg) {
        this.arg = arg;
    }

    public PartialTarget() { }

    public String getArg() {
        return arg;
    }

    private String privateWithArg(String arg) {
        return "Hello privateWithArg! " + arg;
    }

    public String privateMethodCaller(String arg) {
        return privateWithArg(arg) + " privateMethodCall.";
    }
}
```

类注解

在使用 PowerMockito mock static，private，final，enum，constructor 之前需要在测试类上加入如下注解：

```
@RunWith(PowerMockRunner.class)
@PrepareForTest({StaticTarget.class, PartialTarget.class, TypeEnum.class, FinalTarget.class})
```

static

PowerMockito.mockStatic 声明了要 mock static 方法的类:

```
PowerMockito.mockStatic(StaticTarget.class);
StaticTarget.firstMethod("xxx");
```

verify

值得注意的是，它的 verify 方法使用比 Mockiton 更复杂。

需要先声明一下验证目标类的静态方法再紧接着调用一下，表示待验证的目标方法:

```
PowerMockito.verifyStatic(StaticTarget.class); // 1
StaticTarget.firstMethod(invokeParam); // 2
```

也有类似于 Mockiton 的调用次数校验:

```
PowerMockito.verifyStatic(StaticTarget.class, times(1));
PowerMockito.verifyStatic(StaticTarget.class, Mockito.atLeastOnce());
```

private

PowerMock 模拟 private 方法 "privateWithArg" 的返回值并校验 "privateWithArg" 被调用的次数:

```
PartialTarget partialMock = PowerMockito.mock(PartialTarget.class);
doCallRealMethod().when(partialMock).privateMethodCaller(anyString());
PowerMockito.doReturn("mockResult").when(partialMock, "privateWithArg", any());
// *privateMethodCaller* will invoke method *privateWithArg*
String result = partialMock.privateMethodCaller("arg");
Assert.assertEquals(result, "mockResult privateMethodCall.");
```

```
PowerMockito.verifyPrivate(partialMock, times(1)).invoke("privateWithArg", "arg");
```

final

```
PowerMock 校验 mock final 方法
FinalTarget finalTarget = PowerMockito.mock(FinalTarget.class);
String finalReturn = "finalReturn";
PowerMockito.when(finalTarget.finalMethod()).thenReturn(finalReturn);
Assert.assertThat(finalTarget.finalMethod(), is(finalReturn));
```

enum

PowerMock mock enum，这里的 Whitebox.setInternalState 可以设置 TypeEnum fieldName=N 的值为给定的 mock 枚举:

```
String mockValue = "mock title";
TypeEnum typeMock = PowerMockito.mock(TypeEnum.class);
Whitebox.setInternalState(TypeEnum.class, "N", typeMock);
when(typeMock.getTitle()).thenReturn(mockValue);
Assert.assertEquals(TypeEnum.N.getTitle(), mockValue);
Assert.assertEquals(TypeEnum.Y.getTitle(), "TRUE");
```

constructor

构造器 mock 与 verify:

```
String arg = "special arg";
PartialTarget partialWithArgSpy = PowerMockito.spy(new PartialTarget(arg));
whenNew(PartialTarget.class).withNoArguments().thenReturn(partialWithArgSpy);
PartialTarget partialNoArg = new PartialTarget();
Assert.assertEquals(partialNoArg.getArg(), arg);
verifyNew(PartialTarget.class).withNoArguments();
```

完整示例如下：

```
import org.assertj.core.api.Assertions;
import org.junit.Assert;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.Mockito;
import org.powermock.api.mockito.PowerMockito;
import org.powermock.core.classloader.annotations.PrepareForTest;
import org.powermock.modules.junit4.PowerMockRunner;
import org.powermock.reflect.Whitebox;
import static org.hamcrest.core.Is.is;
import static org.mockito.ArgumentMatchers.anyString;
import static org.mockito.Mockito.times;
import static org.powermock.api.mockito.PowerMockito.doCallRealMethod;
import static org.powermock.api.mockito.PowerMockito.verifyNew;
import static org.powermock.api.mockito.PowerMockito.when;
import static org.powermock.api.mockito.PowerMockito.whenNew;

@RunWith(PowerMockRunner.class)
@PrepareForTest({StaticTarget.class, PartialTarget.class, TypeEnum.class, FinalTarget.class})
public class PowerMockTest {

    @Test
    public void testStatic() throws Exception {
        PowerMockito.mockStatic(StaticTarget.class);
        String mockResult = "Static mock";
        PowerMockito.when(StaticTarget.firstMethod(anyString())).thenReturn(mockResult);
        String invokeParam = "any String parameter";
        Assert.assertEquals(StaticTarget.firstMethod(invokeParam), mockResult);

        // Verification of a static method is done in two steps.
        PowerMockito.verifyStatic(StaticTarget.class); // 1
        // StaticTarget.secondMethod();// not invoked
        StaticTarget.firstMethod(invokeParam);// 2
```

```
// use argument matchers
PowerMockito.verifyStatic(StaticTarget.class); // 1
StaticTarget.firstMethod(anyString()); // 2
// atLeastOnce
PowerMockito.verifyStatic(StaticTarget.class, Mockito.atLeastOnce()); // 1
StaticTarget.firstMethod(anyString()); // 2
// times
PowerMockito.verifyStatic(StaticTarget.class, times(1)); // 1
StaticTarget.firstMethod(anyString()); // 2

// partial mocking of a private method & verifyPrivate
// PartialTarget partialNoArgSpy = PowerMockito.spy(new PartialTarget());
PartialTarget partialMock = PowerMockito.mock(PartialTarget.class);
doCallRealMethod().when(partialMock, "privateMethodCaller", anyString());
PowerMockito.doReturn("mockResult").when(partialMock, "privateWithArg", any());
// *privateMethodCaller* will invoke method *privateWithArg*
String result = partialMock.privateMethodCaller("arg");
Assert.assertEquals(result, "mockResult privateMethodCall.");
PowerMockito.verifyPrivate(partialMock, times(1)).invoke("privateWithArg", "arg");

// Final
FinalTarget finalTarget = PowerMockito.mock(FinalTarget.class);
String finalReturn = "finalReturn";
PowerMockito.when(finalTarget.finalMethod()).thenReturn(finalReturn);
Assert.assertThat(finalTarget.finalMethod(), is(finalReturn));

// enum
String mockValue = "mock title";
TypeEnum typeMock = PowerMockito.mock(TypeEnum.class);
Whitebox.setInternalState(TypeEnum.class, "N", typeMock);
when(typeMock.getTitle()).thenReturn(mockValue);
Assert.assertEquals(TypeEnum.N.getTitle(), mockValue);
Assert.assertEquals(TypeEnum.Y.getTitle(), "TRUE");

// verify New
```

```
        String arg = "special arg";
        PartialTarget partialWithArgSpy = PowerMockito.spy(new PartialTarget(arg));
        whenNew(PartialTarget.class).withNoArguments().thenReturn(partialWithArgSpy);
        PartialTarget partialNoArg = new PartialTarget();
        Assert.assertEquals(partialNoArg.getArg(), arg);
        verifyNew(PartialTarget.class).withNoArguments();

        // throw exception
        PowerMockito.doThrow(new ArrayStoreException("Mock secondMethod error")).when(Static
Target.class);
        StaticTarget.secondMethod();
        // AssertJ: Exception assertions
        Assertions.assertThatThrownBy(StaticTarget::secondMethod)
                .isInstanceOf(ArrayStoreException.class)
                .hasNoCause()
                .hasMessage("Mock secondMethod error");
    }
}
```

AssertJ

上面提到的 AssertJ 是 Assert 的一些功能增强，以流式编程的方式调用，下面介绍一些常用的用法:

- isIn,isNotIn 和 matches 用于断言匹配条件。
- filteredOn 可以针对 assertThat 中传入的参数进行过滤，类似 java8 中 Stream() 的 filter 方法。
- extracting 可以针对 assertThat 中传入的元组进行字段提取校验。
- assertThatExceptionOfType 和 assertThatThrownBy 可用于捕获预期的异常。

为了方便使用，AssertJ 还提供了几种常用的异常断言的包装器:

```
// AssertJ provides wrappers for common exception types
Assertions.assertThatNoException();
Assertions.assertThatIOException();
Assertions.assertThatNullPointerException();
Assertions.assertThatIllegalStateException();
Assertions.assertThatIllegalArgumentException();
```

示例如下：

```
import org.assertj.core.api.Assertions;
import org.junit.Test;
import java.util.Arrays;
import java.util.List;
import static org.assertj.core.api.Assertions.tuple;

public class AssertTest {
    @Test
    public void testAssertJ() {
        String title = "foo";
        AssertTarget assertTarget = new AssertTarget(title, 12, TypeEnum.Y);

        String msg = "Illegal Argument error";
        Exception cause = new NullPointerException("cause exception msg");
        Assertions.assertThatExceptionOfType(IllegalArgumentException.class)
                .isThrownBy(() -> assertTarget.throwIllegalArgumentException(msg, cause))
                .withMessage(msg)
                .withMessageContaining("Argument error")
                .overridingErrorMessage("new error message")
                .withCause(cause);

        Assertions.assertThatThrownBy(() -> assertTarget.throwIllegalArgumentException(msg, caus
e))
                .isInstanceOf(IllegalArgumentException.class)
                .hasMessageContaining("Argument error");
```

```
        Assertions.assertThat(assertTarget.getTitle())
                // as() is used to describe the test and will be shown before the error message
                .as("PartialTarget's arg is not match", assertTarget.getTitle())
                .startsWith(title)
                .endsWith(title)
                .contains(title)
                .isNotEqualTo("foo bar")
                .isEqualToIgnoringCase("FOO")
                .isEqualTo(title);

        AssertTarget target1 = new AssertTarget("testTitle", 12, TypeEnum.N);
        AssertTarget target2 = new AssertTarget("titleVal1", 16, TypeEnum.N);
        AssertTarget target3 = new AssertTarget("titleVal2", 18, TypeEnum.Y);
        AssertTarget target4 = new AssertTarget("titleVal3", 20, TypeEnum.N);
        List<AssertTarget> assertTargetRing = Arrays.asList(target1, target2, target3);

        Assertions.assertThat(target1.getNum()).withFailMessage("the num not matches").isEqualTo(1
2);
        Assertions.assertThat(target1.getType().equals(TypeEnum.N)).isTrue();
        Assertions.assertThat(target1).isIn(assertTargetRing);
        Assertions.assertThat(target4).isNotIn(assertTargetRing);
        Assertions.assertThat(target4).matches(e -> e.getNum() > 18 && e.getType().equals(TypeE
num.N));

        Assertions.assertThat(assertTargetRing)
                // extracting multiple values at once grouped in tuples
                .extracting("num", "type.title")
                .contains(tuple(16, TypeEnum.N.getTitle())
                        , tuple(18, TypeEnum.Y.getTitle()));

        Assertions.assertThat(assertTargetRing)
                // filtering a collection before asserting
                .filteredOn(e -> e.getTitle().startsWith("title"))
                .extracting(AssertTarget::getNum)
                .contains(16, 18);  }  }
```

真香

以上针对自己使用的 mock 单元测试的三板斧 Mockito + PowerMock + AssertJ 常用姿势做了小结。

- 利用 Mockiton 做常规类和接口的 mock。
- PowerMock 则可以 mock 静态方法，私有方法，final 方法，枚举，构造函数等。
- AssertJ 流式风格，增强 assert 判断逻辑和校验异常流程。

更多姿势等待大家在实操中继续解锁，利用这些姿势在后续的开发自测中可以更快速的做自我逻辑验证，而我再也不必等待每次项目开发环境的 10min 部署了。

艾玛，真香~

参考文档：

Mockito: https://site.mockito.orgPowerMock: https://powermock.github.io
AssertJ: https://assertj.github.io/doc

Q: 如何学习自动化测试?

作者 | 荣木
出品 | 淘系技术

作为一个测试人员，从业年期从事手工测试的工作是没有太多坏处的，当然，如果一直点来点去那么确实自身得不到提高，这时候选择学习自动化测试是一件很有必要的事情，一来将自己从繁重的重复工作中解放出来，从事一些更有挑战的工作，二来能积累技术知识，厚积薄发完成飞跃，那么技术新人该如何学习自动化测试呢?

看这篇文章或者认真查看回复的朋友，可能是刚从学校毕业，在开发/测试岗位间徘徊，或者是刚踏入软件质量行业不久，亦或是觉得手工测试的"点点点"太枯燥乏味，想在技术深度或者广度上做一些突破。

题主这个问题包含的范围挺广的，要完整的回答好，确实挺难的。

先简单回答下楼主的问题，后续在各个领域大家也可以深入交流下~

对自动化测试个人看法:

自动化是一个老生常谈的话题，也是一个软件领域非常有技术广度和技术深度的活动，特别是在大型软件的生命周期上。

个人觉得开展自动化测试的难度不亚于传统意义上的软件开发。

从产品角度来看:

质量领域本身要求从业人员要全面了解产品、有全局风险意识，例如：产品需求/设计阶段

能否发现设计缺陷、产品测试阶段能否发现深层次的 bug、产品运维阶段能否制定良好的灰度策略、快速发现、定位线上问题，甚至如何做好新/老系统线上过渡切换等等，这里面都有自动化测试可发挥的空间。

从技术的广度和深度来看：

1. 从技术广度来说，不同的技术领域的质量保障需要使用不同的技术（这些技术领域都有一些代表性的工具，但不一定能完全满足实际的项目自动化测试需求），例如有做 JUnit 接口测试的、有做 Web/App/桌面客户端 UI 测试的、有做性能测试的、有做用户体验测试的、有做 AI 算法测试的、有做 IoT 的、有做压测的、有做各种专项（如兼容性、安全、多媒体、网络）测试的等等，实在太多了……。

如果考虑到测试工具本身的可用性、系统性，除知道使用工具以外，可能还需要掌握一些基础开发技能，例如：Java/Node/Python 后台、React/H5 前端、或者 Android/iOS 客户端；

2. 从技术深度来说，想通过开发软件去测试另一个软件是否正常，本身就是一个很具挑战的事情，特别是在黑盒的状态下，举个例子，试想你能否开发一款自动化测试工具能够模拟人的意识形态，它能够对当前多如牛毛的 App 开展自动化测试，很多人此时想起了 Monkey、Appium、AirTest 或者 Applitools，其实这远远不够，因为目前并不具备解决场景构建甚至自我发现缺陷的能力，简单来说，还不具备"认知"App 的能力。这个想法不是天方夜谭，事实上很多人正在往这个方向努力 ing。自动化测试远远不只是在一个已有的工具上开发自己的脚本，达到所谓的一个通过率或覆盖率，更核心是思考如何在软件生命周期各个阶段提升产品研发效能及稳定性甚至用户体验。

技术新人如何学习自动化测试：

1. **首先简单了解下 QA 在软件研发迭代过程中的定位。**

传统软件使用较多的是瀑布模型。测试人员的活动区域是有限的，活动的时间区域主要是提测至上线前。

传统瀑布模型中，QA 发挥的空间比较有限，质量压力都集中在测试阶段。随着软件规模的扩大、部门职能的划分、敏捷迭代模式的发展，互联网或者大型软件项目绝大部分演变成了 DevOps：

DevOps 是软件文化上的一次飞跃，它强调产品、开发、测试、交付、运维各个环节的沟通合作，将敏捷的方式延伸到整个产品。从 QA 的角度也有了测试左移和测试右移的概念。

测试左移：

测试左移的思想是需求阶段、开发架构设计阶段或是未到系统测试或集成测试前就进行测试，目标是降低时间成本、减少风险，从用户角度描述产品行为、从技术角度建立好开发与产品需求的连接，防止产品设计上的雷或缺陷。这有利于减少无效代码的开发、以投入更好的时间在正确的产品上。也可以在代码编写阶段进行单元测试或覆盖率统计。日常工作中，QA 都期望只对修改的代码或受连带影响功能/需求进行测试，从而减少重复回归的工作量，即"精准测试"。但是实际上，往往得到开发同学的回复要么是"最好全回归或者核心流程全回归"，要么"是没关系的，就回归下 A 功能就好"（实际可能已经带雷上线了）。设想如果能够有个工具能够帮我们将需求与相关的代码调用栈联系起来，在相关代码依赖变动时都能够自动评估有效回归范围，可能是"精准测试"实现的一个方向（我相信业界应该已经有人在做了）。

测试右移：

测试右移简单来说是指产品上线以后开展的一系列质量活动。事实证明，在快速迭代以及产品复杂化、多样化的今天，几乎不可能做到 0 缺陷上线，当然，对硬件产品或涉及资金的产品而言，存在缺陷可能意味着产品召回或是资损，会给公司带来巨大损失，对于某些互联网产品而言，由于产品发布的天然优势，一般具备热修复、热发布能力，因此在时间和产品质量维度，可能会更强调快速上线，比如 facebook 就提倡灰度快速上线。因而如何监控产品的稳定性、第一时间发现线上用户问题、用户反馈并使问题及时得到解决、如何了解更好的用户需求（如 AB 测试）变成了 QA 在测试右移活动中的关注点。期间也有大量自动化测试可发挥的空间。

由此可见，QA 发挥的空间是在整个软件的生命周期的，DevOps 的理念也强调流程自动化，我理解的在各个阶段能够代替人工工作、提升测试效率的都可以称之为自动化测试。这也反过来要求 QA 具备更高的软件产品流程/风险意识以及更强的自动化理念、编码落地实践能力。

2. QA 做自动化测试应该掌握哪些技术？

说到具体的技术，其它回复也有提到，感觉整体太散了，初学者可能有点摸不到边，不知从哪里开始，个人建议顺序是这样的：

那让我们先修炼下最基本的内功吧!

软件工程&测试理论基础

各个公司产品形态迥异，因此也制定了不同的软件研发流程。大多数大公司都设置有运营、产品、视觉/交互、开发、测试、运维、技术支持、客服等岗位，应当明白各个角色的职责，以及了解整个产品运转的逻辑。至少应该了解所在公司的研发流程以及当前主流的研发流程（如敏捷开发 Scrum），并在项目过程中积极思考，形成自身的软件意识与理念。在校的同学可以多在网上找找资料，有个大概了解。个人理解，软件工程本身是一个浩大的工程，也在日新月异不断地向前发展，它需要长期积累、不断修炼内功，并在实际项目中实践驱动，从业 2 年、5 年、10 年、20 年都会有不同层次/深度的理解，自动化测试亦是如此。

关于测试理论基础这里不赘述了，网上资料一大把，搜白盒/黑盒、等价类、边界值等关键字就可以找到。

通用计算机基础（其实就是计算机专业相关的大学课程）

建议至少掌握一门编程语言（C/C++/Java/Python，推荐 Python，学习成本相对更简单一些）。相比于特定需求/领域的开发人员来说，测试人员对编码技术要求相对会弱化一些（当然并不意味着不需要极客精神、架构思想）。涉及到 Web、桌面 GUI、Android/iOS 的可以到具体应用再学习相应的框架。

- 掌握基本的数据结构以及在具体程序语言中的应用，例如：list、map。
- 掌握面向对象程序设计的基本思想。
- 掌握一种代码管理工具，如 git、svn。
- 掌握 Linux 的使用及基本命令使用，如：cp、grep、vi/vim 等。
- 掌握关系数据库的基本理论和关系数据库（如 MySQL）SQL 基本使用、NoSQL（如 Redis）的基本使用。
- 掌握基础的计算机网络理论，如 TCP/UDP 协议、IP 划分。

接下来，我们就需要站在巨人的肩膀上了。这部分可以根据实际需要进行学习，涉及的内容实在太多了，我这里主要从 App 自动化测试的角度给出一些工具使用、方向学习建议，大家搜关键字应该都能找到一些资料。

服务端：

1) 白盒单元测试：Junit(Java)、unittest(Python)、gtest(C++)
2) http 接口测试：Postman
3) 抓包工具：Charles、Wireshark
4) 压测：Jmeter，在大厂里面都会有特定的一些写好的工具可以使用。
5) 链路依赖分析：梳理应用间的依赖关系，提供压测模型，大厂里面也有一些工具可以使用。
6) 监控&日志分析：应用稳定性监控，如 qps、rt，服务器负载、cpu 监控等。日志分析这块可以做一些基于规则的错误日志监控、甚至基于 AI 的方式（如：机器学习）对日志大数据进行聚类、问题分析/定位。

客户端（Android/iOS/H5）：

1) UI：Appium、Macaca、Airtest
2) 性能（CPU/内存/帧率）：Android Studio、Instruments(iOS)
3) 稳定性：Monkey
4) 兼容性：各种云真机平台

事实上，即使非常熟练掌握了以上工具，也无法达到完全释放人力的目的，甚至在自动化实践过程中会存在各种各样的问题（例如如何针对具体的场景设计自动化用例、提升覆盖率、如何维护/构造测试数据、如何进行精确校验、如何提高执行稳定性、如何缩短执行时长、如何监控线上问题等等）。

这就需要我们更加深度的去了解产品形态、在已有工具解决不了问题时，怎么去用创新的思维看待各个阶段面临的问题、甚至创造工具，这已经不仅仅只是技术本身的问题了，而是如何去挖掘、思考问题、如何去运用技术的问题了。实际上自动化测试可以归纳为如下几个阶段，这也是近 2 年智能化测试的研究方向：

3. **另外，个人觉得目前市面上对自动化测试其实是存在的一些误区的：**

1) 认为自动化测试无所不能，只要有自动化，人工可以完全释放。对于复杂产品，目前来看，这几乎是不可能的。因此，目前市面上并没有类似"统一宇宙理论"的自动化测试工具。
2) 自动化测试没技术含量（测试没开发有前途），如果仅仅是对工具使用，没有创造力，那确实没有什么太高的技术含量。但是如果是在 DevOps 各个阶段发挥最大价值，个人认为比传统意义的开发岗位难度更高，并且可发挥空间更大。举个例子，能否创造一种工具，能够根据视觉稿或者 App UI 自动生成测试用例（啊？ 怎么可能，试想下人是怎么设计用例的，得益于 AI 技术的发展，这完全有可能，目前也有一些根据视觉稿生成前端代码的工具了）。我不觉得这是一个没技术含量或没有价值的自动化测试。近些年，质量领域的大会越来越多，大家也可以关注关注。例如 QCon、MTSC、TICA、Tid 等等。
3) 自动化覆盖率至高无上。这部分往往来自人们对自动化测试过高的期望，为了提升覆盖率，未考虑好 ROI，以至于南辕北辙，入不敷出。著名的测试金字塔给了最好的解释。
4) 人云亦云，泛而不专。相比于开发人员，个人认为 QA 开展自动化测试需要掌握的技术知识可能会更广泛，例如：开发人员可能专注于 Android、iOS 或者 Java 后端，亦或是某类 AI 算法，即对开发人员的要求是对某一技术掌握的足够深入，对 QA 来说，精力有限的情况下，可能又要懂点服务端、又要懂点客户端、又要懂点算法等等。再加上各种产品、技术形态不一，也衍生出了形形色色的工具和研究方向，初学者容易受到误导，不知所措。自动化测试和开发角色一样，也是一个逐步积累、实践的过程。

关于自动化测试确实有非常多的内容可以交流、学习，篇幅有限，先写到这里啦。以上内容是个人对自动化测一些理解，当然，如果继续往上走，到管理者，需要掌握的知识也远不止这些了。

音视频与图像技术篇

围绕淘宝直播和短视频核心场景，手淘在多年前就开始构建音视频核心技术竞争能力，编解码、低延时直播和直播看点等在 2020 年都取得了不错成果，未来我们会继续在核心场景加大技术投入，构建技术壁垒。

——阿里巴巴资深技术专家 梁舒

技术经典总结

支撑每秒千万图像访问的鹿班 PICASSO 实时渲染引擎

作者 | 郑行涛(轩怒)
出品 | 淘系技术

在读者受益：

1、鹿班 PICASSO 实时合图引擎因何而生

2、实时合图引擎如何支撑每秒千万图像访问

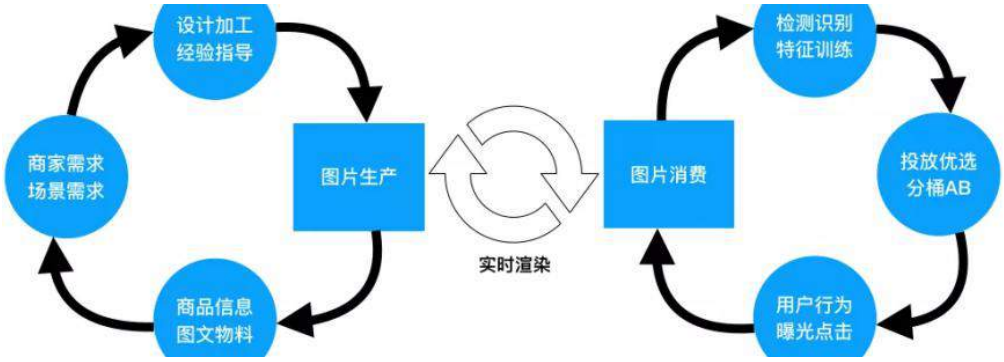
3、实时合图引擎应用场景介绍

缘起

在个性化的公域触达和人货场的私域运营中，图片作为一个信息表达的载体，在过去一直是先生产，后消费。在这个生产-消费的过程中，存在两个循环：

- 商家与设计师的图片制作生产循环；

- 用户与商品的图片投放消费循环；



在生产侧，优秀的设计创意可以对用户注意力带来显著的提升，但由于缺乏标准化的生产机制，难以形成规模化的供给，更多的依靠作为个体的设计师的经验知识来优化生产供给。

在消费需求侧，相比于理解结构化的商品信息而言，机器去理解非结构化图片数据，在当下仍然有诸多的限定与不确定，同时由于无法干预生产，从而很难构建出一套正反馈机制，只能以量取胜。

我们希望能够利用新硬件与新软件的红利，构建生产与消费之间的数据新闭环，形成生产-消费双循环联动与共振。

基于以上出发点，我们在集团图片下行核心链路上构建起实时图像渲染服务，将原先面向存储的图片基础设施增加上面向理解生成的能力。

存储效率与设计能效双提升

营销图片的"千人千面"带来用户转化提升的与流量宏利的同时，给图片的存储成本与设计的生产能效上也带来了巨大的挑战。

在智能个性化算法大规模应用的今天，相同的商品在不同的"人货场"中有不同的展现需求。因"千人千面"普及而产生的大量设计脉络上大致相同仅局部元素不同的冗余图片，带来了存储成本与设计成本的爆炸性上涨。

如上图所示，同一套设计在不同年龄与性别的用户群体中展现的是配色与文案利益点的不同，而在不同尺寸的容器组件上又展现出图像分辨率的不同。

由此看来设计与个性化算法做笛卡尔积带来的图片数量的指数级别增加，若按照传统存储方式保存，财务成本是不可接受的；按照传统用人做图的生产模式，设计成本也非常高昂。

随着实时图像渲染服务的诞生，以上两个问题都迎刃而解。对于个性化算法产生的爆炸性的图片数据，仅需存储最原始的一份结构化设计，而实际展示的图片在访问时实时渲染生成，用高性能计算换取存储成本。

另一方面，实时渲染所存储的图像结构化的数据相较于二进制流数据更易于被设计端所理解。图像分元素控制更能感知用户对设计元素级的敏感性，从而更高效反哺智能化生产。

在大规模个性化算法应用下，实时图像渲染成为了精准控制图像元素级展现的必要条件，而只有做到了图像元素级的控制才能更好感知"人货场"的数据输入进行智能化表达，从而提升用户转换。

挑战

为完成面向理解图像基础设施的全新升级和进一步提升存储与设计效率的目标，我们做出了在集团图像下行核心链路上搭建个性化实时渲染服务的决定。

该服务的主要职责为每次用户访问图像资源时，将结构化的设计数据与个性化需求相匹配，

实时渲染出图像返回给用户。实时图像渲染服务作为图片中心的基础设施服务，既要有设计增量效果又要满足高吞吐、高可用与低延迟(处理<100ms)，这些都给我们系统设计带来了巨大挑战。

高吞吐低延时

按照传统做法，用普通 CPU 机器渲染首焦的 banner 图片，至少需要 300ms 的延时，单机吞吐无法超过 100QPS，这样的性能只能做到图片离线生产，无法支撑实时渲染。

为了满足高吞吐低时延的需求，我们引入了 FPGA 异构计算的架构，将图像编解码与图像融合烧写进 FPGA 板卡中，极大化的利用 PCIE 带宽及异构的优势，增加单机吞吐，降低单位延时。

目前在 88Cores 2.5GHz / 340GB RAM 双板卡 4 VU9P FPGA 的异构机器上，用鹿班模版库历史数据作为压测标准（输出手淘上应用最广的 heic 格式的图像，平均 7 个图层，平均大小在 800*800）最终能单机吞吐稳定到 3000QPS，耗时控制在 100ms 内，相比普通 CPU 机器性能提升超 30 倍。

设计效果全支持

设计效果的丰富度与精细化程度很大程度上影响了图片的表达能力，进而影响图片的转化率。

为了完美还原渐变、描边、阴影、旋转等渲染效果，解决多语言、多字体带来的设计效果上的飞越，我们引入了 skia 渲染引擎，以上为渲染的示例图。skia 渲染引擎能在不影响硬件

性能的情况下生成华丽的设计效果，满足视觉设计上的大部分需求。

缩放&质量&压缩后缀全支持

在手机淘宝与网页淘宝中，访问的所有图片链接都并不是图片的原始格式，而是经过对应后缀进行压缩的或更高压缩率编码的图片。

其中应用最广的高压缩率编码格式 HEIF ，编码的计算复杂度是 JPEG 编码的 10 倍以上，用普通的 CPU 机器来输出这种格式图片效率极低。

尺寸后缀	请求数	占比	TOP N占比	
460x460q90.jpg_.heic	252411355	0.142567635695095		
320x320q90.jpg_.heic	159394373	0.090029542853617		
600x600q90.jpg_.heic	124710335	0.070439214621280		
760x760q90.jpg_.heic	124698585	0.070432577955828		
460x460q90.jpg_.webp	83571050	0.047202817048609		
320x320xzq90.jpg_.heic	64731290	0.036561694979187	0.457233483153618	
-	61849959	0.034934254136342		
320x320q90.jpg_.webp	59496078	0.033604728969467		
760x760q90.jpg_.webp	41622637	0.023509405701994		
460x460q75.jpg_.heic	29466341	0.016643255090308	0.56592512705173	
200x200xzq90.jpg_.heic	27630712	0.015606450361205		
600x600q90.jpg_.webp	26998528	0.015249378555920		
200x200q90.jpg_.heic	26851416	0.015166286374816		
460x460xzq90.jpg_.heic	22873246	0.012919326085358		
460x460q90_.heic	21472297	0.012128038440397	0.636994606869429	

根据统计，每一份原图对应的不同后缀输出格式有 2400+ 种，TOP6 格式覆盖率 46%，TOP15 格式覆盖率仅达到 64%。可以看出在实际的图片应用场景中，所有访问都是带有后缀请求的。

我们支持了集团图片中心的所有缩放参数与格式参数（包括 heic&webp 输出）。由于在 FPGA 中操作，返回缩放与高压缩编码处理过的图片不增加耗时，保证生成的链接在全集团的端与页面上无差别适配。

异地多活与全球部署

在个性化场景下进行实时渲染，CDN 命中率不会很高，因此保证源站的稳定性就成了重中之重。如何避免因单地网络或存储集群不可用进而影响业务整体可用性，如何降低网络流量在垮地域中对穿都是亟须解决的问题。

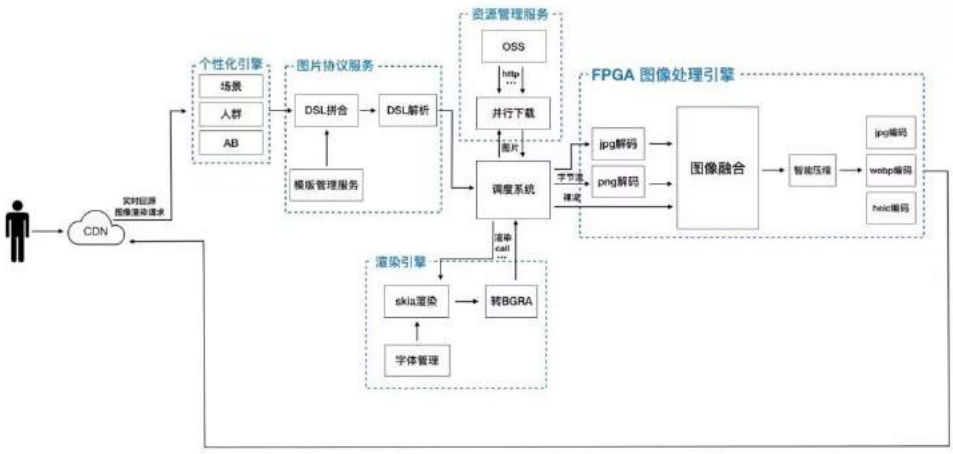
为了响应集团国际化的需求，我们的服务必须在全球都有部署。实时渲染服务必须让存储和计算在一个机房内完成闭环，但又要考虑到数据垮地域复制的安全性。

实现细节

实时渲染服务架设在 CDN 与底层 OSS 存储之间，是一个重计算、低延时、高吞吐的分布式系统。

用户首次访问个性化图片会回源到实时渲染服务源站取图，此时我们会将渲染好的图片经过高压缩码率后吐还给 CDN；之后相同访问请求在更贴近用户的 CDN 边缘节点上就能得到返回。

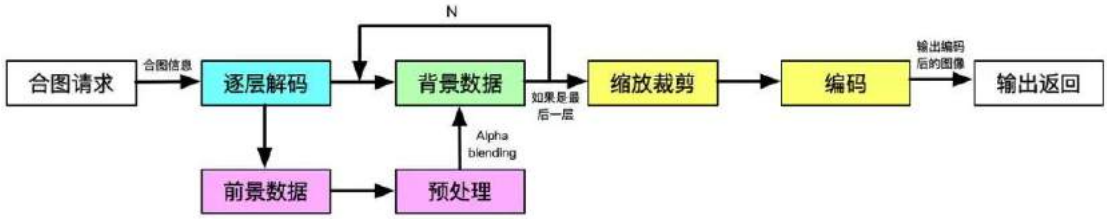
为满足国际化需求，实时渲染服务在全球多地均有部署，保证全球时延无差别。各地服务的存储数据都在本机房内流转，避免跨机房访问数据。每个地域服务的模块划分在下面大图中能够体现：



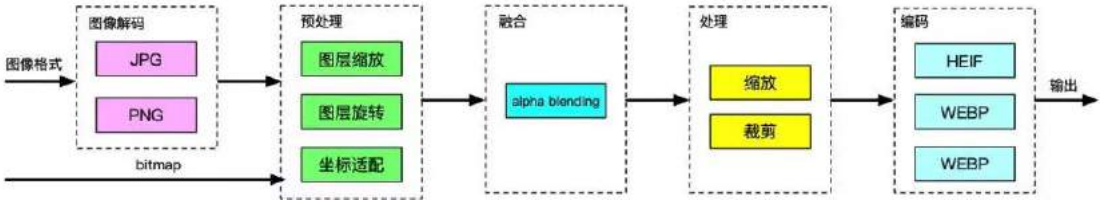
每个地域的流量服务入口都来自于 CDN 的回源请求，按照调用顺序依次经过个性化场景引擎、图片协议服务、资源管理服、渲染引擎、FPGA 图像处理引擎这五大模块。接下来会依次讲解关键技术。

基于 FPGA 的高吞吐低延时实时融合方案

基于智能算法，商品在不同"人货场"的个性化呈现，实质上是将不同的标签元素处理后生成不同的图片。以图层为单元定义合图结构，实现上采用图层依次处理的方案，如下图所示，即循环执行解码、预处理、合图，待合图完成后，再进行用户请求参数（包括分辨率、目标编码格式）的匹配。



图片处理方案实现



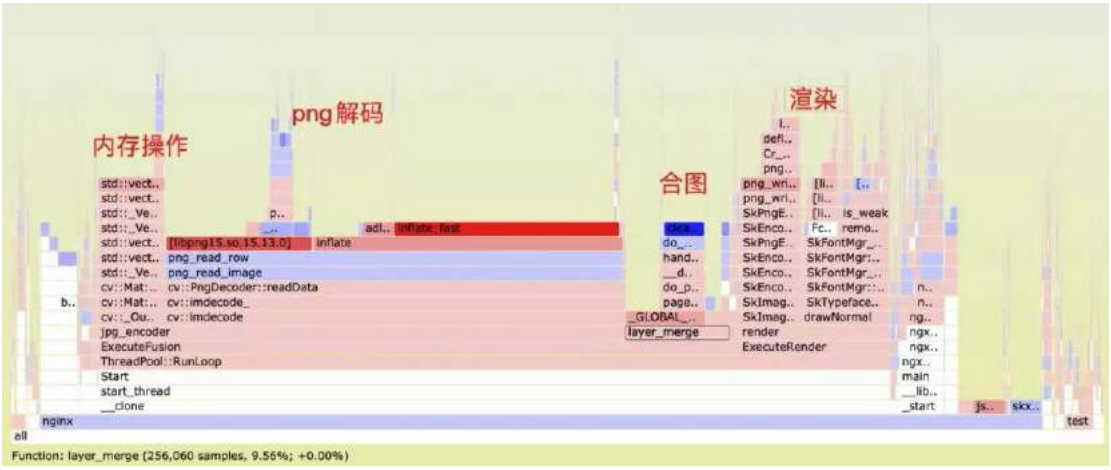
技术挑战与选型

传统的离线生产逻辑将图像合成后以图像格式存储在 OSS，下载端从 OSS 拉图后进行缩放、转码，而在新的实时图像渲染系统中，合图及目标格式的适配在同一个流程中完成，做到用户请求实时处理，支撑"人货场"的个性化表达。

复杂功能的实现带来的是计算量的增加，合图相比于传统的图像处理，单次请求单次合图请

求输入图层数量多（线上平均 6-7 个图层），数据量大，在单次请求内完成多图层的处理，不论从资源占用，还是处理耗时，对服务端来说都存在很大的挑战。

进行 9 个图层的合成，使用软件方案处理，输出为 850x850，吞吐及 RT 情况如下，对于 JPG 输出，从火焰图可以看到，CPU 主要消耗在解码、渲染等重计算处理上，如果使用软件方案来处理，要支撑 30w QPS 的目标处理请求，需要 3750 台机器，成本无法接受，这种重计算逻辑的硬件化加速显得十分必要。



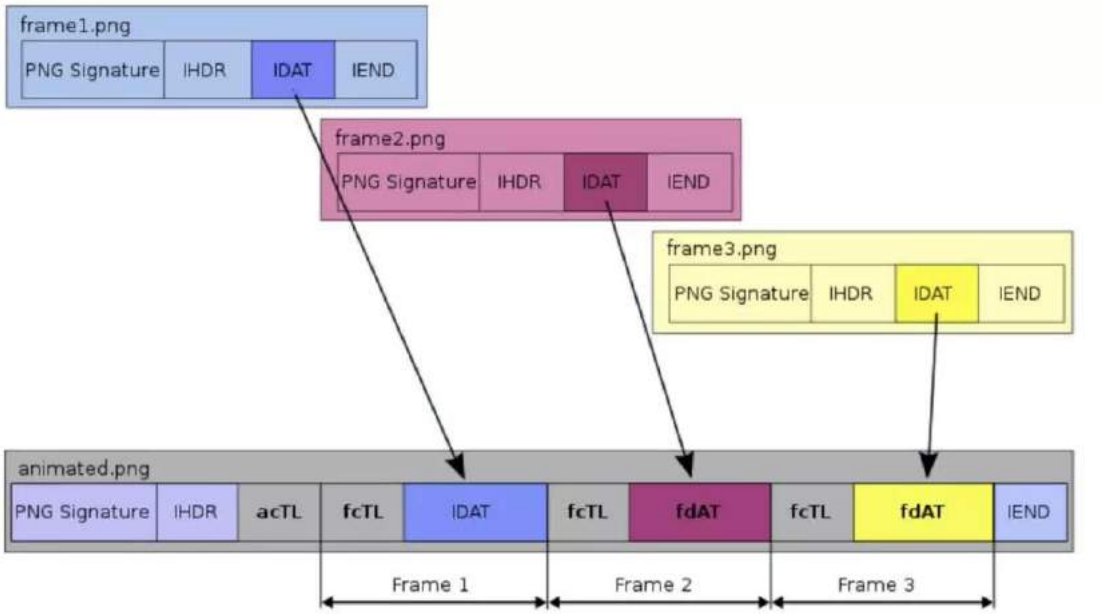
FPGA 由于其硬件并行加速能力和可编程特性，在传统通信领域和 IC 设计领域大放异彩，基于 line buffer 的实现，非常适用于滑窗插值、块匹配、MEMC 等图像处理算法的并行及流水线加速，借鉴图片中心在下行链路使用 FPGA 进行转码的经验与阿里云 AIS 合作，基于异构 FPGA 机器，将重计算的逻辑 offload 至 FPGA 来解决编解码及复杂图像处理问题。

FPGA 实现合图与编解码全流程

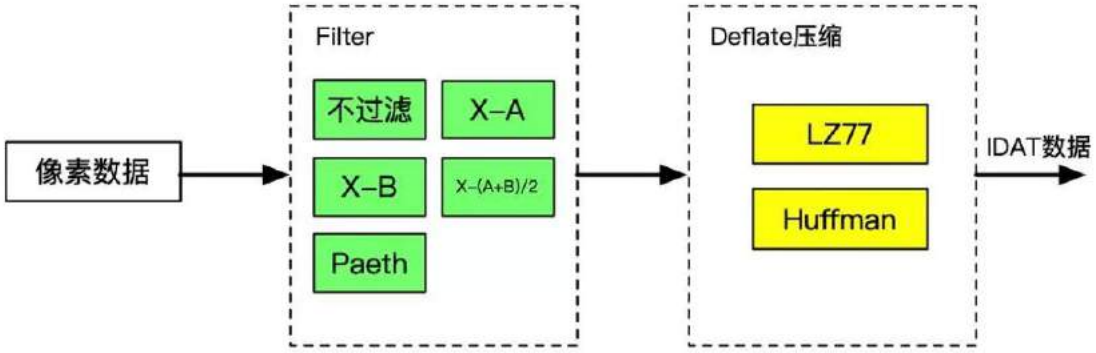
合图的输入流主要包括 PNG、JPG 及 bitmap 数据，对于 PNG 和 JPG 数据，需要先进行解码，针对不同的图层类型自适应选择解码器解码，对于 JPG 格式数据，FPGA 高性能解码方案在 2019 年双 11 时我们已经上线，对于 850x850 的图像，2ms 即可完成解码，而软件 SIMD 加速版本 turbojpeg 需要 10ms 左右。

由于 PNG 带有透明通道，可提升模板元素的呈现效果，设计上对 PNG 情有独钟，在解码端 PNG 的处理量也是最大的。

PNG 的压缩过程是完全无损的，压缩过的文件可以准确的还原出原图，可封装 RGBA8888 带透明的真彩色数据，并且是一种可扩展的封装格式，PNG 文件格式里面包含不同的区块(chunks)，各个区块带有不同类型的数据。典型的 PNG 数据包括四部分，而在此基础上，增加 acTL（动画控制块）、fcTL（帧控制块）、fdAT（帧数据块）即为 APNG 动图格式。



IDAT 的压缩过程主要包括 Filter 和 deflate 两部分，Filter 即对像素做过滤，PNG 编码使用差分对原始像素数据进行 Filter ，该过程无任何压缩损失，并且完全可逆。



PNG 中的 Deflate 与 gzip、zlib 中的 deflate 原理一样，进行像素数据的无损压缩。最终我们与 AIS 合作将解压缩和 Filter 在 FPGA 中实现，大大加速的解码流程，提升吞吐并降低延时。

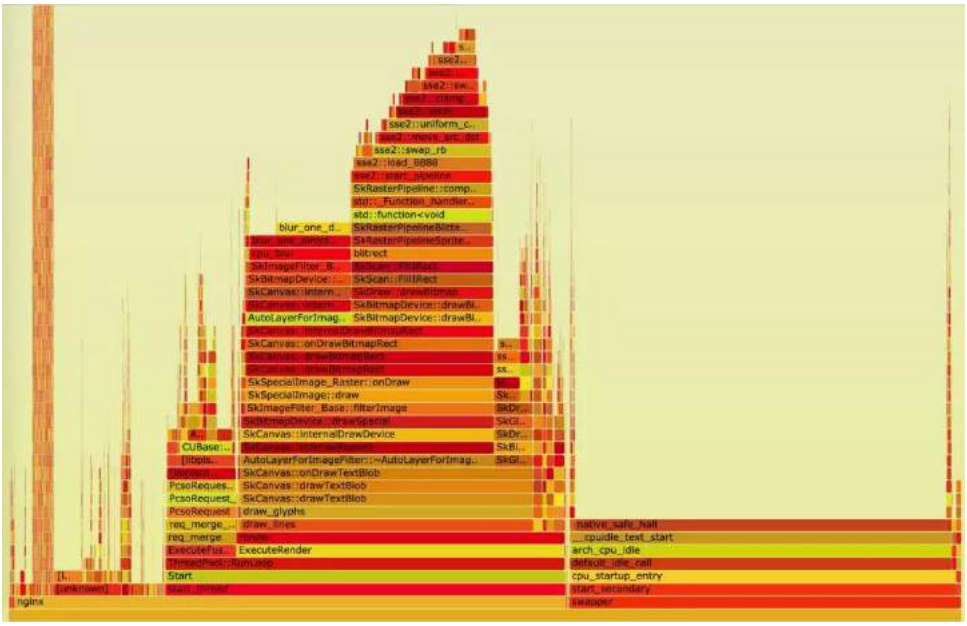
为支持丰富的图层表达，在图层预处理中实现多种图层缩放、裁剪、旋转、坐标适配逻辑，大大拓展图层元素的效果表现力。合图本质为 alpha blending，即根据像素级透明度进行前后景的叠加，有大量像素级的乘加和操作，对于 9 个图层的叠加，最初直接采用像素操作方案，耗时 40ms，后续使用 SIMD 加速，一个指令周期可进行 64 个像素点的操作，大大提升 CPU 的处理效率，最终在服务端，加速后 9 个图层的耗时控制在 1 个 ms。

为支撑不同分辨率、不同质量、不同细节清晰度的请求，我们在服务端实现了大量的图像处理逻辑，包括各种各样的缩放操作（长边等比缩放、短边等比缩放、缩放裁剪、缩放不裁剪）、质量调整、锐化，最终输出调整后的结果。

手淘上目前常用的图片格式为 HEIC/WEBP，而 PC 上常用的格式为 JPG/WEBP ，为支撑淘系业务，需做到支持不同编码格式的输出。从压缩收益来看，HEIC>WEBP>JPG，高的压缩收益带来的是复杂度的提升，相比于 JPEG 编码只需要做中心化、DCT 变换、量化、熵编码，HEVC 帧内编码复杂度非常高（infra pred 包括 35 个预测方向），是同等大小的 JPEG 编码复杂度的 10 倍以上。最终我们在硬件上支持 HEVC/JPG 编码，将软件 1000ms 耗时的处理降低到 2-3ms。

经过 FPGA 硬件化加速，我们 HEIC/JPG 输出链路达到了 3000 QPS 的吞吐，全流程的融合处理 RT 控制在 40ms 内，不加文字渲染合图结果如下，依次进行图层叠加，直至图层合并完成，再进行编码输出。

从优化后的火焰图可以看到，编解码等重计算逻辑基本都已 offload ，目前 FPGA 的资源平均利用率不到 30%，而 CPU 主要消耗在 Render 上，未来性能进一步的优化，主要在于 render（filter 处理重计算）的卸载，这部分处理加速后，整机吞吐将会大大提升。



基于 skia 的 GPU 渲染

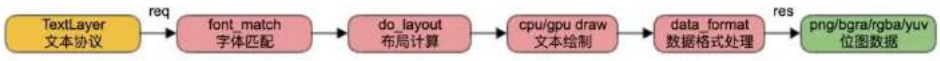
文字渲染业务上功能点主要考虑两个方面：排版、效果，排版指文字的水平/垂直排版、字号自适应缩放的计算，效果则围绕文字渐变、描边、阴影等常用的效果还原。

在技术上文本渲染不仅要支持高并发&低延迟，还要满足不同 os 端的跨平台特性，实时图像渲染支持 android、ios 下的文本渲染的集成，以解决端侧文字效果短板。

为什么选择 skia ？文字渲染有很多开源技术如 Skia、Cairo、Rusttype 等，以下是对部分文本渲染技术的分析：

技术选型	绘制引擎	语言规范	跨平台	排版效果	使用范例
Skia	2d图形绘制引擎	c++	支持跨平台 mac/linux/android	复杂	chrome、android图形引擎
Cairo	2d图形绘制引擎	c	支持跨平台	复杂	firefox图形引擎
Rusttype	类freetype字体引擎	rust	不支持跨平台	简单	

从业务和技术出发点，skia/cario 都可以满足需求，考虑到鹿班当前渲染技术&编辑器基于 puppeteer，而 puppeteer 图形绘制是基于 skia，并且 skia 可集成 webassembly 利用 CanvasKit 在 web 端的渲染能力使得在技术架构上确保前后端的同构。基于 skia 的文字渲染流程如下：



字体匹配

实时图像渲染服务作为公私域全场景的图像基础设施，文字渲染最基本的要求不能出现下图(左一)的 tofu 问题，面对国际化多语言、不同 os 的字体引擎差别等问题，文字匹配在加载指定协议字体时会校验字符是否存在，不存在的字符会走统一的兜底接口，不同 os 端实现各自字体的管理逻辑。

布局计算

在图片设计中，文本的排版影响着整体布局的美观度，从简单的单行文字排版到多行文案排版、从水平排版到垂直排版、非 cjk 文字 90°旋转垂直排版、字号缩放、自动换行等，在 html 中借助 css 布局样式可以很方便地实现以上布局排版，基于 skia 之上实现了自己的文本布局管理器。

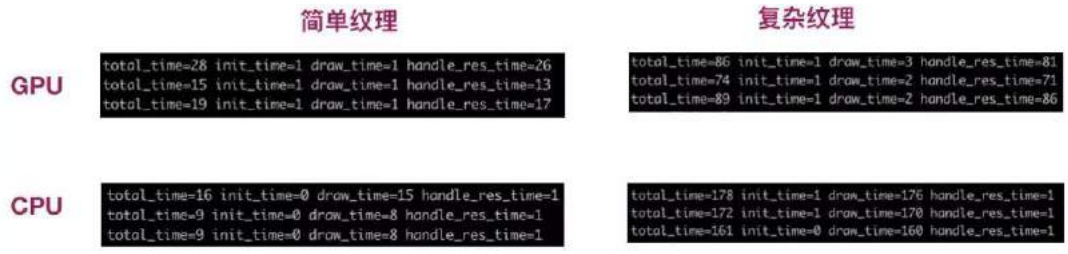
上述文本布局单从文本角度上看，解决了文字元素域的布局问题，在业务中我们常遇见一些广告 banner 拓版需求，其本质也是对图片元素(位图、文本)进行重新布局排版问题，于是我们在此之上实现了拓版的布局自适应引擎。

拓版并不是简单的图片缩放，因为拓版的尺寸通常是非等比的(相对于原尺寸)，图片缩放只是处理位图尺寸等比缩放的一种方式，拓版强调的是位图元素等比、文本元素弹性的尺寸&坐标调整。

点击查看动图演示

cpu/gpu 文本绘制

skia 默认使用 cpu backend，我们分别在 cpu 和 gpu 场景下做了对比测试，在简单文本(无效果加持)图层渲染时，cpu 和 gpu 在性能上没有太大差别，甚至 cpu 还具有一定优势，虽然 gpu 在纯纹理绘制时具有很大优势，但在 readPixels 时 gpu 显存到 cpu 主存下的 io 处理很耗时，所以 gpu 加速用在复杂纹理(阴影、blur)处理或者批量图层渲染时优势才足够明显。

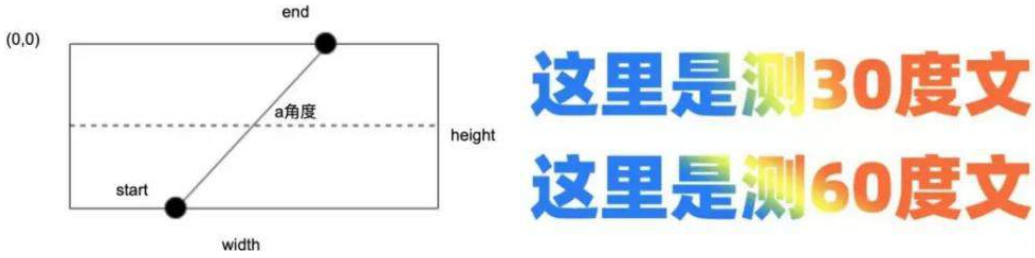


效果还原

效果还原其中效果指的是描边、渐变、阴影三种常见的文本设计效果，还原则对比的是设计师在 ps 的设计效果。skia 支持 stroke/gradient/shadow 底层 api 能力，但在表达 ps 效果上有些吃力更多地是见招拆招，如文字描边 ps 支持内描边、居中描边、外描边，skia 描边是没有类型区分的。

- 描边，skia 对描边的效果的支持需要绘制 2 层纹理叠加而成，叠加的顺序决定了描边的效果：

- 渐变，文字渐变效果支持线性、径向两种，径向渐变较简单不在此赘述，这里简要说下线性渐变角度变化对于 start/end 两点坐标计算，参考示意图如下：



- 阴影，文字阴影主要理解三个变量：dx/dy 阴影的偏移量，blur 阴影模糊半径，注：文字阴影效果对于 cpu 绘制下的性能有很大影响：

性能优化

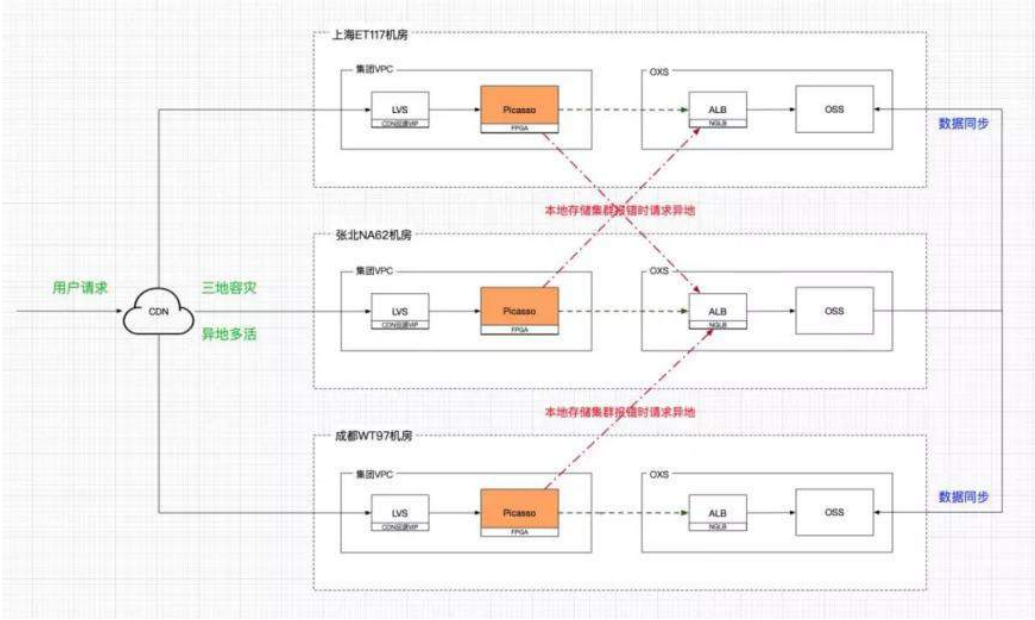
在初期过程中，skia 文本绘制区域是整个画布尺寸而一个文本图层区域相对于画布来讲小得多，全画布绘制使得内存开销更大数据传输也变得更加低效，在经过调整输出最小文本区域时性能提升不少。文字渲染默认返回 png 图片数据，火焰图中看出编码占用了 5%cpu 资源。将输出改为 BGRA 格式后性能有了 30%的收益。通过增加字体缓存的方法，文字渲染性能同样获得了收益。

高可用架构和国际化部署

高可用架构

实时渲染作为直接面向 C 端的链路，对加载时长和稳定性的要求非常高，为降低加载时长、保证用户体验，采用 CDN 缓存 + 实时渲染源站 的二级分层架构：

- 在满足业务实时变化的诉求下，通过 CDN 缓存降低计算量，同时利用其分发网络来降低网络时延；
- 如果未命中 CDN 缓存，则直接请求源站并实时返回处理结果。

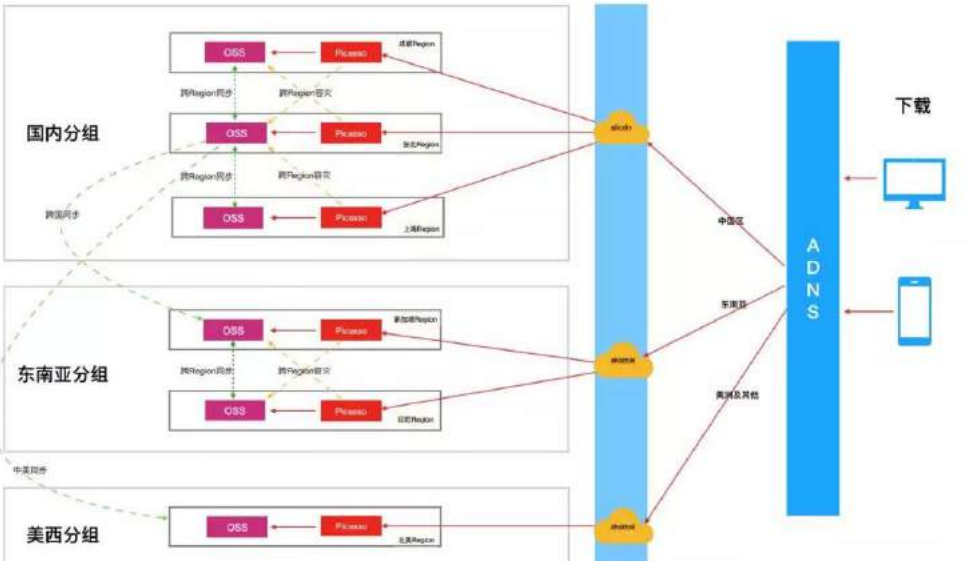


在个性化场景下，CDN 缓存命中率有限，因此，源站高可用成为整个链路的重中之重。因源站对网络、存储的强依赖，为避免单地网络或存储集群不可用影响业务整体可用性，实时图像渲染集群采用三地部署、异地多活模式：

- 渲染的原始数据在三地存储集群间异步复制，计算集群从本地存储集群读取所需数据；
- 当某地出现问题时，通过调整 CDN 回源策略实现异地容灾逃逸（秒级生效）；
- 考虑到存储集群作为一个有状态服务，单机不可用会造成请求成功率下跌，因此，针对单个请求使用跨地域容灾来保障整体成功率。

国际化部署

实时渲染作为基础服务，考虑到赋能 AE、Lazada 业务，进行了国际化部署；在国际化部署中，因实时图像渲染对加载时长、网络的要求较高，所以，必须让存储和计算在一个机房内完成闭环，整体架构如下图：



考虑到模板在电商业务中的复用性及其读多写少的业务特性，我们将设计师模板做了全球同步：3 个分组、5 个 Region；考虑到存储和复制成本，用于合图的商家数据 Native 化：只做分组内容灾、不做分组间同步。

基于设计领域知识的个性场景化引擎

相比文字，图片更容易吸引用户的注意力。但不同的人对图片的偏好是不一样的，如上右图所示，单就牛皮癣而言，不同年龄段、不同消费能力的用户会呈现明显的喜好差异。

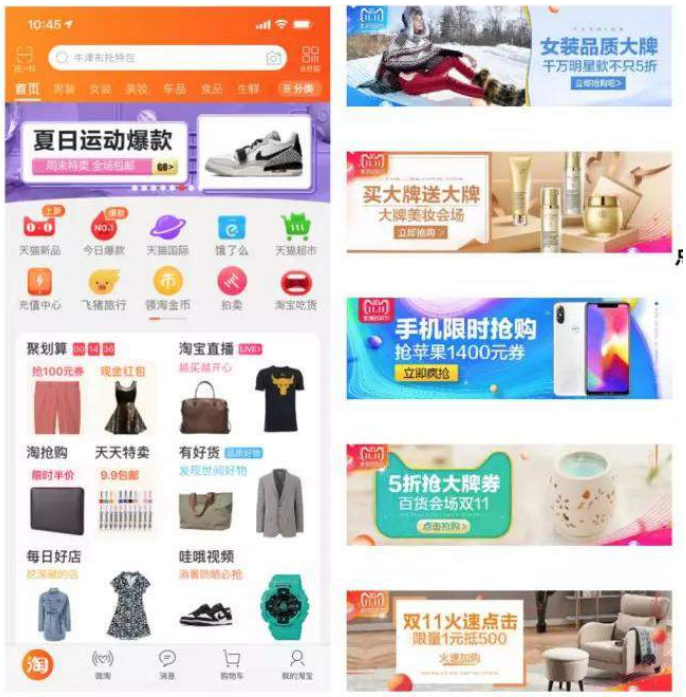
此外，不同的导购链路和场景，用户对图片的注意力也可能产生差异。在主搜和首猜两大场景，我们通过深度测图产品，发现很多典型的 case ，同样的商品图片，仅有细微的设计差异，但在大数据的验证下，持续体现了完全不同的转化效率。由此可见，研究用户在图片上的注意力个性化，一定会给业务带来巨大的增量。

我们经过长期的业务积累，沉淀了一套基于设计领域知识的个性化引擎，结合专家知识+数据驱动，通过感知场景、人群、商品的特征，给出最合适的图片内容，实现图片效率的最优。

应用场景

手淘首焦

在手淘首焦商品个性化基础之上进行图片实时个性化投放，基于设计数据+用户注意力偏好构建图片优化生产模型，提高 banner 转化率。



主图打标

对商品主图增加官方的个性化模板，用智能算法填写符合用户画像的文案利益点，达到统一管控主图氛围的目的。做到公私域展现统一打通，首猜、搜索与详情展示图片实时渲染。

主图打标业务上要求在大促时候需要对图片进行指定氛围管控，利用实时图像渲染+CDN缓存批量刷新，做到了双十一零点千万图片在同一时刻的氛围变化。

店铺

通过动态实时渲染技术与个性化推荐算法结合，支持了"千人千面"的店铺页设计，服务了近百万商家，日均曝光超千万次。

在于店铺的合作中开拓出长条形模版分片渲染的能力，适配手机上的内存淘汰策略，将长条形模版切分成多快渲染，使端上体验如丝般顺滑。

lazada&AE 国际化输出

在与 Lazada、Ali Express 等国际站点的合作过程中，实时图像渲染服务不仅完成了全球化部署，还支持了全世界各地的多种语言以及不同国家的设计风格。

让机器读懂视频：亿级淘宝视频背后的多模态 AI 算法

作者 | 阮彤枭（晓何）、燕保明（元年）、王琳（有邻）
出品 | 淘系技术

本文内容大纲：

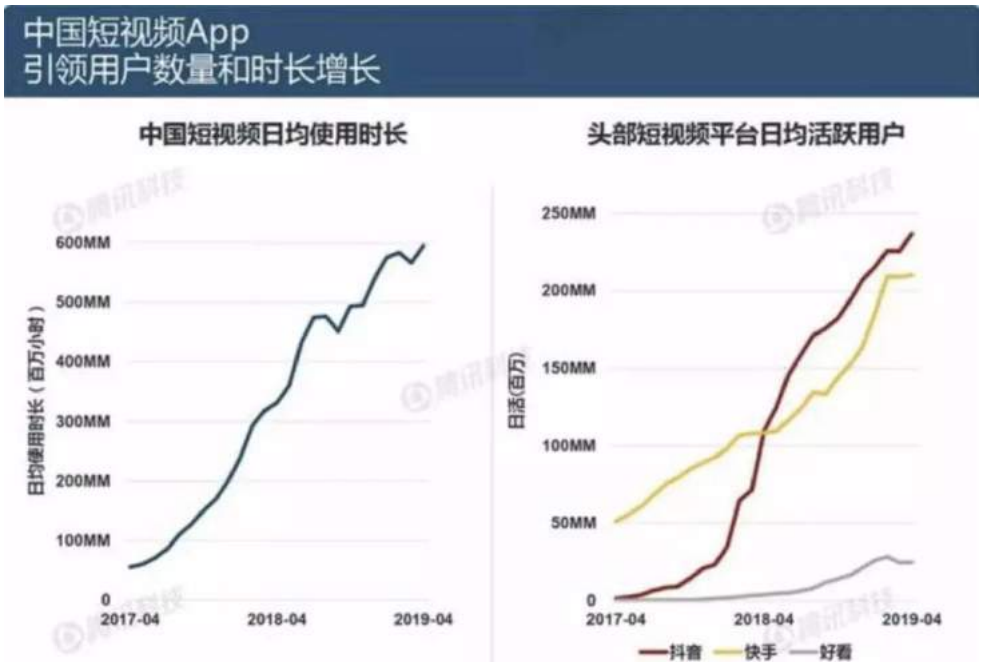
1、多模态在视频分类算法中的应用探索

2、视频分类算法中的层次化分类器的设计

3、总结和展望

背景

随着 4G 的普及和 5G 的推出，内容消费的诉求越来越受到人们的重视。2019 年互联网趋势报告指出在移动互联网行业整体增速放缓的大背景下，短视频行业异军突起，成为"行业黑洞"抢夺用户时间，尽管移动互联网人口红利见顶，新的增长点难以寻觅，但中国短视频人均使用时长及头部短视频平台日均活跃用户均持续增常(如图 1 所示)。



图一

在淘宝，短视频业务一直以来都是非常重要的业务，是淘宝 app 从单一的商品导购 app 走向商品导购+内容消费的多元化 app 的关键所在。相较于单一的商品导购，商品导购+内容消费的模式有效增加用户粘性，提高用户的停留时长，最终获得 GMV 的持续增长。不仅如此，2019 年视频营销发展趋势白皮书指出目前视频内容的转发量已达到图文的 12 倍，视频营销已经成为品牌最爱的营销方式，使用视频营销比不使用视频营销收入增长速度快 49%，且从搜索获得的网站流量多 41%。现如今淘宝每年新增内容数达数十亿，其中视频数占比持续提升，预计到 2022 年视频的占比会超过 50%。如何对规模如此庞大的视频进行内容化理解，高效赋能视频运营和个性化分发变得极为关键。

视频类目体系

视频内容化理解的首要问题是构建一个完备的、层次化的类目体系，相比于传统的淘宝商品类目体系，视频类目体系需要解决两大问题。

- 一是提高运营的精细化能力，提供内容的盘货和定向生产、以及快速搭建会场的能力；
- 二是针对新生产的视频提供冷启动能力，提升分发的效率。

虽然淘宝的商品类目体系可以部分代替视频类目体系的功能，但会导致短视频业务的用户心智和商品导购业务同质化，因此构建一套属于短视频业务自己的视频类目体系已经迫在眉睫。有了构建完成的视频类目体系，另外一个更重要的问题是产出高效的视频分类算法，用于对海量的视频进行标签生成。为了获得好的点击效果，视频生产者往往会采用吸引眼球的封面图或者标题，但是和视频本身的内容关联性很弱。此外，淘宝的视频天然和商品有强关联性，如何利用好商品的信息也非常关键。

不同于抖音美拍等其他视频的内容体系从新闻、科技、金融等领域做分类划分，淘宝的视频类目体系从商品导购、产品功能展示、商品知识获取等角度出发，由行业经验丰富的运营进行设计，包含了 30+一级类目和 150+二级类目，其中一级类目是对视频的领域的划分，例如服饰/家居日用/美食/萌宠等，二级类目是在一级类目的基础之上对视频的子领域的划分，例如一级类目服饰下的熟女穿搭/少女穿搭/儿童服饰/中老年穿搭等，图 2 是视频类目体系的几个案例。在这个体系之上多媒体算法团队投入近半年的时间提出了基于模态注意力机制

的多模态分层视频分类算法。

图二

多模态分层视频分类算法

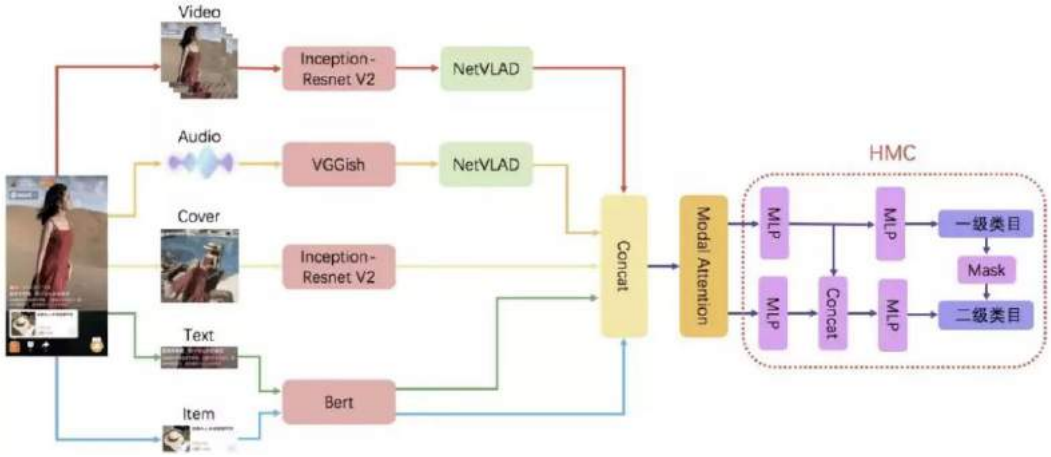
视频分类的难点

- 多模态：淘宝短视频的信息是非常丰富的，有视频/封面图/文本/音频/商品等模态，分别刻画了短视频不同维度的信息，这些信息的展示形式都是非结构化的，如何将非结构化的信息转化成结构化的特征是一大难点。不同模态的信息在不同的视频中对类别的贡献度也是不一样的，小部分视频通过标题就可以简单地推测出类别，但大部分视频的标题有效信息过少，需要兼顾其他模态信息才能推测出类别，因此在算法的训练中如何协同不同模态的特征，达到不同模态特征互补的效果又是另一个难点。
- 层次化 label：单独使用二级类目的 label 虽然也可以进行算法的训练，但是无法使算法达到最优的效果，因为不同一级类目下面的二级类目之间的差距是远大于同一个一级类目下面的二级类目之间的差距的，单独使用二级类目的 label 无法学习到这个信息。因此如何在算法的训练中充分利用一级类目+二级类目的层次化 label 同样是一个难点。
- 模态信息缺失：淘宝视频包含多种不同的模态信息，然而这些视频不一定包含全部的模态信息，有些视频没有外挂商品，有些视频没有对应的视频标题或者摘要，还有的视频没有封面图。如何让算法能够自适应模态缺失的情况也是一个难点。

多模态分层视频分类算法框架图

针对建立高效准确的视频分类算法的迫切需求，解决视频分类中存在的诸多挑战，我们提出了基于模态注意力机制的多模态分层视频分类算法，算法总体框架如图 3 所示。算法的核心主要分为 3 个部分：

1. 预训练模型的选择，
2. 模态融合方法的设计，
3. 多目标的分类器的设计。



图三

预训练模型的选择

随着硬件水平的提高以及大规模的预训练数据集的推出，迁移学习在深度学习任务中扮演的角色越来越重要。尤其是在缺乏训练数据的情况下，使用预训练模型进行迁移学习能够加速 loss 收敛并显著提升下游任务的准确率。

(1) 视觉模态：视频和封面图共同构成了视觉模态信息，视频是视频内容的主体，包含了主要的内容信息，封面图是视频内容的精华，两者可以互相补充。在 VGG16、Inception 系列模型、ResNet 等经典的图像分类模型中，我们选择了 Inception-Resnet v2[1]作为视觉特征提取的模型。这个模型是 2016 年 Google 推出的大规模图像分类模型，既具有 Inception

系列模型的优势，能够通过堆叠不同的 Inception Block 增加网络的宽度提高算法的准确率，还加入了 Resnet 的残差学习单元(如图 4 所示)，残差学习单元的输出由多个卷积层级联的输出和输入元素间相加，能够缓解网络退化的问题，增加深度网络的层数，有效提高视觉特征的的泛化性。

视频特征序列相较于普通的图像特征包含了更加丰富的信息，不同特征之间具有时序相关性。我们采用 NetVLAD(如图 3 所示)作为视频特征的聚合网络。NetVLAD 常出现在近几年国内外视频分类大赛的 top 方案中，以 CNN 的网络结构实现 VLAD 算法，构成了新生成的 VLAD 层，VLAD 算法(如公式 1 所示)统计的是特征 x 和其相应的聚类中心 c 的残差和，a 决定 c 是否是特征 x 距离最近的聚类中心。相比于 Average Pooling，NetVLAD[2]可以通过聚类中心将视频序列特征转化为多个视频镜头特征，然后通过可以学习的权重对多个视频镜头加权求和获得全局特征向量。

$$VLAD(j,k) = \sum_{i=1}^{K} a_k(x_i)\left(x_i(j) - c_k(j)\right)$$

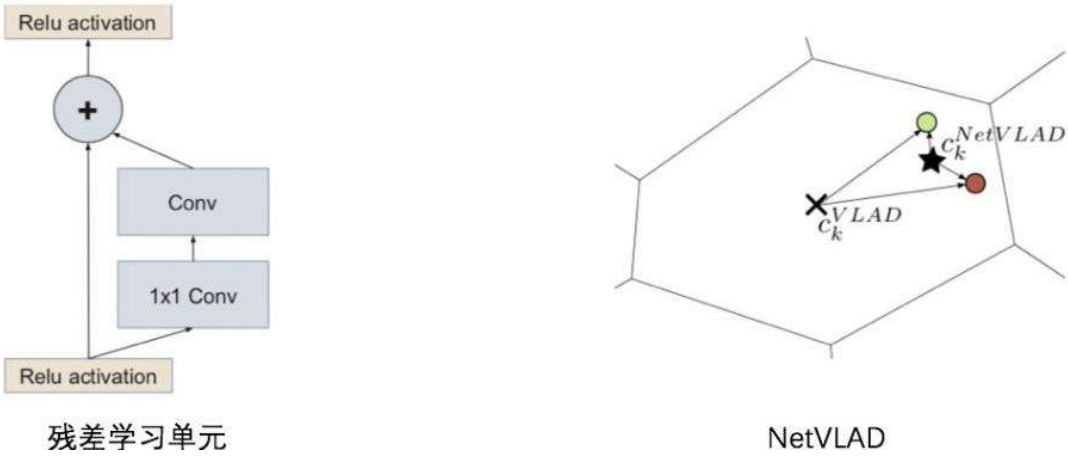
$$a_k(x_i) = \frac{e^{w_k^T x_i + b_k}}{\sum_{j=1}^{K} w_j^T x_i + b_k}$$

公式 1

(2) 音频模态：淘宝视频中包含大量的教程类视频，这些视频内容的关键信息通过音频表现出来，因此在淘宝视频分类中音频模态至关重要。我们首先从淘宝视频中分离音频信号，通过计算 MFCC 特征将音频信号转换为图像输入，然后使用 VGGish[3]提取音频特征序列。音频特征序列与视频特征序列类似，使用 NetVLAD 提取不同镜头对应的音频特征，然后通过可学习的权重融合生成音频模态的全局特征向量。

(3)文本模态：视频内容中的文本包含了视频标题和视频摘要，是视频描述内容的大致概括，对视频分类起到指导性的作用。文本模态，我们使用 Bert 模型生成视频标题和视频摘要的全局特征向量。Bert 是 18 年 Google 推出的大规模文本预训练模型，可谓是 nlp 领域大力出奇迹的代表，Bert 用 12 层的 transformer encoder 将 nlp 任务的 benchmark 提高了一大截。相较于普通的 word2vec，经过海量文本预训练的 Bert 能够在视频分类算法中引入更多的迁移知识，提供更精准的文本特征。

(4) 商品模态：商品模态是淘宝视频区别于站外视频的标志，是体现我们的视频分类算法优势的关键所在。我们沿用文本模态的 Bert 模型生成商品模态的全局特征向量。商品模态在推荐领域常用 item_id lookup 到商品的 embedding 矩阵再接入下游网络，然而我们的视频分类算法是离线学习的，对于新发现的 item_id 不能很迅速地获得它的 embedding 特征，因此我们使用 Bert 模型提取商品的标题和类目名称的文本特征，作为商品模态的全局特征向量。



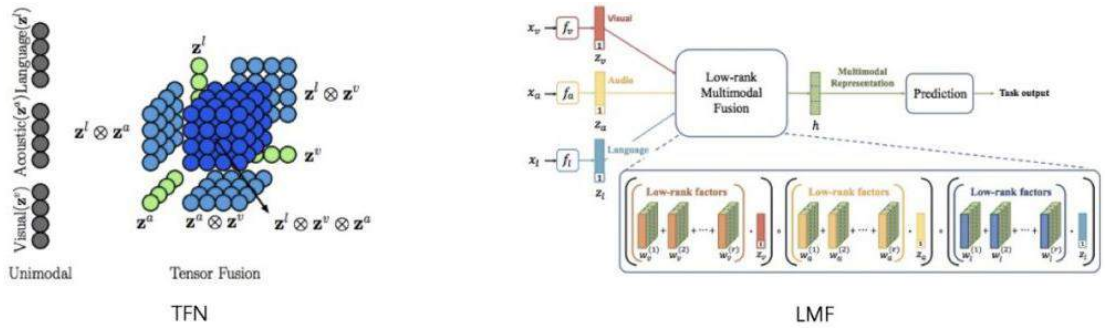
图四

模态融合方法的设计

淘宝视频的多模态信息十分丰富，不同模态之间提供的信息内容并不是完全一致的。如何设计优秀的多模态特征融合方法，充分利用非结构化的多模态信息，将不同模态间的特征对齐到同一特征空间，使得不同模态信息之间取长补短，这是视频分类算法模型中最关键的模块。我们比较了多种不同的多模态特征融合方法，实验结果如图表格 1 所示。

(1) TFN 和 LMF(如图 5 所示)都是将多模态特征映射到不同模态间外积的高维特征空间进行特征融合。TFN[6]通过模态之间的外积计算不同模态的元素之间的相关性，但会极大的增加特征向量的维度，造成模型过大，难以训练。而 LMF[7]是 TFN 的等价形式，利用低秩矩阵的分解，将原本的各模态间的先外积再全连接变换过程，等价为各个模态先单独线性变换到输出维度，之后多个维度点积，可以看作是多个低秩向量的结果的和，LMF 相比 TFN 减少

了很多参数量，是 TFN 的优化版本。但在视频分类的算法中，这 2 种方法的效果都不及预期，分析原因在于视频分类的模态特征长度都在千维左右，即使是 LMF 也会出现参数数量爆炸的情况，为了保证参数量不爆炸就必须先将每个模态特征降维，然而降维本身是有损的，导致降维后的模态特征再外积不如直接利用不同模态间特征拼接。



TFN　　LMF

图五

(1) 淘宝视频不同模态之间信息通常是不一致的，这些不一致的模态信息之间有些内容和类别标签息息相关，有些内容则相关性较低。为了关注那些与类别标签相关性更高的模态信息，降低对于不重要模态信息的关注程度，我们提出了基于 Modal Attention 的多模态特征融合方法。Modal Attention 基于融合的特征向量预测一个模态个数维度的基于多模态联合特征的对于不同模态的重要性分布概率，这个模态分布概率与多模态融合特征做点积，得到对于不同模态特征重要性重新加权过后的新的多模态融合特征。从表格 1 可以看出，基于 Modal Attention 的多模态特征融合方法的准确率显著超过了 TFN 和 LMF，验证了基于 Modal Attention 的多模态特征融合方法的优势。

(3) 为了应对淘宝视频中出现的模态缺失情况，我们使用了 modal 级别的 dropout，在训练的时候以一定比例随机性去除某个模态信息，增加模型对于模态缺失的鲁棒性。在不添加 modal dropout 时，测试数据如果缺失 10%的模态信息，测试精度会下降 3.5%左右；在添加了 modal dropout 后，测试集精度下降不到 0.5%，几乎可以忽略。同时添加 modal dropout 后，就算模态信息不缺失的情况下，还能够提升测试集精度，提高约 0.4%。

Modal	modal fusion method	classifier	level1 accuracy	level2 accuracy
Video	--	MLP	87.5%	76.5%
Cover	--	MLP	84.9%	72.4%
Text	--	MLP	89.4%	79.1%
Item	--	MLP	86.5%	75.3%
All	TFN	MLP	93.9%	84.1%
All	LMF	MLP	94.1%	84.4%
All	Modal Attention	MLP	95.1%	86.8%
All	Modal Attention	HMC	96%	88%
All	Modal Attention	HMC+violation loss	96.6%	88.7%

表格一

层次化的分类器的设计

一般来说，分类任务只有单一的分类目标，然而，淘宝视频的标签体系是一种结构化的分层分类任务，同时具有一级类目和二级类目，一级类目和二级类目之间有依存关系，构成了一种树状的分类体系结构，例如：二级类目的熟女穿搭、少女穿搭、男士休闲等都属于一级类目的服饰类。针对这个任务，我们提出了分层多标签分类器（HMC）。

(1) 我们将 HMC 分类器与非常经典的 MLP 分类器做对比。MLP 分类器直接预测淘宝视频的二级类目标签，然后根据一二级类目之间的对应关系获得一级类目标签。HMC 分类器同时构建了一二级类目各自的分类通道，能够同时预测一二级类目标签，结构图如图 2 所示。这种分类器相比于 MLP，能够隐形的学习一二级类目的依赖关系，能够互相促进，提高分类精度，表格 1 的实验结果证明了这一点。

(2) 基于 HMC 分类器，我们添加了基于类别不匹配的多目标损失函数，具体公式见公式 2。损失函数 L 由三部分构成，分别是一级类目损失 L1，二级类目损失 L2，以及一二级类别不匹配损失 LH。一级类目损失和二级类目损失是一二级类目的交叉熵损失，能够使得网络同时学习到多模态特征与一二级类目的条件概率分布，同时能够隐形的学习到一二级类目之间的依赖关系。然而，仅仅使用一二级类目损失无法保证一二级类目之间的依赖关系，为了缓解这个问题，我们加入了类别不匹配损失，用于惩罚一二级类目不匹配的情况。参数 λ 用来控制一级类目损失和二级类目损失之间的重要性相对程度，因为二级类目数量更多，学习更加困难，需要添加更大的权重去学习。参数 β 用来调节类别不匹配损失对于总体损失函

数的重要性。添加类别不匹配损失之后，一二级类目不匹配的情况大幅度下降，同时分类准确率也获得了提升。

$$L = L_1 + \lambda L_2 + \beta L_H$$
$$L_1,\ L_2 = -\frac{1}{N}\sum_{i=1}^{N}\left[log\left(\hat{Y_{ij}}\right)\right]$$
$$L_H = max\left(0, Y_{class2} - Y_{class1}\right)$$

公式二

总结和展望

Google AI 掌门人 Jeff Dean 在 NeurIPS 举办期间指出在 2020 年多模态学习将会有很大的发展，能够解决更多单模态无法解决的问题。我们提出的基于模态注意力机制的多模态分层视频分类算法方案为淘宝的视频内容化理解奠定了坚实的基础，为淘宝视频的精细化运营能力和冷启动能力做出了应有的贡献。与此同时，我们的方案还根据业务需求提供了无成本的可扩展能力，不仅能够助力图文内容化理解，提升图文的理解深度，同时还能通过精细的视频向量化表达来解决相似视频召回、视频抄袭检测的问题。

基于模态注意力机制的多模态分层视频分类算法方案为淘宝的视频内容化理解开了一个好头，后续我们还会在视频内容化理解的领域内继续耕耘。目前我们正在尝试更细粒度的多模态视频标签算法，力图将目前的 2 级视频类目体系推向 2 级视频类目+多级视频标签的体系，通过视频类目+视频标签的组合形式持续提升淘宝视频的精细化运营能力和冷启动能力。在多模态技术上，我们会继续关注如何更高效地进行模态融合，尝试将预训练技术引入多模态中，通过误差重建的方法让多个模态相互学习，提升模态融合的能力。

reference

Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, inception-resnet and the impact of residual connections on learning[C]//Thirty-First AAAI Conference on Artificial Intelligence. 2017.

Arandjelovic R, Gronat P, Torii A, et al. NetVLAD: CNN architecture for weakly supervised place recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 5297-5307.

Hershey S, Chaudhuri S, Ellis D P W, et al. CNN architectures for large-scale audio classification[C]//2017 ieee international conference on acoustics, speech and signal processing (icassp). IEEE, 2017: 131-135.

Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

Wang W, Bi B, Yan M, et al. StructBERT: Incorporating Language Structures into Pre-training for Deep Language Understanding[J]. arXiv preprint arXiv:1908.04577, 2019.

Zadeh A, Chen M, Poria S, et al. Tensor fusion network for multimodal sentiment analysis[J]. arXiv preprint arXiv:1707.07250, 2017.

Liu Z, Shen Y, Lakshminarasimhan V B, et al. Efficient low-rank multimodal fusion with modality-specific factors[J]. arXiv preprint arXiv:1806.00064, 2018.

Wehrmann J, Cerri R, Barros R. Hierarchical multi-label classification networks[C]//International Conference on Machine Learning. 2018: 5225-5234.

AceRay 极速 3D 渲染：预览速度提升 30 倍

作者 | 淘系技术
出品 | 淘系技术

天猫 618 宣布的 3D 购物时代，相信有很多小伙伴好奇，这背后有哪些"黑科技"？橙子从以下三点为你揭秘——3D 实景复刻、3D 渲染、3D 算法，上周讲了《天猫 618 宣布开启 3D 购物时代，实景逛街背后的技术是这样实现的》，本期将和大家分享 AceRay 极速渲染。

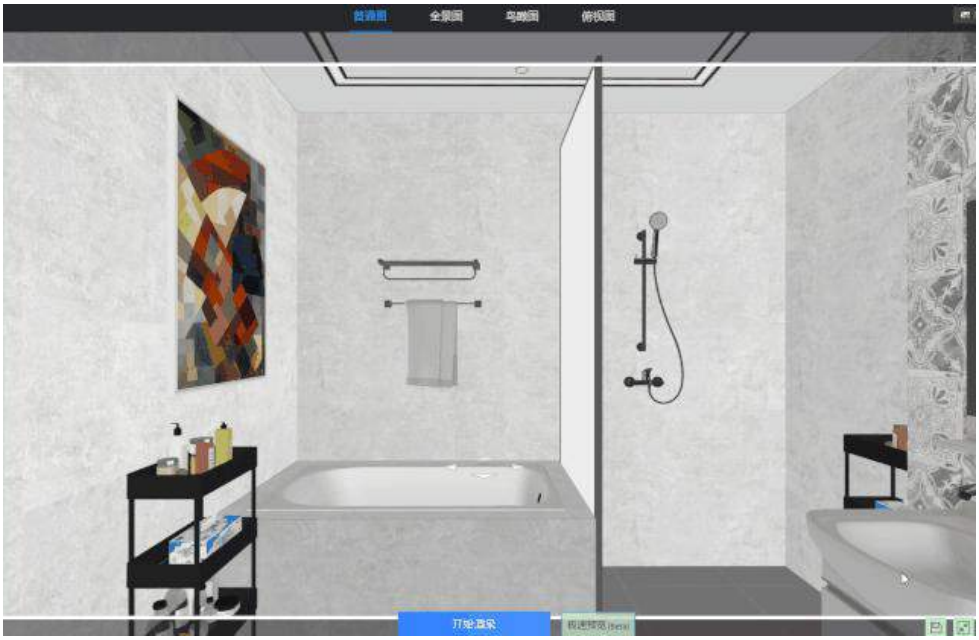
5 月 28 日，天猫正式宣布开启 3D 场景购物时代，3D 实景逛街从此走到了电子商务的最前线。用户足不出户便可逛各大卖场，在虚拟 3D 世界中，更可以享受自由搭配家具的畅快感受。

3D 实景逛街意味着线下的卖场所有数据需要 3D 数字化，这涉及到 3D 内容生产，数据链路整合，3D 渲染技术等多个领域，是一项规模浩大的工程，需要设计师，工程师，以及产业链方面的通力合作才能做到。

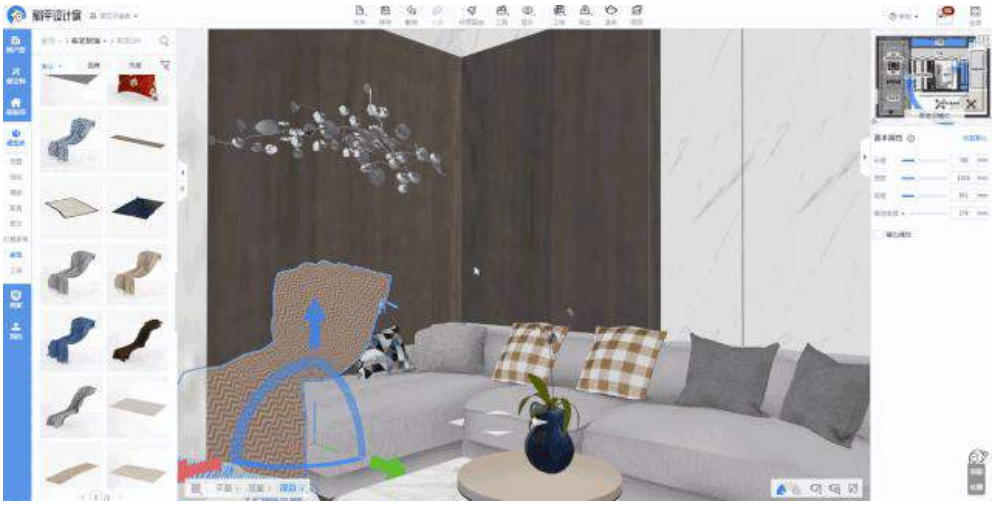
躺平设计家是阿里巴巴淘系技术部联合躺平推出的为商家和设计师服务的免费 3D 设计工具和渲染服务，是一个能够高效生产 3D 内容的平台工具。其中最新推出的依托于淘系技术部 AceRay 云渲染平台的极速渲染功能，大幅度提高了设计预览的速度，可以助力各商家和设计师更方便快速地制作 3D 内容，更好地融入 3D 场景化购物这一未来家装购物模式。

在躺平设计家里，用户可以非常方便地编辑 3D 场景，并利用极速渲染快速查看渲染效果。

点击查看动图演示



点击查看动图演示



点击查看动图演示

AceRay 极速渲染云平台涵盖了大量的工程化设计和最新几何/渲染研究成果，包括 3D 资源的处理，服务器资源调度，以及图形渲染算法。

3D 资源

在进入到渲染前，需要提前准备好渲染资源，这些资源包括模型、材质。

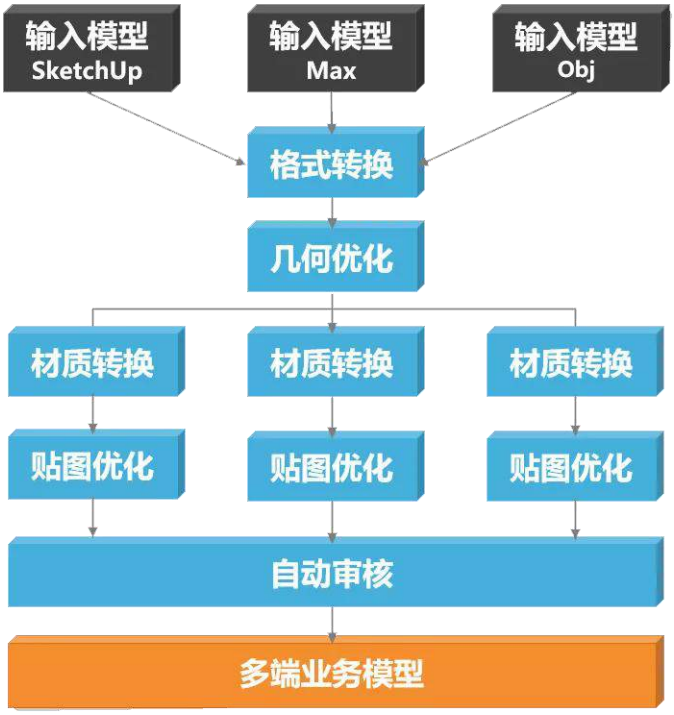
家装行业的业务复杂度很高，需要支持大量不同格式模型上传，同时需要为各个平台产出模型和材质。不同格式模型上传需要转成中间格式；目前家装模型通常三角面数都是非常高，需要通过几何优化手段把面数降下去；家装模型中的贴图通常都非常大，需要根据各个平台优化贴图；等等。

为了处理各个不同的问题，需要大量的服务器处理各个不同的问题，但是这些服务器怎样串通起来，怎样合理的调度能够使得这些服务器的综合效率最高，各个服务器间怎样并发处理大量的模型，为各个平台产出不同的模型格式，需要一个管控平台来处理这些问题。因此我们开发了模型中控平台，支持无上限提交模型，同时并发处理大量模型，为每个模型定制不同的处理流程，能够自由地控制每个模型的处理优先级。

模型中控平台中包含两个核心部分：流程引擎和调度引擎。

开始一个新模型转换流程时，第一步进入流程引擎，分析流程节点，找出第一个需要执行的节点；然后任务交付到调度引擎，调度引擎负责找出最合理的远程执行机器，将任务分给它，并等待它执行完毕；当远程机器执行完毕时，调度引擎将任务交回给流程引擎，流程引擎将当前模型流转到所属流程的下一个节点，再度交付到调度引擎。重复执行这样的过程，直到当前模型的全部流程被执行完。

模型中控平台上可以同时运行 N 种不同的处理流程，一种典型的处理流程如下：



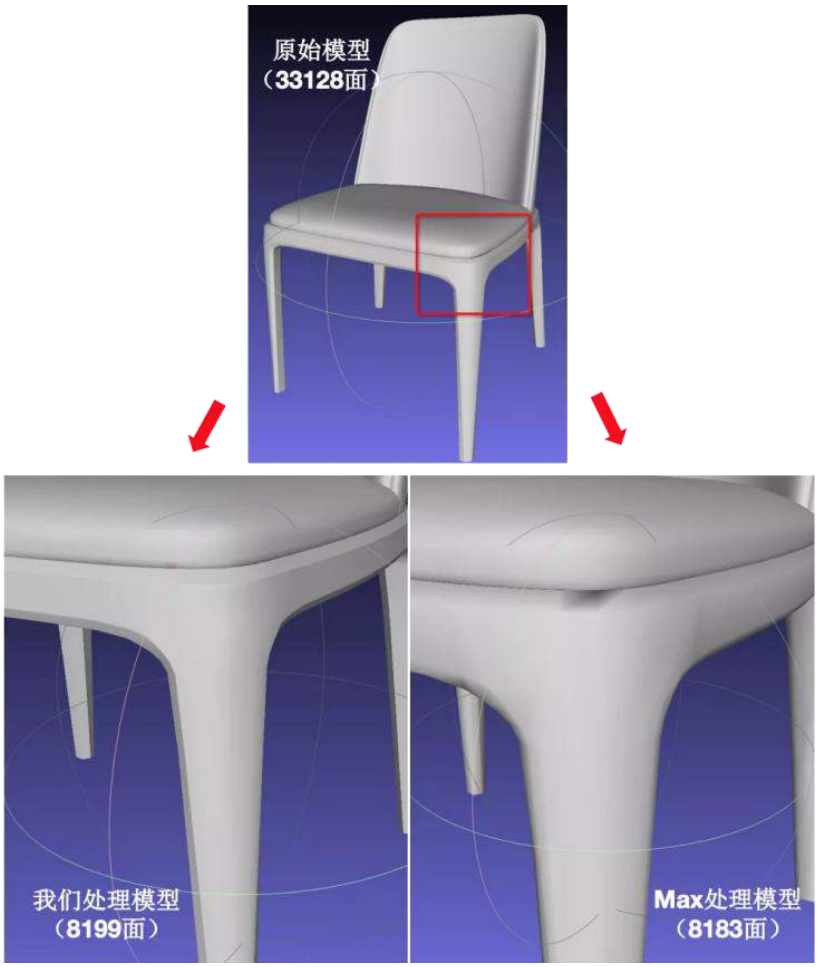
解决了批量处理问题，我们有更多的单点问题需要解决，比如模型问题：

目前设计师或者工厂所涉及的家装模型通常三角面数都是非常高的，材质非常复杂。

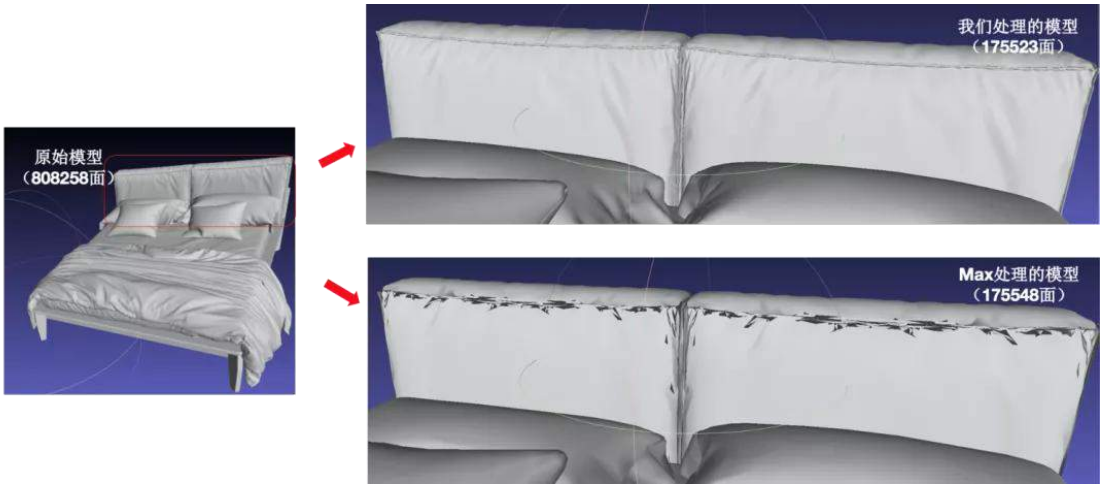
高面数有利于增加模型的细节，但是对于极速渲染来说，这么多的三角面片就不是一件好事了。因此我们需要在渲染之前，预处理这些拥有大量三角面片的模型。目前市面上类似的工具应该说是形形色色，都能做到精简模型面片的功能。但是具体到细节上，减面后需要保持原模型的各种细节，要有精准的法线，uv 等要求，最重要的是一套参数控制所有模型，能够达到非人工干预、自动化、规模化的模型处理工具，在工业界并不存在。因此我们自主开发了一套几何优化工具解决上述问题。

对比 Max 处理的模型，我们处理的模型在外形保持、细节保持和法线、UV 保持都要更好。

细节保持效果：



外形保持效果：



在解决了面数问题后，我们需要进一步解决材质问题。在家装行业，比较常见的一种材质是 Vray 材质，这种材质的参数非常多，计算过程非常复杂，非常不利于极速渲染。而市面上又没有一种 Vray 材质转更简单材质的方法。因为我们自研了一套材质转换方案，将现存的材质转换到更简单的材质来提升极速渲染的速度。

以下是材质转换的效果：

3D 云渲染服务

淘宝的商家依赖于我们的渲染服务，因此我们并非在开发一个单点渲染引擎，而是需要一个服务器集群来处理上大量的请求，这个服务器集群组成了云渲染平台。云渲染平台管理所有的渲染请求类型，包括离线渲染、极速渲染，以及布料仿真。云渲染平台隔离业务方和渲染服务器，并统一了渲染接口，调度器会为每个独立的请求寻找最合适的机器，最快速地完成这次渲染请求。

云渲染平台分为两层：调度集群和渲染集群。调度集群向渲染集群发送任务，渲染集群向调度集群上报状态。

渲染任务首先进入调度集群，调度集群根据上报的状态和发送历史，选择最优的渲染服务器，将任务发送给它；渲染服务器更新并上报它的机器状态，然后开始执行渲染。

AceRay 极速渲染

设计工具通常都使用离线渲染器。离线渲染的效果很好，但是比较大的问题是渲染速度很慢，渲染一张图的时间短则几分钟，长则几小时。对于设计师来说，渲染速度慢无疑会降低生产 3D 场景的效率，特别是某些场景中有需要频繁调整的元素时，快速渲染的需求显得更为迫切。对于消费者，快速的效果预览也十分必要，人们通常没有耐心为了一张效果图等待太长时间。

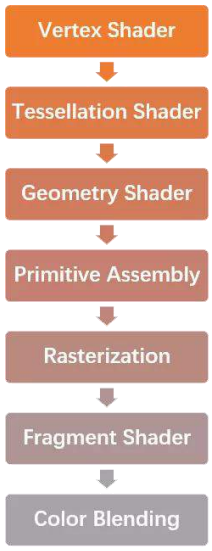
AceRay 极速渲染服务正是为了解决离线渲染速度慢的问题而诞生，渲染器的目标是要帮助设计师提高 3D 场景生产效率，提高消费者 3D 功能使用体验。

谈到渲染，效果永远是排在第一位的。如果没有优秀的渲染质量，那么再快的速度也失去了意义。我们使用 AceRay 云渲染平台，在十几秒内渲染了两张室内家装的效果图。

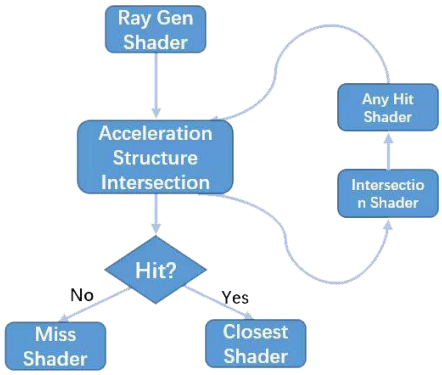
在保证上述的渲染效果时，极速渲染的渲染速度可以达到离线渲染的 20~30 倍，这无疑是一个很具有诱惑力的功能。在躺平设计家里，极速预览功能通常在 3~5 秒就能得到一张预览图像，极大提升了设计师的工作效率。

光栅化和光线追踪

首先简单介绍一下渲染原理。以游戏为代表的实时渲染业界通常使用光栅化渲染器，而以电影特效、设计工具为代表的离线渲染业界通常使用基于光线追踪的渲染器。



光栅化渲染流程



光线追踪渲染流程

光栅化渲染的管线和光线追踪的渲染管线和有着很大的不同。传统的光栅化渲染管线包括顶点处理，图元组装，光栅化以及像素着色等阶段，是一套在游戏和实时渲染业界久经考研的渲染方案。自人类有图形硬件开始，图形硬件就在做一件事情，那就是加速光栅化渲染。然而光栅化渲染有着致命的缺陷，在渲染过程中，物体与物体之间的相互关系缺失。这个缺点导致了涉及到物体相互关系的光照效果都很难计算，需要设计特殊的算法，花费高昂的代价来实现。这些效果包括阴影，间接光照，镜面反射，环境光遮蔽等。正因如此，光栅化渲染的效果在很长一段时间内都难以比肩离线渲染，做不到真正的物理真实的渲染。

而离线渲染使用的基于光线追踪原理，简单来说是从相机的位置出发，向屏幕上的每个像素的位置发射光线，通过模拟光线的各种反射/折射/散射的过程，从而实现真实的渲染效果。光线追踪的原理是简单的，但是要达到真实的渲染效果，每个像素需要大量的光线和计算，这是导致离线渲染速度慢的最主要原因。尽管现在 CPU 通常都是多核设计，但是这些计算硬件在离线渲染巨大的计算量面前还是显得力不从心，要真正能做到能够在十几秒甚至几秒内渲染出效果图，这个效率还是远远不够的。

基于光线追踪的离线渲染器，本质来说是使用光线追踪的功能，利用蒙特卡洛方法，对高维的渲染方程做数值估计。而蒙特卡洛方法使用了随机采样，为了达到较好的渲染质量，需要很高的采样数量。

AceRay 极速渲染采用定制的优化算法，很好的解决了离线渲染中需要很高采样数的问题，在较快的时间内使用光线追踪算法获得平滑的光照效果。

直接光照

间接光照

最终效果

总结

虽然图形学和光线追踪已经有相当长的历史了，基于光线追踪的快速渲染还是一个相对较新的领域。如何更好地利用硬件计算能力，结合当前业务的需求，做出快速而高质量的渲染器是一个相当富有挑战性的工作。虽然用户通常只会接触到最终的渲染效果图，然而渲染之前的模型处理，场景处理，服务器调度等工作是整个渲染服务的基石。3D 业界的数据流目前是非常多样化的，如何兼容各种数据流，处理成为统一的渲染器输入格式，高效调度服务器资源，完成渲染，这背后的工作值得更多的关注和赞赏。

更多极速渲染效果图：

ACERAY

ACERAY

3D 购物时代：实景逛街背后的技术实现

作者 | 淘系技术
出品 | 淘系技术

5 月 28 日，天猫 618 宣布开启 3D 购物时代，躺平设计家携顾家、林氏木业等行业 TOP 品牌，让 3D 场景化购物成为未来家装购物新模式。

天猫 618 首次将 3D 购物技术大规模应用于宜家、顾家等 100 个商业体，消费者可以在手机上逛商场，所见将与亲临现场近乎一样，还能随时替换其中新品，创造更大的想象和展示空间。

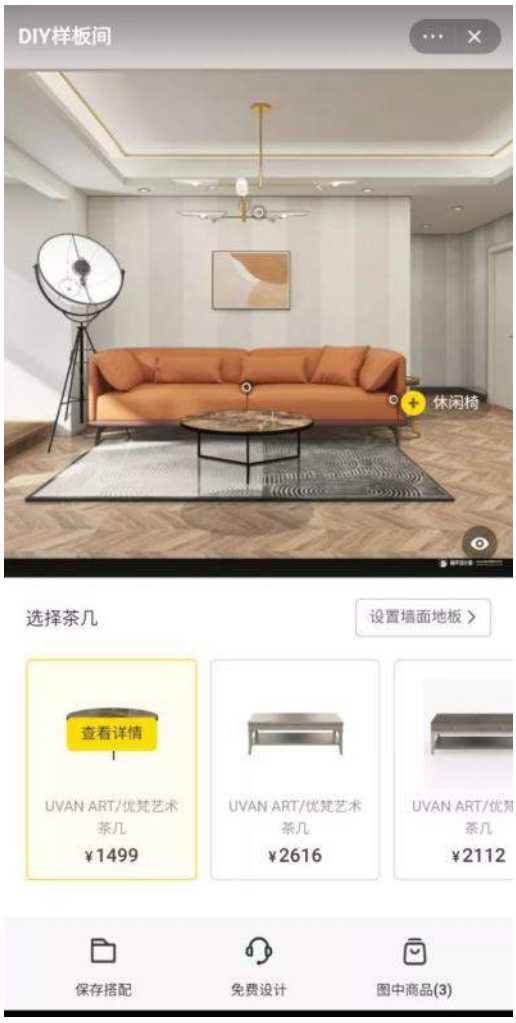
品牌宜家还将 3000 平方米的上海宝山店，按 1:1 比例复刻至天猫，常去宜家宝山店的顾客还能找到熟悉的角落。

5G 投入商用后，3D 购物成为最受科技界期待的新技术之一。作为 618 的绝对主场，天猫 618 每年推出新技术，不断拓展商业边界，同时让消费场景不断拓宽。

此前，受疫情影响，不少线下商业体的客流尚未恢复正常，实体商业格外关注如何通过新技术实现业务的在线化。3D 购物背后的实景技术能够将购物体验从平面陈列式升维为立体场景，这也是电商未来发展的主流方向。

“这可能是疫情以来对我们最有用的新技术。建立 3D 线上展厅，相当于把一个门店覆盖至全国，让逛的人更多。”顾家家居相关负责人表示。

而家装从业者表示，“过去家装电商做不了场景化展示，消费者仅仅看商品照片很难下购买决心。现在 3D 购解决了该难题。”



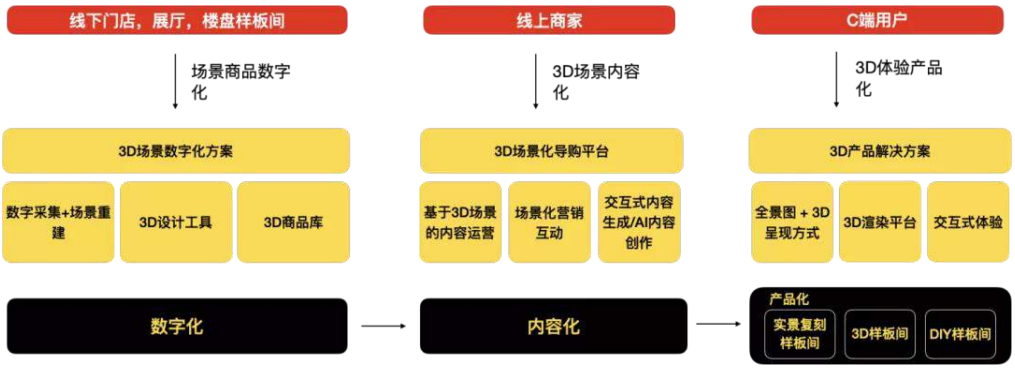
在这次天猫 618 新操作的背后，3D 实景逛街技术被推到了最前台。

3D 实景逛街技术不仅将线下场景 1:1 还原至线上，让用户可以足不出户"云逛街"，其更深的含义是将商家的线下卖场"数字化"，与线上"数字化"资产形成联动，让目前线上传统基于"图文/视频"内容导购类型升级至基于 3D 数字化场景的"场景化导购"，从而提升用户的购物体验。

3D 场景化导购技术涉及到 CV（Computer Vison）、CG（Computer Graphics）、3D 渲染等多领域。同时，要将这些领域串联起来又会是巨大的挑战。

下面，阿里巴巴淘系技术部-躺平技术团队将从"数字化"、"内容化"和"产品化"三个方面给大家分享"3D 场景化导购"项目中的一些思考。

- "数字化"着重在解决如何将商家线下的场景转化成结构化的 3D 描述，这是场景化导购的基石。
- "内容化"着重解决 3D 场景如何区别于传统内容导购，结合商家线上资产，丰富场景化导购体验。
- "产品化"着重在解决 3D 内容如何更好的展现给 C 端用户，提升用户体验。



数字化

这里的数字化主要指：

- 场景本身的数字化
- 场景中物体（商品）的数字化

早在 4 年前，淘系技术部-躺平技术团队就在场景和商品数字化领域开始了不同程度的探索。

2016 年发布的 Buy+将美国 Macy's 百货公司搬至线上

场景数字化

当前，我们对于场景的数字化采用了 2 种解决方案:

- 图像设备采集+3D 图像重建算法
- 使用 3D 设计工具 1:1 制作还原

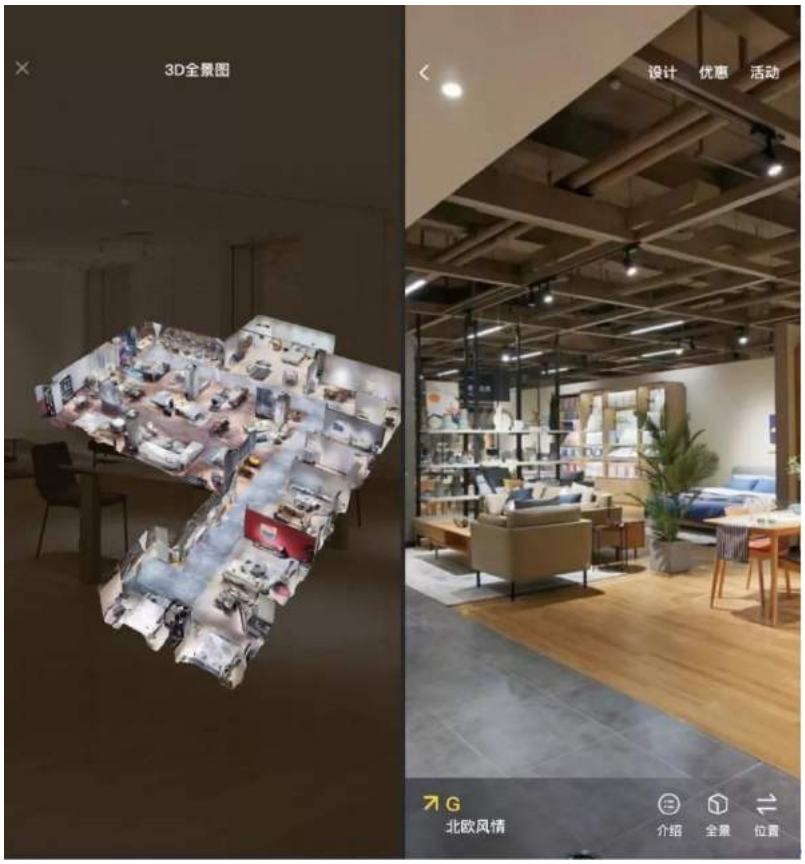
方案 a 通过自主研发的采集设备收集空间深度信息结合场景重建算法，得到场景的 3D 模型。

优势在于:

- 提供"场景"的 3D 信息，几乎 100%还原了真实场景，并且提供照片级的清晰程度。

劣势在于:

- 无法直接理解场景中的物体，如果要更深入一层，需要采用"场景理解""场景分割"算法对场景 3D 数据进一步分析。



方案 a 得到的场景三维信息和全景图

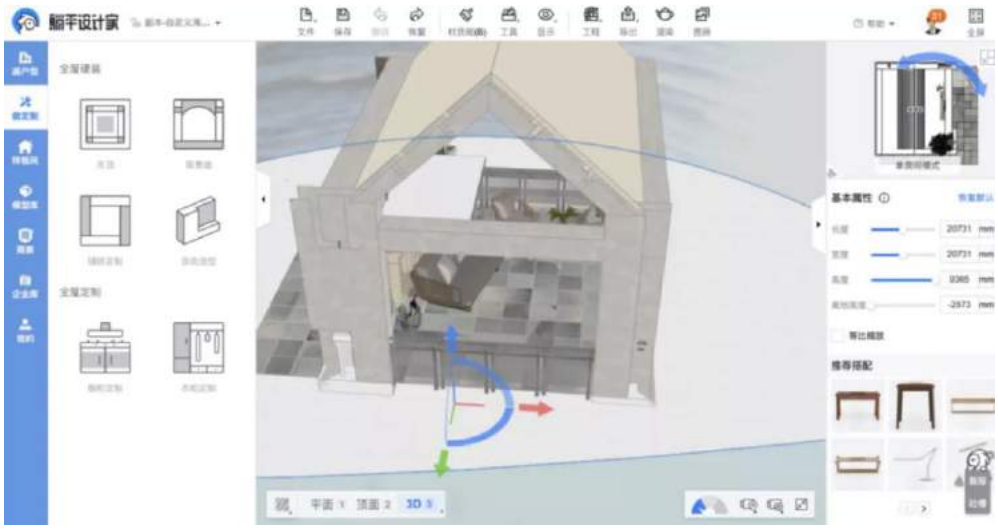
方案 b 依托阿里旗下"躺平设计家"的 3D 设计工具高效地帮助场景重建人员快速的完成场景 1:1 还原，一个重建人员每天可以还原 2-3 个 200 平米左右的线下门店或者楼盘样板间。

优势在于:

- 不仅提供场景的 3D 信息，还可以得到场景中的每一个物体的信息。
- 可以摆脱现实世界对与设计的束缚，将概念场景在线上实现。

劣势在于:

- 视觉效果依赖于离线渲染的效果（略低于真实照片）。



方案 b 使用的躺平设计家 3D-Saas

针对不同的业务，采用的方案也是不同的，核心是对"3D 数字化程度"和"视觉效果"进行平衡。

对于偏真实体验类的业务，如云逛博物馆、云逛宜家会采用方案 a，这类业务更重视还原度和体验效果。

对于设计类需求，如楼盘样板间设计，更多采用方案 b，因为这类业务会频繁出现场景需要根据用户的需求发生调整的情况，更深程度的数字化可以提供更多的调整空间。

商品数字化

模型平台收集了海量的高质量模型以及其与商品的关系，使得阿里生态可以在各种场景利用 3d 模型为原料来生产各类组合后且有商品信息的场景内容。同时，模型平台通过对模型的分析理解深挖，使得模型可以被更智能地放置、组合与检索。

对于海量的 3D 数据，淘系技术部-躺平技术团队建设了一套 3D 基础特征库(3D Foundation Features - 3DFF)，通过计算几何跟图神经网络的方法更细节地挖掘了 3D 数据的特征（如部件语义信息），并且设计了一套开放、规范的 3D 分析算法调度体系，可将多种分析的特征存入模型平台，提供给上层应用。上层应用基于这些更为精细的 3D 特征能获得更好的效果，比如使得场景设计的专家规则更为细致，基于统计的 AI 布局也更为准确。

目前模型平台已经支撑了百万级别的商品模型。

内容化

场景数字化技术，将商家的线下资产搬至线上，相较于一维的文字内容，二维的图片内容，商家拥有了一个三维内容的运营阵地，可以让用户更加真实地接触到其服务和商品。

场景本身即内容，而 3D 技术让商家线上数字资产可以以多种方式融入场景中，让用户在消费场景内容的同时更容易引发情绪，相比传统的内容导购，有了较大的提升。

基于 3D 场景的内容运营

传统的内容导购中，创作者需要拍摄图片、视频，编辑文案、商品锚点，以及通过 PhotoShop 在图文中编辑商品尺寸信息，这些工作在 3D 场景内容编辑中我们使用算法都可以实现自动化，如自动计算场景图片的锚点，并且通过模型平台获取关联的商品信息，甚至可以分析场景，使用 AI 算法生成场景介绍。

场景化的营销互动

在场景化导购中，场景对用户需求不再是单个商品的激发，而是多角度的激发。用户看到一个茶几，也能看到茶几上漂亮的摆件，最终可能是小摆件+茶几的组合让用户产生兴趣。在这种新导购场景里，我们融入阿里积累多年的营销互动玩法，还尝试了多种新的模式。

例如，算法可以分析场景信息，将场景中可以相互搭配的商品提取出来，形成区别于传统单品的组合优惠价。

3D 场景丰富了内容的维度，提升了业务的想象空间。

躺平轻应用中的"组合购"

交互式内容生成

当用户浏览一个场景的时候，可能会有这样的需求"如果刚才看中的商品放到这个场景里是什么样子？"，3D 场景内容提供了这种个性化需求的解法，用户可以将场景中的任何一个商品替换成另一个商品并查看其在场景中的效果。

为了能将一个静态的内容变成可以根据喜好动态变化的内容，我们构建了一个互动式内容生产引擎来自动化赋予静态的 3D 场景 DIY 搭配的能力。

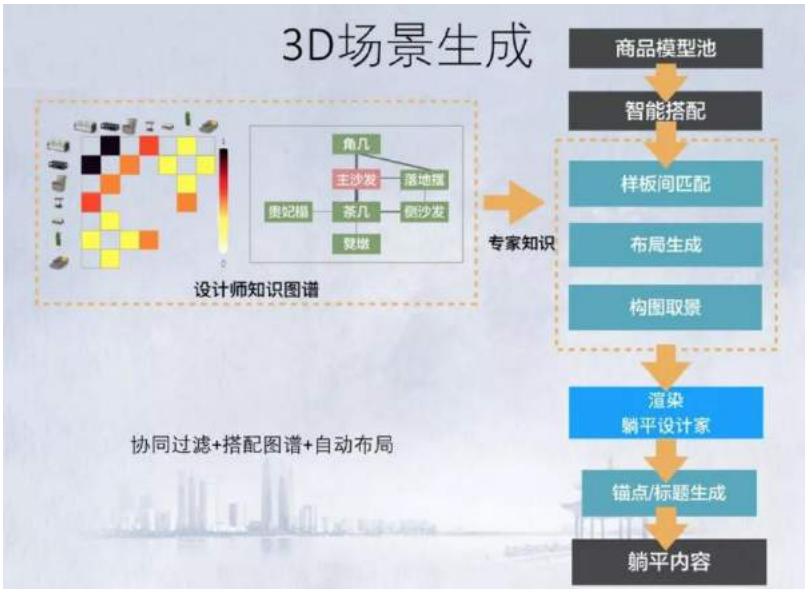
这个内容生产引擎有如下几个核心步骤：

- 分析 3D 场景，根据专家知识图谱来抽取 3D 场景内容中的设计规则。
- 通过分析商品的 3D 特征（3DFF）和基础特征服务（ABFS），结合场景的设计规则确定可替换商品模型候选池。
- 确保模型候选池中的商品模型能够美观地适配场景，这里主要靠 3D 微调算法来完成，算法依据步骤 a 中的设计规则，对原始方案进行分析构建多维度数值约束进行优化求解，实现新的家具组合按照原布局适配到场景中。

AI 内容创作

制作一个精美的 3D 样板间需要耗费设计师大量的精力，为了满足消费者不同的需求，需要设计海量的 3D 样板间，这对商家来说也是一笔巨额的投入，特别是对淘宝上的中小卖家而言，更是无形的拉高了这个行业的门槛。如何为淘宝上数以亿计的消费者提供差异化的服务是我们面临的重大挑战，也是我们面临的重大机遇。

基于日渐成熟的 3D 技术，借助 3D 自动理解、自动创作的能力，通过学习现有 3D 内容的搭配、布局等信息，结合图神经网络(GNN)、视觉技术(CNN)以及自然语言处理相关技术(Transformer)我们让机器拥有了设计师的能力。利用机器生成海量的 3D 导购内容，并通过个性化推荐技术形成精准的需求匹配，将合适的内容分发给合适的消费者，形成新的用户场景下的产品，引导消费者从单品消费走向场景消费，并真正实现一站式购买。



产品化

针对不同的 3D 场景内容，淘系技术部-躺平技术团队投入了大量时间打磨和优化对应的 C 端产品形态，使 3D 场景给用户更好的体验。

场景导购要用户引发情绪，需要接近照片级别的渲染质量，这在用户手机端对渲染技术是一个极大的挑战。

对此，我们将持续投入自建的客户端渲染引擎，同时尝试其他新的渲染解决方案在现阶段尽可能的提供给用户逼真的体验。

3D样板间　　实景复刻样板间　　DIY样板间

点击查看动图演示

AceTiny 自研渲染引擎

AceTiny 是淘系技术部-躺平技术团队自研的一款主打移动端的高性能渲染引擎。AceTiny 引擎为了提升 3D 互动业务制作效能而开发、拥有体量小、运行效率高等特性、支持动态语言开发、PBR 光照以及骨骼动画等引擎常用功能，方便快速地支撑起了需要移动端

渲染的大量业务。AceTiny 提供的性能媲美 Native 级别的渲染接口，可以在移动端高效地完成较大规模的场景渲染，这使得在手机端渲染室内的家居场景成为可能。AceTiny 引擎有着体积小、易于集成、开发上手快、运行效率高这一系列优点，给用户在移动客户端上提供了逼真的 3D 体验。

基于 AceTiny 端渲染完成的 AR 效果

点击查看动图演示

2D&3D 混合呈现

直接使用渲染引擎渲染在手机端效果目前还不能在所有 3D 场景中令人完全满意，用户看惯了高清图文，直接看 3D 的场景渲染效果体验会有损失。对此，淘系技术部-躺平技术团队采用 2D&3D 混合的方式来平衡“交互性”与“效果”。

在实景复刻项样板间目中，我们采用全景图+3D 模型的方式来保证用户浏览效果体验的同时保留空间感。

在 DIY 样板间中，我们放弃部分视角的自由度，使用 2D 渲染效果取代 3D 渲染效果，结合 3D 场景的光影条件对 2D 素材进行重打光、施加阴影，同时使用图像算法将 2D 素材与场景融合，在保证场景内容可交互的同时，也能让效果逼真可信。

结尾

淘系技术部-躺平技术团队深耕"3D 技术在商业中的落地"这一课题多年，在电商中对 AR、VR、3D 都进行过深度探索，在 CV、CG、3D 方向都有长期积累 。而在诸多的 3D 技术选型中，我们首先考量的问题一直是两个：

1. "该技术能否给客户（不管是 B 还是 C）带来真实价值？"，3D 技术给人第一印象是酷炫，在酷炫背后给用户带来了哪方面的提升才是核心，我们希望 3D 技术"酷炫"且"有用"。
2. "该技术是否可被规模化？"，因为只有可被规模化应用的技术，才具带来长久的商业价值，对于目前仍不具备规模化能力的前沿技术，我们保持关注。

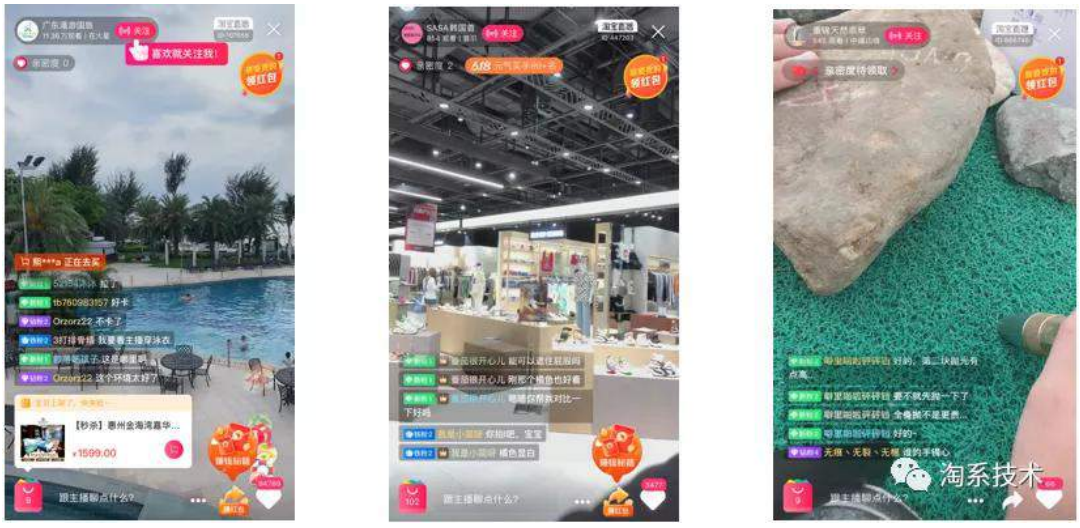
这次 618 天猫大规模的应用 3D 技术，正是印证了这两点。我们会持续整合阿里的图形、图像、3d 技术能力赋能商家，提高商家场景化内容生产效率，降低商家内容制作成本，助力家居行业数字化的革新。

打破传统降噪算法：AliDenoise 的背后技术

作者 | 庄恕、七琦
出品 | 淘系技术

近年电商直播越来越火热，同时移动端直播变得越来越普及，直播场景也越来越多样化，不再限于传统的直播间，开阔喧嚷的户外、喧闹的商场甚至是市场都可以变成主播直播的根据地。影响直播最终观感最直接的两个因素就是画面和声音，场景的复杂化对传统的实时音频信号处理算法提出了更大的挑战，其中最直观的一个现象就是，场景变吵了，声音听起来不干净，这就是音频 3A 算法中的降噪算法需要处理的问题。

背景介绍



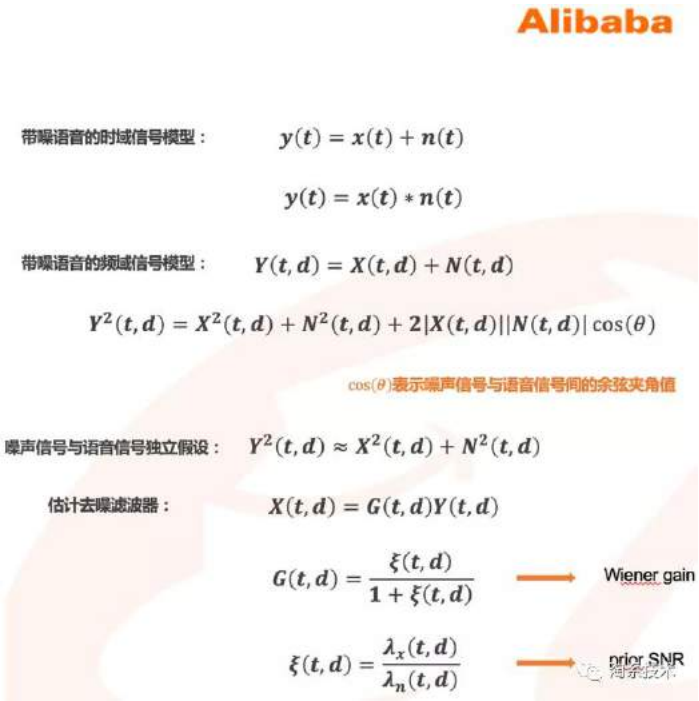
户外旅行&户外代购&玉石市场

传统降噪算法可以部分处理这种噪声，它的方法是：

1. 对时域信号做短时傅里叶变换，就得到了信号的频域表现形式，对幅度谱平方得到信号能量。

2. 在计算了带噪语音的幅度谱后，会借助 vad 模块的结果进行噪声谱 N 的估计，由计算出的功率谱和估计出的噪声谱来计算先验信噪比。
3. 再经过变换得到增益函数，一般取维纳增益，将计算得到的增益函数乘以带噪语音的频域形式 再经过反变换，就得到了增强后的语音的时域信号。

不难看出，这是一个比较复杂的信号分离任务，传统的信号处理方法通过一系列条件约束求得了最终的解析解或迭代解，一般假设噪声信号和语音信号之间相互独立，cosq 为 0，但这通常是不合理的假设，也影响了算法效果的上限。



问题与挑战

传统降噪算法面临的挑战如下：

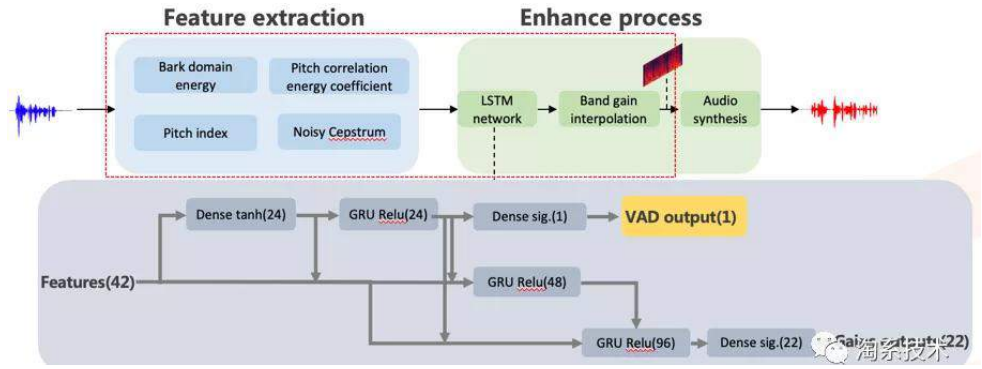
1. 非平稳突发性噪声抑制能力差，比如户外直播场景中突然出现的汽车鸣笛声，传统降噪算法就难以消除干净。
2. 传统算法增强后会有很多残留噪声，这些噪声会导致主观听感差，甚至影响语音信息传达的可懂度。

3. 目前传统算法中求得解析解的过程中存在很多假设，难以适应复杂多变的实际场景，特别是瞬态噪声。
4. 在低信噪比情况下，传统方法的降噪效果几乎是失效的，其中 VAD 模块误判的可能性大大增加，尤其在噪声的能量大于语音的能量时，几乎无法将语音谱恢复出来，而这对天然能量就比较低的辅音结构来说是毁灭性的。

业界做法

针对传统算法存在的弊端，业界在 2014 年开始探索数据驱动类算法，也就是深度学习相关的算法，比较典型的是 RNN Noise 和 CRNN 方法。

RNN Noise 是一种基于传统算法改进的混合方法。整个算法为了保证实时性，训练特征主要以传统信号的特征为主，（涉及 bark 域的能量和 pitch 域的相关能量系数、pitch_index 的值、带噪语音 bark 域的能量和带噪语音 bark 域倒谱相关的信息），一共 42 个，并没有采用深度学习算法中主流的自动提取特征机制，网络结构以 GRU 为主，同时有一个副产物 VAD 保证有效收敛，最终的输出为 bark 域的 22 个增益，经过插值平滑处理后与带噪信号的频域信号相乘，反变换后得到降噪后的时域语音信号。



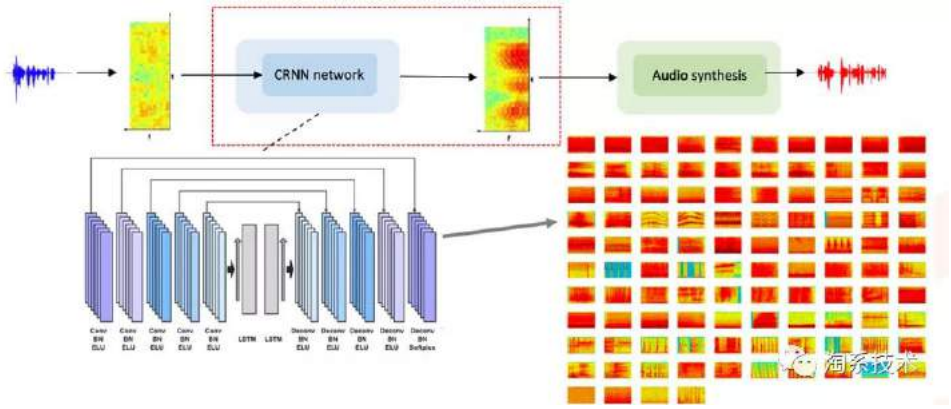
这个算法的核心优势是可实时，性能消耗低，降噪能力也有所增强，但语音损伤比较严重，经过分析，认为原因主要有三个：

1. 数据集问题，rnnoise 的训练数据集中噪声质量是不高的，噪声中含有较多人声部分，这在一定程度上会导致语音损伤大的问题。

2. vad label 制作的不准确性，简单采用一些能量判断法当做 vad 模块的 groundtruth，也会影响最终效果。
3. feature 只有 42 个，网络输出只有 22 个，这样的设计在降低运算量的同时会损失部分详细信息，是会影响最终的效果。

CRNN 方法一个典型的 end2end 的解决方案，输入是 noisy 的频域幅度谱，输出直接是降噪后的频域幅度谱，然后结合 noisy 的相位谱，进行时域数据的合成。网络模型采用了 encoder+lstm+decoder 的架构，无论是语音分离还是增强，这都是比较主流的一种网络

结构，同时为了做到原理上可实时，卷积全部使用了因果卷积，CRNN 在整段增强时效果是非常好的，增强后的主观听感也很好。



但存在的问题也比较大的问题：、

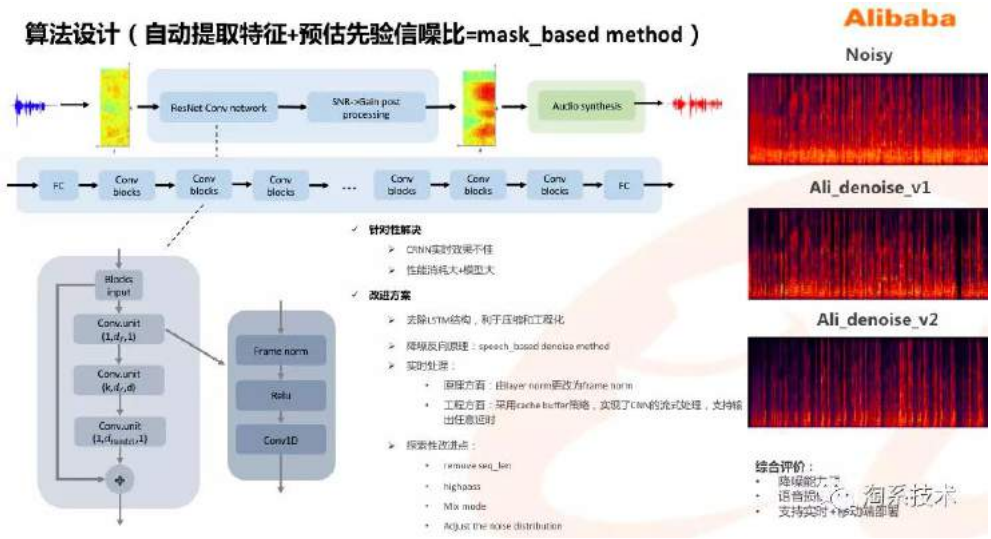
1. 整段处理的输入无法满足流式处理的要求，在模型分帧后效果下降比较明显。
2. 即使采用 cache buffer 的策略保证分帧后的效果，由于模型的感受野较大，性能开销也无法达到移动端实时的要求，同时，模型整段增强时本身消耗就比较大。
3. 中间存在 lstm 层，模型的轻量化比较困难。

AliDenoise

AliDenoise 是综合 RNNNoise 和 CRNN 的一种自研算法，首先为了可实际部署，去除了中间的 lstm 层，此时如果训练目标还是具有时序关系的降噪后频谱，效果会大大受到影响，

所以我们更改了降噪策略，进行反向降噪，改为 speech_based denoise method，只利用 CNN 较强的提取特征能力，学习比较有特色的语谱纹路，将输出抽象为传统信号处理中的先验信噪比这一概念，本质上也是一种增益，这一转化也有益于提升模型对未见噪声的泛化能力，可以通过大量纯净语音的训练反向应对未见过的噪声达到降噪的目的，在保证可实时处理方面，首先更改了模型 norm 的方式，由 layer norm 变为 frame norm，从原理上保证可实时处理，同时客观指标并没有下降，还有轻微的提升；然后在工程实践中，采用 cache buffer 策略，实现了 CNN 的流式处理，同时保证分帧后的效果，可以支持任意延时，这也是目前线上应用的版本。

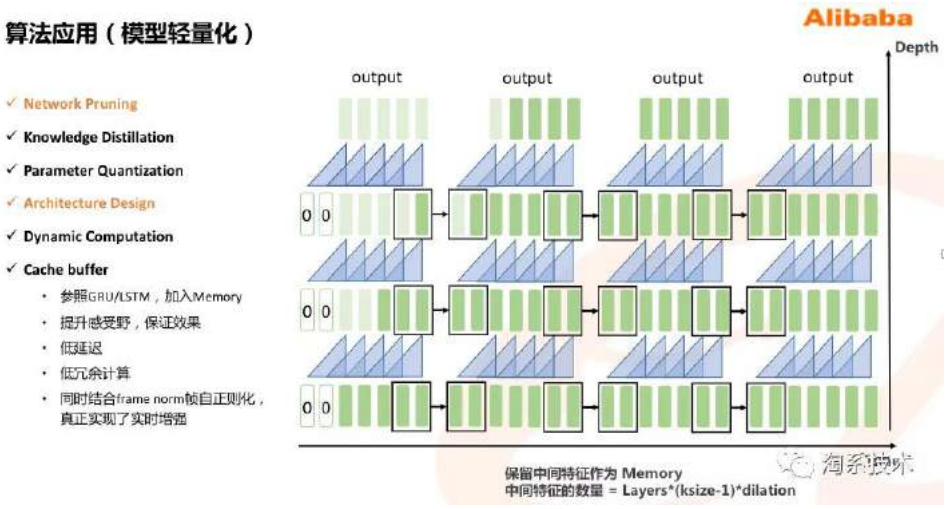
网络结构以 1D 卷积为主，同时加入了 res 结构，在每一个 blocks 中实现了三个卷积单元，通过调整 kernel size 实现不同维度的特征提取，将不同 blocks 连接起来，达到推进抽象特征的深入过程，最终得出期望的时频域上的先验信噪比。除此之外，我还进行了去除 seq_len，添加 highpass，更改数据混合方式，调整训练集中的噪声分布等改进项，从不同角度提升模型的训练效果。



为了节省计算资源，我们采用了模型裁剪手段：

1. 把 neuron 或 weight 剪掉，使得网络变小。通过对网络的分析会发现有些 weight 基本为 0，对输出影响不大或有些 neuron 的 output 总是为 0，那这种情况下就可以把它移除。

2. 使用更少的 bit 来表示权重。
3. 调整网络的结构设计，减少参数量。
4. 模型可以自适应，在资源不充足的情况下，也可以输出一个还不错的结果；在资源充足时就做到最好。



总结下来，传统降噪算法只能处理稳态噪声，非稳态突发噪声处理能力弱，但数据驱动类的算法稳态噪声及瞬时 非稳态噪声均可处理。在性能消耗上，传统算法是比较小的，数据驱动类算法会相对耗一些，为此，我们设计了小模型使用简单结构 以更好的发挥 MNN 的优化效果的策略为主，在实用性上，传统算法经验参数多，场景适应性差，而数据驱动类算法，场景适应性是比较好。

算法效果

我们看一下算法效果，客观指标主要通过 PESQ 语音质量与 STOI 语音可懂度来衡量，我们构建了 ali_test_datasets，含有 1w 多条带噪语音，涵盖街道、咖啡馆等日常环境，分为四个子测试集，_itu 表示数据集中的纯净语音为 itu 标准测试序列，_objective 表示数据集中的噪声为 训练集中未见 但是同分布噪声，相对应的就是数据集中的噪声为训练集中未见且不同分布的噪声，环境噪声强度为-5 到 15db，最大降噪力度控制在-25db，我们和业界标杆的竞品进行对比，最终可以看到在 STOI 可懂度方面的提升中 ali_denoise 和竞品明显优于传统信号处理算法，而 ali_denoise 略高于竞品，在 PESQ 语音质量方面的提升中，

ali_denoise >竞品>rtc，同时在性能消耗方面，ali_denoise 可以支持 iphone6 及以上的机型，竞品目前可以支持 iphone8 及以上的机型。

	降噪能力	PESQ	STOI	延时	性能消耗	杂音残留
noisy		2.09975	0.82369			
webrtc_ns	19dB	2.39652	0.74005	10ms	2%(支持iphone所有机型)	较大
ali_denoise_v2	25dB	**2.68043**	**0.86589**	**86ms**	**8%（支持iphone所有机型）**	**较小**
EVLib	25dB	2.66105	0.83659	58ms	支持iphone8及以上六核机型	

ali_denoise 算法在算法应用落地上主要具有 5 大优势：

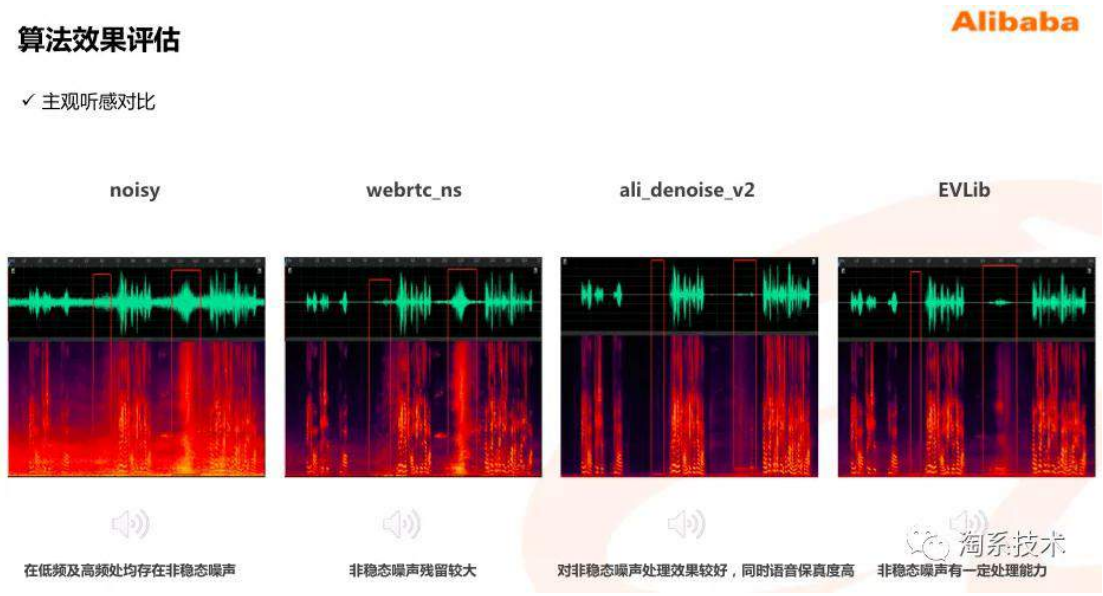
第一，利用开源的中英文纯净语音数据集、噪声集，自录噪声并进行数据清洗，同时针对淘宝直播场景构建突发瞬时噪声数据集，构建了我们私有化数据集。

第二，通过 cache buffer 策略，可实时输出增强后的语音，契合直播场景，同时支持任意延时。

第三，目前深度学习类的增强算法只支持 16khz，ali_denoise 支持 48khz 全频带语音增强处理，其中 0-16k 采用神经网络处理，16-48k 采用传统算法+神经网络混合方式处理

第四，为了保证线上音视频同步，独立智能降噪模块，和采集线程解耦，iphone6 以上机型直播无异常。

第五，由于采用了小型化模型设计和裁剪工作，性能消耗较低。



受新冠疫情的影响，互联网进入全民直播大时代，教育直播，旅游直播，助农直播，电商直播，政务直播等空前兴起。团队研发的智能降噪技术应用到阿里集团内的多个场景。在助农直播中，我们的算法有效去除田间机械的噪声，还原清晰语音；在钉钉在线课堂中，我们智能降噪结合盲源回声消除，助力 2 亿中小学生在家上课，提供高质量的声音体验；在电商直播中，我们的算法支撑了海外代购，户外运动，乐器直播，音乐晚会等全场景语音处理需求。我们秉承"让声音更清晰，让沟通更有效"的理念，追求更高质量的语音体验。

未来展望

AI+实时音频处理这个方向其实还有很多可以探索的，在前端处理算法中，除了我们已经优化的 ANS 算法外，AEC 的非线性残留部分也可以结合深度学习类算法进行优化，可以将参考信号和观测信号的频域特征 concat 起来作为输入，观测信号的类 IRM 指标作为输出，进而得到估计的近端信号；类似的任务还有啸叫抑制及后端处理算法中的带宽扩展、智能 PLC 等等。

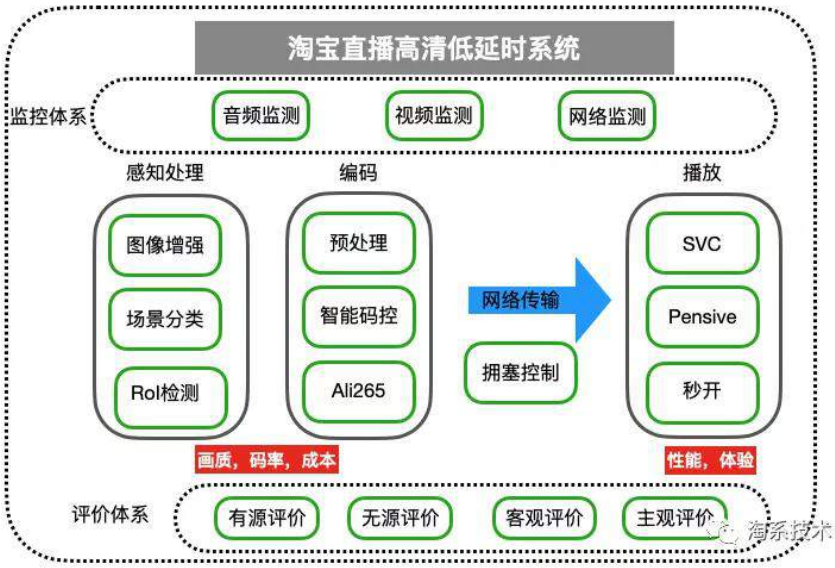
5G 时代|淘宝直播高画质低延时技术探索

作者 | 音视频技术团队
出品 | 淘系技术

前言

目前 5G 新基建正在迈入起航阶段，5G 相比目前广泛使用的 4G，它具有更高的速率，更大的容量，同时延迟更低，可靠性更高。在 5G 时代，视频得益于网络带宽的提升，未来将成为主流的传播媒介。越来越多的业务和应用将视频化，直播化。大量互动的内容将通过 5G 以低延时的方式以视频的形式传输。

5G 将对视频分辨率和清晰度提出越来越高的要求。淘宝作为一个数亿级用户的短视频与直播平台，业务多样，两端用户分布广，设备和网络情况复杂，给多媒体内容存储和分发带来巨大挑战。在内容生产过程中把控好质量和成本，在内容分发和消费过程中确保用户体验，是当前面临的主要问题。为了解决这个问题，我们有两个优化目标，一是在画质不变的前提下降码率，二是在码率不变的前提下提升画面质量。



淘宝直播高清低延时系统架构

在降码率上，我们自研高效编码器，升级播放架构，添加智能 ROI，场景编码，智能码控等工具，有效地降低了视频码率带宽。在这些技术中，高效的编码器能够在质量不变的前提下显著降低码率；场景编码能够根据不同的画面内容配置合适的编码参数；ROI 挑出画面中人眼比较关注的区域交给编码器重点编码；智能码控根据人眼主观特性，消除因为超过人眼阈值而浪费的码字。

在画质上，我们使用前处理增强，去噪，超分高动态范围等算法提高生产内容的观感质量；在体验优化上，通过低延时编码技术，在降低了编码延迟的同时损失很小的码率，增加观众和主播的体验。

围绕着提高问题发现、问题处理效率的出发点，具备数据采集、存储、异常事件收集、智能告警、告警数据运营、可编码诊断平台、故障自动化处理、变更联动等能力。我们搭建了一套基于淘宝直播的全链路监控体系，从音频，视频，网络这三个方面入手去解决目前淘宝直播全链路的现有问题以及将来可能出现的问题。不断去优化整套高画质低延时系统。

与此同时，我们建立了客观质量和主观质量评价体系，采用 vmaf，psnr，ssim 这一系列的指标作为客观质量评价。针对海量无源场景，我们还基于 cnn 建立了无源评价模型，保证无源场景下质量评价的准确性。以这些有效的评价手段来确保"画质不变"，并监控线上视频质量。

端上窄带高清

S265 编码器

带宽成本是视频服务中非常重的基础设施成本，如何在保证视频质量的前提下降低成本是整个链路中至关重要的一环。相机采集到的视频数字信号，数据带宽通常都非常高，以 720p 25fps 为例，带宽高达 263.67Mbps，很难存储和发送。好在视频图像内部，帧与帧之间存在非常高的相关性，采用压缩技术去除相关性后，可以将带宽降低到原来的 100-400 倍。视频压缩标准主要由 ISO（国际标准组织）制定的 MPEG 系列和 ITU（国际电信联盟）主导的 H.26X 系列，每隔十年时间，视频压缩标准升级带来的压缩率会提升一倍。

h265 作为比 h264 更新一代的视频压缩标准，提供了更灵活的编码结构和划分方式，并在在运动补偿、运动矢量预测、帧内预测、变换、去块滤波、熵编码等方面进行了大量改进与优化，得益于这些新的编码工具和特色技术，相同画质下最高可以比 H.264 节省一半码率，为了在不牺牲画质的前提下节省码率，h265 成为我们首选的编码标准。

Ali265 是淘宝自研的高性能 H.265 编码器，对比业界开源的 X265 可实现 BDrate20%以上的增益，对比 X264 则有 40%以上的增益。目前已在淘宝直播，优酷视频，阿里郎会议，V Mate，UC 云盘等业务中上线使用。

Ali265 JCTVC class B~F sequence

	Ali265 VS X265(RC=ABR)		Ali265 VS X264(RC=ABR)	
速度档次	BitSaving@ Same quality	SpeedRatio@ same bitrate	BitSaving@ Same quality	SpeedRatio@ same bitrate
Veryfast	-20.2%	210%	-40.7%	55%
Medium	-18%	396%	-42.3%	66%
veryslow	-21.5%	620%	-50.4%	62%

淘宝直播技术团队联合阿里云团队开发了 s265 编码器，对比业界常用的开源软件 X265，1 pass 单遍编码在相同 psnr 指标下:

- veryslow 速度档次有 28%码率节省。
- medium 速度档次有 36%码率节省。
- crf 模式与 abr 模式节约的码率接近。

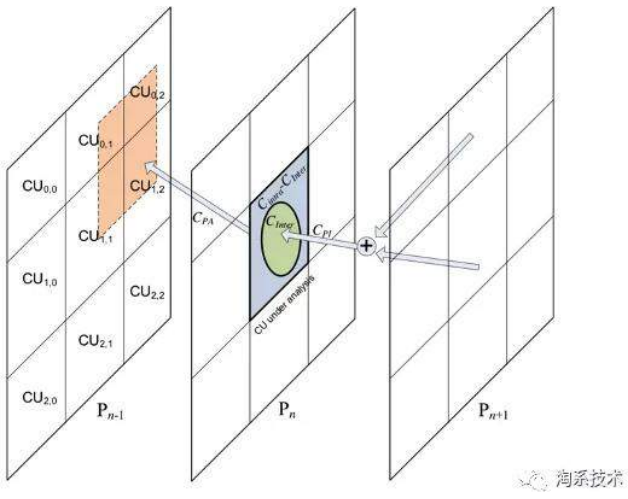
S265 编码从码率控制，编码工具两个方向优化编码质量，并从快速算法，工程算法两方面引入速度优化算法。

码率控制

为了进一步提高压缩质量，在编码器框架标准一定的情况下，编码器算法优化主要的方向是找到策略选择出最优的编码方法和编码参数，从而获得更好的码率节约。合理分配码率是编码器的一个重要工作，码率控制的目标是把码字分配到更有价值的地方，从而在目标码率下使得编码失真降到最低，或者在失真固定的前提下使码率降到最低。码控需要解决两个经典问题，一是帧级码控和块级码控根据目标码率来分配每个 GOP、帧、编码块的码字数量，二是块内编码时以最合理的方式把这些码字分配到每个编码块中。

在帧级别码控中，传统方法统计所有已编码帧的长期复杂度，根据长期复杂度及当前码率之间的比例计算出 QP，这样一来，QP 对帧复杂度越来越不敏感，导致编码质量下降或码率过剩。特别是在计算首帧 qp 时，以往算法采用了一个只和当前码率有关的经验值。我们基于 cutree 理论准确估计预分析长度中 ipb 帧的码率占比和预期编码大小，从而在编码前获得更准确的量化系数。

块级码控分配则受时域 cutree 和空域 AQ 影响。在时域上 IBP 帧的重要性是明显不同的，被后续帧参考的块，不仅影响自已本身的质量，还会影响到后续帧的质量，因此被参考更多的块需要进行高质量编码。cutree 算法根据帧内预测代价和帧间预测代价计算信息的传递比例，算出当前块对后续序列的影响程度，进而调整 qp 偏移。但考虑到在不同的噪声能量，运动强度，纹理边缘强度，以及编码参数下，不同参考块的调节为后续帧的节约比例是不一样的， 所以 s265 通过参数训练的方法，获得多个因素对传递效率的影响，得到一个更准确的信息传递比递，从而更合理地在时域上分配码率。



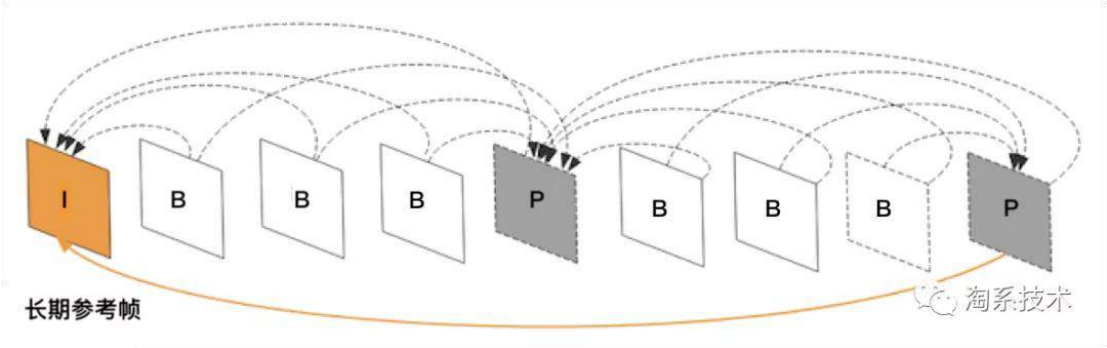
cutree 传递过程

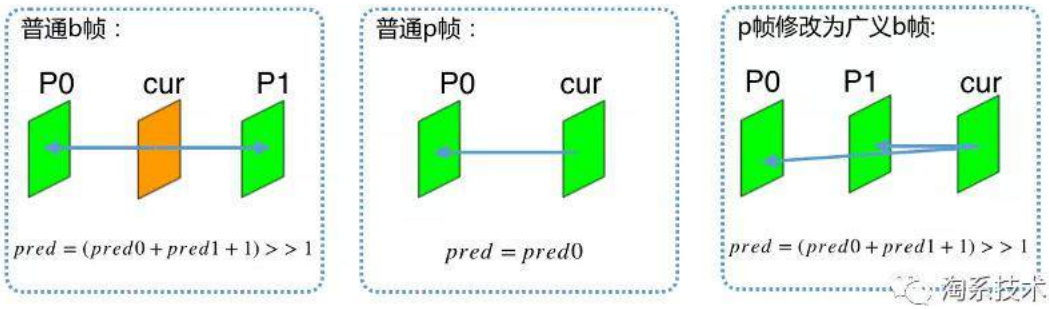
另一方面，空域上各块之间的重要程度也是不一样的。人眼是视频的最终观察者，从人类视觉系统出发，不同的块在人眼中的视觉冗余不相同，比如人眼存在视觉掩蔽效果，它对显著纹理和强边缘附近的噪声不敏感，将码率更多分配向人眼敏感的平坦区域，可以得到更好的主观质量。在编码器中，我们通过计算块的方差能量及边缘能量作为块的代价，研究不同块能量和人眼感知程度之间的关系，估计出块间码率配分对人眼注意力的影响，合理分配码率到更重要的纹理块，提高视频感知编码效率。

编码工具

在编码工具上，S265 对传统的场景切换检测，帧类型决策，SAO，DEBLOCK，两遍编码，RDOQ 等编码工具算法做了改进，并实现一批编码工具。

比如在参考帧模块，有较多的工具可以提高参考效率。首先长期参考帧和广义 B 帧等帧类型可以提高预测质量，长期参考帧针对背景很少发生变化的直播场景，它有效减少信息经过多帧传递带来的损失，引用长期参考帧可将平均 EV 提高大概 0.25dB。而传统 P 帧改为广义 b 帧，采用双向预测取代单向预测从而降低噪声，光照变化，采样误差等预测残差源。



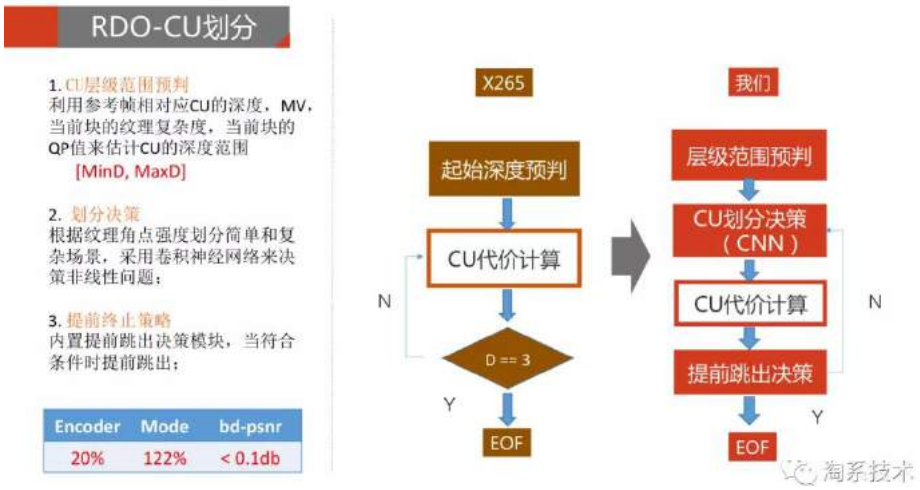


在扩充了帧类型后，我们基于参考强度做 IBP 帧帧类型决策。然后在 minigop 内部，我们使用金字塔结构的参考关系，得到比传统结构获得更短的参考距离。最后，在管理和选择参考帧时，我们考虑到静止块和运动块的区别，静止块倾向于参考质量高的帧，运动块倾向于参考时间近的帧，所以针对场景筛选出这两种类型的参考帧能得到更好的参考质量。

速度优化

HEVC 编码器带来了编码效率的提升，但很多新的编码工具都存在计算复杂度过高的问题。因此，优化编码器速度，在高端机上能打开更多的编码工具，搜索更大的编码模式空间。进一步提升编码质量。在低端机上则能降低 CPU 发烫和编码卡顿的现象。

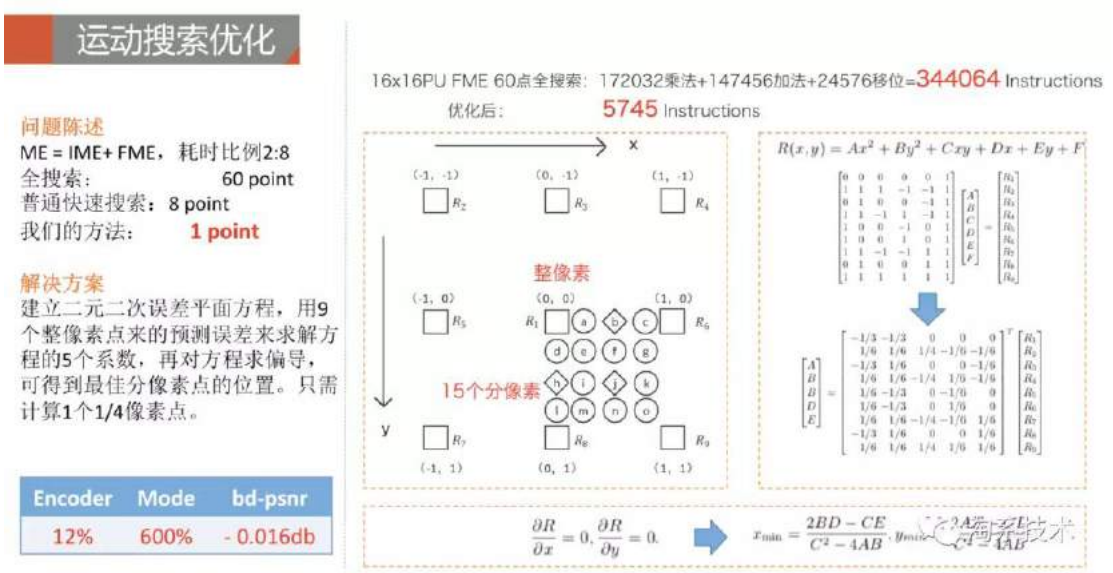
HEVC 可以将图像块从 64x64 划分到 4x4，同时块的类型模式激增，备选的编码模式数量是 h264 的数倍，块划分及模式决策因此成为一个重要的瓶颈。



所以在 RDO 中，减少 CU 划分层级的搜索次数，筛选出一些必要的层级是减少计算量的重要手段。首先利用时间和空间相关性，可以从参考块获取到一些先验信息，再结合本块的运动信息和纹理信息，分析预判出当前块 CU 层级的最大估计层级和最小估计层级。其次，在决策过程中的提前跳出策略也可以大幅降低计算量，我们根据图像纹理的平坦程度，或者各种模式下的 rdcost 对比，提前跳出当前的模式遍历。而在一些图像非线性的场景，我们通过 CNN 深度学习模型辅助决策模式。

进入决策模块的内部，同样存在大量复杂的计算。帧内预测存在 35 种模式，我们可以通过贝叶斯理论，求出最简单的几种模式后，估计出最佳模式最可能出现的位置，从而为帧内模式筛选过程提升一倍速度并将损失控制在 0.01db。另外，帧间预测的运动搜索是从参考帧寻找最佳匹配块的过程，它的分像素搜索需要做 7 抽头或者 8 抽头的插值滤波，计算量很大。我们所以可以利用整像素的信息建立二元二次误差平面方程，估算最佳分像素点的位置，避免了分像素的完整搜索过程。

在评价模式的优劣时通常采用 rdcost 作为模式的代价，它需要计算编码比特数和编码失真。这就需要将编码系数进行熵编码计算码流长度，同时还要将编码系数变换回时域求失真。为了降低 rdcost 的计算量，我们采用了失真和码率的线性估计算法，包括两个部分，其一是量化误差能量在频域计算，利用 IDCT 变换的能量不变性，计算量化余数的平方和估计失真，其二是建立编码系数特征信息和码流大小之间的线性关系，直接从系数特征信息估计出熵编码的大小。通过这个方法可以跳模式代价计算的熵编码过程以及，反变换，反量化，重建，SSE 等过程。节约了大量的计算。



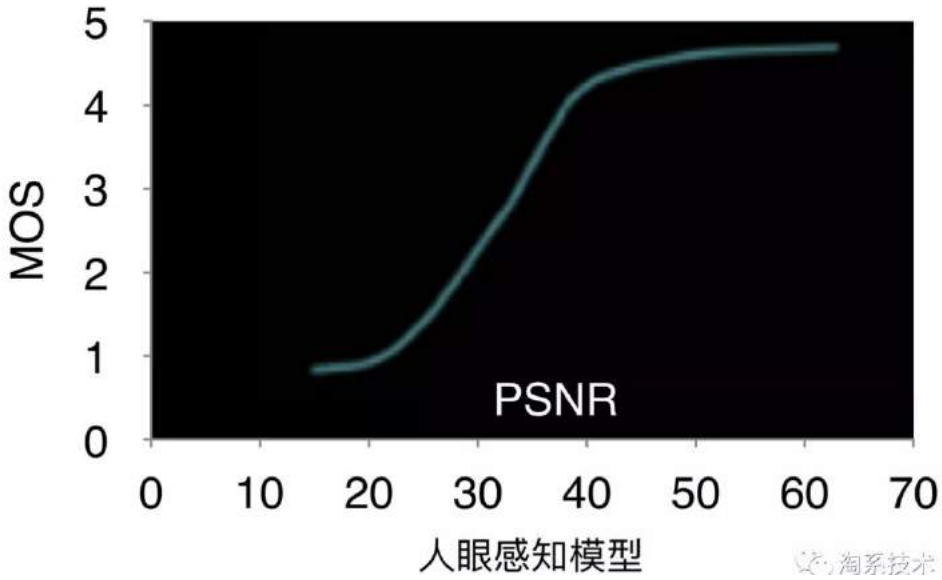
在 rdo 之外，我们还改进了 slicetype 决策算法，动态拉格朗日因子调整算法，快速 deblock 和 sao 决策等。

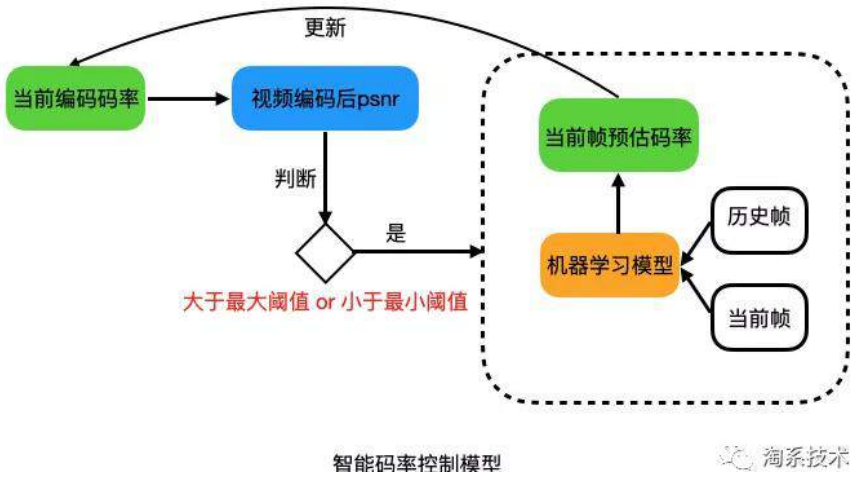
在工程优化方面我们也添加了多项优化，首先是 C 函数优化，我们通过优化流程逻辑，拆分特殊路径，合并分支，查表，循环优化等方法给 rdoq 模块，系数解析，deblock 等模块带来了接近一倍的提升；其次针对密集计算的函数我们 simd 化并优化汇编代码的执行速度。

s265 经过快速算法与工程两个层次上的优化，我们为 HEVC 编码带来了明显的性能提升。从而在低端 iphone 上实现 720P 30 帧每秒的实时编码。

智能码控

智能码控是淘宝自研的码率控制算法，普通 ABR 或 CBR 码率控制为了追求目标码率，在低复杂度场景浪费了大量码率，根据人眼主观质量模型，当 psnr 高于一定阈值后，再提高质量，人眼无法察觉，只会消耗过多码字。我们使用机器学习方法，根据 17 种历史编码信息和待编码帧的复杂度，预估出待编码帧在质量阈值以上的量化系数，并限定在 ABR 目标码率以下，确保每个帧都能以最合适的码率编码。



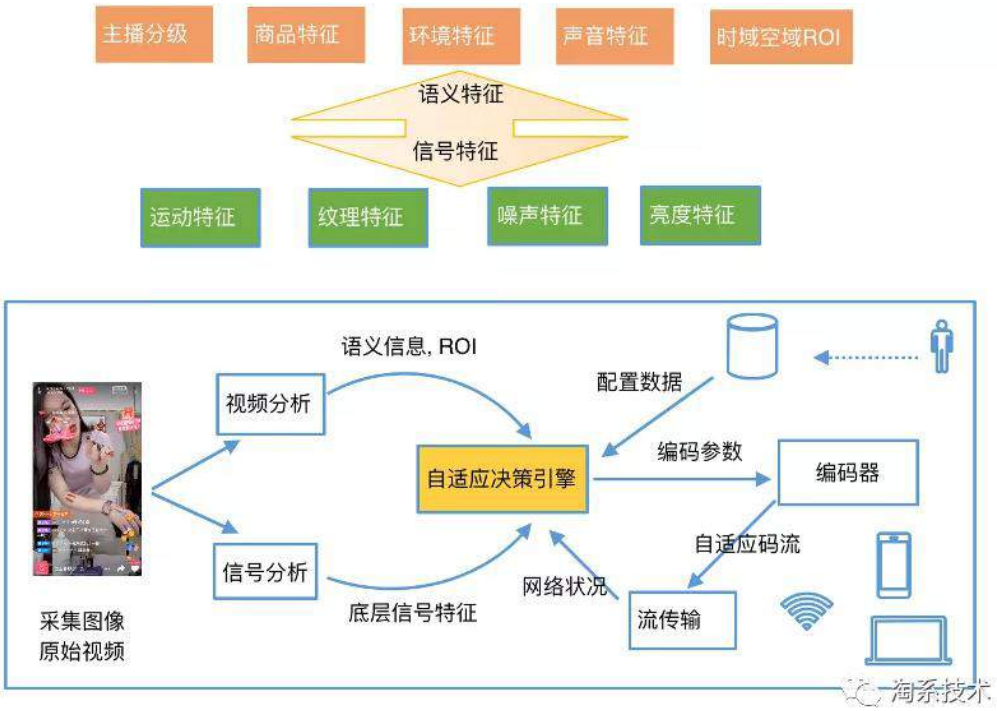


智能码率控制模型

经过淘宝直播线上验证，可达到 15%的省流，在钉钉直播中使用更是节省了 52%的带宽并降低了 62%的推流侧卡顿。

场景编码

由于当前淘宝直播种类的丰富性，各种场景下的纹理，光照，背景，运动程度都是不一样的。户外主播经常走动，画面帧变化幅度频率高。美妆主播大多坐在室内，光照基本上比较偏亮。珠宝类主播主要是拍摄物品，画面多静止不动。面对形形色色的直播场景，单一的编码器配置并不能满足当前淘宝直播的需求，开启或关闭某些编码工具对视频编码效果影响不一致，如何针对内容选择最佳参数成为业界研究的方向。



在此需求下，我们提出了基于不同场景的编码参数配置策略。首先，我们通过多个深度学习与机器学习模型对数万条各种内容的直播视频进行了数据训练分类，包含两个大的特征维度，分别是语义特征和信号特征。语义特征包含：主播分级，商品特征，环境特征，声音特征，时域空域 RoI。信号特征包含：运动特征，纹理特征，噪声特征，亮度特征。通过对不同特征种类的视频集，我们单独使用大规模服务器集进行最佳编码参数搜索，自动化高效地搜索到适合当前视频编码的最佳编码参数组合，在提升画质的同时能尽可能地减少码率消耗。并最终根据编码参数集进行聚类分为多个参数配置项。

在主播需要推流的时候，首先进行标准的编码参数配置进行推流。收集一定的数据之后，我们将得到的视频语义特征和信号特征送入自适应决策引擎，通过里面的深度神经网络进行视频分类，决策出当前视频应该下发的编码参数配置，然后我们将新的参数配置重新送入编码器进行新的推流，以此优化使主播获得当前情况下最优质的视频编码。通过此方法，我们在淘宝直播里面获得了 7-10%的 BDrate 收益。在淘拍场景下获得了 40%的 BDrate 收益。

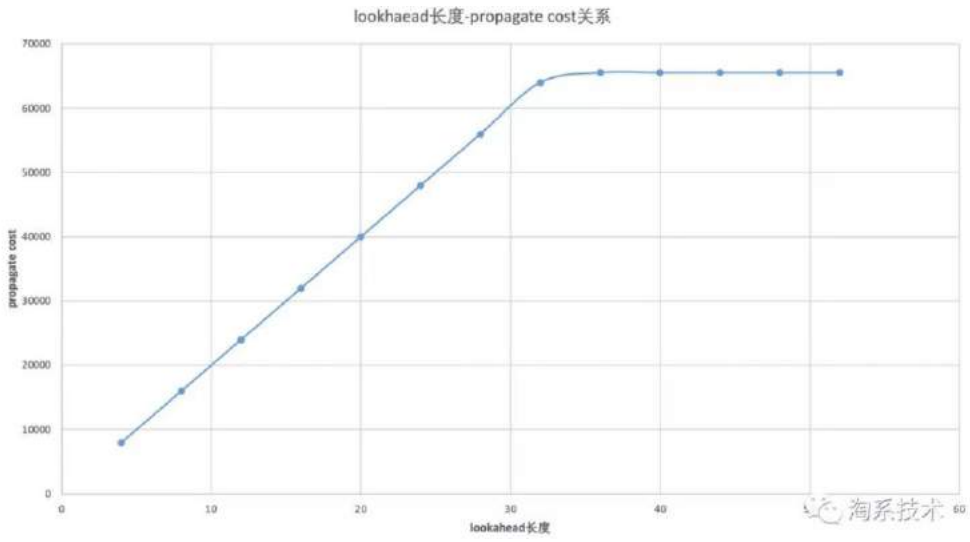
低延时编码

在直播中，低时延意味着高效率和优质体验。试想以下场景：

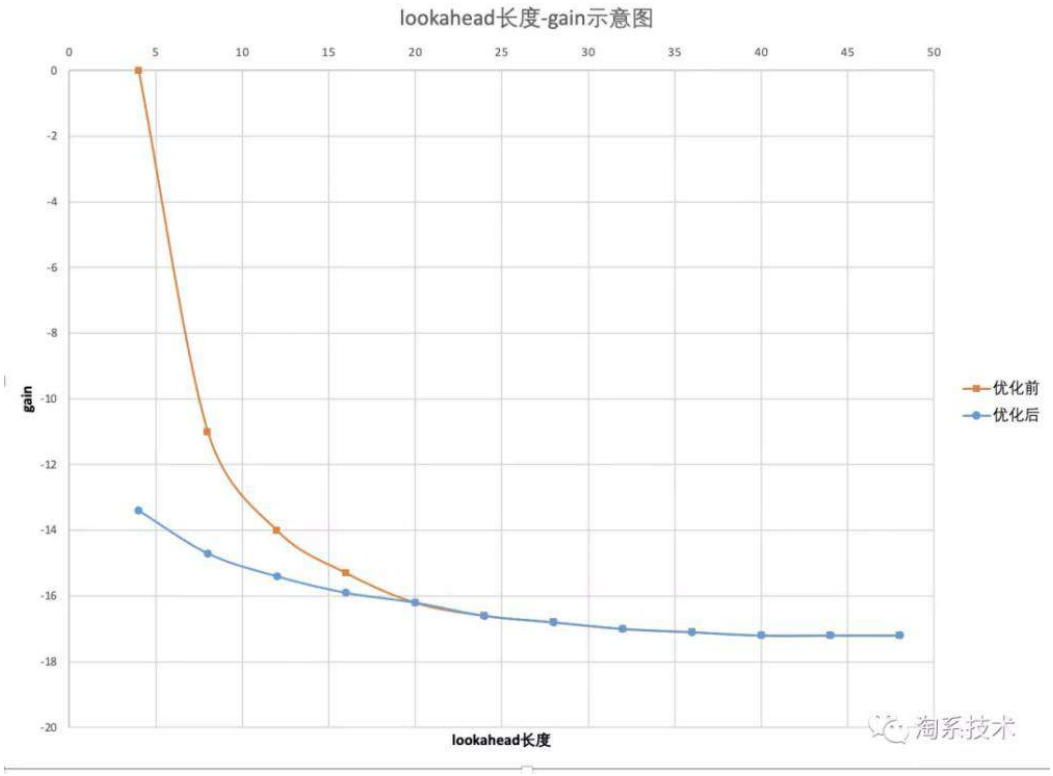
- 场景一：当主播展示下一个商品后，10 秒才收到上一个的商品的提问。
- 场景二：钉钉课堂直播中，老师提问后迟迟得不到学生的反馈，浪费部分时间。

这些场景给用户带来糟糕的体验，使得直播卖货、直播课堂效率低下。当 5G 普及，会带来更低的时延，带来更好的体验，但是当下还是 4G 为主，降低时延有很有必要。

端到端延迟主要分布在采集、编码、传输、转码、分发、播放等各个部分，这部分主要优化编码延迟。编码延迟又分为多线程导致的延迟、缓存帧数延迟、B 帧数带来的延迟等。其中编码延迟最大的一部分来源于编码器缓存，通过分析编码前的缓存图像，可以大大的增加编码效率。如果粗暴的降低编码器缓存，可以实现较低的延迟，但是质量损失比较高。所有产生了一种想法，能不能用较少的缓存去模拟较长的缓存的效果？通过分析 cutree 的原理，结合统计 lookahead 长度跟传递代价的关系，可以发现缓存长度跟传递代价很强的线性关系，如下图所示：



根据场景可以用不同的预测模型变种，最终实现用较短的 lookahead 模拟较长的 lookahead 的效果，测试在直播素材中 lookahead4 优化后比优化前可以节省 13.5%的码率，有效的降低了编码延迟，结果示意图如下。

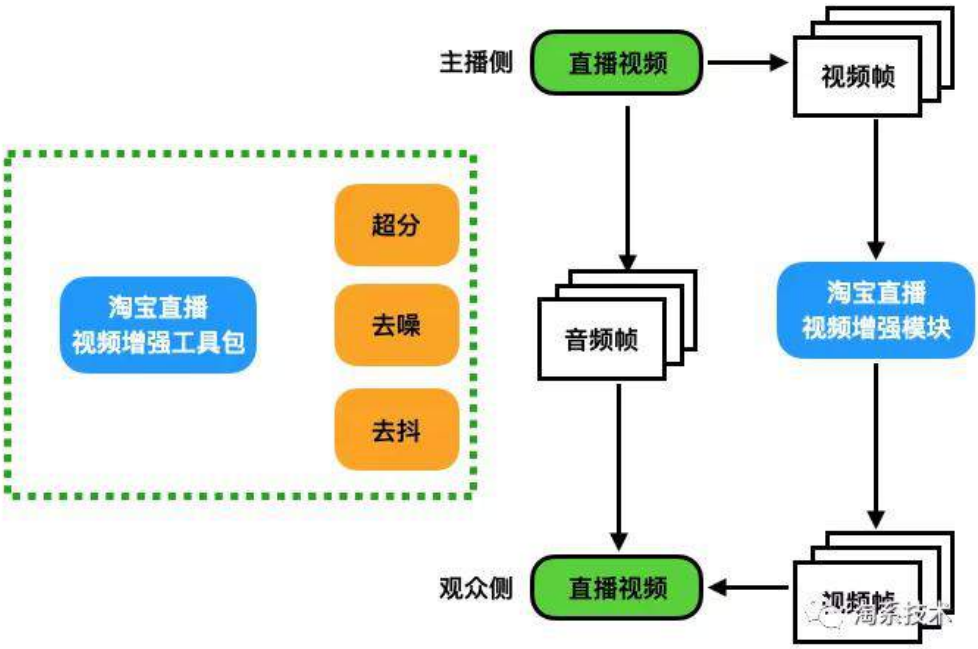


同时，在之前的测试中发现，该优化对场景不敏感，运动简单场景和运动复杂场景提升同样有效。

过去一年，我们采用前述优化，将 265 码流在画质不变的前提下，将码率从 1.4M 下降到 800K。

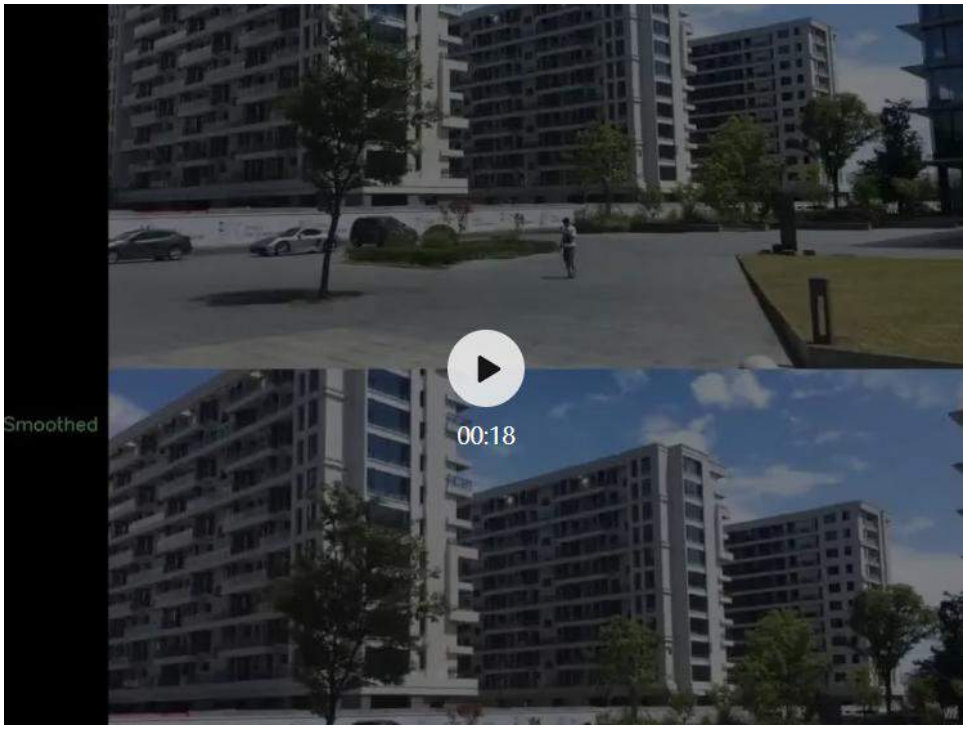
画质增强

在淘宝直播的场景中，大主播有自己的专业设备与团队，直播出来的视频与音频都是比较高质量的。但是针对中小主播，用户的行为不可控。因此产生的结果就是很多中小主播产生的视频质量比较低，收获的观众数量也比较少。针对这种情况，我们选取了用户习惯产生最严重的几种情况，对这一类主播进行了画质增加的，显著提升了用户的直播体验。



下面介绍一些已经有的应用效果。

去抖:



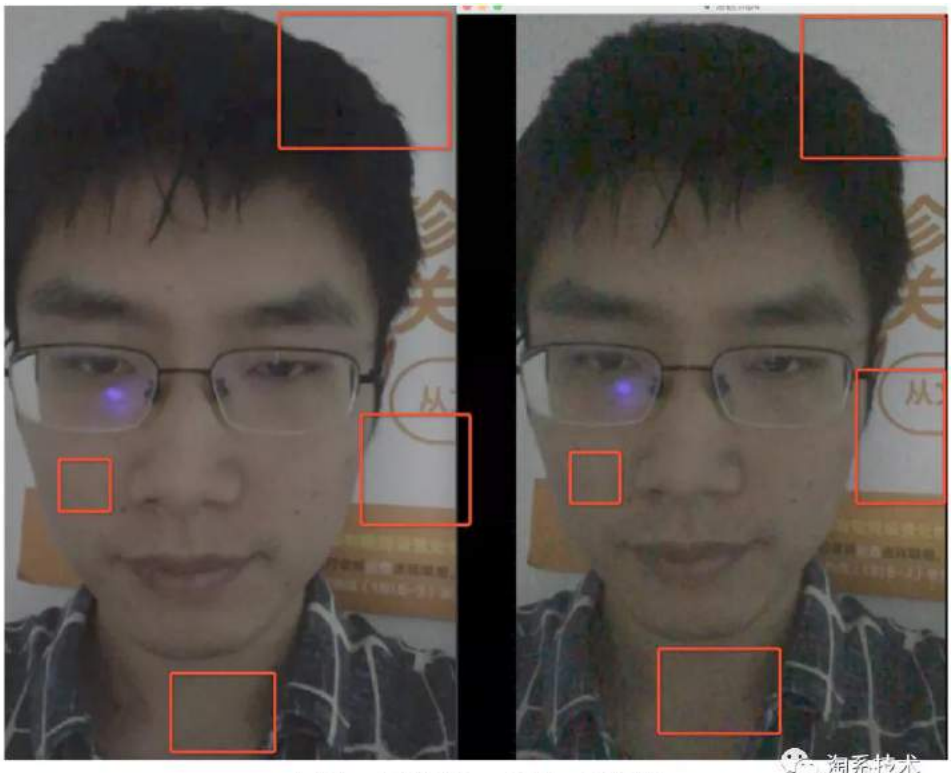
点击可跳转原文查看视频详情

现代编码器能够较好的处理平坦纹理和平移运动，前者通过帧内预测来消除空间相关性，后者通过运动搜索来消除帧与帧之间的时间相关性。但是在视频采集过程中，由于摄像机抖动产生的视频帧抖动，编码器不能够很好的处理；由于抖动剧烈的一般是中小主播，且携带的设备比较老旧，我们考虑从采集源来改善视频帧，最终在这里我们采用相机路径平滑算法来去除视频帧中的抖动。

去噪：

视频直播在灯光不太理想的情况下，摄像头采集的画面会产生明显的飞蚊噪声和高斯白噪声，严重影响用户对视频内容的感受，这种情况下，有必要对视频进行降噪。

现有的很多优秀的云端去噪算法，其实对于移动端来说采用深度学习的方法就不合适。虽然现在有很多移动端深度学习框架，但是毕竟还没有跟机器是配得非常好，针对很多中低端的手机其实跑不动这种生成模型的。基于此，我们在移动端主要是考虑效率，那么我们就采了基于维纳滤波的时域降噪算法方式来实现，进行训练和优化。



左图：去噪图像，右图：原图像

超分：

针对一些小微主播，录播设备只能支持 360p，最终观众端看到的视频会通过插值等传统方法进行放大为 720p。这样获得的视频帧难免产生模糊效果，影响直播观感。得益于深度学习在移动端的优化，我们在部分高端机实现了移动端视频帧的实时超分。

在众多的网络架构中，我们最终选择了性能最佳的 FSRCN 方案，网络的架构图如下所示。在训练过程中，我们精选了 1W+淘宝个品类的高清大图，结合业界的高清开源数据集，利用样本增强技术，训练了 5000 轮左右的模型达到收敛效果。此外，为了消除图像分块带来的边界效应，我们做了图像重叠合并的操作，在增加部分计算时间的情况下带来了更好的超分效果。

为了在手机端实时运行，避免占用过多资源，我们优化反卷积计算，并针对人眼视觉特性，对强纹理和静止区域部分像素进行超分，以此大幅提高移动端的效率。

4K 实战

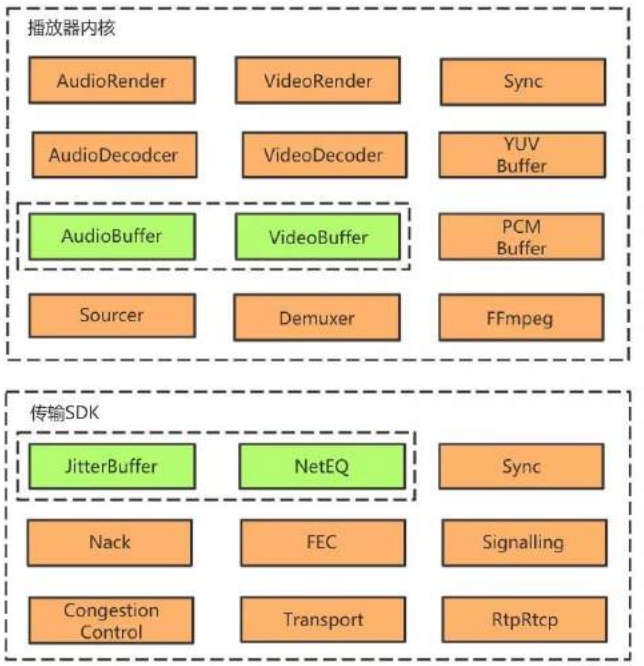
淘宝直播与中国电信浙江公司合作了全国首场 5G 电商直播，利用 5g 直播下的超高清画质和超低时延给消费者更好的线上购物体验。消费者除了可以看到直播间的整体情况，还能够远程自主对局部进行放大，非常清晰地看到商品的细节信息。让线上购物具有逼真的线下现场购物感受。

未来凭借 5G 通信技术与极致的编码优化和画质优化的结合，能给直播带货体系带来深层次的改变。4K，VR，AR 等技术的成熟将会给直播场景带来更多产品和更好的体验，带来更多新型的应用场景，沉浸式体验和全画幅体验使用互动感和真实感更强。通过技术的赋能，淘宝直播间的可看性越来越强，淘宝直播所能覆盖的场景、所能覆盖的领域也会越来越广。

低延迟传输

低延迟播放器

常规播放器的延迟分析。目前基于 TCP 的直播传输技术主要有 HLS 和 RTMP/HTTP-FLV 两个协议，其中 HLS 直播的延迟一般在 10 秒以上，HTTP-FLV 直播的延迟一般在 6 到 9 秒，从推流、cdn 分发到播放的整个直播链路看，延迟的大头来自播放端。在播放器中，几乎每个线程都有自己的缓冲区，这些缓冲区的作用是平滑整个播放链路的抖动，它们的大小决定了播放过程中的播放延迟和播放的流畅性。VideoBuffer 和 AudioBuffer 用来存放待解码的音视频 packet，该缓冲区是为了平滑网络的抖动，推流、CDN 传输和播放下载的抖动都会堆积到播放端，这是常规播放器延迟最大的一个产生点，为提升直播的整体流畅度，缓冲区延迟一般在 5 秒以上。



低延迟播放器架构

基于 TCP 的媒体传输并不适用于低延迟直播场景，主要原因如下：

- 重传慢，TCP 追求的是完全可靠性和顺序性，丢包后会持续重传直至该包被确认，否则后续包也不会被上层接收，且重传超时时间一般 200ms，会造成接收侧帧抖动。
- 上层无法针对优化，TCP 拥塞控制和 Qos 策略在操作系统内核层实现。
- 拥塞判断不准确，基于丢包的拥塞控制跟实际网络情况不符，丢包并不等于拥塞，也会造成发送链路 bufferbloat，链路 RTT 增大。

我们的低延迟传输 SDK 是基于 WebRTC 打造的，使用了 WebRTC 的几个核心模块，包括 RTP/RTCP、FEC、NACK、NetEQ、JitterBuffer、音视频同步、拥塞控制等。NetEQ 和 JitterBuffer 分别是音频和视频的网络抖动缓存区，这是传输 SDK 延迟最大的一个产生点。RTP over UDP 能够更好地对抗公网的丢包，结合自适应缓存和 Qos 优化，确保直播整体流畅度的条件下，我们的 JitterBuffer 的缓冲区延迟能够控制在 700 毫秒以下，直播观看延迟在 1 秒左右。

播放器对低延迟传输 SDK 的接入适配。我们对低延迟传输模块封装了 FFmpeg 的扩展 demuxer，将支持低延时传输协议的 demuxer 注册到 FFmpeg，播放器通过 FFmpeg 打开网络连接读取数据，这种接入方案基本不影响播放器原有逻辑，对播放器改动较小，主要改动点如下：

- 缓冲区大小控制。使用低延迟传输协议拉流时，网络的抖动缓冲区是底层传输模块的 JitterBuffer，播放器层的 JitterBuffer 的缓存应设置为 0 秒，否则会引入多余的延迟。
- 卡顿统计修改。一般播放器根据缓冲区水位大小判断卡顿事件，当缓冲区为空或持续空一段时间，这会导播放画面卡顿，同时触发卡顿事件，播放器的 JitterBuffer 被低延迟传输 SDK 接管后，卡顿事件也应该由低延迟传输 SDK 触发。
- 音频解码流程。从 NetEQ 获取的音频已经是 PCM 数据了，播放器读取的音频数据可直接渲染，如果音频使用硬解，可能会出现解码兼容问题，现象是听不到声音，使用 FFmpeg 软解也是可以兼容的。

低延迟服务器

低延迟传输是一个综合性的问题，要从整体入手，不仅要从设计上考虑，还需要客户端，服务器，数据系统紧密配合。从传输协议设计上采用 rtp/rtcp 方案。基于 udp 半可靠传输，技术成熟，更加适合音视频场景。难点在于既要降卡顿，也要降延迟。我们使用的整体算法策略如下：

- 拥塞控制

拥塞控制 gcc&bbr 算法针对直播场景深度优化，同时兼顾秒开和延迟。

- 分层丢帧

基于 B 帧的 SVC 算法和丢 gop 策略在网络拥塞时保证快速降低码率，解决拥塞。

- 重传控制

重传控制既要抑制重传风暴，也要保障快速重传。

- 平滑发送优化

平滑发送策略防止网络突发，平滑流量。同时针对秒开场景深度定制。重新设计发送机制和算法，发送性能大大提高。

- 秒开优化

服务器和端配合的多种秒开策略，保证极速开播。淘宝直播大盘平均秒开率 94%以上。

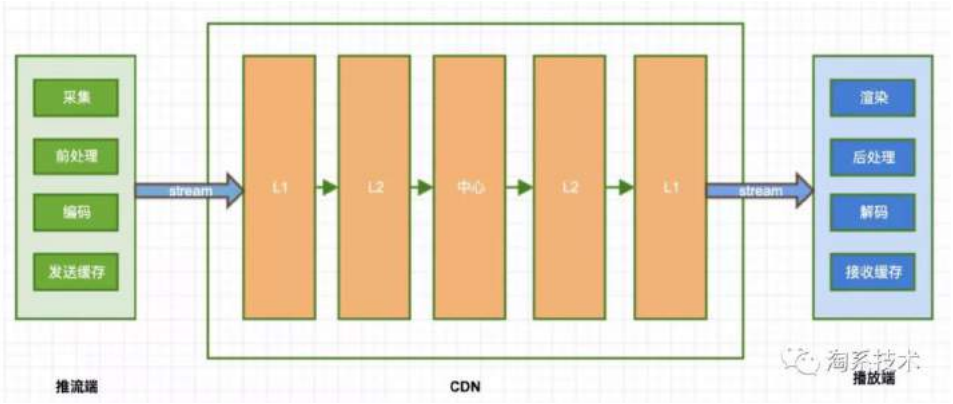
- 信令优化

从信令设计上采用 rtcp app 私有协议，和音视频传输使用一个 socket 连接。建联协议更加精简，保障 1RTT 快速给出媒体数据。

除此之外还进行了大量策略到算法上的改进和优化。上面整体策略，基于数据驱动，针对场景不断迭代优化。

端到端全链路分段统计

我们设计的端到端延迟分段统计系统，既能统计单次播放的延迟，也能统计每个阶段延迟。不依赖 ntp 时间，适合超大规模网络。通过分析不同平台推流端，服务器，播放器各个阶段的延迟情况，大盘展示出来，可以针对专项做优化。



面向未来

伴随着 5G 网络的提速，主播侧到用户侧的延时将会越来越短；移动端本身的性能提升，各种画质增强，图像渲染技术也会慢慢硬件化。移动端的深度学习模型也逐渐变得轻量化。这使得学术界各种越来越先进的创新也得以工程化。近期淘宝直播推出的智能虚拟主播也圈起了一波粉。将来，越来越有趣的玩法也会逐渐推出，使淘宝直播不再仅仅只限于“卖货”。更可能有更多有趣的玩法，例如：主播观众实时游戏互动，虚拟主播完全代替真人直播，观众沉浸式地通过 VR 或者 AR 进行直播购物。这些都会慢慢变成现实。淘宝直播的技术将来会为用户带来越来越丰富的直播购物体验。

音视频精彩问答

Q: WebRTC 有前途吗?

作者 | 淘系技术

出品 | 淘系技术

> 苹果一直不在 safari 上支持 WebRTC，目前也没什么开源方案可以在 iphone 上使用 WebRTC。大家觉得 WebRTC 的前途如何?

在直播大趋势下，现在的 WebRTC 开源软件还不能很好地支持直播。

基于 WebRTC ，我们淘宝直播做了一些方案改造，实现了一秒内的低延迟直播。本文核心内容是对比下各传输方法间的优缺点，以及我们对 WebRTC 的创新实践和未来展望。

低延迟直播技术选型

上图是我们基于 UDP 的两种方案对比，第一种是可靠 UDP 方向，比如 Quic ，另一种是 RTC 方案，比如 WebRTC 。

我们从五个维度对两种方案做了对比：

- 传输方式：Quic 是可靠传输；而 RTC 是半可靠传输，特定情境下可对音视频有损传输，可有效降低延迟。
- 复杂度：Quic 的复杂度非常低，相当于将 TCP 接口换位 Quic 接口即可，RTC 方案的复杂很高，涉及一整套的协议设计和 QOS 保障机制。
- 音视频友好性：Quic 不关心传输内容，对音视频数据透明传输。RTC 对音视频更友好，可针对音视频做定制化优化。
- 方案完备性：从方案完备性方面来讲，Quic 是针对传输层优化，而 WebRTC 可提供端对端优化方案。
- 理论延迟：经我们实验室测试以及线上数据分析，WebRTC 方案的延迟可以达到 1 秒以内。

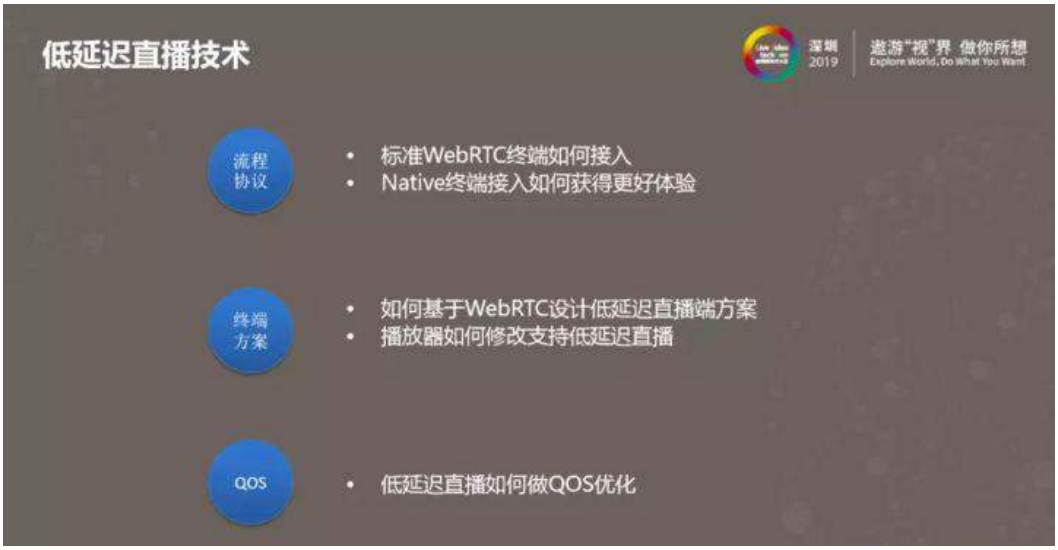
综上，Quic 方案的最大优点是复杂度低，不过这个方案要想达到更低的延迟，也需要引入更多的复杂度。从方案的先进性上看，我们选择 WebRTC 技术作为我们的低延迟方案。同时，我们也相信，RTC 技术和直播技术的融合，是未来音视频传输技术的一个趋势。

阿里云 RTS

RTS 是由阿里云和淘宝直播共建的低延迟直播系统，此系统分两大块：

- 上行接入：可接入三种输入方式，第一种是 H5 终端，使用标准 WebRTC 推流到 RTS 系统中；第二种是 OBS 等传统 RTMP 推流软件，使用 RTMP 协议推流到 RTS 系统中；第三种是低延迟推流端，可以使用我们基于 RTP/RTCP 扩展的私有协议推流到 RTS 系统中。
- 下行分发：它提供两种低延迟分发，第一种是标准 WebRTC 分发，另一种是我们在 WebRTC 基础上的私有协议扩展，淘宝直播目前使用最多的就是基于私有协议的分发。

低延迟直播技术



下面我将重点从流程协议，终端方案介绍低延迟直播技术，主要回答几个问题：

- 标准 WebRTC 终端如何接入
- Native 终端接入如何获得更好体验
- 如何基于 WebRTC 设计低延迟直播端方案
- 播放器如何修改支持低延迟直播

标准 WebRTC 接入流程

播放流程描述：

- 播放端端发送接入请求：通过 HTTP 发送 AccessRequest ，携带播放 URL 和 offer SDP。
- RTS 收到播放的接入请求后，记录 offerSDP 和 URL ，然后创建 answerSDP，生成一次会话 token 并设置到 SDP 的 ufrag 字段，通过 http 响应发送给客户端。
- 客户端设置 answerSDP，发送 Binding Request 请求，请求中 USERNAME 字段携带 answerSDP 中的 ufrag（即 RTS 下发的 token ）。
- RTS 收到 Binding Request，根据 USERNAME 中的 token，找到之前 HTTP 请求相关信息，记录用户 IP 和端口。 借助 Binding Request 的 USERNAME 传递 token 是由于 RTS 是单端口方案，需要根据 UDP 请求中的 token 信息确定是哪个用户的请求。传统的 WebRTC 是根据端口区分用户，RTS 为每个用户设置端口会带来巨大的运维成本。

标准 WebRTC 接入过程会有各种限制：

- 它不支持直播中常用音频 AAC 编码和 44.1k 采样率。
- 其它不支持视频 B 帧、H265 等编码特性，多 slice 编码在弱网下也会花屏。
- WebRTC 建联过程耗时过长，会影响秒开体验。

基于以上的这些问题，我们设计了更为高效、兼容性更好的私有协议接入。

私有协议接入流程

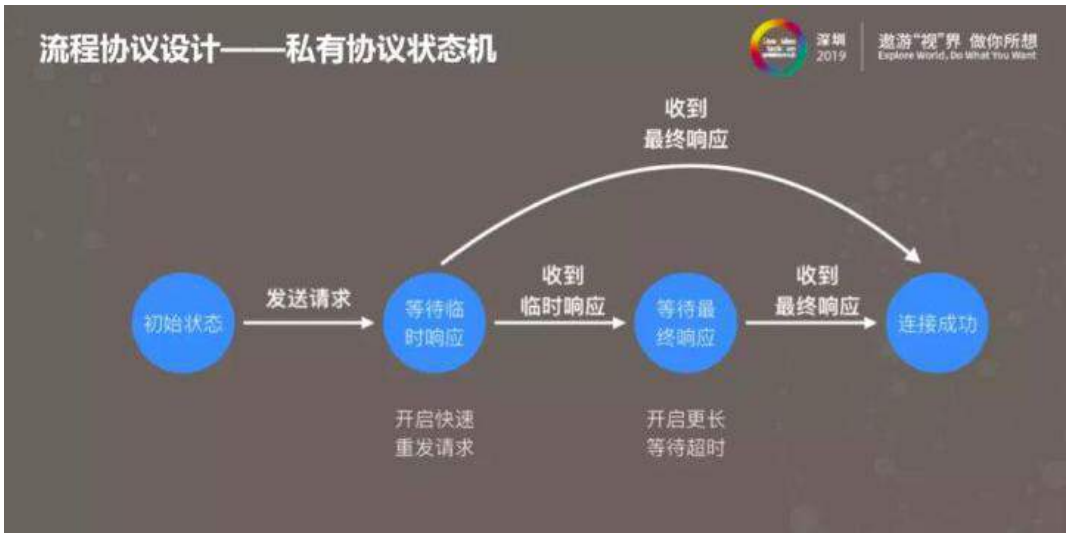
播放流程描述（四次握手建联）：

- 发送播放请求：通过 UDP 发送 PlayRequest ，携带播放 URL ；
- RTS 收到播放请求后，立即返回临时响应，并且开始回源；
- RTS 回源成功后，发送最终响应，携带相关媒体描述（编解码等）；
- 客户端发送最终响应 ACK，通知服务端最终响应接收成功；
- RTS 发送媒体数据：RTP/RTCP，连接建立成功；

对流程的几点说明：

- PlayRequest 的作用是将 URL 告诉 CDN，同时兼具 UDP 打洞功能；
- 协议中信令和媒体在一个 UDP 通道传输；
- 四次握手流程设计，保证建联速度的同时，确保重要信息可靠到达；
- 整个建联过程只有一个 RTT，建联速度快；

私有协议状态机设计



上图是私有协议的流程状态机：

- 初始状态下发送播放请求，然后会进入等待临时相应状态；
- 在临时响应状态会启动毫秒级快速重发定时器，超时未收到临时响应则快速重传播放请求，保证建联速度；
- 收到临时响应后进入等待最终响应状态，这时会开始更长的秒级定时器；
- 收到最终响应建联成功；

过程中临时响应可能丢失或乱序，可能出现提前收到最终响应的情况，会直接从等待临时相应状态直接进入最终状态。

私有协议信令设计

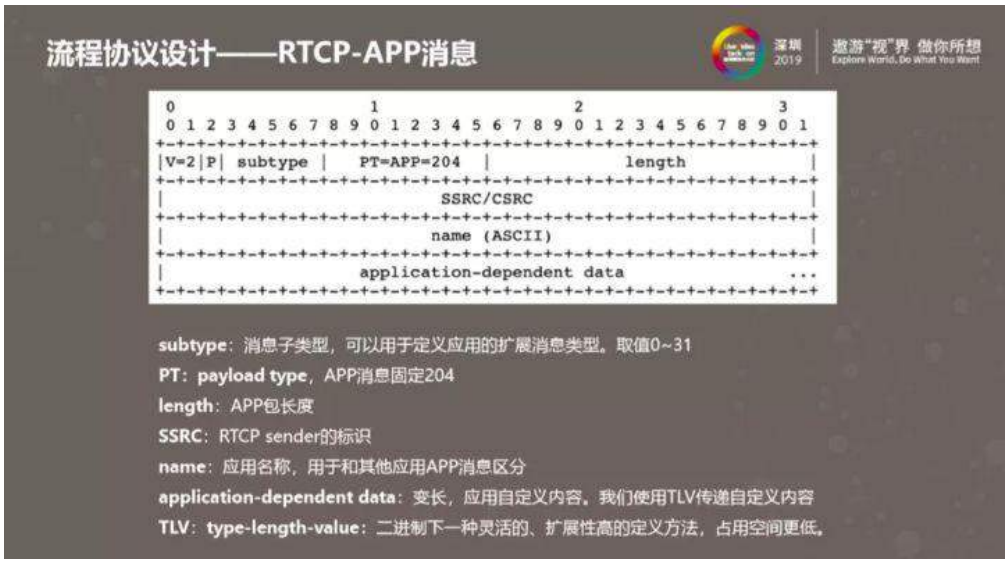
私有信令我们选择使用 RTCP 协议。原因是 RFC3550 定义了 RTCP 的四个功能，其中第四项可选功能描述为：RTCP 可以用在"松散控制"系统中，传递最小会话控制信息。比如，标准定义了 BYE 消息，用于通知源已经不再处于激活状态。

我们在此基础上扩展了建联的信令消息，包括请求、临时响应、最终响应以及最终响应 ACK 。

同时，我们选择 RTCP 消息中的 APP 消息承载我们的私有信令。APP 消息是 RTCP 提供的一种为新应用、新功能使用的一种扩展协议，即它是 RTCP 提供的一种官方扩展方式，应用层可以自己定义消息类型以及内容。此外，选择此协议也基于以下考虑：

1. 使用 RTCP-APP 可以解决私有协议和标准 RTP/RTCP 区分问题。如之前所讲，媒体和信令在同一个通道，服务端收到后如何区分私有协议、RTCP 包以及原生 RTCP 包，RFC3550 有清晰的描述，帮助我们解决了这个问题。
2. 使用此协议可以直接使用现有包分析工具解析抓包。
3. 我们可复用 RTCP 相关定义，例如 payload type、subtype、ssrc 等。

RTCP-APP 消息介绍



介绍下 RTCP-APP 消息的细节，上图是 RTCP-APP 消息头，主要字段如下：

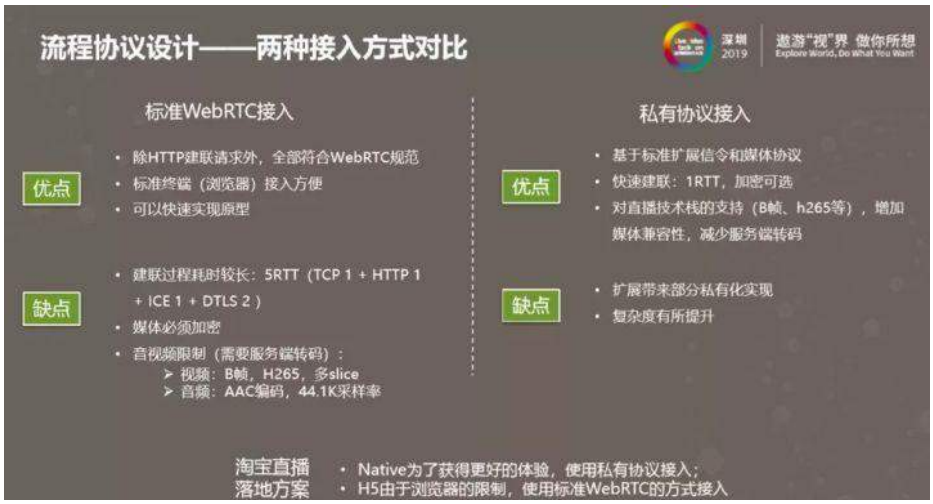
1. subtype 消息子类型，可用于定义私有应用扩展消息，我们私有信令的请求、临时响应、最终响应都是根据 subtype 区分的。subtype 的取值范围是 0 到 31，其中 31 预留将来做扩展的消息类型。
2. payload typeAPP 固定 payload type 是 204。可用于区分其它 RTP 和 RTCP 消息。
3. SSRCSSRC 是 RTCP sender 的标识。
4. Namename 是应用名称，用于区分其它应用 APP 消息。RFC3550 中描述消息类型用两个字段区分，name 确定应用类型，subtype 用于区分消息类型，同一个 name 下可有多个 subtype 类型。
5. application-dependent data 应用层扩展消息内容，我们使用 TLV 格式，即 type、length、value，是一种灵活的、扩展性高的二进制数据格式，它占用空间低。

私有协议媒体部分设计

媒体部分协议主体遵循标准 RTP/RTCP 规范以及 WebRTC 的相关规范，其中 H264 采用 rfc6184 规范，H265 采用 rfc7798 规范。

对 RTP 的扩展部分使用标准的 RTP 扩展，为了和 WebRTC 兼容，标准 RTP 扩展头部定义使用了 rfc5285 标准。

两种接入方式对比



标准 WebRTC 接入的优点：

- 标准 WebRTC 接入除了 HTTP 建联请求外，全部符合 WebRTC 规范。
- 标准终端方便接入。
- 可快速实现原型。

标准 WebRTC 接入的缺点：

- 建联过程耗时长，使用 HTTP 情况下达到 5RTT，选用 HTTPS 会更长。
- 媒体必须加密传输。
- 音视频有相关限制，使用时需要在服务端转码。

私有协议接入优点：

- 基于标准扩展信令和媒体协议，与标准协议差异很小。
- 建联速度快，秒开体验非常好。
- 支持直播技术栈，增加了媒体兼容性，减少了服务端转码成本。

私有协议接入缺点：

- 虽基于标准扩展，但仍然带来了部分私有化实现。
- 使用私有协议后，复杂度有所提升。

淘宝直播落地方案中，为了能够获得更好的体验，Native 端我们使用私有协议接入，目前已在线上大规模运行。

流程协议设计原则

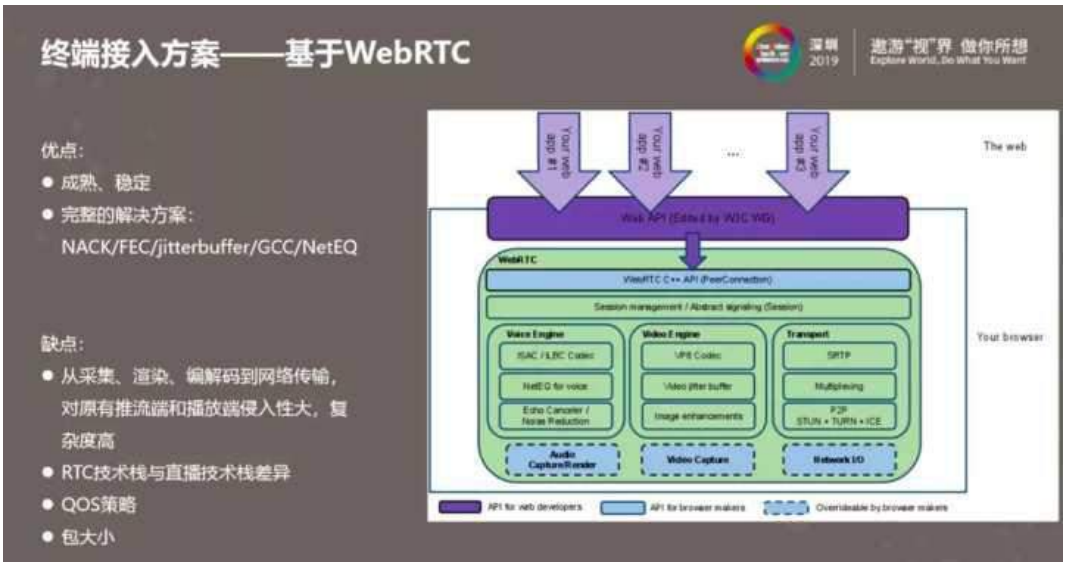
流程协议设计原则有三个：

1. 尽量使用标准，包括 WebRTC 相关规范。这个原则意味着我们和标准 WebRTC 互通，会非常方便。
2. 当标准和体验发生冲突时，优先保障体验。
3. 当需要扩展时，基于标准协议扩展，并且使用标准扩展方式。

我们并没有将 RTP/RTCP 协议全部推翻，使用完全的私有协议，有两个原因：首先是工作量问题，重新设计的工作量会比使用标准协议多很多。其次， RTP/RTCP 协议设计非常精简，久经考验，自己设计不一定能考虑到所有问题。所以我们选择基于标准扩展而非重新设计。

终端接入方案

基于 WebRTC 全模块的接入方案



基于 webrtc 全模块的接入方案，使用 webrtc 的所有模块，通过对部分模块的修改，实现低延迟直播功能。

这个方案的优缺点并存：

优点：

- 经过多年发展，它非常成熟，很稳定，同时提供了完整的解决方案，包括 NACK、jitterbuffer、NetEQ 等可直接用于低延迟直播。

缺点：

- 它的缺点也很很明显。如上图中是 WebRTC 整体架构，它是一个从采集、渲染、编解码到网络传输的完备的端对端方案，对现有推流端和播放端侵入性极大，复杂度极高。
- RTC 技术栈和直播技术栈存在差异，他不支持 B 帧、265 等特性。在 QOS 策略方面，WebRTC 的原生应用场景是通话，它的基本策略是延迟优于画质，这个策略在直播中不一定成立。
- 包大小：所有 webrtc 模块全部加入到 APP 中，包大小至少增加 3M。

基于 WebRTC 传输层的接入方案

我们目前终端的整体接入方案如上图，也是基于 WebRTC，但是我们只使用了 WebRTC 几个核心传输相关模块，包括 RTP/RTCP、FEC、NACK、Jitterbuffer、音视频同步、拥塞控制等。

我们在这些基础模块上进行了封装，将他们封装成 FFmpeg 插件注入到 FFmpeg 中。之后播放器可直接调用 FFmpeg 相关方法打开 URL 即可接入我们的私有低延迟直播协议。这样可极大减少播放器和推流端的修改，降低对低延迟直播技术对原有系统的侵入。同时，使用基础模块也极大减少了包大小的占用。

播放器针对低延迟直播的修改

上图是普通播放器的架构。播放器使用 FFmpeg 打开网络连接，读取音视频帧后会放入播放器缓冲，之后会依次对它进行解码、音视频同步及渲染。

接入低延迟直播系统后，整体架构如图下面部分：FFmpeg 增加低延迟直播插件支持我们的私有协议；将播放器的缓冲设置为 0 秒，FFmpeg 输出的音视频帧直接送入解码器进行解码，然后同步，渲染。我们将播放器原先的缓冲区移动到了我们的传输层 SDK 中，使用 jitterbuffer 可以根据网络质量好坏动态的调整缓冲区大大小。

整个方案对播放器的修改很小，基本不影响原有播放器的逻辑。

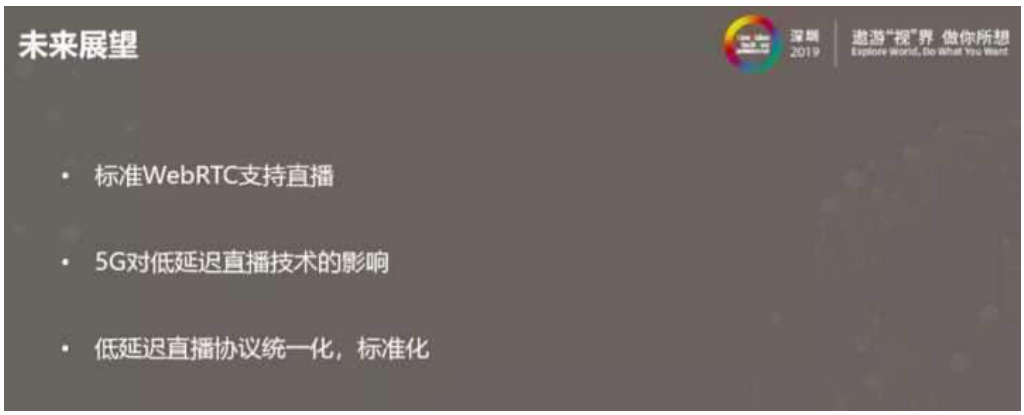
低延迟直播业务价值

低延迟直播技术目前已在淘宝直播中大规模应用，它的上线降低了淘宝直播的延迟，提升了用户的互动体验，这是最直接的价值。

所有的技术优化都是为了创造业务价值，所有体验的提升应该对业务提升有帮助。我们经过线上验证发现，低延迟直播对电商直播的成交有明显的促进作用，其中 UV 转化率提升 4%，GMV 提升 5%。

同时，低延迟直播技术也可支持更多业务形态，例如拍卖直播，客服直播等。

未来展望



我们对低延迟直播技术的未来展望有三点：

1. 现在的 WebRTC 开源软件还不能很好支持直播，希望未来的标准 WebRTC 能很好直播，这样我们可以很方便的在浏览器上做低延迟直播。
2. 5G 到来后，网络环境会越来越好，低延迟直播技术会成为直播行业未来的一个技术方向。
3. 现在各厂商的低延迟直播协议大都存在私有协议，对用户来说，从一个厂商切换到另一个厂商时成本会很高。低延迟直播协议的统一、标准化对直播行业非常重要。一个基本判断是，随着低延迟直播技术的方案和普及，低延迟直播协议在未来一定会走向统一化和标准化。也希望我们国家的技术厂商能够在推动低延迟直播标准化的过程中发出自己的声音，贡献自己的力量。

端智能与 MNN 篇

端智能作为一项创新技术，依托淘系丰富的业务场景，过去 3 年多获得了快速发展。技术体系逐渐完善，从端侧推理引擎 MNN 到端 AI 研发范式，通过 MNN 工作台解决算法模型部署全链路问题。业务应用不断拓展，覆盖视觉、推荐、语音等多种算法领域，端智能技术已经逐渐成为业务创新的核心技术推动力之一。我们一直保持开源开放的态度，希望跟行业一起推动端智能方向的发展。更多信息可以访问我们官网：www.mnn.zone。

——阿里巴巴资深无线开发专家 吕行

技术经典总结

阿里端侧 AI 从探索尝试到逐步展开

作者 | 吕承飞（吕行）

出品 | 淘系技术

人工智能发展已进入"落地为王"阶段，端侧 AI 相比云侧 AI，具有低延时、保护数据隐私、节省云端计算资源等优势，现已成为端侧技术新热点，并且紧贴用户在 AR 特效、搜索推荐等方面有诸多创新应用。我们有幸邀请到吕承飞（吕行）阿里巴巴淘系技术部资深无线开发专家来为我们介绍端侧 AI 的现状以及在阿里的实践。

本文将回顾端侧 AI 的发展过程，以阿里端侧 AI 发展为例展望端侧 AI 的未来。 在将于 11 月份举行的 GMTC 北京 2020 上，吕承飞（吕行）老师作为"端侧 AI "的专题出品人，策划了端侧 AI 专题的方向，将为大家介绍前沿的端侧 AI 创新应用、端侧 AI 技术、端侧 AI 开放能力，让大家在听完之后能结合自己业务快速上手，创造无限可能。

端侧 AI 介绍和发展回顾

端侧 AI 正从尝试应用变成驱动业务创新的核心推动力之一

AI 在智能手机和智能设备中应用越来越广泛，比如短视频 App 中的 AR 特效、人脸门禁等。随着端上算力不断增强和算法快速发展，特别是模型压缩技术和小模型算法设计不断成熟，在端侧运行算法模型成为可能。端侧 AI 简单说就是在终端设备做机器学习算法应用，这里"终端设备"主要是指手机，当然也包括不断出现的各种智能设备和 IoT 设备。目前，端侧 AI 主要是推理运行，当然也慢慢出现了在终端设备做训练，比如联合学习、迁移学习等。相较于云侧 AI，端侧 AI 具有低延时、兼顾数据隐私、节省云端计算资源，以及不依赖网络提供稳定服务等显著优势。

自 2017 年以来，端侧 AI 在底层技术和业务应用等方面都取得了快速发展，逐渐从尝试性应用变成驱动业务创新的核心推动力之一。基于端侧 AI 的人脸检测、人体姿态、手势等算法补齐了 AR 特效中的交互能力，从而可以实现各种有趣好玩的 AR 应用，且在短视频 App 中广泛应用。基于端侧 AI 的实时用户感知和理解能力，对于搜索推荐、安全风控、系统优化等业务都有帮助。在智能硬件 IoT 领域、人脸考勤机、 智能门禁锁，以及车载 ADAS 等应用也都跟端侧 AI 相关。

从技术发展来看：

- 端侧推理引擎逐步成熟，基本解决了算法模型在端侧能不能跑的问题，而且国内框架相比国外具有普遍的优势。国内开源的 NCNN、MNN、MACE 以及 Paddle-Lite 等经过不断打磨优化已经做的非常不错。国外 TFLite 和 Pytorch 也开始重视端侧推理，重点投入，性能等提升明显。
- 除推理引擎之外，端侧 AI 应用和部署仍存在较高门槛，行业公司逐步有一些尝试。比如，阿里淘系在 2020 年 3 月开放 MNNKit，包含人脸检测、手势识别等算法能力。百度和小米也有计划开源多种算法模型，比如，百度最近开源的含口罩人脸检测及分类模型。另外，华为通过一站式开发平台 ModelArts 希望实现端、边、云全场景的 AI 部署。

总的来说，虽然端侧 AI 目前仍处于发展初期，但是相关应用已经展示出其巨大潜力，希望更多人能够了解和应用这项技术。这也是我们本次大会端侧 AI 专题的价值，通过介绍端侧 AI 最新技术进展和行业案例，给广大开发人员提供参考，从而使他们能结合自己业务场景进行探索和尝试，获得业务突破。

阿里端侧 AI 发展状况

端侧 AI 业务应用场景逐步增多并取得明确价值，开源推理引擎 MNN

最早从 2016 年开始公司内部就有业务尝试端侧 AI 能力，并在 2017 年开始有计划的建设和探索端侧 AI 方向，包括我自己和所在的 MNN 团队，也是从那个时候开始尝试端侧 AI，经过这几年的发展，端侧 AI 已经成为移动 App 的基础能力，助力业务发展和创新突破，并且也取得了不错的业务结果。

目前，公司里绝大多数移动 App 都有使用端侧 AI，2019 年初统计使用 MNN 的 App 就超过 20 个。以手淘为例分享几个数据：10 多个场景应用，超过 25 个模型运行，每天运行机器学习和深度学习算法次数超过 500 亿次。

应用场景

- **手淘信息流推荐**

基于端侧 AI 技术可以实现更加实时的用户意图识别，做出更加精准的内容推荐，甚至在端上训练实现“千人千模”，有效提升了原有个性化推荐技术，该技术已在 2019 年双 11 中大规模应用并取得不错业务结果。对信息流的点击量和 GMV 都带来了明显提升。

- **手淘美妆 AR**

基于端侧 AI 的人脸检测能力结合 AR 应用可以给消费者更加真实的购物体验。2019 年双 11 美妆行业推出的 AR 在线试妆功能，能够让用户模拟上妆，有效提升了线上购物体验。

除此之外，支付宝扫福、手淘拍立淘、智能 Push 以及闲鱼智能发布等都是端侧 AI 的典型应用。

技术建设

总体来说，因为我们起步早，体系也相对完整，技术结果显著。基本已经解决算法模型在端侧能不能跑以及跑得好、跑得快等问题，现在我们正在解决规模化应用和业务创新快速验证的问题。具体从如下几方面简单来说：

- **端侧推理引擎**

我们有开源推理引擎 MNN，以及支付宝定制化引擎 xNN，在性能优化、 异构多机型适配，模型压缩等方面有比较多积累，目前也已具备训练能力并支撑业务端上训练任务。

- **端 AI 算法能力**

公司各算法团队已经沉淀人脸检测、手势识别、人体骨骼、OCR 等常用能力，并且构建统一开箱即用算法集支撑业务快速接入应用。

- **端 AI工作台和研发体系**

淘系技术团队正在探索和建设一站式端侧 AI 工作平台"MNN工作台"，降低端侧 AI 应用门槛，实现业务快速创新试错。

端侧 AI 未来展望

端侧 AI 仍处于大规模应用爆发前夜，未来结合 AR、IoT 等技术迎来更大发展

端侧 AI 的应用会向更大范围、更多领域铺开并切实给业务带来价值。从大公司尝鲜到普通公司逐步推开，它会真正成为一项基础能力；从视觉算法到大数据、自然语言处理、语音等

更多算法领域的应用，例如：谷歌发布的 ALBERT 在移动端部署应用已经成为可能；从智能硬件到 IoT 设备，近来 TinyML 技术也在不断成熟，正在成为新的蓝海，具体可以参见 TinyML2020 有关内容。

端侧 AI 相比云侧 AI 一站式机器学习平台整体技术体系还不够完善，针对非专业算法同学如何能够自主解决数据和算法问题产出模型，针对大前端和算法同学如何提高协同效率等些问题的解决势必会进一步推动整个端侧 AI 的发展。淘系内部基于 MNN 正在构建端侧 AI 的完整研发体系，包括端 AI 运行时 +工作台 + 服务端，后续也会逐步向业界开放，一起让端侧 AI 获得更大发展。

5G 技术已经到来，我觉得 5G 技术的发展也会促进端侧 AI 的发展。举例来说，5G 来了，音视频数据会进一步膨胀，当网络不是瓶颈，服务端算力就会成为瓶颈，那么本地的视频内容理解、预处理就显得尤为重要，更实时、更低成本的方案肯定会被应用，业界有些公司已经开始行动。结合 AR 来说，端侧 AI 补充了 AR 的交互能力，5G 补充了 AR 网络传输能力，让高质量的 AR 素材传输变成可能。那么，结合 5G + 端侧 AI + AR 技术可能会出现一些有意思的应用。

总的来说，端侧 AI 经过几年发展，虽然在技术建设和业务应用都有了一些进展，但仍处于大规模应用爆发前夜。随着算力和算法不断进步，应用门槛的不断降低，结合 AR、视频、5G 以及 IoT 等技术，端侧 AI 必将迎来更大的发展。

作者简介

吕承飞，花名吕行，阿里巴巴资深无线开发专家。2013 年加入淘宝，经历手机淘宝超级 App 技术演化完整过程，主导手淘 iOS 架构升级、架构治理、稳定性以及性能等相关工作。2017 年开始端侧 AI 方向探索，现负责端智能团队，构建开源端侧推理引擎 MNN，淘宝 AR 技术框架和美妆 AR 等创新应用，端计算技术框架和搜索推荐等创新应用。

随着算力和算法不断进步，结合 AR、视频、5G 以及 IoT 等技术，端侧 AI 必将迎来更大的发展。欲了解更多端侧 AI 相关案例，请点击 GMTC 北京 2020 大会官网。

MNN 正式版发布：模型训练、异构性能和 Python 工具链

作者 | MNN 团队
出品 | 淘系技术

MNN 的诞生源于淘系技术部的一群对技术充满热情的同学，在充分的行业调研后认为当时的推理引擎如 TFLite 不足以满足手机淘宝这样一个亿级用户与日活的超级 App 。

于是我们从零开始自己搭建了属于阿里巴巴的推理引擎 MNN 。1 年前的这个时候，MNN 在 Github 上开源【1】。它比其他的推理引擎更快更轻量，更符合手机淘宝这样庞大、复杂的生产部署环境。今年 3 月份，MNN 的引擎设计与优化理念还获得了学术界的认可，在 MLSys 2020 【2】上发表了论文，并进行了 oral presentation 。

开源 1 年以来，获益于公司内外的用户反馈和业务推动，MNN 在许多方面都取得了长足的进步：

1. 在阿里巴巴集团内部得到广泛推广，成为了端上推理引擎的事实标准，覆盖了如手机淘宝、手机天猫、优酷、钉钉、闲鱼等 20 多个 App 。
2. 新添了模型训练的支持，从此 MNN 不再是单纯的推理引擎，而是具有推理+训练能力的深度学习引擎。基于 MNN 的训练能力，我们可以进行 Quantization-Aware Training(QAT)。在 MobileNet 上，MNN 量化训练之后的模型准确率几乎不降。
3. 持续投资于异构硬件后端的优化，尤其是利用 ARMv8.2 指令集，获得了两倍的性能提升。
4. 进一步完善 Python 工具链，累计新增超过 150 个接口。
5. 开源了应用层开箱即用的解决方案 MNNKit【3】，包含了人脸跟踪与检测、人像分割、手势识别场景的解决方案。
6. 开办了三期《 MNN 学院》直播（1 期【4】，2 期【5】，3 期【6】），增加了与用户们交流，也获得了忠粉们的大量高质量反馈意见。

截止到今天，MNN 在开源社区获得了近 4000 的 Github Stars，这是大家对我们的工作所投的 4000 张认可票，也是鞭策我们完善 MNN 的动力。近日，MNN 发布了 1.0.0 正式版本。自此，MNN 不再被 Github 贴上“Pre-release”的标签了！相较于 0.2.2 版本，1.0.0 版本的主要升级在于：模型训练、异构性能和 Python 工具链。下面，我们逐项说明。

模型训练

模型构建

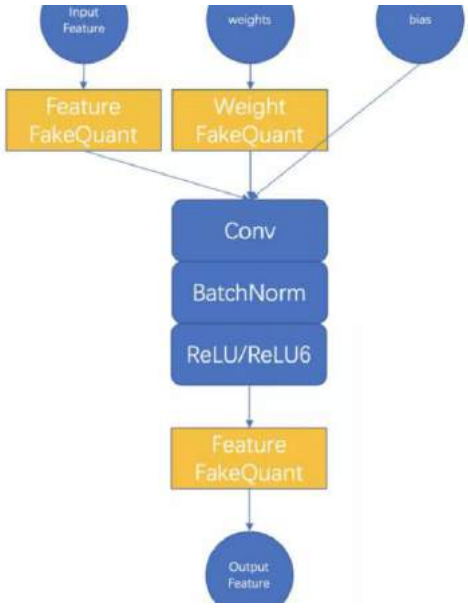
MNN 支持使用 Express （表达式）接口来构建模型，如下例所示，接口还是比较简洁明了的。模型的构建、训练和保存具体可以参考说明文档【7】。

```
VARP x = inputs[0];
x       = conv1->forward(x);
x       = _MaxPool(x, {2, 2}, {2, 2});
x       = conv2->forward(x);
x       = _MaxPool(x, {2, 2}, {2, 2});
x       = _Convert(x, NCHW);
x       = _Reshape(x, {0, -1});
x       = ip1->forward(x);
x       = _Relu(x);
x       = dropout->forward(x);
x       = ip2->forward(x);
x       = _Softmax(x, 1);
return {x};
```

以 MNIST 数据集 + Lenet 网络为例，一个 epoch 60000 张图片，一般可达到 97-98% 的准确率。性能上，同款 MBP 上，MNN 比 PyTorch 和 Caffe 都有明显优势；而手机上，MNN 也达到了完全可用的性能水准。

量化训练

模型量化既可以降低模型大小，又可以利用硬件特性提升推理性能，可谓业务应用必备之选。但美中不足之处在于，模型量化会带来一定的精度损失 —— 对于精度攸关的项目，就难免要做出艰难的选择了。



为此，MNN 借助自身模型训练能力，实现了模型训练量化，具体实现可以参考说明文档【8】。精度和压缩率方面，我们以 MobileNet V2 为例说明：

模型	类型	准确率	模型大小
原始模型	float32	72.324%	13M

MNN 训练量化模型	symm int8	72.456%	3.5M
TensorFlow 训练量化模型	symm int8	71.1%	3.5M

注 1：训练和验证均采用 ImageNet 数据集。训练采用 32 为 batchsize，执行 100 个迭代，即，使用了 3200 张图片进行训练；精度验证则使用了 50000 张图片。

注 2：原始模型为 TensorFlow 官方模型，官方准确率为 71.8%，但因预处理代码上有细微差别，我们测试原始模型的准确率结果稍高于官方； 可以看出，在实现了 73% 模型尺寸压缩的情况下，量化模型的精度甚至要稍高于原始模型。

迁移学习示例

这里节选 MobileNet V2 的 4 分类迁移学习示例，来说明模型的 Finetune，完整示例请参考文档【9】。

```
class MobilenetV2TransferModule :
public Module {public:
  MobilenetV2TransferModule(const char* fileName) {
  // 读取原始 MobilenetV2 模型
  auto varMap   = Variable::loadMap(fileName);
  // MobilenetV2 的输入节点
  auto input    = Variable::getInputAndOutput(varMap).first.begin()->second;
  // MobilenetV2 分类层之前的节点，AveragePooling 的输出
  auto lastVar = varMap["MobilenetV2/Logits/AvgPool"];

  // 初始化一个 4 分类的全连接层，MNN 中可以用卷积来表示全连接层
  NN::ConvOption option;
  option.channel = {1280, 4};
  mLastConv         = std::shared_ptr<Module>(NN::Conv(option));

  // 初始化内部特征提取器，内部提取器设成不需要训练
  mFix.reset(PipelineModule::extract({input}, {lastVar}, false));
  // 注意这里只注册了我们新初始化的 4 分类全连接层，那么训练时将只更新此 4 分类全连接层
  registerModel({mLastConv});
  }
  virtual std::vector<VARP> onForward(const std::vector<VARP>& inputs) override {
```

```
        // 输入一张图片，获得 MobilenetV2 特征提取器的输出
        auto pool     = mFix->forward(inputs[0]);
        // 将上面提取的特征输入到新初始化的 4 分类层进行分类
        auto result = _Softmax(_Reshape(_Convert(mLastConv->forward(pool), NCHW), {0, -1}));
        return {result};
        }
        // MobilenetV2 特征提取器，从输入一直到最后一个 AveragePooling
        std::shared_ptr<Module> mFix;
        // 重新初始化的 4 分类全连接层
        std::shared_ptr<Module> mLastConv;
        };

        int main(int argc, const char* argv[]) {
         std::string trainImagesFolder = argv[2];
        std::string trainImagesTxt = argv[3];
        std::string testImagesFolder = argv[4];
        std::string testImagesTxt = argv[5];

         // 读取模型，并替换最后一层分类层
        std::shared_ptr<Module> model(new MobilenetV2TransferModule(argv[1])); // arg1:
/path/to/mobilenetV2Model
        // 进入训练环节
        MobilenetV2Utils::train(model, 4, 0, trainImagesFolder, trainImagesTxt, testImagesFolder,
testImagesTxt);
        return 0;
        }
```

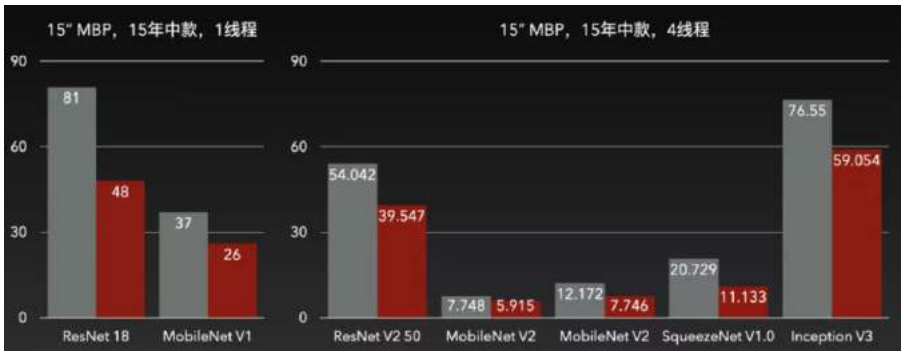
异构性能

x86

在 x86 上，我们重点优化了矩阵乘法。在分析过 AVX 和 Arm 向量乘指令差异后，我们修改了 AVX 下的权重矩阵布局，降低了 I/O 布局，以充分利用 CPU 算力。

此外，我们允许在支持 FMA 扩展的设备上，启用扩展，将乘法和加法合为一条指令，以进一步降低指令耗时。

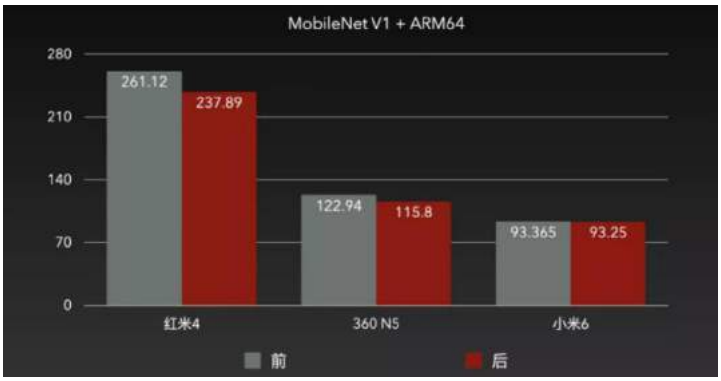
当前，FMA 扩展的启用开关还放置在 CMakeLists.txt 中，后续会在运行时判别。



综合两项优化，x86 上有 30% 左右的性能优化。

ARM64

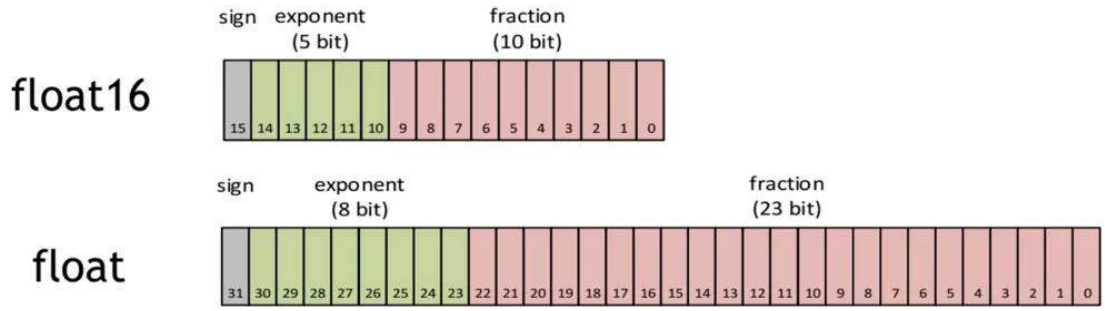
在 ARM64 上，我们面向中低端设备，调整了矩阵乘法的分块策略，矩阵中每个元素的均摊 I/O 降低了 22%；同时，将数据对齐从 32 字节调整为 64 字节，与 ARM 架构 CPU 下场景的 L1 cacheline 匹配；最后，优化了缓存预取。优化结果如下：



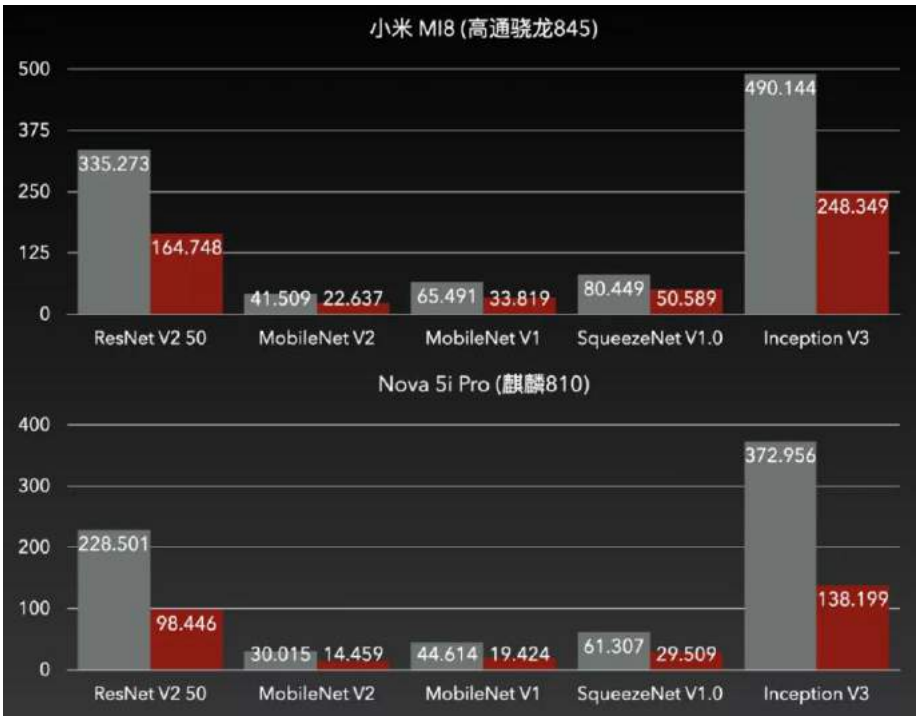
ARMv8.2

ARM 在「Bringing Armv8.2Instructions to Android Runtime」【10】一文中，列举了可以在 Android 运行时中应用的 ARMv8.2 新特性。其中，FP16 extensions 和 Dot Product 可以分别应用于浮点计算加速和量化计算加速。

FP16extensions

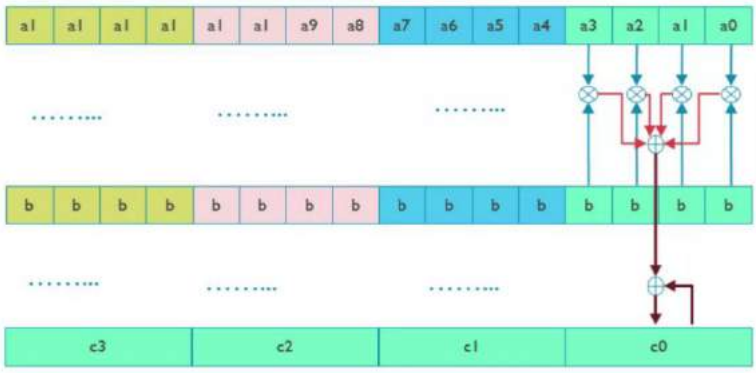


亦记作 asimdhp(Advanced SIMD Half Precision)，是 ARMv8.2 架构的可选扩展。asimdhp 可用时，可以使用相关 SIMD 指令实现 float16 的读写计算。float16 将 float32 所需的位数降低了一半，因此在 SIMD 下，可实现两倍的并发吞吐，从而优化性能。为此，我们在卷积中，采用[N,C/8,H,W,8]的数据布局，新增了部分卷积实现，效果如下：

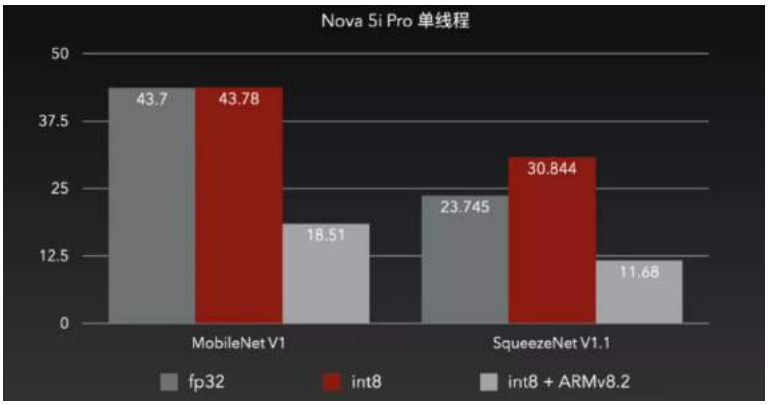


精度上几乎没有下降，但是性能足足提升了一倍。搭配上 MNN 转换工具的--fp16 输出选项，模型大小还能减小一半。一箭双雕。

Dot Product



亦记作 asimddp(Advanced SIMD Dot Product)，是 ARMv8.2 架构的可选扩展。asimddp 可用时，可以使用 SDOT/UDOT 指令实现 int8/uint8 的点积计算。SDOT/UDOT 指令如上图所示，一次可以处理两个 4x4 int8/uint8 数据乘，并累加到 4x1 的 int32/uint32 的寄存器上。这样强大的硬件加速指令，还是双发射的。



实战表现效果也非常明显，在原先 int8 无法发挥效用的设备上，ARMv8.2 也成功实现了耗时减半。

Python 工具链

2019 年的绿盟开发者大会上，我们发布了 MNN 的 Python 前端和 Python 版的转换、量化、可视化工具。而今，Python 又增加了对 MNN Express (表达式)、模型训练的封装，累计新增超过 150 个接口。具体可以参考说明文档【11】。

依然是前文的 Express 构图，使用 Python 改写的版本如下：

```
class Net(nn.Module):
    """construct a lenet 5 model"""
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.conv(1, 20, [5, 5])
        self.conv2 = nn.conv(20, 50, [5, 5])
        self.fc1 = nn.linear(800, 500)
        self.fc2 = nn.linear(500, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.max_pool(x, [2, 2], [2, 2])
        x = F.relu(self.conv2(x))
        x = F.max_pool(x, [2, 2], [2, 2])
        x = F.convert(x, F.NCHW)
        x = F.reshape(x, [0, -1])
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        x = F.softmax(x, 1)
        return x
```

对熟悉 Python 的开发者来说，是不是要亲切上许多呢?

注：目前 Python ExpressAPI 处于 BETA 阶段。我们会根据社区和内部的反馈持续改进 Python API ，包含进行 backward incompatible 的改动。

后续计划

2020 年，我们计划每个季度发布一个稳定版本。

未来的计划，主要集中在性能、训练、NPU 支持和模型压缩。 性能性能是 MNN 的立身之本，相信很多朋友选择 MNN，也主要出于它飙车般的性能。有兴趣的朋友，可以去看看 MNN 发表在今年 MLSys 的论文解读【12】。

- CPU 上，移动设备方面，ARMv8.2 将是新手机的主流，上文所展示 2 倍加速比非常诱人，我们会进一步挖掘 ARMv8.2 的优化空间；其他平台方面， x86 的性能在单机训练、服务端推理的场景中举足轻重，会是性能优化的另一个目标。
- GPU 上，我们会聚焦 Vulkan—— Android 下一代 GPGPU API 的事实标准。

训练

MNN 最新拥有的训练能力已经通过 Express (表达式)接口支持常用模型的训练、量化、蒸馏，我们会进一步完善训练能力，添加更多算子和求导的支持，以支持更多的模型。

NPU 支持

NPU 具有超高的性能、超低的能耗，将是未来手机的标配。NPU 的支持，也是许多 MNN 用户经常在钉钉群里提出的需求。MNN 在未来的 1 年，会逐步支持更多的 NPU，请大家拭目以待！

模型压缩

MNN 目前已经拥有 Post-training quantization 和 Quantization-aware training 的能力。我们会持续投入模型压缩（如蒸馏，稀疏，剪枝，低比特等），给业界提供更多优秀的、即插即用模型压缩算法。

链接：

https://github.com/alibaba/MNN

https://proceedings.mlsys.org/static/paper_files/mlsys/2020/7-Paper.pdf

https://github.com/alibaba/MNNKit

http://mudu.tv/watch/4308076

http://mudu.tv/watch/4397479

http://mudu.tv/watch/4844443

https://www.yuque.com/mnn/cn/kgd9hd

https://www.yuque.com/mnn/cn/bhz5eu

https://www.yuque.com/mnn/cn/vniw5p

https://community.arm.com/developer/tools-software/oss-platforms/b/android-blog/posts/bringing-armv8-2-instructions-to-android-runtime

https://www.yuque.com/mnn/cn/usage_in_python

https://mp.weixin.qq.com/s/ZmKC4fHTNPUj5iGqag19HA

端计算在移动设备上的应用探索

作者 | 李杰(兵长)
出品 | 淘系技术

本文知识点提炼：

1、端计算在移动设备上的应用探索

2、技术方案与核心模块设计

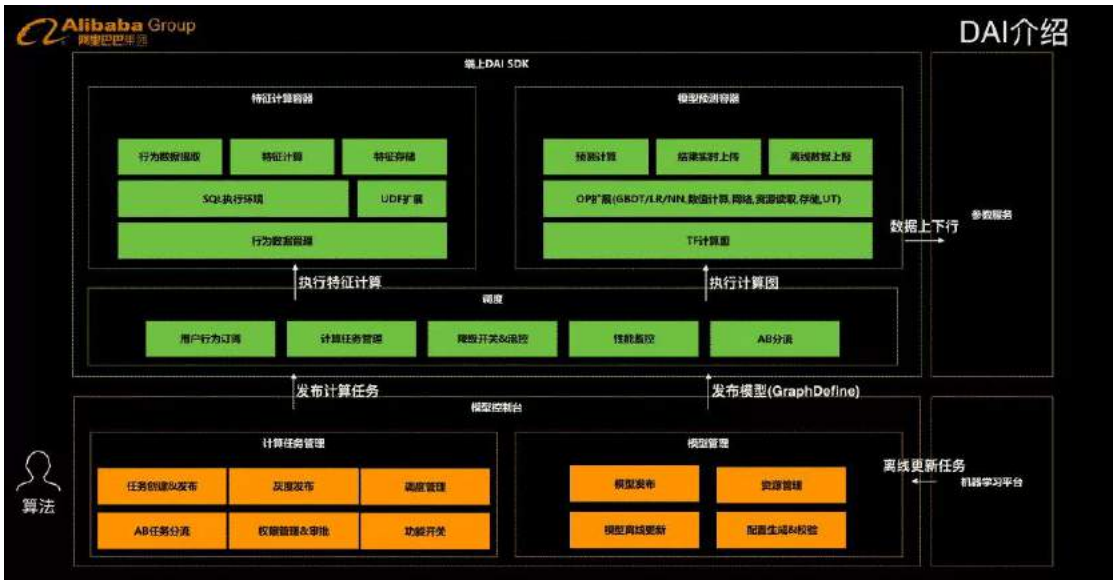
3、总结与展望

背景

传统的云计算，使用的是端侧采集数据，云端处理消费，再反馈给端侧的模式。而伴随着数字化转型的浪潮、万物互联时代的到来，5G、大数据、人工智能等信息技术的快速发展，云计算已经无法特定场景对低延迟的高要求。此时基于路由器、交换机、基站等计算节点的边缘计算因运而生，其具有低延时、低成本、数据安全、数据丰富等特点。

而借鉴边缘计算的思想，并融合手淘电商的业务场景，我们在 18 年初提出并搭建了基于移动设备的第一代 端计算工程系统 DAI 。其基于 TensorFlow 的模型推理能力，将计算、决策前置到移动终端，获取最原始的数据，实时在端侧进行数据分析与决策，端到端的响应耗时可以做到百毫秒级别。同时将过滤后的数据传输上云，与云端形成协同效应，并减少服务端的带宽、运算成本。

18 年双十一期间，端计算在部分场景小范围尝试和落地，并在主会场的猜你喜欢，详情页的看了又看等业务上取得了不错的效果。



端计算 DAI 架构图

面临的挑战

今年我们加大投入，并联合了算法团队、搜索推荐工程团队、手淘基础链路团队，共建端计算的工程体系。随着端计算体系承载的业务数量与复杂度的快速增加，也对 DAI 等基础设施提出来了更多更严峻的挑战。

研发效率

初期的设计是算法同学通过控制台下发 TensorFlow 的 pb(protobuffer) 模型文件，所有的逻辑均在 pb 的网络结构中实现。这种模式下，存在如下一些不足的地方。

- 由于端侧集成的为精简版 TF Mobile ，算法同学编写的 TF 代码在端侧可能存在缺少算子而跑失败的情况。
- 新增或修改 Op 需要 Native 发版实现，周期长。
- if、for 等流程控制在 TF 中难以处理。
- TF 的端侧推理耗时较长，业务决策响应不及时。

稳定性

Android 出于包大小和动态性的考虑，采用了动态下发并加载动态库的模式。但是由于 Android 设备的碎片化，动态加载存在着诸多兼容性的问题，测试也不好验证。同时 JavaScriptCore 本身在 iOS 上是个黑盒，曾在 iOS9 上就出现过大量的 JavaScriptCore 的 Crash 问题。而端计算作为算法处理的基础设施，每日被调用的次数非常庞大。所以任何一个极小的不稳定因素，都有可能被放大。

并且端侧的故障，大部分是由于线上配置发布引起的。手淘对于线上变更有着严格的安全生产流程，涉及发布窗口、验证、灰度、观察等各个环节。而算法同学往往对端侧的指标不熟悉，一些潜在风险未必能及时发现。我们需要在各个环节加强完善设施能力，在风险发生前及时暴露，在发生中将影响减至最低。

任务治理

在年初的时候，我们进行了一次线上业务梳理。发现手淘环境中有 5+的特征提取任务、4+的曝光任务。很多基础的数据特征，在不同的业务场景下都需要使用到，且对于同一特征的加工方式往往相识。若所有的特征均由各业务方自行进行加工，难免会造成开发成本及端上计算成本的浪费。而且无法高效地将已有能力复用到更多业务和 App 上。

场景覆盖

在端计算模式快速发展中，我们关注到部分业务域虽然不具备算法资源，但是希望借鉴端计算的思路，在一些输入因素相对比较固定的场景下，对用户特定的行为进行快速的响应与干预。同时初期 DAI 的触达能力比较单一，仅将执行结果以广播的方式通知到业务方，由业务方自行实现通知后的触达响应逻辑。而一些常规的触达途径，在大部分业务域都是相识的。比如 Push、Poplayer（浮窗）、触发其他模型任务联动等。在这个环节需要有一套统一的多样的触达机制，满足不同场景不同定制。

端计算 2.0 Walle

基于上述问题，我们对 DAI 进行了全面的升级，并改名为 Walle 。希望如电影 Walle 一样，将被遗忘在端上的数据汇集起来，成为挖掘金矿的工厂。

架构设计

Walle 架构图

整体设计上，Walle 由端、云两部分组成。

端侧包括采集层、计算层、触达层三个模块。采集层对接了端侧不同的数据源，进行数据存储与特征加工。计算层内置了 MNN、AliML 等。所有模型任务经过调度系统后会在计算容器中进行实时的决策。决策结果经由触达层的多种途径触达用户。

云侧分为运维平台、数据服务、触达服务三部分。运维平台负责日常的运维监控、数据服务为云端协同提供通道支撑、触达服务配合端上的触达层，进行人群圈定和事件分发。

解决思路

更高效易用的计算容器

为了解决算法模型的迭代部署效率，我们需要一套脚本语言环境来承载复杂控制与业务定制化逻辑。基于新版的计算容器，大幅降低算法同学的认知与学习成本，无缝衔接服务端算法应用流程，极大提升部署与迭代效率。

同时我们使用自研的轻量级深度学习引擎 MNN 替代 TF ，扩展了机器学习计算集 AliML，集成了高性能时序数据库 ProtoDB ，为用户提供了一套低成本、高效、快速迭代的端侧模型预测与训练的执行环境。

计算容器

更夯实的稳定性保障措施

作为逐步大规模应用的端计算基础设施，稳定性可谓重中之重。我们对端计算的开发、发布、运行时、监控、降级等全链路进行了详细梳理，针对一些有风险的环节进行重点保障。

- 开发测试阶段
  - 代码覆盖率。在 SDK 中内置了代码覆盖率与性能热点的采集与上报功能。配合 Jarvis 平台的真机验证系统，可以在发布阶段更全面地度量真机验证的效果，将风险暴露在上线前。
  - Mock 系统。为了实现自动化测试能力，我们开发了 Mock 系统。支持基于基线数据，对端计算任务的入参、出参、异常逻辑进行 Mock 验证。

- 运行时
  - 单机熔断。由于算法模型的迭代频率较高，为了避免在日常的迭代过程中引入新问题，我们在端侧引入了单机熔断的机制。既某个模型任务的执行耗时超过阈值或者执行线程卡死时，我们会重启执行线程，并对当前的模型任务进行一定时间段的熔断处理，以免影响其他模型任务的正常执行。
  - 高危模块移除。移除了 So 动态加载、JS 等存在潜在风险的模块，使用更优雅的方案替代。
  - 疑难问题解决。端计算演进期间，我们攻克了诸多内存 Abort 、Crash 、多线程锁等疑难问题，整体 Crash 率有大幅降低。

- 监控
  - 调试工具。支持验证版本的生效配置，扫码拉取端侧日志，对任务异常进行实时调试排查。
  - 监控大盘。面向 App 运维同学，可全局地观察整个端计算整体关键指标，以及每个任务的资源消耗排名，异常情况排名等。
  - 任务报表。面向算法同学，包含全链路多维度监控，长尾报表等，可直观地实时反馈任务上线后的运行情况。
  - Crash 定向监控。为了更准确实时地定位线上 Crash 问题，我们与 Emas 团队合作开发了模型 Crash 定向监控能力。在 Emas 平台上，现在可以直观地看到所有模型任务的 Crash 分布情况，以及 Crash 调用栈明细。

更体系化的数据能力

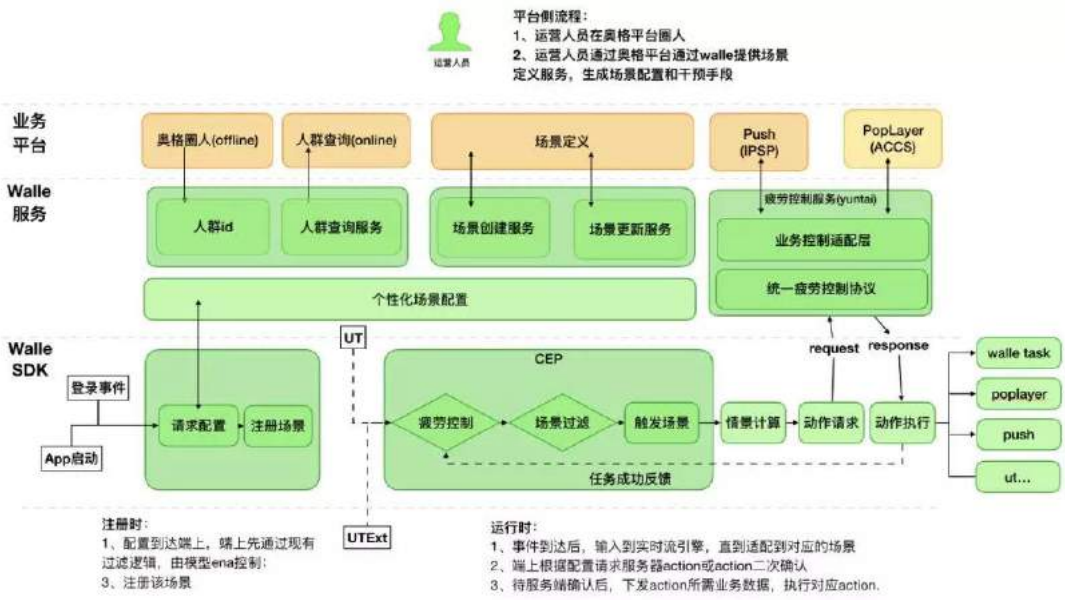
从共享端侧特征、降低重复计算与使用成本、提升特征查询效率等方面考虑，我们建立了端侧基础特征服务 DBFS。基于基础特征分层抽象出了统计特征、用户画像、情景计算等高维业务特征，同时支持算法同学进行特征 op 的自定义扩展。DBFS 目前提供 100+个基础特征 op，10+中间层业务特征 op，涵盖电商场景最常用的点击、收藏、加购、下单等行为。

DBFS 架构图

更丰富的场景覆盖能力

为了满足不同场景对端计算能力的诉求，我们建设了端计算的触达中心，其包含两部分能力：

- 在端侧实现了一套简化的 CEP（复杂事件处理）引擎，支持根据预置的规则序列，使用滑动窗口的模式匹配用户的操作行为，进行实时的用户干预。对于一些轻量化场景，可以直接使用 CEP 来定义行为，而无需引入机器学习模型。
- 同时在触达层面，对接了奥格的人群系统，支持针对特定人群进行 CEP 规则或者模型任务的投放。在行为命中后的业务响应环节，我们统一扩展了 Push、Poplayer、Broadcast、WalleTask、UT、NativeCallback 等多种途径触达用户。



触达中心

总结

今年双十一期间，端计算首次在手淘大规模落地，覆盖主搜、信息流推荐、云主题、会场、智能 Push、红包雨、促升、直播等场景。双十一当日共执行了 2235 亿次运算，在大幅提升 GMV 的同时，也为用户带来了更好的交互体验。同时除了手淘外，目前端计算也已在猫客、闲鱼、AE、CBU、零售通、优酷等 App 有成熟方案落地。

端计算的出现，填补了云计算在网络延时、数据丰富、隐私安全、算力成本方面的不足。而端计算与云计算也会以共存与互补的姿态，拥抱智能化浪潮。随着端计算体系的成熟以及基础设施的完善，相信未来算法同学们会有越来越多的创新项目孵化与应用，我们共同期待。

端智能与前端 AI 探索

作者 | 离青
出品 | 淘系技术

疫情时期的前端 AI

AI 在大前端领域已经火爆了两三年了。随着设备算能的跃升和端侧模型的演化，越来越多的 AI 场景开始涌现。从最初的图片分类，到而今基于人体的百变特效、嵌入日常的语音识别、大众津津乐道的自动驾驶，AI 已经开始进入我们生活的方方面面。

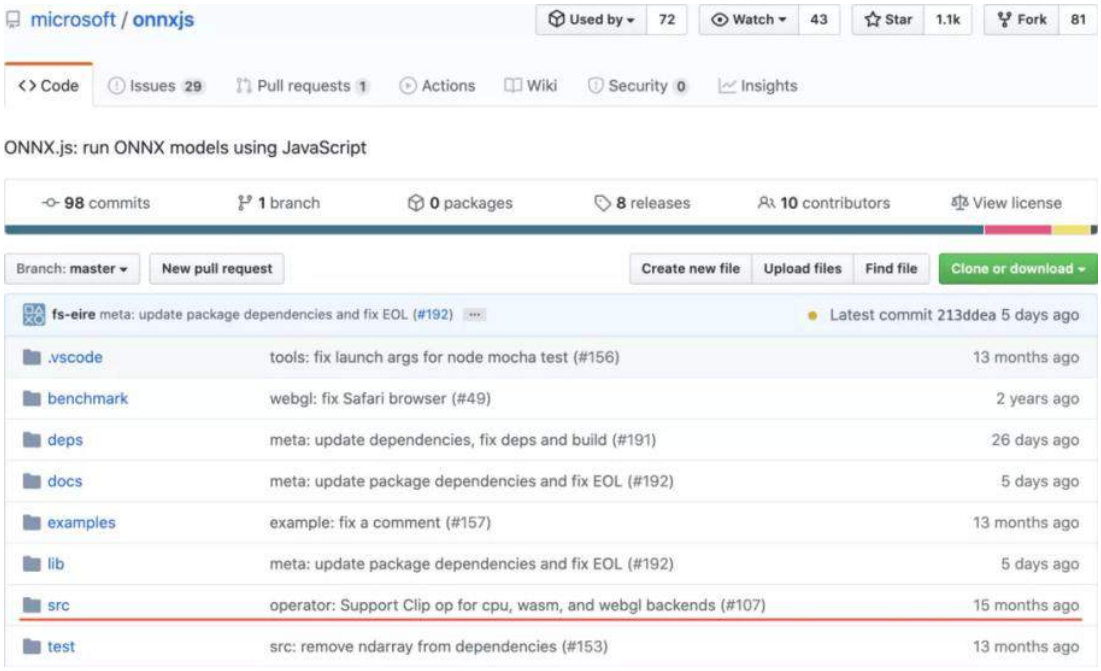
疫情期间，AI 更是频频进入我们的视野 —— 从病毒宿主研究，到病例追踪防控。而我们前端，也不甘寂寞，给人脸照片戴上口罩的 WebApp，就在知乎和朋友圈火了一把。网友 Random Forest 基于 faceapi 实现了人脸的检测，而后，在口罩素材库里选择口罩混合渲染生成最终的图片。

类似这样的趣味 WebApp，还有许多，前端从来不缺点子。那为什么前端的智能化，没有隔壁端侧智能看起来热闹呢？

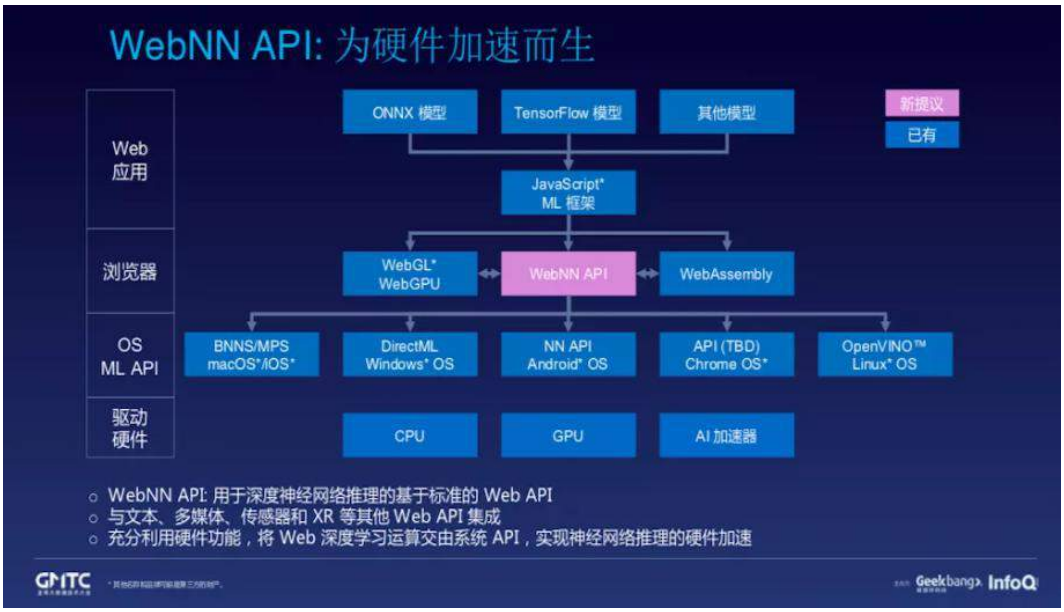
狂欢下的落寞

2017 年之后，为 Mobile App 提供底层加速能力的框架，如雨后春笋般涌现。BAT、小米、华为各自推出了端侧的深度学习框架，连旷世、商汤都要撸起袖子进场。但前端这就要冷清上许多了，除了 TensorFlow JS 之外，没有太多选择。

ONNX JS？除开最近的两次依赖升级，对框架主体的升级，要整整回推到一年多以前了。很难说是一个让人安心的选择。

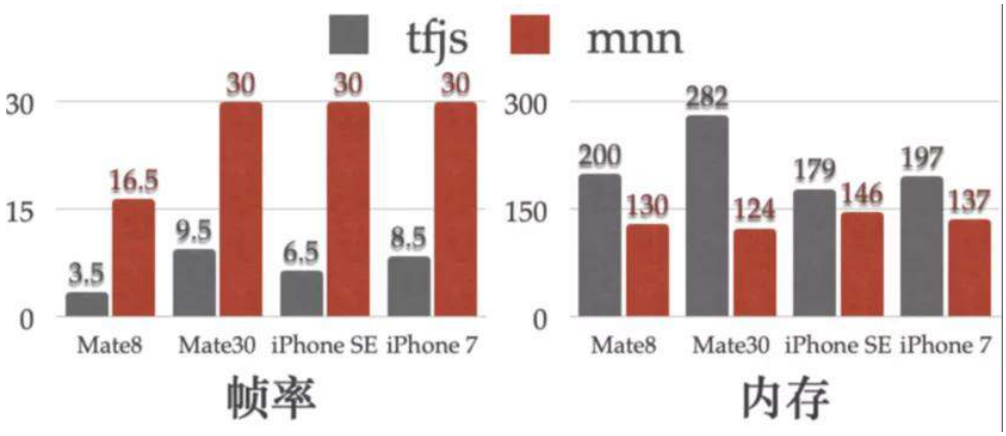


WebNN 的提交则稳定上许多。WebNN 会优先选择平台提供的加速方案，如 DirectML NNAPI；若无，则退而求其次，尝试 WebGL、WebGPU 或 WebAssembly。WebNN 如果可以成为浏览器们的标准实现，则可以以低成本实现计算加速；但相应的，依赖平台众多，也导致支持的特性往往只能选取共性子集。标准推进，前路漫漫。



那么，TensorFlow JS 是不是足够满足业务的需求呢？早些时间，笔者参与了淘宝的 AR 小程序项目，小程序环境类似于浏览器，可以使用 WebGL 和 WebAssembly 加速推理。但即使友军 WebGL 的性能相比于微信已经有不小的改进，也没能挽救 Mate 30、iPhone 7 这样中端机型明显卡顿的局面。最终，我们选择桥接到 Native 的 MNN 实现了加速，效果显著，基本全线设备都有 4 倍以上的帧率，也更节约内存。

小程序帧率 30 封顶，中端以上设备，Native MNN 的帧率就达到上限了。



MNN 再快，也需要有 Native 的支持。在浏览器和小程序上，有既跨平台，又能跑得流畅的解决方案么？

前端武器库

先盘点盘点前端武器库里，可以用于推理加速的装备们 —— 跨平台主要考量覆盖率，流畅度则主要考量性能。

JavaScript

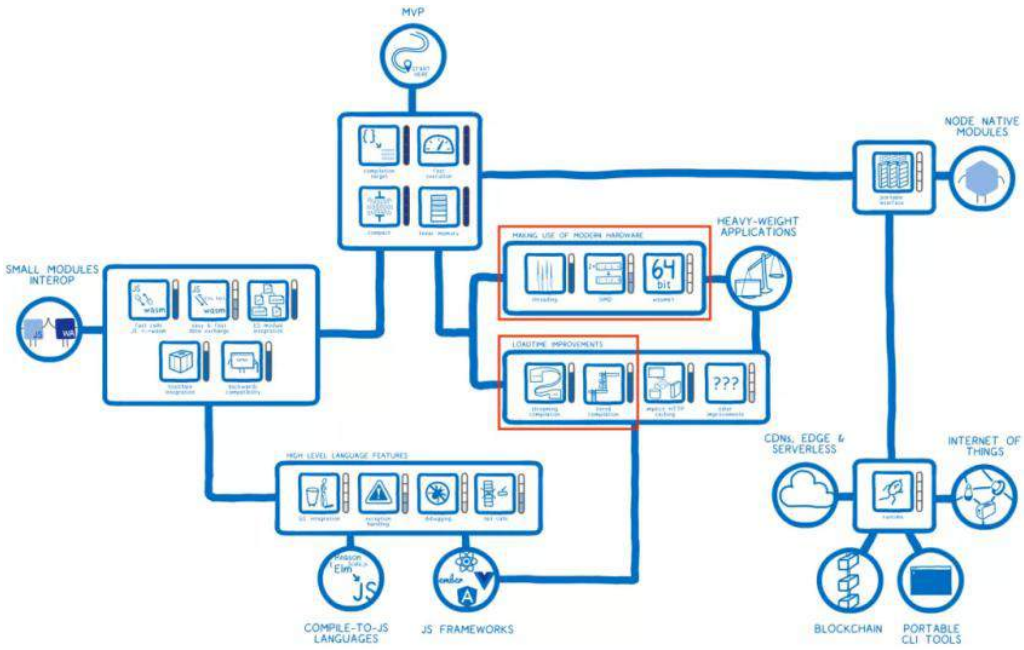
覆盖率 100%。可以通过 asmjs 这样的工具，将 C/C++的实现方案转到 JavaScript 上，是实现成本较低的方案。但性能就不要太指望了，同平台下，比 Native 方案慢上几十倍是再正常不过的事。

WebGL / WebGL2

覆盖率方面，WebGL 和 WebGL2 在浏览器上分别有 97%和 74%，Mobile 和 Desktop 的差距不大，小程序的覆盖率应当也相近。但再考虑到推理加速所必须的 color_buffer_float 拓展，实际情况并没有没有这么乐观。相当比例的中低端 Android 设备 WebGL 支持不佳。因此，仅支持 WebGL 的话，就需要放弃相当一部分用户。

性能方面，WebGL 和 WebGL2 是 tfjs 和 onnxjs 的性能担当，就现阶段而言，比 JavaScript 和 WebAssembly 都要快上许多。

WebAssembly



覆盖率方面，WASM 在浏览器上为 91%，但微信、淘宝、支付宝小程序则都支持 WASM。

性能方面，参考 mozilla 的规划介绍，由于当前缺少多线程、SIMD、64 位等的支持，WASM 的性能还很难与 Native 看齐。Android V8 引擎下，可以通过 JIT 找回一些场子；但 iOS 下，就回天乏术了。

WebGPU

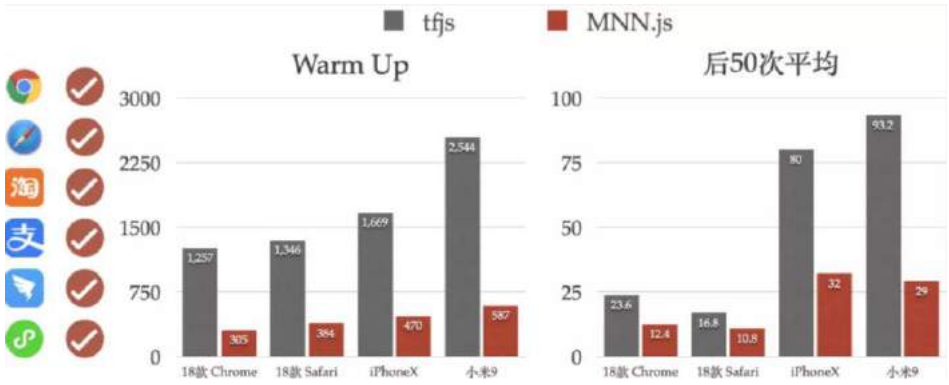
覆盖率一片红...性能再好也指望不上哎。

小结一下，结合 WebGL 和 WebAssembly，可以实现对大部分浏览器和主流小程序的覆盖，也可以实现相对较好的性能。

磨一把更快的刀

我们结合了我们在小程序上的优化经验和 MNN GPU 优化经验，基于 MNN 模型基础设施，实现了全新的 MNN.js。性能上有相当明显的提升，视频场景下的帧率抖动基本消失，内存上也有一定的节省。

效果



测试版本：tfjs@1.7.0 + mnn.js@0.0.3。

测试方法：Warm Up 1 次，后续推理 50 次。MBP 直接测试，iOS/Android 重启测试（避免 Shader 缓存）。

测试环境：

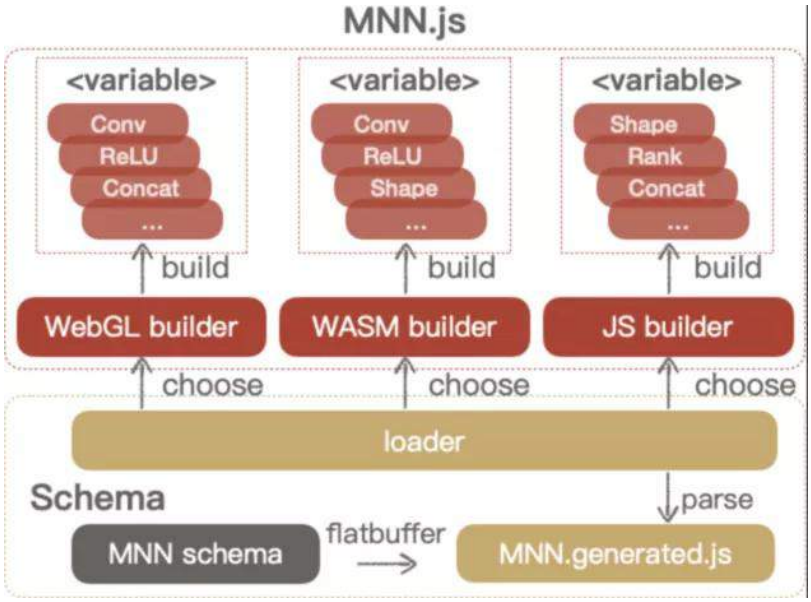
- Desktop：MBP 18 年中款，GPU = Radeon Pro 555X + Intel UHD Graphics 630

- iOS：iPhone X
- Android：小米 9

上数据。性能部分，首次推理上，相较于 tfjs 节约约 70%的耗时；后 50 次平均上，则节约了 35%以上的耗时。兼容性部分，浏览器上，我们测试了 Chrome、Safari 和众多手机的自带浏览器；小程序上，我们测试了淘宝、支付宝、钉钉、微信小程序。具体测试的数据就不再陈列，WebGL 性能表现上与上述基本相近。

MNN.js 当前支持了 38 个算子，除了 MobileNet 这样的简单模型，在多达数百个算子的内部模型上，我们也有近似的性能领先。

框架



在设计上，我们复用了 MNN 的模型格式。一来，可以实现对 Caffe、TensorFlow、PyTorch 模型的转换支持；二来，MNN 的算子替换、层融合等模型优化，也可以延续到 MNN.js 上。

模型加载后，有 WebGL、WebAssembly、JavaScript 三种计算后端可供选择。JavaScript 一般不单独使用，WebAssembly 一般在 WebGL 不支持的情况下使用。

示例

```
async function loadModel() {
  let compiled = await MNN.WasmBuilder.compile(fetch('/path/to/mnn.wasm'));
  let wasm = await MNN.WasmBuilder.create(compiled);
  let js = MNN.JSBuilder.create();

  let builders = [webgl, wasm, js];
  let result = await fetch(url)
  let model = await result.arrayBuffer();
  return MNN.loader.loadMNN(new Uint8Array(model), builders);
  }

  async function run(webcam) {
  let width = webcam.videoWidth;
  let height = webcam.videoHeight;
  let preprocess = {
  sourceFormat: 'rgba',
  destFormat: 'bgr',
  filterType: 'nearest',
  wrapType: 'zero',
  cropAndResize: [[0, 0, width, height], [0, 0, 224, 224]],
  means: [103.94, 116.78, 123.68],
  norms: [1 / 127.5]
  };

  let rgba = capture(webcam);
  await model['data'].setData(rgba, "uint8", [1, height, width, 4], "NHWC", preprocess);
  let result = await model['prob'].getData("NHWC");
  let values = new Float32Array(result.data);
  let sorted = values.slice().sort((a, b) => b - a);
  }
```

接口上，我们延续了 MNN 小程序插件的动态图设计，并提供了图像前处理能力，即使是在视频场景争抢资源的情况下，也不至让性能过分下滑。

前端 AI 去哪儿

深度学习三要素：数据、算法、算能。数据在不断积累，模型在不断翻新，算能也跟上了，AI 应用是不是就水到渠成了？并不是。算法是 AI 应用的筋骨，场景才是 AI 应用的灵魂。谁最熟悉业务困境？谁最了解用户痛点？谁能跨出那一步，去了解算法的边界，点亮创意，连接算法和业务，为 AI 应用填充血肉？

而今，随着小程序基础设施的建设和跨端互通互投战略的推进，App 不再是束缚内容的孤岛，每一个业务都可以小程序化，从而有更多机会触达更多的用户。

淘宝购物小程序作为淘宝开放最重要的阵地，承担着提升购物乐趣、让商家自由经营的重要使命。小程序平台通过 AI/AR 新技术主推淘内导购体验升级。基于人脸、姿态、手势等识别能力，可以创造出更丰富、更新奇的互动玩法，让买家边玩边买。

目前，购物小程序已经打通了店铺首页、商品详情、客服通道、直播、搜索等众多公私域渠道，并在不断加码。小程序插件体系使得外部优秀的算法供应商/引擎服务商可以快捷入淘，为淘内广大的商家群体和服务生态提供技术服务，在给商家创造增量价值的同时，也可以获得相应的商业收益。

详情请参阅招募通道：open.m.taobao.com。

舞台已备。天地广阔。踏南天，碎凌霄，当是吾辈中人。

花絮

在阿里内部的技术交流贴里，~~两个商业互吹~~两名技术小二对上了两段后浪：

那些口口声声 前端 AI 没有机会的人
应该看着你们
就像我一样
我看着你们 满怀羡慕
设备积攒了十几年的算能
所有的分类、识别、分割和推荐
像是专门为你们准备的礼物
数据海量、算法繁荣、引擎强大
深度学习的门槛 被逐步降低
可以尽情地应用
自由学习一类模型 挑选场景
优化一处体验 在无尽的可能中探索

那些抱怨"前端不如原生"和"原生不如前端"的人
应该看着你们
就像我一样
我看着你们 满怀敬意
向你们的专业态度致敬
你们正在把客户端的 变成前端的
把前端的 变成客户端的
把性能差的 变成性能好的
码工与码工之间的壁垒被打破
你们只凭相同的爱好 就能结交千万个商业互吹的朋友
你们拥有了 我们曾经梦寐以求的权利
跨栈写码的权利

我说前端会玩吧。

技术交流

MNN.js 距离开源，还需要一定的打磨。欢迎有志于前端 AI 的你，加入 MNN 钉钉群，与我们交流。

链接：

https://zhuanlan.zhihu.com/p/104070435
https://github.com/tensorflow/tfjs
https://github.com/microsoft/onnxjs
https://webmachinelearning.github.io/webnn/
https://github.com/alibaba/mnn
http://asmjs.org/

聊一聊 MNN 在端智能生态中的应用

作者 | 陈以鎏（离青）
出品 | 淘系技术

前言

2019 年 5 月，淘系技术开源了深度学习推理引擎 MNN，加入到人工智能开源社区中来。自打开源起，MNN 就一直是开源社区的性能标杆之一，是众多后来者的挑战的目标。这固然是来自行业的认可，却也拉开了推理引擎间的性能军备竞赛。这或多或少，背离了我们的初心 —— 人工智能只是工具，靠比拼工具在一时一地的优劣，不能帮助我们繁荣 AI 应用生态。

帮助 MNN 不断成长的，除了我们在性能上的不断压榨，还有众多场景的哺育。MNN 支撑着阿里巴巴众多的人工智能应用，从电商购物到视频直播，从手机应用到智能汽车；同时，也有越来越多的企业选择 MNN，通过人工智能来提升工作能效、优化用户体验。是这些应用，而不是 PPT 上的性能数据，在你我不经意之间，改变了我们的生活。

为了走出性能内卷怪圈，为了给生态引一泓清流，MNN 会陆续为你介绍我们在人工智能应用上的探索。这里的「我们」，不只是阿里巴巴，也包括身为 MNN 用户的你。

优酷体育：宅家街舞 AI 私教？！

今年的疫情对许多行业都产生了深远的影响，这其中就包括运动健身。宅家胡吃海塞，不能去健身房挥汗如雨，沉积的脂肪都堆在了大腿和肚腩上。俗话说，三月不减肥，四月徒伤悲，五月不减肥，六月徒伤悲...就算宅家，也不能被封印在床头！生命！在于运动！

然后你看了看家里的跑步机。

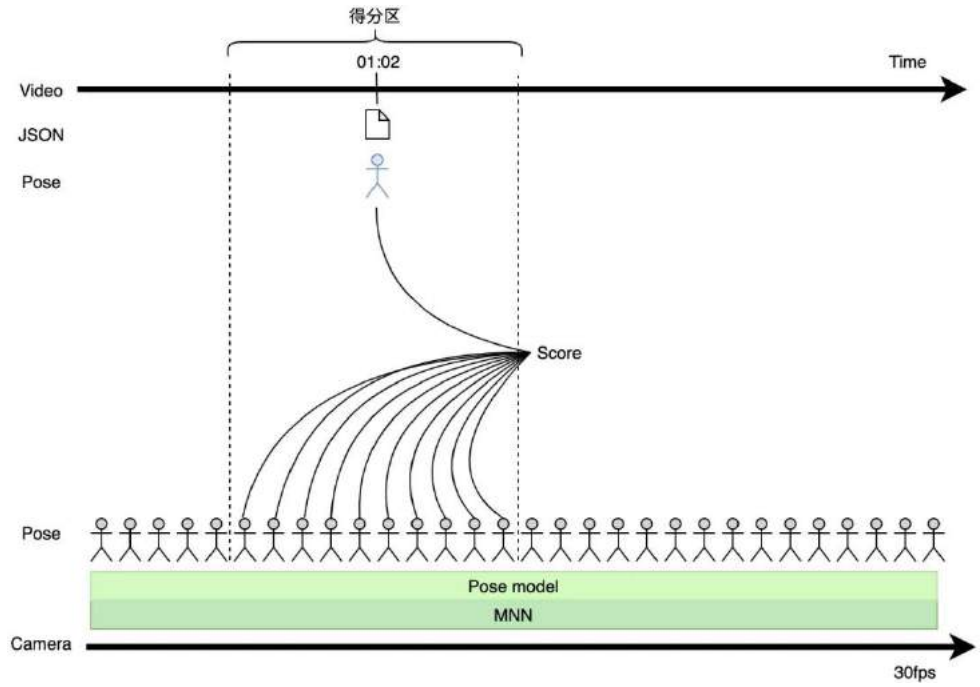
是这样?

点击查看动图演示

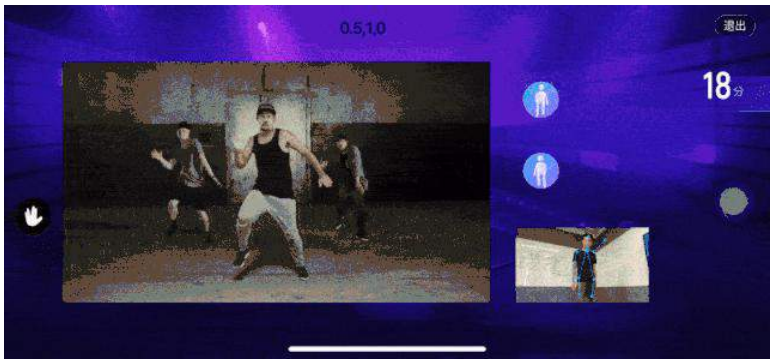
或者是这样?

除了器械，家庭健身需要专业的运动指导，更需要优质的内容，让运动本身不再乏味和无趣。优酷体育 AI 操房，作为中国首家家庭智能操房平台，连接了硬件产商、健身机构和运动达人，通过游戏互动，带你玩转运动 —— AI 操房通过摄像头捕捉用户动作，通过算法实现动作量化分析，再结合上游戏化的方式，把反馈投屏到电视上，让用户的跳操不再枯燥乏味。

为了指导用户做出动作并打分，我们要先根据原始视频设定动作示意和打分标准。运营同学会在 AI 操房的后台，从原始视频中选出关键帧，根据视频骨骼点信息生成动作示意图和描述文件。



在进入操练前，动作描述文件会和视频一同下载下来。在视频播放的同时，摄像头会捕捉用户的动作。关键帧的前后一段时间都会作为对应动作的得分区间。我们会在得分区间内持续比较用户和视频的动作差异，给出 miss、good、perfect 等反馈，并打出分数。区间内的最高得分会保留为动作的最终得分，参与总得分的计算。



点击查看动图演示

这样，就算宅在家里，也能练起街舞了。谁说下一届「这！就是街舞」的冠军，就不能是你呢?

整套方案基于平台化思维，分为前端用户流程和动作编辑后台，整套流程完全开放，可以提供给健身机构或者 KOL，产出定制化的 AI 操房，创造更多的玩法。整套方案还可以无缝迁移到 OTT，为 OTT 用户提供服务。

有意向与优酷体育进行合作的智能硬件产商、科技和 AI 算法平台、健身机构和健身内容创作者们，请发邮件至：zr162261@alibaba-inc.com ，与优酷体育共创 AI 健身操房，做大！做强！

陌陌：直播互动「心」姿势

陌陌在人脸识别、人脸关键点、表情识别、手势识别、身体关键点算法上，有长足的积累。在直播中，陌陌基于人脸识别、人脸关键点，加上渲染、磨皮、美白等技术，为主播实现美颜、眼妆、贴纸等一系列特效；在拍摄器里，陌陌基于人脸关键点，来识别表情和睁闭眼，实现了一系列的特效玩法，比如眨眼识别的眨眼星星特效、嘟嘴识别的吹蒲公英、吹泡泡等。

最初，陌陌并没有使用 MNN 推理引擎，由于客户端上人脸检测、人脸关键点等模型推理和特效渲染都需要占用大量 CPU 和内存，CPU 和内存成为了更多模型和特效应用的瓶颈。在全面升级到 MNN 推理引擎之后，推理速度和内存占用上都有了明显的优化。这样，客户端上就可以放心添加更多功能模块了。

为了增加主播和粉丝的互动，让直播更加有趣，陌陌在直播中推出了送礼物比心的互动玩法 —— 当粉丝给主播送出礼物后，主播可以做出比心手势表示对粉丝的感谢，在客户端识别出手势后，会触发粉丝送礼物的特效。

技术小哥哥素颜出镜比心

功能上线后，使用过的主播都夸效果好，直播间的粉丝数有明显的提升，主播和粉丝之间的互动更多了，粉丝刷礼物也更积极。

后续，陌陌还会在直播间和拍摄器里设计出更多好玩的特效和互动，期待你的体验~

智能应用长征

人工智能可以优化原有流程的体验，也可以开创出全新的玩法，但目前，智能应用的开发流程还很长，整体门槛比较高。

要打造一款有用、有趣的人工智能应用，你需要对人工智能和业务场景都有必要的认知，才能将有效融合两者。同时，你需要为模型训练收集大量的数据，并对数据做出必要的清洗和标注。之后，在众多模型结构中找到适合业务的，经历漫长的训练、验证迭代，得到模型后还需要做必要的优化、压缩。

产出模型并不是端侧 AI 应用的终点，恰恰是起点。以视觉类应用为例，可能 90%的工作量都在模型训练之后 —— 适配 iOS、Android 的相机输入，图片增强特征、裁切、转换，推理结果过滤、提炼，结合物料渲染上屏。工程链路很长，涉及到的移动端编程、图片处理、渲染绘制，还分别要求不同的专业技能。

端智能精彩问答

Q: 如何评价阿里开源的移动端深度学习推理框架 mnn?

作者 | MNN 团队
出品 | 淘系技术

如何评价 Alibaba 出的 mnn 轻量级的深度神经网络推理引擎?

不请自来，正主儿坐镇。：）

说说我们的想法，欢迎讨论交流。

MNN 的诞生源于淘系技术部的一群对技术充满热情的同学，在充分的行业调研后认为当时的推理引擎如 TFLite 不足以满足手机淘宝这样一个亿级用户与日活的超级 App 。

于是我们从零开始自己搭建了属于阿里巴巴的推理引擎 MNN 。1 年前的这个时候，MNN 在 Github 上开源。它比其他的推理引擎更快更轻量，更符合手机淘宝这样庞大、复杂的生产部署环境。

今年 3 月份，MNN 的引擎设计与优化理念还获得了学术界的认可，在 MLSys 2020 上发表了论文，并进行了 oral presentation。

开源 1 年以来，获益于公司内外的用户反馈和业务推动，MNN 在许多方面都取得了长足的进步：

- 在阿里巴巴集团内部得到广泛推广，成为了端上推理引擎的事实标准，覆盖了如手机淘宝、手机天猫、优酷、钉钉、闲鱼等 20 多个 App 。
- 新添了模型训练的支持，从此 MNN 不再是单纯的推理引擎，而是具有推理+训练能力的深度学习引擎。基于 MNN 的训练能力，我们可以进行 Quantization-Aware Training (QAT)。在 MobileNet 上，MNN 量化训练之后的模型准确率几乎不降。
- 持续投资于异构硬件后端的优化，尤其是利用 ARMv8.2 指令集，获得了两倍的性能提升。
- 进一步完善 Python 工具链，累计新增超过 150 个接口。
- 开源了应用层开箱即用的解决方案 MNNKit，包含了人脸跟踪与检测、人像分割、手势识别场景的解决方案。
- 开办了三期《MNN 学院》直播（1 期，2 期，3 期），增加了与用户们交流，也获得了忠粉们的大量高质量反馈意见。

截止到今天，MNN 在开源社区获得了近 4000 的 Github Stars ，这是大家对我们的工作所投的 4000 张认可票，也是鞭策我们完善 MNN 的动力。

近日，MNN 发布了 1.0.0 正式版本。自此，MNN 不再被 Github 贴上"Pre-release"的标签了！

相较于 0.2.2 版本，1.0.0 版本的主要升级在于：模型训练、异构性能和 Python 工具链。下面，我们逐项说明。

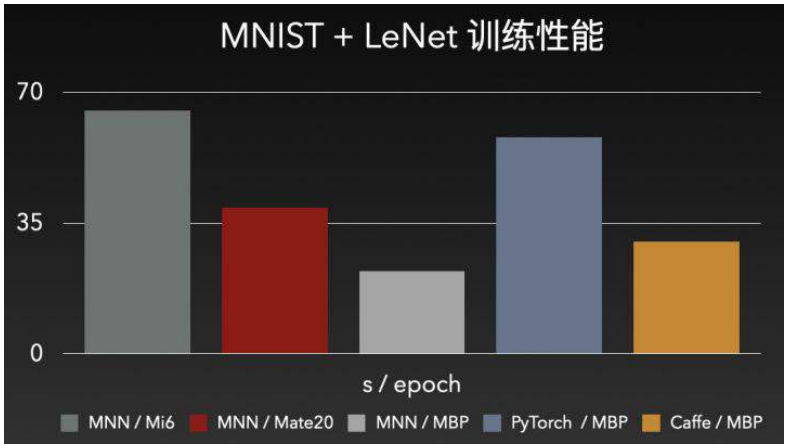
模型训练

模型构建

MNN 支持使用 `Express` （表达式）接口来构建模型，如下例所示，接口还是比较简洁明了的。模型的构建、训练和保存具体可以参考说明文档。

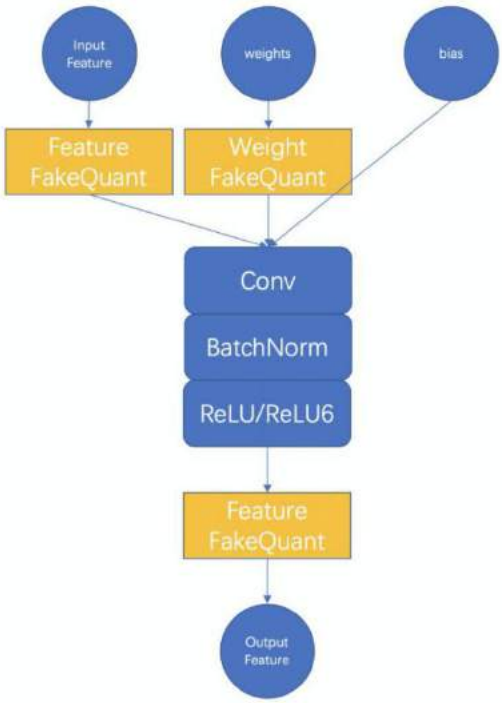
```
VARP x = inputs[0];
x      = conv1->forward(x);
x      = _MaxPool(x, {2, 2}, {2, 2});
x      = conv2->forward(x);
x      = _MaxPool(x, {2, 2}, {2, 2});
x      = _Convert(x, NCHW);
x      = _Reshape(x, {0, -1});
x      = ip1->forward(x);
x      = _Relu(x);
x      = dropout->forward(x);
x      = ip2->forward(x);
x      = _Softmax(x, 1);
return {x};
```

以 MNIST 数据集 + Lenet 网络为例，一个 epoch 60000 张图片，一般可达到 97-98%的准确率。性能上，同款 MBP 上，MNN 比 PyTorch 和 Caffe 都有明显优势；而手机上，MNN 也达到了完全可用的性能水准。



量化训练

模型量化既可以降低模型大小，又可以利用硬件特性提升推理性能，可谓业务应用必备之选。但美中不足之处在于，模型量化会带来一定的精度损失 —— 对于精度攸关的项目，就难免要做出艰难的选择了。



为此，MNN 借助自身模型训练能力，实现了模型训练量化，具体实现可以参考说明文档。精度和压缩率方面，我们以 MobileNet V2 为例说明：

模型	类型	准确率	模型大小
原始模型	float32	72.324%	13M
MNN 训练量化模型	symm int8	72.456%	3.5M
TensorFlow 训练量化模型	symm int8	71.1%	3.5M

注 1：训练和验证均采用 ImageNet 数据集。训练采用 32 为 batch size，执行 100 个迭代，即，使用了 3200 张图片进行训练；精度验证则使用了 50000 张图片。

注 2：原始模型为 TensorFlow 官方模型，官方准确率为 71.8%，但因预处理代码上有细微差别，我们测试原始模型的准确率结果稍高于官方；

可以看出，在实现了 73%模型尺寸压缩的情况下，量化模型的精度甚至要稍高于原始模型。

迁移学习示例

这里节选 MobileNet V2 的 4 分类迁移学习示例，来说明模型的 Finetune，完整示例请参考文档。

```
class MobilenetV2TransferModule : public Module {
public:
    MobilenetV2TransferModule(const char* fileName) {
        // 读取原始 MobilenetV2 模型
        auto varMap   = Variable::loadMap(fileName);
        // MobilenetV2 的输入节点
        auto input    = Variable::getInputAndOutput(varMap).first.begin()->second;
        // MobilenetV2 分类层之前的节点，AveragePooling 的输出
        auto lastVar = varMap["MobilenetV2/Logits/AvgPool"];

        // 初始化一个 4 分类的全连接层，MNN 中可以用卷积来表示全连接层
        NN::ConvOption option;
        option.channel = {1280, 4};
        mLastConv         = std::shared_ptr<Module>(NN::Conv(option));

        // 初始化内部特征提取器，内部提取器设成不需要训练
        mFix.reset(PipelineModule::extract({input}, {lastVar}, false));
        // 注意这里只注册了我们新初始化的 4 分类全连接层，那么训练时将只更新此 4 分类全连接层
        registerModel({mLastConv});
    }
    virtual std::vector<VARP> onForward(const std::vector<VARP>& inputs) override {
        // 输入一张图片，获得 MobilenetV2 特征提取器的输出
        auto pool    = mFix->forward(inputs[0]);
        // 将上面提取的特征输入到新初始化的 4 分类层进行分类
        auto result = _Softmax(_Reshape(_Convert(mLastConv->forward(pool), NCHW), {0, -1}));
        return {result};
    }
    // MobilenetV2 特征提取器，从输入一直到最后一个 AveragePooling
    std::shared_ptr<Module> mFix;
```

```
    // 重新初始化的 4 分类全连接层
    std::shared_ptr<Module> mLastConv;
};

int main(int argc, const char* argv[]) {
    std::string trainImagesFolder = argv[2];
    std::string trainImagesTxt = argv[3];
    std::string testImagesFolder = argv[4];
    std::string testImagesTxt = argv[5];

    // 读取模型，并替换最后一层分类层
    std::shared_ptr<Module> model(new MobilenetV2TransferModule(argv[1])); // arg1: /path/to/mobilenetV2Model
    // 进入训练环节
    MobilenetV2Utils::train(model, 4, 0, trainImagesFolder, trainImagesTxt, testImagesFolder, testImagesTxt);
    return 0;
}
```

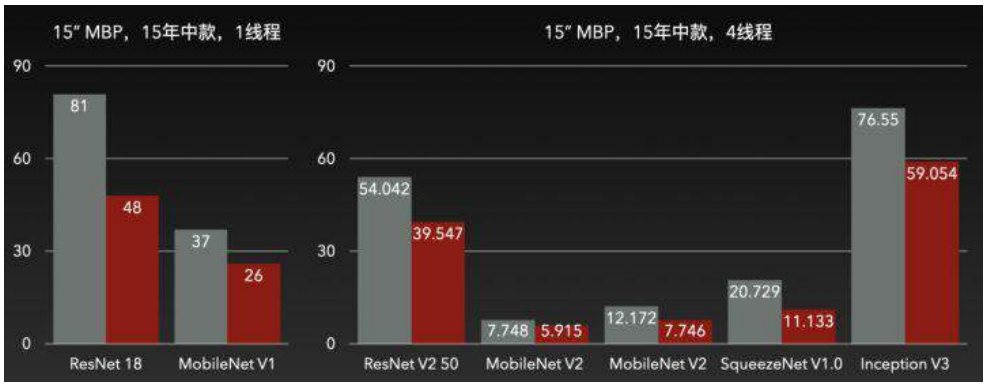
异构性能

x86

在 x86 上，我们重点优化了矩阵乘法。在分析过 AVX 和 Arm 向量乘指令差异后，我们修改了 AVX 下的权重矩阵布局，降低了 I/O 布局，以充分利用 CPU 算力。

此外，我们允许在支持 FMA 扩展的设备上，启用扩展，将乘法和加法合为一条指令，以进一步降低指令耗时。

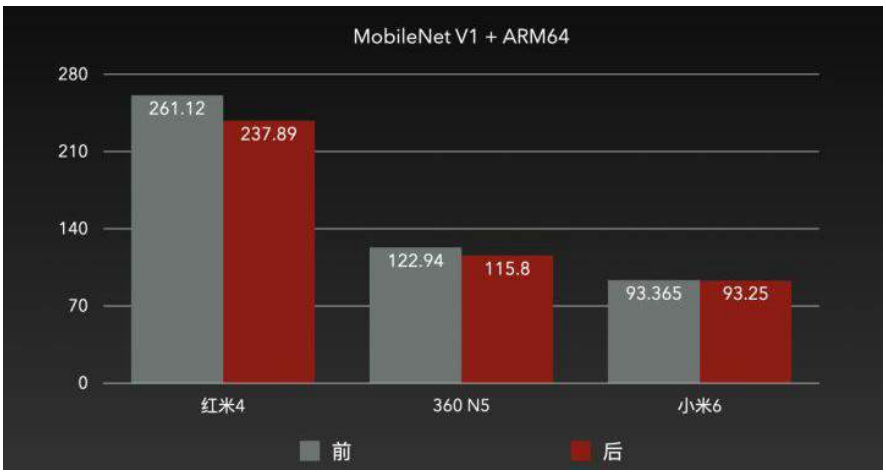
当前，FMA 扩展的启用开关还放置在 CMakeLists.txt 中，后续会在运行时判别。



综合两项优化，x86 上有 30%左右的性能优化。

ARM64

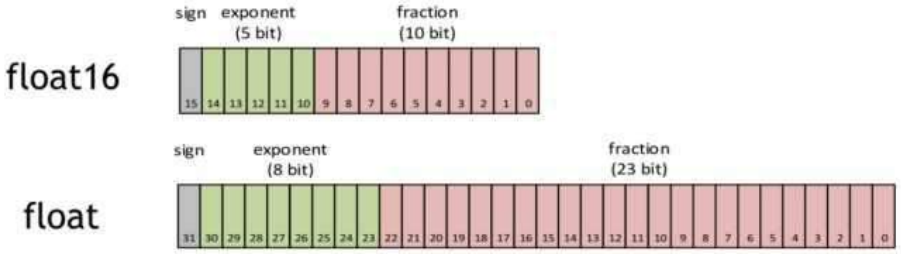
在 ARM64 上，我们面向中低端设备，调整了矩阵乘法的分块策略，矩阵中每个元素的均摊 I/O 降低了 22%；同时，将数据对齐从 32 字节调整为 64 字节，与 ARM 架构 CPU 下场景的 L1 cacheline 匹配；最后，优化了缓存预取。优化结果如下：



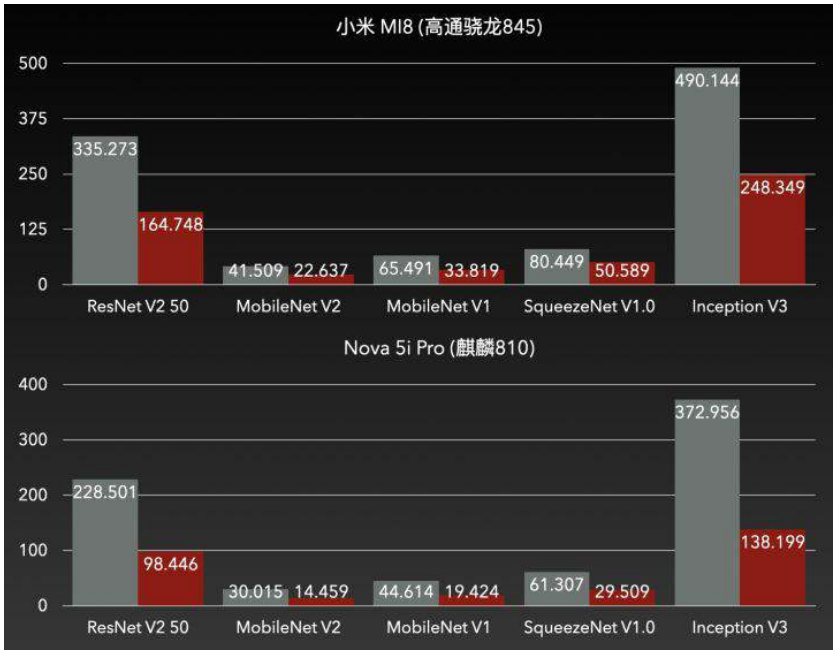
ARMv8.2

ARM 在「Bringing Armv8.2 Instructions to Android Runtime」一文中，列举了可以在 Android 运行时中应用的 ARMv8.2 新特性。其中，FP16 extensions 和 Dot Product 可以分别应用于浮点计算加速和量化计算加速。

FP16 extensions

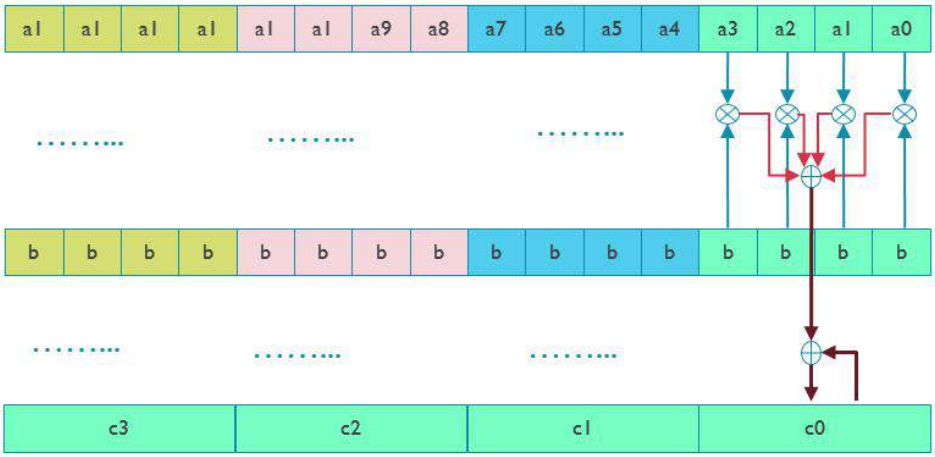


亦记作 asimdhp(Advanced SIMD Half Precision)，是 ARMv8.2 架构的可选扩展。asimdhp 可用时，可以使用相关 SIMD 指令实现 float16 的读写计算。float16 将 float32 所需的位数降低了一半，因此在 SIMD 下，可以实现两倍的并发吞吐，从而优化性能。为此，我们在卷积中，采用[N,C/8,H,W,8]的数据布局，新增了部分卷积实现，效果如下：

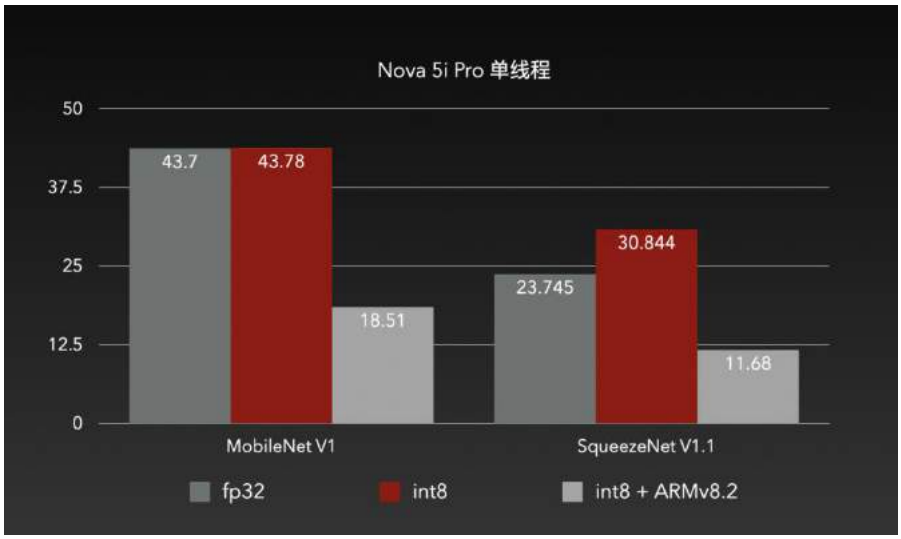


精度上几乎没有下降，但是性能足足提升了一倍。搭配上 MNN 转换工具的--fp16 输出选项，模型大小还能减小一半。一箭双雕。

Dot Product



亦记作 asimddp(Advanced SIMD Dot Product)，是 ARMv8.2 架构的可选扩展。asimddp 可用时，可以使用 SDOT/UDOT 指令实现 int8/uint8 的点积计算。SDOT/UDOT 指令如上图所示，一次可以处理两个 4x4 int8/uint8 数据乘，并累加到 4x1 的 int32/uint32 的寄存器上。这样强大的硬件加速指令，还是双发射的。



实战表现效果也非常明显，在原先 int8 无法发挥效用的设备上，ARMv8.2 也成功实现了耗时减半。

Python 工具链

2019 年的绿盟开发者大会上，我们发布了 MNN 的 Python 前端和 Python 版的转换、量化、

可视化工具。而今，Python 又增加了对 MNN Express (表达式)、模型训练的封装，累计新增超过 150 个接口。具体可以参考说明文档。

依然是前文的 Express 构图，使用 Python 改写的版本如下：

```
class Net(nn.Module):
    """construct a lenet 5 model"""
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.conv(1, 20, [5, 5])
        self.conv2 = nn.conv(20, 50, [5, 5])
        self.fc1 = nn.linear(800, 500)
        self.fc2 = nn.linear(500, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.max_pool(x, [2, 2], [2, 2])
        x = F.relu(self.conv2(x))
        x = F.max_pool(x, [2, 2], [2, 2])
        x = F.convert(x, F.NCHW)
        x = F.reshape(x, [0, -1])
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        x = F.softmax(x, 1)
        return x
```

对熟悉 Python 的开发者来说，是不是要亲切上许多呢?

注：目前 Python Express API 处于 BETA 阶段。我们会根据社区和内部的反馈持续改进 Python API，包含进行 backward incompatible 的改动。

后续计划

2020 年，我们计划每个季度发布一个稳定版本。

未来的计划，主要集中在性能、训练、NPU 支持和模型压缩。

性能

性能是 MNN 的立身之本，相信很多朋友选择 MNN，也主要出于它飙车般的性能。有兴趣的朋友，可以去看看 MNN 发表在今年 MLSys 的论文解读。

- CPU 上，移动设备方面，ARMv8.2 将是新手机的主流，上文所展示 2 倍加速比非常诱人，我们会进一步挖掘 ARMv8.2 的优化空间；其他平台方面，x86 的性能在单机训练、服务端推理的场景中举足轻重，会是性能优化的另一个目标。
- GPU 上，我们会聚焦 Vulkan —— Android 下一代 GPGPU API 的事实标准。

训练

MNN 最新拥有的训练能力已经通过 Express(表达式)接口支持常用模型的训练、量化、蒸馏，我们会进一步完善训练能力，添加更多算子和求导的支持，以支持更多的模型。

NPU 支持

NPU 具有超高的性能、超低的能耗，将是未来手机的标配。NPU 的支持，也是许多 MNN 用户经常在钉钉群里提出的需求。MNN 在未来的 1 年，会逐步支持更多的 NPU，请大家拭目以待！

模型压缩

MNN 目前已经拥有 Post-training quantization 和 Quantization-aware training 的能力。我们会持续投入模型压缩（如蒸馏，稀疏，剪枝，低比特等），给业界提供更多优秀的、即插即用模型压缩算法。

淘系 Flutter 探索实践篇

Flutter 作为新一代的跨平台技术，很好的解决了组织协同和多终端带来的研发成本问题。淘系技术部在过去的一年里，多团队多点开花，构建了基于 Flutter 的基础设施并在多个子领域有所突破。通过 AliFlutter 组织沉淀出的一系列基础设施一方面帮助集团多个 App 快速启动拿到业务结果，一方面兼容并包探索基于虚拟组织下的 Flutter 中台的运作策略。下面让我们一起了解团队在这一年的具体沉淀和收获吧。

——阿里巴巴集团 AliFlutter 项目负责人 宗心

技术经典总结

Flutter 在阿里淘系的体系化建设和业务实践

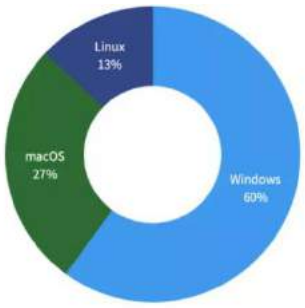
作者 | 张翰（门柳）
出品 | 淘系技术

Flutter 这两年的热度不断提升，行业内投入建设 Flutter 的公司也越来越多，有很明显的上升趋势。

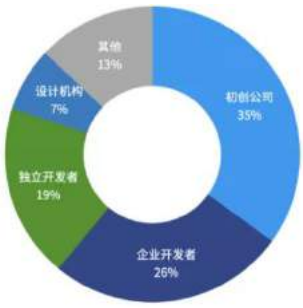
作为一个技术框架，Flutter 该有的功能都有了，但要把它应用到业务中去，还得解决工程问题、复用已有的技术积累、融入业务的工作流等，还要针对特定的业务场景做增强和扩展。所以，我们的核心目标是把 Flutter 从一个单点的技术框架，打造成完整的企业级解决方案。

Flutter 社区发展情况

在官方四月底的统计数据中，自发布以来的 16 个月内，已有 200 万开发者使用 Flutter，三月份的时候也有 10% 的增长，Google Play Store 中发布的 Flutter 应用约有 5 万个，仅在 2020 年 4 月就有近 1 万个应用上传。开发者所在的团队，初创公司最多，占了 35%，26%是企业开发者，19%是独立开发者，还有 7%的设计机构。



开发者使用的操作系统



开发者所在的团队

淘系技术

Flutter 使用者数量排名前五的地区是印度、中国、美国、欧盟和巴西。很神奇，国内开发者对 Flutter 的热情比出生地美国还要高。从各种公开信息来看，头条、微信、QQ、美团、抖音、快手都对 Flutter 保持的密切的关注，而且已经不是小规模尝试了，有些公司已经铺开准备大规模建设了。

简单来讲，Flutter 的发展势头越来越猛，国内大公司都在建设，Flutter 更受新兴业务的青睐。

阿里淘系的业务背景

淘系业务面临较大的竞争压力，整个移动互联网的竞争也越来越激烈，为了吸引并留住用户，对产品的性能、使用体验和迭代效率都有更高的要求，像直播/短视频/沉浸式交互/AR//3D 等新交互形式可以明显的提高业务竞争力，这些技术基本上都是原生能力，所以需要一个和原生技术结合度好，而且又能避免写重复代码的技术方案。Flutter 比较符合这个定位。

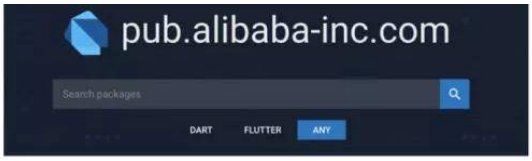
除了淘宝 App 本身，还有一系列淘系应用，如闲鱼、淘宝特价版、直播 App、躺平等新兴应用不断出现，整个阿里已经有 30+ App 在使用或者在尝试 Flutter 了，为了减少重复建设，需要沉淀出通用的技术能力，站在现存的技术积累之上，打造出完整的技术体系。

阿里在 Flutter 的技术积累

手淘从 Flutter 刚诞生以来就对它保持关注，2018 年开始就持续做技术探索，去年启动了 AliFlutter 的项目，希望可以拉通阿里内部 Flutter 相关的技术积累，有了一些初步的成果。

基础设施建设

首先标准化了 Flutter 的工程构建。有统一的代码仓库来管理 Flutter 的源码，有专门的升级发版节奏，确保可以跟进官方的迭代。客户端打包的环境配置比较复杂，还要和阿里自身的开发环境做融合，这些功能已经形成了平台和工具，也开放了部分功能让大家可以定制和扩展自己的引擎。



内部 pub 服务

对接内部打包平台

另外还提供了阿里内部的 pub 库实现包管理，类似前端的 npm，让 Flutter、Dart 的通用功能库有地方可以承接。

而且还与手淘的基础设施做了紧密结合。对接了内部的摩天轮（MTL）打包平台、性能稳定性监控的高可用平台，手淘基础的图片库、网络库、日志、埋点和一部分基础中间件。Flutter 的开发、测试、打包、集成、发布、运维监控等功能，都融入了客户端开发本身的工作流中。

完善 Flutter 的基础设施是后续所有工作的基础，让 Flutter 开发不再处于刀耕火种的工作环境。

内核增强与扩展

把 Flutter 落地到业务的过程，也不是一帆风顺的。性能倒不是瓶颈，但是包体积太大、内存占用高是经常遇到的问题，所以去年有一部分精力花在 Flutter Engine 本身的优化上，分析二进制产物的包结构，压缩 Flutter Engine 和 Dart AOT 编译产物的大小。分析 DartVM 的内存管理机制和 snapshot 功能，不断优化内存。而且有针对性的对图片库的内存、Engine 的启动流程做了优化。

在图形化方面，Flutter Engine 提供了一套兼容性和性能都比较好的图形接口，而且跨平台一致。所以基于 Flutter Engine 封装了 2D Canvas API 透出到 JS 环境，性能和稳定性相比之前的技术都有大幅提升。而且对接到了淘宝小程序的容器环境，支撑互动类的业务。

信息互通

在阿里内部建立了 AliFlutter 信息互通的虚拟组织，淘宝、闲鱼、特价版、飞猪、盒马、UC、钉钉、CBU/ICBU、优酷、本地生活、搜索、天猫精灵等 BU 都有接口人在参与，定期开会同步信息，避免出现重复建设。

另外在社区我们和 Google 官方团队也保持着联系，定期开语音会议。这个合作关系建立之后，在社区也可以混个脸熟，我们提的 issue 和 PR 说不定会被优先处理~

技术探索和实践

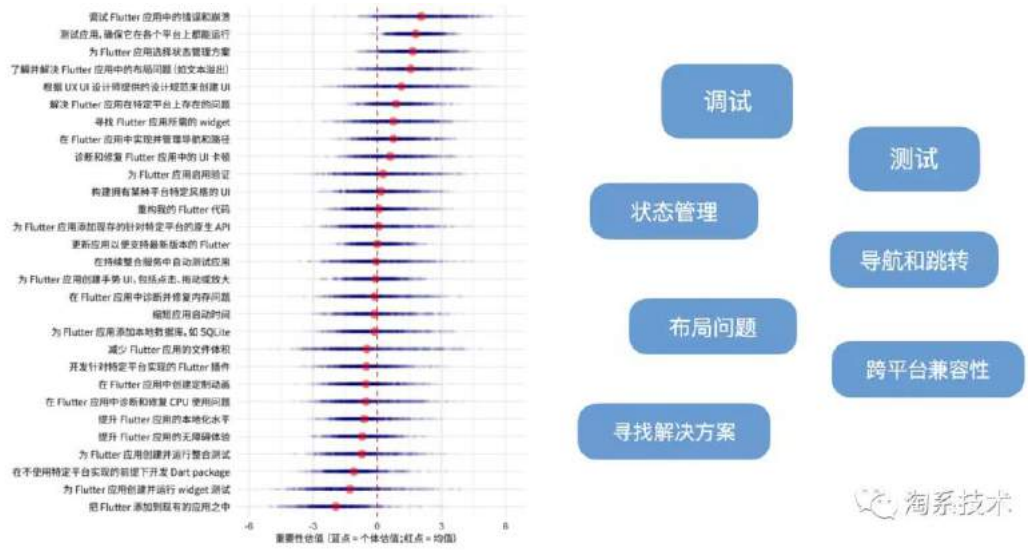
我们团队积累了 15+ 高质量的技术文章分享在内部的技术论坛，在原理分析、性能优化、应用和实践这三个大类里，分析 DartVM 的原理以及背后的 GC 和 Snapshot 机制，研究 Flutter 的渲染管线，思考 Web 生态和 Flutter 对接的方案，优化 Flutter Engine 的启动时间和包大小，解决实际业务中图片库接入、长列表内存占用高等问题。

- 阿里 Flutter 体系化建设
- 闲鱼 Flutter 一体化演进之路
- Flutter Widget 和 CSS 布局原理 PK
- Flutter 和 Web 生态对接

- Flutter 渲染引擎深入分析
- 在 Flutter 体系中对接手淘图片库
- 细化 Flutter List 内存回收，解决大 Cell 问题
- Flutter 瀑布流及通用列表解决方案

目前待解决的问题

下图左侧是 Flutter 官方做过一次问卷调查，评选出了开发者实际开发过程中最关心的问题。



前六项任务摘录出来如下：

	最重要的任务	最难的任务
1	调试 Flutter 应用中的错误和崩溃	解决 Flutter 应用在特定平台上出现的问题
2	测试应用，确保其在各个平台上都能运行	诊断和修复 Flutter 应用中的内存问题
3	为 Flutter 应用选择状态管理方法	诊断和修复 Flutter 应用中的 CPU 使用问题
4	了解并解决布局问题（如文本溢出）	将现有的针对特定平台的原生 API 添加至 Flutter 应用

5	根据 UX/UI 设计师提供的设计规范来创建 UI	诊断和修复 Flutter 应用中的 UI 卡顿
6	解决 Flutter 应用在特定平台上出现的问题	开发 Flutter 插件（针对特定平台的实现）

这是面向所有社区开发者的问卷，我们在实际业务中遇到的问题也类似，总结下来主要是对开发效率、调试测试、解决方案、功能复用的诉求比较强烈。说明 Flutter 作为一个新兴的研发模式，在工程、工具方面做得还不够成熟，生态还不够繁荣，尤其是贴合业务场景的解决方案比较少。

这些问题是我们下一步的工作重点，从实际的业务痛点出发，不断完善 Flutter 的内核能力和平台支撑，在保障产品质量和体验的同时，提升业务的交付效率。

总结与未来规划：Unicorn 独角兽

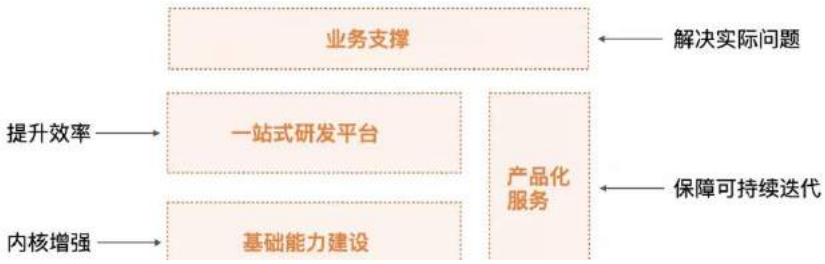
用「独角兽」项目孵化更多「独角兽」。快速迭代，野蛮增长！

解题思路

前面介绍了，这个项目的核心目标是：

- 在业务视角，完善 Flutter 的研发模式和平台支撑，在保障产品质量和体验的同时，提升业务的交付效率。
- 在技术视角，建设产品化的 Flutter 技术体系，将一个单点的技术框架，打造成完整的企业级解决方案。

为了实现上面的目标，我们从四个层面做建设，最下面是对 Flutter 本身内核的增强，然后再向上提供一站式的研发平台提升业务的迭代效率，同时也提供产品化的服务保障整个项目可持续迭代，最上面是业务支撑，解决在业务落地过程中实际遇到的问题。



这个项目并没有推翻任何积累，没有造新轮子，而是用更加体系化的方式，梳理在做的关于 Flutter 的事情，做一个整体规划，并且和集团内所有做 Flutter 技术的团队保持信息同步，以开放的心态做技术，技术成果互通互享。

整体规划

顺着上面的思路，把各项内容展开来看，是这个样子的：

整个项目是在之前的技术积累上进一步做建设，首要原则是降低对 Flutter 本身技术的侵入性，确保架构可持续迭代。因为 Flutter 还是一个快速迭代的技术，一旦开始分化，再跟进官方的进展就很难了，前期接入的业务就会成为历史包袱，要极力避免这种情况。

电商系的业务场景很复杂，不同场景的技术诉求也不一样，所以架构上要做到可灵活扩展，让垂直类的能力可以快速接进来，实现差异化。另外为了降低业务的使用成本，要保证功能的开箱即用，优化开发体验。

关于演进路线，Flutter 本身的技术特点也比较适合历史包袱比较少的新兴业务，所以优先考虑的是新型 App 和垂直类的业务，然后再推广到常规业务中。从技术角度是自底向上做建设，首先保障底盘的稳定性，而不是上来就搭花架子。对接好基础设施之后，下一步是为业务提效，然后再做扩展和对接，最后沉淀出靠谱的技术方案回馈社区。

基础能力增强

基础能力的建设和可以分为三层：

- 应用层：这部分主要解决在业务实践过程中遇到的问题，提供解决方案。闲鱼是 Flutter 的技术先驱，在应用层有很多积累，比如 Boost(混合栈) 和 Candy(游戏化开发)，除此之外我们还有正在做 DX Flutter 容器(模板化/动态化渲染)，卡片级嵌入(multi-window) 等技术方案。
- 引擎层：这部分逐渐走向技术的深水区，分别在启动链路、AOT 编译、DartVM、渲染管线、图形化/光栅化等领域做优化和增强，在业务无感的情况下，提升内核的性能和稳定性。
- 对接层：这部分工作大部分已经完成，继续查漏补缺，不断完善使用体验。

一站式研发平台

回顾前面列的 Flutter 还存在的问题，大部分在于工程链路和生态不够成熟，这是业务使用 Flutter 时遇到的最大阻力。这部分带有业务属性的工作 Google 官方不会为你做，必须要企业自己来做。所以除了完善 Flutter 基础能力之外，面向业务场景，还要有全链路的平台和工具支撑，才能真正提升业务的迭代效率。

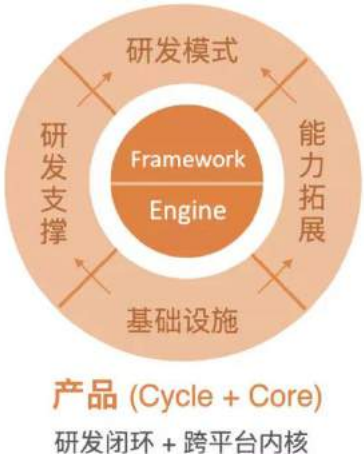
基于手淘首页在用的模板化的框架实现部分 UI 的动态化，加上 Flutter 的跨平台渲染能力，再结合 GAIA 端云一体化的研发模式，形成一个完整的研发闭环，从开发者视角整理工作流，用一个平台提供所有能力。目标是降低 Flutter 在业务中的使用成本，让开发者可以在 60 分钟内完成 App 首次接入，30 分钟内就可以开始写第一行代码。

业务支撑

整个项目首先支撑的是淘系 1+X 的 App，贴身解决业务遇到的实际问题。另一方面也会做一些产品化的事情，把可复用的技术方案沉淀下来，有地方承接和展示，而且会贡献代码和方案到社区，提升大家在公司内外的技术影响力。

产品化

整个技术产品的逻辑是 Cycle + Core，即「研发闭环」+「跨平台内核」。



跨平台内核是对 Flutter 本身的 Framework 和 Engine 持续做增强，保持和官方版本的同步，尽量将通用的代码合回官方仓库。研发闭环是根据当前的业务环境、结合已有的技术积累、提供平台支撑，把功能做得完备，而且好用。
核，深入研究引擎技术；环，不断提高迭代效率。

技术细节

前面是概述，其中几项具体的技术我可以简单介绍一下。

工程和效率

在真正的业务实践里，很少把一整个 App 都用 Flutter，基本上都是把 Flutter 当成一个模块引入，在开发和调试 时还依赖自身的各种库，如网络、图片、安全认证等。而 Flutter 的工具链是要基于壳工程来打包和调试的，很多情况下，在壳工程里没法实现完整的业务逻辑。

所以我们在 Flutter 工具链基础上提供了扫码开发的能力，直接使用完整的 App，如淘宝特价版，扫码连接本地环境，修改代码后可以实时在真机上预览实际效果。
除了对本地开发环境的增强，我们也在考虑和云平台的联动，借助 FaaS 的支持，实现在一个工程里、一套技术栈开发完整的应用逻辑。

模板化开发

Flutter 的动态化能力一直是比较弱的，然而电商系的应用对动态化有强诉求，在淘宝 App 首页和核心交易链路里使用到了一个模板化的渲染框架 DinamicX，可以动态下发模板来驱动页面渲染，使用到的组件以及绑定的业务逻辑是原生语言开发的，如 Java 和 Objective-C，不支持 JS 等依赖脚本引擎的语言，这样性能更好，审核风险更低。

我们正在把这一套成熟的模板化的渲染协议迁移到 Flutter 上来，既不破坏 Flutter 页面本身的性能体验，也在一定程度上提供了动态化的能力。

多 FlutterView 嵌入

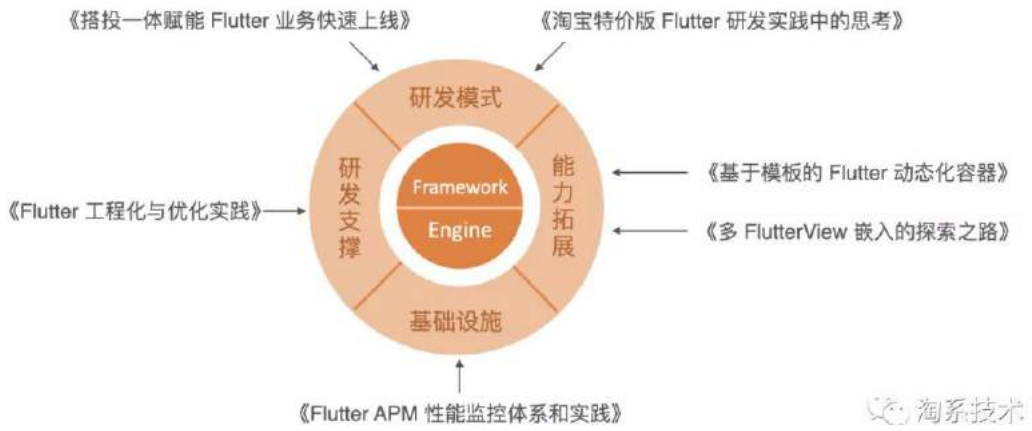
在 Flutter 页面的某个区块使用原生组件可以用 PlatformView，还有一种情况是，在原生渲染的页面里，某一个区块是用 Flutter 实现的。也就是说，不拿 Flutter 写一个完整的页面，而是写一个（或多个）组件嵌入到原生页面中，这项能力在真实的业务场景中也是有诉求的。

Flutter 仓库的 issue 里也有人提出了 multi-window API 的需求，我们也在评估各种实现方案，关键问题是要不要在多个窗口间复用同一个 Engine 实例，以及如何管理多个 Scene 等。这方面我们也在开源社区里讨论技术方案，及时跟进这项功能的实现。

业务实践

闲鱼是使用 Flutter 技术的先驱，积累了比较丰富的实践经验，淘宝特价版也在大范围的使用 Flutter，期待业务开发者们给大家介绍更丰富的细节吧。

如果上面这些资料看得还是不够过瘾，不用着急，我们接下来会安排多场直播，给大家详细介绍其中的细节。



直播的主题还是围绕着我们的产品逻辑，介绍各个模块所做的工作，更多的是实践类的经验，希望对大家有帮助~

Flutter 快速学习和入门方法

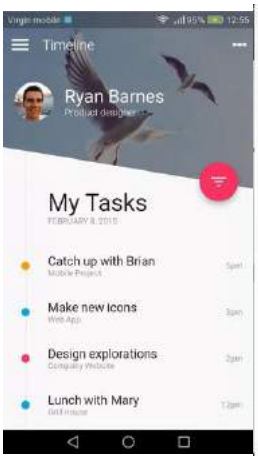
作者 | 钮肇杰（衣客）
出品 | 淘系技术

近两年 Flutter 的热度不断提升，无论在阿里还是外部公司，参与 Flutter 生态建设的人越来越多。Flutter 作为跨端的 UI 框架，未来也有很大的可能像 Rax 一样作为集团内营销活动页面搭建的 DSL。所以目前学习 Flutter，参与 Flutter 生态建设是一件时髦且有价值的事情。

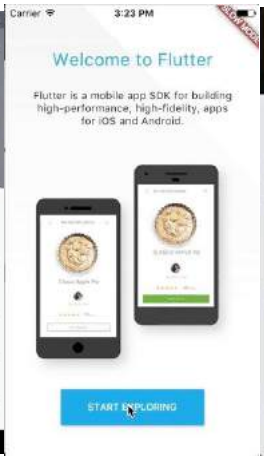
Flutter 是什么

Flutter 是谷歌的移动 UI 框架，可以快速在 iOS 和 Android 上构建高质量的原生用户界面。Flutter 可以与现有的代码一起工作。在全世界，Flutter 正在被越来越多的开发者和组织使用，并且 Flutter 是完全免费、开源的。简单来说，Flutter 是一款移动应用程序 SDK，包含框架、控件和一些工具，可以用一套代码同时构建 Android 和 iOS 应用，并且性能可以达到原生应用一样的性能。

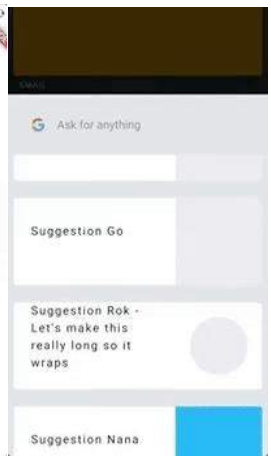
Flutter 简介



点击查看动图演示



点击查看动图演示



点击查看动图演示

配置开发环境

Flutter 开发可以在 macOS，Linux 或 Windows 上完成。虽然您可以在 Flutter 工具链中使用任何编辑器，但 IntelliJ IDEA，Android Studio 和 Visual Studio Code 的 IDE 插件可以简化开发工作。

- 下载 flutter SDK。地址
- 将 flutter 的 bin 目录添加到 path 中。
- 执行 flutter doctor 命令，他会安装 flutter 框架，包括 dart，而且提示你任何其他需要安装的依赖。
- 安装其他依赖。
- 在 IDE 中安装 flutter 插件。

体验

Android Studio (为 Flutter 提供完整的 IDE 体验)

创建应用

1. 选择 File>New Flutter Project
2. 选择 Flutter application 作为 project 类型，然后点击 Next
3. 输入项目名称 (如 myapp)，然后点击 Next
4. 点击 Finish
5. 等待 Android Studio 安装 SDK 并创建项目. 在项目目录中，您应用程序的代码位于 lib/main.dart.

运行应用程序

定位到 Android Studio 工具栏:

1. 在 target selector 中，选择一个运行该应用的 Android 设备. 如果没有列出可用，请选择 Tools>Android>AVD Manager 并在那里创建一个
2. 在工具栏中点击 Run 图标，或者调用菜单项 Run > Run.

VS Code(轻量级编辑器，支持 Flutter 运行和调试)

创建应用

1. 启动 VS Code.
2. 调用 View>Command Palette...
3. 输入 'flutter'，然后选择 'Flutter: New Project' action
4. 输入 Project 名称 (如 myapp)，然后按回车键
5. 指定放置项目的位置，然后按蓝色的确定按钮
6. 等待项目创建继续，并显示 main.dart 文件 在项目目录中，您应用程序的代码位于 lib/main.dart.

运行应用程序

1. 确保在 VS Code 的右下角选择了目标设备
2. 按 F5 键或调用 Debug>Start Debugging
3. 等待应用程序启动

Terminal + 编辑器

创建应用

使用 flutter create 命令创建一个 project:

```
$ flutter create myapp
$ cd myapp
```

运行应用程序

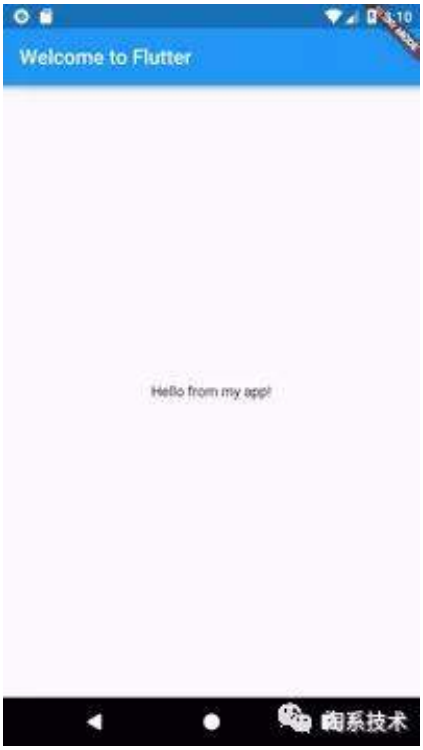
检查 Android 设备是否在运行。如果没有显示，请参照 设置。

```
$ flutter devices
```

运行 flutter run 命令来运行应用程序:

```
$ flutter run
```

如果一切正常，您应该在您的设备或模拟器上会看到启动的应用程序:



项目结构

```
┬
└ projectname
┬
├ android          - Android 部分的工程文件
├ build            -  项目的构建输出目录
├ ios               - iOS 部分的工程文件
├ lib               -  项目中的 Dart 源文件
┬
└ src             -  包含其他源文件
└ main.dart    -  自动生成的项目入口文件，类似 RN 的 index.js 文件
├ test              -  测试相关文件
└ pubspec.yaml -  项目依赖配置文件类似于 RN 的  package.json
```

如何添加 Flutter 项目所需的依赖?

- 在 Android 中，你可以在 Gradle 文件来添加依赖项；
- 在 iOS 中，通常把依赖添加到 Podfile 中；
- 在 RN 中，通常是由 package.json 来管理项目依赖；

Flutter 使用 Dart 构建系统和 Pub 包管理器来处理依赖。这些工具将 Android 和 iOS native 包装应用程序的构建委派给相应的构建系统。

```
dependencies:
  flutter:
    sdk: flutter
google_sign_in: ^3.0.3
```

在 Flutter 中，虽然在 Flutter 项目中的 Android 文件夹下有 Gradle 文件，但只有在添加平台相关所需的依赖关系时才使用这些文件。否则，应该使用 pubspec.yaml 来声明用于 Flutter 的外部依赖项。

iOS 也是一样，如果你的 Flutter 工程中的 iOS 文件夹中有 Podfile，请仅在添加 iOS 平台相关的依赖时使用它。否则，应该使用 pubspec.yaml 来声明用于 Flutter 的外部依赖项。

如何归档图片资源以及如何处理不同分辨率

- 虽然 Android 将 resources 和 assets 区别对待，但在 Flutter 中它们都会被作为 assets 处理，所有存在于 Android 上 res/drawable- *文件夹中的资源都放在 Flutter 的 assets 文件夹中。
- 与 Android 类似，iOS 同样将 images 和 assets 作为不同的东西，而 Flutter 中只有 assets。被放到 iOS 中 Images.xcasset 文件夹下的资源在 Flutter 中被放到了 assets 文件夹中。

在 Flutter 中 assets 可以是任意类型的文件，而不仅仅是图片。例如，你可以把 json 文件放置到 my-assets 文件夹中。

```
my-assets/data.json
```

记得在 pubspec.yaml 文件中声明 assets：

```
assets:
- my-assets/data.json
```

然后在代码中我们可以通过 AssetBundle 来访问它：

```
import 'dart:async' show Future;
import 'package:flutter/services.dart' show rootBundle;

Future<String> loadAsset() async {
return await rootBundle.loadString('my-assets/data.json');
}
```

对于图片，Flutter 像 iOS 一样，遵循了一个简单的基于像素密度的格式。Image assets 可能是 1.0x 2.0x 3.0x 或是其他的任何倍数。这个 devicePixelRatio 表示了物理像素到单个逻辑像素的比率。

Android 不同像素密度的图片和 Flutter 的像素比率的对应关系

```
ldpi      0.75x
mdpi       1.0x
hdpi       1.5x
xhdpi      2.0x
xxhdpi     3.0x
xxxhdpi 4.0x
```

举个例子，要把一个名为 my_icon.png 的图片放到 Flutter 工程中，你可能想要把它放到 images 文件夹中。把图片（1.0x）放置到 images 文件夹中，并把其它分辨率的图片放在对应的子文件夹中，并接上合适的比例系数，就像这样：

```
images/my_icon.png            // Base: 1.0x
imageimages/2.0x/my_icon.png    // 2.0x
imageimages/3.0x/my_icon.png    // 3.0x image
```

接下来就可以在 pubspec.yaml 文件中这样声明这个图片资源：

```
assets:
    - images/my_icon.png
```

现在，我们就可以借助 AssetImage 来访问它了。

```
return AssetImage("images/a_dot_burr.jpeg");
```

也可通过 Image widget 直接使用：

```
@override
Widget build(BuildContext context) {
return Image.asset("images/my_image.png");}
```

如何归档 strings 资源，以及如何处理不同语言？

不像 iOS 拥有一个 Localizable.strings 文件，Flutter 目前没有专门的字符串资源系统。目前，最佳做法是将 strings 资源作为静态字段保存在类中。例如：

```
class Strings {
static String welcomeMessage = "Welcome To Flutter";
}
```

然后像如下方式来访问它：

```
Text(Strings.welcomeMessage)
```

默认情况下，Flutter 只支持美式英语字符串。如果你要支持其他语言，请引入 flutter_localizations 包。你可能也要引入 intl 包来支持其他的 i10n 机制，比如日期/时间格式化。

```
dependencies:
  # ...
  flutter_localizations:
  sdk: flutter
intl: "^0.15.6"
```

要使用 flutter_localizations 包，还需要在 app widget 中指定 localizationsDelegates 和 supportedLocales。

```
import 'package:flutter_localizations/flutter_localizations.dart';

MaterialApp(
localizationsDelegates: [
// Add app-specific localization delegate[s] here
GlobalMaterialLocalizations.delegate,
GlobalWidgetsLocalizations.delegate,
],
supportedLocales: [
const Locale('en', 'US'), // English
const Locale('he', 'IL'), // Hebrew
```

```
// ... other locales the app supports
],
// ...)
```

这些代理包括了实际的本地化值，并且 supportedLocales 定义了 App 支持哪些地区。上面的例子使用了一个 MaterialApp ，所以它既有 GlobalWidgetsLocalizations 用于基础 widgets，也有 MaterialWidgetsLocalizations 用于 Material wigets 的本地化。如果你使用 WidgetsApp ，则无需包括后者。注意，这两个代理虽然包括了"默认"值，但如果你想让你的 App 本地化，你仍需要提供一或多个代理作为你的 App 本地化副本。

当初始化时，WidgetsApp 或 MaterialApp 会使用你指定的代理为你创建一个 Localizations widget。Localizations widget 可以随时从当前上下文中访问设备的地点，或者使用 Window.locale。

要访问本地化文件，使用 Localizations.of() 方法来访问提供代理的特定本地化类。如需翻译，使用 intl_translation 包来取出翻译副本到 arb 文件中。把它们引入 App 中，并用 intl 来使用它们。

更多 Flutter 中国际化和本地化的细节，请访问 internationalization guide ，里面有不使用 intl 包的示例代码。

组件

在 Flutter 中一切皆是 组件，仅仅 Widget 的子类和间接子类就有 350 多个，如此多的组件到底如果学习，真的需要学习 350 多个组件？经济学中有一个著名的 二八定律 ,而我们学习 Flutter 也适用于二八定律，大部分组件是平时很少用到的，因此作为初学者，只需学习那 20% 常用的组件即可。

RenderObjectWidget 及其子类共有 89 个：

ProxyWidget 及其子类共有 34 个：

StatelessWidget 及其子类共有 89 个：

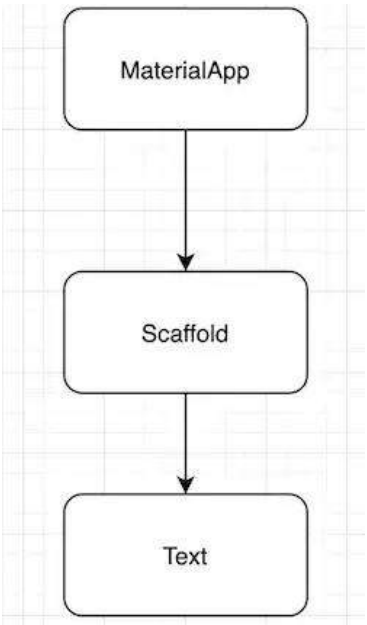
StatefulWidget 的子类最多，高达 141 个

组件树

Flutter 创建 App 的时候，所有的组件最后会生成一个组件树，例如如下代码：

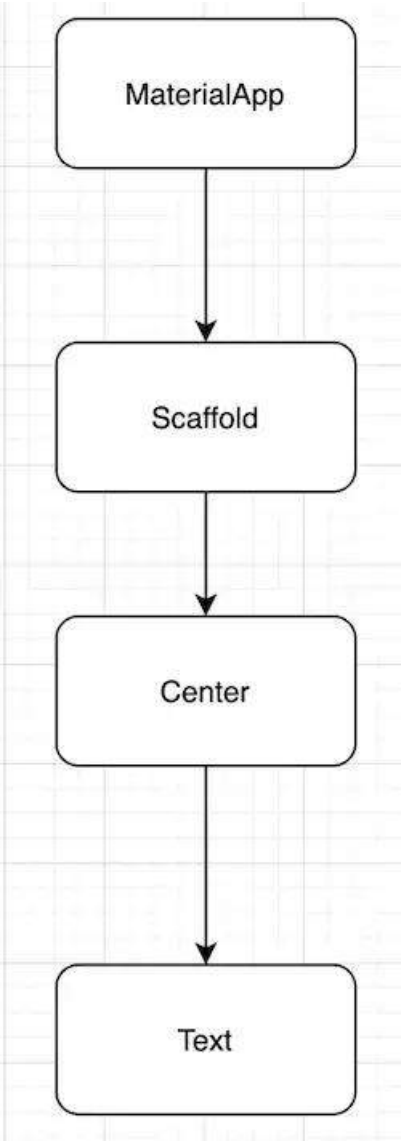
```
void main() {
  runApp(MyApp());
  }

  class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
    title: 'Flutter Demo',
     theme: ThemeData(
    ),
    home: Scaffold(
    body: Text('Hello world!'),
),
 );
}
}
```

main 函数是应用程序开始的地方，运行 MyApp 组件。生成的组件树如下：



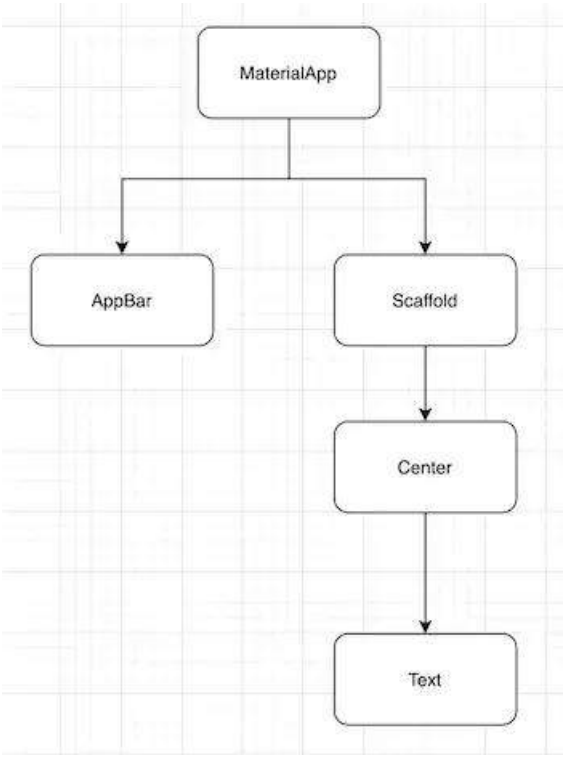
让 Text 组件居中，修改如下：

```
Scaffold(
body: Center(
child: Text('Hello world!'),
),
)
```



给应用程序添加 AppBar：

```
Scaffold(
appBar: AppBar(),
body: Center(
  child: Text('Hello world!'),
  ),
    )
```



StatefulWidget vs StatelessWidget

Flutter 中组件分为 无状态组件（StatelessWidget） 和 有状态组件（StatefulWidget）两种。它们唯一的区别就是运行时 重新加载 组件的方式不同，StatelessWidget 组件重新加载时重新创建当前组件的实例，而 StatefulWidget 组件重新加载时不会重新创建实例，而是重新执行 build 函数。

StatelessWidget 组件创建的方式：

```
class StatelessWidgetDemo extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
  return Container();
}
  }
```

build 函数返回当前组件，此组件一旦创建将不可改变，build 函数只能执行一次。如果想重新绘制此组件，只能重新创建此组件新的实例。

StatefulWidget 组件创建的方式：

```
class StatefulWidgetDemo extends StatefulWidget {
  @override
  _StatefulWidgetDemoState createState() => _StatefulWidgetDemoState();
}

class _StatefulWidgetDemoState extends State<StatefulWidgetDemo> {
  @override
  Widget build(BuildContext context) {
    return Container();
}
}
```

StatefulWidget 组件的创建方式和 StatelessWidget 不同，State<> 中的 build 函数返回当前组件，有状态的组件可以在其生命周期内多次重绘，即多次调用 build 函数，而不是创建一个新的实例。

StatefulWidget 组件重绘需要调用 setstate 方法，setstate 会使其自身及其子组件重绘，所以尽量封装 StatefulWidget 组件，避免无效的重建和重绘，影响性能。

快速书写小技巧：在 Android Studio 和 VS Code 中 输入 stl 然后点击回车，可以快速创建 StatelessWidget 组件，同理输入 stf 点击回车，可以快速创建 StatefulWidget 组件，这是编辑器 Live Templates 的功能。

Material vs Cupertino

Flutter 中包含两套风格的组件，分别是 Material 和 Cupertino ，Cupertino 是 iOS 风格的组件，命名都带 Cupertino 前缀，比如 CupertinoSlider 、 CupertinoDatePicker 等，Material Design 是由 Google 推出，旨在为手机、平板电脑、台式机和"其他平台"提供更一致、更广泛的"外观和感觉"。

Flutter 使用一套代码在不同的平台上表现一致，它不会根据不同的平台绘制不同的外形，比如使用 AlertDialog 弹出警告框，不管在 Android 上，还是在 iOS 上效果是一样。

但有一些功能 Flutter 区分平台，比如 ListView 滑动到底部时继续滑动，Android 底部会出现淡蓝色（默认情况下）拱形，而 iOS 上则没有，这是因为 Flutter 在封装此组件时在代码中区分了平台，所以在查看 Flutter 源码到过程中会经常看到根据不同的平台做不同处理的情况。

总结

Flutter 学习成本主要还是在组件的熟悉上，能熟练常用组件就可以上手项目了，至于其他的控件只需大概浏览一下，做项目的时候遇到一些功能能够想起 Flutter 已经提供了此组件就可以了。

前端工程师学习 flutter 的方法经验

作者 | 朱炎（一晟）
出品 | 淘系技术

当下最火的跨端技术，当属于 Flutter ，应该没人质疑吧。一个新的技术的趋势，最明显的特征，就是它一定想把"前浪"拍死在沙滩上。这个前浪，就是"react Native","weex"。目前随便在搜索引擎上 搜索"Flutter reactNative",就全是这两个技术的对比，评测。

前言

一股股浓浓 ： 不服来 “掰” 啊 ！ ！ ！ 的味道。

是的，错过了 react Native，weex 这些 “炸” 翻前端的技术，不能在错过 Flutter 了，这年头，你不会一门跨端技术，怎么好意思说自己是【前端】。

2020 年 5 月，Flutter 团队带来了的第一个稳定版本，Flutter 1.17.x 版。

按照官方的说法，新版本关闭了自 Flutter 1.12 版本的 6,339 个问题，从 231 位贡献者那里合并了 3164 个 PR，并修复了许多错误。除了质量改进之外，新版本还在增加了一些新功能：

- 移动性能和应用程序包大小的改进

  运行速度将提高 20%-37%,iOS 动画减少 40%的 CPU / GPU 使用率；Flutter Gallery 示例，包大小减少了 18.5%，快速滚动浏览大图像时，减少了 70%的内存。

- Metal 对 iOS 性能提升 50%

  开发者可以使用 Metal API 直接访问底层 GPU，iOS 设备上，Flutter 将默认使用 Metal，使得应用在绝大多数情况下都运行得更快，渲染速度平均提升约 50%。

- 新的 Material Widget

  这是一个新的 widget NavigationRail 组件，它由 Google Material Design 团队设计并实现。 NavigationRail 非常适合在移动和桌面设备之间切换的应用。还增加了新的 Animations 库，提供了实现新的 Material motion 规范的预构建动画。

- Dart DevTools for Flutter

  这是一个神器，用过 chrome DevTools 的一定不陌生，Dart DevTools 用于 Dart 和 Flutter 代码调试和性能分析的工具。这套工具集功能非常全，包含性能、UI、热更新、热重载、log 日志等很多功能。（这个工具以后会专门一期介绍）。

- **新的 Flutter 文本主题和 Google 字体**

  在新版本中，Flutter 团队在不破坏现有 Flutter 应用的同时，完成了对 2018 年 Material Design 规范文字排版缩放 (Type Scale) 的实现。也就是说 TextTheme API 是向前兼容的，对于旧的废弃名称只做警告。同时新的 主题也可以搭配 新的 Google Fonts for Flutter v1.0 字体。

- **无障碍功能和国际化**

  更加全面的工作，对滚动、文本框以及其他输入 widget 的无障碍功能进行了修复。GitHub 上有在这个版本中完成修复的无障碍功能完整列表。国际化方面，完成了三星键盘输入法的 bug。

- **其他重要改动**

  在使用 pushReplacement 时，保留之前的路径，继续执行动画

  弃用 UpdateLiveRegionEvent

  在高速滚动时延迟图像解码

  文本选取溢出 bug（Android)

  空绘图对象的断言缓存提示设置

  实时图像缓存

  可视范围内，确保构造块高度计算选区准确

  在 gen_l10n 中生成消息查询

  从 RouteSettings 中移除 isinitialroute

  将 mouse_tracking.dart 移动到 rendering

具体细节可以参考官方地址：https://medium.com/flutter/flutter-spring-2020-update-f723d898d7af

每次迭代都有令人小激动的改进，但是对于刚接触 Flutter 的 web 前端同学来说还是……

好了，回到标题，笔者作为一名传统 web 前端，想从前端最熟悉的视角“躺”着把 Flutter 了解一遍，不要敬仰，平视它！！！先从兴趣开始。

正文

Flutter 环境的搭建，其实有很多资源可以参考。这里就不赘述了（知道有很多坑，在后续文章中,有机会把个人遇到的坑汇总一下 ）。

有兴趣可以参考 「flutter 安装环境的搭建」(https://flutter.cn/docs/get-started/install/macos) ，在这里建议各位，一定要自己亲自搭一下环境，跑一下官方 demo，小马过河，焉知深浅，自己定的位才是最准确的。

有了环境，「配置编辑器」 (https://flutter.cn/docs/get-started/editor)。

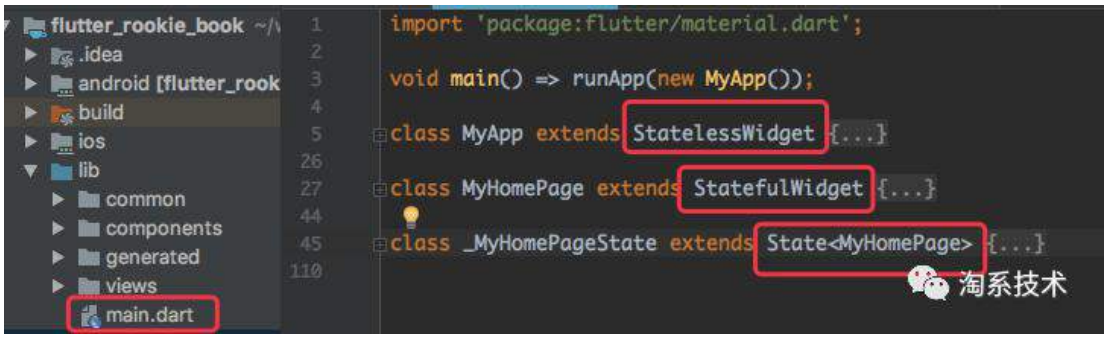
然后「创建一个 Flutter 项目」(https://flutter.cn/docs/get-started/codelab)。

项目创建完成，可以先用 flutter run 跑一下。

```
flutter doctor // 让 flutter 医生检查一下环境和配置，得到四个 ☑ 后，恭喜你通过考核。
flutter run // 奔跑吧...
```

好了，跑起来了吧，你会看到一个计数的官方示例，点击加号图片可以做加运算。

这时候我们看项目的 project 目录里 有一个入口文件叫 main.dart。然后打开 main.dart 就像下面这样:



（为什么你们看到代码比我的长，因为我折叠了!!!）这不是重点，重点是每个类继承的都是一个尾号为 Widget 的字符。

聪明的你一定会觉得 Widget 和 Flutter 有着某种神秘的联系。

“Binggo！"，是的，Flutter 有两个重型武器，一个叫 Dart ,另一个就是 Widget 了。

Dart 一切皆来自 Object，Flutter 的组件皆来自 Widget。

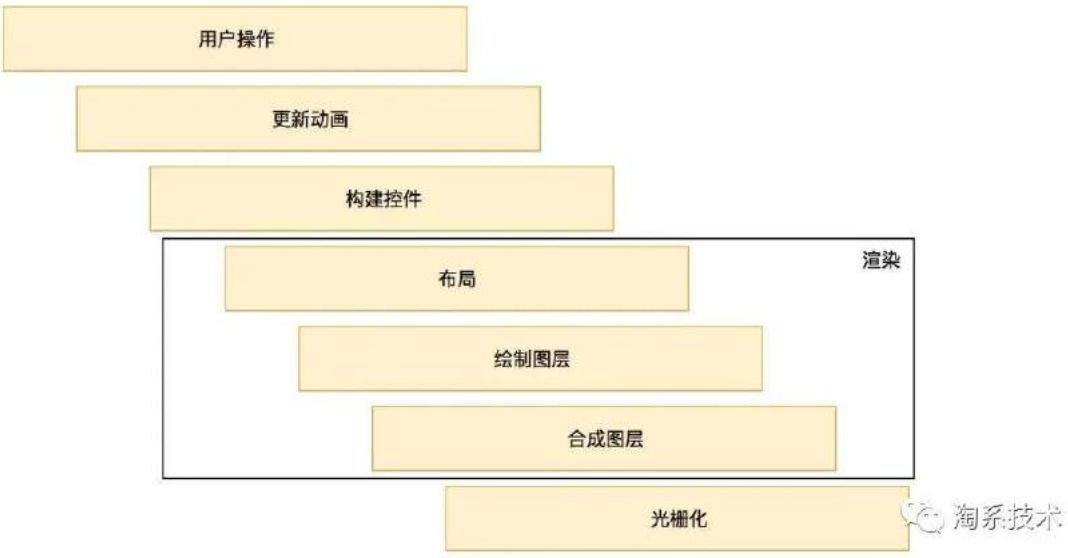
关于强大的 Dart,今天暂且不表,后面有时间可以独立篇幅来聊聊 Dart。

先祭出一张 Flutter 的架构老图。

在 Flutter 的世界里，包括 views，view controllers，layouts 等在内的概念都建立在 Widget 之上。

Widget 是 Flutter 组件的抽象描述。所以掌握 Flutter 的基础就是学会使用 Widget 开始。

在 Flutter 界面渲染过程分为三个阶段：布局、绘制、合成，布局和绘制在 Flutter 框架中完成，合成则交由引擎负责：



Flutter 通过组合、嵌套不同类型的控件，就可以构建出任意功能、任意复杂度的界面。

它包含的最主要的几个类有：

```
class WidgetsFlutterBinding extends BindingBase with GestureBinding, ServicesBinding,
SchedulerBinding,PaintingBinding, RendererBinding, WidgetsBinding { ... }

abstract class Widget extends DiagnosticableTree { ... }
abstract class StatelessWidget extends Widget { ... }
abstract class StatefulWidget extends Widget { ... }
abstract class RenderObjectWidget extends Widget { ... }
abstract class Element extends DiagnosticableTree implements BuildContext { ... }
abstract class RenderObjectElement extends Element { ... }

class StatelessElement extends ComponentElement { ... }
class StatefulElement extends ComponentElement { ... }
```

上面这些类的主要作用如下:

- 基于 Flutter 控件系统开发的程序都需要使用 WidgetsFlutterBinding，它是 Flutter 的控件框架和 Flutter 引擎的胶水层。
- Widget 就是所有控件的基类，它本身所有的属性都是只读的。
- RenderObjectWidget 所有的实现类则负责提供配置信息并创建具体的 RenderObject Element。
- Element 是 Flutter 用来分离控件树和真正的渲染对象的中间层，控件用来描述对应的 element 属性，控件重建后可能会复用同一个 element。
- RenderObjectElement 持有真正负责布局、绘制和碰撞测试（hit test）的 RenderObject 对象。
- StatelessWidget 和 StatefulWidget 并不会直接影响 RenderObject 创建，只负责创建对应的 RenderObjectWidget
- StatelessElement 和 StatefulElement 也是类似的功能。

很复杂是吧，先不用管，简单表述 Widget 是这样的:

- Widget = 样式(css) + 标记语义(标签) + 组件化(官方) + 数据绑定(props)

“什么？这不就是 react 吗？"

- 对， React 的概念和 Flutter 的 Widget 是有相通性的。

“既然有 react 的概念,难道还有 state,setState 吗？“

- 又对，Flutter 还真有 类似 state 状态机制的概念，而且也确实有 setState 的方法。

“dome 里有两个基类,StatelessWidget 和 StatefulWidget 是做什么用的？”

- StatelessWidget 和 StatefulWidget,这里两个类特别重要,几乎所有的组件都是基于他们创建的。
- StatelessWidget 是状态不可变的 widget,称为 无状态 widget。初始状态设置以后就不可再变化。如果需要变化需要重新创建。
- StatefulWidget 可以保存自己的状态,称为 有状态 widget。Flutter 首先保存了初始化时创建的 State,状态是通过改变 State,来重新构建 Widget 树来进行 UI 变化。改变状态的方法，就是我们用的最多的神器"setState"，而单纯改变数据是不会引发 UI 改变的，这个概念和我们的 React 一样一样的。

如果你是初次接触 Flutter 可以不用记忆这么多组件基类，只用记住以下式子就可以，不夸张的说，熟悉这个式子就可以开发 Flutter 项目了:

拆解

围绕着 widget 的构成，我们来拆解分析一下，标记语义，样式，组件化。

标记语义

为什么不称为“模版”,“标签”,“element”，而叫"标记语义"，是因为 flutter 的 widget 结构并不只是“模版”，“标签”，“element"。widget 描述结构更像是 React 的虚拟 dom 阶段，浓缩了相关上下文，以对象化的结构展示。

我们先来看看，React 创建出来的虚拟 dom 结构(伪代码):

```
var newTree = el('div', {'id': 'container'}, [
el('h1', {style: 'color: red'}, ['simple virtal dom']),
el('p', ['Hello, virtual-dom']),
el('ul', [el('li'), el('li')])])
```

再来看看,flutter 用 Dart 创建的 widget 代码结构:

```
Widget build(BuildContext context) {
  return new Column(
  children: <Widget>[
  new Container(
    padding: new EdgeInsets.only(top:100.0),
    child: new Text('这是一个组件')
    ),
new Container(
  decoration: new BoxDecoration(border: new Border.all(width:1.0,color: Colors.blue)),
  padding: new EdgeInsets.all(20.0),
   child: new Text('来自输入框:'+active)
          )
      ],
    );
  }
```

是不是这样看就熟悉很多了。

注:很多前端 er 会不习惯这中书写方式，目前有开发者在社区推动，在编译前，使用 jsx 标签，编译后再解析成标记树,比如这个 DSX 设计的提案。

虽然 jsx->标记树 还只是提案，但其实可以帮助我们更容易理解,此提案想表达的样式像这样:

```
class MyScaffold extends StatelessWidget {
  build(context) {
  return <Material>
<Column>
```

```
<MyAppBar
  title={<Text
  text='Example title'
  style={Theme.of(context).primaryTextTheme.title},
  />}
/>
<Expanded>
<Center>
<Text text='Hello, world!'/>
</Center>
</Expanded>
</Column>
</Material>;
  }
}
```

上面这段 jsx 要是用 Dart 来写是什么样的？如下:

```
class MyScaffold extends StatelessWidget {
    @override
  Widget build(BuildContext context) {
  return Material(
    child: Column(
    children: <Widget>[
    MyAppBar(
    title: Text(
    'Example title',
    style: Theme.of(context).primaryTextTheme.title,
), // Text
), // MyAppBar
Expanded(
  child: Center(
  child: Text('Hello, world!'),
              ), // Center
            ), // Expanded
          ], // <Widget>[]
      ), // Column
     ); // Material
            }
          }
```

虽然 Flutter 与 react 这么类比多少有些牵强，但可以个人总结一些方法，方便理解：

- 开头大写的类名 相当于 jsx 的标签名。
- child 相当于 jsx 的 子标签+布局。
- children 相当于 jsx 的 群组子标签（和 child 还是有区别的)。
- 其他属性相当于 jsx 的 props。
- 大家有没有注意，第二段 dart 语法结尾都会带上 " // " 注释符号，这个是编辑器 IDE 在识别是 Flutter 项目后，自动追加上去的，像 jsx 语言的标签封闭，方便发现标注的起始节点。

样式

对于 Flutter 样式的理解,可以查看官方的这篇文档，也是同样用类比的方式，很直观的了解，HTML、css 样式和 flutter 之间的联系。

https://flutterchina.club/web-analogs/

笔者摘选其中样例的重点部分，对比展示来说明(由于篇幅问题，父子关系 css 结构，用 tab 方式来表示):

文本样式：

```
.demo1 {
  background-color: #e0e0e0;
  width: 320px;
  height: 240px;
  font: 900 24px Georgia;
  letter-spacing: 4px;
  text-transform: uppercase;
    }
```

```
var demo1 = new Container(
  child: new Text(
    "Lorem ipsum".toUpperCase(), // 对应 左边的文本转换大小写
    style: new TextStyle( // 对应 左边的 font
      fontSize: 24.0
      fontWeight: FontWeight.w900,
      fontFamily: "Georgia",
      letterSpacing: 4.0,
        ),
    ),
    width: 320.0,  // 对应 左边的 width
    height: 240.0,  // 对应 左边的 height
    color: Colors.grey[300],  // 对应 左边的 background-color
    );
```

样式居中：

```
.demo2 {
  display: flex;
  align-items: center;
   justify-content: center;
  }
```

```
var demo2 = new Container(
  child:  new Center( // 对应左边的整个 flex 属性
  child:  new Text("Lorem ipsum")
        )
      );
    );
```

设置最大(小)宽度：

```
    .container{
    width:300px
    .demo3 {
```

```
    width: 100%;
    max-width: 240px;
      }
    }
```

```
// 对于嵌套容器，如果父级的宽度小于子级宽度，则子级容器将自行调整大小以匹配父级。
  var container = new Container(
    child: new Center(
    child: new Text("Lorem ipsum"),
    decoration: new BoxDecoration( ... ), // constraints 属性,创建一个新的 BoxConstraints 来设置 mi
nWidth 或 maxWidth
    width: 240.0, // 对应左边的 max-width
    ),
    width:300.0
    ),
```

旋转组件：

```
    .redbox {
      transform: rotate(15deg);
            }
```

```
var container = new Container( // gray box
  child:   new Transform( // 对应左边的 transform
  child:   new Container(   ... ),
    alignment: Alignment.center, // 对应左边的 transform
    transform: new Matrix4.identity() // 对应左边的 transform
     ..rotateZ(15 * 3.1415927 / 180),
       ),
      )
    );
```

缩放组件：

```
    .redbox {
transform: scale(1.5);
}
```

```
var container = new Container( // gray box
  child:   new Transform( // 对应左边的 transform
   child:   new Container(   ... ),
  alignment: Alignment.center, // 对应左边的 transform 的中心
  transform: new Matrix4.identity() // 对应左边的 transform
  ..scale(1.5),
        ),
    )
  );
```

设置绝对位置和相对位置：

```
    .greybox {
        position: relative;
        .redbox {
        position: absolute;
        top: 24px;left: 24px;
          }
        }
```

```
var container = new Container( // grey box
  child: new Stack( // 相对跟容器位置 relative
  children: [
  new Positioned( // 相对父容器位置 absolute
  child:   new Container( ... ),
  left: 24.0,
  top: 24.0,
      )],
    )
  );
```

颜色渐变：

```
.redbox {
background: linear-gradient(180deg, #ef5350, rgba(0, 0, 0, 0) 80%);
  }
```

```
var container = new Container( // grey box
  child: new Center(
  child: new Container( // red box
  child: new Text( ... ),
    decoration: new BoxDecoration(
    gradient: new LinearGradient( // 对应左边的 background: linear-gradient
    begin: const Alignment(0.0, -1.0),
    end: const Alignment(0.0, 0.6),
    colors: [
      const Color(0xffef5350),
      const Color(0x00ef5350)
                ],
                ),
              )
            ),
        )
      );
```

圆角：

```
.redbox {
border-radius: 8px;
  }
```

```
var container = new Container( // grey box
  child: new Center(
  child: new Container( // red circle
  child: new Text( ... ),
    decoration: new BoxDecoration(
     borderRadius: new BorderRadius.all(
      const Radius.circular(8.0),
            ),
        )
      ),
    )
);
```

阴影：

```
.redbox {
box-shadow: 0 2px 4px rgba(0, 0, 0, 0.8),
                0 6px 20px rgba(0, 0, 0, 0.5);}
```

```
    var container = new Container( // grey box
      child: new Center(
      child: new Container( // red box
      child: new Text( ... ),
        decoration: new BoxDecoration(
          boxShadow: [ // 对应左边的  box-shadow
        new BoxShadow (
        color: const Color(0xcc000000),
        offset: new Offset(0.0, 2.0),
        blurRadius: 4.0,
        ),
        new BoxShadow (
        color: const Color(0x80000000),
        offset: new Offset(0.0, 6.0),
          blurRadius: 20.0,
                    ),
                ],
            )
          ),

        ));
```

画圆：

```
.redcircle {
  text-align: center;
  width: 160px;
  height: 160px;
  border-radius: 50%;
  }
```

```
    var container = new Container( // grey box
      child: new Center(
      child: new Container( // red circle
      child: new Text( ... ),
        decoration: new BoxDecoration(
        color: Colors.red[400],
        shape: BoxShape.circle, // 画圆和圆角不太一样，用的是 BoxShape 绘制图像能力
      ),
            width: 160.0,
            height: 160.0,
             ),
          )
      );
```

内联样式：

```
// css 的内联结构
 .greybox {
font: 900 24px Roboto;
  .redbox {
    em {
    font: 300 48px Roboto;
      font-style: italic;
      }
       }
      }
```

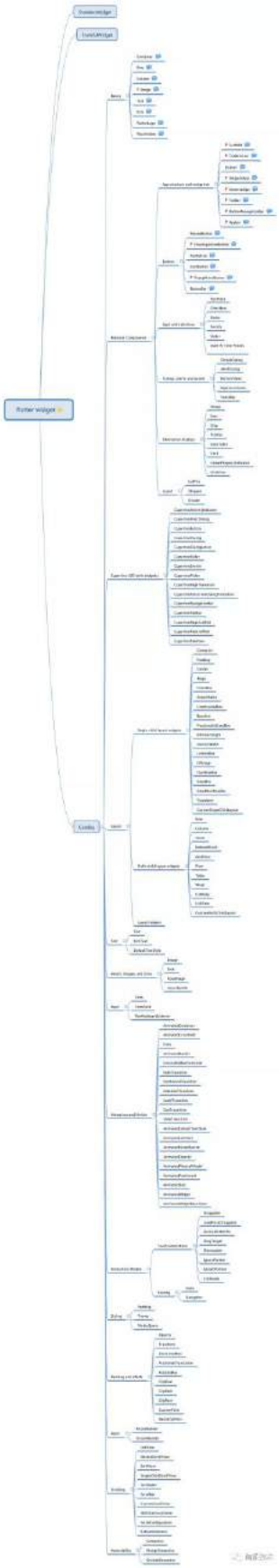
```
    var container = new Container( // grey box
      child: new Center(
      child: new Container( // red box
      child:   new RichText(
      text: new TextSpan(
      style: bold24Roboto,
      children: [
        new TextSpan(text: "Lorem "), // 继承内联样式
        new TextSpan(
        text: "ipsum",
        style: new TextStyle( // 具有自定义样式的单独样式
        fontWeight: FontWeight.w300,
```

```
            fontStyle: FontStyle.italic,
            fontSize: 48.0,
                                ),
                            ),
                        ],
                    ),
                ),
            ),
        )
    );
```

组件化

官方的 widgets 目录

点击每一个 card 后，里面还有子 card，以一个展开的纬度来看，是这样的:



这真是一个庞大的组件系统，这不是社区提供的，而是官方的。

flutter 团队事无巨细的实现了目前市面上基本上能见到的组件方式和类型。
个人认为这样做优缺点并存的。

先说缺点:

- 学习成本增加，曲线也还是比较陡的。
- 组件直接的继承关系路径比较零乱，比如
  继承自 Widget 是这些大类道还清晰:
  PreferredSizeWidget ProxyWidget RenderObjectWidget StatefulWidget StatelessWidget。
  但是，StatelessWidget 和 StatefulWidget 的子类就过于平行化了，名称上晦涩，没有抽象架构化，分层或者塔型级别。

StatelessWidget:

StatefulWidget

3. 对自定义组件定义模糊，有这么庞大的组件库，到底以后是有个一个更系统的第三方组件库去替换它，还是说 Flutter 官方就不建议使用第三方组件,这个也未有定论。

再说优点：

1. 可以阻止以后轮子泛滥，在团队僵持不下使用哪个轮子库时，最好的理由就是"官方"二字，因为使用官方，可以弱化和规避一些问题,比如: 版本迭代不同步，性能瓶颈，规范不统一等问题,也能快速支持官方的辅助工具。
2. 正是由于官方的标准划一，为自动化编译，自动化搭建，测试调优，可视化带来便利，甚至为 Ai 前端模型化业务场景，提供支撑，都说前端的组件像搭乐高，Flutter 就是颗粒标准统一的乐高。
3. 天下之势，分久必合，合久必分。

- 前端在经历了 flash 一统 pc 页面的富媒体时代，后被乔布斯和 H5 瓦解；
- 之后又有 H5 的繁盛和框架、语法、构建模式的乱战；
- 再到有"别再更新，老子学不动"的呼声下，希望有个一统江湖的跨平台系统出现；
- 让开发者更专注于，提高业务内容，创造 "新， 酷，炫" 的展现形势上下功夫。
- 也许，真有一个前端江湖的王者的诞生。

最后

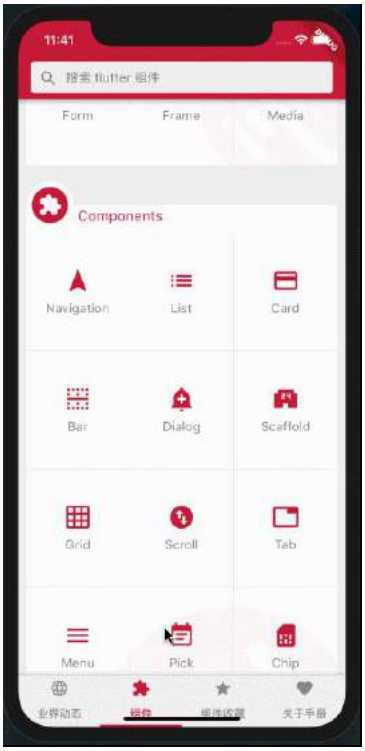
实际的 Flutter Widget 要复杂的更多的多，在眼花缭乱的 Widget 组件中,笔者想用自己的一些理解，去粗取精，来逐步理解 Flutter 这个新家伙 ,文中有理解不到位的地方,欢迎大

家指正。笔者团队也正在开发一套《Flutter 菜鸟手册》的 APP,帮助大家熟悉复杂的 Flutter Widget。

原文链接（https://github.com/ryan730/PersonalBlog/issues/1)

更多学习 Flutter 的小伙伴，欢迎入 QQ 群 Flutter Go :679476515

《Flutter GO》项目地址 [alibaba/flutter-go](https://github.com/alibaba/flutter-go) 或点击阅读原文



点击查看动图演示

对比详解 Flutter Widget 和 CSS 布局原理

作者 | 张翰(门柳)
出品 | 淘系技术

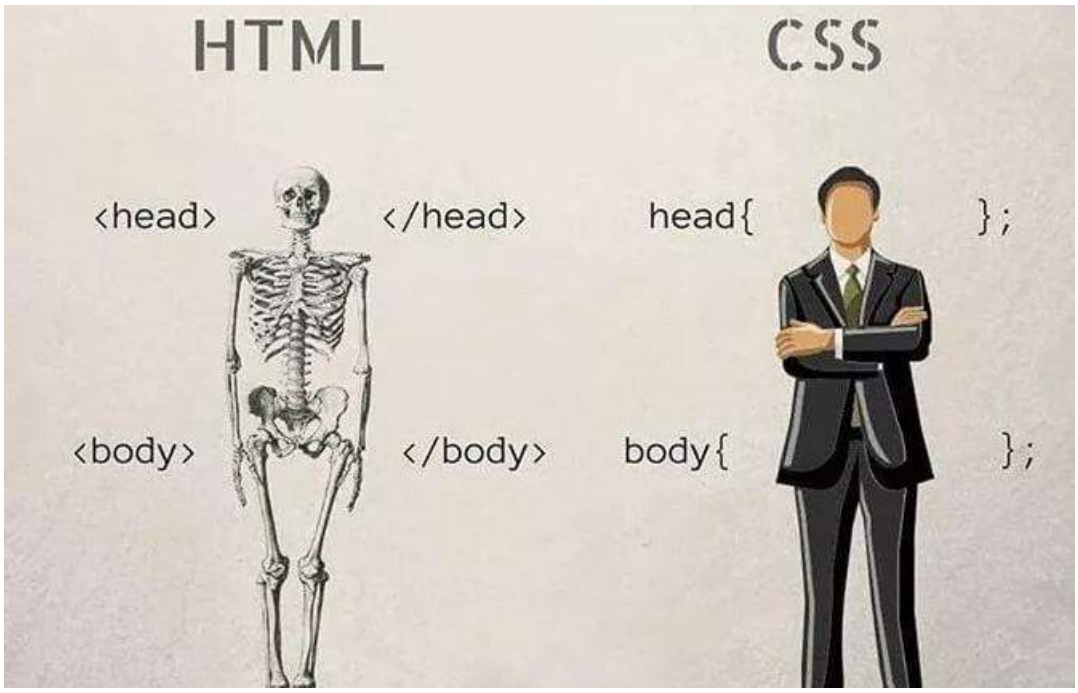
我在之前一篇文章《打破重重阻碍，Flutter 和 Web 生态如何对接？》提到了这么一句话："CSS 和 Widget 的对接也是很繁琐的过程，而且存在完备性问题"。我直接给了结论，没有给出原因。

现在填上这个坑，这篇文章专门对比 Flutter Widget 的布局原理和 CSS 布局原理的差异，分享在对接过程中会遇到的问题和解决方案，帮大家理一理思路，内容可以分为这几部分：

- CSS 和 Widget 参赛选手介绍
- Let's Battle! 从五个角度正面硬刚
- Love & Peace 讨论取长补短的可行性
- Happy Ending 最重要的环节

CSS

CSS 是 Cascading Style Sheets（层叠样式表）的简写，是一种用来描述样式的标记语言，最初的想法诞生自 1994 年，1996 年落成第一版规范（参考 20 Years of CSS）。HTML 描述了页面的结构，CSS 描述页面呈现出来的样子，这对 CP 已经配合工作了二三十年，依然是布局圈里最高效的组合。

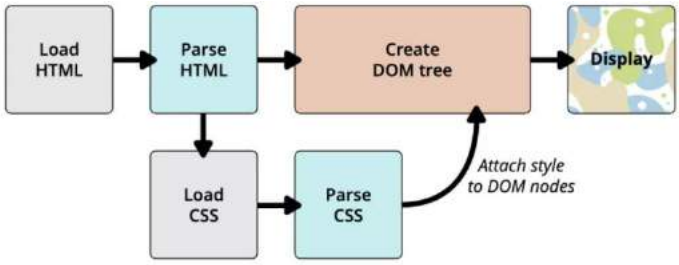


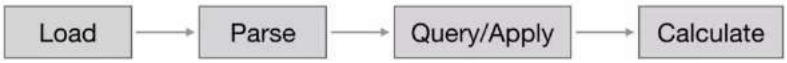
CSS is Awesome

CSS 描述布局很高效，这点无需质疑。后续出现的各种布局方案，前端框架 CSS in JS、XML 描述文件、Flutter 等等，即使没有直接照搬 CSS 的功能，也深受 CSS 设计的影响。CSS 很容易上手，经典的盒模型一学就会，文本相关的属性看名字就知道怎么回事，Flexbox 也是很好用的，但是 CSS 逐渐出现了一些很难驾驭的功能，多列布局就稍微难了一点，CSS Grid ... 这代码也太难写了吧，我觉得这是给 AI 设计的，不是让人手写的，再加上 clip-path, filter, css houdini 等等，功能越来越强大，可以做滤镜、画皮卡丘、画油画、甚至还可以做游戏。功能很强大，但是不实用，上述这些功能在生产环境基本上都是用 JS 实现的，没时间去磨 CSS。

渲染布局流程

浏览器里 CSS 的渲染过程，简化一下可以总结为 加载、解析、查询并作用到 DOM 节点、计算布局 这四个过程。浏览器首先加载 HTML 文件，遇到 <style> 标签开始加载 CSS 和解析 CSS，根据选择器的特性，计算出每个节点的样式，然后作用到 DOM 树上。



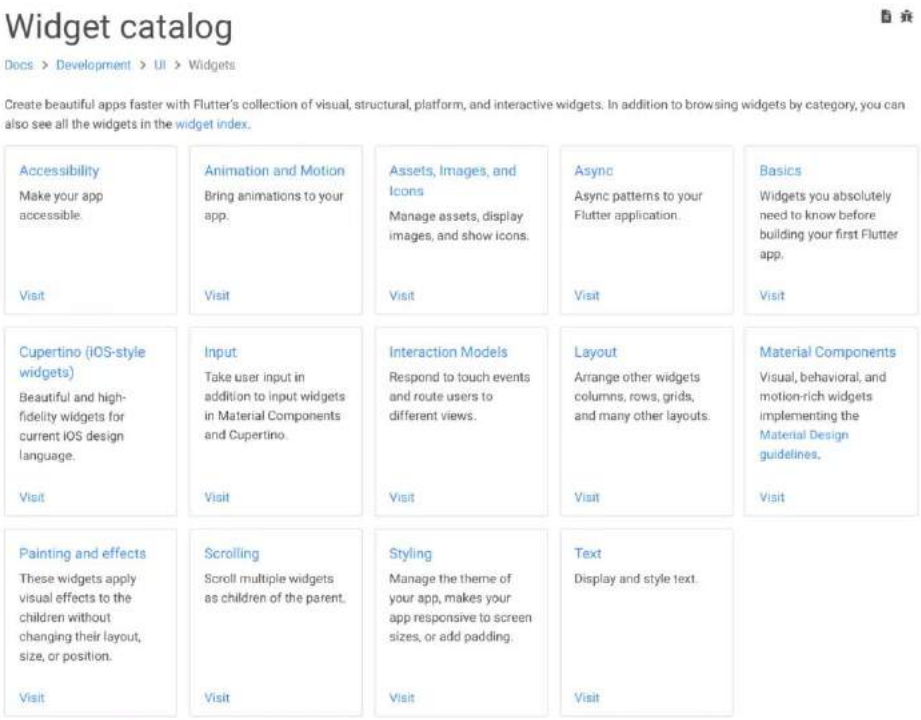


然后浏览器根据带有 ComputedStyle 的 DOM 树生成 LayoutTree，节点的 display 特性不同，生成的 LayoutObject 的类型也不同，然后布局算法会多次遍历这课树，计算出每个节点的 Rect。这个过程是很复杂的，并不是一个 DOM 节点就对应一个 Layout 节点，要考虑 display:none、伪元素、文本节点、shadow dom 等等，而且整个过程是同步的，在解析出的 DOM 节点已经布局完成后，如果浏览器又解析到一个 <style> 标签，那就又要查询匹配其中的 CSS，把新样式作用到 DOM 上，再计算一遍布局。

布局完成之后是执行 Paint，把布局信息一层一层的提交到 compositor 线程，然后再划分成一块一块的交给 GPU 线程去绘制。我觉得最后这两步是比较快的，在主线程里的 Layout 和 Paint 才是最耗时间的，而且和 JS 代码的执行搅在了一起。

Flutter Widget

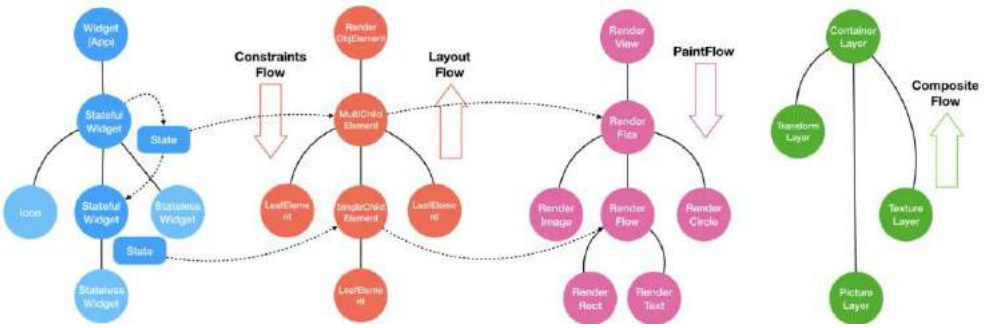
和 CSS 不同，Flutter Widget 设计得很细致，分类清晰功能明确，不像 CSS 那样各种属性耦合在一起，互相影响。Widget 的设计是比较原子化的，基本不会互相影响，为了保持布局算法的高效，对 Widget 的嵌套方式有要求。



Flutter 设计得比较合理，有一个原因不容忽视，Flutter 在 Google 的开发团队和 Chrome 有很大渊源，有些人已经参与制定 CSS 规范很多年，都是在这个领域里很有经验的人。整体上没有 CSS 那么多的冗余和历史包袱，新框架都可以吸取以前的教训，站在巨人的肩膀上设计得越来越好用。换句话说，如果让 CSS 的设计者们，抛开历史包袱重新设计，不考虑向下兼容，很有可能就设计成了现在 Flutter Widget 的样子。

渲染布局流程

关于 Flutter 的渲染流程，官方文档 Flutter 工作原理就是很好的学习资料，最大的亮点是次线性的布局，有接近 O(n) 的性能，比 CSS 高效得多。



具体过程大家去官网学习吧，这里就放一张图，本文的重点是 Battle！

Let's Battle!

Round 1: 背后的大佬

在真正开始比较之前，先看一下他们背后是什么样的组织在支撑和运营着他们。

CSS 背后是 W3C，这是业界认可的标准化组织，而且被各大浏览器实现，浏览器厂商也在积极的推到标准的发展。CSS 是个开放的技术，它背后的大佬是 W3C、Chrome、Safai、Firefox 等等一系列盈利或者非盈利组织，大家互惠互利共同发展。

但是 Flutter 背后是是只有 Google，虽然也是开源的，但是设计与实现都是由 Google 的团队来主导，其他人都是在使用，真正有能力有机会参与开发的很少，PR 都是小修小补。框架和标准的一个差别，就是会不会向下兼容，框架有可能明天宣布推出 2.0，带上牛逼的优化和 breaking change，不向下兼容，是司空见惯的事。

从这个角度看，CSS 虽然臃肿，但毕竟是个标准化的技术，生命力顽强。你现在写的 CSS 代码，五年之后依然可以运行，现在写的 Flutter 代码到五年后就不好说了。假如 Google 宣布不维护 Flutter 了，很可能社区里就瞬间丧失了信心，那 Flutter 就死了；假如 Chrome 宣布不支持 CSS 了，CSS 依然活得好好的，Chrome 很可能会死（参考 IE），FireFox 要笑醒了。

Round 2: 学习成本

要说学习成本，当然是 CSS 高效，先不讨论原理，先从几个侧面的案例说明一下。

市面上能出现"零基础，三个月成为前端高手！"的培训班，也侧面说明了前端学习成本低，其实他们的口号是错的，三个月肯定学不会前端，能学会的只有用 CSS 切页面。但是没有培训班开"三个月掌握 Flutter"的课，一个熟练的前端开发者三个月学会是可能的，客户端上

手更快一点，没有编程基础的人学完肯定一脸懵逼的要求退钱了。

另外微信可以举办"青少年微信小程序编程创意营"，面向中小学生，小程序的 UI 就是受限的 HTML + CSS 来写的，但是 Flutter 要搞这种比赛的话，就得面向有熟练编程经验的人了。

```
<template>
  <div class="greybox">
    <div class="redbox">
      Lorem <em>ipsum</em>
    </div>
  </div>
</template>

<style scoped>
  .greybox {
    background-color: #E0E0E0;
    width: 320px;
    height: 240px;
    font: 900 24px Roboto;
    display: flex;
    align-items: center;
    justify-content: center;
  }
  .redbox {
    background-color: #EF5350;
    padding: 16px;
    color: #FFFFFF;
  }
  .redbox em {
    font: 300 48px Roboto;
    font-style: italic;
  }
</style>
```

HTML + CSS

```
var container = Container( // grey box
  child: Center(
    child: Container( // red box
      child:  RichText(
        text: TextSpan(
          style: bold24Roboto,
          children: <TextSpan>[
            TextSpan(text: "Lorem "),
            TextSpan(
              text: "ipsum",
              style: TextStyle(
                fontWeight: FontWeight.w300,
                fontStyle: FontStyle.italic,
                fontSize: 48.0,
              ),
            ),
          ],
        ),
      ),
      decoration: BoxDecoration(
        backgroundColor: Colors.red[400],
      ),
      padding: EdgeInsets.all(16.0),
    ),
  ),
  width: 320.0,
  height: 240.0,
  color: Colors.grey[300],
);
```

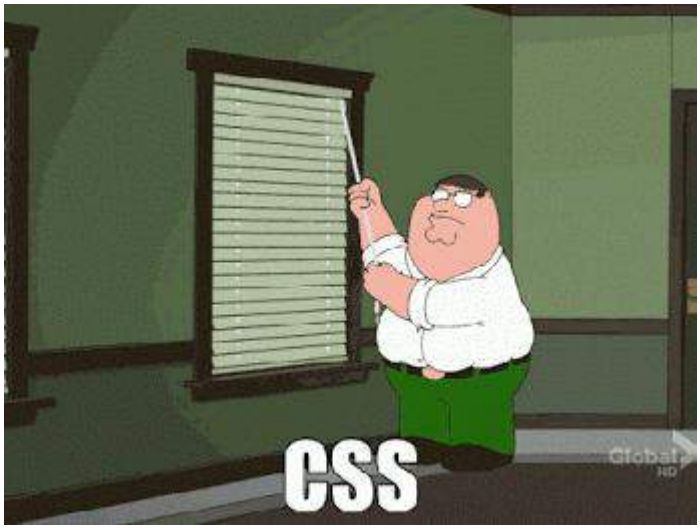
Flutter Widget

如果从语法角度考虑的话，CSS 容易学是因为它只是一种描述性语言，不含复杂的编程逻辑，设计目标就是用来描述"我想要的 UI 是什么样子"的，是面向结果的描述，而 Widget 是要通过写 Dart 代码来实现的，UI 和代码逻辑写一起，通过代码一行行描述"我怎么把 UI 组合出来"的，是面向过程的描述，所以 CSS 更直观一些，写出来的代码更容易让人理解。还有个小原因，就是像 Fluter Widget 这样层层嵌套的代码，写起来和改起来都很麻烦，太依赖编辑器，复制粘贴不太方便（差不多的话，是可以抄一下代码的嘛）。

另外 CSS 诞生了这么多年，学习资料简直多到爆！本身规范事无巨细，还有 MDN 、CSS Tricks 、CodePen 等网站即授之以渔又授之于鱼，各种线上培训也把知识框架和学习路线都给你安排的明明白白的。相比之下，Flutter Widget 就只有官网，学习资料和社区生态都还差很多。

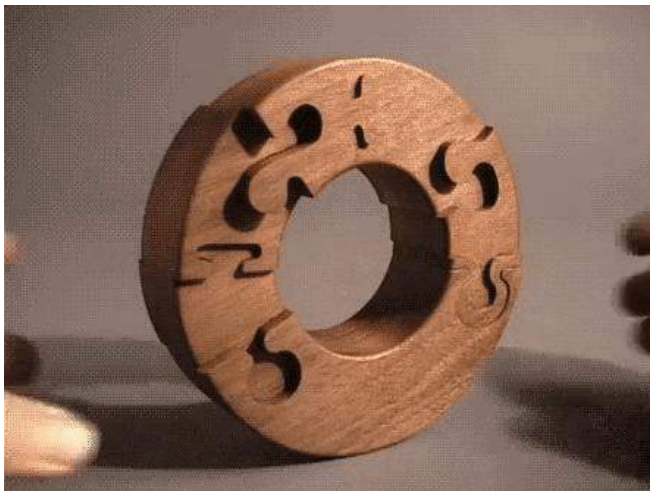
Round 3: 开发效率

CSS 上手虽然简单，但是很难掌握，没有个几年的开发经验，没被它虐过千百遍，是根本驾驭不住它的。



点击查看动图演示

假如要画出来 CSS 的学习曲线的话，初期肯定是快速上升，到了一定高度后就变慢了，甚至还迷之下降。但是 Flutter 刚开始学习时要了解很多概念，要转变思维，上手稍微慢了一点，但是越学越快。说个不恰当的比喻，写 CSS 可以看做是操作提线木偶，写 Widget 就相当于是搭乐高积木。



点击查看动图演示

因为 CSS 各种属性之间是可以互相影响的，输入和输出不是简单的对应关系，你以为你写了 width: 100px 它的宽度就一定是 100px 了？就像是绑了几百根线的木偶，让你拉动其中的五根来做一个 OK 的手势，你牵了其中一条线，动的可能不只是一个部位，而是整个上半身。

Flutter Widget 就更加原子化，而且对于谁可以嵌套谁是有要求的，就像是拼乐高积木，每块积木都很小，但是有明确型号的卡口，要先符合它的设计，然后再发挥创意拼装成各种造型。迫使你按照理想的方式去组合 Widget，避免写出性能太差的代码，CSS 就是任意属性都可以和任意属性在写在一起，而且标准里总能给出你一个合理的解释，所以写出来的代码很混乱，也增大了布局的难度。

所以 CSS 的开发效率初期较快，但是积累沉淀较难，大规模协作很难管理（耦合度高，全局作用域等），而 Flutter 的封装性更好，有利于协作，开发效率会越来越快的。

Round 4: 天下武功，唯快不破!

要问 Flutter 的 Widget 和 CSS 谁更快？大家的共识也是 Flutter 更快吧。我觉得 Flutter 和 CSS 布局相比有两大性能优势：一个是次线性的布局算法，一个是更合理的线程模型。

Widget 的布局原理比 CSS 高效，这是牺牲了一部分灵活性换来的，以后扩展 Widget 时，都要遵循这些设计才能继续保持高效。CSS 简单的属性背后可以深挖出特别复杂的细节，使得布局模型越来越复杂，渲染管线越来越长，光 display 就有十几二十个值，每一条都对布局影响巨大，方便了开发者，但是给布局性能带来很大挑战。

关于线程模型，也是浏览器一直被诟病的性能瓶颈，主线程太忙了，JS 的执行、HTML/CSS 的解析、DOM 的构建，布局的运算，全都在主线程。相比之下 Flutter 划分的四个线程就比较均衡，GPU 线程做的工作和浏览器差不多，但是宿主平台（Android/iOS）的代码跑在 Platform 线程里，Flutter Framework 主要运行在 UI 线程里，另外还有 IO 线程实现网络和图片、字体等文件的加载。

Round 5: 未来的发展

分享几个关于 CSS 现状的数据，W3C 官方定义的 CSS 样式有 520 条，Chrome 平台上统计的样式有 703 条，包括了一些带前缀的样式，有大量样式的使用率很低。支付宝小程序的同学总结过 Top 100 小程序里用到的不同 CSS 样式，有 184 条。

总结一下就是：W3C 标准里定义了 520 条样式，Chrome 还额外支持 180+ 条私货，但是常用的样式不超过 200 条。

CSS 有大量历史包袱，我自己也感觉大部分样式我都用不到，有些甚至是最佳实践里禁止使用的，学会 50 条 CSS 后就可以写 80% 种情况的布局了，对于难写的样式，我多加几层标签再写点 JS 也能实现。但是这些样式不能废弃，必须继续支持，新增一条好用的属性时，要解释清楚和现在所有属性的适配情况。

CSS 是负重前行，注定要越变越复杂。Widget 则是轻装上阵，而且解耦比较好，是可插拔可组合的，如果未来想废弃一些 Widget，把这部分 Widget 从主包里拆出来放到独立插件里就行了，想用的话自行引入。整个迭代过程受历史包袱的影响很小。

Love & Peace: 可否实现对接?

Battle 已经结束，友谊第一比赛第二，不讨论胜负，下面进入 Love & Peace 环节。

Flutter 的压线性布局让人眼馋， CSS 的灵活度又深受大家喜爱，可不可以取其精华去其糟粕，让开发者写 CSS 但是底层用 Flutter 来渲染？有这种想法的不是一个人，所以有很多方案把前端框架或小程序对接到 Flutter 上，在实现的时候，就会遇到如何把 CSS 转换为 Widget 的问题。

技术可行性

技术上当然是可行的，我在前一篇文章《打破重重阻碍，Flutter 和 Web 生态如何对接？》里介绍了各种实现方案，自己也写代码做过对接（用的 C++魔改方案），跑通了整个渲染链路。

我介绍一下我的实现方式，可以简单分成三步：

1. 解析 CSS 语法

我写了个精简版的 CSSOM，是标准 CSSOM 的子集。用来实现样式表和样式属性和增删改查、样式值的解析与计算、选择器的匹配和查询等功能。（功能是独立的，有需要的话自取）

CSSOM 主要是为了处理 CSS 的上层语法，转成一致的数据格式，为下一步的转换做准备，有一部分 CSS 的语法是在这个过程实现的，例如 CSS 选择器，包括伪类选择器和选择器关系等，还有 @media 和 @keyframes 等功能，也可以实现 CSS variable 以及 calc()。无论上层是直接写 CSS 还是 CSS in JS，都保证下一步转换时输入的数据格式一致，可以简化后续的实现。

2. 实现 CSS 属性和 Widget 数据格式的映射

这部分要把 CSS 的基础数据格式转成构建 Flutter Widget 所依赖的数据格式。例如 CSS 的 color 属性会转成 Flutter 的 Color 类，margin 和 padding 会转成 EdgeInsets 类，flex-direction 将被转成 Axis 枚举，把文本相关的属性转换成 TextStyle 类。

这部分也是做原子性的转换，技术上看起来比较简单，只是转换数据格式而已，但是需要对 CSS 和 Widget 的设计都比较了解，知道同一个概念在双方语义中的对应关系，得搞清楚里边的技术细节。这个对应关系如果转换错了，后续的布局怎么调都调不对（过来人的经验...）

3. 构建 Widget 树

拿到 CSSOM 传来的数据，掌握了 CSS 和 Widget 数据结构的语义转换，然后在结合 HTML 定义的结构，就可以生成真正的 Widget 树了。

构建 Widget 树就不能只考虑 CSS 了， HTML + CSS 才和 Widget 对等。这里还要处理不同类型 HTML 节点和 Widegt 的对应关系，节点上带的布局样式不同，生成的节点也不同，Widget 的层次深度比 HTML 的要深。例如普通的 div 标签，如果仅包含普通盒模型样式，就转换成 Container；如果包含了 flex 相关的属性，根据具体配置的不同，转成 Flex/Center/Row/Column 等；如果包含了绝对定位就要转成 Positioned/Stack。这个过程也是很繁琐，包含了大量细节，需要理解默认 HTML 标签的语义、CSS 的层模型以及 BFC 等等，还需要理解 Flutter Widget 之间的嵌套限制，组合成 CSS 想要的效果。

使用限制

对于开头提到的问题，前面的技术可行性分析回应了「繁琐」，下面讨论一下「完备性」。

CSS 是灵活的，Widget 是受限的，把一个灵活的语法转换成受限的实现，注定是不完备的。

在 CSS 里，任意属性可以和任意属性写在一起，W3C 标准里总有一个明确的解释方式，HTML 和 CSS 是没有错误的，只有不符合预期。但是在 Flutter 里，某个 Widget 里可以放哪些 Widget 是明确的限制的，例如 Positioned 外层必须有个 Stack、Center 只能有一个子 Widget，不符合预期的嵌套是会报错的，写出的代码不会出现匪夷所思的混用，所以布局算法可以很快。从这个角度讲，CSS 是 O(n!) 的复杂度，而 Widget 是多项式复杂度，用 Widget 去实现 CSS 注定是不完备的。（这难道是个 P 和 NP 问题……？）

在限制 CSS 写法的情况下，能不能对接到 Widget 的实现？这个是可以的。

想要 Widget 次线性布局的性能，就必须牺牲 CSS 的一部分灵活性。想要用技术突破这个限制？那就要改这套次线性的布局算法了，改完之后就不再是 Flutter 了，性能优势也没有了。

以我的实践经验来看，Widget 只能支持一定范围内的 CSS 样式。它对 CSS 的使用限制不在于样式条数，并不是说某个样式实现不了，而是在于样式的混用，即使支持了 500 条 CSS，但是某些属性依然不能同时使用，外层用了样式 A 内层再用样式 B 就是无效的，C 和 D 写一起就只有 C 有效。我评估 Widget 对 CSS 支持的范围上限会比现在的 ReactNative/Weex 还要大一些，能够满足大部分业务和小程序的需求，然而业务需求是会增长的，达到支持范围上限以后就很难再扩大了，就需要教育开发者了，对开发体验有影响。

参考：

https://www.w3.org/Style/CSS20/
https://una.im/CSSgram/
https://projects.lukehaas.me/css-clip-path-pokemon/
https://css-art.com/pure-css-lace/
https://codepen.io/finnhvman/pen/xJRMJp
https://flutter.cn/docs/resources/inside-flutter
https://github.com/flutter/flutter/wiki/The-Engine-architecture#threading
https://www.w3.org/Style/CSS/all-properties.html
https://www.chromestatus.com/metrics/css/popularity

Flutter 渲染引擎：底层原理解析

作者 | 万红波（远湖）
出品 | 淘系技术

前言

Flutter 作为一个跨平台的应用框架，诞生之后，就被高度关注。它通过自绘 UI ，解决了之前 RN 和 weex 方案难以解决的多端一致性问题。Dart AOT 和精减的渲染管线，相对与 JavaScript 和 webview 的组合，具备更高的性能体验。

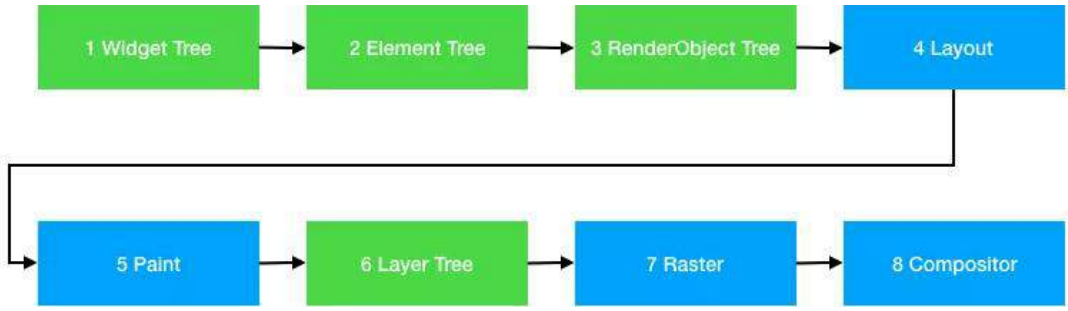
目前在集团内也有很多的 BU 在使用和探索。了解底层引擎的工作原理可以帮助我们更深入地结合具体的业务来对引擎进行定制和优化，更好的去创新和支撑业务。在淘宝，我们也基于 Flutter engine 进行了自绘 UI 的渲染引擎的探索。本文先对 Flutter 的底层渲染引擎做一下深入分析和整理，以理清 Flutter 的渲染的机制及思路，之后分享一下我们基于 Flutter 引擎一些探索，供大家参考。

本文的分析主要以 Android 平台为例，IOS 上原理大致类似，相关的参考代码基于 stable /v1.12.13+hotfix.8 。

渲染引擎分析

渲染流水线

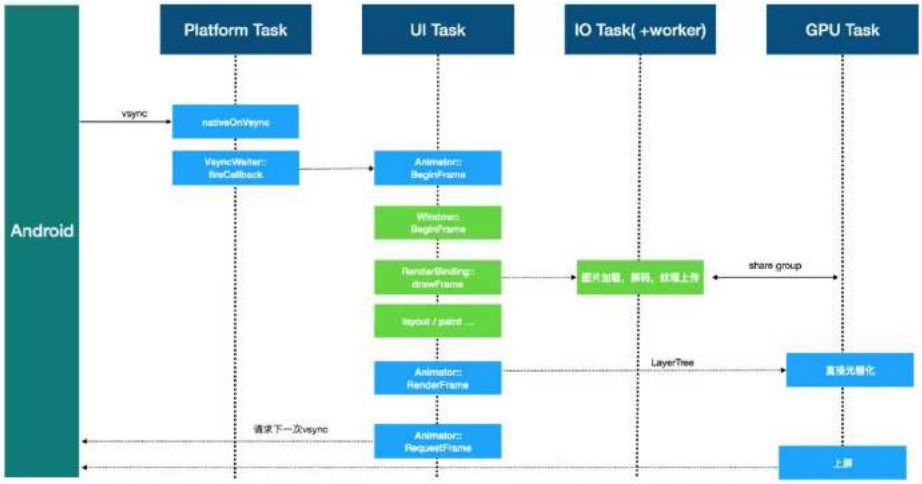
整个 Flutter 的 UI 生成以及渲染完成主要分下面几个步骤：



其中 1-6 在收到系统 vsync 信号后,在 UI 线程中执行，主要是涉及在 Dart framework 中 Widget/Element/RenderObject 三颗树的生成以及承载绘制指令的 LayerTree 的创建，7-8 在 GPU 线程中执行，主要涉及光栅化合成上屏。

- 1-4 跟渲染没有直接关系，主要就是管理 UI 组件生命周期，页面结构以及 Flex layout 等相关实现，本文不作深入分析。
- 5-8 为渲染相关流程，其中 5-6 在 UI 线程中执行，产物为包含了渲染指令的 Layer tree，在 Dart 层生成，可以认为是整个渲染流程的前半部，属于生产者角色。
- 7-8 把 dart 层生成的 Layer Tree，通过 window 透传到 Flutter engine 的 C++代码中，通过 flow 模块来实现光栅化并合成输出。可以认为是整个渲染流程的后半部，属于消费者角色。

下图为 Android 平台上渲染一帧 Flutter UI 的运行时序图：



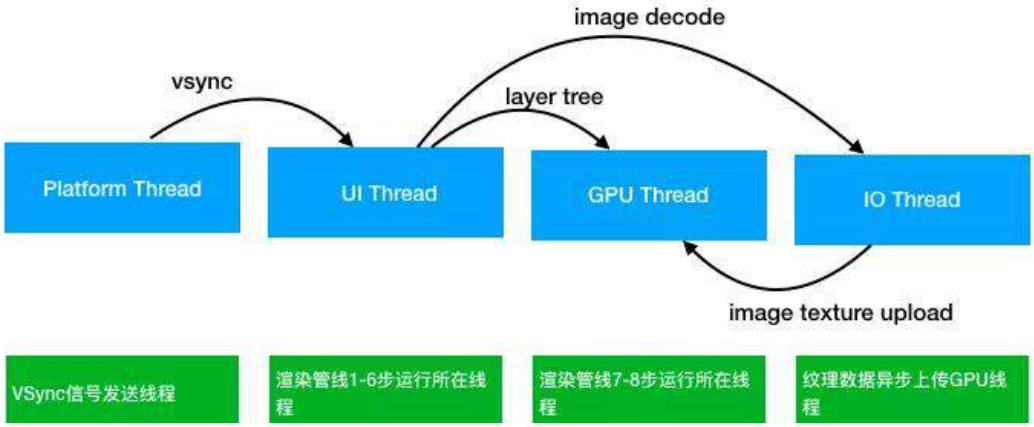
具体的运行时步骤：

1. flutter 引擎启动时，向系统的 Choreographer 实例注册接收 Vsync 的回调。
2. 平台发出 Vsync 信号后，上一步注册的回调被调用，一系列调用后，执行到 VsyncWaiter::fireCallback。
3. VsyncWaiter::fireCallback 实际上会执行 Animator 类的成员函数 BeginFrame。
4. BeginFrame 经过一系列调用执行到 Window 的 BeginFrame，Window 实例是连接底层 Engine 和 Dart framework 的重要桥梁，基本上所以跟平台相关的操作都会由 Window 实例来串联，包括事件，渲染，无障碍等。
5. 通过 Window 的 BeginFrame 调用到 Dart Framework 的 RenderBinding 类，其有一个方法叫 drawFrame ，这个方法会去驱动 UI 上的 dirty 节点进行重排和绘制，如果遇到图片的显示，会丢到 IO 线程以及去 worker 线程去执行图片加载和解码，解码完成后，再次丢到 IO 线程去生成图片纹理，由于 IO 线程和 GPU 线程是 share GL context 的，所以在 IO 线程生成的图片纹理在 GPU 线程可以直接被 GPU 所处理和显示。
6. Dart 层绘制所产生的绘制指令以及相关的渲染属性配置都会存储在 LayerTree 中，通过 Animator::RenderFrame 把 LayerTree 提交到 GPU 线程，GPU 线程拿到 LayerTree 后，进行光栅化并做上屏操作（关于 LayerTree 我们后面会详细讲解）。之后通过 Animator::RequestFrame 请求接收系统下一次的 Vsync 信号，这样又会从第 1 步开始，循环往复，驱动 UI 界面不断的更新。

分析了整个 Flutter 底层引擎总体运行流程，下面会相对详细的分析上述渲染流水线中涉及到的相关概念以及细节知识，大家可以根据自己的情况选择性的阅读。

线程模型

要了解 Flutter 的渲染管线，必须要先了解 Flutter 的线程模型。从渲染引擎的视角来看，Flutter 的四个线程的职责如下：



- Platform 线程：负责提供 Native 窗口，作为 GPU 渲染的目标。接受平台的 VSync 信号并发送到 UI 线程，驱动渲染管线运行。
- UI 线程：负责 UI 组件管理，维护 3 颗树，Dart VM 管理，UI 渲染指令生成。同时负责把承载渲染指令的 LayerTree 提交给 GPU 线程去光栅化。
- GPU 线程：通过 flow 模块完成光栅化，并调用底层渲染 API（opengl/vulkan/meta），合成并输出到屏幕。
- IO 线程：包括若干 worker 线程会去请求图片资源并完成图片解码，之后在 IO 线程中生成纹理并上传 GPU ，由于通过和 GPU 线程共享 EGL Context，在 GPU 线程中可以直接使用 IO 线程上传的纹理，通过并行化，提高渲染的性能

后面介绍的概念都会贯穿在这四个线程当中，关于线程模型的更多信息可以参考下面两篇文章:《深入了解 Flutter 引擎线程模型》《The Engine architecture》

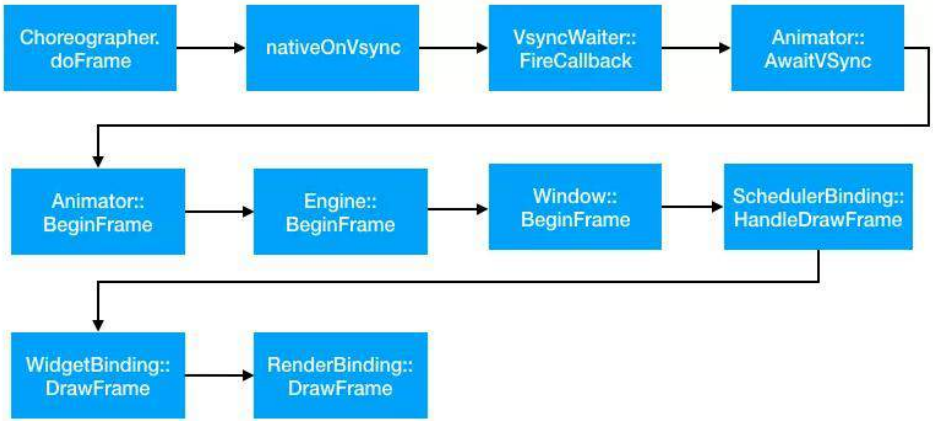
VSync

Flutter 引擎启动时，向系统的 Choreographer 实例注册接收 Vsync 的回调函数，GPU 硬件发出 Vsync 后，系统会触发该回调函数，并驱动 UI 线程进行 layout 和绘制。

```
    @ shell/platform/android/io/flutter/view/VsyncWaiter.java
    private final FlutterJNI.AsyncWaitForVsyncDelegate asyncWaitForVsyncDelegate = new
FlutterJNI.AsyncWaitForVsyncDelegate() {
     @Override
    public void asyncWaitForVsync(long cookie) {
    Choreographer.getInstance().postFrameCallback(new Choreographer.FrameCallback() {
```

```
    @Override
    public void doFrame(long frameTimeNanos) {
    float fps = windowManager.getDefaultDisplay().getRefreshRate();
    long refreshPeriodNanos = (long) (1000000000.0 / fps);
    FlutterJNI.nativeOnVsync(frameTimeNanos, frameTimeNanos + refreshPeriodNanos, cookie);
          }
        });
      }
    };
```

下图为 Vsync 触发时的调用栈：



在 Android 上，Java 层收到系统的 Vsync 的回调后通过 JNI 发给 Flutter engine，之后通过 Animator,Engine 以及 Window 等对象路由调回 dart 层，驱动 dart 层进行 drawFrame 的操作。在 Dart framework 的 RenderingBinding::drawFrame 函数中会触发对所有 dirty 节点的 layout/paint/compositor 相关的操作,之后生成 LayerTree，再交由 Flutter engine 光栅化并合成。

```
    //@rendering/binding.dart
      void drawFrame() {
        assert(renderView != null);
        pipelineOwner.flushLayout();
          pipelineOwner.flushCompositingBits();
          pipelineOwner.flushPaint();
        renderView.compositeFrame(); // this sends the bits to the GPU
        pipelineOwner.flushSemantics(); // this also sends the semantics to the OS.
        }
```

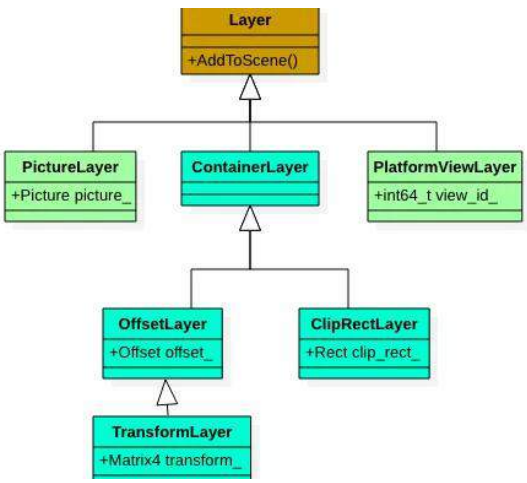
图层

在 Dart 层进行 drawFrame 对 dirty 节点进行排版后，就会对需要重新绘制的节点进行绘制操作。而我们知道 Flutter 中 widget 是一个 UI 元素的抽象描述，绘制时，需要先将其 inflate 成为 Element，之后生成对应的 RenderObject 来负责驱动渲染。通常来讲，一个页面的所有的 RenderObject 都属于一个图层，Flutter 本身没有图层的概念，这里所说的图层可以粗暴理解成一块内存 buffer，所有属于图层的 RenderObject 都应该被绘制在这个图层对应的 buffer 中去。

如果这个 RenderObject 的 RepaintBoundary 属性为 true 时，就会额外生成一个图层，其所有的子节点都会被绘制在这个新的图层上，最后所有图层有 GPU 来负责合成并上屏。

Flutter 中使用 Layer 的概念来表示一个层次上的所有 RenderObject，Layer 和图层存在 N:1 的对应关系。根节点 RenderView 会创建 root Layer，一般是一个 Transform Layer，并包含多个子 Layer，每个子 Layer 又会包含若干 RenderObject，每个 RenderObject 绘制时，会产生相关的绘制指令和绘制参数，并存储在对应的 Layer 上。

可以参考下面 Layer 的类图，Layer 实际上主要用来组织和存储渲染相关的指令和参数，比如 Transform Layer 用来保存图层变换的矩阵，ClipRectLayer 包含图层的剪切域大小，PlatformViewLayer 包含同层渲染组件的纹理 id，PictureLayer 包含 SkPicture(SkPicture 记录了 SkCanvas 绘制的指令，在 GPU 线程的光栅化过程中会用它来做光栅化)



渲染指令

当渲染第一帧的时候，会从根节点 RenderView 开始，逐个遍历所有的子节点进行绘制操作。

```
//@rendering/view.dart
//绘制入口，从 view 根节点开始，逐个绘制所有子节点
  @override
  void paint(PaintingContext context, Offset offset) {
  if (child != null)
   context.paintChild(child, offset);
  }
```

我们可以具体看看一个节点如何绘制的：

1. 创建 Canvas

绘制时会通过 PaintContex 获取的 Canvas 进行，其内部会去创建一个 PictureLayer，并通过 ui.PictrureRecorder 调用到 C++层来创建一个 Skia 的 SkPictureRecorder 实例，再通过 SkPictureRecorder 创建 SkCanvas，最后把这个 SkCanvas 返回给 Dart 层去使用。

```
//@rendering/object.dart
  @override
  Canvas get canvas {
    if (_canvas == null)
    _startRecording();
    return _canvas;
     }

     void _startRecording() {
    assert(!_isRecording);
    _currentLayer = PictureLayer(estimatedBounds);
    _recorder = ui.PictureRecorder();
    _canvas = Canvas(_recorder);
     _containerLayer.append(_currentLayer);
    }
```

2. 通过 Canvas 执行具体绘制

Dart 层拿到绑定了底层 SkCanvas 的对象后，用这个 Canvas 进行具体的绘制操作，这些绘制命令会被底层的 SkPictureRecorder 记录下来。

3. 结束绘制，准备上屏

绘制完毕时，会调用 Canvas 对象的 stopRecordingIfNeeded 函数，它会最后会去调用到 C++的 SkPictureRecorder 的 endRecording 接口来生成一个 Picture 对象，存储在 Picture Layer 中。

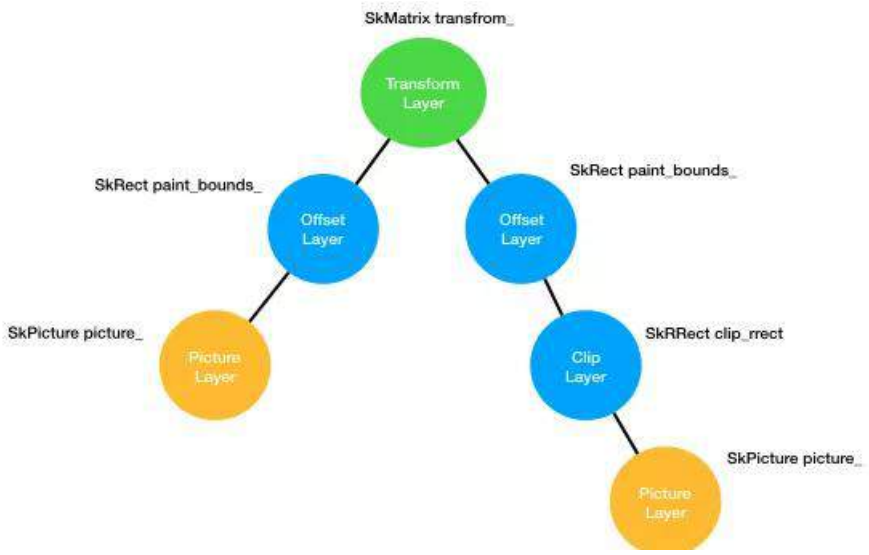
```
//@rendering/object.dart
  void stopRecordingIfNeeded() {
    if (!_isRecording)
    return;
     _currentLayer.picture = _recorder.endRecording();
    _currentLayer = null;
     _recorder = null;
    _canvas = null;
      }
```

这个 Picture 对象对应 Skia 的 SkPicture 对象，存储这所有的绘制指令。有兴趣可以看一下 SkPicture 的官方说明。

所有的 Layer 绘制完成形成 LayerTree，在 renderView.compositeFrame()中通过 SceneBuilder 把 Dart Layer 映射为 flutter engine 中的 flow::Layer,同时也会生成一颗 C++的 flow::LayerTree，存储在 Scene 对象中，最后通过 Window 的 render 接口提交给 Flutter engine。

```
//@rendering/view.dart
  void compositeFrame() {
    ...
    final ui.SceneBuilder builder = ui.SceneBuilder();
  final ui.Scene scene = layer.buildScene(builder);
  _window.render(scene);
  scene.dispose();
  }
```

在全部绘制操作完成后，在 Flutter engine 中就形成了一颗 flow::LayerTree，应该是像下面的样子:



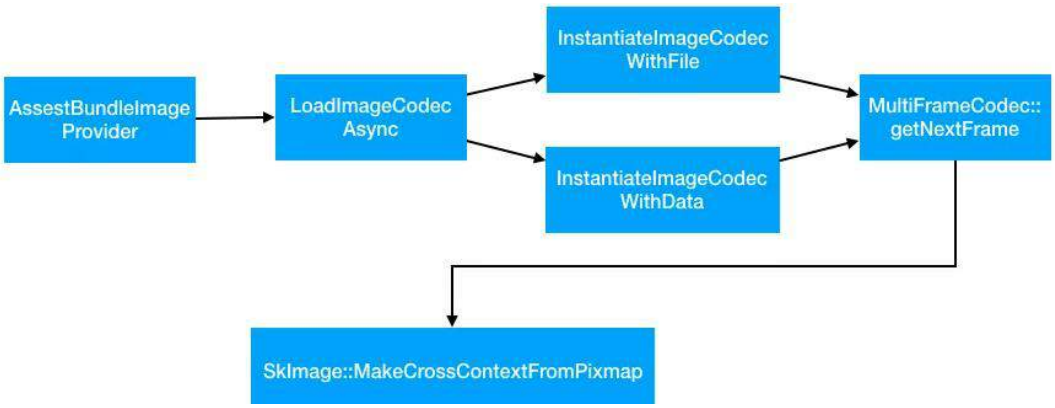
这颗包含了所有绘制信息以及绘制指令的 flow::LayerTree 会通过 window 实例调用到 Animator::Render 后，最后在 Shell::OnAnimatorDraw 中提交给 GPU 线程，并进行光栅化操作,代码可以参考:

@shell/common/animator.cc/Animator::Render
@shell/common/shell.cc/Shell::OnAnimatorDraw

这里提一下 flow 这个模块，flow 是一个基于 skia 的合成器，它可以基于渲染指令来生成像素数据。Flutter 基于 flow 模块来操作 Skia，进行光栅化以及合成。

图片纹理

前面讲线程模型的时候，我们提到过 IO 线程负责图片加载以及解码并且把解码后的数据上传到 GPU 生成纹理，这个纹理在后面光栅化过程中会用到，我们来看一下这部分的内容。



UI 线程加载图片的时候，会在 IO 线程调用 InstantiateImageCodec*函数调用到 C++层来初始化图片解码库，通过 skia 的自带的解码库解码生成 bitmap 数据后，调用 SkImage::MakeCrossContextFromPixmap 来生成可以在多个线程共享的 SkImage，在 IO 线程中用它来生成 GPU 纹理。

```
//@flutter/lib/ui/painting/codec.cc
sk_sp<SkImage> MultiFrameCodec::GetNextFrameImage(
fml::WeakPtr<GrContext> resourceContext) {
...
//  如果 resourceContext 不为空，就会去创建一个 SkImage，
//  并且这个 SkImage 是在 resouceContext 中的，
  if (resourceContext) {
    SkPixmap pixmap(bitmap.info(), bitmap.pixelRef()->pixels(),
    bitmap.pixelRef()->rowBytes());
    // This indicates that we do not want a "linear blending" decode.
    sk_sp<SkColorSpace> dstColorSpace = nullptr;
    return SkImage::MakeCrossContextFromPixmap(resourceContext.get(), pixmap,
    false, dstColorSpace.get());

    } else {

    // Defer decoding until time of draw later on the GPU thread. Can happen
    // when GL operations are currently forbidden such as in the background
    // on iOS.       return SkImage::MakeFromBitmap(bitmap);
     }
    }
```

我们知道，OpenGL 的环境是线程不安全的，在一个线程生成的图片纹理，在另外一个线程里面是不能直接使用的。但由于上传纹理操作比较耗时，都放在 GPU 线程操作，会减低渲染性能。目前 OpenGL 中可以通过 share context 来支持这种多线程纹理上传的，所以目前 flutter 中是由 IO 线程做纹理上传，GPU 线程负责使用纹理。

基本的操作就是在 GPU 线程创建一个 EGLContextA，之后把 EGLContextA 传给 IO 线程，IO 线程在通过 EGLCreateContext 在创建 EGLContextB 的时候，把 EGLContextA 作为 shareContext 的参数，这样 EGLContextA 和 EGLContextB 就可以共享纹理数据了。

具体相关的代码不一一列举了，可以参考：

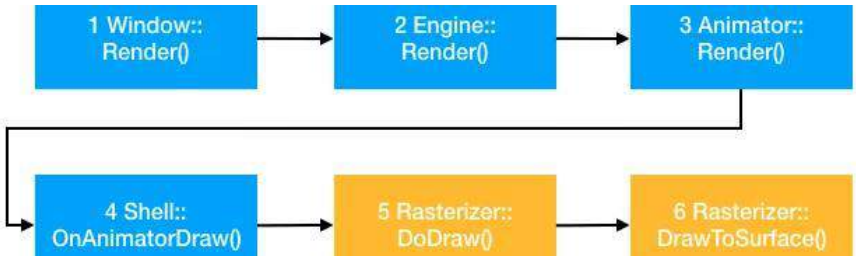
@shell/platform/android/platform_view_android.cc/CreateResourceContext
@shell/platform/android/android_surface_gl.cc/ResourceContextMakeCurrent
@shell/platform/android/android_surface_gl.cc/AndroidSurfaceGL
@shell/platform/android/android_surface_gl.cc/SetNativeWindow

关于图片加载相关流程，可以参考这篇文章：TODO

光栅化与合成

把绘制指令转化为像素数据的过程称为光栅化，把各图层光栅化后的数据进行相关的叠加与特效相关的处理成为合成这是渲染后半段的主要工作。

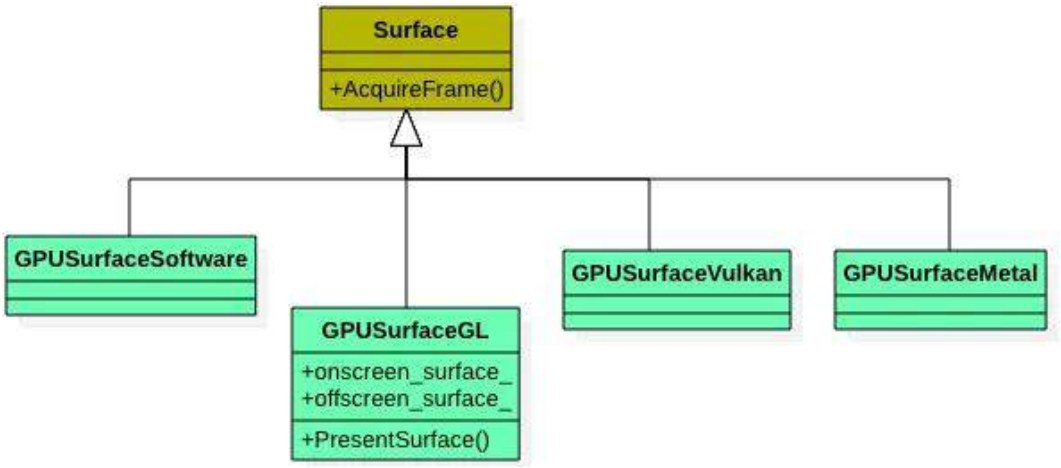
前面也提到过，生成 LayerTree 后，会通过 Window 的 Render 接口把它提交到 GPU 线程去执行光栅化操作，大体流程如下：



1-4 步，在 UI 线程执行，主要是通过 Animator 类把 LayerTree 提交到 Pipeline 对象的渲染队列，之后通过 Shell 把 pipeline 对象提交给 GPU 线程进行光栅化，不具体展开，代码在 animator.cc&pipeline.h

5-6 步，在 GPU 线程执行具体的光栅化操作。这部分主要分为两大块，一块是 Surface 的管理。一块是如何把 Layer Tree 里面的渲染指令绘制到之前创建的 Surface 中。

可以通过下图了解一下 Flutter 中的 Surface，不同类型的 Surface，对应不同的底层渲染 API。



我们以 GPUSurfaceGL 为例，在 Flutter 中，GPUSurfaceGL 是对 Skia GrContext 的一个管理和封装，而 GrContext 是 Skia 用来管理 GPU 绘制的一个上下文，最终都是借助它来操作 OpenGL 的 API 进行相关的上屏操作。在引擎初始化时，当 FlutterViewAndroid 创建后，就会创建 GPUSurfaceGL，在其构造函数中会同步创建 Skia 的 GrContext。

光栅化主要是在函数 Rasterizer::DrawToSurface 中实现的：

```
//@shell/rasterizer.cc
RasterStatus Rasterizer::DrawToSurface(flutter::LayerTree& layer_tree) {
FML_DCHECK(surface_);
...
    if (compositor_frame) {
  //1.执行光栅化
  RasterStatus raster_status = compositor_frame->Raster(layer_tree, false);
  if (raster_status == RasterStatus::kFailed) {
  return raster_status;
  }

  //2.合成
  frame->Submit();
  if (external_view_embedder != nullptr) {
  external_view_embedder->SubmitFrame(surface_->GetContext());
   }

  //3.上屏
  FireNextFrameCallbackIfPresent();

   if (surface_->GetContext())
  {
  surface_->GetContext()->performDeferredCleanup(kSkiaCleanupExpiration);
   }

   return raster_status;
  }

  return RasterStatus::kFailed;
  }
```

光栅化完成后，执行 frame->Submit()进行合成。这会调用到下面的 PresentSurface，来把 offscreen_surface 中的内容转移到 onscreen_canvas 中，最后通过 GLContextPresent()上屏。

```
//@shell/GPU/gpu_surface_gl.cc
bool GPUSurfaceGL::PresentSurface(SkCanvas* canvas) {
...
  if (offscreen_surface_ != nullptr) {
     SkPaint paint;
     SkCanvas* onscreen_canvas = onscreen_surface_->getCanvas();
     onscreen_canvas->clear(SK_ColorTRANSPARENT);
```

```
        // 1.转移 offscreen surface 的内容到 onscreen canvas 中
        onscreen_canvas->drawImage(offscreen_surface_->makeImageSnapshot(), 0, 0,
&paint);
         }
        {
         //2. flush  所有绘制命令
        onscreen_surface_->getCanvas()->flush();
           }
        //3  上屏
        if (!delegate_->GLContextPresent()) {
        return false;
        }
        ...
        return true;
           }
```

GLContextPresent 接口代码如下，实际上是调用的 EGL 的 eglSwapBuffers 接口去显示图形缓冲区的内容。

```
//@shell/platform/android/android_surface_gl.cc
bool AndroidSurfaceGL::GLContextPresent() {
  FML_DCHECK(onscreen_context_ && onscreen_context_->IsValid());
return onscreen_context_->SwapBuffers();
}
```

上面代码段中的 onscreen_context 是 Flutter 引擎初始化的时候，通过 setNativeWindow 获得。主要是把一个 Android 的 SurfaceView 组件对应的 ANativeWindow 指针传给 EGL，EGL 根据这个窗口，调用 eglCreateWindowSurface 和显示系统建立关联，之后通过这个窗口把渲染内容显示到屏幕上。

代码可以参考：

@shell/platform/android/android_surface_gl.cc/AndroidSurfaceGL::SetNativeWindow

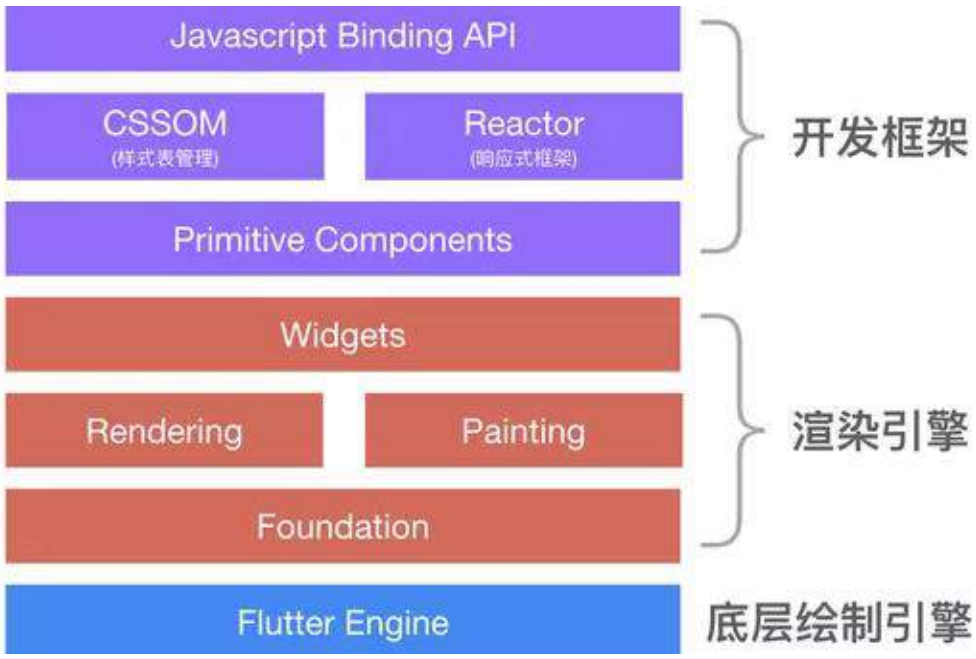
总结以上渲染后半段流程，就可以看到 LayerTree 中的渲染指令被光栅化，并绘制到 SkSurface 对应的 Surface 中。这个 Surface 是由 AndroidSurfaceGL 创建的一个

offscreen_surface。再通过 PresentSurface 操作，把 offscreen_surface 的内容，交换到 onscreen_surface 中去，之后调用 eglSwapSurfaces 上屏，结束一帧的渲染。

探索

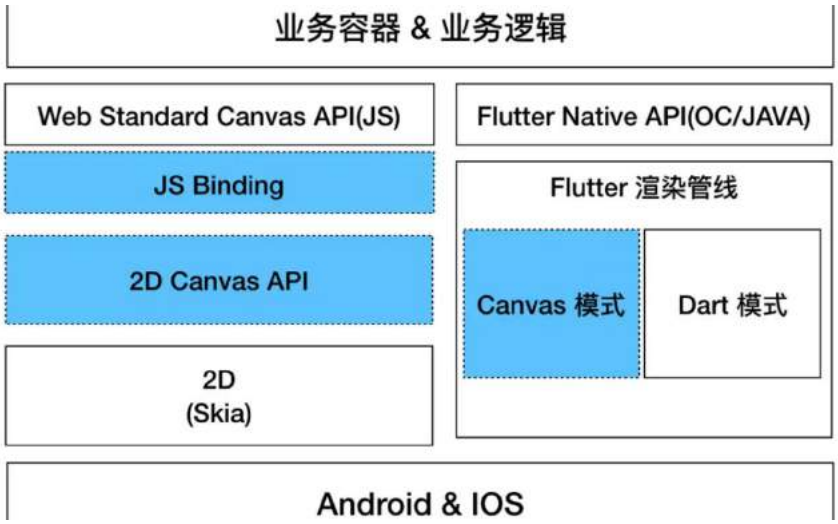
深入了解了 Flutter 引擎的渲染机制后，基于业务的诉求，我们也做了一些相关的探索，这里简单分享一下。

小程序渲染引擎



基于 Flutter engine，我们去除了原生的 dart 引擎，引入 js 引擎，用 C++重写了 Flutter Framework 中的 rendering，painting 以及 widget 的核心逻辑，继续向上封装基础组件,实现 cssom 以及 C++版的响应式框架，对外提供统一的 JS Binding API，再向上对接小程序的 DSL，供小程序业务方使用。对于性能要求比较高的小程序，可以选择使用这条链路进行渲染，线下我们跑通了星巴克小程序的 UI 渲染，并具备了很好的性能体验。

小程序互动渲染引擎



受限于小程序 worker/render 的架构，互动业务中频繁的绘制操作需要经过序列化/反序列化并把消息从 worker 发送到 render 去执行渲染命令。基于 flutter engine，我们提供了一套独立的 2d 渲染引擎，引入 canvas 的渲染管线，提供标准的 canvas API 供业务直接在 worker 线程中使用，缩短渲染链路，提高性能。目前已经支持了相关的互动业务在线上运行，性能和稳定性表现很好。

总结与思考

本文着重分析了 flutter engine 的渲染流水线及其相关概念并简单分享了我们的一些探索。熟悉和了解渲染引擎的工作原来可以帮助我们在 Android 和 IOS 双端快速去构建一个差异化高效的渲染链路。这在目前双端主要以 web 作为跨平台渲染的主要形式下，提供了一个更容易定制和优化的方案。

参考链接:

https://github.com/flutter/flutter/wiki/The-Engine-architecture
https://github.com/google/skia/blob/master/include/core/SkPicture.h#L27

Flutter页面卡顿原因详解

作者 | 檀婷婷(三莅)
出品 | 淘系技术

背景

高性能高流畅度一直是 Flutter 团队宣传的一大亮点，也是当初闲鱼选择 Flutter 的重要因素之一，但是随着复杂业务的应用落地，通过 Flutter 页面和原生页面滑动流畅度对比，我们开始产生怀疑，因为部分 Flutter 页面流畅度明显低于 Native，是 Flutter 的宣传言过其实还是我们开发人员使用姿势有问题，今天我们就来具体分析下。

Flutter 渲染原理简介

优化之前我们先来介绍下 Flutter 的渲染原理，通过这部分基础了解渲染流程以及主要耗时花费。

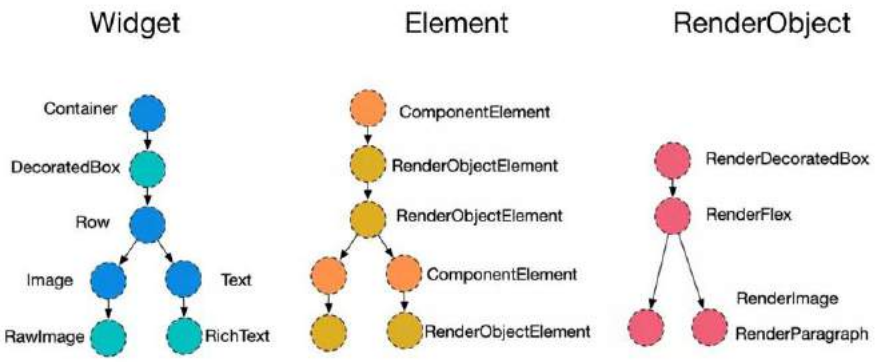
Flutter 视图树包含了三颗树：Widget、Element、RenderObject

- Widget: 存放渲染内容、它只是一个配置数据结构，创建是非常轻量的，在页面刷新的过程中随时会重建。
- Element: 同时持有 Widget 和 RenderObject，存放上下文信息，通过它来遍历视图树，支撑 UI 结构。
- RenderObject: 根据 Widget 的布局属性进行 layout，paint ，负责真正的渲染。

从创建到渲染的大体流程是：根据 Widget 生成 Element，然后创建相应的 RenderObject 并关联到 Element.renderObject 属性上，最后再通过 RenderObject 来完成布局排列和绘制。例如下面这段布局代码:

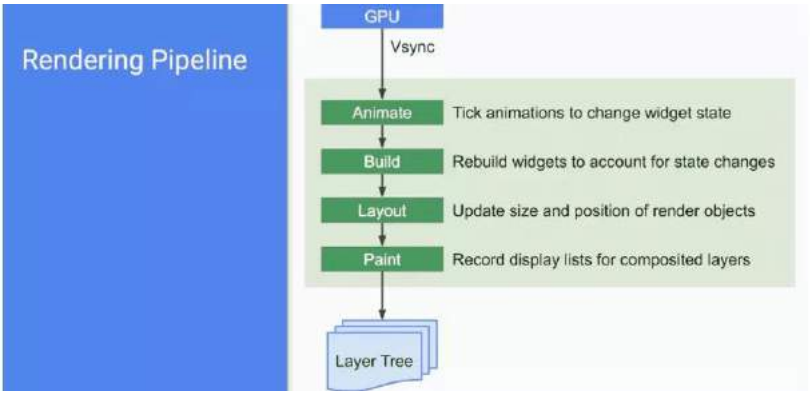
```
Container(
      color: Colors.blue,
      child: Row(
        children: <Widget>[
Image.asset('image'),
Text('text'),
  ],
  ),
);
```

对应三棵树的结构如下图：



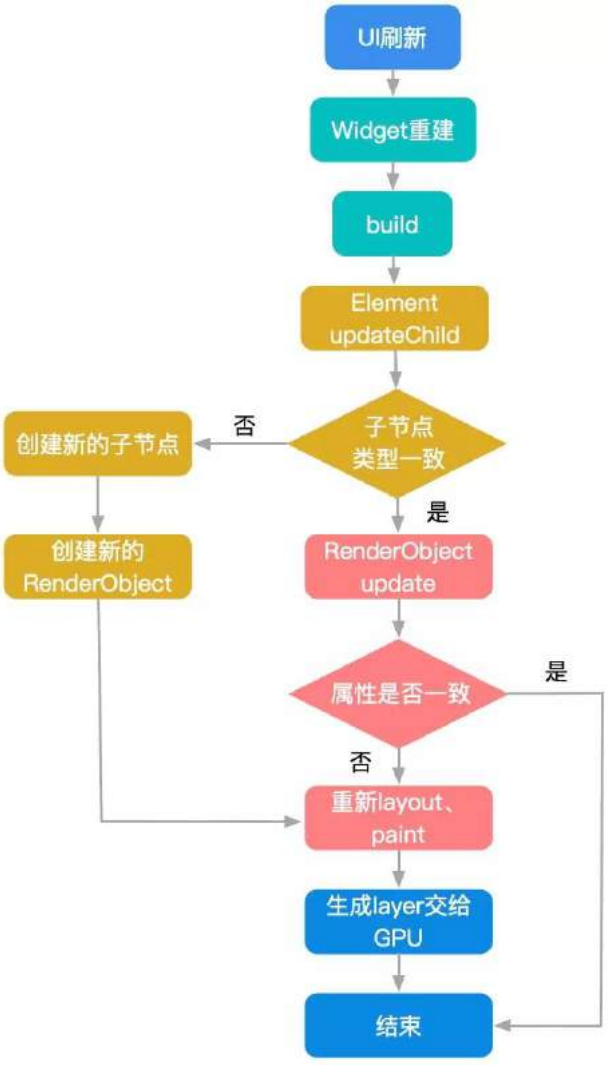
了解了这三棵树，我们再来看下页面刷新的时候具体做了哪些操作？

当需要更新 UI 的时候，Framework 通知 Engine，Engine 会等到下个 Vsync 信号到达的时候，会通知 Framework 进行 animate，build，layout，paint，最后生成 layer 提交给 Engine。Engine 会把 layer 进行组合，生成纹理，最后通过 Open GI 接口提交数据给 GPU， GPU 经过处理后在显示器上面显示，如下图：



结合前面的例子，如果 text 文本或者 image 内容发生变化会触发哪些操作呢?

Widget 是不可改变，需要重新创建一颗新树，build 开始，然后对上一帧的 element 树做遍历，调用他的 updateChild，看子节点类型跟之前是不是一样，不一样的话就把子节点扔掉，创造一个新的，一样的话就做内容更新，对 renderObject 做 updateRenderObject 操作，updateRenderObject 内部实现会判断现在的节点跟上一帧是不是有改动，有改动才会别标记 dirty，重新 layout、paint，再生成新的 layer 交给 GPU，流程如下图:



到这里大家对 Flutter 在渲染方面有基本的理解，作为后面优化部分内容理解的基础。

性能分析工具及方法

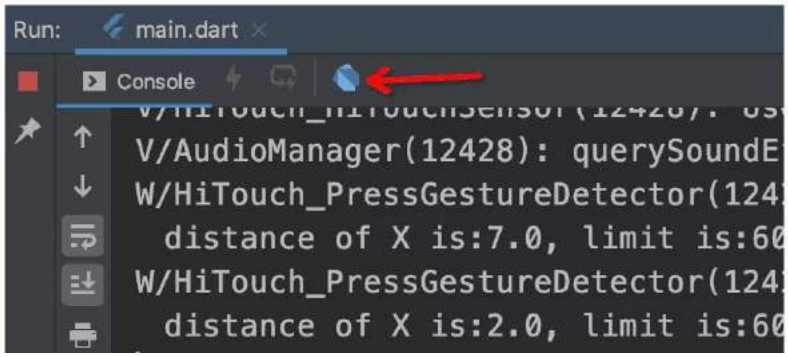
下面来看下性能分析工具，注意，统计性能数据一定要在真机+profile 模式下运行，拿到最接近真实的体验数据。

performance overlay

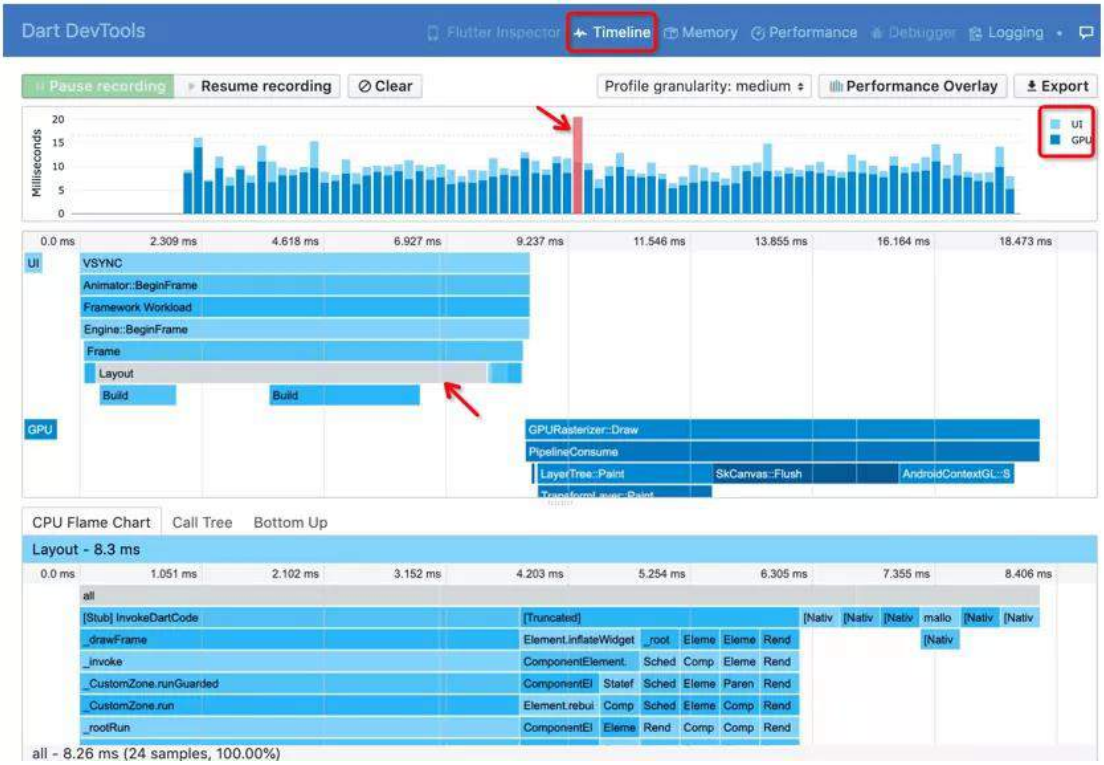
平时常用的性能分析工具有 performance overlay，通过他可以直观看到当前帧的耗时，但是他是 UI 线程和 GPU 线程分开展示的，UI Task Runner 是 Flutter Engine 用于执行 Dart root isolate 代码，GPU Task Runner 被用于执行设备 GPU 的相关调用。绿色的线表示当前帧，出现红色则表示耗时超过 16.6ms，也就是发生丢帧现象。

Dart DevTool

另一个工具是 Dart DevTool ，就是早期的 Observatory，官方提供的性能检测工具。它的 timeline 界面可以让逐帧分析应用的 UI 性能。但是目前还是预览版，存在一些问题。profile 模式下运行起来，点击 android studio 底部的菜单按钮，会弹出一个网页。



点击顶部的 Timeline 菜单



这个时候滑动页面，每一帧的耗时会以柱形 bar 的形式显示在页面上，每条 bar 代表一个 frame，同时用不同颜色区分 UI/GPU 线程耗时，这个时候我们要分析卡顿的场景就需要选中一条红色的 bar（总耗时超过 16.6ms），中间区域的 Frame events chart 显示了当前选中的 frame 的事件跟踪，UI 和 GPU 事件是独立的事件流，但它们共享一个公共的时间轴。

选中 Frame events chart 中的某个事件，以上图为例 Layout 耗时最长，我们选中它，会在底部 Flame chart 区域显示一个自顶向下的堆栈跟踪，每个堆栈帧的宽度表示它消耗 CPU 的时长，消耗大量 CPU 时长的堆栈是我们首要分析的重点，后面就是具体分析堆栈，定位卡顿问题。

debug 调试工具

另外还有一些 debug 调试工具可以辅助查看更多信息，注意，只能在 debug 模式下使用分析，拿到的数据不能作为性能标准：

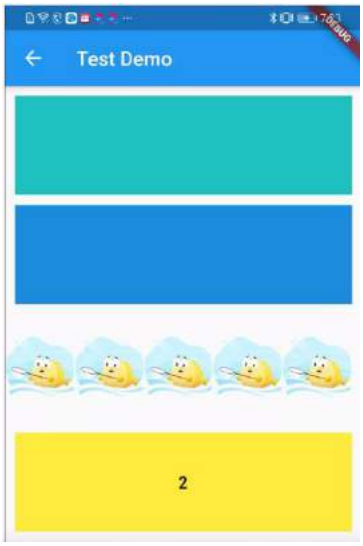
- debugProfileBuildsEnabled：向 Timeline 事件中添加每个 widget 的 build 信息
- debugProfilePaintsEnabled：向 timeline 事件中添加每个 renderObject 的 paint 信息
- debugPaintLayerBordersEnabled：每个 layer 会出现一个边框，帮助区分 layer 层级
- debugPrintRebuildDirtyWidgets：打印标记为 dirty 的 widgets
- debugPrintLayouts：打印标记为 dirty 的 renderObjects
- debugPrintBeginFrameBanner/debugPrintEndFrameBanner：打印每帧开始和结束

实例分析

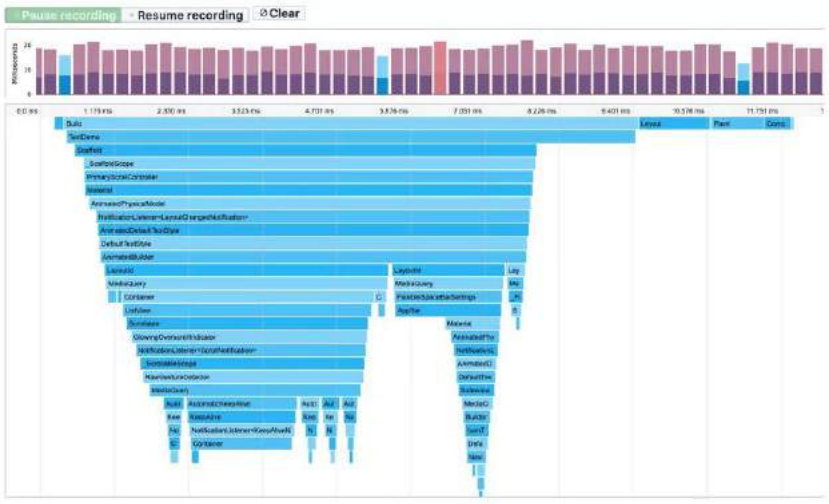
了解这些工具下面我们来看个简单的 demo 具体分析下，一个由 Column、Container、ListView 嵌套的布局，其中有个定时器控制 Text 中显示的文本实时更新。

```
class TestDemo extends StatefulWidget {
@override
State<StatefulWidget> createState() {
  return _TestDemoState();
}
}
class _TestDemoState extends State<TestDemo> {
int _count = 0;
Timer _timer;
...
@override
Widget build(BuildContext context) {
  return new Scaffold(
      appBar: new AppBar(
        title: new Text("Test Demo"),
      ),
      body: content()
  );
}
Widget content(){
  Widget result = Column(
    children: <Widget>[
      Container(...),
      Container(...),
      Container(...),
      Container(
```

```
            ...
            child: Center(
              child: Text(
                _count.toString(),
              ),)), ],
  );
  return result;
}
}
```



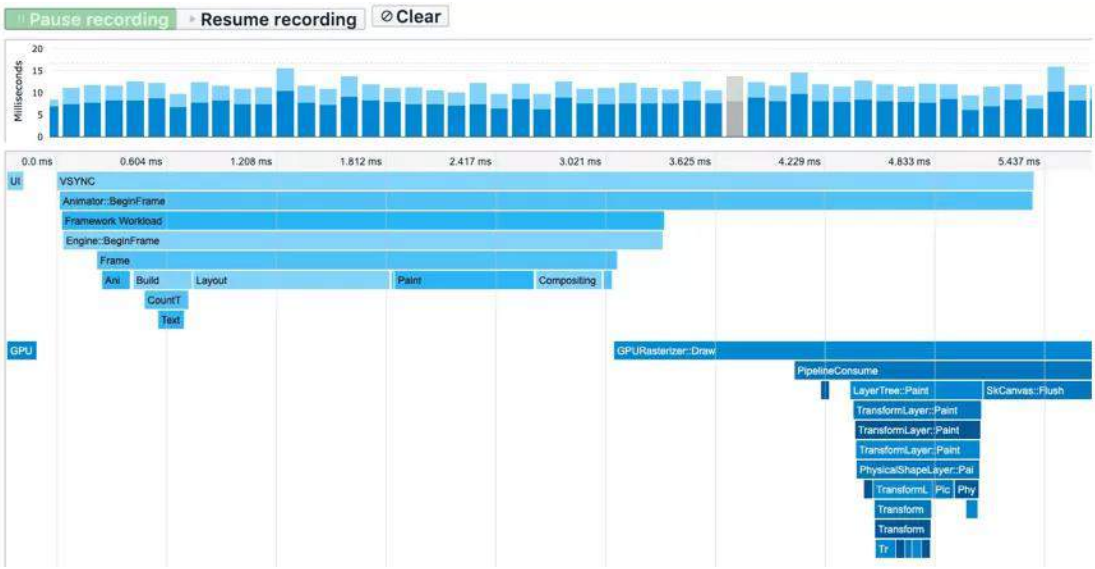
大部分 widget 都是静态的，只有黄色 Container 中包含一个内容一直刷新的 Text ，这个时候我们打开 debugProfileBuildsEnabled，用 Timeline 分析下它的渲染耗时，可以通过 Frame events chart 中显示的 build 层级非常深:



结合第一部分渲染原理我们了解到，每次定时器刷新 text 数字的时候，整个页面 widget 树都会重新 build，但其实只有最底层 Container 中的 Text 内容在改变，没有必要刷新整颗树，所以这里我们的优化方案是提高 build 效率，降低 Widget tree 遍历的出发点，将 setState 刷新数据尽量下发到底层节点，所以将 Text 单独抽取成独立的 Widget，setState 下发到抽取出的 Widget 内部。

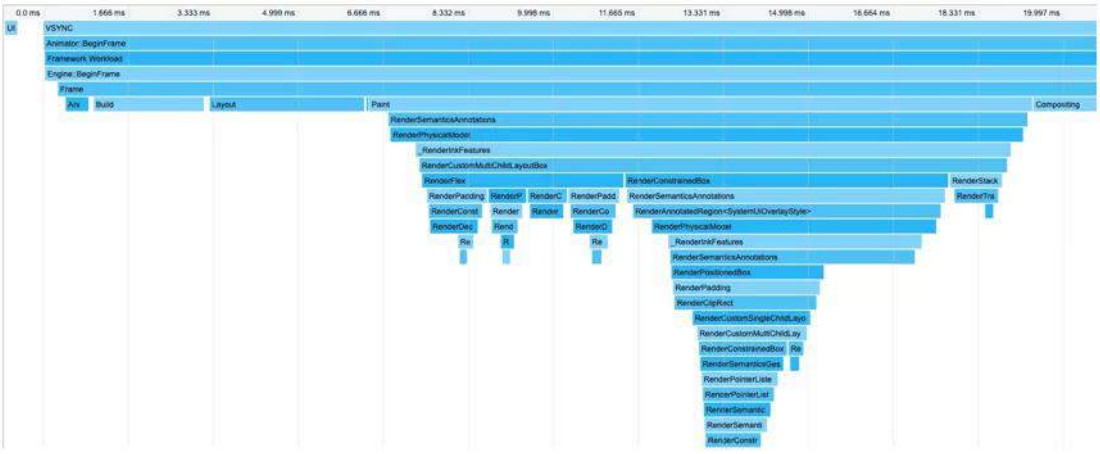
```
class _TestDemoState extends State<TestDemo> {
...
Widget content(){
  Widget result = Column(
    children: <Widget>[
      ...
      Container(
          ...
          child: Center(
            child:
                CountText()
          )),],
  );
  return result;
}
}

class CountText extends StatefulWidget {
@override
State<StatefulWidget> createState() {
  return _CountTextState();
}
}

class _CountTextState extends State<CountText> {
int _count = 0;
Timer _timer;
...
@override
Widget build(BuildContext context) {
  return Text(
    _count.toString(),
    style: TextStyle(fontSize: 18, fontWeight:FontWeight.bold),);
}
}
```

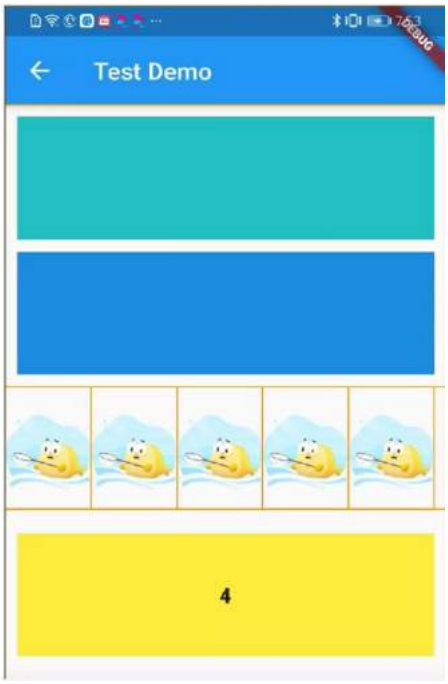
修改后的 Timeline 显示如下图：



build 层级明显减少，总耗时也明显降低。

接下来分析下 Paint 过程有没有可以优化的部分，我们打开 debugProfilePaintsEnabled 变量分析可以看到 Timeline 显示的 paint 层级：

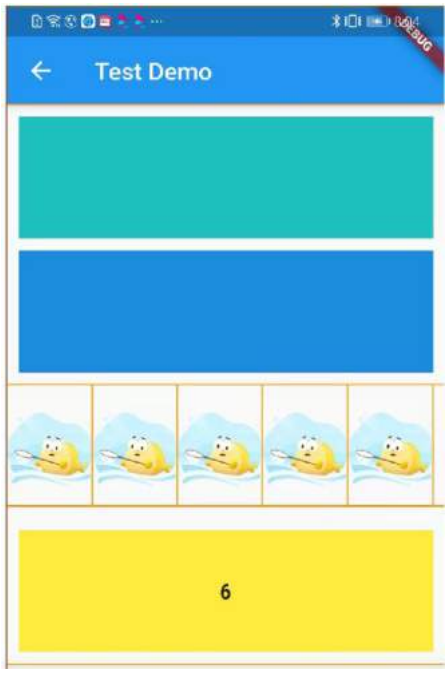




通过 debugPaintLayerBordersEnabled=true;显示 layer 边框可以看到不断变化的 Text 和其他 Widget 都是在同一个 layer 中的，这里我们想到的优化点是利用 RepaintBoundary 提高 paint 效率，它为经常发生显示变化的内容提供一个新的隔离 layer，新的 layer paint 不会影响到其他 layer。

```
RepaintBoundary(
            child: Container(
                margin: EdgeInsets.fromLTRB(10,20,10,10),
                height: 100,
                width: 350,
                color: Colors.yellow,
                child: Center(
                    child: CountText()
)
),
),
```

看下优化后的效果:



可以看到我们为黄色的 Container 建立了单独的 layer，并且 paint 的层级减少很多。

常见问题总结:

- 提高 build 效率，setState 刷新数据尽量下发到底层节点。
- 提高 paint 效率，RepaintBoundry 创建单独 layer 减少重绘区域。

这两个我们之前的例子已经具体分析过:

- 减少 build 中逻辑处理，因为 widget 在页面刷新的过程中随时会通过 build 重建，build 调用频繁，我们应该只处理跟 UI 相关的逻辑。
- 减少 saveLayer（ShaderMask、ColorFilter、Text Overflow）、clipPath 的使用，saveLayer 会在 GPU 中分配一块新的绘图缓冲区，切换绘图目标，这个操作是在 GPU 中非常耗时的，clipPath 会影响每个绘图指令,做相交操作，之外的部分剔除掉，所以这也是个耗时操作。
- 减少 Opacity Widget 使用，尤其是在动画中，因为他会导致 widget 每一帧都会被重建，可以用 AnimatedOpacity 或 FadeInImage 进行代替。

以上内容介绍了些 Flutter 常见的性能问题以及我们怎么用工具检测这个问题，在平时开发过程中要留意规避这类问题。

Flutter-DX 案例分析

近期我们做了个 Flutter 端的动态化模板渲染方案 Flutter-DX，它使用集团 DinamicX 的 DSL，通过下发 DSL 模板，在 Flutter 侧实现动态解析渲染。具体介绍可以参考之前的文章：

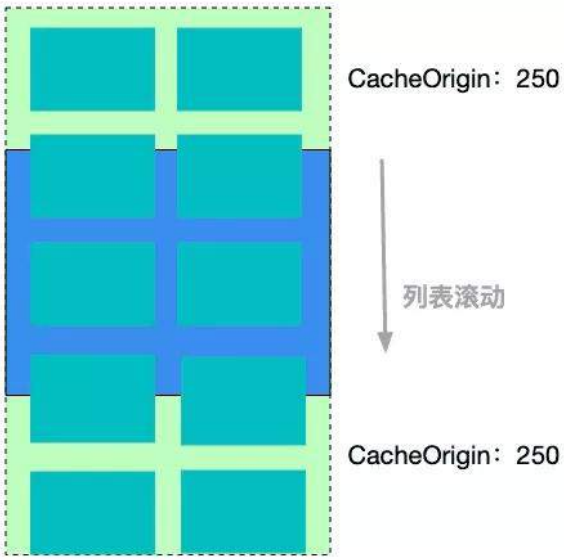
《如何在 Flutter 上实现高性能的动态模板渲染》
《做一个高一致性、高性能的 Flutter 动态渲染，真的很难么？》

这里不再详细介绍。

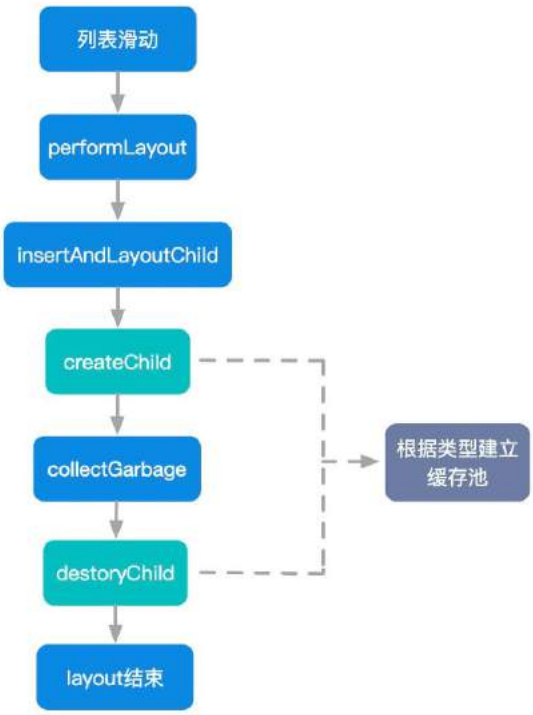
尽管进行了一次渲染架构升级，很大程度上提升性能表现，但是通过高可用线上统计，发现在长列表场景下 fps 值没有达到预期值，所以需要进一步分析哪些操作导致的耗时问题。

以搜索页页面结构为例，外部是 GridView 的容器，里面都是一个个 DX 模板组成的宝贝 card，滑动过程中发现流畅度要明显偏低所以我们做了以下的优化措施：

针对 Sliver 滑动的优化，sliver 在滑动过程中，有一个超出屏幕上下 250 像素的一个缓存区：

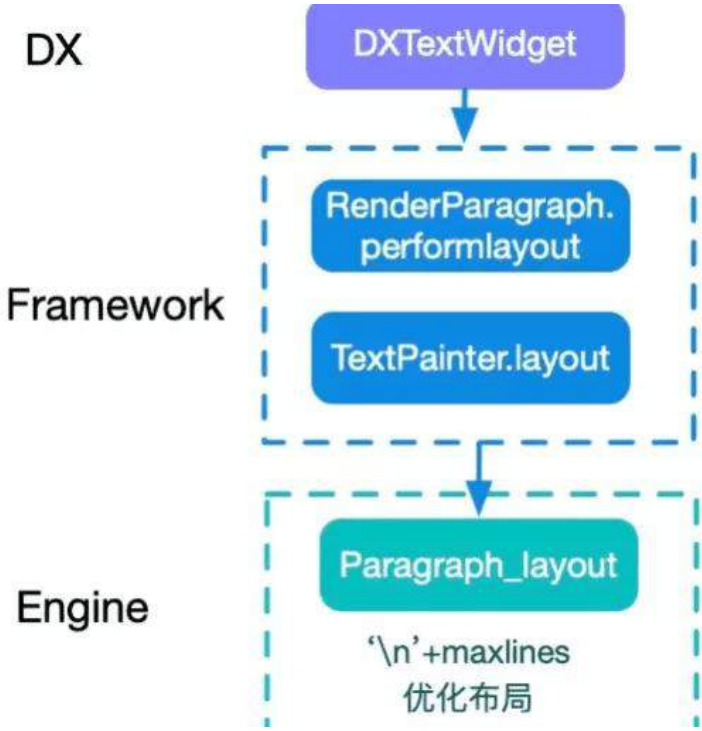


在列表滚动过程中，DX card 不断的被重建和销毁，没有任何缓存机制，我们在其中加了个缓存池，流程如下，避免 element 不断的被销毁和创建，一定程度提高流畅度。



通过 Timeline 分析发现 TextPaint 的 layout 耗时显著，进一步对比分析发现，同样的 UI 显示，带换行符的长文本长度 layout 耗时明显偏高。

后来确认带换行符的文本会影响布局效率，具体分析可以查看 issue:



这里我们做的优化措施是在判断只有一行文本显示的情况下，截取换行符前的内容作为 text 文本，从而提升 TextPaint layout 效率。

除此之外，还有一些减少布局层级和简化 build 流程，预加载缓存等措施，实现将 FPS 提升 3 个点，达到一定程度的优化效果。

总结

以上内容分析了 flutter 的渲染原理以及遇到卡顿问题可以用哪些工具从哪些方向入手分析，Flutter 虽然一直宣称流畅度是一大亮点，但也存在一定的优化空间，以及需要开发者掌握一定的开发技巧才能达到更丝滑的体验。

Flutter 精彩问答

Q: Flutter 凉了吗?

作者 | MNN 团队
出品 | 淘系技术

> 早年间大热的 flutter 好像没火起来，并没有看到有国内大厂完全使用 flutter 开发应用。即使一般体量的 app 都没见过，能搜到的都是网友自己开发的 demo 级别产品，不禁怀疑 flutter 是不是凉了。

去年，2019 年无疑是 Flutter 技术如火如荼发展的一年。

每一个移动开发者都在为 Flutter 带来的"快速开发、富有表现力和灵活的 UI、原生性能"的特色和理念而痴狂，从超级 App 到独立应用，从纯 Flutter 到混合栈，开发者们在不同的场景下乐此不疲的探索和应用着 Flutter 技术，也在面临着各种各样不同的挑战。

阿里巴巴集团内也有越来越多的业务和团队开始尝试 Flutter 技术栈，从闲鱼的一支独秀引领潮流，到如今淘宝特价版、盒马、优酷、飞猪等 BU 业务相继入局，Flutter 的业务应用在集团内也已经逐渐形成趋势。

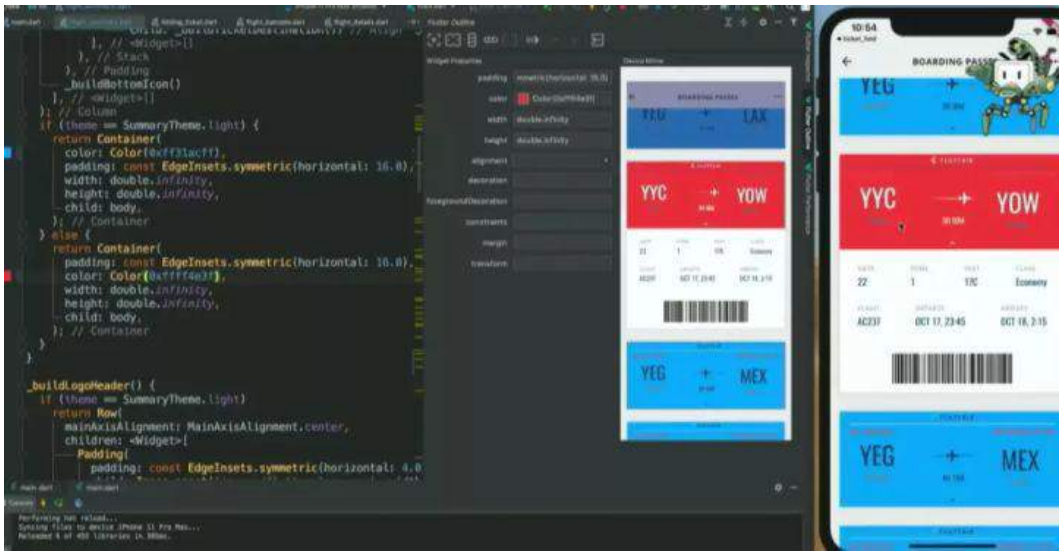
那么，是什么原因让集团内越来越多的开发者选择拥抱 Flutter 技术栈？Flutter 的哪些优势吸引了集团 Native 开发者们通过 Flutter 开发并交付业务？

我们分享一下阿里集团内进行 Flutter 体系化建设的情况，以供参考和交流。

从技术上看，阿里巴巴新零售淘系技术部的思牧认为 Flutter 最核心的 3 个特点最为吸引开发者：

- 极高的开发与交付效率，良好的开发体验
- 优秀的跨多端多平台能力
- 极强的 UI 表现力

先说一下**开发效率**。从集团电商业务属性出发，业务响应效率及其背后的研发效率从来都是最为重要的指标。在保证体验的前提下，尽可能的提高研发效率，就意味着更高的生产力。传统的 Native 业务研发 iOS/Android 双端需要分别投入，研发成本高，端差异性大且依赖端侧发版，这也是为什么集团电商业务的活动类技术栈一直较为依赖前端体系，从 H5 到 Weex 到小程序，很大程度上就是在追求研发和交付效率以及灵活性。如今 Flutter 很好的解决了跨端一致性问题，一套代码无差异的同时跑在 iOS 与 Android 两端；开发体验基本接近前端，支持 on device 的 Hot Reload ，去年年底 Flutter 又推出了在 Android Studio 中通过插件实现实时预览并支持交互的 Hot UI 能力，以及 Layout Explorer 可视化布局，让 Flutter 的开发效率和前端效率基本持平。

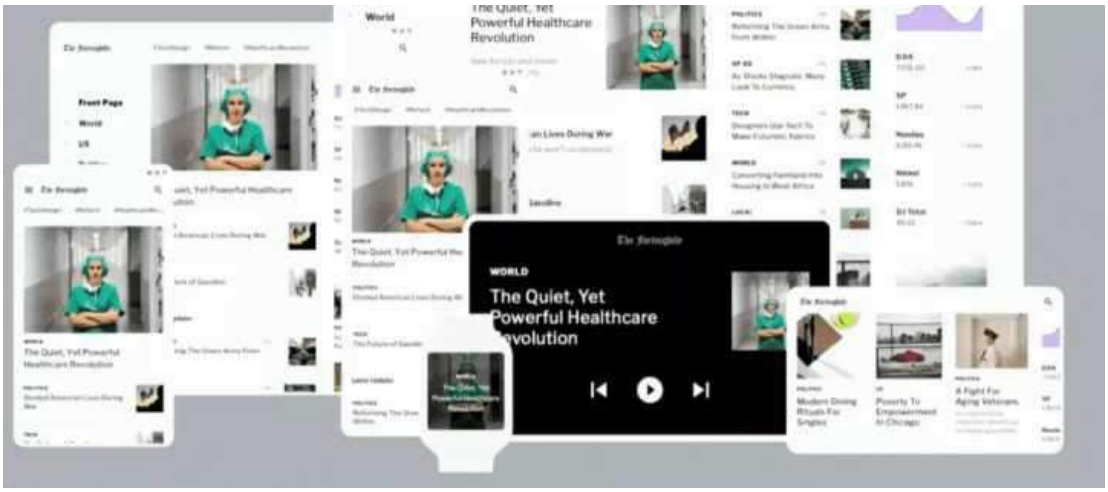


电商业务发展到当前阶段，已经不再仅仅局限于移动端场景，越来越多的业务需求对跨端跨平台性提出了更高的要求。钉钉/千牛桌面端应用场景，甚至天猫精灵、线下门店等业务场景，从长远看都需要一个比 Web 性能一致性更好适配成本更低的多端方案。目前跨多端

技术方案主要依赖于浏览器和前端体系，但浏览器本身的沙盒属性、与系统较低的结合度、以及在低端设备上较差的性能都降低了研发效率和用户体验，提高了业务的交付门槛。

可以说目前集团内的跨多端多平台方案是实质缺失的。Flutter 从设计上就天然支持多平台开发，它的底层基于 Skia 跨平台图形引擎，向上构建出了一整套平台无关的渲染体系和事件处理体系，并紧贴 Native 研发模式自定义了基于 widgets 的声明+响应式编程范式，对系统能力依赖度低，并具备出色的跨平台还原度；支持多平台也是 Flutter 的战略目标之一。

目前除了 iOS 和 Android ，官方宣布支持的平台有 Mac、Windows 和 Web，Linux 也在开发中，它的技术特性也让将 Flutter 移植到 Linux based IoT 平台上成本很低，同时 Flutter 还是未来 Google 的下一代操作系统 Fuschia 的官方应用研发框架。可以说 Flutter 已经具备了成为下一代跨多端多平台研发模式的一切条件，围绕 Flutter 建立集团的多端多平台研发体系是非常可行的选择。



最后讲一下 **UI 表现力**。电商业务重体验，重交互，尤其对于流量精细化运营场景，富交互的游戏化表现方式已经成为流量促活的重要手段。在 UI 表现力方面，前端体系一直具备着优势，通过 CSS3 强大的动画能力，开发者可以非常容易的实现复杂的动画效果和交互体验，而基于 Native UI ，需要借助各种动画特效三方库，双端开发体验不一致，实现复杂且交付效率低。

Flutter 很好的解决了这个问题，从补间（Tween）动画、基于物理属性的动画，到相对复杂的页面间 Hero 动画、parallax 交错动画等特效，Flutter 都可以跨平台低成本的高效实现。

Flutter 体系化建设现状

目前集团内有多个业务 BU 均已开始尝试应用 Flutter 技术栈，涵盖了从电商详情业务、导购频道，到 Feeds 流、游戏化交互以及国际化等多个业务场景。目前 Flutter 技术在集团应用的痛点在于，研发基础设施的中台基建不够完善，研发支撑能力与数据运维能力未实现标准化，集团 Flutter 开发者生态还未完全拉通，暂时未能形成合力。这些问题将是我们后面在集团层面建设 Flutter 技术体系的重点。

另一方面从行业趋势上看，Flutter 技术已经成为越来越多行业伙伴重点投入的技术建设方向。字节跳动、美团等公司均建设了自己的 Flutter 工程化体系，并服务了各自的业务场景；腾讯也基于 Flutter 在多个 App 上进行了应用尝试，并在 Flutter 渲染能力服务小程序的场景下做了有益探索。行业伙伴们在 Flutter 技术上的投入力度和决心，一方面让我们对 Flutter 技术的应用前景和社区更有信心，另一方面也让我们感到联合集团各方力量共建 Flutter 生态的必要性和紧迫性。

手淘的尝试和思考

最后，简单讲一下我们从 18 年到现在在 Flutter 上做的探索和思考。手淘从 18 年 10 月开始探索 Flutter 渲染引擎应用在小程序场景；19 年下半年开始建设 Flutter 基础能力，并服务了淘宝特价版业务，在引擎、图片库、内存优化和加载性能等关键技术上做了沉淀；同时通过对 Flutter 的引擎改造，封装出 Flutter 2D Canvas 能力，向上支持小程序 Canvas 组件及小游戏引擎，服务 2D/2.5D 游戏化业务，并在业务场景中落地。

在这个过程中，我们也沉淀了解决内存问题和图片问题等方案，以及 Flutter 技术与 Web 技术的对比与思考，取得了一定的技术及业务价值。 通过这些尝试，加深了我们对 Flutter 技术的掌控力和理解。在我们看来， Flutter 的横空出世，完全可以被看作吹响了 Native 体系复兴的号角。在保持 Native 性能优势的前提下，Flutter 带来了优秀的跨端一致性、

贴近前端的研发效率，以及强大的 UI 表现力，为集团业务使用 Native 技术栈带来了新的可能。

从业务应用上看：Flutter 目前带来的最大价值是研发效率的提升。在基建和 native 扩展能力完备的前提下，开发基于 Flutter 的纯 Dart 业务的人效比之前各端分别开发的效率提高了接近 2 倍，单位时间内的需求响应能力也相应提高了接近 2 倍，目前已在闲鱼和特价版业务开发中得到了很好的工程化验证。

从适应场景看：Flutter 目前比较适合承载富图文内容，如详情、Feeds 流、用户主页等常规业务开发，以及 2D/2.5D 游戏场景以及富动效业务。Flutter 通过单端技术栈可以同时满足以前需要 iOS、Android 以及前端技术栈分别负责的业务场景，甚至可以通过端云一体化的开发模式使用 Dart 负责一部分服务端业务逻辑开发，可以帮助业务团队拓展业务边界的同时，实现前后端研发能力闭环。Flutter 目前的限制在于，动态性能力及前期的投入成本。

前期投入成本主要指技术学习与团队研发模式升级的成本，涉及到技术路线选择，是我们和每个业务团队需要一起思考和判断的，这里不展开谈。动态性能力是 Flutter 的相对短板，目前能够通过 Flutter 模板化技术实现基于模板的组件级动态化能力，但基于性能、审核及对原生 Flutter 体系的侵入性等多种因素，目前还不能去直接实现 UI + 逻辑动态化能力。Flutter Web 方案虽然不存在审核限制，但受限于浏览器 DOM API 与 widgets 体系的差异性，目前仍旧存在较严重的性能瓶颈和渲染差异性，仅可作为降级的备用方案，暂时无法作为动态化的主要实现方案。

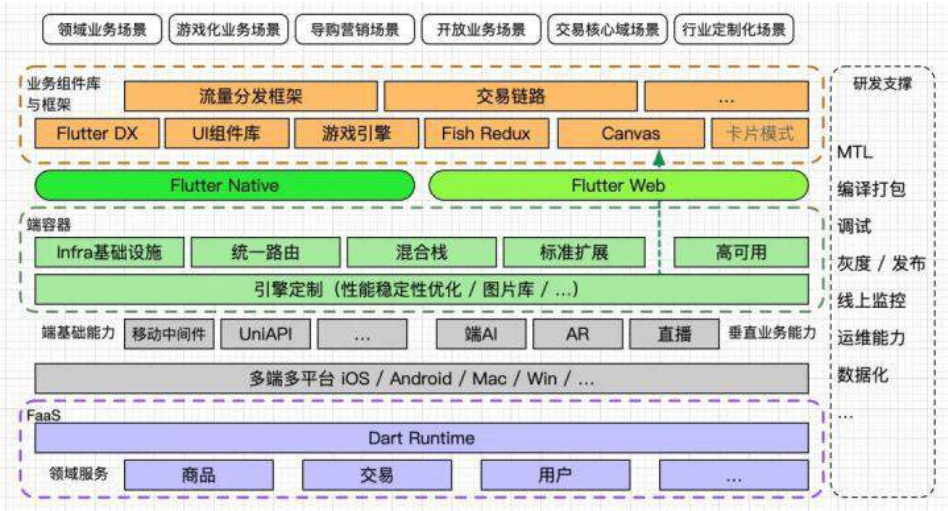
未来在动态化方向的探索也将是个长期的博弈过程。如果后面我们可以解决好 Flutter 动态化的问题，那么 Flutter 完全有机会成为集团业务的核心研发模式之一。综上我们认为，入局 Flutter 的时机已成熟，合力共建 Flutter 集团移动生态，这件事情大有可为。

AliFlutter - 经济体移动小组 Flutter 共建项目

在这样的背景下，经济体移动技术小组今年也将端侧架构治理的重点方向放在了 Flutter 上。移动技术小组从战略角度提出了 AliFlutter 项目，目标非常明确：

- 从经济体层面拉通 Flutter 体系建设，打造 Flutter 的公共技术基础设施，制定 Flutter 容器、中间件与 API 标准，建设 Flutter 研发支撑与数据运维能力，复用关键技术；
- 联合各 BU 构建经济体 Flutter 技术社区，沉淀共享集团 Flutter 技术及业务组件、能力与解决方案，服务集团 Flutter 业务，共建集团 Flutter 技术生态；
- 在经济体层面构建 Flutter 的对外影响力，联合各 BU 一致对外，打造阿里在行业内的 Flutter 技术制高点。

为经济体的 Flutter 技术体系"建基础、育社区、扛大旗"，我们责无旁贷。未来 AliFlutter 的整体架构如下所示：



AliFlutter 建设的第一步，就是要构建集团的 Flutter 基础设施、提供公共容器与组件、研发支撑服务与标准化研发流程，为集团的 Flutter 业务提供一个基础的 Flutter 公共研发服务能力，搭建好技术共建共享的基础和平台；

第二步，我们要服务好 Flutter 业务应用，探索业务应用模式与 Flutter 技术特性的结合点，并在过程中打磨技术，形成针对业务特点的解决方案与技术沉淀，真正盘活集团内的 Flutter 社区共建氛围与生态；

第三步，面向未来，解决好 Flutter 应用的几个核心关键问题：跨端与交互能力、业务研发效率与业务交付效率，通过技术赋能业务，让 Flutter 真正成为集团移动业务的核心研发模式。接下来，就详细讲一讲每个阶段 AliFlutter 所做的工作和面向未来的思考。

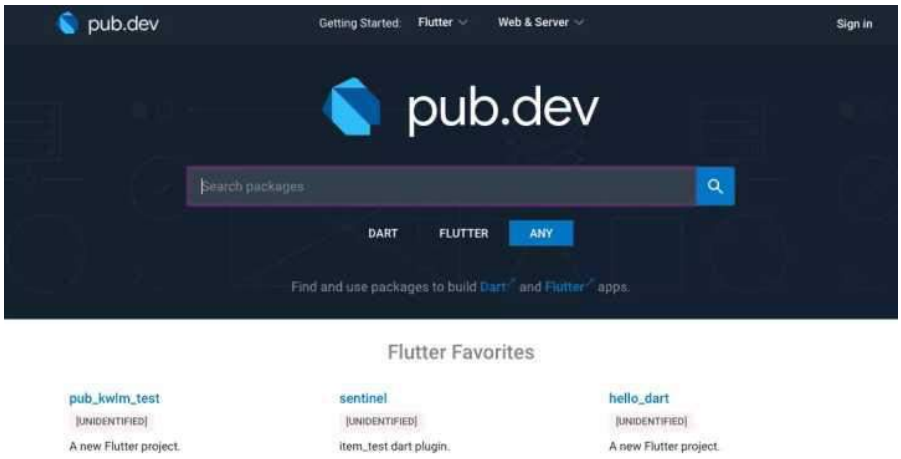
基础设施建设

从 19 年 10 月 AliFlutter 项目启动开始到现在，我们基本构建起了一套 Flutter 的公共基础设施，包括 Artifacts 与 pub 库，Flutter 标准容器与 API 标准，并实现了 Flutter 的构建打包自动化，定义了标准的引擎定制工作流与业务研发工作流。目前基础设施已经具备支撑集团 Flutter 业务研发的能力，并支持各 BU 按需定制。

Artifacts 仓库

产物服务器主要是为了配合引擎定制，加速通过 Flutter 命令获取 Engine 中间产物的后端服务，便于统一 Flutter 开发者的工作环境。开发者可通过设置 FLUTTER_STORAGE_BASE_URL 来将 Flutter 工具链获取 artifacts 的地址指向该服务，同时通过 namespace 便可实现定制化的获取 artifacts 的能力以及内网加速服务。 Namespace 设计为区分不同 BU 的引擎产物，同时提供了公共 namespace 来存储公共产物，确保定制性和公共能力的按需配置；若后端存储上不存在需要获取的产物地址，则会触发从 Flutter 官方镜像做一次获取并缓存在服务端。 各 BU 可按需定制引擎，并按规定的路径格式上传至自己 namespace 中，即可实现从 namespace 中获取定制版本的引擎中间产物。

pub 仓库



类似于 Node.js 世界的 npm，Flutter 使用 pub 来管理三方组件与依赖。考虑到易用性、安全性等需求，为了管理集团内的公共二方组件，我们也搭建了内网环境的 Flutter pub 库。该库的目标是成为集团的 pub 发布平台，管理集团内所有二、三方 pub package。用户可通过设置 PUB_HOSTED_URL 指向内部地址，来实现通过 Flutter 工具链获取配置以及发布二方 pub 的能力，用户也可以通过 Web portal 的方式访问 pub 库并查询已发布的 pub 组件。

容器、中间件与 API

对于业务的接入而言，现阶段核心要解决的问题就是提供一个统一的 Flutter 运行时容器，以及一系列集团标准化移动中间件的 Flutter 封装与 API 能力，并提供集团标准的插件扩展方式供业务方独立开发业务功能。

鉴于集团应用基本上均以混合栈为主，我们将 FlutterBoost 作为 Flutter 容器混合栈的基础，并配合集团标准路由与导航中间件提供了统一的混合栈路由导航能力，业务通过标准路由注册即可快速实现 Flutter 页面和 Native 页面的混合导航能力。

容器通过对接高可用平台，提供了初始化性能埋点与 Crash 数据上报等标准监控能力，为 Flutter 业务的技术性能分析和问题排查提供了基础。集团移动端积累了一整套的标准中间件体系，包括网络库、图片库、push 消息、配置下发、数据采集与监控等一系列基础能力，在 Flutter 体系内无缝使用移动中间件能力对于业务是刚需。

同时，小程序体系建设过程中形成的一系列标准 API，也很大程度上实现了一个完整的小程序运行环境的底层能力抽象，对于 Flutter 体系标准化的访问系统能力，实现平台无关的跨端能力是个非常好的补充。

我们联合淘宝中间件团队与小程序团队，对基础中间件和小程序 API 实现做了 Flutter 侧的封装与标准化，未来也将对 Flutter 中间件和 API 能力进行系统支持。

标准化 Flutter 构建

由于 Flutter 研发体系较新，且构建 Pipeline 相较传统的移动构建流程又存在一定特殊性，产物构建配置复杂耗时长易出错，给 Flutter 业务的构建和发版带来了很大阻碍。因此我们也联合研发支撑部的同学，以插件的形式实现了 Flutter 脚本化的构建流程，支持双端自动整包打包和 Flutter Module 打包。 目前 AliFlutter 的构建流程默认使用 AliFlutter 的 Flutter 仓库以及集团内部 pub 仓库，引擎产物也统一按配置从 artifacts 仓库获取，较好的实现了支持 Flutter 业务的自动化构建需求。

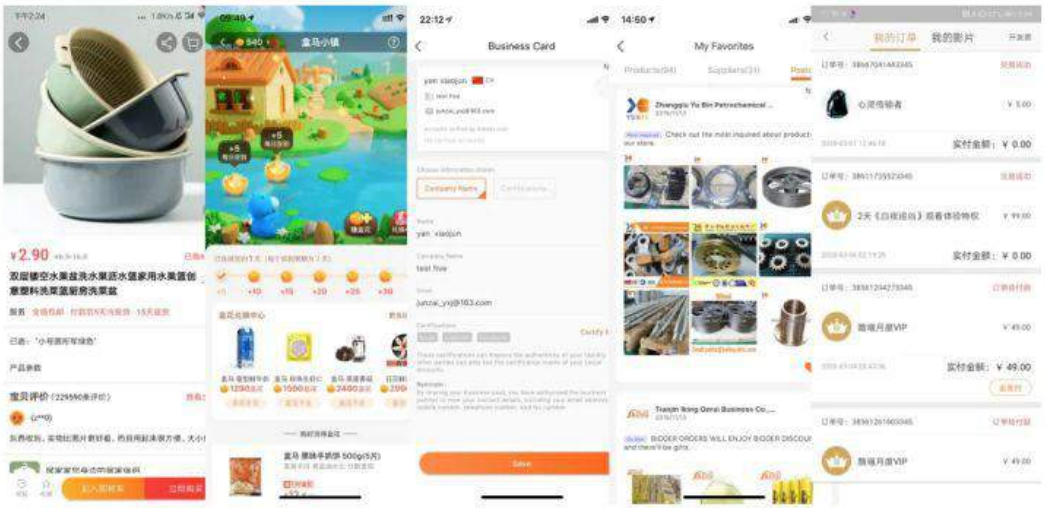
业务应用

在夯实 Flutter 集团共建基础之后，第二步，我们 AliFlutter 在业务应用方面也做了大量工作。一方面通过原生 Flutter 的工程化能力持续服务淘系与集团业务；另一方面通过 Flutter Canvas 项目服务了小程序场景及游戏化场景下的互动业务

淘系与集团业务支撑

目前淘宝特价版已完成详情业务的 Flutter 改造并上线，采用 Flutter 使业务在需求节奏不变的情况下人力投入减少一半，对缓解业务研发压力起到了明显的作用；同时应用的整体性能和稳定性与 Native 基本持平。

后续特价版将基于 Flutter 继续拓展业务改造范围，并沉淀基于 Flutter 的业务域解决方案。



目前盒马、ICBU 、优酷也基于 AliFlutter 进行了容器接入升级与业务适配，盒马依托闲鱼的 Flutter 游戏引擎实现了盒马小镇业务，ICBU 在主链路相关页面使用了 Flutter，优酷则基于 Flutter 实现了会员订单页等场景。

同时我们也在和钉钉及 Google 一起探索 Flutter 桌面端的解决方案。

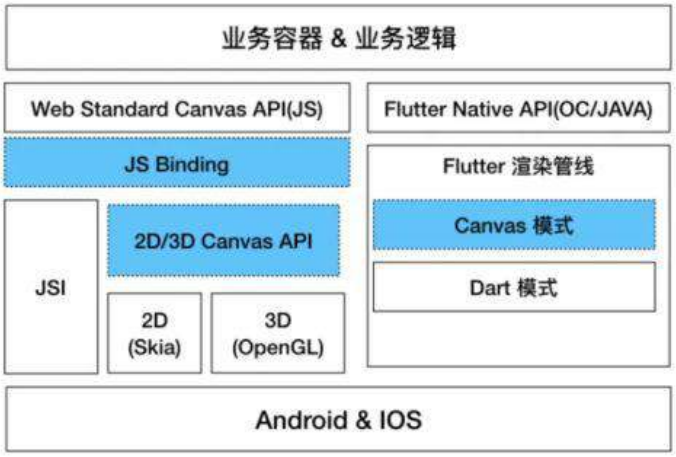
Flutter Canvas

在电商活动营销中互动场景日益增多，对性能要求持续提升的前提下，如何提供一个高性能且稳定的 Canvas 基础能力服务好富交互的互动场景就成为了一个重点的课题。

在小程序场景中 Canvas 作为承载互动游戏的主要能力发挥了重要作用。然而受限于小程序架构下 app context 和 page context 的隔离设计，存在从 app worker 到 page renderer 的通信瓶颈，无法充分发挥出 web canvas 的性能，如果有一个 native 版的 canvas 实现将可直接在 native 层对接 app worker ，降低通信成本，充分发挥 Canvas 的性能。

Flutter 底层基于 Skia，其性能和移动端复杂异构机型的适配性均得到过长期的检验，且 Flutter 基于浏览器的设计实现了一条平台无关的渲染管线，并对浏览器实现做了极大的简化，提供了很好的可靠性和性能。那么如果能够将这条渲染管线直接用于向业务容器提供

Canvas 能力，通过 binding 方式直接向小程序和小游戏容器提供与 Web Canvas 一致的标准 API，一方面可以复用 Flutter 的底层能力，为非 Dart 环境提供渲染支持，另一方面可以借助 Flutter 简化高效的渲染管线实现提供更好的渲染性能。



目前 Flutter Canvas 已落地手机淘宝，并在小程序运动银行业务进行了灰度试点，初步具备了承载小程序 Canvas 业务的能力；其性能在 Android 低端机上的表现有优势，可以作为 Web Canvas 方案的有益补充。

未来 Flutter Canvas 一方面将借助 Flutter 渲染管线的跨平台与高性能特点，以及 Flutter 对 Vulkan 和 Metal 的适配支持，在移动端获得更好的适配性以及性能；同时将继续实现 3D API ，支撑未来互动类的业务应用。

未来建设

扎根业务之后，接下来的第三步，我们要紧贴 Flutter 体系在阿里集团未来的建设目标，持续回答好 Flutter 面向未来建设路径中的几个关键问题。那么首先，Flutter 体系在阿里集团的建设目标应该是什么？个人以为：Flutter 应成为阿里集团未来跨多端多平台的核心业务

研发模式之一。那么，我们目前离这个目标还有多大差距？在我看来，如果要想让 Flutter 成为业务的核心研发模式，那么必须解决好跨端能力、交互能力、业务研发效率以及业务交付效率四个核心问题。

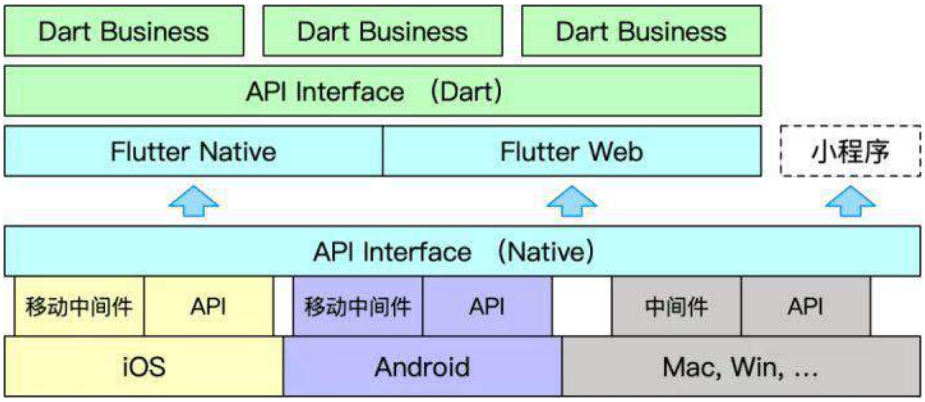
- **从跨端能力看**：Flutter 虽然已具备了很好的跨多端能力与高还原度，但涉及到平台能力时，仍然需要通过各端扩展实现，还未形成小程序体系这样的标准化的容器和 API 封装能力。那么如何更好的解决 Flutter 的容器化问题，让业务不感知平台差异性？
- **从交互能力看**：Flutter 如何利用好自身交互能力的优势，在提供媲美前端的富交互体验的同时，降低 Native 富交互特性开发的门槛，真正吸引 Native 开发者使用 Flutter 技术开发业务？
- **从业务研发效率看**：虽然 Flutter 的 Hot Reload/Hot UI 机制已经让开发 Native 页面的效率追上了前端，但在工程解耦方面仍然有很大提升空间，目前还无法高效的支持多业务团队并行开发；另一方面如何与现今流行的 Serverless 能力结合，实现端云一体研发模式，使业务实现研发闭环，也需要实践的检验。
- **从业务交付效率看**：目前 Flutter 仍属于 Native 方案，依赖端侧发版，交付效率低，无法很好的承载电商系灵活性和实效性的需求；那么如何解决 Flutter 的动态化，帮助业务实现快速迭代？

解决好这几个问题，才能真正让 Flutter 成为集团移动业务的核心研发模式，为集团业务研发带来一个飞跃性的提升。下面讲讲我们在这几个方向的思考和探索。

提升跨端能力：Flutter 容器化

从工程角度看，虽然 Flutter 通过 Skia 跨平台图形渲染和自建事件体系基本实现了对宿主平台的最小依赖，但对于平台侧能力，目前 Flutter 还未也没有必要从应用框架角度做到一个统一的抽象，这就需要我们根据业务的诉求和特点进行有选择的封装。小程序 API 就做了一个非常好的示范，目前阿里小程序体系提供的 API 达到了 200+，很好的对移动端的 UI、多媒体、文件缓存、网络、设备能力、数据安全以及业务相关能力进行了封装，让业务开发者在小程序侧针对 API 进行系统能力调用，无需关心平台实现。 因此 AliFlutter 容器接下来的规划就是从工程体系的角度，提供一套标准化的 API 能力，以规范并抽象移动端

的端基础能力，使业务尽量少甚至不关心平台差异性，专注于业务；同时借助标准化 API 的能力，实现跨多端多平台部署。



从移动端架构角度看，各个时期的跨平台方案都对 API 能力有着共同的诉求，从 H5 到 Weex ，再到后面的小程序，以及 Flutter 等容器环境，进行了多轮的 API 重复建设，造成了缺少 API 接口的标准化定义，以及缺少实现层统一管控的现状。如果能够在 API 的 native 实现上做到接口统一，再通过各个容器分别提供接口供业务使用，可以更好的做到实现收口，并在统一实现层跨容器实现对系统资源的统一调度、管控和度量。

提升交互能力：UI + 游戏引擎

前文提到过，Flutter 目前最大的价值在于研发效率的提升，这是吸引业务团队应用 Flutter 技术的起点；但仅仅依靠研发提效还远远不够，当通过各种工程化手段解决好当前研发痛点，提升研发效率之后，如何说服业务继续使用 Flutter 体系进行业务开发？Flutter 带来的长远价值在哪里？个人认为，这个落脚点应该在通过游戏交互能力的泛化，打破 UI 与游戏引擎的边界，用游戏化的方式创造更有表现力的交互体验，创造新的业务玩法和价值。 大家知道传统的 UI 和游戏引擎是相互独立的两个体系，在 H5 应用中，往往是通过 DOM 或者上层应用框架做 UI，通过建立在 canvas 上的 H5 游戏引擎实现游戏能力。

如果在游戏应用中有 UI 的需求，解决方案一般是自建一套简单的 UI 体系与事件体系，通过自绘的方式在游戏中叠加 UI ，独立游戏引擎亦是如此。Flutter 从技术原理上看更像是一个建立在 Skia 图形库上的游戏引擎，它通过细粒度的 widgets 设计向上构建 UI 系统；

同样得益于这样的细粒度设计，我们也完全可以直接通过 widgets 能力组合出一个完整的游戏引擎，提供 Game，Scene 及 Sprite 动效等 widgets 并扩展对应的 elements 和 render objects，并与 UI 体系共用一套事件处理机制、分层与渲染合成机制。

这样做就相当于打破了原来 H5 中 DOM UI 和 Canvas 游戏的边界，让两个体系在 widgets 概念下完美融合起来，通过游戏引擎的能力赋能 UI 实现更多以前 UI 体系难以低成本实现的动效效果（比如一只小盒马一口吃掉了一个订单组件等等）。我们相信，这个方向的探索将会进一步释放 Flutter 的技术潜力，带来更多的业务可玩性与创造性，解放产品和设计的想象力，为业务创造更多价值。

提升研发效率：工程解耦与端云一体化

Flutter 工程解耦

前端体系的研发效率很大程度上来自于基于 URI 的统一路由体系带来的页面间解耦性，与页面内基于 Web API 的标准化带来的内聚性。然而目前的 Flutter 研发模式，仍需要多个业务团队工作在同一个工程下，互相之间存在源码依赖，未来如果跨业务团队大规模应用 Flutter 技术，必将拖慢业务的研发效率。 从工程解耦角度看，目前 AliFlutter 容器通过混合栈与标准路由能力基本实现了页面研发的解耦，未来的容器化建设通过提供小程序对等的 API 能力封装，业务对平台无感知，能够让我们有机会解耦业务研发，实现与小程序开发接近的研发体验和效率。

理想的方案是：业务可以从业务维度创建一个独立的 Dart 工程，只包含业务相关的页面和逻辑代码，通过 Flutter 的 Hot UI 开发页面，通过 IDE 提供的基于 Flutter Web 的能力本地预览工程并调试 API 与系统调用，完成研发期工作；也可生成预览二维码，使用预装有 AliFlutter SDK 环境的宿主应用扫码预览；研发与构建链路分离，云端主动拉取业务仓库代码参与整包构建。以此达到类似小程序研发方式的前端研发体验，同时实现业务间的研发解耦和并行发布，提高业务的交付效率。

端云一体化

如今 Serverless 概念越来越多的受到业务研发的重视和应用，集团在新一代的端云一体化研发模式上的探索这一年多来也做的如火如荼。结合轻量级容器环境、多语言支持能力与统一的 API 服务端编程，端侧同学可以很容易的使用客户端语言如 Java、JS、Swift 甚至 Dart 来开发服务端业务能力，实现服务编排、服务端 FaaS 业务逻辑与 API 自动生成，达到前后端工程体系归一，业务研发闭环的效果。

目前闲鱼在 Dart FaaS 云端一体化的探索走在了前面，通过集团的容器规范实现了 Dart function 容器，并联合服务端为部分业务需要的领域服务抽象出来 BaaS 层（存储、消息队列等），并封装了面向前端的 BFF（Backend for Frontend）能力层，使移动端开发者可以很容易的使用 Dart 封装 FaaS 业务逻辑，同时进行移动端和服务端 FaaS 开发，大大提高了业务研发效率。通过将原有的端侧请求接口、组装数据并转换为 ViewModel 的逻辑都后移到了服务端，经过字段映射与页面编排，移动端可直接获取 ViewModel 并刷新页面，通过 BinderAction 双向交互状态数据，有效屏蔽了通信细节，提高了开发效率。

云端一体化除了带来了研发与协同效率的提升，同时也重塑了生产关系，使端侧业务从只关注端侧体验和逻辑的开发角色，变成能够向业务整体结果负责；同时也让服务端更专注于领域服务的沉淀。Flutter 良好的跨端特性，能够屏蔽掉端差异化，配合 Flutter 容器化改造，更近一步的简化了业务的全链路研发模式。未来如何在 FaaS 模式下，沉淀出一套可以服务集团业务研发的通用端侧和服务端通信调度框架，让集团 Flutter 开发者和业务都能共享到 Serverless 技术和云端一体化提效的红利，是需要我们共同去探索和定义的新问题。

提升交付效率：Flutter 动态化

实现动态化是交付效率提升的重要方式。对于电商强运营强时效性特性来说，动态化几乎是一个必备的诉求，但从技术上看也是一个非常敏感的需求，是一个在系统厂商平台管控下长期博弈的过程。在我看来，动态化技术需要解决的核心问题，是在保证业务发布确定性的前提下，寻求技术性能、业务迭代效率与灵活性三者之间合理的平衡点。

下面讲讲我们在 Flutter 动态化上的两个思路与尝试：Flutter 模板化方案与 Web on Flutter。Flutter 模板化方案动态模板化方案是集团内较为成熟的一套基于 Native 技术的模板化方案，专注于 UI 模板渲染，没有执行逻辑和运行时环境，目前被广泛应用于电商系的一些核心业务场景。同时该方案提供了配套的组件平台，支持在线模板编辑、预览、测试及发布整套流程，结合发布平台形成了一整套完善的业务开发生态闭环。 在 Flutter 体系下，目前闲鱼团队依据标准模板协议在 Flutter 侧实现了一套 Dart 版 SDK ，通过模板下发实现了 Flutter 端的轻量级动态化组件编排能力；并通过一年多的迭代逐步解决了渲染性能与渲染一致性问题，较好的赋能了 Flutter 业务的组件动态化能力。

那么，未来 Flutter 与动态模板化方案有没有更好的结合点？答案是肯定的。从 DSL 角度看，目前模板的写法基本上来自 Android XML ，对组件开发者尤其是非 Android 开发者不够友好，具备一定的学习成本；而 Flutter 的结构与属性均可通过 widgets 表达，可以极为灵活且平台中立的用声明式代码构建 UI 并绑定数据，易于开发者编写组件并通过 Flutter 框架独立调试与测试；在移动端运行时，可以按需按场景翻译成 native 组件或 Flutter 组件。未来我们也将持续在这个方向上做探索。

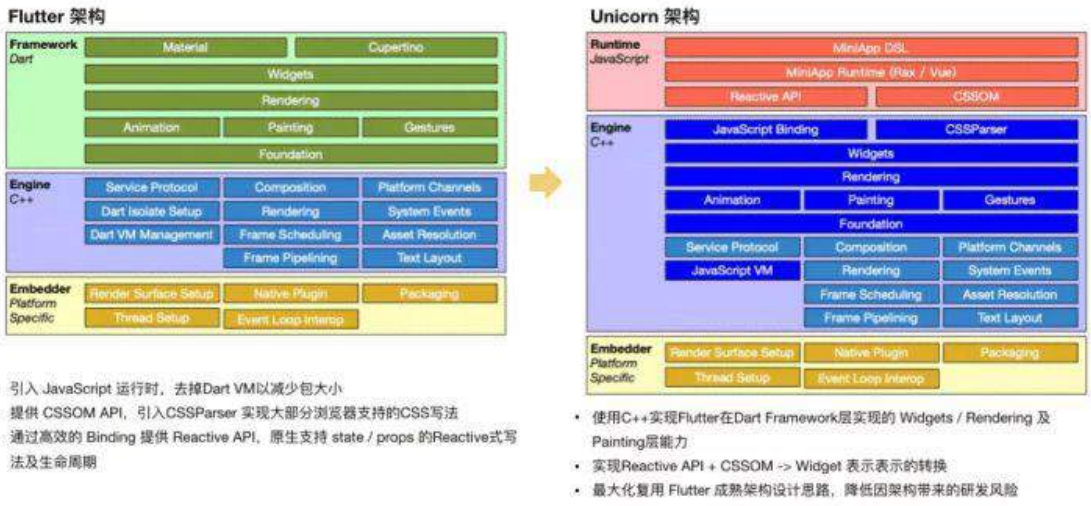
Web on Flutter

相比贴近 Native 研发模式的模板组件化渲染方案，Web on Flutter 希望通过类 H5 的 DSL + JS ，借助 Flutter 的渲染能力，实现贴前端技术栈的动态化能力。 目前基于 Web 渲染的小程序方案存在启动耗时高，渲染性能距原生 UI 有较大差距等性能问题，这些问题很大程度上都源自于浏览器引擎的设计历史包袱（渲染管线复杂、CSS multi-pass layout 及 legacy 实现等），以及 JS 到 Native 通信效率低（bridge）。

Flutter 的设计思路源自浏览器，一方面直接吸收了前端框架近年来的演进成就，原生支持了声明式-响应式编程范式，提高了移动端的研发效率；另一方面，Flutter 紧贴 native 开发模式有限定义了 UI 结构、布局与渲染的必要元素，在满足 native UI 开发模式的前提下简化了能力定义与布局算法（single-pass layout 与 repaint boundary 等概念），很大程度上简化了渲染管线的复杂度，直接为 Flutter 带来了接近原生的性能体验。

同时，Flutter 的 widgets 设计巧妙，结构布局属性等基础元素均使用 widgets 表达，且可通过基础 widgets 的组合来构成复杂组件，这种细粒度+组合能力设计让 Flutter 有极强的表现力，并具备向上对接各种研发模式的可能性。 因此，通过 widgets 组合小程序 DSL，支持小程序 CSS 有限集，实现渲染层替换浏览器引擎，并对接 JS 引擎支持 JS 执行能力，是一个将 Flutter 应用于小程序生态的合理探索方向。

相比把开发者惯坏了的浏览器，这种方案在 CSS 的能力支持上必然是受限的，无法满足所有 CSS3 标准的实现，更多通过紧贴 Flutter widgets 的现有能力以及必要的 widgets 扩展，在不破坏 single-pass layout 的前提下组合出 CSS 能力。但从 Flutter 原生开发的角度看，只要 Flutter 现有的原生能力能够满足业务需求，那么受限的 CSS 实现也一样可以提供和 Flutter 对等的能力解决业务问题。同时，通过受限的 CSS 可以换来与 Flutter 相当的高性能，与基于 Web 的实现相比，在性能上带来了质变。



我们在 18 年底通过一个内部项目实现了这个思路的原型，通过使用 C++ 重写 Flutter 的 widgets、rendering、painting 及事件管理等 Dart framework 中的中低层能力，并在 widget 上用 C++ 实现了 CSSOM + DSL -> Widgets 的响应式框架，直接从 C++ 层提供 render 实现，将传统由 JS 承担的模板展开、tree-diff 计算与渲染工作交给 C++层，通过 Flutter 提供的 Widgets tree 到 RenderObjects tree 的 diff 能力实现，显著提高了性能。

从实现的简单的 demo 看，相对小程序的 web 渲染性能有了大幅的提升。这种方案的问题在于和 Google 代码库分裂后的长期维护性。“Flutter 和 Web 生态如何对接”这篇文章对集团内在这个方向上尝试的几种思路做了较为全面的对比，未来我们也将继续在这个方向上做深入和持续的探索。

总结与展望

Flutter 体系化建设目前在淘系刚刚起步，仍然有大量工作需要去做，我们在基础设施、工程化以及通过 Flutter 定义和收敛业务域研发模式上做的许多建设性的事情，正朝着把 Flutter 打造为统一移动应用基础研发框架，助力业务回归移动端研发模式这个大目标一点点迈进。 在移动技术小组启动 AliFlutter 项目之前，闲鱼技术部已经在 Flutter 技术建设上做了大量探索和投入。

一方面通过 Flutter 技术赋能业务提效升级，拿到了很好的业务成绩；另一方面沉淀 Flutter 的技术与业务解决方案，并通过开源反哺社区，在海内外 Flutter 技术社区中建立了显著的技术影响力与领导力，也涌现出一众 Flutter 技术专家。 接下来 AliFlutter 的重点任务，就是要和闲鱼、财富等先驱应用者以及盒马、钉钉、飞猪、优酷、饿了么、CBU 等 Flutter 的实践者一起，在集团层面把 Flutter 的场子建起来，把集团 Flutter 生态拉起来，让技术和经验能够共同沉淀和分享，一起来把 Flutter 技术体系在阿里的应用生态内做大做强，真正成为集团业务的核心研发模式，并让每个参与者都能从中受益。

我们一直坚信 Flutter 技术的先进性和应用前景，未来我们将继续立足淘系服务集团业务，和集团开发者携手前行，在 Flutter 这个技术方向上坚定的走下去。

也欢迎关注我们的知乎账号，我们还将分享更多 Flutter 相关的技术干货与实践心得。

淘系实战项目篇

双 11 总结：2020 天猫双 11 前端体系的建设与挑战

作者 | 双 11 前端 PM-海文
出品 | 淘系技术

今年双 11 的整体节奏从之前的“光棍节”变为“双节棍”，具体业务上也有很多变化和调整，应了阿里的土话“唯一不变的是变化”。面对这些变化，是挑战也是机会，我们要做的就是，“既要”高效支撑保障业务先赢，“又要”确保体验和稳定性带给用户极致体验，“还要”追求创新让前端持续演进。为了实现“既要、又要、还要”，包括技术方案、流程机制、人员组织等各方面都进行了大量的设计和保障。最终第一次双峰的双 11 圆满结束，淘系前端也实现了自己的目标，包括应用大量优化手段和创新方案带来业务转化提升；将 FaaS、PHA、ESR 等技术应用在更多场景，分别向服务端、客户端、CDN 节点进一步拓展了前端的能力和边界；应用视觉还原、一体化研发等提升研发效率，大幅缓解资源瓶颈等等。下面会整体介绍一下淘系前端在今年双 11 的思考和沉淀，希望对大家有所助益。后续也会有各个专项的系列文章，希望大家持续关注。

变化 & 挑战

今年的双 11，首先感受到的就是源源不断的变化。

单峰变双峰：双 11 从之前的一个波段变成今年的两个波段，大促的三个阶段预售、预热、正式也都对应的翻倍。首先带来的是研发工作量的大幅增加，在时间排期不变、工作量增加、人员不变的情况高效的完成需求研发是第一重挑战；其次面对 6 个阶段的状态变化，如何保持准确切换、稳定运行、体验流畅是在双峰期间要重点保障的内容；最后面对超过 20 天的超长作战期，安全生产、人员状态保持、快速反应都需要有强力的组织和机制进行保障。

图：双 11 节奏

首页大改版：最新的淘宝首页首屏内容有颠覆性的变化，比如首屏内容简化，推荐提前，频道作为内容嵌入推荐等。各个业务在缺少固定的流量入口的情况下，包括运营策略、产品策略、设计方案、技术方案都需要积极调整。同时在各个场景的推荐能力也需要持续增强，今年双 11 通过将坑位数扩展到 1000+，理论可达无限扩坑；通过智能 UI 提升用户点击率。

图：手淘版本对比

业务变化：业务创新和新玩法层出不穷，包括 mini 详情、旗舰店、价格表达、笔笔返、芝麻购等在内的很多业务都是全新的表达、颠覆式的升级。即是业务上新的尝试，在技术上也要解决架构选型、对账、一致性表达、排期等问题。

做好本职

首先要做的就是做好本职工作，保障需求研发和稳定性。需求研发方面，我们通过 D2C 实现了大部分 UI 模块自动开发、通过建设 Eva 互动体系降低互动研发成本、通过 Serverless 的一体化研发提升研发和运维效率，使前端不再成为资源瓶颈。稳定性上，也通过一系列机制和工具体系进行保障。同时增加一块大家平时可能不太关注的资损防控的策略和方案。

D2C 研发提效

去年双 11 我们设立了研发效率专项，核心就是通过设计稿生成代码(Design to Code，D2C)平台 Imgcook 来提升研发效率。最终在去年的双 11 大促会中承接了 78.94% 的新增模块代码自动生成，代码可用率达到 79.34%。

今年前端智能化助力前端研发模式升级，数个 BU 共建前端设计稿识别算法模型和数据集，设计稿生成代码技术体系全面升级，如对 UI 多态、直播视频组件、循环的智能识别增强等。今年双 11 会场承接了 90.4% 的新模块代码智能生成，代码可用率达到 79.26%（对比去年升级设计稿智能检查能力，视觉稿无需人工辅助调整）。得益于 D2C 的研发提效，今年并没有出现往年借调资源投入会场开发的情况。相比传统模块开发模式，使用设计稿生成代码技术后编码效率（模块复杂度和研发耗时比值）提升 68%，固定人力单位时间模块需求吞吐量增加约 1.5 倍。

图：D2C 操作流程

互动研发升级

在电商领域，互动是一个重要的用户增长方案，在提升用户黏性、活跃以及拉新上都发挥着重要的作用。今年双 11，淘系互动团队推出了“超级星秀猫”，我们不盖楼、不开车，全民参与养猫出道，3 只风格各异的萌猫咪一经问世，瞬间俘获了无数消费者的心。通过 EVA 互动体系一整套解决方案，大幅提升研发效率，支撑全民养猫猫在手淘、猫客、支付宝等多个 APP 互通。借助客户端能力及 EVA 互动体系将性能与内存良好控制，让多数用户体验高清稳定的互动，实现 0 故障及秒开，同时星秀猫参与人数再创新高。后续的系列文章将具体阐述淘系互动前端团队是如何做到双 11 互动又快又好又稳，内容涵盖互动基础、EVA 研发体系和全局稳定性方案 3 个方面。

图：互动效果图

Node FaaS 一体研发

Serverless 云+端研发模式通过打通页面代码和服务代码进行一体研发，使得前端可以从前台页面到后端服务完整支撑，节省中间沟通和联调成本。在天猫榜单以及 V 榜的落地，使得双 11 Node FaaS 相关业务整体研发效率提升 38.89%。行业导购双 11 需求也在云+端的新模式下支撑外包快速入场，使得整体提效约 20%。

稳定性保障

稳定性保障贯穿从项目启动到结束的整个双 11 周期，下面从几个重点方面进行简单的介绍：

变化评估：每年的双 11 都是站在巨人的肩膀上，都经过了上一次双 11 的考验。主要的风险就变成新增的部分以及变化的部分，这里的变化既包括技术上的变化也包含人员上的变化。要做到对变化的充分评估，在 99 大促进行验证，并且保证 99 大促后不再进行变化，以一个稳定的状态迎接双 11。

压测：首先要进行流量评估，借鉴去年数据的同时结合今年的变化，进行相应的机器、带宽等资源准备。完成单线路压测，保证在预估流量模型下，自己的服务和上下游都能够运转正常。进行全链路压测，核心验证在 0 点高峰时各业务并发的情况的运转情况，尤其是一些底层公共服务，以及优先级的保障情况。

兜底&预案：兜底一般指在大流量或其他不可控因素的情况下，如何将用户体验和业务损失降低到最小。预案需要评估可能遇到的各种情况，以及对应的处理方案。

验收：功能预演，按照用户的所有使用路径进行操作，目前这个工作仍是人工。时间穿越，将页面和系统的状态都调整为活动时间来验证，需要打通上下游的各个系统并形成联动。机型验收，基本分为高端机、中端机、低端机，分别进行验收，很多业务都需要针对低端机做功能降级。稳定性验收，单独页面的性能和稳定性各自保障，但业务叠加后很可能存在问题，尤其像会场、互动、直播、旗舰店等内存消耗大户，互相之间都有引流，切换后很难保证，需要整体全链路验收。

变更&应急：历次的故障数据表明，大部分的问题都是由于变更导致的，如何做好变更管控尤为重要。根据时间分为弱管控、强管控期；根据业务等级分为集团核心应用、BU 核心应用、非核心应用等；建立变更的 CR 和审批的机制。应急主要指在核心活动期间，问题、舆情、故障等流转机制，针对问题发现、定位问题、修复问题时间作出要求，不同等级如何决策作出安排。

监控：淘系前端持续进行监控能力的建设和升级。需要保障大促高峰的可用性以及报警的实时性，覆盖所有的业务场景。针对越来越复杂的场景，需要端到端的监控和数据分析平台。灰度过程缺少度量和定点监控。根据这些问题和需求，jstracker 提供了安全生产的整体解决方案，打造端到端的前端监控与数据分析平台，打造实时监控、多端覆盖、数据分析、智能化的数据平台。同时根据页面情况、错误日志、源站数据、FaaS 日志等打造了双 11 的前端数据大盘。

资损防控

一直以来前端资损防控是平台非常薄弱的一环，前端触发的资损和舆情问题不在少数。之前全靠开发同学的经验和意识来保证，缺少体系化的资损防控能力。去年开始组织了团队层面的集中筛查和人工预演，对人力和时间的消耗非常巨大，并且很难保证质量并进行积累和沉淀。所以为了能有一种成本更低、预防效果更好的方式进行资损防控，2020 年 S1 伊始，就重点对资防做相关产品化的设计和实现。同时今年也重点增加了商家、运营中后台侧的资损防控。

我们将资损防控氛围了三个阶段，研发阶段、操作阶段、运行阶段。研发阶段给存在资损风险的仓库打标，将常规的价格、优惠、默认文案等 case 进行枚举，通过静态扫描、UI 测试用例扫描等方式进行防控。操作阶段，主要是指商家、运营进行优惠、权益等设置的阶段，通过表达方式统一（避免 5 折、0.5 折造成理解差异）、二次确认、限定边界值、低价预警等进行防控。运行阶段有快照对比、服务端数据对账等方式，运行阶段的防控相对滞后，发现时很大概率已经造成实际的影响。

然而，目前仍是预防为主，不能百分之百保障没有资损故障发生，接下来我们还在构思链路级别的、生产环境上的防控手段，建设一些告警和自动止血为平台保驾护航。

业务价值

做好本职的基础上，我们希望给业务带来增量价值。本章从会场性能优化提升转化、基础链路新方案提升转化、唤端技术定制策略提升精准率、智能 UI 为不同人群提供不通过 UI 提升点击等 4 个方面来介绍。

性能提升

会场是每年双 11 的主角之一，会场的用户体验自然也是每年最关注的点。在日趋复杂的业务需求下，如何保障我们的用户体验不劣化甚至能更优化是永恒的命题。今年分别使用了预渲染方案和 SSR 方案进行优化，首先是重新定义了秒开的标准，从原来的前端时间升级到从用户点击经过跳转到页面可视的时间，增加了客户端路由、webview 启动等时间，使体验的衡量更贴近用户真实的体感。覆盖了包括主会场、行业会场、外投会场等数十个场景。

预渲染

预渲染是在今年双 11 会场中使用的技术方案，用于提升用户打开会场的体验。将原有 H5 页面渲染流程中的 WebView 的初始化、页面资源加载、部分 JS 的执行等耗时的操作，提前执行，在离屏状态下完成页面"渲染"。当用户真正点击进入会场的时候，复用这个提前"渲染"的页面，大大节省打开会场的时间。用户打开会场的整体平均耗时缩短了 200ms~700 ms 左右，秒开率提升 10%-14%。优化对中低端机绝对收益更高，已实现在低端机上实现秒开会场。让用户逛会场体验更流畅了，尤其中低端手机效果更加明显。在后续的文章也会讲述包括预渲染、数据快照、并行请求等性能优化方面的实践与思考。

图：中低端机型预渲染效果对比图

SSR

今年在不改变现有架构，不改变业务的前提下，在会场上使用了 ServerSideRendering 技术，将秒开率提高到了新的高度（82.6%）；在用户体验得到优化的同时，业务指标如点击率等也有明显的增长，带来了不错的业务价值。后续的系列文章汇中会详细介绍前端在解决工程化、业务效果评估上的具体实践与方法论；服务端在解决前端模块代码于服务端执行、隔离和性能优化上的思考和方案。

图：中低端机型 SSR 效果对比图

基础链路

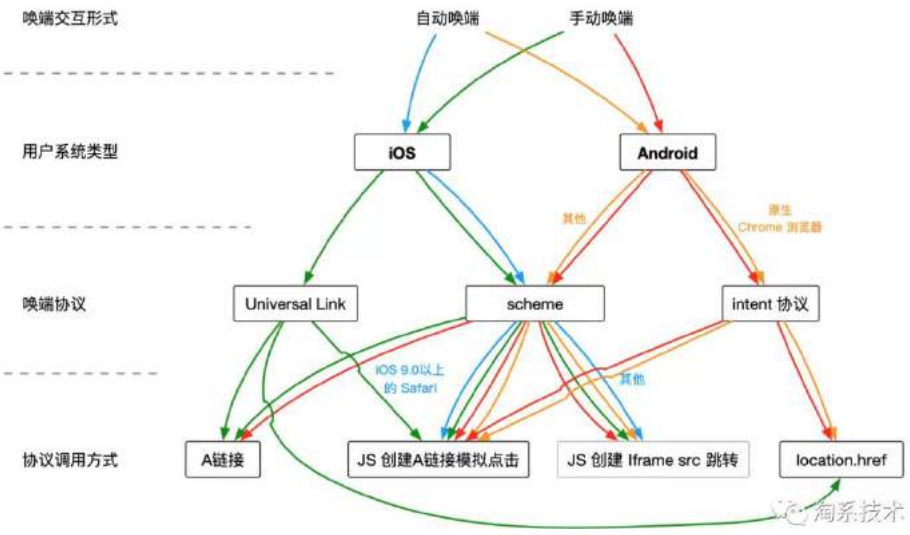
基础链路是电商核心的链路，包含首页、商品详情、微详情、交易（下单、订单、购物车、支付成功）、信息流、我的淘宝等基础业务。现有的技术方案是手淘内使用 Native 版本，追求极致的体验和稳定性；站外流量、包括支付宝在内的阿里系 App 使用 H5 版本，追求灵活性和可用性。随着支付宝容器化体系的完善，在其他 App 中的内聚，基础链路新的容器化版本具备了孵化的土壤；同时 H5 的一些弊端，比如资源都在远端、Native 能力使用限制等也可以得到优化。

借助之前的"新奥创"和"DinamicX"方案（主要解决业务定制以及安卓、iOS、H5 的三端一致，实现一处开发、三端生效），容器化版本得以快速扩展，实现四端一致。性能数据上，加载时间对比 H5 版本有 2s 的提升，基本达成秒开的目标；业务数据上，容器化版本对比 H5 版本 UV 转化率提升 70+%。

目前已覆盖支付宝、特价版、优酷、高德、淘小铺、一淘等 App，以及通过百川 SDK 集成在众多外部媒体 App。业务上也接入了每日必抢、大牌直降、淘宝特价、淘宝直播、百川媒体、优酷、小铺、轻店、花呗等业务。

唤端技术

随着流量见顶、电商大战进一步升级，如何做好用户增长是各大公司必须完成的命题。用户增长涉及的面非常广泛，今年淘系前端聚焦在唤端技术，即外部流量拉起手淘 App 的技术体系。唤端技术的门槛很低，简单到只需要拼一个类似 URL 的 scheme 就可以触发唤端。唤端技术又很复杂，不同的渠道、不同的 OS、不同的 App 都有可能针对唤端协议有限制，并有各种各样的兼容性问题；唤端链路中不同业务可能都有自己的业务定制需求，例如参数的透传；唤端链路的效率更是被关注的核心点，不同场景不同业务在效率上可能都不一样，因此还需要对唤端效果进行监测和对比。为了解决这些复杂的问题，我们在唤端技术上进行了又一次升级，建设了可定制的唤端策略，打造了详细的唤端 AB 测试链路。从本次双 11 的效果看，不同场景下的唤端效率（唤端成功率）相对提升了 25~40%不等。



图：唤端策略图

智能 UI

随着移动互联网和推荐系统的发展，人和商品的精准匹配为业务带来了效率的大幅提升。越来越多的精细化手段逐渐应用于个性化推荐领域，比如场景化推荐、人群定投技术等。同时商品的信息比以往任何时候都要丰富（买家秀，品牌背书，无忧购服务等），不同的用户对于内容的 UI 表达有着差异化的诉求，因此通过为不同人群找到合适的 UI 表达一定能带来业务效果的提升。

项目的最早期，我们通过 AB 实验直接定量测试，明确了相同的 UI 方案在不同的场景会产生差异，不同的 UI 方案在相同场景下也会产生差异。也就是说，针对不同场景使用不同方案是有意义的。2020 年双 11 大促我们第一次大规模采用智能 UI 产品化方案落地了多个前端模块，包括猜你喜欢模块、商品模块、店铺模块等，覆盖了双 11 的预售和正式开卖阶段，承受了流量洪峰的考验，且带来了稳定的增长。覆盖 300 多个会场，最高的会场 PV 点击率相对提升 10%+。

技术升级

伴随业界的技术演进和业务的发展，我们在技术上相比去年也有了新的尝试和迭代升级，其中典型的包括 FaaS 的深度使用、PHA 渐进式的体验增强、边缘节点渲染的应用等。

FaaS

Serverless，一块深水的坚冰，逐步从深海付出了水面，阿里淘系从去年在大促实践开始，逐渐将 Serverless 应用到前端领域方方面面。今年双 11 首先是在覆盖场景方面，FaaS 从淘宝行业拓展到会场和营销业务，业务的复杂度得到极大的丰富。能力进一步提升，支撑的业务量级也从 2k QPS 提升到 5W QPS，CPU 水位从去年的高 QPS 规模时，精力花费降低约 50%。在研发体验方面，打造解决方案体系，单元保障、大促管控、专家系统、函数盘点等能力，运维提效约 50%。在研发体验方面，打造解决方案体系，降低研发门槛，支持外包快速入场。

PHA

PHA 全称 Progressive Hybrid App，是提升 Hybrid 体验的一种应用框架。提升页面加载速度和交互体验的渐进式 Web 应用，使用 PHA 开发的应用本质上没有脱离前端开发和 W3C 标准，但依然拥有原生应用的特性和体验。或许你有想到 PWA，但 PHA 有比 PWA 更强的 UI 能力和更快的加载速度。目前已经在手淘、特价版、Lazada、CBU 等多个客户端落地，支持了 618、双 11 等多个大促。PHA 联合客户端、前端团队、数据分析团队，跨栈协同，在性能优化方向上也做了很多优化工作，梳理全链路性能埋点、定义新的性能口径（从用户点击到可视），使用了预加载、预渲染、资源加速下载、离线资源等优化手段。

ESR

现在的渲染节点主要是在终端或是服务端，对应 CSR（Client Sider Rendering）和 SSR（Server Side Rendering），分别有适用的场景以及优势和弊端。现在借助阿里云的能力可将渲染转移到 CDN 节点，也就是我们要介绍的 ESR（Edge Side Rendering），即能为前端提供渲染能力，同时也能将大量 CDN 机器上的计算资源利用起来。

阿里云推出了 CDN 轻量编程环境——EdgeRoutine，这为我们提供了一个新的尝试方向。我们可以在 CDN 节点去做提前渲染的事情。CDN 的访问策略是会去寻找离用户最近的节点，就像快递运输的最后一公里一样，总会派送到离客户最近的分拨点。这么看来页面的网

络调度时长是非常有优化空间的。并且我们还可以利用 CDN 节点资源共享的特性，将部分数据缓存到 CDN 节点上，减少远程的数据请求。

这套方案对于数据刷新率不高、访问量极大的页面，ESR 搭配 CDN 的缓存能力是非常适合用的。以达人页为例，首屏时间约能提升 50%左右。现在 ER 的技术才刚刚起步，应用场景比较局限，能力上还有很多不足，体系也需要不断地建设，但这个新技术为前端提供了更多可能，需要我们不停的去探索和完善。

双 11 PM 初体验

双 11 作为电商年度最核心的节日，各方面投入的力度和资源都是最大的。作为参加过 8 次双 11 的老兵，作为前端 PM 是第一次，有很多不一样的感受。

复杂：首先是业务上，有双 11 定制和特有的主会场、主互动、猫晚等，还有淘系内部本身就有导购、行业、营销、直播等数十个业务，同时联动支付宝、优酷、本地生活、阿里妈妈、菜鸟等多个集团 BU，与商家、ISV、物流、媒体等的协同和合作。技术上同样复杂，前端的页面从开发、搭建、源站、CDN 的全部链路，以及 Node FaaS 的容器、中间件、容量准备、流量调配、机房部署等。管中窥豹，对于整个体系的认知还需要更进一步的探索。

流程：双 11 作为电商业务每年的大考，已经摸索出一套成熟的流程机制。包括人员的组成、沟通机制、时间排期、组织保障等各个方面都有很细致的机制进行保障。

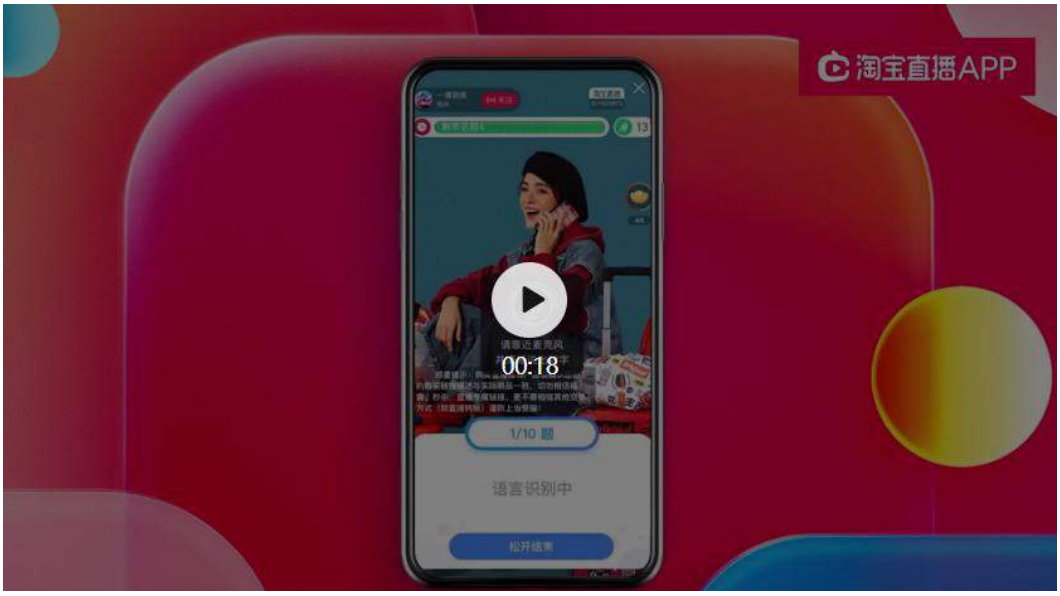
协同：双 11 是非常好的节点，可以让各团队、各岗位、各 BU 之间形成联动，集中力量将如此庞大的体系进一步完善。很多技术的升级和突破都是在双 11 落地并进一步推广的。这次预渲染的方案就是客户端和前端紧密协同，在很短的时间内实现并验证的。

多维：看问题的视角可以更多维，不同技术岗位视角，全链路视角，业务的视角。以一次变更的审批为例，之前更多关注变更的代码实现上，对上下游的影响、对稳定性的影响、对业务的影响、是否引入新的风险、影响的范围等等都需要进行综合衡量，做一个判断往往需要从中进行取舍，而不单单是技术上的 1 和 0。

双 11 总结：淘宝直播“一猜到底”——移动端实时语音识别技术方案及应用

作者 | 端智能语音项目组

出品 | 淘系技术

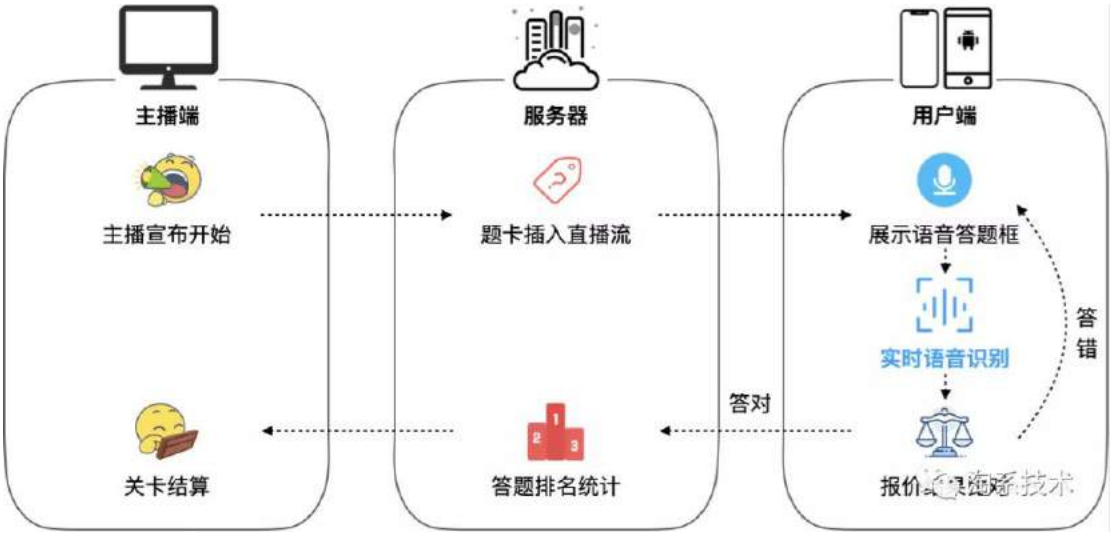


点击可跳转原文查看视频详情

过去一年淘宝直播快速发展，截止 2020 年 9 月底，80 个淘宝直播产业基地在全国落地开花，从农村走出 10 万农民主播，直播真正意义上成为帮助商家和消费者完成交易的利器，同时通过各种互动玩法让直播购物变得有趣好玩。在 2020 年双 11 开始阶段，淘宝直播 App 升级了 18 年直播答题「点题成金」的玩法，推出「一猜到底」新玩法。如果说传统的直播答题是「选择题」，一猜到底的玩法更像是几万人同时在线的「抢答题」，将答题方式从文字选择升级成语音抢答，给出猜中价格高低提示，让用户增加了更多的参与的乐趣。

为了实现比肩综艺现场的直播竞猜体验，我们一次压上了由达摩院语音实验室、阿里云 PAI 团队、淘系技术直播 App 和端智能 MNN 团队组成的全明星阵容，通力协作之下，一举实现了工业界首个用于直播的移动端语音识别。

业务流程和技术挑战



「一猜到底」整体玩法链路如上图所示，主播口播开始后，用户需要在人数和时间未满前，按住按钮，通过语音报出价格，系统通过本地语音识别能力进行识别和结果比对，提示用户所报价格"过高"还是"过低"，直到答对或者超时结束。在每一关有限的作答时间内，用户往往需要多次竞答，才能逼近商品的真实价格。于是，实时语音识别能不能准确且快速地识别用户的报价，直接决定了「一猜到底」的成败。

不同于一般的语音识别应用，一场顶流的淘宝直播，可以聚集百万乃至千万的用户围观。这么多用户同时进行语音识别，会出现非常多的请求，如果采用云端识别对服务压力和服务质量都有非常大的挑战。项目开始时实验了端侧和云侧识别的两种方案，发现云侧方案难以支撑这样的活动，最终选择了端侧方案，确定端侧识别方案之后，发现也不是康庄大道，主要存在以下技术难点：

- **高精度高性能的本地语音识别**

目前行业比较成熟的是服务端的语音识别方案，完全照搬服务端方案到移动端也不现实，需要创建一套适合移动端运行的语音识别方案。同时，直播场景下的语音答题噪声较大，对语音识别的准确度要求较高，语音识别速度也会对用户的答题速度造成巨大影响。

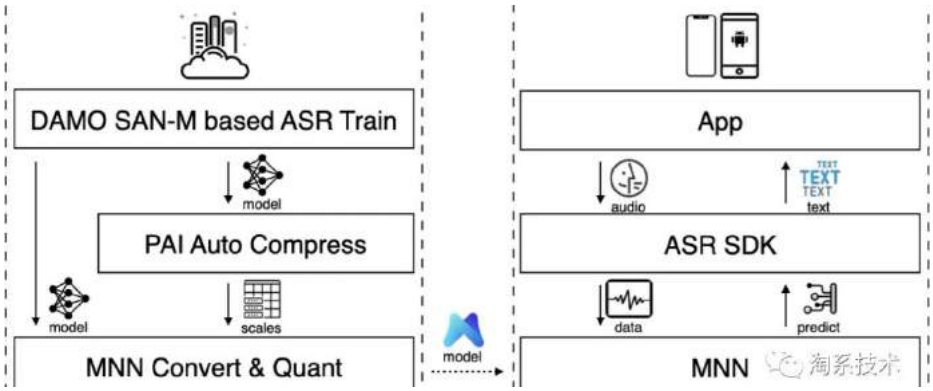
- **语音模型和资源包体积过大**

考虑到活动特性，端侧的语音识别引擎需要内置在包内，而且越小越好。经过客户端研发评估，如何做到 15MB 以内甚至更小的语音模型是关键，因此需要极致的模型压缩能力支持。

- **端侧资源有限，性能压力大**

直播场景本身就已经很占用资源，叠加直播场景下做语音识别，对语音识别过程中的 CPU、内存占用，都有很大的要求，高性能的推理和优化成为模型落地的最大拦路虎。

移动端实时语音识别技术大揭秘

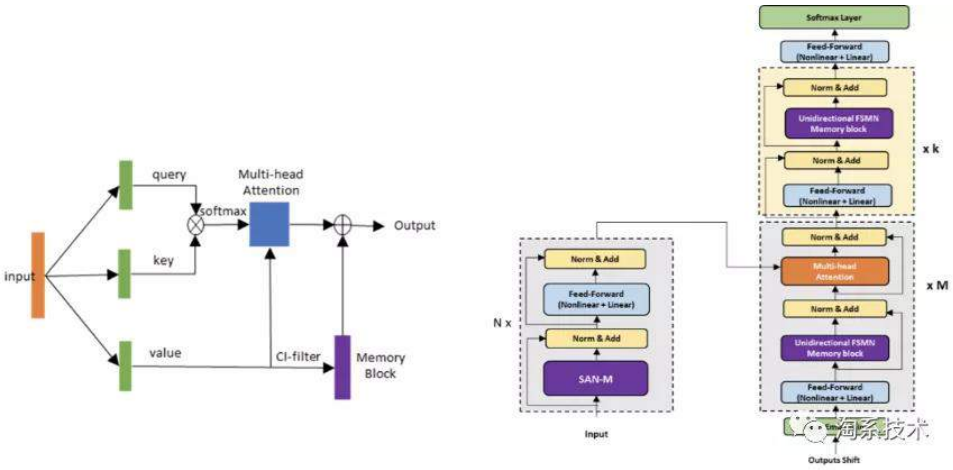


阿里达摩院语音实验室早在 2015 年就研发出了第一代移动端离线语音识别方案，近来结合 PAI 模型压缩、MNN 高性能推理引擎，实现了移动端离线和流式端到端语音识别方案，满足语音指令、语音识别、实时翻译等场景需求。根据「一猜到底」项目需求，我们选取"基于 SAN-M 的离线端到端语音识别"方案，通过极致的模型压缩和性能优化，最终实现模型大小小于 15MB、内存占用低于 60MB、1s 语料识别快于 50ms 的高性能方案。

基于 SAN-M 的离线端到端语音识别

目前，最具代表性的离线端到端语音识别模型 LAS[1]和 Transformer[2]都是基于 Attention-Encoder-Decoder 的。LAS 采用基于 BLSTM 的 Encoder 和基于单向 LSTM 的 Decoder；而 Transformer 则采用 Multi-head Self-Attention 模块组建 Encoder 网络，采用 Masked Multi-head Self-Attention 组建 Decoder 网络。

在公开评测任务集上，Transformer 较 LAS 在性能上有优势，同时由于采用了 Multi-head，训练并行化效率更高。我们分析了 Self-Attention 和 DFSMN memory block[3,4]之间的关联性：Self-Attention 可以理解为采用了 context-dependent 系数进行全局建模，而 DFSMN 的 memory block 则采用了 context-independent 系数进行局部建模。对于语音识别，局部声学建模和全局语义建模都非常重要，因此我们提出了如下图所示的 SAN-M 模型结构，高效地融合了 Self-Attention 和 DFSMN memory block。

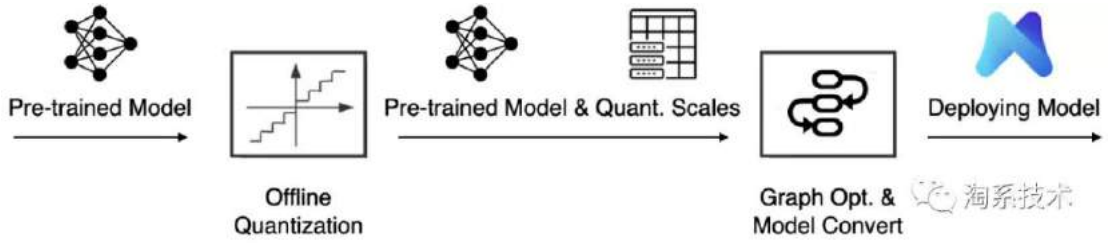


SAN-M 模块如上左图所示，将 Self-Attention 和 DFSMN memory block 融合一个模块，有效的结合了 Self-Attention 的全局长时建模能力和 memory block 的局部长时建模能力。基于 SAN-M 模块构建了如上右图的 Encoder-Decoder 离线语音识别系统（SAN-M-E2E-ASR），并在开源的 1000 小时 AISHELL-2 中文识别任务中获得了当前该任务的最优性能（CER=5.61%）；在工业量级的 2 万小时中文识别任务中，该系统也显著优于我们之前线上的 CTC 系统和标准 Transformer 系统。

针对本次识别场景，我们最终实现了不到 40MB 的端到端模型，而识别性能则可以媲美上一代整体超过 100GB 大小的云端 DFSMN-CTC 系统。我们在 finetune 数据上进行了不同维度的挑选和搭配，并做了不同策略的数据扩增来覆盖多样的识别情况。针对模型输出的 token，也进行了一定压缩，并拉低了与本次任务无关的 token 概率来降低误识别率。在 ITN 模块，我们采用精小的 FST（Finite State Transducer）来实现规则网络，用状态转移来实现文字到阿拉伯数字的转换，通过边上权重来控制其转换方向，并在简略读法、谐音、容错上也做了一系列路径优化。

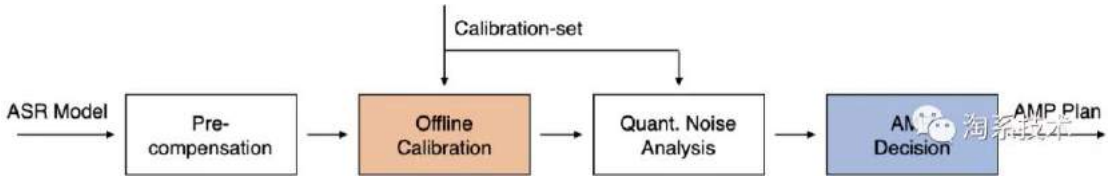
基于 PAI-MNN 云端一体化模型压缩

虽然达摩院语音实验室通过定制化语音识别模型设计，将原有的 170MB 模型裁剪至不到 40MB，但是考虑到移动端的资源情况，我们还需要通过 PAI-MNN 云端一体化模型压缩方案，进一步将模型基本无损地压缩到 15MB 以内。



从训练、模型压缩到优化部署的 PAI-MNN 云端一体方案

PAI 混合精度量化流程



PAI 混合精度量化流程

上图显示了 PAI 团队 (PAI: Platform of A. I. in Alibaba)研发的无数据标注干预的自动混合精度量化流程（Label-free AMP Pipeline, AMP: Automatic Mixed Precision），包括量化误差预补偿、离线标定、量化噪声分析与混合精度决策四个阶段，主要创新点包括：

- **支持端到端 Transformer 的离线后量化：**
  - PAI 团队的后量化方法，引入了循环张量探针，以支持端到端 Transformer 的离线后量化。
  - 相比于拆图量化、量化训练等，端到端后量化具备快捷、高效的优势。

- **集成了丰富的后量化策略，为后量化的精度鲁棒性提供了坚实保证，基本策略包括：**
  - KL 算法的改进，能够有效减少输入/输出张量的量化噪声；
  - EasyQuant（参考文献 [5]）的使用，可进一步减少输入/输出张量的量化误差，尤其能改善 INT7 等更低精度量化的效果；
  - Bias Correction（参考文献 [6]）通过补偿网络权重的量化偏差（均值与方差的偏差），以减少权重量化噪声；同时对 Bias Correction 的适当改进，增强了对 SAN-M ASR 模型的补偿效果；
  - ADMM（参考文献 [7]）亦可优化权重量化参数，减少权重量化噪声；也适当改进了 ADMM 的使用，从而在交替方向迭代范围内，确保权重量化误差最小；
  - Weight Adjustment（参考文献 [8]）在 Kernel weight 按 Per-tensor 量化时，通过 Per-channel 形式的等价均衡变换，可以减少 Weight 量化误差。

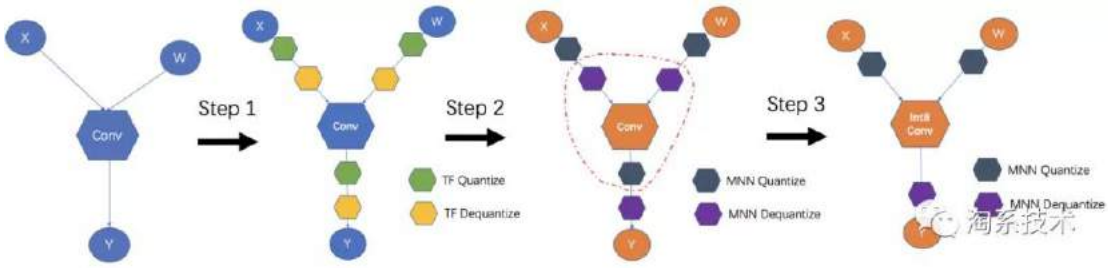
- **无 Label 干预的混合精度量化流程：**
  - 该流程从模型输入到混合精度决策，无需数据标注（Label）的干预，简洁易用、快捷有效；
  - 量化误差按逐层统计，且能准确反映每个网络层的量化敏感度，为混合精度（INT8/FP32 混合）决策提供了可靠基础；
  - 通过控制回退的网络层数，可选择出精度与模型容量折中最佳的帕累托最优解，完成多目标优化；
  - 生成的混合精度量化表，能够对接移动端推理框架 MNN，以生成低延迟、高推理精度的运行时推理引擎；从而构成了完整的工具链路，即从混合精度量化、到移动端的推理部署；
  - AMP Pipeline 不仅适用于移动端，也适用于 CPU/GPU 优化部署，体现了 PAI 云端一体的优势所在。

基于 PAI AMP Pipeline，有效实现了 SAN-M 模型的离线后量化（PTQ: Post-training Quantization）。为了保持算法模型识别精度，经 AMP INT8 量化之后（回退 3 个 Op，分类层保留为 FP32 实现）。

为了解决压缩率的问题，MNN 模型转换和优化工具对回退的算子统一使用权重 8bit 存储、float 计算的方式进行优化，进一步压缩模型大小。通过一套统一格式的模型压缩文件，经过 PAI AMC 优化的模型可以顺滑无缝地转换到 MNN 的格式。

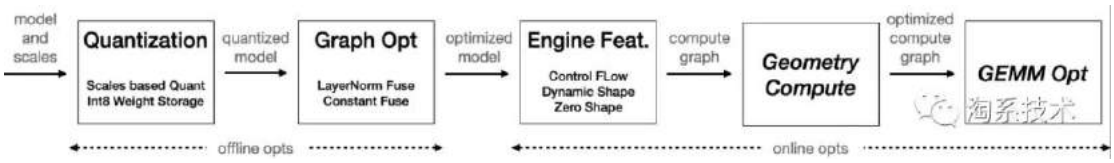
MNN 模型转换工具基于现有的图优化流程，根据该模型压缩文件将 float 模型转换成 MNN 模型的同时完成离线量化，具体过程如下：

- 根据量化表中提供的 tensor name，在 TensorFlow 的计算图中生产和消费该 tensor 的边上同时插入一个自定义的量化和反量化算子。
- 将 TensorFlow 的计算图转换成 MNN 的计算图，其中自定义的量化和反量化算子转换成 MNN 量化（FloatToInt8）和反量化（Int8ToFloat）算子。
- 算子融合：将支持量化的算子、输入的反量化算子和输出的量化算子融合成一个 Int8 的算子。
- 最后消除成对的 MNN 量化和反量化算子。



最终，SAN-M 模型在众包测试集上的 WER 绝对损失低于 0.1%、SER 绝对损失低于 0.5%、理论压缩比约为 3.19 倍。

基于 MNN 推理引擎的实时高性能计算



为了在移动端上实现实时的端到端语音识别模型推理计算，MNN 在全链路上做了诸多优化。

端到端语音识别模型基于 Transformer 结构，包含一个对输入音频特征编码的 Encoder 和一个自回归解码的 Decoder。这类模型结构要求 MNN 支持 Control Flow、Dynamic Shape 和 Zero Shape 等特性，因此，MNN 首先在框架层面对这些特性进行了支持和完善：

- MNN 重构了 Control Flow 支持方案，提供用户透明的 functional control flow 实现，并支持了 TensorFlow 1.x 的控制流模型转换，为用户提供一站式的部署体验。
- 对于 Dynamic Shape 的支持，MNN 将整图按照动态形状算子划分为多个分段子图。在代码层面，一个子图对应一个 Module，Module 支持嵌套，即整图被表达为一个由 Module 组成的调用树，树的每个叶子节点可以使用一个 Session 来执行，Session 每次执行前 resize，重新进行 shape 推理和分配内存。
- Zero Shape 指的是模型中某些 Tensor 的 shape 存在 0 值，比如（1，0，256），这种情况大多是为了给 while-loop 中某些循环变量提供初始值而引入的。MNN 在形状推理和执行逻辑上对 Zero Shape 进行了支持。

之后，MNN 根据达摩院模型新增了 LayerNorm Fuse、Constant Folding、重复 Reshape 算子消除等图优化方法。图优化之后的计算图更容易和其他优化方法组合使用，比如，Constant Folding 后 MatMul 的一个输入可能被替换成一个 Constant 节点，因此就可以转换成 FullyConnected 或 Conv1x1 进行加速，并且也更容易利用模型压缩方法对权重进行量化。

而后，语音模型的耗时重点仍然是矩阵乘法。MNN 通过更优矩阵乘分块、基于 NC4HW4 布局优化前后内存布局转化、Strassen 算法改进等策略，优化了整体的卷积和矩阵乘的性能，ARM 架构上性能提高了 10%-20% ，保障了语音模型的高效运行。

同时，MNN 最新提出的几何计算机制也在实时语音识别起到了重要作用。几何计算是 MNN 为了解决设备碎片化问题而提出的一种新机制，其核心在于把坐标映射标准化，以便统一实现与优化。在几何计算的支持下，我们可以较简单地合并相邻的纯形变算子，从而降

低访存需求，提升模型运行性能。

最后，在 PAI-MNN 云端一体化模型压缩的加持下，我们利用量化表和有限回退机制，在精度损失可控的前提下，进一步降低了移动端上的计算总量。

RTF（real time factor），即实时率，表示识别一秒钟音频需要的耗时。

	模型大小	算子数量	RTF	内存占用
优化前	35MB	2341	0.04048	233MB
优化后	13MB	1623	0.01852	54MB

在这一系列组合拳之下，我们才最终在目标设备上，将 RTF（real time factor）降低到了目标值 0.02 以下，从而实现实时语音识别，让「一猜到底」得以走到每一个用户的面前。

总结与展望

通过这次项目合作，基于高性能推理引擎 MNN，结合一流的语音模型设计和模型压缩技术，我们已经能在移动端上实现实时的语音识别，并通过了双 11 核心场景的考验。除了上述离线端到端语音识别之外，我们还实现了更复杂的流式端到端语音识别，能够低延迟地流式输出识别结果，可以应用到语音实时翻译等场景。

在硬件算力、模型设计、模型压缩和推理引擎飞速发展的共同推动下，CV、Data 的 AI 应用场景已经日趋成熟，ASR、NLP 的规模化应用也已指日可待，端侧 AI 的应用场景仍在持续发酵。

淘系技术部端智能团队，基于淘系丰富业务场景，持续进行端侧 AI 技术建设和业务创新实践。除开源推理引擎 MNN 之外，我们将上百次 AI 应用开发实践中积累的经验沉淀为 MNN 工作台。MNN 工作台将极大降低 AI 应用门槛，将 AI 研发的效率提升数十倍，让"技术小白"也能快速上手，轻松设计自己的 AI 应用。MNN 工作台已于近日正式对公众开放，

赶快进入 MNN 官网下载体验吧。

MNN 官网：http://www.mnn.zone

参考：

[1] Chan W, Jaitly N, Le Q, et al. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition[C]//2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2016: 4960-4964.

[2] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.

[3] Zhang S, Lei M, Yan Z, et al. Deep-fsmn for large vocabulary continuous speech recognition[C]//2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018: 5869-5873.

[4] Zhang S, Lei M, Liu Y, et al. Investigation of modeling units for mandarin speech recognition using dfsmn-ctc-smbr[C]//ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019: 7085-7089.

[5] Di Wu, Qi Tang, Yongle Zhao, Ming Zhang, Ying Fu, Debing Zhang, "EasyQuant: Post-training Quantization via Scale Optimization", arXiv preprint 2006.16669, 2020.

[6] Ron Banner, Yury Nahshan, Elad Hoffer, Daniel Soudry, "Post-training 4-bit quantization of convolution networks for rapid-deployment", arXiv preprint 1810.05723, 2018.

[7] Cong Leng, Hao Li, Shenghuo Zhu, Rong Jin, "Extremely Low Bit Neural Network: Squeeze the Last Bit Out with ADMM", arXiv preprint 1707.09870, 2017.

[8] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, Max Welling, "Data-Free Quantization Through Weight Equalization and Bias Correction", arXiv preprint 1906.04721, 2019.

双 11 总结：阿里亿级长连网关的云原生演进之路

作者 | 邓波(光锥)
出品 | 淘系技术

AServer 接入网关承载整个阿里集团的入口流量，负责亿级用户的长链保活，支持上万路由策略转发，是连接上亿用户与后端几十万服务节点的桥梁，在今年双十一需要支撑亿级在线用户、千万级 QPS、生效上万条 API 管控策略，做到了安全可靠的转发路由，并保障了用户体验如丝般顺滑。
在大规模业务流量与管控支撑的背后，需要对系统每一个细节的精确把控，消除每一个潜在的风险点。
借助云原生架构可以极大地简化运维操作，降低了潜在的风险，今年双十一阿里 AServer 接入网关上千台规模的 Pod 平稳扛过峰值。本文主要介绍阿里 AServer 接入网关如何从上一代架构拥抱变化，全面云原生的演进之路。

架构演进背景

每年双十一大促都是对阿里所有服务最严峻的考验，尤其对 AServer 接入网关来说，作为阿里集团第一道门户，需要抵御大促峰值带来的流量洪峰，清洗攻击流量，所需集群规模巨大。

巨大集群规模，以及对机器性能极致要求，导致了运维上的复杂性；随着接入业务的增多，所支持的业务场景扩宽，业务对路由策略的灵活性、生效的实时性要求变高，对路由策略的动态编排能力有着强烈诉求；由于业务的多样性，业务线不同封网节奏，以及故障隔离性，衍生出对流量隔离的稳定性诉求。

运维的复杂性、动态编排的诉求、流量隔离以及对性能的极致要求，推动着 AServer 接入网关不断演变和成长，紧跟业务的发展步伐的同时，逐步降低运维成本的，增强系统稳定性，

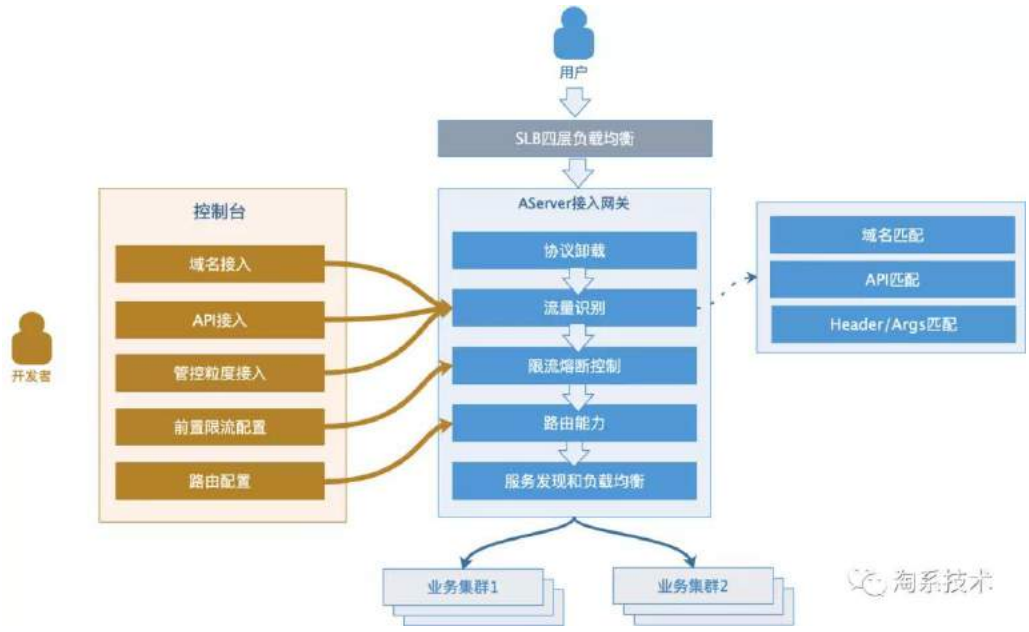
能够一次又一次经受住双十一的考验。

业务背景

作为阿里集团 AServer 接入网关，承载整个阿里集团入口流量，最开始支持域名转发策略的 tengine 网关，根据域名 转发到后端不同服务，业务形态相对简洁。

来到 All in 无线时代，为优化手机端侧的用户体验，同时降级服务端人员的开发成本，集团自研了 MTOP（Mobile Taobao Open Platform）API 网关，为客户端和服务端提供了一致的 API 平台，相同的域名，仅通过 URI 携带的 API 信息转发到对应业务，接入网关需要支持按照 API（通过 URI 区分）的路由转发能力，几年时间迅速增加到数万规则。

随着业务发展越来越精细化，期望对同一 API 下的不同业务场景进行细分，如针对双十一大促会场的来源，手淘、支付宝、其他外投页面等场景进行更精细化控制，为适应业务发展，网关需要支持精细化的管控能力，根据业务请求参数、请求头进行管控和分流。每一个请求都要从上万灵活的配置规则中匹配出唯一的路径，同时又要保持极高的性能，是一件极具挑战性的事情。



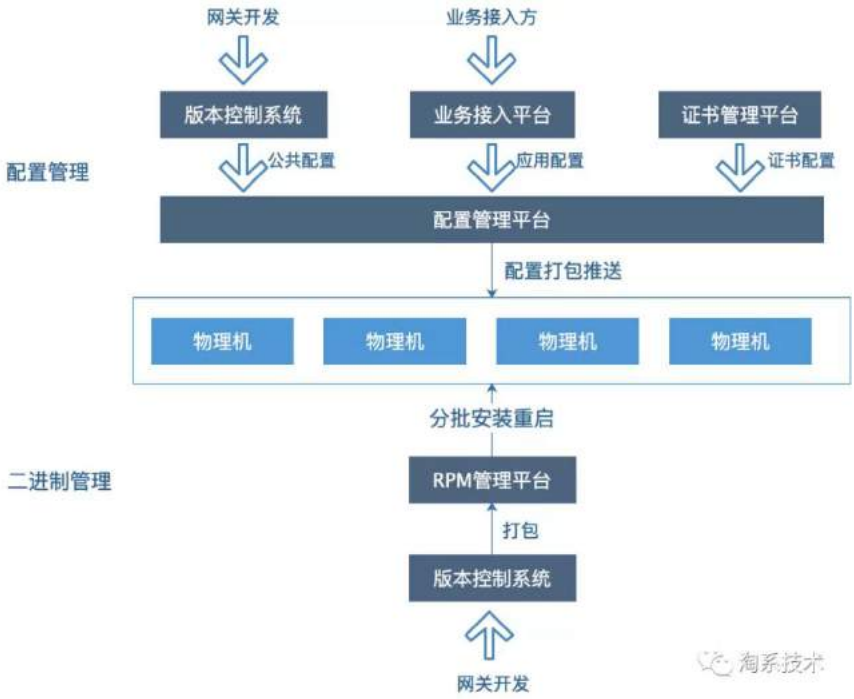
业务模型图

运维体系背景

最开始基础配套的基础设施并不完善，网关层基于 tengine 搭建，最简单快速的方案便是使用物理机，部署进程和配置即可完成服务搭建。随着业务增长，配置管理就成为瓶颈，网关层需要一个强有力的配置管理平台，通过标准化的方式生成业务配置，配套自研的配置管理平台把配置划分为应用配置、公共配置管理、证书配置三部分。

- **公共配置**：通过 Git 版本管理方式生成 tengine 运行的基本配置，如启用模块配置，tengine 运行逻辑配置。
- **应用配置**：通过标准的模板生成业务所需的 tengine 配置。
- **证书配置**：由于证书存在有效期，为了防止过期时忘记更新，还承担了证书自动更新任务。

最初的系统部署架构：



该方案可以实现业务自助接入，通过配置管理平台的模板生成 tengine 配置，再由定时推送到网关机器并进行进程的 reload，生效配置。

通过这种运维方式，不依赖基础设施，可以快速演进，但随着业务增长以及集群规模的上涨，物理机的运维方式弊端逐步显现，野蛮生长的时代过去，作为阿里服务入口，稳定性成为了重中之重，物理机的二进制发布依赖人工部署，需批量执行命令安装 rpm 包，并且分批 restart 进程，这一切都是黑屏操作完成。

此种运维方式显然无法满足现在的稳定性需求，通过手工发布，极易出现误操作导致系统性故障。另外物理机运维很难进行一致性保障，包括二进制的一致性，机器本身环境的一致性检查（如内核参数等），过去的手工运维方式显然已经跟不上时代的步伐。

解决发布和环境一致性问题的最优方案便是容器化技术，随着集团基础设施的完善，接入网关容器化改造扫除了障碍，把不变量（系统配置、二进制）打包成一体进行发布，把变量（应用配置、公共配置、证书）继续沿用配置管理平台进行管理，配合容器化技术进行调整。

容器化改造后的发布和配置变更流程：

容器化架构，简化了建站、扩缩容操作，发布效率有了极大的提升，增加审批流程，系统化卡点，避免了人为操作可能导致故障，发布流程还可对接监控系统，自动告警并暂停发布。

核心问题

随着电商业务发展越来越快，规模化达到瓶颈以后，业务就会有更多的横向扩展，精细化程度越来越高，迭代速度也随之变高，网关层适应业务的变化的成本也来越高，由此带来的核心问题：

- **运维操作复杂性**：由于对性能的极致要求，网关集群对机器有着特殊的要求；由于网关配置管理的特殊性，导致了运维操作的复杂性；特殊性的存在无法很好对接集团现有运维体系，需要进行运维体系的升级；
- **动态编排能力不足**：随着接入业务的增多，所支持的业务场景扩宽，业务对路由策略的灵活性、实时性要求越来越高，静态配置不论生效的实时性还是策略灵活性都难以满足业务发展需求，需要支持路由策略的动态编排能力；
- **流量隔离成本较高**：缺乏轻量级业务范围隔离能力，通过新建集群实现的成本过高，为支持业务线不同封网节奏，以及支持故障隔离性，需要轻量级的多集群流量隔离方案。

云原生近年来的飞速发展，也为网关层提供了更好的架构选择。

云原生架构

为解决接入网关现存问题，结合集团业务场景以及云原生的开源体系，开启了 AServer 接入网关的云原生演进之路，为了分步验证，分解三个阶段逐步实现：运维体系升级，服务治理&网关 Mesh 化，南北向架构拆分。接下来对每一个步骤进行详细的演进说明。

运维体系升级

待解决问题

- 通过容器化升级部署，极大的简化了部署运维方式，能够解决当时最突出的问题，但仅仅改造部署方式还远远不够。

- 由于接入网关有着特殊性（如需要对接配置管理平台，有着大量的 VIP 需求），无法直接对接集团的基础设施，开发了独立的定制化化的运维工具，扩缩容流程需要多个基础组件通过非标接口配合进行，极大的影响了运维产品的迭代效率。
- 故障机置换机器等操作，依赖外部系统轮询检测，并且集团基础设置系统跟定制运维平台对接才能处理，有较大时延。
- 运维操作脱离集团运维体系。

演进思路

随着集团内针对云原生应用设计的统一基础设施 ASI（Alibaba Serverless infrastructure）的逐步完善，提供了基于原生 K8S API 的完整云原生技术栈支持。

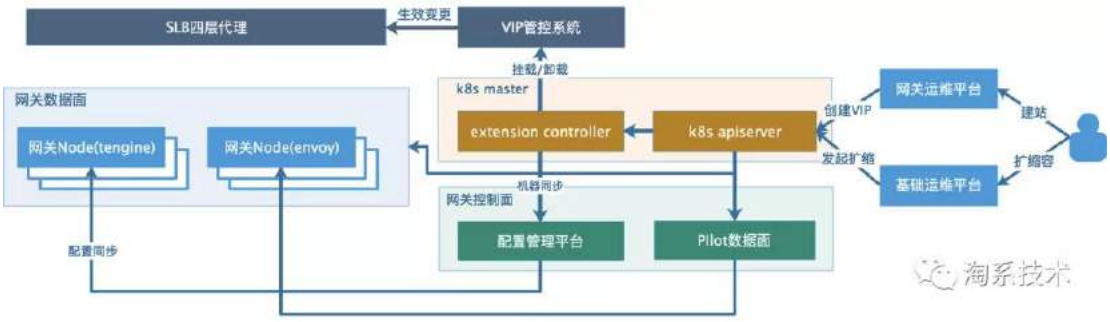
云原生方案可编排能力很强，通过自定义实现 k8s 扩展，非常容易抹平网关层的特殊性，ASI 原有的自动化运维手段，可直接应用于网关层。

网关层对机型的特殊性，可以通过节点池划分来实现，网关机器节点池可自定义机型以及内核参数，消除网关运维上的特殊性，统一管理运维。

演进方案

通过 k8s 自身的 Controller 扩展能力，自定义容器编排，在扩缩容时可以监听 Pod 变更事件对配置管理平台进行机器增删操作，同时也可以挂载/卸载 VIP，抹平运维上的特殊性，并且所有资源都通过声明式 API 定义，方便运维。

对于网关运维，还需要保留一个非常简单的运维平台，仅做建站之用，对比普通应用，网关建站需要在对应区域创建 VIP，进行域名绑定等操作，轻量且易维护：



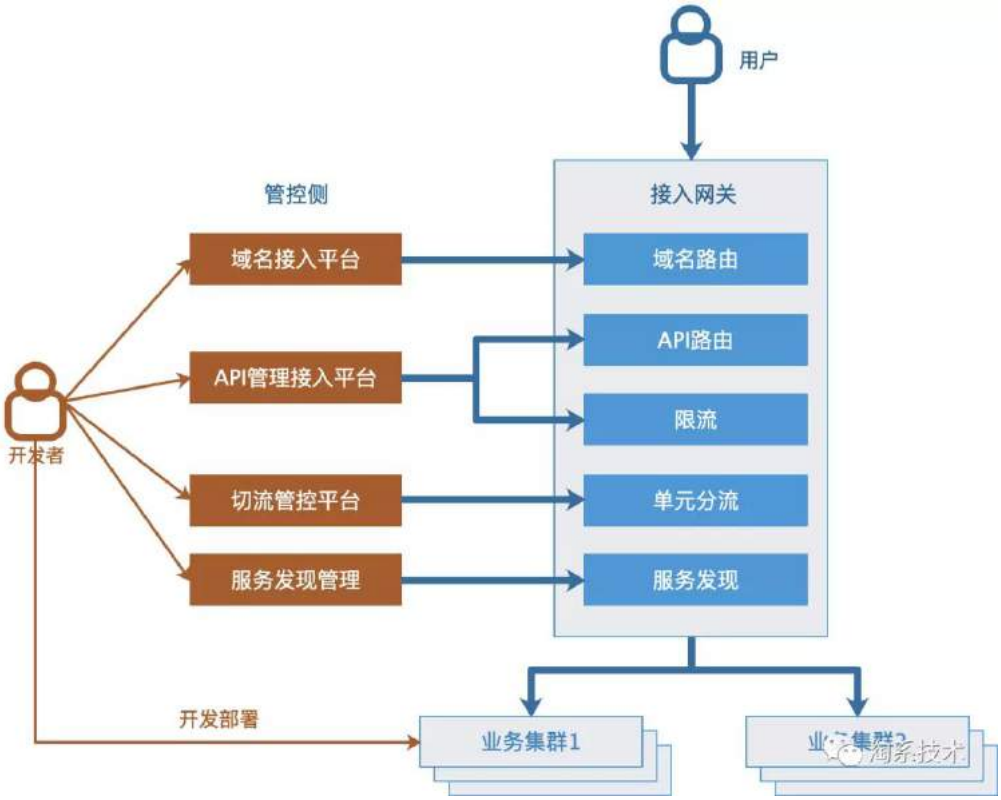
通过进行 ASI 化改造，使得接入网关的运维融入集团 ASI 云原生体系（提升交付效率，去除特殊化运维），通用能力下沉至 ASI 和基础系统，同时具备了风险隔离、自恢复、弹性能力

- **风险隔离**：使用 Sidecar 能力隔离安全和工程能力，避免二者的互相干扰，安全能力出现异常，只影响流量清洗，降级安全能力后，不至于服务整体不可用；
- **自恢复**：对于容器自愈能力，原有容器化方式依赖外部应用的轮询检测，不论是准确性和实时性达都有所欠缺，升级 ASI 后，通过容器自身的检测，可在 3-5 分钟内识别并置换故障容器；
- **弹性能力**：自动弹性能力，通过 ASI 的改造，各系统的对接方式可以使用标准的声明式 API，整合集团内各组件，极大的简化了扩缩容操作，为自动弹性提供了支持；
- **屏蔽机型差异**：通过节点池划分，针对网关应用可使用特殊的机型，底层配置屏蔽差异，无需特殊操作。

服务治理&网关 Mesh 化

待解决问题

随着网关层接入的业务类型增多，需要支持上万条 API 路由规则，而且路由策略也越来越精细化，使用 tengine 原生能力无法满足业务需求。通过定制开发 tengine 模块，非标的定义方式，过去几年中可以很好适应业务的发展，但随着业务诉求愈发精细化，定制开发 tengine 模块的成本也逐步变大



原有架构

- 路由配置通过模块配置+原生配置组合而成，多个模块配置共同决定路由策略，分散的配置无法对一个请求完整的路由路径的识别；
- 通过功能模块划分，想要按照业务粒度的进行增量更新较难实现；
- 基于 tengine 架构，动态变更能力不足，域名变更每日定时推送配置并生效，无法满足业务快速迭代需求；
- 非标准协议直接跟不同管控平台直接对接，对接成本较高，并且不容易收口管控；
- 对于不同业务线（如淘系、优酷），要做到资源隔离，由于多数模块配置使用静态的公共配置，建站成本较高。

演进思路

如何动态编排、精细化的控制路由策略，是在云原生体系下首要考虑的问题。参考对比业界网关层做法，如 Kong，Ambassador 等，主流网关数据面实现都是基于 nginx 或者 envoy，不同产品的扩展性、动态编排能力、成熟度的对比情况：

	数据面实现	框架扩展性	动态编排能力	开源	成熟度
Istio	Envoy	扩展性强，自定义filter实现	强：通过xDS协议动态下发按配置	开源	社区成熟，应用广泛，但聚焦于东西向路由
Kong	Nginx	扩展性强，nginx-lua实现	强：通过API动态添加服务和路由	区分社区版和企业版	社区成熟
Ambassador	Envoy	企业版本支持扩展	强：通过xDS协议动态下发按配置	区分开源版本、社区版本、企业版本，开源版本功能较少	商业公司支持
Nginx Unit	Nginx	弱	强：通过REST API配置	开源	无控制面实现，私有协议，周边配套较弱
nginMesh	Nginx	弱	弱	开源	已不在积极开

淘系技术

从动态性、标准性、性能方面综合考虑，使用 envoy 作为数据面更适合云原生演进方向：

- **动态性和灵活性**
  - envoy 实现的标准 xDS 协议足够灵活，并且可全部动态配置和变更。
  - envoy 扩展性足够好，可通过实现 filter 扩展实现集团内特有的路由逻辑。

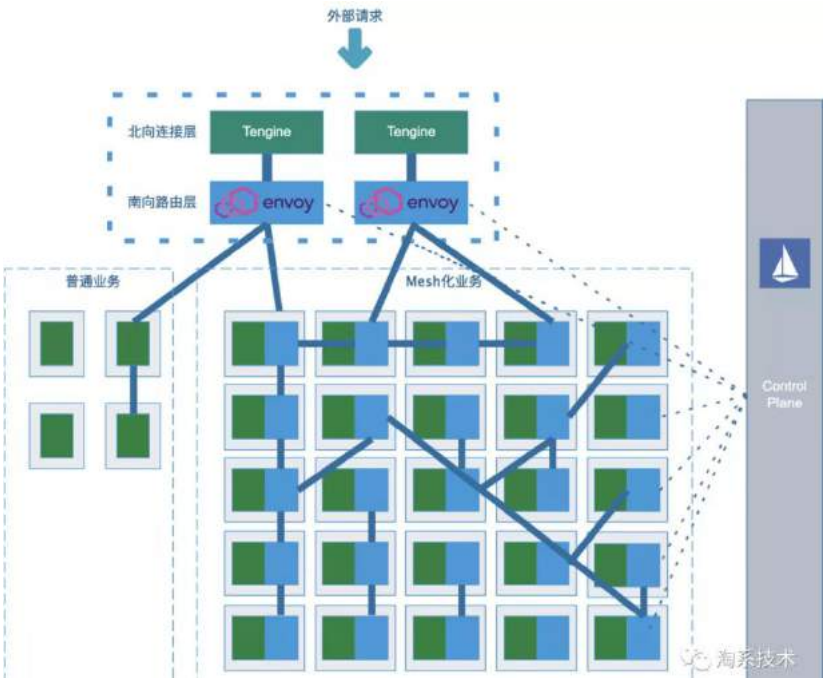
- **标准性**
  - istio 标准组件，社区支持力度大，发展迅速。
  - 阿里集团 mesh 化使用 istio 技术方案，使用 envoy 作为数据面选项能够跟集团业务管控的统一化。

- **性能**
  - C++实现，性能足够好，而开发效率比 tengine 高。

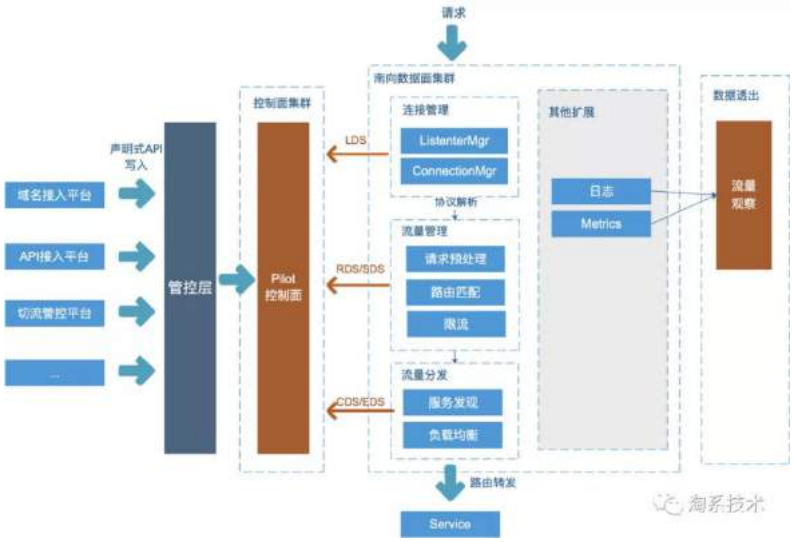
而 envoy 不足之处在于作为 istio 标准组件，东西向路由能力较强，作为南北向需要进行一定性能和稳定性优化，但长远来看，动态性和标准性更为重要。

演进方案

复用集团 Pilot 作为统一的控制面组件，实现网关自身的 Mesh 化：



控制面为提供各透出的业务产品写入，需提供一层管控逻辑进行权限的收口，各产品通过 k8s 声明式 api 写入路由策略，再由 Pilot 控制面转换为 xDS 数据面协议，实时同步给数据面 Envoy，南向路由网关的实现架构：



由于集团的配置规模较大，数十万的路由规则和数千应用，几十万业务节点，开源体系鲜有如此规模。Pilot + Envoy 方案应用于南北向网关后，需要对原生组件做一定的优化和定制，以解决规模化引起的性能和稳定性问题：

- **Pilot 支持 SRDS 协议**：解决大规模 API 配置导致的线性匹配性能问题。
- **增量配置更新**：实现并完善控制面增量更新能力，避免全量更新导致变更半径扩大风险。
- **节点变更优化**：解决几十万业务节点的状态变更对控制面和数据面性能影响。
- **扩展定制**：针对集团特有的路由规则，定制 filter 实现。

通过对开源体系进行定制和优化，可以很好的对接集团内的需求，通过灵活的配置组合，通过快速迭代控制面透传的能力，实现集团内不同业务的特殊需求。

南北向拆分

待解决问题

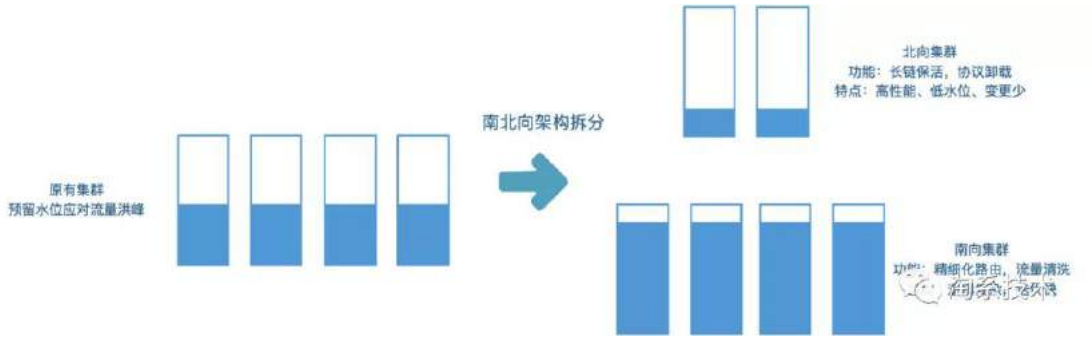
网关作为用户跟业务的桥梁，对用户端保活长链，协议优化，让用户尽可能快速稳定的连到集团；对业务支持灵活的路由和熔断限流策略，负载均衡。虽然连接保活跟路由转发作为网关的整体能力透出，但二者的迭代效率的诉求，以及业务特点都有较大差异。

在一些大促活动场景，即使有预期外的流量洪峰，网关层作为保护业务服务的屏障，仍然可以做到稳如磐石，依赖于高性能和水位的预留。考虑到保活长链，协议优化有这较长的迭代周期，性能极高；路由转发和流量清洗由于策略灵活复杂，资源消耗自然相对较高，如把二者进行架构拆分，可以极大的提升整体资源的使用率。

演进思路和方案

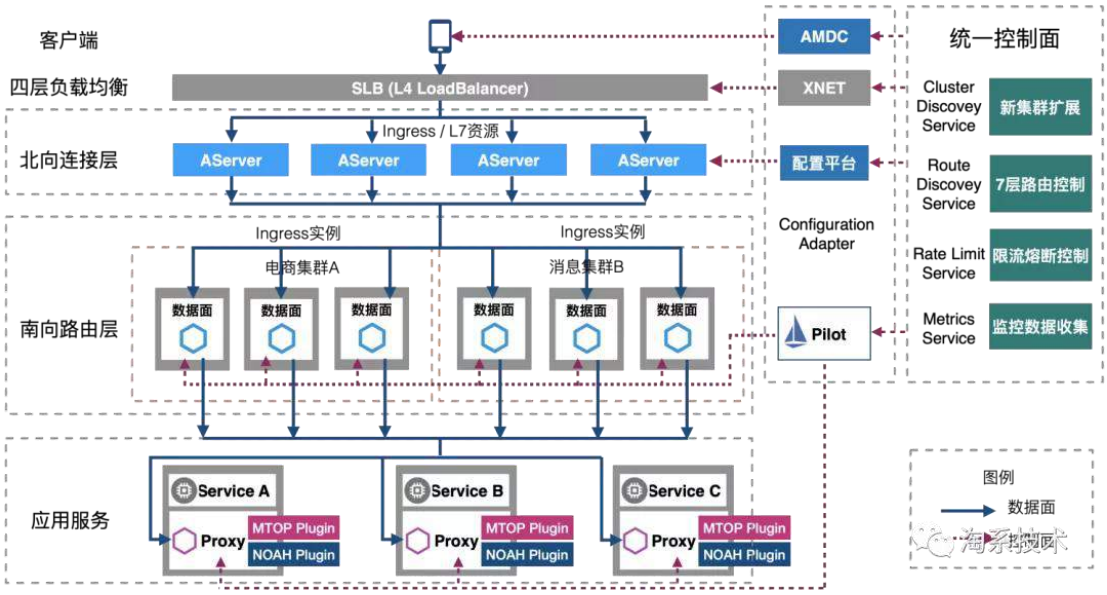
把协议卸载、长链保活等跟客户端交互，并且能够保有极高性能的模块，单独拆分为北向集群，由于性能很好，只需要少量的机器，便可筑高坝挡洪流；对于跟业务路由策略相关，以及安全清洗能力，消耗性能较多，拆分到南向集群，通过北向的高坝保护南向集群不会过载，

南向集群可减少预留水位，进而提升整体的资源利用率，如此可做到即能够提升资源利用率，又可进行灵活配置适应业务快速发展的需求。



整体架构

通过三个阶段演进，终局架构图如下：



AServer 接入网关云原生架构

- **统一控制面**：通过集团统一控制面进行服务接入、服务发现、熔断限流控制，起到变更收口统一处理的作用；

- **北向连接层**：基于 tengine 承载亿级在线用户和流量洪峰，起到高水坝作用，提升南向路由层资源利用率；
- **南向路由层**：基于 Envoy 通过 Pilot 转换 xDS 协议动态下发路由策略，实现动态编排路由、轻量级流量隔离方案；
- **云原生基座**：运维操作建立在集团统一基础设施 ASI，屏蔽网关差异性，降低运维复杂性。

未来

阿里 AServer 接入网关一步步向云原生演进，每次演进都是基于长久以来困扰我们的问题，但又不仅仅止步于解决问题，同时基于当前时代的解决方案，云原生架构改造也远不是终点，云原生的优势也尚未完全发挥。技术的升级最终是为产品服务，云原生升级之后，让我们有了一个强有力的引擎，接下来需要做的就是利用这个引擎改造产品形态，让基于网关之上的开发者最终受益。

产品整合

什么样的状态才是一个网关产品最好的状态呢？开发者每天都在使用，但又无需关心网关的存在，这样存在感最低的状态或许就是最优的状态。当前接入网关从产品形态上暴露了一些实现细节，一个入口应用上线需要通过若干不同系统交互才能完成接入，在云原生改造完成后，可以更好的实现 All in One，产品上做到一体化和闭环。

快速弹性

虽完成 ASI Pod 升级改造，可自动化执行故障机置换，机器迁移等操作，降低了运维成本，但上云最重要的一项能力就是快速弹性，如可以在双十一峰值大促前快速扩容，大促过后快速缩容，可极大减少为准备大促所保有的机器资源，从而节省巨量的成本。当然其中要解决的问题也很多，如安全性可靠性，弹性的实时性，这都需要配合云的基础设施共同建设，真正发挥云上的优势。

双 11 总结：手淘双 11 最新实践：PopLayer 弹层领域研发模式升级

作者 | 倪群（络天）
出品 | 淘系技术

背景

近年来，各大 APP 内的弹层需求逐渐增多，以手机淘宝为例，日常的弹层上线频率为单端每月 50 次左右，而在大促期间可以达到 240 次以上。在手淘内，各类弹层业务都会通过 PopLayer 中间件的能力进行投放。但业务往往会遇到开发弹层难、慢、稳定性差的种种困难。对比于往年业务研发成本较高的现状，PopLayer 在今年提出了【低研发搭投模式】来解决这类问题，形成一套快速搭建+可视化+多端多场景通用的解决方案，在日常与大促期间得到了广泛应用：

- **研发效率升级**：弹层业务的上线成本从 3 天+，降低到 2 小时左右
- **业务覆盖率高**：双 11 大促期间的业务覆盖率达到 75%
- **稳定性极佳**：大促期间线上 0 故障

在各类 APP 都逐渐走向存量时代，精细化流量运营的今天，弹层作为一个可以随时随地产生内容并带来高流量的强运营手段，已经从低频需求，变成了面向各类人群投放的高频需求。作为业务支撑方向的中间件，如何为业务提效，将业务的关注重点从开发转向内容运营，助力其完成触达矩阵，成为了一件非常值得探索的事情。

PopLayer

弹层，是一种强触达用户的交互形态。PopLayer 的定义，则是一个可以在任何 APP 页面上，在指定时间内，对页面无侵入地弹出任何内容的弹层中间件。其业务定位，则为触达各领域

用户的重要流量场。

为了便于理解，下面以手淘首页近期 Pop 为例，将手淘内的 Pop 业务分类举例介绍（本文中 Pop 即指弹层）：

1. 大促氛围打造

点击可跳转原文查看视频详情

开售倒计时提醒

2. 增强用户体验

死亡恢复浏览飘条

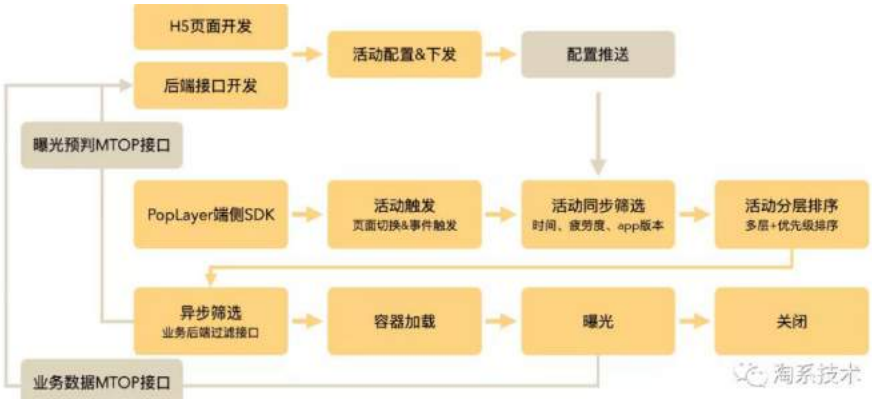
3. 红包发放&提醒

左图：星秀猫开奖，右图：红包催用提醒飘条

4. 用户指引

点击可跳转原文查看视频详情

可以看到，弹层业务的交互形态是灵活多变的，业务目标诉求也各有不同。其背后有着各自业务层面的复杂诉求和增长目标。PopLayer 为此提供了一套端侧弹层管理 SDK+投放系统的整体解决方案。



PopLayer 常规投放&生效流程

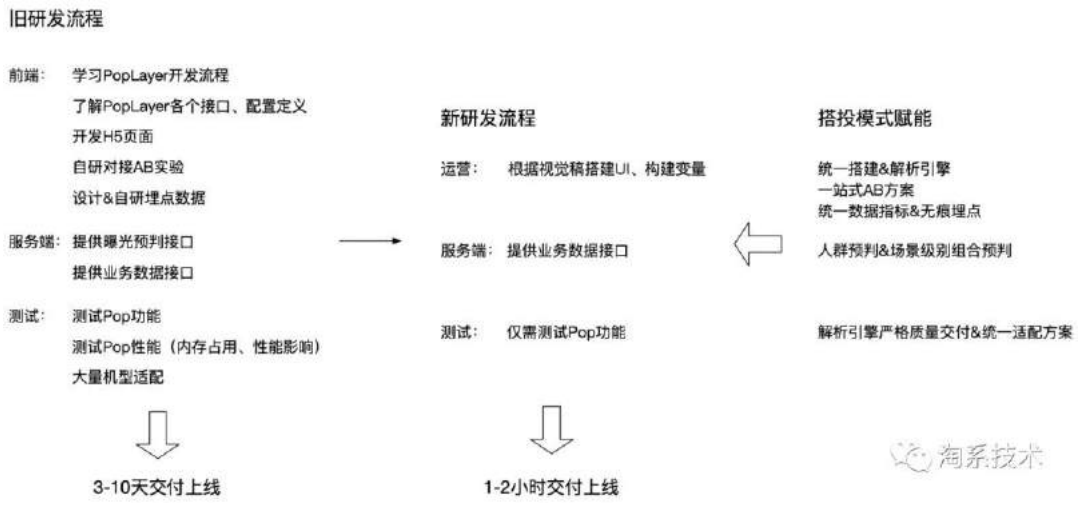
注：PopLayer 是以端侧中间件为核心进行建设的，其中每个环节都有比较复杂的链路可以展开，这里我们不展开讨论端侧细节，主要讨论研发效率方面。

在这套流程中，对业务方负担最重的，也是研发耗时最重的，便是前端页面的研发以及服务端接口的研发。且各个业务的曝光预判接口不断累积，也带来了非常大的资源浪费与 QPS 压力。随着弹层业务逐年增多，这套模式的弊端越来越凸显：

- **研发效率低下。**以日常期间观察，研发一个图文类型弹层，至少需要一个前端人员投入三天以上时间+一个后端人员投入三天以上时间+测试人员投入一天以上。
- **运营效率低下。**运营策略往往受限于研发成本、资源调控、以及上线时间等等问题，而无法灵活展开与快速迭代。尤其在大促期间，很多快速决策的运营玩法无法及时且稳定地落地，丧失了关键时刻获取流量的机会。
- **研发质量难以保证。**不同于整页研发，弹层存在一些特殊需要注意的问题。而研发人员接入 PopLayer 的流程熟悉程度往往有限，很容易因缺乏经验而产生线上异常，比如只有背景黑色遮罩弹出而内容加载失败（可以想一想会是什么状况）
- **业务数据指标没有统一标准，**无法形成客观统一的业务指标，无法通过数据快速定位问题，无法形成有效的数据沉淀与对比。
- **整体方案难以沉淀复用。**

这样的研发成本，对于 Pop 这类往往需要快速响应的业务需求，是远远不能满足的。尤其在大促期间，对时效性要求很高，一个 Pop 从决策到上线，可能仅仅只有 1-2 个小时的响应时间，一旦错过时机对业务的流量损失是巨大的。

经过建设搭投模式，这套陈旧野生的研发流程终于得到了改变。如今，通过新模式，一个常规的单图 Pop 几分钟就可以完成搭投。业务方可以彻底解放双手，集中精力在更加优质的内容编排与制作上。



PopLayer 搭投模式对研发流程的影响

PopLayer 搭投模式

Pop 业务背景分析

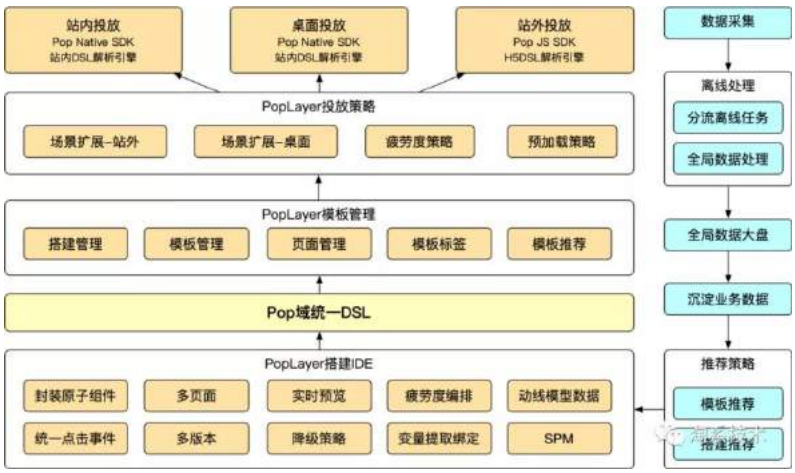
经过长期与业务深入合作，我们发现弹层的需求往往有一定的规律可循。PopLayer 领域下的业务特征大体如下：

- **UI 结构轻量**：主要为底图+内部图文混合的 UI 结构，视觉复杂度有限。
- **点击交互可枚举**：跳转页面、关闭 Pop、发送后端接口、切换内部子页面等。
- 组件复用性低、整体复用性高：每个 Pop 内部可复用的组件几乎不存在，更应该以一个完整的 Pop 作为一个模板进行维护和复用。
- **Pop 特有逻辑较多，**比如疲劳度规则、各类数据来源变量解析等。

那么实现一套统一标准的搭建-投放链路，从前后端等各个方面逐个击破，来承载业务的大部分高频需求，支持其快速、无研发迭代上线，便成了解决这类问题的首选方案。

得益于这套标准化的前端协议规则，我们可以将 PopLayer 的触发范围，从 APP 站内触达，向其他流量场横向扩展，比如 Android 桌面、H5 环境等，这部分后文将会展开讨论。

搭投模式架构



搭投模式方案框架图

我们通过锁定 无研发搭建+多端多领域统一框架 的解决方案，支持运营与业务快速完成各类弹层小时级上线。其链路主要包括如下几个部分：

- **搭建**
  - 设计一套 Pop 业务域内的统一业务描述 DSL，来描述 Pop 的全部 UI 架构、数据提取规则、交互逻辑等等内容。以其为核心，完成搭建与各端各领域的解耦。
  - 搭建 IDE，提供友好的编辑界面、实时动态预览、真机预览、搭完即投等业务服务，最终产生标准 DSL 内容。

- **投放**
  - 探索除 APP 站内之外的更多触达领域，包括 Android 桌面环境、H5 环境。
  - 研发 DSL 运行时解析引擎，并完成统一体验的 Pop 渲染及交互。

- **ServerLess 服务**
  - 提供一体化人群预判服务。
  - 提供权益、AB 与模板搭建的打包配置能力，无需业务方自建实验、自研权益对接。
  - 将单场景多 Pop 情况下的预判 QPS 压力，降低为单场景组合预判模式，有效降低服务端压力。

搭建

搭建与 DSL

DSL，即领域特定描述语言，是为了解决特定领域问题而形成的编程语言或规范语言。在 Pop 业务域下，我们无需形成编程语言，甚至追求尽可能低研发，所以这里的 DSL 即为一种 Pop 业务域范围内的规范描述语言。

Pop 的 DSL 格式为常用的 JSON 格式。其整体结构为 pages-UI 动线、props-变量解析、requests-请求接口、env-环境全局配置。

下面我们从交互动线结构、变量解析、事件结构、疲劳度几个方面分别介绍 DSL 描述的主要内容。

1. 交互动线与 UI 结构

交互动线

点击可跳转原文查看视频详情

以如上的测试 Demo 为例，我们可以看到其基本动线为：展示开奖图文，点击后，进入第二页红包获取图文。但实际针对部分不同策略的用户，会如第二个视频，直接展示其红包获取图文。

对应到 DSL，我们提供了多子页面+多版本的描述方案，即通过创建多个子页面+每个子页面的多个版本来完成动线素材，并通过设置事件动作，完成动线串联。对应到 DSL 的结构，即通过 pages+vers 以树形结构分别描述各个子页面版本。其整体示例如下：

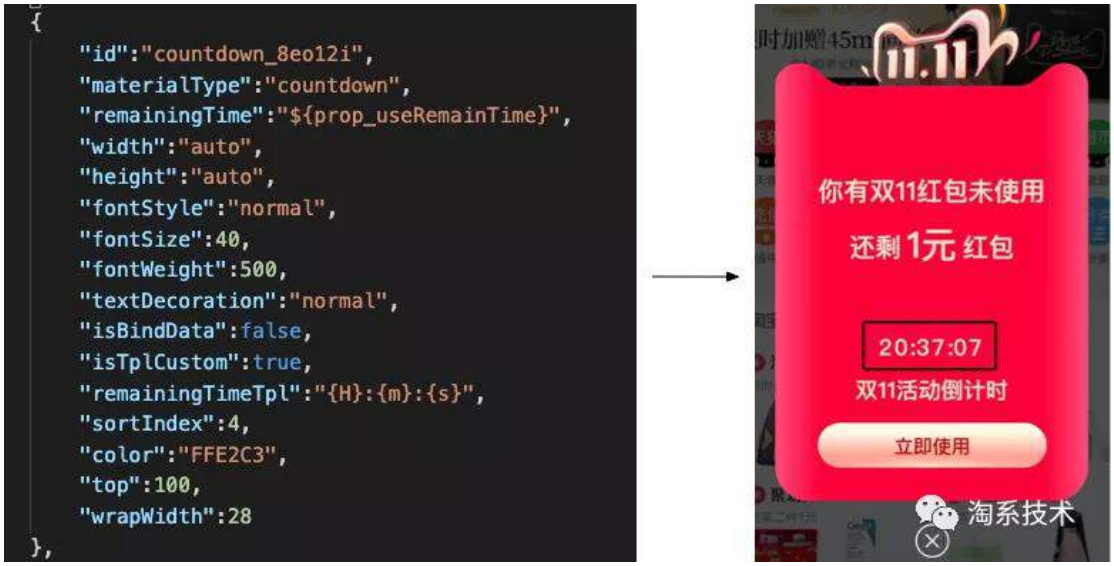
布局版型

Pop 的布局版型是多种多样的，但基本可归类为如下几种：居中、四角挂角、四边贴边。DSL 设计中，每一个子页面都可以单独设置其布局版型。不同的版型，会以不同的布局逻辑计算其大小位置。

居中　　挂角　　贴边

UI 组件

Pop 形态下的 UI 组件，基于围绕着如下几个类型展开：图片、文本、视频、容器、倒计时、点击热区等。即通过提供大图或视频为背景，并通过容器+内部组件形成内部复杂的界面布局。我们针对各个组件提供了统一的布局配置+各自不同的素材配置。以一个倒计时组件为例：



2. 变量数据提取与绑定

变量数据提取

Pop 的内容与服务端数据做绑定时，需要提供一套提取数据的描述方案。而数据来源因 Pop 的整体链路设计，存在多个可能来源。我们通过 指定数据来源+提供插入式 Mtop 接口配置+接口数据提取 Function 完成数据提取的设置，形成一个变量。仍以上述 Demo 为例，其中红包金额的变量为服务端 Mtop 接口返回数据。其提取流程示例：

即通过预判 MTOP 接口数据源，通过 JSONPath，并指定其数据位置完成提取。在某些较为复杂的情况下，有时数据来源需要多层解析（JSONPath+URLDecode+URLParse），那么也支持其设置串行多层解析。

变量绑定

解析结束的变量，即视为一种全局资源，其可以绑定到各种内容与其他数据上，哪里需要哪里搬。比如图片地址、文本内容、toast 内容、跳转地址、MTOP 请求参数等等。其实现方案为常用的字符串模板表达式${prop_name}，进行运行时替换。

3. 事件结构

大多情况下，Pop 内的事件，即为用户点击事件，但随着业务复杂度的提升，例如视频播放结束、视频加载失败、倒计时结束等时机也需要响应事件，我们便提供了统一的事件描述，方便挂载到各个组件事件配置上。而事件的类型。即为跳转场景、切换子页面、发送 MTOP 接口、关闭 Pop 等，我们分别对这些事件提供了对应的封装描述。此处细节较多暂不展开。

4. 疲劳度

疲劳度是 Pop 投放策略的重要组成部分之一。疲劳度的设计分为疲劳度规则+疲劳度消耗规则。例如 Pop 需要用户每天曝光不设限，但点击后当天不再弹出。那么其疲劳度规则为一天一次，而消耗规则即为点击时消耗。通过这样的实现方式，则可以非常灵活的实现各类疲劳度需求，做到想怎么弹就怎么弹。

在 DSL 的曝光、关闭、以及每个事件结构中，均有疲劳度消耗规则，而疲劳度整体规则，则通过不同的疲劳度表达式完成配置。

搭建 IDE

IDE 的核心功能，即为业务用户提供一个实时可视化、随时可真机预览、搭完即投的一站式搭建编辑器。其产出，则是产生一份描述业务完整需求的 DSL 内容。目前已为业务提供包括页面搭建、数据管理、曝光判定、疲劳度规则、降级策略、埋点配置等方面的搭建服务。

搭建 IDE

投放

如方案框架图所示，搭投模式的目标不仅仅是在 APP 站内完成 Pop 投放，还需要在 Android 桌面、站外 H5 这样的环境里完成一站式多端投放。我们可以把目前涉及的几个流量域，称为投放领域。

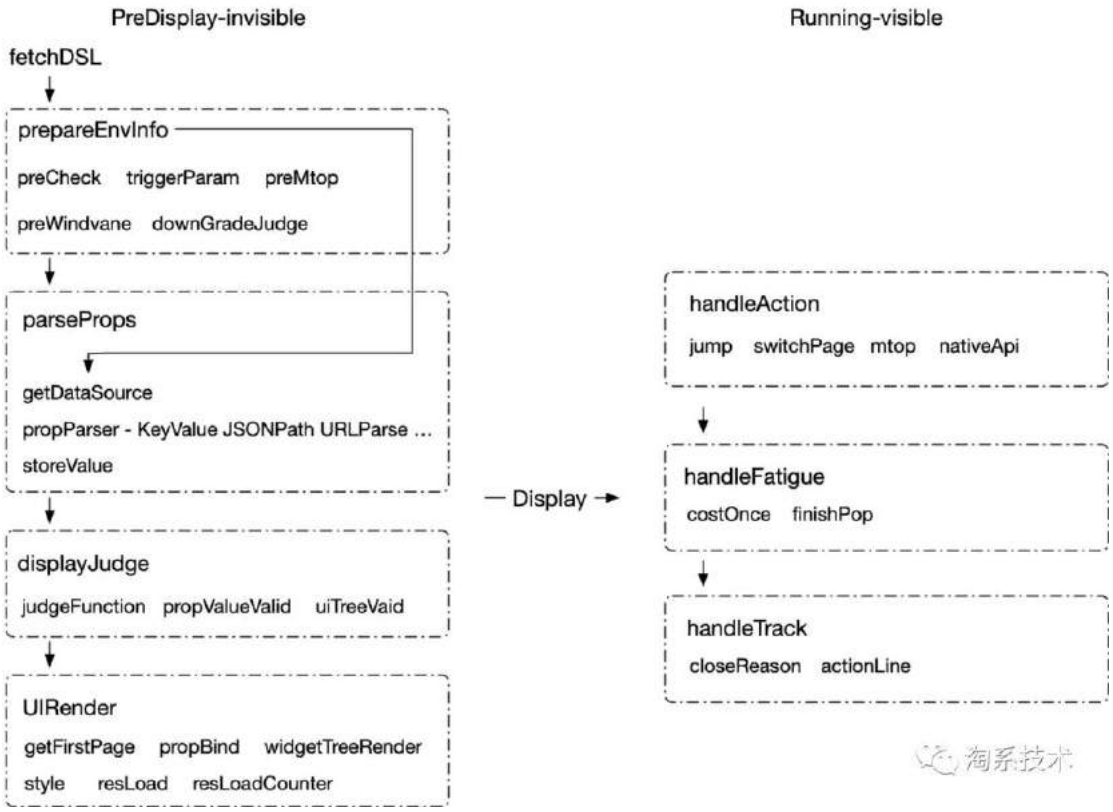
APP 站内的触发流程，即 PopLayer 端侧中间件，功能上有非常丰富的积累，可支撑几乎所有 Pop 业务的各方面诉求，此处不进行展开，本文将从弹出 Pop 后的解析引擎、Android 桌面的投放领域支持方面进行介绍。

运行时解析引擎

针对不同的投放领域，需要形成各自的运行时解析引擎，目前我们完成了 APP 站内引擎：负责站内+Android 桌面的解析渲染，以及 H5 站外引擎：负责 H5 环境下的解析渲染。这里我们主要针对站内引擎进行介绍。

PreDisplay + Running

解析引擎的主体工作流程，分为 PreDisplay 阶段：获取 DSL、获取各环境数据、解析变量、完成 UI 渲染并曝光，以及 Running 阶段：在曝光后的事件交互处理。



解析引擎工作流

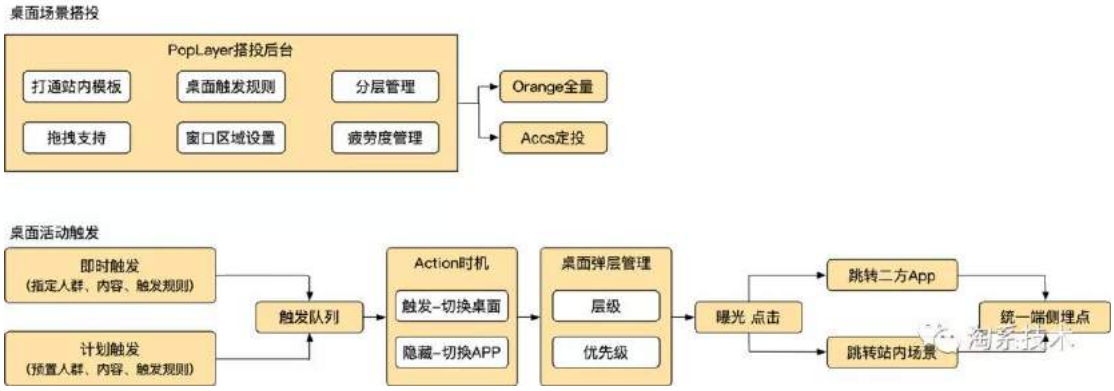
在执行 display 之前，Pop 为隐形状态，用户无感知。经过如上图的 DSL 解析、同步各类环境数据、变量解析、曝光判定、素材加载等流程后，通过 display 接口，完成最终曝光。

为了达到双端统一的渲染效果、高适配性、以及高性能渲染的要求，站内引擎的底层载体目前为 Rax 方案。基于 Rax 完善的工程化支持，我们得以完成一系列上层方案，无需过度关注动态性、适配性等问题。

Android 桌面流量域打通

对于手淘这样日活流量足够大的 APP，其 Android 桌面的触达流量价值同样是巨大的。相比 APP 站内的 Pop 触达，其更加拥有包括加强唤端、二方流量交换这样的独特价值。在有规则规范的投放前提下，我们可以通过端侧中间件建设，把 Pop 搭投的能力无差别的输出到桌面环境，使其成为 Pop 触达生态的一环。其具体的触达形态，则可以是顶部消息、挂角提醒等。其底层实现方案为 Android 悬浮窗。

桌面 Pop 的效果 Demo 如下：



桌面 Pop 方案框架

首先，我们将桌面与站内进行了包括端侧、投放侧的搭投能力对齐，使一个搭建产生的页面，即可以投放在手淘里，也可以投放在桌面上。为此我们抹平了底层方案不同带来的差异，包括：

- 搭投模式与站内一致，同样采用标准 DSL+解析引擎完成渲染。
- 通过控制 Window 添加次序来对齐层级管理。
- 通过控制视窗大小位置，控制其可绘图区域；通过搭建输出可视区域位移量，对视窗内容进行位移还原窗口内容。

另外，我们提供了桌面环境的特殊处理：

- 增加了切换桌面触发时机（计划触发，适合计划常驻），并打通了 ACCS 消息触发时机（即时触发，适合消息类型）。
- 增加了自由拖拽、边侧自动吸附功能。

由于桌面环境的特殊性，应避免对用户形成严重的干扰。那么桌面触达的规则管理则十分重要。目前我们设计了如下避免过度干扰的规则：

- 桌面环境的 Pop 必须有明确明显的关闭按钮。
- 切换其他 APP 时，需要将 Pop 内容进行隐藏，对于 Android 高版本则进行倒计时后自动关闭设置。
- 桌面的弹出管理底层与站内一致，采用分层分优先级管理，并对一次桌面切换的曝光次数进行上限设置。

ServerLess 相关建设

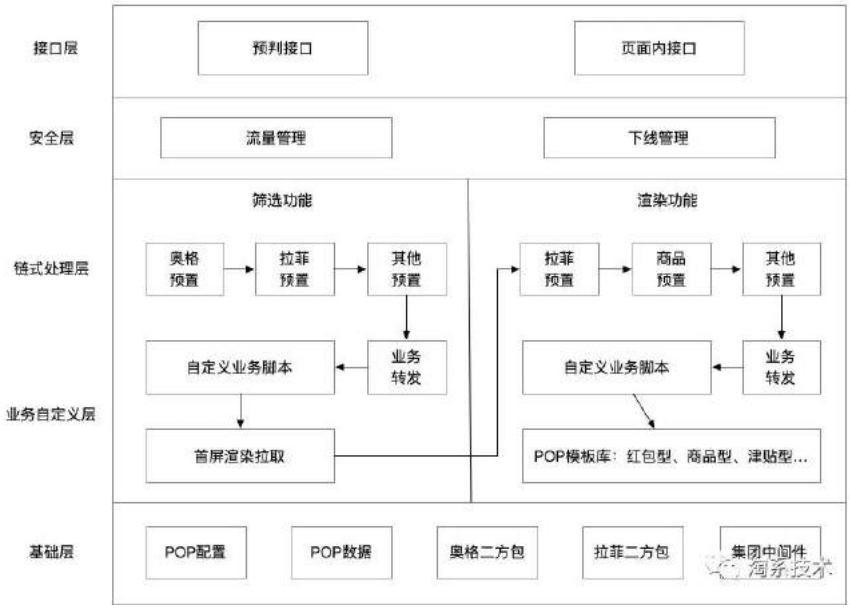
从上述投放流程图可以看到，业务对于曝光预判、业务数据方面都是需要服务端的人力投入的。即除前端的研发成本问题，服务端同样面临类似的问题。我们梳理业务目前痛点如下：

- **人力消耗大，大促时效性差**
- **机器资源消耗大**
  - 全量配置下发+全量接口预判的模式，导致单活动机器资源消耗大；
  - 单场景（比如手淘首页）下的 Pop 往往存在多个，活动之间筛选独立进行，导致机器消耗总量增长快（QPS 总量随活动数线性增加且无上限）。
- **稳定性风险高**
  - 临时开发的模式，加上人员开发质量层次不齐，稳定性很难保障。
  - 业务需要自己投入精力维护稳定性，特别是每次大促的时候应对突发流量。

为此我们实现了对业务进行一站式托管服务。核心目标为：

- 实现权益、导流这两个业务领域的无研发极速上线
- 降低机器资源消耗，在线活动数量不再受机器资源限制
- 托管业务全年 0 故障

通过拆解上文的投放流程图，可以看到服务端的工作主要包括曝光预判接口，以及页面内的业务数据接口。我们针对两部分分别进行部分托管建设，架构图设计如下：



ServerLess 服务架构

- 针对曝光预判接口，我们提供了单场景多活动的人群预判复用能力，即将人群圈选的预判模式统一集中管理，底层与奥格人群平台二方包打通，上层单场景仅透出一个整合接口。从过去每次切换页面触发 N 次预判接口，变为仅触发一次。业务也无需自研人群接口，仅需把人群包 ID 进行配置即可。
- 针对内容数据接口，我们仍在建设中。计划通过底层打通了拉菲权益平台二方包，将权益类型（红包、优惠券等）直接整合进搭建体系中，业务无需进行复杂的权益能力对接，仅需提供权益 ID 配置即可。

整体效果

除文章开头提到双十一期间的业务覆盖率已经达到 75%之外，得益于搭投模式对研发效率的提升，今年双十一期间，手淘内 Pop 的业务量和整体流量也有了大幅度飞跃：

除此之外，今年我们快速稳定地响应了大促期间的全部紧急需求，避免出现过去几年因封网、研发效率等问题带来的无法上线 Pop 的情况。

写在最后

PopLayer 目前除手淘外，已经服务了集团众多 APP，包括天猫、淘宝特价版、闲鱼、淘宝直播、饿了么、Lazada、零售通、AE 等等。后续也将继续以手淘为核心，服务更多的集团业务。

通过双十一大促期间以及日常的业务覆盖率，我们印证了搭投模式+ServerLess 对业务的价值。站在业务的角度思考，Pop 这类"既轻量又复杂"的业务域，经过一番深挖的底层支持，可以大幅度破除业务的桎梏，让其解放双手，去快速通过"提出 idea-搭投-AB-看数据-再次迭代"的模式得到最佳的业务结果。这套研发模式的优化，从思考如何研发变为如何尽可能封装研发，对于相对轻量级的业务域来说也是有输出价值的。

后续，我们一方面将会继续完善相关建设，将 AB、标签+推荐系统、引擎加载页面性能优化等等进行深度挖掘，从研发效率提升，升级到业务价值提升；另一方面也会将 Pop 的建设经验沉淀成流量域方法论的一部分，输出到其他流量域中，为业务探索与构建更有价值的流量增长矩阵。

向量召回在躺平 APP 的实践

作者 | 陈雷慧（豆苗）
出品 | 淘系技术

背景

躺平 APP 是"躺平"这个大生态中生活记录社区，记录生活记录家。

该业务场景中，个性化推荐在充分利用流量实现精细化运营、促进信息流动方面有着不可或缺的地位。在业务成长初期，其内容推荐也面临着如下两个问题：

- **用户冷启动**：对于无任何行为或行为稀疏的用户，难以有效的捕捉到用户的兴趣偏好，进行个性化推荐；
- **内容冷启动**：受限于有限的流量，也存在一定程度的内容冷启动，小众内容得不到推荐展示机会；

目前，业务处在拉新促活的阶段，因此我们迫切的需要缓解上述两个问题：

- 一方面要使得内容能够精准的触达新增和低活用户，提升留存，将其转化为平台的活跃用户；
- 另一方面也要保障长尾小众社区用户的内容有推荐展示的机会，避免小众用户的流失。

一般来说，推荐系统采用召回->排序两阶段的架构。召回阶段从海量内容池中召回数千条内容生成候选集，排序阶段利用用户、内容侧丰富的特征、上下文信息和复杂的模型对候选集中的内容进行打分排序，最终为用户返回数十条内容。

因而，召回是推荐系统的基础，很大程度决定了推荐效果的上限。随着表征学习的发展，向量召回逐渐成为推荐系统中提高召回准确性和多样性的一种行之有效的方案。

鉴于此，我们在原有的个性化召回和热门召回的架构上，新增一路个性化向量召回。并且，考虑到躺平 APP 是一个以图片为内容主要载体的 UGC 社区，我们主要从网络表征学习和多模态表征学习两个角度进行了相关的探索。

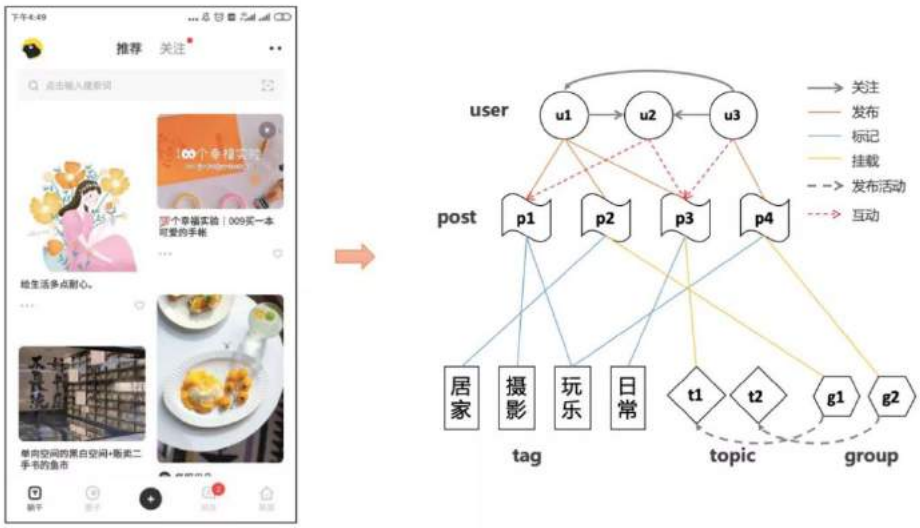
本文旨在分享向量召回在躺平 APP 社区上的应用与实践，内容将会分为以下三个部分：

- 第一部分为网络表征学习在召回上的落地
- 第二部分简单介绍利用多模态信息学习内容的向量表征
- 最后一部分对本文进行总结与展望

希望能对大家有所帮助与启发。

Graph Embedding 的实践

用户在躺平 APP 社区的行为，如发布、点击、点赞、评论和收藏等都可以抽象为网络关系图。因此 Graph Embedding 非常自然地成为学习社区中用户与内容 embedding 的一项关键技术。



目前落地的模型大致分为两类：直接优化节点的浅层网络模型和基于GNN的深层网络模型。

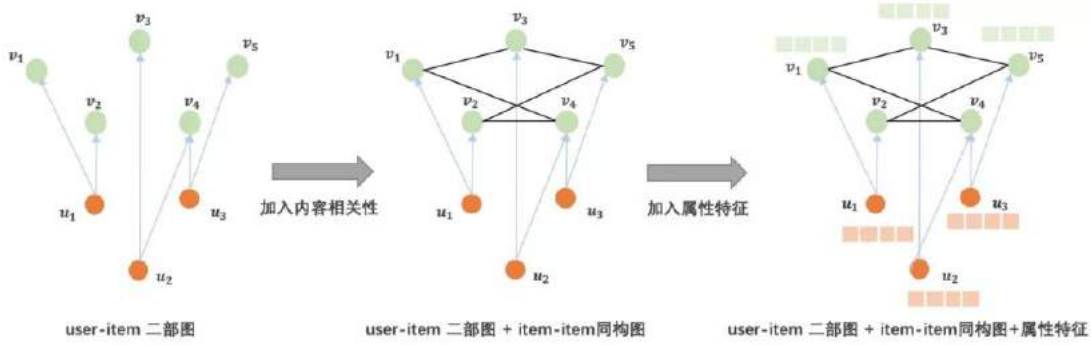
前者包括基于用户行为理解内容，学习内容向量表征的 item2vec[1]，用于扩充 i2i 召回；同时学习用户与内容向量表征的异构网络表示学习模型 metapath2vec[2]，用于提高内容召回的多样性；以群体行为代替个体行为的 userCluster2vec[3]，缓解新用户冷启动问题。

后者包括采用邻域聚合的思想，同时融入节点属性信息，通过多层节点聚合使得网络中的拓扑结构能够有效捕获的 GraphSAGE[4]，以及将 attention 机制运用邻域信息聚合阶段的 GAT[5]，对用户与内容向量表征进行更加细致化学习。

本节主要介绍 GraphSAGE 和 GAT 在社区内容推荐的应用与实践。

网络构图

我们将用户与内容的交互行为构建为 user-item 二部图。考虑社区用户行为数据的稀疏性，我们在 user-item 二部图网络中加入描述 item-item 相关性的同构网络，避免孤立节点无法学习有效向量表征问题。



user-item 二部图网络

user-item 二部图网络描述用户对内容的偏好，包含两类节点：user（用户）节点 $U = \{u_1, u_2, \dots\}$ 和 item（内容）节点 $V = \{v_1, v_2, \dots\}$，以用户近 7 天的历史行为数据为一次 session，构建网络的边。

以用户u_i 与内容 v_j 为例，节点间边 e_{ij} 的权重 w_{ij} 的计算方式为：

$$w_{ij} = \frac{c_{ij}}{\sum_{j=1}^{|V|} c_{ij}}$$

其中，c_{ij} 为用户u_i 对内容v_j 互动次数，一次点击、点赞、评论、收藏均为一次互动。

item-item 同构网络

item-item 同构网络衡量内容之间的相关性，仅包含一类节点：item（内容）节点 $V = \{v_1, v_2, \dots\}$。从全站用户互动的内容序列挖掘得到内容之间的相关性。

属性特征

本次训练主要使用的是 id 类属性特征。用户侧使用如年龄、性别、地区等特征，内容侧使用标签、挂载的圈子和话题等特征。

技术实现

相比较于采用浅层网络直接优化节点的 Graph Embedding 模型，如 DeepWalk、node2vec 和 metaPath2vec 等，基于 GNN 的深层网络通过卷积(GCN)、注意力机制（GAT）和门控网络（GaAN[6]）等技术，对图的全局及局部拓扑结构进行更细致化的学习，并能够自然得融入节点的属性特征。

GraphSAGE

GNN 模型在向量召回应用的第一阶段，我们主要选择归纳学习——GraphSAGE 来学习用户与内容的向量表征。该模型在对邻居节点进行聚合前，先对邻居节点进行采样得到一个子

图，并用该子图代替原图进行邻域聚合计算，从而有效地避免了 GCN 由于需要融合所有邻居节点而无法完成大规模数据向量表征学习的弊端。伪代码如下：

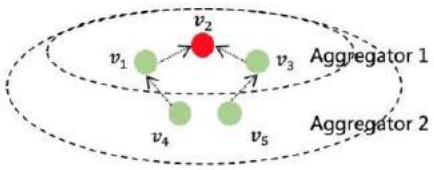
Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

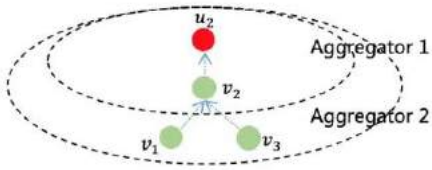
Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

1 $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
2 **for** $k = 1...K$ **do**
3 **for** $v \in \mathcal{V}$ **do**
4 $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
5 $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
6 **end**
7 $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
8 **end**
9 $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

伪代码 4-5 行为邻域聚合层。以生成节点 v 第 k 层向量表征 h_v^k 为例，首先聚合采样得到的节点 v 所有近邻节点 $N(v)$ 的第 k-1 层向量表示，从而得到节点 v 第 k 层的邻域聚合特征 $h_{N(v)}^k$，并与节点 v 的 k-1 层向量表征 h_v^{k-1} 拼接起来，最终通过一层全连接转换得到节点 v 在第 k 层的向量表征 h_v^k。



Item邻域聚合

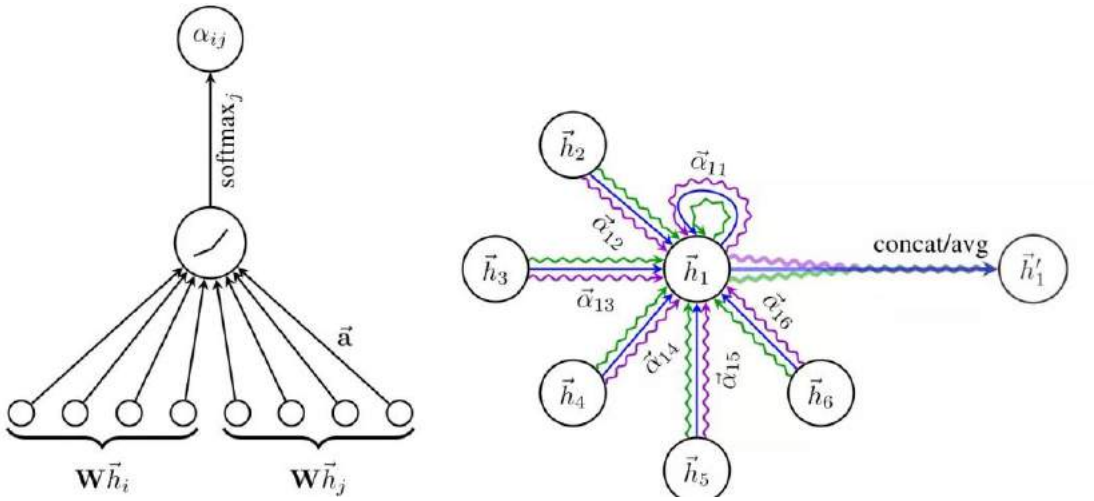


user邻域聚合

为了在充分学习到网络的拓扑结构信息的同时控制训练成本，我们进行了多组离线实验，最终发现仅采用二阶的邻域聚合就能获得比较理想的效果。具体地，通过聚合 2 阶邻域节点的 embedding，生成一阶邻域节点 embedding，再聚合一阶邻居 embedding，并融合自己本身 embedding，完成自身 embedding 的更新。需要注意的是，内容侧从 item-item 同构网络中采样得到邻居节点的子图，用户侧从 user-item 二部图中采样得到邻居节点。采用 Cross Entropy 的分类损失函数进行模型训练。

GAT

考虑用户行为序列中存在许多噪声，如误点击等，且用户兴趣在具有多样性同时也具有侧重性。在 GNN 模型在向量召回应用的第二阶段，我们进一步引入 attention 机制到邻域信息聚合阶段，借鉴 GAT[5]思想，对节点邻域节点进行差异化的信息组合，从而对用户兴趣进行更准确捕捉，提升召回效果。



针对图结构数据，GAT（graph attention networks）使用 masked self-attention 层使得图中的每个节点可以根据邻节点的特征，为其分配不同的权值，聚焦在相关输入，且 GAT 无需使用预先构建好的图。

实验证明，GAT 模型可以有效地适用于（基于图的）归纳学习问题与转导学习问题。

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

我们基本沿用原始论文中的聚合方案，在各节点采样完成的邻居节点上进行 Attention 操作，详细原理参考[5]。

对于 user，通过$u-i-i-i...$的路径进行邻居采样；对于 item，则仅通过$i-i-i-i...$的路径进行邻居采样用于后续 embedding 聚合阶段。在正负样本采样阶段，对于 user 节点，我们沿着 path 进行随机采样；对于 item 节点，借助 item 间的相似性系数，沿着 path 按照权重高低进行采样。

效果

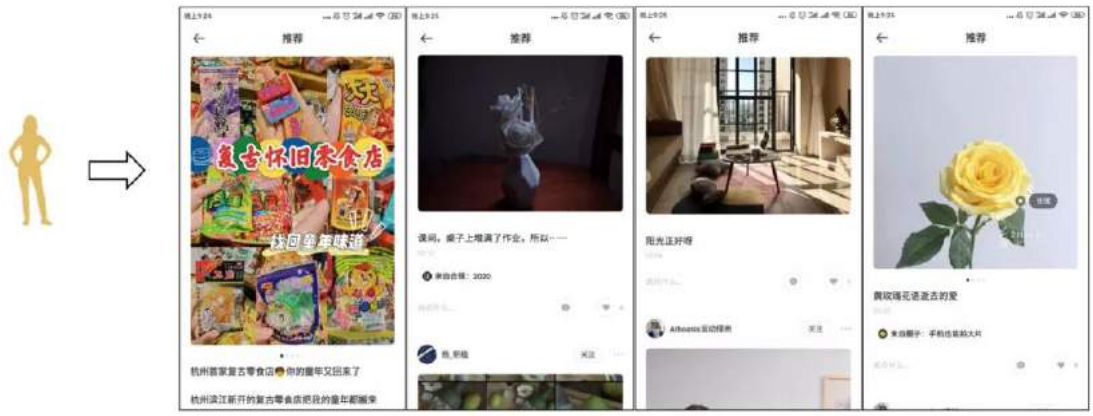
我们将训练得到的用户与内容 embedding 用于躺平 APP 内容推荐的召回的阶段，以用户的 embedding 作为 trigger，寻找与其相似的内容，并送入后续的精排模型，进行打分。

A/B Test

用户点击率和曝光深度均有正向提升，加入属性信息的效果比未加入的更优。

case study

随机挑选一名女性和男性用户，观察其召回效果。其中，女性用户倾向于消费美食、手工 DIY 和居家生活类内容，男性用户则偏好潮玩、摄影和居家生活类内容。两位用户召回结果如下：



基本可命中用户的兴趣偏好。

多模态表征学习的实践

利用多模态信息学习内容的向量表征，是缓解由用户行为为主导的推荐系统中内容冷启动问题行之有效的方法之一，工业界也有很多相关的落地实践。受此启发，我们也希望从内容本身所包含的信息出发，挖掘内容之间的相关性。

主feeds双列流

详情页单列流

通过对社区内容调研发现，整体内容上存在以下两个明显的特点：

- 短文本随意性较大，字数较少，以阐述心情、描述生活状态为主；
- 图片质量较高，大都能清晰地展现出人、物或景，但也存在多图且内容发散的问题，即一条内容的多张图之间的关联性较弱；
- 无论是主 feeds 双列流还详情单列流，都是以用户发布的内容首图来进行展示。

综合考虑，我们尝试通过从内容首图的图片理解角度出发，学习到腰部和长尾内容的图片向量表征，并基于向量召回的技术，提高小众内容召回效果，缓解内容冷启动问题，提升推荐效果。

技术实现

关于图片特征的提取的方法有很多种，其中最为典型的就是利于基于 ImageNet 训练的 ResNet[7]网络去提取图片特征向量。

但这种方法也有一定的局限性，首先，用于预训练的 ImageNet 图像数据的类别信息与躺平 APP 的内容类目体系存在较大差异；其次，ResNet 网络学习出的图片向量维度较高，影响向量召回的效率。

因此，利用躺平的内容类目体系对基于 ImageNet 训练的 ResNet 网络进行 fine-tuning，进而得到躺平内容的图片向量。

我们在预训练 ResNet 网络上添加了两层全连接对图片进行有监督的标签分类。该模型在产出标签的同时，也可以从中间网络层中提取出面向内容和业务理解的图片向量特征副产。

利用向量召回技术，将提取出的图片向量作为一路召回，添加进召回环节，用以缓解内容冷启动问题。

显然，标签的选择，对图片向量的效果有着至关重要的作用。

细粒度的标签，可以提高召回的准确性，但内容同质化现象较为明显，无法彰显社区的多样性。

粗粒度的标签，则无法保障召回的准确性。

我们依次从图片叶子标签、一级标签和二级标签等不同粒度的标签着手，对图片进行分类实验，最终确定以图片二级标签作为分类网络的优化目标，得以在保证产出的图片在拥有较高召回准确性的基础上兼具一定的泛化能力，对用户的兴趣偏好，能够进行发现性的探索。

效果

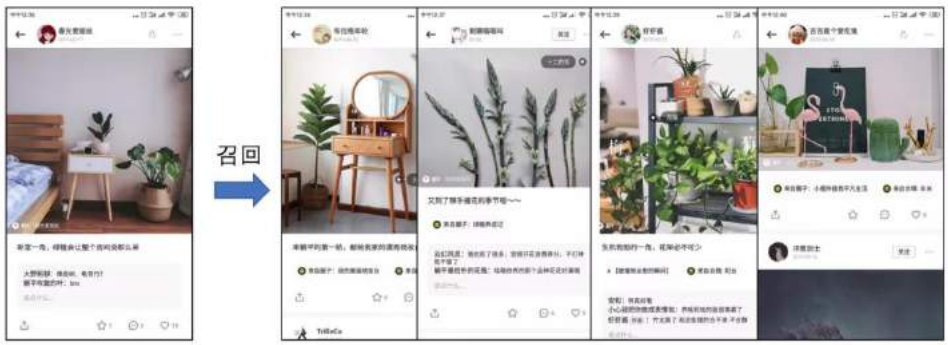
我们将训练得到的图片向量用于躺平 APP 内容推荐的召回的阶段，以用户的历史偏好以及实时点击内容的图片 embedding 作为 trigger，寻找与其相似的内容，并送入后续的精排模型，进行打分。

A/B Test

在上线环节，我们对用户近期交互的内容首图向量进行向量召回，最终点击率有正向提效。

case study

随机挑选内容，离线观测召回效果，示例 case 如下图所示。从结果上来看，一些用户行为稀疏的相似内容可以被召回，起到缓解内容冷启动的作用。



总结展望

本文对躺平 APP 拉新促活阶段所面临的用户和内容冷启动问题的缓解方案进行了讨论。在综合考虑社区内容本身特点和业务的实际需要，在原有的召回架构上增加一路向量召回，期望提高召回结果的准确性和多样性。

首先，从社区属性出发，基于用户交互行为构建图网络，并利用基于归纳学习的 GraphSAGE，对用户和内容进行表征，通过引入 attention 机制，改善向量表征学习的侧重点。相比较于传统的基于统计的召回，归纳学习的 GNN 模型在进一步提高行为稀疏用户召回内容精准性的情况下也增强了多样性。

其次，从图片内容社区角度出发，学习到腰部和长尾内容的图片向量表征，并通过向量召回的技术，提高小众内容召回效果，彰显社区的多样性，避免展示在推荐位的内容永远是热门的大众内容。期望促进信息流动，优化小众群体的生产与消费内容的体验，从而提高这部分用户的留存。

但是，在 Graph Embedding 向量召回建模方面，目前我们只利用了点击数据，在处理好数据不足和不平衡的情况下，后续可以融入对不同用户行为的建模，从而进一步挖掘用户行为的隐含兴趣偏好；也可以在保留行为顺序基础上结合 GNN 进行用户长短周期兴趣建模；在多模态内容表征学习方面，整体设计较为基础，后续会融入挂载圈子、话题和内容文本等信息，探索出更为丰富的内容表征，也欢迎大家一起交流。

参考文献：

[1] Barkan O, Koenigstein N. Item2vec: neural item embedding for collaborative filtering[C]//2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP). IEEE, 2016: 1-6.

[2] Dong Y, Chawla N V, Swami A. metapath2vec: Scalable representation learning for heterogeneous networks[C]//Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. 2017: 135-144.

[3] Grbovic M, Cheng H. Real-time personalization using embeddings for search ranking at airbnb[C]//Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018: 311-320.

[4] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 1025–1035.

[5] Veličković P, Cucurull G, Casanova A, et al. Graph attention networks[J]. arXiv preprint arXiv:1710.10903, 2017.

[6] Zhang J, Shi X, Xie J, et al. Gaan: Gated attention networks for learning on large and spatiotemporal graphs[J]. arXiv preprint arXiv:1803.07294, 2018.

[7] He K , Zhang X , Ren S , et al. Deep Residual Learning for Image Recognition [J]. 2015.

阿里拍卖全链路导购策略首次揭秘

作者 | 王海平（感知）
出品 | 淘系技术

拍卖，是一个大家熟悉的词汇，但对在线拍卖以及在线拍卖的个性化推荐，大家可能不那么熟悉。本文将对阿里拍卖业务背景先做介绍，进而描述业务中的浅库存约束及拍品冷启动问题，并分别提出经线上 AB 实验验证的解法，最后是对解法适用范围的讨论。

前言

业务背景介绍：

- 拍卖中的约束条件：浅库存、时效性
- 交易链路及频道页的布局
- 拍卖导购与传统频道页的区别
- 研究对象：在浅库存和时效性约束下的拍下率优化问题

技术点：

- 运用拉格朗日乘子法求解在浅库存约束下的流量调控问题，并分析了上述调控模型适用的必要条件；
- 从 Knowledge Graph Embedding 和概率图模型两个角度来描述拍品的关键属性，以优化拍品的冷启动问题。

值得一提，今年诺贝尔经济学奖颁发给了保罗·米尔格罗姆（Paul R.Milgrom）和罗伯特·威尔逊（Robert B.Wilson），他们因"对拍卖理论的改进和拍卖新形式的发明"获奖。本文中的拍卖业务，均采用英式升价拍卖，更多从流量分配和效率优化角度探讨，尚未从拍卖机制

设计上做尝试。

阿里拍卖简介

说到拍卖，脑海中就很容易涌现出这样的画面：拍卖师站在台上拿着锤子装作要一锤定音，台下一堆人 1w,5w 的往上加价。所卖的标的物，往往是翡翠、字画、瓷器这样的珍品。那么线上拍卖又是什么呢？

每年双十一的零点，优衣库这样的店铺都会选出一部分商品，让前 100 名购买者以极低的价格拿到。这种饥饿营销可以成功带来流量，背后则是通过捡漏+浅库存+时间窗口的形式让消费者之间形成竞争。有一个频道页，时时刻刻都在进行这样的消费者 PK，在特定的时间段对稀有的商品多轮出价，这就是在线的阿里拍卖。

比双十一更严苛的是，阿里拍卖的库存一般只有一件；与线下拍卖行也不同，在线拍卖可以同时进行几千上万场拍卖，且一件拍品可以多达 4w 人围观，1000 多人订阅，最终 31 人出价，经历了 253 轮竞拍。啥拍品？？今年 6 月，一套离西湖边仅 100 多米的北山街栖霞岭小区，有一套法拍房起拍价仅为 260 万元。你~ 有没有一些心动呢？

补充拍卖交易链路的小知识：

围观：就是点击进商品详情页，热门的标的物往往有过万的围观群众。

订阅：除了即刻拍在线拍卖一般有排期的进行。拍品一般分为预热，开拍和结束拍卖三个时间点，珍品往往当天结束，而房产等大资产会有多天甚至 30 天的拍卖时间。订阅好比是闹钟，用于提醒用户该关注了。

交保：缴纳保证金，交保意味着正式入场。

出价：各方竞价，截止时间内价格最高者胜出。

支付：支付尾款，违约且当前价高于保留价需支付违约金（保证金）。

拍品主要分以下三块；

- **大资产**：如法拍房、二手车、股权、破产公司的厂房、商铺，不一而足；
- **高消品**：名牌手表、名家字画、古董玉器、名酒钱币等等；
- **大众商品**：少量淘系商家供给，也包括部分闲鱼商品。如下图：

上文提到了起拍价 260w 西湖边法拍房就是大资产。和其他频道页不同，来线上拍卖的人有刚需偏好，也有情怀，但更多是投资。当一个价格洼地出现，且是网红资产，短时间内会迅速产生聚集效应。可这是我们想看到的吗？试想我们把上面的西湖边房产推送给每个用户，作为业务核心指标点击、订阅、交保那是蹭蹭蹭的往上涨，但抱歉只有一套。所以成交量只有可怜的 1 套。在浅库存和时间窗口的约束下，用户效率最大化并没有带来频道整体收益最大化。不妨引申，拍卖业务中的导购，和传统频道页导购究竟有哪些不同？

带着问题，我们先看下阿里拍卖频道页的布局。其中，主入口是手机淘宝轻应用第 8 个坑位的 icon，此外还通过短信 push，首猜云主题和商品投放，主搜等渠道。拍卖首页是承接流量的主阵地，模块包括：搜索框、子频道 icon、四个子业务模块和猜你喜欢。

阿里拍卖导购 VS 传统商品导购

单从频道页看，拍卖的 feeds 流和传统猜你喜欢没有太大区别。也许是表象使然，刚接手业务后，我们用近一个月时间尝试把阿里拍卖推荐按照传统频道页推荐来做，结果用户侧的效率有明显提升，但和业务对焦发现，核心指标拍品拍下率不升反降，甚至是降的厉害。被打脸后我们不禁反思，我们错在哪里？

先看业务指标：

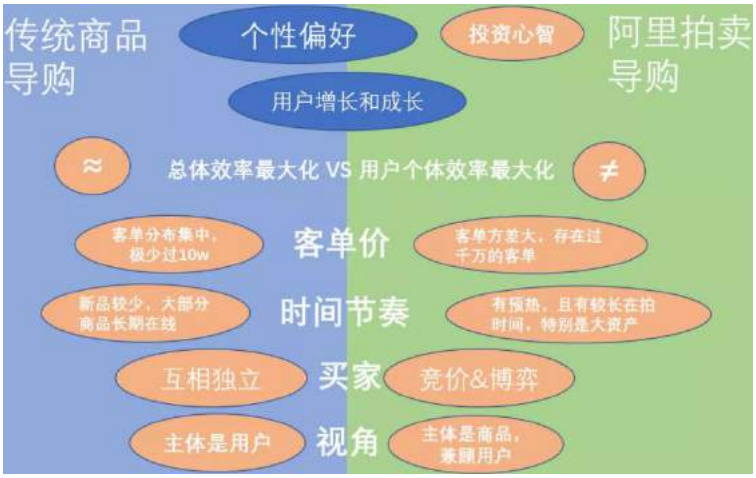
拍下率 = 拍下商品数/总商品数= 1 - 流拍率

解读：

1. 业务最终目标是 GMV，可是拍卖中的价格差从几百到几亿，直接量化存在困难。因而，业务和技术一致认同将目标先拆解为拍下率。
2. 在浅库存约束下，要考虑到长尾商品，并对头部商品流量去中心化。
3. 从业务侧，相比于酒类玉器等珍品，房产等大资产的拍下率优化更为重要，业务要求珍品和大资产的指标分开统计。
4. 拍下是一个延迟过程，特别是大资产的开拍时间可能长达 10 多天，这个算法天级别的优化带来困难；业务建议，可以将订阅率和交保率作为大资产算法优化目标。
5. 影响 GMV 的因素，除了拍下率外还有溢价率（可简单理解成高于市场价的百分比），这也是传统拍卖关心的指标，这一块暂未做尝试。
6. 对新用户尽量避免因调控而导致流失。

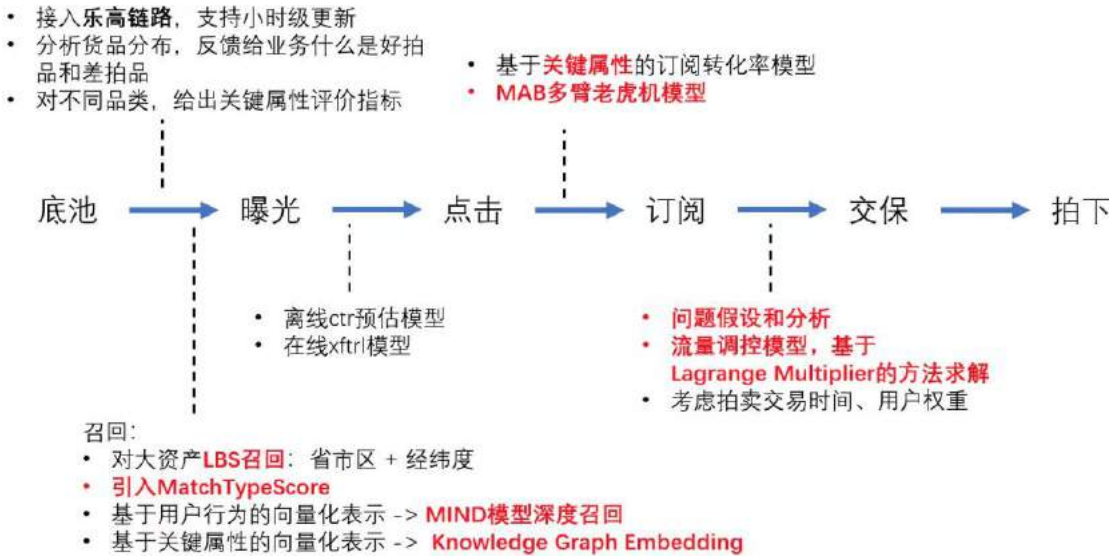
回到问题，在传统猜你喜欢对商品、内容导购时，我们知道要把好的品给对的人。好的品往往是经过群体行为检验的，模型判断出的高点击、转化或关注的品。

如果真有一个品大家都喜欢，那么大概率是运营强控置顶，比如 1499 元的茅台。匹配的过程往往是从用户视角，让商品经过选品->召回->排序->重排序这样的漏斗，并最终展示每页 10~20 个商品。但在浅库存的约束下，如果一样东西抢手，给过多的人看就造成了流量浪费；而如果一样东西有潜在价值但受关注过少、最终流拍，就更加可惜。在拍卖的导购中，更关心一个商品是否会卖，而不是某个用户是否去买。两者的区别，小结如图：



整体框架

按照从底池->曝光->点击->订阅->交保->拍下的漏斗，我们在各个环节做了尝试，标红部分是有较大效果提升。其中，从交保->拍下在传统拍卖中更加关注，特别是提升溢价率，这部分目前还没有尝试。



在求解过程中，会遇到如下一些问题：

1. 浅库存约束下，如何均衡流量分配与效率？在此约束下，是不是更适合做 Exploration and Exploitation?
2. 怎么判断一个拍品好是不好？对好的拍品，我们如何做冷启动？
3. 如果拍下率是优化目标，那么拍卖的流量怎么量化，与拍下率的关系是什么？
4. 拍品分为预热、开拍和结束拍卖三个关键时间节点，该如何考虑拍卖时间节奏？

带着这些问题，我们将从模型调控和新品冷启动两个角度来分析。

流量调控

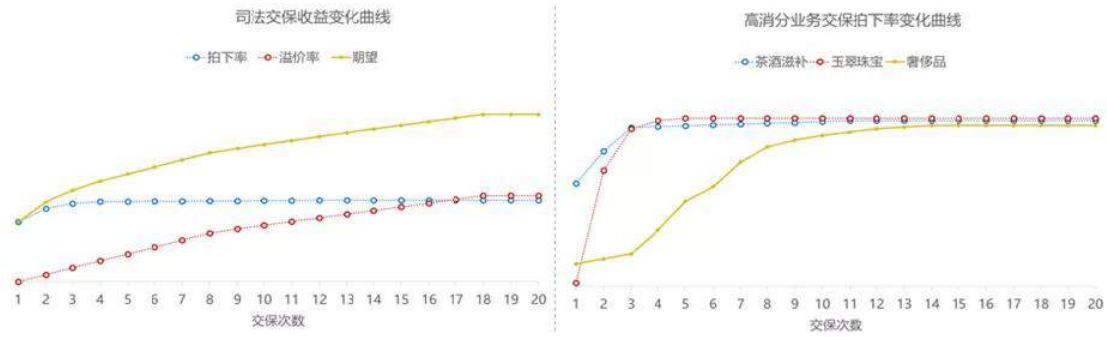
整体思路

布洛和柯伦伯(1996)证明了，在非常一般的私人价值环境中，以及更一般的共同价值环境中，一个简单的无保留价格，有 N+1 个买方的升价拍卖比任何一个只有 N 个买方的最优拍卖都具有更好的盈利表现。在线上拍卖，买方可定义为缴纳保证金的人。这个定理也启示了我们，交保证金的人数与最终商品是否拍下，可能有强相关性。那么究竟如何量化交保人数与拍下率，以及如何定义我们的目标函数，并构建调控模型。我们进行了如下工作：

1. 拟合了交保-拍下率效用函数，并给出相应假设
2. 定义了优化目标函数，并用拉格朗日乘子法求解
3. 在模型中引入时序和流量权重
4. 讨论了上述拉格朗日模型的适用范围

数据分析

通过拉取历史成交数据，可发现交保人数与拍下率呈如下关系，其中不同曲线代表了不同品类，如手表、字画、二手车等：



通过对交保人数与拍下率的关系做了定性，可以发现：

- 交保人数越多，拍下率越高。
- 直观的看，该曲线的二阶导<0，意味着随着人数增加，拍下率的边际收益下降。
- 当交保达到一定的人数时，边际收益已经非常微弱，此时流量的边际成本不变，则可认为是流量浪费，这里的临界点称为最优交保人数。
- 不同品类的曲线和最优交保人数有较大区别。

问题假设

如何定量？我们不妨从假设开始。

假设 1：交保次数与拍下率满足如下关系：

$$U_j = 1 - a^{-k_j}$$

其中k_j是商品 j 交保次数，U_j是 j 的拍下概率，a 是参数并按品类区分。本函数的假设是根据交保-拍下率曲线拟合得到，对不同业务需要先假设函数关系，再验证是否满足次模性。

假设 2：最优交保人数是指，当交保次数k_i达到某个值后，边际收益明显下降，即拍下率斜率。

$$U_j'(n) \approx U(n+1) - U(n) < \varepsilon$$

超过了则认为会造成流量浪费。$U_j'(n)$物理含义是商品 j 交保次数从$n \to n+1$后对成交概率的边际收益。可以说，拍卖流量调控的切入点，就是最大化边际收益。

假设 3：在某个时间窗口内，每个用户存在稳定的投资预期。

来拍卖频道页的用户大多是投资和捡漏心智，虽然用户的投资预期的具体数值难以估计，但我们不妨假设用户期望交保的商品次数，在短时间内是一个常数。具体的取值并不影响最终排序，在后面的部分会阐述。

拉格朗日模型 1.0

考虑目标函数：

$$argmaxL = \sum_{i,j} U_j x_{i,j} - \lambda \sum_{i,j} x_{i,j} lnx_{i,j}$$

$$s.t. 0 \leq x_{i,j} \leq 1$$

$$\forall i, \sum_j x_{i,j} = s_i$$

$$\forall j, \sum_i x_{i,j} \leq I_j$$

其中，$x_{i,j}$是用户 i 对商品 j 的交保概率，$\sum_{i,j} x_{i,j} lnx_{i,j}$是最大熵正则项，用于控制流量均衡；$U_j$是交保次数与拍下率的函数；$I_j$是商品 j 的交保上限。考虑假设 3，当交保次数 k 足够大时，边际收益接近于 0，则认为此时的 k 为交保上限；s_i是引入的一个与用户 i 相关的变量，表示用户 i 的投资预期，由假设 4 可知s_i恒定。该问题的拉格朗日形式如下：

$$L(x, \alpha, \beta) = \sum_j U_j \sum_i x_{i,j} - \lambda \sum_{i,j} x_{i,j} lnx_{i,j} + \sum_j \alpha_j (I_j - \sum_i x_{i,j}) + \sum_i \beta_i (s_i - \sum_j x_{i,j}) \quad (1)$$

KKT:

$$\nabla_x L(x, \alpha, \beta) = 0$$

$$\alpha \geq 0$$

$$\forall j, \alpha_j (I_j - \sum_i x_{i,j}) = 0$$

$$\forall j, \sum_i x_{i,j} - I_j \leq 0$$

$$\forall i, \sum_j x_{i,j} - s_i = 0$$

推导：

$$\frac{\partial L}{\partial X_{i,j}} = U_j - \lambda(lnx_{i,j} + 1) - \alpha_j - \beta_i = 0$$

$$lnx_{i,j} = \frac{1}{\lambda}(U_j - \alpha_j - \beta_i) - 1$$

$$x_{i,j} = exp[\frac{1}{\lambda}(U_j - \alpha_j - \beta_i) - 1] = exp[\frac{1}{\lambda}(U_j - \alpha_j)] * exp[-\frac{\beta_i}{\lambda} - 1] \quad (2)$$

$$\sum_j x_{i,j} = s_i = \sum_j exp[\frac{1}{\lambda}(U_j - \alpha_j)] * exp[-\frac{\beta_i}{\lambda} - 1] = exp[-\frac{\beta_i}{\lambda} - 1] * \sum_j exp[\frac{1}{\lambda}(U_j - \alpha_j)] \tag{3}$$

$$\frac{s_i}{exp[-\frac{\beta_i}{\lambda} - 1]} = \sum_j exp[\frac{1}{\lambda}(U_j - \alpha_j)]$$

即：

$$s_i * exp[\frac{\beta_i}{\lambda} + 1] = \sum_j exp[\frac{1}{\lambda}(U_j - \alpha_j)]$$

令：

$$Z = \sum_j exp[\frac{1}{\lambda}(U_j - \alpha_j)]$$

则：

$$Z = s_i * exp[\frac{\beta_i}{\lambda} + 1]$$

$$\boldsymbol{\beta_i = \lambda(lnZ - lns_i - 1)}$$

把 Z 带入由（2）式得：

$$x_{i,j} = exp[\frac{1}{\lambda}(U_j - \alpha_j)] * exp[-\frac{\beta_i}{\lambda} - 1]$$

$$x_{i,j} = \frac{s_i}{Z} exp[\frac{1}{\lambda}(U_j - \alpha_j)]$$

又由原式式(1)可知，的梯度为：

$$\alpha_j^n = max(0, \alpha_j^{n-1} - \eta(I_j - \sum_i x_{i,j}^{n-1}))$$

其中，$\sum_i x_{i,j}^{n-1}$ 是拍品 j 是当前交保人数，$\eta(I_j - \sum_i x_{i,j}^{n-1})$越大，说明离调控目标还有距离，则$\alpha_j^n$会变小，从而使分配概率$\boldsymbol{x_{i,j}}$增大。考虑$x_{i,j} = \frac{s_i}{Z} exp[\frac{1}{\lambda}(U_j - \alpha_j)]$，在个性化排序中是在同一用户下比较商品得分，因而最终的计算可以忽略$\boldsymbol{s_i}$变量和$\boldsymbol{Z}$。

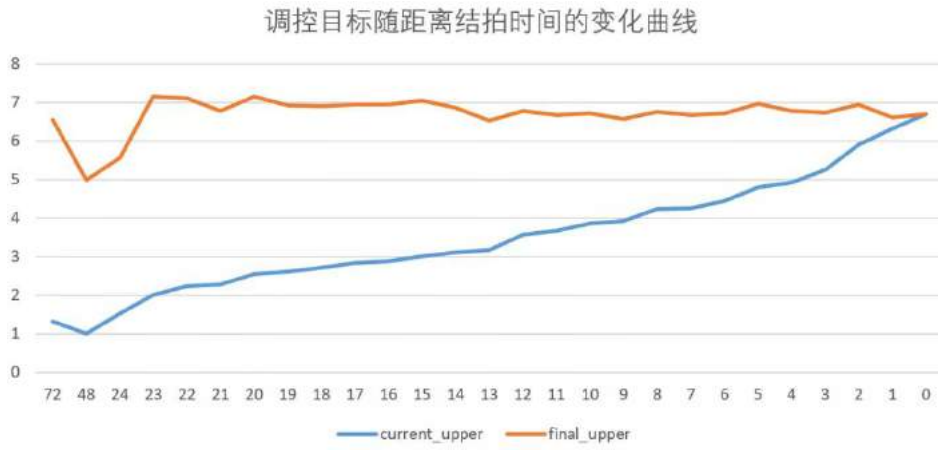
拉格朗日模型 2.0

带有时序的调控目标

在上述 1.0 模型中，最优交保人数I_j是调控目标。然而，我们忽略了拍品起拍和结束时间的影响。试想，有两个同类目的拍品，效用函数U_j一致、最优交保人数I_j一致、结束时间点也一致、当前拍下的人数也相同。但一个离结束拍卖还有 2 小时，一个还有 1 天。那显然，我们应该优先考虑快结束的拍品。如何在模型中体现这件事呢？根据日志，我们把拍卖的结束时间点和离结束时间的小时数做如下二维表：

距离拍卖结束时间	拍卖点点结束 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
24~48小	33.76%	32.56%	33.49%	40.10%	32.83%	35.32%	30.25%	27.18%	32.54%	33.38%	30.02%	29.91%	21.48%	24.73%	16.73%	12.91%	17.46%	16.55%	29.58%	30.68%	24.77%	22.58%	19.78%	20.10%
24	2.09%	1.19%	0.65%	0.59%	0.15%	1.50%	1.58%	1.91%	2.79%	2.92%	3.16%	3.18%	3.73%	2.18%	2.05%	1.40%	1.29%	1.11%	2.23%	3.65%	3.39%	4.71%	4.41%	6.69%
23	1.21%	0.24%	0.40%	0.59%	0.27%	2.16%	0.84%	1.56%	2.49%	2.62%	2.61%	2.27%	2.89%	2.07%	2.26%	1.10%	1.13%	1.44%	2.52%	4.02%	2.91%	3.54%	5.25%	2.23%
22	0.83%	0.21%	0.40%	0.48%	0.44%	1.12%	1.56%	1.69%	2.02%	2.28%	2.11%	2.25%	2.67%	1.75%	2.20%	0.84%	1.26%	1.74%	2.70%	3.64%	2.49%	2.32%	1.91%	1.18%
21	0.38%	0.20%	0.45%	0.60%	0.73%	1.46%	1.86%	2.53%	2.28%	2.23%	2.41%	2.00%	2.29%	1.84%	1.90%	1.13%	1.84%	2.23%	3.46%	3.10%	1.82%	1.42%	1.02%	0.69%
20	0.71%	0.32%	0.79%	1.27%	0.81%	1.39%	1.50%	1.48%	2.56%	2.17%	2.35%	2.23%	2.26%	1.66%	2.29%	1.40%	2.00%	2.83%	2.92%	1.92%	1.11%	0.84%	0.62%	0.57%
19	1.13%	0.02%	1.45%	1.35%	1.00%	0.54%	1.60%	1.68%	2.70%	2.54%	2.14%	2.23%	1.81%	2.12%	2.95%	1.73%	2.75%	2.61%	2.17%	1.17%	0.67%	0.40%	0.48%	1.02%
18	1.59%	1.37%	1.78%	1.25%	1.49%	1.67%	1.12%	1.91%	2.54%	1.77%	1.78%	2.02%	1.92%	2.92%	3.14%	2.64%	2.62%	1.71%	1.25%	0.62%	0.36%	0.42%	0.70%	1.66%
17	2.41%	1.40%	1.81%	1.69%	2.37%	1.48%	1.43%	1.48%	1.97%	1.75%	1.70%	2.18%	2.58%	3.88%	3.77%	2.53%	1.95%	1.27%	0.78%	0.42%	0.31%	0.48%	1.07%	2.08%
16	2.36%	1.74%	2.19%	1.63%	2.59%	1.60%	1.58%	1.18%	2.12%	1.38%	1.72%	3.38%	3.69%	4.05%	3.32%	6.13%	1.17%	0.54%	0.48%	0.42%	0.41%	1.01%	1.58%	2.54%
15	2.58%	1.61%	2.72%	1.80%	1.52%	1.26%	1.84%	1.08%	1.58%	1.56%	2.72%	3.49%	6.81%	3.06%	2.45%	2.94%	0.92%	0.42%	0.36%	0.24%	0.90%	1.41%	2.29%	3.12%
14	3.00%	1.52%	2.10%	1.35%	1.44%	1.08%	1.69%	1.28%	2.97%	2.06%	4.20%	4.67%	5.98%	1.82%	1.71%	1.67%	0.49%	0.47%	0.53%	0.88%	1.24%	1.93%	2.78%	3.78%
13	2.78%	1.93%	1.98%	1.48%	2.32%	1.46%	1.61%	1.36%	2.41%	2.88%	7.08%	4.06%	3.55%	1.50%	0.72%	1.19%	0.30%	0.38%	1.32%	1.46%	1.90%	2.80%	5.13%	3.64%
12	2.55%	1.99%	2.47%	2.05%	1.20%	0.90%	2.62%	2.02%	3.39%	3.71%	3.69%	2.75%	1.91%	0.79%	0.72%	1.15%	0.42%	0.62%	1.62%	1.86%	2.06%	2.42%	3.98%	2.69%
11	2.54%	1.67%	1.60%	1.54%	1.22%	1.36%	2.03%	3.99%	4.43%	4.18%	2.15%	1.41%	1.17%	0.69%	0.61%	1.17%	0.83%	1.19%	2.32%	2.51%	2.73%	2.50%	3.95%	2.64%
10	2.39%	1.50%	2.07%	1.64%	1.47%	2.48%	4.28%	5.39%	5.07%	3.28%	1.33%	0.83%	0.79%	0.52%	0.72%	2.27%	1.23%	1.87%	2.46%	2.78%	2.36%	2.69%	3.96%	2.65%
9	2.25%	1.69%	1.50%	1.39%	2.35%	4.25%	5.06%	5.47%	3.14%	2.07%	0.83%	0.53%	0.55%	0.76%	1.06%	3.10%	2.05%	2.24%	3.10%	2.36%	2.55%	2.70%	3.74%	3.06%
8	2.26%	1.60%	2.27%	1.89%	3.20%	3.07%	5.44%	2.79%	2.14%	1.14%	0.62%	0.54%	0.73%	1.08%	1.76%	4.09%	2.07%	3.12%	3.03%	2.60%	2.58%	2.70%	3.94%	3.02%
7	2.25%	2.23%	2.21%	2.41%	2.59%	3.87%	2.97%	2.73%	1.41%	0.92%	0.49%	0.67%	1.50%	1.63%	2.52%	4.33%	2.79%	4.12%	3.83%	3.36%	2.97%	3.09%	3.88%	3.79%
6	2.67%	2.76%	2.60%	3.31%	3.77%	2.65%	2.81%	1.73%	0.90%	0.65%	0.72%	1.15%	2.32%	2.59%	3.59%	5.64%	3.31%	5.53%	5.46%	2.54%	2.79%	3.50%	3.79%	3.44%
5	4.00%	4.62%	4.50%	3.06%	7.34%	2.51%	3.21%	1.06%	0.75%	1.02%	1.23%	1.93%	2.91%	3.71%	6.60%	7.62%	5.67%	8.20%	3.53%	2.50%	3.18%	3.70%	3.00%	4.27%
4	5.17%	8.65%	4.94%	4.63%	3.74%	1.95%	1.48%	1.05%	1.06%	1.61%	2.11%	3.40%	6.71%	4.35%	6.04%	9.67%	13.31%	8.33%	3.96%	2.96%	3.79%	4.18%	4.50%	5.93%
3	10.21%	8.77%	6.54%	5.86%	3.33%	2.27%	1.73%	1.74%	2.55%	2.95%	4.54%	9.63%	7.04%	4.92%	10.06%	10.14%	11.55%	10.98%	5.92%	4.13%	5.93%	6.03%	7.06%	5.70%
2	6.30%	11.29%	11.15%	10.93%	10.03%	6.44%	3.12%	5.72%	6.14%	9.00%	12.06%	10.68%	10.38%	14.21%	14.58%	10.27%	15.43%	10.77%	9.55%	5.78%	12.93%	11.83%	6.88%	10.22%
1	1.57%	7.82%	7.56%	8.89%	10.82%	14.02%	11.81%	18.04%	8.35%	9.93%	9.62%	4.68%	1.97%	7.70%	6.61%	3.52%	6.36%	6.30%	9.01%	8.96%	10.67%	9.83%	2.38%	2.50%

其中，横轴是拍卖结束的时间点，纵轴是距离结束时间的小时数。取值为该时间段的交保人数占比=商品交保人数 / 商品最终总交保人数，如在下午 14 点结束的拍品集合中，离结束还差 2 小时交保人数占比是 14.58%。基于统计数据，我们可以得到I_j随时间变化的曲线，如下：



从而可以得到：

$$I_j(t) = I_j * weight(endTime, deltaTime)$$

上述的例子中的两个拍品，假设都是 14 点结束，则里结束还差 2 小时的拍品当前调控的目标值为$I_1(t) = I_1 * weight(14, 2) \approx 0.339 * I_1$,距离结束还差 1 天的拍品调控目标值为$I_2(t) = I_2 * weight(14, 2) \approx 0.167 * I_2$。

流量评估中引入用户权重

在拍卖业务中，除去黑产有五类用户：超级 VIP 用户，VIP 用户，至少购买一次的用户，至少交保一次的用户，小白用户。在背景篇中，有介绍说不同用户群体在拍卖频道中的行为差异极大，如超级 VIP 用户和小白用户的交保率相差 12 倍，而拍下率则相差 26.4 倍！

考虑有两种情况，拍品 1 和 2 起拍和结束时间一致，品类一致，效用函数U_j一致、最优交保人数I_j一致。且当前交保人数都是 8 人，但拍品 1 是由 8 个超级 VIP 用户交保，而拍品 2 是由 8 个小白用户交保。那么从直观上，就可以认为拍品 1 的拍下率高于拍品 2。同样，如何在模型中体现?

考虑效用函数$U_j = 1 - a^{-k_j}$拟合的是交保次数k_j与拍下率U_j的函数关系。通过引入用户权重可将k_j改写为：

$$k_j(F_j) = \sum_i counter(i, j) * weight(j)$$

其中，F_j表示对商品j交保人群的分布，$weight(j)$仍基于统计数据，上述 5 类人群的取值权重为 5:4:3:2.5:1。使用改写后的$k_j(F_j)$重新拟合曲线，得到新的效用函数$U_j(F_j) = 1 - a^{-k_j(F_j)}$。

引入时序和用户权重后，模型调整为：

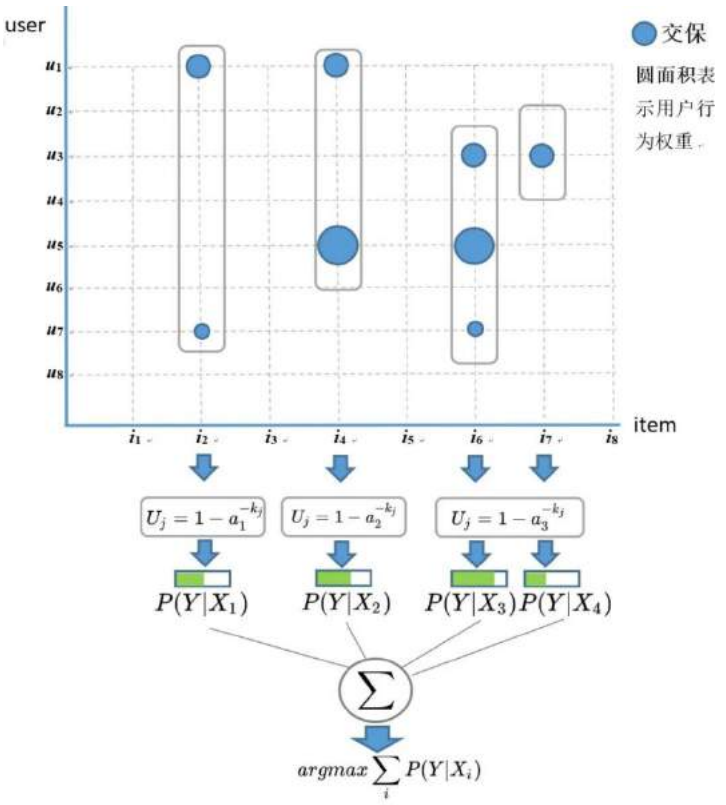
$$\alpha_j^n = max(0, \alpha_j^{n-1} - \eta(I_j(t) - \sum_i x_{i,j}^{n-1}))$$

$$x_{i,j} = \frac{s_i}{Z} exp[\frac{1}{\lambda}(U_j(F_j) - \alpha_j)] \propto exp[\frac{1}{\lambda}(U_j(F_j) - \alpha_j)]$$

浅库存约束下的流量调控问题

回到假设，我们通过建立拉格朗日模型来求解浅库存的拍卖流量分配问题。这里一定有人问，我的业务适合这种调控模型吗？上述模型适用的必要条件是什么？

不妨小结下，我们假设了拍品交保人数和拍下率满足$U_j = 1 - a^{-k_j}$函数关系，并设定了目标函数$argmaxL = \sum_{i,j} U_j x_{i,j} - \lambda \sum_{i,j} x_{i,j} ln x_{i,j}$，其中 L 是算法调控目标。如果从问题出发，考虑用户总数是 N，可得向量空间$X = (x_1, x_2, \ldots, x_n) \in 2^N$，其中$x_i \in \{0, 1\}$表示是否交保。在已知部分用户已交保的前提下，商品 j 拍下率可表示为$U_j = P(Y|X_j) = 1 - a^{-k_j}$，则整体拍下率期望可表示为$E = argmax \sum_j P(Y|X_j)$，如图：



注：user 对 Item 的交保行为可想象为在图示中的格子里画圈，会发现在某一个商品（如i_6）上的边际收益越来越少，且全局的边际收益也在逐步减少，这就是次模性的含义。此外，要尽量找某一列还没画过圈的商品，但此类商品缺少历史行为不确定要哪个更好，这就是下一章要提到的冷启动的问题。

围绕着$U_j = 1 - a^{-k_j}$函数关系，不妨记为F_j。F 的表现是，拍下率随着交保人数的增加而增加，但斜率逐渐下降，这是边际收益递减性质的一个朴素体现。在数学上，以次模性(Submodularity)描述函数的边际收益递减性质，具有次模性的函数被称为次模函数(Submodular Function)。其标准定义为：对给定的离散集合 S，满足：

$$f(A \cup \{a\}) - f(A) \geq f(B \cup \{a\}) - f(B), \forall a \in S, \forall A \subset B \subset S$$

其中，集合函数$f: 2^{|S|} \to R$是从向量空间到实数的映射。随着次模函数的自变量集合内元素扩充，向该集合内加入一个新元素所能带来的增益会越来越小。例如，假设商品i_6和i_7属于同一类目，因而效用函数相同都是$U_j = 1 - a_3^{-k_j}$，当前状态下i_6已经有$\{u_3, u_5, u_7\}$交保，而i_7只有u_3交保，此时如果新增u_8对两个拍品都交保，则边际收益满足：$f(\{u_3\} \cup \{u_8\}) - f(\{u_3\}) \geq f(\{u_3, u_5, u_7\} \cup \{u_8\}) - f(\{u_3, u_5, u_7\})$。上述次模现象在其他淘宝浅库存的业务中也有体现，如闲鱼。那么次模性与问题是否可解的关系是什么呢?

性质 1：函数$F_j : U_j = 1 - a^{-k_j}$满足次模性。

证明：考虑在 maximum 的目标下，次模性 submodularity 与 concavity 等价[Polyhedral aspects of Submodularity, Convexity and Concavity]，不难推导F_j满足下述表达式，即满足了次模性的充分必要条件。

$$f(a + s) - f(a) \geq f(b + s) - f(b), a \leq b, s > 0$$

性质 2：拍下率的期望函数 E 满足次模性。

由性质 1，函数 F_j 具有次模性，又因为不同商品共用同一个向量空间 $X = (x_1, x_2, \ldots, x_n) \in 2^N$，由次模性的 conditioning 性质，即：

Conditioning: FS submodular on V, W subset of V
Then $F'(S) = F(S \cup W)$is sumodular

因而期望 $E = argmax \sum_j P(Y|X_j)$ 也满足次模性。即为拍下率期望 E 的最大化问题存在多项式解法。

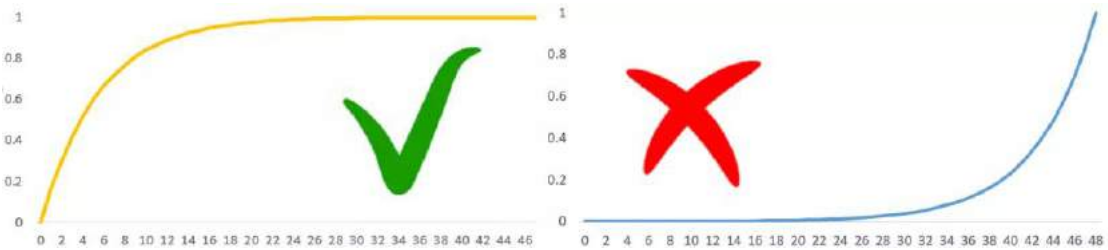
综上，在拍卖问题中，对交保次数与拍下率两个变量围绕边际收益递减的性质展开，在假设效用函数时，需要考虑函数具有次模性(Submodularity)，从而问题可转化为 maximum 下的 concavity 或 minimum 下的 convexity，并进一步设定目标函数和约束条件，再利用拉格朗日对偶性求解。

所以，上述模型适用条件是：

- 浅库存约束
- 效用函数满足次模性

更多 tips：

- 考虑效用函数 $U_j = 1 - a^{-k_j}$，当库存量为 1 时 U_j 为拍下率，当库存量>1 时 U_j 可以理解为商品 j 拍下数的期望值。在拍卖中大多数拍品的库存为 1，但上述模型对库存量>1 的浅库存问题同样适用，这里浅库存定义是商品存量 < 用户需求量，购买者之间存在博弈；
- 同品类下效用函数参数 a 取值相同，此时流量均分是最优策略；在不同品类之间，参数 a 取值不同，当 a 越大时曲线越陡峭，说明边际收益越大，则可以倾向给更多流量；
- λ如何调试？我们分析了λ与以下变量有关：由命题一，品类数越少则越趋近于流量均分策略，从而λ越小；库存越小，同理，流量趋于均分，从而λ越小。
- 如果是求解 maximum 问题，一个直观的次模性函数如图：



新品冷启动

整体思路

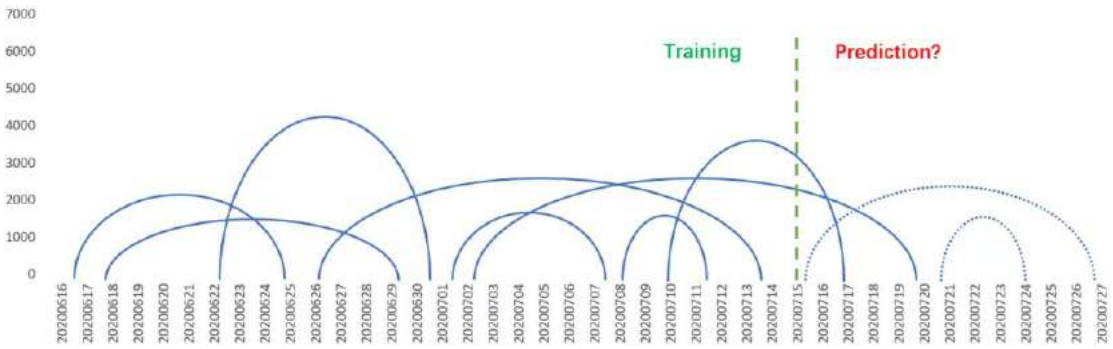
对拍品价值的分析，不论是从点击率还是转化率出发，都离不开日志，但第一步在这里就卡壳。因为拍卖的商品时时刻刻在更迭，大多数 Item 只有一次生命期，不论是流拍还是拍下，都不再出现。除非是循环拍的珍品和法拍房，考虑占比小这里不讨论。商品更迭快，特别体现在字画古玩等高消拍品，大约每天更新 40%。推荐系统始终面对冷启动问题。而且，需要及时更新商品库，延迟超过两天，各种指标就会腰斩。关于如何求解拍卖冷启动问题，我们进行了如下工作：

- 多臂老虎机模型做流量调控
- Knowledge Graph Embedding，基于关键属性对商品表征
- PGM 概率图模型，基于不同品类下关键属性构建 Bayer Network，并进行 inference learning
- 对关键属性有效性的评估

一个例子

拍卖商品的生命期往往是：预热-开拍-结束拍卖，这意味着每天都有大量新品涌入频道页，这也必然给推荐带来冷启动困难。我们先看个例子：

上图选出的是 4 款较受欢迎的爆款拍品，考虑到库存只有一件，实际上这些拍品流量都过载了。从这四个爆款也可以看出，拍品有明确的生命期，考虑更多拍品生命期很短，比较形象的图示如下：



有点像多个叠放在一起的石拱桥。这样的品类更新方式带来了挑战是：

- **模型训练。**比如在离线 CTR 预估模型中，3 天，7 天，15 天的 item 统计特征往往占特征权重的大头，而在拍卖问题中，高消品每天约有 40%的商品更迭。
- **新品孵化。**很多高消品在线时间很短，甚至直到下线依然只有很少流量光顾而最终流拍。目前，拍卖整体流拍率达到 70%，如何及时发现好的拍品，算法急需尝试 Explore & Exploit。
- **工程上保证链路稳定，**一旦超过 2 天未更新商品迟会骤减，整体指标甚至腰斩。

Explore & Exploit

回忆一下，拍品的交易链路是曝光->围观->订阅->交保->出价->成交，其中围观->订阅是拍卖冷启动最关注的阶段，订阅->交保则是流量调控关注的阶段。E&E 算法思路比较简单，只需走通实时链路并定义清楚调控目标，就可以实验。

$$mabScore = \overline{x_j}(t) + \sqrt{\frac{2lnt}{T_{j,t}}}$$

符号	多臂老虎机	个性化推荐
	老虎机	拍品
	手里游戏币个数	每天总围观次数
$T_{j,t}$	在第j个老虎机上已尝试次数	对第j个拍品已围观次数
t	在所有老虎机上已尝试次数	当天总围观次数
$\overline{x_j}(t)$	在第j个老虎机上累计成功的概率	第j个拍品从围观->订阅的转化率

拍品的关键属性

拍卖的结果取决于三个因素，拍卖的规则、拍卖对象的估值、竞标者所有的信息。在阿里拍卖中，拍卖规则始终为公开的英式升价拍卖；竞标者所有信息，可以理解为用户画像，包括用户实时行为，以及用户年龄、性别和购买力等属性。上述信息往往与个性化链路集合，比如用户行为分为淘系的和拍卖站内的，都作为个性化召回链路的 trigger；年龄等属性则作为 rank model 的特征。这与传统个性化链路做法基本一致。

然而，阿里拍卖频道页与传统频道页不同，如上文所分析，用户往往带有捡漏的心智而来，因而首先需要对拍卖对象的估值。作为平台，则需要提供尽可能多的信息便于他准确估值。这些信息在本章中称为关键属性。譬如，钱币的品相、面值、年代，酒类的产地、年份和度数；属性值一般可枚举，例如玉石的种地类型包含{冰种，糯种，冰糯种}。商品详情页如图:

对冷启动缺少日志的商品，关键属性可以部分弥补。但难点是关键属性只适用于部分品类，如酒类的度数只用于酒。从模型训练的角度，更希望采用通用属性做特征；而如果每个类目单独训练一个模型，不仅繁琐且日志也不充分。针对上述问题，我们做了属性评估，并尝试了两种模型来表达，以下分别介绍。

关键属性的评估

同一个品类的关键属性可能看似很多而实际可用很少。这里就需要：

- 属性有作用域，这里选择后台二级类目，只有同二级类目的商品才有可比性。
- 属性是有限集，每个属性的取值也是有限集，例如，手表{机芯类型，表带材质，适用对象，价格范围}，而机芯类型{自动机械表，石英表，手动机械表，机械表}。
- 属性及属性值的选择，需要从以下几个维度判断：
  - 覆盖度：属性及所选属性值包含的商品量级。例如法国产区与酒类强相关，但覆盖不足。
  - 均匀性：有的属性看似商品多但其实不能用，商品分布绝大部分都集中在一个属性值。

chandi_shoubiao_n	invest_level	item_num	ratio
德国	hot	574	0.4429012345679012
瑞士	hot	14475	0.2999937825122795
国内	hot	160	0.299625468164794
欧美	hot	278	0.1674698795180723

- 区格性： 有的属性取值与投资价值没啥关联，使得这些属性值之间没有区格，譬如书法/绘画品类的字画性质：会发现这里的 ratio 非常接近，说明字画性质的原稿和原创没有区格。
- 保序性：比如钻石切工，从完美到一般是有序的集合，所期望对应的热度也是有序的。

Knowledge Graph Embedding

我们通常利用三元组（triple）来表示一条知识，格式为（头实体，关系，尾实体），例如三元祖（阿里巴巴，创始人，马老师）来表述"马老师是阿里巴巴创始人之一"这条知识。随着三元组的不断积累，实体之间的关系愈发丰富，实体、关系类型愈发丰富，所构成的知识图谱（Knowledge Graph）便更加强大。在这个"万物皆可 embedding"的时代，图谱也能够对图上的每个节点、关系都产生一个向量表征，用于下游任务。

知识表示便是一种产出向量表征的方法，该方法主要是将是实体和关系通过某种方式映射到某个向量空间中，进而利用链接预测（link predict）任务的方式结合损失函数进行训练。链接预测任务是给定（头实体，关系）来预测尾实体，或者给定（关系，尾实体）来预测头实体，例如给定（阿里巴巴，创始人），期望能够预测到马老师等创始人而非 xxx，预测正确的 loss 会更小，错误 loss 会更大，进而通过反向传播进行训练。

知识表示发展至今，也有众多方法，其中最经典，最简单的也正是 TransE 方法，该方法将实体和关系都映射到同一个向量空间，通过计算欧几里得范数得到距离，距离越小表示向量运算更为正确，最终损失函数通过 hinge loss 计算得到损失后反向传播至实体与关系，不断的调整实体与关系的向量表征，尽可能使得:

$$f(head, realtion) = tail$$

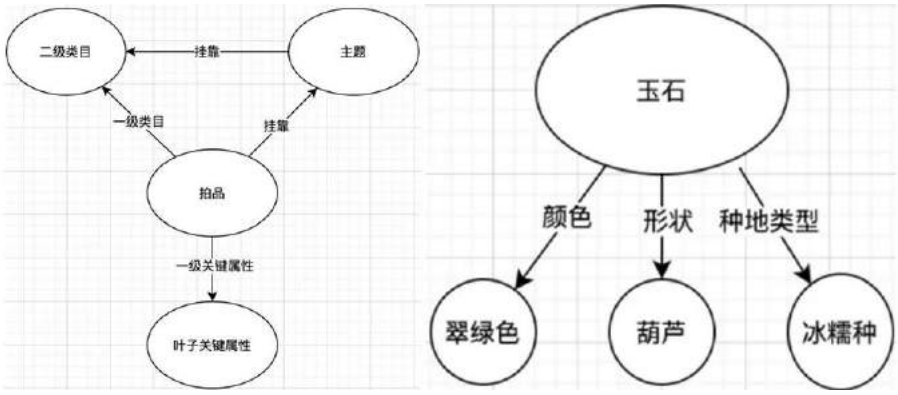
其中距离函数为$d_r(h,t) = ||h + r - t||_{L1/L2}$该距离越小表示该三元组的向量表征越正确。由于该算法是一个无监督的算法，故训练过程采用手动构建负样本并期望正负样本差距较大，且大于某个 margin 即可，如下公式所示：

$$L(h,r,t) = max(0, d_{pos} - d_{neg} + margin)$$

其中d_{pos}为正样本距离，d_{neg}为负样本距离，其中负样本为$S'_{h,r,t} = \{(h',r,t)|h' \in E\} \cup \{(h,r,t')|t' \in E\}$。一般来说，头尾实体不同时替换，可能会替换成一个正样本；且负样本的构造是整个模型训练的关键，负样本过于简单则起不到训练效果，负样本过于难学习可能难以收敛，故负样本构建上本文选择了同叶子类目下进行构建负样本，相比于全局下的负样本构建，在珍品三元组替换到了大资产的房产其实是没有学习价值的。

模型训练后评估过程是针对训练集中的三元组，将待替换实体替换为训练集中的每一个实体，然后计算距离，最后将所有三元组距离进行排序，计算正确的三元组的排名计算得到 Mean Rank 和 Hit@N 进行衡量。其中 Mean Rank 表示排名的平均值，该值越小越好；Hit@N 一般看 Hit@10、Hit@50 等，其表示排名在前 N 个的占比，该值越大越好。

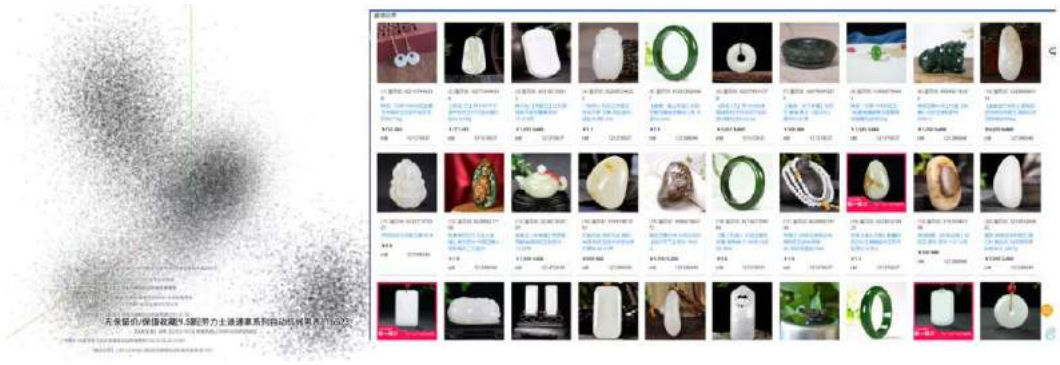
三元组示意图：



根据 embedding 聚合后的可视化结果

左图：向量低维可视化成簇

右图：i2i 召回可视化



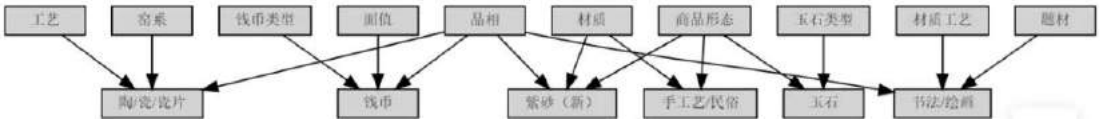
如何使用?

通过关键属性这样的静态特征，使用 transE 算法可得到 Item 的 embedding 特征，可以作为：

1. 模型特征，如离线 CTR 预估的 GBDT 模型，之前在天猫新品业务中尝试，可以提升点击率和新品冷启动效率。
2. 得到 embedding I2I，swing i2i 或是 slim 算法更加依赖行为日志，在日志充分、新品较少的业务并不 work。但在拍卖这样商品更迭频繁，新品占比高的业务，基于 knowledge graph embedding 可以作为补充，提高覆盖率。具体的 AB 效果见后文。

PGM 概率图模型

PGM 一般适用于当特征之间不满足条件独立性假设时，通过特征之间的依存关系构建 graph，并将第一层节点视为先验概率，上游节点视为下游的条件概率分布，并最终计算得到联合概率分布的过程。如图：

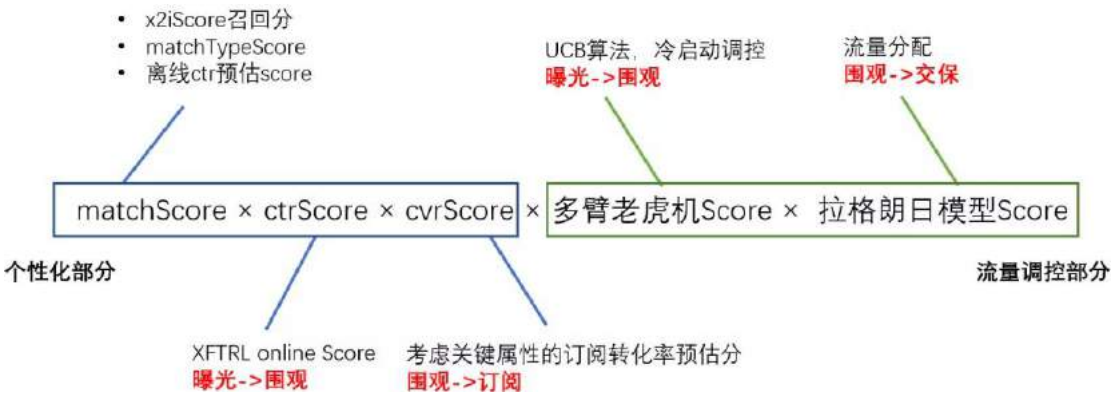


root_node	root_value	root_value_name	prior_prop
caizhi_shoushi	43581	冰糯种	0.2362
caizhi_shoushi	30650	白玉	0.1674
caizhi_shoushi	24776682	羊脂白玉	0.1146
caizhi_shoushi	43580	冰种	0.1281
caizhi_shoushi	55088	碧玉	0.0507
caizhi_shoushi	43582	糯种	0.0133
caizhi_shoushi	30648	青玉	0.0196
tuan_shoushi	43591	佛	0.0476
tuan_shoushi	43590	观音	0.0652

整体流程是：

- 从覆盖率、区格、均匀和保序四个维度做特征评估和选择
- 根据订阅日志，构建先验概率分布和条件概率分布，譬如 P(题材='山水画')，P(订阅='yes'|题材='山水画' & 品相='十品')
- 基于 pgmpy 包，构建 Bayesian Networks
- 做 inference learning，算法采用 Belief propagation，其中参数优化使用 BDeu 评分函数
- 各类目关键属性统一映射到一个 score，可以直接使用，或将其作为 GBDT 模型特征。

流量调控与个性化分数融合



Tips:

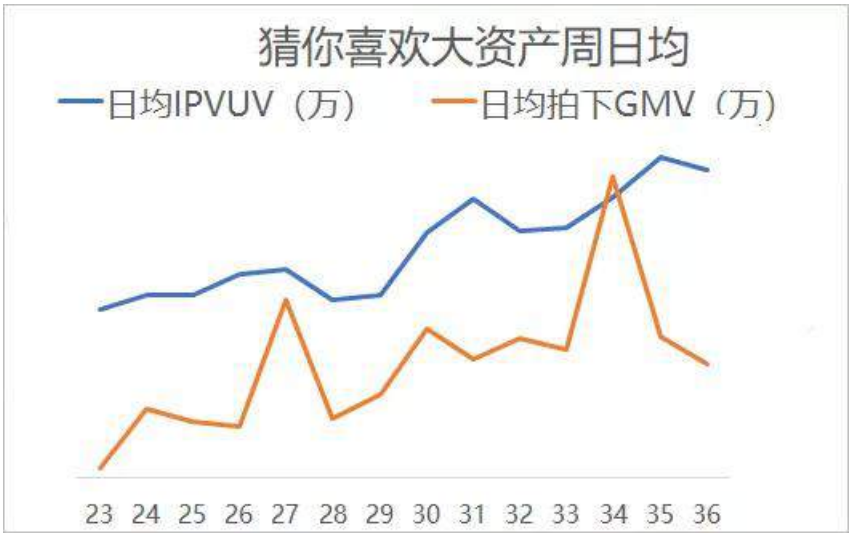
- 流量调控部分的 score，仅限同品类内的比较，比如在比较字画与房产这种跨类目品类，只考虑个性化分。
- 对新用户不做调控，特别是没有一次交保的小白用户。

整体效果

- GMV 和 IPVUV

经过半年的优化，拍卖首页猜你喜欢核心指标变化如下，考虑到 6.10 号有改版（猜你喜欢从第三屏到第二屏），这里可以着重看下近三个月的指标变化。对比六月，八月份日均 GMV+98%，日均引导 IPVUV+28%，拍下 UV+69%，30 天拍品拍下率从+381%，均明显高于大盘增幅。

- **主要驱动是大资产，可以看下大资产 6.10~9.10 的 IPVUV 和 GMV:**



- AB 实验结果明细

下一步工作

从算法角度，一直有两个问题没探讨，其一是用户冷启动；其二是价格。如何让新用户破冰？即第一次交保。除了技术层面，这里还包括信任层面。毕竟拍卖不论是 B 端还是 C 端，都是很专业的事情。不过，猜你喜欢更有义务来承接。其次就是价格，高客单是拍卖的特点，而价格本身也是决定某拍品是否有人出价、最终拍下重要因素；此外，拍卖的品往往用于投资，具有很高的价格弹性。如何在推荐链路中融入价格，既需要，也很困难。

最后是站在业务角度，英式升价拍卖也不一定是在线拍的最优机制，譬如今年诺奖获得者提出的同步多轮拍卖(SMRA)，也是针对"赢家诅咒"问题做了很好的解法。此外，他们还有一个证明是"当竞标者在投标过程中对彼此的估价了解得更多时，这种拍卖形式会给卖家带来更高的预期收入"这一点也值得关注。因为，竞标者对于拍卖品的估价可能与其他竞标者估价相互关联。也就是说，如果某个竞标者对拍卖品的估价发出强烈的信号，其他竞拍者看到这一点，自己的估值信号也会变得强烈。当越来越多的相关私人信息被揭示，传统的升序英式拍卖就会比降序的荷式拍卖产生更高的收益。或许，我们也不能总是独立的看待每个竞拍者，对吧？

特别感谢：淘系商家自运营及智能场景组、淘系商业机器智能团队及阿里拍卖事业部，也希望更多的小伙伴了解和走进阿里拍卖业务。

淘系娱乐化互动技术经验总结

作者 | 渚薰
出品 | 淘系技术

在人口红利开始萎缩，各个 App 都开始以娱乐化互动作为重要手段来争夺普通民众的在线时长。因此在诸如互动业务领域内，我们就需要打造核心轻量化、研发速度快、用户体验好的互动技术。它是 Web 技术的一块特定领域，它融合了图形、渲染、动画以及前端工程和软件工程等知识。本文会把这些领域的知识有条理的梳理一遍呈现给大家。

解惑娱乐互动：认知场景，知道娱乐化互动的概念和意义。

为什么要做娱乐化的互动

想必很多做业务的同学，已经听过无数次产品同学把互动这个词挂在嘴边了。那么为什么一个在电商业务，需要把互动扯进来呢？我们来看看 2020 财年阿里巴巴投资者大会上，一张非常有份量的 PPT ：

PPT 中，列举了不少淘系内的大型互动产品，比如金币庄园、淘宝人生，以及叠猫猫和双 11 战队等大促互动。这些互动无外乎是吸引用户、持续"消耗"用户时间的有效手段。

同时，这类互动也会通过金币、红包、虚拟形象等利益或内容渗透到商业中，直接或间接的带来 GMV 的增长。从战略的角度来说"娱乐化互动产品矩阵、加速拉动用户活跃度提升"，就意味着娱乐化的互动所带来的价值是非常可观的。

用户面前的甜点

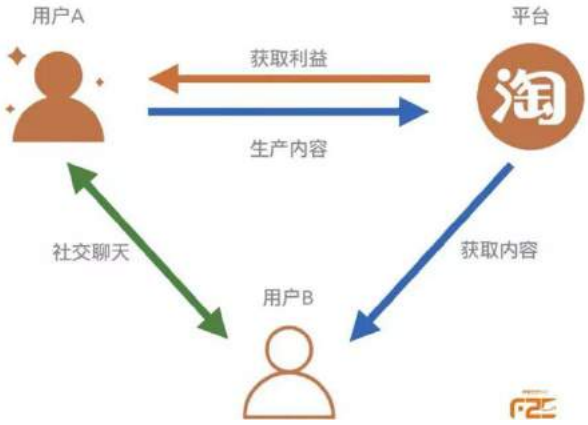
为什么互动会如此吸引用户，这要从用户喜欢的三类事物说起：内容、社交和娱乐。

- 内容：比如直播、短视频、信息流、咨询等是依靠内容本身来吸引用户的。
- 社交：比如朋友圈、陌生人交友等是通过人与人之间产生联系来吸引用户的。
- 娱乐：比如影音、书籍、游戏等是通过赋予各类知识、情感享受来吸引用户的。

我们常常会发现，内容被社交化，比如抖音。又会发现娱乐被社交化，比如王者荣耀。很少会发现独立某类事物的产品存在，都是融合了内容、社交和娱乐的综合产物。

用户是如何互动的

互动可以理解为一种行为，它把内容、社交这类事物的产生在人人之间、人与平台之间传递起来。比如这张图：



用户和用户之间，用户和平台之间想要得到这些事物，就需要“互动”。这种过程它可能持续很久，也可能是一瞬间的事情。娱乐化能让这些过程变得更有趣味性，比如借助好玩的形式获得平台发出的红包，又比如用双 11 战队互相 PK 的方式让用户间完成社交聊天《知乎-如何看待 2019 双 11 盖楼挑战？》。种种有趣的玩法和现象下，用户乐于其中并最终获得自己想要的事物，这就是我认为的娱乐化互动的意义。

所以总结下，互动并不是一种事物，娱乐化互动也并不是游戏，它是一种有趣、好玩的获得目标事物的过程。这个过程，用趣味的场景或玩法是非常有效的手段。

如何阅读正文

说了这么多，还只是个引子。接下来的内容都是基于这几年淘系互动团队在互动技术上的沉淀和经验阐述的。如果，你已经是一位资深的互动技术架构师，希望你能找到互动技术和业务的结合点，跨领域形成融合。如果，你是一位互动业务的前端，希望你能分析自身业务处于哪一个阶段、哪一个层次，并更加助力你的业务。如果，你仍然是一位非互动业务的前端，希望你能拓展知识领域、储备相关技能，迎接娱乐化浪潮。

解读知识领域：勾勒互动技术的图谱，跨领域拓展知识。本文接下来从前端开发者的视觉来科普互动领域的技术体系。

技术知识图谱

要开发互动 H5，到底需要学习多少知识？网传曾有怎么一份知识树（出自 miloyip 的 Game Programmer）。这份知识树从最基础的知识到专业技能覆盖的非常全，但这肯定会“逼退”很多想要进入这个领域的前端。但我的目标是让前端开发者能开发娱乐化的互动，而不是真正的开发一款游戏，所有我就对这些知识进行了重新归类整理。

第一，编程基本功是每位技术同学都需要掌握的，特别是计算机科学和软件开发，和具体的技术栈或者专业知识领域不强相关。也就是说，这一项对于大多数前端同学来说，是必选项，难度可以给一颗星。

第二，实战应用技巧是对一个专业领域开发模式的总结，有一些不同于其他领域的技巧、架构和思维。学习开发模式可以从学习该领域的一个框架开始，就像你在学习一个 MVVM 框架（React、Vue、Angular）一样。业界这样的框架或引擎有很多，比如大名鼎鼎的 Unity，以及 Web Game 中有着广泛好评的 Phaser、Egret、Laya，以及阿里已开源的 Hilo 和孵化中的 EVA 。所以在这一项上，对于前端同学来说，可以找准一款框架进行学习或深入研究，难度可以给一星半。

第三，数学/物理学是所有科学的基础^-^。互动中的基础科学，同样包括了代数、几何、力学、声学、光学等等。这些知识在高中、大学的教学课本里基本都会涉及到。掌握基本知识后可以大致理解各种模型设计，深入研究后可以挖掘更加富有创意的模型。在这一项上，可能大家在毕业后会渐渐丢弃，重拾还是有点难度的，可以给二颗星。

第四，图形学是渲染的底层技术，AI 是某类玩法的核心要素。这些知识只有当你准备去从事这方面的项目开发时，才会专门用到，又或者最初只需了解一二即可。当然，它对于前端技术栈的同学来说，完全是一个从未涉足过的新领域，难度可以给到三颗星。

重构学习之路

经过分类归纳后，就可以梳理需要学习知识以及方式方法。

首先，如果觉得自己基本功还不行，就恶补它，做到能熟练掌握编程语言、数据结构、算法、容器特性，以及设计模式、敏捷开发等软件工程的基础理论，当然也不能纸上谈兵，得实际应用在日常开发中。然后，适时掌握一个或两个你觉得最优秀的框架，在实际项目中使用它，并吃透它。这样下来，你就是一名合格的互动业务前端开发者了。

假如编写逻辑并不能满足你悸动的心，那么我们就继续：回去翻翻高中大学课本，看看已经放下了许久的知识，当你能熟练掌握了大学课本之后，通过这些知识，就能帮助你去理解和认知技术名词和代码实现。然后再苦心钻研 GPU 工作原理、图形语言的精妙之处，或许你就能创作一些有趣的特效出来。

再然后，如果你的需求方希望你研发一款在线实时多人对战的玩法，那么你就硬着头皮去看下 AI 机器人的专题，那么就可以写出一些看起来并不那么笨的 AI 机器人出现在真实玩家中间。 最后，理想是当一名架构师的话，那请把上面的这些阶段和知识都精通吧。

明确角色和职责

学习目标和结果产出，也和你想要担任的角色和职责是相关联。

假如你非常热衷于投入到业务的研发中，那么作为一名开发者，你的职责就是提供可持续迭代、低代码研发的解决方案。

假如你对美有特殊的见解，希望创作神奇梦幻般的效果，那么作为一名技术美术，你的职责就是提供辅助创作的艺术工具以及许多精美的预设素材。

假如你总是操着一份想要制定标准的心，那么作为一名架构师，你的职责就是提供完备稳定的顶层设计和架构方案。

解剖互动技术：深入互动去理解架构，不要放过任何细节。 前文中有提到淘系团队正在孵化的 EVA，名字听上去挺耳熟，不过这个不重要，重要的是我会借 EVA 的架构为大家剖析下互动的部分技术方案，以便大家对互动应该做什么技术有所认知。

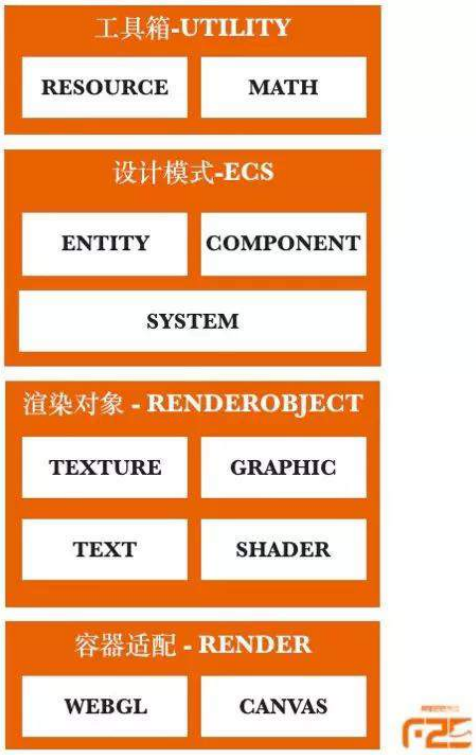
从体系到架构

淘系业务很复杂，整个前端体系也非常庞大，但好在有统一的业务平台去支撑产品运营侧的需求，以及有淘系大生态来保证工程、构建、容器、Web 框架等。有了这些强大的靠背，对于互动技术来说，更多的关心互动域自身的研发平台和核心基础。先看这张大图：

挑其中两个来分析下它们的架构设计、分别是：EVA JS，EVA Store。

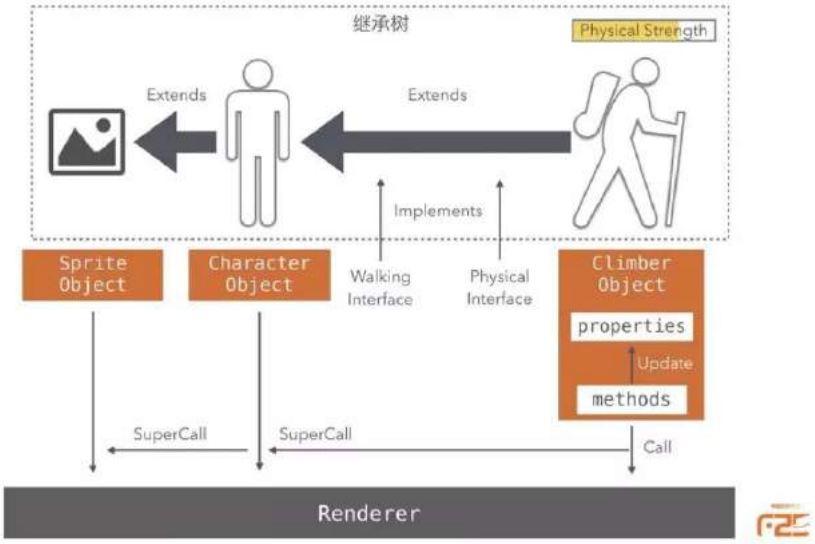
EVA JS 设计模式

EVA JS 也就是互动框架，为了承载互动业务的轻量化引擎，基本采用了业界成熟的架构实现，拥有 ECS、Loader、场景等基础能力，依赖 Hilo 渲染，并提供丰富的扩展插件。核心部分的系统分层大体是这样的：

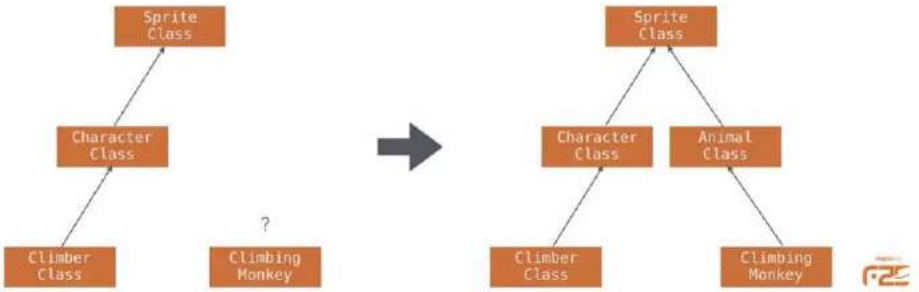


其中，ECS 是 EVA Core 中的核心，全名 Entity Component System，是一种设计模式，也是参考了游戏领域的架构设计，和我们常见的 OOP+MVC 有着比较大的差异。

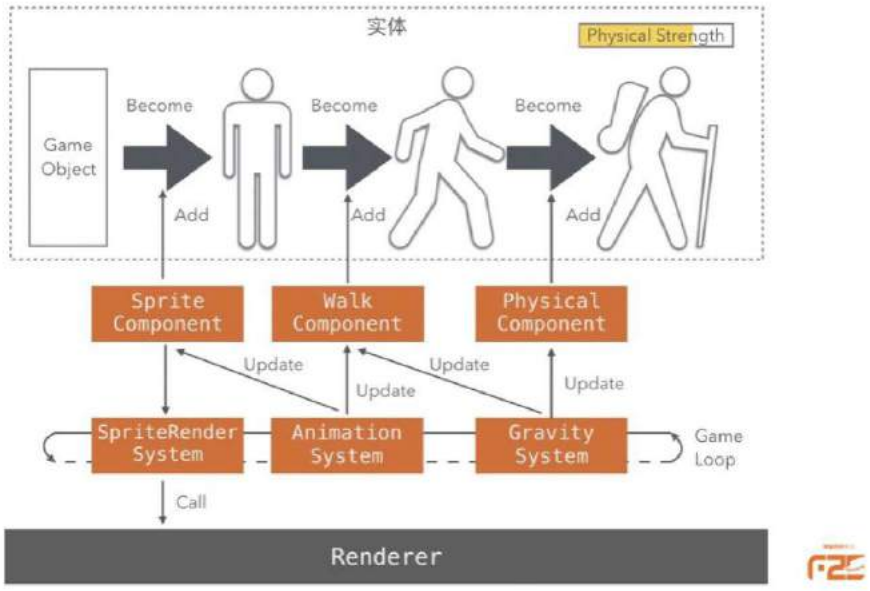
举个生动的例子，在程序里创建一个爬山者，赋予它体力的属性和行走的能力，并且行走会消耗体力。在面向对象的设计模式下，编程方式就是通过对继承和实现接口来完成的。



从子类到基类，需要逐步抽象具体行为。那么问题来了，因为继承关系是一颗树，当遇到某个具体对象（叶子节点）无法继承某个基类（父节点）时，那么就必须再创造一个基类（另一个父节点），并且寻找可以挂载的祖父节点。例如，我们通过对爬山者的抽象，创造了一个人的基类。人的基类继承自渲染对象的基类下，此时如果要再实现一个爬山猴，那么就又要创造一个动物的基类，同样继承自渲染对象的基类。



但它们两者的很多行为都是一样的，比如体力属性、行走能力。这些行为虽然可以用接口的方式实现，但不免就显得很繁琐。因为，每设计一个类（增加了一个节点），就增加了维护一棵树的成本（它总是会被继承的）。那么 ECS 又是怎么做的呢?



在最初，爬山者就是一个空对象。当给这个空对象添加了一个渲染组件，并且提供了一张人的图片时，它就变成了一个人。接着添加一个行走组件，就变成了一个可以行走的人。最后添加一个体力组件，就变成了一个有体力的可行走的人。接下来，系统就开始登场了，当重力系统介入的时候，这个人开始消耗体力了；当动画系统介入时，行走的人动起来了；当渲染系统介入时，当前行走的状态就显示在了屏幕上。这些系统在一个循环里不断运行，每一帧都在改变这些组件的数据和状态。

那么如何实现爬山猴呢？仅仅只需要再新建一个空对象，添加这些组件，并赋予不同的参数值就可以了。

这个时候应该不难察觉，在一个场景里实体是非常多的，比起维护继承树，维护能力组件的成本要低的多。这里并不是在比较这两种设计模式的优劣，而是诠释在特定的场景下需要用合适的设计。ECS 不光摆脱了继承树的负担，它还减少了数据通信的成本。

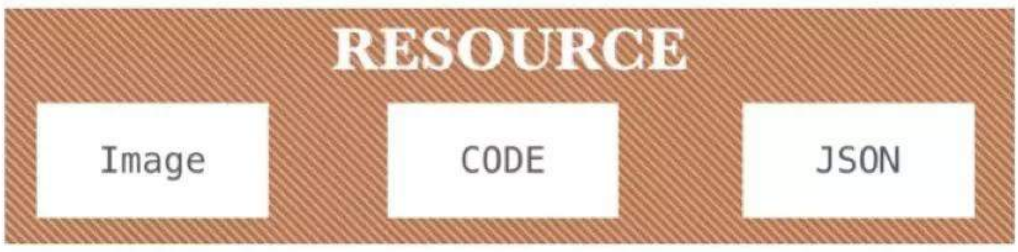
因 Game Loop 的存在，改变数据后每一帧读取到的都是最新的数据，所以并不需要用单向绑定或事件机制来触发的数据对视图的刷新。因此也就不需要 MVC 当中的 Controller，整个 ECS 可看成一个自运作自更新的 View，只是数据存放在了 Component 里，改变数据的 Action 存放在了 System 里。

EVA Store 设计模式

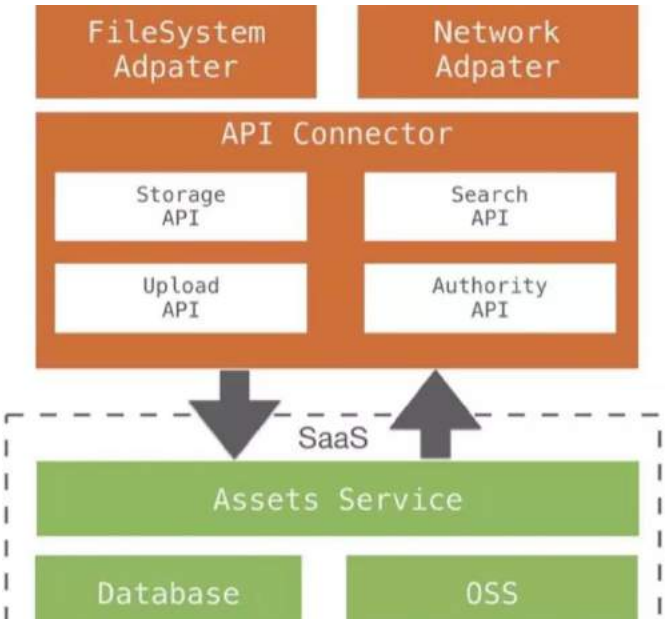
在互动中需要大量的素材，例如雪碧图（Sprite）、雪碧图动画（SpriteAnimation）、骨骼动画（SkeletalAnimation）等。

它们自然是交给引擎来读取并展示的。但从原始资源到可被引擎读取的素材之间，实际会需要一些预处理，比如对资源的检测、对图片的压缩、对数据的优化等。一些引擎会附带一些工具来对原始资源进行预处理。EVA Store 一方面成为这类资源输入和素材输出的集散地，另一方面集合多种素材预处理工具，可以给视觉和开发同学带来许多便捷。

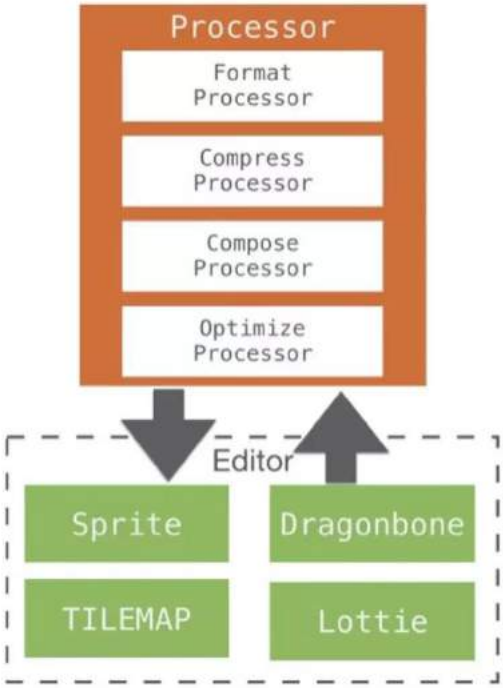
EVA Store 接收到的是一些原始资源，比如设计师给你切好的图片、从专业软件导出的动画文件（一般包含图片和 JSON），或者是一份 Shader 代码。



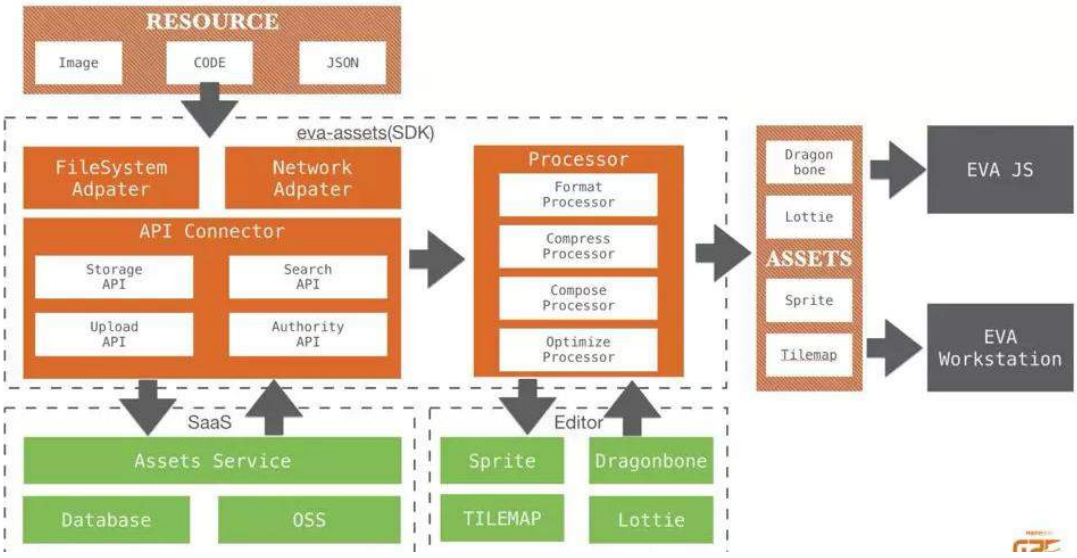
进入 EVA Store 时，首先需要把这些资源落库方便进行管理，通常会基于一套 SaaS 服务来管理和检索它们。



落库后，这些资源会被交由预处理工具来进行针对性的处理，比如可以将图片合成雪碧图、可以检测龙骨的正确性、以及最后对图片和数据的压缩优化等。这些预处理，既可以是自动化的，也可以通过一些编辑器来可视化的进行。



最后输出的就是符合引擎和工作站要求的素材。



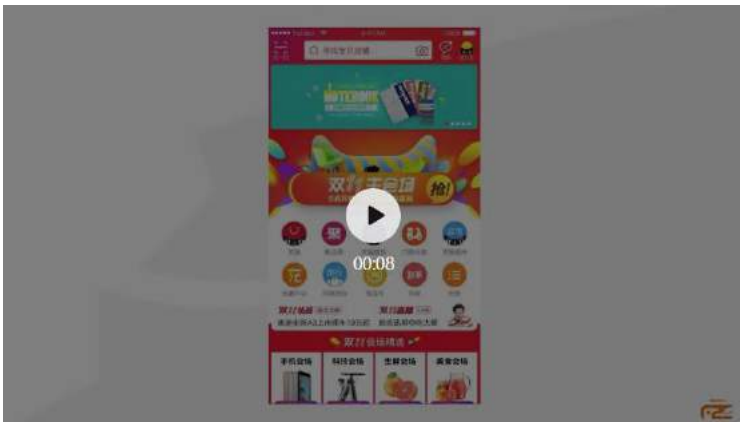
图中可以清楚的看到从原始资源到标准素材的流进路线，其中核心部分就是 eva-assets 。它是一个 JS SDK ，集中了 SaaS 服务调用、资源预处理，文件系统和网络的适配器，使其既能在 Web 环境运行，也能在 Node 环境运行。

EVA Store 基于这个 SDK 完成素材中心网站的功能接入。它也可以被 Editor 或 EVA Workstation 使用来管理和处理这些素材。 如果这些架构的剖析看的还不过瘾，可以关注后续淘系互动出品的更多技术分享和文章。

洞察时代演变：无论是坚持业务开发的同学，还是专精于架构研究的同学，都需要会创造价值。 所有业务都不是一朝变成现在的形态的，淘系互动业务近三年也经历了一些变化，我把它们简单划分成三个时代。

动画时代

在这个时代，产品希望在原本枯燥的界面上，用动画来丰富需要表达的事物。例如，大促开始时的一段揭幕动画或是引导点击按钮的一段引流动画。这个时代的动画，基本不需要太多交互，展示给用户动画播完就结束了。



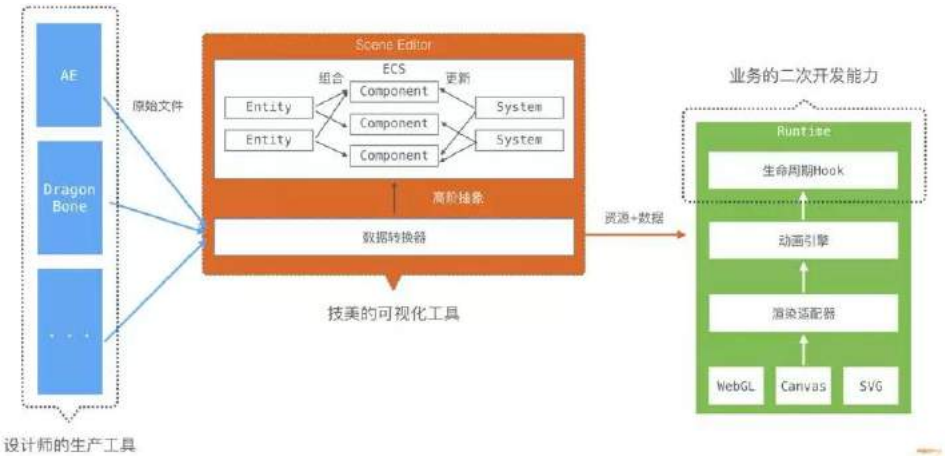
点击可跳转原文查看视频详情

针对这样的产品需求，前端的解决方案会有这么三种：

实际开发中，在拿到设计师的动画 Demo 稿或者仅有产品提供的动画示意稿时，真要把它还原出来可能要花费大量的精力，有时跟 Demo 稿比起来也会差强人意。

所以这个时代，重点要解决生产品质的问题，也就是动画还原的精细度。这时候，你可能会想到古老的 Flash，这个在 90 年代特别风靡的动画制作软件，设计师制作完动画后，工程师们可以通过 ActionScript 来执行一些业务逻辑，并轻松的运行在 Web 环境下。

的确，Flash 的制作和开发工作流是非常"先进"的，它让设计师和工程师们能做自己最擅长的事情，且互相配合的非常默契。那么在当下，是否还有这样的工作流可以让它们各司其职呢？答案当然是有的。



在上图中，设计师的生产工具变得更加专业和强大，比如 After Effect、DragonBone，这让设计师们拥有了广阔的创作空间。这些生产工具可以保存为源文件或者导出视频、帧序列、GIF 等。

但源文件不能在 Web 上运行，视频、帧序列、GIF 也不是我们想要的。这就需要一个把源文件转成对 Web 友好的动画格式（图片资源+JSON 数据），这一步很多生产工具也会内置进去。

但在一些复杂的动画场景里，需要多段动画或多种动画，这个时候就需要有一个场景编辑器来组合这些动画。比如 Unity 编辑器就可以是这样的一个场景编辑器，它可以把多种动画拼接在一起，变成一个完整的动画场景。之后，把重组的资源和数据再导出给 Web。

在 Web 上，就需要一套能解析这个动画格式的引擎，它可以被叫做动画引擎。这个动画引擎在运行时解析数据、分析数据，并逐帧绘制到画布上，完成动画内容的播放。迄今为止，这样的动画引擎也是非常多的，而绝大多数引擎也会提供支持播放这些动画的插件，EVA JS 同样可以支持雪碧图动画、骨骼动画、Lottie 动画等常见的动画格式。

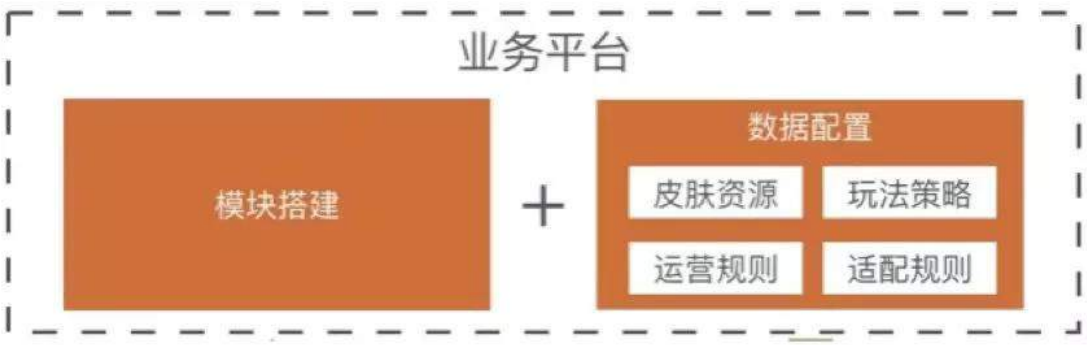
借助这样的工作流和工作模式，能 99%的还原动画精细度，且因为过程中并不需要开发者来还原动画展现，整体的效率也能得到极大的提升。 大家如果对动画的制作和原理感兴趣，也可以顺道去阅读我早几年在大会上分享的两个话题：《畅想动画制作的乐趣》和《渐进式动画解决方案》。

互动时代

来到了互动时代，这个时候产品已经不满足单纯的动画展示给用户，而是期望和用户产生更多的交互。在动画时代沉淀下来的工作流和引擎，基本能满足这类需求。然后核心问题是，在不同的大促、不同的活动中，产品希望能复用这些作品，加以变化并且能快速上线，设计和开发也因为花了较大精力去完成某个作品，希望能让它发挥更多的价值。所以这个时代，重点要解决生产关系的问题。

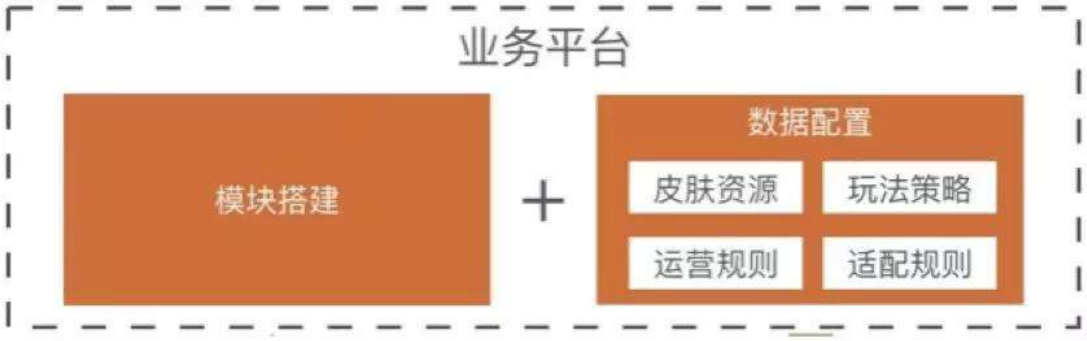
解决这样的问题，最佳实践通常是利用模块化搭建的方式。把最初的项目沉淀为一个模块，这个模块封装了主体逻辑，并提供了可运营化的服务功能。

在沉淀这个模块的时候，一定要和产品以及设计充分沟通，把整个作品以产品化的方式进行重新梳理，并且预想可复用的场景，增加模块的灵活性。之后需要有个业务平台面向这类需求以搭建的方式选择这些模块并配置相关的数据。



模块和业务平台的关系是一对多的，也就是交付给业务平台的模块，是统一标准、统一体系的，不同业务特性可以架设不同的平台，走不同的投放逻辑和运营策略，这些是业务特有的，并不附属于模块本身。

搭建的页面中，除了互动类的模块还可以添加其他商品类的模块等等。这样带来的好处是，作为开发和设计同学只需面向模块开发、交付符合标准的模块即可，并不需要面向某个业务项目进行开发。



有了这样一层关系后，以往优秀的互动产品就可以通过模块的方式，赋能给更多的业务，从而发挥更大的价值。

点击可跳转原文查看视频详情

娱乐化时代

到了现今，已经全面进入了娱乐化的时代。在这个时代，商业和互动结合的非常紧密。淘系在 2 年前渐渐孵化并生长出多个千万级别的娱乐化互动产品，包括金币庄园、天猫农场、淘宝人生等。这些产品不约而同的用娱乐化互动的手段，同自运营积分体系、全网商家、特色卖家等相互合作联通，实现互动流量变现的高价值。在这些产品的孕育和发展中，比起以往仅存在于大促活动的互动项目，有着维护成本上的差异。所以在这个时代，重点要解决生产迭代的问题。

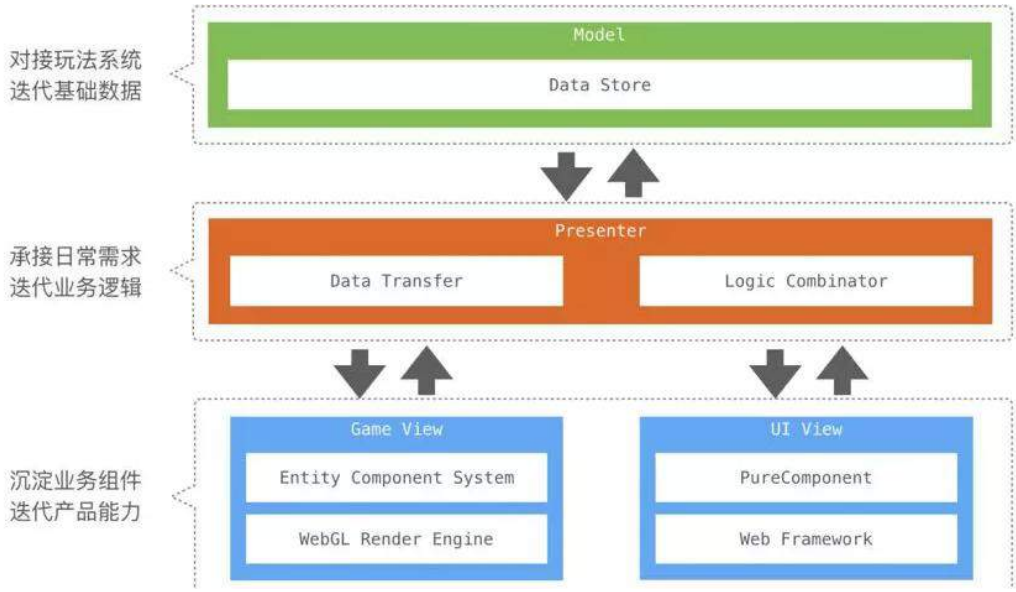
但这里说的迭代，并不是想说把项目怎么维护好，单测搞起来，做敏捷开发之类的事情。因为迭代的意义是能给让业务不断的往前推进，而不单只是让代码不出问题。所以，在分析整个业务推进的过程中，有几个方面是需要去关注的。

第一，产品要和业务的区分开，产品是承载功能，而业务是把功能在某个时间段以某种外化的形式表达给用户。比如，弹窗组件是一个产品，而金币庄园升级的提示弹窗就是个业务。所以通常这类产品组件，也可以叫做业务组件，它高度抽象业务中的某些行为，并长期沉淀提供给业务日常运营使用。这种业务组件其实和互动时代的模块很像，也是同一种产品化的思路，只不过颗粒度可能更小一点。在项目长期的运作中，我们都要面向某种产品能力进行迭代，而不要陷入单次的业务逻辑中。

第二，在有了业务组件后，业务逻辑就可以架在业务组件之上，专注于短周期内的迭代。这部分业务逻辑还可以用编排的方式来提高效率。关于逻辑编排的技术和方案，淘系互动团队会在之后的文章中为大家分享。

第三，业务不断向前发展，背后的数据也在不断的产生变化来适应业务的需求。数据变化的周期相对来说是要长一点的，而一旦发生变化，就会牵连到业务组件以及业务逻辑。我们通常希望，服务端在返回数据时，尽量不要去动原来的数据结构，只增加类型或可枚举的值，事实当然不可能经常如愿以偿。因此也需要适时的抽象产品能力，让一个基础的系统来提供稳定数据形态。在互动中，比如有任务系统、养成系统、集换系统等等。这些基础系统的数据在一定周期内是稳定的，并且可以满足不同的业务需求，完成不同的项目。

当需要面向多个独立迭代的事物时，前端在代码上的分层架构也要能适时的去配合这些迭代，并减少相互间的耦合。大家可能已经想到了 MVP（或 MVC）架构。



上面的图基本很清楚了，不再赘述，只不过在娱乐化互动里，有专门的一个 Game View 来展示互动场景，同时会有一个 UI View 来展示交互界面。至于为什么要把互动场景和交互界面区分开，主要是因为互动场景是逐帧刷新的，而交互界面不需要这么高频的刷新，分离后能使用各自合适的设计模式来处理。比如互动场景就可以用 ECS，而交互界面可以用 PureComponent。

人人可开发、处处有互动

到文末了，不免想打个小广告，淘系技术部前端互动团队的使命是"人人可开发、处处有互动"。基于这样的目标，我们会提供统一的互动产品和研发平台来支撑互动业务的研发。同时整个淘系技术部前端团队，以技术驱动商业发展，不设界限的在前端相关领域深入突破，如互动、搭建、AI、Node、Serverless、中后台、云研发、终端技术等领域，以技术探索未来。

如果你也想一起定义未来，就快点加入我们一起改变世界吧！简历投递到邮箱：zhuxun.jb@alibaba-inc.com。

链接：

https://www.zhihu.com/question/352540203/answer/874963931

https://img.alicdn.com/tfs/TB1rEeExaL7gK0jSZFBXXXZZpXa-810-19136.png

https://github.com/miloyip/game-programmer

https://unity.com/

http://phaser.io/

https://www.egret.com/

http://layabox.com/

https://yq.aliyun.com/articles/69561

https://yq.aliyun.com/download/166

DinamicX：手淘信息无障碍技术方案全解

作者 | 大饱
出品 | 淘系技术

目前中国有 1700 多万视障人士，他们渴望购物，也希望在任何情况下都能平等的获取他们想要的信息，手淘作为全国最大的购物 App，我们也希望通过技术让视障消费者能更好的享受移动互联带来的便利，这既是公益，也是义务。
本文将和大家分享手淘在使用 DinamicX 支持无障碍的技术方案，并给出了相关示例，希望对移动端开发者有所启发。

什么是无障碍?

无障碍范围很广，一般是指在发展过程中没有阻碍，活动能够顺利进行。比如给腿脚不便的人在一些公共场合比如火车站、机场、商场等地方设置无障碍电梯，无障碍厕所，或者给听觉障碍的人提供助听器等等。

换句话说：为失能人士提供与非失能人士同等机会。这里所说的失能根据具体形式和严重程度各不相同，但主要可以分为四种：认知、视觉、听觉，以及活动能力。
当然失能也分为两种永久性失能和情境性失能。

- **永久性失能**：
  - 视觉障碍、听觉障碍、坐轮椅或者行动不便等肢体障
- **情境性失能**：
  - 开车的时候试图用手机：汽车晃动导致情境性的视觉障碍、肢体障碍、注意力障碍
  - 开会时聊天软件发来语音：查看消息发出声音会影响周围同事（好的聊天软件提供语音转文字的功能便是无障碍的一种）
  - 出国旅游语言不通：情境性的口头沟通障碍

- 买东西时拎着大包小包：情境性的肢体障碍

无障碍的新定义：确保每个用户意图都被理解

某种程度而言，信息无障碍是智能产品交互设计中针对特殊人群的一个功能，它可以让人们更加平等地享用产品在硬件和软件上的各项功能。

阿里无障碍的历程

DinamicX 对无障碍的支持

目标与定位

通过技术手段，帮助视障人群更好地感受世界的美好，让用户在使用 app 的时候能够顺畅的获取信息、利用信息。

DinamicX

DinamicX SDK 作为支撑手淘基础核心链路重要的一环，首当其冲肩负起支持无障碍的功能。

手淘基础核心链路?

手淘首页、详情、购物车、下单、订单、订单列表、我的淘宝都属于手淘核心链路，目前上述页面 UI 都是使用 DinamicX 作为渲染引擎来绘制。

什么是 DinamicX?

DinamicX 的定位是一个提供三端统一能力的客户端动态化解决方案，为无线基础链路上的高性能和高可用提供基础保障。我们希望通过社区化运营不断丰富 DinamicX 的能力和内容，提高渲染性能和稳定性，将 DinamicX 打造成一个集团内的客户端动态化体系的标准化方案。

动态模板解决方案核心技术：一个包含模板的下载、加载、解析、渲染的引擎，帮你动态生成 View。

讲完 DinamicX，接下来我们讲 DinamicX 对无障碍的支持。

DinamicX 对无障碍的支持主要分为两部分：

- SDK 本身对无障碍的跨平台支持
- 模板开发平台进行卡口校验

作为一个跨平台统一的动态化解决方案，势必要抹平端与端的差异，以及降低业务方(模板开发者)想支持无障碍的认知成本，我们团队全体成员包括 Android、iOS、以及测试同学讨论了很久，确保两端统一的情况下，勾勒出统一无障碍行为。

技术方案

系统原生的无障碍

iOS 系统原生的无障碍

原生 iOS 的几种逻辑：

- View 设置了 isAccessibilityElement=YES，无论是否设置了 accesibilityLabel，所有它的子节点，都不可获得焦点；
- UILabel 的 isAccessibilityElement 属性默认是 NO，但只要主动地设置过值，就算设置的是 NO，也无法在父容器下自动读出；
- 如果需要父容器获得焦点后自动读取出里面所有 UILabel 的文字，需要 isAccessibilityElement = NO，并且 accessibilityElementsHidden = NO。Label 的 isAccessibilityElement 必须保持原始默认值，不能设置任何值；
- 如果父节点嵌套，并且所有父节点的 accessibilityElement 都设置为 off，会自动将这个父节点所有的子节点的 TextView 的 accessibilityLabel 顺序读出，这意味着所有自动阅读的文字最终都在根节点上被读出。

系统无障碍 API

API	说明	举例
accessibilityLabel	给当前view设置无障碍需要朗读的信息	_view.accessibilityLabel = _accessibilityText
accessibilityHint	动作提示 如'双击打开'	_view.accessibilityHint = _accessibilityHint
accessibilityValue	读控件本身的值，如选中/未选中	_view.accessibilityValue = _accessibilityValue
isAccessibilityElement	控制View是否能启用无障碍功能	_view.isAccessibilityElement = YES 开启无障碍功能可以获得焦点
		_view.isAccessibilityElement = NO 不可以获得焦点
accessibilityElementsHidden	控制View以及子View是否隐藏无障碍功能	_view.accessibilityElementsHidden = YES关闭该控件以及子控件的所有无障碍功能
		_view.accessibilityElementsHidden = NO 开启该控件以及子控件的所有无障碍功能

Android 系统原生的无障碍

Android 的 View 无障碍状态总共分为 3 种：

- **没有无障碍信息，**如 ImageView、View 等 默认就是没有无障碍信息
- **有无障碍信息，**如 ImageView 设置 setContentDescription，或者 TextView 自带无障碍信息就是它本身的 text
- **有无障碍信息的可交互控件，**比如 ImageView 设置 setContentDescription 的同时，又设置了 setOnClickListener， 比如 TextView 设置了 setFocusable(true)，或者 EditText、CheckBox 这种默认就是有无障碍信息的可交互控件

这 3 种无障碍状态在它的父 layout 之中的关系：

		Layout		
	setImportantForAccessibility	YES	YES	NO
	setContentDescription	不设置	设置	无效
		若有控件B 则可选中，且朗读控件B的信息 若没有控件B 则不可选中	可选中，朗读设置的文字	不可选中，不可朗读
child	无无障碍信息 控件A	不可选中，不可朗读	不可选中，不可朗读	不可选中，不可朗读
	有无障碍信息 控件B	不可选中，可朗读，其父区域被选中	不可选中，不可朗读	可选中，可朗读
	有无障碍信息的可交互 控件C	可选中，可朗读	可选中，可朗读	可选中，可朗读

系统无障碍 API

API	说明	举例
setContentDescription	给当前view设置无障碍需要朗读的信息	view.setContentDescription("购买")
setImportantForAccessibility	控制View是否能启用无障碍功能	view.setImportantForAccessibility (View.IMPORTANT_FOR_ACCESSIBILITY_YES) 开启无障碍功能
		view.setImportantForAccessibility (View.IMPORTANT_FOR_ACCESSIBILITY_AUTO) 系统决定
		view.setImportantForAccessibility (View.IMPORTANT_FOR_ACCESSIBILITY_NO) 不启动
		view.setImportantForAccessibility (View.IMPORTANT_FOR_ACCESSIBILITY_NO_HIDE_DESCENDANTS) 不启动当前以及子View的无障碍功能
setFocusable	支持当前View获得焦点	setFocusable(true)

DinamicX SDK 定义无障碍属性

抹平两端差异，简化无障碍逻辑，DinamicX 提供了两个无障碍属性来支持无障碍功能。

属性名称	描述	值类型	备注
accessibility	当前节点及子节点是否可获得无障碍焦点	on	当前节点可获得无障碍焦点，若该节点为Layout，则子节点都无法获得焦点
		off	当前节点无法获得无障碍焦点，若该节点为Layout，则子节点根据配置决定是否可获得焦点
		off_child	当前节点无法获得无障碍焦点，若该节点为Layout，则子节点也无法获得焦点
		auto	取决于系统控件的默认值，若该节点为Layout，则可选中，并朗读内部所有没有主动设置无障碍的Text
accessibilityText	给当前view设置无障碍需要朗读的信息	例："跳往详情"	此时点击该控件，会读出跳往详情的语音

xml 示例

如下表示该控件在触摸到的时候，会被选中，且朗读出"跳往详情页"的文案

```
    <ImageView
      width="100"
      height="100"
      accessibility="on"
      accessibilityText="跳往详情页"
      onTap="@openUrl{'detail'}"
      imageUrl="https://img.alicdn.com/tfs/TB1FuMQQFXXXXXLXXXXXXXXXXXX-420-420.jpg"
/>
```

统一两端无障碍行为

下图代表的是两端目前统一行为，描述了 Layout 与子节点在无障碍属性各种 value 值碰撞下的情况。

		Layout			
		on	off	auto	off_child
		可选中，不朗读内部任何Text	不可选中	可选中，朗读内部所有没有主动设置无障碍的Text	不可选中
child	on	不可选中	可选中		不可选中
	off		不可选中		
	auto		取决于系统控件的默认值		
	off_child		不可选中		

端上的处理

为达到上图所展示的两端一致的行为，端上各自做了自己的处理。

iOS

下图表示 SDK 根据模板属性到系统 API 的映射

	accessibility			
	on	off	auto	off_child
isAccessibilityElement	☑	☐	☐	☐
accessibilityElementsHidden	☐	☐	☐	☑

Android

以下图片表示 SDK 根据模板属性到系统 API 的映射：

Android 对 Layout 和非 Layout 的 View 需要区别对待。

Layout 节点对无障碍的处理：

	accessibility			
	on	off	auto	off_child
setImportantForAccessibility	YES	NO	YES&setContentDescription(null)	NO_HIDE_DESCENDANTS
setFocusable	YES	不处理	不处理	不处理

非 Layout 节点对无障碍的处理：

	accessibility			
	on	off	auto	off_child
setImportantForAccessibility	YES	NO	不处理	NO
setFocusable	YES	不处理	不处理	不处理

案例演示

模板示例：

```
<LinearLayout
backgroundColor="#eeeeee"
height="match_content"
width="375"
orientation="vertical"
disableFlatten="true"
>
<LinearLayout
marginLeft="@triple{@data{cellType},20,50}"
backgroundColor="#f2f2f2"
height="match_content"
width="match_parent"
orientation="vertical"
disableFlatten="true"
accessibility="auto"
>
<!--auto 代表点击的时候，该 layout 下面的 text 信息都可以读出来-->
<TextView
width="match_content"
```

```
height="match_content"
textColor="#ff051b28"
textSize="12"
marginTop="20"
marginBottom="20"
text="这是一个 textView"
/>

<TextView
width="match_content"
height="match_content"
textColor="#ff051b28"
textSize="12"
marginTop="20"
marginBottom="20"
text="这是一个有焦点的 textView"
accessibility="on"
onTap="@rTap{}"
accessibilityText="这是一个有焦点的 textView"
/>

 <FastTextView
width="match_content"
height="match_content"
 textColor="#ff051b28"
textSize="12"
marginTop="20"
marginBottom="20"
text="这是一个 FastTextView"
/>

<TextView
width="match_content"
height="match_content"
textColor="#ff051b28"
textSize="12"
marginTop="20"
marginBottom="20"
accessibility="off"
text="这是一个不需要被朗读的 textView"
 />
```

```
<ImageView
width="100"
height="100"
marginLeft="20"
marginTop="12"
borderWidth="3ap"
borderColor="#FF0000"
accessibility="on"
accessibilityText="这是一个 ImageView 点击"
onTap="@rTap{'测试'}"
imageUrl="https://img.alicdn.com/tfs/TB1FuMQQFXXXXXLXXXXXXXXXXXX-420-420.jpg"
/>
</LinearLayout>
</LinearLayout>
```

模板示例手机演示

看如下视频，由于 Layout accessibility 设置了 auto 属性，因此该 Layout 会被选中，并朗读内部含有无障碍信息的 Text，但是第二个和第四个 TextView 是不会朗读的，第二个配置了 onTap&accessibility="on"，因此此时它属于一个可交互的控件，是需要单独被选中的，第四个 accessibility="off"，因此此时它是关闭无障碍这个功能的，因此也没法选中朗读，且不会被 Layout 选中朗读。

点击可跳转原文查看视频详情

无障碍校验卡口

支持是一方面，引导开发同学去写是另一方面。

事实上现在好多动态化的方案，包括 native 本身都会支持无障碍功能，但是这种支持是单向的，如果你只是支持，但是开发者不去支持，那最终这个产品无障碍功能依旧是缺失。

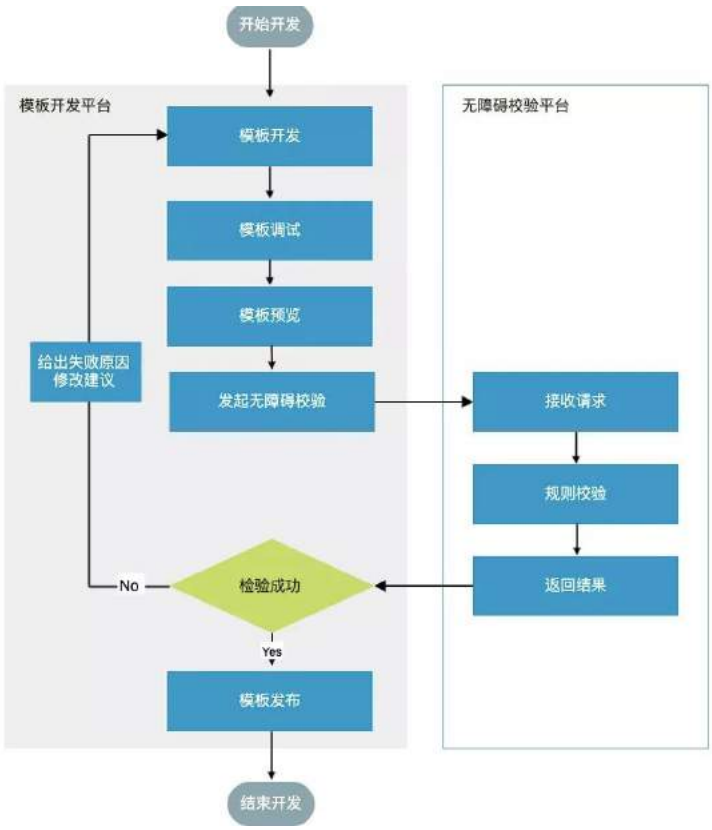
开发者为什么不去支持呢?

- 第一 无障碍的公益宣导不够，优先级不高，开发本身没有这个意识，无障碍测试用例缺失。
- 第二 无障碍功能的支持有一定的成本，且没有一套标准和规范告知什么情况下需要无障碍，且如何支持。
- 第三 流程上没有监督和管控，开发有可能会忘记。

为了更好的支持帮助视障用户使用手机淘宝，同时帮助业务方定位发现无障碍的错误，减少无障碍的测试回归工作量，我们发起了无障碍校验卡口，智能检测无障碍问题，通过调用无障碍服务来判断模板是否合格，以此确保每一个模板的发布都是支持无障碍的。

添加无障碍校验卡口这才是无障碍工作最关键的一环，目前由于手淘的核心链路都使用的 DinamicX，且 DinamicX 模板都在组件平台开发，因此只要我们加上这卡口，你想不支持无障碍都不行，否则你的动态模板发布不了。

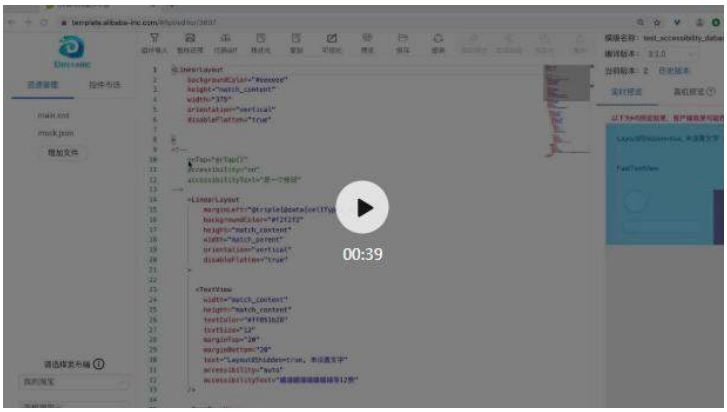
拥有无障碍校验卡口功能的 DinamicX 开发模式流程图:



目前无障碍卡口校验的相关规则（有些校验规则也是为了抹平两端差异而加的）：

- 非交互性控件，如 ImageView、FrameLayout、LinearLayout 等，若有设置 onTap 属性，则会检查是否含有无障碍属性，若没有则校验不通过，并给出建议：需要设置" accessibility= on"，开启无障碍焦点，同时设置 "accessibilityText=xx" 属性。
- 非交互性控件，如 ImageView、FrameLayout、LinearLayout 等，若有设置 accessibility=on 的时候，必须同时设置 accessibilityText=xx。
- 子 View 设置 onTap 属性的时候，必须保证它的父 Layout 没有设置 accessibility=on，否则该子 View 是不能获取焦点的。
- 如果 Layout 设置了 auto 属性，TextView 不能只设置 onTap，还要设置 accessibility=on，否则获取不到焦点。
- accessibility 属性不能设置动态表达式。

如下视频，假设 Layout 节点上面设置 onTap 点击事件，那么校验卡口会提醒你该节点需要设置无障碍信息：



点击可跳转原文查看视频详情

现阶段整个手淘首页、详情、购物车、我的淘宝、订单详情、订单列表等核心页面所开发的模板都会经过该卡口的校验。

愿景

也许我们做的不一定是最好的，但是我们会一直努力去做，不为别的，只是为了让手淘在大众心中特别是盲人的心中除了是一个购物 app 之外，更是一个温暖的产品，一个让盲人感动的产品。

希望有一天我们的开发同学开发模板的时候，再也不需要弹起那个卡口的校验，而是写模板的那一刻，已经想起了那些拿着手机耳边听的人群。

希望有一天手淘是盲人心目中最喜欢的购物产品。

希望有一天看到这篇文章的同学们能够在心里有那么一丝触动，不是道德绑架，而是在未来某一天盲人谈起手淘那种由衷的感谢可以带给自己心灵的那种慰藉。

希望有一天看到这篇文章的同学能够感受到：无障碍是一件公益，做完心里暖暖的，技术除了有价值以外，还可以有温度的。

重视无障碍 重视公益 从我们做起！

借助手机旁白功能，视障者用耳朵购物

淘宝无障碍实验室工程师闭着眼睛做无障碍测试

2020 技术资讯&观点解读

2020 双 11 技术总指挥汤兴：今年双 11 是技术和商业的完美共振

作者 | 淘系技术

出品 | 淘系技术

“ 今年的双 11 是全球极大内容电商场的超级爆发，消费者、技术、内容与商业生态之间每一秒都在产生激烈共振，实时性、复杂性和持续峰值的叠加令其成为全球技术顶峰。我们的使命就是让技术成为双 11 坚如磐石的稳态，让消费者感受到更顺滑的体验。”

——汤兴

阿里巴巴副总裁 双 11 技术总指挥 汤兴（平畴）

今年是第 12 个双 11，作为双 11 的新一轮的元年，不仅仅是每年最大的购物节狂欢节，更是 2020 年最大的一次内需爆发。4982 亿，双 11 再创记录。这背后是阿里巴巴经济体的数字化建设支撑了全社会的双 11，支撑了如此大体量的双节棍。快跟着小编来了解下今年双 11 最新最快的核心技术解读。

淘宝直播，缔造全球内容电商场的技术顶峰

“今年天猫双 11 是最为特殊的一年。”淘宝天猫总裁蒋凡说，作为疫情之后全球最大的消费季，很多商家对“双 11”有更强的期待。从 10 月 21 日，淘宝直播强势开启今年双 11 的预售环节，淘宝直播俨然成为商家在内容营销媒体上的第一选择。那从技术上，淘系直播技术系统如何做能够支持在线高并发，同时还能保障主播和用户之间的互动实时性、交易的准确性？

淘系自研全球实时传输网络——GRTN

GRTN 是新一代多媒体传输网络，实现了流量、算力的智能调度，通过大数据计算，汇聚 CDN 及客户端数据，采用自动化控制理论，自动控制 CDN 带宽。同时淘宝直播的内容生产实现了端+云智能调度能力，依据端的性能、云的容量，动态调整生产位置，降低资源成本，保障了系统稳定性。

相较于其他直播平台的 3~5 秒延迟，GRTN 让淘宝直播的延时控制在 1 秒以内。今年双 11 是 GRTN 首次大规模应用于淘宝直播电商，极低的互动延时带来互动体验的质的提升。从而实现在千万级在线并发情况下，仍能够让主播自信的喊出“123，开抢！”

同时，基于 GRTN 动态容量伸缩的特性，保障了我们在直播带宽消耗快速增长的同时，成本同步下降 30%。

淘系内容中台服务阿里巴巴集团 10+业务

端侧智能助力直播创新玩法

随着手机端侧算力迅速发展，极大的提升了直播内容的交互可玩性和内容理解准确性。依托于淘系自研并开源的 MNN 端侧推理引擎，淘宝直播间推出“语音猜价格”挑战，观众在直播间也能实现语音交互，动动嘴就可以响应主播发出的猜产品价格任务。

淘宝直播的观众遍布全国，口音差异较大，对识别算法提出了很高的要求；同时，如果使用 Siri 等基于云端的语音识别技术，那本就无法满足直播场景强互动的识别时延，在一场直播动辄百万、千万的观众数量面前，将会雪上加霜，端侧识别引擎同样面临巨大挑战。为了实现实时语音交互，此次淘宝直播采用了全新的移动端语音识别方案。基于全新的网络结构，快语速、重口音等问题，高难度场景中的语音识别错误率降低近三成；同时，推理引擎 MNN 也针对语音识别模型做了重点优化，达成 50 毫秒内完成语音交互过程，速度高达传统语音识别的 20 倍。

平台与商家共同完成数智化运营

数字化新基建成就史上最大规模商家全链路压测

以往由于平台和商家 IT 基础设施存在"水位差"，平台很难帮助商家进行系统的改造升级。随着阿里面向商家与生态伙伴推出新一代数字化基建体系，借助云原生技术引擎、可持续全链路压测与云 IT 治理能力等平台成熟的技术，帮助商家与生态伙伴重构系统的高可用基线，使得越来越多的商家具备了和阿里一样的互联网级超大规模数据应用系统能力。

在双 11 备战期间，淘系技术团队通过帮助 100 多家 ERP 服务商与品牌商家实施订单洪峰全链路压测，使得商家系统每秒能处理的订单规模上万，这在传统模式下是完全无法想象的；通过向商家与生态伙伴提供新的 IT 治理手段与工具，帮助商家系统发现并修复上千个数据访问性能问题；在购后履约阶段，通过大数据技术应用，让今年双 11 每一笔订单都能更为有效地对接物流和仓储调度，提升消费者收件体验和时效的同时，帮助商家进一步降低了发货成本，发货效率提升 20%以上；以上一系列的技术升级，赋能百万商家，覆盖 90%以上的双 11 订单，为今年双 11 核心交易链路的平稳运行保驾护航。

在过去，我们将阿里的全链路压测称为大促备战的"核武器"，放眼未来，阿里巴巴通过电商与云原生技术的进一步融合，将更好形成与百万商家的数据化实时协同，为实现真正的线上线下零售一体化、智能化经营，奠定了良好的系统基础。

智能新品孵化，助力商家快速增长

新品作为每个品牌/商家的增长核心驱动力之一，在过去，传统新品的研发主要痛点有两个：新品的研发周期长，新品的研发成功率低。目前在天猫上，已经有超过 1600 多家品牌商，通过新品平台来尝试进行趋势、测款、试销、量产、诊断、引爆等全流程智能化。整体的新品研制流程能比过去缩短 60%。

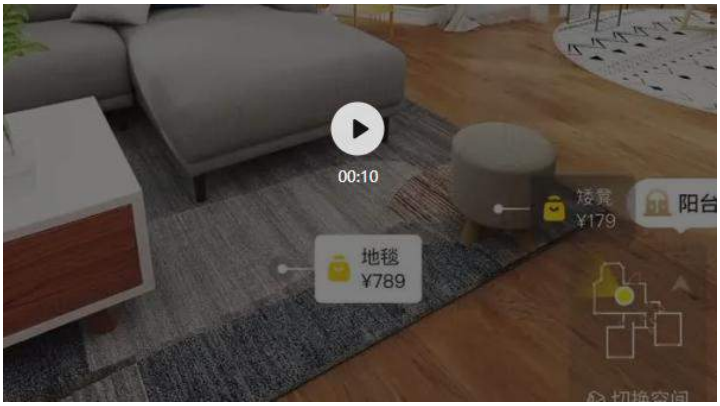
淘系新品平台通过整合业界趋势生态机构和时尚媒体，结合阿里巴巴平台海量数据，将抽象的趋势语言结构化和数字化。并采用 NLP 新词发现，图像识别等智能技术，为趋势构建了体系化的识别和预测模型，快准全的跟踪和预测未来趋势。在此基础上，构建完整的市场分析和评估体系，将趋势转化为生意机会。与品牌商共建品牌知识库，并通过线上的精准人群测款调研能力、新品试销诊断，帮助品牌商快速调整新品策略，赋能商家新品研发打造极致爆款。以产品化+ISV 的方式赋能行业和商家，提升其对新品开发的确定性，降低试错成本。

淘系技术投入建设新品平台已有两年多，致力于构建从研发新品到打爆新品的一体化新品技术体系，也已经促成犀牛智造、躺平智造等柔性制造模式更快落地。 3D 场景购“利器”助推

线上家装业爆发

我们“盖”了一栋 100 层的大楼

这个双 11，天猫盖了一座 100 层的 3D 虚拟家装城，包含几万套家装样板间，可以 360 度无死角呈现家居全景。消费者躺在家里，点击手机屏幕就能挑选自己喜爱的家装产品，体验实景逛街的感觉。与此同时，还可以自主搭配、一键购物。当然，还可同步享受品牌商家双 11 的优惠促销价格，以及线下门店的售后服务。



点击可跳转原文查看视频详情

在这座 100 层大楼应用了 5G、VR、AR 技术的应用，驱动 3D 样板间 2.0 的体验升级。

在双 11 前夕，淘系技术联合躺平设计家共同研发并推出了 AI 智能样板间。在 11 月 3 日的技术沟通会上，双 11 技术总指挥汤兴说："通过 AI 智能样板间，可以帮助商家可以在 3 秒钟之内，设计出一套 3D 样板间，仅仅在今年双 11 期间，就为客户节省了近 1 亿元设计费。"

AI 智能样板间通过不断学习前期积累的大量优秀设计案例，能将最新潮前沿的设计方案呈现给消费者。值得一提的是，智能样板间与淘宝主流量渠道进行了打通，能够为商家带来精准流量，通过大数据技术，实现对投放设计的实时追踪，及时感知消费偏好，实现千人千面投放。

3D 样板间虚拟直播，碰撞新火花

同时今年双 11，"3D 样板间+直播"再升级，首次采用绿幕技术，帮助商家和主播在虚拟的样板间中进行直播。这背后则是淘系技术与躺平设计家在 3D 购物技术领域带来的新突破，即一将 3D 样板间技术与绿幕直播技术进行融合，使消费者在观看主播讲解时，仿佛置身于实景样板间之中。

主播在前面讲解，可以利用绿幕“抠像”技术换成 3D 样板间的背景。如此，置身绿幕的主播，在直播镜头里，却仿佛置身在实景样板间中。而绿幕技术和 3D 样板间加持后，主播可以随时切换不同的场景，瞬间“平移”至其他样板间，为商家大大节约了成本。

左图直播背景是绿幕，右图观众视角的背景是 3D 样板间

仅 11 月 1 日，就有上万名消费者进入商家直播间，体验在线 3D 样板间虚拟直播的新形式。在此期间，用户平均逛样板间的时长达到 42 秒，单个 UV 的价值较日常提升 5 倍多。

技术君说：每一年的双 11，零点交易的系统数字都在不断提升，而淘系技术的突破和创新远不止与此，我们将持续打造全球领先的线上新零售技术平台。

WWDC 20 Core Image：iOS 图像处理技术追踪

作者 | 张若舟(越伯)
出品 | 淘系技术

Core Image 是苹果官方提供的图像处理框架，通过丰富的 built-in（内置）或自定义 Filter（过滤器）高效处理静态图片、动态图片或视频。开发者还可以通过构造 Filter 链或自定义 Core Image Kernel 来实现更丰富的效果。

在 WWDC20 中，苹果官方针对 Core Image 技术在以下三方面做了优化：Core Image 对视频/动图的支持、基于 Metal 构建 Core Image（CI）Kernel 以及 Core Image 的 Debug 支持。

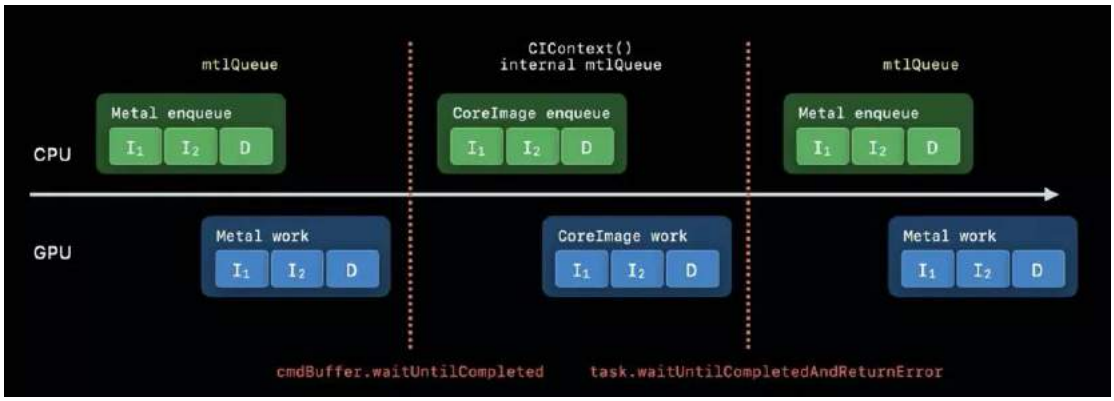
这三方面会在下文逐一提到，文末笔者也会浅谈 Core Image 在手淘图片库中的应用可能以及对 Core Image 技术的展望。

优化 Core Image 对视频/动图的支持

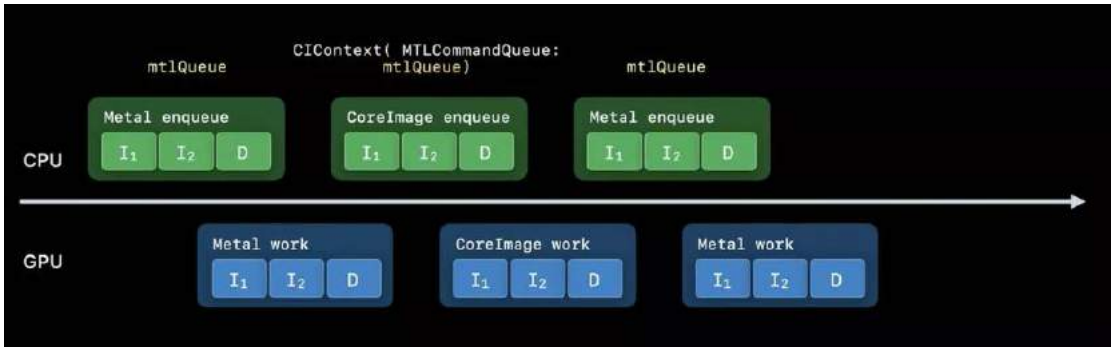
创建 CIContext

创建 CIContext 时，需要遵循一个 view 一个 context 的原则。由于视频的每一帧都会发生变化，将 CIContext 的 cacheIntermediates 属性设置为 false 可以大大减少内存消耗。

如果在使用 Core Image 时将同时运用 Metal（作为输入或输出），通过设置 MTLCommandQueue 属性创建 CIContext 将会是较好选择。在不使用 MTLCommandQueue 的情况下，每一个 Metal 或 CoreImage 执行的任务都在不同队列中并以 wait 命令分隔开，导致任务执行效率低。通过设置 MTLCommandQueue 创建的 CIContext 和相应的 Metal 任务在同一队列中，能提高 app 的运行效率。



图一：不使用 MTLCommandQueue 的工作流程



图二：使用 MTLCommandQueue 的工作流程

编写 Core Image Kernel（在 Metal 中实现）

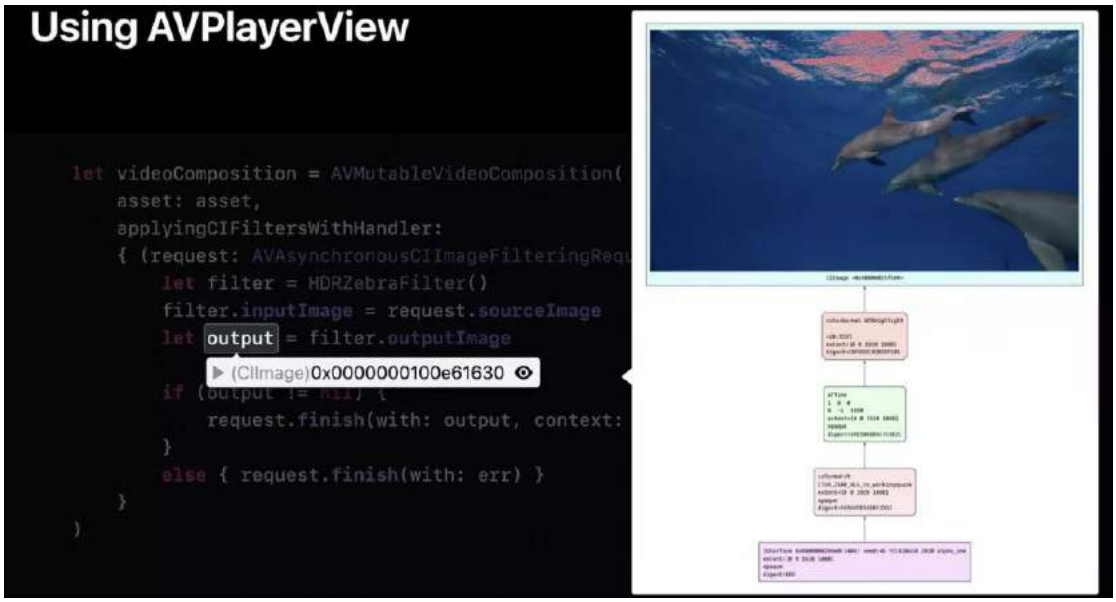
为了将效果处理得更丰富，通过 Metal 来实现自定义 CI Kernel 是个高效的选择。苹果官方提供了的许多方便部署的内置工具（都通过 Metal 实现），如内置 CI 滤镜。通过 Metal 实现自定义 CI Kernel，不仅 app 的 runtime 编译时间将会大大减少（这段工作会移至 app 构建完成后进行），开发者还能获得高性能语言特性（如 gather-reads、group-writes、半精度浮点数）、高效开发体验（如缩进检查、缩进高光）等功能。

选择合适的 View 类

如果要对视频/动图应用特效，静态内容 View 如 UIImageView 或 NSImageView 应当被避免。AVPlayerView 和 MetalKit View（MTKView）是个两个不错的选择。前者为简单选择，后者为进阶选择。

使用 AVPlayerView 时，需要创建 AVMutableVideoComposition 对象，CI 滤镜在 block 中执行图像处理任务。在进行断点 debug 时，通过点击 CIImage 对象地址右侧的眼睛图示可以浏览 CI 滤镜处理流程的详细信息。

官方提供的案例中，Core Image 还将 10 位的 HDR 视频帧数据自动从 HLG 转化成了 Core Image working space。



图三：CI Image 断点测试中展现的处理流程

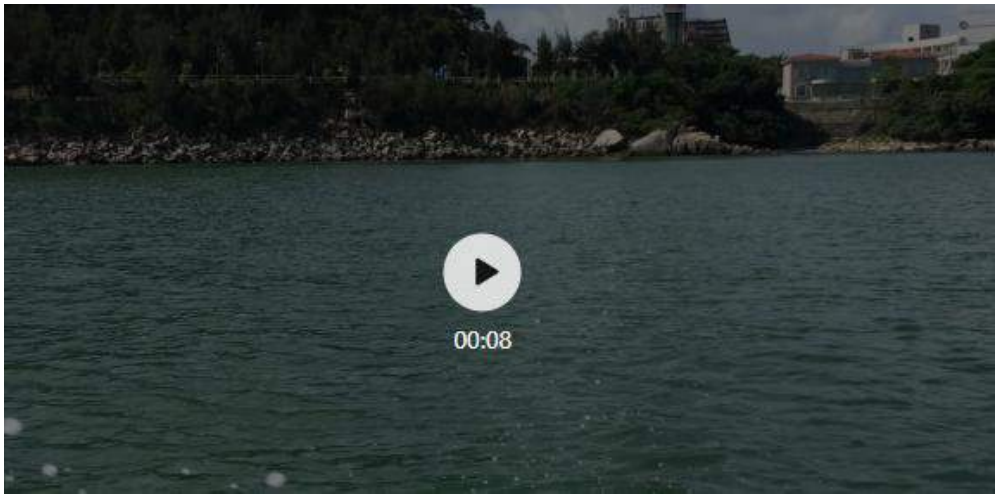
使用 MTKView 时，开发者需要以 frame 和 device 作为参数重载 init 方法。VIew 对应的 CIContext 也将在 init 函数中被创建。如果我们在 macOS 中开发支持 HDR 的 view，color-Pixel-Format 属性需要被设定为 rgba16Float，wants-Extended-Dynamic-Range-Content 属性需要被设定为 true。设定完 init 方法后，开发者需要实现 draw-in view 方法。需要注意的是，此处并未直接将 Metal 材质传入 CIRenderDestination 函数，而是创建了一个会返回 texture 的 block。这使得 CIContext 能在前面的帧尚未完成时将 Metal 工作入队。之后该方法会执行渲染任务（至指定目的地）并创建 command buffer 将当前绘制结果渲染至 view。

本人也亲自尝试了通过 Core Image 处理视频的整个流程。

以下案例使用 CIVortexDistortion 滤镜对视频进行逐帧处理并渲染，展示内容包含核心代码、原视频、CI 滤镜处理后视频以及断点测试的滤镜逐帧处理图示。

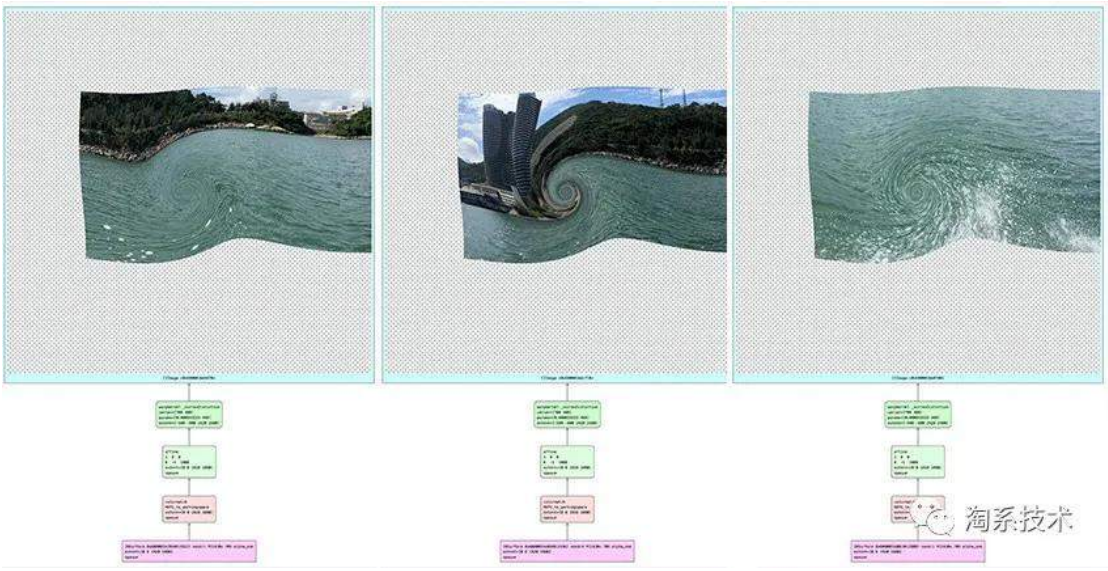
```
let filepath: String? = Bundle.main.path(forResource: "test_video", ofType: "MOV")
let fileURL = URL.init(fileURLWithPath: filepath!)
let asset = AVAsset(url: fileURL)
let item = AVPlayerItem(asset: asset)
item.videoComposition = AVMutableVideoComposition(asset: asset) { request in
  let filter = CIFilter(name: "CIVortexDistortion")
  filter?.setValue(request.sourceImage, forKey: kCIInputImageKey)
  filter?.setValue(NSNumber(400), forKey: "inputAngle")
  filter?.setValue(NSNumber(1200), forKey: "inputRadius")
  filter?.setValue(CIVector(x: 700, y: 400), forKey: "inputCenter")
  let out = filter?.outputImage
  request.finish(with: out ?? request.sourceImage, context: nil)
  }
  avPlayer = AVPlayer(playerItem: item)
```

测试核心代码



原视频（笔者本人取材）点击可跳转原文查看视频详情

添加 CIVortexDistortion 滤镜后的视频 点击可跳转原文查看视频详情



图四：断点调试时 Core Image 对每帧的处理流程

基于 Metal 构建 Core Image Kernel

使用 CI Kernel 有诸多优势，包括上文提及的缩短 runtime 编译时间、高性能语言特性（如 gather-reads、group-writes、半精度浮点数）、高效开发体验（如缩进检查、缩进高光）。

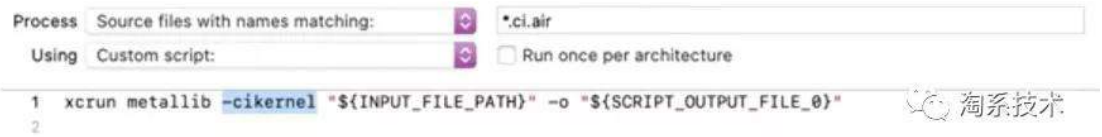
基于 Metal 构建 CI Kernel 有 5 步流程，会在下文进行逐一介绍。

在项目中增加自定义构建规则

苹果官方推荐在项目 target 中增加两项自定义构建规则。第一个构建规则针对以".ci.metal"为后缀名的文件。该构建规则会创建一个以".ci.air"为后缀名的二进制输出文件。

图五：针对"*.ci.metal"文件的构建规则

第二个构建规则针对以".ci.air"为后缀名的文件（上一个构建规则的输出结果）。该构建规则会在 app 的资源文件夹内创建以".ci.metallib"为后缀名的输出文件。



图六：针对"*.ci.air"文件的构建规则

在项目中增加.ci.metal 资源

在 Xcode 提供的创建面板中选择 Metal File 即可。开发者对 Metal File 进行命名时需要以".ci"作为后缀名，这样项目中新生成的文件会以".ci.metal"作为后缀名。

编写 Metal Kernel

便携 Metal Kernel 需要 include CoreImage.h 头文件，用来使用 Metal 和 Core Image 提供的各种类。官方提供的范例编写了一个 CIColorKernel，输入参数为 coreimage::samle_t 对象（表示输入图片的一个像素）、time 和 coreimage::destination 对象，返回 float4 像素。

```
// MyKernels.ci.metal
#include <CoreImage/CoreImage.h> // includes CIKernelMetalLib.h
using namespace metal;

extern "C" float4 HDRZebra (coreimage::sample_t s, float time, coreimage::destination dest)
{
    float diagLine = dest.coord().x + dest.coord().y;
    float zebra = fract(diagLine/20.0 + time*2.0);
    if ((zebra > 0.5) && (s.r > 1 || s.g > 1 || s.b > 1))
        return float4(2.0, 0.0, 0.0, 1.0);
    return s;
}
```

淘系技术

图七：苹果官方提供的代码范例：Metal Kernel 编写

苹果官方为开发者提供了描述 CI Kernel 中 Metal Shader 语言的文档，详情见「Metal Shading Language for Core Image Kernels」①。

加载 Kernel 并应用于新图像（基于 Swift）

Kernel 会被 CI 滤镜的子类使用。苹果官方推荐开发者在实例化滤镜的 CIKernel 对象时使用静态属性（static property），这种情况下加载 metallib 资源的工作仅会执行一次（在首次需要时）。CI 滤镜的子类也必须重载输出图片的属性，Kernel 将在 getter 中进行图像处理并创建新图像。

```
class HDRZebraFilter: CIFilter {
    var inputImage: CIImage?
    var inputTime: Float = 0.0

    static var kernel: CIColorKernel = { () -> CIColorKernel in
        let url = Bundle.main.url(forResource: "MyKernels",
                                  withExtension: "ci.metallib")!
        let data = try! Data(contentsOf: url)
        return try! CIColorKernel(functionName: "HDRzebra",
                                  fromMetalLibraryData: data)
    }()

    override var outputImage : CIImage? {
        get {
            guard let input = inputImage else {return nil}
            return HDRZebraFilter.kernel.apply(extent: input.extent,
                                               arguments: [input, inputTime])
        }
    }
}
```

淘系技术

图八：苹果官方提供的代码范例：Kernel 加载与使用

Core Image 的 Debug 支持

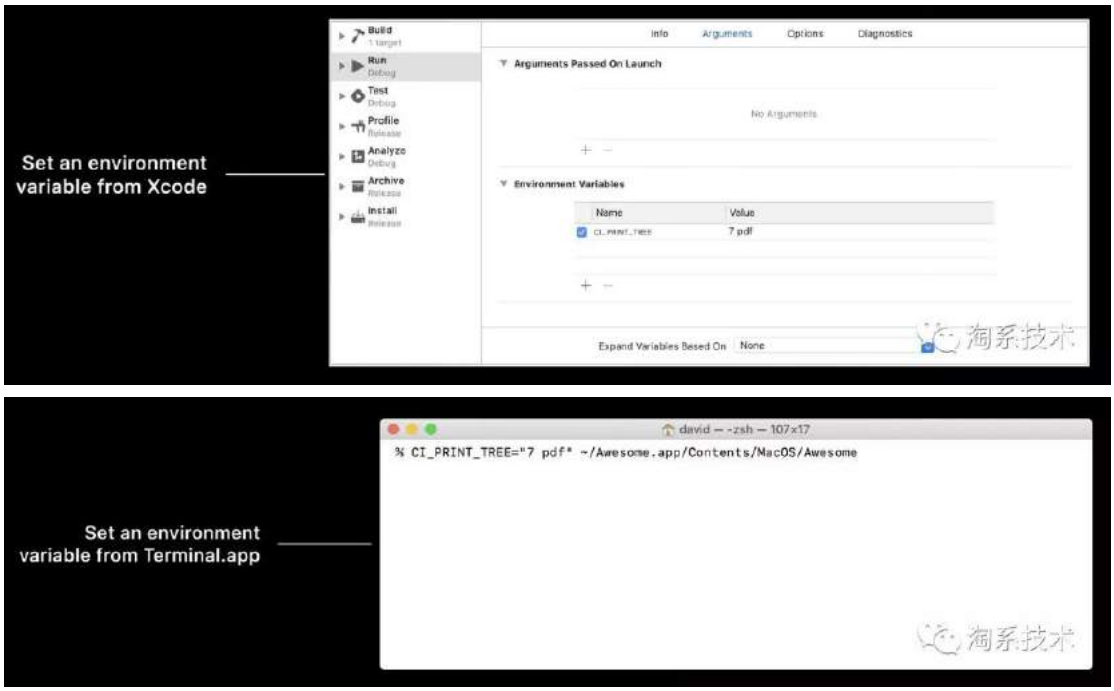
苹果官方在 WWDC20 详细介绍了 Debug 特性：CI_PRINT_TREE。

什么是 CI_PRINT_TREE

CI_PRINT_TREE 的基础框架与 Xcode 提供的 Core Image Quick Look 支持相同。Core Image Quick Look 为开发者提供了快捷可视化的 Core Image 图片（详见上文图三），而 CI_PRINT_TREE 支持几种不同的模式和选项用来查看 Core Image 如何优化和渲染图像。

如何启用 CI_PRINT_TREE

苹果官方提供了 CI_PRINT_TREE 的两种启动方式。最常用的方法是编辑 Xcode target scheme，在 Arugments 窗体下的环境变量列表中加入 CI_PRINT_TREE 并设置值。另一种方法是在 Terminal.app 中通过命令行启动 CI_PRINT_TREE（需要在执行应用程序前设定）。



图九：启用 CI_PRINT_TREE 的两种方式

如何控制 CI_PRINT_TREE

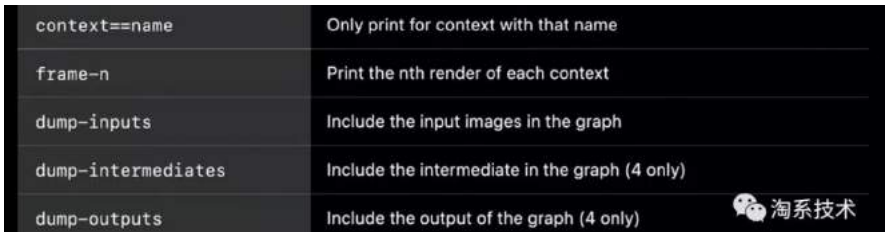
CI_PRINT_TREE 的字符串格式为"<graph type> <output type> <options>"

graph type：表示 Core Image render 的若干 stage，包括 type-1 初始图像（有助于查看被使用的色彩空间）、type-2 优化后的图像（有助于查看 core image 对 render 的优化效果）、type-4 级联图像（有助于查看各 stage 如何级联于 GPU 程序，以便了解 render 需要多少中间缓存）以及 type-7（输出图像 type1、2 和 4）。

图十：苹果官方对 graph type 四个 stage 的描述

output type：输出格式可以是 pdf 或 png。在 macOS 上 trees 会被存储在临时项目文件夹，在 iOS 上 trees 会被存储在文档（Documents）目录下。如果 output type 没有确定，core image 会把 tree 以紧凑文本格式输出在标准输出（stdout）。通过设置 CI_LOG_FILE ="oslog"，文本也可以前往 Console.app（在 iOS 开发中更为方便）。

options：对于 CI_PRINT_TREE，开发者可以设定额外的选项。如通过设定 context==name 来限制输出（仅输出名字相同的 context），或是通过设定 frame-n 来框定具体输出 context 的哪一帧。更多 option 及详情请见图十一。设定 option 对 debug 能提供很大帮助，但也需谨慎使用，因为生产这些文件需要额外的时间和内存。



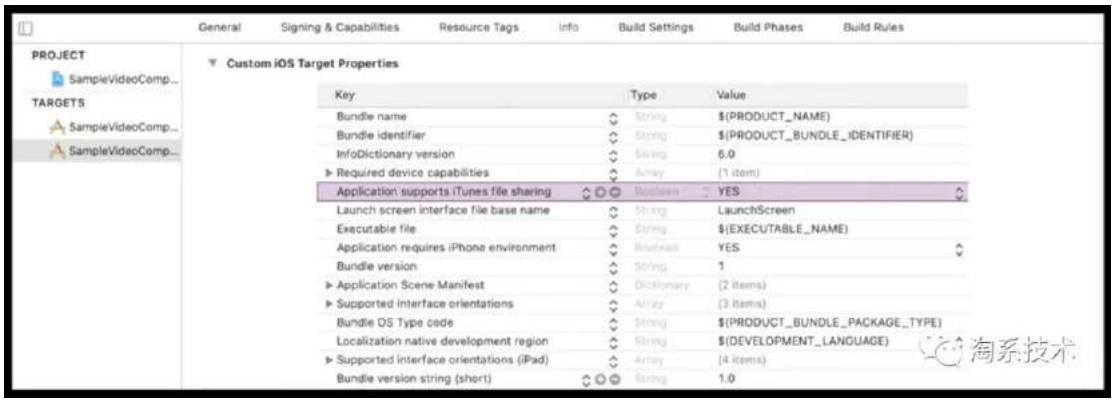
图十一：苹果官方提供的 option

图十二：type 设定为 7 时 tmp 文件夹下的文件

如何获得 CI_PRINT_TREE 文件

在 macOS 中，开发者只需要进入"/tmp"文件夹就能找到生成的 CI_PRINT_TREE 文件。需要注意的是沙盒应用会使用特有的临时存储文件夹。

在 iOS 中，开发者需要将 Custom iOS Target Properties 中的"Application supports iTunes file sharing"项设为 YES（图十三）。这样生成的 CI_PRINT_TREE 文件可以在连接中的 iOS 设备上被找到并拖拽至 macOS 存储中。

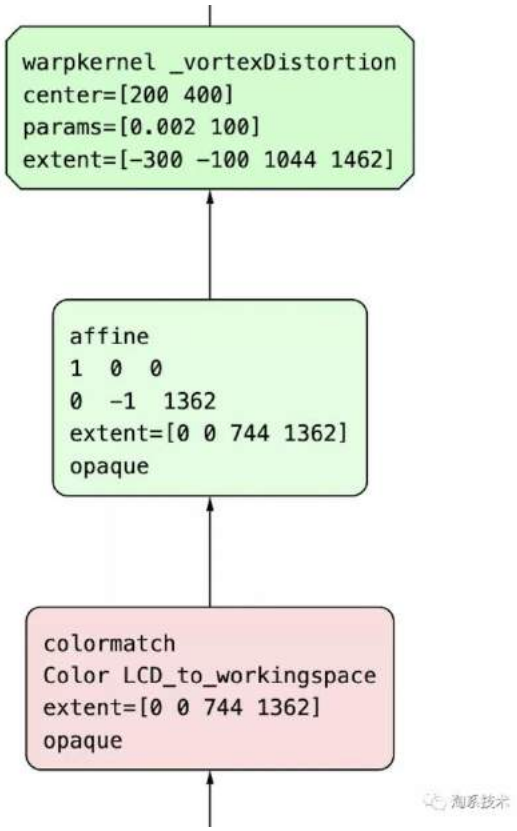


图十三：Custom iOS Target Properties 中进行设置

如何解释 CI_PRINT_TREE 文件

读 CI_PRINT_TREE 时，需要遵循以下规则：

- 输入在底层，输出在顶层。
- 绿色节点代表卷曲内核（warp kernel），红色节点代表颜色内核（color kernel）。



图十四：绿色节点与红色节点示例

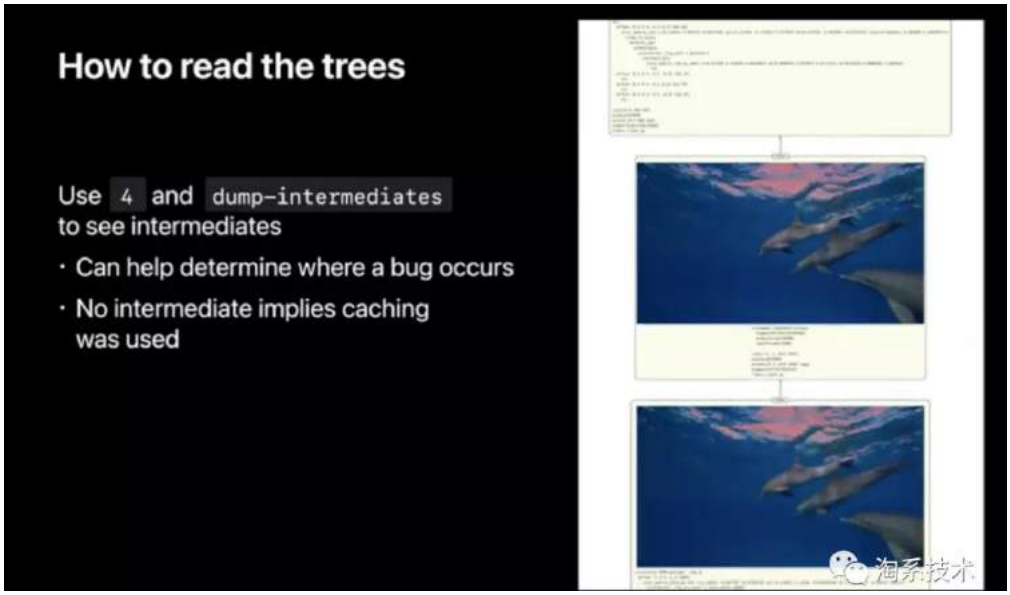
- 在树的初始位置（initial tree）很容易找到颜色搭配节点（colormatch nodes），里面记录了搭配前后的色彩空间名称。苹果官方提供的案例为 ITUR_2100_HLG_to_working space，即 HLG 色彩空间转化为 Core Image 线性色彩空间。

```
colormatch
ITUR_2100_HLG_to_workingspace
roi=[-1 -1 1922 1082]
extent=[0 0 1920 1080]
opaque
digest=812B7EBA01D82797
```

图十五：苹果官方案例中 initial tree 对色彩空间的描述

- 每个节点会显示 Region of Interest（ROI），表示该节点在 render 中被使用的范围。

如果开发者在 CI_PRINT_TREE 控制字符串中选择 type-4 并在 option 中设定 dump-intermediates，产生的级联图片会展示中间缓存的每一次 pass（除了 output pass）及其耗时、像素点数量和像素点格式（用来查找耗时大、占内存大的 pass）。这对 render 内追踪错误非常有帮助。如果树中没有展示中间图，那么说明这张图在先前渲染的时候已被缓存，因此 Core Image 没有渲染它的必要。



图十六：设定 dump-intermediates 的 debug 效果展示

Core Image 在手淘图片库中的应用可能

手淘图片库中的 CDN 图片适配处理库（TBCDNImage）的核心目的是为不同终端设备、网络环境下的图片展示提供最优解。目前考虑的维度主要是终端设备硬件和网络状态，考虑的参数则是图片尺寸、压缩比率、锐化等图片属性。随着苹果在 Core Image、端智能（CoreML）、硬件支持（自研芯片）等方面进行技术提升，手淘的 CDN 图片适配处理库可以考虑增加"图片内容"作为新的维度，增加亮度、对比度、滤镜、图片种类等新参数。以下为部分应用场景：

- 识别亮度较暗的图片，提升亮度做 CDN 图片适配处理。
- 判断图片内容种类（如美食），根据不同内容种类的图片增加适合的滤镜做 CDN 图片适配处理。

- 根据移动终端设备屏幕亮度（或深/浅色模式）修改图片色调做 CDN 图片适配处理，达到护眼效果。

对 Core Image 技术的展望

总结全文，WWDC20 对 Core Image 技术的提升主要在三方面：

- 优化 CI 对视频/动图的支持，包括开发流程简化、逐帧处理性能提升等。
- 允许开发者更自由的构建 Core Image Kernel，使 CI 的特效处理更加丰富。
- 针对 CI 开发流程提供更高效的 Debug 支持。

随着苹果未来自研芯片的底层硬件支持将提供视频流流畅的逐帧处理与渲染。笔者认为 Core Image 技术将会在以下场景有较大应用价值：

- 直播滤镜/特效功能原生化（摆脱自研或第三方 API），实现质量更高的实时滤镜渲染。
- 视频拍摄增加滤镜功能（如淘宝或咸鱼的商品视频录制）。

参考：

https://developer.apple.com/metal/MetalCIKLReference6.pdf
https://github.com/duzhaoquan/ImagesVideoFilters

WWDC 20 无线网络优化实践，带来哪些启发?

作者 | 徐杰(无宸)
出品 | 淘系技术

网络技术作为互联网应用赖以存在的技术基础，速度与安全永远是其核心使命，本次 WWDC 的网络类 topic 涵盖内容基本还是围绕这两个点来展开。本次 WWDC 网络类 session 在基础网络技术上譬如新协议、新算法方面着墨并不多；也未提出新的类似 NS URLSession / Network.framework 之类的新网络组件。站在应用视角，本次 WWDC 网络类 session 可分为两大类：

- 无线网络体验优化实践在系统层面的标准化；
- 本地网络应用的权限管控增强。

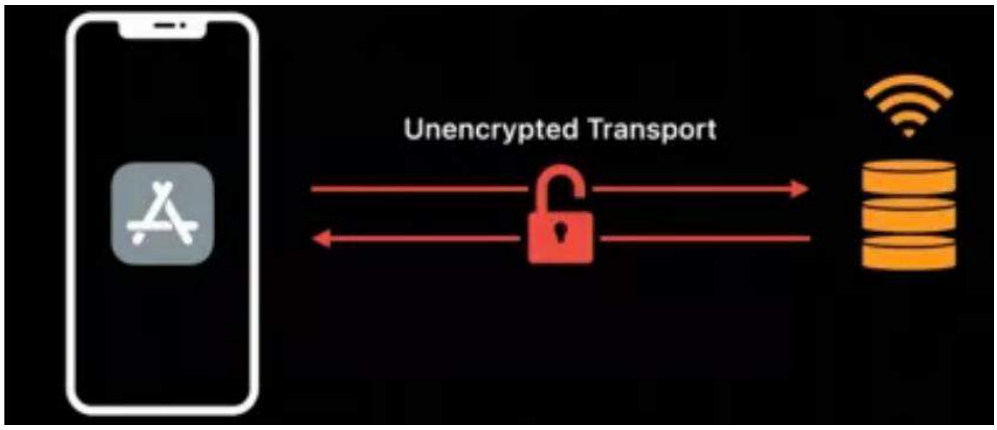
在第一类议题中，我们看到很多已经在手淘中的类似实践，或标准或自研，说明手淘在网络技术的开发与应用上还是较为深入和前沿的，基本走在全球业界前列。根据我们手淘的业务特点，笔者重点关注第一类 session，并简单探讨该新技术可以我们带来什么样启发和变化。

使用加密 DNS

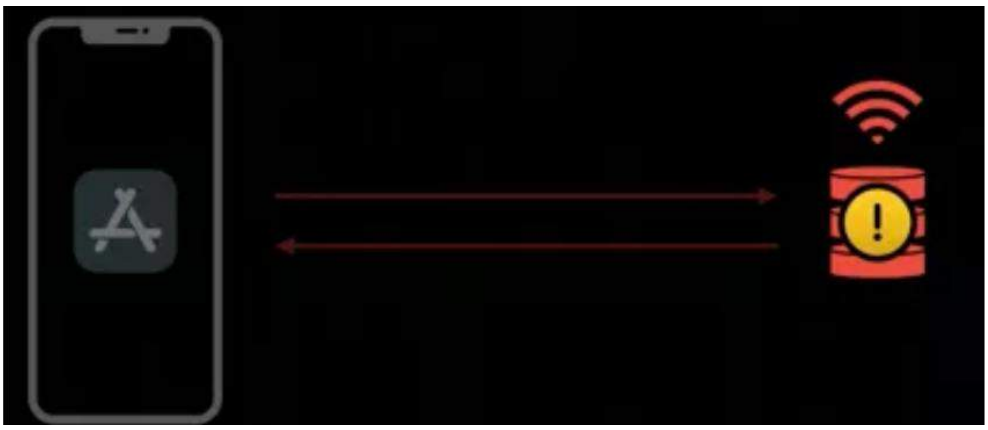
DNS 解析是网络的连接的第一步，这里提到的"加密 DNS"是什么、它解决什么问题?

解决什么问题

一是传统 Local DNS 的查询与回复均基于非加密 UDP，我们常见的 DNS 劫持问题



二是 Local DNS Server 本身不可信，或者本地 Local DNS 服务不可用。



其实针对 DNS 解析过程中以上两个问题，在实践中早就有了解决方案，就是 HTTPDNS，各大云厂商也都有成熟产品售卖，那苹果这里的加密 DNS 与我们的现有 HTTPDNS 有何不同呢？

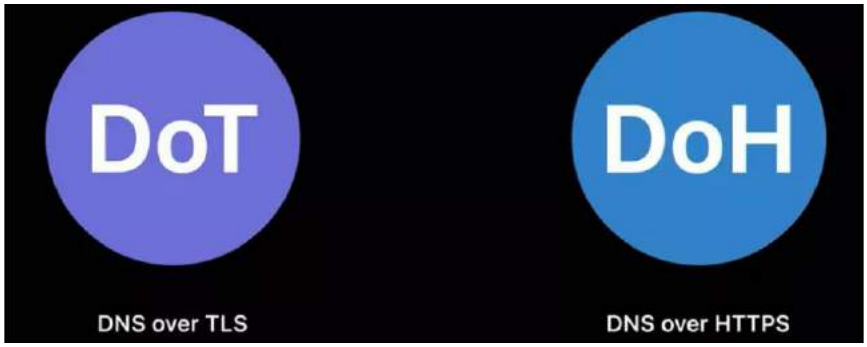
现有 HTTPDNS 有两个很大的问题：

- 一是对业务的侵入性，即如果某个网络连接需要使用 HTTPDNS 的能力，首先他需要集成服务商提供的 SDK，引入相应的 Class，然后修改网络连接的阶段的代码；
- 二是面临各种技术坑，比如 302 场景的 IP 直连处理、WebView 下 IP 直连如何处理、Cookie、以及 iOS 上的老大难的 SNI 问题等，这些都需要业务开发者付出极大的努力和尝试。

iOS 14 上的 Encrypted DNS 功能很好的解决了现有 HTTPDNS 的存在的问题。

规范与标准

iOS 14 开始系统原生支持两种标准规范的 Encrypted DNS，分别是 DNS over TLS 与 DNS over HTTPS。



具体协议标准可以参见：rfc7858（DoT）、rfc8484（DoH）

如何实现

iOS 14 提供了两种设置加密 DNS 的方法。第一种方式是选择一个 DNS 服务器作为系统全局所有 App 默认的 DNS 解析器，如果你提供的是一个公共 DNS 服务器，你可以使用 NED NSSettingsManager API 编写一个 NetworkExtension App 完成系统全局加密 DNS 设置。或者如果你使用 MDM(Mobile Device Management)管理设备的企业设置；你可以推送一个包含 DNSSettings paload 的 profile 文件完成加密 DNS 设置。

使用 NetworkExtension 设置系统域全局 DNS 服务器示例代码：

```
// Create a DNS configuration

import NetworkExtension

NEDNSSettingsManager.shared().loadFromPreferences { loadError in
    if let loadError = loadError {
        // ...handle error...
        return
    }
    let dohSettings = NEDNSOverHTTPSSettings(servers: [ "2001:db8::2" ])
    dohSettings.serverURL = URL(string: "https://dnsserver.example.net/dns-query")
    NEDNSSettingsManager.shared().dnsSettings = dohSettings
    NEDNSSettingsManager.shared().saveToPreferences { saveError in
        if let saveError = saveError {
            // ...handle error...
            return
        }
    }
```

上述代码首先通过 NEDNSSettingsManager 加载配置，加载成功后，创建一个基于 DoH 协议的 NEDNSOverHTTPSSettings 实例，对其配置 DNS IP 地址和域名，DNS IP 地址是可选配置的。然后将 NEDNSOverHTTPSSettings 实例配置到 NEDNSSettingsManager 共享实例的 dnsSettings 属性，最后保存配置。

一条 DNS 配置包括 DNS 服务器地址、DoT/DoH 协议、一组网络规则。网络规则确保 DNS 设置兼容不同的网络。因为公共 DNS 服务器是无法解析本地网络的私有域名，比如只有企业 WiFi 网络内的 DNS 服务器可以解析员工访问的私有域名，这类情况就需要手动指定网络规则兼容企业 WiFi 网。

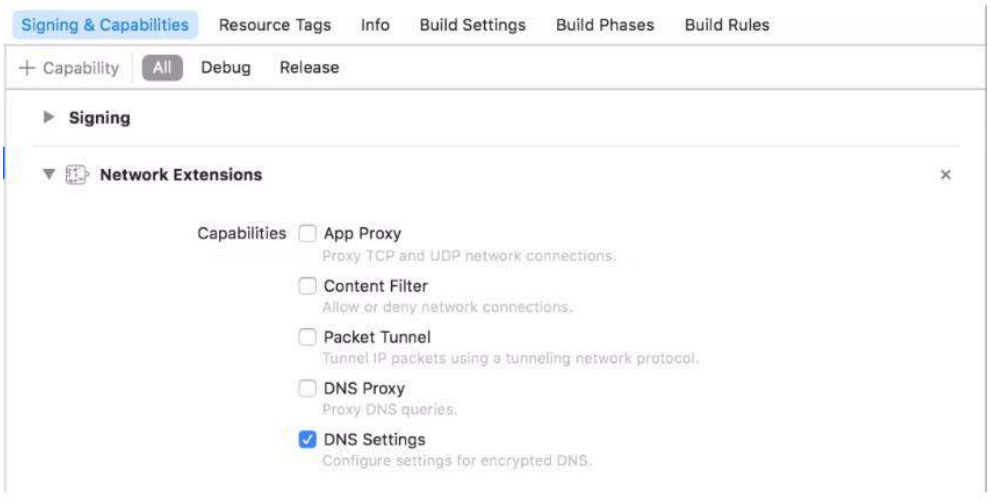
网络规则可以针对不同网络类型定义行为，比如蜂窝网、WiFi、或者具体的 WiFi SSID。在匹配的网络下，你可以禁用配置的全局 DNS 设置，或者对私有域名不使用 DNS 设置。

而在一些情况下，兼容性会自动处理。比如强制门户网络(captive portal)，手机在连接上某个 WiFi 的时候，自动弹出一个页面输入账号密码才能连接网络。这种情况下系统域全局 DNS 配置会做例外处理。相类似的，对于 VPN 网络，在 VPN 隧道内的解析将使用 VPN 的 DNS 设置，而不使用系统域 DNS 配置。

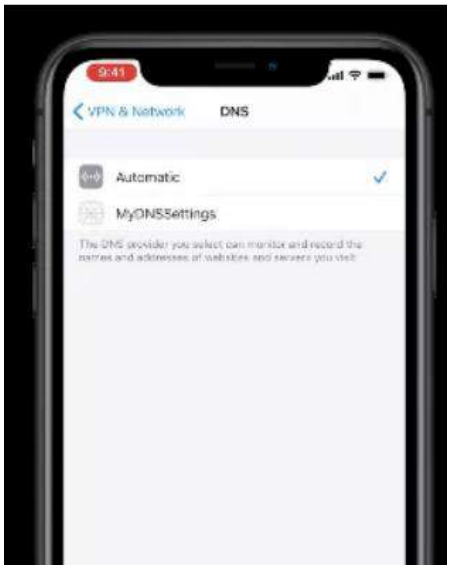
网络规则设置示例代码：

```
// Apply network rules

let workWiFi = NEOnDemandRuleEvaluateConnection()
workWiFi.interfaceTypeMatch = .wiFi
workWiFi.ssidMatch = ["MyWorkWiFi"]
workWiFi.connectionRules =
    [ NEEvaluateConnectionRule(matchDomains: ["enterprise.example.net"],
                               andAction: .neverConnect) ]

let disableOnCell = NEOnDemandRuleDisconnect()
disableOnCell.interfaceTypeMatch = .cellular

let enableByDefault = NEOnDemandRuleConnect()

NEDNSSettingsManager.shared().onDemandRules = [
    workWiFi,
    disableOnCell,
    enableByDefault
]
```

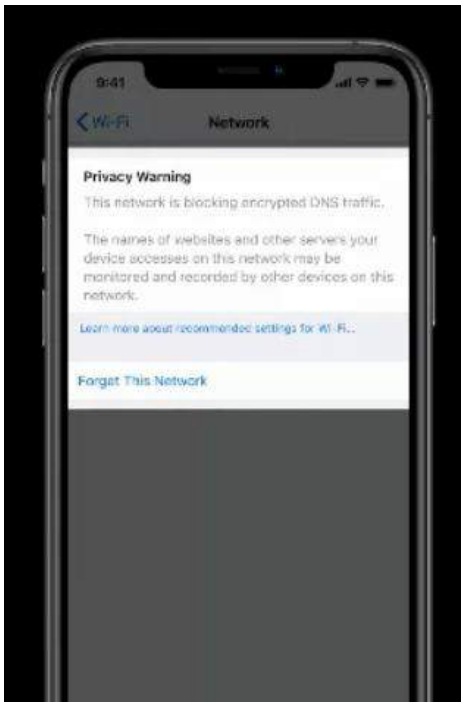
上述代码设置了三个网络规则，第一个规则表示 DNS 配置应该在 SSID="MyWorkWiFi"的 WiFi 网络生效，但对私有企业域名 enterprise.example.net 不开启。第二个规则表示规则在蜂窝网下应该被禁止使用;第三个 NEOnDemandRuleConnect 表示 DNS 配置应该默认开启;因为配置 DNS 是系统支持的，所以在编写 NetworkExtension App 时不需要实现 Extension 程序，只需要在 Network Extensions 中勾选 DNS Settings 选项。



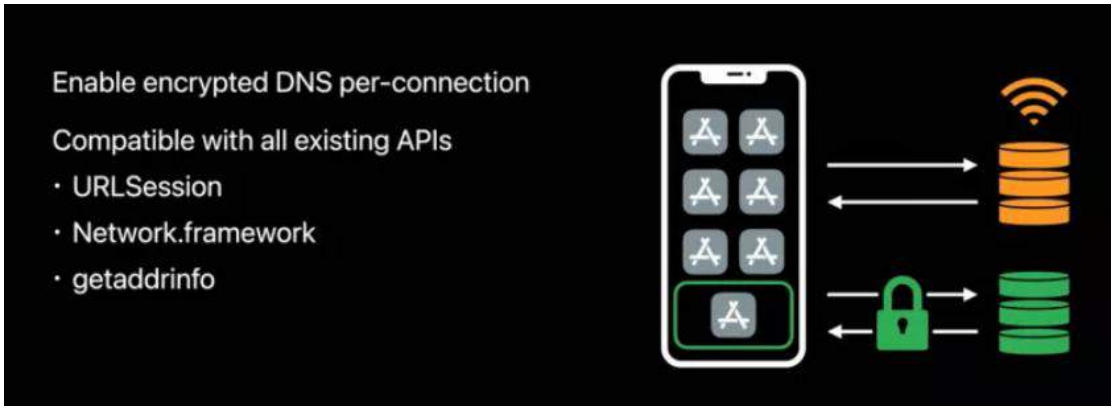
运行 NetworkExtension App，DNS 配置将会被安装到系统，为了让 DNS 配置生效，需要前往设置->通用->VPN & Network->DNS 手动启用。



一些网络可能会通过策略阻止你使用加密的 DNS 服务器。这些网络尝试分析 DNS 查询请求来过滤流量。对于此类网络，系统会被标记隐私警告提示，在该网络下的网络连接将会失败。



第二种方式是针对单个 App 的所有连接或部分连接启用加密 DNS。



如果你只想为你的 App 使用加密 DNS，而非涉及整个系统域。你可以适配 Network framework 的 PrivacyContext，对你的整个 App 开启加密 DNS，或者仅对某一连接开启。不管使用的是 URLSessionTask，或 Network framework 连接或 getaddrinfo 的 POSIX API，这种方式都有效。

对单个连接使用加密 DNS 示例代码：

```
// Use encrypted DNS with NWConnection

import Network

let privacyContext = NWParameters.PrivacyContext(description: "EncryptedDNS")
if let url = URL(string: "https://dnsserver.example.net/dns-query") {
    let address = NWEndpoint.hostPort(host: "2001:db8::2", port: 443)
    privacyContext.requireEncryptedNameResolution(true,
        fallbackResolver: .https(url, serverAddresses: [ address ]))
}

let tlsParams = NWParameters.tls
tlsParams.setPrivacyContext(privacyContext)

let conn = NWConnection(host: "www.apple.com", port: 443, using: tlsParams)
conn.start(queue: .main)
```

验证请求是否使用加密 DNS：

```
import Network

conn.requestEstablishmentReport(queue: .main) { report in
    if let report = report {
        for resolution in report.resolutions {
            switch resolution.dnsProtocol {
            case .https, .tls:
                print("Used encrypted DNS!")
            case .udp, .tcp:
                print("Used unencrypted DNS")
            default:
                // Ignore unknown protocols
        }
    }
}
```

如果你想在 App 范围内使用加密 DNS，你可以配置默认的 PrivacyContext；App 内发起的每个 DNS 解析都会使用这个配置。不管是 URLSessionTask，还是类似 getaddrinfo 的底层 API。

```
import Network

if let url = URL(string: "https://dnsserver.example.net/dns-query") {
    let address = NWEndpoint.hostPort(host: "2001:db8::2", port: 443)
    NWParameters.PrivacyContext.default.requireEncryptedNameResolution(true,
        fallbackResolver: .https(url, serverAddresses: [ address ]))
}

let task = URLSession.shared.dataTask(with: ...)
task.resume()

getaddrinfo(...)
```

现有实践对比及启发

与现有实践对比

上文已经提到过 HTTPDNS 产品，手淘的域名解析，采用的是类似于 HTTPDNS 原理，但更加复杂的调度方案。但是相比于这种系统层面的标准化方案，解决了现有实践中的两大难题：

- 对业务的侵入性：业务必须修改现有网络连接的阶段的代码；
- 交互的标准兼容性不足：IP 直连下的 302 问题、Cookie 问题、SNI 问题等 。

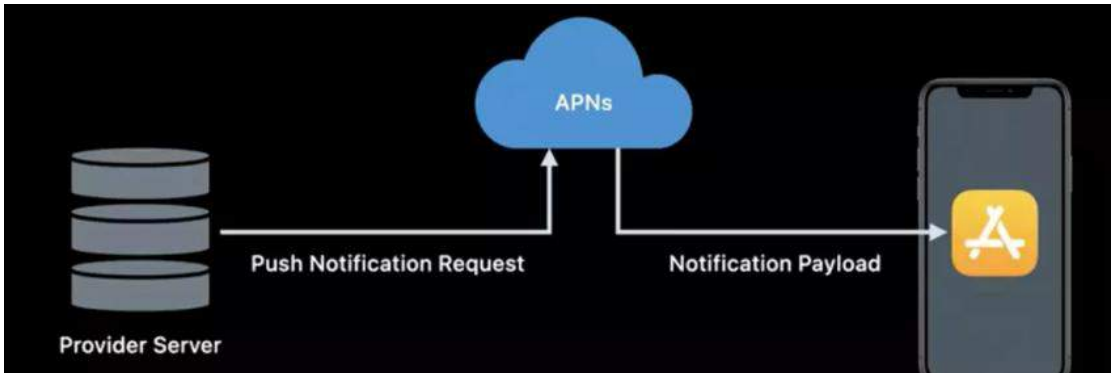
带来的变化

一是在应用内部，我们可以对业务透明提供提供全局的域名调度或者域名兜底解析能力,不再是只有用到特定组件或 SDK 才可以。二是苹果放开系统级别 DNS 接管后，用户设备上的服务相比原 Local DNS 的"中立性"如何管控？对特定业务是否甚至会造成恶化？如果对外部应用的这种"中立性"疑问或担心确实存在，则这种系统标准化的加密 DNS 对手淘这种大型应用则是必选项。

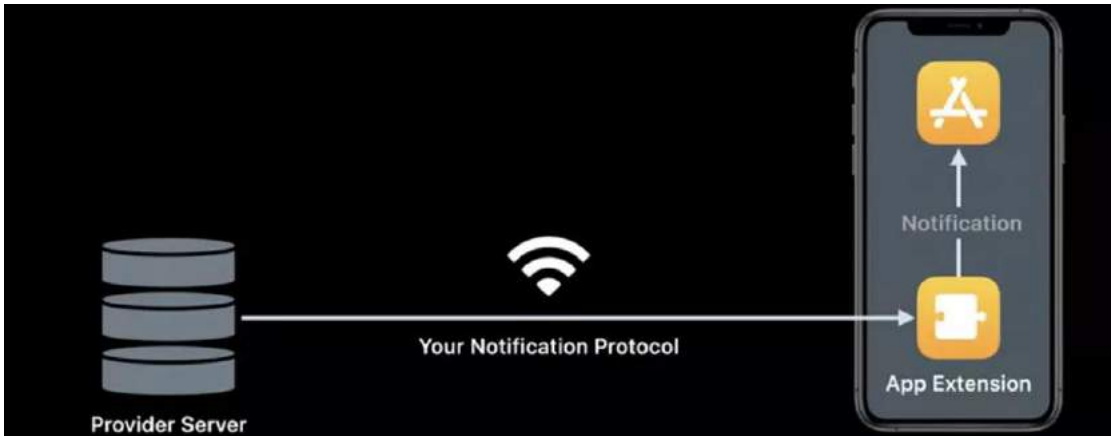
受限网络中推送

解决什么问题

当向 iOS 设备推送消息时，业务服务器需要将消息先发送给 APNS 服务器，APNS 服务器再将消息转换为通知 payload 推送给目标设备。如果设备所在的 WiFi 网络没有连接互联网或者当前网络受限，比如在游艇、医院、野营地这些地方，设备没有与 APNS 服务器建立有效连接，APNS 消息投递将会失败。



对于一些 App 来说，接收推送消息是 App 的一项非常重要的功能，即使在没有互联网连接的情况下也需要持续稳定工作。为了满足这一需求，iOS 14 中增加了本地推送连接(Local Push Connectivity)API，这是 NetworkExtension 家族中新的 API。本地推送连接 API 允许开发者创建自己的推送连接服务，通过开发一个 App Extension，在指定的 WiFi 网络下可以直接与本地业务服务器通信。



上图中，App Extension 主要负责保持与本地业务服务器之间的连接，以及接收业务服务器发来的通知。因为没有了 APNS，开发者需要为业务服务器与 App Extension 定义自己的通信协议。主 App 需要配置具体在哪个 WiFi 网络下使用本地推送连接。当 App 加入到指定的 WiFi 网络，系统会拉起 App Extension，并在后台持续运行。当 App 断开与指定 WiFi 网络的连接，系统将停止 App Extension 运行。

如何实现

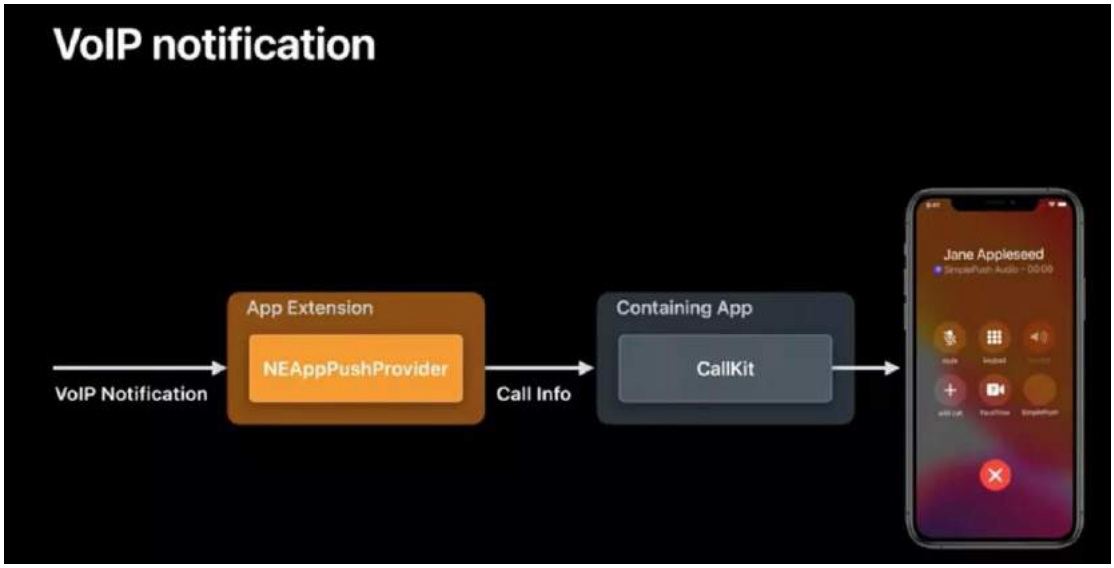
本地推送连接对那些推送功能非常重要，而设备所在网络受限的场景非常适合。而对于常规的推送需求，依然推荐使用 PushKit 或 UserNotification API 处理 APNS 推送消息。每台设备和 APNS 服务器之间只建立一条 APNS 连接，设备上所有 App 都公用这一条连接，所以 APNS 非常省电。

APNS 与本地推送连接对比：

Push Notifications	Local Push Connectivity
Requires APNs connectivity	Requires local server connectivity
Any network	Specific Wi-Fi networks
Apple APNs protocol	Custom protocol

新的本地推送连接 API 由两个类组成：NEAppPushManager 和 NEAppPushProvider。NEAppPushManager 在主 App 中使用，主 App 使用 NEAppPushManager 创建一个配置，配置中指定具体哪个 WiFi 网络下运行本地推送连接。你可以使用 NEAppPushManager 加载/移除/保存配置。NEAppPushProvider 则在 App Extension 使用。在 App Extension 中实现一个 NEAppPushProvider 的子类，子类中需要覆盖生命周期管理方法，这些方法在 App Extension 运行和停止时被调用。

App Extension 主要处理两类推送，一类是常规的推送通知，一类是 VoIP 呼叫通知。如果是常规的推送通知，App Extension 收到消息后，可以使用 UserNotification API 构造一个本地推送显示推送信息。如果是 VoIP 呼叫通知，App Extension 使用 NEAppPushProvider 类将呼叫信息报告给系统。如果此时主 App 不在运行，系统将唤醒主 App，并将消息投递给它，最后主 App 再使用 CallKit API 显示呼叫界面。



下面是在主 App 中使用 NEAppPushManager 的示例代码：

```
// Create Configuration

import NetworkExtension

let manager = NEAppPushManager()
manager.matchSSIDs = [ "Cruise Ship Wi-Fi", "Cruise Ship Staff Wi-Fi" ]
manager.providerBundleIdentifier = "com.myexample.SimplePush.Provider"
manager.providerConfiguration = [ "host": "cruiseship.example.com" ]
manager.isEnabled = true

manager.saveToPreferences { (error) in
    if let error = error {
        // Handle error
        return
    }
    // Report success
}
```

上述代码创建了一个 NEAppPushManager 实例，并配置实例的各个属性值。matchSSIDs 表示在指定的 WiFi 网络下才启用本地推送连接。providerBundleIdentifier 表示 App Extension 的包名，providerConfiguration 是传给 App Extension 的一些配置，在 App Extension 内可以通过 NEAppPushProvider 的 providerConfiguration 属性获取。isEnabled 表示使用这个配置开启本地推送连接。最后调用 saveToPreferences 方法持久化配置。下面是 App Extension 实现 NEAppPushProvider 子类的示例代码：

```
// Manage App Extension life cycle and report VoIP call

class SimplePushProvider: NEAppPushProvider {

    override func start(completionHandler: @escaping (Error?) -> Void) {
        // Connect to your provider server
        completionHandler(nil)
    }

    override func stop(with reason: NEProviderStopReason,
                       completionHandler: @escaping () -> Void) {
        // Disconnect your provider server
        completionHandler()
    }

    func handleIncomingVoIPCall(callInfo: [AnyHashable : Any]) {
        reportIncomingCall(userInfo: callInfo)
    }
```

系统调用 start(completionHandler:)方法启动 App Extension，在这个方法内 App Extension 与本地业务服务器建立连接。当 App Extension 停止运行，系统调用 stop(with:)方法，

在这个方法内 App Extension 断开与业务服务器的连接。handleIncomingVoIPCall(callInfo:)方法在收到 VoIP 呼叫时被调用，在方法内 AppExtension 调用 reportIncomingCall(userInfo:)将该事件上报给系统。随后系统将会唤醒主 App，并将呼叫信息传递给主 App。主 App 处理系统传入的呼叫信息示例代码：

```
// Handling incoming VoIP call in the containing app

class AppDelegate: UIResponder, UIApplicationDelegate, NEAppPushDelegate {
    func application(_ application: UIApplication,
                     didFinishLaunchingWithOptions launchOptions:
                     [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        NEAppPushManager.loadAllFromPreferences { (managers, error) in
            // Handle non-nil error
            for manager in managers {
                manager.delegate = self
            }
        }
        return true
    }

    func appPushManager(_ manager: NEAppPushManager,
                        didReceiveIncomingCallWithUserInfo userInfo: [AnyHashable: Any] = [:]) {
        // Report incoming call to CallKit and let it display call UI
    }
}
```

以上代码在 AppDelegate 的 didFinishLaunchingWithOptions 方法内使用 NEAppPushManager 加载所有配置，并设置每个 NEAppPushManager 示例的代理属性。系统会在主线程中将接收到呼叫信息通过 didReceiveIncomingCallWithUserInfo 方法投递给主 App。主 App 必须通过 CallKit API 将呼入信息上报给 CallKit 展示呼叫界面。

价值场景

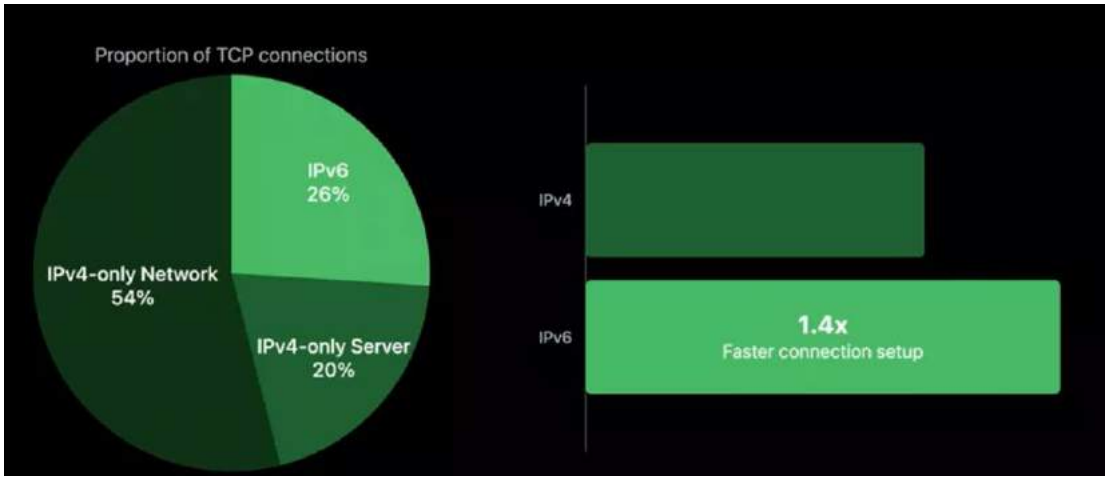
如果只考虑推送本身，对于手淘或者大部分消费类应用来说，笔者认为价值并不大，因为此类 APP 不可能在一个封闭的本地网络里去部署资源来提供服务能力。这里一个可应用的点在于：设备一旦进入特定网络环境，触发 App Extension，进而唤起主 App，应用可在后台完成一定事务。因为 iOS App 一直缺乏后台服务能力，这种特定网络环境的触发唤醒，极大的补充了这一能力。

现代网络技术的应用

苹果在这次 WWDC 中，把一些较新的网络技术，对应用的体验提升，做了一个简单综述，包括 IPv6、HTTP/2、TLS1.3，MTCP、以及 HTTP/3。这些技术在手淘基本都有涉及，有些是已经是大规模部署、有些是正在逐步推进中。对各个应用来说，如果已经在应用这些技术了，则云端均尽可能标准化，便于进一步推广和复用。这里简单的对苹果的综述做一个搬运：

IPv6

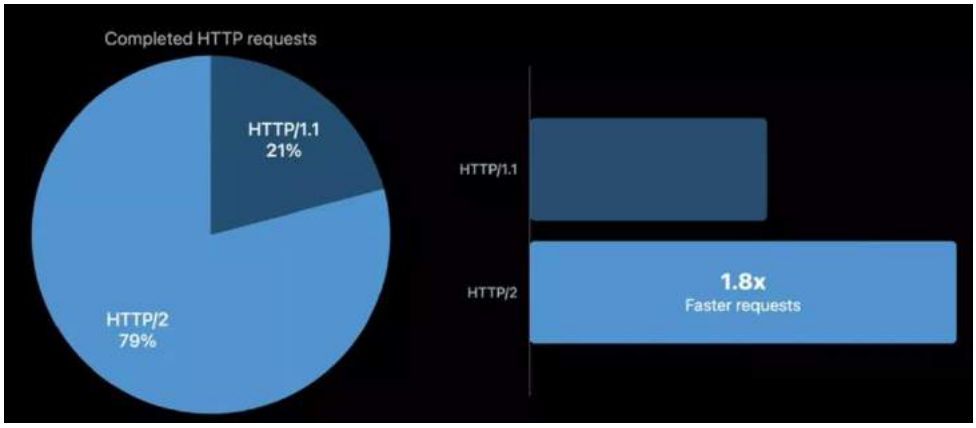
苹果根据最新统计，苹果全球设备 TCP 连接占比中，IPv6 占比 26%，IPv4 占比 74%，其中 74%的占比中有 20%是因为服务端没有开启 IPv6 支持。在建连时间方面，由于减少了 NAT 使用，提高了路由效率，IPv6 的建连时间比 IPv4 快 1.4 倍。开发者只需使用 URLSession 和 Network.framework API，IPv6 网络适配将自动支持。



以上是苹果的数据。阿里巴巴集团从 18 年开始大力推进 IPv6 的建设，目前我们在 IPv6 整体应用规模上在业界是属于头部大厂。但根据我们应用的实际效果数据，以及业界友商的应用数据，性能提升并不明显。以及工信部的 IPv6 建设目标来看，性能提升也不是 IPv6 建设的目标，只要达到 IPv4 同等水平即可。IPv6 的意义就在解决 IPv4 地址空间枯竭的问题，更多的在规模、安全，而不是性能提升。就企业而言，例如可以降低 IPv4 地址的购买费用；就国家而言，IPv6 突破了 IPv4 中国境内无 DNS 根结点的风险。IPv6 目前阶段在国内尚处于发展中，基础运营商覆盖、家用网络接入设备支持、应用服务的支持，正在快速发展中。

HTTP/2

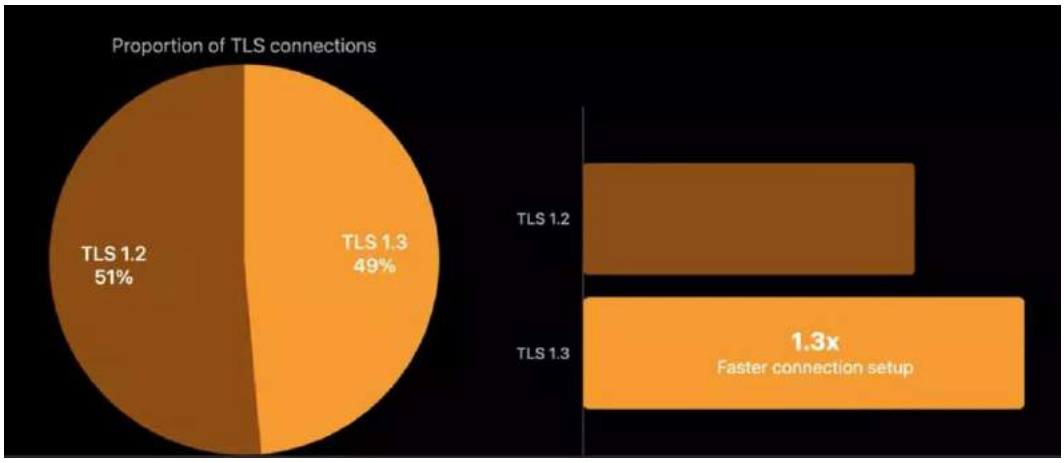
HTTP/2 的多路复用特性使得对同一服务器的多个请求复用到单个连接上，不必等待前一个请求响应结束才能发送下一个请求，不仅节省了时间也提升了性能。头部压缩特性提升了带宽利用率，通过简化消息内容，从而降低消息的大小。根据最新统计，在 Safari 中 HTTP2 Web 流量占比 79%，HTTP/2 比 HTTP/1.1 快 1.8 倍。如果服务端支持 HTTP/2，URLSession 将默认使用 HTTP/2。



在手淘业务中，全面应用 HTTP2 已经有三四年之久，其使用规模比苹果统计的结果要高的多，取得了巨大的体验提升收获。手淘的核心流量分为两类：API 请求 MTOP 于图片 CDN 请求，这两类的 HTTP2 的流量占比约超过 98%，基本实现核心流量全部长连化。但当前手淘的 HTTP2 技术在设备支持之前即大规模应用，协议栈完全自研，为进一步提升建联速度，又做了一些预置证书的优化。下一步将逐步使基础网络依赖标准化平台能力，降低对私有协议栈的依赖，可降低包大小以及提升产品复用体验。

TLS1.3

TLS1.3 通过减少一次握手减少了建连时间，通过形式化验证(Formal Verification)与减少被错误配置的可能性，提高了通信安全。从 iOS 13.4 开始，TLS1.3 默认在 URLSession 和 Network.framework 开启。根据最新统计，在最新的 iOS 系统上，大约 49%的连接使用 TLS1.3。使用 TLS1.3 比使用 TLS1.2 建连时间快 1.3 倍。如果服务端支持 TLS1.3，URLSession 将默认使用 TLS1.3。



目前手淘的 HTTP/2 的 TLS version 仍然还是基于 TLS1.2，不过为了解决低版本 TLS 的性能之殇，手淘网络通过预置证书与自定义 SSL 握手流程，创新自研了 slight-ssl 协议，实现了 0rtt 的效果。TLS 1.3 标准在系统层面的全面支持，对应用来说是一个明确的技术趋势信号，我们的网络协议需尽快标准化，对网络管道两端来说，客户端应用层网络接口需全面升级到 NSURLSession 或 Network.framework，实现对系统标准能力的应用；云端也许全面支持 TLS1.3，可不依赖端侧 SDK 即可提供更新安全、高效、标准的网连接。

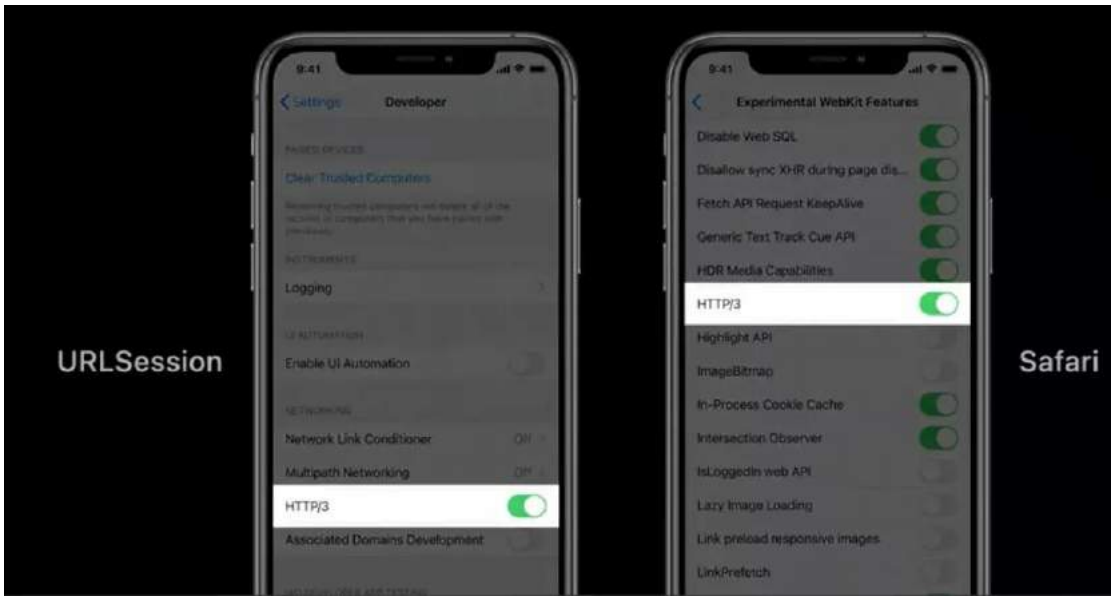
MultiPath TCP

MultiPath TCP 允许在一条 TCP 链路中建立多个子通道。当设备切换网络时，单个 TCP 连接依然可以继续使用。苹果的 Apple Music 与 Siri 服务都使用了 MultiPath TCP。Apple Music 在使用 MultiPath TCP 之后,音乐播放卡顿次数减少了 13%,卡顿持续时间减少了 22%。开启 MultiPath TCP 需要客户端和服务端都支持才能生效，服务端支持 MultiPath TCP 可参考：http://multipath-tcp.org/

其实手淘在这方面也有类似的优化尝试：多网卡：同时通过 Wi-Fi 与蜂窝网连接目标服务器，提升数据传输速度。其技术原理与 MTCP 不一样，但也是想在上层起到类似作用：通过多路连接，提升数据交换带宽。业界也有类似的产品，例如华为的 LinkTurbo。

HTTP/3

HTTP/3 是下一代 HTTP 协议，它是构建在新的 QUIC 传输协议之上，QUIC 协议内建了对 TLS1.3 的支持，并提供了与 HTTP/2 一样的多路复用功能，并进一步减少了队头阻塞的发生，让单个请求或相应的丢失不影响到其他请求。使用 QUIC 的 HTTP/3 还具有较高的保真度信息，以提供改进的拥塞控制和丢包恢复。同时也包括内建的移动性支持，这样网络切换不会导致正在进行的操作失败，可以无缝在不同网络之间切换。不过 HTTP/3 目前还处于草案阶段，iOS 14 和 MacOS Big Sur 包括了一个对使用 URLSession 的 HTTP/3 的实验预览支持，这个功能可以在开发者设置中开启。同时 Safari 的 HTTP/3 支持也可在开发者设置中开启。



在手淘中，我们的 QUIC 应用应该会早于苹果系统先行支持，目前已经在灰度中。

WWDC 20 Apple Widget：下一个顶级流量入口？

作者 | 王浙剑(柘剑)
出品 | 淘系技术

2020 年 6 月 22 日，苹果召开了第一次线上的开发者大会 - WWDC20。这可谓是一次可以载入史册的发布会，宣布了 ARM 架构 Mac 芯片、软硬件的生态大统一、iOS 14 系统界面大改等一系列激动人心的消息。

当然，最让我感兴趣的就是让 iOS 界面大改的 Widget 了。过去几年，iOS 的桌面交互体验可谓是一言难尽，Widget 的加入无疑是一次比较大的破局。在看发布会的时候，我的脑海里就浮现出一个问题："这会是下一个互联网公司竞争的流量入口吗？"

先不抛结论，让我们先看一下 WWDC20 介绍了哪些新东西。

什么是 Widget?

Widget 不是一个小型的 App，它是一种新的桌面内容展现形式，主要是用于弥补主应用程序无法及时展示用户所关心的数据。如下图所示：

一个优秀的 Widget 需要有三个特点：简单明了（Glanceable）、恰当展示（Relevant）、个性化定制（Personalized）

简单明了（Glanceable）

Widget 不是一个小型的 App，这句话被反复提起。一般用户每天进入主屏幕的次数超过 90 次，但停留的总时长不过几分钟。通常来说用户只会在主屏幕上停留片刻时间，就会跳转到其他地方，所以并不需要任何复杂的交互设计来增强 Widget 的作用，也不需要复杂的样式来丰富 Widget 的内容，简单明了的内容才是 Widget 的关键。

和安卓的 Widget 不太一样，苹果设计的 Widget 并不支持任何交互行为，也不建议大家设计过于复杂的样式来呈现内容，这也非常符合苹果对于主屏幕的改进一直保持克制的特点。

恰当展示（Relevant）

苹果期望 Widget 可以和正在执行或者考虑的事情紧密的结合。比如，早上起床，用户最关心天气怎么样，Widget 可以展示一下天气情况；起床后，用户就要了解一下一天的行程，Widget 可以展示一下 Reminders 中的内容；等到一天忙完了，准备睡觉的时候，可以用 Widget 打开音乐稍微放松一下。为此，苹果系统提供了一个叫智能叠放（Smart Stacks）的功能，智能叠放是一个 Widgets 的集合。系统会根据每个人的习惯，借助端智能的能力，自动的显示准确的 Widget 在最顶部。

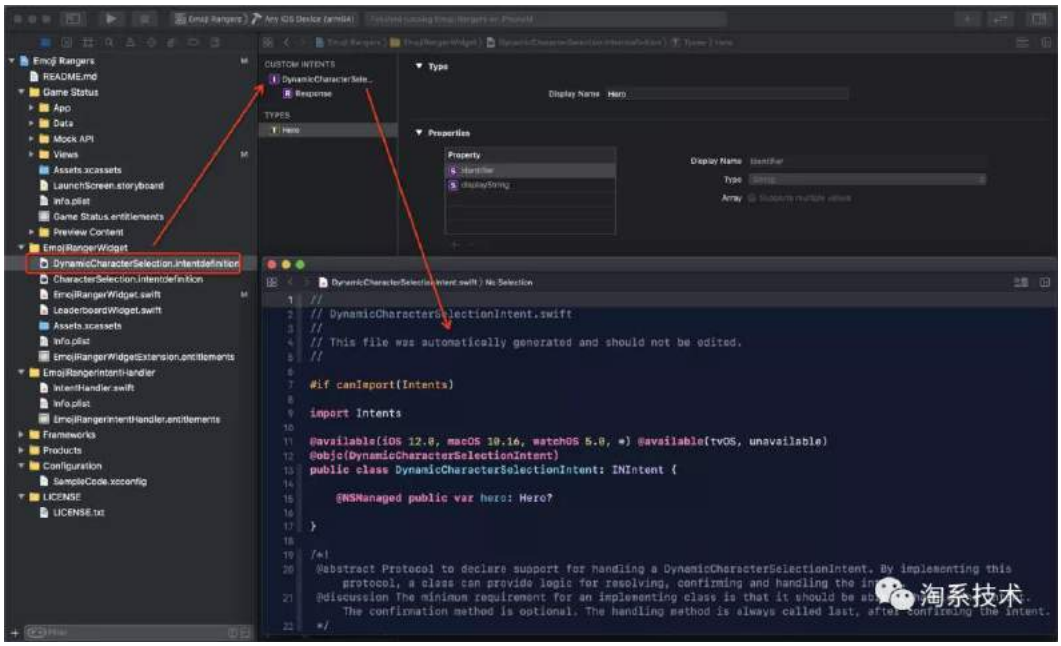
当然，苹果也考虑到了一些特殊的场景，比如 Widget Gallery 浏览时，提供了 Snapshot 的能力给到开发者可以定制展示样式，当加载内容的时候提供了 Placeholder UI API 而不是单调的 loading 加载框来避免过多的白屏的尴尬局面。这些设计的目的只有一个，苹果期望 Widget 可以在任何特定的场景都可以展示合理的样式。

个性化定制（Personalized）

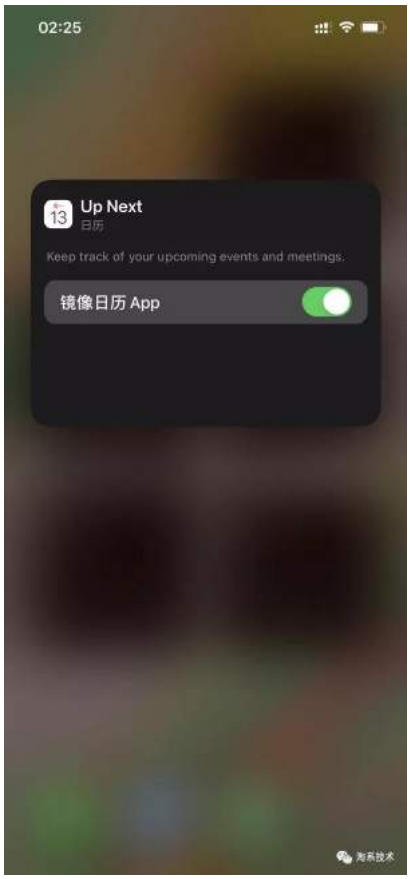
Widget 需要一定的定制能力，比如当我添加一个天气 Widget，我只需要关心杭州的天气怎么样。为了实现这个能力，苹果给 Widget 提供了 Configuration 的能力。顾名思义，就是可配置。一共有两种配置类型：

- dianStaticConfiguration，也就是用户无需配置，展示的内容只和用户信息有关系。
- IntentConfiguration，支持用户配置及用户意图的推测功能。

IntentConfiguration 的实现是基于 Intents.framework，开发过 SiriKit 和 Shortcuts 一定知道 Intents API 是用于了解用户意图的。其实就是一个智能的表单系统，开发者创建一个 SiriKit Intent Definition File 之后，只需要简单的配置，Xcode 会自动帮你生成对应的代码和类型。

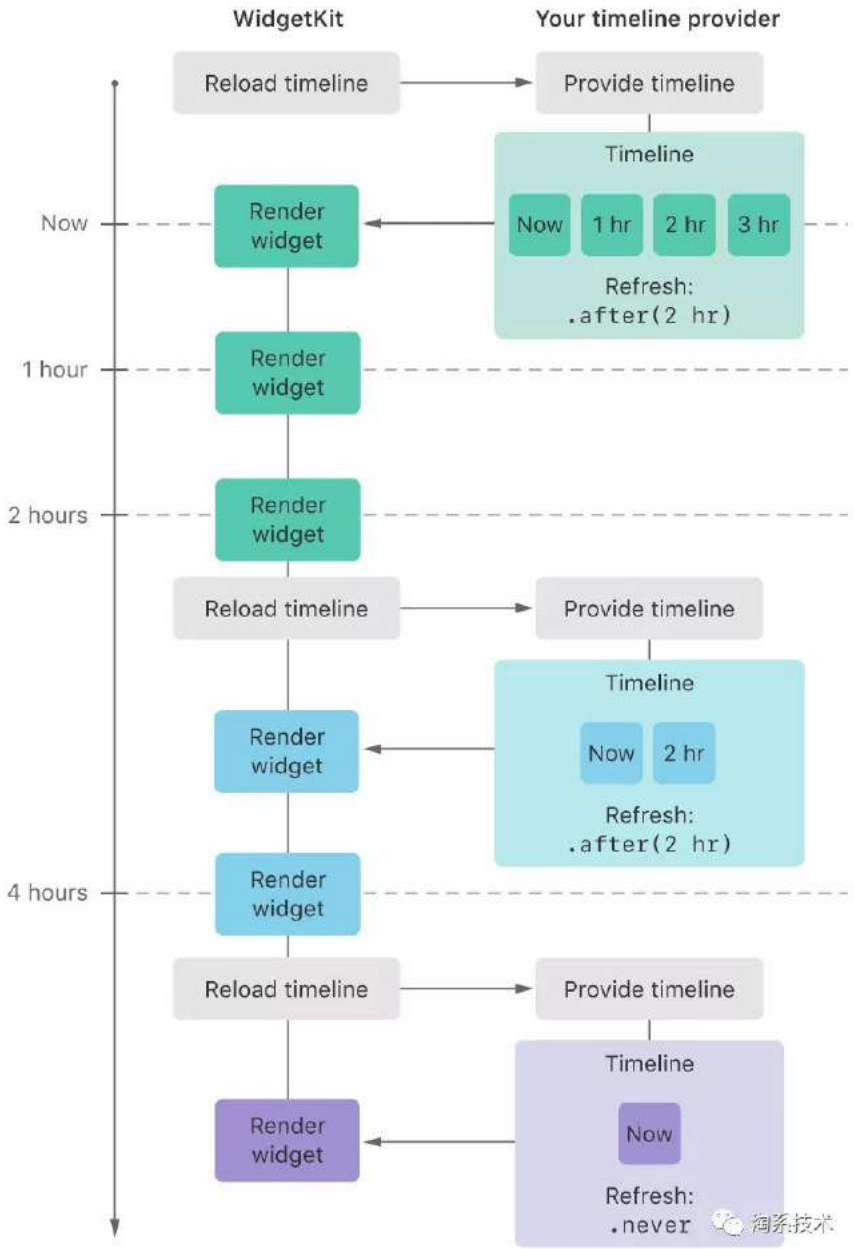


当开发者编写完配置之后，会借助 Intents.framework 的能力，在运行的时候直接绘制出一个配置页面（如下图所示），开发者并不需要关心如果编写这个页面。



Widget 的刷新方式

Widget 的刷新方式是很特别的，相当的克制。在展开讲刷新方式之前，要讲一个概念，叫 Timeline。顾名思义，就是时间线，下面的图就是一条 Timeline。



当系统的 WidgetKit 调用 Reload Timeline API 之后，会要求 Widget Extension 的 Timeline Provider 提供一组 TimelineEntry 和 ReloadPolicy，用来后续刷新页面。

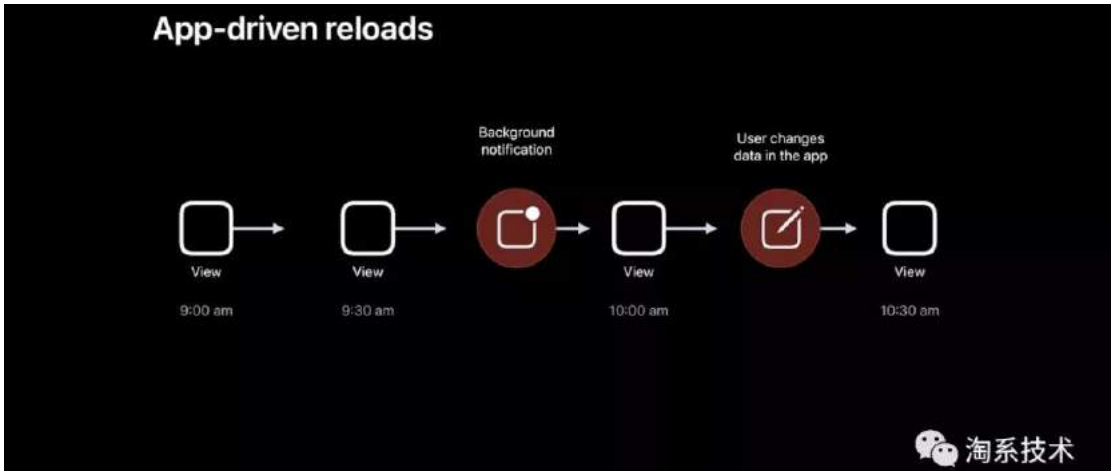
这里的概念比较多，我们一个一个来解释。

首先，Widget 的刷新完全由 WidgetCenter 控制。开发者无法通过任何 API 去主动刷新 Widget 的页面，只能告知 WidgetCenter，Timeline 需要刷新了。

系统提供了两种方式来驱动 Timeline 的 Reload。System Reloads 和 App-Driven Reloads。

System Reloads: 这个行为由系统主动发起，会调用一次 Reload Timeline 向 Widget 请求下一阶段刷新的数据。系统除了会按时发起 System Reloads 之外，还会借助端智能的能力，动态决策每个不同的 TimeLine 的 System Reloads 的频次。例如被查看次数很大程度上直接决定了 System Reloads 的频率。当然还有一些由于设备环境变化触发的行为也会触发 System Reloads，比如设备时间进行了变更。

App-Driven Reloads：指的是 App 请求 Widget 下一阶段刷新的数据。这里也要分两种场景，应用在前台运行和应用在后台运行。当应用在前台运行的时候，App 可以直接请求 WidgetCenter 的 API 来触发 Reload Timeline；而当应用处于后台时，后台推送（Background Notification）也可以触发 Reload Timeline。



注意，前面所提到的 Reload Timeline 并不是直接刷新 Widget，而是 WidgetCenter 重新向 Widget 请求下一阶段的数据。而 Timeline Provider 就是提供这个数据的对象。

而 Timeline Provider 提供的数据有两部分，一部分是 TimelineEntry，另外一部分是 ReloadPolicy。

TimelineEntry 是某个时间节点下 Widget 需要呈现的视图信息和时间点。
而 ReloadPolicy 则是接下来这段时间 Timeline 的刷新策略，一共有三种：

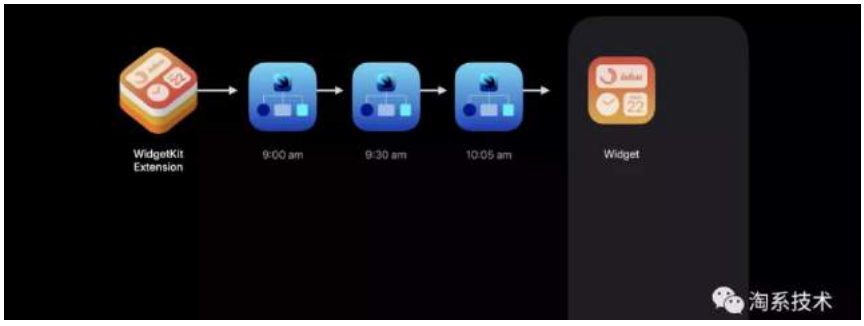
- atEnd: 是指 Timeline 执行到最后一个时间片的时候再刷新。
- atAfter: 是指在某个时间以后有规律的刷新。
- never: 是指以后不需要刷新了。什么时候需要重新刷新需要 App 重新告知 Widget。

当 Timeline Provider 提供完下一阶段的数据之后，就会停止运行。系统也会根据 entry 的信息，到点对 Widget 的展示内容进行刷新。值得一提的是，WidgetKit 会把 Timelines 所定义的 Entries 对应的 Views 结构信息缓存到磁盘，然后在刷新的时候才通过 JIT 的方式来渲染。这使得系统可以在极低电量开销下为众多 Widgets 处理 Timelines 信息。

简而言之，苹果对 Widget 的刷新相当的克制。开发者无法直接决定 Widget 刷新，只能提供刷新策略。具体的时间和节奏全部由系统来控制。苹果这么做，大概率是为了提高主屏幕的性能和减少电量开销上的考虑。

Widget 和 SwiftUI

Widget 只能用 SwiftUI 来进行开发，确切的说，Widget 的本质是一个随着时间线而更新的 SwiftUI 视图。



当我最开始知道这个限制的时候，说实话是相当震惊的。众所周知，SwiftUI 是一个去年才发布的新技术，而且最开始的时候 SwiftUI 是相当不稳定的，以至于苹果自己都是建议开发者暂时不要用到生产环境上，Widget 作为系统主屏幕的功能，强制使用这么新的技术，会不会太激进了?

显然是不会。苹果要求 Widget 只能使用 SwiftUI 主要是基于几点考虑:

1. SwiftUI 经过一年的发展，有了很大的提升，不仅可以使用 SwiftUI 来构建整个应用程序，而且在一些方面已经优于基于 UIKit 的开发方式了。具体的内容，大家可以看一下《详解 WWDC 20 SwiftUI 的重大改变及核心优势》。
2. 苹果正在布局跨平台，大统一的策略。Widget 作为系统的核心功能，使用 SwiftUI 是唯一的选择。SwiftUI 精美的 DSL 设计，使得开发者使用一套代码在 iOS、iPadOS、macOS、watchOS 和 tvOS 等多个平台展示不同的样式可以轻松的实现。（Widget 只会在 iOS、iPadOS 以及 macOS 上展示）。
3. 使用了 SwiftUI 使得 Dynamic Type 和 Dark Mode 等问题适配起来成本很低。
4. 只有使用 SwiftUI 才能达到很多对于 Widget 的限制。倘若可以使用 UIKit 开发者可能有无数种办法绕过苹果的限制。比如开发无法使用 UIViewRepresentable 来桥接 UIKit，只要使用任何 UIKit 的元素会直接 Crash。
5. 将 Swift 语言和 SwiftUI 的重要程度提升了一大截。

Widget 的展示形式

一个 App 可以对应多个 Widget Extension

你可以使用 WidgetBundle 来进行组装。苹果并没有对 Widget Extension 有数量上的限制。所以为了避免大家开发过多的 Widget Extension 导致搜索起来麻烦，在 Widget Gallery 中只能看到一个条目。

点击查看动图演示

一个 Widget Extension 一共只有三种尺寸

考虑到简单明了的特点以及手机屏幕的空间有限的问题。苹果只提供了三种样式可以选择，systemSmall（2 2 icon 区域）、systemMedium（24 icon 区域）、systemLarge（4 * 4 icon 区域）。

同一种 Widget 可以被多次添加到主屏幕中

而且对于每一个 Widget 来说，都有其对应的独立 TimeLine，相互独立，互不干扰。

开发者无法开发智能叠放（Smart Stacks）

开发者无法开发一个 Widget 的集合。智能叠放（Smart Stacks）是一个系统特有的能力，对于开发者来说，唯一可以做的就是主动提供相关性信息。前文提到了 Timeline 的数据又一组 TimelineEntry 组成，而每个 TimelineEntry 除了包含时间点和视图信息以外，还可以包含一个 TimelineEntryRelevance 对象，用来表示这个 entry 的相关性。

不可交互，只可点击

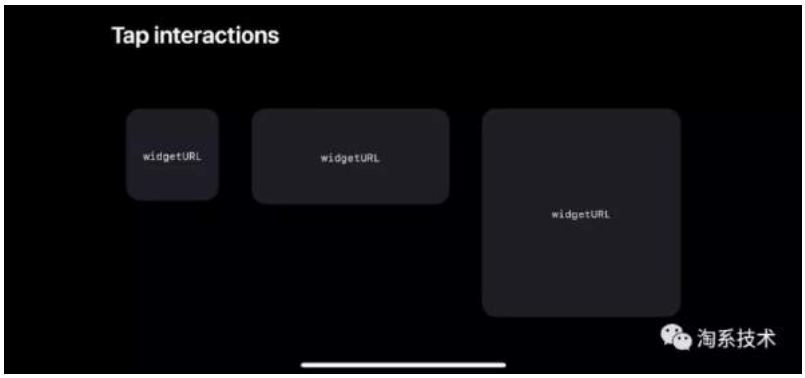
Widget 的 UI 是无状态的，不支持滚动，也不支持像 Switch 一样的互动元素。唯一开放的能力只有通过点击和 DeepLink 来唤起主 App。

苹果提供了两种 API 给到开发者，第一种是 SwiftUI widgetURL API，代码如下所示：

```
struct EmojiRangerWidgetEntryView: View {

    var entry: Provider.Entry

    var body: some View {
        HStack(alignment: .top) {
            AvatarView(entry.character)
                .foregroundColor(.white)
            Text(entry.character.bio)
                .padding()
                .foregroundColor(.white)
        }
        .padding()
        .widgetURL(entry.character.url)
    }
}
```

淘系技术

而 widgetURL 的可点击区域如下：



对于 systemSmall 类型来说，只支持 widgetURL 的方式，但是 systemMedium 和 systemLarge 还可以使用 SwiftUI Link API，代码如下所示：

```
    var body: some View {
        VStack(spacing: 48) {
            ForEach(
                characters.sorted { $0.healthLevel > $1.healthLevel }, id: \.self) { character in
                Link(destination: character.url) {
                    HStack {
                        Avatar(character: character)
                        VStack(alignment: .leading) {
                            Text(character.name)
                                .font(.headline)
                                .foregroundColor(.white)
                            Text("Level \(character.level)")
                                .foregroundColor(.white)
                            HealthLevelShape(level: character.healthLevel)
                                .frame(height: 10)
                        }
                    }
                }
            }
        }
    }
}
```

淘系技术

而 Link 的可点击区域如下：

同时，为了性能和耗电量的考虑。Widget 不能展示视频和动态图像。所以期待通过动效吸引用户眼球的方式可以暂时息熄火了~

总结与展望

Widget 的出现，让 iOS 系统的桌面有了破局，一定会有很多产品都期待借助 Widget 来丰富自己产品的内容表达。

但是，Widget 设计的初衷是简单明了的在恰当的时机展示一些带有个性化定制的内容，为了不让主屏幕的整体使用体验变得复杂，Widget 从技术上就做的很克制，限制了很多很多的能力。因此我认为 Widget 不会成为下一个互联网公司竞争的流量入口，它会成为 App 提高用户体验的利器。

从技术角度看，SwiftUI Only 这种看似"激进"的策略其实也是一种信号，其实也是在告诉大家苹果对于 Swift 以及 SwiftUI 的重视程度。

虽然，从目前来看 Pure SwiftUI 的设计，可以做的事情真的很少，但是我也相信，苹果会不断优化 Pure SwiftUI 的能力。让开发者可以以最低的开发成本，适配更多的平台。

最后，也期待大家可以好好研究一下 Widget，结合自己的产品，给到用户极致的用户体验。

参考：

iOS 14 Preview

Widgets code-along

Meet WidgetKit

What's new in SwiftUI

Add configuration and intelligence to your widgets

Build SwiftUI views for widgets

Creating a Widget Extension

Building Widgets Using WidgetKit and SwiftUI

Making a Configurable Widget

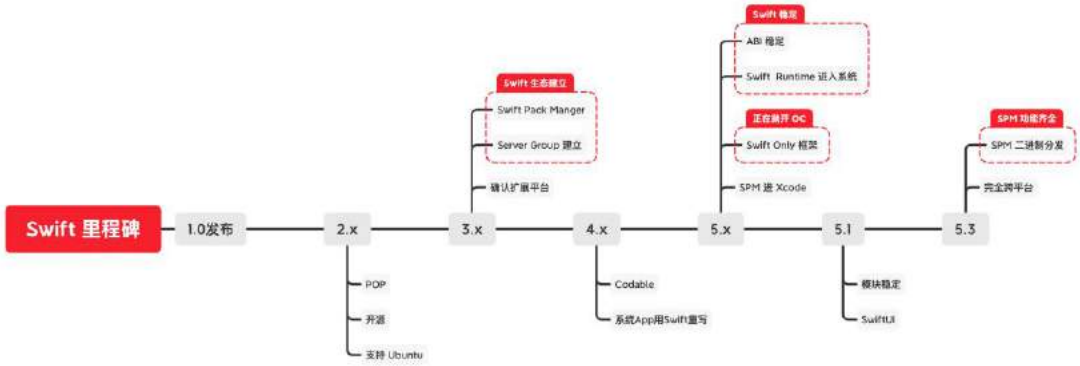
Keeping a Widget Up To Date

WWDC 20 Swift 5.3 又更新了什么新奇爽快的语法?

作者 | 蒋志远(星志)
出品 | 淘系技术

Swift 在 WWDC14 正式发布到 2019，经过 5 年的不断迭代，这其中经历了标准库变动，语法的增减。首先使用 Swift 作为开发语言的开发者们都苦不堪言，戏称《Swift 从入门到重学》，几乎每一年 Swift 都会迎来比较大的改动，甚至 API 都发生了变化。WWDC 19 苹果发布了 Swift 5.0，苹果终于宣布 Swift 的 ABI 稳定。这标志着 Swift 这门语言已经趋于稳定，在 2019 至 2020 的迭代中，Swift 5.2 也做到了模块稳定，之前的大修大改已经不会在出现了。

Swift 发展里程碑



有意思的是，在 WWDC16 中有一页 PPT 写下了 Goals for Swift 3：

- Develop an open community
- Portability to new platforms
- Get the fundamentals right
- Optimize for awesomenss

如今已经 2020 年，再回头看这些目标，Swift 5.3 几乎完全实现了。Swift 可谓一直不忘初心，朝着认为正确的方向不断的努力。

语法的不断成熟

每年 WWDC 比较期待的一个点就是看看 Swift 又加了一些什么新奇爽快的语法。我先列举一下 Swift 5.3 新增的语法:

- [SE-0249] KeyPath as Function
- [SE-0279] Multiple trailing closure
- [SE-0281] Type-based program entry point (@main)
- [SE-0266] Synthesized comparable conformation for enum types
- [SE-0269] Increased availability of implicit self in closure
- [SE-0276] Multi-pattern catch clauses
- [SE-0280] Enum cases as protocol witnesses
- [SE-0267] Where clauses on contextually generic declarations
- [SE-0270] Collection operations on noncontiguous elements
- [SE-0264] Standard library preview package
- [SE-0253] Callable values of user-defined nominal types
- [SE-0263] String initializer with access to uninitialized storage

上面列举的 12 个语法改动，有兴趣的同学可以到 swift-evolution 去查看详细的内容。其中比较有意思的几个改动就是 [SE-279]，[SE-281]和[SE-0269]，我们来细细评味一下。

Design for SwiftUI

[SE-279] Multiple trailing closure 多重尾闭包这个新特性算是一个非常大的改动。这个特性的出现影响了 Swift 非常重要的地方，ABI 设计。先来几个官方的例子:

```
// 1. Single closure argument -> trailing closure
UIView.animate(withDuration: 0.3) {
  self.view.alpha = 0
}
// 2. With trailing closure
UIView.animate(withDuration: 0.3, animations: {
  self.view.alpha = 0}) { _ in
  self.view.removeFromSuperview()
}
// 3. Multiple closure arguments -> no trailing closure
UIView.animate(withDuration: 0.3, animations: {
  self.view.alpha = 0}, completion: { _ in
  self.view.removeFromSuperview()
})
// 4. Multiple trailing closure arguments
UIView.animate(withDuration: 0.3) {
  self.view.alpha = 0} completion: { _ in
  self.view.removeFromSuperview()
}
```

上面的代码中 2 的情况是经常出现的，尾闭包的含义就不够明确，以前最佳的方式应该为 3，但是两个闭包被套在一个小括号里，使得括号多重嵌套，可读性下降。为了解决这个问题，苹果使用了第 4 中方法--多重尾闭包。可以从代码看出可读性得到了很大的提升。

这一看似优雅的设计其实引入了很多语言的复杂度，有了这个特性你甚至可以写出类似下面的代码，看上去就像是创造了一个 if-else 语句，但其实它是一个函数。

```
func when<T>(
  _ condition: @autoclosure () -> Bool,
  then: () -> T,
  `else`: () -> T
) -> T {
  condition() ? then() : `else`()
}// using syntaxwhen(2 < 3) {
  print("then")
} else: {
  print("else")
}// equivalent to:
when(2 < 3, then: { print("then") }, else: { print("else") })
```

这就对 SDK 的开发者提出了更高的要求，需要小心的使用这个特性来设计 API，防止这个特性被滥用导致代码的可读性下降，甚至造成歧义。苹果在 WWDC20 中也强调了这一点。

还有两个特性其实很简单，[SE-0281] 是 @main 来标记程序入口，[SE-0269] 是让开发者在特定的情况下不需要再写多余的 self.。

苹果推出这几个语言特性，就是为了让战略性的项目 SwiftUI 的 API 更加简洁明了。

可以看下面的例子:

```
@main // <- @main
struct MyApp: App {
    var body: Scene {
        WindowGroup {
            Text("Hello World!")
        }
    }
}//------
Gauge(value: acidity, in: 3...10) {
    Image(systemName: "drop.fill")
} currentValueLabel: { // <- multiple trailing closure
    Text("\(acidity, specifier: "%.1f")")
} minimumValueLabel: {
Text("3")
} maximumValueLabel: {
    Text("10")
}//------
struct ContentView : View {
    var landmark: Landmark
    var body: some View {
        Button(action: {
            landmark // <- no self.
        }) {
            Text("Button")}
    }}
```

苹果也不是第一次为了 SwiftUI 改 Swift 的语法，[SE-0255] Implicit returns from single-expression functions 就是为了让 SwiftUI 的 body 不用写 return，强化 DSL 风格。

苹果为了 SwiftUI，推出了 functionBuilder 来实现 DSL 形式的代码；为了 SwiftUI，推出了许多让它更漂亮的语法特性；为了 SwiftUI，不惜为语言引入更多的复杂度。

下限低，上限高

Swift 的最初的设计方向就导致了它是一门下限很低上限很高，入门容易精通难，使用容易设计难的语言。它一系列的语法糖和语法设计，包括类型推断系统，都是让它成为十分亲民的语言，而它的一系列语法特性，让它在设计 API、SDK 实现的时候其复杂程度也不亚于其他语言。

Swift is a general-purpose, multi-paradigm, compiled programming language developed by Apple Inc. for iOS, iPadOS, macOS, watchOS, tvOS, and Linux.

---- Wikipedia

维基百科的后半部分可能有些局限了，但是前半部分说明了它是一门多编程范式的语言，Swift 可以支持 面向对象编程(OOP)，面向协议编程(OPP)，声明式编程(DP)，函数式编程(FP)，泛型编程(GP)。

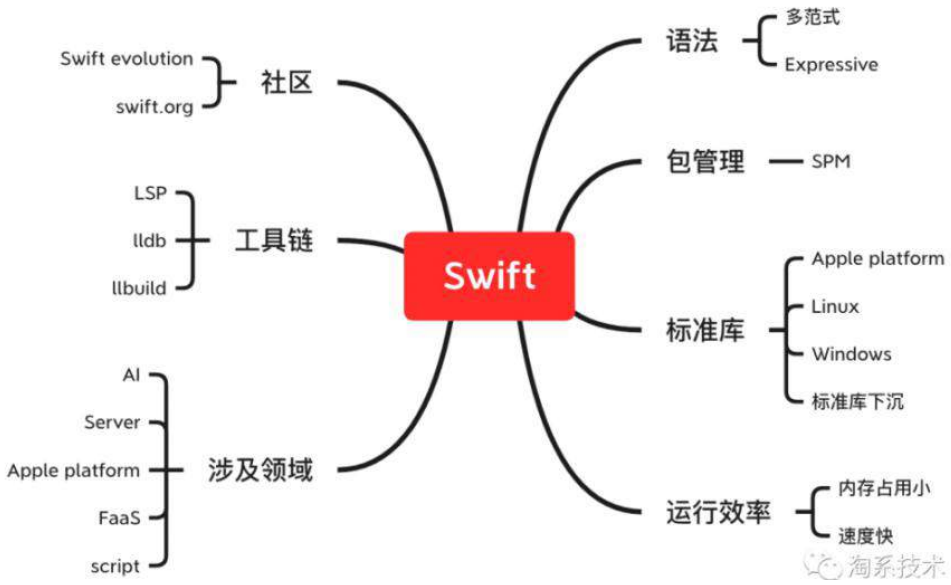
Objective-C 已经发展了这么多年，如此成熟，为什么现在苹果要开始抛开它？Objective-C 本来就是生于一个面向对象编程范式起飞的一年，与 C++ 一样为了拓展 C 命令式编程范式而诞生的语言。当时 C 语言虽然也可以实现 OOP，但是语法设计成为了限制。在当代计算机编程语言研究演进下，出现了很多编程范式的新理论，如函数式编程，元编程等等，同样 OC 可以通过它的方式来实现，但是语言的冗杂和 OOP 的设计已经不能更好的表达这些概念，就犹如纯 C 来表达 OOP 一样。

苹果推出 Swift 就是为了摆脱 OC 的束缚，让它能更好的践行现代的编程理论，所以才会诞生出 SwiftUI，才会有 Combine、map/filter/reduce 等这些库和 API。类型推断和元编程的理论也让 Swift 在保留强类型的环境下还能保持如此简洁优雅，可读性强的代码。而这些是 OC 无法做到的。尝试使用 OC 对这些现代概念做表达的 SDK 都会显得十分冗杂。

现在 Swift 语法已经不会再大变，有如此现代、安全、稳定、富有想象力的语言，有什么理由不真香呢?

土壤已经肥沃

我们先通过下面的脑图感受一下 Swift 更迭到 5.3 已经取得的成就。



现在 Swift 生态环境已经自成一派，有一套完整的工具链保证开发，有更独立的标准库让它可以自由迁移，包管理让丰富了它的 SDK，同时还具有 Native 语言的各种优势。在这么多年的发展中，它的能力已经触及到了 AI、Server、Mobile Device、FaaS，强大的标准库让它甚至可以当脚本语言使用。

Swift 如今已经不再孤立无援，开源让 Swift 变得生机勃勃，合理优雅的设计和开放的态度让全球的开发者们都在不断的完善它。

在 Swift 的大生态中，包管理工具是最值得一说的。一门语言它的能力，取决于它是否有强有力的 SDK，还有就是获取他们的途径。基础的 Swift 的能力都由工具链和标准库提供，而强有力的 SDK 最好的方式就是通过包管理工具快速获取。Swift Package Manager 就是 Swift 的军械库。

Swift Package Manager

看一下 Swift Package Manager(SPM) 的发展历程，SPM 从 Swift 3.0 时就发布，当时只支持 Git 远端仓库，支持支源码发布。WWDC 19，Xcode 引入了一个新的架构 XCFramework，一个旨在能打包多个平台 framework 的文件结构。WWDC20 SPM 宣布正式支持发布二进制 framework。

WWDC20 SPM 走出的这一步标志着功能的完善。为什么这么说，因为在 iOS 届，解决了 Xcode 无包管理工具难题，让众多优秀的开源库能被快速获取的包管理工具 -- Cocoapods。在 SPM 出现之前，Cocoapods 的功能已经非常完善，打败了当时另外的一款包管理工具 Carthage。

Cocoapods 作为基于 Xcode 开发的第三方包管理工具，通过修改 Xcode 的工程信息来实现处理各个包的依赖问题。它支持依赖的二进制分发、源代码分发，基于 framework 良好的资源管理。现在许多公司的大型 App 也是用 Cocoapods 做模块化组件分离，通过二进制依赖的方式来提高打包速度。

然而比较可惜的是它工作流是脱离 Xcode 的，Podfile 更新等操作都是在工程之外。语言使用的是 Ruby，虽然使用了一些操作使 Podfile 变成了 DSL，但是对于开发者而言是有一定门槛的，特别是需要编辑发布 SDK 所使用的 podspec 时，对照文档，无代码提示的编写让写正确配置文件都成了难事。Cocoapods 还不得不设计出 podspec lint 来帮组开发者确认这个事情。

对于 Swift PM 这个亲儿子而言，就不存在这个问题，SPM 已经深入的集成到了 Xcode 中，所有的操作都可以利用 Xcode 完成。在编写 Swift Package 时，Xcode 现在已经具备将 Package.swift 作为工程项目打开，并且现在工程模板中已经包含了 Swift Package。强大的 Xcode 自动补全让编写 Package.swift 不再是一件难事。如果脱离 Xcode，利用 swift-lsp 结合现在热门的文本编辑器，如 VSCode 等都可以实现相应的功能。SPM 现在已经是独立于 Xcode 的存在，可以为 Swift 提供强有力的支持。

SPM 在功能层面已经不逊于 Cocoapods，在 WWDC20 时，SPM 已经支持以二进制库形式分发 Package，SPM 也可以管理包中的资源和本地化，基本的能力已经与 Cocoapods 差异不大，然而最关键的是，SPM 是纯 Swift ，开源，还有苹果官方支持的包管理工具。

新瓶酿酒酒更香

国外的不少 APP 已经迁移到了 Swift；三方开源库如 AFNetworking 已经用 Swift 重写为 Alamofire，Lottie 已经完全被 Swift 重写替代。苹果也推出了许多 Swift Only 的库。苹果也在利用 Swift 为 UIKit，GCD 等基础库不断提供更具有表现力的 API。

Uber 完成了迁移，收获了 Swift 极强的稳定性。Alamofire，SnapKit 用 Swift 重写，获得了更加具有表现力的 API，开发者更容易接受且喜爱。Reactive 系列的编程风格在 Swift 上大放异彩。

机会与展望

现在 Swift 发展了这么多年，遍地开花，Objective-C 已经有在逐渐被废弃的趋势。现在很有可能都很难招到认真学习过 OC 的应届生了。

Swift 在集团内部的发展，还有很多的工作要做，这其中充满了挑战。我们也在积极探索 Swift 在手淘的落地，取得了 Swift 5.1 能模块在手淘中正确运行起来的阶段性成就。但是比较遗憾的是还有很多模块是无法让 Swift 正确跑起来的，以现在的工具链来说甚至源码依赖调试都无法做到。

但是这都是暂时的，现在时机已经成熟，Swift 的语言特性，SPM，工具链，标准库都已经足够强大。未来，我们会尽快大力升级 Swift 的基建。让 Swift 的花在集团内开起。先定个小目标，希望 Swift 能成为集团 iOS 客户端开发的首选语言。

参考：

SwiftUI 背后那些事儿

WWDC20 What's new in Swift

WWDC19 What's new in Swift

WWDC18 What's new in Swift

WWDC17 What's new in Swift

WWDC16 What's new in Swift

WWDC20 Swift packages: Resources and localization

WWDC20 What's new in SwiftUI

Swift 5 时代的机遇与挑战到底在哪里？

Swift Evolution

WWDC 20 Metal 新特性：大幅度提升 iOS 端性能

作者 | 陈昱(岑彧)
出品 | 淘系技术

作为较早在客户端侧选择 Flutter 方案的技术团队，性能和用户体验一直是闲鱼技术团队在开发中比较关注的点。而 Metal 这样的直接操作 GPU 的底层接口无疑会给闲鱼技术团队突破性能瓶颈提供一些新的思路。
本文将会详细阐述一下这次大会 Metal 相关的新特性，以及对于闲鱼技术和整个淘系技术来说，这些新特性带来了哪些技术启发与思考。

前言

Metal 是一个和 OpenGL ES 类似的面向底层的图形编程接口，通过使用相关的 api 可以直接操作 GPU ，最早在 2014 年的 WWDC 的时候发布。Metal 是 iOS 平台独有的，意味着它不能像 OpenGL ES 那样支持跨平台，但是它能最大的挖掘苹果移动设备的 GPU 能力，进行复杂的运算，像 Unity 等游戏引擎都通过 Metal 对 3D 能力进行了优化，App Store 还有相应的运用 Metal 技术的游戏专题。

闲鱼团队是比较早在客户端侧选择 Flutter 方案的技术团队,当前的闲鱼工程里也是一个较为复杂的 Native-Flutter 混合工程。作为一个 2C 的应用，性能和用户体验一直是闲鱼技术团队在开发中比较关注的点。而 Metal 这样的直接操作 GPU 的底层接口无疑会给闲鱼技术团队突破性能瓶颈提供一些新的思路。

下面会详细阐述一下这次大会 Metal 相关的新特性，以及对于闲鱼技术和整个淘系技术来说，这些新特性带来了哪些技术启发与思考。

Metal 相关新特性

Harness Apple GPUs with Metal

这一章其实主要介绍的是 Apple GPU 的在图形渲染上的原理和工作流，是一些比较底层的硬件原理。当我们使用 Metal 进行 App 或者是游戏的构建的时候，Metal 会利用 GPU 的 tile-based deferred rendering (TBDR)架构给应用和游戏带来非常可观的性能提升。这一章主要就是介绍 GPU 的的架构和能力，以及 TBDR 架构进行图像渲染的原理和流程。总之就是号召开发者们使用 Metal 来构建应用和游戏。因为这个 session 没有涉及到上层的软件开发，就不对视频的具体内容进行赘述了。详情可见：Harness Apple GPUs with Metal

Optimize Metal apps and games with GPU counters

这一章主要介绍了 Xcode 中的 GPU 性能分析工具 Instrument，这个工具现在已经支持了 GPU 的性能分析。然后从多个方面分析了 GPU 的性能瓶颈，以及性能瓶颈出现时的优化点。总体来说就是通过性能分析工具来优化我们的 App 或者游戏，让整个画面更加流畅。整个章节主要分为五个部分：

总体介绍

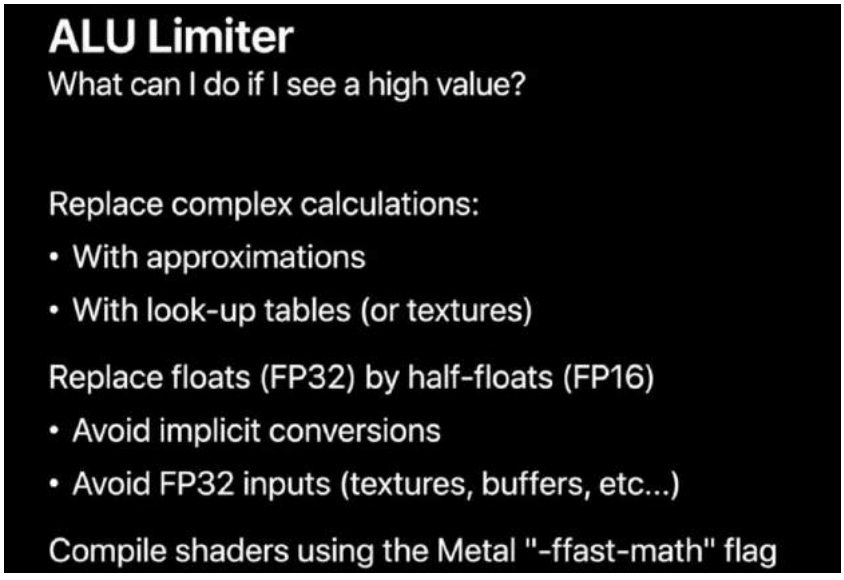
这个环节主要是快速回顾了一下 Apple 的 GPU 的架构和渲染流程。然后因为很多渲染任务都需要在不同的硬件单元上进行，例如 ALU 和 TPU。他们对不同的吞吐量有着不同的度量。有很多 GPU 的性能指标需要被考虑，所以推出了 GPU 性能计数器。这个计数器可能测量到 GPU 的利用率，过高和过低都会造成我们的渲染性能瓶颈。关于计数器的具体使用，参考官方的 video 效果会更好：Optimize Metal apps and games with GPU counters（6：37~9：57），主要使用了 Instrument 工具，关于工具的全面详细的使用可以参考 WWDC19 的 session videoGetting Started with Instruments

性能瓶颈分析

这一章主要介绍了造成 GPU 性能瓶颈的各个方面以及它们的优化点。主要分为六个方面，如下图所示：

1. Arithmetic（运算能力）

GPU 中通常通过 ALU（Arithmetic Logic Unit）来处理各种运算，例如位操作，关系操作等。他是着色器核心的一部分。在这里一些复杂的操作或者是高精度的浮点运算都会造成一些性能瓶颈，所以给出以下建议来进行优化：

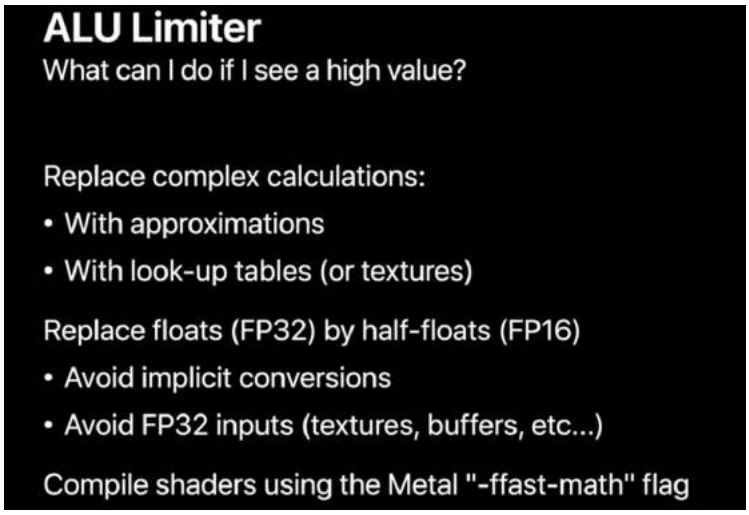


如上图所示，我们可以使用近似或者是查找表的方式来替换复杂的运算。此外，我们可以将全精度的浮点数替换为半精度的浮点数。尽量避免隐式转换，避免 32 位浮点数的输入。以及确保所有的着色器都使用 Metal 的"-ffast-math"来进行编译。

2. Texture Read and Write

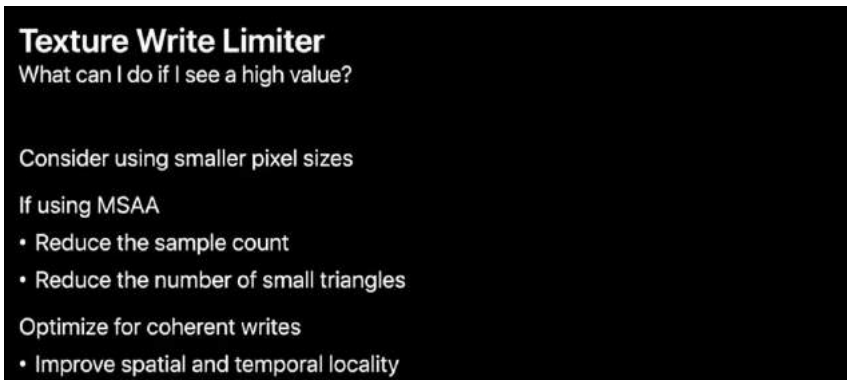
GPU 通过 Texture Processing Unit 来处理纹理的读写操作。当然在读写的过程中也会遇到一些性能瓶颈问题。这里从读和写两个部分分别来给出优化点：

Read



如上图所示，我们可以尝试使用 mipmaps。此外，可以考虑更改过滤选项。例如，使用双线性代替三线性，降低像素大小。确保使用了纹理压缩，对 Asset 使用块压缩（如 ASTC），对运行时生成的纹理使用无损纹理压缩。

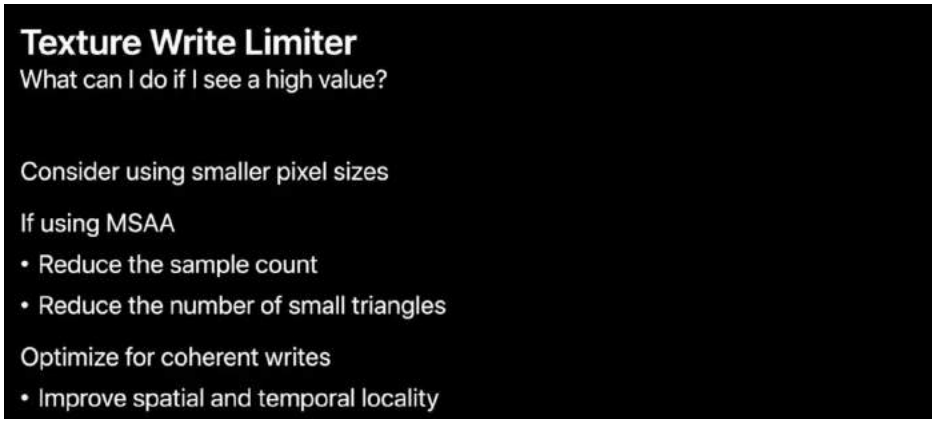
Write



如上图所示，我们应该注意到像素的大小，以及每个像素中唯一 MSAA 样本的数量。此外，可以尝试一些优化一些逻辑写法。

Tile Memory Load and Store

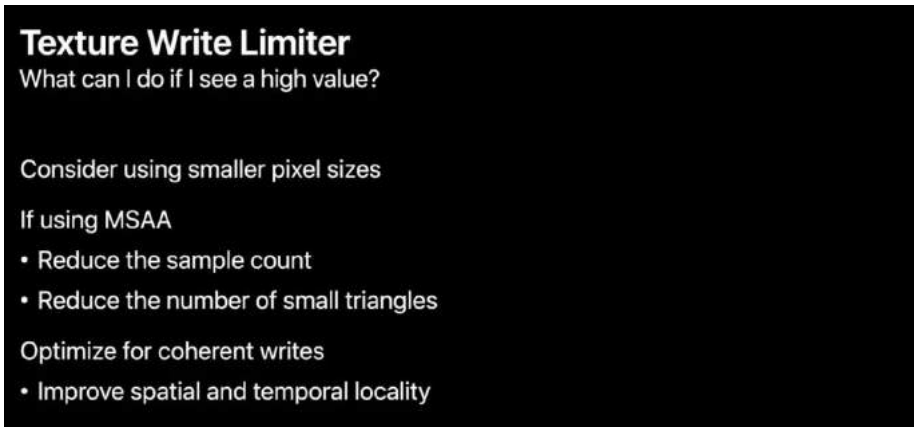
图块内存是一组存储 Thread Group 和 ImageBlock 数据的高性能内存。当从 ImageBlock 或是 Threadgroup 读取或写入像素数据时，比如在使用 Tile 着色器时或者是计算分派时，可以访问到 Tile 内存。那当使用 GPU 性能计数器发现这个方面的性能瓶颈时，我们可以如下图所示进行优化。



考虑减少 threadgroup 的并行，或者是 SIMD/Quadgroup 操作。此外，确保将线程组的内存分配和访问对齐到 16 字节。最后，可以考虑重新排序内存访问模式。

Buffer Read and Write

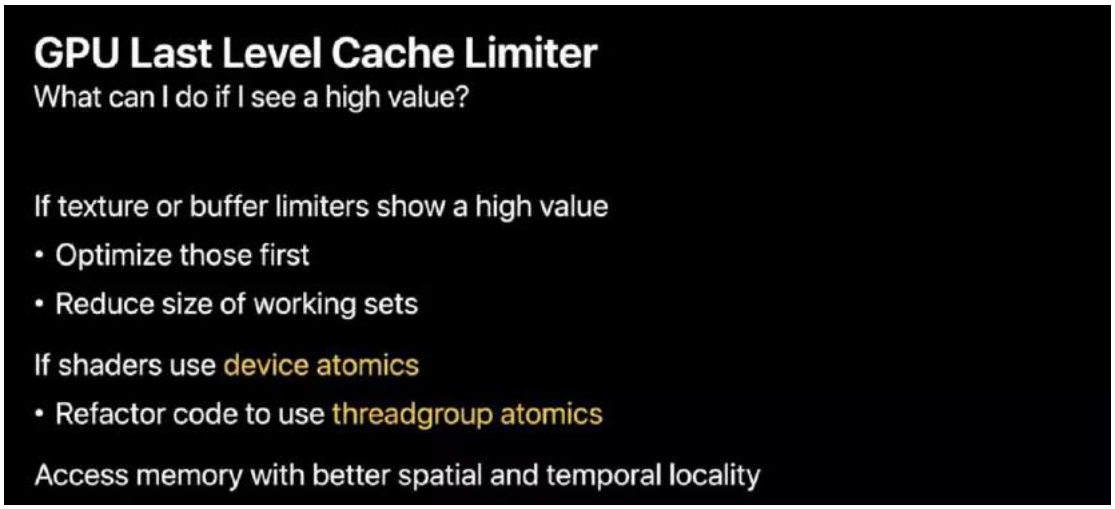
在 Metal 中，缓冲区只被着色器核心访问。在这个地方发现了性能瓶颈。我们可以如下图所示进行优化：



可以更大力度的压缩打包数据，例如使用例如 packed_half3 这样小的类型。此外，可以尝试向量化加载和存储。例如使用 SIMD 类型。避免寄存器溢出，以及可以使用纹理来平衡工作负载。

GPU Last Level Cache

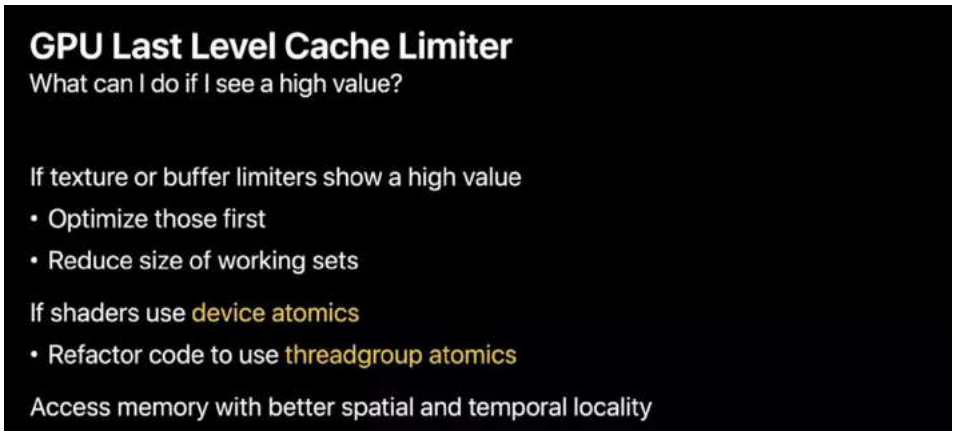
如果在这个方面，我们的 GPU 性能计数器显示一个过高的值。我们可以如下图这样优化：



如果纹理或者是缓存区也同样显示一个过高的值，我们可以把这个优化放到第一优先级。我们可以考虑减小工作集的大小。如果 Shader 正在使用 Device Atomics，我们可以尝试重构我们的代码来使用 Threadgroup Atomics。

Fragment Input Interpolation

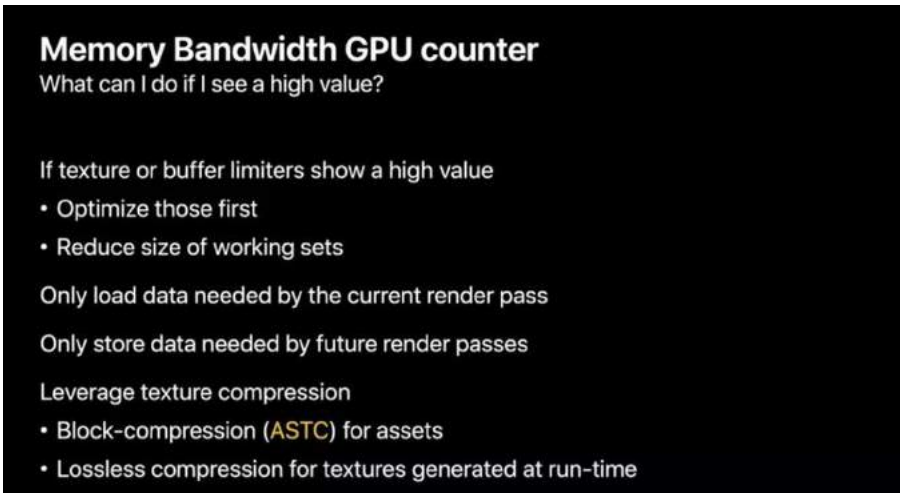
分段输入插值。分段输入在渲染阶段由着色器核心进行插值。着色器核心有一个专用的分段输入插值器。这个是比较固定和高精度的功能。我们能优化的点不多，如下图所示：



尽可能的移除传递给分段着色器的顶点属性。

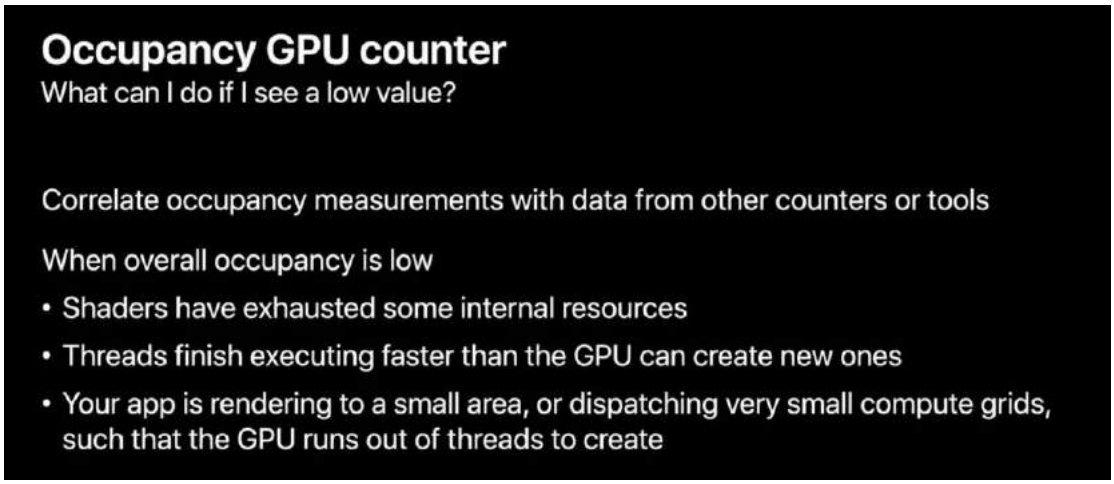
内存带宽

内存带宽也是影响我们 GPU 性能的一个重要因素。如果在 GPU 性能计数器的内存带宽模块看到一个很高的值。我们就应该如下图所示来进行优化：



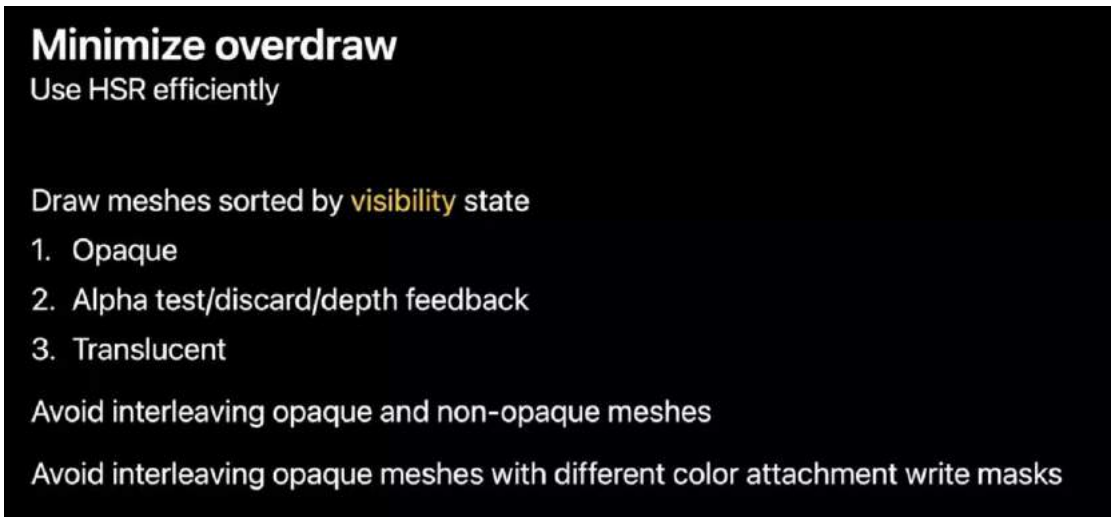
如果纹理和缓存区也同样显示比较高的值，那优化优先级应该排到第一位。优化方案也是较少 Working Set 的大小。此外，我们应该只加载当前渲染过程需要的数据，只存储未来渲染过程需要的数据。然后就是确保使用纹理压缩。

Occupancy



如果我们看到整体利用率比较低，这意味着 Shader 可能已经耗尽了一些内部资源，比如 tile 或者 threadgroup 内存。也可能是线程完成执行的速度比 GPU 创建新线程的速度快。

避免重复绘制



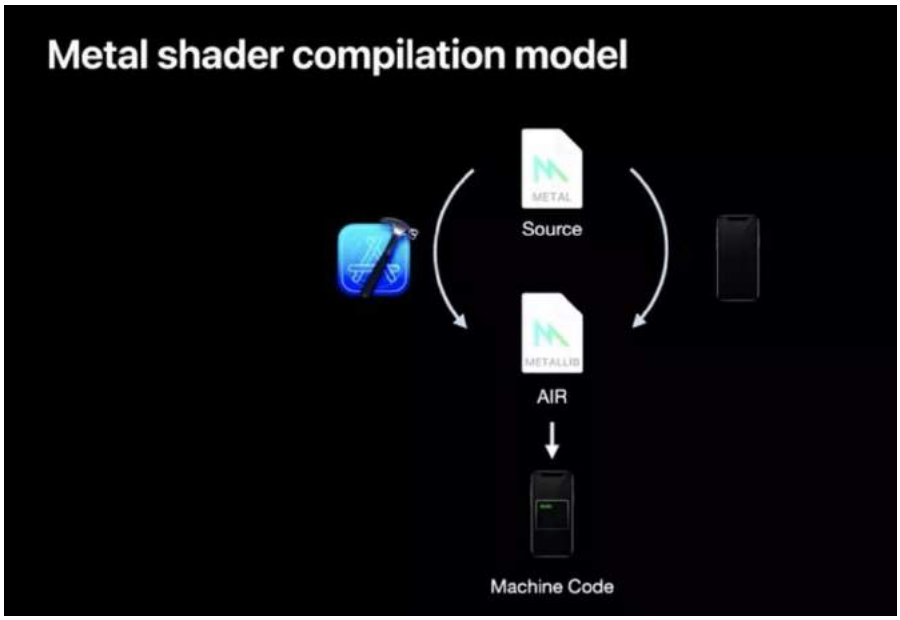
我们通过 GPU 计数器可以统计到重复绘制的区域，我们应该高效使用 HSR 来避免这样的重绘。我们可以如图所示的顺序来进行绘制。

Build GPU binaries with Metal

这一章主要给开发者们介绍了一种使用 Metal 的编程工作流，可以通过优化 Metal 的渲染编译模型来增强渲染管线，这个优化可以在应用程序启动，特别是首次启动时大大减少 PSO(管线状态对象)的加载时间。可以让我们的图形渲染更加的高效。整个章节主要分为四个部分：

Metal 的 Shader 编译模型概述

众所周知，Metal Shading Language 是 Apple 为开发者提供的 Shader 编程语言，Metal 会将编程语言编译成为一个叫做 AIR 的中间产物，然后 AIR 会在设备上进一步编译，生成每个 GPU 所需的特定的机器码。整个过程如下图所示：



上述过程在每个管线的生命周期中都会发生，当前 Apple 为了加速管线的重新编译和重新创建流程，会缓存一些 Metal 的方法变体，但是这个过程还是会造成屏幕的加载耗时过长。

而且在当前的这个编译模型中，应用程序不能在不同的 PSO（管线状态对象）中重用之前生成的机器码子程序。

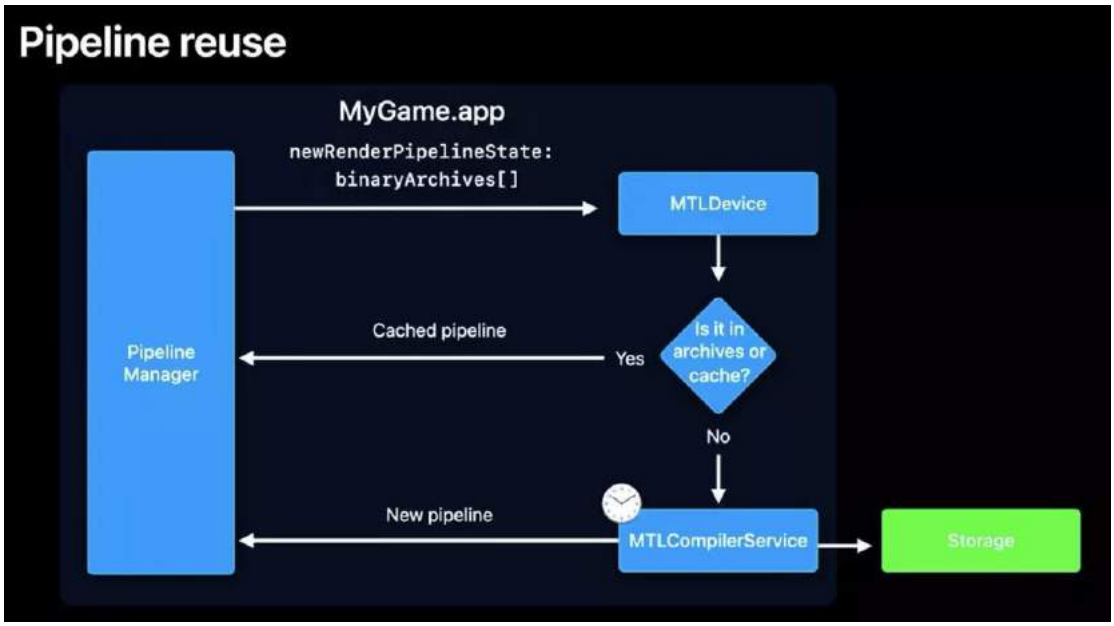
所以我们需要一种方法来减少这个整个管线编译（即源代码->AIR->GPU 二进制代码）的时间成本，还需要一种机制来支持不同 PSO 之间共享子程序和方法，这样就不需要将相同的代码多次编译或者是多次加载到内存中。这样开发者们就可以使用这套工具来优化 App 首次的启动体验。

Metal 二进制文件介绍

Metal 二进制文件就是解决上述需求的方法之一，现在开发者们可以直接使用 Metal 为二进制文件来控制 PSO 的缓存。开发者可以收集已编译的 PSO，然后将它们存储到设备中，甚至可以分发到其他兼容的设备中（同样的 GPU 和同样的操作系统），这种二进制文件可以看做一种 Asset。下面是一些例程和示意图：

```
//创建一个空的二进制文件
let descriptor = MTLBinaryArchiveDescriptor()
descriptor.url = nil
let binaryArchive = try device.makeBinaryArchive(descriptor:descriptor)
```

```
//Populating an archive
// Render pipelines
try binaryArchive.addRenderPipelineFunctions(with: renderPipelineDescriptor)
// Compute pipelines
try binaryArchive.addComputePipelineFunctions(with: computePipelineDescriptor)
// Tile render pipelines
try binaryArchive.addTileRenderPipelineFunctions(with: tileRenderPipelineDescriptor)
```



```
//重用已编译的方法

// Reusing compiled functions to build a pipeline state object from a file

let renderPipelineDescriptor = MTLRenderPipelineDescriptor()
// ...
renderPipelineDescriptor.binaryArchives = [ binaryArchive ]

let renderPipeline = try device.makeRenderPipelineState(descriptor:
renderPipelineDescriptor)

//序列化
let documentsURL = FileManager.default.urls(for: .documentDirectory, in: .userDomainMask).first!
let archiveURL = documentsURL.appendingPathComponent("binaryArchive.metallib")
try binaryArchive.serialize(to: NSURL.fileURL(withPath: archiveURL))
```

```
//反序列化
let documentsURL = FileManager.default.urls(for: .documentDirectory,in: .userDomainMask).first!
let serializeURL = documentsURL.appendingPathComponent("binaryArchive.metallib")
let descriptor = MTLBinaryArchiveDescriptor()
descriptor.url = NSURL.fileURL(withPath: serializeURL)
let binaryArchive = try device.makeBinaryArchive(descriptor: descriptor)
```

总的来说就是这个 Metal 二进制文件可以提供开发者手动管理管线缓存的方法，这样就可以从一个设备中获取这些文件并部署到其他兼容的设备上，在 iOS 环境下，极大地减少了第一次安装游戏或应用以及设备重启后的管道创建时间。可以优化应用的首次启动体验和冷启动体验。

Metal 对动态库的支持

动态库将允许开发者编写可重用的库代码，却可以减少重新编译程序的时间和内存成本，这个特性将会允许开发者将计算着色器和程序库动态链接。而且和二进制文件一样，动态库也是可序列化和可转移的。这也是解决上述需求的方案之一。

在 PSO 生成的时候，每个应用程序都需要为程序 library 生成机器码，而且使用相同的程序库编译多个管线会导致生成重复的机器码。由于大量的编译和内存的增加，这个可能会导致更长的管线加载时间。而动态库就可以解决这个问题。

Metal Dynamic Library 允许开发者以机器码的形式动态链接,加载和共享工具方法。代码可以在多个计算管线中重用，消除了重复编译和多个相同子程序的存储。而且这个 MTLDynamicLibrary 是可序列化的，可以作为应用程序的 Asset 使用。MTLDynamicLibrary 其实就是多个计算管线调用的导出方法的集合。

大致的工作流程如下：我们首先创建一个 MTLLibrary 作为我们指定的动态库，这个可以将我们的 metal 代码编译为 AIR。然后我们调用方法 makeDynamicLibrary，这个方法需要指定一个唯一的 installname，在管线创建时，linker 将会使用这个名字来加载动态库。这个方法可以将我们的动态库编译成为机器码。这就完成了动态库的创建。

对于动态库的使用来说：通过设置 MTLCompileOptions 里的 libraries 参数，就可以完成动态库的加载和使用了。代码如下：

```
//使用动态库进行编译
let options = MTLCompileOptions()
options.libraries = [ utilityDylib ]
let library = try device.makeLibrary(source: kernelStr, options: options)
```

开发工具介绍

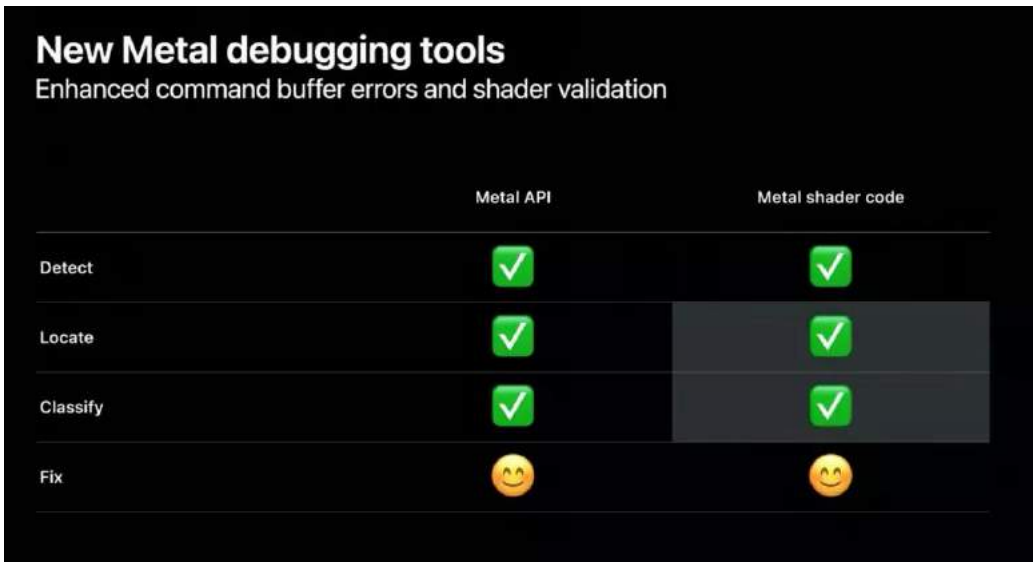
这个部分主要介绍了构建 Metal 二进制文件和构建动态库的具体工具和方法。以视频的形式可能会更好的表现，详情可见：Build GPU binaries with Metal （从 22：51 开始）

Debug GPU-side errors in Metal

这一章主要介绍的是 GPU 侧的 bug，当前如果我们的应用程序出现了 GPU 侧的 bug，他的错误日志常常都不能让开发者很直观的定位到错误的代码范围和调用栈。所以在最新的 Xcode 中，增强了关于 GPU 侧的 debug 机制。可以像在代码侧发成的错误一样不但能定位到错误原因，还有错误的调用堆栈和各种信息都可以详细的查看到。让开发者能更好的修复代码造成的 GPU 侧的渲染错误。

Enhanced Command Buffer Errors

这是当前的错误日志上报，我们可以看到 GPU 侧的错误日志不像 Api 的错误日志一样可以让开发者很快的定位到错误原因和错误的代码位置。



而最新的 Metal debugging 工具就增强了这方面的能力，让 Shader 的 code 也可以像 Api 代码一样提供错误定位和分类能力。

我们通过以下代码便可以启用增强版的 commandbuffer 错误机制

```
//启用增强版的 commandbuffer 错误机制
let desc = MTLCommandBufferDescriptor()
desc.errorOptions = .encoderExecutionStatus
let commandBuffer = commandQueue.makeCommandBuffer(descriptor: desc)
```

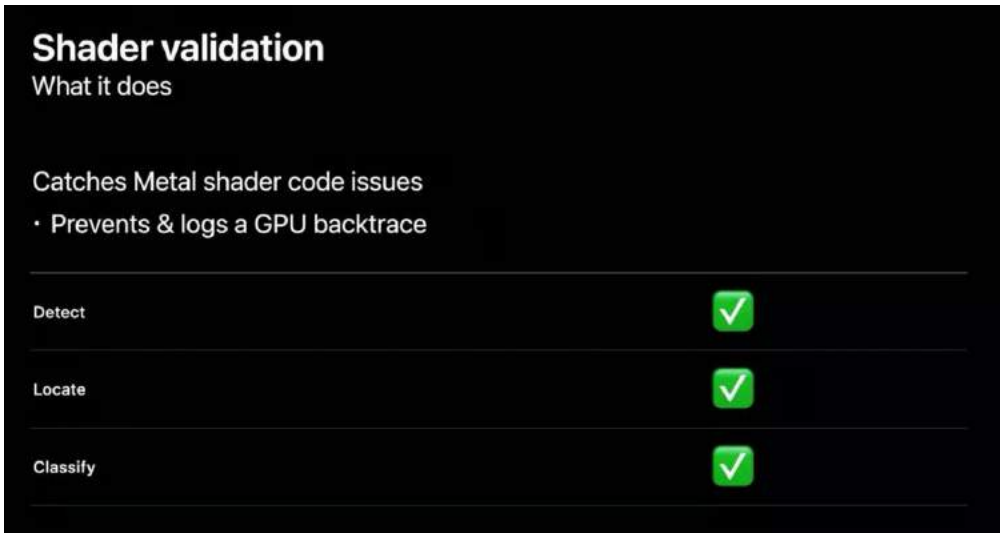
错误一共有五种状态：

我们也可以通过以下代码来打印 error：

```
//打印 commandbuffer 的错误
if let error = commandBuffer.error as NSError? {

    if let encoderInfos =
        error.userInfo[MTLCommandBufferEncoderInfoErrorKey]
        as? [MTLCommandBufferEncoderInfo] {

        for info in encoderInfos {
            print(info.label + info.debugSignposts.joined())
            if info.errorState == .faulted {
                print(info.label + " faulted!")
            }
        }
    }
}
```

开发者可以在开发时和测试时启用优化版的错误机制

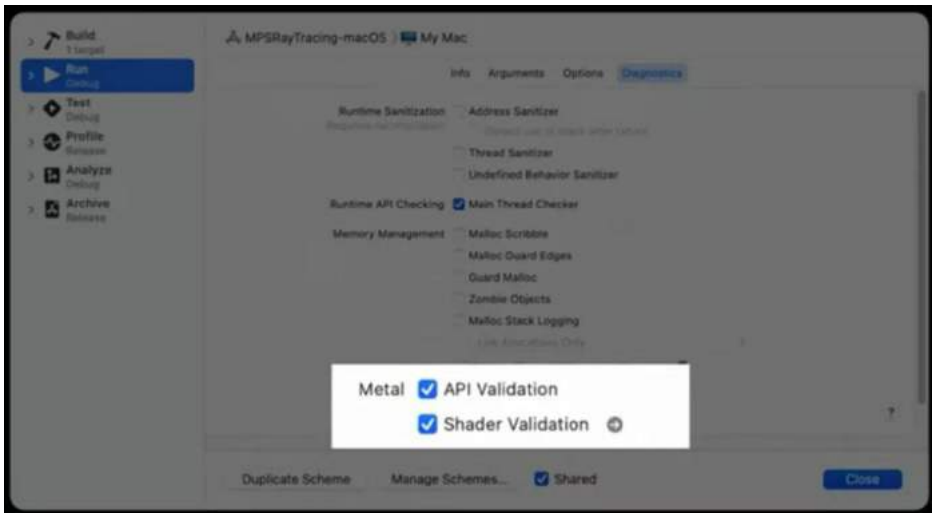
Shader Validation



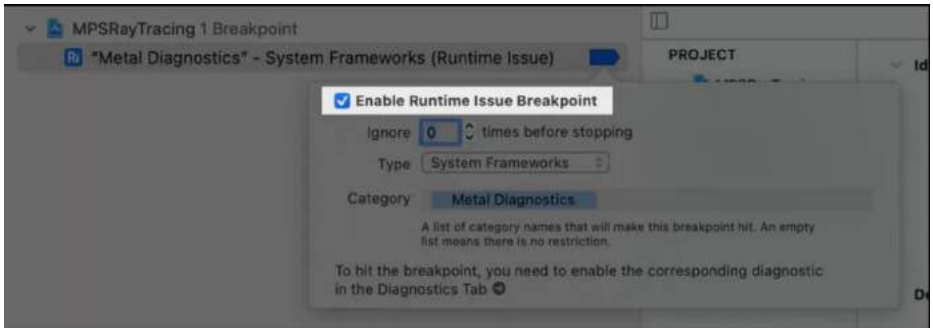
如上图所示，这个功能可以在 GPU 侧发生渲染错误时自动定位和 catch 到错误并定位到代码，以及获取回溯栈帧。

我们可以在 Xcode 中按照以下流程来开启这个功能：

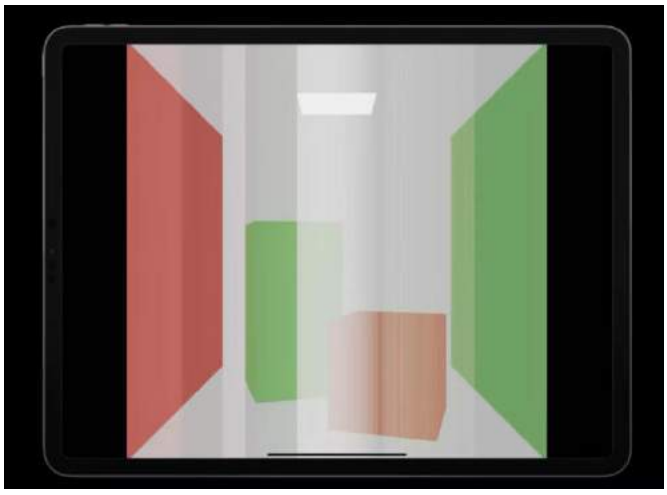
1. 开启 Metal 中的两个 Validation 选项：



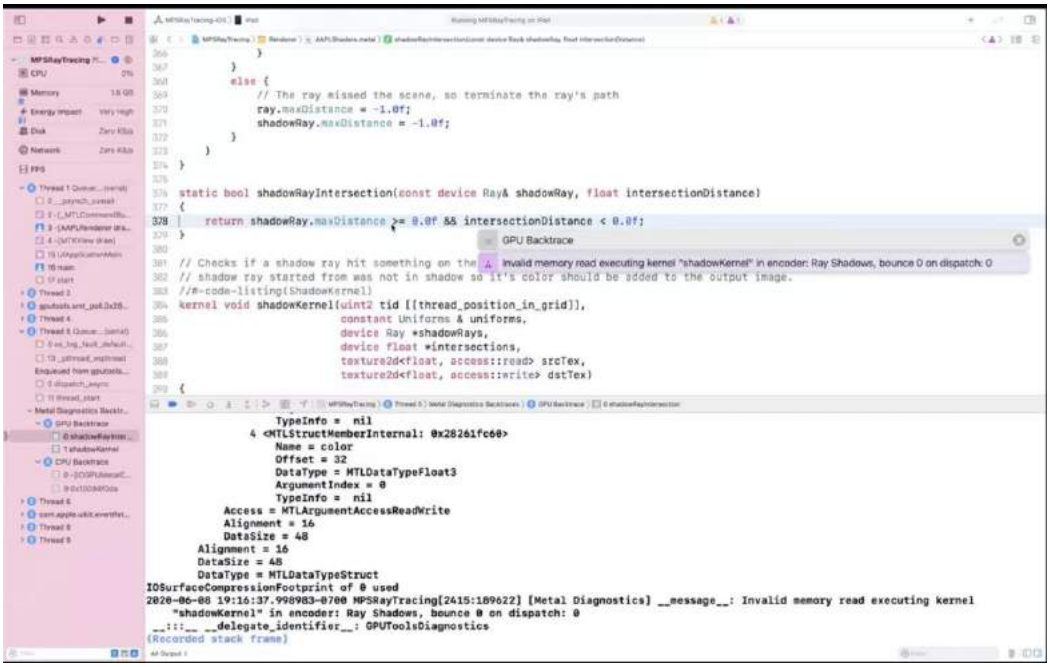
2. 开启 issue 自动断点开关并配置类型和分类等选项：



Video 中用了一个 demo 来展示整个工作流，具体参见 Debug GPU-side errors in Metal（11：25~14：45）大致流程如下图所示：



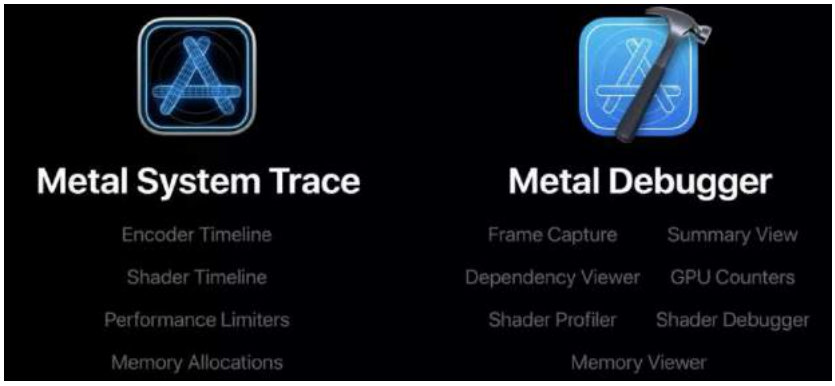
这是一个 Demo 应用程序，很明显它在渲染上出现了一些异常，但是因为是 GPU 侧的问题，所以开发者很难定位。但是通过上述的工作流开启 Shader Validation 之后。



Xcode 会自动断点到发生异常的地方，并展示出异常信息，这样就可以极大的提升开发者的错误修复效率。

Gain insights into your Metal app with Xcode 12

这一章主要讲的是 Xcode12 给 Metal App 提供了更多调试和分析的新工具。大致如下图所示：



主要分为两个部分：

Metal Debugger

这个工具可以让开发者在 App 运行时，获取到想分析和调试的任何一帧，然后再进入 Xcode 提供的各种分析界面，总体情况，依赖情况，内存，带宽，GPU，Shader 等各种具体的界面来对这一帧进行更加详细的分析和调试。整个过程使用视频的方式可能会更加高效，所以这里不会进行详细的赘述和分析。详情可以参见 Gain insights into your Metal app with Xcode 12

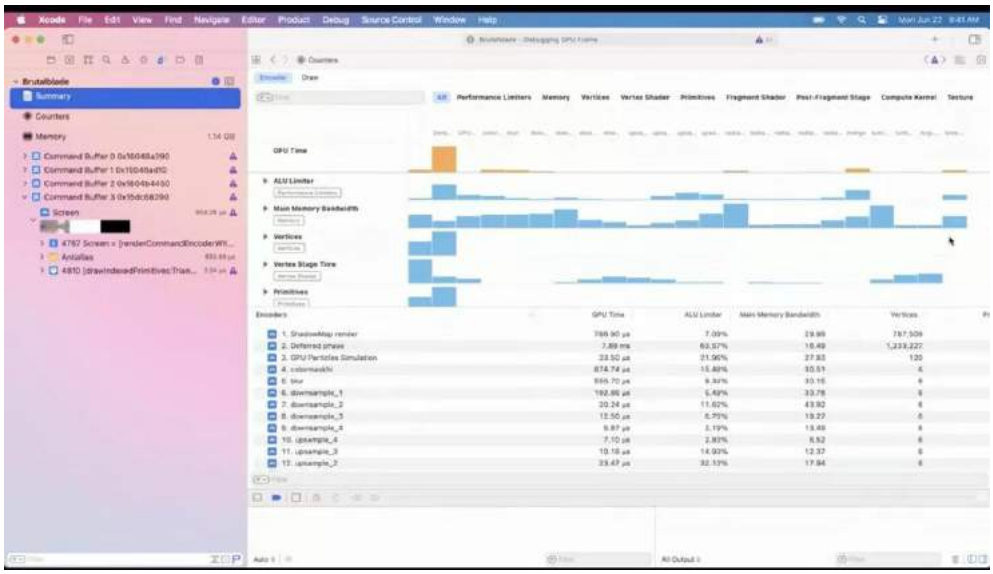
Metal System Trace

整个工具跟之前提到过的 Debugger 相比，他的功能主要是让开发者可以随着时间的推移来捕获应用程序的各种信息和特征，可以让开发者很好的调试一些例如终端，帧丢失，内存泄漏等问题。而 Debugger 主要是对某一帧进行调试和分析。

他提供了一个叫做编码时间线的工具，可以让开发者查看到 GPU 在应用运行中的运行各种命令缓冲的情况。然后提供了一个叫做着色器时间线的工具，可以让开发者查看到各种着色器在代码运行期间运行的过程。然后还有 GPU 计数器的工具，这个工具我们在前文进行了详细的分析，主要是用于解决 GPU 的绘制性能问题的工具。然后最后一个工具就是内存分配跟踪工具，可以让开发者查看到应用程序运行过程中各种内存的分配和释放，可以帮助开发者解决内存泄漏问题或者是降低应用内存占用。

技术启发与思考

WWDC 20 关于 Metal 的 Session 中，比较重要的就是官方提供了很多可供开发者进行 GPU 级别的调试工具以及性能分析工具。给比较成熟庞大而复杂的工程突破性能瓶颈，提供更加优秀的用户体验提供了一些思路。



闲鱼作为一个电商类 App，随着功能和增多和以及工程的复杂化，在所难免的会遇到性能瓶颈，而闲鱼团队当前面对挑战的方式是从工程级别来进行优化。从 Flutter 的角度来看，WWDC 20 对于 Metal 的调试工具和性能分析工具的完善，无疑提供了更多的优化思路。这为未来运行在 iOS 上的应用的调优和突破性能瓶颈带来了新的思路和可能性。

对于跨平台框架，Apple 有自家的 SwiftUI，这也是此次大会的重点项目。不过无论是 Flutter，还是 SwiftUI，大家最后对应用的性能瓶颈突破和优化一定是殊途同归的，也就是深入到 GPU 级别来进行开发和调试以及性能分析。对于未来的客户端开发人员，理解 GPU 和进行 GPU 级别的编程肯定是不可或缺的技能点之一。

参考：

https://developer.apple.com/videos/play/wwdc2020/10602/?spm=ata.13261165.0.0.33874c47d0rPKo

https://developer.apple.com/videos/play/wwdc2020/10605/?spm=ata.13261165.0.0.33874c47d0rPKo

WWDC 20 iOS14 隐私适配及部分解决方案

作者 | 盛兰雅（岚遥）
出品 | 淘系技术

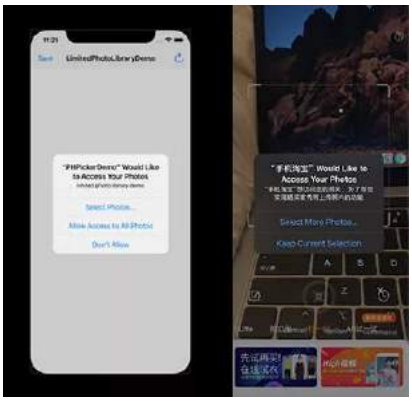
在刚刚结束的线上 WWDC 2020 发布会上苹果向我们展示了新的 iOS14 系统。iOS14 的适配，很重要的一环就集中在用户隐私和安全方面。

在 iOS13 及以前，当用户首次访问应用程序时，会被要求开放大量权限，比如相册、定位、联系人，实际上该应用可能仅仅需要一个选择图片功能，却被要求开放整个照片库的权限，这确实是不合理的。对于相册，在 iOS14 中引入了 “LimitedPhotos Library” 的概念，用户可以授予应用访问其一部分的照片，对于应用来说，仅能读取到用户选择让应用来读取的照片，让我们看到了 Apple 对于用户隐私的尊重。这仅仅是一部分，在 iOS14 中，可以看到诸多类似的保护用户隐私的措施，也需要我们升级适配。

适配点

相册

iOS14 新增了“Limited Photo Library Access” 模式，在授权弹窗中增加了 Select Photo 选项。用户可以在 App 请求调用相册时选择部分照片让 App 读取。从 App 的视角来看，你的相册里就只有这几张照片，App 无法得知其它照片的存在。



iOS14 中当用户选择：

“PHAuthorizationStatusLimited” 时，如果未进行适配，有可能会在每次触发相册功能时都进行弹窗询问用户是否需要修改照片权限。

对于这种情况可通过在 Info.plist 中设置：

“PHPhotoLibraryPreventAutomaticLimitedAccessAlert”的值为 YES 来阻止该弹窗反复弹出，并且可通过下面这个 API 来主动控制何时弹出 PHPickerViewController 进行照片选择。

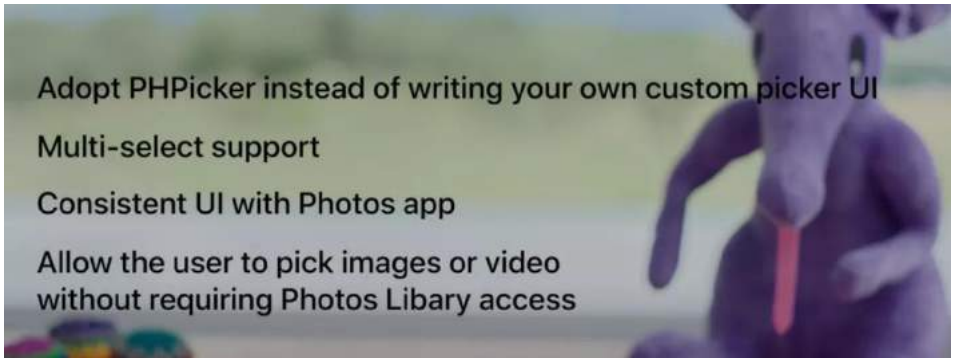
```
[[PHPhotoLibrary sharedPhotoLibrary] presentLimitedLibraryPickerFromViewController:self];
```

在 iOS14 中官方推荐使用 PHPicker 来替代原 API 进行图片选择。PHPicker 为独立进程，会在视图最顶层进行展示，应用内无法对其进行截图也无法直接访问到其内的数据。

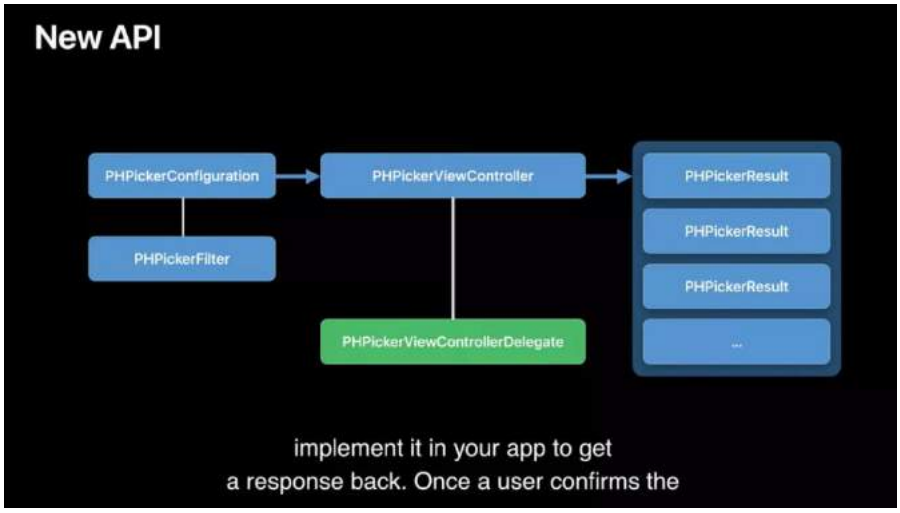
UIImagePickerController -> PHPickerViewController， UIImagePickerViewController 功能受限，每次只能选择一张图片，将逐渐被废弃。

```
typedef NS_ENUM(NSInteger, UIImagePickerControllerSourceType) {
    UIImagePickerControllerSourceTypePhotoLibrary API_DEPRECATED("Will be removed in a future release, use PHPicker.", ios(2, API_TO_BE_DEPRECATED)),
    UIImagePickerControllerSourceTypeCamera,
    UIImagePickerControllerSourceTypeSavedPhotosAlbum API_DEPRECATED("Will be removed in a future release, use PHPicker.", ios(2, API_TO_BE_DEPRECATED)),
} API_UNAVAILABLE(tvos);
```

PHPicker 支持多选，支持搜索，支持按 image，video，livePhotos 等进行选择。



新 API 及迁移 demo:




```
@interface ViewController () <PHPickerViewControllerDelegate>
  @property (weak, nonatomic) IBOutlet UIImageView *imageView;
@property (nonatomic, strong) NSArray<NSItemProvider *> *itemProviders;
  @end
  @implementation ViewController
  - (void)viewDidLoad {
        [super viewDidLoad];
        // Do any additional setup after loading the view.
}
  - (IBAction)button:(id)sender {
        // 以下 API 仅为 iOS14 only
        PHPickerConfiguration *configuration = [[PHPickerConfiguration alloc] init];
        configuration.filter = [PHPickerFilter videosFilter]; // 可配置查询用户相册中文件的类型，支持三
种
     configuration.selectionLimit = 0; // 默认为 1，为 0 时表示可多选。
        PHPickerViewController *picker = [[PHPickerViewController alloc] initWithConfiguration:confi
guration];

        picker.delegate = self;
        // picker vc，在选完图片后需要在回调中手动 dismiss
     [self presentViewController:picker animated:YES completion:^{
        }];
  }
  #pragma mark - Delegate
  - (void)picker:(PHPickerViewController *)picker didFinishPicking:(NSArray<PHPickerResult *> *)res
ults {
```

```
        [picker dismissViewControllerAnimated:YES completion:nil];
        if (!results || !results.count) {
            return;
        }
        NSItemProvider *itemProvider = results.firstObject.itemProvider;
        if ([itemProvider canLoadObjectOfClass:UIImage.class]) {
            __weak typeof(self) weakSelf = self;
            [itemProvider loadObjectOfClass:UIImage.class completionHandler:^(__kindof id<NSItem
ProviderReading>  _Nullable object, NSError * _Nullable error) {
                if ([object isKindOfClass:UIImage.class]) {
                    __strong typeof(self) strongSelf = weakSelf;
                    dispatch_async(dispatch_get_main_queue(), ^{
                        strongSelf.imageView.image = (UIImage *)object;
                    });
                }
            }];
        }
    }
```

需要注意的是，在 limit Photo 模式下，AssetsLibrary 访问相册会失败；在 writeOnly 模式下，AssetLibrary 也会有显示问题。建议还在使用 AssetsLibrary 的同学尽快迁移到新 API。

授权相关：旧 API 废弃，增加 PHAccessLevel 参数。如果再使用以前的 API 来获取权限状态，

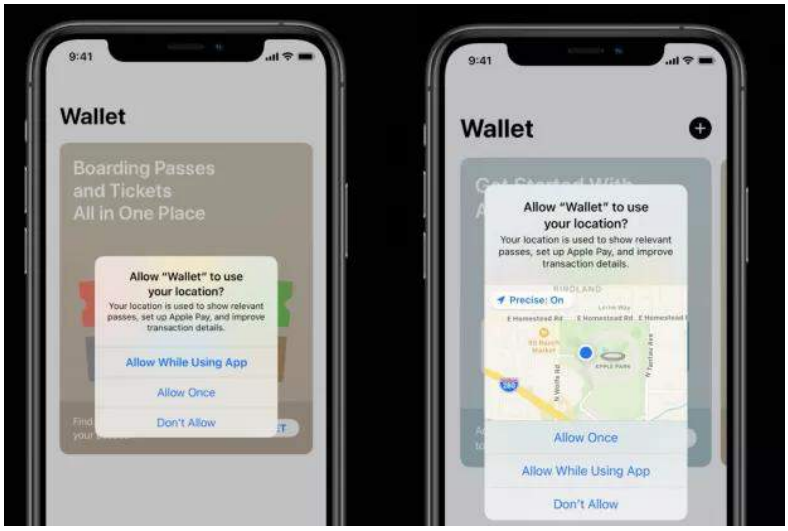
PHAuthorizationStatusLimited 状态下也会返回 PHAuthorizationStatusAuthorized

```
#pragma mark -
OS_EXPORT
@interface PHPhotoLibrary : NSObject

+ (PHPhotoLibrary *)sharedPhotoLibrary;

#pragma mark - Library access authorization status

/// Replaces \c +authorizationStatus to support add-only/read-write access level status
+ (PHAuthorizationStatus)authorizationStatusForAccessLevel:(PHAccessLevel)accessLevel API_AVAILABLE(macosx(10.16),
    ios(14), tvos(14));
+ (void)requestAuthorizationForAccessLevel:(PHAccessLevel)accessLevel handler:(void(^)(PHAuthorizationStatus
    status))handler API_AVAILABLE(macosx(10.16), ios(14), tvos(14));

/// Deprecated and replaced by authorizationStatusForAccessLevel:, will return \c PHAuthorizationStatusAuthorized if the user has chosen limited photo
    library access
+ (PHAuthorizationStatus)authorizationStatus API_DEPRECATED_WITH_REPLACEMENT("+authorizationStatusForAccessLevel:",
    ios(8, API_TO_BE_DEPRECATED), macos(10.13, API_TO_BE_DEPRECATED), tvos(10, API_TO_BE_DEPRECATED));
+ (void)requestAuthorization:(void(^)(PHAuthorizationStatus status))handler
    API_DEPRECATED_WITH_REPLACEMENT("+requestAuthorizationForAccessLevel:handler:", ios(8, API_TO_BE_DEPRECATED),
    macos(10.13, API_TO_BE_DEPRECATED), tvos(10, API_TO_BE_DEPRECATED));
```

```
typedef NS_ENUM(NSInteger, PHAccessLevel) {
    PHAccessLevelAddOnly = 1, // 仅允许添加照片
    PHAccessLevelReadWrite = 2, // 允许访问照片，limitedLevel 必须为 readWrite
} API_AVAILABLE(macos(10.16), ios(14), tvos(14));
// 查询权限
PHAccessLevel level = PHAccessLevelReadWrite;
PHAuthorizationStatus status = [PHPhotoLibrary authorizationStatusForAccessLevel:level];
    switch (status) {
        case PHAuthorizationStatusLimited:
            NSLog(@"limited");
            break;
        case PHAuthorizationStatusDenied:
iOS14 隐私适配及部分解决方案 <          30

            NSLog(@"denied");
            break;
        case PHAuthorizationStatusAuthorized:
            NSLog(@"authorized");
            break;
        default:
            break;
}
// 请求权限，需注意 limited 权限尽在 accessLevel 为 readAndWrite 时生效
[PHPhotoLibrary requestAuthorizationForAccessLevel:level handler:^(PHAuthorizationStatus status) {
    switch (status) {
        case PHAuthorizationStatusLimited:
            NSLog(@"limited");
            break;
        case PHAuthorizationStatusDenied:
            NSLog(@"denied");
            break;
        case PHAuthorizationStatusAuthorized:
            NSLog(@"authorized");
            break;
        default:
            break;
    }
}];
```

定位

在 iOS13 及以前，App 请求用户定位授权时为如下形态：一旦用户同意应用获取定位信息，当前应用就可以获取到用户的精确定位。

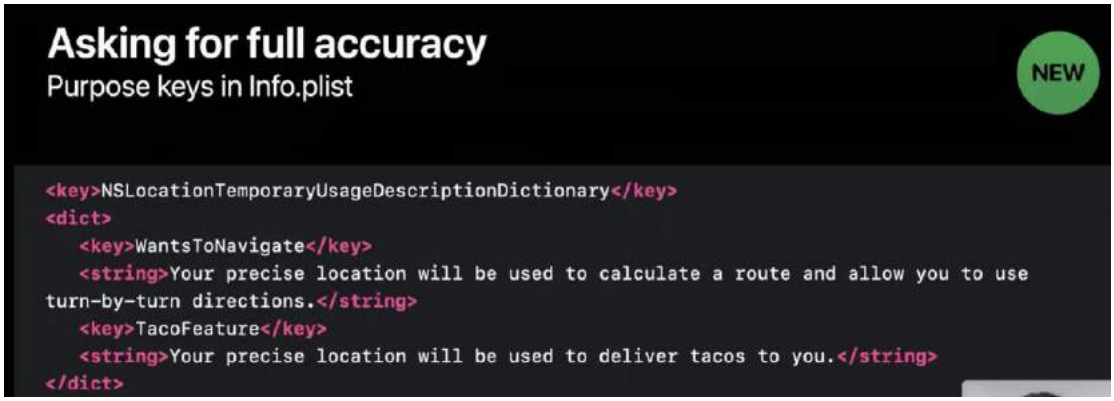


iOS14 新增用户大致位置选项可供用户选择，原因是大多数 App 实际上并不需要获取用户到用户最准确的定位信息。iOS14 授权弹窗新增的 Precise 的开关默认会选中精确位置。用户通过这个开关可以进行更改，当把这个值设为 On 时，地图上会显示精确位置；切换为 Off 时，将显示用户的大致位置。

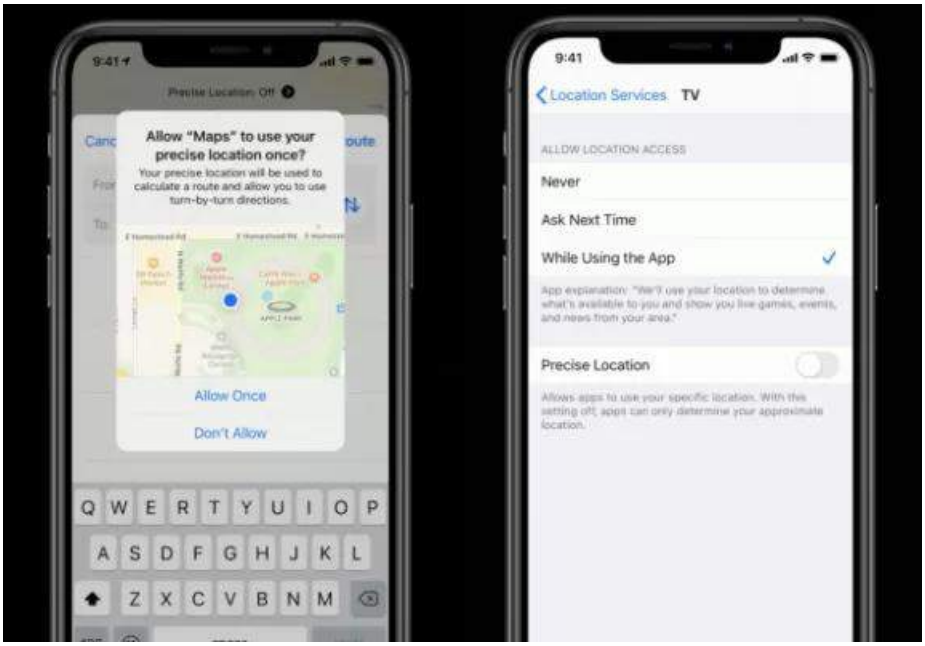
对于对用户位置敏感度不高的 App 来说，这个似乎无影响，但是对于强依赖精确位置的 App 适配工作就显得非常重要了。可以通过用户在“隐私设置”中设置来开启精确定位，但是可能用户宁可放弃使用这个应用也不愿意开启。这个时候，iOS14 在 CLLocationManager 新增两个方法可用于向用户申请临时开启一次精确位置权限。

```
 *      callbacks occur.  That is, it will be called on the runloop where the
 *      CLLocationManager was originally initialized.
 */
- (void)requestTemporaryFullAccuracyAuthorizationWithPurposeKey:(NSString *)purposeKey completion:(void(^ _Nullable)(NSError *
    _Nullable))completion API_AVAILABLE(ios(14.0), macos(10.16), watchos(7.0), tvos(14.0));

/*
 *  requestTemporaryFullAccuracyAuthorizationWithPurposeKey:
 *
 *  Discussion:
 *      This is a variant of requestTemporaryAccurateLocationAuthorizationWithPurposeKey:completion:
 *      which doesn't take a completion block.  This is equivalent to passing in a nil
 *      completion block.
 */
- (void)requestTemporaryFullAccuracyAuthorizationWithPurposeKey:(NSString *)purposeKey API_AVAILABLE(ios(14.0), macos(10.16), watchos(7.0),
    tvos(14.0));
```

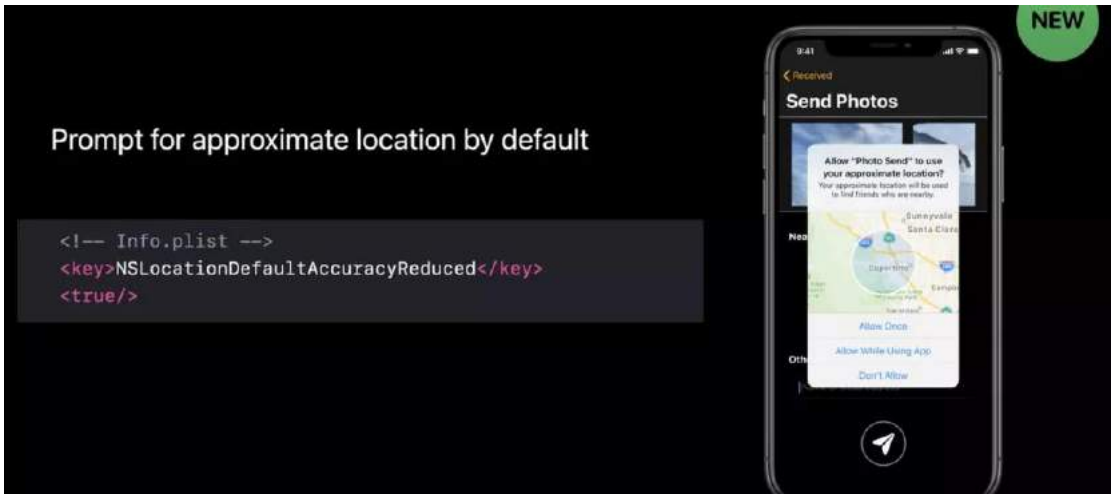
使用方式也很简单，需要首先在 Info.plist 中配置"NSLocationTemporaryUsageDescriptionDictionary"字典中需要配置 key 和 value 表明使用位置的原因，以及具体的描述。



在本例中，key 即为获取用户权限时传的 "purposeKey"，最终呈现给用户的就是左图，右图为当 App 主动关闭精确定位权限申请。

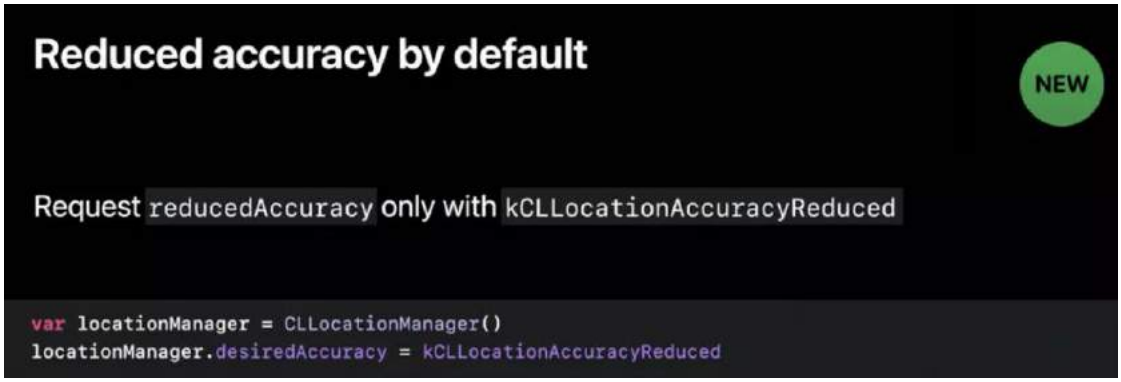


对于地理位置不敏感的 App 来说，iOS14 也可以通过直接在 info.plist 中添加 NSLocationDefaultAccuracyReduced 为 true 默认请求大概位置。



这样设置之后，即使用户想要为该 App 开启精确定位权限，也无法开启。

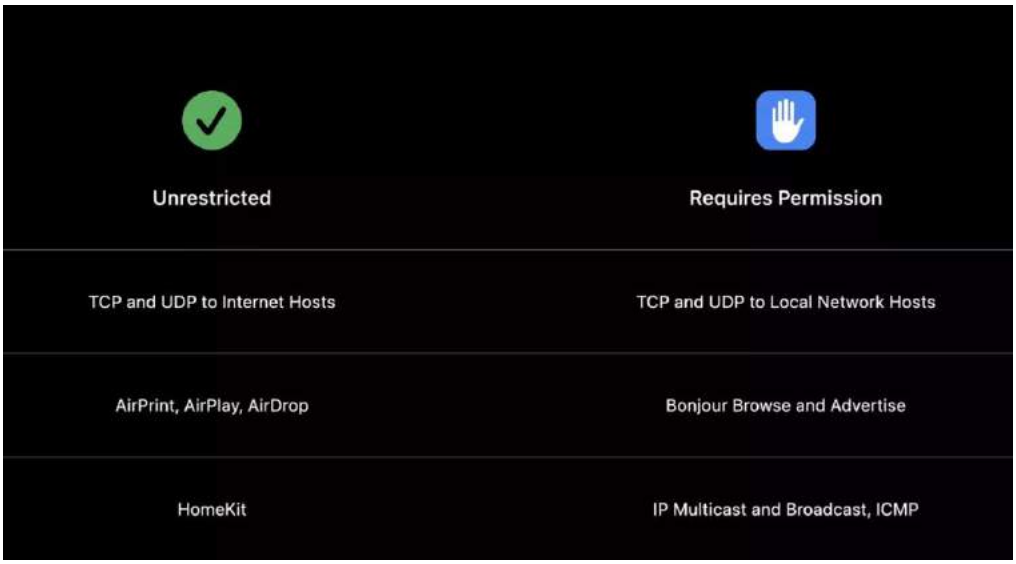
也可以直接通过 API 来根据不同的需求设置不同的定位精确度。



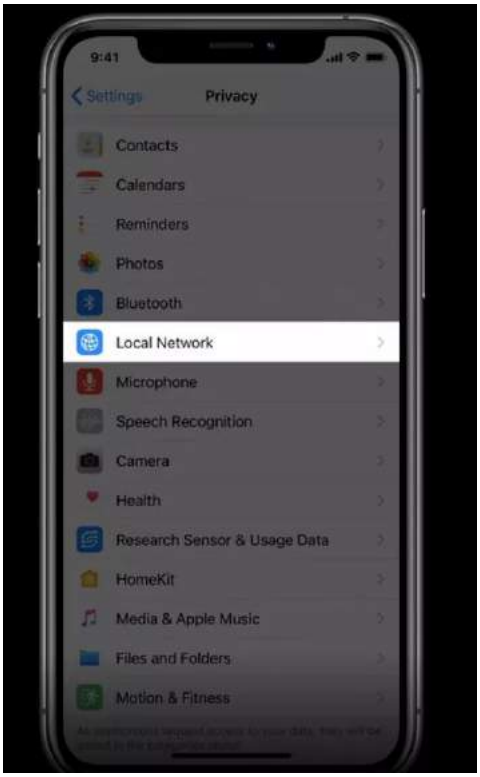
需要注意的是，当 App 在 Background 模式下，如果并未获得精确位置授权，那么 Beacon 及其他位置敏感功能都将受到限制。

Local Network

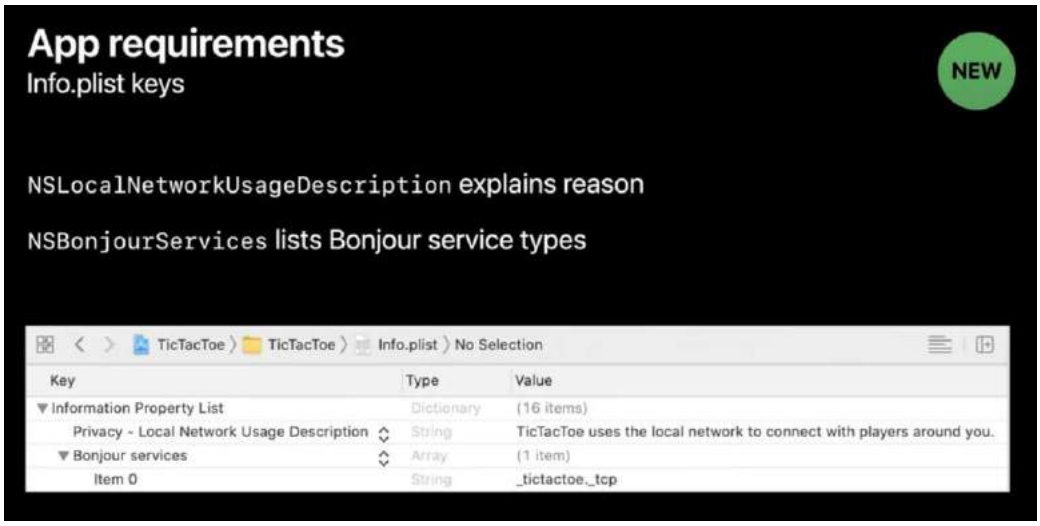
iOS14 当 App 要使用 Bonjour 服务时或者访问本地局域网，使用 mDNS 服务等，都需要授权，开发者需要在 Info.plist 中详细描述使用的为哪种服务以及用途。下图为需要无需申请权限与需要授权的服务：



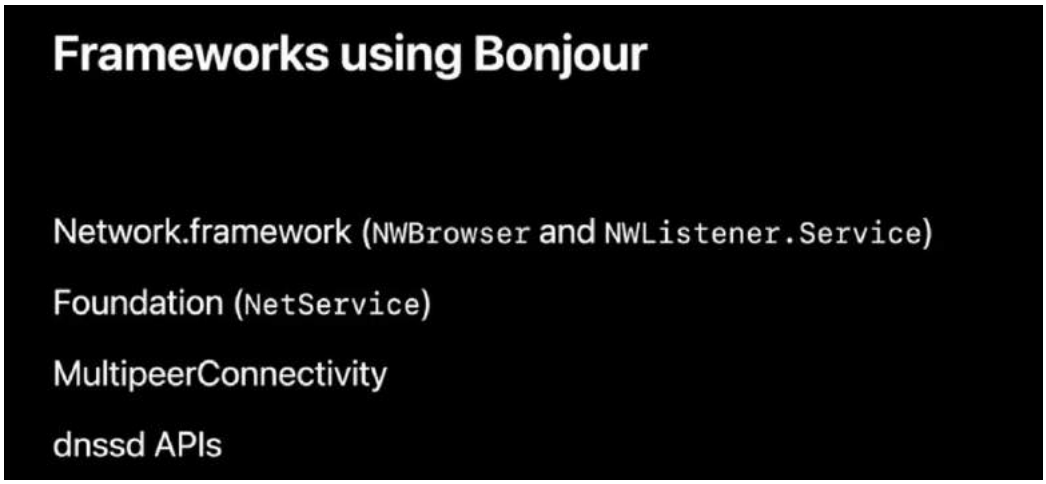
在 "隐私设置" 中也可以查看和修改具体有哪些 App 正在使用 LocalNetwork：



如果应用中需要使用 LocalNetwork 需要在 Info.plist 中配置两个选项，详细描述为什么需要使用该权限，以及需要列出具体使用 LocalNetwork 的服务列表。



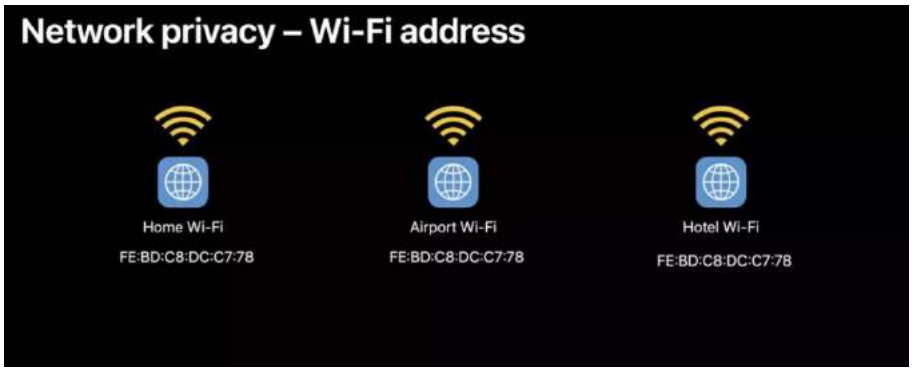
对于使用了下列包含 Bonjour 的 framework，都需要更新描述.:



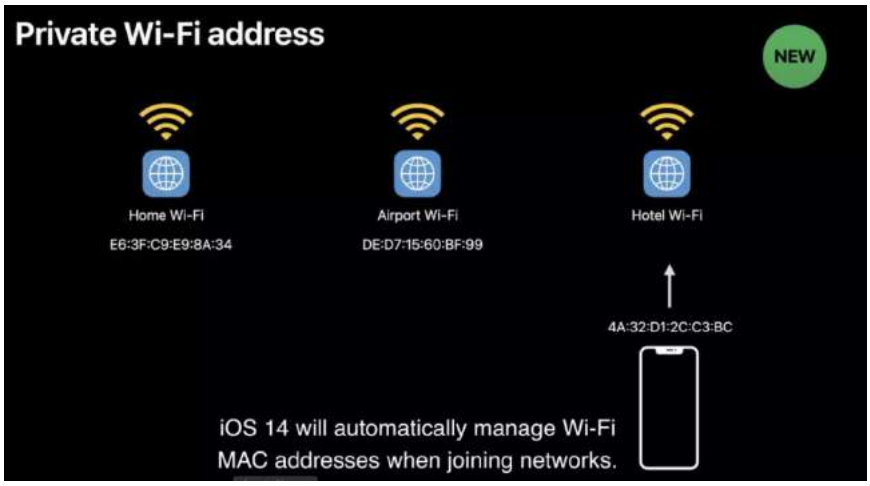
Wi-Fi Address

iOS8 - iOS13 ，用户在不同的网络间切换和接入时，mac 地址都不会改变，这也就使得网络运营商还是可以通过 mac 地址对用户进行匹配和用户信息收集，生成完整的用户信息。iOS14 提供 Wifi 加密服务，每次接入不同的 WiFi 使用的 mac 地址都不同。每过 24 小时，mac 地址还会更新一次。需要关注是否有使用用户网络 mac 地址的服务。

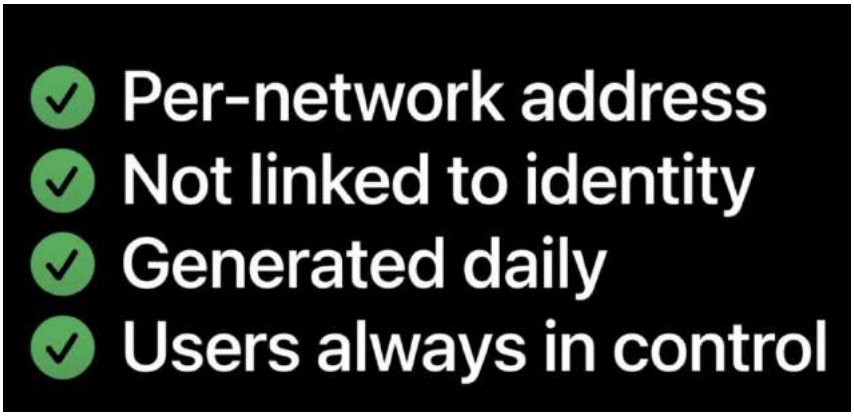
下图为 iOS13 及之前用户接入网络时 mac 地址并不会进行改变:



下图为 iOS14 用户接入 Wi-Fi 时 mac 地址的变化情况：



并且用户也可以自行选择是否开启 private Wi-Fi address：



剪切板

在 iOS14 中，读取用户剪切板的数据会弹出提示。

弹出提示的原因是使用 UIPasteboard 访问用户数据，访问以下数据都会弹出 toast 提示。

```
@property(nullable,nonatomic,copy) NSString *string API_UNAVAILABLE(tvos) API_UNAVAILABLE(watchos);
@property(nullable,nonatomic,copy) NSArray<NSString *> *strings API_UNAVAILABLE(tvos) API_UNAVAILABLE(watchos);

@property(nullable,nonatomic,copy) NSURL *URL API_UNAVAILABLE(tvos) API_UNAVAILABLE(watchos);
@property(nullable,nonatomic,copy) NSArray<NSURL *> *URLs API_UNAVAILABLE(tvos) API_UNAVAILABLE(watchos);

@property(nullable,nonatomic,copy) UIImage *image API_UNAVAILABLE(tvos) API_UNAVAILABLE(watchos);
@property(nullable,nonatomic,copy) NSArray<UIImage *> *images API_UNAVAILABLE(tvos) API_UNAVAILABLE(watchos);

@property(nullable,nonatomic,copy) UIColor *color API_UNAVAILABLE(tvos) API_UNAVAILABLE(watchos);
@property(nullable,nonatomic,copy) NSArray<UIColor *> *colors API_UNAVAILABLE(tvos) API_UNAVAILABLE(watchos);

// Queries
```

兼容方案：如果应用访问剪切板仅仅用于判断是否为 URL 格式，则 iOS14 新增了两个 API 可以用于规避该提示。如果应用想直接访问剪切板的数据，暂时可能无法做到规避该提示。iOS14 新增两种 UIPasteboardDetectionPattern。

```
typedef NSString * UIPasteboardDetectionPattern NS_TYPED_ENUM API_AVAILABLE(ios(14.0));

/// NSString value, suitable for implementing "Paste and Go"
UIKIT_EXTERN UIPasteboardDetectionPattern const UIPasteboardDetectionPatternProbableWebURL API_AVAILABLE(ios(14.0));

/// NSString value, suitable for implementing "Paste and Search"
UIKIT_EXTERN UIPasteboardDetectionPattern const UIPasteboardDetectionPatternProbableWebSearch API_AVAILABLE(ios(14.0));
```

上面的两个 API 可用于规避提示，但只能用于判断剪切板中是否有 URL，并不是真正的访问剪贴板数据，也拿不到剪切板的真实数据。下面两个 API 可以获得具体的 URL 信息，但是会触发剪切板提示。并且实测当用户剪切板中包含多个 URL 时只会返回第一个。

```
// Detection

/// Detects patterns in the first pasteboard item.
///
/// @param patterns Detect only these patterns.
/// @param completionHandler Receives which patterns were detected, or an error.
- (void)detectPatternsForPatterns:(NSSet<UIPasteboardDetectionPattern> *)patterns
                completionHandler:(void(^)(NSSet<UIPasteboardDetectionPattern> * _Nullable,
                                           NSError * _Nullable))completionHandler NS_REFINED_FOR_SWIFT API_AVAILABLE(ios(14.0));

/// Detects patterns in the specified pasteboard items.
///
/// @param patterns Detect only these patterns.
/// @param itemSet Specifies which pasteboard items by their position. Nil means all items.
/// @param completionHandler Receives which patterns were detected per item specified,
///                          or an error.
- (void)detectPatternsForPatterns:(NSSet<UIPasteboardDetectionPattern> *)patterns
                      inItemSet:(NSIndexSet * _Nullable)itemSet
                completionHandler:(void(^)(NSArray<NSSet<UIPasteboardDetectionPattern> *> * _Nullable,
                                           NSError * _Nullable))completionHandler NS_REFINED_FOR_SWIFT API_AVAILABLE(ios(14.0));

/// Detects patterns and corresponding values in the first pasteboard item.
///
/// @param patterns Detect only these patterns.
/// @param completionHandler Receives which patterns and values were detected, or an error.
- (void)detectValuesForPatterns:(NSSet<UIPasteboardDetectionPattern> *)patterns
              completionHandler:(void(^)(NSDictionary<UIPasteboardDetectionPattern, id> * _Nullable,
                                         NSError * _Nullable))completionHandler NS_REFINED_FOR_SWIFT API_AVAILABLE(ios(14.0));

/// Detects patterns and corresponding values in the specified pasteboard items.
///
/// @param patterns Detect only these patterns.
/// @param itemSet Specifies which pasteboard items by their position. Nil means all items.
/// @param completionHandler Receives which patterns and values were detected per item specified,
///                          or an error.
- (void)detectValuesForPatterns:(NSSet<UIPasteboardDetectionPattern> *)patterns
                    inItemSet:(NSIndexSet * _Nullable)itemSet
              completionHandler:(void(^)(NSArray<NSDictionary<UIPasteboardDetectionPattern, id> *> * _Nullable,
                                         NSError * _Nullable))completionHandler NS_REFINED_FOR_SWIFT API_AVAILABLE(ios(14.0));
```

使用示例

```
NSSet *patterns = [[NSSet alloc] initWithObjects:UIPasteboardDetectionPatternProbableWebURL, ni
l];
[[UIPasteboard generalPasteboard] detectPatternsForPatterns:patterns completionHandler:^(NSSet<
UIPasteboardDetectionPattern> * _Nullable result, NSError * _Nullable error) {
    if (result && result.count) {
            // 当前剪切板中存在 URL
    }
}];
```

相机和麦克风

iOS14 中 App 使用相机和麦克风时会有图标提示以及绿点和黄点提示，并且会显示当前是哪个 App 在使用此功能。我们无法控制是否显示该提示。

会触发录音小黄点的代码示例：

```
AVAudioRecorder *recorder = [[AVAudioRecorder alloc] initWithURL:recorderPath settings:nil error:nil];
[recorder record];
```

触发相机小绿点的代码示例:

```
AVCaptureDeviceInput *videoInput = [[AVCaptureDeviceInput alloc] initWithDevice:videoCaptureDevice error:nil];
AVCaptureSession *session = [[AVCaptureSession alloc] init];
if ([session canAddInput:videoInput]) {
    [session addInput:videoInput];
}
[session startRunning];
```

IDFA

IDFA 全称为 Identity for Advertisers ，即广告标识符。用来标记用户，目前最广泛的用途是用于投放广告、个性化推荐等。

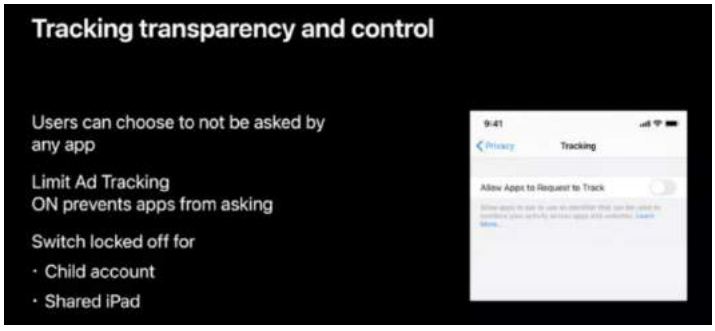
在 iOS13 及以前，系统会默认为用户开启允许追踪设置，我们可以简单的通过代码来获取到用户的 IDFA 标识符。

```
if ([[ASIdentifierManager sharedManager] isAdvertisingTrackingEnabled]) {
    NSString *idfaString = [[ASIdentifierManager sharedManager] advertisingIdentifier].UUIDString;
    NSLog(@"%@", idfaString);
}
```

但是在 iOS14 中，这个判断用户是否允许被追踪的方法已经废弃。

```
#import <Foundation/Foundation.h>

NS_ASSUME_NONNULL_BEGIN

API_AVAILABLE(ios(6), macosx(10.14), tvos(6))
@interface ASIdentifierManager : NSObject

+ (ASIdentifierManager *)sharedManager;

@property (nonatomic, readonly) NSUUID *advertisingIdentifier;
@property (nonatomic, readonly, getter=isAdvertisingTrackingEnabled) BOOL advertisingTrackingEnabled
    API_DEPRECATED("This has been replaced by functionality in AppTrackingTransparency's
    ATTrackingManager class.", ios(6, 14), macosx(10.14, 10.16), tvos(6, 14));

- (void)clearAdvertisingIdentifier API_UNAVAILABLE(ios, macos, tvos);

@end

NS_ASSUME_NONNULL_END
```

iOS14 中，系统会默认为用户关闭广告追踪权限。



对于这种情况，我们需要去请求用户权限。首先需要在 Info.plist 中配置" NSUserTracking UsageDescription " 及描述文案，接着使用 AppTrackingTransparency 框架中的

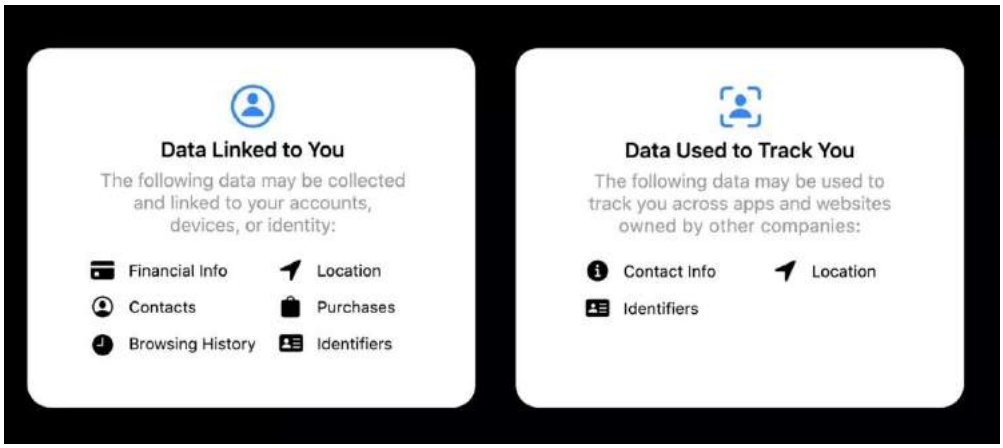
ATTrackingManager 中的 requestTrackingAuthorizationWithCompletionHandler 请求用户权限，在用户授权后再去访问 IDFA 才能够获取到正确信息。

```
#import <AppTrackingTransparency/AppTrackingTransparency.h>
#import <AdSupport/AdSupport.h>
- (void)testIDFA {
    if (@available(iOS 14, *)) {
        [ATTrackingManager requestTrackingAuthorizationWithCompletionHandler:^(ATTrackingMan
agerAuthorizationStatus status) {
            if (status == ATTrackingManagerAuthorizationStatusAuthorized) {
                NSString *idfaString = [[ASIdentifierManager sharedManager] advertisingIdentifie
r].UUIDString;
            }
        }];
    } else {
        // 使用原方式访问 IDFA

    }
}
```

对于用户拒绝授权 UserTracking 的情况，可以考虑接入苹果的 SKAdNetwork 框架进行广告分析。感兴趣的同学可进一步了解：https://developer.apple.com/documentation/storekit/skadnetwork

上传 AppStore

更加严格的隐私审核，可以让用户在下载 App 之前就知道此 App 将会需要哪些权限。目前苹果商店要求所有应用在上架时都必须提供一份隐私政策。如果引入了第三方收集用户信息等 SDK，都需要向苹果说明是这些信息的用途。



总结

对于这次 iOS14 的隐私权限大升级和新尝试，体现了苹果对于用户隐私的尊重。

从用户角度来说，近年来越来越精准的广告投放让我们越来越感觉自己被"监视"着，此次升级后，我们有了更多保护自己隐私的方式以及避免广告骚扰的方法，苹果此举无疑会加大我们对其的好感度和信任感。但从另一个角度来说，对于 IDFA 的限制，可能会导致之前许多依靠广告投放收入的免费 App 难以继续维持生计，也可能也会导致免费 App 的数量有所降低。从开发者的角度来说，除了对 iOS14 隐私升级的积极适配外，也让我们感受到了 iOS14 中对于用户隐私的重视无疑会提高获取用户行为信息的成本。

冲击最大的应该就是广告行业，对于目前的推荐算法和用户拉新都会受到影响，如何在充分尊重用户隐私的前提下进行广告的精准投放对于开发者和广告商来说都是一个不小的机遇和挑战。

参考：

WWDC 2020 Apple Developer

Developer Documentation

WWDC 20 详解 SwiftUI 的重大改变及核心优势

作者 | 姜沂（倾寒）
出品 | 淘系技术

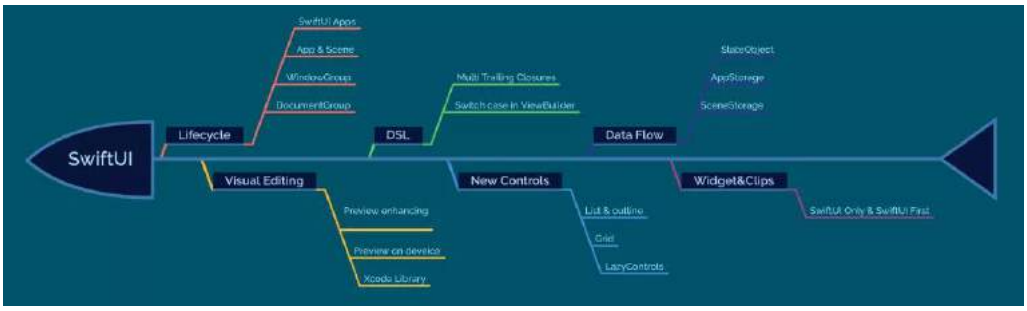
6 月 23 日凌晨 1 点，苹果 WWDC20 开发者大会在线上以主题演讲的方式，在 Apple Park 进行直播。
23-26 日，苹果公开了 100 多个面向开发者的视频，内容涵盖 Swift / SwiftUI 、App Clips、Widgets、Privacy & Security 等等方面。
对于开发者和程序员来说，我们有哪些新发现和新思考？
淘系技术客户端团队，将给大家带来一系列关于新系统背后的启发，欢迎交流讨论。

前言

SwiftUI 是苹果公司于 2019 年推出的 Apple Platform 的新一代声明式布局引擎，笔者于去年第一时间升级 Beta 尝鲜全家庭，并在短时间内迅速落地了基于 SwiftUI 的内部 APP，也分享了几篇关于 SwiftUI 的文章， 但 SwiftUI 1.0 基本没有任何公司敢用在正式上线的主 APP 上，API 在 Beta 版本之间各种废弃，UI 样式经常不兼容，大列表性能差，彼时都标识着 SwiftUI 还称为一个 Toy Framewrok。

随着 WWDC 20 相关新特性和介绍视频的释出，都明确的宣告着 SwiftUI 元年已经到了，SwiftUI 已经成长为新时代的布局引擎。

以下从几个方面分享关于 SwiftUI 的重大改变及核心优势。



PS: 需要读者对 Swift 及 SwiftUI 1.0 有一定熟悉。

SwiftUI Apps

苹果在最近几年的动作中一直在搞 Apple Platform 统一的事情，从最近几年的 iPad 多任务 多窗口，到 Mac Catalyst 再到今年更进一步直接推出了 Apple silicon 芯片更是从硬件上做到了真正统一(话外音:你们在软件上玩的那些跨平台的都是小玩意,硬件才是王道)。

还提供了 Rosetta2 Universal2 帮助开发者基本无成本的迁移到新平台上。但是作为软件工程师还是要更多的关注软件生态的变化。首先了解下创建 APP 时的变化。

可以看到创建新工程时有了一套全新的模板基于 SwiftUI App Lifecycle 的跨平台项目。

代码也从原本的基于 UI/NS HostViewController 变成了基于 APP 的声明式描述，下面是代码的前后对比。

Before

```
class SceneDelegate: UIResponder, UIWindowSceneDelegate {
    var window: UIWindow?
    func scene(_ scene: UIScene, willConnectTo session: UISceneSession, options connectionOptio
ns: UIScene.ConnectionOptions) {
        let contentView = ContentView()
        if let windowScene = scene as? UIWindowScene {
            let window = UIWindow(windowScene: windowScene)
            window.rootViewController = UIHostingController(rootView: contentView)
```

```
            self.window = window
            window.makeKeyAndVisible()
        }
    }
}
```

After

```
import SwiftUI
@main
struct MyApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```

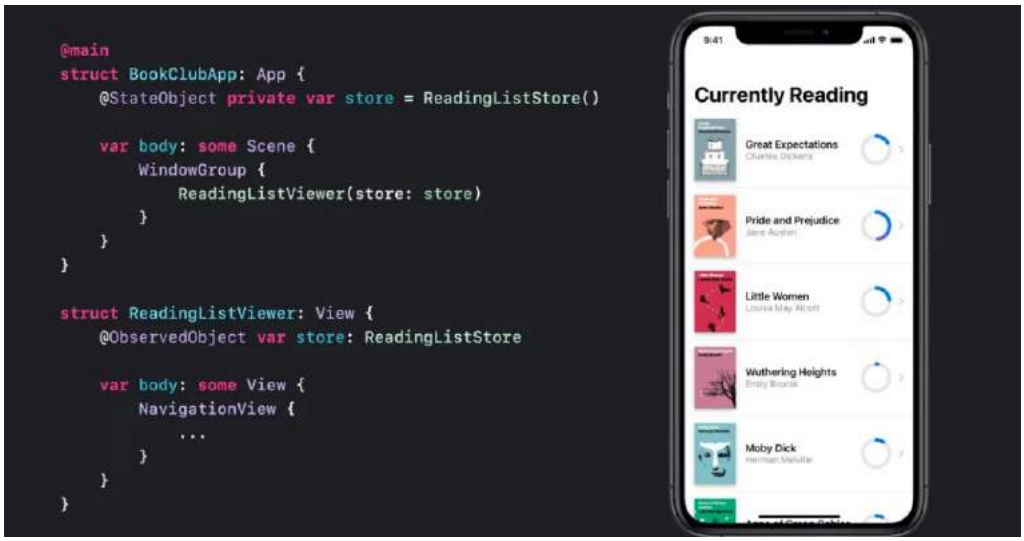
其中@main 是 Swift 5.1 新增的 Attribute 标记了应用程序的入口点，更多请参看 SE-0281-main-attribute.md

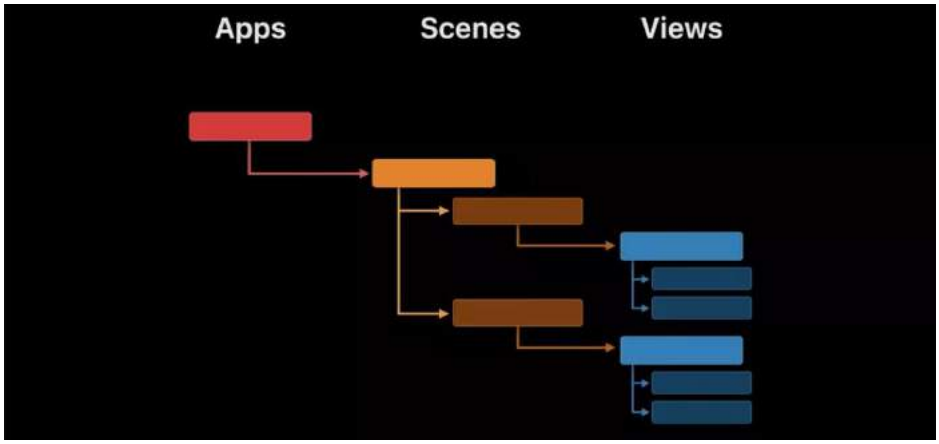
乍看好像只有代码精简了不少，很多人会认为这个简洁程度还不如 Flutter 的 main() => runApp(MyApp());.

但最重要的变化是这是第一次跨平台代码，完全无需引入任何 UIKit APPKit WatckKit 等相关 Framewok，即可直接运行在不同平台上。这意味着我们后续在 UI 布局系统上可以逐渐摆脱对传统命令式 UI 编程的依赖。达到真正的平台无关。

SwiftUI 将整个原有的平台差异部分抽象为 App 和 Scene，对于一个 mac/iOS/iPad/watch/tv/..应用，来说 App 代表了整个应用，Scene 代表了与 Window 相关的多窗口，有些设备只有一个 Scene 有些则有多个，虽然不同的 OS 确实存在差异，但是在语义层面达到了一致。

其次一个没有历史包袱的 APP，也可以完整的从 Swift APP lifecycle 风格式的模板开始，无需再和传统的 UIKit/APPKit 等混合。这也意味着可以达到 APP 完全 Declared and State-Driven。





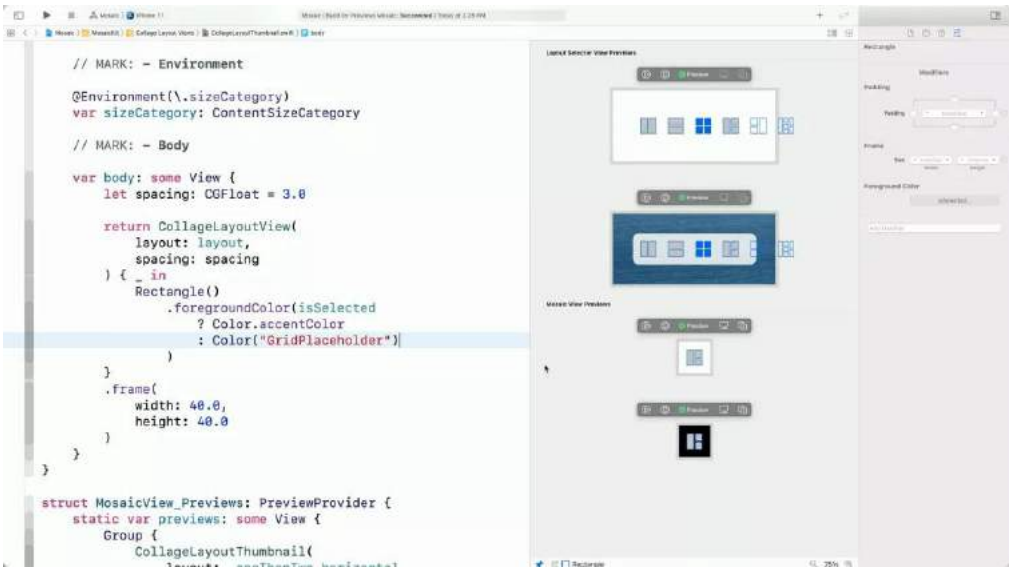
Viusal Editing

Preview

在传统的利用 DSL 可视编程框架或者平台，诸如 Web Flutter 等技术，都是开发者编写好对应的代码，运行在对应的平台或者调试工具上。SwiftUI 作为苹果最重要的软件层战略框架，更是和 Xcode 深度结合，在运行之前就可以完整的预览你所编写的界面。

强大的 Preview 可以让你既可以从编写 DSL 到立即预览效果，也可从预览的 Canvas 画布中直接修改效果在代码编辑器中生成代码，这对于日常开发的效率有非常大的提高，尤其是在 UI 微调时，效果尤为突出。

Xcode12 可以在 Canvas 上同时预览多个不同设备环境的界面,也可以直接投射到真实的设备上来预览。



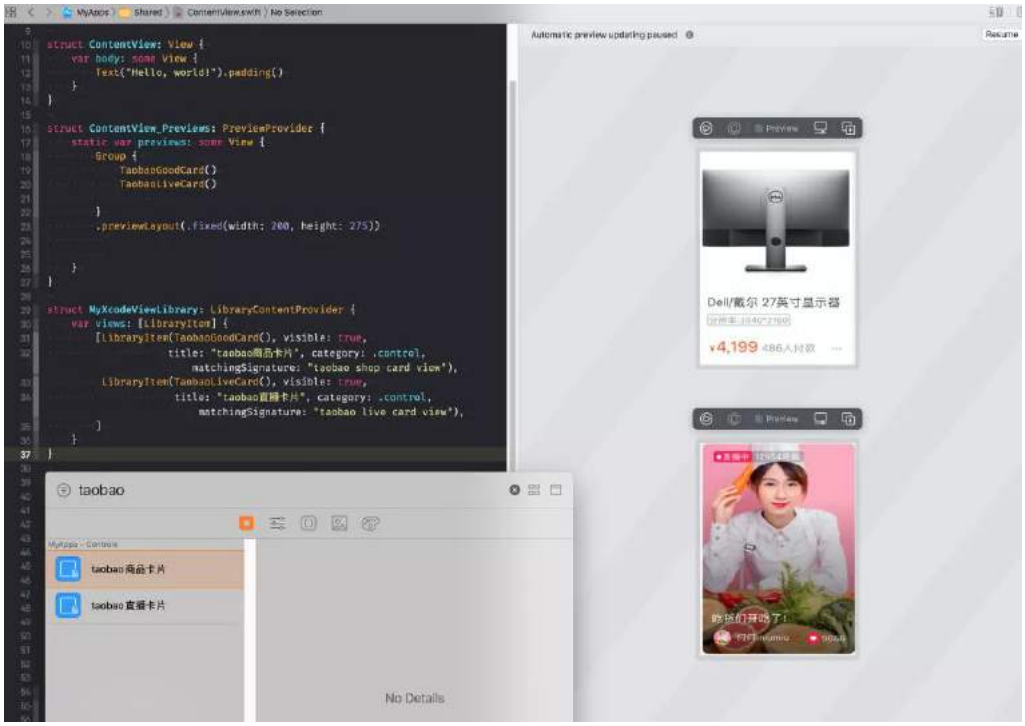
对于日常开发来说，编写一个 UI 界面通常依赖外部的网络/磁盘/其他数据，才能正常的构建，这也造成了 UI 开发虽然是开发中较为简单的一步，但同时也是最耗时的一步，有了预览功能，可以把很多繁琐的工作前置解决掉，对于研发效率会有非常大的提高。

Xcode Library

在编写真实项目中，一个公司的 APP UI 包含成百上千种风格的 View 组件，对于 UI 组件丰富的产品,如果一个新需求可以由现有的组件组合,那么需求交付的时间也会大大缩短。

但是对于一个大型的开发团队而言，一个开发同学是很难知道公司内到底有多少种组件库，而且即便知道有某种组件库，开发同学初期看到的也是代码，一般需要书写一定的 Demo 才可以用眼睛感知到这个组件到底是否是我想要的。

在 Xcode 12 中提供了更强大的工具，一个自定义组件，只需要遵守一个 LiberyContent Provider 协议就可被 Xcode 识别，可以像系统控件一样直接从 Xcode 里面识别并预览。对于一个大型团队来说，此功能可以大大提高找寻组件和查看组件样式的效率。




```
// Without trailing closure:
        UIView.animate(withDuration: 0.3, animations: {
          self.view.alpha = 0
        }, completion: { _ in
          self.view.removeFromSuperview()
        })
        // With trailing closure
        UIView.animate(withDuration: 0.3, animations: {
          self.view.alpha = 0
        }) { _ in
          self.view.removeFromSuperview()
        }
        // Multiple trailing closure arguments
        UIView.animate(withDuration: 0.3) {
          self.view.alpha = 0
        } completion: { _ in
          self.view.removeFromSuperview()
        }
```

DSL

随着 Swift5.3 和 SwiftUI2.0 的推出，SwiftUI 在 DSL 上也更富有表现力， Swift 支持了多重尾闭包语法和在 ViewBuilde 里面支持 Switch Case 语句。

Multiple Trailing Closures

虽然社区对多重尾闭包的讨论上一直存在争议问题，但最终 Swift5.3 还是接受并实现了，在普通命令式编程的地方使用会有一定的困惑性，但是在 SwiftUI 中 DSL 也更有声明式的味道。

```
// Without trailing closure:
        UIView.animate(withDuration: 0.3, animations: {
          self.view.alpha = 0
        }, completion: { _ in
          self.view.removeFromSuperview()
        })
        // With trailing closure
        UIView.animate(withDuration: 0.3, animations: {
          self.view.alpha = 0
        }) { _ in
          self.view.removeFromSuperview()
        }
        // Multiple trailing closure arguments
        UIView.animate(withDuration: 0.3) {
          self.view.alpha = 0
        } completion: { _ in
          self.view.removeFromSuperview()
        }
```

Switch Case Support

在 SwiftUI 的 ViewBuilder DSL 体系中也支持了 Switch case 语法。

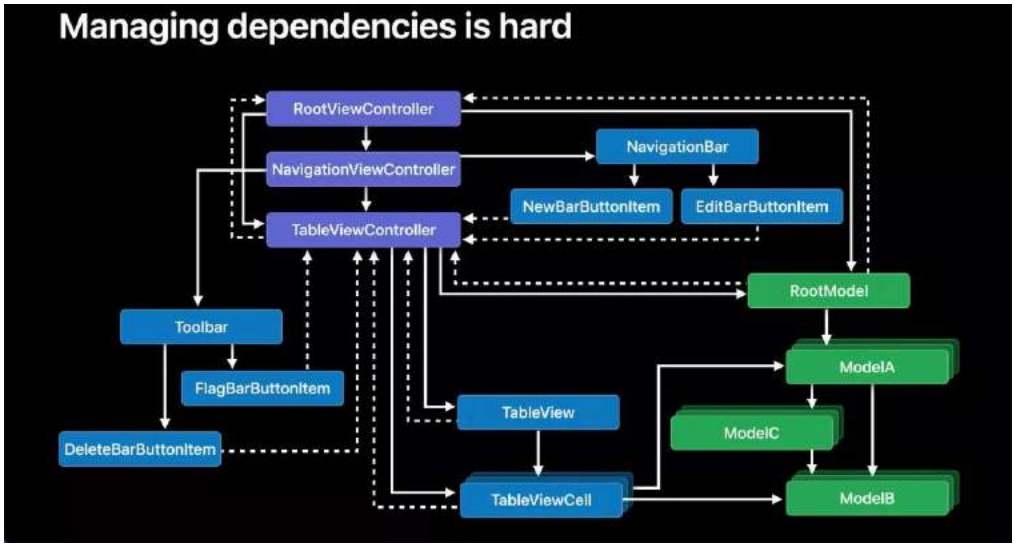
```
var body: some View {
            switch c {
            case .a:
                return Text("A")
            case .b:
                return Text("B")
            case .c:
                return Text("C")

            }
        }
```

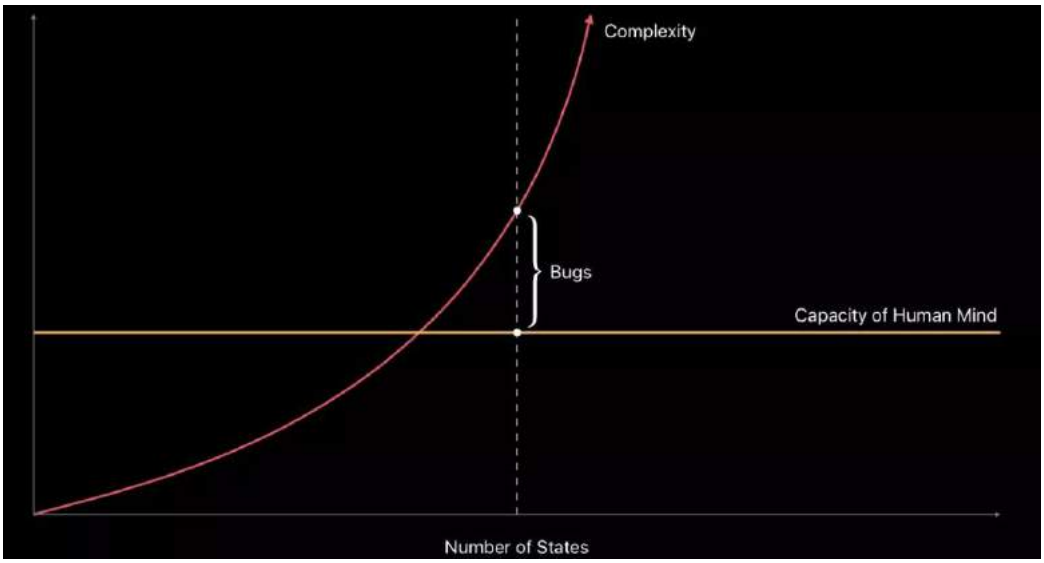
Data Flow

在使用传统命令式编程编写 UI 代码时，开发者需要手动处理 UIView 和 数据之间的依赖关系，每当一个 UIView 使用了外部的数据源，就表明了 UIView 对外部的数据产生了依赖，当一个数据产生变化时，如果意外的没有同步 UIView 的状态，那么 Bug 就产生了。

处理简单的依赖关系是可控的，但是在真实项目中，视图之间的依赖关系是非常复杂的，假设一个视图只有 4 种状态，组合起来就有 16 种，再加上时序的不同，情况就更加复杂。

人脑处理状态的复杂度是有限的，状态的复杂度一旦超过人脑的复杂度，就会产生大量的 Bug，并且修掉了这个产生了新的 Bug。

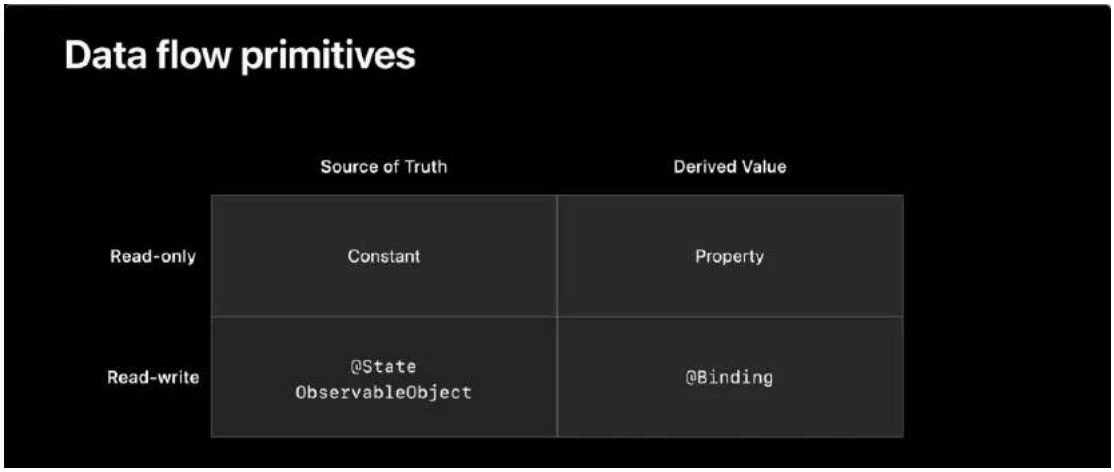




那么 SwiftUI 是如何解决这个问题的?

SwiftUI 的框架提供了几个核心概念:

1. 统一的 body 属性，SwiftUI 自动从当前 App 状态集自动生成基于当前状态的快照 View。
2. 统一的数据流动原语。



关于 SwiftUI 中的 Data Flow 是如何消除视图和状态不一致的，请参考去年撰写的文档系列文章《深度解读 SwiftUI 背后那些事儿》。

今年 SwiftUI 2.0 新增的 StateObject 数据流原语让 SwiftUI 在重复创建 View 时避免重复创建 ObservedObject 从而提高 View 重建的性能。

SceneStorage 和 APPStorgae 让一些可持久化的数据变得更加简单且具有语义化。

New Controls

前面提到的，新增的 DSL 语法 SwiftUI App Lifecycle，以及 Xcode Library Preview 其实本质上都是对去年 SwiftUI 1.0 锦上添花的新扩展。

真正重要的是今年新增的各类新控件，其中通过导出来自 Xcode11.5 和 Xcode12.0 beta 版本的 Swift 声明文件，可以观察到整个声明文件从原来的 10769 行增加到 20564 行。

新增了约 87 个 struct 16 个 protocol。有了这些丰富的组件才可以更好的构建我们的 APP 。

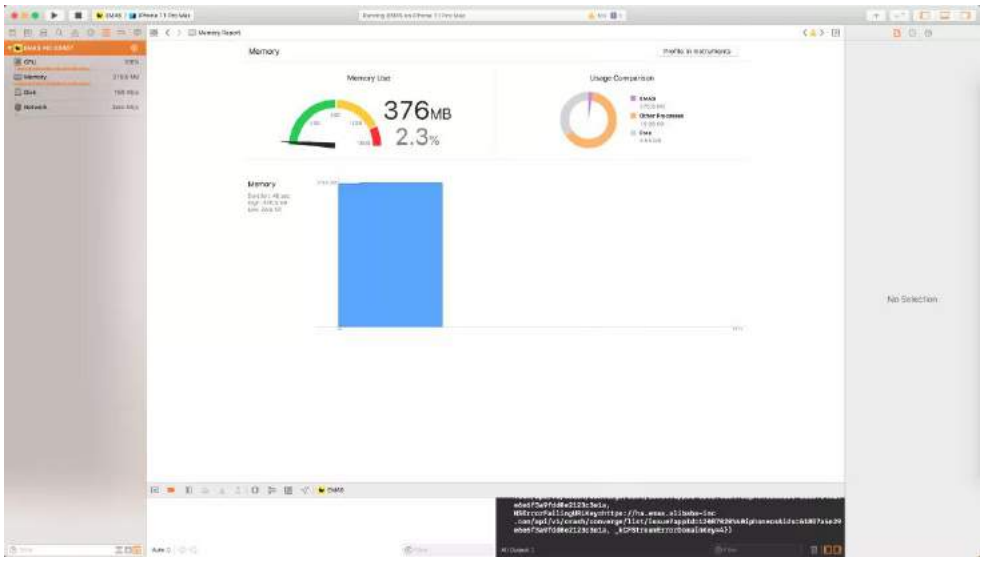
大列表组件

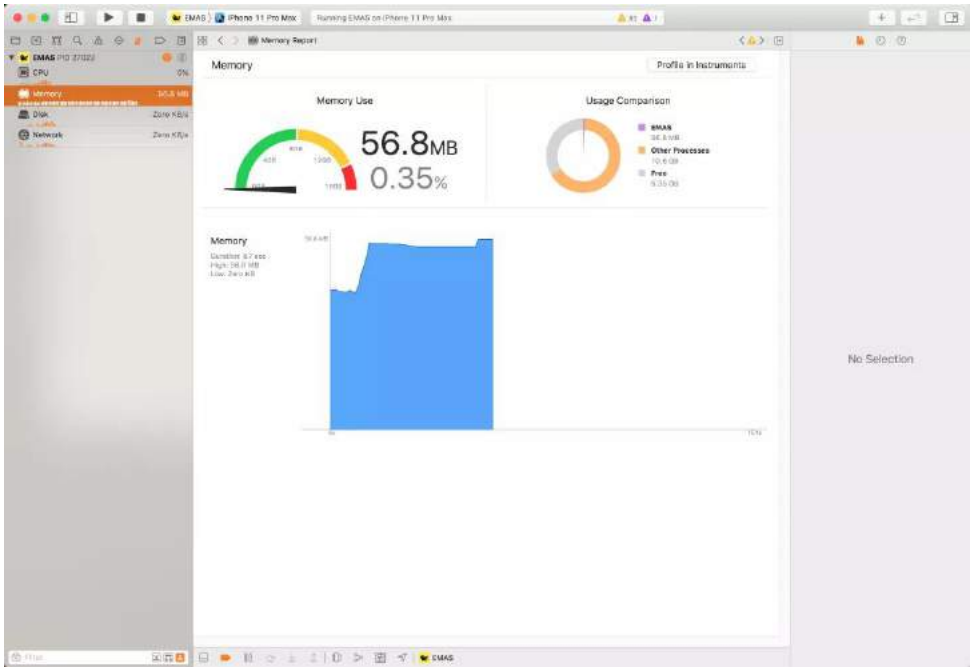
在任何一款 APP 中都会存在类似大列表组件，如淘宝 APP 里面的某家店铺里面商品列表流，首页的信息流，都是具有超长内容的列表页数据。对于长列表页来说，过长的 UI 页面会导致过多的内存占用，在用户的设备中，内存是最为重要的指标，对于目前国内的 APP 市场，低端手机仍然占据大量的市场，对于这些设备来说，一旦内存超标，APP 就很容易 OOM，这会导致用户体验非常差，在现有竞争关系激烈的市场环境下，体验差意味着会失去用户。

对于传统的命令式编程来说，我们可以主动控制 UITableViewCell 的重用，自建缓冲池等一系列手段去优化我们的 APP 内存占用，但是对于 SwiftUI 1.0 来说，系统提供的控件并没有有效的办法去让我们控制页面的渲染，对于大列表页面就容易出现内存占用过高的问题。

SwiftUI 2.0 推出了 LazyHStack 和 lazyVStack 加上 List 渲染模式默认就是 Lazy 的直接解决了最大的性能问题。

笔者以去年使用 SwiftUI 编写的 Emas App 为例，当列表页（并无大图）加载到 500 个时， APP 使用内存已经达到了将近 360MB 。而只需要切换到 Xcode12 API 调整为到 LazyVStack 内存占用直接降低 300MB 。





Widget and Clips

苹果与 WWDC 20 推出的 WidgetKit 支持的 API 是 SwiftUI Only，虽然已经可以混合部分 UIkit 里面的 View，但相信没有历史包袱 最低支持版本为 iOS14 的 Widget 没有人会选择笨重的命令式 API。

同理 Clips 也一样。这里因为篇幅原因就不做展开，后续会有专门的文章分析相关技术。

Swift & SwiftUI 的机会在哪里?

笔者曾经在公司推动集团升级了基建，支持了 Swift 开发环境也在淘宝落地了一些场景，但是集团内一直有一些质疑的声音， 引入 Swift 到底有什么用?

SwiftUI 又是 N 年后才可以用上的小玩意，Objective-C 不够用吗？现在笔者可以回答这些质疑的声音， Swift 未来的机会在 效率，体验和苹果的技术红利。

效率

从研发效率上来说， Swift 对比 Objective-C 的精简程度不言而喻，笔者在淘宝 APP 上线的模块代码量下降了 40 %。

但更进一步，如果编写 UI 界面从 UIKit 转向了 SwiftUI 代码量直接少了不止一倍。更少的代码意味着更快的交付，在目前竞争激烈的市场会有更多的试错场景。

关于使用 UIKit 编写代码转向 SwiftUI 的代码量对比，读者可以参考开源 APP MovieSwiftUI 直观了解。

体验

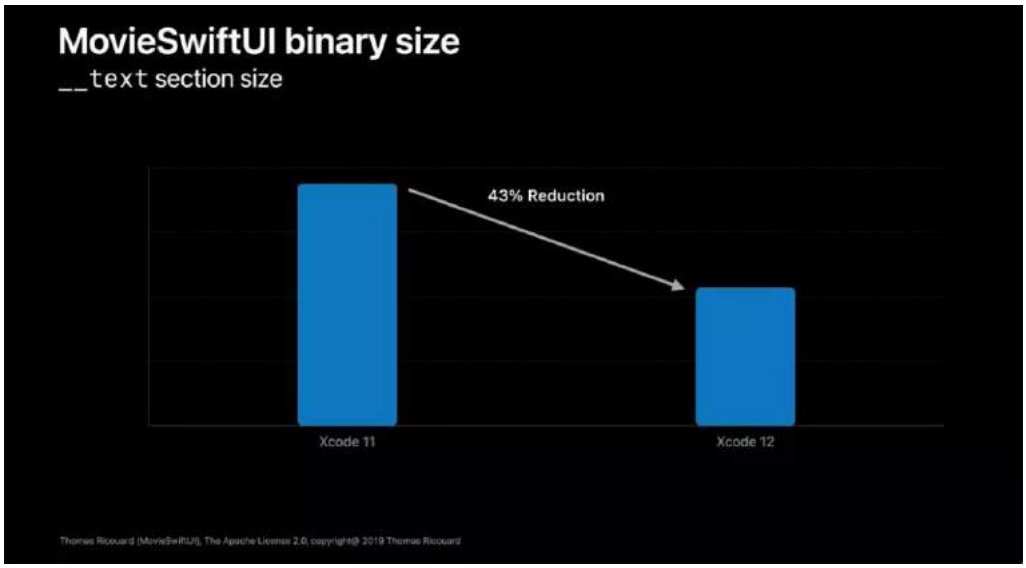
读者可能比较困惑对于切换语言和框架，对体验看上去没有任何帮助，但事实真是这样吗?

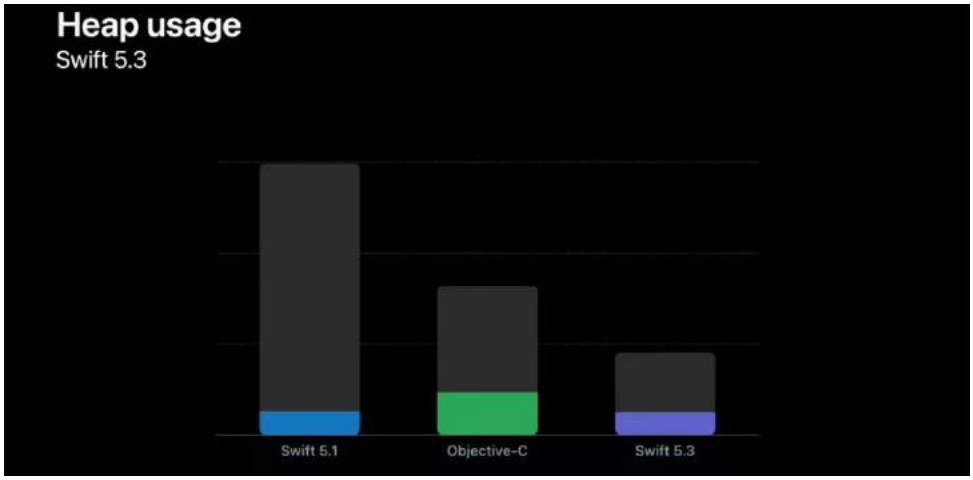
首先引入 Swift 后，由于 Swift 语言设计之初便对安全性列为最重要的目标，Swift 的引入会让代码尽可能的减少未定义的行为，减少 Crash 意味着 APP 的稳定性提高，体验自然更佳。

其次虽然 Swift 同样的语言出于对安全性考虑编译处理的指令会比 Objective-C 更多，但是如果 UI 部分都用 SwiftUI 来写呢?

更少的代码意味者更小的包大小，目前国内巨头 APP iOS 端 APP 包大小都朝着 200 MB 奔去，如果能减少更多的代码对包大小也可以在 200MB 的限制下承载更多而业务。对用户的体验也有较大的提升。

更进一步由于 Swift 选择使用值类型构建整个 APP，值类型的有点在于更扁平化的内联数据结构去分配内存，而不是使用更多间接指针引用，减少了大量不必要的堆内存消耗，意味着整体内存使用量的降低。对整个 APP 的稳定性也有较大的提高。





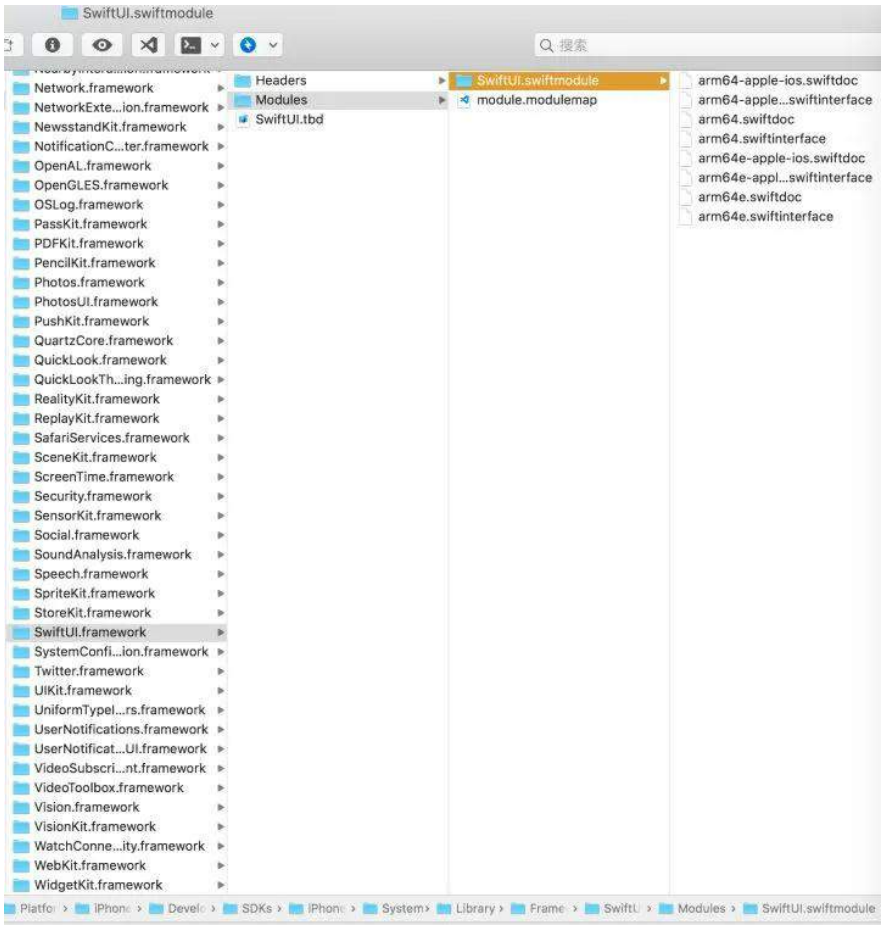
苹果的选择

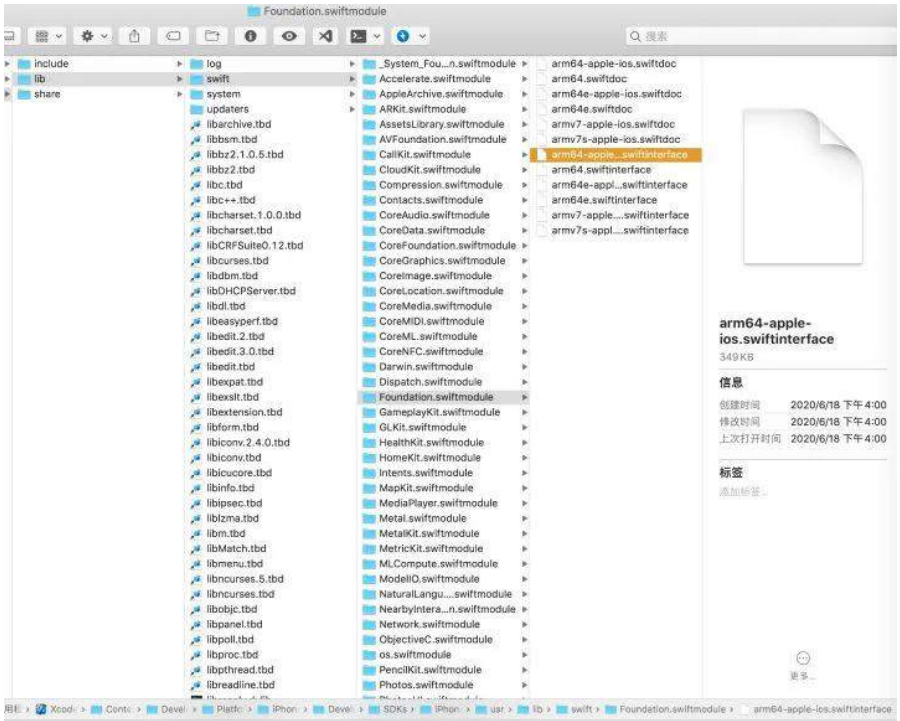
Swift 做为苹果的战略语言已经发展的越来越壮大，自 2019 年 Swift ABI 稳定后，苹果在 Swift 的投入越来越大。我们可以进入/Applications/Xcode-beta.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS.sdk/usr/lib/swift ，/Applications/Xcode-beta.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS.sdk/System/Library/Frameworks ，https://github.com/apple 和 https://github.com/swift-server 看到， 自 iOS 13 以来 苹果新增了约 10+ Pure Swift Library ，10 + Open Source Swift Library，以及针对 144 个公开 Framework，根据 Swift Style 重新设计了 57 个 Framework 的 API。

从以下数据：

1. 从 WWDC17 后 苹果已经不再使用 Objective-C 做 Sample Code 演示；
2. https://developer.apple.com/不再更新 Objective-C 相关的文档；
3. WidgetKit 是 SwiftUI only；
4. App Clips 10M 的包大小， SwiftUI 是最合适的框架；
5. 开源社区逐步放弃 Objecive-C 如 Lottie；

可以判断，Swift 是未来 Apple 平台的唯一选择，越是有包袱的大厂 APP，从现在还不尽早储备，在未来越会寸步难行。


SwiftUI.swiftmodule
Headers
Modules
SwiftUI.tbd
SwiftUI.swiftmodule
module.modulemap
arm64-apple-ios.swiftdoc
arm64.swiftdoc
arm64.swiftinterface
arm64e-apple-ios.swiftdoc
arm64e.swiftdoc
arm64e.swiftinterface
Network.framework
NewsstandKit.framework
OpenAL.framework
OpenGLES.framework
OSLog.framework
PassKit.framework
PDFKit.framework
PencilKit.framework
Photos.framework
PushKit.framework
QuartzCore.framework
QuickLook.framework
RealityKit.framework
ReplayKit.framework
SafariServices.framework
SceneKit.framework
ScreenTime.framework
Security.framework
SensorKit.framework
Social.framework
SoundAnalysis.framework
Speech.framework
SpriteKit.framework
StoreKit.framework
SwiftUI.framework
Twitter.framework
UIKit.framework
UserNotifications.framework
VideoToolbox.framework
Vision.framework
VisionKit.framework
WebKit.framework
WidgetKit.framework


Foundation.swiftmodule
include
lib
share
log
swift
system
updaters
Accelerate.swiftmodule
AppleArchive.swiftmodule
ARKit.swiftmodule
AssetsLibrary.swiftmodule
AVFoundation.swiftmodule
CallKit.swiftmodule
CloudKit.swiftmodule
Compression.swiftmodule
Contacts.swiftmodule
CoreAudio.swiftmodule
CoreData.swiftmodule
CoreFoundation.swiftmodule
CoreGraphics.swiftmodule
CoreImage.swiftmodule
CoreLocation.swiftmodule
CoreMedia.swiftmodule
CoreMIDI.swiftmodule
CoreML.swiftmodule
CoreNFC.swiftmodule
Darwin.swiftmodule
Dispatch.swiftmodule
Foundation.swiftmodule
GameplayKit.swiftmodule
GLKit.swiftmodule
HealthKit.swiftmodule
HomeKit.swiftmodule
Intents.swiftmodule
MapKit.swiftmodule
MediaPlayer.swiftmodule
Metal.swiftmodule
MetalKit.swiftmodule
MetricKit.swiftmodule
ModelIO.swiftmodule
NearbyInteraction.swiftmodule
Network.swiftmodule
ObjectiveC.swiftmodule
os.swiftmodule
PencilKit.swiftmodule
Photos.swiftmodule
arm64-apple-ios.swiftdoc
arm64.swiftdoc
arm64e-apple-ios.swiftdoc
arm64e.swiftdoc
arm7-apple-ios.swiftdoc
arm7s-apple-ios.swiftdoc
arm64-apple-ios.swiftinterface
arm64.swiftinterface
arm64e-apple-ios.swiftinterface
arm64e.swiftinterface
arm7-apple-ios.swiftinterface
arm7s-apple-ios.swiftinterface
arm64-apple-ios.swiftinterface
349 KB
信息
创建时间 2020/6/18 下午4:00
修改时间 2020/6/18 下午4:00
上次打开时间 2020/6/18 下午4:00
标签

我们需要做些什么?

Swift

1. 我们已经做了什么?
2. 一套支持 Swift 二进制的研发环境。
3. 300+ 支持了混编的淘系 SDK。
4. 手淘落地了 6 个模块。
5. 集团新增了约 20 个 支持 Swift 的 APP。
6. 10 多场技术培训。
7. 169+ 语雀知识沉淀。
8. 300+ 工程师的集团 Swift 官方组织。
9. 2 个 技术创新产品。

经过去年横向组织大家共同的努力，我们已经已经支持了横向大基建。包括研发环境，工具支撑，沉淀了大量的文档，还有相关的技术课程。

要朝什么方向去努力

目前集团对于 Swift 的呼声越来越高，我们大量的工程师希望的去使用 Swift 。目前首先要做的事情是依托 Swift 和 SPM 提升我们的开发体验，升级我们的中间件，使业务可以大量的用起来 Swift ，提高我们的研发效率和代码质量。

1. 升级基于 SPM 的新的包管理体系。
2. 升级老旧基础库，打磨新一代基建。
3. 引入新的 Swift 特有库 赋能业务。

SwiftUI

虽然前文提到了 SwiftUI 的众多优点，包括研发效率，体验的提高，但是在国内的环境中 SwiftUI 也有它致命的弊端。

1. iOS 14 才可放心的使用。
2. 只支持 Apple Platform，这和国内的要支持 Mobile Platform 从理念上冲突。

大型 APP 要解决的是如何部署到低版本操作系统上和安卓平台上，毕竟很多公司还在支持 iOS 9 对于升级到最低支持 iOS 14 好像还需要一个世纪那么漫长，而且国内的设备占比大头还是以 Android 巨多 。

虽然可以看到 Swift 语言也在逐渐支持 Android 平台，但是也看到苹果对于安卓平台的 SwiftUI 并没有太大兴趣。

从体验上 Flutter 远不如 SwiftUI 这种亲儿子效果好， 但对于国内跨端欲望旺盛的市场来说 SwiftUI 还是比不过 Flutter， 不过既然 SwiftUI DSL 层已经基本固定，那么也有可能投入人力直接在低版本操作系统上实现一套自建的 SwiftUI 引擎，或者将 Swift UI 引擎移植到安卓平台，比如对接 Flutter 或者直接对接 Android Native。

比起 Flutter 引入双端带来的包大小增量和体验不一致的情况， SwiftUI 保留 iOS 平台体验，只侵入一端的选择显然要更好一点。

不过短期内我们可以在 Clips 和 Widget 场景下开始使用 SwiftUI， 毕竟 SwiftUI 快速的开发效率对和较低的包大小占用非常适合这样的场景，我们可以在业务场景中练兵储备我们的 SwiftUI，并积极在主 APP 中尝试。

参考：

SwiftUI 背后那些事儿
MovieSwiftUI
SE-0281-main-attribute.md

Add custom views and modifiers to the Xcode Library

Structure your app for SwiftUI previews

Introduction to SwiftUI

What's new in SwiftUI

App essentials in SwiftUI

Visually edit SwiftUI views

Stacks, Grids, and Outlines in SwiftUI

Build document-based apps in SwiftUI

Data Essentials in SwiftUI

Build a SwiftUI view in Swift Playground

Build SwiftUI apps for tvOS

Build SwiftUI views for widgets

Build complications in SwiftUI

What's new in Swift

Swift packages: Resources and localization

Distribute binary frameworks as Swift packages

Explore logging in Swift

Create Swift Playgrounds content for iPad and Mac

Embrace Swift type inference

Explore numerical computing in Swift

Unsafe Swift

Safely manage pointers in Swift

Explore numerical computing in Swift

Explore Packages and Projects with Xcode Playgrounds

Use Swift on AWS Lambda with Xcode

2020 前端开源领域技术展望

作者 | 展炎、御术、偏右
出品 | 淘系技术

导读：可以说开源前端领域正如偏右所说的渡过了波澜不惊的 2019 年，2020 年开启了下一个 10 年，站在这个起点展望前端的未来，必会面临巨大的变化和挑战。说实话，展望接下来的十年会发生什么？十年对于互联网，对于前端来说太长了，很难想象十年后会是什么一个场景，但这路上有些事情在我们看来正在发生。

TypeScript 是当下的政治正确但前端终将语言无关

毫无疑问 TypeScript 将成为很长一段时间的主流，大型前端开源项目大都已经或正在全面拥抱 TypeScript，他能让我们拥有很多面向对象语言、强类型语言才具备的先进特性，能帮助我们提升代码质量，降低团队协作上的成本和风险，如果我们的项目还未转过来，是时候动手了。

但一个开源项目的质量跟语言本身并无直接关系，更别说 TypeScript 本身依旧还只是带「静态类型检查」的「弱类型」语言，OO 并不是必须的，用原生 JS 也一样能写出优雅的封装与继承，更让人期待的是 WebAssembly（https://www.zhihu.com/question/362893829） 的潜力被真正开发出来后，前端终将语言无关，前端在研发的是人机交互界面上的产物。

浏览器技术和云计算的进化将加速前端构建和资源加载的变革

Webpack 5 正式版发布，通过物理缓存大幅提升构建性能，在短期内还会是主流构建工具；但随着 ES Modules 成为主流，CJS 淡出，ESM CDN 大范围使用后构建工具将转向 Bundless，Webpack 不可避免也会逐渐退出历史舞台。特别是当 HTTP/2 到来，支持请

求与响应的多路复用来，HTTP 首部字段被大大压缩，传输效率会得到极大的提升，届时我们将不再需要对资源文件进行合并操作，构建工具也会随之加速进化。

C/S 到 B/S 再到 C/S

历史总在不断重复轮回中循环上升，互联网，特别是云的出现让软件从 C/S 走向了 B/S，甚至如 WebAssembly 技术的出现，我们能更低成本的将原先的客户端软件移植到浏览器上去，一切看上去都是朝着 B/S 发展。然而我们也要看到另外一个趋势，自成生态的巨型应用如微信、支付宝、Facebook 等正在收敛浏览器上软件至他们的客户端上，背后的典型技术是小程序、PWA，我们甚至已经看到了出现在 PC 客户端上小程序。而借助如 Electron 这类项目，很多 B/S 模式的软件服务在推出自己的客户端以谋求用户更专注的使用体验，特别是很多传统以 C/S 模式提供服务的软件推出客户端会更贴合用户的心智。这是我们看到的历史轮回，但接下来的这段 C/S 历程跟过去的 C/S 所用到的技术已经完全不一样了，不管是小程序、PWA 还是 Service Worker、Electron、这 Client 端上开源项目会迎来一个春天。

新兴交互场景将唤起 Micro UI

Micro Frontends （https://micro-frontends.org/） 正风生水起，然而它不过是已经风靡多年但仍备受争议的 Microservice Architecture（https://microservices.io/）在前端的应用，可以帮我们更好地去解决复杂应用的开发和维护问题。在人机交互，用户如何在一个复杂的产品中找到此时此刻最需要的一个 UI，仍然是一个未被很好地解决的问题。用户常常迷失在一个个大型 APP 复杂的 UI 中，然而很多时候我们只是需要一个能高效完成某项工作的 UI，姑且称为 Micro UI 吧。在银河帝国等科幻小说及科幻电影中，常常能看到一种叫"信息窗"的黑科技，这样的每个信息窗就是一个 Micro UI。随着 AI、AR、VR 、IOT 持续推进话，设备将越来越懂人，信息和数据的呈现也将越来越生动化，这些在科幻中才出现的东西在未来 10 年一定有一些会变成现实 ， 这种 UI 的开发方式也将为前端的开发方式带来质的变化。

RPA & Marchine Driven UI

在 AI 和 IOT 时代，如果你然坚持认为应用一定是一个需要由专业人士设计开发，并且由用户通过手机主动使用 Web 产品的话，那么你将有较大概率过未来。

RPA 这个远古时代活跃在游戏外挂的古老的技术在 AI、标准化产品飞速发展的正在焕发着升级，UiPath（https://www.uipath.com/）、Zapier（https://zapier.com/）、Microsoft Power Platform（https://flow.microsoft.com/zh-cn/）、UI Bot（https://www.uibot.com.cn/）等工具正在改变很多行业的工作方式，而以 Google Assistant 为代表的智能助理正在改变人们使用电脑的方式。作为前端，我们要考虑是：UI 不仅是被人消费的，还可以被机器消费，甚至只需要描述你的意图，这些非常需要一种高效的方式让你的应用可以被机器人来驱动。甚至应用已经不再是一个当下我们能看到的，而是一个个机器人了，而我们提供的则是：RPA 机器人制作工具、可以被 RPA 来驱动的标准 UI。如何设计与实现可以被机器人、人同时使用的 UI 是我们面临的一大挑战。

图形技术依旧会是界面上的热点

尽管脑机交互已经在实验室里孵化，但未来十年眼睛依旧还是人类获取外界信息最主要的通道，图形化可以比文字快十倍甚至万倍的速度清晰有效地传达信息，这是人类经历漫长进化习得的强大视觉化思考本能。在数据愈加丰富、使用场景愈加多样的下一个十年，如何高效的从数据中获得洞察依旧会有强烈的诉求，前端这个最接近数据可视化能力模型要求的工种依旧会是这个领域的主力。而另一方面，未来会出现新的「屏」，不管是 IOT 下的实屏还是 VR、AR 下的虚屏，当下已经相当完备和标准化的绘图 API 最有机会被新「屏」优先支持和采用，在这基础上利用图形技术实现一套新的 UI 基础设施甚至语言都是有可能的。

前端智能化

前端近年来一直尝试提升效率，可视化研发、搭建体系、工程&工具、端多解决方案等等都是为了解决效率，但天花板已然出现，下一步必然会引入智能化能力，在原来的技术体系&产品上，注入自然语言识别、图像识别的能力，使得效能的发生一系列质的变化，来突破这

个天花板。更大层面上，运用智能化能力来解决工作中的问题，会逐步成为前端的必备能力和思考方式。

就算不想去蹭人工智能的热，事实上是你不得不接受并开始学习人工智能，大量简单如千人千面的 Banner、营销活动页已经纯粹通过人工智能自动化生产，并不需要前端去参与研发，复杂如中后台的表单、表格、图表展现也已经可以通过提供接口智能推导出最佳的页面实现，只需要一行代码输入数据就能出可视化图表，甚至从设计稿直接生成页面也将变成现实。前端智能化的项目将会越来越多，任何时候开始都不晚，但不开始就晚了。

端到端的研发

随着云原生和业务平台能力逐步完备，前端运用 Serverless 能力来完成端到端的研发工作会逐步成为一种趋势，在这个趋势的背后，会面临两个挑战，一方面是需要将现有的工具和平台延展到 Serverless 上，在支持 Serverless 的同时，必须将端到端一体化的来看待，包括：研发工具、监控体系、质量体系等等；另一方面对人员自身能力和意识有了新的要求。前端这个岗位的职能边界和能力范围的会发现变化。

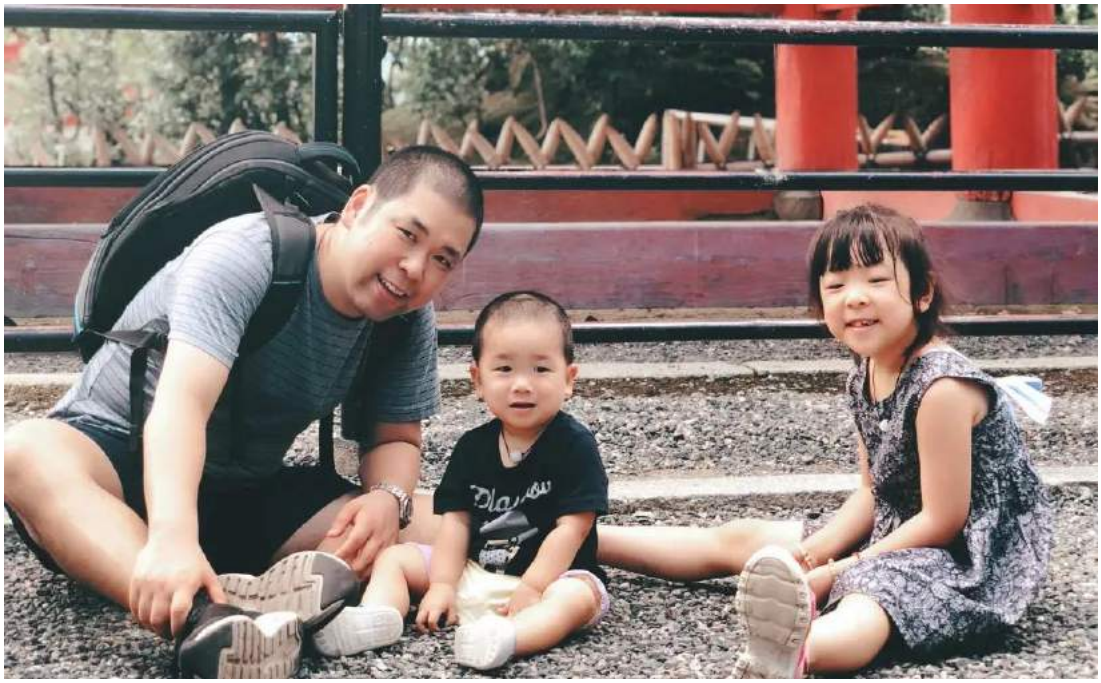
基于前端智能化与端到端的研发方向上，是未来前端开源上的最具吸引力的项目；而在整体的层面上，前端具备吸引力的项目会有两个方面的特征：

1. 在现有相关体系上提出新的理念和思想；随着框架、构建库等基础设施收敛，类似的重复性的开源项目会减少，但创新并不会停止，等到更多关注和认可的一定是能力在基础设施上提出新的理念和思想的项目。
2. 在现有相关体系上与其他领域产生链接；有些技术未必是刚刚出现，但今天重新受到前端业内的关注，比如 WebAssembly，因为大家发现了新的运用场景。未来将现有前端技术与其他技术领域相结合的项目和设计，会受到开源社区的更多关注。

阿里专家跨平台思路：Weex、RN 还是 Flutter?

作者 | 黄刚（腾渊）
出品 | 淘系技术

随着移动互联网的发展，Android 和 iOS 呈两分天下之势，跨平台开发也逐渐成为移动领域的热门话题之一。怎么看待其背后的发展逻辑？跨平台开发的难点是什么？未来的发展方向如何？阿里巴巴淘系技术部资深无线技术专家黄刚（腾渊）受 InfoQ 采访邀约，分享了跨平台开发的发展趋势和变化。



黄刚（腾渊）

黄刚（腾渊） 阿里巴巴淘系技术部资深无线技术专家，现淘系技术部基础链路负责人，负责首页、商品详情、交易、信息流等核心业务。2014 年加入阿里巴巴，2018 年双十一淘宝技术部技术队长，2019 年淘系技术部技术队长。

前言

腾渊认为：从早期比较流行的 Hybrid App，到后来的 React Native、Weex，再到现在比较火爆的小程序和 Flutter，可以看到跨平台的发展趋势是框架变得越来越重。如果当前的业务情况下前台呈现比较不稳定并且整个前台开发占比较高，那么应用跨平台框架的收益就比较高了。到底哪个方向更正确、哪个框架更好其实是没有标准答案的，更多需要开发者因时制宜地选择。

腾渊还会在即将召开的 QCon 全球软件开发大会 2020（北京站）上担任「移动新生态趋势下的应用实践」专题的出品人，感兴趣的读者可以关注。以下为黄刚（腾渊）的观点精华。

如何选择适合自己的跨平台开发框架?

在移动互联网时代，当 Android 和 iOS 奠定了整个移动 OS 的地位后，跨平台研发就一直是一个大的研究方向或者说大的课题。这个话题其实很有意思，因为很多时候大家会有些默认的假设，比较容易忽视这个方向出现或者变热的前提条件。

从 PC 时代开始，就有一些像 Qt 之类的非常优秀的跨平台软件研发框架。但是我们回过头来看，那个时候面向个人用户的跨平台研发很难说是一个非常热的方向，其中最主要的原因就在于 PC 时代 Windows 的绝对统治地位。假设今天我们在移动互联网上只有一个 OS 占绝对统治地位，那可能今天也不太会有业界同仁在研究跨平台研发这个事情了，这是一个大的前提。

在移动时代，在 Android、iOS 作为移动 OS 双巨头的格局下，天然就会带来一个重复开发导致移动应用开发成本增大的问题。要开发一个应用，服务端、前端都可以是一套班子，客户端基本需要 Android 和 iOS 两套开发班子，所以不难理解跨平台研发这个话题的起点在于降低研发成本。在这个基本问题下，往下衍生的问题就是多平台的开发增加的额外成本到底是多少，在整个研发生命周期中的占比是多少，跨平台的开发框架能多大程度上解决这个问题？只有这个问题回答清楚了，我们才能明确适合自己的跨平台开发框架应该具备哪些特征，这个跨平台框架在一个 App 应用中的使用范围是多大。

从早期比较流行的 Hybrid App 到后来的 React Native、Weex，再到现在比较火爆的小程序和 Flutter，可以看到跨平台研发框架变得越来越重。是不是一定越重、越新的框架越好？答案是不一定。

从应用的 Life Cycle 来看，研发阶段只是其中一个阶段，是否具有长久的可维护性、可运维性也是需要重点考虑的问题。再到研发阶段本身，整个研发是从前端到后端 End-to-End 的，跨平台更多地是集中在大前端领域内的话题，如果在当下的业务形态里，前台展现是高度产品化、比较稳定或者对于性能以及交互的要求极度苛刻的，那么 Cross-Platform First 未必是一个理想的选择。

一方面是多平台开发工作在整个研发成本里的占比不高，ROI 未必高；另一方面是 Cross-Platform First 是以牺牲平台特性为代价来达到跨平台的一致性（本质上跨平台研发框架也基本无法做到多平台上表现得完全一致性）。在达到一致性表现的过程中，工程上的填坑成本可能更高。

反过来讲，如果当前的业务情况下，前台的呈现比较不稳定并且整个前台开发占比较高，那么应用跨平台框架的收益就比较高了，在阿里比较典型的例子就是 Weex 在大促会场上的应用。

我们都应该理解，从 Hybrid 的方案到 React Native、Weex 再到 Flutter，本质上都是在研发成本、灵活性、性能体验三者间找一个平衡点，只是大家切入的点不太一样，最终导致整个解决方案有了不同。假设说现在你要做一个新的 App，可能整个开发团队是多前端、少客户端的，那么我可能会比较建议大家多考虑 Hybrid 的模式；如果对性能要求比较高，就可以考虑用用 Weex 或者 React Native；反过来，如果是客户端同学比较多，那么考虑下 Flutter 未尝不可。

其实我们也可以看到，像 Google 同时在推 Flutter 和 Kotlin，这两个东西本质是不一样的，Flutter 更多地体现 Cross-Platform First，Kotlin 更多地体现 Platform First，到底哪个方向更正确，哪个框架更好，其实是没有标准答案的，更多需要开发者因时制宜地进行选择。

阿里各跨平台开发框架的发展及应用

当下阿里用的比较多的应该是 Weex、DinamicX 和小程序这几个框架，在不同场景下解决不同的核心问题。任何框架都不能脱离当时发展的背景讨论，这几个框架是因为在不同阶段解决了当时手淘的一些核心问题，所以才能在各条业务线上广泛应用。

我们从 2015 年开始做 Weex，2016 年大规模应用。当时我们大促会场页面从研发效率以及发布的灵活性考虑，统一使用的是 H5，但在当时，性能和体验都有些问题。同样，当时手淘上众多的导购业务因为很难接受 H5 的性能与体验，强烈要求使用 Native 做页面开发，这样就又带来了另外一个问题，即 App 的包大小的问题。Weex 在这个背景下诞生，在尽量保留 H5 研发效率以及部署灵活性的前提下，尽量提升页面的性能体验。

不难想象，在这种情况下，我们整个开发框架必然是面向前端、全面兼容 H5 的研发模式。当然其中 Vue 和 Rax 这样的前端框架也是功不可没的。因为 Weex 的出现，大幅度提高了当时会场页面的性能，而且满足了大部分导购频道对于性能的要求。导购业务在性能基础体验能够满足的情况下，其实很多导购业务还是倾向于更灵活的研发与部署模式，所以当时很多的导购业务还从 Native 转回到了 Weex 进行开发，间接帮助了手淘 App 包大小的控制。

我们从 2017 年开始做 DinamicX，2018 年大规模应用。这个框架的诞生就要顺着刚才的历史接着往下讲。Weex 解决了原先使用 H5 的业务的性能诉求，但是我们发现 Weex 在一些产品化程度很高的业务域内，依然不能满足性能要求，比如说首页这类一级 Tab 页，所以这种业务基本都还是保留了 Native 的开发模式。

虽然 Native 的开发模式被保留了下来，但是不意味着这些业务对于开发的平台一致性、研发效率、部署效率没有要求。为了解决这些问题，DinamicX 使用了类似于 Android layout 的声明式布局。相较于 Weex，DinamicX 并没有引入脚本引擎的能力，通过牺牲部分灵活性，来达到性能的极致。通过这样的方式，能够做到在性能基本和 Native 开发持平的情况下，比较灵活地调整页面布局，再通过和我们内部的新奥创研发平台的结合，实现页面布局与业务逻辑的分离。通过这样的方式，DinamicX 这个框架基本解决了我们在基础产品业务

域内的研发效率以及性能体验的平衡问题。

再来讲讲小程序框架，现在我们更多地是使用在了商家应用场景上，核心是为了解决我们在集团多个 App 之间业务快速部署以及外部商家开发的页面快速入驻手淘并且能够得到有效管控的问题。小程序最近比较热，这个点上我就不展开了。

至于 Flutter，我们内部使用比较深入的是闲鱼的同学。对于跨平台领域的解决方案，我不太秉持只有一个正确答案的想法，对于终极解决方案这个结论我还是比较存疑的，但有一个需要探索的重要方向是可以确定的：决定一个跨平台解决方案的成败有两个重要因素，一个是技术产品化程度，或者说是工程化水平；另外一个是整个开发者生态的构建。如果从这两个因素来看的话，可以明确都还有一段很长的路要走。手淘今年也成立了专门的团队在研究 Flutter 的应用，我们可以拭目以待。

5G 时代给移动领域带来的新机会

5G 也是大家非常关注的热点，我们内部的讨论也会比较多。毫无疑问，5G 将能带来一些爆点场景，但是这方面的预测是相当困难的。现在市面上关于 AR/VR 和多媒体应用的畅想理论上都没有什么问题，但是从商业的角度来讲，我们还不知道整个变化对于成本方面的影响，而成本的考量对于应用场景是极其重要的。

我举个简单的例子，很多 App 在视频自动播放下都有一个设置选项，叫做“Wi-Fi 下自动播放”，绝大多数 App 在使用移动流量播放流媒体之前一般都会给用户弹一个提醒，“当前在用移动流量播放，可能会产生资费”。大家想一想，这个背后的本质是不是就是关于成本的考量。这个限制其实对于整个应用的场景影响是非常大的，正如 4G 的流量费用大幅降低让移动互联网走到了亿万用户手中，5G 的使用成本将极大影响它的使用场景。但是我们依然可以做一个大胆的预测，当 5G 帮助大家极大突破了网络传输速率的束缚后，后面紧接而来的可能是对终端设备算力大幅提升的诉求，最后才创造出一个个崭新的应用场景，这是一个从网络到终端设备到应用场景整体升级的过程。

我个人最近比较关注 On-Device AI 的应用实践，如果说跨平台解决的是节约研发成本的问题，On-Device AI 的应用是真正会产生业务增量的技术新赛道。过去的一年中，在 On-Device AI 上手淘有了比较大规模的应用，一个典型的场景就是信息流这个商品、内容等的混合智能推荐场景。我们通过更低的资源消耗、更多的数据输入、更实时的感知，在整个信息流场景导购侧有了非常大幅度的提升。过去一年的实践只是一个开端，我们齐头并进的应用场景还包括 AR 美妆、消息 Push、直播等等，这块的想象空间是非常大的。

小结

在这一篇文章里，我们介绍了跨平台开发的发展趋势和变化，如何选择适合自己的跨平台开发框架？阿里各跨平台开发框架的发展及应用，以及 5G 时代给移动领域带来的新机会。未来「淘系技术」将持续与大家分享前端前沿技术和应用实践。

第二部分 职场成长技巧与解惑

- ✧ 职场成长技巧&故事
- ✧ 一线职场&学习问答实录

职场成长技巧&故事

设计转研发，台湾程序员的成长之路

作者 | 黄泰成(斯肯)
出品 | 淘系技术

作者简介：黄泰成（花名：斯肯），是一位来自台湾的程序员，加入阿里巴巴大约 5 年多，目前在阿里巴巴淘系前端团队负责互动方向，过去从设计跨到客户端、后端、前端，先后经历过社群、安全以及目前的电商领域，目前专注在终端解决方案以及互动技术；喜欢自己动手作各种工具，常写些小工具帮助自己生活通过软硬链接将生活中的片段自动化；由于过去的所学为设计，平常也喜欢去看各种艺术、设计、摄影展，到处旅行体验生活与各种不同的文化，认真生活快乐工作。

起点 - 数位媒体设计

我本科读的是数位媒体设计，数位媒体设计在地化说法应该是数字媒体设计或多媒体设计，课程中主要分为动画与人机互动两个部分，其中人机互动因为在设计系，所以写代码相关的课程从来没有开成过，所以人机互动部分也是偏向视觉展现，加上蹩脚的程序或找人合作完成，那时接触比较多的就是 flash & processing，靠着这两个加上万能的 google 在毕业前在台湾国美馆的工作坊和学长姐做了一台大型的多点触控互动艺术作品，值得骄傲的就是时隔一年后 Microsoft 的 Surface Lab(不是现在的 Surface)用完全相同的架构发表了第一台多点触控桌。

大学这四年影响我最深的就是其中一门课是基础设计，这门课每周一个主题一次汇报，有单人有多人，训练了我上台说话不会紧张以及抓重点的能力，设计是一种表达，将你的想法、

情感用作品表达给对方，以及面向问题先了解本质原因，再思考如何用最简单的方式解决。

转变 - 设计转研发

因为过去所学最初的工作自然是选择了交互设计，但是移动端正在兴起，ios3 以及 Android 2.x 正热门，很自然的选择了无线的方向，但是很快的遇到研发说“这个交互无法实现”，心想按照官方交互规范怎么可能做不出来？就开始请教好朋友 google，没几天出了 demo，又过了几周转为正职 ios 开发，按照公司需求也同步开始写起 Android；但也很快的碰到瓶颈，UI 开发非常在行但是资料结构、架构完全不行，因此开始大量补课、看代码，还记得那时 iOS 没有 ARC，因此所有的内存需要自己管理，每天光是算内存 retain count 是多少就头大，还好很快的 iOS4 开始引入 ARC，这点真的降低很多门槛。

设计转研发对我来说很自然的过程，就像以前做设计一样，选择了最佳的方案解决问题，只是这个方案完全自己动手；上手这么快必须感谢以前的学长姐，大学的数位媒体设计对应的研究所是设计运算研究所，研究方向是探讨设计如何结合程序、大数据结合，因此招生也是一半设计一半研发，这些学长姐确实给了我不少启发，然而这个研究所在我毕业前因为斗争没了，不然放到今天肯定是热门的研究所，先前提出的智慧界面想法也是来自于其中一门课 - 衍生设计。

创业 - 收获满满的过程，更加全面的思考

其实原本只是想换工作，只是想找个能把之前无线开发经验延续，没想到我是这公司的第 1 号研发，自己建立了一个 10 多人的团队，团队包含客户端、后端和设计师，当时公司是以接案为主（俗称外包），我的技术能力在这段时间非常快速的成长，管理上从 0 开始很快的仍然陷入了误区，那时招到的人都是技术上能力不及我的，我就是团队的天花板以及瓶颈，不懂得如何让团队成长；那年 24 岁，很快的我发现管理不是我要的，以及我想做产品不想接案，因此离职，又去了新的创业团队。

在新的团队中我是专职 iOS，是个作回忆共享服务；做产品最重要的就是增长，日常最重要的就是思考如何增长， 9 人团队工作模式就是自己提出想法、讨论、开发、发布、看数

据，当然需求不一定是开发，有可能是发传单、办线下活动、BD 商家、运营论坛、回 facebook/twitter 等，在不具备大数据情况下我们根据 GA 找出了不同渠道来的用户来的规律，并让我体会到 Larry Page 所提到“如何让公司 10 倍数增长？”这样的变革肯定是做对了什么，某个功能解决了用户的刚需，而不是保持现状作精细化运营。

当然肯定最终失败，不然我也不会在阿里，在阿里前这段期间被迫作两次转型，第一次是暂时 ios 上找不到增长点，但是后端很多需求做不完，因此在 2 小时技术灌顶教学后转为转职后端；第二次是投资团队的母公司不想再投钱，要求我们回母公司，母公司没 iOS 缺，但是很缺前端，当时主管和我说了一句“你有两条路，1 转前端 2 离职”，当时选择了前端，作密码管理服务，期间研究了一堆加解密方法，将公司核心的 C++ 加密移植到 JS，整理了全公司前端的脚手架以及开发规范，将 nodejs 推进到线上服务，一年后以前端入职了阿里。

阿里 - 找到对的舞台，努力发光发热

虽然前端入职，但对自己的定位已经不再是的前端这个岗位，而是终端技术，对我来说最重要就是如何提升用户在终端的体验；从入职以来就一直在天猫作营销活动的业务，重点在终端基础服务，直到近期团队合并才变为淘系并且开始正式接触互动的业务，也许一年后可以来写写互动的事。

天猫的营销活动每年节奏很明确，简单可以分为两个阶段，双 11 和准备明年的双 11，我第一年就是负责双 11 主会场，印象很深刻的是业务上每天都有大改动，每天改的很累，回头来看除了累以外没了，现在来看最大问题就是没有数据支撑，因此改动的 ROI 无从判断，以及性能稳定性在那时是凭感觉，缺少量化数据；业务上还有一个点很重要，我相信所有人都有听过“这些都很重要”，当会有这个反应就表示都不重要，没想清楚，这时最好的方式就是问清楚问题的本质，在你清楚知道本质后给出最佳方案这才是个好研发，而不是作需求翻译机，最好的技术应该是了解业务，并且能提前思考业务下一步，仰望星空脚踏实地的做好每一件小事，会比起你急于做好一件大事重要，就像赛德克巴莱电影里面的一句话“好的猎人懂得等待”，在正确的路上别着急，是你的他跑不走，不是你的也别抢着要。

刚入职天猫我也是满腔热血，觉得技术整体工程链路和外部相比简直落后太多，但做完双 11 后开始了解到为什么会有这样的设计，不贴合业务发展的技术架构就是耍流氓，世界上会把技术淘汰的只有市场，而不会是技术本身，别因为自己站在比较好的位置或卡口上就乱定规则，又或是拿着业务背书就开始作小闭环，做之前要想这个事放大 10 倍会是什么？最终局会或最好的状态是什么？本质解决了什么问题？影响了什么人？生产关系改变了什么？举个例子，假设我是作高铁的，今天用户的需求是快速的见面开会，那么我的竞争对手不一定会是开更快的高铁或飞机，而是视讯会议，但这并不代表高铁开的更快更准时就没意义，而是快速的见面开会这个问题不一定是我该去解决。双 11 后直接找了主管说明我不想再作业务，想把基础做好，基础最完美的状态应该是让业务感受不到技术，因为他就应该像是水电，看到水龙头打开就有水，看到插座就是有电，用户要的其实只有符合预期的结果，惊喜不应该出现在基础。

开始管理职起初还是比较不适应，毕竟带团队与带项目还是有很大的差异，尤其是在阿里，带团队第一个反应是想起过去的经验，首先第一个想法就是如何别让自己成为瓶颈，如何让自己的产能下降 50%，但是让团队每个人提升 10%，因为现在你的目标不再是自己而是团队，要的是总体的提升，如何启发新的种子，让每个种子再成为新的大树；记得以前老师说过一件事，“今天你可能对你的老师、父母不满，但很显然的在这个阶段你无法反抗，现在该做的事就是把这些不满记下，提醒将来某一天别成为这样的人”，然而很多事当你真的到那个位置后确实是身不由己，但你知道这样对于其他人可能感受不好，应该尽可能去避免，换位思考能帮助你非常多，如果每个人都能为对方多想一点，世界会更美好。

做产品？我曾经想过再从研发转产品，将我的技能树再点开新篇章，设计、研发、产品都体验后就可以再去创业了（误，但最后想想还是个年轻冲动的想法，首先应该思考为什么会有这样的想法？因为觉得我能做的更好，但是因为这个点我就必须转产品吗？转了产品难道不会再觉得研发或设计达不到我的标准又自己开始动手吗？要学着从有限中找寻无限，学习接受不完美也是种美；如果你身边也有人有类似想法应该鼓励他，鼓励拥有冒险探索的心，不设边界的尝试各种可能，这不是立竿见影的事，但能帮助你对于事情思考的多面的切入。最后身为一位中年身材走样的油腻大叔分享几个经验：

1. 不设边界探索各种可能性，世界上唯一能限制你的只有自己，一位设计师可以变为研发，一位中英文长年不及格被当的人今天能正常的用中英文和人交谈，没有做不到的事，只看有没有这份决心。
2. 多思考问题本质，大胆假设小心求证，深入思考但是用最简单的方式解决问题，把问题放大 10 倍会发现很多事都不是现在该关心的事。
3. 保持专注，所有事都重要就表示都不重要，耐得住寂寞，因为相信所以看见，好的猎人懂得等待。
4. 当迷茫时就回到原点，想想你的初心。
5. 换位思考能帮助你解决大多的问题，尤其是人与人之间。
6. 加班做事只能帮助你熟练技能，多思考多累积能帮你接近卓越，从研发来看写每一行代码都思考如何明天别再写相同代码，让别人别再写相同代码，一年过后你会发现完全不一样。
7. 坚持就是胜利，毅力决定了差距，回头看看自己这些年坚持了什么？
8. 相信你的队友，相信自己的选择，选你所爱，爱你所选，选择被融合比起融合别人是更高的高度。
9. 知道如何种出又大又甜的木瓜吗？就是不断的让它有危机意识，对自己、对团队都应该保持危机意识。
10. 别总是想着要赢，想着不能输能让你内心更加坚强。
11. 身体是你的，平安健康的成长就是给家人最好的交代，看看雷神，多吃蔬菜。

7 年晋升 3 次，他如何摆脱业务前端的职业迷惑

作者 | 林晚
出品 | 淘系技术

在成长的未知道路上，我们总会遇到各种各样的问题，但是，所有的迷茫与逆境都能够帮助我们成长，我们要抓住每一个机会让自己进步，而不是徘徊不前。
淘系前端开发同学——林晚，今天就来和大家分享他这七年的成长经历，以及如何摆脱业务前端的职业迷茫感。

林晚淘系技术部——多媒体前端团队

个人经历

我的经历相对来说比较简单，毕业后一直在阿里。

07 到 13 年我在武汉大学学习通信工程专业，同时辅修了动画双学位。

13 年毕业后加入阿里巴巴国际站，在 B2B 阶段我对动画和创新交互比较感兴趣，共申请了 6 项相关专利，有的已经获得授权，赚得了人生的第一桶金：5 万 。

15 年转岗到蚂蚁口碑，当时主要负责互动营销和质量检测。

18 年了解到淘宝直播，想在多媒体领域持续深耕，最近两年主要负责 ALive 直播开放和多媒体前端领域的建设。

三个阶段都有一定的成长，也得到了认可。下面就带着大家走进我的七年，希望能给处于迷茫期的你一些指引。

阶段回顾

每个阶段的回顾，按照下面几个阶段来划分，总结了下我在每个阶段的关键词：

- 折腾点什么（点）
- 主导点什么（线）
- 突破点什么（面）
- 引领点什么（体）

折腾点什么

初入职场

大家还记得自己做过的第一个需求是什么吗？

我的第一个需求是做一个类目选择器，索引、排序。我记得当时是周五，我了解完需求和师兄说，这个需求我周末弄两天，下周一就可以做好。当时师兄笑了一下。因为那个时候我还不清楚一个需求的完整流程，需要经过评审、排期、开发、联调、提测最后才能发布。

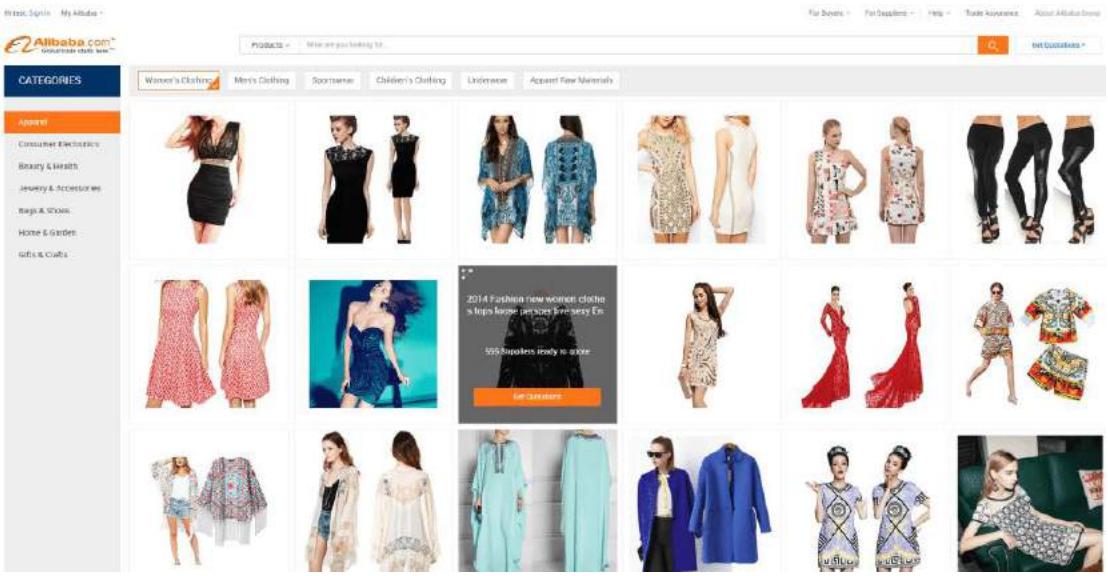
后来慢慢熟悉了公司里的工作流程，开始闷头做项目，一个接一个，很充实，但平淡。第一个绩效我就拿了 3.25，绩效沟通完我给自己写了几个字"知耻而后勇"，我得折腾点什么。

举个栗子

折腾点什么呢？和大家分享个例子：当时喜欢用百度来搜索美女图片，但是列表的浏览体验不太好，于是自己设计实现了一个可以任意方向无限漫游的效果，每一个方向代表一种风格类型，可以实时推荐。大概是这样一个效果：

点击查看动图演示

大晚上的撸完代码，看到这么多取之不尽的美女挺兴奋。又想能不能跟工作结合起来，于是包装了下概念，申请了一个商品加载和推荐的专利。申请完专利，又想能不能在业务里用起来，于是自己做产品设计，找运营聊场景，最后在俄罗斯的一个采购活动上完成了上线，业务效果 List 到 Detail 的转化率比传统的列表形式提升了 3.5 倍。



在国际站一共申请了类似的 6 项专利，其中一项已经获得国家专利局授权，还有的被法务部推荐去申请了海外专利。

阶段成长和迷茫

国际站，让我完成了从学校到职场的转变。现在回过头看，在国际站我的成长是非常大的，对我的职业发展影响也很大。总结了一下，主要有以下几个方面的成长：

- **职业性**：避免情绪化对工作产生影响；结构化思考和表达。
- **业务和数据思维**：养成深入了解业务的习惯，数据化思维看产出和价值。
- **技术结合业务**：不为技术而技术，结合业务场景推进技术建设。
- **团队视角**：将个人的能力变成团队的能力（比如结合自己在动画方面的经验，在团队里组织了动画兴趣小组、搭建动画平台竹马小站）。

在这个阶段，我也产生了一些迷茫，自己折腾的一些事情，和业务场景不太匹配，国际站都是 B 端的场景，而自己更多的关注在 C 端的效果和交互，加上集团 all in 无线的大趋势，出现了 H5 工程师，自己要不要转型？经过考虑后做了一个决定：走向 C 端，转岗到了当时的创新业务支付宝口碑。

主导点什么

业务扛把子

在口碑的阶段，自己的职业度更加地成熟，能去主导一些核心的事情了。业务方面成为了扛把子，负责口碑一些核心的频道和栏目的建设，比如大牌抢购、旅游版等；结合之前动画领域的沉淀，团队里动画游戏类的营销活动我也成了一号位。

如何把业务做好？和大家分享一个大牌抢购的案例。

做业务之前需要充分的理解和思考：

- **业务的站位是什么？**我对口碑 C 端的业务做了一个分类：围绕"找门店"、"找优惠"、"找内容"这三个维度，对数据进行各种形式和场景化的运营。大牌抢购频道在整个 C 端业务中的站位，就是针对"找优惠"设计的一个抢购形式的运营场景。

- **业务的特点是什么？**因为有多个场次，场次有多个状态，以及券来源、状态、类型都非常丰富，导致了券的处理复杂度非常高；另外因为是抢购场景，所以对券信息的时效性要求比较高，库存状态和领取状态等信息都需要做到及时更新。
- **挑战和解法是什么？**比如券处理复杂度高很高，我抽象了一个统一的数据结构，这个结构对券的各种维度做了清晰的划分，各种维度交叉组合可以得出很多的种类，为了保证数据结构变动或扩展之后，视图能方便管理，我做了一层展示归一，得出一套展示模式，并且可扩展。

再比如做口碑年度榜单，我也没把它当成一个一次性的动画需求来做，而是通过这个项目沉淀一套动画类需求的标准化处理流程：动画分层、动效拆解、业务解耦、技术实现。为后续的动画类需求提供更高效、更灵活、高复用地处理方案。

技术攻坚

技术方面主导了口碑的真机检测平台砚台，从 0-1 构建了口碑的真机检测方案。这是一个比较复杂的项目，服务端通过 chair 应用来实现，在 chair 应用中又包括 socket 服务和 http 服务，socket 服务主要负责 group 分组、agent 管理、任务事件处理、数据接收和中转等工作，http 服务主要提供页面、接口、数据的处理和读写，数据存储使用的是 basement 提供的 db 服务；跟 socket 服务保持长链接的 agent 层，负责手机设备的管理，心跳检查，并轮询上报给服务端，还负责命令的接收和下发，数据采集等工作；最后是真机设备，通过 adb 连接到 agent 主机。

整个系统比较复杂，自己主导了整个方案的设计并联动客户端完成落地，中间过程也解决了一些关键的技术问题，比如 chair 如何实现 socket 服务、进程间如何通信、docker 部署多容器 socket 如何同步等。

阶段成长和迷茫

在口碑经过业务和技术的历练，主要有以下几个方面的成长：

- **业务主人翁**：负责的业务当成自己的孩子来养，每天多体验、每天多思考
- **做业务更需要技术思维**：大牌抢购抽丝剥茧、别有洞天
- **系统架构和技术攻坚**：砚台整体架构、chair 应用结合 socket 服务

在口碑待了 3 年，我又开始迷茫了：我做的这些事情都是线条的，重点频道、营销业务、真机检测，我后续的发展还是继续沿着这几条线深入吗，再继续做几年，我的变化会有多大？这里其实引申出来的问题是：线条型发展，我的技术领域是什么？

考虑到自己学的是通信工程和数字媒体相关，毕业设计也是做的基于 WebRTC 的视频会议系统，有一些多媒体的基础，这个领域的壁垒深、空间大，于是想专注于多媒体前端领域发展，转岗到了淘宝直播。

突破点什么

翻车与逆袭

刚到直播的时候接手的第一个项目是直播间里的亲密度，一个客户端同学写的 H5 页面，当时在直播间里调试简直苦不堪言，端侧方案其实是很不成熟的。

在日常迭代还能勉强应付，一到大型晚会有很多复杂的实时互动，问题就暴露的淋漓尽致。18 年双 11 的潮流盛典是整个直播团队最黑暗的阶段，连续几个通宵 cr 代码修 bug，晚会现场还是问题不断，现场一边直播一边改 bug 发布，甚至降级下线一些功能，这个项目差一点就做垮掉。

从北京回来后我们痛定思痛，开始重构直播端侧方案，抽象直播容器，提供配套的工程调试链路，几个月后的双 12 和芒果 TV 合作的人民的宝贝落地了这套方案，从现场可以看到一片祥和，这套方案后来演变发展成了现在的 ALive。

直播间的互动率和停留时长是很核心的指标，我们创新突破启动了媒体智能项目，给淘宝直播带来了流媒体互动的方案。

还有基础的播放器，我们的 VideoX 作为技术项目也进入了多媒体领域的深水区。

灵魂四拷问

上面提到的几个项目，我们在项目初期都会思考几个问题：客户是谁、解决什么问题、使用什么技术方案、带来什么价值。这几个问题思考下来，项目要不要做、怎么做，基本就清晰了，接下来就是细化方案和执行落地。

项目	客户是谁	解决什么问题	使用什么技术方案	带来什么价值	
				业务价值	技术价值
ALive 多媒体开放	一/二/三方直播业务	• 直播业务定制能力弱 • 直播业务接入门槛高	• 动态化方案 • 直播容器方案 • 小程序方案 • ALive工程体系建设	开播场次 DAU GMV	研发效能 直播性能
媒体智能	主播/达人、消费者	多媒体（直播、短视频）场景下 • 流媒体玩法缺失 • AI/AR玩法工程化方案缺失	联动MediaAI Studio，RACE渲染，PixelAI播间团队技术能力，建设工程化方案、高效、规模化生产流媒体玩法	观看时长 互动率	生产效率 跨端流媒体渲染
VideoX 统一播放器	前端开发人员	pc/h5/weex/小程序 多场景播放器接入门槛高	播放内核解耦，统一播放器组件	播放场景覆盖率	研发效率 播放器稳定性
多媒体生产端	主播/达人	多媒体生产工具CEF开发效率低下，满足不了业务节奏，功能不完备	Electron+OBS+MLT重构，统一生产工具	开播场次 视频量	研发效率 稳定性 画质/低延时

同时团队里会产出一张大图，让大家既能看到全貌，又能了解彼此之间的关联，更好的单点突破、全面协作，凝聚成一颗心，共同打好一场仗。

阶段成长和迷茫

在直播的这个阶段，总结了下我主要有以下几个方面的成长：

- **行业洞察和预判**：全面了解自己的业务，竞对的动态；预判趋势，提前布局，反推业务（比如 ALive 直播小程序）。
- **创新突破**：停留时长、互动率下降，如何抢占用户时长，直播视频场景流媒体互动创新。
- **技术深水区**：播放器除了多协议支持、低延时优化、单实例控制等，自研播放内核。
- **影响力**：ALive、媒体智能、VideoX 打造团队技术品牌，跨 BU 影响力。

现阶段依然有些迷茫，但是已经不是方向上的迷茫感了，而是在多媒体领域，前端的核心价值是什么？如何做深做厚？接下来会尝试在集团层面打造 Web Media 的体系。

引领点什么

近 2 年的工作成果得到了认可，晋升到新的层级，对自己也有新的挑战。我觉得接下来，得引领点什么。

去年开始也在往这方面做一些尝试，比如拉通集团组织了多媒体专场分享，倾团队之力打造多媒体前端知识图谱和配套的前端手册，整理经济体多媒体前端大图等，接下来会继续在集团层面建设多媒体前端体系 Web Media，今年也加入了 W3C 的 MEIG 媒体与娱乐兴趣组 https://www.w3.org/groups/ig/me/participants ，目前在这个组织里跟进和推动 W3C 关于媒体相关的标准。

一些感悟

关于业务前端的职业迷茫感，如何摆脱或者说衰减，我觉得最重要的是得找到一个自己愿意持续学习、有领域知识积累的细分方向。我自己找到了，我会在多媒体这个领域持续发展，现在我没有方向上的迷茫。工作了 3-5 年的同学应该需要回答这样一个问题，自己的技术领域是什么？前端工程化、nodejs、数据可视化、互动、搭建、多媒体？如果确定了自己的技术领域，业务前端的迷茫感应该会衰弱很多。

大家做业务，都有很大的业务压力，但公司对我们的要求是除了业务还要体现技术价值，这就需要我们做事情之前有充分的思考。在评估一个项目的时候，要想清楚 3 个问题：业务的目标是什么、技术团队的策略是什么，我们作为前端在里面的价值是什么。如果 3 个问题都想明白了，前后的衔接也对了，这事情才靠谱。

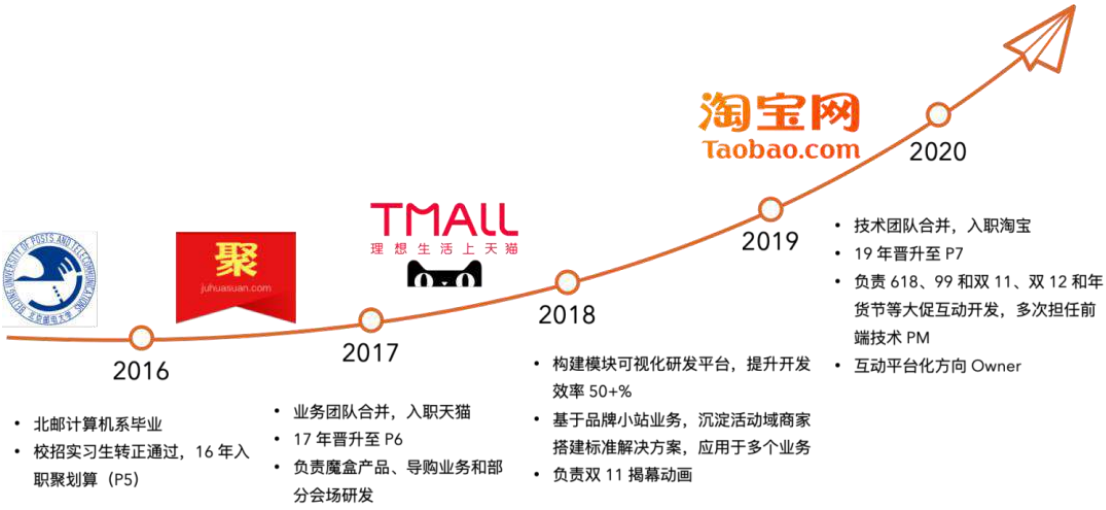
希望大家还是能像最初来阿里的时候一样，能多折腾，保留这种折腾劲，甚至是孩子气，如果你还有的话。

校招实习生4 年升 P7，她想分享的 3 个技能与 2 个反思

作者 | 珑晴

出品 | 淘系技术

作者简介：珑晴——淘系技术部前端技术专家，16 年校招实习转正进入的阿里，当时是在聚划算前端团队，随着业务变化一路从聚划算到天猫至今加入淘系技术部，负责日常活动营销的同时，也多次参与大促会场&互动的研发，在支持业务的过程中不断挖掘提炼创新，最终完成从前端小白到前端技术专家的蜕变。



本文，将结合笔者的 4 年，从以下几点具体谈谈我的前端成长之路。

- 职业经历
- 关键技能
- 一些反思

职业经历

职场初探-实习的那几个月

首先是实习期，当时是北邮的一个师兄内推来的杭州这边。实习期间，主要负责 2 块东西，XList 和周末淘宝， XList 是聚划算之前的一个解决无尽滚动的框架，当时主要还是在熟悉团队的工具链、看源码，顺便给其他业务打打杂。

- 学习 Zepto / Less / Jucier / Gulp 等，掌握 移动端开发测试 基本技能
- 负责 XList 的维护，基于 XList 框架研究模拟滚动相关和移动端性能优化 问题
- 开发周末淘宝业务

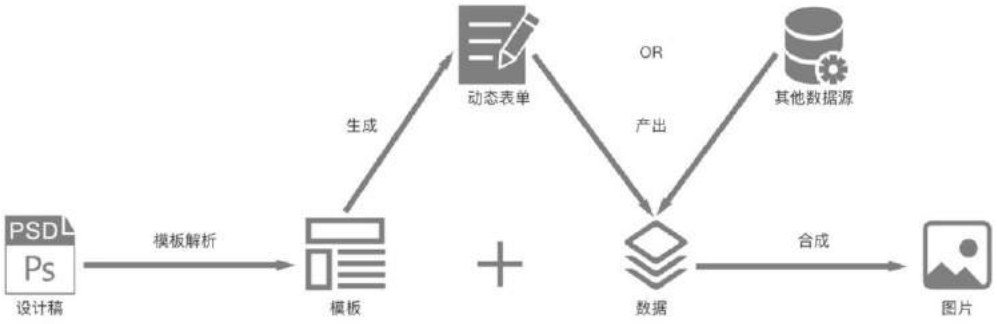
这个时期的重点是团队融入和兴趣探索，对实习生本身能力要求并不高，所以这个阶段的同学们不要过于焦虑，重点还是看自己的个人意愿与团队的匹配度来进一步决定是否要继续待在这样的团队。

新人菜鸟-入职第 1 年

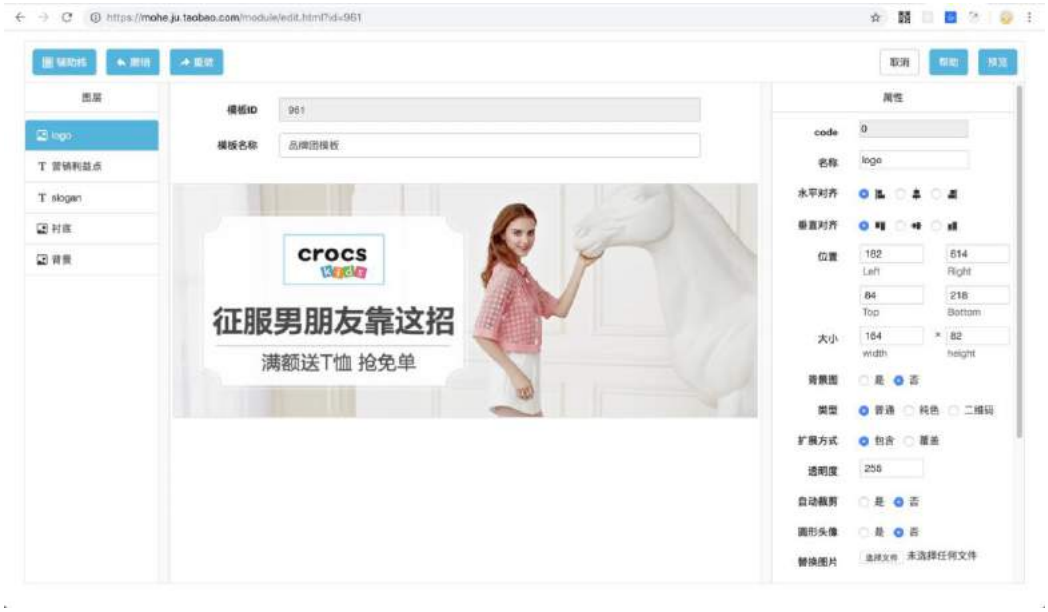
然后是入职第 1 年，我的菜鸟时期。我入职的时候，刚好赶上聚划算整个在做品牌升级，当时存在一个商家素材不规范的问题，亟需解决以提升用户体验。

不规范的商家素材

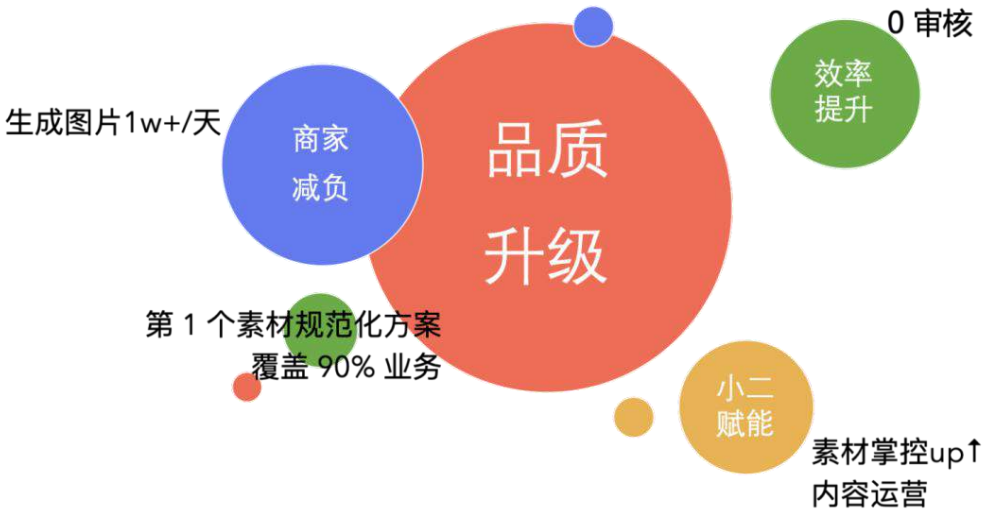
我就在师兄的指导下，设计了这么一个基于 PSD 解析的规范化合图方案，这是技术方案的简图。



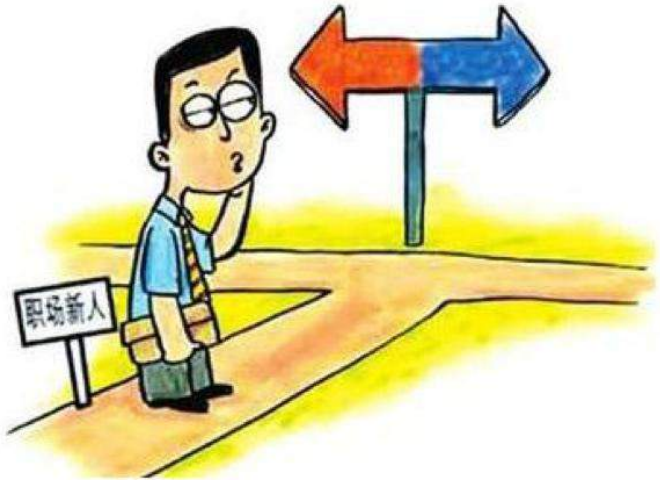
在 PSD 解析的基础上，我还提供了一个可视化界面给设计师调控实现配置生成图片，界面是参考 sketch 实现的。



这个方案在聚划算整个实施之后，也取得了很好的业务结果，覆盖了 90% 的业务线，平均每天合图超过 1w 张。而这个工具，也是当时团队内部第一个完整的 node 应用。



总的来说，第 1 年是新人时期，在这段时间，要开始学习识别业务需求，能进行方案设计，推动个人技术栈的成型。这个时期，我们完成的是从学生到社会人身份上的转变，在做业务的同时，一定要时不时的抬抬头，把自己做的东西拿出来跟大家分享，既增进了团队对自己的了解，也促进了自己对知识的进一步总结概括。

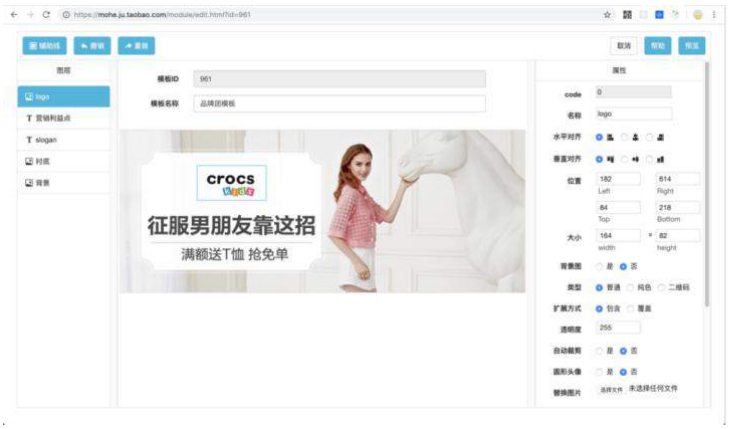


独当一面-升级打怪

接下来，就是作为新人成功 landing 之后的升级打怪了。这个时期，我给自己的定义是独当一面。这里主要结合我个人做的另一个工具阐述。

坑位研发，是我们电商前端工程师日常开发过程中绕不开的事情，这是一些常见的商品坑位。

受之前做的配置生成图片工具的启发，我萌生了用编辑器产出布局的想法。



配置生成图片

编辑器产出布局

然后，我就做了一个坑位可视化开发工具，左边是编辑器界面，右边是消费者侧渲染的 demo。通过这个工具，设计人员简单复制粘贴、拖拖拽拽就可以生成一个新坑位，极大地降低了 UI 的开发成本。

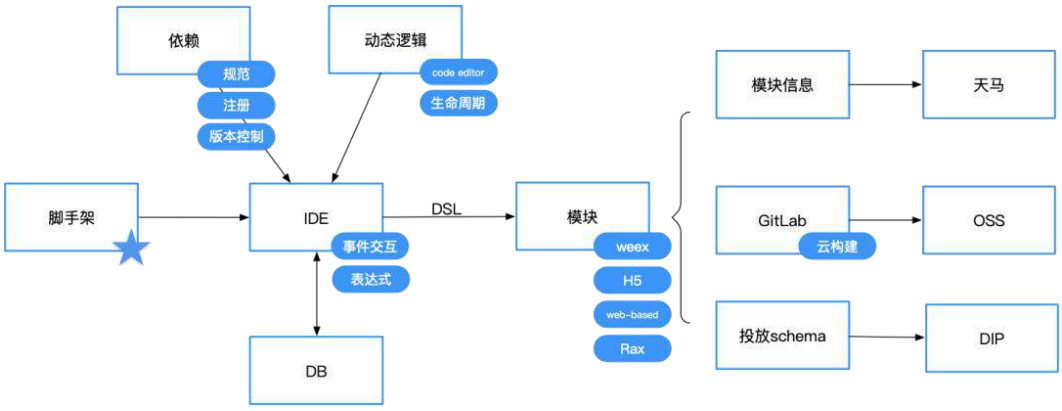
UI 编辑器

这是当时在会场上的应用情况，做到了业务变化的分钟级响应。

但是，工具在推广过程中，也遇到了一些问题，包括如下：

- 不支持存在事件交互、数据处理等逻辑的场景
- 运行时解析方式在端上性能不佳
- 脱离正常的研发流程
- 。。。

当然，这些问题也正好给了我独立 owner 且重构产品的机会。于是，我结合了当时天猫的研发体系，推出了融入开发体系的模块可视化研发方案，即 2.0 版本。这是 2.0 版本的完整链路图。



下图是 2.0 版本上线之后取得的结果，也是在那一年拿到了年度 3.75，并成功晋升。

从我个人角度，独当一面这个时期的重点是在能游刃有余的完成分发到自己的工作之外，建设核心能力、寻求突破。这个时期，已经对业务有了较为深刻的了解，可以从零到无设计一块东西，技术上能够对接业务方，技术之外要能做项目管理，协同多方有效完成任务。

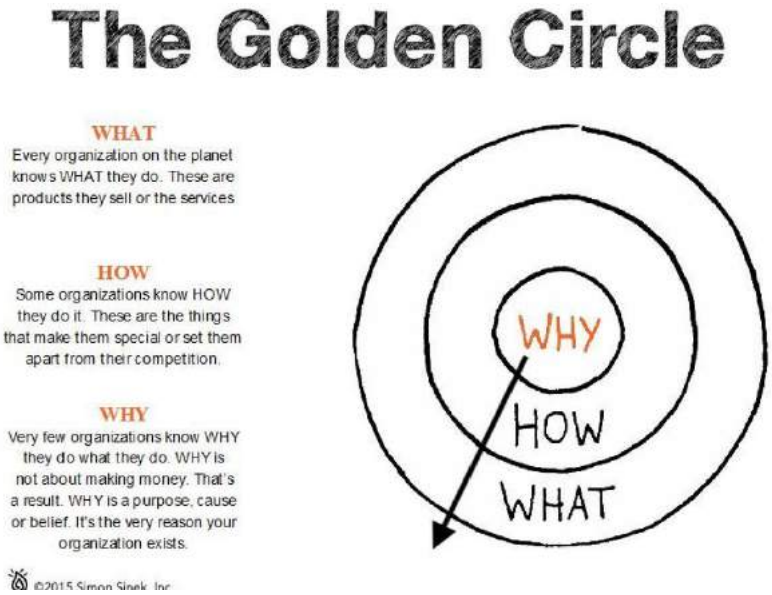
关键技能

接下来，我将重点介绍下我过去 3 年多解锁的一些关键技能。

技能一：跳出技术视角

第 1 个关键技能是跳出技术视角。不少开发都有类似【我只是一个技术，不应该参与 QA、PM 的工作】的想法，实际上也确实有人在技术路线越走越精深，并得到了认可。然而，凡

事都有个但是，很多人技术水平非常高，但一直得不到认可，这种现象更为常见。尤其对于我们前端来说，很难彻底脱离上下游、脱离业务方来完成工作，甚至除了 QA 和 PM 的工作之外，还需要向业务方“推销”自己的工具。一个更加“全面”的前端才能更加顺风顺水。那么，怎么做到更加全面呢？可以从以下几个方面入手，比如从 why 出发做事情，做业务的同学，关注业务数据、跟进线上问题，做技术产品的同学，提供出色的产品服务文档。

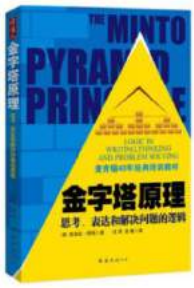


技能二：追求极致

第 2 个关键技能是追求极致，也可以理解为我们常常称赞的匠人精神。以我为例，我入职第一年就做了个在线编辑器，在完成基础功能的同时，还增加了快捷键、辅助线、自动吸附等功能，不断打磨编辑器的产品体验，因此也吸引了很多小伙伴的加入，共同把这个产品推广到更多的业务域。

技能三：结构化的表达

第 3 个关键技能是结构化的表达。俗话说的好，酒香也怕巷子深，结构化的表达，是我们在完成事情的前提下，更好地让别人（比如你的老板）get 到你的产出及价值。结构化的表达离不开结构化的思维方式，大家可以看看金字塔原理这本书。除此之外，在建立结构化思维的过程中，好的画图工具也能实现事半功倍的效果。



既然是复盘，也来做下自我批判，也是对大家的一些建议。

一些反思

反思一：No 三点一线

程序员长时间工作、加班，三点一线的奔波，对身体伤害其实蛮大的。身体是革命的本钱，要保持一个良好的健身习惯，不需要很频繁或者很专业的锻炼，其实每周 1 次的健身房养成习惯就好。保持一个良好的身体状态，工作时其实也能更有精力。

反思二：Keep learning

经常会有人说，做业务一直在 CRUD，感觉工作几年下来没什么成长。诚然，有意思、有挑战的工作确实不多见，更多的是一些琐碎的、重复的工作。所以，在工作之外，也需要花一些时间，关注下业内资讯、新闻，看看书、写写博客、参与一下 GitHub 感兴趣的项目。keep learning，保持对新技术的关注~

好书推荐

最后向大家推荐《凤凰项目》这本书，他是以小说的形式，讲述了一个凌乱的无可救药的运维项目组，是如何一步步达成最后高效且舒心的工作状态。通过该书，能学会有效的自我、团队、项目管理方法，让开发不再深陷于无尽的业务交付。

关于读书你必须要知道的 3 种方法

作者 | 爱读书的

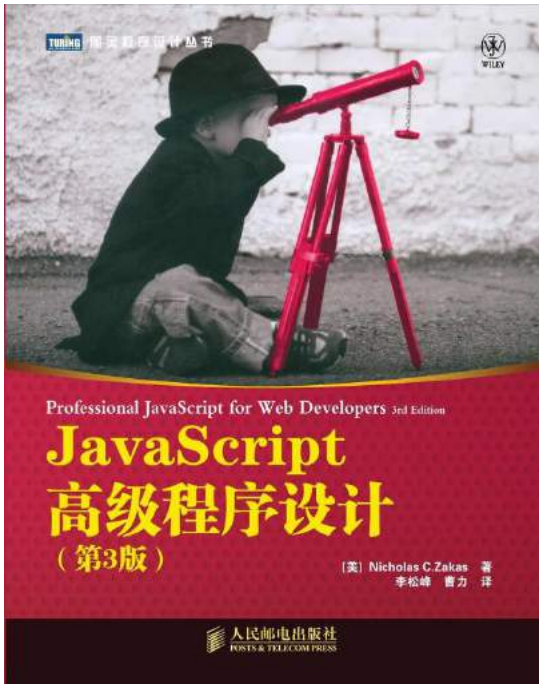
出品 | 淘系技术

来自北京的技术菜鸟张小跳提问：

最近一直在看 JavaScript 红皮书，从去年 9 月开始看的，到现在才慢吞吞看到事件部分。做了学习笔记，也做了简单的 demo，但笔记感觉基本就是在抄书，没什么压缩的空间……

我书架上囤积了不少技术类，还有思维类书籍，几乎都在吃灰。之前看过的几本，也没太多印象。

想问问，技术类或者改善技术思维的书籍，有什么好的阅读方法呢？到底怎样读书才能把读到的知识转化为能力呢？



来自淘系技术的专业大牛回答:

If I had eight hours to chop down a tree, I'd spend six hours sharpening my axe.

—— Abraham Lincoln

工欲善其事，必先利其器，读书也是如此。

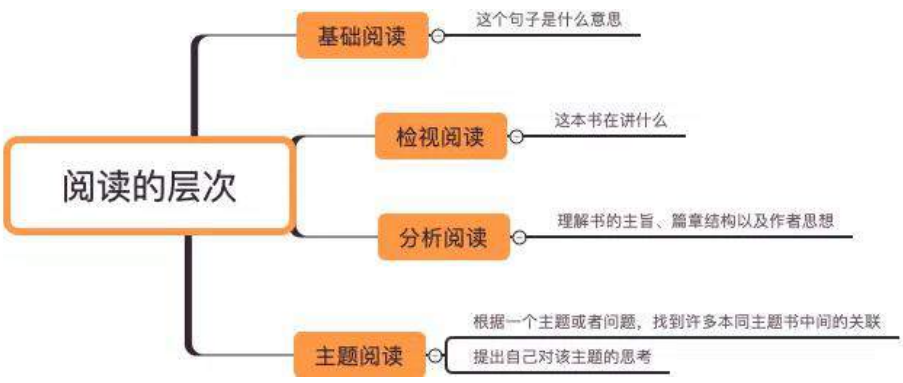
每个人其实都是最优秀的自觉经济家，当我们说畏难的时候，我们畏的其实不是困难本身，而是困难背后所暗含的时间成本。

下面，我将围绕【泛思维类书籍阅读法】和【专业技术书籍阅读法】两个部分，给你一些建议。

泛思维类书籍阅读法：抓住主题

科学、历史、哲学、物理这类书籍，能够帮助人提升思想的层次和认知的境界。

对于这样的书，阅读有 4 个层次，可以分为基础阅读、检视阅读、分析阅读和主题阅读。（如下）



围绕这 4 个层次，我们可以提炼出这样一套核心方法。

读书开始前，写下这 4 个问题。

- 整本来说，这本书在讲什么？（检视阅读）
- 这本书的细节部分说了哪些？是如何展开的？（分析阅读）
- 这本书的观点有道理吗？是整体有道理，还是部分有道理？（分析阅读）
- 这本书与你的关系是什么？（主题阅读）

每一本读完后，强制自己输出对这些问题的答案。想要从书中获取思想的进步，你就要和这本书"产生交互"。它必然不是一个消遣的过程。

特约嘉宾建议

读书首在选书。然后读书先看脉路，最后顺序看细节，看完后核心是结合自身经历的链接和思考，然后回去再去读温故而知新。好的书需要读很多遍其义自现。核心还是读书同时的思考和融会贯通。每本书里能有几点和自身经历相结合有所感悟就是值得的。

——阿里巴巴副总裁 淘系技术部负责人 平畴

无论是数学、科学、技术、管理，还是历史、文学，读书的过程，就是根据一维的文字流，在立体的思维空间中重新建构。
但我们为什么要做这样的建构呢？怎样才是值得的？还是回归到缘，缘是一种空无中诞生的关联。这些建构和自己，和自己的其他建构之间要有联系。联系是构成意义之网的丝线。
宇宙诞生于空无一物，这里一定有美好的东西，有有价值的东西，好书浓缩了人类的观察、记录和总结。错过了了它们，多可惜啊。

——淘系技术部资深算法专家 乐田

技术书籍阅读法：明确目的

技术类的书籍分支很细，原理、架构、代码、安全……对于不同技术背景、不同问题目的的你来说，实用性程度不一样。记住以下 3 种方法，能够帮你解决大部分阅读难题。

1. 设定读书目标，带着问题选择和阅读

闭上眼睛，想象一下周围有多少橙色物体，心里默数一下，睁开眼睛，是不是感觉双目所及的橙色物体一下子都飞入眼帘呢？

这就是主动意识的力量。

技术类的书籍，就是要带着问题找答案。

首先，要围绕你的目标，寻找适合的书单。

其次，带着你的关键疑惑，用倍数阅读法找你要解决的问题，可以跳读和略读。在快速阅读中记录核心关键词，每 10 个关键词可以归纳成 3-5 个重点或一系列动作。

特约嘉宾建议

如何围绕目标选书单？假设，你遇到的问题是——【如何设计一个秒杀系统？】，书单的列表可能要包括：

原理侧：CAP 原理

架构侧：静态文件分离、CDN 设计、Redis 缓存原理、数据库乐观锁设计、限流方案设计、负载均衡策略

代码侧：库存拍减、扣减、和预扣，关单回补设计，库存预热

安全侧：如何防黄牛，防恶拍，防 DDOS 攻击

——淘系技术部技术专家 殷浩

我在技术类书籍的阅读中，关键重点是识别出，作者提到的哪些点是我没有涉猎到的，或者是我涉猎过，但作者提出了不一样的看法和解法。比如说我们做交互行为的场景，可能会遇到作者提出某个按钮做一些微动，带来创新的动画效果，这种东西我都会比较感兴趣，并且去做验证。

——淘系技术部前端技术专家 大漠

2. 设定行动目标，学以致用良好循环

知识付诸行动才有转化为能力的效果。

有些同学疑问为什么自己边看边敲，做笔记，实现阶段性的小效果，最后还是如同水流过的石头一样，风一吹，太阳一晒，什么都没有留下。

关于如何设置实践和行动，我推荐你分这样 2 步走。

首先，输入-笔记-实践-完善笔记、整理沉淀

真正有效的学习行动不是一个【对着书敲一遍就=实践完毕】的过程，这也是大多数技术同学的误区。

- 根据你的目的和问题，摘录关键词和行动，形成简单的笔记。
- 利用新学的知识在工作中展开实践，或者独立做一个小的项目。
- 过程中记录下自己实践的感受和踩坑的关键点。
- 回过头来完善自己的笔记，并最终整理输出。

然后，组团实践与输出，获得不同的见解

你有一个思想，我有一个思想，当我们在一起时，就有了 2 个思想。

特约嘉宾建议

在我们淘系现在的 MNN 团队，我们是这样展开读书学习的。列举一个时下热门的技术话题，团队里每个成员分头去阅读和搜集资料。
这个过程中，我们不仅看书，也看最新的博客和资讯，关注业界大佬对这个话题的评价和看法，根据自己的理解，形成自己的观点。然后同事会定期聚集在一起，共同讨论对这个话题的看法。
即便过程中我们看的是同样的资料，在不同的视角下，依然经常得出不同的结论。
这种碰撞和交流，确实给我们带来了更深的技术理解。

——淘系技术部高级技术专家 玄裳

3. 对于特别艰森的技术论文或内容的阅读方法

还有一类内容，属于你拿起就想睡觉，每个字都认识，每句话都看不明白。

在时下大热的机器学习、深度学习、神经网络等领域表现尤其明显。

这里我推荐诺贝尔物理奖获得者，理查德·费曼（Richard Feynman）在他的自传里提到的阅读方法——

- 仔细审阅这篇论文的辅助材料 （supporting material），直到掌握了相关的知识基础、足以理解其中的艰深想法为止。

面对一个晦涩难懂的主体，脚注、尾注、参考文献等，不放过任何一个细小的东西。你应该分而化之，切成小知识块，再逐个对付，你最终能填补所有的知识缺口，否则，这些缺口将阻挠你理解整个想法。

找一篇该领域的核心文章，顺蔓摸瓜研究完所有的辅助材料，你也会对这个领域留下整体的理解和印象。

~~4. 一本万利的神仙级读书方法——量子速读法~~

“遇事不决，量子力学。”

闭上眼睛，打开感知，2 分钟内看完一本 10 万字书籍，还能妥妥复述出来。

没有什么书，是量子速读法不能解决的。

只要你真敢信。

（友情提示：第 4 种方法高深玄妙，非专业人士请勿模仿）

德国哲学家尼采，在提及正确的阅读方法时说过——

It teaches how to read well: i.e. slowly, profoundly, attentively, prudently, with inner thoughts, with the mental doors ajar, with delicate fingers and eyes.

单靠阅读是无法让你形成自己的思想体系、价值判断和技术能力的，持续思考和输出才是致胜之道。

希望你在 2020 年的这个读书日确有所获，把你那一满满书架真正消化吸收吧~

阿里大牛推荐书单

阿里巴巴副总裁 淘系技术部负责人-平畴

书单:

- 《孙子兵法》（孙武）
- 《刷新》（ [美] 萨提亚·纳德拉 ）
- 《商业的本质》（ [美] 杰克·韦尔奇）

杂志:

- 《哈佛商业评论》

新闻:

- bloomberg

阿里巴巴淘系技术部资深算法专家-乐田

书单:

- 《深度探索 C++对象模型》（ [美] Stanley B·Lippman ），不懂对象模型就不是好的C++工程师。
- 《支持向量机导论》（[美]克里斯蒂亚尼尼），一本小册子，伴随了我上下班的地铁两个月，翻了三遍。基本的概念和思想永不过时。
- 《罗马人的故事》15 卷本，人类童年时期的热情、理想和勇气，让我有遇到同道的共鸣。

- 《QED》（ [美] 理查德·费曼 ） 给我一个新的视角来看量子世界，作者最大限度地保证了内容的严谨科学。
- 《枢纽》（施展）探讨了何谓中国，从古至今，从国内到世界。从内部视角介绍了中原、草原、高原、海洋的统一联系，从外部视角讨论了中国作为连接海洋秩序到大陆秩序的桥梁的可能。
- 《债务危机》（ [美] 瑞·达利欧 ），宏观经济关乎每个人的生存和发展状态，它可能揭示了经济繁荣-衰退周期的最可靠的底层规律。

阿里巴巴淘系技术部高级技术专家-玄裳

书单：

- Effective Java, Joshua Bloch
- Java Concurrency in Practice,Brian Goetz
- Java 方面非常经典的两本书。

- Deep Learning, Ian Goodfellow

如果要了解深度学习，这本是非常系统客观的，自认为作为门外汉来入门很合适。

阿里巴巴淘系技术部前端技术专家-大漠

《图解 CSS3：核心技术与案例实战》作者

书单：

- 《兼济天下：用户无障碍体验的可访问性设计》
- 近期正在阅读的一本，在无障碍方面讲得不错。

极简而高效的沟通管理法

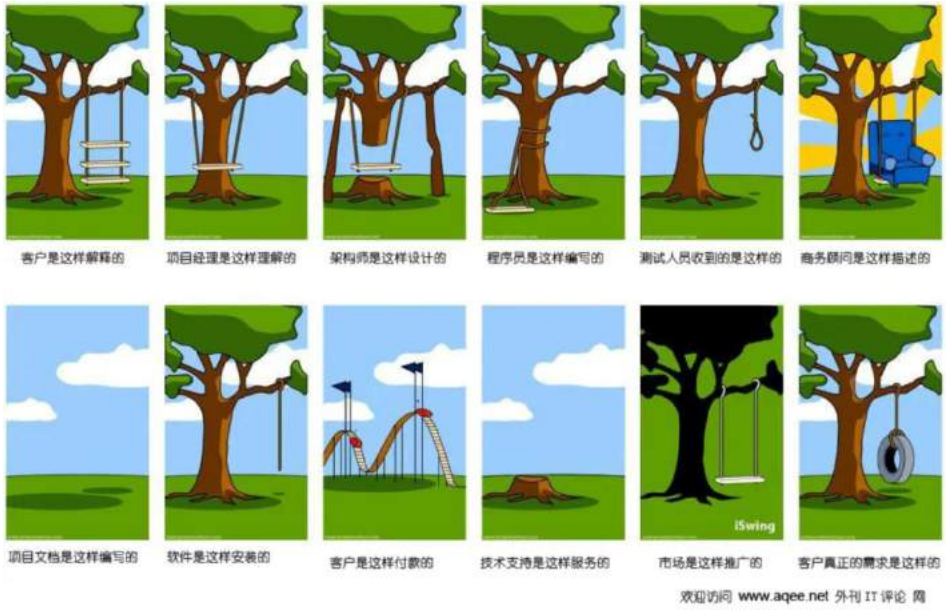
作者 | 鲁佳(鹿迦)
出品 | 淘系技术

世界上有两件最难的事：把别人的钱装进自己的口袋；把自己的思想装进别人的脑袋。

为什么沟通那么重要

谁都知道在工作中沟通是非常重要的，那大家有没有真正想过为什么沟通这么重要呢？首先我们想想看，在你的日常工作时间分配上，“沟通”所用的时间占比多少呢？日常的会议、在钉钉上的讨论、处理邮件等等，反正我个人是远远超过了 50%的。这就和管理学上的一个著名理论很契合了：

管理者 50%以上的时间用在了沟通上，但同时工作中 50%以上的障碍都是在沟通中产生的。这就是“双 50%理论”，这直观的反应了不论是在占用时间上，还是对工作的影响程度上，沟通都是至关重要的。



为什么团队里开会大家总是很沉默，有想法的同学也不吭声？上面这个图大家应该在各种不同的渠道都看到过吧，相信大在第一次看到这个图的时候都会和我一样会心一笑吧。

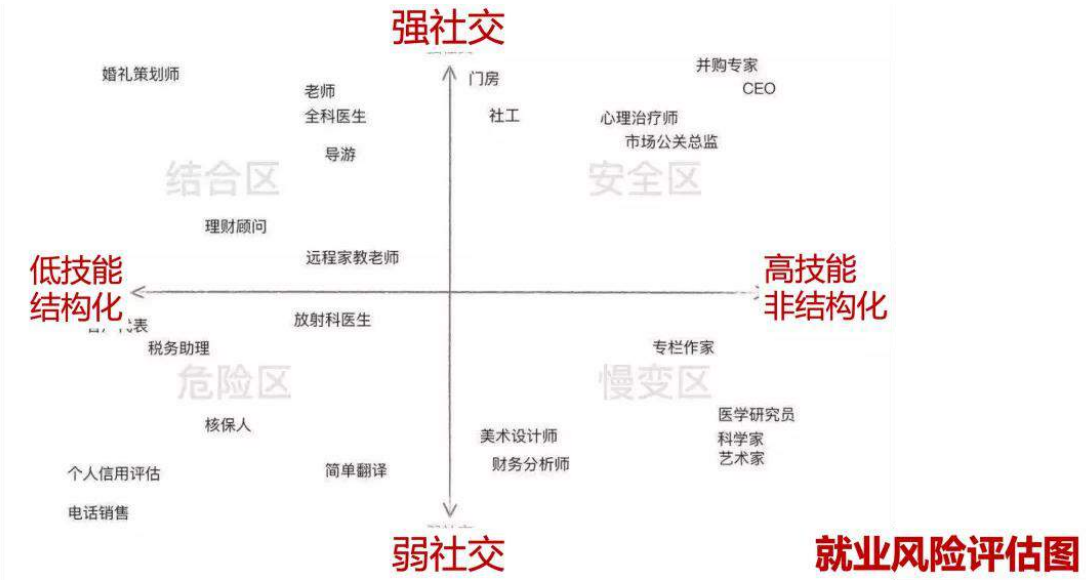
为什么这张图能有这样的效果呢？因为这个图所表现出来的问题真的非常好的影射了我们日常实际工作的情况：由于沟通不到位的问题导致一个产品在不同的角色的理解上造成了这么巨大的差异。那么在像阿里这样的互联网公司，工作量常见的沟通问题都要哪些呢？这里做一些简单的列举：

- 为什么推动兄弟团队配合这么难，他们配合的进度完全不可控？
- 为什么工作布置给了下属，但是最后又都堆积到了我这里？
- 为什么会议讨论留下来的 action 执行起来总是拖拖拉拉完成不了？
- 为什么老板最近的消息经常已读不回？
- 为什么整天被钉钉消息淹没了，个人的事情只能在夜深人静加班做了？
- ……

这么多的沟通问题，那沟通问什么这么困难呢？这里从最常见的情况来看，特别是在 1 对 1 的沟通中，主要有三个主要的原因：

既然沟通的问题很多，导致问题的原因也非常复杂，那我们的出路在哪里呢？是不是解决办法就是标准化我们的工作，然后通过标准化、流程化来让我们的工作减少沟通的需求，然后大家都可以在自己的工位安安静静写代码、写方案、这样就万事大吉了呢？答案是否定的。

我们来看这个象限图，这个是李开复博士在对未来 AI 的设想中提到的一个模型：

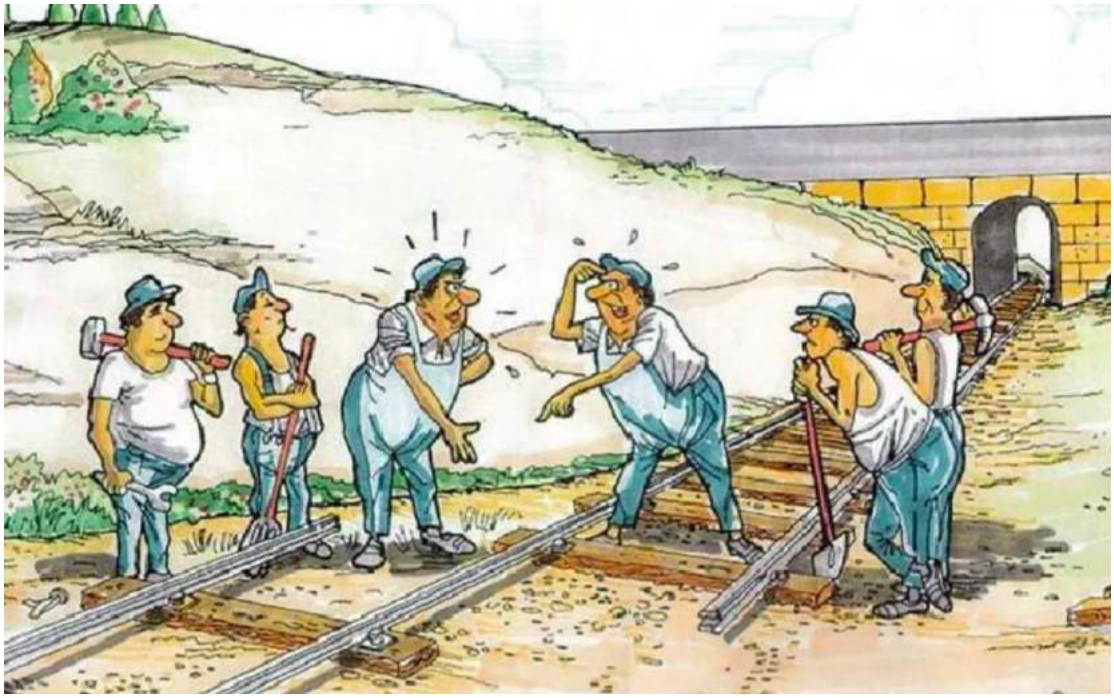


可以看到，在区间图的左下角的岗位对应的是低社交需求，并且是结构化的领域，在这样的领域的岗位，未来 5-10 就会被 AI 机器人所取代。

如果我们的工作能够简单的通过定义标准化的接口做到按流程执行，从而避免或减少沟通，那我们的工作的价值就会降低，从而最终被机器取代，就像流水线上的工人正在大量被智能机器人所取代那样。

我想说的意思不是说为了体现工作的价值，要故意提高工作的沟通成本，相反，我们的确需要通过归纳总结梳理标准话的流程、甚至是工具化的手段来降低一个领域的沟通，但我们同时工作在找更复杂、更需要去沟通的场景中，去创造更大的价值。所以我们要做的是保持竞争力，持续提升沟通能力。这个才是大家每个人要持续努力的方向。

沟通的基础

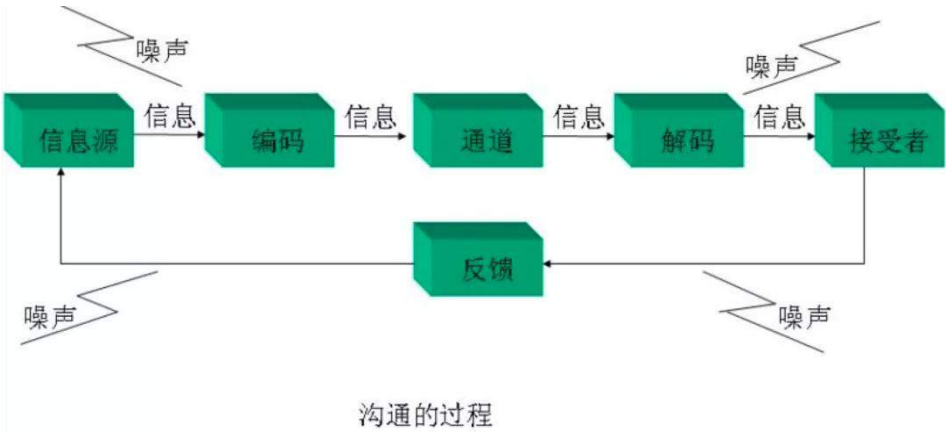


有一个西方的谚语是这么说的：世界上有两件最难的事：把别人的钱装进自己的口袋；把自己的思想装进别人的脑袋。就像图上的这个场景一样，在沟通遇到问题的时候，大家往往都会觉得自己是对的，都想要说服对方，但有都总是困难重重。

沟通失败的原因总是多种多样，要么是表达上的问题，要么是倾听和反馈上的问题，要么是环境上带来的问题。列举下来，一般的情况有：

- 缺乏信息或知识
- 没有说明重要性
- 只注重了表达、没有注重倾听
- 没有完全理解对方的话，以至于询问不当
- 没有注重反馈
- 时间不够
- 不良情绪
- 职位的差距、文化的差距

首先，我们来一起了解一下沟通模型。沟通是从信息发送者出发，到信息接受者收到消息，然后给出反馈到信息发送者的一个完整闭环。



沟通的过程

这里的编码，可能是文字，也可以是声音，甚至是手势、表情；通道可以是钉钉、邮件、视频、也可以是面对面；各种外界环境可能带来噪音，比如心情不好，或者大家背景不同、缺少共同语言，并行沟通的影响等等。

可以看到，一个最小闭环的沟通过程，因为涉及多个环节，其实形成了一个较长的链路，链路一旦长了，在链路上的各个环节都容易出现问题，最终就会将问题层层放大。这里就必须给大家介绍一下沟通漏斗的理论了。

这里必须要知道沟通漏斗的形成过程。举个例子，比如一个开发的 leader，发现最近项目开发中引入的问题比较多，产品好几次遇到严重的线上质疑，于是他决定开个会整改，晚上睡不着觉心里想怎么解决呢？

想到了 10 个方面（100%）的问题的，需要与下属员工讲。当第二天站在众人面前开会时，由于没休息好并且紧张，想到的 10 件事情，只说出了其中的 8 件，漏掉了 2 件，也就是只说出了 80%。而这 8 件事在当众讲的时候，由于场地的限制、人员的多少等关系影响，造成员工真正听到的是有 6 件事（60%）。

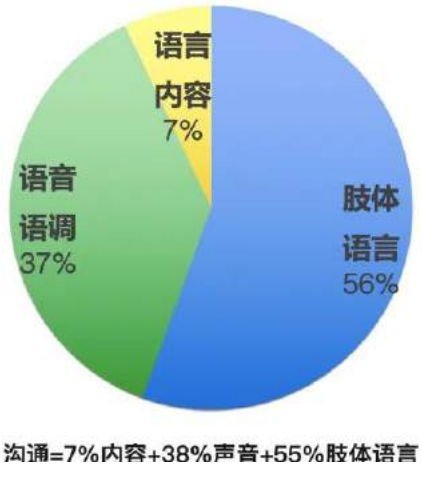
又由于员工水平、知识背景的差异，我们的技术 leader 语言表达的高低、语言口音方言等实际情况，真正被别人理解、消化的东西大概只有 40%。等到这些人去落地执行时，由于每件事情与个人关系的大小、执行的意愿强弱等因素，造成到最后得到执行的已经变成 20% 了；虽然这个比例比较夸张，实际并没有这么严重，但是这个沟通造成的漏斗确是实实在在存在的，我们只能想办法将他变得影响越小越好。

那么到底有没有办法减少或消除这个沟通造成的漏斗现象呢？我们从说、听、问三个角度来看一下。

沟通基础一：表达

沟通的基础，第一条就是表达，就是发送信息。我们的马老师是公认的演讲高手，他的表达能力就非常强。

在表达上，除了我们大家比较了解的，需要有易懂的语言、清晰地逻辑，我们还需要注意什么呢？看马老师的形象我们就清楚了，那就是语气、表情、肢体动作这些。



沟通=7%内容+38%声音+55%肢体语言

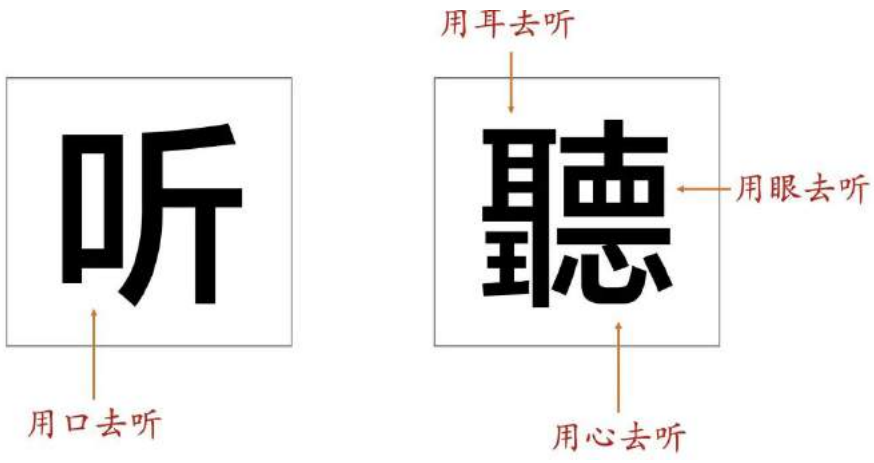
在沟通管理领域，有一个非常著名的理论，“沟通 = 7%内容 + 38%声音 + 55%肢体语言”，这充分说明了身体语言在沟通过程中的重要性。这里，给大家在沟通表达时的身体语言提供一些建议：

- 语言和身体语言必须保持一致
- 设法一开始引起对方的注意
- 如果必要，需要重复、改述
- 用对方熟悉的语言
- 要：正面面对、点头、微笑、身体前倾、目光接触
- 不要：双手抱在胸前、眼神游离、干别的事

再举一个我们日常中最常见的例子。日常我们用钉钉做沟通比较多，同样的一句话，如果用不同的语调会表示不同的态度、表达不同的意思（如“这个需求提测了吗？”，是询问还是催促？）。所以我在钉钉上询问别人类似的问题时，总担心被误解，一般都会增加一个笑脸的表情包，不知道大家有没有这样的做法呢？^_^

沟通基础二：倾听

沟通的第二个基础就是听。沟通的关键往往是接收者、听者，而非说话者。越是成功的人，往往越是擅于倾听的人。



相对于简体字（用口听），繁体字的"听"更能体现倾听的精髓（看来我们古人还是的智慧还是很了不起的）：

- "耳王"。既然是听，最重要的器官无疑是耳朵，所以听的要义就是以耳为王，带着耳朵来，把倾听放在第一位。
- "十四"。其中的"四"可以看成"目"，也就是眼睛。听不光用耳朵，还要用眼睛。对方的眼神、表情、肢体语言，都包含着丰富的信息，是与对方提供的资料或实物相配合的
- "一心"。要达到沟通的目的，收到最佳的效果，必须要用心，而且要一心一意。是用心去听讲，用心去观察，用心去提问，用心去思考。

倾听也分成几个不同的层次：

- 听而不闻
- 假装倾听
- 选择性倾听
- 专注的倾听
- 设身处地的倾听

其他第五层和第四层的区别是，专注的听还是为了对听到的内容作出反应，但设身处地的听是为了了解对方，而不是为了做反应，目的是通过交流了解对方的观念、感受，真正做到同理心。

C 沟通基础三：提问

沟通的第三个基础就是提问，会提问的人更是沟通达人。擅长说的人，沟通的主要时间都是他在讲，不用担心冷场，而且他可以讲得很精彩。典型的就是培训老师；擅长问的人，和他说话时他说的不多，但往往提出一些发人深省的问题，让你娇躯（或者虎躯）一震，打开一个新的世界。所以，善于提问的人才是正在引导沟通方向的人。

方式	场景和要点
封闭式提问	节省时间，但可能收集信息不全，并使气氛紧张
开放式提问	信息全面，但浪费时间，且沟通方向不易控制
引导式提问	假设场景，能了解对方的思维能力和经验程度
反问式提问	扭转被动局势，拉动对话走上你的频道

一般提问的方式有四种，不同的场合环境和目的需要采用不同的提问方式：

- 封闭式：你同意这个方案吗？我们能按期完成这个任务吗？
- 开放式：你有什么好的建议吗？
- 引导式：如果资源的问题解决了，你第一步打算怎么做？
- 反问式：如果你一味地回答对方问题，就会很容易陷入被动局势。如何让谈话走上你的频道？反问式提问便是一个高招， 例如："我想问一下，这个需求开发排期是怎么样的？""那业务的运营节奏是怎样的呢？"

团队内沟通和跨团队沟通

接下来给大家介绍三组最常见的沟通场景，这些沟通场景也是日常工作中最容易遇到困难的地方。首先介绍团队内的沟通和跨团队的沟通。

团队内沟通

团队内的沟通是大家每个人首先要面的的场景。在团队里，一般来说有 3 种沟通的模式。

链式　　轮式　　网式

- **链式**
  - 严格遵循正式的指挥链和沟通链，信息线性传递，如在军事化的组织里就比较常见；

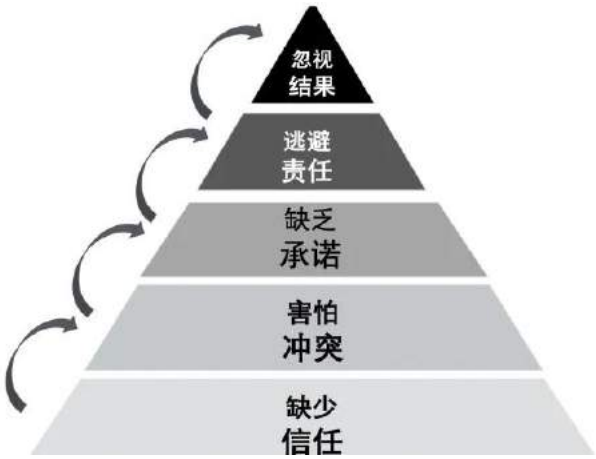
- **轮式**
  - 轮式沟通仰赖核心人物作为所有成员沟通的引线和中心节点，可以激发强而有力的领导人物出现；

- **网式**
  - 允许所有团体成员积极主动的沟通，所有成员均能自由的贡献自我，没有信息的中间节点，也没有明显的领导者角色；

这里，我们对这三种模式做了一个对比：

	模式		
影响	**链式**	**轮式**	**网式**
速度	中等	快	快
准确性	高	高	中等
领导人数的出现	中等	高	无
成员满足感	中等	低	高

在我们像我们阿里这种互联网的公司里，如果没有特别的原因，我们会更倾向引导团队超第三种模式，“网状”的沟通模式发展。推荐网状的沟通模式并不是说我们对准确性的要求不高，相反，因为我们有很多非常有效的工具，能够保障我们在网状沟通中仍然保持较高的信息准确性，比如语雀、Aone、钉钉等。在团队沟通里，我们就不得不提团队沟通协作的五大障碍。



- 首先在团内里经常出现的问题是"缺少信任"，这个在新组建的团队里特别常见。
- 缺少信任之后就会给团队带来第二个问题，那就是"害怕冲突"，最常见的就是在会议上大家都沉默不发言。因为缺少信任，大家不知道自己不同的意见会导致别人什么样的反应，为了安全，不发言是最稳妥的。
- 害怕冲突的团队自然会带来第三个问题"缺乏承诺"。可以想象，如果自已不同的观点没有表达出来，那么团队最终达成的结论，我是不会有太强的承诺的，因为自己对这个结论心里并不认同。
- 缺乏承诺继续会导致问题，那就是"逃避责任"，没有承诺的结论如果后续导致了不好的结果，团队成员自然不想对这个结果承担责任，他会心里想，这个本来就不是我希望的做法，看，果然出问题了吧。
- 逃避责任的最终就会团队成员"忽视结果"，如果团队成员自己团队的结果都无感了，那么这个团队也就名存实亡了。

如果大家对团队沟通协作的五大障碍有兴趣，希望更深入的了解，推荐大家看一本书《团队协作的五大障碍》。

那问题来了，对于团队里沟通协作的五大障碍之首的"信任"，要怎么培养呢？在团队里建立信任我们有很多办法，比如从小的来说我们要加强团队成员的相互了解，在阿里，我们常见的有破冰（想必大家都经历过）、team building 等等，还有一个建立信任的说法是"一起同过窗,一起扛过枪"，这样大家就比较容易建立相互信任的关系。

不过建立信任也不一定是这样刻意，或者一定要经历历史性的大战役才能培养，其实"信任"是在一次次达成并保持的小的约定的基础上建立起来的。比如大家约定的一个文档产出时间，大家约定的一个会前准备工作，只要类似这种小小的约定能持续达成，大家间的信任就能有效的建立。

团队里的冲突往往是破坏团队沟通氛围的重要因素，不希望团队里有成员的冲突往往是大家共同的想法。不过这里要想和大家聊一聊的是，冲突对于团队的沟通一定是件坏事情吗？

答案是否定的，团队可以从健康的富有成效的分歧及处理中得到成长。对团队不利的是"破坏性冲突"，但对团队有价值的是"建设性的冲突"，我们一起来看看这两种冲突有什么区别：

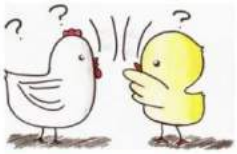
建设性冲突	破坏性冲突
团队都积极参与来探寻各种各样的可能性。	大家更感兴趣的是怎么样赢得争论。
每个人都参与到讨论中来。	由少数人主宰，其他游离在外。
讨论关注在一些事实的因素上： 5W (who, what, when, why, how, where)	对话中充满了观点的对抗和人身攻击。
讨论持续关注在大家有争议的观点上，不会延伸到别的话题上。	讨论持续从一个话题转移到另一个话题上。
组员都相互鼓励他人的观点输入，不论他们是否认同这些想法。	组员们在讨论中争夺控制权，并且压制别人的发言，对不同的观点就吹毛求疵。。

在团队里，阻碍健康冲突的原因一般有：

- 缺乏信任、尊重和理解
- 过强势的管理风格
- 团队成员不稳定或跨地域

跨团队沟通

在跨团队沟通我们经常遇到的各种问题，我们先简单给大家罗列一下：



专业有gap，缺少共同语言

目标达不成一致（部门墙）

进度不透明

职责不清、问题推诿

- 专业有 gap，缺少共同语言。比如业务开发的团队依赖中台的能力，如果业务开发团队不了解中台产品的当前现有的体系，讲不清楚对中台需求的需求，就会出现这种情况，最终导致沟通意愿受损，沟通出问题。
- 目标达不成一致（受部门墙的影响），比如你计划一个战略级的项目规划，但是需要其他部门来支持你，在前期调研的时候很配合，真正开始实施了，你也许发现他们的态度发生了变化，他们不打算真的做较大投入了，因为这个和他们团队的目标不匹配。
- 进度不透明。和兄弟团队约定好了交付的时间，但是中间过程没有沟通，完全是个黑盒，到了约定的时间快到了才发现很多东西还没有准备好。
- 职责不清、问题推诿。沟通中经常遇到大家因为分工不清导致工作相互推诿，一些灰色的工作找不到负责人，一旦灰色领域出了问题，又没有人愿意来背锅。

那面对这些问题，我们该怎么解决了？这里给大家提供一个跨部门沟通的原则：

- 首先是要大家一起确定目标：一个共赢的目标，不能只是你的目标，一开始就要设定一个大家都能实现自己价值的目标，这样对方才有意愿和你合作。
- 第二个就是要明确分工、职责、时间、责任人等，做到"亲兄弟、明算账"，俗话说的"丑话说在前面"也是这个意思。
- 第三个就是一开始要确认好沟通机制，做到信息互通，比如周会、周报等，确保过程中信息透明。
- 第四个就是在问题的处理上要有责任、有担当、同理心，不能是有好处大家都凑过来，一旦出了问题大家就翻脸不认人，和人一样，团队也要注意信誉。

因为跨团队沟通往往双方不像在团队里那样，双方可能不是很了解，没有太多的信任感，所以当双方刚坐下来，还没开始沟通，大家的座位方式已经极大的影响了沟通的结果。这里给大家介绍一下几种不同沟通座位可能带来的影响。

向上沟通和向下沟通

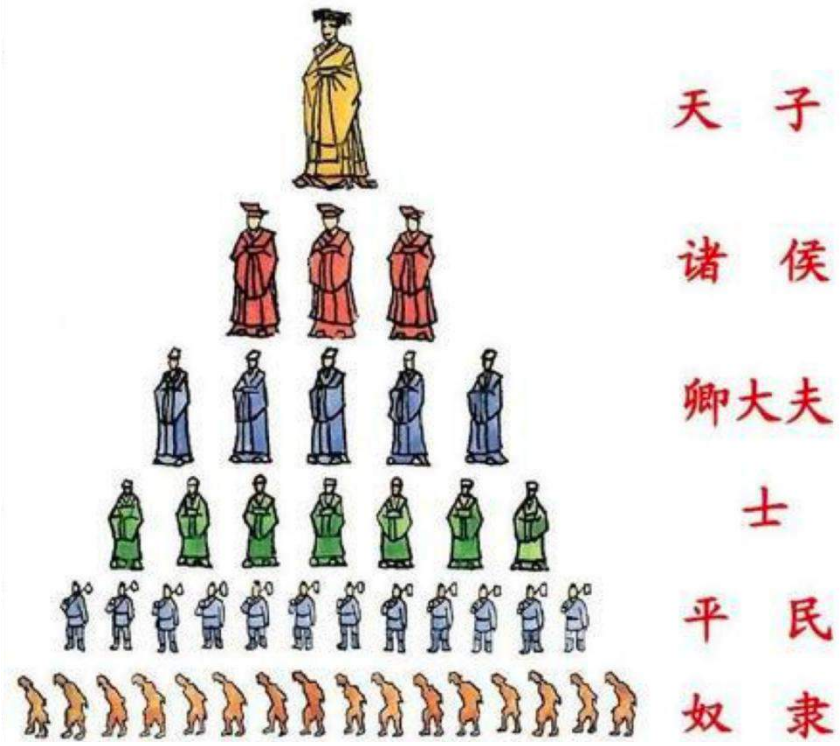
第二组经常遇到问题的沟通场景就是向上沟通和向下沟通。

向上沟通

讲到"向上沟通"，我们就在提一下"向上管理"，先看一下这两个概念的解释。

- 向上管理，为了给上级和自己取得更好的结果而有意识地配合上级一起工作的过程，可以说，这是让上级改变的过程；向上管理不是领导上级，而是通过向上沟通、向上协调去影响及引导上级的看法与自己协调一致；
- 向上沟通，向上沟通是向上管理的主要手段之一，一般指和上级有目的性的沟通行为，从而向上传递必要的信息或影响其决定；

向上沟通是一种"有目的性的"向上管理行为，如果是那种在 teambuilding 时候很放松的闲聊，我们暂不归类到向上沟通里面。那向上沟通的难点是什么呢？



- **传统观念**
  - 中国人已提到向上管理和向上沟通，可能就各种各样的问题都出来了。就如上图，这个是西周的"天子统三公、三公治诸侯"的一个金字塔，这就是中国过去传统的一个很明确、很强烈的等级观念，比如常见的"官大一级压死人"；在中国，这个是一个历史文化的传承，即使到了今天一个自有民主的社会环境，即使在像阿里这样开明的互联网公司，这中固有的观念还是会潜移默化的影响我们。

- **缺乏主动**
  - 缺乏主动性，一定要等到出了大篓子，兜不住的时候才想到要去通报主管。信息不流通，会让主管没有安全感，会有失控感。永远不要让主管从别人那里知道本应该你告诉他的情况。

- **信息片面**
  - 你想通过沟通去影响你的老板，如果你发现他现在有一个想法不对，你希望引导他按照你的思路去决策；这个背后还有一个可能性是你获取的信息没有你老板多，你做出的判断是片面的，甚至是错误的。

- **直线思维**
  - 直线思维，比如上级说："小明，你这个事儿做得不对"，"我这事哪儿做得不对了！"。那这种沟通方式肯定就没办法谈下去了，是吧，如果这样沟通下去的结果是什么？可以想象，大家可以掏出手机看阿里味儿了。你可以不认同他的观点，或者他处理问题的方式，但是一定不要在公开的场合当面怼。即使在当场觉得那个方式行不通，处理上存在偏差，也可以在会后，私下沟通，在执行过程中进行修正。

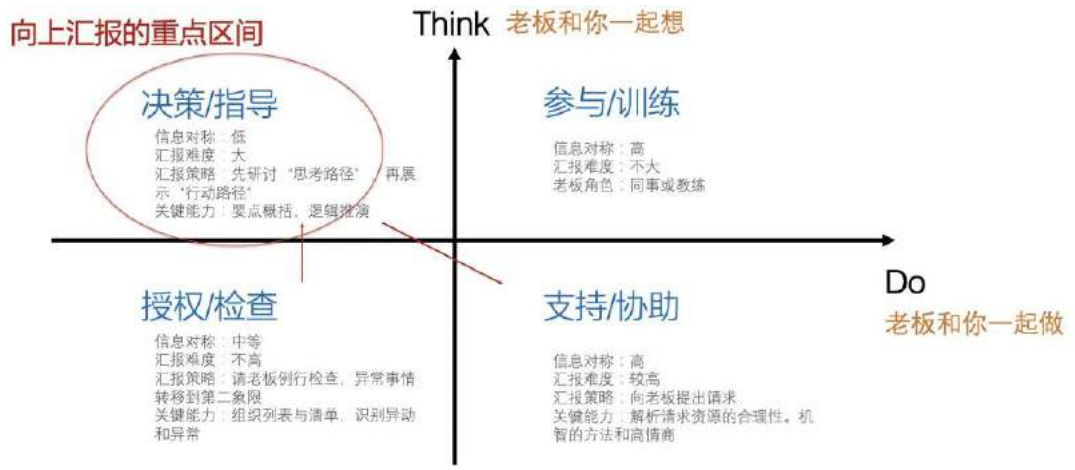
那么向上沟通的常见问题是哪些呢？

- **有问题、有疑虑不敢沟通**：
  - 领导展现出一种事事有反馈、积极复盘的认真工作态度，留下良好的第一印象。
  - 执行时遇到问题，把问题及时反馈，等于把责任分担了出去，提高解决问题的成功率。
  - 对于自己来说，主动汇报也能使你理清自己的工作，锻炼自己解决问题的能力，提升效率。
  - 主动汇报，让老板信任，因为不汇报给领导带来失控感。主动汇报，既可以让你的工作"可视化"，也确保执行方向正确。这样做的好处有：

- **啰里啰嗦说不清问题和影响**：
  - 给老板汇报一定要提前准备，思路清晰，表达清楚，提高效率。关于怎么给老板做汇报，后面还还展开讲一下。

- **只说问题不给方案**：
  - 充分思考，提前准备好方案和建议，让老板做选择题而不是问答题。这个不仅提高了和老板沟通的效率，更重要的是要让老板看到你的思考，做到为老板分担。

讲到向上汇报，那我们就需要先来讲一下向上汇报的思考框架，通过这个思考框架，你能判断当前的内容是否需要汇报，已经用什么策略来做汇报。



第一个象限和老板一起进入重要级别非常高的项目。此时，老板就是你的同事。一起策划一起干活。这个象限里，老板是完全的“参与”。你们之间信息是对称的，这时，重要的汇报形式可能都消失了，因为你们随时都知道对方的信息。

第二个象限，老板的角色要给你“指导与决策”。这里老板很关心，但具体进展和细节他不清楚，这里的内容往往是汇报中最重要的，也是向上汇报能力的重点。重点汇报里很有一个可能目标是有困难需要老板来支持的，比如资源不够，需要老板协调，你直接告诉老板需要他做什么就好了（这里就转移到了第四象限）。

第三个象限都是老板授权给你的事情，是你的职责所在。虽然信息在这里也有不对称，但是老板在宏观面上会知道你的内容（甚至他当年还干过你这个工作），或者内容的重要性不能进入老板的“管理仪表盘”（不涉及他要做的高级别管理决策）。这部分内容只要表达：你在按部就班地在如期进行就好，老板只需阶段性的检视一下就行，不是汇报的重点。常见的误区是：这部分内容总是错误地摆上 PPT，向老板事无巨细的读一遍，大家都很沉闷，成为最没水平的汇报。这个区间里如果有异常情况，异常情况需要转移到第二区间处理。

第四象限的内容，其缘由和理由都无需论述，老板和你已经达成共识，信息是对称的。但是，这里有下属难以完成的任务，或者超出职权的任务，或者是新范畴的任务，最常见的是下属需要领导帮助解决某种资源。这里老板的角色是“帮助与支持”。

前面说到要主动汇报，那汇报最好的时机是什么时候呢？这里提供了 4 种最基本的向上汇报时机，大家可以参考：

时间点	作用	建议
制定工作计划后	让老板了解你的规划，得到支持和指导	请求支持要说明理由
工作有一定进展时	让上级了解工作情况，避免出大错	尽量简明扼要
工作中出现意外时	让上级知道意外的原因以及后果	提供解决方案，将功补过
工作完成时	汇报整体工作情况，体现自我价值	要有亮点，避免流水账式罗列

现在要介绍一下向上汇报的方法了：

- **内容精简扼要**：要相信老板都是看过成千上万张 PPT 的，所以他们大多数时候都能够一眼看出 PPT 中所要表达的意思，因此 PPT 的信息需要精简扼要。同时需要使用大量的数据来佐证自己的观点，尽量从客观的角度来陈述。
- **直奔主题**：如上面的描述，老板的时间安排极其紧凑，甚至会在汇报过程中由于突发事件而离开，所以汇报一开始就要开门见山的陈述自己的观点或要求，然后再罗列理由和证据，最后再重新陈述一下自己的观点即可。以上即为"电梯游说"的 PREP 结构：Position（观点）、Reason（理由）、Evidence（证据）、Position（观点）。
- **做好充分准备**：汇报的内容也需仅有数页 PPT，但是很多时候领导都会顺藤摸瓜的询问一些相关问题。所以作为汇报人，需要同时将相关的信息都要提前了解清楚，以防被问的哑口无言，不但会影响汇报进度，而且会被认为不尽职。
- **掌握 10/30 原则**：30 分钟的时间计划中，汇报时间最多占用 10 分钟，剩下的 20 分钟留给讨论时间，往往能够产生价值的信息会在讨论时间内产生。

PREP 汇报方法

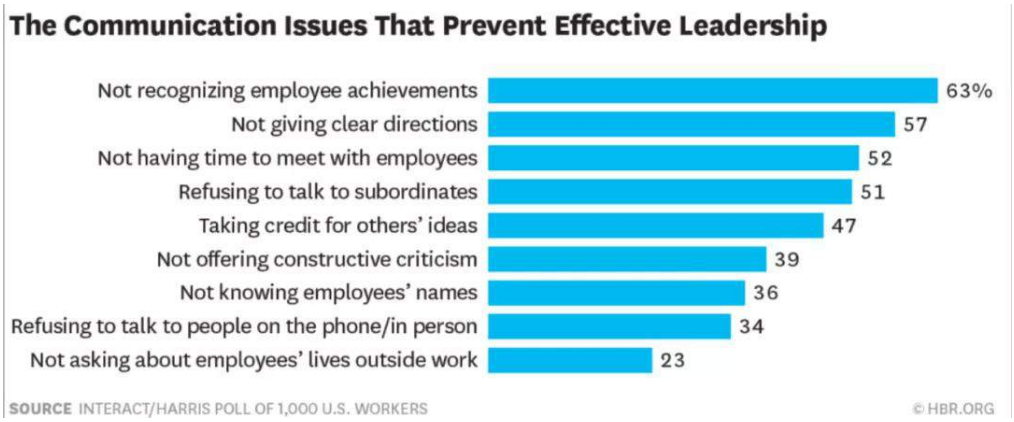
出自麦肯锡的 30 秒电梯理论。麦肯锡要求公司员工能够在最短的时间，把想表达的事情说清楚、讲明白。因此，作出了 PREP 的汇报结构，要求汇报者能够站在一个决策者而不是执行者的角度上汇报工作。其中由四个部分组成：

- **观点**：就是首先用一两句话说清楚你想要表达的核心观点，包括：希望领导原则上同意、定下一步计划或者尽快看看你的详细材料等等。
- **理由**：你需要仔细分析你的汇报材料，凝结成不超过三个理由，然后试着用简洁而有力的语言表达出来。
- **证据**：主要是说明汇报重点中包涵的具体细节，例如：具体现象、数据和效果估计等等。
- **重复观点**：最后，再一次重复你的要求

向下沟通

向上沟通有很多困难，那是不是就意味着向下沟通就简单了呢？你的下属都向你汇报，那还有什么难度吗？

是的，有带团队的同学都能体会，考虑到你每个下属的情况都不相同，向下沟通并不容易。并且如果不考虑到这些差异，糟糕的向下沟通对管理者和团队来说都是一个灾难。“哈佛商业评论”就有一个调研表明：91%的员工表示，上级糟糕的沟通会损害其领导力！



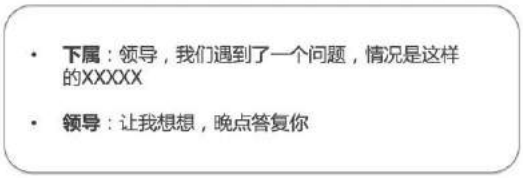
那和下属沟通的技巧都有哪些呢?

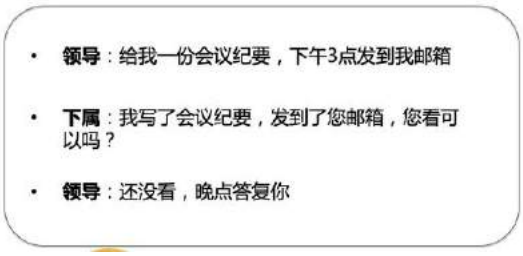
- **发布指令目的清晰，目标明确**
  - 说明任务的重要性，明确任务的 SMART 目标；如果说得不好听，一个含糊的目标（如我们要重视用户体验），你的下属怎么知道你是不是一时兴起，随口说的呢?

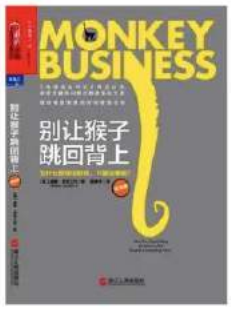
- **给与反馈关注行为及影响**
  - 反馈分为两类：积极性反馈和发展性反馈。顾名思义，积极性反馈是为了表达对被指导者的赞扬与认可，发展性反馈则用于指出需要改善的地方。这里区别于父母常用的唠叨模式，我们同样要有结构，即行为+影响+态度（认可或不足）；如，不能简单粗暴的说 xx 没有团队精神。

- **建立定期的 one one 沟通渠道**
  - Oneone 沟通，展现了重视，给与了确定性的准备时间，面对面的沟通更容易进入到更深层面的沟通；

向下沟通还有非常重要的点是，上级怎么处理下级扔过来的问题。当下属找你的时候，他们很多时候不是在寻求解决方案，而是寻求一个能解决问题的人。下属向管理者报告时，应该引导他说"我有一个问题"，而不能说"我们有一个问题"。下属提出问题的时候，应该让他理解这个问题是他的。这里举两个列子：







正确的对话方式："我暂时不能给你建议。你要把你负责的事情再考虑下，下午 4 点半，来我的办公室，提出可行的进一步想法。"，或者"我 24 小时内没空处理这个问题，你继续去想，明天再给我汇报新的进展。"

如果大家对怎么处理和下属中间任务职责的问题感兴趣的话，推荐大家去看一本书：《别让猴子跳回背上》。

主管的工作，是协助下属完成工作，而不是替下属工作。授人以鱼不如授人以渔的这个道理大家都懂。管理者的目的是培养出更多可以独当一面的员工，如下面的图所示，主管要尽量避免出现自有层级为 1、2 级的员工。不要帮助下属去完成他们分内的事情，培养下属解决问题和思考的能力。面对下属的求助，多问问他"你觉得呢"，让下属自己去想办法解决问题。

- 5 级、等待指示——连该做什么都不知道；
- 4 级、请示要做什么——知道该做什么，但不知道该怎么做；
- 3 级、提出建议，按照裁断的结果行动——能提出备选方案，但需要主管进行方案选择；
- 2 级、行动，但需立即指示——能决定合适的行动方案，但具体执行时会随时请示；
- 1 级、独立行动，主管只需要看他的例行报告。

自由层级第 1 层级与第 2 层级是争取来的特权——他们不是与生俱来的天然权利。这种特权获得来自于你能向上司成功的推销自己的想法和建议而慢慢积累的名声。这种名声也只有在第 3 层级自由量表中才能培养出来。

同步沟通和异步沟通

最后一组要介绍的是同步沟通和异步沟通，这个组词就不像前面那两组那么常见了，不知道大家是不是都理解这两个词的意思？

- **同步沟通**
  - 同步沟通是指你发出一个消息，接收方立即处理这个信息，并马上回复。

- **异步沟通**
  - 异步沟通指的是你发出一个消息，并不期待立即得到反馈的沟通方式。

按照上面的定义，大家就可以把我们日常的沟通方式做一个归类了，比如"会议"、"面对面讨论"、"电话"就属于典型的同步沟通；"邮件"、"语雀"、"Aone"就是典型的异步沟通了。

那"钉钉"和"微信"应该归在哪一类呢？微信和钉钉一般认为是同时具备同步沟通和异步沟通的特征的，过程中有随时变化的可能，就看大家在沟通的过程中是否实时。不过考虑到钉钉有消息阅读状态的提示，一般我们认为钉钉更加接近同步沟通的方式。

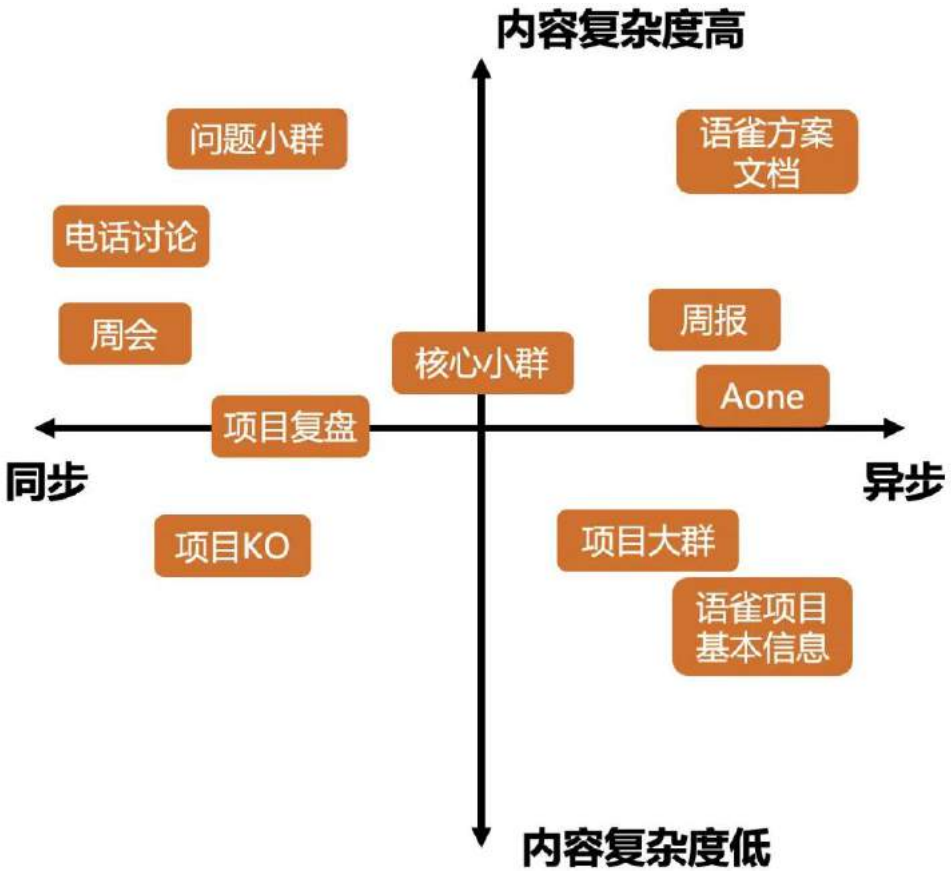
下面我们就对这两种沟通的方式和场景做一个对比：

	同步沟通	异步沟通
优点	互动性好、响应快、仪式感强	减少打断、成本低、可以批量处理 准备充分、信息归类清晰
问题	打断工作、成本大、容易堵塞	响应慢、对系统和参与人的要求高
应用场景	复杂的协作、需要密集对焦 紧急情况，强调仪式感的场合	较大人员规模的沟通，事项条目较多， 需要充分思考的方案准备
建议	避免在较大规模的人数时做同步沟通 关注会议效率，减少同步沟通频率	规范化沟通形式，选择合理承载工具

随着在阿里钉钉的深度使用，同步沟通的比重越来越大，不可否认这种方式让整体的工作效率得到了很大的提升，但任何事情过量就会带来问题。同步沟通常见的问题有：

- 频繁打断，大家的工作时间都被切割成了碎片，妨碍了深度工作的开展。
- 在线的优先级优于高产。不在线的人连发声机会都没有，事情就已经决定了。于是大家都得被迫实时在线，被迫参加每个可能有关的会议。
- 造成不必要的压力。实时在线的期望使人们丧失对时间表的控制。工作时间内，大家响应式回应各方请求，而不是主动地按自己的计划行事。有研究表明，因为常被打断，人们反而会努力把事情做得更快（做得更差为代价），并由此感到更大的压力和困惑；
- 导致低质量的讨论和次优方案。因为需要马上回复，人们没有充分的时间来考虑周全。第一时间给出的方案往往不是你能给出的最优方案。

那在日常的项目管理中，我们怎么有效的利用好各种沟通方式，让这些沟通方式能更好的发挥他们的作用，同时避免一些副作用呢？这里我把项目中最常见的沟通方式放在了一个象限空间中（横坐标是同步或者异步，纵坐标是沟通内容的复杂程度），希望大家可以对比一下自己的日常项目，做个参考。



最后，也是最重要的

终于到了最后（感谢有人能看到这里）。

本文讲了很多的不同沟通场景中的技巧，这些沟通技巧有的时候换个角度也可以看成沟通中的套路。在这里，我需要强调一下，一个沟通的成败，套路的东西会有效果，但是效果也是有限的，沟通中更重要的还是要看沟通的动机。你沟通的目的是不是希望要实现"1+1>2"的效果，你是不是有一个开放的心态，是不是重视差异化、视每次冲突是成长的机会。只要是有一个真诚的心，就算沟通技巧上问题重重，我相信别人还是愿意和你合作。这就是所谓的："多一点真诚，少一点套路"！

一线职场&学习问答实录

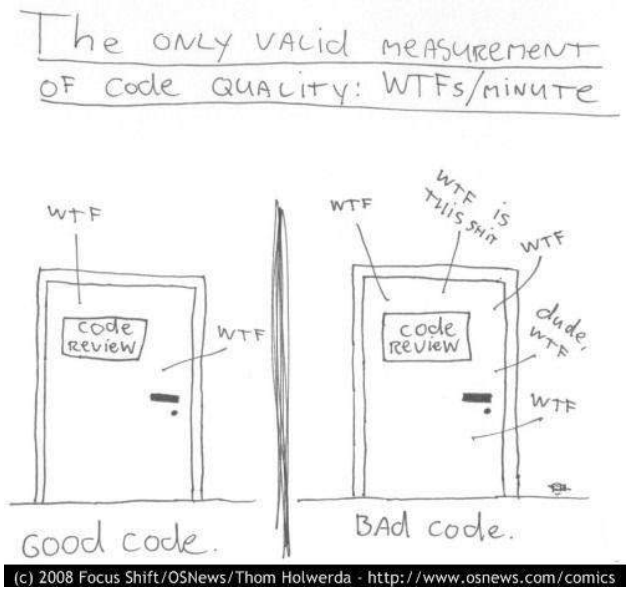
Q: 如何写出优雅的代码?

作者 | 泽畔

出品 | 淘系技术

先说好，衡量代码质量的唯一标准是阅读该代码时说脏话的次数。

（WTF/min ）(ノ'□')ノ ┻┻



这句话来自 Robert C. Martin

Bob 大叔对于好代码的理解非常有趣。

我们编写的代码，除了用于机器执行产生我们预期的效果以外，更多的时候是给人读的，这个读代码的可能是后来的维护人员，更多时候是一段时间后的作者本人。

今天，我们就来聊一聊，什么代码是优雅的代码，怎样写出优雅的代码。

任何一个傻瓜都能写出计算机可以理解的代码。唯有写出人类容易理解的代码，才是优秀的程序员。—— Martin Fowler

谈到好代码，首先跳入自己脑子里的一个词就是：整洁。

好的代码一定是整洁的，给阅读的人一种如沐春风，赏心悦目的感觉。

整洁的代码如同优美的散文。—— Grady Booch

好代码的特性

很难给好的代码下一个定义，相信很多人跟我一样不会认为整洁的代码就一定是好代码，但好代码一定是整洁的，整洁是好代码的必要条件。整洁的代码一定是高内聚低耦合的，也一定是可读性强、易维护的。

- 高内聚低耦合
- 高内聚低耦合几乎是每个程序员员都会挂在嘴边的，但这个词太过于宽泛，太过于正确，所以聪明的编程人员们提出了若干面向对象设计原则来衡量代码的优劣：
- 开闭原则 OCP (The Open-Close Principle)
- 单一职责原则 SRP (Single Responsibility Principle)
- 依赖倒置原则 DIP (Dependence Inversion Principle)
- 最少知识原则 LKP (Least Knowledge Principle)) / 迪米特法则 (Law Of Demeter)
- 里氏替换原则 LSP (Liskov Substitution Principle)
- 接口隔离原则 ISP (Interface Segregation Principle)
- 组合/聚合复用原则 CARP (Composite/Aggregate Reuse Principle)

这些原则想必大家都很熟悉了，是我们编写代码时的指导方针，按照这些原则开发的代码具有高内聚低耦合的特性。换句话说，我们可以用这些原则来衡量代码的优劣。

但这些原则并不是死板的教条，我们也经常会因为其他的权衡（例如可读性、复杂度等）违背或者放弃一些原则。比如子类拥有特性的方法时，我们很可能打破里氏替换原则。再比如，单一职责原则跟接口隔离原则有时候是冲突的，我们通常会舍弃接口隔离原则，保持单一职责。只要打破原则的理由足够充分，也并不见得是坏的代码。

可读性

代码只要具有了高内聚和低耦合就足够好了吗？并不见得，我认为代码还必须是易读的。好的代码无论是风格、结构还是设计上都应该是可读性很强的。可以从以下几个方面考虑整洁代码，提高可读性。

命名

大到项目名、包名、类名，小到方法名、变量名、参数名，甚至是一个临时变量的名称，其命名都是很严肃的事，好的名字需要斟酌。

名副其实

好的名称一定是名副其实的，不需要注释解释即可明白其含义的。

```
/**
* 创建后的天数
**/
int d;
```

```
int daysSinceCreation;
```

后者比前者的命名要好很多，阅读者一下子就明白了变量的意思。

容易区分

我们很容易就会写下非常相近的方法名，仅从名称无法区分两者到底有啥区别（eg. getAccount()与 getAccountInfo()），这样在调用时也很难抉择要用哪个，需要去看实现的代码才能确定。

可读的

名称一定是可读的，易读的，最好不要用自创的缩写，或者中英文混写。

足够短

名称当然不是越长越好，应该在足够表达其含义的情况下越短越好。

格式

良好的代码格式也是提高可读性非常重要的一环，分为垂直格式和水平格式。

垂直格式

通常一行只写一个表达式或者子句。一组代码代表一个完整的思路，不同组的代码中间用空行间隔。

```
public class Demo {

    @Resource

    private List<Handler> handlerList;

    private Map<TypeEnum, Handler> handlerMap = new ConcurrentHashMap<>();


    @PostConstruct
```

```
    private void init() {
        if (!CollectionUtils.isEmpty(handlerList)) {
            for (Handler handler : handlerList) {
                handlerMap.put(handler.getType(), handler);
            }
        }
    }

    publicResult<Map<String, Object>> query(Long id, TypeEnum typeEnum) {
        Handler handler = handlerMap.get(typeEnum);
        if (null == handler) {
            return Result.returnFailed(ErrorCode.CAN_NOT_HANDLE);
        }
        return handler.query(id);
    }
}
```

如果去掉了空行，可读性大大降低。

```
public class Demo {
    @Resource
    private List<Handler> handlerList;
    private Map<TypeEnum, Handler> handlerMap = new ConcurrentHashMap<>();
    @PostConstruct
    private void init() {
        if (!CollectionUtils.isEmpty(handlerList)) {
            for (Handler handler : handlerList) {
                handlerMap.put(handler.getType(), handler); } } }
```

```
    public Result<Map<String, Object>> query(Long id, TypeEnum typeEnum) {
        Handler handler = handlerMap.get(typeEnum);
        if (null == handler) {
            return Result.returnFailed(ErrorCode.CAN_NOT_HANDLE);
        }
        return handler.query(id); }
}
```

类静态变量、实体变量应定义在类的顶部。类内方法定义顺序依次是：公有方法或保护方法 > 私有方法 > getter/setter 方法。

水平格式

要有适当的缩进和空格。

团队统一

通常，同一个团队的风格尽量保持一致。集团对于 Java 开发进行了非常详细的规范。（可点击下方阅读原文，了解更多内容）

类与函数

类和函数应短小，更短小

类和函数都不应该过长（集团要求函数长度最多不能超过 80 行），过长的函数可读性一定差，往往也包含了大量重复的代码。

函数只做一件事（同一层次的事）

同一个函数的每条执行语句应该是统一层次的抽象。例如，我们经常会写一个函数需要给某个 DTO 赋值，然后再调用接口，接着返回结果。那么这个函数应该包含三步：DTO 赋值，调用接口，处理结果。如果函数中还包含了 DTO 赋值的具体操作，那么说明此函数的执行

语句并不是在同一层次的抽象。

参数越少越好

参数越多的函数，调用时越麻烦。尽量保持参数数量足够少，最好是没有。

注释

别给糟糕的代码加注释，重构他

注释不能美化糟糕的代码。当企图使用注释前，先考虑是否可以通过调整结构，命名等操作，消除写注释的必要，往往这样做之后注释就多余了。

好的注释提供信息、表达意图、阐释、警告

我们经常遇到这样的情况：注释写的代码执行逻辑与实际代码的逻辑并不符合。大多数时候都是因为代码变化了，而注释并没有跟进变化。所以，注释最好提供一些代码没有的额外信息，展示自己的设计意图，而不是写具体如何实现。

删除掉注释的代码

git 等版本控制已经帮我们记录了代码的变更历史，没必要继续留着过时的代码，注释的代码也会对阅读等造成干扰。

错误处理

错误处理很重要，但他不能搞乱代码逻辑

错误处理应该集中在同一层处理，并且错误处理的函数最好不包含其他的业务逻辑代码，只

需要处理错误信息即可。

抛出异常时提供足够多的环境和说明，方便排查问题

异常抛出时最好将执行的类名，关键数据，环境信息等均抛出，此时自定义的异常类就派上用场了，通过统一的一层处理异常，可以方便快速地定位到问题。

特例模型可消除异常控制或者 null 判断

大多数的异常都是来源于 NPE，有时候这个可以通过 Null Object 来消除掉。

尽量不要返回 null ，不要传 null 参数

不返回 null 和不传 null 也是为了尽量降低 NPE 的可能性。

如何判断不是好的代码

讨论了好代码的必要条件，我们再来看看好代码的否定条件：什么不是好的代码。Kent Beck 使用味道来形容重构的时机，我认为当代码有坏味道的时候，也代表了其并不是好的代码。

代码的坏味道

重复

重复可能是软件中一切邪恶的根源。—— Robert C.Martin

Martin Fowler 也认为坏味道中首当其冲的就是重复代码。
很多时候，当我们消除了重复代码之后，发现代码就已经比原来整洁多了。

函数过长、类过大、参数过长

过长的函数解释能力、共享能力、选择能力都较差，也不易维护。
过大的类代表了类做了很多事情，也常常有过多的重复代码。
参数过长，不易理解，调用时也容易出错。

发散式变化、霰弹式修改、依恋情结

如果一个类不是单一职责的，则不同的变化可能都需要修改这个类，说明存在发散式变化，应考虑将不同的变化分离开。

如果某个变化需要修改多个类的方法，则说明存在霰弹式修改，应考虑将这些需要修改的方法放入同一个类。

如果函数对于某个类的兴趣高于了自己所处的类，说明存在依恋情结，应考虑将函数转移到他应有的类中。

数据泥团

有时候会发现三四个相同的字段，在多个类和函数中均出现，这时候说明有必要给这一组字段建立一个类，将其封装起来。

过多的 if...else 或者使用 switch

过多的 if...else 或者 switch ，都应该考虑用多态来替换掉。甚至有些人认为除个别情况外，代码中就不应该存在 if...else 。

总结

本文首先一句话概括了我认为的好代码的必要条件：整洁，接着具体分析了整洁代码的特点，又分析了好代码的否定条件：什么样的代码不是好的代码。仅是本人的一些见解，希望对各位以后的编程有些许的帮助。

我认为仅仅编写出可运行的代码是远远不够的，还要时刻注意代码的整洁度，留下一些漂亮的代码，希望写的代码都能保留并运行 102 年！

参考：

《重构——改善既有代码的设计》Martin Fowler
《代码整洁之道》 Robert C. Martin
《Java 开发规约中文版 》阿里巴巴集团约码项目组

Q: 如何看待 2021 年秋招算法岗灰飞烟灭?

作者 | 阅谦
出品 | 淘系技术

18 年是否值得进入，19 年供大于求，20 年一片红海诸神黄昏。

去年都诸神黄昏了，今年换个词——"灰飞烟灭"。

21 届找算法相关工作的同学不妨进来谈谈感想。

阅谦，2019 年阿里校招算法岗，目前在阿里巴巴淘系技术部担任高级算法工程师。从过来人的角度，给学弟学妹们聊聊算法求职&晋升之路上的感想。算法岗位的竞争是激烈的，不再赘述这个情况增加焦虑，主要聊聊他的成长心得。

关于我自己

我是从 2016 年 5 月份开始基础机器学习个领域的，之前对这个领域完全没有了解，本科期间只做过一些 app 开发和网站开发的工作。

之后在导师的指导下学习吴恩达教授的《机器学习》课程入门，在这期间了解到 Coursera 这个网站，后来也为我提供了很多学习资源。

读研期间主要从事强化学习的研究，不过其实具体做的东西就比较杂了 ~ 大概就是我是一块砖，哪里需要我我就往哪里搬

关于算法的学习主要是通过自学和参加各类竞赛的方式，这个之后会细说

本文的一切内容只是根据我自己的经历整理得出来的，可能并不适合每个人，甚至也有很多属于弯路，只作为一个参考就好~

我还记得刚入学的时候研三即将毕业的师兄在就业分享会上讲过下面一段话：

- 知道自己想做什么
- 知道自己能做什么
- 积极明确的职业规划&找到身边的职业榜样
- 基础知识建设很重要
- 经历成就简历
- 面试是有一定运气成分的
- 机会总是留给有准备的人
- 从事一份最适合自己的工作

基础知识建设

- 基础《统计学习方法》和周志华老师的西瓜书，进阶的可以看 ESL 或 PRML。
- 领域内的书和 paper，CV,NLP，强化学习这个就每个人都不一样了，找自己适合的看
- 然后再推荐一下 Coursera 这个网站，如果你不太喜欢看书，喜欢看视频的话，这个很适合，另外这里还有编程实践的模块，理论和实践相结合，挺不错的。
- 读研期间大概在这个网站上刷了 50 多门课，有人问哪来这么多时间，读研期间我基本都是逃课的（不建议大家学习），然后看视频基本是 1.5-2 倍速播放~
- 后来也是机缘巧合帮助黄海广博士一起整理了一门课程的笔记，受到了广泛的好评~

关于竞赛

在实验室工作之余，自己也参加了一些相关的竞赛，由于时间原因和个人能力原因可能名次不是太好，不过还是学到了很多东西，也认识了很多小伙伴和大佬们，收获还是很大的。

在实习期间也看到过一些同学的简历，基本都是有比赛经历在里面的，感概现在整体水平是水涨船高了。

建议大家 「初期还是以学习为主，之后可以好好去做一个为自己的简历加分」 ~ 另外就是比如因为时间久远或者是组队分工的原因，如果一些比赛自己不是对完整方案非常清楚，就不建议写在简历上了， 「一定要写自己熟悉的」

下面是简单例了下自己大概好好做过的一些比赛(其实还有几个划水的，就不上了):

- 2016.10 - 2016.11 2016CCF 大数据竞赛-搜狗用户画像挖掘 rank 26/894
- 2017.03 - 2017.04 kesci-携程出行产品销量预测 rank2/245
  https://github.com/shenweichen/ctrip14
- 2017.04 - 2017.05 京东 JData 算法大赛 高潜用户购买意向预测 rank 13/4242
- 2017.05 - 2017.06 腾讯社交广告竞赛 移动 App 广告转化率预估 rank 20/1650
  https://github.com/shenweichen/Tencent_Social_Ads2017_Mobile_App_pCVR

- 2017.11 - 2017.12 2017CCF 大数据竞赛 蚂蚁金服-商场中精确定位用户所在店铺 rank 20/2845

关于论文

硕士期间能发相关领域顶会的人应该是在学术方面很优秀了，但是据我了解大部分同学还是没有的，不过没关系，我们不能发，但是我们可以多读多写，领域内经典 papar 一定要了解，发展脉络心里要清楚，对于一些经典的方法，抽空可以去实现一下。

多读 paper，一方面了解一下业内的最新进展，一方面自己的知识也可以更体系化。

我自己的话也是抽空实现了一下，https://github.com/shenweichen/DeepCTR

不仅能加深自己的理解，放在简历里也是加分项。

后来又和小伙伴一起开发了 DeepMatch 。

关于面试

如果把面试的难度看作一个随机变量，那我们能做的就是将它的方差降低到最小。

我觉得面试官从拿到你的简历开始，就已经给了你一个打分，剩下的过程就是尝试去印证他打分的合理程度。所以在写简历的时候一定要确保简历上的内容都是自己做过且非常熟悉的。

一般一场算法岗位的技术面试考察的内容无非涉及一下几块：ML/DL/RL 算法基础知识，结合简历和业务应用的相关知识，基本的编程能力，数理统计和智力逻辑题。

我们在学习一个算法或者模型的时候，一定要学着类比和关联，比如很多人说自己熟悉逻辑回归 LR，那么问问自己，下面这些内容你都了解吗？

- LR 归一化问题，什么情况可以不归一化，什么情况必须归一化，为什么？
- 提到 LR 损失函数要能知道交叉熵，为什么是它，以它为损失函数在优化的是一个什么东西，知道它和 KL 散度以及相对熵的关系。
- 提到 LR 的求解方法，比如 SGD，知道 SGD 和 BGD 的区别，知道不同的 GD 方法有什么区别和联系，二阶优化算法知道什么，对比 offline learning 和 online learning 的区别。
- 提到调参，知道模型不同超参数的含义，以及给定一个特定情况，大概要调整哪些参数，怎么调整？
- 提到 LR 的正则，知道 l1l2 的原理，几何解释和概率解释。
- LR 的分布式实现逻辑是怎么样的，数据并行和模型并行的区别，P-S 架构大概是怎么一回事？
- LR 作为一个线性模型，如何拟合非线性情况？特征侧比如离散化，交叉组合，模型比如引入 kernel，又可以推广到 FM 等 model 上。

个人感觉如果一场面试中大部分时间是你在向面试官输出你的知识，而不是等他来问这样的一问一答的话，那么基本就是 ok 的了。

关于编程题目，leetcode 多刷刷，一般 medium 居多。
关于概率统计和智力逻辑题，这个网上也可以网上搜搜看看~

关于晋升

昨天系统里正式公布了晋升结果，非常荣幸的从一名算法工程师成为了高级算法工程师。过去一年里的努力和付出获得了认可，曾经的困难，委屈和不被理解随着时间的推移也逐渐被忘记。非常感谢在阿里的这 480 天里帮助和指导过我的师兄，同学，同事和老板们，感恩相遇。

阿里是一家很棒的公司，尤其是对于刚刚毕业的应届生来说，在这里可以充分的将学校书本里所学的知识和企业的实际应用需求进行连接，接触到行业内的领先技术，获得一个非常快速的成长。同时身边充满了各种大牛，每个人都有自己独特的优势和亮点，值得学习。

当然，今天这篇文章不是来夸公司的，主要还是希望借此机会分享一些的想法，尤其是「刚刚毕业工作或即将毕业工作的同学们」 ~

放低姿态，从最简单的做起

刚毕业的同学，尤其是「算法岗」的同学，很容易产生一个想法。就是师兄或者主管给我安排的任务是太简单，没什么技术含量或者看起来没什么用。再想想自己手握 X 篇顶会，Y 个竞赛 Top，Z 个上千 star 的开源项目，拒掉了多少家 offer 来到这里，就让我做这个？就这？没错，就这。

- 首先工作的安排是取决于业务的形态和发展需要的，主管不会去浪费你一个人力去做那些真正没有用的事情的，一定是业务确实有需要。
- 第二，对于新同学来说，做一些简单的任务也是让自己熟悉工作的一个过程，从简单的任务不会觉得学习曲线过于陡峭。
- 第三，通过这些相对简单的任务，可以展现出自己做事情的方法和逻辑，当主管觉得你做事还算靠谱后，自然会安排更有难度更有挑战的工作。

保持学习，向身边优秀的同学看齐

其实这句话从读研的时候，实验室的师兄就有和我们讲过。到了工作的时候，依然被不断的提起，想必道理大家都懂。

不要因为自己有了工作了就懈怠了，请保持你当时在找工作时的学习态度，技术行业的发展是很快的，今天的 state of the art 就是明天的 baseline。

去发现身边同学的优点和亮点，观察他们是怎么看待和思考问题以及做事的方法，有机会的话可以多向他们请教和交流，一定会有收获的！

迎接挑战，承担更多的责任

其实这里想说的是，走出舒适区，每每当你感到痛苦和艰难的时候，正是你成长和收获最大的时候。

每一次困难和挑战都是一次学习的机会。做已经掌握或者熟悉的事情，虽然很容易取得结果，但是对个人来说并没有带来成长。

有时候可能一件事情对自身能够带来的价值是有限的，但是如果你把坑都趟平了，可能会让那个整个小组或者团队的同学受益，那你愿不愿意去做。这种时候就有两种想法：

a. 我去做了，这样大家都可以受益 b. 反正早晚有人做，我等着就行了

但如果每个人 b 想法，那么可怕的事情就发生了，陷入了死锁，每个人都在等待别人去做，整个团队的发展也就收到了阻碍。

皮实抗压，培养强大的心力

在过去的一年里，也有过很多次想要放弃的念头，不顺的时候晚上会焦虑的难以入睡，但内心也有另一种声音告诉自己，再试试看看，也许是哪里细节没注意到，多给自己一点耐心。减少玻璃心，减少浮躁，成为一个内心强大的人。心力，脑力，体力三者缺一不可。

成就别人就是成就自己

每个个体的力量有限，每个人擅长的事物也不尽相同，发挥自己优势和长处去帮助身边的同学一起进步，在未来的某一天或许会有意想不到的收获 ~

最后，非常幸运能够加入这样一个团队，在这里收获了很多，也成长了很多，接下来会继续向着新的目标努力~

学弟学妹们一起加油！

Q: 前端 p6 什么水平？如何衡量？

作者 | 磐冲
出品 | 淘系技术

前端 p6/p6+应当有怎样的技术素质，水平达到什么程度！

官方亲自答。

这是一份超详细的阿里前端 p6/p6+的水平和素质参考路线。

我们想要的同学 | JD

业务背景

淘宝内部最大创新项目之一，大团队已有百人规模，大部分项目处于保密阶段，前景远大

职位描述

1. 负责组件库与业务页面开发。
2. 带领团队完成技术产品实现。
3. 负责大型多应用架构设计。
4. 利用前端技术与服务端协同完成团队业务目标。

职位要求

0. 掌握图形学，webgl 或熟练使用 threejs 框架，熟练 canvas 相关渲染及动画操作的优先。
1. 熟练掌握 JavaScript。
2. 熟悉常用工程化工具，掌握模块化思想和技术实现方案。
3. 熟练掌握 React 前端框架，了解技术底层。同时了解 vue 以及 angular 等其他框架者优先。

4. 熟练掌握 react 生态常用工具，redux/react-router 等。
5. 熟悉各种 Web 前端技术，包括 HTML/XML/CSS 等，有基于 Ajax 的前端应用开发经验。
6. 有良好的编码习惯，对前端技术有持续的热情，个性乐观开朗,逻辑性强，善于和各种背景的人合作。
7. 具有 TS/移动设备上前端开发/NodeJS/服务端开发等经验者优先。

翻译一下 JD

为什么起这个标题呢？因为有很多人看到职位描述，可能就在和自己做的事情一一比对，把关键字都核对上。而很多前端同学看到职位要求第一条里的图形学，可能就开始打退堂鼓了。或者看到几个关键字自己都认识，就觉得没问题，还挺简单的。

就这样望而却步真的好吗？为什么职位描述看着简单，面试却这么难呢？你真的读懂这份职位描述了吗？

现在，不妨先停一下，就上面的问题，我们来细细品一下。什么叫读懂职位描述呢？从我个人的理解，读懂职位描述，应该是读懂这个职位需要哪些基础能力，以及可能遇到哪些挑战。我们写自己简历的时候，“精通 react”和“熟练使用 react”，相信大家不会随意去写。同样的，JD 里面的：掌握、熟练掌握、了解、熟悉，也不是随意写的，这代表了团队对新同学的能力要求。

回想写自己简历的时候，我们会对这个前缀扪心自问一下。因为会担心一旦写了精通，面试官的问题会更难，甚至觉得只有源码倒背如流的人，才能称得上精通。当然也会有同学非常自信，用 react 做过几个项目，就写上了精通 react。

这两种都可以称为精通，也都不可以。没有客观标准，又怎么去衡量呢？而标准在哪里呢？所以在这里，我从阿里面试官角度，给出我认为的标准，尽可能的做到客观可量化。那么，基于上面这份职位标准，我来翻译一下职位要求：

首先，总览全部的要求，会发现这个职位虽然提到了 3d 相关的技能，但是大部分却是应用开发相关的能力，所以这个职位并不是想找专业的 3d 领域同学，而是需要一个工程化能力强，对 3d 有了解的同学。

0. **掌握图形学，webgl 或熟练使用 threejs 框架，熟练 canvas 相关渲染及动画操作的优先。**

初级：

- 学习过图形学相关知识，知道矩阵等数学原理在动画中的作用，知道三维场景需要的最基础的构成，能用 threejs 搭 3d 场景，知道 webgl 和 threejs 的关系。
- 知道 canvas 是干嘛的，聊到旋转能说出 canvas 的 api。
- 知道 css 动画，css 动画属性知道关键字和用法(换句话说，电话面试会当场出题要求口喷 css 动画，至少能说对大概，而不是回答百度一下就会用)。
- 知道 js 动画，能说出 1~2 个社区 js 动画库，知道 js 动画和 css 动画优缺点以及适用场景。
- 知道 raf 和其他达到 60fps 的方法。

中级：

- 如果没有 threejs，你也能基于 webgl 自己封装一个简单的 threejs 出来。
- 聊到原理能说出四元数，聊到鼠标操作能提到节流，聊到性能能提到 restore，聊到帧说出 raf 和 timeout 的区别，以及各自在优化时候的作用。
- 知道怎样在移动端处理加载问题，渲染性能问题。
- 知道如何结合 native 能力优化性能。
- 知道如何排查性能问题。对 chrome 动画、3d、传感器调试十分了解。

高级：

- 搭建过整套资源加载优化方案，能说明白整体方案的各个细节，包括前端、客户端、服务端分别需要实现哪些功能点、依赖哪些基础能力，以及如何配合。

- 设计并实现过前端动画引擎，能说明白一个复杂互动项目的技术架构，知道需要哪些核心模块，以及这些模块间如何配合。
- 有自己实现的动画相关技术方案产出，这套技术方案必须是解决明确的业务或技术难点问题的。为了业务快速落地而封装一个库，不算这里的技术方案。如果有类似社区方案，必须能从原理上说明白和竞品的差异，各自优劣，以及技术选型的原因。

1. **熟练掌握 JavaScript。**

初级：

- JavaScript 各种概念都得了解，《JavaScript 语言精粹》这本书的目录都得有概念，并且这些核心点都能脱口而出是什么。这里列举一些做参考：
- 知道组合寄生继承，知道 class 继承。
- 知道怎么创建类 function + class。
- 知道闭包在实际场景中怎么用，常见的坑。
- 知道模块是什么，怎么用。
- 知道 event loop 是什么，能举例说明 event loop 怎么影响平时的编码。
- 掌握基础数据结构，比如堆、栈、树，并了解这些数据结构计算机基础中的作用。
- 知道 ES6 数组相关方法，比如 forEach，map，reduce。

中级：

- 知道 class 继承与组合寄生继承的差别，并能举例说明。
- 知道 event loop 原理，知道宏微任务，并且能从个人理解层面说出为什么要区分。知道 node 和浏览器在实现 loop 时候的差别。
- 能将继承、作用域、闭包、模块这些概念融汇贯通，并且结合实际例子说明这几个概念怎样结合在一起。
- 能脱口而出 2 种以上设计模式的核心思想，并结合 js 语言特性举例或口喷基础实现。
- 掌握一些基础算法核心思想或简单算法问题，比如排序，大数相加。

2. **熟悉常用工程化工具，掌握模块化思想和技术实现方案。**

初级：

- 知道 webpack，rollup 以及他们适用的场景。
- 知道 webpack v4 和 v3 的区别。
- 脱口而出 webpack 基础配置。
- 知道 webpack 打包结果的代码结构和执行流程，知道 index.js，runtime.js 是干嘛的。
- 知道 amd，cmd，commonjs，es module 分别是什么。
- 知道所有模块化标准定义一个模块怎么写。给出 2 个文件，能口喷一段代码完成模块打包和执行的核心逻辑。

中级：

- 知道 webpack 打包链路，知道 plugin 生命周期，知道怎么写一个 plugin 和 loader。
- 知道常见 loader 做了什么事情，能几句话说明白，比如 babel-loader，vue-loader。
- 能结合性能优化聊 webpack 配置怎么做，能清楚说明白核心要点有哪些，并说明解决什么问题，需要哪些外部依赖，比如 cdn，接入层等。
- 了解异步模块加载的实现原理，能口喷代码实现核心逻辑。

高级：

- 能设计出或具体说明白团队研发基础设施。具体包括但不限于：
- 项目脚手架搭建，及如何以工具形态共享。
- 团队 eslint 规范如何设计，及如何统一更新。
- 工具化打包发布流程，包括本地调试、云构建、线上发布体系、一键部署能力。同时，方案不仅限于前端工程部分，包含相关服务端基础设施，比如 cdn 服务搭建，接入层缓存方案设计，域名管控等。
- 客户端缓存及预加载方案。

3. **熟练掌握 React 前端框架，了解技术底层。同时了解 vue 以及 angular 等其他框架者优先。**

初级：

- 知道 react 常见优化方案，脱口而出常用生命周期，知道他们是干什么的。
- 知道 react 大致实现思路，能对比 react 和 js 控制原生 dom 的差异，能口喷一个简化版的 react。
- 知道 diff 算法大致实现思路。
- 对 state 和 props 有自己的使用心得，结合受控组件、hoc 等特性描述，需要说明各种方案的适用场景。
- 以上几点 react 替换为 vue 或 angular 同样适用。

中级：

- 能说明白为什么要实现 fiber，以及可能带来的坑。
- 能说明白为什么要实现 hook。
- 能说明白为什么要用 immutable，以及用或者不用的考虑。
- 知道 react 不常用的特性，比如 context，portal。
- 能用自己的理解说明白 react like 框架的本质，能说明白如何让这些框架共存。

高级：

- 能设计出框架无关的技术架构。包括但不限于：
- 说明如何解决可能存在的冲突问题，需要结合实际案例。
- 能说明架构分层逻辑、各层的核心模块，以及核心模块要解决的问题。能结合实际场景例举一些坑或者优雅的处理方案则更佳。

4. **熟练掌握 react 生态常用工具，redux/react-router 等。**

初级：

- 知道 react-router，redux，redux-thunk，react-redux，immutable，antd 或同级别社区组件库。
- 知道 vue 和 angular 对应全家桶分别有哪些。
- 知道浏览器 react 相关插件有什么，怎么用。
- 知道 react-router v3/v4 的差异。
- 知道 antd 组件化设计思路。
- 知道 thunk 干嘛用的，怎么实现的。

中级：

- 看过全家桶源码，不要求每行都看，但是知道核心实现原理和底层依赖。能口喷几行关键代码把对应类库实现即达标。
- 能从数据驱动角度透彻的说明白 redux，能够口喷原生 js 和 redux 结合要怎么做。
- 能结合 redux，vuex，mobx 等数据流谈谈自己对 vue 和 react 的异同。

高级：

- 有基于全家桶构建复杂应用的经验，比如最近很火的微前端和这些类库结合的时候要注意什么，会有什么坑，怎么解决

5. 熟悉各种 Web 前端技术，包括 HTML/XML/CSS 等，有基于 Ajax 的前端应用开发经验。

初级：

- HTML 方面包括但不限于：语义化标签，history api，storage，ajax2.0 等。
- CSS 方面包括但不限于：文档流，重绘重排，flex，BFC，IFC，before/after，动画，keyframe，画三角，优先级矩阵等。
- 知道 axios 或同级别网络请求库，知道 axios 的核心功能。

- 能口喷 xhr 用法，知道网络请求相关技术和技术底层，包括但不限于：content-type，不同 type 的作用；restful 设计理念；cors 处理方案，以及浏览器和服务端执行流程；口喷文件上传实现；
- 知道如何完成登陆模块，包括但不限于：登陆表单如何实现；cookie 登录态维护方案；token base 登录态方案；session 概念；

中级：

- HTML 方面能够结合各个浏览器 api 描述常用类库的实现。
- css 方面能够结合各个概念，说明白网上那些 hack 方案或优化方案的原理。
- 能说明白接口请求的前后端整体架构和流程，包括：业务代码，浏览器原理，http 协议，服务端接入层，rpc 服务调用，负载均衡。
- 知道 websocket 用法，包括但不限于：鉴权，房间分配，心跳机制，重连方案等。
- 知道 pc 端与移动端登录态维护方案，知道 token base 登录态实现细节，知道服务端 session 控制实现，关键字：refresh token。
- 知道 oauth2.0 轻量与完整实现原理。
- 知道移动端 api 请求与 socket 如何通过 native 发送，知道如何与 native 进行数据交互，知道 ios 与安卓 jsbridge 实现原理。

高级：

- 知道移动端 webview 和基础能力，包括但不限于：iOS 端 uiwebview 与 wkwebview 差异；webview 资源加载优化方案；webview 池管理方案；native 路由等。
- 登陆抽象层，能够给出完整的前后端对用户体系的整体技术架构设计，满足多业务形态用户体系统一。考虑跨域名、多组织架构、跨端、用户态开放等场景。
- mock 方案，能够设计出满足各种场景需要的 mock 数据方案，同时能说出对前后端分离的理解。考虑 mock 方案的通用性、场景覆盖度，以及代码或工程侵入程度。
- 埋点方案，能够说明白前端埋点方案技术底层实现，以及技术选型原理。能够设计出基于埋点的数据采集和分析方案，关键字包括：分桶策略，采样率，时序性，数据仓库，数据清洗等。

6. **有良好的编码习惯，对前端技术有持续的热情，个性乐观开朗，逻辑性强，善于和各种背景的人合作。**

初级：

- 知道 eslint，以及如何与工程配合使用。
- 了解近 3 年前端较重要的更新事件。
- 面试过程中遇到答不出来的问题，能从逻辑分析上给出大致的思考路径。
- 知道几个热门的国内外前端技术网站，同时能例举几个面试过程中的核心点是从哪里看到的。

高级：

- 在团队内推行 eslint，并给出工程化解决方案。
- 面试过程思路清晰，面试官给出关键字，能够快速反应出相关的技术要点，但是也要避免滔滔不绝，说一堆无关紧要的东西。举例来说，当时勾股老师面试我的时候，问了我一个左图右文的布局做法，我的回答是：我自己总结过 7 种方案，其中比较好用的是基于 BFC 的，float 的以及 flex 的三种。之后把关键 css 口喷了一下，然后 css 就面完了。

7. **具有 TS/移动设备上前端开发/NodeJS/服务端开发等经验者优先。**

- 根据了解的深度分初/中/高级。
- 知道 TS 是什么，为什么要用 TS，有 TS 工程化实践经验。
- 知道移动端前端常见问题，包括但不限于：rem + 1px 方案；预加载；jsbridge 原理等。
- 能说出大概的服务端技术，包括但不限于：docker；k8s；rpc 原理；中后台架构分层；缓存处理；分布式；响应式编程等。

JD 的要求很难吗？

首先，感谢你能看到这里，如果你是仔细看的，那么我更加感动了。而且你已经用实际行动，证明了你的学习能力和耐心。上面那么大篇幅的 JD 翻译，有一个问题，大家应该都有答案了：为什么职位描述看着简单，面试却这么难呢？然而，有些同学可能会嘲讽起来：写了那么多，我认识的有些阿里 P6，P7 也不是都会啊，大厂都是螺丝钉，也就面试时候问问，实际工作不还是 if else，何况我又遇不到这些场景，我怎么可能知道。

在这里，我想严肃的说明的是：

1. 我所认识的淘宝前端，以及我所在团队的 P6 同学，上面初级都能做到，中级至少覆盖 60%，高级覆盖 20%；P6+同学，中级覆盖 80%以上，高级覆盖 50%以上；P7 同学高级覆盖 80%以上。
2. 我们团队的前端，每一个人都负责多个复杂业务项目(客观数据上：至少对接 20+服务端接口，5 个以上 router 配置，涉及多个用户角色的综合业务系统)，以及一些通用能力，比如组件库等。不存在一个人只接一条业务线，只负责维护某几个组件这种螺丝钉式的工作。我不知道大厂都是螺丝钉的言论为什么会被复用到互联网企业，我个人感受是，如果我在阿里的工作是螺丝钉，那么我以前几份工作可能勉强算是螺纹。另外，如果你想要晋升，那么维护好这几个业务系统只是你的本职工作，晋升时请提供一些更高层面的思考和技术产出。
3. if else 也分鲜花和牛粪。有的人写的是[].reduce，而有的人写的是 var temp = ''; for () { temp += 'xxx' }。另外，如果不知道原理，那么类似 webpack 这种明星级的技术产品，将永远与你无缘。冷静下来想想，webpack 难道也只是 if else 吗？是，又不全是。

聪明的你应该看出来了，上面 JD 翻译里的初级、中级和高级，对应的就是我认为的，阿里 p6/p6+/p7 的能力标准，同时也是一张知识图谱。

初级的要求更偏实际应用和基础原理，中级的要求是基于原理的拓展和复杂技术架构的应用，高级的要求是对跨栈、跨端，多领域结合产出综合方案的能力。

而且，我们对技术的要求，都是能够与实际业务场景结合，或者能对提升工作效率有帮助的。

空谈和尬想，或者只是百度来的文章，没有经过内化，那么面试过程中将被瞬间拆穿。
有时我会在 boss 直聘上直接打字面试，有时我也会听到面试过程中，电话那头传来键盘敲击的声音，甚至有时候我会主动让面试的同学去百度一下，或者挂电话思考一下，过 15 分钟再聊。

我敢这么面试，因为我知道，我要的答案你查不出来，我看的是你真正理解的东西。能搜索到的，我不在乎，我也希望你去查，来为你更好的表现综合能力。

我相信每个人都是能快速成长的，只是每个人缺少的东西不同。有的人少了些脚踏实地，有的人少了些登高望远的机会，更多的人或许只是没有找到那条正确的路。

我希望这篇文章能够帮助到正在前端领域努力的人，也希望这一篇文章就能成为指路明灯之一。但同时我也深知，每个人都是不一样的。

也欢迎关注我们淘系技术的官方知乎号，后续我还会分享更多前端领域成长的心得和方法。

一起加油。

Q: 做了两年前端开发，平时就是拿 Vue 写写页面和组件，简历的项目经历应该怎么写得好看？

作者 | 磐冲
出品 | 淘系技术

你负责的业务是什么？(学会发现问题)

我接触过不少同学，有说在小公司，被业务需求压着。既然大家都说在做业务，那么，正看到这里的你，能不能 5 分钟说明白，你负责的业务是什么？

这个问题我在活动群的 github issue 活动中，带有业务理解标签的题目里经常会问到，可是大部分同学都没有说到位，甚至答非所问。

这里谈谈我个人对业务的理解，或许没有普遍意义，所以仅供参考。

业务最核心的要素是业务本身的价值

一家公司，或者一个部门，做的事情有许许多多，零零散散。也有很多事情合到一起，促成了一件大事的时候。那么，我们是把那些零散的事情都看成业务？还是只把那一件大事看成业务呢？我认为都可以。决定权在于这件事是否逻辑自洽，以及是否具有独特的价值。

接下来让我们拿着一个例子来说，假设你在开发一个营销活动页，这个页面能够给公司带来 3000 人的新用户，这些人有可能会购买公司的产品，从而带来收入。

这里明显可以感受到，营销是一个业务线，他的商业逻辑是投放页面 -> 拉新回流 -> 商品销售，价值在于新用户的触达，以及商品销售利益。基于这两点，我们就值得投入精力，因为做的越好，公司业绩越好。

那么，做个页面就是亮点了？

1. 当然不是，但是亮点已经离我们很近了。如果你想要有亮点，那你需要保持思考。在上面的例子中，我们有许多可以优化和验证的事情。
2. 营销页每天换内容，怎么快速替换？
3. 营销部门人越来越多了，页面每天要 10 个，一个人怎么做得完？
4. 前端的人也越来越多了，改个组件不能只靠复制黏贴，怎么管理？
5. 拉新回流效率具体有多高？新人真的有买我们的商品吗？这么多人投入，都是要工资的，卖出去的商品能够发我们的工资吗？
6. 转化率低了，怎么才能提升？
7. 这个按钮写错个样式到了右边，居然点的人特别多？那下次是不是都应该放右边？

上面列举的几个问题，估计很多同学日常都有做类似的事情。但问题是，这些事情是你想做的，还是产品让你做的？这些事情能诞生什么出来呢？

1. 运营配置后台与投放策略
2. 营销搭建体系
3. 工程化研发套件
4. 业务埋点与数据分析系统
5. 数据仓库与数据分析后台
6. A/B test 系统

至少在我看来，如果面试的同学上来自我介绍的时候，能够讲一下上面例子中遇到的问题，之后再说做了下面对应的某一个系统，那么，这就是绝对够分量的亮点。只可惜这样的同学少之又少，大部分同学是因为产品说要做就去做了。

所以，你真的想过业务是什么吗？有为业务想过什么吗？有了你，业务有什么不同吗？
可以开始写代码了？(学会思考的方式)

好了，假设我们思考了一下，想了点东西出来，接下来我们可以开始写代码了.....吗？

做一个有亮点的技术产出，可不是撸起袖子就能快速干出来的，当然，如果你是个天才，那请自便。如果和我一样是普通人，那么请先做好技术方案设计。而设计的第一步，就是做一个 ppt 工程师，画图。

图，是思想的结晶

在上面提到过的 github issue 活动里，大部分同学的业务大图或者技术架构图，都没法说明白先表达的意思。

几个最典型的问题是：

1. 思路混乱：下面几个框在写业务的系统，上面画了一个 vue 或者 webpack 的框。
2. 层级混乱：底层写的是 native 容器，上层画了个 api gateway。
3. 答非所问：要求画业务大图，结果画了一堆前端脚手架的关键字，或者画成了流程图。

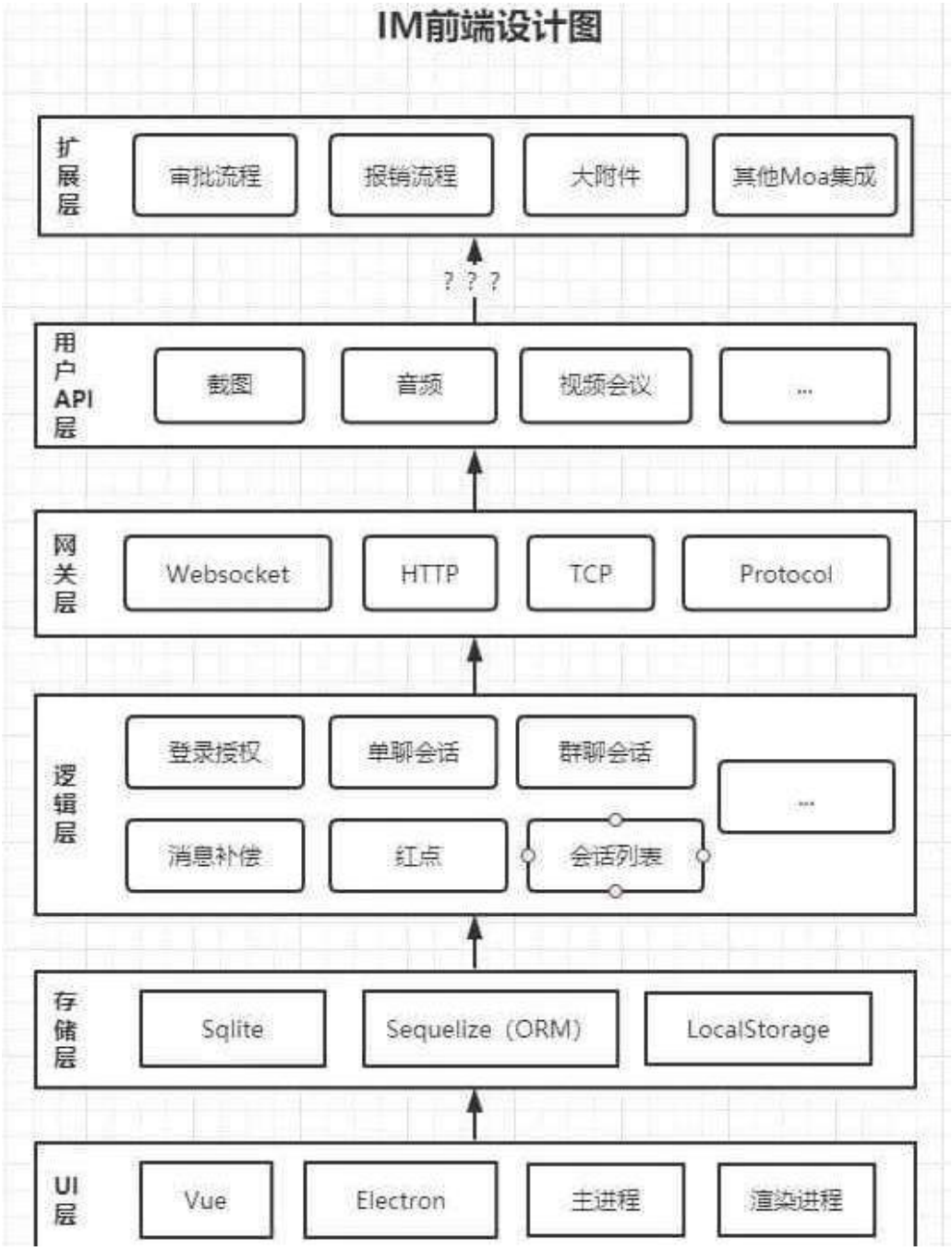
如果看到这里，不明白画图是干什么的同学，可以去查一下架构图是什么，以及如何做程序设计。这经常是被大家忽略的事情，虽然很多同学在大学里学习的时候，都学过相关的课程，但是估计大部分都还回去了。

怎么画好一张图?

这里不做具体的展开，毕竟我自己也不是画图高手，每次画图也是迟迟不知如何下笔。只给到几个建议，供大家参考。

同时，以一个模拟面试同学的案例来做参考。

原图：



第一步，先想明白这张图要表达什么？

这位同学说他参加过很多技术会议，看那些分享的 ppt 里面的大图，都很酷炫，自己平时也有总结(这点非常好)，但是总画不出那种图来。面试过程中我问了这位同学，这张图他想表达什么，答案是他想说明白消息通信业务的技术方案。但是，这张图并不能表达出一个技术方案来。

这张图第一个问题是不够完整，他只有一条主链路，对于 IM 这样的复杂技术产品，主链路只是冰山一角，如果真的只做了主链路，那么代表思考不够，早晚会出现线上故障。

第二个问题在于含义不明与层次混乱。最下面的 UI 层有个箭头指向存储层，那是指渲染进程会去调用 localStorage？那再向上 2 级的网关层呢？UI 层会调用网关层？这里显然逻辑是不通顺的。

第二步，图里的每一个大块必须是同一个领域或类似概念的，每一个框都有意义

在这个问题上，这位同学做的还是很好的，但也还是有些小问题，比如 UI 层里的两个进程。这两个框显得意义不明，在没有描述的情况下，至少我是不明白他想表达的意思，而实际在沟通过程中，他也觉得这里挺奇怪的。

第三步，画完回顾一下是否描述清楚了第一步里的核心逻辑

很多时候我们一气呵成画了一张大图，结果一不小心容易画成一张流程图，把怎么写代码的思路也画到图上了。这就会导致图上有些地方是模块划分，而有些地方则是细节流程，整体就很失调。这只能通过反复的回顾和思考，进行自我调整了。

最后，我给出当时模拟面试时，对于这个业务的粗略设想：

知道原理有什么用？(技术如何赋能)

有了大图，我们终于可以开始实现亮点了……吗？

现实很残酷，哪怕我们想出了一个大饼，并不代表我们能吃到嘴里，从图变成面饼，我们需要太多的中间步骤。

而摆在技术人面前的问题是：如果有面粉了，你会揉面吗？你揉面的技术能保证烤出来的饼好吃吗？

知其然，而后使其然

我认为这就是为什么我们要了解原理。曾经有一位模拟面试的同学，在最后互问互答的时候问了我一个问题，怎么看待面试造火箭，平时拧螺丝？我觉得有点冤枉，因为一面大部分问的都是怎么拧螺丝，以及螺丝的型号，二面开始也就问问怎么造飞机，但是真的进入工作状态，阿里的场景里，至少在我们团队的场景里，我们就是在造火箭，只是造火箭的时候必须要拧拧螺丝，没螺丝你敢上？

有同学又不服了，我会拧螺丝，和我需要知道用什么螺丝有什么关系。那么上面那个烤饼，你能告诉我放白芝麻好吃还是放黑芝麻好吃吗？我相信大厨一定能回答上来，他甚至连小麦原产地都会和你掰扯一下。为什么到了同样需要匠心的编码领域，我们就不用关心用什么螺丝了呢？

当时我给这个同学举了个实际的例子：简历中有提到上传，那你能不能当场告诉我，这个上传是服务端 http 接口配合你上传，还是用阿里云 oss？用 oss 是服务端每次加签，还是用 sts，还是前端直接加签？http 上传你用什么 contentType？用 form 表单组件提交还是自己通过 xhr 发送？如果需要登录鉴权怎么办？如果出现跨域问题怎么办？两种场景都有，都要实现，怎么封装组件？

什么？你说你要百度一下？你要百度一天？那我为什么不聘用那个不用百度的人呢？一天的工资算上 5 金这些成本，月薪 20k 来算，估计也得有小 2000 了，如果我把这 2000 增加到一个懂原理的大神手里，我们岂不是双赢，为什么要等你去搜索呢？只是个简单的上传文件功能，也就是页面里的一个豆腐块，这么小的螺丝，里面却有大大的学问。而日常工作中我们遇到类似的问题有非常多，具体可以参考我上一篇文章的解读，这里就不重复了。

任务的拆解

对于平时愿意学习的同学，到这一步可能开始陷入迷茫了，我之前也遇到过类似的困惑，那就是：要不要造轮子

我们经常会发现好像什么都能做，比如：你有的，我改改也能实现；社区有个差不多能用的，要不要直接用；好像大图上都有差不多的，那是不是拼拼凑凑就可以了，这个方案是不是没什么好做的了。

从我个人来说，每次画图我都会陷入这样的思考，还常常会钻牛角尖，为了整点差异化，故意换一些思路去做，这样能保证这个饼是我的。但最后我都会绕出来，这得益于上面画图的第三步，每次画完我都会重新回顾一下我真正想做的事情是什么。我认为这也是是否造轮子的一个评判标准：从业务的价值出发，思考真正核心的目标，并且为之努力，如果有现成的轮子，能满足业务核心诉求，那就放手去用。

首先，现实往往是这样的，当我们放手去用的时候，会发现这个轮子好像不那么好用，或者这个轮子没人维护了，又或者业务变化太快，轮子自己觉得顶不住了。机会自然会来到身边，而触发这些机会的，是我们不断的站在业务的视角去思考问题，业务的变化一定比一个平台化的轮子要来得快。

其次，真正核心的系统一定是紧贴业务，而且很难大范围复用的，好的技术架构在设计的时候，讲究的是够用即可，过度设计大部分就是没用的设计。在之后的迭代中，会随着业务的不断变化，被带动着自我进化，那最终的产物也自然是和业务形态非常贴合。所以，我个人在选择的时候，一些核心的轮子，该造就造起来，但这些轮子一定是带有业务特色的，比如

我会去造一个业务组件库，但是我绝不会去造一个 antd。

最后，随着事物的演变，分久必合合久必分，单一业务用的好的系统一定是可以在更高的视角上抽象、整合的，在整个过程中，每个人的成长就会是我们想要的亮点了。或许在简历上你写下的是一个已经废弃的系统，但是它的灵魂在你心里，也存在于把他整合了的系统里，这种亮点在个人介绍的时候，一定是能侃侃而谈的。

从 1 到 10 能做什么？(思考方式的抽象)

终于，我们经历的各种抉择，投入了大量的时间，把一个亮点做出来了，完成了美好的从 0 到 1，可有时候我们会发现的问题：从 0 到 1 看上去有很多要做的，做完了，从 1 到 10 还能做什么？

这个问题我个人也没有太多话语权，因为这两年总是在做从 0 到 1 的事情，甚至和我老板也聊过这个，总感觉自己没有个确定的事情。从 0 到 1 做一次挺爽的，一直做，不会一直爽，却只会让人觉得心慌，毕竟谁能保证永远能想出从 0 到 1 的事情呢？

而静下来反思之后，我发现事情并不是这么一刀切的，谁能说明白现在做的事情是 0 到 1，还是 1 到 10 呢？这里的边界其实并没有那么明确，但抽象看，他们都是同一个套路
业务/技术思考 => 发现痛点 => 产出方案 => 拆解实现

伴随着这个闭环，业务永远在变化，而变化又会带来新的问题，只要保持一个思考的状态，没有必要区分具体再哪个阶段，因为你总能找到可以实现自我价值的地方，发现属于你的亮点。

链接：

https://github.com/vianvio/FE-Companions/issues

Q: 一份优秀的前端开发工程师简历是怎么样的?

作者 | 前端团队
出品 | 淘系技术

作为阿里巴巴淘系的面试官，我觉得自己有资格回答这个问题。
而要回答这个问题，首先需要清楚地认识到「一份优秀的技术人简历是怎么样的」，而优秀的前端开发工程师的简历则无非是在前者的基础上增加与技术领域相关的细节。因此，我打算分两个部分来回答这个问题：

- 一份优秀的技术人员的简历是怎么样的?
- 一份优秀的「前端开发工程师」简历是怎么样的? （以阿里淘系前端 P6 ~ P7 级别的岗位为例）

一份优秀的技术人员的简历是怎么样的?

总的来说，一份优秀的技术人员的简历应该满足以下三个因素：

1. 技术能力的描述真实清晰，切中招聘方需求；
2. 项目经历的描述完整，具体，价值点突出；
3. 简历整体留给面试官很好的印象；

下面让我们依次看看要达成以上三点，需要满足哪些要素。

1. 技术能力

通常，「技术能力」这个部分将紧接着你的个人简介之后，放在简历的核心版面。这样设计是有道理的，因为它能够帮助雇主更快的判断你的技能是否与需求相吻合。

因此在制作这一部分内容时，应该考虑以下两点：

(1) 写什么?

应聘者在技术能力的描述上通常会犯两个错误：要么无用的东西写的太多，要么有用的东西写的太少。这里的多和少是相对于「雇主的招聘需求」而言的。

我建议每个应聘者在撰写简历的这部分时，都能够精心设计所需展示的技能，将自己熟练掌握的技能中与雇主需求重合的部分放在醒目的位置。

但无论如何，不宜展示以下内容：

- 自己对该技能只是了解而雇主又不需要的技能；
- 一些没有复杂度的，常识性的技能，例如面试高级开发职位时，表明自己掌握 Git 的使用方式或某些具体业务库的使用；

2) 怎么写?

当我们知道该写什么技能之后，我们还需要知道如何恰当的描述这些技能，通常我们会以：「了解，熟悉，熟练掌握，精通」这几个形容词来描述技能的熟练程度，让我从面试官的角度来分享一下我认为这几个词背后的含义：

- **了解**：表示你听说过这个概念，甚至了解与此概念有关的基本原理；
- **熟悉**：表示你通过 Demo 的形式实践过某个技术，或做过一两个与该技术有关的项目，但缺乏沉淀；
- **熟练掌握**：表示你在工业级环境下，通过数个项目的实践已经掌握了某种技术的核心原理，并能够灵活地应用在开发中；
- **精通**：表示你通过很多次的项目实践和潜心研究，已经对某种技术的原理和应用掌握到近乎尽善尽美的程度；

你应该尽力客观，诚实地描述自己的技术水平，既不要蓄意地夸大，也不应该妄自菲薄。在撰写该部分内容时，我建议你按照技能的熟练程度自高向低的排列，同时对于雇主明确表示需要的技能给予更高的优先级。

2. 项目经历

讲完了技术能力，接下来将分享简历中最重要的部分 -- 「项目经历」。如果一份简历满分是 100 分，项目经历所占的分数应该是 50 分以上。所以务必请你格外留意。

1) 使用 STAR 原则描述项目经历

不知道你是否听说过撰写项目经历的「STAR」原则？之所以这么问，是因为近年来我时常在各大论坛看到有关这一原则的普及文章，可奇怪的是，我很少见到有人在简历中将其付诸行动，考虑到实行这一原则的投入产出比，这实在令人感到奇怪。

事实是这样的，如果你坚持不使用 STAR 原则描述你的项目经历，我十有八九无法在你的简历中获得任何值得推敲的信息。

让我描述一下我经常看到的一类描述：

我在该项目中完成了 XXX，YYY 需求，运用了 a，b，c 技术。

这种描述方式的弊病在于，它除了告诉我求职者确实有在工作之外，再没有其他有用的信息。

让我们看看遵从 STAR 法则的项目经历描述中透露了哪些信息：（顺便一提，STAR 法则是指在描述中依次出现：Situation（情景），Task（任务），Action（行动）和 Result（结果）信息的描述一件事的原则）

XXX 项目出现 XXX 问题，我作为 XXX，负责其中的 XXX 部分，我通过 XXX 方式（或技术方案）成功解决了该问题，使 XXX 提高了 XXX，XXX 增长了 XXX。

当然，这个充满了 XXX 的范例只是抛砖引玉，目的只是为了向你展示遵从 STAR 法则，如何充分地向雇主展现你在项目开发中的产出的价值，取得的成就。并通过你的解决方案了解到你多方面的能力。

2）项目重质不重量

除了使用 STAR 法则描述项目经历，还需注意的一点是：千万不要堆叠你的项目经历！好的简历应该是短小精悍的，正如同没人感兴趣你在小学时期是否是三好学生一样，雇主通常也并不感兴趣你五年前做过哪些项目（除非它真的很优秀！）。

理想情况下，简历最好不要超过 1 页，至多不超过 2 页是一个很好的尺度。鉴于这个篇幅，你的项目经历最好也只需要包含你近 1 年做过的，你觉得有代表性的项目即可。

3. 简历印象

说完了技能和项目经历，最后让我们谈谈撰写简历时需要遵守的一些原则。当一份简历投递雇主手中时，雇主通常会大概看一下这个简历，凭经验和直觉来判断是否继续阅读这份简历，而接下来我想与你分享的，便是我认为一份好简历应该具备的「好味道」。

1）撰写简历三大原则：清晰，简短，必要；

正如标题所呈现的，一份好简历应该满足以下三个特征：

- **清晰**：这表明简历的内容应该是没有歧义，易于理解的，同时简历整体还要富有逻辑；
- **简短**：无论是生活还是工作中，我们都需要拥有一种「抓住重点」的能力，因此优秀的求职者应该尽可能在简历中就展现这一点，而展现的方式就是，尽量写出不超过一页的简历，同时让它充满诱惑；
- **必要**：招聘本身是一个相互匹配的过程，彼此展示必要的信息，能够帮助彼此最大化的节约时间，提升效率。确保简历中出现的内容都是雇主期望了解的很容易就能博得雇主的好感。

相信遵从以上三个部分的建议，你将有能力写出一份优秀的技术人员简历。

而对于前端工程师而言，则需要在此基础上添加一些技术领域内「特性」的内容，让我们进入下一个主题。

一份优秀的「前端开发工程师」简历是怎么样的?

1. 技术能力

技术能力达到什么水平才称得上是一名「优秀的」前端开发工程师呢？这里我们以「阿里巴巴淘系前端团队」的对 P6 ~ P7 级别的岗位要求做一个事例（查看职位详情）：

1. 有扎实的前端基础，了解业界先进的技术概念和开发方式，有实践经验；
2. 具备跨终端的前端开发能力，在 Web（PC + Mobile）/ Node.js / Hybrid App /
3. NativeApp 三个方向上至少精通一个方向，具备多个的更佳；
4. 至少能够熟练应用一款前端框架（ Vue.js / React / Angular / WEEX），熟悉源码和框架原理的优先；
5. 至少熟悉一门非前端的语言（Java / PHP / C / C++ / Python / Ruby），并有实践经验；
6. 具备良好的团队协作精神，能利用自身技术能力提升团队整体研发效率，提高团队影响力；
7. 对前端技术有持续的热情，个性乐观开朗，逻辑性强，善于和各种背景的人合作。
8. 具有 AB 实验的理论知识和实践经验的优先。

1~4 条描述了阿里巴巴对前端技术能力的要求，如果你经常浏览各大厂的招聘信息就不难发现，除了第 2 条希望应聘者具备跨端开发的能力之外，其余 3 条都频繁出现在各大厂的能力要求中，参照我们之前提到的技术能力的评价标准翻译过来，则是要求您在以下三个方面的技术能力能够达标：

- 至少熟练掌握 HTML，CSS 和 JavaScript；

- 至少熟练掌握一门前端框架；
- 至少熟悉一门非前端编程语言；

以我的经验来看，如果你的 1，2 条就已经符合标准，第 3 条的要求就不会非常严苛。因此如果你能够再添加上你在工作中掌握的其他熟练掌握以上级别的技术能力，就可以在这个版块做到十分抢眼了。

2. 项目经历

还记得我们之前讲过描述项目经历的最佳实践吗？

- 运用 STAR 法则；
- 精简；

除此之外，还有非常重要的一点需要注意，即「针对招聘需求选取合适的项目侧重点进行描述」。我们还拿刚才的招聘需求举例，在此招聘需求中的岗位描述里，提到了以下几项：

- 根据产品需求开发 / 维护高质量的跨终端（PC + Mobile / Node.js / Hybrid App ）项目；
- 结合当下开发场景，建立或优化提升工作效率的工具；
- 在理解产品业务的基础上，提升产品的用户体验，能够通过技术驱动业务的发展；
- 关注前端前沿技术研究，通过新技术服务团队和业务。

通过这份岗位描述，我们可以大致看出这份岗位的工作内容包含以下几个方面：

- 跨终端项目开发；
- 内部提效工具开发；
- 产品体验优化；
- 技术创新与赋能；

因此，如果你过往有类似的项目经历，并取得了一定的成就，那么在撰写项目经历这个版块时，就完全没有理由不在这里着重体现出来。

你的项目经历应该尽可能包含招聘方所需求的技术，并通过描述类似的工作经历展示出你对于未来的工作内容早已有相关的经验和准备，你越是表现出你能越快上手这份工作，你的简历竞争力就越强。

3. 简历印象

对于前端开发工程师，需要额外注意的是，你最好尽可能凸显出你的沟通表达能力，团队协作能力与自我学习能力。我认为这三大能力对于一名优秀的前端工程师而言，是非常重要的。

前端开发这个岗位站在设计与技术的十字路口，自然要和感性与理性两方面的人打交道，在工作中的很多时候，实际上充当着团队润滑剂的作用。因此你越是展现出这方面的特点，就越能博得雇主的好感。

在整个简历中，你有 2 次展现出你以上特点的机会，请你务必珍惜：

- 在项目经历描述中，通过交代清楚你在团队中的位置，以及大略描述你在团队中起到的作用；
- 在简历的「自我评价」中，写明你的软实力，并用三两句话举出一个事例来证明你所言非虚；

事实上，很多应聘者在简历中会忽视「自我评价」这个版块的设计和撰写，这是很令人遗憾的，我认为这个版块恰恰是简历中可以非常有亮点的一部分。因为它彰显了应聘者的个性与与众不同。

虽说如此，在此处仅仅是写「沟通表达能力强」是不够的——「沟通表达能力强：曾多次担任团队内技术分享主讲人」这样的「特点 + 事实证明」的描述显然是一种更好的表达方式。

结尾

至此，文章终于到了尾声。总结一下，我们谈论了简历制作过程中需要注意的以下三个部分，并分别给出了一些建议：

- 技术能力：先写岗位所需能力，再写加分能力，不要写无关能力；
- 项目经历：只写明星项目，描述遵循 STAR 法则；
- 简历印象：简历遵循三大原则：清晰，简短，必要；

而对于前端工程师而言，我们也结合阿里巴巴淘系前端团队的一个真实的 JD，在以上的三个方面，给出了如下的建议：

1. 技术能力：

- 至少熟练掌握 HTML，CSS 和 JavaScript；
- 至少熟练掌握一门前端框架；
- 项目经历：根据 JD 岗位职责，调整项目经历描述的侧重点；
- 简历印象：在项目经历与自我评价中凸显三大素质：沟通表达，团队合作，自我学习；

希望我所分享的经验能对你有所帮助，也希望你始终抱着诚实，用心的心态撰写你的简历，我的以上建议既可以作为你简历制作时的自查手册，也可以作为你未来发展的一项重要参考，真心希望你能认可，喜欢。

最后，祝你终能如愿进入心仪的企业工作。

Q: 想学习推荐系统，如何从小白成为高手？

作者 | 九胜

出品 | 淘系技术

> 现在在看推荐系统实践，会 JAVA，正在加 Python，看完这本书下一步该怎么做，求赐教。

关于推荐系统，如果忘掉所有的公式和代码，忘记所有的语言描述，脑海里就剩下几张图景，会是什么？

一张二维表格，一个拓扑图，一条时间线。

这三幅图景，是我看待推荐算法的三种视角，分享给大家便于理解。

想要学习推荐系统，除了具体的书籍和教程，更重要的是，要建立起自己的知识架构，慢慢地，往上填充具体的知识点，从而系统化地掌握某一块知识。

视角一：矩阵视角

在脑中想象一个二维的表格，每一行代表一个用户，每一列代表一个物品，表格里的每一个点代表用户对物品的操作，这个操作可以是评分，点击，点赞。其中，有些格子记录了行为，有些格子是空的。到这里，我们就建立了基本的矩阵视角，推荐问题转化成了如何补上那些空格子。

用户对物品的评分等于相似用户对该物品评分的加权平均值，这就是 user-base 的协同过滤了。换一个方向，用户对物品的评分等于该用户对其他物品的评分按物品相似加权平均值，这就是 item-base 的协同过滤。度量用户之间的相似度，把矩阵的一行——对物品的评分向量作为该用户的表示向量，那么用户之间可以计算向量的距离，可以选择任何距离公式，如余弦距离，皮尔森距离。对于物品之间的相似度，换一个方向即可。

对于任何两个物品，可以计算它们的评分差值。具体来说，两个物品有一批共同的历史评分用户，也就是矩阵里两列有交集的行，每一行可以计算一个差值，将差值平均起来，作为两个物品的距离。和上面的距离不同的，这个差值可以想象成物理中的位移，带着符号的。推荐时，某用户对于某个物品的评分，等于某用户对其他物品评分加上这个位移，再进行平均得到的平均评分。和上面的 item-base 一样的，都是列向量计算相似度，只不过相似度由距离变成了位移。这就是著名的 Slope-One 算法。

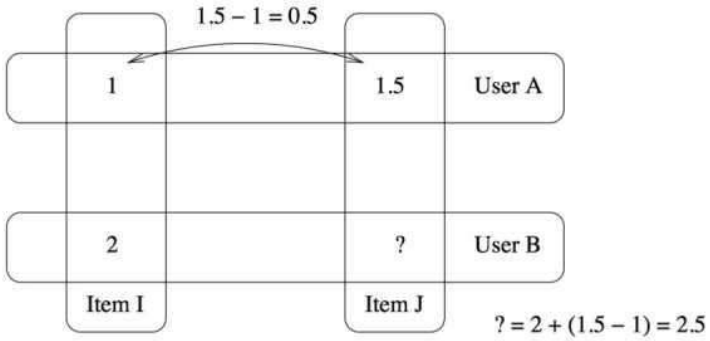
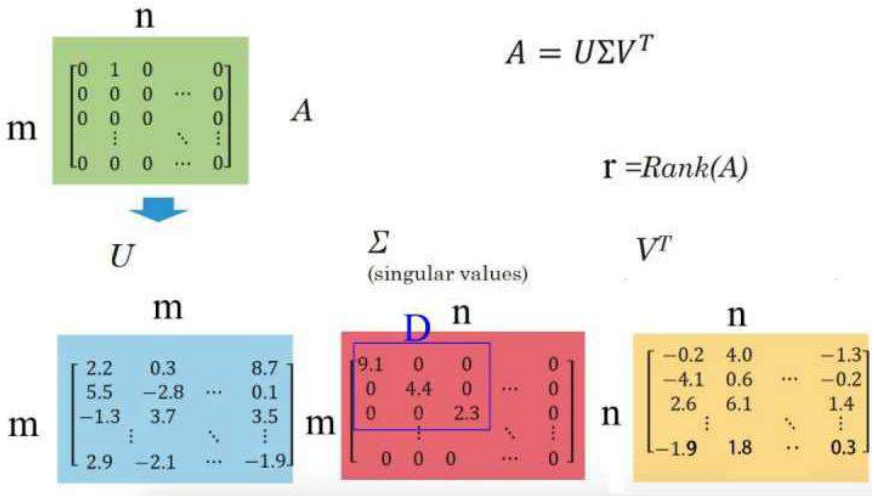


Figure 1: Basis of SLOPE ONE schemes: User A's ratings of two items and User B's rating of a common item is used to predict User B's unknown rating.

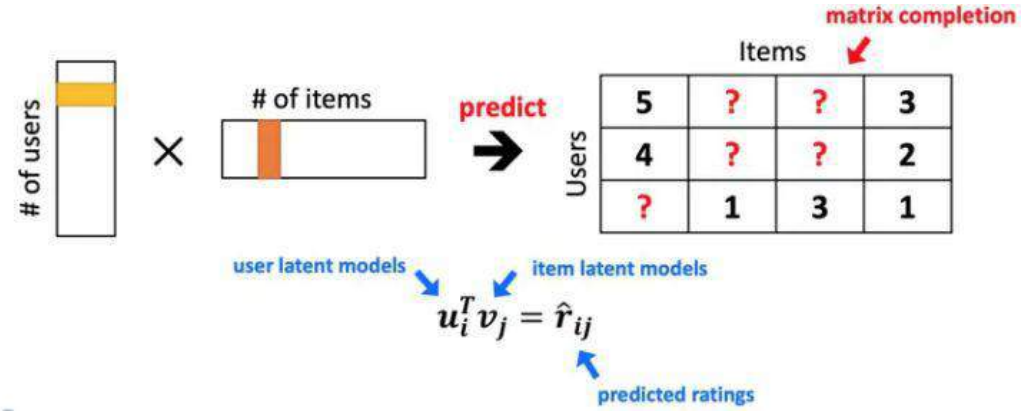
物品直接的相似度，除了上面的启发式算法，能不能通过数据本身学习所得？这就诞生了SLIM(Sparse Linear Methods)方法。矩阵 A是 $n \times m$评分矩阵，要学习一个 $m \times m$维的物品相似的矩阵 W。 A的每一行是用户的历史评分， W的每一列是每一个物品和该列对应物品的相似度，计算内积即为该用户对该列物品的评分，通过梯度下降训练来拟合真实评分。其中， W非负体现了相似度的物理意义；对角线限制为 0 避免对角线全都学习到 1 完美过拟合；添加 L1 正则产生稀疏的 W，使得结果在大规模物品集上可用; W的每一列的学习都可以看作一个线性回归模型，训练时可以彼此相互独立，因而可以分布式学习。

$$\begin{array}{ll} \underset{W}{\text{minimize}} & \frac{1}{2}\|A - AW\|_F^2 + \frac{\beta}{2}\|W\|_F^2 + \lambda\|W\|_1 \\ \text{subject to} & W \geq 0 \\ & \text{diag}(W) = 0, \end{array}$$

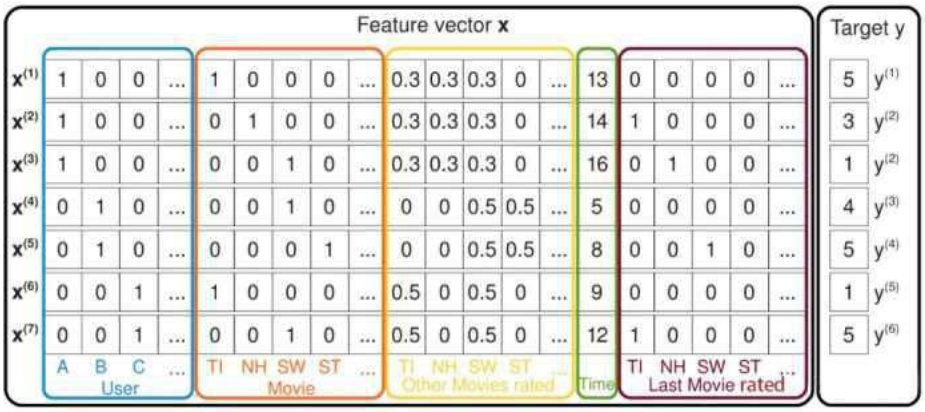
在矩阵视角下，很自然可以进行矩阵分解。SVD 矩阵分解将 n 个用户 m 个物品的大矩阵分解成三个矩阵相乘，中间的矩阵越靠近左上角的特征值越大，代表矩阵分解的主要成分，也就是说保留左上角的 $k \times k$维矩阵 D，其余的都置为零，将原来的等于变为约等于。将蓝色和红色的矩阵合并，得到一个 $m \times k$维的矩阵，每一个行代表一个 k维的用户向量，对于黄色矩阵保留其前 k行（后面的不影响计算了），每一列代表一个物品向量，用户和物品向量的内积也就是矩阵相乘后对应矩阵的值，也就是空缺处的评分，将向量索引起来就可以推荐了。



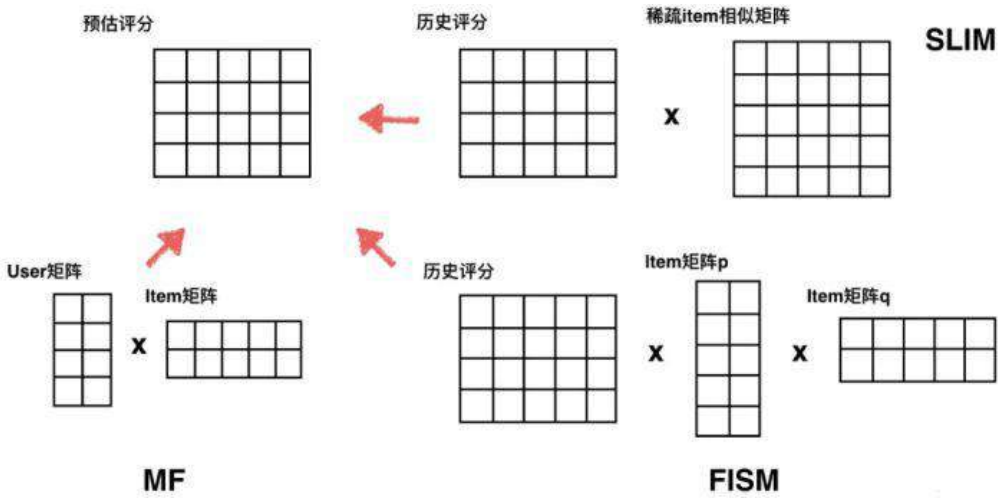
要使用 SVD 分解，待分解矩阵要是稠密的，稀疏的评分矩阵要按照统计学方法填充，如填充均值。另外，SVD 过拟合现象严重，泛化误差太大。在 2006 年 Netflix Prize 的百万推荐大奖赛上，Simon Funk 在博客公开 FunkSVD 算法。直接将评分矩阵分解成两个矩阵相乘，$n \times k$维度的用户矩阵，每一行是用户的隐式向量表示，$k \times m$维的物品矩阵，每一列是物品的隐式向量表示，用户和物品向量的内积即为预估的评分。那如何进行分解呢？随机初始化矩阵，使用均方误差作为 loss，梯度下降进行学习。这个过程中还可以加入正则项，降低泛化误差。由 FunkSVD 开始，基于 Matrix factor（MF）的方法大放异彩。



在 MF 的基础上，考虑推荐中的 side information，如用户的年龄性别，物品的类目价格。用户和物品自身或属性称作一个 field，field 之间可以两两进行矩阵分解，这个被称作二阶项，类似 BiasSVD 考虑每一个 field 都有一个 bias，这个被称作一阶项，再加上一个全局的 bias 项。这就是著名的 Factorization Machines（FM）。

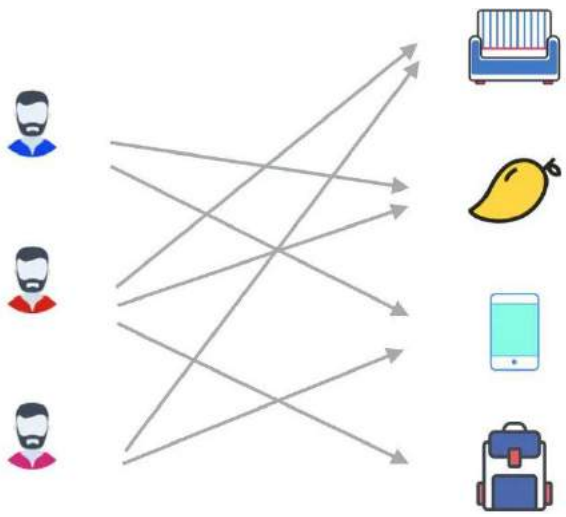


如果把上面介绍的 SLIM 和 MF 解结合起来，将物品的相似度矩阵 W 分解成 $P \times Q$ 两个低维矩阵，用户对某物品的评分，等于他过去评分过的物品在 P 中对应的向量和 Q 中该物品向量内积的和，这就是 FISM 算法。相比 SLIM 的稀疏处理，变为分解降维。最后再附上一张图，说明 MF，SLIM 和 FISM 之间的关系。



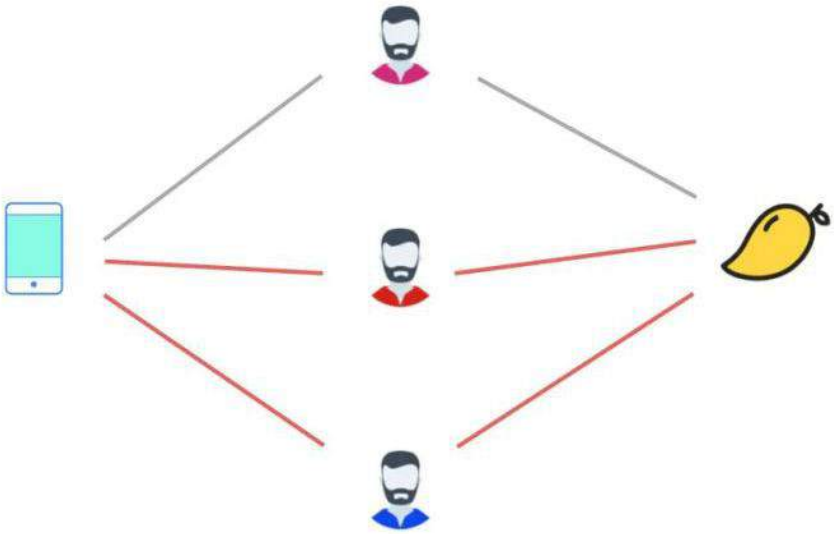
视角二：图视角

把用户和物品看作顶点，用户的评分在用户和物品之间建立起边，就得到了一个二部图；在二部图的基础上添加更多的顶点和边，形成一个更为复杂的图，辅助二部图的计算。在图的视角下，推荐问题转化成了在图上寻找高效的链接模式。

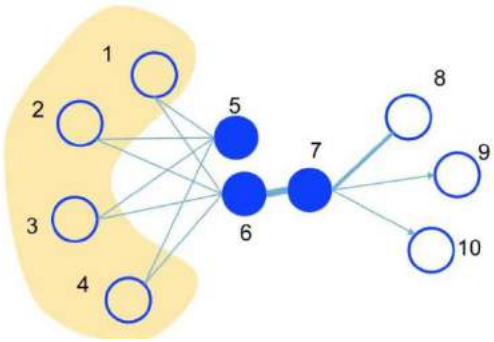


我们认为在同一个用户的历史行为中，那么两个物品之间有一条边，现在要计算两个物品之间的相似度，最朴素的思想就是数一数他们之间有多少条边。考虑每一条边权重不一样，边是通过用户建立的，用户的点击的物品越多，对应边的权重就越小。这就是 Adamic/Adar 算法的思想。

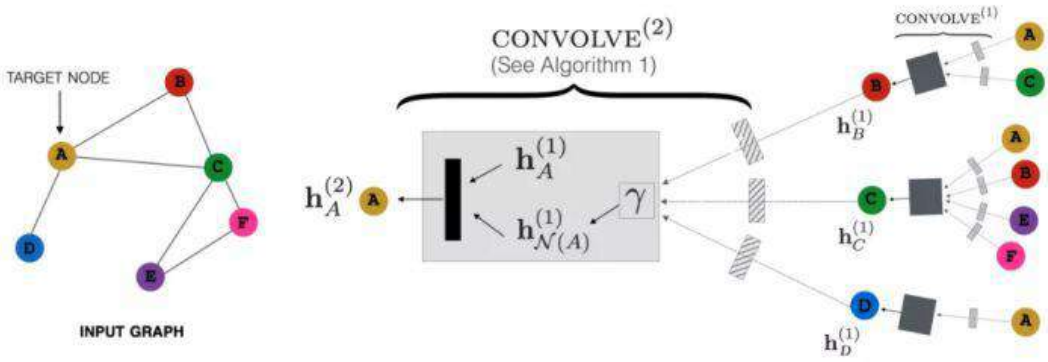
阿里著名的协同过滤推荐算法 swing，寻找图中更加稳固的形状，共同评分过两个物品的用户集合中，每两个用户和这个两个物品形成了一个四边形(下图红边为一个 swing 结构)，统计有多少个这样的结构，每一个结构的权重是不同的，这个结构里两个用户共同评分过的物品的数量越多权重就越小。



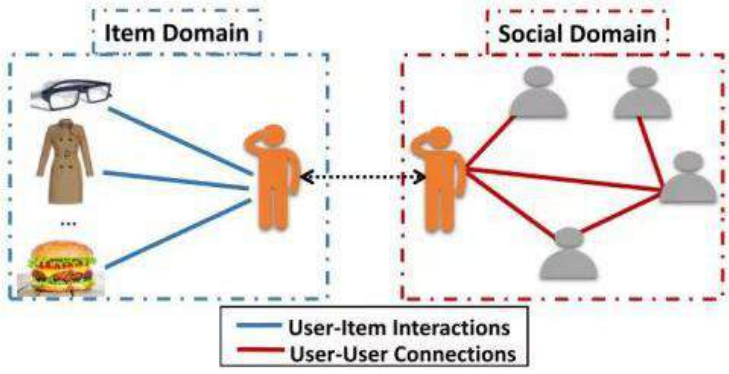
从用户和物品的二部图出发进行构图，再结合隐因子模型(Latent Factor Model)，就进入了 Graph-Embedding 的领域。DeepWalk 算法在图上随机游走深度优先遍历得到序列，然后和 word2vec 类似地使用 Skip-Gram（A 和 B 序列中相邻，用 A 的 embedding 作为特征最大化 B 的选中概率）进行训练。Node2Vec 算法在 DeepWalk 的基础上，考虑随机游走的方式，引入深度优先和广度优先的权衡，能够得到更好的更灵活的顶点隐式表示。LINE 算法考虑顶点的二阶相似，两个顶点有边为一阶相似，两个顶点有共同的邻居顶点为二阶相似，它虽不做随机游走，但可以看作是广度优先的采样。Graph-Embedding 取得了顶点的 embedding，计算相似度可以得到用户物品距离，物品物品距离，用于推荐。



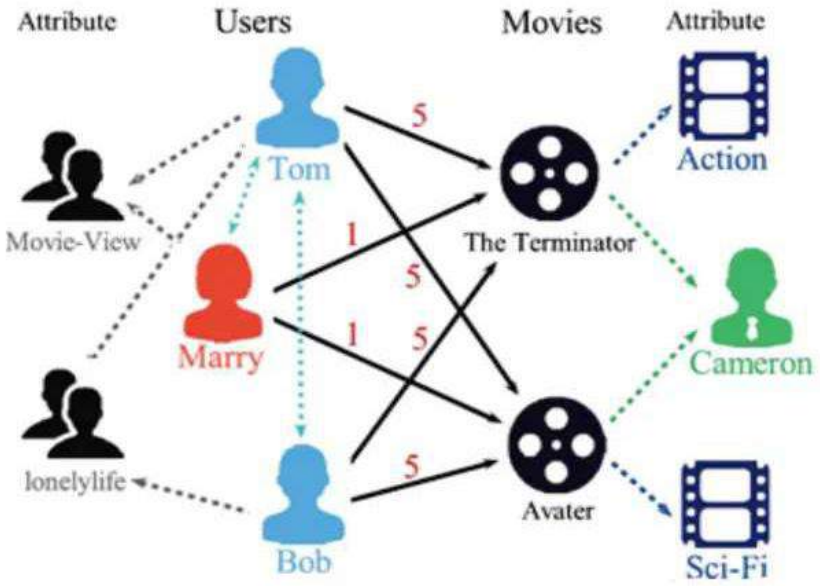
GCN（图卷积）接收拓扑图作为网络输入，可以计算每一个顶点更好的表示，相比 graph-embedding 可以有监督地为推荐目标而训练。但 GCN 在运算时，每一层都要输入整个图，在推荐系统里，物品和用户都可以是百万级别以上，实际中无法使用。GraphSAGE 通过 RandomWalk 采样，解决了这个问题，用在推荐领域就是 PinSage 算法。从某顶点出发，深度优先走 k 步，得到多个子图，组成一个 batch 进行训练，。然后按照采样的反方向做前向传播，这就是一个 k 层的图网络，下图是一个 k 为 2 的例子。



在用户和物品的二部图基础上，用户和用户根据社会关系建立起边来，这就是社会化推荐。

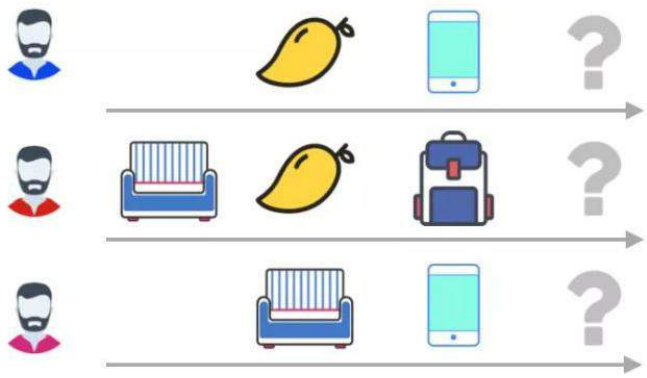


在用户和物品的二部图基础上，增加物品的属性作为顶点，建立新的边，就得到了一个异质信息网络。比如一个电影推荐系统，除了用户和电影外，还有导演，演员，电影类型，导演拍摄电影，电影属于某种类型，演员出演电影，导演与演员合作，诸如此类就能建立很多边。其中一类推荐算法叫做 meta-path，通过专家经验人工挑选出一些图中路径，如用户->演员->电影，用户->导演->电影，这样的路径称之为 meta-path，计算每一条 meta-path 的权重，将用户和物品间的所有 meta-path 联合计算评分。



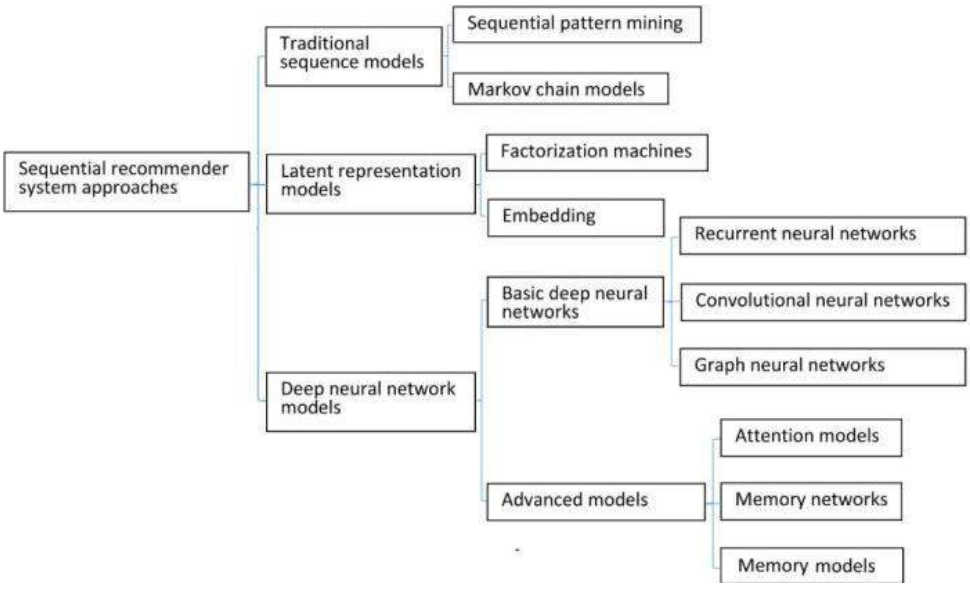
视角三：时间线

把用户对物品的行为想象成一条时间线，我们已知当前时刻前用户的物品行为序列，推荐问题被转化成了预测下一个时刻用户发生行为的物品。



假设序列中下一个物品只与上一个物品有关，可以使用马尔科夫模型 MC（MarkovChains），序列中相邻的物品间进行矩阵分解。结合上文提到的用户和物品间矩阵分解 MF，用户，当前行为物品和下一个物品三者之间两两进行矩阵分解，将三个值加起来拟合评分，就得到了 FPMC（Factorizing Personalized Markov Chains）算法。

Translation-based 推荐在序列建模中引入 Metric Learning（把行为关系和高维空间距离映射起来），用户 u，当前行为物品 i，下一个物品 j三者向量化表示，训练使得它们满足 $u+i\approx j$，推荐时只需拿到用户历史行为的物品向量加上用户向量得到下一个物品向量，然后在推荐集合中 KNN 寻找即可完成推荐。 以前模型的输入形式有限，人们通过特征处理将数据组织成模型可以接受的形式；随着深度学习的发展，数据越来越倾向于保存其原有的形式，人们通过模型设计来学习有效的模式。在时间线的视角下，直接用深度模型结构建模序列，预测下一物品，形成了一个可以发挥想象力和燃烧算力的领域——Sequential/Session-base 推荐。在 2016 年的时候，RNN 是处理序列问题的标配，它从 NLP 领域走来，诞生了 GRU4Rec 算法。受到 NLP 领域 Char-CNN 启发，CNN 的结构也逐渐用于建模序列结构，Attention 机制大火之后，RNN+Attention，CNN+Attention，RNN+CNN+Attention 被枚举了一遍。随着 google 老师的 BERT 取得 NLP 领域巨大成就，Self-Attention 以及多层的 Transformer 结构开始成为序列建模的常规配置。最近的文章里，图神经网络(GNN),Memory networks,变分自编码器(VAE)也成为了序列推荐领域的深度乐高积木。



在 CTR 预估领域，越来越多的模型直接将用户历史行为序列按照时间顺序排列，使用深度模型结构直接建模。

总结

其实如果要细数，还有一个视角叫做高维空间视角。用户和物品都是一个高维度空间里的点，空间里点之间的距离越近，代表着物品和物品越相关，用户对物品越偏好，推荐问题转化成了如何将用户和物品嵌入到高维空间里。典型的主题如 Metric Learning。不过这个视角的正交性不好，深度学习席卷推荐系统后，embedding 是个太常见的思路，前面很多的方法也都是最终把问题转化成了高维空间嵌入，如 graph-embedding，Transition-base 推荐。为了避免归类上的纠结；再加上任何一个深度网络作为 Encoder 把用户和物品 embedding，都可以归在这个视角，没有那么多令人印象深刻的典型方法，就不做单独梳理了。

To My Best Knowledge，我把自己认为推荐系统里经典且令人印象深刻的方法归在三种视角中——矩阵，图，时间线。本来想谈谈认识的，写着写着写多了，变成了一篇梳理文章。希望能对大家从偏算法的角度理解推荐系统有所助益。

小白到高手的学习，从理解开始。

引用：

https://blog.csdn.net/baimafujinji/article/details/6474273
http://xtf615.com/2018/05/03/recommender-system-survey/
Sequential Recommender Systems: Challenges, Progress and Prospects
Translation-based Recommendation

Q: 你写论文时发现了哪些非常神的网站?

作者 | 九胜
出品 | 淘系技术

> 文理科皆可，本硕博不限，专业不限。只要给你写论文带来便利的，能给人相见恨晚之感的都可以说说。

工科硬核英文论文工具。

先讲讲本文提到的工具使用流程。

在 Overleaf 使用 Latex 进行论文编辑，随时可以转化成 PDF 看效果；写一句话时，塑料英语不知道如何下笔，谷歌翻译-中翻英帮忙确立句子轮廓；粘贴到 Grammarly 中进行修改，过程中根据语法错误提示，避免低级错误，对那些常用词进行同义替换（常见词往往不精准）；如果遇到了一些词语不知道如何搭配，Linggle 通过通配符匹配，给出文献中使用的频率和例句。

为了写作，必须要阅读大量的论文。谷歌在大多数都能轻松找到 PDF，偶尔遇到收费的，这时候到 SCI-HUB 上碰碰运气；大量阅读过的论文，通过 Mendeley 来管理，它除了一个 PDF 阅读器的基本功能，按文献的维度（如作者，会议，年份）结构化和检索，方便的论文导入，多端同步；需要引用的时候，直接从 mendeley 里导出 BibTex 格式的片段；想要写公式的时候，自己用 latex 搭出公式来挺费劲的，在别人论文里见过类似的，Mathpix 可以轻松把图片转化为 latex 源码，略作修改就用到自己文章里了。
下面逐一给出简要的介绍，和工具的地址。

在线的 latex 编辑和编译工具：overleaf

论文最终展现出来的就是一个 PDF 格式的文档。

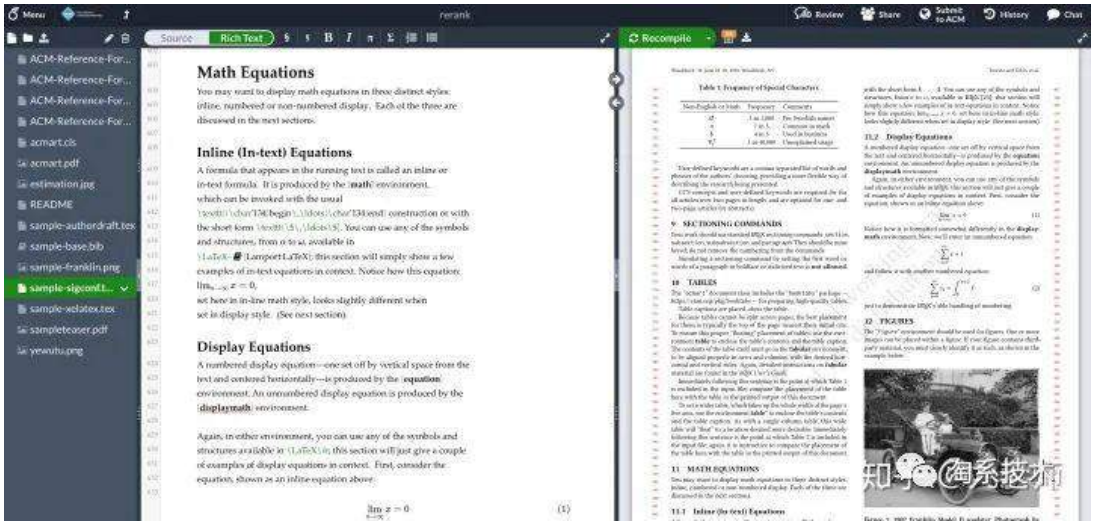
当然可以使用 word，但光排版这件事情，就能耗费你一半的精力。

正确的答案是，使用 latex，它是一个专业的排版工具，按照 latex 的语法进行写作，执行编译就能够得到 PDF 文件。它的语法包含了如何排版，虽然相比 word 上手要慢，但在排版这件事情上，入门级别的 latex 语法，你要达到精通 word 的水平。

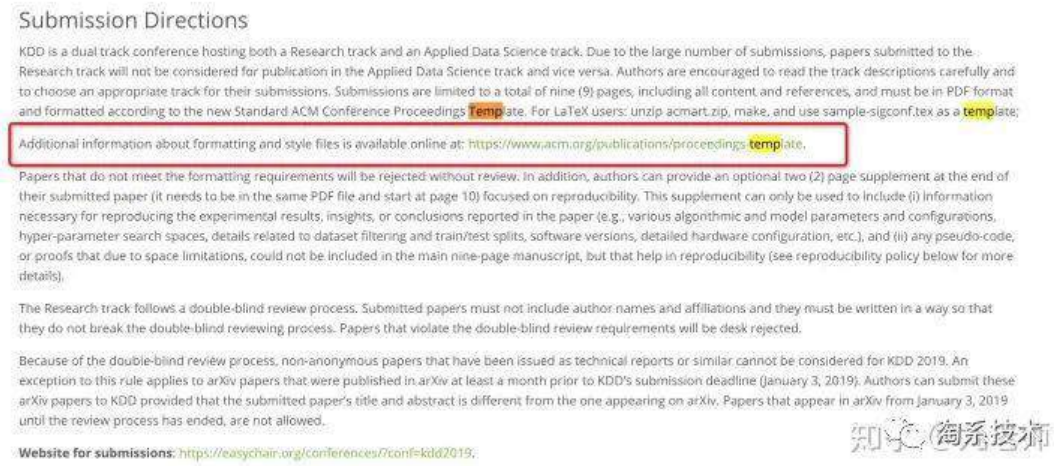
latex 如何使用呢？当然，要安装编译器，再安装编辑器，本地一通配置，偶尔会遇到些问题，凭着强大的谷歌搜索，倒也不是什么难事。配置本地环境，不如直接使用在线编辑器。

Overleaf, Online LaTeX Editor www.overleaf.com

- 注册即用，免去本地 latex 环境安装的痛苦。
- 多人合作，共同编辑。
- 富文本编辑模式，比写 latex 源码舒服些。
- 随时可以完成在线编译，查看 PDF。



按照 overleaf 的开始流程，有选择模板的过程，模板怎么选，还是要看投稿的期刊或者会议的要求。以 KDD 为例，在 KDD 2019 Call for Research Papers 页面上，给出了模板格式，看看能不能在 overleaf 上找到，即使没有，一会提供下载，自己上传到 overleaf。



Submission Directions

KDD is a dual track conference hosting both a Research track and an Applied Data Science track. Due to the large number of submissions, papers submitted to the Research track will not be considered for publication in the Applied Data Science track and vice versa. Authors are encouraged to read the track descriptions carefully and to choose an appropriate track for their submissions. Submissions are limited to a total of nine (9) pages, including all content and references, and must be in PDF format and formatted according to the new Standard ACM Conference Proceedings Template. For LaTeX users: unzip acmart.zip, make, and use sample-sigconf.tex as a template;

Additional information about formatting and style files is available online at: https://www.acm.org/publications/proceedings template.

Papers that do not meet the formatting requirements will be rejected without review. In addition, authors can provide an optional two (2) page supplement at the end of their submitted paper (it needs to be in the same PDF file and start at page 10) focused on reproducibility. This supplement can only be used to include (i) information necessary for reproducing the experimental results, insights, or conclusions reported in the paper (e.g., various algorithmic and model parameters and configurations, hyper-parameter search spaces, details related to dataset filtering and train/test splits, software versions, detailed hardware configuration, etc.), and (ii) any pseudo-code, or proofs that due to space limitations, could not be included in the main nine-page manuscript, but that help in reproducibility (see reproducibility policy below for more details).

The Research track follows a double-blind review process. Submitted papers must not include author names and affiliations and they must be written in a way so that they do not break the double-blind reviewing process. Papers that violate the double-blind review requirements will be desk rejected.

Because of the double-blind review process, non-anonymous papers that have been issued as technical reports or similar cannot be considered for KDD 2019. An exception to this rule applies to arXiv papers that were published in arXiv at least a month prior to KDD's submission deadline (January 3, 2019). Authors can submit these arXiv papers to KDD provided that the submitted paper's title and abstract is different from the one appearing on arXiv. Papers that appear in arXiv from January 3, 2019 until the review process has ended, are not allowed.

Website for submissions: https://easychair.org/conferences/?conf=kdd2019;

知乎 @淘系技术

走过这一步，已经可以编译出模板 PDF 了，可以照猫画虎地写起来了。

谷歌翻译

开始写作前，要纠结一个问题，是先写中文再翻译，还是直接写英文？我觉得，怎么快怎么来，避免拖延，快速开始。英语不过关，翻译软件来凑。

有时候不知道怎么开始一个句型，先中文拿到英文句型的轮廓，再靠着自己的塑料英文加上更多的辅助工具完善，不失为一个好的选择 。

Google 翻译 translate.google.cn

一个秘传用法：中翻英，改英文，英翻中，改中文。。。反复迭代

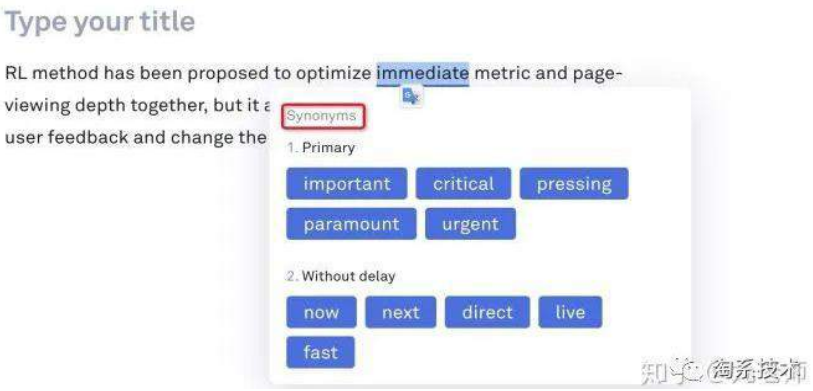
grammarly：在线的语法纠错神器

Write your best with Grammarly. www.grammarly.com

在这编辑文章的一句或一段话，语法出错了会有提示，低级的语法错误都能够避免。

语法纠错

除了语法纠错之外，还有同意替换功能，我的塑料英语能想到的词汇都太过常见，不够精准（逼格不足），选中词就可以同义替换了。



同义替换

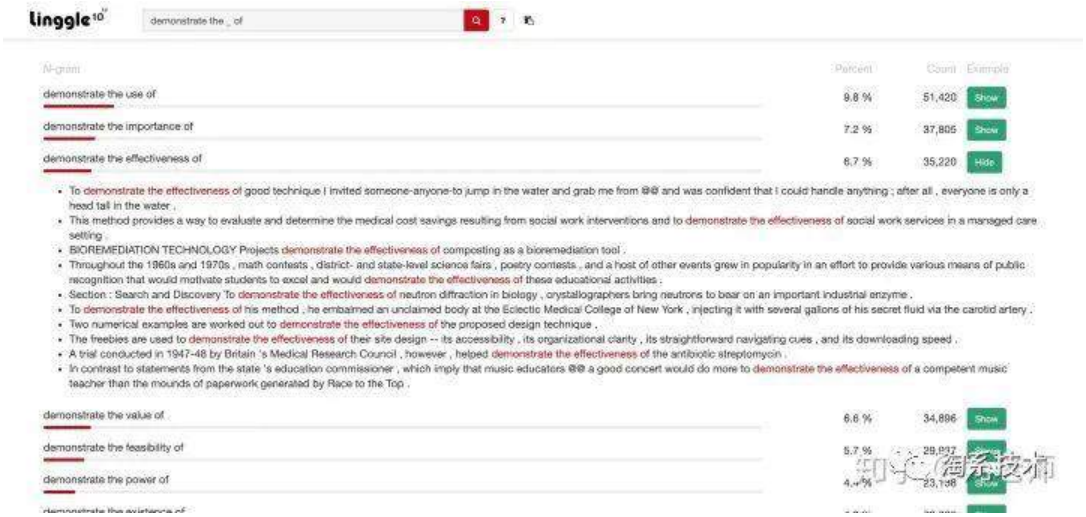
linggle:解决词语搭配问题

上面的 grammarly 可以语句的粗修，而它并不解决词汇搭配的问题，哪些短语自然，意思更精准，linggle 就能完成词汇的精修。

通过短语加通配符，它能够匹配到合适的短语，填补空缺，并给出词频和例句。

Linggle 10^12- Language Reference Search Engines linggle.com

除了下图使用了下划线匹配任意词，还有更加复杂的通配符，在首页就有教程，简单易用。



SCI-HUB：论文检索

谷歌都找不到 PDF 的时候，用它试试。

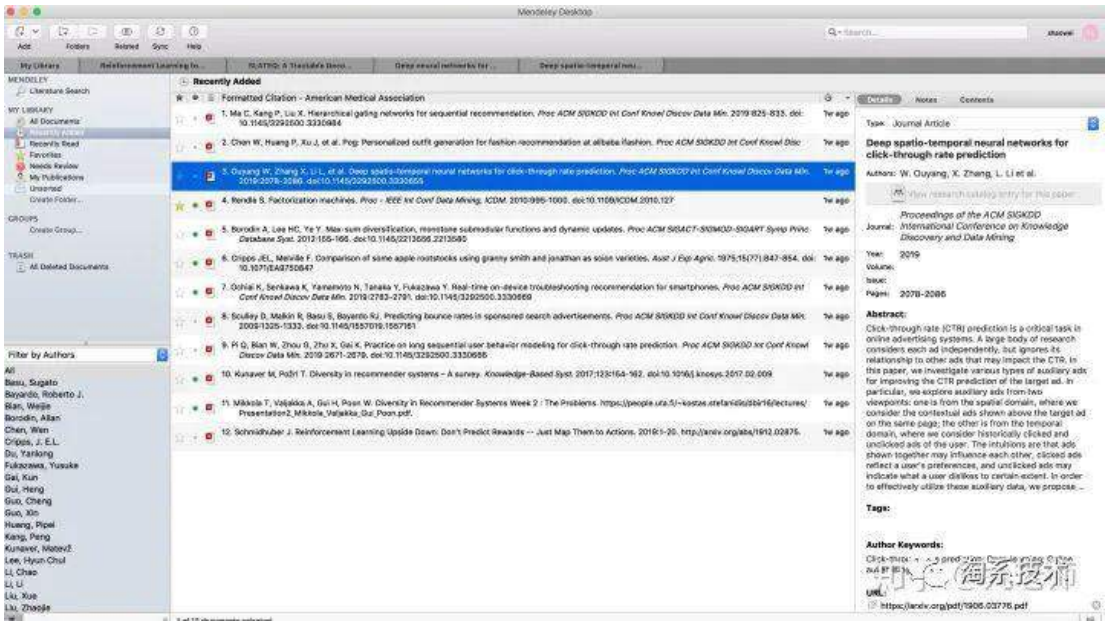
https://www.sci-hub.shop/ www.sci-hub.shop

牛逼哄哄的论文阅读器：mendeley

写作前，一定要大量阅读相关工作，一个靠谱的论文管理和阅读工具非常重要。

Reference Management Software & Researcher Network www.mendeley.com

- 论文结构化和索引，可以自己打标签，方便检索。
- 论文链接粘贴复制导入。
- 多端同步。
- 论文引用，BibTex 格式获取。



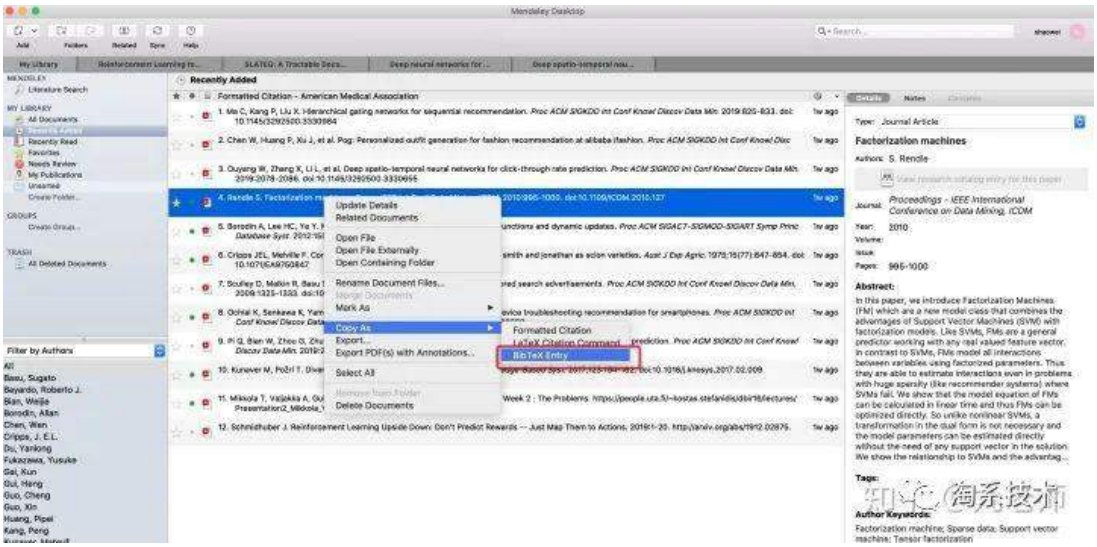
在论文中引用相关工作是不可避免的。latex 在线编辑工具中，有一个 bib 后缀的文件，里面有类似下边的片段。

```
@Article{Abril07,
  author        = "Patricia S. Abril and Robert Plant",
  title         = "The patent holder's dilemma: Buy, sell, or troll?",
  journal       = "Communications of the ACM",
  volume        = "50",
  number        = "1",
  month         = jan,
  year          = "2007",
  pages         = "36--44",
  doi           = "10.1145/1188913.1188915",
  url           = "http://doi.acm.org/10.1145/1219092.1219093",
  note          = "",
}
```

在引用的时候，latex 文档中使用\cite{Abril07}就可以引用了。

那这个片段是哪里来的?

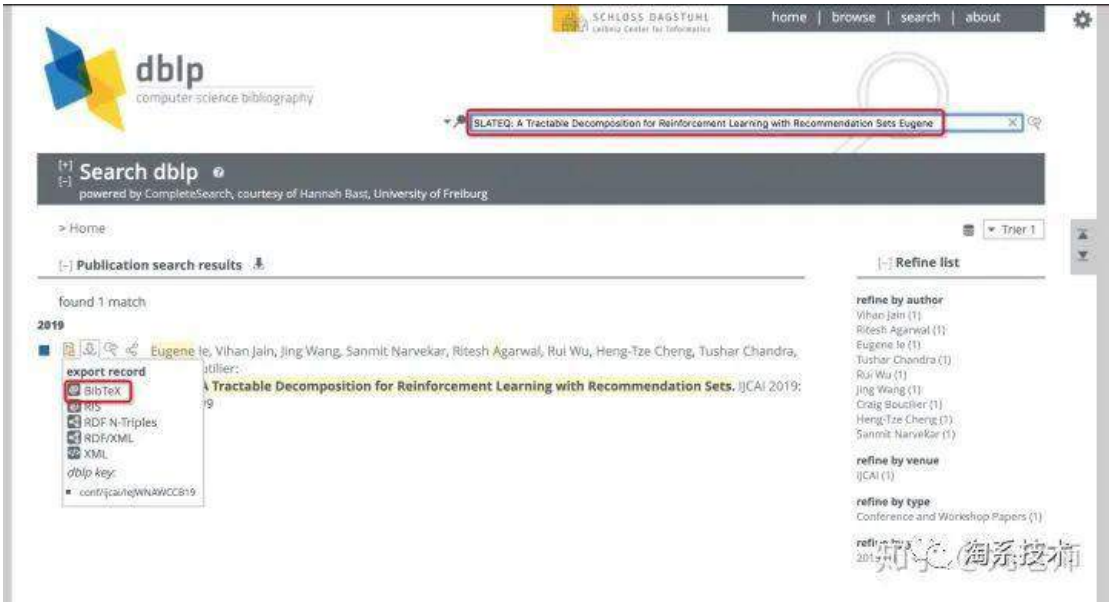
可以使用上面提到的 mendeley，方便地获取 BibTex 引用片段。



或者，推荐一个网站：

computer science bibliography dblp.uni-trier.de

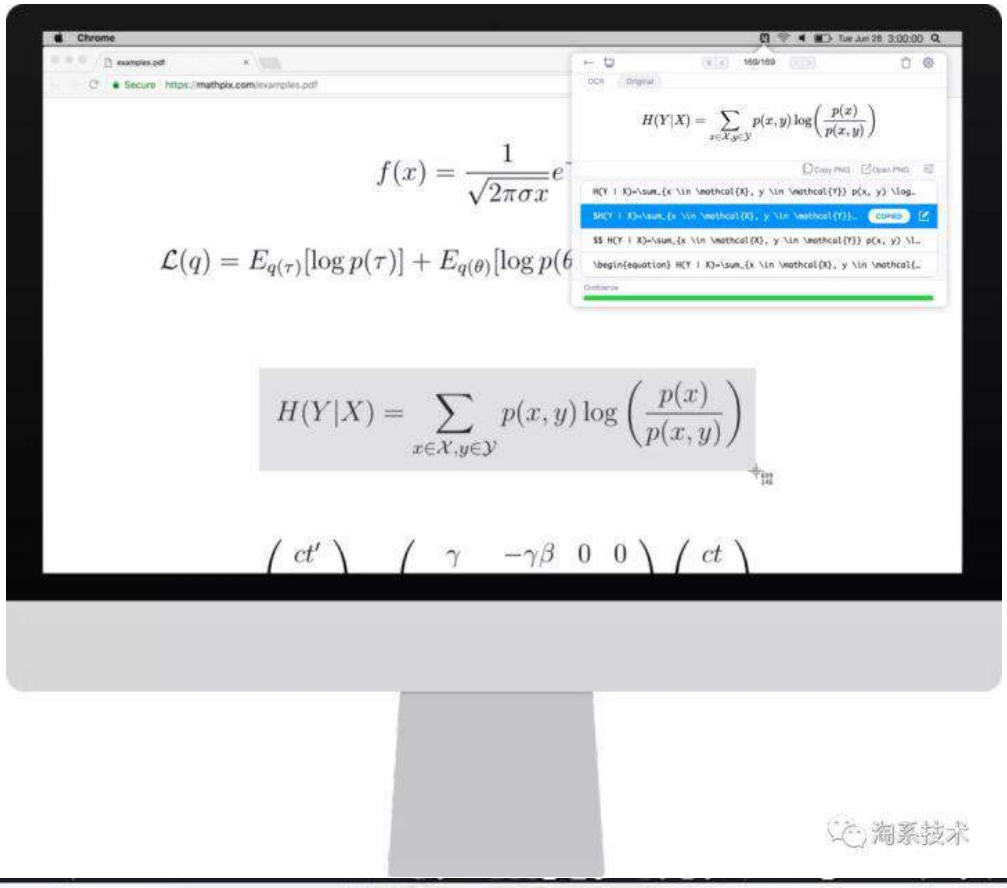
搜索论文题目，点击 BibTex，就能看到那一段引用片段了。

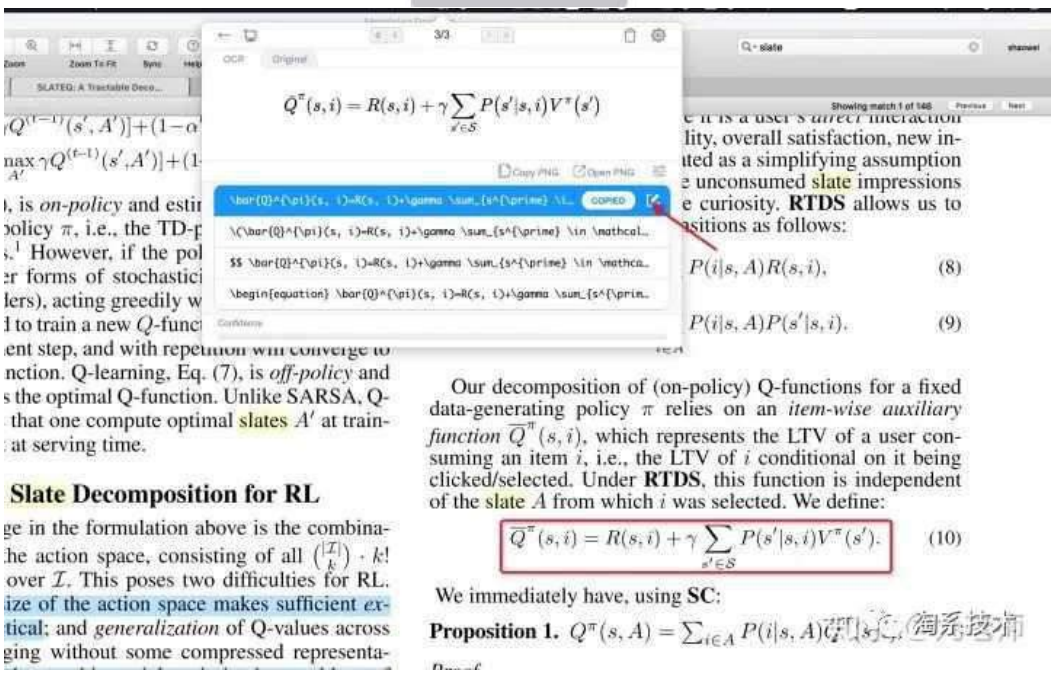


mathpix：公式图片转 latex

数学公式转 latex，有时候自己写个公式还挺费劲的，现在轻松截图，公式在手。

Mathpix Snip mathpix.com





Q: 2020 年 WEB 前端学习顺序?

作者 | 晓问
出品 | 淘系技术

2020 年 WEB 前端的学习先后顺序是什么？PC 端必须要掌握哪些知识和框架？移动端要掌握哪些知识和框架？TypeScript 和 node.js 是否为可学可不学？新手应该按照什么先后次第学习呢？新手感觉前端知识比较杂，犹如蛇吞象不知从哪里开始下口。

0x00 先说结论

我们可以把学习路线比作游戏中的段位上分，在不同的分段都有自己的定位和要锻炼的事情：

1. **青铜** - 从零开始小学生：怀着满腔的热血，看到了这一个行业的希望和未来，准备开始学习 Web 开发知识。
- 先通过 w3cschool 等免费学习资源把 HTML、CSS 和 JavaScript 的基本操作学会了。
- 写一个简单的表白页面送给你的女/男朋友，展示一下自己努力的成果，如果没有就当我没说。

2. **白银** - 懵懵懂懂初学者：懂得如何使用 HTML、CSS 和 JavaScript 三大件来实现基本页面开发功能。
- 选择一个可以覆盖多种场景、可以随自己意愿调整难度的项目尝试实现，如博客系统、记账本、Markdown 编辑器等。
- 从 React 和 Vue 这两个框架中选择一个进行学习。

3. **黄金** - 轻车熟路新玩家：懂得使用框架来实现上面所举例项目。
- 学习 Redux、Vuex 或者 MobX 等状态管理工具，并将他们使用到前面的项目中。
- 思考状态管理工具为你的项目带来了什么好处。

4. **铂金 V** - 初出茅庐新司机： 懂得如何使用脚手架创建项目，并且能将代码结构根据模块化的思想进行安排。
- 学习 TypeScript，对前面的项目进行重写，注重对数据结构和类型的控制。
- 学习 Node.js，试着配合数据库实现一个比博客系统更为复杂的 CMS（内容管理系统），如 图书馆管理系统、仓库管理系统。

5. **铂金 I** - 基本上手好司机：如果是懂得如何利用 Node.js 或 TypeScript 编写业务代码的。
- 思考在前面使用框架开发的项目中，有哪些代码是重复冗余的，有哪些逻辑是可以在多个组件之间共用的。
- 学习利用 ES2015 或更新的 JavaScript 标准，逐步替换使用框架所编写的代码。

6. **钻石 V** - 淡定自然老司机：如果是对逻辑抽象、模块封装有了一定的理解和经验的。
- 思考如何使用纯 JavaScript 对业务组件中的非渲染、非 DOM 相关代码进行抽象。
- 引入单元测试工具，对纯逻辑代码进行测试，争取覆盖率达到 80% 以上。

7. **钻石 I** - 赛道新手初学者：如果上面的条件你都已经满足了。
- 思考不同的代码哲学（OO、FP 等）、不同的代码结构（MVC、MVVM 等）的区别。
- 思考不同的框架之间设计的初衷，思考不同的编程语言中对同一类问题不同解法的区别。

到这里我划了一条从 0 到高级前端工程师级别的纯技术路线。相信有不少有经验的同学会发现中间我省略了不少内容，但不难发现路线中从前半段的“学习”逐步变成后半段的“思考”。

优秀的工程师除了需要有在纯技术领域的沉淀以外，还需要更多对技术、团队、ROI（投资回报率）的思考，当然这依然不足以支撑我们平稳地渡过“程序员 35 岁危机”，前面的路还有很长，钻石往上还有王者呢，谁说程序员就是青春饭碗的？

回想起很多年前我也跟你一样是一个完全的新手，从 0 开始慢慢自学摸索 Web 开发，甚至后来我也没有进入科班学习计算机，那么来听听我作为一个“前人”是如何完全靠自学至今的故事吧。

0x01 我的从 0 开始

我是一个完全从自学开始的前端工程师，想起来第一次接触前端就是初中那会特别流行合租 VPS 然后注册一个 .tk 的免费域名。而作为一个刚入门 Web 开发不久的小屁孩来说，用这种方式一探"大人的世界"属实让人兴奋。

而当时最流行的博客管理软件就是用 PHP 写的 WordPress，作为一个十分成熟的 CMS 软件来说 WordPress 当时就有了非常丰富的社区资源，比如主题、模板、插件等等。而作为一个十分注重个性化的小屁孩来说，当然是要自己做一个主题的啊！于是我就从此踏上了 Web 开发的不归路，在此之前我所接触的都是 Visual Basic 这样的 Native 的语言。

以 WordPress 主题作为切入点，我开始学习 PHP 用于调用 WordPress 的 API 并输出内容、学习 HTML 用于写主题的模板、学习 CSS 用于"装潢"我的博客、学习 jQuery 用于实现页面动态效果。

是的，那个时候基本上大部分人接触的是 jQuery 而不是 JavaScript，一个 $ 函数就可以完成非常多的效果这让我第一次感受到了"框架"所带来的价值。于是便一步一步地发生了以下事情（不一定完全对，毕竟时间过太久了）：

- 我发现页面上的一些样式效果无法在 IE 浏览器上正常显示，于是我就开始到网上学习 CSS 在 IE 的各种特殊处理，包括 reset.css、normalize.css 等工具的使用；
- 每次点击链接都要刷新页面，在那个网速不怎么好的年代体验非常糟糕，于是乎就开始研究怎么用 jQuery/JavaScript 实现不需要刷新页面的情况下切换页面的内容；
    - 通过查看文档发现浏览器支持一种叫做 XMLHTTPRequest 的技术，可以让我们不需要通过跳转的方式从服务器获取到信息，从这里开始了解到 HTML、XML 和 JSON 三种不同格式的区别；
    - 第一次知道了可通过服务器传递 JSON 格式的纯数据，然后前端通过 JavaScript 对数据进行解析，并且结合前端的模板引擎渲染成完整的 HTML；

- 从这里又可以学习到如何通过 URL 中的 path、query、hash 以及 POST 和 PUT 请求正文等信息向服务器传递信息，服务器通过这些信息动态地对各种数据进行处理并返回结果；
- SPA（Single Page Application）开发习惯初见雏形；

这样我就来到了“白银”阶段了。

0x02 接触 Node.js

当我正在愉快地设计着 WordPress 的自定义主题时，偶然间我在某前端网站上了解到了一个新的技术 —— Node.js。与它的相遇改变了我以后的学习路径，影响至今。

2009 年 Ryan Dahl 发布了一个基于 Chrome JavaScript V8 引擎开发的程序运行环境 Node.js，它允许开发者在除了浏览器以外的地方运行 JavaScript 语言，并且提供一些标准库允许 JavaScript 脚本启动进行启动一个 HTTP 服务端应用这种以前在浏览器无法完成的事情。

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(8124, "127.0.0.1");
console.log('Server running at http://127.0.0.1:8124/');
```

这一份代码是 2010 年写在 Node.js 官网的一段实例代码，机缘巧合之下我被这么一段简单的代码深深地吸引住了，虽然当时安装它仍需要从 GitHub 上克隆整个项目代码到本地并依次运行以下指令：

```
$ ./configure
$ make
$ make install
```

这一次编译就得花上至少十分钟，但完成安装后运行上面的一段代码，并在浏览器中打开 http://127.0.0.1:8124/，然后在浏览器上看到 Hello World 字样时仿佛新世界的门打开了。因为当时我所接触过的服务端程序只有 PHP，而 PHP 本质上就是一个模板引擎，它并不能很直观地处理请求本身而是借助 CGI 进行响应。能做更多的事情，这件事情对刚学习编程不久的新手来说是具有很大诱惑力的。

从这里开始，Node.js 配合 npm 便开始了长达 10 年的快速发展。从纯服务端应用开发，到开发工具、工程工具，再到如今的 FaaS（Function as a Service，Serverless）开发方式。Node.js 已经成为 Web 工程师不可或缺的一项技能，不管是用来开发服务端应用还是开发工具类应用，甚至是使用 Electron 开发桌面端应用还是配合 React Native 开发移动端 App，Node.js 能让前端工程师了解更多系统级别的概念，如网络、I/O、内存、文件系统等等，这些很多都是原本在浏览器端上看不到的。而学习这些知识对你理解前端开发背后的一些原理有非常好的价值，就跟学习算法一样。

结论：请学习 Node.js 和其中涉及到的一些基本计算机原理。

0x03 框架时代

当我在做 WordPress 主题的时候，绝大部分的主题开发者都会在前端做一些简单的效果，甚至有甚者会通过 JavaScript 实现一些原本只能通过后端来完成的事情，比如文章列表、文章内容的加载和渲染。

而当年这些主题开发者基本上都会使用 jQuery 来进行这些 JavaScript 的操作，因为纯手写 JavaScript 在当时来说非常的繁琐（ES4 时代，很多现在被广泛使用的原生 API 都仍未具备）所以当时 jQuery 就是大家的首选方案。

从非常早的 PrototypeJS、后来的 jQuery、进入 MVC 时代的 Backbone，AngularJS 开启 MVVM 模式，React 引入 FP 的概念，Vue 成功开启了渐进式开发体验的道路。一路下来一地的鸡毛，被各路人马诟病前端领域一个月开发一个新框架，“学不动了”。然而作为

一个也写过框架、写过工具类库的开发者，我很喜欢用一个经常用于泛科技领域的例子来类比前端领域：科技的终极目标，就是让人民感觉不到科技。

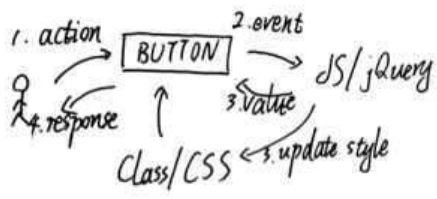
jQuery 时代，前端开发者使用 JavaScript 的模式是从页面中获取 DOM 元素，添加事件，然后通过 `class` 和 `style` 对页面进行动态地变更，以完成对用户行为的响应；

Backbone 时代，原本用在桌面端软件开发中的 MVC 模式被引入到了前端开发中，前端开发者们发现 Web 开发的复杂度已经需要用这些更成熟的开发模式进行管理了；

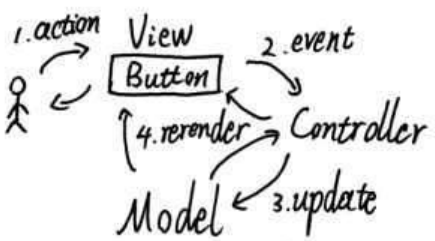
AngularJS 时代，从这里开始 Google 把数据双向绑定模式带到前端开发中，将原本需要通过 JavaScript 控制 DOM 元素这一繁琐的操作变成了只需要关心 Model 层需要改动什么内容即可。

而 Vue 则将这种模式的开发成本降低到了一种相当可观的程度，让很多新手开发者也能很简单地入手这种便捷的开发模式。

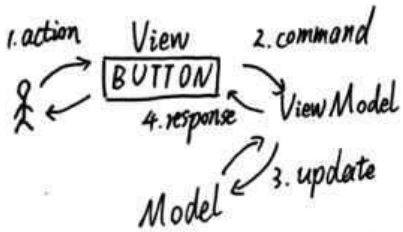
React 时代，Facebook 的科学家们把函数式编程的思想引入到前端开发中，注重的是数据链路的可跟踪、可回溯、可管理，让整个数据链路是尽可能以单链路流转。



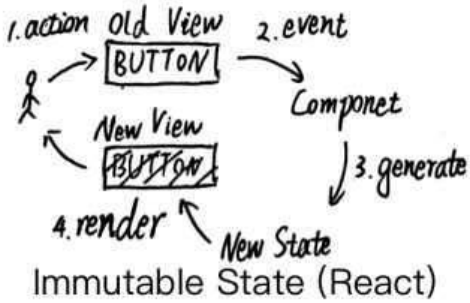
jQuery/Vanilla JS



MVC



MVVM (Angular/Vue)



Immutable State (React)

虽然前端领域常被说“一个月一个新框架”，但实际上每一个框架在迭代的过程中都是解决了它们所在业务场景的实际需求的，并不是“拍脑袋”地想要把每一个技术细节做出一个 break change。

而目前我目前推荐的学习的框架是 React 和 Vue：

- 同样都是目前最流行的框架之一，而且可以预见未来 3~5 年内都是能满足找工作的需求的；
- React：引入函数式编程（Functional Programming）的概念，使得写代码的思路更加严谨，更具有可维护性和逻辑可导性；
- Vue：将 MVVM 模式变得非常简单易于入手，把 Progressive JavaScript Framework 的定位实践得非常到位。且如今 Vue 3.0 的 Composition API 更是在某种程度上将 Hooks 的玩法实现得比 React Hooks 更优；

结论：请不要害怕学习！不要惧怕新技术！

0x04 工程之路

虽然我在接触了框架和 Node.js 之后，发现 JavaScript 除了能实现一般只用于展示内容和呈现简单交互以外还能做更多的事情。但本质上还是围绕着多个页面进行页面上 DOM 元素的控制，而直到我打开了 Google 的一些网站时，我才发现原来网站除了能叫页面以外，还能称之为“应用”。自从 Google 上线了一个完全不需要刷新页面就能完成所有事情而且体验很不错的 GMail 之后，我们发现网页原来也是可以承载那么复杂的逻辑和应用场景的。大家的热情异常地高涨，想着能不能让自己所负责的项目也有这么厉害高级的样子。但随着项目不断地复杂，代码规模也变得非常难以管理，而这个时候就需要工程化的引入。

工程化协作

对于企业来说除了研发效率要足够高以外，研发链路的安全、合规也是同样重要的。什么叫安全合规？可管理的代码版本、可控制的发布流程、可管控的灰度机制，都是大厂用于保证

项目流程稳定进行的必要工具。有很多初学者或者还没有大公司经验的同学在写项目时都是单打独斗的，但更多的一线项目都需要至少 2~3 个甚至更多的人员一同参与开发的。而这种时候，因为每个人的水平和开发习惯都是不一致的，而这些不一致就直接导致整体研发效率和项目进度受到极大的影响。所以就需要一种能够让大家在一个水平线上进行开发的模式，工程化需求便应运而生。

- 版本控制
  - Git：GitHub、GitLab、Coding……
  - SVN：BitBucket、Google Code……

- 代码样式检查工具 JavaScript/TypeScript：ESLint
- 测试工具
  - 单元测试：Karma、Jest、Mocha……
  - 持续集成：CI

- ……

工程化开发工具

从直接将 JavaScript 代码用 <script> 标签，到需要将 jQuery 文件和主要程序文件分别引入，再到 Node.js 出现后使用 npm 进行依赖库管理并使用 webpack 进行打包和压缩。工程类工具的发展见证着前端工程近十年的发展历史，对目前我们所常用的工程工具有更好的了解和实践，绝对是通往优秀路上不可或缺的一步。

- 依赖包管理工具：npm、yarn
- 打包工具：webpack、rollup
- 脚手架工具：create-react-app……

工程化开发语言

相信很多同学都听说过 JavaScript 诞生之初的一些轶事，比如根本没有特别多的严谨思考，或者在非常多的场景中十分地晦涩，比如隐性转换等。有人认为 JavaScript 能发展到如今的地位跟它的这种“灵活度”或者“松散度”有关联，虽然在某种程度上确实因为这种特性造成的 JavaScript 学习门槛比较低而间接导致。但就如我上面所说，当项目规模和人员规模不断发展乃至膨胀过后，这些特性会逐渐表现出来非常糟糕的体验：

1. 团队之间因为没有良好的技术文档沉淀，信息不对等的情况直接导致代码在没有良好的单元测试时出现逻辑冲突；
2. 第三方依赖库的 API 在设计上大量使用了 JavaScript 松散的特性，导致使用方在引用时频繁出现“迷惑”的状态；
3. 当需要使用 JavaScript 与其他语言（特别是强类型语言）进行交互时，JS 过于松散的习惯会让对接方感到非常迷惑，对于双方的实际接入成本会比前期预估的大得多；

为了解决这种情况，来自不同编程领域的大牛们都纷纷开始想办法，于是乎便诞生了非常多的“轮子”：

- Java 系：Scala.js、ClojureScript
- Go 系：GopherJS
- Microsoft：TypeScript
- Facebook：Flow、Reason

目前 TypeScript 已经影响了前端乃至整个 Web 领域的开发生态，TypeScript 之父 Anders Hejlsberg 创造过 Turbo Pascal、Delphi、C# 等在整个计算机科学领域都举足轻重的语言，而 TypeScript 又再次创造出翻天覆地的变化：

- 强类型的引入能让我们在写代码的时候从值优先的思维转变成类型优先；
- 强类型的引入能帮助开发工具 (IDE 等) 更好地为开发者提供便利性能力，如智能补全、类型检测、编译时检查等等；
- TypeScript 可以让 JavaScript 更好地与其他语言进行交互，甚至转换为其他语言；

工程化通用组件

当需求不断变多后，“爱偷懒”的工程师们就会把经常用到的内容进行抽象，比如从很早以前就有的 ExtJS、Twitter 工程师发布的 Bootstrap 再到今天的 Ant Design、Element UI 等，都帮助我们更快更好更稳定地完成一些通用页面能力的开发。

- React：Ant Design、Fusion Design
- Vue：Element UI、iView、Ant Design of Vue

0x05 逻辑抽象能力

随着我对 JavaScript 应用的编写经验不断增加，我所尝试的技术和场景也在不断地变得更加复杂。而当逻辑代码变得越来越复杂时我也渐渐发现一个新的问题，很多时候我所编写的逻辑代码是相似的，但相似之余其中的一些细节不尽相同，而这些代码往往是后期维护成本最高的。这就让我感到十分困惑，如何让我的代码写起来没有那么繁琐的同时，又不丢失原本代码的应有逻辑呢？这就让我想起了之前学习的框架，它们的实现原理不就是把原本我们写得非常繁琐的逻辑代码进行压缩，让我们写起来更加简洁直观吗？

这是我曾经面试过的一位校招候选人写的代码，其背景是用于快速判断自走棋类游戏中不同的增益能力（Buff）的成立状态。但显然这样的代码在实际开发中是绝对不允许存在的：代码逻辑过于冗余；

1. 一旦通用判断逻辑出现变动，需要每一个都进行手动维护；
2. 没有良好的可维护性；
3. 所以我便提出如何让这些代码写得更加“优雅”和利于维护。

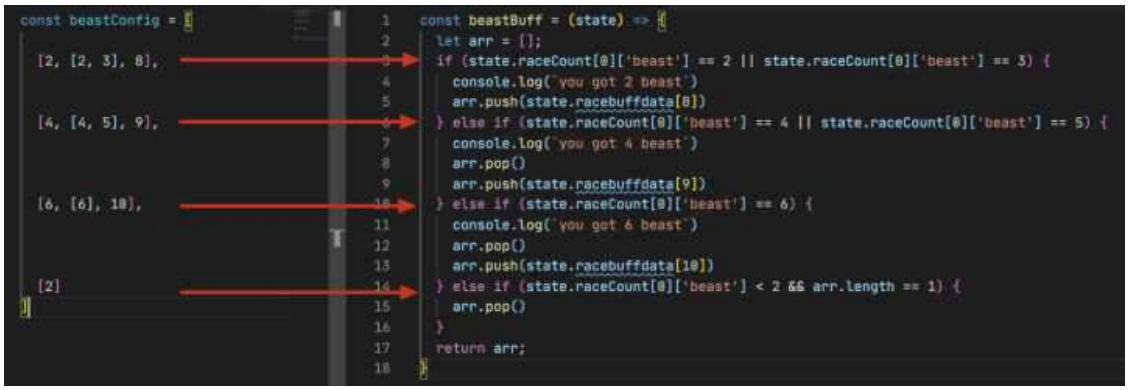
```
export default {
  beastBuff: (state) => {
          let arr = [];
          if (state.raceCount[0]['beast'] == 2 || state.raceCount[0]['beast'] == 3) {
                console.log(`you got 2 beast`)
                arr.push(state.racebuffdata[8])
```

```
        } else if (state.raceCount[0]['beast'] == 4 || state.raceCount[0]['beast'] == 5) {
            console.log(`you got 4 beast`)
            arr.pop()
            arr.push(state.racebuffdata[9])
        } else if (state.raceCount[0]['beast'] == 6) {
            console.log(`you got 6 beast`)
            arr.pop()
            arr.push(state.racebuffdata[10])
        } else if (state.raceCount[0]['beast'] < 2 && arr.length == 1) {
            arr.pop()
        }
        return arr;
    },
    caveclanBuff: (state) => {
        let arr = [];
        if (state.raceCount[1]['caveclan'] == 2 || state.raceCount[1]['caveclan'] == 3) {
            console.log(`you got 2 caveclan`)
            arr.push(state.racebuffdata[11])
        } else if (state.raceCount[1]['caveclan'] == 4) {
            console.log(`you got 4 caveclan`)
            arr.pop()
            arr.push(state.racebuffdata[12])
        } else if (state.raceCount[1]['caveclan'] < 2 && arr.length == 1) {
            arr.pop()
        }
        return arr;
    },
    demonBuff: (state) => {
        let arr = [];
        if (state.raceCount[2]['demon'] == 1) {
            console.log(`you got 1 demon`)
            arr.push(state.racebuffdata[5])
        } else if (state.raceCount[2]['demon'] < 1 && arr.length == 1) {
            arr.pop()
        }
        return arr;
  }
  // ...
}
```

我们不难发现这几个 `xxxBuff` 函数中的逻辑都非常接近，但也各有不同。那么如何能将这段代码进行优化和抽象呢？我当时给 TA 提出了一份示例代码：

```
const beastConfig = [
  [2, [2, 3], 8],
  [4, [4, 5], 9],
  [6, [6], 10],
  [2]
]
```

这份代码中的每一个数字在上面的 beastBuff 函数中都可以一一找到，那么要怎么将它们复用到逻辑代码中，实现与原本的代码一样的功能呢？



我同样给他写了一份参考答案：

```
const generateBuff = (rate, configArr) => {
  return state => {
    const arr = []

    for (const [ output, conditions, index ] of configArr) {
      if (conditions && index) {
        // Buff calculating
        const isHit = conditions.some(condition => state.raceCount[0][race] == condition)
        if (isHit) {
          console.log(`you got ${output} ${race}`)
          arr.pop()
          arr.push(state.racebuffdata[index])
          break
        }
      } else if (state.raceCount[0][race] < output && arr.length === 1) {
        // Last condition
        arr.pop()
```

```
        }
      }

      return arr
    }
}

export default {
  beastBuff: generateBuff('beast', [
      [2, [2, 3], 8],
      [4, [4, 5], 9],
      [6, [6], 10],
      [2]
  ]),

  caveclanBuff: generateBuff('caveclan', [
      [2, [2, 3], 11],
      [4, [4], 12],
      [2]
  ]),

  // ...
}
```

原本代码里面通过 hard code 实现的判断逻辑，通过观察其中的共同点，并思考能否通转换为可抽象部分，这同样也是一名优秀的工程师所必须具备的能力。

Be D.R.Y.! (Don't repeat yourself)

0x06 更高层次的思考能力

随着我对不同业务、不同场景和不同代码难度的不断探索和研究，我发现在前端领域乃至整个编程领域里，不同的框架和架构层出不穷地发展，其实在根本上就是各种实际业务场景在寻找更合适的 Better Practice（更好实践）。

就如前面的所说的那样，不同的框架作者在开发的时候会采取不同的代码结构甚至代码哲学，这些不同的思维角度可能在框架的源码中并不会直接表现出来。但我不会说研读源码完

全没有用！因为研读源码最起码可以学习其中的一些 trick 或者代码习惯。

但更重要的是理解从 API、系统架构上进行思考，因为只有多思考了，你才能逐渐变得比其他人更加对不同的技术游刃有余。

EOF

这一个流程并不是严谨的学习路线，更多的是我个人的一些经验总结。

当然除了我所提到的学习知识点以外，还有很多不同的分支对应着不同的实际业务和场景，比如配合 Electron 开发桌面端应用、配合 React Native/Flutter 开发移动端应用、配合 Node.js/QuickJS/FibJS 开发嵌入式应用、配合 TensorFlow.js 开发适用于前端甚至适用于边缘计算的机器学习应用、配合 WebAssembly 将 Web 应用的使用体验提升到接近原生应用的境界……

关于 JavaScript 有一个很有名的预言：

凡是能用 JavaScript 重写的，终将会使用 JavaScript 重写

无论这句话会不会最终完全实现，但目前我们已经能看到很多应用逐渐通过 Web 应用的形式云端化，比如 Photoshop、音视频编辑软件、代码编辑器甚至是大型游戏等等原本我们完全没想到可以运行在浏览器中。

前端开发困难吗？不困难、门槛相对比较低。简单吗？不简单，通过相信看到这里的你也已经有所体会了。当然实际要如何选择路线和方向，还是你自己所遇到的经历和机遇来决定的。

第三部分 年度精选必读好书

- ✧ 技术硬核参考
- ✧ 商业思维养成
- ✧ 文化&科普
- ✧ 工具&方法

技术硬核参考

- 《深度探索 C++对象模型》（[美] Stanley B·Lippman ）
- 《你不知道的 JavaScript》（[美] Kyle Simpson ）
- 《深入理解 ES6》（[美] Nicholas C. Zakas ）
- 《程序员修炼之道》（[美] Andrew Hunt / David Thomas ）
- 《架构整洁之道》（[美] Robert C. Martin ）
- 《支持向量机导论》（[美] 克里斯蒂亚尼 ）
- 《代码的未来》（[日] 松本行弘 ）
- 《Head First 设计模式》（[美] 弗里曼 ）
- 《领域驱动设计》（[美] Eric Evans ）
- Effective Java , Joshua Bloch
- Java Concurrency in Practice, Brian Goetz
- Deep Learning, Ian Goodfellow

商业思维养成

- 《刷新》（[美] Satya Nadella）
- 《商业的本质》（[美] Jack Welch）
- 《规模：复杂世界的简单法则》（[英] Geoffrey West）
- 《债务危机》（[美] Ray Dalio）
- 《重构》（[美] Martin Fowler）

文化&科普

- 《The Book of Why》（[美] Judea Pearl / Dana Ma）
- 《魔鬼经济学》（[美] 史蒂芬·列维特）
- 《罗马人的故事》15 卷本（[日] 盐野七生）
- 《孙子兵法》（孙武）
- 《博弈论》（王力哲）
- 《QED》（[美] 理查德·费曼）
- 《枢纽》（施展）
- 《叩响命运的门》（马小平）
- 《星空的琴弦-天文学史话》（汪洁）
- 《激荡三十年》（吴晓波）

工具&方法

- 《金字塔原理》（[美] Barbara Minto ）
- 《高效能人士的七个习惯》（[美] 史蒂芬·柯维 ）
- 《软技能 - 代码之外的生存指南》（[美] John Z. Sonmez ）

第四部分 淘系开源项目盘点

- 项目一：Rax
- 项目二：飞冰（ICE）
- 项目三：MNN
- 项目四：coobjc
- 项目五：FlutterBoost
- 项目六：Sandbox
- 项目七：JVM-Sandbox-Repeater
- 项目八：Midway
- 项目九：Pipcook
- 项目十：3D-FUTURE & 3D-FRONT
- 项目十一：TransmittableThreadLocal

第四部分 淘系开源项目盘点

开源推动着互联网技术的发展。多年来，阿里巴巴淘系技术一直积极拥抱开源事业，无论是开源软件的应用、回馈以至自研技术的开源都非常活跃，近两年我们更是开源了 MNN、飞冰 ICE、3D-FUTURE & 3D-FRONT 等项目,在开源社区中，也获得了广泛开发者的支持和使用。

以下是淘系新兴和经典开源项目介绍，欢迎参与和交流。

项目一：

Rax

所属领域：前端

项目地址：

https://github.com/alibaba/rax

项目介绍：

Rax 是一个可支持同时开发 Web/Weex/小程序多端的框架。使用Rax 可以一次开发，多端运行，解放重复工作，专注产品逻辑，提升开发效率。

项目二：

飞冰（ICE）

所属领域：前端

项目地址：

https://github.com/alibaba/ice

项目介绍：

飞冰（ICE）是一个企业级的中后台全方位解决方案，具体如下：

1. 研发框架 icejs：通过 icejs 研发框架提供完整的标准化的 React 应用开发模式和最佳实践。

2. 研发工具 iceworks：通过 Iceworks 提供了可视化辅助和代码编辑辅助的手段，提高您的开发效率及体验

3. 微前端 icestark：面向大型工作台的微前端解决方案，通过引入框架应用和子应用的概念，框架应用负责系统整体布局以及子应用的注册、加载与渲染，同时在设计原则上约定「子应用尽量保持跟传统单页面应用一样的开发体验」，保证子应用自身可独立运行、存量应用可快速迁移适配、增量应用跟传统方式开发体验一致

4. 工程工具 build-scripts：高度可扩展的工程解决方案，通过微内核 + 插件的设计原则，所有核心能力通过插件承载，支持不同体系的工程构建。

项目三：

MNN

所属领域：人工智能

项目地址：

https://github.com/alibaba/mnn

项目介绍：

轻量级深度学习推理引擎，具有高性能、通用、易用的特点。

项目四：

coobjc

所属领域： iOS

项目地址：

https://github.com/alibaba/coobjc

项目介绍：

Coobjc 为 objective-C 和 Swift 语言提供了协程能力，可以用于异步调用的场景，解决 iOS 中通过 block 调用的各种缺点。支持 await、generator 和 actor model。

项目五：

FlutterBoost

所属领域：客户端

项目地址：

https://github.com/alibaba/flutter_boost

项目介绍：

FlutterBoost 是一个 Flutter 插件，在原生应用中集成 Flutter 的混合解决方案，共享一个 flutter 引擎，提供混合栈管理，将 Flutter 像 Webview 那样来使用。

项目六：

Sandbox

所属领域：Java

项目地址：

https://github.com/alibaba/jvm-sandbox

项目介绍：实现了一种在不重启、不侵入目标 JVM 应用的 AOP 解决方案。

项目七：

JVM-Sandbox-Repeater

所属领域：Java

项目地址：

https://github.com/alibaba/jvm-sandbox-repeater

项目介绍：

基于 JVM-Sandbox无侵入式 AOP 架构，实现了 Java 服务端的无侵入录制回放的基础协议框架。

项目八：

Midway

所属领域：前端

项目地址：

https://github.com/midwayjs/midway

项目介绍：

Midway 即使一个面向未来的全栈 Web 开发框架，也是一个 Serverless 开发框架，在一定条件下，Web 应用和 Serverless 函数的互相迁移，和前端完美的融合开发，发布。

项目九：

Pipcook

所属领域：前端

项目地址：

https://github.com/alibaba/pipcook

项目介绍：提供前端可用的视觉和 NLP 能力。

加入我们 ： lyz122260@alibaba-inc.com。

项目十：

3D-FUTURE & 3D-FRONT

所属领域：3D

项目地址：

https://tianchi.aliyun.com/specials/promotion/alibaba-3d-future

https://tianchi.aliyun.com/specials/promotion/alibaba-3d-scene-dataset

项目介绍：

3D 视觉与几何研究是未来主流探索方向。在这起步阶段，业界研究缺乏相对完整且整理较好的 3D 数据集。阿里巴巴淘系技术部与躺平设计家针对这一现象，与国际知名学者合作开源大型 3D 家居数据集 3DFUTURE 与 3D-FRONT，旨在推动业界 3D 研究的迅速发展。

Transmittable ThreadLocal

项目十一：

TransmittableThreadLocal

所属领域：后端

项目地址：

https://github.com/alibaba/transmittable-thread-local

项目介绍：分布式跟踪系统、日志收集记录系统上下文、Session 级 Cache、应用容器或上层框架跨应用代码给下层 SDK 传递信息。

“拥抱开源、回馈开源、融合开源和回报开源”是阿里的开源历程，而我们的每一项重要开源技术都离不开业内广大开发者的参与和贡献。未来手淘将会和大家共同努力，推动中国开源软件生态系统的发展。

第五部分 2020 淘系 A 类顶会论文收录

- 数据与数据挖掘领域
- 信息检索领域
- 移动计算领域
- 数据库领域
- 机器学习和计算神经科学领域
- 人工智能综合领域
- 交叉/综合/新兴领域

大数据与数据挖掘领域

ICDE 2020

Two-Level Data Compression using Machine Learning in Time Series Database

Xinyang Yu, Yanqing Peng, Feifei Li, Sheng Wang, Xiaowei Shen, Huijun Mai, Yue Xie *Alibaba Group*
{xinyang.yu, y.peng, lifeifei, sh.wang, leyu.sxw, huijun.mhj, jimmy.xy}@alibaba-inc.com

***Abstract*—The explosion of time series advances the development of time series databases. To reduce storage overhead in these systems, data compression is widely adopted. Most existing compression algorithms utilize the overall characteristics of the entire time series to achieve high compression ratio, but ignore local contexts around individual points. In this way, they are effective for certain data patterns, and may suffer inherent pattern changes in real-world time series. It is therefore strongly desired to have a compression method that can always achieve high compression ratio in the existence of pattern diversity.**

In this paper, we propose a two-level compression model that selects a proper compression scheme for each individual point, so that diverse patterns can be captured at a fine granularity. Based on this model, we design and implement AMMMO framework, where a set of control parameters is defined to distill and categorize data patterns. At the top level, we evaluate each sub-sequence to fill in these parameters, generating a set of compression scheme candidates (i.e., *major mode selection*). At the bottom level, we choose the best scheme from these candidates for each data point respectively (i.e., *sub-mode selection*). To effectively handle diverse data patterns, we introduce a reinforcement learning based approach to learn parameter values automatically. Our experimental evaluation shows that our approach improves compression ratio by up to 120% (with an average of 50%), compared to other time-series compression methods.

I. Introduction

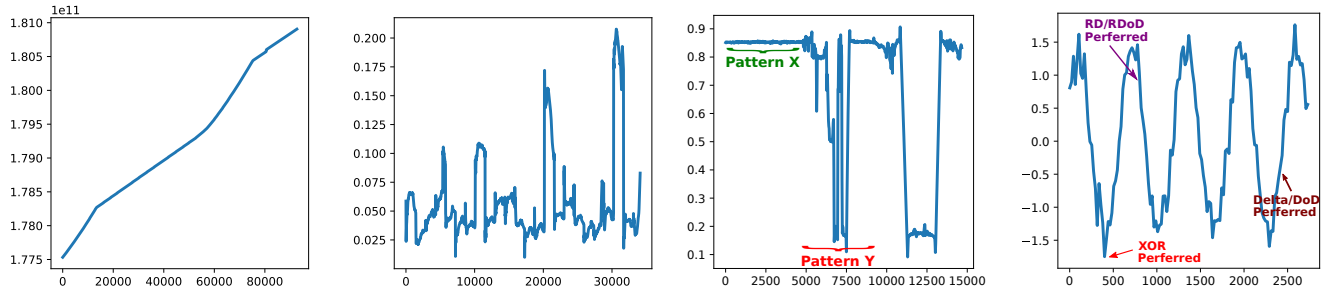
Nowadays, time series data is being continuously generated from a wide range of applications, such as finance [5], Internet services [23] and Internet of things [12]. Such increasing demand for storing and processing time series data facilitates the emergence and development of time series databases (TSDBs), such as OpenTSDB [21] and InfluxDB [9]. One of the key component in these databases is an effective compression scheme that reduces storage footprint, leading to two performance boosts. First, more records can reside in the memory cache for fast access. Second, lower data transfer cost is required between devices (e.g., RAM to disk, and to FPGA or GPU if they are used to speedup processing).

There are plenty of compression schemes to use off the shelf [1], [3], [15], [22]–[25]. However, the effectiveness of a scheme largely depends on its input, where each is only effective on a certain types of inputs (i.e., patterns). Hence in a TSDB, especially a cloud TSDB service that manages data from various sources, it is difficult to rely on a pre-selected compression scheme to sustain high compression ratio for any data. Figure 1 illustrates four examples of real-world time-series data. As shown, these data are distinct from each other, making a single compression scheme difficult to achieve satisfactory compression ratio on all of them. Ideally, we hope that a TSDB could automatically search for the scheme (as well as its parameters) that achieves high compression ratio for a given piece of data. We refer to this problem as *adaptive compression scheme selection*.

This problem gets more challenging in real worlds, since time series data in many applications are awfully complex: even within one time series, the pattern can change over time (Figure 1c); and different parts from a single piece of data may prefer different schemes (Figure 1d). Therefore, in order to achieve good compression ratio, we cannot simply apply one compression scheme for the entire data piece. Instead, we have to examine the data in a fine granularity, and decide the appropriate scheme for sub-pieces of the original data. In the extreme case, we could break down the data into individual points and find the optimal compression scheme for each data point. However, this is almost impractical since both the search space for the optimal scheme and the number of points contained in a TSDB are huge, incurring prohibitive computation cost if we perform the search for each point.

In this paper, we work towards enabling automatic per-point compression scheme selection to achieve satisfactory compression ratio in real-world applications. Note that the per-point approach (needed as illustrated in Figure 1c and Figure 1d) introduces extra space overhead on meta data compared to a global compression scheme. To balance the trade-off between scheme selection efficiency and compression efficiency, we propose a two-level compression model, which is able to adaptively select proper compression schemes for each point. It is built on a concept called *parameterized scheme space* that abstracts the entire search space as a set of scheme templates and control parameters. At the top level, for each *timeline* (a sequence of continuous points), we construct a *scheme space* (a small search space) by filling in values for control parameters. At the bottom level, the most effective scheme from the space is selected for each point.

However, there remain two challenges in the proposed model. First, how to construct a parameterized scheme space that has the potential to always perform well against unpredictable patterns. It is desired that the entire search space is small but covers most cases. Second, how to fill in values of control parameters for a chosen timeline. It is hard to distill hidden patterns from timelines and translate them into parameter values, both effectively and efficiently. For the first challenge, we design and develop AMMMO



(a) Finance (b) Internet monitoring (c) Resource utilization (d) Sensing device

Fig. 1: Four use cases from real scenario; different use cases have different patterns. Fig 1c shows an example that different periods from data could have different patterns; Fig 1d illustrates the preferred compression scheme at different parts of data.

(Adaptive Multi-Model Middle-Out) framework. In particular, it divides the entire search space into four sub-spaces, called *major modes*, designed with insightful preferences (i.e., transform/bitmask/offset-preferred or mixed in Section IV-B). Each major mode further contains four *sub-modes*, which are constructed compression schemes. For the second challenge, we propose both rule-based and learning-based approaches to address it. In particular, for learning-based approach, since labels (i.e., ground truths) for optimal parameter settings are unavailable, we apply reinforcement learning on a neural network structure to learn parameters interactively. When the compression ratio drops due to pattern changes, this network can be easily retrained.

In summary, we make following major contributions:

- We introduce a two-level model to select compression schemes for each individual point, which addresses inherent data diversity in time series. A parameterized scheme space is proposed to facilitate the representation of the entire search space.
- We design a compression framework AMMMO, following the two-level model. This framework encompasses four major modes to categorize typical patterns, and defines sophisticated control parameters to help construct sub-modes, i.e., compression schemes.
- We design a neural network structure with reinforcement learning to tune parameter values automatically. It solves the issue that no labeled training samples available in the context of data compression.
- We conduct extensive experimental evaluations using large-scale real datasets. The results show that our approach improves compression ratio up to 120% (with an average improvement 50%), compared to other time-series compression methods.

The rest of the paper is organized as follows. We first discuss background and motivation in Section II. We then introduce the two-level scheme selection model in Section III, present the AMMMO framework in Section IV, followed by reinforcement-learning-based model selection in Section V. We evaluate our approach in Section VI and conclude in Section VII.

II. Preliminaries

A. Data Compression

The data compression can be regarded as a process to transform a byte sequence in some representation (e.g., floating numbers for metric value) into a new byte sequence that contains the same information but with less bytes. As surveyed in [16], there are many universal compression techniques, such as (static or adaptive) Huffman and arithmetic coding, which are ubiquitous in real-world applications. Besides, there are also compression algorithms tailored for certain data types. For example, HEVC [30] and H264 [10] are designed for videos, while gzip, 7zip and snappy are designed for texts.

Based on different byte alignment strategies for compressed bit streams, compression algorithms can be categorized into *byte-level* and *bit-level* [29]. In general, the bit-level compression achieves higher compression ratio, while the byte-level compression avoids costly bit shift operations, and is more suitable for partition and parallel processing. In this work, we will focus on byte-level compression since it has higher throughput and is more hardware-friendly.

B. Time Series Data

As defined in [21], [24], a time series consists of many data points, each of which is uniquely identified with a timestamp, a metric name, a measured value, and a set of tags. A simple example is shown in Table I. The combination of tags allows users to record different data properties and issue queries intuitively. Among all fields, timestamps and measured values dominate the storage consumption. Therefore, major task is to compression values in these two fields effectively.

Although general-purposed compression algorithms can be directly applied to time series data, these data have their unique characteristics (detailed in Section III-A) that could potentially help us to tailor more effective compression scheme for them.

C. Reinforcement Learning

Reinforcement learning (RL) is a type of machine learning concerned with how to take appropriate actions to maximize reward in a particular situation. It could be useful for training without ground truth [32].

Reinforcement learning has been widely applied in many interactive scenarios, such as game playing [7], [28] and

TABLE I: An example of raw time-series data on CPU load requests.

timestamp	metric name	value	tagA	tagB
1386806400	cpu.load	0.07	node=alpha	type=system
1386806402	cpu.load	0.13	node=alpha	type=user
1386806403	cpu.load	0.05	node=beta	type=user
1386806405	cpu.load	0.48	node=beta	type=user

resource arrangement [18], [19]. In this work, we explore machine learning based compression methods for TSDBs. Since it is hard to obtain ground truth (i.e. optimal compression scheme and configuration), reinforcement learning becomes a promising solution to facilitate our learning process.

D. Related Work

There are several compression algorithms designed for time-series data in the literature. [3], [15] present lossy compression methods running on sensors over a time series stream. Gorilla [23] adopts a lossless algorithm that first conducts delta-of-delta operations on timestamps and xor operations on measured metric values in server side. However, its bit-level compression model is not friendly to heterogeneous computing, such as GPU and FPGA, which is prevalent for computation acceleration. In the parallel middle out algorithm [25], a byte-level compression algorithm is proposed achieving higher throughput. It adopts only one mode for contiguous 8 data points hence achieves lower compression ratio. Similar, [26] develops a SIMD based algorithm for integer data, achieves higher performance while its compression ratio improvement is minor. Sprintz [1] presents a time series compression for the Internet of Things. Its focus is on 8/16 bit integers and is not optimized for floats. PBE [22] utilizes dynamic programming to achieve optimal compression scheme for monotonically increasing time series data, but it cannot handle general time series data with ups and downs. Compression planner [24] presents an interesting idea that constructs and selects compression plan dynamically on GPU from various compression tools like Scale, Delta, Dictionary, Huffman, Run Length and Patched Constant. However, since it tries to compress all the time-series data several times with various tools and then pick the best one, this is not suitable for high-throughput scenarios. ModelarDB [11] develops a model-based algorithm which segments data into 2 major static models, Gorilla and constant function model which performs well in regular time series and lossy mode. In data mining area, lossy compression for time-series data is widely adopted [13], [17]. However, these methods are unacceptable for a commercial-level TSDB production system.

To support commercial systems for various applications, one of our objectives is to design a lossless byte-level compression framework for time series data that can achieve both high compression ratio and high throughput.

III. Two-level compression scheme selection

A. Time-Series Data Characteristics

Based on our observations from production environments, we summarize several inherent characteristics in time-series data, which play a crucial role in guiding the design of an effective and practical data compression algorithm:

- **Time Correlation**. A time series contains data points continuously collected over a time period. These points are usually sampled with a pre-defined interval (e.g. per second). This leads to two consequences: consecutive timestamp values advance at a relatively fixed rate; and consecutive metric values are always close to each other. These observations should be utilized by compression schemes for good compression ratio.
- **Pattern Diversity**. Time series data can be generated by diverse applications and domains, e.g., finance, Internet, and IoT, from which data patterns may vary dramatically. Furthermore, patterns of a time series may vary over time due to circumstance changes. A compression strategy designed for certain patterns often performs poorly on others, which must be carefully handled in a practical compression solution.
- **Data Massiveness**. As data points are often generated at high rate in time series applications, the underlying storage engine requires both high write throughput (e.g., tens of millions operations per second) and fast processing capacity (e.g., tens of petabytes per day). It is therefore essential to have a compression component that not only helps reduce data volume, but also completes (de-)compression tasks fast.

B. Model Formalization

According to data compression fundamentals [16], a data compression process consists of two stages: a *transform* stage that transform data from one space to another which is more regular; and a *differential coding* that use various coding methods to represent the differential value after transform. There are many primitives available in both stages, e.g., Delta, Scaling, Prediction, Dictionary for transform (IDCT, intra-prediction etc in H264 [10]), and Huffman coding, Arithmetic coding, Run-length differential coding (e.g. VLC, CABAC in H264). By selecting different primitives in each stage, along with corresponding parameters, we can derive a large number of compression schemes (Section IV-A) for time-series data.

Due to the pattern diversity issue discussed above, a single manually-selected compression scheme cannot sustain satisfactory compression ratio for all scenarios. At the other extreme, it is also prohibitive to enumerate all possible schemes for each individual point to obtain the local-best compression ratio. Hence, to ensure that pattern diversity can be gracefully handled with acceptable overheads, we propose two-level time-series compression scheme selection model. At the top level, shared contextual patterns within a consecutive point sequence (i.e., timeline) are detected in order to limit the search space of potentially effective compression scheme candidates

Algorithm 1: Basic Two-Level Compression

```
   Input: a time series ts, scheme spaces sps
1  while ts.empty() == false do
       /* read next timeline from ts                    */
2      timeline = ts.readNext();
       /* select scheme space for timeline              */
3      space = sps.select(timeline);
4      for each point p in timeline do
           /* select scheme for the point               */
5          s = space.select(p);
6          compress p using scheme s;
```

(i.e., space selection). Here, we say points within a sequence share a 'contextual pattern' if they potentially have good compression ratio under the same set of compression scheme candidates, e.g., the majority of delta-of-delta difference can be represented by 2-byte-value. At the bottom level, by searching over these scheme candidates, we are able to find an effective scheme for each individual point (i.e., scheme selection). Since the top-level phase is conducted once for a timeline, a heavy but effective algorithm is affordable. However, for the bottom-level phase, the efficiency and efficacy are both critical.

In the context of this paper, a *timestamp* is represented as a 64-bit integer, and a *measured metric value* is a 64-bit floating number. In our data compression framework, all data points in a time series with the same metric and tags are modeled as a sequence of ⟨timestamp, value⟩ pairs. We denote the i-th point in the sequence as $\langle t_i, v_i \rangle$. Note that a time series is compressed separately as two sequences $T = \{t_1, t_2, ..., t_n\}$ and $V = \{v_1, v_2, ..., v_n\}$.

1) Scheme Selection Modeling: In this section, we explain how the scheme selection problem is modeled as a two-level procedure. Suppose that we have a collection of transform primitives P_{trans} (e.g., xor of two consecutive values $\{v_i, v_{i-1}\}$, delta-of-delta of $\{v_i, v_{i-1}, v_{i-2}\}$) and a collection of differential coding primitives P_{code} (e.g., 6-bit bitmask coding, 1-byte offset coding). We define several important notions as follows:

Definition 3.1 (Compression Scheme): A compression scheme is the basic unit to compress a single value. It is represented as a tuple $s = \langle a, b, \lambda_a, \lambda_b \rangle$, where $a \in P_{trans}$, $b \in P_{code}$ and λ_a, λ_b are parameters associated to a, b respectively[1]. The scheme contains all information required to compress a value.

Definition 3.2 (Scheme Space): A scheme space S contains a collection of compression schemes $S = \{s_1, s_2, ..., s_n\}$. Each data point belongs to a single scheme space S_i, and is compressed by a specific scheme $s_j \in S_i$.

Given a time series and a set of scheme spaces, we propose the *basic* scheme selection model for each point, as shown in Algorithm 1. For each *timeline* (a consecutive point sequence), a scheme space is selected. Then it is easy to find the most effective scheme for each point, by simply compressing the point using different schemes and keeping the best one.

[1]We use one parameter per scheme for simplicity. Each scheme can have multiple parameters instead.

However, from this simplified model, some critical challenges are not addressed yet:

- How to have a concise scheme space representation, instead of explicit member list? A concise representation facilitates the automatic space selection without human interaction. (Section III-B2)
- How to construct scheme spaces that has the potential to perform well against diverse patterns? It is desired that the entire search space is small in size but covers most frequent patterns. (Section IV)
- How to choose the most suitable scheme space among all candidates? It is challenging to distill hidden patterns from time series and associate them to scheme spaces effectively and efficiently. (Section V)

2) Parameterized Scheme Space: In order to support fast space selection, the number of space candidates must be kept small, which inherently declines the capability of handling different patterns. Instead of having pre-defined static spaces, it is therefore demanded that these candidates can keep adapting themselves to pattern changes, e.g., by leveraging machine learning techniques. However, the member list representation used in Definition 3.2 treats each compression scheme as an independent element, and hence is difficult to be learned by machine learning tasks. Alternatively, we propose a concise representation called *parameterized scheme space*, which divides a scheme space into two parts: a static *scheme template* set and a configurable *variable* set.

Definition 3.3 (Scheme Template and Variables): A scheme template is an extension to a specific compression scheme. It replaces one or more elements in $\langle a, b, \lambda_a, \lambda_b \rangle$ with variables, which are determined at runtime. A variable can represent either a scheme $(i.e., a, b)$ or the parameter associated to a scheme $(i.e., \lambda_a, \lambda_b)$

For example, $\langle X, \texttt{offset}, \lambda_X, \lambda_{\texttt{offset}} \rangle$ with variable X means that the coding scheme is offset coding and the transform scheme is determined by the value of X. Similarly, $\langle \texttt{xor}, \texttt{bitmask}, \lambda_{\texttt{xor}}, Y \rangle$ with variable Y means that xor transform and bitmask coding are used, and the parameter (e.g., byte shift size) of bitmask coding is determined by the value of Y.

Definition 3.4 (Parameterized Scheme Space): A parameterized scheme space contains a collection of scheme templates and a collection of associated variables. A variable can present in multiple scheme templates, which introduces correlations between different schemes within the space.

With the concept of parameterized scheme space, we can transform the task of finding scheme spaces into the task of determining variable values. These values can be adjusted dynamically to catch up pattern changes.

IV. AMMMO Compression Framework

The two-level scheme selection model provides a general-purpose abstraction for data compression. Based on this model, we design AMMMO (Adaptive Multi-Model Middle-Out) compression framework, which is a specific implementation tailored for time-series data. In this section, we discuss key design choices made in AMMMO, including scheme primitives,

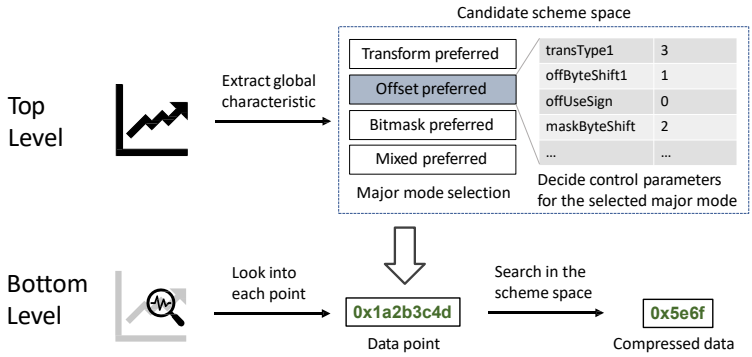


Fig. 2: The two-level compression framework AMMMO for compressing time-series data.

TABLE II: List of differential coding primitives. 0x and 0b prefixes stand for hexadecimal and binary formats.

Primitive Name	Parameter	Format	Example		
			Input	Param	Output
`Offset`	`offByteShift`	1 byte: 1-bit control bits || 7-bit value 2 bytes: 2-bit control bits || 14-bit value 3 bytes: 3-bit control bits || 21-bit value	0x 00 17 00 00 00	3	control bits || 0b10111
`Bitmask`	`maskByteShift`	1 value: (up to 6-bit) bitmask || value 2 values: 6-bit bitmask || value1 || value2	0x 10 00 26 84 00	1	0b1011 || 0x 10 26 84
`Trailing-zero`	`N/A`	2-bit (or 3-bit) trailing-zero control bits || 3-bit non-zero control bits || value	0x 00 14 04 09 00	`N/A`	1 (0b01) || 3 (0b010) || 0x 14 04 09

TABLE III: List of transform primitives.

Id	Primitive Name	Operation Description
0	`delta`	$v_i - v_{i-1}$
1	`reversed delta (RD)`	$v_{i-1} - v_i$
2	`xor`	v_i xor v_{i-1}
3	`delta-of-delta (DoD)`	$(v_i - v_{i-1}) - (v_{i-1} - v_{i-2})$
4	`reversed delta-of-delta (RDoD)`	$(v_{i-1} - v_{i-2}) - (v_i - v_{i-1})$
5	`delta xor (DX)`	$(v_i - v_{i-1})$ xor $(v_{i-1} - v_{i-2})$

scheme spaces (i.e., control parameters, major modes and sub-modes), and an intuitive rule-based method for scheme space selection. These choices of primitives and parameter settings are empirical and have been proven to be effective in practice. However, they are orthogonal to our model and can be replaced or extended arbitrarily. An overview architecture of AMMMO framework is shown in Figure 2.

A. Compression Scheme Primitives

Recall that a compression scheme consists of a transform stage and a differential coding stage (or called encoding stage). In AMMMO, we use a set of primitives to build these stages.

Transform primitives. At the transform stage, for the i-th data point, we apply transform operations to its value v_i, which additionally takes previous two points v_{i-1}, v_{i-2} as input. Table III lists six transform primitives used in our framework, which are proven effective from our empirical evaluations in production environment. Note that those transform primitives suffering low throughput (e.g., dictionary) are excluded from our implementation.

Encoding primitives. The transform stage converts each 8-byte raw value into a 8-byte differential value. At the encoding stage, this value is to be encoded using less bytes. Table II lists three encoding primitives designed in our framework, and we detail them as follows:

- *Offset coding* uses a global parameter `offByteShift` to indicate from which byte the non-zero value starts. In our implementation, three coding lengths (1/2/3 bytes) are supported, each with own `offByteShift` value.
- *Bitmask coding* has a bitmask (with up to 6 bits) to indicate whether a byte has non-zero values. To handle the case that only high-order bytes have non-zero value, a global parameter `maskByteShift` is defined to remove low-order bytes from the mask. To further reduce the number of control bits, two consecutive values can share one bitmask, halving the amortized bits per value.
- *Trailing-zero coding* separates the 8-byte value into three parts: trailing-zero bytes, non-zero bytes and leading-zero bytes. In this format, 2 or 3 bits (trailing-zero control bits) are able to indicate up to 3 or 7 bytes of trailing zeros; and 3 bits (non-zero control bits) representing range $[1, 8]$ instead of $[0, 7]$, indicate the number of non-zero bytes (up to 8) to be coded after trailing zeros are removed.

B. Compression Scheme Space Definition

AMMMO applies different scheme selection strategies to metric values and timestamp values. For metric values, the two-level selection model is used, which determines a major mode (i.e., scheme space) followed by a sub-mode (i.e., scheme), as shown in Figure 2. Timestamp values only use bottom level (sub-mode) selection model as their patterns are more regularized.

Based on compression scheme Definition 3.1 and primitives design IV-A, AMMMO constructs nine (first four specify a and b, and last five specify λ_a and λ_b) control parameters as listed in Table IV (defining possible compression scheme spaces), which major mode (each with four sub-modes) is further defined in Table V.

TABLE IV: The control parameters in AMMMO which defines a compression scheme space in top level.

Parameter	Bits	Range	Description
majorMode	2	[0, 3]	Indicate the selected major mode
transType1	3	[0, 5]	Refer to a transform in Table III
transType2	3	[0, 5]	Similar to transType1
transType3	3	[0, 5]	Similar to transType1
offByteShift1	3	[0, 7]	Byte offset for 1-byte offset coding: offByteShift = offByteShift1
offByteShift2	1	[0, 1]	Byte offset for 2-byte offset coding: offByteShift = offByteShift1 - offByteShift2
offByteShift3	1	[0, 1]	Byte offset for 3-byte offset coding: offByteShift = offByteShift1 - offByteShift2 - offByteShift3
offUseSign	1	[0, 1]	Indicate if offset coding use sign
maskByteShift	3	[0, 5]	Byte offset for bitmask coding

TABLE V: The major modes and sub-modes in AMMMO with control parameters.

Id	Major Mode	Control Bits	Sub-Mode Compression Scheme
0	Transform Preferred	0b00	transType1 with 6-bit bitmask
		0b01	transType2 with 6-bit bitmask
		0b10	transType3 with 6-bit bitmask
		0b11	transType1 with 6-bit trailing-zero
1	Bitmask Preferred	0b00	transType1 with 6-bit bitmask
		0b01	transType2 with 2-value bitmask
		0b10	transType3 with 6-bit bitmask
		0b11	transType1 with 6-bit trailing-zero
2	Offset Preferred	0b1	transType2 with 1-byte offset
		0b01	transType2 with 2-byte offset
		0b000	transType2 with 3-byte offset
		0b001	transType1 with 5-bit trailing-zero
3	Mixed	0b1	transType1 with 1-byte offset
		0b01	transType2 with 2-value bitmask
		0b000	transType3 with 5-bit bitmask
		0b001	transType1 with 5-bit trailing-zero

A space (i.e., major mode) contains 4 sub-modes to adaptively compress each point at bottom level. This choice is a balance between the coverage of search spaces and the overhead of meta data and computation. Note that the definitions of the major modes and sub-modes are based on experimental results and are further extended or replaced when dealing with different workloads. In our current design, a timeline requires 20 bits (Table IV) of meta data for top-level mode selection, and an average of 2 bits (Table V) for bottom-level mode selection.

C. Compression Procedure

In AMMMO, 32 data points are processed as a block with header and data segment, and timestamps and metrics value are compressed separately:

Timestamp compression. Timestamps in a timeline usually have fixed intervals, such as 1 second. This pattern can be easily captured by the delta-of-delta transform operation. However, it is common that some points violate this pattern, e.g., due to sampling jitter or network delay. Inspired by Gorilla [23], we take advantage of the delta-of-delta operation, and try to align the irregular data in bytes to speed up. To simplify transform process, we consider the byte representation of each point as a 64-bit integer. Detailed compression procedure is shown in Algorithm 2. We maintain two data segments, i.e., *header* for delta-of-delta outputs and *data* for irregular data

Algorithm 2: Compression for Timestamp Values

```
   Input: timestamp sequence ts
   Output: compressed bitStream bs
1  bitStream header, data;
   /* write first two points (8 bytes each)                    */
2  data.append(ts[0], ts[1]);
3  for each point ts[i] where i ∈ [2, ts.length) do
4      d = (ts[i] − ts[i-1]) − (ts[i-1] − ts[i-2]);
5      if d == 0 then header.appendBit(0) ;
6      else
7          header.appendBit(1);
8          if d > 0 then d = d − 1;
9          if d ∈ [−4, 4) then
10             data.append(1b'1 —— 3b'd); // 4 bits
11         else if d ∈ [−32, 32) then
12             data.append(2b'01 —— 6b'd); // 8 bits
13         else
14             if bitmask coding is preferred then
15                 data.append(3b'001 —— 5b'bitmask);
16                 data.append(bitmask-encoded d);
17             else
18                 data.append(3b'000 —— 5b'control-bit);
19                 data.append(zero-trailing-encoded d);
   /* final output concatenates header and data                */
20 return bs.write(header, data)
```

Algorithm 3: Compression for Metric Values

```
   Input: metric sequence ms
   Output: compressed bitStream bs
1  bitStream header, data;
2  data.append(ms[0]);
3  for each point ms[i] where i ∈ [1, ms.length) do
4      d = ms[i] − ms[i-1];
5      if d == 0 then header.appendBit(0) ;
6      else
7          header.appendBit(1);
8          for each mode m from sub-modes do
9              calculate compression ratio r for d;
10             if r ¿ best_ratio then
11                 best_ratio = r;
12                 best_mode = m;
13         data.append(best_mode.compress(d));
14 return bs.write(header, data)
```

points. The first two points of a timeline are stored in *data* in their raw formats (line 2). For subsequent points, we calculate their delta-of-delta values and indicate in *header* (line 3-7). When irregular value occurs (i.e., delta-of-delta is not zero), they are encoded in different lengths (e.g., 4, 8, 16 bits) and appended to *data* segment (line 8-19) [2].

Metric value compression. The compression for metric values is similar to timestamps, as shown in Algorithm 3. Here, deltas of consecutive metric values are used as indicators (line 3-7), instead of delta-of-deltas. For each point different from its predecessor (i.e., $d \neq 0$), we try all 4 sub-modes to find the best compression scheme for d. For each sub-mode, we construct the corresponding scheme to encode d and calculate compression ratio (line 8-12). We then choose the sub-mode with the best ratio and append compressed d into the $data$ segment (line 13).

D. Rule-based Scheme Space Selection Algorithm

The remaining challenge is to select proper compression scheme space, i.e., fill in values for all parameters in Table IV. A rule-based algorithm (Algorithm 4) invokes metric *benefit*

[2]Since $d = 0$ is unoccupied, we utilize it to cover a larger range (line 8).

Algorithm 4: Rule-based Scheme Space Selection

```
Input: metric value sequence ms
Output: control parameter values params in Table IV
/* PART I: calculate the benefit_score                    */
/* benefit_score: the total number of bytes can be
   saved against the worst case 9-byte original
   representation per point (i.e. 1 byte of control
   bits and 8 bytes of differential value) for
   different compress schemes in a timeline               */
1  a 6 × 6 array with zero initializations: benefit_score;
2  for each point ms[i] where i ∈ [1, ms.length) do
3      for each transform mode tm[j] in Table III do
4          for each coding format cf[k] in Table II do
5              benefit = calculateBenefit(ms[i], tm[j], cf[k]);
6              benefit_score[j][k] += benefit;
7
   /* PART II: calculate the params based on
      benefit_score                                       */
   /* best_majorMode_score represents the best score
      among 4 major modes                                 */
8  best_majorMode_score = 0;
9  for each majorMode mm[i] where i ∈ [0, mm.length) do
10     majorMode_score = 0;
       /* best_subMode_score represents the best score
          among 4 sub modes of the majorMode mm[i]        */
11     best_subMode_score = 0;
12     for each subMode sm[j] where j ∈ [0, sm.length) do
           /* find the array indexes in benefit_score that
              match mm[i] and sm[j], say s and t          */
13         s, t = findIndex(mm[i], sm[j]);
14         if benefit_score[s][t] > best_subMode_score then
15             best_subMode_score = benefit_score[s][t];
16     majorMode_score += best_submode_score;
17     if majorMode_score > best_majorMode_score then
18         params = findParams(mm[i]);
19
20 return params
```

score to represent a compression scheme's efficiency, and the whole process is separated into two parts generally. Firstly, *benefit score* for all the possible submodes are constructed (line 2-6). Secondly, the best major mode is selected based on the benefit scores, and then fill control parameter values accordingly (line 8-18).

However, there are issues that limit its efficiency and are difficult to resolve, for example: whether *benefit score* is a good enough metric? Moreover, when the definitions of Table II ˜ V need to be adjusted (e.g., to tune compression performance), such an analysis process has to be manually re-designed and implemented, which is complex and time-consuming. Therefore, it is appealing to have an efficient, automatic and adaptive method to tune these parameters, which motivates us to adopt machine learning approaches.

V. Machine Learning for Mode Selection

From Table IV, we have $2^3 \cdot 4 \cdot 6^4 \cdot 8 = 331,776$ possible combinations for control parameters in total. It is prohibitive to enumerate all cases to find the best *control setting*. Alternatively, getting the appropriate control setting can be formulated as a classification problem, which supervised machine learning approaches are good at solving. However, it is infeasible in our context to generate ground truths for training since:

- Suppose a training sample has 32 points (256B), there will be 256^{256} possible cases in theory. And for each sample, we need random 331,776 times to find the best compression ratio and take it as label. The sample organization and computation consumption to create dataset is prohibitive.
- Supervised learning requires the label to be only one or a few, while given a sample, it's likely that multiple control settings are able to achieve the same compression ratio, which leads to multiple labels associated with a sample.
- Once the primitive and parameter definition is refined, the original training set becomes useless.

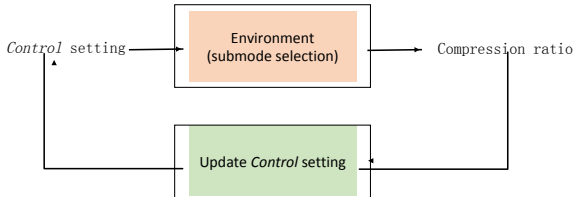


Fig. 3: An overview of policy gradient RL based control setting generation model.

Hence, we propose a framework that adopts reinforcement learning to tune control parameters automatically in two steps:

1) *Learn control settings from blocks.* Each timeline is further divided into fixed-size blocks (e.g., 32 points per block in our implementation). A neural network (Section V-B) is trained via reinforcement learning (Section V-A) to interactively learn proper settings of control parameters for each block .
2) *Determine the control setting of a timeline.* The trained network is used to generate control settings for all blocks. After that, the best setting for the entire timeline is determined following a statistical strategy (Section V-C).

A. Training with Reinforcement Learning

Figure 3 illustrates the overview of our learning flow. In particular, a neural network takes a block as a state, generates a control setting as the action. Under this setting, the submode selection module computes the compression ratio as the feedback to adjust the neural network interactively.

For a given block, its control setting (or called control option) is defined as *cop* = $\{ op_0, op_1, ..., op_8\}$, where op_i belongs to the value range of i-th control parameter listed in Table IV. For example, op_0 is `majorMode` and its value belongs to range [0, 3]. Let $r_i(cop)$ represent the compression ratio of the i-th block with control setting *cop*. The objective of the control setting generation network is to find the *cop* that maximizes the compression ratio:

$$J(\theta) = E_{x \sim p(x|\theta)}[f(x)] \tag{1}$$

where: x stands for a control setting; θ is the policy defined by the network (Section V-B) that generates the *cop*; $p(x/\theta)$ is the probability to generate control setting x; and $f(x)$ represents the reward to the RL network, i.e., compression ratio $r_i(x)$. The network aims to maximize the expected reward of control setting decision x under probability distribution $p(x/\theta)$.

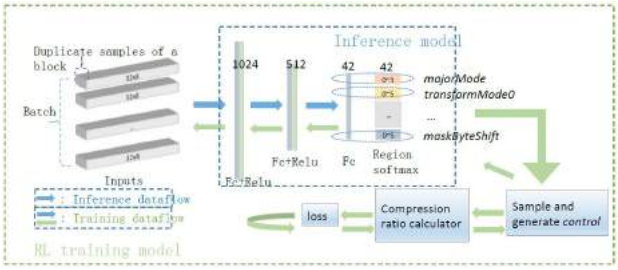


Fig. 4: Neural network architecture.

Decisions are the control option cop_i drawn from the softmax distribution in θ.

The weights in the network are learned using Adam [14] optimizer based on the policy gradients computed via the REINFORCE equation [32]:

$$\begin{aligned}\nabla_\theta J(\theta) &= \nabla_\theta E_x[f(x)] \\ &= E_x[f(x)\nabla_\theta logp(x)] \\ &\approx \frac{1}{N}\sum_{i=1}^{N} f(cop_i)\nabla_\theta logp(cop_i)\end{aligned} \tag{2}$$

We sample N control settings to estimate ∇_θ as specified in Formula 2. It implies that the network is updated towards the direction of better compression reward $f(cop_i)$.

For N different sampled settings of a block, we should activate the network to generate higher possibility for those *cop*s with high rewards. To achieve this goal, block-level normalized compression reward $f_n(x)$ and summarized cross entropy $Hcs(x)$ are applied:

$$f_n(x) = (f(x) - mean(f(x)))/std(f(x)) \tag{3}$$

$$Hc(x) = -(p(x)logp(x) + (1-p(x))log(1-p(x))) \tag{4}$$

$$Hcs(x) = \sum_{i=x:op0}^{x:op8} Hc(i) \tag{5}$$

Consider that the SGD scheme uses M blocks as a batch, each with N samples, the final loss function to be minimized is:

$$\frac{1}{M*N}\sum_{i=1}^{M*N}(f_n(cop_i) * Hcs(cop_i)) - \lambda * H(cop) \tag{6}$$

where $H(cop)$ is the average entropy value of all *cop* used as entropy regularization that avoids network converge to local optimum [20]. We use regularization parameter $\lambda = 0.01$ to leverage it.

B. Neural Network Architecture

Figure 4 shows the neural network structure and overall data flow of training and inference. In the RL training stage (module), a batch of total M blocks, each of which has N duplicate samples, are fed into the network. First, the blocks pass through the network and generate $M \cdot N$ control setting candidates. Second, for each field in a control setting, we sample with associated probabilities to determine its value. Third, compression ratios under each control setting are calcuated and then are used to derive the loss according to Formula 6. Last, backward propagation is applied and weights are updated accordingly. In the inference stage (module), either a single block, a batch, or all blocks of a timeline can be fed into the network. We pass through the network and generate control setting for each block. Then for each field in a control setting, we choose the highest probability value among all the blocks as the final setting.

TABLE VI: Neural network layers.

Layer	Type	Size	Activation
1	Fully-connected	1024	ReLU
2	Fully-connected	512	ReLU
3	Fully-connected	42	None
4	Region Softmax	42	None

Table VI lists layer configurations used in the network. The basic input unit is a block, which has the size $32 \cdot 8$ bytes. The fully connection is applied to layer 1, 2 and 3, in which the first two layers have the ReLU activation function. The size of the hidden node in layer 3 is 42, which is the number of all available values in nine control parameters. Each parameter field could make its own decision independent from others. Hence, rather than using the global softmax as in image classification tasks, we introduce a *region softmax* that conducts softmax on each field, instead of using all 42 logits.

C. Determine Timeline Control Setting

A timeline has only one global control setting that takes effect on all blocks. We use statistic strategy to assemble blocks of control setting to one. The procedure is: consider all these parameters as independent from each other, select the most frequent values for each field in control parameters. Note, some other strategies (i.e., select the most frequent combination of control settings, or choose the most frequent value for `majorMode` first, and then select most frequent values for other fields from those blocks belong to this mode) have slight difference in the final result view.

VI. Experiments

To evaluate the proposed AMMMO framework, we test it on datasets collected from our business operations[3], ranging from IoT applications to performance monitoring of large clusters. In addition, a public timeseries repository UCR [4] is also used. We compare our method against other state-of-the-art compression methods in terms of both the compression ratio and efficiency. Also, we evaluate AMMMO framework using various mode selection algorithms, showing the advantages of machine learning based methods.

A. Experiment Setup

Datasets. We collect a representative set of time series datasets that exhibit different patterns, characteristics and sizes. In particular, 28 datasets are selected, where 8 of them are from IoT applications and 20 of them are from

[3]https://github.com/JonyYu/ATimeSeriesDataset.

cluster monitoring. They are of different sizes, from 10k to 500k points[4]. For the purpose of applying machine learning methods, these datasets are further divided into a test set A (for training) and a test set B (for validation), as listed in Table VII[5].

TABLE VII: Datasets with 28 selected time series.

Test Set A					
Name	Points	Name	Points	Name	Points
IoT0	430,737	IoT6	430,413	Server35	147,395
IoT1	429,745	IoT7	313,539	Server41	136,594
IoT2	428,390	Server30	158,188	Server43	29,233
IoT3	344,581	Server31	147,385	Server46	154,585
IoT4	306,736	Server32	165,395	Server47	140,199
IoT5	372,868	Server34	140,194	Server48	157,051

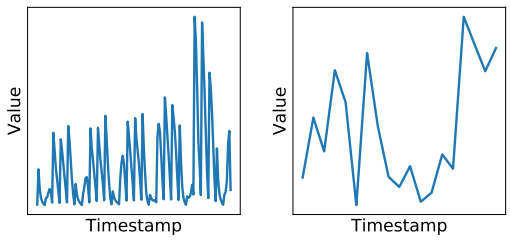
Test Set B			
Name	Points	Name	Points
Server57	26,779	Server94	140,198
Server62	32,569	Server97	158,194
Server66	135,409	Server106	136,478
Server77	136,598	Server109	153,438
Server82	143,798	Server115	165,384

Also, there is a well-known time series datasets named UCR [4], which is a repository of 85 univariate time series datasets from various domains, commonly used for benchmarking time series algorithms. Since most of them are small, and each has several different patterns (pattern id 0, 1, 2, ...). We choose the longest 8 datasets, and concatenate all the pattern 0 to form a single longer time series. We also separate this dataset into test sets A and B in Table VIII.

Figure 5 illustrates two of the datasets. As shown in figure, the data from different datasets exhibit different characteristics. Even in one dataset, the internal data patterns are complex.

TABLE VIII: 8 longest time series datasets in UCR

Test Set A		Test Set B	
Name	Points	Name	Points
HandOutlines	641,796	CinC_ECG_torso	8,190
Haptics	19,638	InlineSkate	16,929
StarLightCurves	155,496	MALLAT	6,138
UWaveGestureLibraryAll	115,168	Phoneme	4,092



(a) Server48 (b) Haptics

Fig. 5: Illustration of two of the datasets.

Methods and Variants. Three open-sourced baseline compression methods and six AMMMO variants are used to evaluate the performance of AMMMO framework and machine learning mode selection.

- **Gorilla** [2]: a state-of-the-art commercial bit-level compression algorithm applied in server side. It applies delta-of-delta on timestamp values followed by a variable length coding, while applies xor on metric values followed by Huffman coding.
- **MO (Middle-Out)** [25]: a byte-level compression algorithm good for parallel processing, which takes 8 points as a process block. It applies xor on timestamp values followed by byte-aligned differential coding, while applies xor on metric values followed by trailing zero coding on the differences.
- **Snappy** [6]: a general-purpose compression algorithm developed by Google. It is byte-level and is used by InfluxDB [9], KairosDB [8], OpenTSDB [21], RocksDB [31], the Hadoop Distributed File System [27] and numerous other projects.
- **AMMMO_Lazy**: AMMMO framework with a fixed control setting, i.e., set `majorMode` to 0, `transTypes` to 2/5/0, and all other parameters to 0.
- **AMMMO_Rnd5000Best**: Control setting is selected from random generations. Generate 5000 times and select the one with the best compression ratio. Although we do not cover all possible settings, it still gives a good approximation for the optimal ratio that AMMMO could achieve. Note that this method is unaffordable in production deployment, and we only use it as a reference point.
- **AMMMO_Rnd1000Best**: Generate random control settings 1000 times, report the best compression ratio.
- **AMMMO_Rnd1000Avg**: Generate random control settings 1000 times, report the average compression ratio.
- **AMMMO_Analyze**: Scheme space selection is done by rule-based analysis algorithm (Section IV-D).
- **AMMMO_ML**: Scheme space selection is done by machine learning (Section V).

B. Compression Ratio

In this section, we evaluate the compression ratio achieved by AMMMO framework on both timestamp and metric values. We also demonstrate the advantages of using machine learning methods.

1) Timestamp Compression: For timestamp values, we use one-level mode selection in AMMMO, and hence all variants have the same behavior. Therefore, we omit their internal comparisons and only provide results for Gorilla, MO and AMMMO.

Figure 6 shows the compression ratios for all 18 time series in test set A. There is no result for Gorilla on server monitoring applications, as it cannot handle the case that the difference between two consecutive timestamps is larger than 2 hours (which exists in all these time series). As can be seen, Gorilla and AMMMO perform much better than MO. It is because that delta-of-delta operations are more suitable for timestamps compared to xor operations. Moreover, AMMMO achieves the best results in all cases, since it uses 3 bits to handle

[4]Each dataset is treated as a single timeline for simplicity. However, in real world, a long time series (with diverse patterns) is always split into multiple timelines to mitigate the pattern diversity within a timeline.

[5]In test set A, 8 out of 18 datasets contain constant metric values over long periods (e.g., with most values being zero), which can be easily compressed. Hence, we only include them in timestamp evaluation.

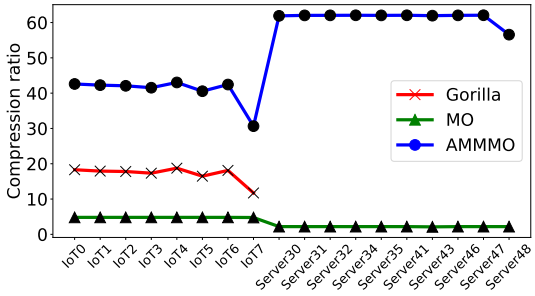
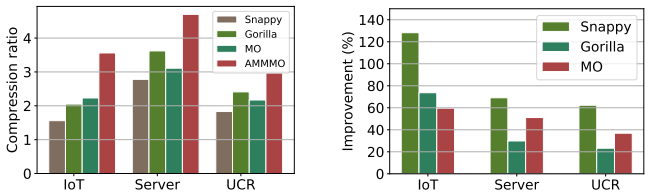


Fig. 6: Compression ratios on timestamp values.



(a) Compression ratio

(b) Improvement of AMMMO over other schemes

Fig. 7: Comparison of compression ratios.

sample jitter pattern and uses adaptive multiple modes (e.g., offset, bitmask, trailing-zero) to handle other patterns. We can observe that the overall compression ratio improvement in AMMMO are significant: an 8-byte raw value only costs 1 to 2 bits in average after compression. Note that Gorilla also requires only 4 bits in average, indicating that timestamp compression is quite effective in most cases.

2) Metric Value Compression with AMMMO: Recall that AMMMO framework enables variants with different mode selection strategies as listed in Section VI-A. To clearly show its overall capacity, we define AMMMO as the best compression ratio achieved by all variants.

We define X *improved compression ratio over* Y as $\frac{(X-Y)}{Y}$. Figure 7a shows the average compression ratio on all tests in IoT/Server/UCR datasets. From the result, Snappy achieves worst performance since it does not consider characteristic of timeseries much; Gorilla is slightly better than MO in general (MO is better in IoT cases since IoT data is usually simpler), because it uses bit level packing operation; AMMMO is significantly better in all these cases. Figure 7b shows the improved compression ratio comparison. The average improved ratio of AMMMO over Snappy, Gorilla, and MO are around 87%, 42% and 49%.

Figure 8 shows the detailed compression ratios on each time series in test set A. For better readability, the x-axis is sorted by Gorilla's compression ratio. As can be seen, Snappy is worst, Gorilla and MO are close, and AMMMO variants significantly outperform others in most cases; an exception is `UWaveGesture`, where snappy achieves very good compression ratio and AMMMO_Rnd5000Best achieves slightly better result. Note that AMMMO_Rnd5000Best is better than AMMMO_Rnd1000Best in most cases except for `IoT5` and `Server43` (due to randomness). This indicates that AMMMO_Rnd5000Best is a proper approximation of AMMMO's

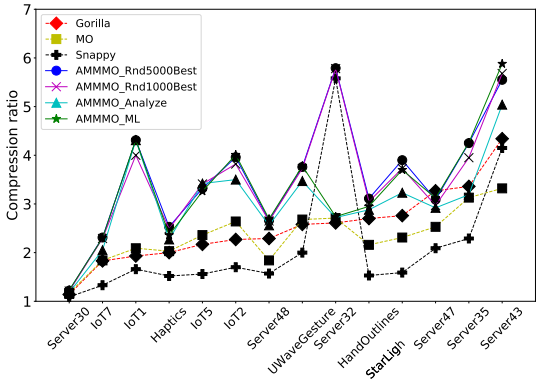


Fig. 8: Comparison of metric value compression ratios.

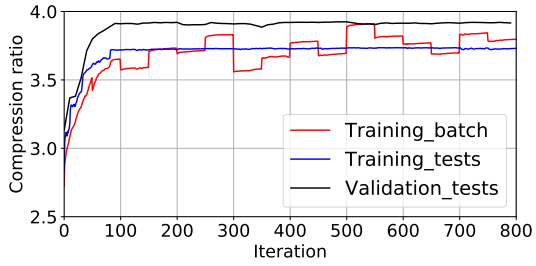


Fig. 9: Training curve of compression ratio on training batch, training tests, and validation tests.

best performance. For practical variants AMMMO_Analyze and AMMMO_ML, machine learning approaches outperform manually-determined rules in most cases. And AMMMO_ML is very close to or even better (`Server43`) than AMMMO_Rnd5000Best, except `UWaveGesture`.

3) AMMMO with Machine Learning: In this part, we focus on the evaluation of whether AMMMO_ML is able to learn proper parameter values from training data, i.e., test set A. Therefore, we compare it with other variants on validation test set B. In the training stage, we set batch size M to 896, block duplicate sample N to 64, and start learning ratio to 0.0005. Figure 9 illustrates the average compression ratio achieved in training batch, training tests and validation tests. Thanks to the effective loss function design, the RL scheme learns how to compress quickly. These ratios increase rapidly in first 100 iterations, and after that the curve becomes stable. Suppose we have a time series with 1024 blocks. According to our findings, we only need to train 100 iterations to obtain acceptable compression ratio, taking around 100 seconds.

AMMMO has a large number of control setting options, and it is critical to have a mode selection strategy that can find a proper one. Figure 10 shows the compression ratios achieved by different mode selection strategies. As can be seen, AMMMO_Rnd1000Avg is the worst, and even AMMMO_Lazy outperforms it. Both AMMMO_ML and AMMMO_Analyze sustain good performance in most cases. Moreover, AMMMO_ML achieves astonishing compression ratio on `Sever115` and `Sever97`. We observe that, AMMMO_ML performs worse than AMMMO_Rnd1000Best on `Sever109`. With a closer look, we observe that block-level

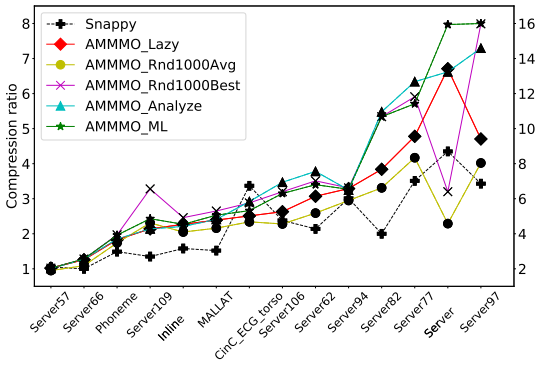
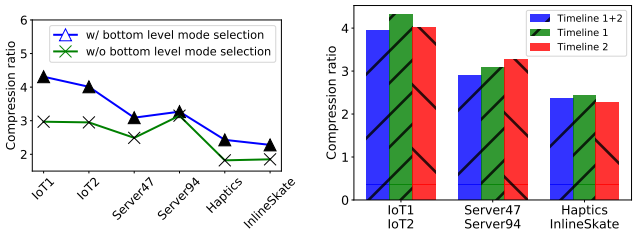


Fig. 10: Metric value compression ratios from different methods. Right vertical axis is for `Server97` and left axis for other datasets.



(a) Bottom level mode selection (b) Compression on split timelines

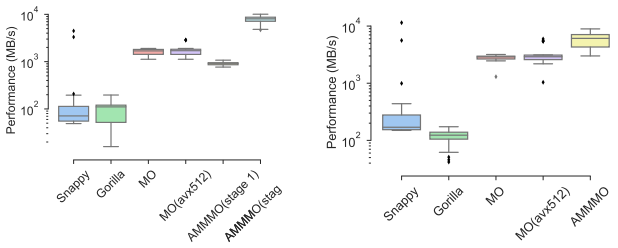
Fig. 11: Comparison of different mode selections.

compression ratios are still good in AMMMO_ML. However, these parameter values are quite diverse, which makes the statistic polices hard to derive a proper global control setting.

In summary, AMMMO_ML is able to learn patterns from training set A and achieve excellent compression ratio on validation set B. This confirms that machine-learning based mode selection is effective, and can help to automate the parameter tuning process. In fact, our initial AMMMO only contains six control parameters and two major modes. This automatic mode selection facilitates the discovery of nine effective parameters and four representative modes as the backbone of our AMMMO framework.

4) Parameter Verification: It is interesting to check whether AMMMO_ML figures out meaningful control setting. Table IX lists the timeline control setting decisions made by different AMMMO variants, on `IoT1`, `IoT2`, `Server35` and `Server48` in set A. The results show that AMMMO_ML is able to find almost the same setting for `maskByteShift` and `offsetByteShift`, which indicates that it understands the effect of data resolution after transform. It also performs well on `majorMode` selection and transform types selection.

Figure 10 shows the importance of top-level mode selection, which indicates the advantages of AMMMO_ML against other variants that do not have mode selection. It is also interesting to look at bottom-level mode selection effect. Figure 11a shows the result when only submode 4 (i.e., similar mode as Gorilla in Table V) is used, the performance is significantly worse as can be expected. For the timeline length, if a timeline contains different patterns in each sub-parts, splitting it into



(a) Compression (b) Decompression

Fig. 12: Compression and decompression performance comparison among different schemes.

shorter timelines is able to take advantage of different scheme spaces. Figure 11b shows that compression on split timelines (i.e., shorter and with less diverse patterns) achieves better compression ratio.

C. Compression Efficiency

Due to the massiveness of time series, the compression and decompression efficiency also plays a crucial role in compression algorithms. Here we provide throughput for both operations achieved by AMMMO, Gorilla, MO and Snappy.

On a CPU+GPU platform[6], the compression and decompression performance of Gorilla, MO and Snappy are illustrated in Figure 12. From the result, we can see that Snappy's performance varies a lot. In most cases, the performance is around 60MB/s (150MB/s). In some rare cases, it can reach up to 4.5GB/s (11.5GB/s), because the compression ratio is less than 1.1 and hence there is no or few compression. Gorilla and MO are much stable, and the average performances are around 100MB/s (100MB/s) and 1.3GB/s (2.9GB/s). In several cases, Gorilla's decompression performances is worse than compression performances. It is because that Gorilla is a bit-level coding, where the whole process has to be serialized and involves frequent bit operations and condition checks during decompression. With the same CPU resource, MO's performance is much better than Gorilla, which is attributed to the simpler algorithm and byte-level coding. AMMMO is also byte-level, and we evaluate its performance on GPU. Both its compression and decompression performances are stable. Compression procedure in AMMMO involves 2 stages[7] , mode selection stage and compression stage (Section IV-C). The performance is around 900MB/s and 7.5GB/s[8] respectively. For decompression, the performance is around 5.6GB/s, which is slightly worse compared to compression performance. It is because compression procedure can leverage higher parallelism during sub-mode selection for many points.

[6]CPU: Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz; GPU: Nvidia gp100 with 16 GB video memory; Memory: 512 GB.

[7]The mode selection performance is for learning-based inference. In real deployments, not all timelines need mode selection, since we usually can reuse the last mode for subsequent timelines (due to time correlation). If the inference performs badly, we can retrain the network, which costs about 100s.

[8]This performance includes data copy through PCIe. The compression kernel itself achieves around 30GB/s.

TABLE IX: Control settings selected by different AMMMO variants.

	Algorithm	major Mode	trans Type1	trans Type2	trans Type3	offByte Shift1	offByte Shift2	offByte Shift3	offUse Sign	maskByte Shift
IoT1	Analyze	3	0	5	0	3	0	0	0	1
	RandomBest	3	0	5	3	3	1	1	0	1
	ML	3	0	5	4	3	1	0	0	1
IoT2	Analyze	2	0	2	0	3	1	0	0	0
	RandomBest	2	5	0	3	3	1	0	0	0
	ML	2	0	0	5	3	1	0	0	0
Server35	Analyze	2	0	5	0	5	0	0	0	0
	RandomBest	3	4	2	4	5	0	0	1	5
	ML	3	3	2	5	5	0	0	0	5
Server48	Analyze	2	5	5	0	4	0	0	0	0
	RandomBest	3	0	5	4	6	1	1	0	4
	ML	3	4	5	0	4	0	0	0	4

VII. Conclusion and Future Work

In order to improve compression ratio for time-series data, we propose a two-level compression scheme selection model to fully utilize diverse data characteristics. We design and implement AMMMO framework, a byte-level parallel-oriented compression method outperforming state-of-the-art approaches in terms of compression ratio and efficiency. It achieves 50% better compression ratio compared with Gorilla and MO. In this framework, we introduce a machine learning based method to learn diverse data patterns, and it helps perform mode selection and tune parameters automatically. We have used this automatic method to discover nine control parameters and four major modes, which in turn advises the design of AMMMO internal structures.

As future work, we will explore how to improve AMMMO by adjusting major mode definition, working with major modes, integrating with other compression tools and building blocks (for example support scaling). Another valuable direction is to design another neural network to replace statistic polices in generating timeline control settings. Lastly, it is an interesting open problem to extend the proposed framework to compress other data types.

References

[1] D. W. Blalock, S. Madden, and J. V. Guttag. Sprintz: Time series compression for the internet of things. IMWUT, 2(3):93:1–93:23, 2018.

[2] M. Burman. Time series compression library, based on the facebook's gorilla paper, 2016. https://github.com/burmanm/gorilla-tsc.

[3] H. Chen, J. Li, and P. Mohapatra. Race: time series compression with rate adaptivity and error bound for sensor networks. In 2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems, pages 124–133. IEEE, 2004.

[4] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. The ucr time series classification archive, 2015.

[5] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In Proceedings of the 1994 ACM SIGMOD international conference on Management of data, SIGMOD '94, pages 419–429. ACM, 1994.

[6] S. Gunderson. Snappy: A fast compressor/decompressor, 2015.

[7] X. Guo, S. P. Singh, H. Lee, R. L. Lewis, and X. Wang. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In NIPS, pages 3338–3346, 2014.

[8] B. Hawkins. Kairosdb: Fast time series database on cassandra.

[9] InuxDB. Open source time series, metrics, and analytics database, 2015.

[10] ITU-T and I. J. 1. Advanced video coding for generic audiovisual services, 2012. https://www.itu.int/rec/T-REC-H.264.

[11] S. K. Jensen, T. B. Pedersen, and C. Thomsen. Modelardb: Modular modelbased time series management with spark and cassandra. In Proceedings of the VLDB Endowment, volume 11. VLDB, 2018.

[12] S. D. T. Kelly, N. K. Suryadevara, and S. C. Mukhopadhyay. Towards the implementation of iot for environmental condition monitoring in homes. IEEE sensors journal, 13(10):3846–3853, 2013.

[13] E. J. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In KDD, pages 206–215. ACM, 2004.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In ICLR, 2015.

[15] I. Lazaridis and S. Mehrotra. Capturing sensor-generated time series with quality guarantees. In Proceedings of the 19th International Conference on Data Engineering, pages 429–440. IEEE, 2003.

[16] D. A. LELEWER and D. S. HIRSCHBERG. Data compression. In ACM Computing Surveys 19, volume 1 of ACM 1987, 1987.

[17] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, pages 2–11. ACM, 2003.

[18] A. Mirhoseini, A. Goldie, H. Pham, B. Steiner, Q. V. Le, and J. Dean. A hierarchical model for device placement. In Proceedings of the International Conference on Learning Representations, ICLR '18, 2018.

[19] A. Mirhoseini, H. Pham, Q. Le, M. Norouzi, S. Bengio, B. Steiner, Y. Zhou, N. Kumar, R. Larsen, and J. Dean. Device placement optimization with reinforcement learning. In International Conference on Machine Learning, ICML '17, 2017.

[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Proceedings of the 33rd International Conference on International Conference on Machine Learning, volume 48 of ICML '16, pages 1928–1937. ACM, 2016.

[21] OpenTSDB. A distributed, scalable monitoring system, 2013.

[22] D. Paul, Y. Peng, and F. Li. Bursty event detection throughout histories. In ICDE, pages 1370–1381. IEEE, 2019.

[23] T. Pelkonen, S. F. J. Teller, P. Cavallaro, Q. Huang, J. Meza, and K. Veeraraghavan. Gorilla: A fast, scalable, in-memory time series database. In Proceedings of the VLDB Endowment, volume 8 of VLDB '15, pages 1816–1827, Dec. 2015.

[24] P. Przymus and K. Kaczmarski. Dynamic compression strategy for time series database using gpu. In New Trends in Databases and Information Systems, pages 235–244. Springer, 2013.

[25] Schizofreny. Middle-out compression for time-series data, 2018.

[26] B. Schlegel, R. Gemulla, and W. Lehner. Fast integer compression using simd instructions. In Proceedings of the Sixth International Workshop on Data Management on New Hardware, volume 48 of DaMoN'10, pages 34–40. ACM, 2010.

[27] J. Shieh and E. Keogh. isax: disk-aware mining and indexing of massive time series datasets. Data Mining and Knowledge Discovery, 2009.

[28] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. G. Aja Huang, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. Nature, 550:354–359, Oct. 2017.

[29] E. Sitaridi, R. Mueller, T. Kaldewey, G. Lohman, and K. A. Ross. Massively-parallel lossless data decompression. In 2016 45th International Conference on Parallel Processing, ICPP '16, pages 242–247. IEEE, 2016.

[30] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand. Overview of the high efficiency video coding (hevc) standard. IEEE Trans. Circuits Syst. Video Technol., 22(12):1648–1667, Dec. 2012.

[31] F. D. E. Team. Rocksdb: A persistent key-value store for fast storage environments.

[32] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8(3):229–256, May 1992.

2020 IEEE 36th International Conference on Data Engineering (ICDE)

Cross-Graph Convolution Learning for Large-Scale Text-Picture Shopping Guide in E-Commerce Search

Tong Zhang[1], Baoliang Cui[2], Zhen Cui[1], Haikuan Huang[2], Jian Yang[1], Hongbo Deng[2], Bo Zheng[2]
1 School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
2 Alibaba Group, Hangzhou, China
{tong.zhang,zhen.cui,csjyang}@njust.edu.cn; moqing.cbl@taobao.com; {haikuan.hhk,dhb167148,bozheng}@alibaba-inc.com

***Abstract*—In this work, a new e-commerce search service named text-picture shopping guide (TPSG) is investigated and deployed to one of the most popular shopping platforms called Taobao. Different from traditional services that only contain text options, the TPSG provides pairs of text terms and user-friendly pictures for shopping guide, named text-picture options (TPOs). Instead of manually labeling pictures, we aim to automatically recommend personalized pictures in TPOs. To this end, we build a large-scale graph model on a great amount of data about users, pictures, and terms. Accordingly, a cross-graph convolution learning (CGCL) method is proposed to facilitate the accurate and efficient inference on the constructed graph. To separate the cue of personalized preferences of users to commodities, we factorize the entire mixture-relation graph involving attributes/relations of users and commodities into the user graph, the commodity graph, and the cross user-commodity graph which just characterizes the preferences. Further, we introduce powerful graph convolution to learn more effective representation of these graphs. To reduce the computation burden, specifically, we generalize graph convolution and propose a tensor graph convolution method to learn representation on cross graphs. We conduct extensive offline and online experiments on the large-scale datasets. The results show that the proposed CGCL is very effective and the TPOs recommendation method outperforms manual/advanced selection methods.**

***Index Terms*—large-scale online e-commerce, text-picture shopping guide, text-picture option, cross-graph convolution learning**

I. Introduction

Nowadays, online shopping has become one of the most popular consumption fashions, where customers can search for their desired commodities by entering several keywords (also called queries) on e-commerce search engines. In many cases, users' queries could be very broad and ambiguous, and thus have numerous matching candidates. For example, the query terms of "dress" and "women's shoes" are too broad to find those proper commodities for users. To offer users a better search experience, many platforms provide some assistant text options related to the original queries, as shown in the left part of Fig. 1. After users click one of the options, the search engine will do search again by appending the term of clicked option to the original query. This will help users to narrow their intention and find proper commodities.

Corresponding Authors: Baoliang Cui and Zhen Cui

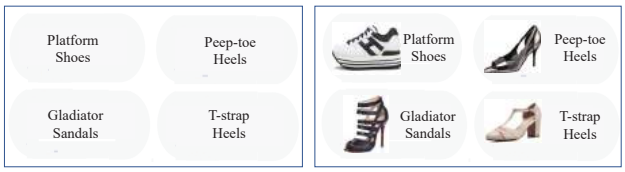


Fig. 1: The examples of previous text option (left) and our designed TPOs (right) based on the query of "women's shoes".

However, it's hard for users to understand the exact meaning of such terms occurred in options, especially if users are buying such commodities for the first time. For example, most users have no sense about the unfamiliar term "gladiator sandals" if only given the text. Thus the configuration of text options will heavily degrade users' search experiences. To address this problem, we investigate a new service, called text-picture shopping guide (TPSG), to help users better understand the exact meaning of terms. An option of TPSG, named text-picture option (TPO), is defined by a term together with a related picture, as shown in the right part of Fig. 1. In contrast to the traditional text options, TPSG helps users better understand the meaning of such terms by providing visual pictures. The statistic experiment on our online e-commerce marketplace also concludes that the click through rate (CTR) of TPOs is about 30% higher than only text options, which shows the superiority of TPOs.

The gain of TPOs motivates us to study how to automatically recommend the proper TPOs. Initially, on the Taobao platform, pictures in TPOs are manually annotated and pre-fixed for those terms, one of which is only related to one constant picture. However, there are several disadvantages. First, this picture is not personalized because different users are with different preferences. Second, outer conditions may influence the selection, such as seasonality, fashion and so on. Third, manual annotation costs a large number of labor resources. Therefore, designing an effective and automatic recommendation system of TPOs is crucial to cater to different users and improve the quality of search service. However, it is challenging to do this. The reasons are three folds:

(1) The factors affecting users' preferences are complicated. Firstly, users' characteristics such as "gender", "age" and

2375-026X/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDE48307.2020.00147

IEEE computer society

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 21,2020 at 02:41:21 UTC from IEEE Xplore. Restrictions apply.

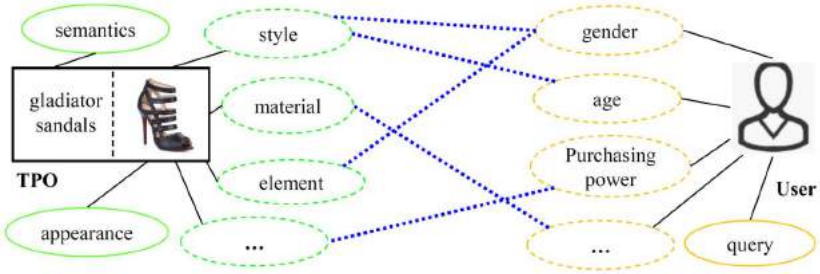


Fig. 2: An example illustration of factor analysis in the TPSG task. Factors of users are colored in orange while commodities are colored in green. Ellipses with solid lines represent explicit information and ellipses with dotted lines represent underlying attributes. Blue dotted lines denote interactions across users and commodities.

"purchasing power" have an important effect on their preferences. Secondly, the semantics of terms and visual appearances of pictures might affect users' preferences. Finally, users might care about the underlying attributes of pictures such as "style" and "element", as shown in Fig. 2.

(2) The cold start problem. There are many new commodities together with many new users every day, which makes the online TPSG task nontrivial. Currently, the existing popular algorithms [1]–[5], e.g. collaborative filtering methods based on matrix completion [6] and deepwalk [3], [4], cannot work well in this case because of the severe cold start problem.

(3) The high computational complexity. A large number of text terms and pictures on Taobao marketplace cause explosive combinations of TPOs.

Considering these issues aforementioned, we need to address two crucial problems:

(i) How to model this intricate system, especially preference relations of customers to commodities? This contains the selection/purification of attributes and the definition of connection relations within/across users and commodities.

(ii) How to infer personalized TPOs from a great number of candidates for the online TPSG task? This recommendation process should be accurate with tolerable running time on the online platform.

In order to achieve more systematic and comprehensive modeling, three aspects of information are considered and jointly modeled, including the text of queries and terms, the appearance of pictures, and the attributes of users and commodities. To represent the intricate shopping system, we abstract and define those useful attributes as well as their connection relationships, and then integrate all of them into a large-scale graph model. To further mine the cue of users' preferences, we factorize the graph into three types of subgraphs: the user graph, the commodity graph, and the cross user-commodity graph which describes the preferences of different users to commodities. To learn effective representation on the irregular graphs, we introduce the powerful graph convolution, and extend it to cross-graph convolution learning (CGCL) for the cross-graph inference case. Specifically, we propose the tensor graph convolution to efficiently learn node representation by diffusing context information across different graphs. The tensor graph convolution consists of two main stages, the parameterized cross-graph construction and the factorized graph filtering. After extracting high-level features from users and TPOs, our CGCL predicts preference scores of users to TPOs. Finally, the TPOs are ranked according to the preference scores, and the top-ranked TPOs are recommended to users.

We first test our CGCL on the offline data collected from Taobao platform, and compare it with several baseline methods. Next we deploy our CGCL to the online shopping guide system of the Taobao platform, and conduct the online A/B tests. Both offline and online tests demonstrate the effectiveness of our proposed CGCL on the TPSG task.

Our contributions are summarized as follows:

(i) To the best of our knowledge, this is the first research literature to investigate the TPSG task, and to deploy it to a large-scale online e-commerce platform.

(ii) We propose a cross-graph convolution learning method to model the intricate large-scale online shopping system, and further develop the tensor graph convolution to infer personalized TPOs from a great amount of data.

(iii) The offline and online tests on a large e-commerce platform validate the effectiveness of our proposed method.

II. Related work

To the best of our knowledge, there are few public research literatures to investigate the TPSG problem on online e-commerce platforms. In general, two aspects of research are mostly related to our work: graph convolution from the view of technical line and personalized recommendation from the view of application line. Below we briefly overview them.

Graph convolution: In recent years, graph convolutional neural networks (GCNNs) are given more attention in the field of artificial intelligence [7]–[10]. As the generalization of conventional convolutional neural networks (CNN), GCNNs are adapted to irregular graph data by combining CNN together with the spectral graph theory, and show the superiority in dealing with irregular data. The basic idea of GCNNs originated from the work in [7], where spectral graph theory was studied

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 21,2020 at 02:41:21 UTC from IEEE Xplore. Restrictions apply.

for convolutional filtering on graphs. Afterward, a fast localized graph convolution was proposed [10] by approximating global filtering with parameterized polynomials, which results into more robust and high efficient local filtering. In the parallel period, many variants on the spectral convolutions were proposed, such as graph convolutional networks (GCNs) [9] and Cayleynets [11], etc. Currently, GCNN begins to flourish in the fields of data mining and pattern recognition [12], [13], and has been applied to many tasks such as action recognition [14], [15], user recommendation [6] and text classification [10].

Personalized recommendation: Numerous researches have been done on personalized recommendation in recent years. They can be roughly divided into three categories, collaborative filtering [1]–[4], [6], [16]–[22], content-based filtering [23]–[28] and hybrid recommendation methods [29]–[31]. Collaborative filtering offers new recommendations by finding similar rating patterns based on collected ratings of items by users, while content filtering makes use of similarities between items and users to recommend new items. For collaborative filtering, some previous methods mathematically modeled the recommendation task as matrix completion problems, including the classical matrix completion [32] and geometric matrix completion [6], [33]. Vassilis et al. [33] aimed to obtain a low-rank solution on matrix completion that structured by the proximities of rows and columns encoded by graphs. The low-rank solution was also constrained to be smooth on these graphs. Moreover, based on the graph theory, the graph auto-encoder framework [34] and recurrent multi-graph neural networks [6] were proposed by considering the user rating prediction as matrix completion tasks and employing graph/multi-graph convolution to learn the user-item interactions. For content filtering methods, DNNs are often employed to encode information of customers and commodities for computing their similar scores. For instance, Schrauwen et al. [25] employed DNNs for the task of music recommendation, and Oh et al. [26] constructed DNNs for recommending news to readers by analyzing users' preferences. Covington et al. [35] learned the embedding features of the user's watch history, and recommended personalized Youtube videos according to the computed similarities between the user and videos. Moreover, to take advantage of both collaborative filtering and content-based filtering, hybrid recommendation methods were proposed to exploit the information of both user interaction and semantics of user/commodity content.

For the online large-scale TPSG problem on shopping platforms such as Taobao, the collaborative filtering and hybrid recommendation based methods are hardly deployed due to high degree dynamics caused by a huge number of newly introduced commodities and registered users every day. Therefore, we follow the content-based framework to solve the TPSG problem. However, different from previous works, we approach the TPSG task from a new perspective by modeling the preferences of users. In view of the key influence of the pictures in TPOs to users' preference, we propose a cross graph convolution learning method by specifically deriving a tensor graph convolution operation to model interaction between underlying attributes of customers and commodities.

III. Preliminary of Graph convolution

Different from image/video data with regular lattices, irregular graph data is more flexible in topology structures and the standard convolution cannot favor this case well. Recently, graph convolution methods [9], [10] are proposed to extract more discriminative features from graphs. Below we briefly overview the classic graph convolution [7], [36].

We denote a graph with a triple tuple $\mathcal{G} = (\mathcal{V}, \mathbf{A}, \mathbf{S})$, where $\mathcal{V}$ is a set of n nodes, $\mathbf{A} \in \mathbb{R}^{n\times n}$ is an adjacent matrix between nodes, $\mathbf{S} \in \mathbb{R}^{n\times d}$ is the signal (also named feature) matrix of the graph. For a user graph in the TPSG task, for example, we take user attributes (e.g., age, gender and purchasing power) as nodes, and the relationships between nodes define the adjacent matrix $\mathbf{A}$. Each element A_{ij} describes the connection strength between the i-th node and the j-th node (e.g., the correlation score between "gender" and "purchasing power"). Each node is associated with a d-dimension feature vector as its description. For example, the "age" node is transformed into one hot vector, the "material" node in a commodity graph is projected into a real vector through word embedding.

The degree of a node is the number of edges incident to this node. Accordingly, the degree matrix $\mathbf{D}$ is defined as a diagonal matrix, i.e., $\mathbf{D} = \mathrm{diag}([d_1, d_2, \cdots, d_n])$, where $d_i = \sum_j A_{ij}$. The normalized Laplacian matrix is $\mathbf{L} = \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{I}_n$ is an $n \times n$ identify matrix. Since the Laplacian matrix $\mathbf{L}$ is positive semidefinite, it has a set of nonnegative real eigenvalues $\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$, and the corresponding eigenvectors $\{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n\}$ are the complete orthonormal basis, known as Fourier modes. The Laplacian matrix can be diagonalized by the Fourier basis $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n] \in \mathbb{R}^{n\times n}$. It can be written as $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^\intercal$, where $\Lambda = \mathrm{diag}([\lambda_1, \lambda_2, \cdots, \lambda_n])$. Then the graph Fourier transform [36] on a graph signal $\mathbf{s} \in \mathbb{R}^{n*}$ is defined as $\widehat{\mathbf{s}} = \mathbf{U}^\intercal\mathbf{s} \in \mathbb{R}^n$, and the inverse Fourier transform is $\mathbf{s} = \mathbf{U}\widehat{\mathbf{s}}$.

Suppose the convolution operator on a graph is the filter f, Fourier transform converts the filter f as $\hat{f}(\Lambda)$, where $\hat{f}(\Lambda)$ is the spectral operator. Equivalently, we derive it in the space domain as

$$\mathbf{s}' = \mathbf{U}\hat{f}(\Lambda)\widehat{\mathbf{s}} = \mathbf{U}\,\mathrm{diag}([\hat{f}(\lambda_1), \cdots, \hat{f}(\lambda_n)])\mathbf{U}^\intercal\mathbf{s}. \quad (1)$$

If $\hat{f}$ takes the polynomial approximation, the spectral filtering may be parameterized as

$$\hat{f}_\theta(\Lambda) = \mathrm{diag}([\hat{f}_\theta(\lambda_1), \hat{f}_\theta(\lambda_2), \cdots, \hat{f}_\theta(\lambda_n)]), \quad (2)$$

where $\hat{f}_\theta(\lambda_i)$ is a polynomial function of λ_i with the parameter θ. In general, we may take the plain polynomial

$$\hat{f}_\theta(\lambda_i) \approx \sum_{k=0}^{K-1} \theta_k \lambda_i^k, \quad (3)$$

*To simplify the derivation, we consider the case of a signal (i.e., one dimension feature) for each node. $\mathbf{s} \in \mathbb{R}^n$ can be viewed as one column of $\mathbf{S}$. The conclusion can be easily extended to the case of multiple signals.

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 21,2020 at 02:41:21 UTC from IEEE Xplore. Restrictions apply.

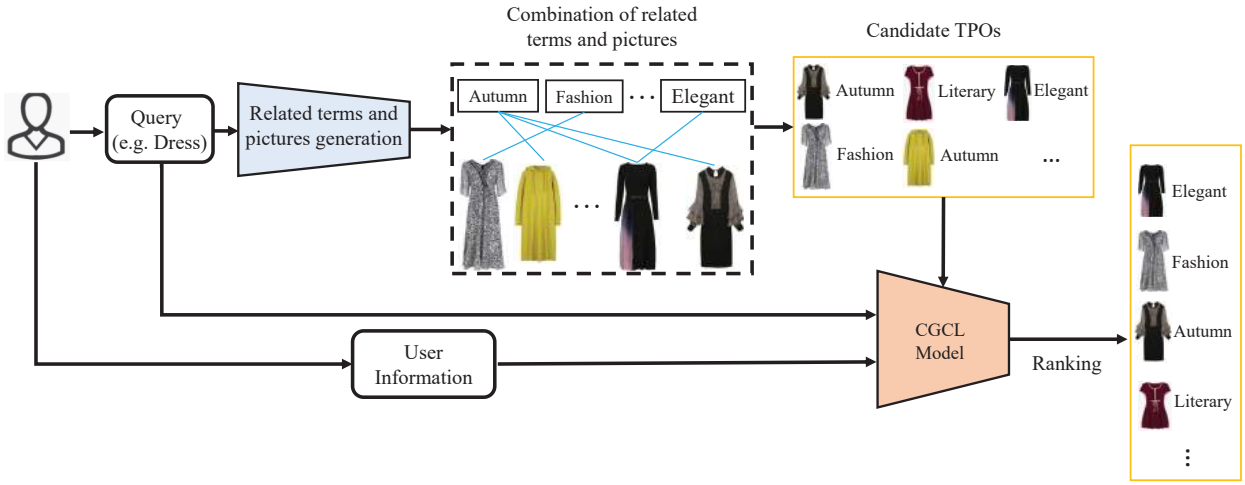


Fig. 3: The flow chart of TPSG system.

where θ_k is the coefficient (i.e., filter parameter) of the k-order polynomial term. By substituting Eqn. (3) into Eqn. (1), after a series of matrix calculation, we can derive graph filtering as

$$s' = \sum_{k=0} \theta_k \mathbf{L}^k \mathbf{s}. \tag{4}$$

Furthermore, according to the literature [37], $\mathbf{L}$ can be replaced by $\mathbf{A}$ as they have the same Fourier basis. Then the spectral graph filtering can also be rewritten as

$$\mathbf{s}' = \sum_{k=0}^{K-1} \theta_k \mathbf{A}^k \mathbf{s}. \tag{5}$$

Such a polynomial approximation avoids the use of singular value decomposition, which usually has high computational complexity with $O(n^3)$. The convolution process is the weighting aggregation on those k-hop neighbors. The larger k is, the more distant the aggregated nodes are. For example, when $k = 1$ in Eqn. (5), i.e., the polynomial term $\theta_1 \mathbf{A}\mathbf{s}$, the signals of the adjacent nodes are summarized with the weights of edges, and then transformed into the other space through the filtering parameter θ_1. In fact, the matrix $\mathbf{A}^k$ computes the k-step reachable possibilities between nodes, if we normalize $\mathbf{A}$ or $\mathbf{A}^k$ into possibility values.

IV. TPSG System Overview

We show the whole procedure of TPSG system in Fig. 3. Given a query from a user, e.g., "dress", the search engine first generates lots of relative terms. For each term, a number of corresponding pictures are attached. The combination of these initial terms and attached pictures will produce a great number of candidate TPOs. Then, our designed framework is to recommend the personalized TPOs from these candidate terms and related pictures. It should be noticed that one picture/term may correspond to multiple terms/pictures.

To build this system, we first find factors related to the TPSG task, e.g., attributes and relationships as shown in Fig. 2. Generally, there are some explicit factors such as user queries, commodity pictures, textual terms, etc. The underlying factors include attributes of users such as "gender", "age" and "purchasing power", and attributes of commodities such as "style", "element" and "material". Besides, the interaction relationships between user and commodity attributes are also important factors, and are often different for different users. For example, older users may pay more attention to "material" while younger users may care more about appearance.

We integrate all these factors including attributes and relationships into a graph framework, and then infer personalized TPOs on graphs. To recommend proper TPOs for different users, we propose a cross-graph convolution learning method (in Section V) to predict the scores between users and TPOs, and then take those TPOs ranked with the highest scores as the recommendation. In order to train the model, offline dataset is collected based on the log of Taobao marketplace (please refer to the details in Section VI-A). At the online testing stage, TPOs are ranked and selected according to the prediction scores of our CGCL model.

V. Cross-Graph Convolution Learning

The entire network architecture of our proposed CGCL is shown in Fig. 4. Considering all factors within/across users and commodities, we may construct a single huge graph to describe all attributes and all potential connections, which will increase the computation burden on the inference of TPOs. To efficiently mine the cues of users' preferences, we construct two separate graphs by virtue of local connectivity, the user graph and the commodity graph. In each graph, attributes are treated as nodes and the relations between attributes are viewed as edges. In the recommendation system, one user graph may link to several commodity graphs when user likes several commodities, and conversely one popular commodity may also be liked by several users. To model complicated connections between users and commodities, we build a cross

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 21,2020 at 02:41:21 UTC from IEEE Xplore. Restrictions apply.

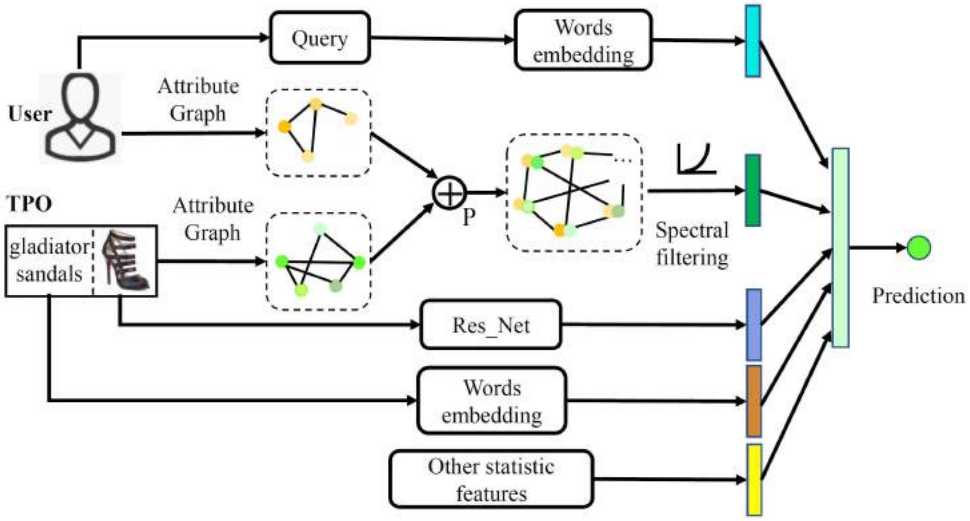


Fig. 4: The proposed cross-graph convolution learning architecture.

user-commodity graph to describe the interactions between user and commodity graphs. Thus the recommendation of TPOs may be understood as the selection/prediction of those interesting or important interactions. To make more accurate prediction, we introduce the powerful graph convolution to learn more discriminative representation. Specifically, we extend it to the cross-graph case and propose the cross-graph convolution learning. To reduce computation cost, we further derive a tensor filtering calculation on cross-graph.

Parameterized cross-graph construction. Let $\mathcal{G}_u = (\mathcal{V}_u, \mathbf{A}_u, \mathbf{S}_u)$ and $\mathcal{G}_c = (\mathcal{V}_c, \mathbf{A}_c, \mathbf{S}_c)$ denote the user and commodity graphs respectively, in which $\mathcal{V}_u, \mathcal{V}_c$ are the node sets with n_u, n_c nodes, $\mathbf{A}_u \in \mathbb{R}^{n_u \times n_u}$ and $\mathbf{A}_c \in \mathbb{R}^{n_c \times n_c}$ are the adjacent matrices, $\mathbf{S}_u \in \mathbb{R}^{n_u \times d_u}$ and $\mathbf{S}_c \in \mathbb{R}^{n_c \times d_c}$ are the signal/feature matrices of user and commodity graphs respectively. The features of nodes are determined by the attributes therein, which are related to those defined factors. For those explicit factors such as the query words and terms, we employ their embeddings as the features. For the pictures, we employ the ResNet [38] to extract appearance features. Besides, we also use some statistic features of terms and commodities, e.g., the occurrence frequency and popularity degree of goods.

Below we define the cross graph $\widetilde{\mathcal{G}} = (\widetilde{\mathcal{V}}, \widetilde{\mathbf{A}}, \widetilde{\mathbf{S}})$, which bridges users and commodities. The node set consists of pairwise combinations of nodes respectively from $\mathcal{V}_u$ and $\mathcal{V}_c$, i.e., $\widetilde{\mathcal{V}} = \{(v_i, v_j) | v_i \in \mathcal{V}_u, v_j \in \mathcal{V}_c\}$ and $|\widetilde{\mathcal{V}}| = n_u n_c$. Next, we derive the conjunctive signal matrix $\widetilde{\mathbf{S}} \in \mathbb{R}^{n_u n_c \times (d_u + d_c)}$ and conjunctive adjacency matrix $\widetilde{\mathbf{A}} \in \mathbb{R}^{n_u n_c \times n_u n_c}$ in the form of tensor decomposition.

For the signal matrix $\widetilde{\mathbf{S}}$, we formulate it as

$$\widetilde{\mathbf{S}} = [\widetilde{\mathbf{S}}_u, \widetilde{\mathbf{S}}_c] = \left[\mathbf{S}_u \otimes \mathbf{1}_{n_c \times 1},\ \mathbf{1}_{n_u \times 1} \otimes \mathbf{S}_c\right], \tag{6}$$

where the operation $\otimes$ means the tensor product, the square bracket $[\cdot, \cdot]$ denotes the concatenation of its inputs in columns, $\mathbf{1}$ is a matrix with all element values equal to 1, and $\widetilde{\mathbf{S}}_u \in \mathbb{R}^{n_u n_c \times d_u}$, $\widetilde{\mathbf{S}}_c \in \mathbb{R}^{n_u n_c \times d_c}$. More intuitively, in the cross-graph, the feature of one node (v_i, v_j) concatenates the features of the node $v_i \in \mathcal{V}_u$ and the node $v_j \in \mathcal{V}_c$ therein.

Further, we construct the adjacency matrix $\widetilde{\mathbf{A}}$ of the cross-graph, each element of which describes the connection relationship between two node pairs. Note that node pairs from $\widetilde{\mathcal{V}}$ are defined cross user and commodity graphs. We choose the Kronecker sum operation $\oplus$ to define conjunctive similarities of two given separate graphs:

$$\widetilde{\mathbf{A}} = \mathbf{A}_u \oplus \mathbf{A}_c = \mathbf{A}_u \otimes \mathbf{I}_{n_c} + \mathbf{I}_{n_u} \otimes \mathbf{A}_c, \tag{7}$$

where $\mathbf{I}$ is an identity matrix. Once the user/commodity graphs are fixed, the interaction relations cross graphs would be invariant based on the calculation of Eqn. (7). However, the recommendation of TPOs needs to learn the cross-graph relations $\widetilde{\mathbf{A}}$, rather than static relations. To address this problem, we take the parameterizing strategy in the computation of adjacent matrix on cross-graph. Concretely, we parameterize the Kronecker sum operation to learn an optimal conjunctive adjacency matrix from $\mathbf{A}_u$ and $\mathbf{A}_c$. The parameterized Kronecker sum operation, denoted as $\oplus_p$, is formulated as follows,

$$\widetilde{\mathbf{A}} = \mathbf{A}_u \oplus_p \mathbf{A}_c = \mathbf{A}_u \otimes \mathbf{\Lambda}_u + \mathbf{\Lambda}_c \otimes \mathbf{A}_c, \tag{8}$$

where $\mathbf{\Lambda}_u = \mathrm{diag}(\mathbf{w}_u), \mathbf{\Lambda}_c = \mathrm{diag}(\mathbf{w}_c)$ are diagonal matrices with the parameter vectors $\mathbf{w}_u \in \mathbb{R}^{n_c}, \mathbf{w}_c \in \mathbb{R}^{n_u}$ to be learnt.

Factorized graph filtering. According to the formulation of graph convolution in Eqn. (5) of Section III, we can define the cross-graph filtering with the polynomial form

$$\widetilde{\mathbf{S}}' = \sum_{k=0}^{K} \theta_k \widetilde{\mathbf{A}}^k \widetilde{\mathbf{S}} = [\sum_{k=0}^{K} \theta_k \widetilde{\mathbf{A}}^k \widetilde{\mathbf{S}}_u, \sum_{k=0}^{K} \theta_k \widetilde{\mathbf{A}}^k \widetilde{\mathbf{S}}_c], \tag{9}$$

where $\widetilde{\mathbf{A}}^k$ describes the k-hop reachable relationship (also called k-hop receptive field) in the cross-graph, θ_k is the polynomial coefficient to be learnt, K is the total order number of

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 21,2020 at 02:41:21 UTC from IEEE Xplore. Restrictions apply.

polynomial. As the cross-graph is with the multiplicative node number $n_u n_c$, the straightforward computation for Eqn. (9) will need high computation cost.

Below we derive a high efficient filtering by using tensor calculations. To simplify the derivation, we consider the case that the signal/feature $\widetilde{\mathbf{S}}$ is one-dimension (denoted as $\widetilde{\mathbf{s}}$). The following conclusion can be easily extended to multi-dimensional signal/feature. Thus the basic calculation in graph convolution can be derived as

$$\widetilde{\mathbf{A}}\widetilde{\mathbf{s}} = (\mathbf{A}_u \otimes \mathbf{\Lambda}_u + \mathbf{\Lambda}_c \otimes \mathbf{A}_c)\widetilde{\mathbf{s}} \tag{10}$$

$$= vec(\mathbf{\Lambda}_u mat(\widetilde{\mathbf{s}})\mathbf{A}_u^\intercal) + vec(\mathbf{A}_c mat(\widetilde{\mathbf{s}})\mathbf{\Lambda}_c), \tag{11}$$

where $vec(\cdot)$ denotes the vectorization of a matrix through column by column stacking, and $mat(\cdot)$ is the reverse process of $vec(\cdot)$ which transforms a column vector into a matrix. This derivation in Eqn. (11) reduces the computation complexity from $O(n_u^2 n_c^2)$ to $O(n_u^2 n_c + n_c^2 n_u)$. Further, we can have the following derivation for Eqn. (11),

$$\tilde{\mathbf{A}}\tilde{\mathbf{s}} = vec(\mathbf{\Lambda}_u mat(\tilde{\mathbf{s}})\mathbf{A}_u^\intercal) + vec(\mathbf{A}_c mat(\tilde{\mathbf{s}})\mathbf{\Lambda}_c) \tag{12}$$

$$= vec(\mathbf{\Lambda}_u \mathbf{1}_{n_c}\mathbf{s}^\intercal \mathbf{A}_u^\intercal) + vec(\mathbf{A}_c \mathbf{1}_{n_c}\mathbf{s}^\intercal \mathbf{\Lambda}_c) \tag{13}$$

$$= vec((\mathbf{\Lambda}_u \mathbf{1}_{n_c})(\mathbf{s}^\intercal \mathbf{A}_u^T)) + vec((\mathbf{A}_c \mathbf{1}_{n_c})(\mathbf{s}^\intercal \mathbf{\Lambda}_c)), \tag{14}$$

where the calculation from Eqn. (12) to Eqn. (13) is based on Eqn. (6) by using the equation,

$$mat(\tilde{\mathbf{s}}) = mat(\mathbf{s} \otimes \mathbf{1}_{n_c}) = \mathbf{1}_{n_c}\mathbf{s}^\intercal. \tag{15}$$

Until now, the computation complexity is reduced to $O(n_u^2 + n_c^2)$ for Eqn. (14), where $\mathbf{1}_{n_c}, \mathbf{s}$ are vectors and $\mathbf{\Lambda}_u$, $\mathbf{\Lambda}_c$ are diagonal matrices.

For high-order cases, we take the recursive calculation as

$$\widetilde{\mathbf{A}}^k\widetilde{\mathbf{s}}_1 = \widetilde{\mathbf{A}}(\widetilde{\mathbf{A}}^{k-1}\widetilde{\mathbf{s}}) \tag{16}$$

$$= vec(\mathbf{\Lambda}_u mat(\widetilde{\mathbf{A}}^{k-1}\widetilde{\mathbf{s}})\mathbf{A}_u^\intercal + \mathbf{A}_c mat(\widetilde{\mathbf{A}}^{k-1}\widetilde{\mathbf{s}})\mathbf{\Lambda}_c). \tag{17}$$

The derivation process from Eqn. (10) to (17) takes a series of tensor and matrix calculations. It can be easily extended to multi-dimension input signals.

Loss function. Besides cross graph convolution learning on original graph features, a fully connected network is employed to learn high-level features from the original embedding or appearance features. These two parts of features from graph convolution and fully connection network are concatenated and fed into an additional fully connected layer followed by a softmax layer to predict the final preference scores.

In the objective function, we employ the focal loss to learn parameters of the model, which can well alleviate the problem of class imbalance. Given a pair of user and TPO, suppose the prediction score from our CGCL model is $\mathbf{o}_i = [o_{i0}, o_{i1}]$ where o_{i0}, o_{i1} represent their dissimilarity and similarity scores. Then the focal loss can be written as:

$$\zeta = -\sum_{i=1}^{N}\sum_{j=0}^{1} \tau(c, y_i) \times (1 - o_{ij})^\gamma \times \log(o_{ij}), \tag{18}$$

where γ is the focusing parameter, y_i is the label of the i-th training sample and

$$\tau(x, y) = \begin{cases} 1, & \text{if } x = y; \\ 0, & \text{otherwise.} \end{cases}$$

The focal loss reduces the loss contribution for easy samples, which are more likely negative samples due to their larger amount, and meanwhile increases the importance of misclassified examples of most positive samples.

VI. Experiments

We evaluate our proposed CGCL model on the popular Taobao platform. Two stages of experiments are conducted, including offline validation and online A/B test. In the offline validation, the TPSG model is trained on some historical search records of the online e-commerce platform, and then this model is further deployed on the platform for online test. To achieve comprehensive evaluations, we compare our proposed CGCL with several baseline methods in both offline and online processes. In the following parts, we first describe the experiment setup including the offline dataset, baseline methods and implement details, then we report and analyze the experimental results.

A. Experiment Setup

Offline dataset construction. The offline dataset is collected from the log data of three days on the online Taobao platform, where the data sampled from two days are used for training and another one day for testing. Each record in the log data corresponds to an input query and contains the necessary information of the user and the presented TPOs, including the basic attributes of the user and commodity, the textual content of the term, etc. In the process of sample labeling, TPOs clicked by users can be treated as positive samples, but the non-clicked samples do not surely mean the irrelevance. To obtain negative samples with high quality, we further take some strategies, e.g., only if the effectively displaying TPOs in a page are all not clicked, can they be treated as negative samples. After collection, the final offline dataset contains about 3.88 million different users, 250 thousand commodities and 4 million different TPOs. There are totally 6 million samples for training and 120 thousand samples for testing, where the training set consists of about 0.5 million positive and 5.5 million negative samples, respectively.

The baseline methods. Few existed algorithms can be directly applied to TPSG as it is a new task. Considering the large-scale characteristic, high dynamics and high speed requirement of online Taobao platform, the content-based filtering methods are employed as baselines. The collaborative filtering methods such as matrix completion [6] and deepwalk [3], [4], are not considered as they are not available for the online Taobao platform. To verify our proposed cross-graph convolution algorithm, we adapt two deep frameworks, i.e., DNN and GCNN, to the TPSG task as the baseline methods. The GCNN-based model applies vanilla graph convolutions separately on user and commodity graphs, and then combines the output

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 21,2020 at 02:41:21 UTC from IEEE Xplore. Restrictions apply.

features through fully connected layers. The DNN-based model just employs fully connected layers to learn features from user and commodity attributes.

Implement details. For the user graph construction, three attributes of users, including "gender" (3 categories including "NULL"), "age" (9 groups, larger number means higher age), and "purchasing power" (8 levels, larger number means higher power), are extracted from either the registered personal information or statistics of purchasing history. Each attribute of the user as a node in the user graph is described with a one-hot vector, e.g., "age" is represented as a 9-dimension one-hot vector as it is divided into 9 groups. The one-hot vector is further projected into a 5-dimension feature space through matrix projection (fully connected layer here). For the commodity graph, 11 attributes of commodities are selected based on the commodity description provided by online sellers, e.g., "style", "element", "brand", etc. Different from the user attributes, the embedding features with 16 dimensions of the commodity attributes are obtained by `word2vec` [39], whose model is trained on the words of commodity attributes from the online e-commerce platform.

In addition to the user and commodity attributes, other text inputs, including user queries and text terms in TPOs, are also embedded into 16-dimension features through `word2vec`. Moreover, for pictures in TPOs, we use the 152-layer ResNet [38], which is pretrained on some pictures of commodities of the Taobao platform, to extract the 2048-dimension appearance features from the fully connected layer. Other statistic features are set to 9-dimension vectors. Each extracted feature above passes through an additional fully connected layer for high-level feature learning, where the output dimension is traversed in the range of [16, 32, 48, 64, 128]. Then, the obtained features are concatenated with the output of the cross-graph convolution to predict the similarity score. Besides, the polynomial order number K is traversed in the range of [1,2,3,4]. Finally, in the CGCL model, we decide the values of parameters through cross validation, based on which the dimension of fully connected layers is set to 32 and K is 3. In the training process, we run the CGCL as well as baseline methods with ten thousand rounds by using a learning rate of 0.001.

B. Offline testing

We train our proposed CGCL based on the offline dataset and compare the performance with the baseline methods. For a comprehensive evaluation, we employ multiple popular metrics used for ranking performance evaluation, including the hit rate ratio (HR@K) [40], [41], normalized discounted cumulative gain (NDCG@K) [40], [41], area under the ROC curve (AUC) [42] and group AUC (GAUC) [43]. HR@K measures whether the user's intended choice emerges in the set of Top-K recommendation; NDCG@K is a position-aware metric measuring Top-K ranking performance based on its position in the recommendation list, and formally it is defined as

$$NDCG@K = \frac{1}{N_u}\sum_{j=1}^{N_u}\frac{DCG@K(j,\pi_j)}{DCG@K(j,\pi_j^*)}, \tag{19}$$

where

$$DCG@K(j,\pi_j) = \sum_{k=1}^{K}\frac{2^{R_{j\pi_{jl}}}-1}{log_2(l+1)}. \tag{20}$$

For the DCG definition in Eqn. (20), π_{jl} means the index of the l-th ranked TPO for the j-th user, $R_{j\pi_{jl}}$ is the rating given to the π_{jl}-th TPO by the j-th user. AUC is a popular metric in recommendation systems to measure the overall ranking performance. Besides, we also employ GAUC to evaluate the performance. GAUC measures the ranking result based on the statistic of each user, and finally provides a weighted average of AUC over all users. As GAUC focuses on prediction for each user, it is more valuable to evaluate personalized recommendation systems. Formally, GAUC is defined as

$$GAUC = \frac{\sum_j N_{imp}^j * AUC_j}{\sum_j N_{imp}^j}. \tag{21}$$

where N_{imp}^j is the number of impression (i.e., the displaying TPOs number for user), and AUC_j corresponds to AUC of the j-th user.

The comparison results are reported in Table I. We have several observations: (i) our CGCL outperforms the baseline methods (GCNN and DNN) in all cases, which indicates the superiority of the proposed cross-graph convolution. (ii) The parameterized Kronecker sum operation can boost the performance by comparing CGCL and CGCL_np, where the latter directly uses Kronecker sum as in Eqn. (7). (iii) The graph based method GCNN is superior to DNN because the connections are additionally taken into consideration. Besides, we also observe that the values of HR@8 are almost the same for all methods. The reason is that the clicks of the top-8 TPOs are almost 90% of the whole in statistics.

C. Online A/B testing

To test the practical performance, we deploy our CGCL as well as the baseline methods to the large-scale online Taobao platform to perform online A/B test. In this process, users are randomly and evenly distributed to a certain number of buckets, where each bucket is deployed with one algorithm. Two types of CTR metrics highly related to economic benefits, i.e. CTR and unique visitor (UV) CTR, are reported in continuous 7 days. Concretely, CTR and UV CTR are defined as

$$\text{CTR} = \frac{\text{Number of clicks of TPOs}}{\text{Number of impressions of TPOs}}, \tag{22}$$

and

$$\text{UV CTR} = \frac{\text{Number of users who clicked TPOs}}{\text{Number of users who viewed TPOs}}. \tag{23}$$

The results are given in Table II. We first compare two types of TPOs, manual annotation and automatic recommendation of pictures, to verify the effectiveness of introducing an automatic framework for the TPSG task. Due to the limitation of labor resource, only the terms of 20 categories of commodities are attached with manually labeled pictures. To achieve a fair comparison, we specifically train a DNN-based model on this

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 21,2020 at 02:41:21 UTC from IEEE Xplore. Restrictions apply.

TABLE I: The comparisons of TPSG on offline dataset.

Method	HR@1	HR@2	HR@3	HR@4	HR@8	NDCG@2	NDCG@3	NDCG@4	NDCG@8	AUC	GAUC
DNN-based	0.304	0.520	0.679	0.788	0.930	0.434	0.513	0.561	0.614	0.616	0.601
GCNN-based	0.322	0.540	0.694	0.796	0.933	0.453	0.530	0.574	0.626	0.652	0.638
CGCL_np	0.326	0.545	0.695	0.797	0.933	0.458	0.532	0.577	0.629	0.653	0.640
CGCL	**0.335**	**0.556**	**0.702**	**0.800**	**0.934**	**0.468**	**0.540**	**0.584**	**0.634**	**0.667**	**0.657**

TABLE II: The comparisons of different methods through online A/B testing on both 20 and 179 categories.

Method	CTR of 20 categories	UV CTR of 20 categories	CTR of 179 categories	UV CTR of 179 categories
DNN_mrp	base	base	–	–
DNN-based	+11.5 %	+10.5 %	base	base
GCNN-based	+15.8 %	+13.2 %	+4.4%	+3.5%
CGCL	**+17.7** %	**+14.5** %	**+9.1**%	**+6.8**%

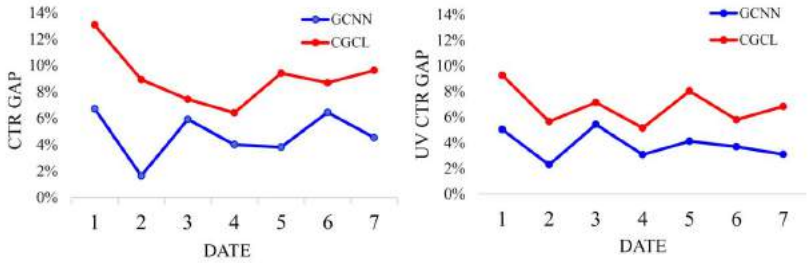


Fig. 5: The CTR and UV CTR gaps of GCNN and CGCL over DNN in one week.

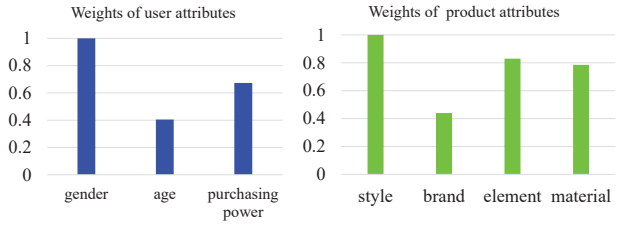


Fig. 6: The weights (also parameter values) of attributes of users and commodities.

part of categories as the automatic framework for TPOs. Then, the pictures of DNN-recommended TPOs are replaced with the manually selected ones, which is referred to as DNN_mrp (i.e., DNN-selected terms with manually recommended pictures). The comparison results are shown in the middle column of Table II. DNN achieves better performance than DNN_mrp in both CTR and UV CTR. The gain can reach more than 10%. It demonstrates that automatic recommendation becomes feasible and can substitute manual labeling in the tedious and cumbersome task of TPSG.

Next we compare our CGCL with the baseline methods on the commodities of 20 and 179 categories. The online test results are shown in Table II. Our CGCL achieves the best online performance in both situations of 20 and 179 categories. Here we focus on analysing the results of 179 categories as they provide a more comprehensive evaluation. Treating the DNN-based method as the standard baseline, GCNN improves the performance of CTR and UV CTR by 4.4% and 3.5% respectively. The reason behind is that the connections are used in the GCNN while DNN does not use them. Further, the CGCL outperforms GCNN with 4.7% higher CTR and 3.3% higher UV CTR, which can be attributed to the effect of cross graphs between users and commodities. Therefore, the above online A/B tests demonstrate the superiority of our proposed cross-graph convolution method for the TPSG task.

Moreover, the comparisons of continuous 7 days on the commodities of 179 categories are plotted in Fig. 5. We give the relative gaps of GCNN and CGCL over DNN, i.e., the values of two CTR metrics of DNN are subtracted from those of GCNN and CGCL. As shown in this figure, the performances of GCNN and CGCL are persistently higher than DNN, and CGCL also achieves stable performance gains when comparing to GCNN.

D. Analysis

In addition to verifying the effectiveness of cross-graph convolution over baseline methods, we further provide some insights with the following analysis: (i) analyse preferences learned by the cross-graph convolution, (ii) show some examples of personalized TPOs learned by our CGCL.

Analysis of preferences. According to Eqn. (8), the learnt parameter matrices $\mathbf{\Lambda}_u$ and $\mathbf{\Lambda}_c$ are related to the cross patterns

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 21,2020 at 02:41:21 UTC from IEEE Xplore. Restrictions apply.

Fig. 7: Online recommended examples of our CGCL. Left part: recommended TPOs with the query of "Coat"; Right part: recommended pictures in TPOs with the query of "dress" and the term of "Medium skirt". "Age-2", "Age-4" and "Age-5" denote the age ranges of [18, 24], [30, 34] and [35, 39], respectively.

of preferences. Specifically, the parameter w.r.t one specific attribute (node), e.g., "gender" in the user graph, represents the correlation degree between this node and all nodes of the commodity graph. Similarly, the parameter w.r.t a commodity attribute represents the interaction degree between it and all nodes of the user graph. Further according to Eqn. (9), we can conclude that higher values of diagonal elements in both $\mathbf{\Lambda_u}$ and $\mathbf{\Lambda_c}$ indicate the stronger connections among the corresponding nodes of the two graphs.

Based on the above analysis, we visualize some learned parameters in $\mathbf{\Lambda}_u$ and $\mathbf{\Lambda}_c$ in Fig. 6. Note that in the experiment the parameters can converge into a relatively stable point after about eight thousand training rounds. For better exhibition, we normalize the largest values of these parameters in $\mathbf{\Lambda}_u$ and $\mathbf{\Lambda}_c$ to 1 respectively. From Fig. 6, we can observe that users of different ages and purchasing powers pay much attention to "style", "element" and "material", but less attention on "brand". Comparing to "gender" and "purchasing power", the attribute of "age" has a lower connection degree to commodity attributes.

Further, we also give the statistical preferences of different "genders" for "style" of backpack and T-shirt in Table III. The results indicate there is a large gap between the preferences of males and females, which coincides with the high connection degree between "gender" and "style" as shown in Fig. 6.

Online examples of TPOs. We show two examples of personalized TPOs in Fig. 7, which are recommended by our CGCL. "Age-2", "Age-4" and "Age-5" denote the age ranges of [18, 24], [30, 34] and [35, 39], respectively. In the right part of Fig. 7, we display two specific attributes, i.e. "age" and "purchasing power", as their connection strengths to commodities are relatively lower than "gender". As shown in these examples, our proposed CGCL may recommend suitable TPOs to users of different ages, genders and purchasing powers. It also indicates that our proposed CGCL can achieve personalized recommendations for users, even though the learned connection degree of "age" and "purchasing power" is relatively low.

TABLE III: The statistical preferences of different genders for the attribute of "style".

	backpack	T-shirt
male	sporty, leisure, business	fashion, leisure, sporty
female	Korean, cute, cartoon	young, Korean, cute

VII. Conclusion

In this paper, we investigated a new search service named TPSG and then deployed it to one popular online e-commerce platform Taobao. The TPSG system well recommends personalized TPOs, which are more user-friendly to shopping guide than traditional text options. To facilitate better recommendation, we proposed the cross graph convolution learning method by modeling the TPSG system into one mixture graph, which describes all attributes/relations of users and commodities. Further, we decomposed the entire huge relation graph into three types of graphs so that personalized preferences can be well represented. Accordingly, we developed an efficient tensor graph convolution method to learn representation on the factorized graphs, i.e., user graphs, commodity graphs and cross user-commodity graphs. The experiments of both the offline and online A/B tests demonstrated the effectiveness of our proposed CGCL method for the personalized recommendation of TPOs. In future work, we will introduce attention mechanism into our cross-graph convolution method for better self-adaptation.

VIII. ACKNOWLEDGMENT

The authors would like to thank Professor Qiong Luo and the anonymous reviewers for their valuable comments which help to improve this paper. This work was supported by the National Natural Science Foundation of China (Grants Nos. 61906094, 61772276, 61972204, 61971236), the Natural Science Foundation of Jiangsu Province (Grants Nos. BK20190452, BK20190019), and the fundamental research funds for the central universities (No. 30919011232).

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 21,2020 at 02:41:21 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang, "Representation learning for attributed multiplex heterogeneous network," *arXiv preprint arXiv:1905.01669*, 2019.

[2] M. Qu, J. Tang, J. Shang, X. Ren, M. Zhang, and J. Han, "An attention-based collaboration framework for multi-view network representation learning," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 1767–1776.

[3] H. Zhang, L. Qiu, L. Yi, and Y. Song, "Scalable multiplex network embedding." in *IJCAI*, vol. 18, 2018, pp. 3082–3088.

[4] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2017, pp. 135–144.

[5] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[6] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 3697–3707.

[7] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[8] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[10] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016, pp. 3844–3852.

[11] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2018.

[12] S. Rhee, S. Seo, and S. Kim, "Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification," *arXiv preprint arXiv:1711.05859*, 2017.

[13] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," *arXiv preprint arXiv:1801.03226*, 2018.

[14] C. Li, Z. Cui, W. Zheng, C. Xu, and J. Yang, "Spatio-temporal graph convolution for skeleton based action recognition," *arXiv preprint arXiv:1802.09834*, 2018.

[15] C. Li, Z. Cui, W. Zheng, C. Xu, R. Ji, and J. Yang, "Action-attending graphic neural network," *IEEE Transactions on Image Processing*, vol. 27, no. 7, pp. 3657–3670, 2018.

[16] C. Chu, Z. Li, B. Xin, F. Peng, C. Liu, R. Rohs, Q. Luo, and J. Zhou, "Deep graph embedding for ranking optimization in e-commerce," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 2007–2015.

[17] M. Tomassini and L. Luthi, "Empirical analysis of the evolution of a scientific collaboration network," *Physica A: Statistical Mechanics and its Applications*, vol. 385, no. 2, pp. 750–764, 2007.

[18] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon, "Collaborative filtering with graph information: Consistency and scalable methods," in *Advances in neural information processing systems*, 2015, pp. 2107–2115.

[19] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[20] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 974–983.

[21] J. Zhang, X. Shi, S. Zhao, and I. King, "Star-gcn: Stacked and reconstructed graph convolutional networks for recommender systems," *arXiv preprint arXiv:1905.13129*, 2019.

[22] T. Bansal, D. Belanger, and A. McCallum, "Ask the gru: Multi-task learning for deep text recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 107–114.

[23] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The adaptive web*. Springer, 2007, pp. 325–341.

[24] B. Bai, Y. Fan, W. Tan, and J. Zhang, "Dltsr: A deep learning framework for recommendation of long-tail web services," *IEEE Transactions on Services Computing*, 2017.

[25] A. Van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Advances in neural information processing systems*, 2013, pp. 2643–2651.

[26] K.-J. Oh, W.-J. Lee, C.-G. Lim, and H.-J. Choi, "Personalized news recommendation using classified keywords to capture user preference," in *Advanced Communication Technology (ICACT), 2014 16th International Conference on*. IEEE, 2014, pp. 1283–1287.

[27] D. Wang, Y. Liang, D. Xu, X. Feng, and R. Guan, "A content-based recommender system for computer science publications," *Knowledge-Based Systems*, vol. 157, pp. 1–9, 2018.

[28] C.-N. Ziegler, L. Schmidt-Thieme, and G. Lausen, "Exploiting semantic product descriptions for recommender systems," in *Proceedings of the 2nd ACM SIGIR Semantic Web and Information Retrieval Workshop*, 2004, pp. 25–29.

[29] E. Aslanian, M. Radmanesh, and M. Jalili, "Hybrid recommender systems based on content feature relationship," *IEEE Transactions on Industrial Informatics*, 2016.

[30] A. Gunawardana, C. Meek *et al.*, "A unified approach to building hybrid recommender systems." *RecSys*, vol. 9, pp. 117–124, 2009.

[31] R. Burke, "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[32] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, p. 717, 2009.

[33] V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst, "Matrix completion on graphs," *arXiv preprint arXiv:1408.1717*, 2014.

[34] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[35] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 191–198.

[36] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "Signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular data domains," *arXiv preprint arXiv:1211.0053*, 2012.

[37] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE transactions on signal processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[39] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[40] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender systems handbook*. Springer, 2011, pp. 1–35.

[41] T. Ebesu, B. Shen, and Y. Fang, "Collaborative memory network for recommendation systems," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 2018, pp. 515–524.

[42] T. Ge, L. Zhao, G. Zhou, K. Chen, S. Liu, H. Yi, Z. Hu, B. Liu, P. Sun, H. Liu *et al.*, "Image matters: Visually modeling user behaviors using advanced model server," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2018, pp. 2087–2095.

[43] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1059–1068.

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 21,2020 at 02:41:21 UTC from IEEE Xplore. Restrictions apply.

2020 IEEE 36th International Conference on Data Engineering (ICDE)

PoisonRec: An Adaptive Data Poisoning Framework for Attacking Black-box Recommender Systems

Junshuai Song†, Zhao Li*‡, Zehong Hu*, Yucheng Wu†, Zhenpeng Li*, Jian Li* and Jun Gao†‡
†The Key Laboratory of High Confidence Software Technologies, Ministry of Education,
Department of Computer Science, Peking University, China
Email: {songjunshuai,wuyucheng,gaojun}@pku.edu.cn
*Alibaba Group, China
Email: {lizhao.lz,zehong.hzh,zhen.lzp,zeshan.lj}@alibaba-inc.com
‡The corresponding authors

***Abstract*—Data-driven recommender systems that can help to predict users' preferences are deployed in many real online service platforms. Several studies show that they are vulnerable to data poisoning attacks, and attackers have the ability to mislead the system to perform as their desires. Considering the realistic scenario, where the recommender system is usually a black-box for attackers and complex algorithms may be deployed in them, how to learn effective attack strategies on such recommender systems is still an under-explored problem. In this paper, we propose an adaptive data poisoning framework, PoisonRec, which can automatically learn effective attack strategies on various recommender systems with very limited knowledge. PoisonRec leverages the reinforcement learning architecture, in which an attack agent actively injects fake data (user behaviors) into the recommender system, and then can improve its attack strategies through reward signals that are available under the strict black-box setting. Specifically, we model the attack behavior trajectory as the Markov Decision Process (MDP) in reinforcement learning. We also design a Biased Complete Binary Tree (BCBT) to reformulate the action space for better attack performance. We adopt 8 widely-used representative recommendation algorithms as our testbeds, and make extensive experiments on 4 different real-world datasets. The results show that PoisonRec has the ability to achieve good attack performance on various recommender systems with limited knowledge.**

***Index Terms*—Data Poisoning Attack, Recommender System, Reinforcement Learning**

I. Introduction

Recommender system, as a data-driven way to predict online users' potential preferences, is playing an increasingly important role in online service platforms. The basic idea behind recommender systems is collaborative filtering (CF) [1]–[3], which can help us make predictions about users' potential interests by collecting other users' preference information. In recent years, many recommendation algorithms have been proposed and achieved great success in practical applications [4]–[9].

There have been some evidences showing that existing recommender systems are vulnerable under data poisoning attacks (also known as shilling attacks) [1], [10], [11]. Some attackers may inject some fake data into a recommender system, after which the target items may be recommended to users more frequently than before. For example, Yang et al. [12] have successfully performed the item promotion attack on several real-world popular services like Youtube and eBay where the co-visitation recommender system [12] is deployed. It is profitable for attackers, and we mainly consider such an item promotion problem in this work.

Fig. 1. An attack example. The attacker tries to promote the target item *Hat* by frequently clicking *Hat* and another popular item *Chair*.

In the literature, there are mainly two categories of attack methods proposed. The first category is manually designed heuristic methods, which follow simple rules to conduct attacks [1], [10], [11], [13], [14]. For example, a common strategy is to frequently click the target item and another popular item so that the recommender system may be misled that the target item is closely related to the popular one as illustrated in Figure 1. Then, users who have bought the popular item may receive recommended results including the target item. An obvious drawback of methods in this category is that the simple rules cannot always achieve satisfying attack performance on various recommender systems.

Another category of attack methods is learning-based [12], [15]–[17]. Most of them are particularly designed for given recommender systems [12], [15], [16] where specific recommendation algorithms are deployed. In addition, they usually requ
infc
R

2375-026X/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDE48307.2020.00021

IEEE computer society

Authorized licensed use limited to: Zhejiang Tmall Technology Co.Ltd.. Downloaded on August 26,2020 at 12:14:24 UTC from IEEE Xplore. Restrictions apply.

practical. They are always required protecting users' privacy, and users' profile information and interactions with the system are confidential. For security reasons, they will not disclose their system composition, algorithms, etc. We consider such a realistic black-box scenario, which means the recommender system is totally agnostic to attackers. It is obvious that attacking a black-box recommender system is very challenging. On the one hand, we only have very limited background knowledge about the recommender system. We know nothing about the log data, the system components, and the recommendation algorithms, etc. On the other hand, there are various algorithms developed in the system, which might have been upgraded and be far more complex than before [3], [6], [9] with the development of the deep learning technique. It is very difficult and also time-consuming to directly design effective attacks for each of them.

To our best knowledge, it is still an under-explored problem to learn effective attack strategies on different complex black-box recommender systems. Model-free reinforcement learning [18]–[20] could help with this goal, which does not make any assumption, and requires no (or limited) knowledge on the interactive environment (the recommender system to be attacked in this work). Inspired by its success in many areas [21]–[24], we adopt it as the core component of our solution.

We propose PoisonRec, an adaptive data poisoning attack framework. Since PoisonRec is decoupled from the interactive recommender system under the model-free reinforcement learning architecture, it can learn effective attack strategies on various recommender systems, and adopt any suitable advanced reinforcement learning method. It repeatedly injects generated fake user behaviors into a recommender system, and improves its own attack strategies through available reward signals until the model converges. Specifically, we model the sequential attack behavior trajectory as an MDP. Given the state s_t that contains a variable-length attack trajectory at step t, we use the long short-term memory (LSTM) network to embed them into a fixed-length hidden variable, based on which the next action a_t can be sampled. For the reward signal, we consider the observable number of Page View (PV) [22], [25] on a pre-defined target item set under the black-box setting. To overcome the low convergency problem on a large action space in reinforcement learning, we further reformulate the action space through a Biased Complete Binary Tree (BCBT), which can obviously improve the model performance and convergency efficiency of PoisonRec. We adopt real-world data from Steam, MovieLens and Amazon, and choose 8 different widely-used recommendation algorithms as our attack testbeds. Extensive experiments show that PoisonRec has the ability to adaptively learn effective attack strategies on different kinds of recommender systems.

The remainder of this paper is organized as follows. We first review preliminaries in Section II, and then propose our PoisonRec framework in Section III. Section IV reports the experimental results. Finally, we conclude the paper in Section V.

II. Preliminaries

We review the development of recommender systems in recent years, and discuss existing attack methods.

A. Development of Recommender Systems

Recommender systems can predict users' preferences, and have been greatly developed in recent years. ItemKNN [12], [26] is an item-based CF method, and it recommends items through calculating similarities between items. MF [2] is a basic model-based CF recommendation algorithm. BPR [5] optimizes MF through a pair-wise ranking loss. With the development of the deep neural network technique, many deep learning-based methods [3], [4] are also proposed. He et al. [3] give NeuMF to replace the inner product in MF with a neural architecture, so that it can learn an arbitrary user-item preference function from data. Suvash et al. [27] utilize the auto-encoder paradigm and propose AutoRec, which can directly predict users' potential interests towards all items. Besides, sequential and graph structure information proved to have obvious benefits in predicting users' preferences. Balázs et al. [6] and Zhou et al. [28] utilize users' historical behaviors to predict the next item that the user may click or purchase. Wang et al. [9] exploit the high-order structure information in the user-item graph, and achieve much better recommendation performance.

B. Attack Methods on Recommender Systems

The most common attack methods against recommender systems are manually designed heuristic methods [1], [10], [11] like popular attack and random attack. They can be applied to any type of recommender systems, but the results are usually not satisfying. Recent studies on attacking recommender systems achieve much better performance [12]–[17]. Seminario et al. [13], [14] propose to select "power users" and "power items" in advance when developing the attack strategy. Yang et al. [12] try to attack a simple non-personalized co-visitation recommender system. They model the attack strategy as a constrained linear optimization problem, by solving which they can spoof the recommender system to make recommendations as their desires. Christakopoulou et al. [17] propose a method that can be adopted to attack various recommender systems, in which they first generate fake rating scores for the multiple controlled user accounts to initialize a matrix, and then use an *approximate gradient* method to iteratively update the matrix for obtaining the effective attack strategy.

III. PoisonRec: An Adaptive Data Poisoning Attack Framework

In this section, we first summarize some frequently-used notations in Table I, and give an overview of real recommender systems and attackers' knowledge requirements under the black-box setting. Then, we formulate the attack problem, and finally propose the novel adaptive data poisoning attack framework, PoisonRec.

Authorized licensed use limited to: Zhejiang Tmall Technology Co.Ltd.. Downloaded on August 26,2020 at 12:14:24 UTC from IEEE Xplore. Restrictions apply.

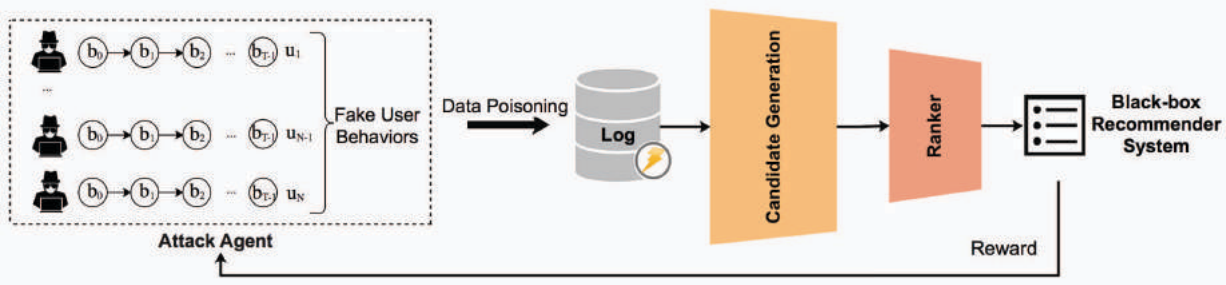


Fig. 2. The overall framework of PoisonRec.

Notation	Explanation
C_u	It is the candidate set that a user u may interested in.
L_u	It is the list recommended to a user u.
I	It represents the original (existing) item set in a recommender system.
I_t	It is the target item set that the attacker wants to promote, which consists of 8 new items in this paper.
N	It is the number of attackers.
τ	It means an attack trajectory that consists of T items.

TABLE I
FREQUENTLY-USED NOTATIONS.

A. Overview

1) Recommender System: A typical recommender system usually consists of at least two separate components: Candidate Generation and Ranker. Candidate Generation [4] is responsible for selecting hundreds of items from the entire corpus into the candidate set C_u for each user u. Ranker [3], [6], [9], [29] is used to rank the items in C_u, which can more accurately estimate the user preference scores on items, and then the top-k items with the highest scores could be selected as the final recommendation result L_u.

2) Attack Background: We discuss the attacker's goal, knowledge, and capability in this part.

Attacker's Goal. We introduce a widely-used concept, the number of Page View (PV) [22], [25], that measures items' exposure within a certain period of time in the real online recommender system. We formally defined the number of PV on a pre-defined target item set I_t as $RecNum$. Specifically, we give:

$$RecNum = \sum_u |L_u \cap I_t| \tag{1}$$

where u is a user accessing the system, L_u indicates the recommended results, and I_t is the target item set. Though attackers cannot obtain L_u for other users when attacking a real recommender system under the black-box setting, $RecNum$, as a statistical indicator, is usually available. Attackers know how many users have viewed their products, how many goods they have sold, etc. Given a target item set I_t, the attacker's goal is to increase $RecNum$ as much as possible. In addition, it is worth explaining how data poisoning attack affects $RecNum$. The recommendation model in a recommender system is always trained on all users' log data. If the attacker injects some fake user data into a recommender system, this part of data will affect the training of the recommendation model in this attacked recommender system, and get the model to learn something it should not. If the attacked recommender system is misled that a target item is a popular one, many other users may receive this target item in their recommendation results, leading to a high $RecNum$.

Attacker's Knowledge. We focus on learning highly effective strategies to attack various complex recommender systems under the black-box setting.

- Attackers do not know any detail about the target recommender system, *e.g.*, what components it contains, or what the Candidate Generation and Ranker algorithms are.
- We focus on recommender systems with implicit feedback [5] instead of explicit user-item rating information, in which the interactions between other users and the systems are private, and cannot be used as attackers' background knowledge. Attackers can only crawl basic item information like item title, item description and item popularity (the sales volume of an item).
- Attackers can only use available signals like the number of PV on the pre-defined target items to improve the attack strategy under the black-box setting. More specific information like the user-item preference scores given by the Ranker algorithm in a recommender system is agnostic to attackers.

Attacker's capability. When attacking real-world recommender systems, attackers can perform different operations, like click and purchase, on items by order, which means that some fake behavior data is injected into the history log of a system. Attackers can also do some operations like changing the attributes, but we do not discuss these cases. Other different types of attacks like removing logs from the recommender system, which require strict permission, are not allowed here.

As illustrated in Figure 2, there is an attack agent in PoisonRec. It interacts with a black-box recommender system through generated fake user behavior data, and then improves the attack strategies based on available reward signals. For convenience, we use the click behavior for the attack trajectory in the following parts.

Authorized licensed use limited to: Zhejiang Tmall Technology Co.Ltd.. Downloaded on August 26,2020 at 12:14:24 UTC from IEEE Xplore. Restrictions apply.

B. Problem Definition

N is the number of attackers. I is the original item set in a recommender system, and I_t is the pre-defined target item set. $\tau = \{a_0, a_1, ..., a_{T-1}\}$ is a T-length attack trajectory. The possible discrete space that an attacker may search is in the form of:

$$\tau = (a_0, a_1, ..., a_{T-1}),\ a_t \in I \cup I_t \tag{2}$$

where a_t refers to the sampled item to be clicked at step t, and $t \in [0, 1, ..., T-1]$. After receiving all attackers' fake trajectories $\boldsymbol{\tau} = [\tau^0, \tau^1, ..., \tau^{N-1}]$, they can be injected into a recommender system. Then, the goal is to find the policy network $\pi_\theta(\cdot)$ of which $R(\boldsymbol{\tau})$ is maximized:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau^i \sim \pi_\theta(\cdot)}\{R(\boldsymbol{\tau})\} \tag{3}$$

where τ^i is sampled from the policy network $\pi_\theta(\cdot)$, $i \in [0, 1, ..., N-1]$, and $R(\boldsymbol{\tau})$ is the reward function, which will be further introduced in Equation 4.

C. Model Design

We formulate the attack trajectory as an MDP in reinforcement learning [20], [30], [31]. Since the N attackers are homogeneous, we require that they share the same policy network $\pi_\theta(\cdot)$ as in [32]. In the following, for convenience, we discuss the model design with a single attacker ($N = 1$), while the definition can be expanded to multiple attackers in a trivial way.

State. S is the state space, and we use $s_t = \{u, a_0, a_1, ..., a_{t-1}\} \in S$ to represent the state at step t, which consists of the current attacker u and all the selected sequential actions (items) so far.

Action. A is the action space, and we have $|A| = |I| + |I_t|$. The action at time step t is to predict the item a_t that needs to be clicked, and a_t belongs to $I \cup I_t$.

Reward. As discussed in the Attacker's Goal in Section III-A2, the number of PV ($RecNum$) is a realistic and available evaluation indicator that can be used as the reward signal to guide the optimization process of PoisonRec. The reward function is defined as:

$$R(\tau) = \sum_{t=0}^{T-1} \gamma^t R_t(s_t, a_t) = RecNum \tag{4}$$

where $\tau = \{a_0, a_1, ..., a_{T-1}\}$ is the sampled attack trajectory, $R_t(s_t, a_t)$ is the received reward at step t, and γ is the discount rate. $RecNum$ is calculated after the complete attack trajectory is injected into the recommender system, and we set $\gamma = 1.0$, $R_t(s_t, a_t) = 0,\ 0 \le t \le T-2$ and $R_t(s_t, a_t) = RecNum,\ t = T-1$. With multiple attackers, the set of sampled trajectories $\boldsymbol{\tau}$ is injected into a recommender system together, and we have $R(\boldsymbol{\tau}) = RecNum$, which is then used to direct the training process of PoisonRec.

Policy Network. There are two parameterized neural networks, an LSTM network [33] and a deep neural network (DNN), in the policy network $\pi_\theta(\cdot)$ used here. In practice, given the state s_t=$\{u, a_0, a_1, ..., a_{t-1}\}$, we first exploit the LSTM network to embed it into a fixed-length hidden variable. Formally, we define:

$$h_t = \text{LSTM}(e_u, e_{a_0}, e_{a_1} ..., e_{a_{t-1}}) \tag{5}$$

where a_t represents the sampled item at step t, e_u and e_{a_t} means the feature representation for user u and item a_t, respectively. We let $|h| = |e|$, and set the size of all hidden layers in LSTM to be equal to $|e|$.

Then we sample next action a_t from the policy network, whose output follows a multinomial distribution on the action space A. For a given h_t, we sample a_t according to:

$$a_t \sim \pi_\theta(\cdot|s_t) = \frac{exp(\mathcal{D}(h_t) \cdot e_j)}{\sum_{j \in I \cup I_t} exp(\mathcal{D}(h_t) \cdot e_j)} \tag{6}$$

where e_j is the feature representation of item j, $\mathcal{D}(\cdot)$ is a DNN whose output dimension is equal to $|e_j|$, and $\mathcal{D}(h_t) \cdot e_j$ is the point-wise product between them. More specifically, we adopt a 2-layer DNN with $Relu$ as the activation function for $\mathcal{D}(\cdot)$, and the size of each layer is also set to $|e|$.

Algorithm 1 PoisonRec Framework

Input: policy network $\pi_\theta(\cdot)$, learning rate α, the number of attackers N, the length of an attack trajectory T, the number of sampled training examples in each training step M, the batch size B, and the number of epochs K.
output: the learned policy network $\pi_\theta(\cdot)$.
function POISONREC($\pi_\theta(\cdot)$, α, N, T, M, B, K)
    Initialize the parameters in the policy network $\pi_\theta(\cdot)$.
    for $step \leftarrow 1, 2, ...$ **do**
        for each $m < M$ **do**
            Sample $\tau^u \leftarrow \{a_0, a_1, ..., a_{T-1}\}$ for each attacker u from the policy network $\pi_\theta(\cdot)$.
            Get the poisoning dataset $D^p \leftarrow \boldsymbol{\tau}$ that consists of the N attackers' trajectories.
            DataPoisoning(D^p).
            $R(\boldsymbol{\tau}) \leftarrow RecNum$ according to Equation 1.
            Store the current example.
        end for
        for each $k < K$ **do**
            Sample a batch of examples where $B \le M$.
            Normalize $\boldsymbol{R}(\boldsymbol{\tau})$ according to Equation 8.
            Update $\pi_\theta(\cdot)$ with normalized rewards following Equation 7.
        end for
    end for
end function

function DATAPOISONING(D^p)
    // Fake data works in a black-box recommender system, which is agnostic to the attacker.
    Reload the Ranker R.
    Update R with D^p.
end function

Authorized licensed use limited to: Zhejiang Tmall Technology Co.Ltd.. Downloaded on August 26,2020 at 12:14:24 UTC from IEEE Xplore. Restrictions apply.

D. Model Solving

Considering the expensive interaction cost, we exploit the highly popular PPO method in PoisonRec, which can obviously improve the efficiency of sample utilization.

We adopt the *clipped surrogate objective* in PPO [20], and get the update rule of $\pi_\theta(\cdot)$:

$$\theta = \theta + \alpha \min \left(\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} R^i(\boldsymbol{\tau}), \right. \\ \left. cilp(\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)}, 1-\epsilon, 1+\epsilon) R^i(\boldsymbol{\tau}) \right) \nabla \log \pi_\theta(a_t|s_t) \tag{7}$$

where α is the learning rate, $p_\theta(a_t|s_t)$ and $p_{\theta'}(a_t|s_t)$ are the sample probabilities of a_t given s_t under the current parameters θ and the old parameters $\theta^{'}$, respectively. $R^i(\boldsymbol{\tau})$ could be the $RecNum$ within a period of time after the recommender system is poisoned by the i-th attack trajectory set $\boldsymbol{\tau}$ in a training batch, where $\boldsymbol{\tau}$ consists of N attackers' trajectories. ϵ $(= 0.1)$ is the hyperparameter used to clip the probability ratio defined in PPO, and $clip(x, a, b)$ limits x between $[a, b]$.

Note that, $R(\boldsymbol{\tau})$ is usually a large discrete number, leading to the difficulty of convergency in a reinforcement learning algorithm. Here, we normalize $\boldsymbol{R}(\boldsymbol{\tau}) = [R^i(\boldsymbol{\tau})]_{0 \le i \le B-1}$ in a sampled batch with Normal (Gauss) Distribution:

$$\hat{R}^i(\boldsymbol{\tau}) = \frac{R^i(\boldsymbol{\tau}) - \mu(\boldsymbol{R}(\boldsymbol{\tau}))}{\sigma(\boldsymbol{R}(\boldsymbol{\tau}))} \tag{8}$$

where $\mu(\boldsymbol{R}(\boldsymbol{\tau}))$ is the mean of $\boldsymbol{R}(\boldsymbol{\tau})$, $\sigma^2(\boldsymbol{R}(\boldsymbol{\tau}))$ is the variance of $\boldsymbol{R}(\boldsymbol{\tau})$, and B is the batch size. $\hat{\boldsymbol{R}}(\boldsymbol{\tau}) = [\hat{R}^i(\boldsymbol{\tau})]_{0 \le i \le B-1}$ is the normalized rewards.

The complete training process is described in Algorithm 1. We also give a kind of possible attack scenario that might happen in a real online recommender system in function $DataPoisoning(D^p)$.

E. Optimization Through Biased Complete Binary Tree

In previous sections, we have fully introduced the framework of PoisonRec. Benefitting from the model-free reinforcement learning architecture, PoisonRec can be used to adaptively attack various recommender systems.

However, it suffers the low convergency problem when directly applied to learn attack strategies on a large action space ($|A| = |I \cup I_t|$). Intuitively, since we focus on the item promotion attack in this work, some actions on the target items I_t are always needed. If we directly sample items from A $(= I \cup I_t)$ as shown in Equation 6, the probability of sampling the target items at each step is only $\frac{|I_t|}{|I|+|I_t|}$ ($|I_t| \ll |I|$), which is very small, and brings many difficulties to obtain effective attack strategies. Moreover, for a T-length attack trajectory, there are total $|A|^T$ possibilities in the search space [18], [20]. $|A|^T$ is a huge number, and it is difficult for PoisonRec to converge under such a kind of setting.

In this part, we propose an important optimization technique to solve the issues discussed above. We reformulate the action space in PoisonRec through a Biased Complete Binary Tree (BCBT). There are mainly two improvements:

- Priori Knowledge. It could be seen as a priori knowledge that it is necessary to sample target items in the attack strategies. We propose a two-stage sampling process, in which PoisonRec first decides which item set (I or I_t) to click, and then chooses one item from the selected set. It can be seen that we built a two-layer tree for sampling items, so that we obtain a biased sampling probability distribution on items in I and I_t. The probability of sampling target items could be increased to about 0.5 ($\gg \frac{|I_t|}{|I|+|I_t|}$) at the beginning of the training.
- Hierarchical Structure. We argue that a hierarchical action space [34] is beneficial for PoisonRec. It not only largely reduces the sampling time complexity on the action space (Section III-E2), but also effectively speeds up the model convergency (Section III-E3).

Algorithm 2 The Sampling Process on BCBT

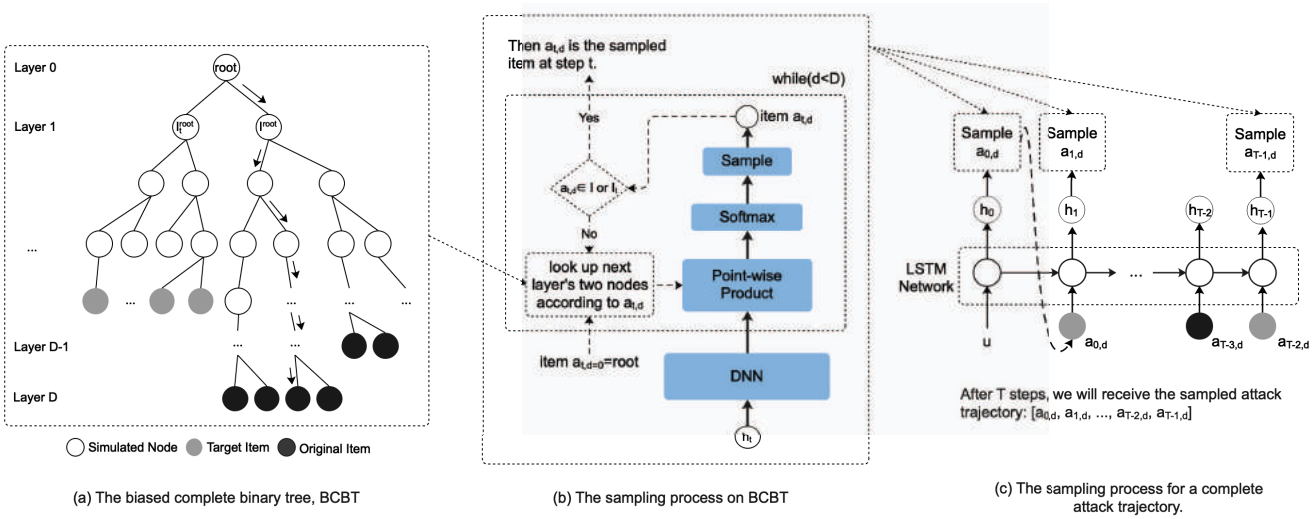
Input: Step t, the hidden variable h_t, the built BCBT.
output: The sampled leaf node (real item) on BCBT.
$d \leftarrow 0$ and $a_{t,d} \leftarrow root$.
while $a_{t,d}$ is not a leaf node ($a_{t,d} \notin I$ *or* I_t) **do**
    Look up next layer's two nodes on BCBT according to $a_{t,d}$.
    Compute point-wise product between $\mathcal{D}(h_t)$ and the two nodes' features, and get $[o_1, o_2]$.
    Sample the next node: $o \sim$softmax$([o_1, o_2])$.
    $d \leftarrow d + 1$.
    $a_{t,d} \leftarrow o$.
end while
Return $a_{t,d}$

In the following, we introduce the construction of BCBT, the sampling on BCBT, and the new update rule of PoisonRec with BCBT in sequence.

1) Construction of BCBT: We adopt the complete binary tree for the hierarchical structure. When constructing BCBT, we first build a complete binary tree (Hierarchical Structure) for I_t and I, respectively, and then merge these two trees with a new root node (Priori Knowledge).

More specifically, we introduce the construction of the complete binary tree for I in detail (the same for I_t). A complete binary tree is a binary tree, and each layer except the last has 2^d nodes (d is the depth of the current layer). Furthermore, the nodes at the last layer are required to be left-aligned. When building a complete binary tree, we need to repeatedly insert nodes from left to right in one layer of the tree. If a layer is full, we insert nodes into the next new layer. The building process is stopped when the number of leaf nodes is equal to $|I|$. All non-leaf nodes are simulated nodes that are used to assist the sampling process on the tree (Section III-E2). For leaf nodes, we assign real items in I to them (details will be discussed later). Since only leaf nodes (real items) have their own features, we randomly initialize

Authorized licensed use limited to: Zhejiang Tmall Technology Co.Ltd.. Downloaded on August 26,2020 at 12:14:24 UTC from IEEE Xplore. Restrictions apply.



(a) The biased complete binary tree, BCBT (b) The sampling process on BCBT (c) The sampling process for a complete attack trajectory.

Fig. 3. The generation of a complete attack trajectory in PoisonRec with BCBT.

trainable feature representation for all simulated no-leaf nodes, whose dimension is equal to that of real items. There is an example of BCBT illustrated on the left of Figure 3.

2) Sampling on BCBT: With a built BCBT, we can sample items following the tree structure more efficiently. The original sampling process in Equation 6 is split into a series of actions on BCBT. At each step t, we need to sample a complete path on BCBT from the root node to a leaf node, and the time complexity of sampling an item can be reduced from $\mathcal{O}(|I \cup I_t||e|)$ to about $\mathcal{O}(\log(|I|)|e|)$. We give the complete pseudo-code of sampling an item on BCBT in Algorithm 2, and there is also an example illustrated in the middle of Figure 3.

3) Updating Rule of PoisonRec with BCBT: For a series of nodes sampled from different layers of BCBT at step t: $[a_{t,1}, a_{t,2}, ..., a_{t,d}]$, we need to update all of them, because one original sampling step is split into multiple ones. The update rule given in Equation 7 is redefined as follows:

$$\theta = \theta + \alpha \min \left(\frac{p_\theta(a_{t,d}|s_t)}{p_{\theta'}(a_{t,d}|s_t)} \hat{R}^i(\boldsymbol{\tau}), \right. \\ \left. cilp(\frac{p_\theta(a_{t,d}|s_t)}{p_{\theta'}(a_{t,d}|s_t)}, 1-\epsilon, 1+\epsilon) \hat{R}^i(\boldsymbol{\tau}) \right) \nabla \log \pi_\theta(a_{t,d}|s_t) \tag{9}$$

where $a_{t,d}$ is the sampled node at step t and level d on the built BCBT, and $d \in [1, 2, ...]$.

Then we discuss how BCBT benefits the model convergency. Each sampled path on BCBT consists of several simulated non-leaf nodes and a real item (leaf node), where all the non-leaf nodes could be viewed as the ancestors of the real item. Following Equation 9, if $p_\theta(a_{t,d}|s_t)$ of each $a_{t,d}$, $d \in [1, 2, ...]$, in a sampled path is updated, the sampling probabilities of other real items that have ancestors in this path will also be affected. To ensure the effectiveness, we emphasize that the design of BCBT should be meaningful and reasonable. Here, we make an assumption:

Assumption 1: Items with close popularity have similar attack properties in a recommender system. For a sampled attack trajectory set, replacing the items with similar ones will not obviously affect its attack performance.

When building BCBT with Assumption 1, there will be clear requirements on the trees built for I and I_t. Taking the tree for I as an example, we need to sort the items in I by their popularity, and then assign them to the leaf nodes by order (from left to right). In such a way, items with closer popularity will share more ancestors. After updating a sampled path on BCBT, the sampling probabilities of items that have higher similarity with the real item in this path will be more affected. These impacts among items are usually positive and can help PoisonRec greatly improve the model convergency. The effectiveness of BCBT will be further discussed and proved in detail in our experiments.

The right of Figure 3 illustrates the generation process for a complete attack trajectory. At step t, PoisonRec models $(u, a_{0,d}, a_{1,d}..., a_{t-1,d})$ through an LSTM network to get h_t (Equation 5), and then samples $a_{t,d}$ with the built BCBT.

F. Complexity Analysis

We analyze the time and space complexity of PoisonRec (Algorithm 1) in this part.

The operations in Algorithm 1 mainly include 1) the attack trajectory sampling based on the policy network $\pi_\theta(\cdot)$, 2) the normalization of a batch of rewards, and 3) the optimization of $\pi_\theta(\cdot)$. We use e to represent the feature representation of users and items, and the size of each layer in the LSTM network and the DNN is set to be equal to $|e|$. To sample T-length trajectories with $\pi_\theta(\cdot)$ for N attackers, PoisonRec needs to perform the calculations in Equations 5 and 6. Specifically, $\mathcal{O}(NT(|e|^2 + |e|^2 + |I \cup I_t||e|))$ time complexity is required (only the major parts are remained), where $\mathcal{O}(|e|^2)$, $\mathcal{O}(|e|^2)$ and $\mathcal{O}(|I \cup I_t||e|)$ are for the calculation in the LSTM network [33], the DNN and the multinomial distribution on the action

Authorized licensed use limited to: Zhejiang Tmall Technology Co.Ltd.. Downloaded on August 26,2020 at 12:14:24 UTC from IEEE Xplore. Restrictions apply.

space A $(= I \cup I_t)$, respectively. With BCBT, since the original sampling process in Equation 6 is split into a series of actions as discussed in Section III-E2, we do not need to calculate the multinomial distribution on A. The sampling complexity is reduced to $\mathcal{O}(NT(|e|^2 + |e|^2 + \log(|I|)|e|))$. For the normalization of a batch of rewards, it requires $\mathcal{O}(B)$ time complexity, where B is the batch size. For the optimization of $\pi_\theta(\cdot)$ on a batch of examples, PoisonRec needs to calculate the gradient and update the parameters of the two neural networks and the features of related items. With BCBT, real items are assigned to the leaf nodes on it. When sampling items, only about $\log(|I|)$ nodes on BCBT are involved each time. The time complexity for optimization is $\mathcal{O}(BNT(|e|^2 + |e|^2 + \log(|I|)|e|))$. Considering the outer iterations in Algorithm 1, the overall time complexity could be viewed as $\mathcal{O}(S(M + KB)NT(|e|^2 + \log(|I|)|e|))$, where S is the number of training steps.

As for the space complexity, the features of nodes on BCBT and the parameters in the LSTM network and the DNN are stored in the main memory, while others like the space for storing training examples are simply ignored. The leaf nodes (real items) on BCBT have $\mathcal{O}(|I \cup I_t||e|)$ space complexity, and the simulated non-leaf nodes require additional $\mathcal{O}(|I \cup I_t||e|)$. For the two neural networks, both of them need about $\mathcal{O}(|e|^2)$. The overall space complexity is $\mathcal{O}(|I \cup I_t||e| + |e|^2)$.

IV. Experiment

In this section, we first conduct experiments to show the effectiveness of the optimization, BCBT, for PoisonRec. Then we analyze the learned attack strategies on different recommender systems. Finally, we compare PoisonRec with other existing attack methods.

A. Settings

We conduct experiments to evaluate PoisonRec on different real-world datasets, including Steam[1], MovieLens-1m[2] and Amazon[3].

Datasets. Steam, MovieLens-1m and Amazon are all available public datasets.

- **Steam.** Steam is one of the world's most popular game hubs. There are about 6,000 games and 200,000 interactions between users and games. For each user u, we use $\{b_1, b_2, ..., b_k\}$ to represent his/her historical behaviors, and the recommender system can use the first $(k-1)$ behaviors to predict his/her k-th interaction on items. We filter users to guarantee that each user u has at least 3 behaviors $\{b_1, b_2, ..., b_k\}$ $(k \geq 3)$.
- **MovieLens-1m.** MovieLens is collected and made available from the MovieLens web site. MovieLens-1m contains 1 million ratings from 6000 users on 4000 movies. In our experiments, we follow the same pre-processing strategy as [3], which regards all user-item rating pairs as positive samples. In addition, we also remove users with sparse rating behaviors.
- **Amazon.** It is a public widely-used e-commerce dataset collected from Amazon. We select the sub-datasets related to the categories of **Phone** and **Clothing**. For each user u that has k behaviors, we can use his/her historical behaviors to predict the next click item.

For all datasets derived from Steam, MovieLens and Amazon, the features we use are user id and item id. The learned id embedding [3], [35] are regarded as the feature representation of users and items. For item popularity, we simply estimate it through the statistical results in the datasets. We split each dataset into train set, validation set and test set, in which for a user's k behaviors $\{b_1, b_2, ..., b_k\}$, b_k is in the test set, b_{k-1} is in the validation set, and others are in the train set [6], [28]. The statistics are shown in Table II.

Dataset	# Users	# Items	# Samples
Steam	6,506	5,134	180,721
MovieLens	5,999	3,706	943,317
Phone	27,879	10,429	166,560
Clothing	39,387	23,033	239,290

TABLE II
STATISTICS OF ALL DATASETS.

Recommender Systems. For Candidate Generation, we adopt a random method for evaluation efficiency in our experiments, as ranking all items and selecting hundreds of them as the candidates each time through a candidate generation model are time-consuming. Randomly selecting some candidate items and observing whether target items get relatively high scores among them in Ranker can reasonably reflect how well the target items are promoted. For Ranker, we choose 8 different widely-used representative recommendation algorithms as our testbeds, including ItemPop [5], CoVisitation [12], PMF [35][4], BPR [5][5], NeuMF [3][6], AutoRec [27][7], GRU4Rec [6][8], and NGCF [9][9]. ItemPop is a traditional recommendation algorithm, in which items are recommended based on their popularity. CoVisitation is an item-based CF recommendation algorithm. Others have been described in Section II. All of them are popular in the literature of recommender systems.

For experimental evaluation, we need to specify the recommendation process of the recommender system. For a user u, we first produce a candidate item set, and then rank items in it. More specifically, we use randomly-selected 92 original items in I and the 8 target items in I_t to make up a candidate item set each time. For the recommendation results, we assume that each user only views k $(= 10)$ items, and define the top-k items with the highest estimated preference scores in Ranker as L_u. If target items frequently appear in users' recommendation results, there will be a high $RecNum$.

[1]https://www.kaggle.com/tamber/steam-video-games/
[2]https://grouplens.org/datasets/movielens/
[3]http://jmcauley.ucsd.edu/data/amazon/

[4]https://github.com/fuhailin/Probabilistic-Matrix-Factorization
[5]https://github.com/gamboviol/bpr
[6]https://github.com/hexiangnan/neural_collaborative_filtering
[7]https://github.com/gtshs2/Autorec
[8]https://github.com/hidasib/GRU4Rec
[9]https://github.com/xiangwang1223/neural_graph_collaborative_filtering

Authorized licensed use limited to: Zhejiang Tmall Technology Co.Ltd.. Downloaded on August 26,2020 at 12:14:24 UTC from IEEE Xplore. Restrictions apply.

Attack baselines. We compare PoisonRec with 6 different attack methods. Popular Attack, Random Attack, Middle Attack and PowerItem are heuristic methods, while ConsLOP and AppGrad are learning-based attack methods. Note that, PowerItem and ConsLOP require the system log in their attack background knowledge, and we involve them as our competitors to better illustrate the advantages of PoisonRec.

- **Popular Attack.** Popular Attack refers to injecting fake click behaviors on target and popular items. We select the top $k\%$ (*e.g.*, k=10) of the most popular items from I into a popular item set I_p. In our experiments, we require randomly clicking a target item in I_t and a popular item in I_p alternately.
- **Random Attack.** Similar to Popular Attack, Random Attack means randomly clicking an item in I and a target item in I_t alternatively.
- **Middle Attack.** We further give Middle Attack, in which we randomly select an item from I_t, I_p or $(I \setminus I_p)$ at each step. An important feature of this method is that it may click several target items continuously.
- **Power Item Attack.** Power Item Attack [13] (PowerItem for short) selects influential items first when attacking a recommender system, and we use the *in-degree centrality* [13] for the power item selection, which performs well in the original paper. In our experiments, we require clicking power items and target items alternately in PowerItem.
- **ConsLOP.** ConsLOP is built upon the specific non-personalized recommender system CoVisitation [12]. It models the attack as a *constrained linear optimization problem* (ConsLOP), and can determine: 1) which items the attacker should inject fake co-visitations to, and 2) how many fake co-visitations should be injected to each item. The $N * T$ fake user behaviors in this paper are redefined as $\frac{N*T}{2}$ co-visitations, where a co-visitation means one click behavior on a target item and one following click on a selected original item. ConsLOP is a single item promotion attack method, so we only remain one target item in I_t to perform the attack.
- **AppGrad.** AppGrad [17] initializes a rating matrix $\widetilde{\boldsymbol{M}}$ for N attackers through a generative adversarial network that is trained on the collected user-item rating data. Then $\widetilde{\boldsymbol{M}}$ is iteratively updated by approximately calculating the gradient of $f(\widetilde{\boldsymbol{M}})$ until the model converges, where $f(\widetilde{\boldsymbol{M}})$ is the pre-defined loss function for attack. We set $f(\widetilde{\boldsymbol{M}}) = -RecNum$. In addition, to apply this method in our experiment settings, we make three necessary changes. First, we consider attacking black-box recommender systems with implicit feedback. Thus, we do not generate rating scores, but use discrete behaviors that are sampled with our priori knowledge to initialize $\widetilde{\boldsymbol{M}}$. Second, we use $\widetilde{\boldsymbol{M}}_{ij}$ to represent that user i takes $\widetilde{\boldsymbol{M}}_{ij}$ clicks on item j, and only remain T behaviors for each attacker for fair comparison (the same setting with all competitors). Third, AppGrad does not model the order of behaviors, and we randomly decide it.

Implementation Details. For all the recommendation algorithms and the compared attack methods, without additional statements, we use the default parameters given in their papers. For PoisonRec, we set the size of all layers to 64, and use Adam as the optimizer with a learning rate $\alpha = 2e-3$. When running Algorithm 1, we set $M = B = 32$, $K = 3$. For the number of attackers and the length of each attack trajectory, we set $N = 20$ and $T = 20$ by default.

B. Effectiveness of BCBT

We study the effectiveness of BCBT. Specifically, we compare four different types of designs.

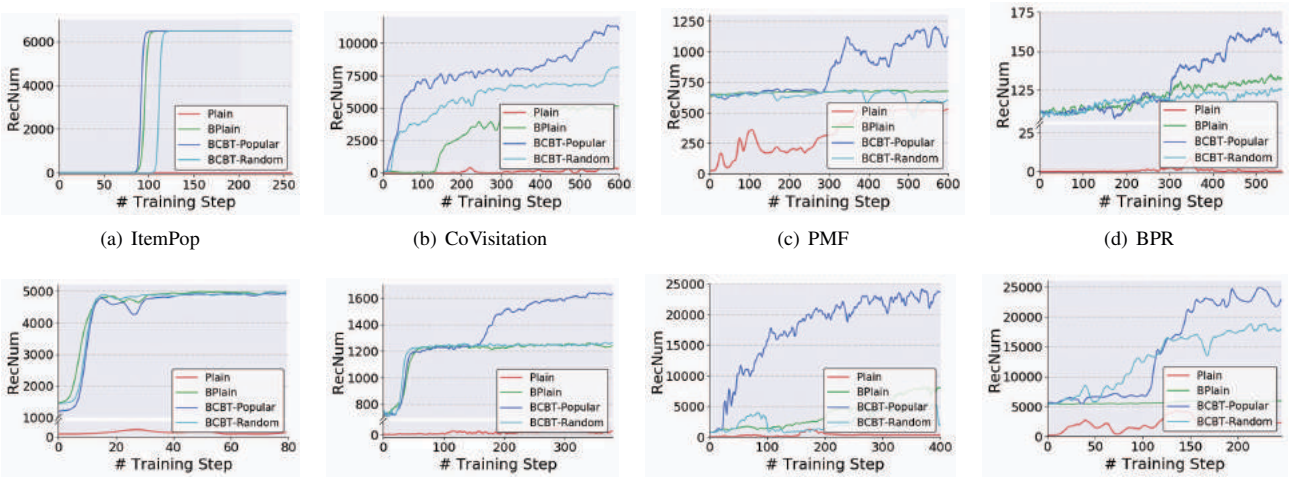
- Plain. It is the basic design of the action space given in Section III-C, with which we need to directly sample items from A as illustrated in Equation 6.
- BPlain. We give BPlain to show the advantages of the priori knowledge, in which we first choose an item set between I_t and I, and then determine which item to click in the selected item set.
- BCBT-Popular. We propose to reformulate the action space through BCBT (with the priori knowledge and the hierarchical structure) for better attack performance in Section III-E. With Assumption 1, we put items with close popularity together, and use BCBT-Popular to represent it.
- BCBT-Random. In BCBT-Random, we randomly assign real items to the leaf nodes on BCBT, and use it to test the rationality of Assumption 1.

At first, compared with the basic Plain, we test the efficiency advantage of BCBT, which greatly reduces the time complexity of Algorithm 1 as discussed in Section III-F. We focus on the time used by PoisonRec and perform experiments on different sizes of item sets, from $3,000$ to $30,000$. As we expect, PoisonRec with BCBT always runs faster than that with Plain, especially on large item sets because of its logarithmic time complexity related to $|I|$. Specifically, on an item set of size $3,000$, PoisonRec with BCBT spends about $1.41s$ to perform a complete training step, while PoisonRec with Plain takes $1.93s$. When the size of the item set reaches $30,000$, the time cost of them increases to $2.33s$ and $15.69s$, respectively. PoisonRec with BCBT runs more than 6 times faster than PoisonRec with Plain, which obviously shows that BCBT has advantages on efficiency.

Then we test the benefits of BCBT-Popular on attacking different recommender systems through equipping PoisonRec with these four different designs on the action space, respectively. Taking the results on Steam as an example, we illustrate how the model converges with the training process in Figure 4.

Advantages of the Priori Knowledge. We say that when performing the item promotion attack in a recommender system, some interactions on the target item set I_t are always needed. The basic design in Plain leads to very a small probability of sampling the items in I_t. Consequently, the generated attack strategies always perform poorly, and there is only a small chance of receiving obvious reward signals.

Authorized licensed use limited to: Zhejiang Tmall Technology Co.Ltd.. Downloaded on August 26,2020 at 12:14:24 UTC from IEEE Xplore. Restrictions apply.



(a) ItemPop (b) CoVisitation (c) PMF (d) BPR

(e) NeuMF (f) AutoRec (g) GRU4Rec (h) NGCF

Fig. 4. The attack performance when attacking different recommender systems on Steam with the training process.

Within a limited number of training steps, PoisonRec with it performs not well. We build a two-layer tree structure in BPlain, with which we can decide which item set to click (I_t or I) at first. Under such a kind of setting, PoisonRec can find more effective attack strategies at the beginning of the training. From Figure 4, we can see that PoisonRec with BPlain always has a better initialization for the training, and can achieve higher $RecNum$.

Advantages of the Hierarchical Structure. We also introduce a hierarchical structure for the action space. The original item sampling process is split into a series of actions on it, and the update rule in Equation 7 is also redefined as Equation 9, which brings potential to speed up the model convergency as discussed in Section III-E3. With Assumption 1, the impacts among items are meaningful and helpful. In our experiments, PoisonRec with BCBT-Popular gets obvious advantages on the model performance and convergency efficiency.

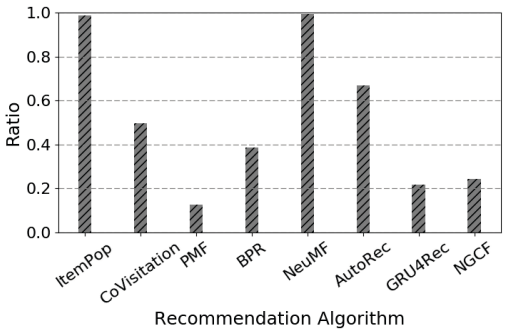


Fig. 5. The ratios of the clicks on I_t to the total clicks for the learned attack strategies on different recommendation algorithms.

We also observe that BPlain gets similar results with BCBT-Popular when attacking ItemPop and NeuMF. We make further investigation, and calculate the ratios of the clicks on I_t to the total clicks for the learned attack strategies of PoisonRec with BCBT-Popular on different recommendation algorithms in Figure 5. For ItemPop and NeuMF, we even receive ratios close to 1.0, which means that clicking items in I_t only is already a good attack strategy on them. Compared with BPlain, BCBT-Popular is further equipped with the hierarchical tree structure, which mainly benefits the exploration on the items in I that have different popularity. If we only need to click the items in I_t that consists of 8 new items, it will be hard for BCBT-Popular to utilize the advantages of the hierarchical tree structure, so BPlain and BCBT-Popular achieve similar performance on ItemPop and NeuMF. When attacking other recommender systems, there are more than 20% clicks on target items most of the time, which experimentally demonstrates the necessity of building a biased probability distribution on items in I_t and I with the priori knowledge.

Besides, to test the rationality of Assumption 1, we give BCBT-Random. In our experiments, PoisonRec with it performs worse than PoisonRec with BCBT-Popular, which clearly illustrates the importance of building a meaningful and reasonable hierarchical tree structure. Gathering items with close popularity (Assumption 1) may not be the optimal solution, but it can bring us considerable advantages. In the following, we adopt PoisonRec with BCBT-Popular by default.

C. Attack Strategies Analysis

In this part, we analyze the learned attack strategies on different recommendation algorithms in detail. Here, we also take the results on Steam as an example. We first visualize all items based on the learned id embeddings in different recommendation algorithms through t-distributed stochastic neighbor embedding (t-SNE) [36][10], and then illustrate the learned attack strategies by circling the clicked items in Figure 6, while ignoring the click order for simplicity. Nodes with different colors are items that have different popularity,

[10]t-SNE is a common technique for dimensionality reduction, and is particularly well suited for the visualization of high-dimensional data.

Authorized licensed use limited to: Zhejiang Tmall Technology Co.Ltd.. Downloaded on August 26,2020 at 12:14:24 UTC from IEEE Xplore. Restrictions apply.

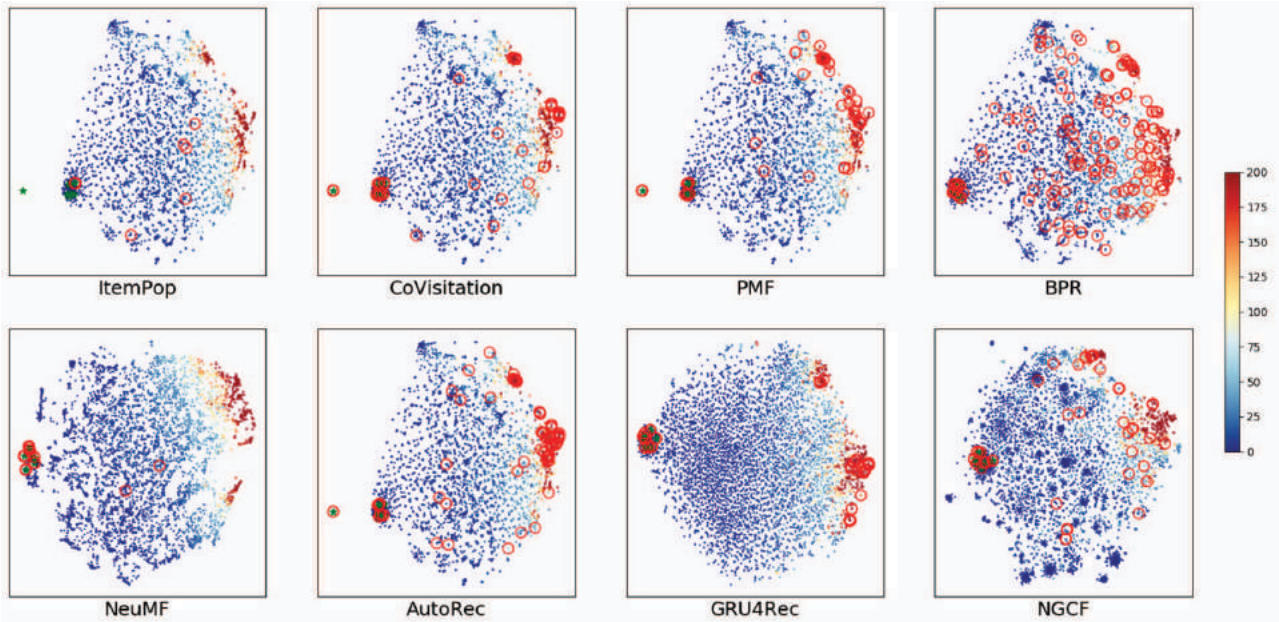


Fig. 6. The visualization of the items and the learned attack strategies on different recommendation algorithms.

and target items are represented by green stars (generally distributed on the left side of each sub-figure). In such a way, we can clearly see the distributions of clicked items in the learned attack strategies on different recommendation algorithms. For ItemPop, CoVisitation and AutoRec, we use the learned item id embedding in PMF for them.

For ItemPop, it is a traditional recommendation algorithm, in which items with higher popularity are more likely to be recommended. We can see that the learned attack strategy of PoisonRec on it involves only one target item. PoisonRec indeed learns to repeatedly click the one selected target item in I_t, so that the target item looks like a popular one, and can receive much more exposure than before. CoVisitation is an item-based CF algorithm. It uses users' sequential behaviors to construct an item-to-item graph, and then calculates item similarities based on this graph. When attacking it, PoisonRec prefers to click target items and original items simultaneously. The learned attack strategies on ItemPop and CoVisitation are obviously different. Putting aside the visualization in Figure 6, when we directly observe the learned attack strategies, we further find that PoisonRec learns to click target items and original items alternately on CoVisitation.

For other recommendation algorithms, PoisonRec also shows the preference for clicking target items and original items at the same time. On PMF, AutoRec and GRU4Rec, PoisonRec prefers to click target items and popular items. However, with limited cost to perform the attack, clicking the most popular items is not always a good solution. More popular items are generally more robust, which means that more fake data may be required to poison them. When attacking BPR and NGCF, clicking some of the slightly popular items receives better performance. As for NeuMF, clicking target items only seems an effective way to poison it.

Overall, there are notable differences among the learned attack strategies on different recommender systems. We need to determine: 1) the items to click, 2) the times to click on each selected item, 3) the order of clicks (*e.g.*, on GRU4rec), etc. **Fixed attack strategies cannot always achieve good results.** We model the sequential attack trajectory as the MDP, and PoisonRec can adaptively learn effective attack strategies on different recommender systems.

D. Comparison with Other Attack Methods.

We conduct the comparison of different attack methods, and Table III reports the results of $RecNum$ on all test datasets and recommender systems.

Random Attack, Popular Attack, Middle Attack and PowerItem are heuristic attack methods. To compare these 4 methods, we use the 8 recommendation algorithms and the 4 datasets to construct 32 testbeds, and then count the number of times that each attack method performs the best. Note that, since there is a high average item frequency ($943317/3706 \approx 254$) in MovieLens, it is difficult for all methods to build a fake popular item with limited cost in this dataset. All methods receive $RecNum = 0$ when attacking ItemPop on MovieLens, so we do not include it in the statistical results.

The specific results are shown in Table IV. We find that we cannot simply say which heuristic method is the best. Different methods receive obvious different attack results on different recommendation algorithms and datasets. Most of the time, Popular Attack and Middle Attack perform better, but they also receive lower $RecNum$ than Random Attack and PowerItem in some cases.

For ConsLOP, it is particularly designed for attacking CoVisitation. With full knowledge about the recommender

Authorized licensed use limited to: Zhejiang Tmall Technology Co.Ltd.. Downloaded on August 26,2020 at 12:14:24 UTC from IEEE Xplore. Restrictions apply.

Dataset	Method	ItemPop	CoVisitation	PMF	BPR	NeuMF	AutoRec	GRU4Rec	NGCF
Steam	Random	7	278	653	114	1,362	667	783	2,203
	Popular	6	1,895	541	106	599	738	1,331	1,093
	Middle	2	530	609	116	449	643	1,347	798
	PowerItem	6	1,794	534	107	588	661	1,401	852
	ConsLOP	8	4,715	633	121	648	683	2,401	1,699
	AppGrad	5,421	135	686	122	2,914	1,256	5,052	8,094
	PoisonRec	**6,496**	**10,917**	**1,211**	**163**	**4,994**	**1,643**	**24,319**	**25,013**
MovieLens	Random	0	492	2,282	2,012	412	11,117	236	6
	Popular	0	1,420	4,237	1,927	10	10,471	1,367	13,015
	Middle	0	120	2,415	2,055	10	10,896	282	12
	PowerItem	0	1,136	4,286	1,972	545	10,691	1,264	11,230
	ConsLOP	0	**2,162**	4,246	1,624	2	11,578	714	11,493
	AppGrad	0	118	3,580	2,044	2,604	12,124	4372	24
	PoisonRec	0	1,552	**7,050**	**2,442**	**2,742**	**12,472**	**18,525**	**21,577**
Phone	Random	2,020	464	10,432	4,282	4,794	2,822	2,826	8,784
	Popular	2,409	2,368	9,939	3,846	1,290	3,885	2,454	8,048
	Middle	4,946	208	9,050	3,565	5,981	2,627	3,699	9,552
	PowerItem	2,358	1,824	10,880	3,779	1,978	3,046	944	7,408
	ConsLOP	2,074	**6,234**	10,787	4,099	1,648	4,694	2,858	9,136
	AppGrad	61,792	131	11,238	4,187	26,800	4,700	4,072	10,852
	PoisonRec	**82,032**	5,683	**12,195**	**4,530**	**28,646**	**4,873**	**8,513**	**12,324**
Clothing	Random	54,820	413	1,848	2,827	4,656	11,270	7,786	7,376
	Popular	53,265	1,262	1,704	2,803	2,424	12,032	11,827	9,468
	Middle	61,156	125	1,699	3,077	4,733	9,768	12,005	5,672
	PowerItem	57,508	686	1,810	2,678	2,525	11,664	7,234	8,592
	ConsLOP	52,921	**3,312**	1,814	2,842	2,294	11,981	15,490	7,524
	AppGrad	180,432	62	3,216	3,816	8,808	13,472	13,424	11,090
	PoisonRec	**218,275**	2,239	**3,363**	**4,656**	**12,592**	**14,245**	**22,013**	**14,391**

TABLE III
RECNUM OF DIFFERENT ATTACK METHODS WHEN ATTACKING DIFFERENT RECOMMENDER SYSTEMS ON 4 DATASETS.

Method	Steam	MovieLens	Phone	Clothing	All
Random	4	1		1	7
Popular	2	3		3	10
Middle	1	1		4	10
PowerItem	1	2		0	4

TABLE IV
THE NUMBER OF TIMES THAT EACH ATTACK METHOD ACHIEVES THE BEST PERFORMANCE AMONG THE FOUR HEURISTIC METHODS.

system, there is no surprise that it can achieve high attack performance on CoVisitation. Specifically, ConsLOP achieves the best results on MovieLens, Phone and Clothing, and performs better than other attack methods except PoisonRec on Steam. For other recommendation algorithms, we use the learned attack strategies of ConsLOP on CoVisitation to attack them. The performance is not as good as before. ConsLOP is particularly optimized on CoVisitation, and the learned attack strategies just achieve limited attack performance on other different recommendation algorithms.

Indeed, ConsLOP is an attack method designed for promoting a single target item [12]. In our experiments, we randomly select one target item from I_t, and then use ConsLOP to calculate the near-optimal solution. However, the constraint of performing attacks on a single target item limits the performance of it. In PoisonRec, we design a relatively comprehensive search space. It is directly optimized on multiple target items, and has opportunities to allocate the limited attack cost ($N*T$ fake user behaviors) on different target items more effectively, leading to better attack performance. PoisonRec also shows the ability to promote multiple target items simultaneously when attacking other recommender systems. For example, on ItemPop, PoisonRec successfully learns to promote 3 and 6 target items at the same time on Phone and Clothing, respectively, and achieves high performance.

For AppGrad, PoisonRec also shows obvious advantages over it. Though AppGrad achieves comparable results with PoisonRec on ItemPop and NeuMF sometimes, PoisonRec performs much better on all other recommender systems. On ItemPop and NeuMF, clicking target items only is already a good attack strategy (Figure 5). We randomly sample discrete behaviors with our priori knowledge to initialize the matrix $\widetilde{M}$ in AppGrad, and nearly half of the clicks are performed on target items at the beginning of the training. When attacking these two recommender systems, AppGrad easily converges to click target items only, and achieves relatively high performance. However, on other recommender systems, the performance of AppGrad is obviously inferior to that of PoisonRec. AppGrad does not incorporate the sequential information on behaviors, which is a serious limitation. When attacking recommender systems that are sensitive to the order of behaviors like CoVisitation and GRU4Rec, it performs not well. In addition, PoisonRec directly explores the attack strategies in the discrete search space, while AppGrad optimizes $\widetilde{M}$ in the continuous numerical space. At each iteration, the elements in $\widetilde{M}$ need to be rounded to the closest integers for obtaining discrete click behaviors [17], which affects AppGrad's performance. More importantly, both PoisonRec and AppGrad need to explore attack strategies in a very large search space, which is related to the number of items in the recommender system. Compared with AppGrad, PoisonRec is further enhanced by BCBT, and

Authorized licensed use limited to: Zhejiang Tmall Technology Co.Ltd.. Downloaded on August 26,2020 at 12:14:24 UTC from IEEE Xplore. Restrictions apply.

can find high-performance attack strategies more efficiently.

V. Conclusions

In this work, we perform a systematic study on attacking different black-box recommender systems. With very limited knowledge about the attacked system, we exploit the reinforcement learning technique and propose a flexible adaptive data poisoning attack framework, PoisonRec, to efficiently find effective attack strategies. In addition, an optimization technique BCBT is devised to strengthen PoisonRec's attack ability. We conduct extensive experiments on 8 representative recommendation algorithms and 4 different public datasets. The results show that BCBT brings obvious advantages on the model performance and convergency efficiency, and PoisonRec can achieve better attack performance in most cases compared with existing attack methods.

Acknowledgment

This work was partially supported by NSFC under Grant No.61832001, and Alibaba-PKU joint Program.

References

[1] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre, "Collaborative recommendation: A robustness analysis," *ACM Transactions on Internet Technology (TOIT)*, vol. 4, no. 4, pp. 344–377, 2004.

[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[3] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[4] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM conference on recommender systems*. ACM, 2016, pp. 191–198.

[5] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 2009, pp. 452–461.

[6] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," *arXiv preprint arXiv:1511.06939*, 2015.

[7] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 353–362.

[8] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 635–644.

[9] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," *arXiv preprint arXiv:1905.08108*, 2019.

[10] S. K. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 393–402.

[11] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness," *ACM Transactions on Internet Technology (TOIT)*, vol. 7, no. 4, p. 23, 2007.

[12] G. Yang, N. Z. Gong, and Y. Cai, "Fake co-visitation injection attacks to recommender systems." in *NDSS*, 2017.

[13] C. E. Seminario and D. C. Wilson, "Attacking item-based recommender systems with power items," in *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 2014, pp. 57–64.

[14] C. E. Seminario and D. C. Wilson, "Assessing impacts of a power user attack on a matrix factorization collaborative recommender system." in *FLAIRS Conference*, 2014.

[15] M. Fang, G. Yang, N. Z. Gong, and J. Liu, "Poisoning attacks to graph-based recommender systems," in *Proceedings of the 34th Annual Computer Security Applications Conference*. ACM, 2018, pp. 381–392.

[16] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *Advances in neural information processing systems*, 2016, pp. 1885–1893.

[17] K. Christakopoulou and A. Banerjee, "Adversarial attacks on an oblivious recommender," in *Proceedings of the 13th ACM Conference on Recommender Systems*. ACM, 2019, pp. 322–330.

[18] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[19] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[21] Y. Wang, Y. Tong, C. Long, P. Xu, K. Xu, and W. Lv, "Adaptive dynamic bipartite graph matching: A reinforcement learning approach," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1478–1489.

[22] Z. Li, J. Song, S. Hu, S. Ruan, L. Zhang, Z. Hu, and J. Gao, "Fair: Fraud aware impression regulation system in large-scale real-time e-commerce search platform," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1898–1903.

[23] K. Lolos, I. Konstantinou, V. Kantere, and N. Koziris, "Adaptive state space partitioning of markov decision processes for elastic resource management," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 2017, pp. 191–194.

[24] Y. Hu, Q. Da, A. Zeng, Y. Yu, and Y. Xu, "Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application," *arXiv preprint arXiv:1803.00710*, 2018.

[25] F. Garcin, B. Faltings, O. Donatsch, A. Alazzawi, C. Bruttin, and A. Huber, "Offline and online evaluation of news recommender systems at swissinfo. ch," in *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 2014, pp. 169–176.

[26] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

[27] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 111–112.

[28] C. Zhou, J. Bai, J. Song, X. Liu, Z. Zhao, X. Chen, and J. Gao, "Atrank: An attention-based user behavior modeling framework for recommendation," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[29] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based Recommendation with Graph Neural Networks," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, ser. AAAI '19, vol. 33, no. 1, 2019, pp. 346–353.

[30] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[31] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[32] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 2017, pp. 66–83.

[33] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.

[34] R. Takanobu, T. Zhuang, M. Huang, J. Feng, H. Tang, and B. Zheng, "Aggregating e-commerce search results from heterogeneous sources via hierarchical reinforcement learning," in *The World Wide Web Conference*. ACM, 2019, pp. 1771–1781.

[35] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *International Conference on Neural Information Processing Systems*, 2007.

[36] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

Authorized licensed use limited to: Zhejiang Tmall Technology Co.Ltd.. Downloaded on August 26,2020 at 12:14:24 UTC from IEEE Xplore. Restrictions apply.

KDD 2020

Prediction and Profiling of Audience Competition for Online Television Series

Peng Zhang[1], Chuanren Liu[2*], Kefeng Ning[1], Wenxiang Zhu[1], Yu Zhang[1]
[1] Alibaba Group, Hangzhou, China, {freeman.zp,kefeng.nkf,wenxiang.zwx,daoji}@alibaba-inc.com
[2] Business Analytics and Statistics, University of Tennessee, Knoxville, USA, cliu89@utk.edu

ABSTRACT

Understanding the target audience for popular television series is valuable for online video platform to manage advertising sales, purchase video licenses, and compete with other video service platforms. Existing studies in this domain generally focus on using data mining and machine learning techniques to recommend television series to individual users or predict the popularity of television series. However, knowing only the popularity of television series can limit our ability to answer more in-depth questions and develop more intelligent applications. In this paper, we develop a data-driven framework to model and predict audience competition patterns for popular online television series. Specifically, we first construct a sequence of dynamic competition networks of television series by mining the detailed viewership records. Then, we design a hybrid modeling framework, i.e., Dynamic Deep Network Factorization (DDNF) for predicting the future competitiveness. Our framework adopts the deep neural network (DNN) and the knowledge-base (KB) embedding to incorporate static features, and integrates the Long-Short Term Memory (LSTM) network to learn dynamic features of the television series. Finally, extensive experiments on real-world data sets validate the effectiveness of our approach compared with state-of-the-art baselines in predicting the audience competition for existing and new television series.

KEYWORDS

Audience Competition, Network Prediction, Network Factorization, Online Television Series.

1 INTRODUCTION

Popular television series play an important role for online video streaming services by attracting new audiences, retaining existing audiences, and improving the competitiveness of the online video platforms. Indeed, the main audience traffic of large-scale online video platforms is often attracted by popular television series, which are important advertising resources. Therefore, understanding the target audience for popular television series is valuable for the online video platform to manage advertising sales, purchase video licences, and compete with other video service platforms [18, 24].

To understand the target audience for popular television series, one can use data mining and machine learning techniques to predict the popularity of television series. Research efforts along this line often focus on the predictive methodology, such as autoregressive models [8], support vector regression [31], and deep neural networks [11]. To further improve the prediction accuracy, research attentions have also been devoted to incorporating problem-specific domain knowledge, such as social networks [16], cross domain information [30], and early view patterns [20, 26].

However, knowing only the popularity of television series can limit our ability to answer more in-depth questions and develop more intelligent applications. For example, for the advertisers using the online video platform, it will be very useful if we can describe typical profiles of the potential audience of an upcoming television series, so that the advertisers can decide whether the series is suitable to promote their brands. For the online video platform, knowing such fine-grained information about the target audience is essential for video recommendation [10], advertisement inventory management [3, 13], and advertisement pricing [7].

In this paper, we propose to model the audience competition patterns for popular online television series to address complex questions infeasible for existing models. We first show that the audience competition patterns provide intuitive and insightful audience profiling for online television series. For instance, as shown in Figure 1, we can inspect the competition patterns associated with a particular show to understand the target audience. Given the historical competition patterns quantified with detail viewership records, we then develop predictive algorithms to forecast future audience competition patterns, so that the online video platform and the advertisers can proactively strategize their business decisions. In particular, with the audience competition patterns, it is also possible to estimate the impact of a new television series on existing television series. Such knowledge can help the online video platform to schedule television programs to maximize the share of viewership and the revenue of advertisement and reach the billion-dollar market. At present, most audience analyses is based on limited information such as feedback from online trailers, which may not fully capture real audience interest. Overall, the prediction of the audience competition among popular television series is of significant interest in the online video industry.

To the best of our knowledge, there has been little research that quantitatively analyzes audience competition for online television series. There are some recent studies on competitiveness analysis in other domains, such as the competitive product detection based on customer behaviors [19], the competitive relationship and competition intensity for shared bicycle apps [23], and the phenomenon of job hopping among competing companies [34]. The fast-evolving business environments in the online video industry call for novel

* Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD'20, August 22–27, 2020, San Diego, California USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

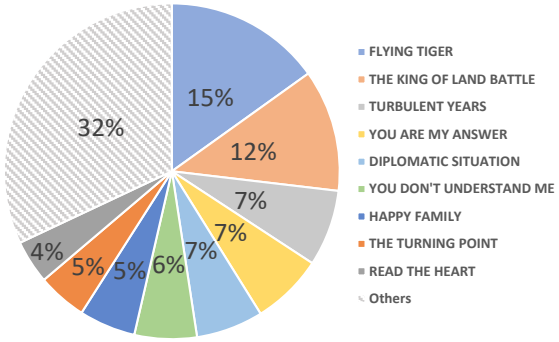


Figure 1: Audience profiling with competition patterns. The visualization is based on the competitiveness of the predator show "ON THE ROAD" where the competitiveness weights are multiplied with the corresponding viewership of the prey shows and then normalized into percentages.

research investigations on the audience competition for online television series with data-driven and intelligent approaches.

To bridge the research gap, we introduce a data-driven framework for effective prediction of future competitiveness among television series. Specifically, we first construct dynamic competition networks where each network node represents a television series and each network edge quantifies the degree of competition between two television series (a predator and prey). The dynamic competition networks are measured by mining the detailed viewership records. Then, we develop a hybrid modeling approach, i.e., Dynamic Deep Network Factorization (DDNF), to predicting the future competition networks, where the predators and the prey are both represented with the time-dependent latent factors. In addition, the latent factors of the predators and the prey are influenced by rich information of the television series. We design deep learning modules in our model to integrate various sources of information about the television series, including static features (e.g. the genre, plot, cast) and dynamic features (e.g. the viewership, ratings, comments at each time slice). For static features, we use knowledge-base (KB) embedding to represent the relations among television series (based on the actors, directors, etc.), and then use the Deep Neural Network (DNN) architecture to learn compact representations for television series; For dynamic features, we adopt the Recurrent Neural Network (RNN) architecture with Long-Short Term Memory (LSTM) to extract the most relevant dynamic information for the prediction of audience competition. By integrating dynamic competition networks with static features and dynamic features, our model can predict future competitions for not only existing television series but also the new ones (as new nodes) in the dynamic competition networks. Finally, the extensive experiments conducted on real-world data sets validate the effectiveness of our approach compared with other state-of-the-art benchmarks.

Our major contributions in this paper can be summarized as follows: 1) We introduce the competitiveness analysis problem for online television series. Our definition and prediction for the audience competition networks of television series enable deeper understanding of the target audience for popular television series. 2) We propose a data-driven hybrid learning approach consisting of KB, DNN, and LSTM to integrate static features and dynamic features and predict future competition networks of television series. 3) Our approach can predict the competitiveness for not only existing television series but also the new ones. New television series often attract more traffic and attention, so our approach can help the online video platform in scheduling television programs and maximizing the revenue of advertisement.

2 RELATED WORK

Competitiveness Analysis. Competitiveness analysis involves the early identification of potential risks and opportunities by gathering and analyzing information in the competitive environment. As part of competitive intelligence (CI), competitiveness analysis is valuable to support executives and managers making strategic decisions [28]. There are some recent studies on competitiveness analysis in various application domains. For instance, Valkanas et al. [32] analyzed the competitiveness between two items based on whether they compete for the attention of the same groups of customers. Ouyang et al. [23] measured competitiveness among mobile bike-sharing apps based on app store data and microblogging data. Zhang et al. [34] used the number of job transitions to represent the competitiveness between two companies and designed an evolving tensor factorization model for predicting the future talent flows. However, to the best of our knowledge, there has been little research that quantitatively analyzes audience competition for online television series with state-of-the-art data-driven techniques.

Matrix/Tensor Factorization. Matrix/tensor factorization is one of the most popular techniques to build recommender systems [1, 6] and has also been successfully used in competitiveness analysis. Specially, Tensor Factorization (TF) can be seen as a generalization of Matrix factorization (MF), where a multi-way data cube representing observations in different contexts (e.g., time, space, and event) is factorized [29]. Popular algorithms for matrix factorization include the Singular Value Decomposition (SVD), Probabilistic Matrix Factorization (PMF) [21], and Non-negative matrix factorization (NMF) [15]. Tensor factorization can be deemed the dynamic factorization of a series of matrices. Taking the application of matrix factorization in recommender systems as an example, since user preferences for items (e.g., products, movies) are drifting over time, it is important to take the temporal dynamics of user preferences into account. Koren [14] proposed the temporal model for the tensor factorization approach by introducing time-variant biases for individual users and items. Zhang et al. [33] considered the evolving user preferences by learning a transition tensor for each user's latent vectors between consecutive time windows.

Another focus in the literature is improving accuracy by incorporating additional information, such as BMFSI [27] and HBMFSI [25]. In BMFSI model, the regression coefficient and side information are augmented in factor matrices such that the rating matrix is estimated by the sum of the collaborative prediction part and regression part against side information. HBMFSI is a method for Bayesian matrix factorization with side information by associating Gaussian-Wishart priors on mean vectors with corresponding side information via regression. Zhang et al. [34] designed a dynamic latent factor based Evolving Tensor Factorization (ETF) model for

predicting the future talent flows. It considers temporal dynamics and also uses some additional information.

Our approach in this paper can capture the temporal dynamics in the time-dependent competition networks and construct the evolving competition factors of television series. This approach allows us to effectively forecast competitiveness for both existing and new television series.

Knowledge Representation Learning. Knowledge bases can learn informative network representation (embeddings) with entities as network nodes and relations as network links [2, 5]. There are mainly two types of knowledge representation models, namely semantic matching models and translational distance models. The semantic matching models such as RESCAL [22] can explain and measure the confidence of relations by calculating the similarity of the entities concerning the relation types in the representation space. The translational distance models use distance-based scoring functions to measure the confidence of the relation triplet including a head entity, a tail entity, and a relation type. In this paper, we use TransE [4], a translational distance model to learn representations of television series based on cast/crew members. By treating each relation triplet as a translation from the head entity to the tail entity, TransE is a simple yet efficient knowledge representation approach achieving state-of-the-art performances in various applications.

Recurrent Neural Networks. Recently, with active developments and widespread applications, deep learning techniques have become popular tools to address complicated learning tasks. In this paper, we use Recurrent Neural Networks (RNN) to capture the temporal dynamics of television series to predict the competition networks. There are many variants of the RNN model, such as time series [9] and popularity prediction [17]. We adopt the Long Short Term Memory (LSTM) networks [12] to learn dynamic embeddings of the television series for its flexibility and superior performances.

3 PROBLEM STATEMENT

We use a large-scale online video platform to set up and evaluate the research on the audience competition problem for online television series (including drama shows and variety shows). Specifically, we extract and predict the competition networks for popular television series. To extract the competition networks from historical data, we measure the *audience competition* between two television series based on the detailed viewership records. Because most of the television series are updated weekly and the audience behavior are reasonably stable week by week, we use week as the time slice to extract and predict the dynamic audience competition patterns. Therefore, the extracted competition networks are represented in a three-dimensional tensor, with values quantifying the competitiveness between two television series during the particular week.

Formally, suppose there are N television series and M audiences that are active on the online video platform. Based on the watch logs (i.e., viewership records), we can construct the dynamic competition networks (i.e., a tensor) R where the t-th matrix slice $R^t \in \mathbb{R}^{N\times N}$ measures the competition relationships of the N television series during the t-th time slice. To define R^t_{ij} with the aggregated behavior of the M audiences for the two television series, we calculate:

$$R^t_{ij} = \frac{1}{M^t_{ij}} \sum_{m=1}^{M^t_{ij}} \frac{\delta^t_{im}}{\delta^t_{im} + \delta^t_{jm}},$$

where δ^t_{im} denotes the frequency of the i-th television series being watched by the m-th user during the t-th time slice. Note that we aggregate the behaviors of M^t_{ij} users who have watched at least one of two shows during the time slice. Intuitively, the aggregated competitiveness R^t_{ij} measures the attraction of the i-th television series (predator) against the j-th television series (prey) during the particular time slice.

Along this line, the problem to predict dynamic competition networks can be formulated as:

Problem 1. *Given a sequence of competition networks $R^1, \cdots, R^T$, which are constructed by the historical watch logs, and the corresponding information of the television series, the goal is to predict the next competition network R^{T+1}.*

Table 1: Summary of notations and definitions.

R^t	The competition network at time slice t;
U^t_i	The temporal latent factor of predator i at time slice t;
V^t_j	The temporal latent factor of prey j at time slice t;
P_i	The static feature embedding of predator i;
Q_j	The static feature embedding of prey j;
G^t_i	The dynamic feature embedding of predator i at time slice t;
H^t_j	The dynamic feature embedding of prey j at time slice t;
X_i	The static features of predator i, as input for $DNN(\cdot, W_p)$;
X_j	The static features of prey j, as input for $DNN(\cdot, W_q)$;
Z^t_i	The dynamic features of predator i at time slice t;
Z^t_j	The dynamic features of prey j at time slice t;
W_p	The parameters of DNN for predator shows;
W_q	The parameters of DNN for prey shows;
W_g	The parameters of $LSTM$ for predator shows;
W_h	The parameters of $LSTM$ for prey shows;

Table 1 lists important notations used throughout this paper. Figure 2 provides an intuitive demonstration of the audience competition patterns and the forecast problem. The network edges between the television series indicate the competition relationship: the arrows indicate the competition direction, and the thickness indicates the degree of the competition. Specifically, suppose the current time is T. At the time slice $T-2$, there were four television series competing with each other. At time slice $T-1$, 'COUNTRY LOVE 11' disappeared because it had been updated which was followed by a significant drop in viewership. In addition, from $T-1$ to T, the competitiveness of 'ON THE ROAD' against 'ALL IS WELL' became stronger. In other words, competitiveness patterns change over time. Our task is to predict the competitiveness of the television series during the time slice $T+1$. The solution should be able to predict for new television series, such as 'ROYAL NIRVANA'.

4 THE PROPOSED MODEL

In this section, we develop the Dynamic Deep Network Factorization (DDNF) framework to model the dynamic competition networks considering temporal dynamics in the historical competition networks as well as the static and dynamic information of the network nodes (i.e., television series). With the learned model, we aim

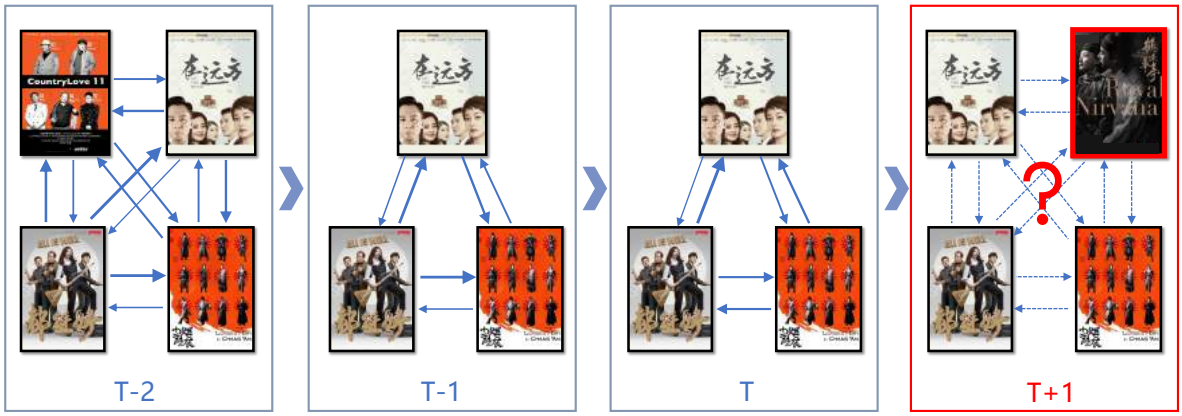


Figure 2: The example of dynamic competition networks.

to forecast future competitive relationships of existing and new television series for large-scale online video streaming services.

Our framework incorporates temporal dynamics, static features, and dynamic features to capture the competitive relationship between a predator show and a prey show. Specifically, the proposed method is a hybrid modeling approach based on the tensor factorization framework with the following components:

- **Temporal Latent Factors:** We infer the latent factors to model the observed competition networks. Since the competition relationships are dynamic in nature, we use temporal latent factors denoted as U_i^t and V_j^t for the predator node i and the prey node j during the time slice t, where the factors have a prior distribution defined with U_i^{t-1} and V_i^{t-1} (i.e., the factors during the time slice $t-1$), respectively.
- **Deep Embedding of Static Features:** In this component, we combine KB and DNN to construct informative embedding vectors, such as P_i and Q_j, for the static features of the shows. We use KB to extract relationship information between shows, and DNN is used to learn all static information.
- **Recurrent Embedding of Dynamic Features:** In this component, we use LSTM to capture information about the dynamic features and produce the dynamic latent vector G_i^t and H_j^t. Specifically, the input of LSTM is a historical sequence of dynamic features, which includes the show status, the number of comments, the number of likes, etc.

Our framework uses these components jointly to embed the dynamic competition networks as well as the static and dynamic features. We optimize the embeddings with the objective to accurately forecast future competition networks under practical requirements on scalability and efficiency of the framework for large-scale platforms. In the joint model, we approximate R_{ij}^t by:

$$\mathbb{E}[R_{ij}^t] = U_i^t V_j^t + P_i Q_j + G_i^t H_j^t, \tag{1}$$

where $\mathbb{E}$ is the operator for expected value and $\Theta_{ij}^t = \{U_i^t, V_j^t, P_i, Q_j, G_i^t, H_j^t\}$ represents the latent factors relevant with R_{ij}^t. Accordingly, the conditional distribution over the observed competitiveness at time slice t is defined as:

$$\Pr(R^t|\Theta,\sigma) = \prod_{i=1}^{N}\prod_{j=1}^{N}[\mathcal{N}(R_{ij}^t|U_i^t V_j^t + P_i Q_j + G_i^t H_j^t, \sigma^2)]^{I_{ij}^t},$$

where $\mathcal{N}(x|\mu,\sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2, and I_{ij}^t is the indicator function that equals to 1 if the value R_{ij}^t is available and equals to 0 otherwise. The latent factors $\Theta = \{U, V, P, Q, G, H\}$ for all time slices together will be optimized with respect to the available values in the historical competition networks.

The rest of this section describes the design details of the model components and finally the learning objective function.

4.1 Temporal Latent Factors

Intuitively, the competition relationships will gradually change over time and latent factors during each time slice will be influenced by the corresponding latent factors in the previous time slices. Note that, other information (e.g., static and dynamic features) influencing the competition networks will be incorporated by other modules in our framework. Therefore, in the temporal latent factor module, we specify that:

$$\Pr(U_i^t) = \mathcal{N}(U_i^t|U_i^{t-1}, \sigma_u^2 I),$$
$$\Pr(V_j^t) = \mathcal{N}(V_j^t|V_j^{t-1}, \sigma_v^2 I),$$

for time slices $t = 2, 3, \cdots$, and all nodes i, j in the competition networks. Since no temporal influence for the first time slice $t = 1$, we simply specify:

$$\Pr(U^1) = \prod_{i=1}^{N}\mathcal{N}(U_i^1|0, \sigma_u^2 I),$$
$$\Pr(V^1) = \prod_{j=1}^{N}\mathcal{N}(V_j^1|0, \sigma_v^2 I).$$

Table 2: The attribute features of television series.

Type	Feature
Numerical	The number of participant's previous views
	The number of episodes
Categorical	The genres of television series
	The operation level of television series
	Whether the show is exclusive on our platform
Text	The plots of television series

The overall probability of the latent factors during all time slices is computed as:

$$\Pr(U) = \prod_{i=1}^{N} \mathcal{N}(U_i^1|0, \sigma_u^2 I) \prod_{t=2}^{T} \prod_{i=1}^{N} \mathcal{N}(U_i^t|U_i^{t-1}, \sigma_u^2 I),$$
$$\Pr(V) = \prod_{j=1}^{N} \mathcal{N}(V_j^1|0, \sigma_v^2 I) \prod_{t=2}^{T} \prod_{j=1}^{N} \mathcal{N}(V_j^t|V_j^{t-1}, \sigma_v^2 I).$$

4.2 Deep Embedding of Static Features

The future competition networks are naturally influenced by a large amount of side information of the predator nodes and the prey nodes. We first consider a set of static features such the genres, plots, actors, directors of the television series. We group the static features into two types, relation features and attribute features, which will be embedded in different ways.

4.2.1 Relation features. Each television series is usually produced by many casts and crew members, who can attract particular target audiences and in turn influence the competitiveness of the show. Therefore, two television series with shared cast/crew members would have related latent factors in modeling the competition networks. We use the knowledge-base representation to embed such relation features in our framework.

Specifically, we first describe common notations. h denotes a head entity, r denotes a relation, and t denotes a tail entity. The bold letters $\mathbf{h}$, $\mathbf{r}$, $\mathbf{t}$ denote the corresponding embedding representations. The knowledge base is then constructed for a set of triplets (h, r, t) to represent the relation between a show h and a member t with the relation type r.

As introduced in related work, TransE [4] represents a relation by a translation vector $\mathbf{r}$ so that the pair of embedded entities in a triplet can be connected by $\mathbf{r}$ with low error. Therefore, to learn the KB embeddings, we minimize the embedding loss of all triples

$$\sum_{\{(h,r,t)\}} \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|,$$

subject to constraints $\|\mathbf{h}\| = \|\mathbf{t}\| = 1$ for all head and tail entities. The learned KB embedding provides a general and compact representation for KB information, so we take the trained $\mathbf{h}$ as the input of the DNN in our model. To ensure that the embeddings can capture comprehensive relationship information, we use all shows of the past few years to build the knowledge base.

4.2.2 Attribute features. We consider three types of attribute features of the television series as shown in Table 2. These features capture content, quality, and popularity of the television series from different perspectives. For instance, television series with different genres often attract different audience groups. Romantic shows are more attractive to young women, while martial arts attract more men. The operational level of television series reflects the investment and promotion cost.

The numerical features will be normalized by subtracting mean and dividing over standard deviation. The categorical features will be used with one-hot encoding. For the text data, we use natural language processing technology to extract pre-trained word embedding from the plots of television series.

4.2.3 Neural network embedding. Finally, the relation features and the attribute features are concatenated into an input vector X for a neural network embedding model, which is composed of several fully connected layers. The output of the embedding model will be used to model the dynamic competition networks. Considering that the predator and prey shows have different embeddings in the competitiveness analysis, their embeddings are separately learned using two neural networks with the same structure but different weights. Formally, the static features are modeled as:

$$\begin{aligned} P_i &= DNN(X_i; W_p), \\ Q_j &= DNN(X_j; W_q), \end{aligned} \tag{2}$$

where X_i (X_j) is the vector of static features of the i-th predator (j-th prey) television series; W_p and W_q are the neural network weights for predator and prey shows, respectively.

4.3 Recurrent Embedding of Dynamic Features

The dynamic features of the television series are important to effectively predict future competition networks. Particularly, the dynamic nature of these features can help our framework to capture various types of dynamic (e.g., seasonal, trend) patterns in the competition relationships. We consider four dynamic features: status of the show, the number of views, the number of likes, and the number of comments. Specifically, the status feature refers to the show status including: pre-release, current, recent post-release, and 30 days post-release. Generally, the competitiveness of the 'current' shows will continue to increase, while the competitiveness of the 'post-release' shows will gradually decline. The number of views can capture popularity trends of the show and thus reflect changes in competitiveness. The number of likes and comments of the television series have similar effects.

Long Short Term Memory (LSTM) networks have been shown effective to capture and characterize the dynamics in longitudinal data. In our framework, we concatenate the dynamic features into a vector (e.g., Z_i^t) as the input of the LSTM component, and use the output as the embedding of dynamic features. Similar to static features, the dynamic feature embeddings of predator and prey nodes are separately learned using two LSTMs with same structure but different modeling weights. Formally, the dynamic feature embeddings are modeled as:

$$\begin{aligned} G_i^t &= LSTM(G_i^{t-1}, Z_i^t; W_g), \\ H_j^t &= LSTM(H_j^{t-1}, Z_j^t; W_h), \end{aligned} \tag{3}$$

where Z_i^t (Z_j^t) is the vector of dynamic features of the i-th predator (j-th prey) television series during the t-th time slice; W_g and W_h are the LSTM weights for predator and prey shows, respectively.

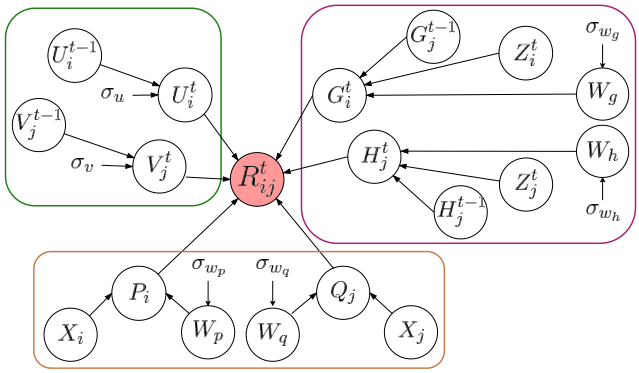


Figure 3: The graphic model of DDNF.

4.4 The Learning Objective Function

The graphic representation of our model is shown in Figure 3, which decomposes the competitiveness R_{ij}^t into temporal latent factors, static embedding vectors, and dynamic embedding vectors. According to the graphical model, we can compute the posterior distribution to optimize the modeling coefficients. First, we represent all features with the $F = \{X, Z\}$ and all weights of the neural networks with $W = \{W_p, W_q, W_g, W_h\}$. Based on Equation 2 and Equation 3, we know that P, Q, G and H are intermediate variables that can be obtained through W and F, so that we only optimize U, V, W. It follows that:

$$\begin{aligned}&\Pr(U^t, V^t | W, R^t, F^t, \sigma, \sigma_u, \sigma_v) \propto \\ &\Pr(R^t | U^t, V^t, W, F^t, \sigma) \Pr(U^t | U^{t-1}, \sigma_u) \Pr(V^t | V^{t-1}, \sigma_v),\end{aligned} \tag{4}$$

for time slices $t > 1$. Then maximizing the log-likelihood of the posterior distribution in Equation 4, which is equivalent to minimizing the sum-of-squared-errors loss function, we have:

$$\begin{aligned}&L(U, V, W) = \\ &\frac{1}{2}\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N} I_{ij}^t (R_{ij}^t - U_i^t V_j^t - P_i Q_j - G_i^t H_j^t)^2 \\ &+ \frac{\lambda_u}{2}\sum_{t=2}^{T}\sum_{i=1}^{N}\left\|U_i^t - U_i^{t-1}\right\|_F^2 + \frac{\lambda_v}{2}\sum_{t=2}^{T}\sum_{j=1}^{N}\left\|V_j^t - V_j^{t-1}\right\|_F^2 \\ &+ \frac{\lambda_u}{2}\sum_{i=1}^{N}\left\|U_i^1\right\|_F^2 + \frac{\lambda_v}{2}\sum_{j=1}^{N}\left\|V_j^1\right\|_F^2 + \frac{\lambda_w}{2}\sum\|W\|_F^2,\end{aligned} \tag{5}$$

where the last term $\frac{\lambda_w}{2}\sum\|W\|_F^2$ is the regularization on the feature embedding modules; P_i, Q_j, G_i^t and H_j^t can be obtained according to Equation 2 and Equation 3, respectively; $\lambda_u = \sigma^2/\sigma_u^2, \lambda_v = \sigma^2/\sigma_v^2, \lambda_w = \sigma^2/\sigma_w^2$, $\sigma_w = \left\{\sigma_{w_p}, \sigma_{w_q}, \sigma_{w_g}, \sigma_{w_g}\right\}$, and $\|.\|_F$ denotes the Frobenius norm. We use Tensorflow to implement our optimization, which updates the modeling coefficients with the back propagation algorithm. Finally, with the learned modeling coefficients, we can forecast the R_{ij}^{T+1} by Equation 1.

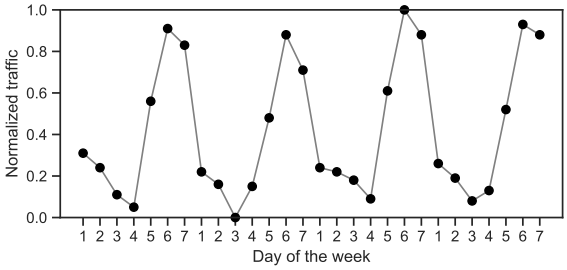


Figure 4: The normalized traffic per day.

Table 3: The statistics of our data sets and knowledge bases.

Data sets	Drama	Variety
# shows	~300	~100
# audiences	~200m	~100m
# competitiveness	~900k	~100k
Knowledge bases (KB)	**Drama KB**	**Variety KB**
# shows	~2k	~1k
# members	~20k	~10k
# relations	~130k	~60k

5 EXPERIMENTAL RESULTS

To validate the effectiveness of our approach, we conduct extensive experiments on real-world data sets. We describe the data and experiment details in Section 5.1, and present the results and analysis in Section 5.2.

5.1 Experiment Setup

5.1.1 Data sets and statistics. We use real-world data sets from a leading online video platform to study the audience competition problem. The data sets contain the detailed viewership records and the characteristics of television series, which covers tens of thousands of television series and hundreds of millions of active users (audience). Based on the viewership records, we construct the dynamic competition networks as introduced in Section 3.

Considering there are only weak competitions between drama and variety shows, we construct two data sets according to the genres of the television series: 1) *Drama Shows*: Popular drama shows are often competitive and valuable for the platform. We chose the most popular drama shows in nearly a year to evaluate our research, and use their viewership records to generate the competition networks. Meanwhile, as shown in Figure 4, the audience traffic has a clear weekly seasonality on our platform. Therefore, we set the length of time slice to be one week. 2) *Variety Shows*: Similarly, we use the viewership records of the top popular variety shows in 2019 to generate another set of competition networks. The detailed statistics of the two data sets are shown in Table 3. For each data set, we use all shows of the same genres (drama or variety) in the past five years to construct the knowledge-base system, where the number of shows, members, and relations are reported in Table 3.

5.1.2 Training and testing. For both drama and variety data sets, we choose the previous T weeks and the $T + 1$ week as the training and testing set, respectively. For example, when $T = 4$, we use the previous four weeks in our data as the training set to predict the

Table 4: The summary of all methods.

Models	Tensor factorization	Temporal dynamics	Additional features
PMF	√	×	×
BTMF	√	√	×
HBMFSI	√	×	√
ETF	√	√	√
DDNF	√	√	√

competition network of the next week. In addition, we conduct ten-fold cross validation for evaluation, and the final results are averaged from ten runs.

5.1.3 Evaluation metrics. In our experiments, we use three metrics for evaluating the performance of competitiveness prediction, namely Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Normalized Deviation(ND). Their definitions are described as follows:

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_i - \hat{y}_i)^2},$$

$$MAE(y, \hat{y}) = \frac{1}{n}\sum_{t=1}^{n}|y_i - \hat{y}_i|,$$

$$ND(y, \hat{y}) = \frac{\frac{1}{n}\sum_{t=1}^{n}|y_i - \hat{y}_i|}{\frac{1}{n}\sum_{t=1}^{n} y_i},$$

where y_i is the actual value, $\hat{y}_i$ is the prediction value, and n is the number of predicted instances. Generally, the lower values of these metrics are, the better prediction performance is obtained. In addition, since RMSE is a part of our loss function (see Equation 5), so it is suitable to investigate whether our model is over-fitting in the training stages.

5.1.4 Comparison methods. To evaluate the performance of our method in competitiveness prediction, we select a number of state-of-the-art methods as baselines. Specifically, we first choose the classic tensor factorization method. Then, we select some methods that can model temporal dynamics for our competitiveness prediction problem. In addition, as our method uses additional information, we choose some methods that can learn additional information. The summary of all methods are shown in Table 4.

- **PMF**[21] models the user preference matrix as a product of two lower-rank user and movie matrices. PMF is a classic tensor factorization method and offers a probabilistic foundation for regularization with Gaussian observation noise.
- **BTMF**[33] extends the conventional matrix factorization method by assuming user preferences change over time, and models the dynamics with a transition matrix for user latent vectors between consecutive time slices.
- **HBMFSI**[25] places Gaussian-Wishart priors on mean vectors and precision matrices of user and item factors, such that the mean of each prior distribution is regressed on additional information.
- **ETF**[34] not only considers temporal dynamics but also uses additional information to learn a dynamic latent factor for predicting the future data.

For methods without temporal dynamics, such as PMF and HBMFSI, we train these models on the training tensors directly. The representation of $T+1$ time slice is obtained by calculating the average representation of previous T time slices. For methods that use additional information, such as HBMFSI and ETFB, we slightly modified them to fit our data. In particular, the ETF considers both temporal dynamics and additional information, which is suitable for our data and serves as a good baseline.

5.1.5 Parameter settings. All modules of our method are implemented using neural networks, we tune hyper-parameters using grid search method. In order to get the best hyper-parameters settings, we use the ten-fold cross validation to evaluate the performance of our proposed method. For the temporal module, we vary the dimension of latent vector D in $\{10, 20, \cdots, 50\}$, and the initial variance of the Gaussian distribution in $\{0.01, 0.02, \cdots, 0.1\}$. For the DNN and LSTM, we vary the hidden layer size in $\{16, 32, \cdots, 256\}$, the number of hidden layers in $\{1, 2, \cdots, 5\}$, the activation function in $\{relu, tanh, sigmoid\}$, the batch size in $\{64, 128, \cdots, 2048\}$, the initial learning rate in $\{0.001, 0.002, \cdots, 0.05\}$, and the dropout rate in $\{0.1, 0.2, \cdots, 0.9\}$. Finally, we set D to 10, initial variance to 0.02, hidden layer size to 128, hidden layers to 2, batch size to 256, learning rate to 0.005, dropout rate to 0.3 and all activation functions are $relu$.

5.2 Results and Analysis

5.2.1 Overall performance. Figure 5 compares the RMSE, MAE and ND values of DDNF with that of other baseline methods[1], and the detailed results are shown in Table 5. Obviously, for both of the two data sets, the classic tensor factorization based method (PMF) has the worst performance, and other methods with temporal dynamics or methods that use additional information have better performance. Then, ETF and DDNF with both temporal dynamics and additional information are better than other methods. In addition, DDNF performs better than ETF because our method can learn more from temporal dynamics and additional information. Finally, on the drama data set, our framework DDNF outperforms all existing methods and achieves 0.087 RMSE, 0.058 MAE and 0.117 ND, that are relative improvements of 7.45%, 10.8% and 10.7% in RMSE, MAE, and ND compared with ETF. Similarly, on the variety data set, DDNF outperforms all existing methods and achieves average 0.095 RMSE, 0.062 MAE and 0.124 ND, that are relative improvements of 10.4%, 12.7% and 12.7% in RMSE, MAE, and ND compared with ETF.

5.2.2 Effectiveness of static and dynamic components. In this subsection, we use controlled experiments to explore the effectiveness of the static and dynamic components for our framework. Table 6 presents the performance for ablation study, in which S represents the static component and D represents the dynamic component. On both two data sets, performance get worse when static or dynamic component is removed, which means that both static and dynamic components improve the performance of our proposed method.

5.2.3 Accuracy on different testing weeks. To further investigate the validity and stability of the methods for predicting the audience

[1]In the overall experiment, we set the training window $T = 12$. We also provide the result for other T in Section 5.2.4.

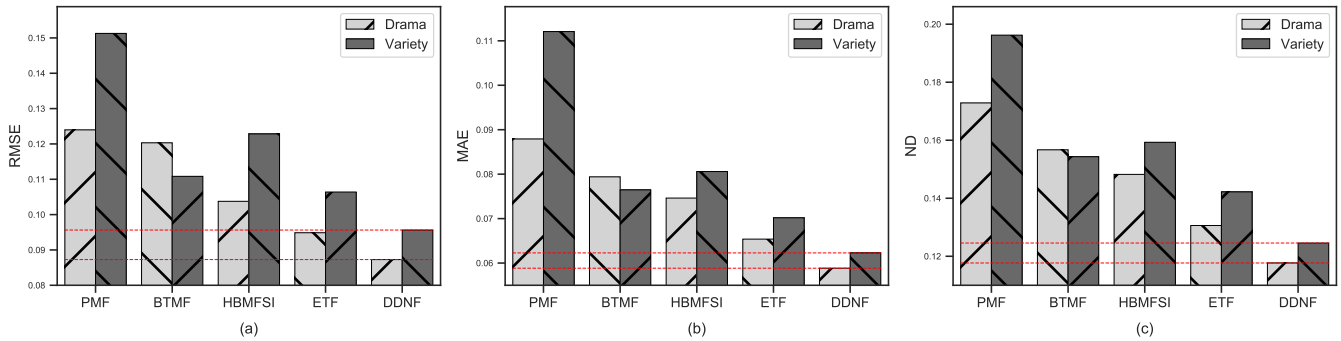


Figure 5: RMSE, MAE and ND of all methods on both data sets.

Table 5: Performance comparisons with ten-fold cross validation, where 'mean' denotes the average, 'std' denotes the standard deviation, and 'ir' denotes the improvement ratio of DDNF over the comparison baselines.

Data sets	Models	RMSE			MAE			ND		
		mean	std	ir	mean	std	ir	mean	std	ir
Drama	PMF	0.124	0.0073	29.8%	0.087	0.0087	33.3%	0.173	0.0089	32.4%
	BTMF	0.121	0.0042	28.1%	0.079	0.0045	26.6%	0.157	0.0051	25.5%
	HBMFSI	0.104	0.0038	16.4%	0.075	0.0029	22.7%	0.148	0.0037	20.9%
	ETF	0.094	0.0015	7.45%	0.065	0.0012	10.8%	0.131	0.0021	10.7%
	DDNF	**0.087**	0.0011		**0.058**	0.0014		**0.117**	0.0019	
Variety	PMF	0.151	0.0085	37.1%	0.112	0.0079	44.6%	0.196	0.0097	36.7%
	BTMF	0.111	0.0025	14.4%	0.076	0.0022	18.4%	0.154	0.0036	19.5%
	HBMFSI	0.123	0.0047	22.8%	0.081	0.0035	23.5%	0.159	0.0055	22.1%
	ETF	0.106	0.0013	10.4%	0.071	0.0009	12.7%	0.142	0.0017	12.7%
	DDNF	**0.095**	0.0016		**0.062**	0.0015		**0.124**	0.0021	

Table 6: Performance of our framework for ablation study ("w/o" is short for "without").

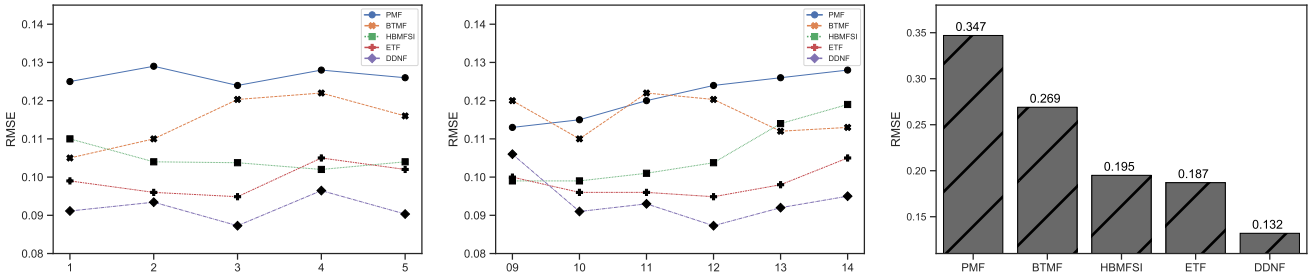
Data sets	Component	RMSE	MAE	ND
Drama	w/o S and D	0.097	0.067	0.134
	w/o D	0.093	0.063	0.125
	w/o S	0.091	0.061	0.123
		0.087	**0.058**	**0.117**
Variety	w/o S and D	0.109	0.073	0.147
	w/o D	0.099	0.065	0.131
	w/o S	0.096	0.063	0.128
		0.095	**0.062**	**0.124**

competition, we further compare all methods in different weeks. Specifically, five consecutive calendar weeks are chosen as the testing weeks, with the training time window unchanged. Figure 6a illustrates the performances in different testing week on the drama data set. Most of the methods are stable, while our DDNF method is better than other methods consistently in all testing weeks.

5.2.4 Effects of training window size. Figure 6b presents the prediction accuracy under different training window size T on the drama data set. Generally, for PMF and HBMFSI, larger training window size leads to worse accuracy, as they cannot model the temporal dynamics in the input data. However, for other methods, as the training window size increases, the prediction accuracy increases first but then decreases. This is because when the window is small, the model just needs more historical information, and when the window size is too large, the historical information is redundant, resulting in a decrease in accuracy. In addition, the optimal window size of DDNF is larger than baseline methods, because DDNF has stronger capabilities for processing historical information and better prediction performance.

5.2.5 Accuracy on new television series. Accurate prediction of the competitiveness for new television series is more valuable yet challenging. To investigate the prediction performance for the new television series, Figure 6c shows the accuracy comparison of all the methods. Our method DDNF achieves $RMSE = 0.132$ on the new television series, which is worse than $RMSE = 0.087$ on all television series. The errors of other methods also increase to different extents. Still, our method DDNF outperforms other methods, providing a 29.4% relative improvement over ETF and a 32.3% relatively improvement over HBMFSI. The reason is that for new television series, there is no temporal dynamics to learn, so the errors increase as expected. But DDNF can still effectively estimate the competitiveness by embedding known features of the new shows.

5.2.6 Audience profiling for new television series. We choose the new show 'ROYAL NIRVAN' as an example for audience profiling. First, we use DDNF to predict the competitiveness of 'ROYAL NIRVANA' against other drama shows in the first week of its launch. Then, the competitiveness weights are multiplied with the respective viewership and then normalized into percentages. Finally, we



(a) Performance in different testing weeks (b) Performance by different training window sizes (c) Performance for new shows

Figure 6: Performance comparison in different scenarios.

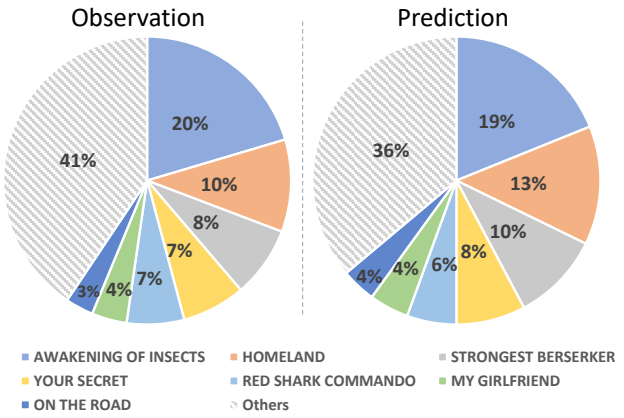


Figure 7: Audience profiling for new show "ROYAL NIRVAN" based on predicted competition networks.

compare the results with the percentages obtained from the observed audience traffics. As shown in Figure 7, the observation (left) is very close to the prediction (right), and both show that the audiences are mainly from 'AWAKENING OF INSECTS', 'HOMELAND', etc. Such audience profiling enabled by our method is valuable to support managerial tasks for the online video streaming services. Specifically, we can decide release time and update strategy for new shows according to the competitiveness patterns. If the new show is too competitive for the existing shows, we can postpone or stagger the release times to ensure the overall popularity of the platform.

6 CONCLUSION

In this paper, we introduce a data-driven approach to predict future audience competition networks for television series. Specifically, we first define the audience competition between shows and construct a sequence of dynamic competition networks by mining the detailed viewership records. Then, we design a hybrid modeling framework for predicting the future competitiveness, which consists of three modules, i.e., temporal latent factors for temporal dynamics, knowledge base (KB) and neural network for static features, and LSTM for embedding dynamic features. Finally, the extensive experiments conducted on real-world data sets validate the effectiveness of our approach compared with state-of-the-art benchmarks and its prediction ability for new shows.

REFERENCES

[1] Charu C Aggarwal et al. *Recommender systems*, volume 1. Springer, 2016.

[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[3] Srinivas Bollapragada and Marc Garbiras. Scheduling commercials on broadcast television. *Operations Research*, 52(3):337–345, 2004.

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[5] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

[6] Fidel Cacheda, Víctor Carneiro, Diego Fernández, and Vreixo Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web (TWEB)*, 5(1):1–33, 2011.

[7] Jiyoung Cha. Predictors of television and online video platform use: A coexistence model of old and new video platforms. *Telematics and Informatics*, 30(4):296–310, 2013.

[8] Biao Chang, Hengshu Zhu, Yong Ge, Enhong Chen, Hui Xiong, and Chang Tan. Predicting the popularity of online serials with autoregressive models. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1339–1348. ACM, 2014.

[9] Sakyasingha Dasgupta and Takayuki Osogami. Nonlinear dynamic boltzmann machines for time-series prediction. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[10] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296. ACM, 2010.

[11] Hongjian Dou, Wayne Xin Zhao, Yuanpei Zhao, Daxiang Dong, Ji-Rong Wen, and Edward Y Chang. Predicting the popularity of online content with knowledge-enhanced neural networks. In *ACM KDD*, 2018.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] Christina ML Kelton and Linda G Schneider Stone. Optimal television schedules in alternative competitive environments. *European Journal of Operational Research*, 104(3):451–473, 1998.

[14] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456, 2009.

[15] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[16] Haitao Li, Xiaoqiang Ma, Feng Wang, Jiangchuan Liu, and Ke Xu. On popularity prediction of videos shared in online social networks. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 169–178. ACM, 2013.

[17] Dongliang Liao, Jin Xu, Gongfu Li, Weijie Huang, Weiqing Liu, and Jing Li. Popularity prediction on online articles with deep fusion of temporal process and content features. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 200–207, 2019.

[18] Carolyn A Lin. Audience fragmentation in a competitive video marketplace. *Journal of Advertising Research*, 34(6):30–39, 1994.

[19] Q. Liu, X. Zeng, C. liu, H. Zhu, E. Chen, H. Xiong, and X. Xie. Mining indecisiveness in customer behaviors. In *2015 IEEE International Conference on Data Mining*, pages 281–290. IEEE, 2015.

[20] Changsha Ma, Zhisheng Yan, and Chang Wen Chen. Larm: A lifetime aware regression model for predicting youtube video popularity. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 467–476. ACM, 2017.

[21] Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.

[22] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pages 809–816, 2011.

[23] Yi Ouyang, Bin Guo, Xinjiang Lu, Qi Han, Tong Guo, and Zhiwen Yu. Competitivebike: Competitive analysis and popularity prediction of bike-sharing apps using multi-source data. *IEEE Transactions on Mobile Computing*, 2018.

[24] Shinjini Pandey, Goutam Dutta, and Harit Joshi. Survey on revenue management in media and broadcasting. *Interfaces*, 47(3):195–213, 2017.

[25] Sunho Park, Yong-Deok Kim, and Seungjin Choi. Hierarchical bayesian matrix factorization with side information. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[26] Henrique Pinto, Jussara M Almeida, and Marcos A Gonçalves. Using early view patterns to predict the popularity of youtube videos. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 365–374. ACM, 2013.

[27] Ian Porteous, Arthur Asuncion, and Max Welling. Bayesian matrix factorization with side information and dirichlet process mixtures. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[28] Michael E Porter. *Competitive strategy: Techniques for analyzing industries and competitors*. Simon and Schuster, 2008.

[29] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 635–644, 2011.

[30] Suman Deb Roy, Tao Mei, Wenjun Zeng, and Shipeng Li. Towards cross-domain learning for social video popularity prediction. *IEEE Transactions on multimedia*, 15(6):1255–1267, 2013.

[31] Tomasz Trzciński and Przemysław Rokita. Predicting popularity of online videos using support vector regression. *IEEE Transactions on Multimedia*, 19(11):2561–2570, 2017.

[32] George Valkanas, Theodoros Lappas, and Dimitrios Gunopulos. Mining competitors from large unstructured datasets. *IEEE Transactions on Knowledge and Data Engineering*, 29(9):1971–1984, 2017.

[33] Chenyi Zhang, Ke Wang, Hongkun Yu, Jianling Sun, and Ee-Peng Lim. Latent factor transition for dynamic collaborative filtering. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 452–460. SIAM, 2014.

[34] Le Zhang, Hengshu Zhu, Tong Xu, Chen Zhu, Chuan Qin, Hui Xiong, and Enhong Chen. Large-scale talent flow forecast with dynamic latent factor model. In *The World Wide Web Conference*, pages 2312–2322. ACM, 2019.

A Dual Heterogeneous Graph Attention Network to Improve Long-Tail Performance for Shop Search in E-Commerce

Xichuan Niu[1], Bofang Li[2], Chenliang Li[3], Rong Xiao[2], Haochuan Sun[2], Hongbo Deng[2], Zhenzhong Chen[1]
1. School of Remote Sensing and Engineering, Wuhan University, China
{niuxichuan,zzchen}@whu.edu.cn
2. Alibaba Group, Hangzhou, China,
{bofang.lbf,xiaorong.xr,haochuan.shc,dhb167148}@alibaba-inc.com
3. School of Cyber Science and Engineering, Wuhan University, China
cllee@whu.edu.cn

ABSTRACT

Shop search has become an increasingly important service provided by Taobao, the China's largest e-commerce platform. By using shop search, a user can easily identify the desired shop that provides a full-scale of relevant items matching his information need. With the tremendous growth of users and shops, shop search faces several unique challenging problems: 1) many shop names do not fully express what they sell, i.e., the semantic gap between user query and shop name; 2) due to the lack of user interactions, it is difficult to deliver a good search result for the long-tail queries and retrieve long-tail shops that are highly relevant to a query.

To address these two key challenges, we resort to graph neural networks (GNNs) which have various successful applications in arbitrarily structured graph data. Specifically, we propose a dual heterogeneous graph attention network (DHGAT) integrated with the two-tower architecture, using the user interaction data from both shop search and product search. At first, we build a heterogeneous graph in the context of shop search, by exploiting both the first-order and second-order proximity from *user search behaviors, user click-through behaviors* and *user purchase records*. Then, DHGAT is devised to attentively adopt heterogeneous and homogeneous neighbors of query and shop to enhance representations of themselves, which can help relieve the long-tail phenomenon. Besides, DHGAT enriches semantics of query text and shop name by compositing the titles of the relevant items to alleviate the semantic gap. Moreover, to enhance the graph representation learning, we augment DHGAT with a regularized neighbor proximity loss (NPL) to explicitly learn the graph topological structure and train whole framework in an end-to-end fashion. Compelling results from both offline evaluation and online A/B tests demonstrate the superiority of DHGAT over state-of-the-art methods, especially for long-tail queries and shops.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
KDD'20, August 2020, San Diego, USA
© 2020 Copyright held by the owner/author(s).
ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

CCS CONCEPTS

• **Information systems → Document filtering**; **Retrieval models and ranking**.

KEYWORDS

Shop Search, Graph Neural Networks, E-commerce

ACM Reference Format:
Xichuan Niu[1], Bofang Li[2], Chenliang Li[3], Rong Xiao[2], Haochuan Sun[2], Hongbo Deng[2], Zhenzhong Chen[1]. 2020. A Dual Heterogeneous Graph Attention Network to Improve Long-Tail Performance for Shop Search in E-Commerce. In *Proceedings of KDD 2020 (KDD'20).* ACM, New York, NY, USA, Article 4, 11 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Search engine is the key functionality in driving rapid growth of e-commerce platforms, such as Taobao[1] and Amazon[2]. Due to the large volume of users and items, it is challenging to deliver satisfied results that meet information needs of customers. Therefore, great efforts have been made to enhance the performance of e-commerce search engine, which plays a significant role in improving user experience and promoting the sales revenue of online merchants. Specifically, a shop in an e-commerce platform often sells items relevant to a specific category, e.g., a shop sells all kinds of sportswear. Also, the promotion campaign is often launched in a shop-wide manner, where a larger discount could be offered with a bigger order. Hence, it becomes more convenient for a user to purchase relevant items in a single shop and enjoy an overlaid promotion. Nowadays, shop search has become an important services for Taobao platform. To our knowledge, no previous effort has been made towards improving the performance of shop search in e-commerce.

Shop search, however, confronts several distinct challenges. Firstly, the user queries and shop names are often very short in nature. There exists semantic gap between user queries and shop names. That is, regularly user queries are more casual while sellers describe their shops more formally. And sometimes a shop name can not reflect what they sell or serve completely, which causes more difficulties to conform user's intent. So simple lexical matching very likely falls short in these cases. Besides, unlike product search,

[1]http://taobao.com/
[2]https://www.amazon.com/

user's shop search behavior and the feedback signals (clicks or purchases) are quite sparser. Note that a user can directly search for interested items rather than shops. Also, they can access the desired shops through previous orders or favorites collection. It is widely known that most of the exposures are popular shops or queries of high frequency. For the long-tail queries and shops, we usually do not have adequate feedback data to train a ranking model, leading to the long-tail bias. These hot shops could domainte the search results and hence degrade the user experience. Moroever, when a long-tail query is issued, it could be difficult to generate a good ranking result.

It has been shown that modelling the e-commerce interactions as structured graph attains significant results. [25] generates node sequence through random walk and the Skip-Gram model [20] is used to train graph embeddings. Then the learned vertice representations are further injected into click through rate (CTR) prediction task. [30] leverages DeepWalk [21] and Node2vec [11] embedding techniques to integrate click-graph features into a unified neural ranking framework also in a two-stage manner. These two-stage approaches are not directly designed for search optimization targets (e.g. CTR, CVR), since the graph based representation learning works as a separate component. This pipelined optimization scheme leads to inferior expressive ability for e-commerce search scenes.

Recently, Graph Neural Networks (GNNs) have achieved magnificent success in various natural language processing and information retrieval tasks [25, 30, 31]. The aim of GNNs is to generalize neural network algorithms to non-Euclidean domains (such as graphs) for robust feature learning. GNNs mainly derive the node representation by aggregating the features appearing in the neighborhood. That is, the underlying graph structure determines how much information can be assimilated in order to facilitate the representation of node itself. Several works have utilized GNNs to deliver promising performance for ranking and recommendation applications [24, 26, 28]. It is expected that modeling homogeneous and heterogeneous relations on the basis of various user interactions and rich semantic information in e-commerce would provide a promising avenue to enhance long-tail performance of shop search.

To this end, in this paper, we propose a heterogeneous graph neural network for shop search in e-commerce. Specifically, we develop a dual heterogeneous graph attention network to boost the representation learning of queries and shops, named DHGAT. The main goal of DHGAT is to enrich the represenations of queries and shops by exploiting various kinds of semantic signals in the e-commerce platform. Here, we build a heterogeneous graph in the context of shop search by exploiting the first-order and second-order proximity from three kinds of user interactions: *user search behaviors, user click-through bahaviors* and *user purchase records*. Both the homogeneous and heterogeneous relations included in the graph can help us derive robust graph-based representations for queries and shops. By introducing a hierarchical attention mechanism, DHGAT can selectively identify the relevant homogeneous and heterogeneous neighbors to update the representation for self node. To further bridging the semantic gap incurred by the mismatch between queries and shops, DHGAT exploits the semantic textual information provided by the relevant items to derive text-level representations for queries and shops. At last, both graph-based representations and text-level representations along with multiple user features are composited and fed into a two-tower network for relevance estimation. For model training, we introduce a regularized neighbor proximity loss as an auxiliary task to enhance the representation learning to retain the graph topological structure, leading to better search performance.

The main contributions of our work can be summarized as follows:

(1) We propose a dual hierarchical graph attention network for e-commerce shop search task. A heterogeneous graph is constructed to perform graph based representation learning for both shops and queries, which includes both first-order and second-order proximity from various user interactions in e-commerce. To the best of our knowledge, this is the first attempt to address the task of shop search in E-Commerece.

(2) The proposed DHGAT selectively leverages heterogeneous and homogeneous relations of queries and shops simultaneously to derive the robust semantic representations for shop search, especially for long-tail performance enhancement. The textual information provided by the relevant items is also exploited to further alleviate the semantic gap.

(3) To explicitly learn the topological structure within graph, we add a regularized neighbor proximity loss to main CTR prediction loss. The whole framework is trained in an end-to-end fashion. Offline evaluation and online A/B test results prove the effectiveness and efficiency of our proposed model.

2 PRELIMINARIES

In this section, we first introduce the problem setup and describe heterogeneous graph. Then we give a detailed description towards the graph construction in context of shop search in e-commerce.

2.1 Problem Setup

Shop Search In E-Comemerce. Given a set $< \mathcal{U}, \mathcal{Q}, \mathcal{S} >$, where $\mathcal{U} = \{u_1, u_2, ..., u_m\}$ denotes the set of m users, $\mathcal{Q} = \{q_1, q_2, ..., q_n\}$ denotes the set of n queries, $\mathcal{S} = \{s_1, s_2, ..., s_p\}$ denotes the set of p shops. When a user u issues a query q, the shop search engine is required to retrieve the most relevant shops as a ranking list where the more relevant shops are ranked higher (i.e., with a higher relevance score).

In context of shop search in e-commerce, there are various kinds of entities (i.e., users, items, shops and queries) and interactions between them. It is natural to model this complex system as a heterogeneous graph, where user, query, shop, item are taken as different types of nodes, and the edges between them reflect the relatedness in terms of the associated interactions.

Heterogeneous Graph. A heterogeneous graph is defined as a network $G = (\mathcal{V}, \mathcal{E})$ consisting of a node set $\mathcal{V}$ and an edge set $\mathcal{E}$. It is also associated with a node type mapping function $\phi(v) : \mathcal{V} \rightarrow \mathcal{O}$ and an edge type mapping function $\psi(e) : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{O}$ and $\mathcal{R}$ denote the set of all node types and the set of all edge types respectively and $|\mathcal{O}| + |\mathcal{R}| > 2$. Here, an edge type indicates the connection between two particular types of nodes.

As mentioned above, for shop search, we can build a heterogeneous graph to include all related types of entities as nodes. For instance, a query node can have homogeneous query neighbors, heterogeneous shop and item neighbors. We plan to take advantage of the information encoded in the heterogeneous graph to facitate the task of shop search and improve user experience. Given a heterogeneous graph $G = (\mathcal{V}, \mathcal{E})$, we devise a novel heterogeneous graph representation learning framework to embed each node as a dense vector representation: $\mathcal{F}_\Theta : \mathcal{V} \rightarrow \mathbb{R}^{|\mathcal{V}| \times d}$, where d is the dimension size and $d \ll |\mathcal{V}|$, and Θ denotes the learnable parameters. Then, for a given shop and query pair, the corresponding representations are fed into a neural network for relevance estimation.

2.2 Graph Construction

Since the efficacy of the graph representation learning highly relies on the underlying graph structure, a key step is to construct a heterogeneous graph that precisely holds the semantic relatedness between different entities in context of shop search. Here, we first collect different kinds of user interactions in the e-commerce platform as the raw data for graph construction. These interactions include *user search behaviors, user click-through behaviors, user purchase records*. Specifically, there are three types of nodes in the constructed graph: Shop (S), Query (Q) and Item (I). By exploiting the first-order and second-order proximity from the raw interaction data, we build three edge types for these three node types correspondingly, which cover both *homogeneous* and *heterogeneous* neighbors for each node.

Homogeneous Neighbors. The homogeneous neighbors refer to the nodes of the same type. This kind of relations often reveal the shared characteristics between the two nodes to some extent. For example, two queries that share the same search intent, or two shops that provide very similar items. For queries and shops, we utilize the following rules of second-order proximity to establish the homogeneous neighbors: 1) users mostly follow a changeless purpose within a search session. Hence, the queries belonging to same session constitute each other's homogeneous neighbors; 2) the queries that lead to the click of same shop often reveal overlap of user's search intent. Hence, the queries matching this criterion are formulated as each other's homogeneous neighbors; 3) the shops that are clicked under the same query are likely to function similarly (e.g., provide the similar items). Hence, these shops are considered as each other's homogeneous neighbors.

Heterogeneous Neighbors. For heterogeneous neighbors, first of all, we make use of click-through data for shop search directly. That is, the shops clicked under a query become the query's heterogeneous shop neighbors and vice versa. Note that this first-order proximity offered by the shop search click-through data could be very sparse since a large proportion of shops are not displayed. The resultant heterogeneous relations could not fully cover the semantic relatedness between shops and queries, leading to inferior long-tail performance.

Therefore, to enrich the heterogeneous query-shop relations extracted through the shop search click-through data, we propose to transfer knowledge from product search, the main search engine for most e-commerce platforms. Due to the large amount of product search click-through data, we can leverage the second-order query-shop relations to augment the corresponding first-order relations. Specifically, for an item provided by a shop and purchased under a query, the corresponding shop and query are considered as each other's heterogeneous neighbor respectively.

Note that exploiting the second-order query-shop relations from the product search click-through data may not alleviate the data sparsity problem to its fullness, because the long-tail phenomenon still exists in the product search scenario. Moreover, the name of a shop is very short due to the restriction made by the platform. It is hard to express all the items sold in the shop by a short shop name. It is expected to alleviate long-tail problem by encoding the semantic information of the constituent items to bridge this semantic gap. Hence, we propose to further exploit the textual semantics covered by the item titles. For each query, we treat the items sold under it as the former's heterogeneous item neighbors. Similarly, the items belong to the same shop are considered as the shop's heterogeneous item neighbors. As will be discussed in the next section, the proposed DHGAT exploits the item neighbors and their associated titles to derive the text-level representations for queries and shops.

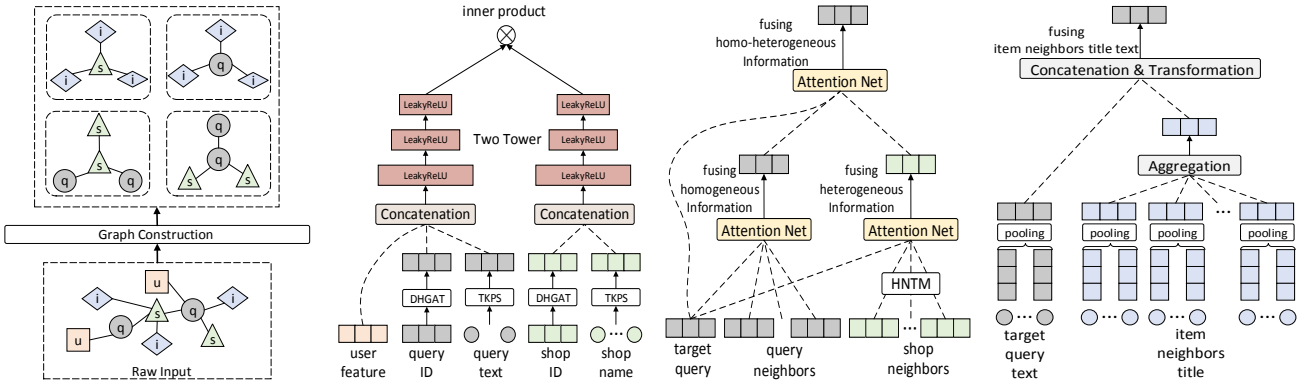
After establishing the edges between nodes, we further assign weight to each edge to indicate the relatedness between the two nodes. For the edge between each shop and item pair, the edge weight is set to be 1. For the other edge types, the edge weight is simply set to be the number of corresponding user interactions. An illustration for graph construction is shown in Figure 1(a), where the homogeneous and heterogeneous neighbors derived from the user interactions are shown as the bottom two after graph construction, while the product search based item neighbors are shown as the upper two.

3 METHOD

In this section, we formally present our proposed model DHGAT to tackle the challenges mentioned in Section 1. For each part, we start from the motivations and intuitions, and then introduce the technical details. Next, we present objective function and some training techniques that can enhance the model performance.

3.1 Overview

The framework architecture of DHGAT is illustrated in Figure 1(c). The basic idea of the proposed model DHGAT is to design a heterogeneous GNN for enriching the representations of queries and shops. As shown in Figure 1(b), our model takes a triple set $< u, q, s >$ as input, where u and q stands for user u issues a query q, s represents a candidate shop. We samples by weight a fix-sized set of neighbors for query and shop respectively, including homogeneous ones and heterogeneous ones. Different types of neighbors are aggregated through a hierarchical attention layer to strengthen ID embedding representations. Dual heterogeneous graph attention networks are operated for queries and shops at the same time. Moreover, borrowing item transaction history from product search (Figure 1(d)), we clarify and enhance semantics of query text and shop name by exploiting their heterogeneous item neighbors' title. Multiple user features are incorporated to yield personalized



(a) Graph Construction (b) Two Tower (c) DHGAT (d) TKPS

Figure 1: The whole pipeline of our model: (a) Graph Construction, (b) Two Tower Architecture, (c) Dual Heterogeneous Graph Attention Network, (d) Transferring Knowledge from Product Search.

search results. At last, user-query representations and shop representations are sent into a tow-tower architecture to get predicted score, according to which a sorted shop list will be displayed.

3.2 Dual Heterogeneous Graph Attention Network

The key challenge of applying GNNs to shop search is how to compose homogeneous and heterogeneous neighbors of queries and shops. Firstly we denote query's homogeneous neighbors as $\mathcal{N}_o(q)$, query's heterogeneous neighbors as $\mathcal{N}_e(q)$, similarly $\mathcal{N}_o(s)$, $\mathcal{N}_e(s)$ are represented as shop's homogeneous neighbors and shop's heterogeneous neighbors respectively. Since some hot queries and shops may have tons of graph neighbors, we begin by sampling $2N$ neighbors for each query and shop, N homogeneous ones and N heterogeneous ones. Sampling weights are calculated and normalized as stated in Section 2.2.

After getting sampled neighbors for queries and shops, a natural idea is to aggregate neighborhood information for reinforcing self node representation. Take query node as a example:

$$\mathbf{h}^t_{\mathcal{N}(q)} = AGGREGATE(\{\mathbf{h}^{t-1}_v, \forall v \in \mathcal{N}_o(q) \cup \mathcal{N}_e(q)\}) \tag{1}$$

$$\mathbf{h}^t_q = COMBINE(\mathbf{h}^{t-1}_q, \mathbf{h}^t_{\mathcal{N}(q)}) \tag{2}$$

where $\mathbf{h}^t_q$ is ID embedding of query node q at layer t, $AGGREGATE$ denotes neighbor aggregation function, such as averaging or max-pooling operation, and $COMBINE$ denotes the function that merges the aggregated neighborhood representation with the node's representation.

However, we have both homogeneous query and heterogeneous shop neighbors for query node, simply adopting one overall aggregation may lose much information. We devise a hierarchical attentive aggregation strategy to discriminate the influence of different types of neighbors. Specifically, we resort to a two-level attention mechanism, where the first level focuses on aggregating neighbor nodes within each type and the second level targets at aggregation over different types. For query node q, the first level attention over homogeneous query neighbors is formulated as:

$$\mathbf{h}^t_{\mathcal{N}_o(q)} = \sum_{i \in \mathcal{N}_o(q)} \alpha^t_{q,i} \mathbf{h}^{t-1}_i \tag{3}$$

where $\alpha^t_{q,i}$ is the normalized attention weight of homogeneous neighbor node i at layer t:

$$\alpha^t_{q,i} = \frac{exp(LeakyReLU(\mathbf{a}^\top_{q,t}[\mathbf{h}^{t-1}_q \| \mathbf{h}^{t-1}_i]))}{\sum_{k \in \mathcal{N}_o(q)} exp(LeakyReLU(\mathbf{a}^\top_{q,t}[\mathbf{h}^{t-1}_q \| \mathbf{h}^{t-1}_k]))} \tag{4}$$

where $\mathbf{a}_{q,t}$ stands for each layer using its own attention parameters, the operator $\|$ denotes concatenation, and we use $LeakyReLU$ as the activation function.

Likewise, for query node q, its heterogeneous shop neighbors are combined at the first level attention. But as query embedding and shop embedding don't share the same vector space, we propose to add a heterogeneous neighbor transformation matrix (HNTM) before the attention layer:

$$\mathbf{h}^t_{\mathcal{N}_e(q)} = \sum_{j \in \mathcal{N}_e(q)} \alpha^t_{q,j} \mathcal{W}_{QS} \mathbf{h}^{t-1}_j \tag{5}$$

where $\mathcal{W}_{QS}$ is the semantic transformation matrix for query's heterogeneous shop neighbors, and $\alpha^t_{q,j}$ is calculated as:

$$\alpha^t_{q,j} = \frac{exp(LeakyReLU(\mathbf{b}^\top_{q,t}[\mathbf{h}^{t-1}_q \| \mathcal{W}_{QS}\mathbf{h}^{t-1}_j]))}{\sum_{k \in \mathcal{N}_e(q)} exp(LeakyReLU(\mathbf{b}^\top_{q,t}[\mathbf{h}^{t-1}_q \| \mathcal{W}_{QS}\mathbf{h}^{t-1}_k]))} \tag{6}$$

where $\mathbf{b}_{q,t}$ is the attention parameter at layer t for measuring weights of heterogeneous neighbors.

Furthermore, after obtaining $\mathbf{h}^t_{\mathcal{N}_o(q)}$ and $\mathbf{h}^t_{\mathcal{N}_e(q)}$, we apply the second attention to fuse both homogeneous and heterogeneous information as followed:

$$\mathbf{h}^t_{\mathcal{N}(q)} = \beta^t_{\mathcal{N}_o(q)} \mathbf{h}^t_{\mathcal{N}_o(q)} + \beta^t_{\mathcal{N}_e(q)} \mathbf{h}^t_{\mathcal{N}_e(q)} \tag{7}$$

$$\beta^t_{\mathcal{N}_*(q)} = \frac{exp(LeakyReLU(\mathbf{c}^\top_{q,t}[\mathbf{h}^{t-1}_q \| \mathbf{h}^t_{\mathcal{N}_*(q)}]))}{\sum_{k=\{\mathcal{N}_o(q),\mathcal{N}_e(q)\}} exp(LeakyReLU(\mathbf{c}^\top_{q,t}[\mathbf{h}^{t-1}_q \| \mathbf{h}^t_k]))} \tag{8}$$

where $\mathbf{c}_{q,t}$ is the parameter of second level attention at layer t, $\beta^t_{\mathcal{N}_*(q)}$ is the attention weights to differentiate two types (i.e., homogeneous and heterogeneous) of neighbors.

$\mathbf{h}^t_{\mathcal{N}(q)}$ is the aggregated neighborhood representation as a result of hierarchical attention. Then we do the *COMBINE* operation regarding to $\mathbf{h}^t_{\mathcal{N}(q)}$ and $\mathbf{h}^{t-1}_q$. First we concatenate them and employ a linear transformation:

$$\mathbf{h}^t_q = LeakyReLU(\mathcal{W}^t_q \cdot [\mathbf{h}^{t-1}_q \| \mathbf{h}^t_{\mathcal{N}(q)}]) \tag{9}$$

where $\mathcal{W}^t_q$ is the learnable transformation parameter. $\mathbf{h}^t_q$ is the query node representation at layer t that has fused both homogeneous and heterogeneous information.

On the other hand, for shop node s, the hierarchical attention mechanism works in an identical way, except the attention parameters are distinct, and the semantic transformation matrix should change from $\mathcal{W}_{QS}$ to $\mathcal{W}_{SQ}$.

3.3 Transferring Knowledge from Product Search

The reason why we need to transfer knowledge from product search (TKPS) is three fold: (1) frequently user queries are ambiguous and shop names may don't reflect what they sell, so we don't aggregate neighbor's texts through hierarchical attention framework since much noise may be carried by them, (2) user behaviors of shop search are fairly sparser while transaction history of product search contains richer information, (3) incorporating items (sold under a query or owned by a shop) title texts can contribute to mitigate the semantic gap between user queries and shop names.

We adopt uniform term embedding scheme described in [9], which represents queries, shops and items with one set of word term embeddings. As shown in Figure 1(d), still take a piece of query q composed of n terms as example, let its terms be $(w_{q,1}, w_{q,2}, ..., w_{q,n})$. At the embedding layer, each term is converted into a low-dimensional dense vector representation and q is expressed as $(\mathbf{t}_{q,1}, \mathbf{t}_{q,2}, ..., \mathbf{t}_{q,n})$. Then we derive the text representation of query q from its term embeddings:

$$\mathbf{e}_q = f(\mathbf{t}_{q,1}, \mathbf{t}_{q,2}, ..., \mathbf{t}_{q,n}) \tag{10}$$

where f means the operation function applied to the terms. In our experiments, we adopt the average function. Similarly, text representations of shop and item are derived as $\mathbf{e}_s, \mathbf{e}_i$ respectively.

Afterwards we non-uniformly sample N item neighbors for query q according to the weights defined in Section 2.2, denoted as $\mathcal{N}_m(q)$. We aggregate the item neighbors text as followed:

$$\mathbf{e}^t_{\mathcal{N}_m(q)} = AGGREGATE(\{\mathbf{e}^{t-1}_i, \forall i \in \mathcal{N}_m(q)\}) \tag{11}$$

where $\mathbf{e}^{t-1}_i$ is item text representation vector at layer $t-1$ and $\mathbf{e}^0_i = \mathbf{e}_i$. Here instead of attentive aggregation, we design the *AGGREGATE* function as mean pooling. Owing to the fact that query itself may hold vague meaning, so query-guided attention may bring in mistaken information.

Then we combine $\mathbf{e}^{t-1}_q$ and $\mathbf{e}^t_{\mathcal{N}_m(q)}$ to get $\mathbf{e}^t_q$:

$$\mathbf{e}^t_q = LeakyReLU(\mathcal{W}^t_{qe} \cdot [\mathbf{e}^{t-1}_q \| \mathbf{e}^t_{\mathcal{N}_m(q)}]) \tag{12}$$

where $\mathcal{W}^t_{qe}$ is the parameter matrix. Correspondingly we can acquire text representation of shop s at layer t: $\mathbf{e}^t_s$, which fuses textual message of shop names and item neighbor titles.

3.4 Incorporating User Features

To better characterize users and retrieve personalized search results, we augment the DHGAT model with additional user features. In our framework, user features are represented in a multi-field multi-hot encoding form. Each field contains multiple discrete categorical features or bag-of-words (BoW) features that are analogous semantically (details of all user features can be found in Appendix A.4). For example, one input instance has three features: *[gender=male, city=beijing, interests=cook&running]*. It is translated into several high-dimensional sparse features via field-aware one-hot encoding:

$$\underbrace{[0,1]}_{gender}\ \underbrace{[1,0,0,0,...,0]}_{city}\ \underbrace{[0,1,...,0,0,1]}_{interests}$$

Then the raw sparse features are fed into the embedding layer to generate low-dimensional real-valued dense vector. The feature embedding is used as the field embedding when the field is single-valued. And if the field is multi-valued, we apply mean pooling to the feature embeddings to get the field embedding. The result of embedding layer $\mathbf{u}$ is a wide concatenated vector of all fields.

3.5 Two-tower Architecture

The two-tower model architecture is shown in Figure 1(b). As can be seen, we first concatenate DHGAT output of query $\mathbf{h}_q$ (fusing homogeneous and heterogeneous neighbor IDs) and TKPS output of query $\mathbf{e}_q$ (fusing item neighbor titles), then combine the concatenation vector and user feature embedding $\mathbf{u}$, which is fed into left tower for more feature fusions. The right tower accepts as input concatenation result of $\mathbf{h}_s$ and $\mathbf{e}_s$ with regard to shop. In our experiments, two-tower are implemented as multiple layer perceptrons (MLPs) whose hidden neurons decrease as the height rises.

3.6 Objective and Model Training

At the last layer L of two-tower, we obtain user-query representation and shop representation $\mathbf{h}^L_{uq}$ and $\mathbf{e}^L_s$. The probability of user u clicking shop s after issuing a query q can be predicted by:

$$\hat{y}_{uqs} = nn(\mathbf{h}^L_{uq}, \mathbf{e}^L_s) \tag{13}$$

For implicit feedback, $nn(\cdot)$ can be a fully-connected layer with a sigmoid activation function. In practice, taking into account efficiency and scalability, we utilize the simplest vector inner product.

The loss function measures the discrepancy between predicted value $\hat{y}_{uqs}$ and ground-truth value y_{uqs}. Here we adopt the most widely used loss function in CTR prediction task, i.e., binary cross entropy:

$$\mathcal{L}_1 = -\sum_{(u,q,s)} y_{uqs} log(\hat{y}_{uqs}) + (1 - y_{uqs}) log(1 - \hat{y}_{uqs}) \quad (14)$$

Besides, using CTR (i.e. click or not click) as the sole supervisory signal to train the model may not capture those information encoded in graph topological structure effectively. We are required to explicitly involve graph neighbor relations in the loss function. Hence, we add a novel neighbor proximity loss to the main CTR prediction loss, which draws closer homogeneous neighbors in the high dimensional space and pushes farther randomly sampled negative nodes:

$$\mathcal{L}_2(\mathbf{h}_i) = -log(\sigma(\mathbf{h}_i^\top \mathbf{h}_j)) - \sum_{k \sim P(i)}^{N_{neg}} log(\sigma(-\mathbf{h}_i^\top \mathbf{h}_k)) \quad (15)$$

where $\mathbf{h}_i$, $\mathbf{h}_j$ and $\mathbf{h}_k$ denotes embeddings of the current node (query or shop), the homogeneous neighbor node in a pair, and the randomly sampled negative node. N_{neg} is the number of sampled negative nodes, we leave the concrete details of negative sampling in Appendix A.1.

Combining CTR prediction cross entropy and neighbor proximity loss, we reach the following complete loss function for DHGAT:

$$\mathcal{L} = \mathcal{L}_1 + \alpha \mathcal{L}_2 + \lambda \|\Theta\|_2^2 \quad (16)$$

where $\|\Theta\|_2^2$ is the L2-regularizer on parameters and embeddings, α and λ are balancing hyper-parameters of neighbor proximity loss and L2 regularization respectively. We use AdaGrad optimizer [8] to minimize the loss.

4 OFFLINE EXPERIMENTS

In this section, we conduct experiments on Taobao real-world dataset to evaluate our proposed model against up-to-date state-of-the-art methods. We then examine hyper-parameter sensitivity and perform the ablation study to analyze effects of different neighbor types.

4.1 Dataset

We evaluate our proposed DHGAT model based on a large-scale real world dataset collected from Taobao shop search platform, which serves over 500 million monthly active users. The dataset covers ten consecutive days of user's click records from the shop search log of Taobao mobile App during November 2019. We hold the first nine days as training set and leave the tenth day for testing, which is refered as *Normal* test set. For the purpose of transferring knowledge of product search, we collect item transaction history of the same period also from Taobao mobile App. To verify the generalization ability of the model, we filter out a subset that only consists of the pairs of query and shop that don't appear in the training set, which we call *Hard* test set. And we further construct a *Long Tail* test set to validate the strength of graph-based model on the long-tail problem, where we select those queries or shops that emerge only once in the training set. The basic statistics of training set, three test sets as well as graph neighbor relations are listed in Appendix A.3.

4.2 Experimental Settings

4.2.1 Baseline Methods. We compare our model with three categories of baseline methods: (1) classical information retrieval and semantic matching methods (NCF [13], DSSM [15], DNN), (2) two-stage graph embedding injected searching methods (GEPS [30], TGE-PS [4]), (3) end to end graph neural networks based methods (PinSage [28], GAT [24], HAN [26]). We also report the performance of two variants of our model (DHGAT, DHGAT$_{NP}$) to test the effectiveness of the components. Hyperparameter settings and implementation details for baselines and our method are introduced in Appendix A.1 and A.2.

- NCF [13]: It is the classical neural network method for top-N recommendation. Here we feed it with the query ID representation and shop ID representation instead of user and item.
- DSSM [15]: This model is proposed for web search semantic matching. We employ the latest BERT [5] to extract text features and measure the similarity of query and shop.
- DNN: Content-enhanced deep learning searching model, which concatenates ID representations with text feature embeddings fed through several MLP layers.
- GEPS [30]: A two-stage method that utilizes DeepWalk [21] and Node2vec [11] graph embedding techniques to enable neural retrieval models to exploit graph structured data for automatic feature extraction.
- TGE-PS [4]: A latest inductive graph embedding approach which uses text-driven graph embedding to generate node embeddings from rich texts.
- PinSage [28]: The state-of-the-art web-scale GNN recommender system with GraphSage [12] as the backbone GNN model. We adopt the optimal implementation released in [12] and report the results based on three neighbor aggregation techniques: Mean (PinSage_M), Max-pooling (PinSage_P), LSTM (PinSage_L).
- GAT [24]: The state-of-the-art attention-based GNN method that leverages attention mechanism to aggregate neighborhood nodes. We report the results of single head and multi-head attention (i.e., K = 8), respectively.
- HAN [26]: The state-of-the-art heterogeneous network embedding model, which takes the importance of nodes and meta-paths into consideration simultaneously.
- DHGAT$_{NP}$: It is a variant of DHGAT, which is non personalized, i.e., removes the additional user features.
- DHGAT: It is our complete model.

4.2.2 Evaluation Protocol. We adopt four commonly used performance metrics for offline evaluation: *Area Under the receiver operating characteristic Curve* (AUC), *Group AUC* (GAUC), *Mean Reciprocal Rank* (MRR), *Hit Ratio at Rank K* (HR@K). Note that GAUC is different from AUC because it can measure the discrepancy between prediction and ground truth under one specific query and combine all queries using frequency as weight. Intuitively, MRR cares about the ranking positions of positive shops while HR@K accounts for whether test shops are present in top-k list. In our experiments, we calculate the HR@1 and HR@5 to evaluate accuracy in the first few retrieval results.

4.3 Overall Performance

We evaluate baselines and our algorithm using three test datasets to explore the model's performance under different scenarios. The overall results of different methods under five indicators are presented in Table 1, from which the following observations can be made:

(1) In all three test datasets, the proposed DHGAT model significantly outperforms all the compared baselines on all five evaluation metrics. Specifically, it offers the average relative performance gain of 4.94%, 4.04%, 2.68%, 4.15%, 1.01% in AUC, GAUC, MRR, HR@1, HR@5 over the best baselines respectively. In particular, we find that the improvements of DHGAT on *Hard* and *Long Tail* are higher than *Normal* compared with baselines. The results indicate that DHGAT is able to successfully aggregate information of homogeneous and heterogeneous neighborhoods and combine item neighbors title from product search. The incorporation of graph neighbor relations helps to strengthen the representation learning of queries and shops and further enhance the prediction performance, especially when the model needs to deal with generalization and long-tail problem.

(2) Pertaining to baseline methods, we observe that DNN and NCF are actually better than DSSM, which demonstrates that classical semantic matching algorithm cannot handle the semantic gap between user queries and shop names. The two-stage graph embedding injected models (GEPS, TGE-PS) performs worse than end to end GNNs based ones. This is probably because these two methods are not directly optimized for specific CTR prediction task, thus limiting the expression ability of generated node embeddings. As for the GNNs based approaches, we find in most cases performance order on different metrics is: HAN > PinSage > GAT. It is not surprising since HAN is capable of exploiting heterogeneous relationships while the other two approaches only model homogeneous information. Note that Mean and LSTM aggregators achieve considerable gain over Max-pooling aggregator on *Hard* and *Long Tail* dataset. And mutli-head GAT(K=8) does perform better than single head version.

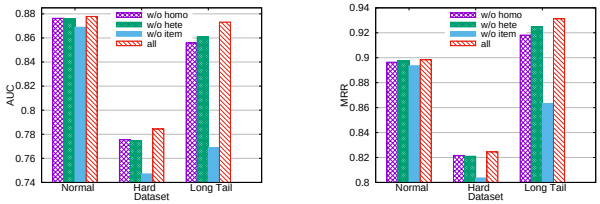
(3) We also show the results of a non personalized variant of DHGAT, i.e, DHGAT$_{NP}$. From Table 1, we can tell that DHGAT$_{NP}$ already exceeds all the baseline methods, which is attributed to its representation power of queries and shops in both identification and text. Moreover, we further augment DHGAT$_{NP}$ with additional user features to fetch personalized search results. The performance gains of DHGAT indicate that incorporation of user features is necessary for better characterizing the user and understanding his intention.

(4) As mentioned above, different evaluation metrics can crystallize different aspects of a model. DHGAT shows strong performance on all five metrics, which reveals that it not only can decide whether a shop meets user requirements, but also is able to arrange a satisfactory ordered list for display.

4.4 In-depth Analysis

To better understand the performance of the proposed model, we conduct a series of in-depth analysis.

Differentiating the importance of neighbor types. To examine which neighbor types play important roles in predicting user's click decisions, we conduct ablation tests by removing different neighbor types. *w/o homo, w/o hete, w/o item* and *all* denote removing query-query edges & shop-shop edges, removing query-shop edges & shop-query edges, removing query-item edges & shop-item edges and the complete model, respectively. We only report the results with respect to AUC and MRR as similar performance patterns are also observed for other metrics.



(a) AUC (b) MRR

Figure 2: Effects of different neighbor types.

As is shown in Figure 2, removing any kind of neighbor will lead to poor performance on both metrics, which means that all three neighbor types contribute to enhancing representation learning. On the other hand, DHGAT assigns highest importance to item neighbor, proving the necessity of transferring knowledge from product search. Homogeneous and heterogeneous neighbors are nearly equally important on *Normal* and *Hard* dataset, nevertheless, homogeneous neighbors appears more critical on *Long Tail* dataset. One plausible explanation is that when query or shop is very unpopular, its possible popular homogeneous neighbors determine more.

Hyperparameter sensitivity. Figure 3 and Figure 4 are illustration of hyperparameter sensitivity of neighbor count and α with respect to AUC and MRR on all three test datasets. Note that the x-axis of Figure 4 is logarithmic.

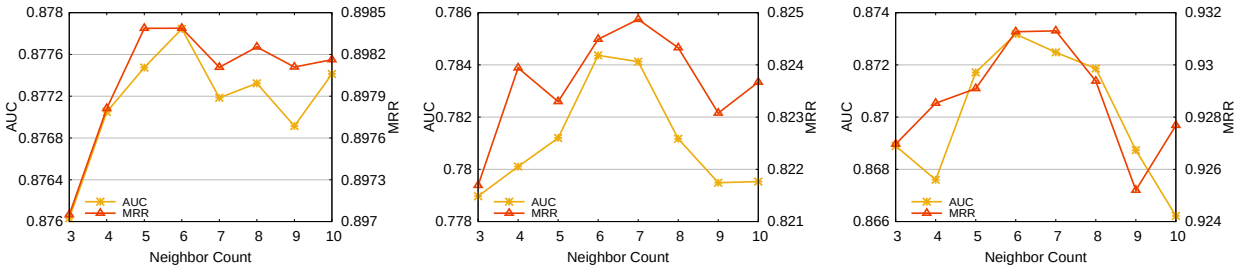
As is depicted in the two figures, we vary the sampling neighbor count from 3 to 10 and α from 0 to 1 while keeping other parameters fixed. The yellow line represents the AUC value, and the red line indicates the MRR metric. From Figure 3, we observe that the best performance is obtained when neighbor count is 6 or 7. Either sampling more or less neighbors will result in performance degradation to some extent. This is reasonable because too few neighbors carry insufficient information while too many would inevitably introduce much noise. Similarly, we see from Figure 4 that free of neighbor proximity loss (i.e., $\alpha = 0$) causes the model to perform inferiorly while pushing too much weight on it (i.e., $\alpha = 1$) also gives rise to worse results. Hence, we set $\alpha = 0.001$ in our experiments.

5 ONLINE EXPERIMENTS

We deploy our proposed model DHGAT on the search matching stage of shop search in Taobao. The graph neighbor relations cover search logs of all training days, and can be updated incrementally each day. Under the framework of A/B tests (i.e., bucket tests), one bucket is selected for baseline and another bucket for our model. For whole bucket, the goal is to increase user activity and maximize the revenue. DHGAT achieves performance gains of **2.506%** on

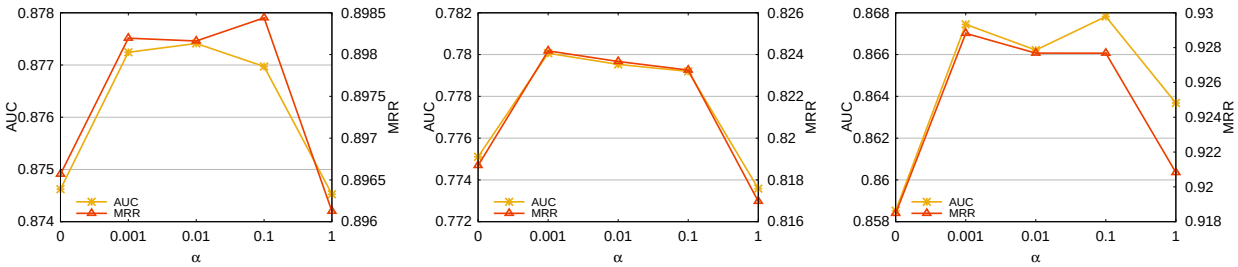
Table 1: Performance comparison for baselines and DHGAT. The * indicates the best performance of the baselines. Best results of all methods are highlighted in boldface. Improvement over the best baseline are shown in the last row.

Method	Normal					Hard					Long Tail				
	AUC	GAUC	MRR	HR@1	HR@5	AUC	GAUC	MRR	HR@1	HR@5	AUC	GAUC	MRR	HR@1	HR@5
NCF	0.8657	0.8517	0.8930	0.7621	0.9289	0.7326	0.7277	0.8040	0.5896*	0.8831	0.7237	0.6581	0.8563	0.8841	0.9926
DSSM	0.8076	0.7871	0.8363	0.6320	0.8938	0.6909	0.6810	0.7524	0.4774	0.8503	0.6968	0.6424	0.8277	0.8596	0.9927
DNN	0.8651	0.8522	0.8920	0.7590	0.9300*	0.7519*	0.7370*	0.8037	0.5847	0.8880*	0.8090*	0.7086*	0.8876*	0.9073*	0.9943*
GEPS	0.8656	0.8521	0.8932	0.7619	0.9274	0.7304	0.7265	0.8036	0.5886	0.8826	0.7187	0.6539	0.8477	0.8777	0.9924
TGE-PS	0.8645	0.8513	0.8918	0.7610	0.9268	0.7282	0.7199	0.8003	0.5855	0.8830	0.7174	0.6542	0.8467	0.8758	0.9921
PinSage_M	0.8670	0.8521	0.8928	0.7618	0.9275	0.7406	0.7267	0.8007	0.5836	0.8768	0.7758	0.6770	0.8690	0.8938	0.9935
PinSage_P	0.8647	0.8510	0.8926	0.7615	0.9265	0.7279	0.7244	0.8029	0.5864	0.8818	0.7348	0.6655	0.8660	0.8897	0.9936
PinSage_L	0.8665	0.8525	0.8929	0.7614	0.9283	0.7393	0.7270	0.8009	0.5840	0.8794	0.7613	0.6723	0.8663	0.8915	0.9934
GAT	0.8295	0.7971	0.8491	0.6484	0.8992	0.7103	0.7017	0.7734	0.5167	0.8623	0.7214	0.6576	0.8539	0.8805	0.9924
GAT(K=8)	0.8578	0.8427	0.8859	0.7453	0.9234	0.7237	0.7213	0.7969	0.5705	0.8792	0.7111	0.6564	0.8489	0.8764	0.9931
HAN	0.8687*	0.8548*	0.8942*	0.7640*	0.9294	0.7453	0.7325	0.8046*	0.5893	0.8842	0.7425	0.6681	0.8611	0.8877	0.9924
$DHGAT_{NP}$	0.8778	0.8640	0.8984	0.7730	0.9346	0.7844	0.7651	0.8245	0.6290	0.9048	0.8732	0.7562	0.9313	0.9427	**0.9961**
DHGAT	**0.8791**	**0.8654**	**0.8990**	**0.7736**	**0.9366**	**0.7883**	**0.7657**	**0.8252**	**0.6320**	**0.9071**	**0.8800**	**0.7581**	**0.9314**	**0.9435**	0.9959
Impv.	1.20%	1.23%	0.54%	1.26%	0.71%	4.84%	3.89%	2.56%	7.20%	2.15%	8.78%	6.99%	4.93%	3.99%	0.17%



(a) Normal Dataset (b) Hard Dataset (c) Long Tail Dataset

Figure 3: Parameter sensitivity of neighbor count.



(a) Normal Dataset (b) Hard Dataset (c) Long Tail Dataset

Figure 4: Parameter sensitivity of α.

number of transactions and **0.342%** on SPV (shop page view) compared to NCF. As for long-tail problem, we filter out those shops that have less than 10 exposures in the training set. On these shops DHGAT improves CTR (click through rate) by **48.1%** and CVR (conversion rate) by **72.8%** on the base of NCF.

6 RELATED WORK

Our work in this paper is closely related to the research of e-commerce search and graph neural networks.

E-commerce Search. E-commerce search is an important problem that has been widely studied both in academic and industrial community. Early studies mainly focus on how to support search for structured product information [6, 19]. Later, e-commerce search studies move to exploit search logs for improving searching performance [7, 22], especially for personalized search [1, 16]. There are also a variety of work on applying learning to rank techniques to e-commerce search for optimization of different retrieval metrics [14, 17]. Recently, research efforts have been attracted to latent embedding based e-commerce search models [2, 23, 32], which jointly learn distributional representations of users and products for better retrieval results.

Graph Neural Networks. Graph neural networks aim to generalize neural network models to graph structured data, such as convolution based GNN [18], attention based GNN [24] and sample and aggregation based GNN [12]. Lately researchers also deployed GNNs in recommender systems or e-commerce search to show consistent performance gains: leveraging GNNs to model user's interactions with items [28], to model session based temporal transitions [27], to model social homophily in user preference [10] and to model attributed multiplex heterogeneous network in e-commerce [3].

7 CONCLUSION AND FUTURE WORK

In this paper, we propose a dual heterogeneous graph attention network with transferring knowledge from product search for shop search in e-commerce. DHGAT is a deep and end to end framework that takes advantage of graph to enhance both ID representation and text representation. Specifically, it employs a hierarchical attention framework to compose homogeneous and heterogeneous neighbors simultaneously. Borrowing item neighbor title text from product search helps relieve the semantic gap between user queries and shop names. Besides, the proposed neighbor proximity loss provides strong additional guidance for learning graph topological structure. DHGAT further incorporates user features to better characterize user intention and retrieve personalized shop search results. We conduct extensive experiments in large scale offline evaluation and online A/B test. The results demonstrate the significant superiority of DHGAT over strong baselines and the effectiveness of the usage of heterogeneous graph.

For future work, we plan to investigate how to efficiently introduce user historical behavior of both shop search and product search. We also will explore more sophisticated relationships existing in user interaction heterogeneous graph and leverage them to enhance e-commerce search performance.

REFERENCES

[1] Qingyao Ai, Daniel N Hill, SVN Vishwanathan, and W Bruce Croft. 2019. A zero attention model for personalized product search. In *CIKM*. 379–388.
[2] Qingyao Ai, Yongfeng Zhang, Keping Bi, Xu Chen, and W Bruce Croft. 2017. Learning a hierarchical embedding model for personalized product search. In *SIGIR*. 645–654.
[3] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation Learning for Attributed Multiplex Heterogeneous Network. *arXiv preprint arXiv:1905.01669* (2019).
[4] Liheng Chen, Yanru Qu, Zhenghui Wang, Lin Qiu, Weinan Zhang, Ken Chen, Shaodian Zhang, and Yong Yu. 2019. Sampled in Pairs and Driven by Text: A New Graph Embedding Framework. In *WWW*. 2644–2651.
[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[6] Huizhong Duan and ChengXiang Zhai. 2015. Mining coordinated intent representation for entity search and recommendation. In *CIKM*. 333–342.
[7] Huizhong Duan, ChengXiang Zhai, Jinxing Cheng, and Abhishek Gattani. 2013. A probabilistic mixture model for mining and analyzing product search log. In *CIKM*. 2179–2188.
[8] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.
[9] Shaohua Fan, Junxiong Zhu, Xiaotian Han, Chuan Shi, Linmei Hu, Biyu Ma, and Yongliang Li. 2019. Metapath-Guided Heterogeneous Graph Neural Network for Intent Recommendation. In *KDD*. 2478–2486.
[10] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *WWW*. 417–426.
[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. 855–864.
[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1024–1034.
[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.
[14] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *KDD*. 368–377.
[15] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*. 2333–2338.
[16] Dietmar Jannach and Malte Ludewig. 2017. Investigating personalized search in e-commerce. In *The Thirtieth International Flairs Conference*.
[17] Shubhra Kanti Karmaker Santu, Parikshit Sondhi, and ChengXiang Zhai. 2017. On application of learning to rank for e-commerce search. In *SIGIR*. 475–484.
[18] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[19] Soon Chong Johnson Lim, Ying Liu, and Wing Bun Lee. 2010. Multi-facet product information search and retrieval using semantically annotated product family ontology. *Information Processing and Management* 46, 4 (2010), 479–493.
[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*. 3111–3119.
[21] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. 701–710.
[22] Ning Su, Jiyin He, Yiqun Liu, Min Zhang, and Shaoping Ma. 2018. User intent, behaviour, and perceived satisfaction in product search. In *WSDM*. 547–555.
[23] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2016. Learning latent vector spaces for product search. In *CIKM*. 165–174.
[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[25] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *KDD*. 839–848.
[26] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. 2022–2032.
[27] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *AAAI*. 346–353.
[28] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *KDD*. 974–983.
[29] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *KDD*. 793–803.
[30] Yuan Zhang, Dong Wang, and Yan Zhang. 2019. Neural IR Meets Graph Embedding: A Ranking Model for Product Search. In *WWW*. 2390–2400.
[31] Jun Zhao, Zhou Zhou, Ziyu Guan, Wei Zhao, Wei Ning, Guang Qiu, and Xiaofei He. 2019. IntentGC: a Scalable Graph Convolution Framework Fusing Heterogeneous Information for Recommendation. In *KDD*. 2347–2357.
[32] Kui Zhao, Yuechuan Li, Zhaoqian Shuai, and Cheng Yang. 2018. Learning and Transferring IDs Representation in E-commerce. In *KDD*. 1031–1039.

A APPENDIX

In the appendix, we firstly give the implementation details of our proposed model and compared methods to help in reproducibility. Then we describe more about the used dataset, including preprocessing procedures, graph neighbor construction and dataset statistics. Finally, we provide details of user features which contribute to characterizing customer intentions and retrieving personalized results.

A.1 Implementation Details of DHGAT

Minibatch Implementation. We adopt mini-batch training technique to calculate the gradients of Eq. (16). As described in Algorithm 1, for each iteration, we consider B triple sets $< u, q, s >$. For each pair $< q, s >$, the homogeneous and heterogeneous neighbors $\mathcal{N}_o(q)$, $\mathcal{N}_e(q)$, $\mathcal{N}_m(q)$, $\mathcal{N}_o(s)$, $\mathcal{N}_e(s)$, $\mathcal{N}_m(s)$ as well as the CTR signal y_{uqs} will be together input into the model for training. After one epoch is finished, the batches partition will be randomly reset to make the model more fully optimized. In each minibatch, since numbers of neighbors are different for queries and shops, we sample by weight a fix-sized set of neighbors of each node instead of using its full set neighbors. Note that if some node has too few neighbors, the sampled set may contain duplicates. In particular, if one node doesn't have any neighbors, we will instead use itself to fill the sampled set. To make computation more efficient, we leverage a negative sampling strategy during batch training when calculating neighbor proximity loss (See below for details).

Algorithm 1: Minibatch Implementation of DHGAT

```
Initialize model parameters Θ randomly;
for iteration in 1, 2, ... do
    Pick a minibatch of B triple sets < u, q, s > and the
      corresponding labels y_uqs;
    Sample by weight (B × 6N) neighbors N_o(q), N_e(q),
      N_m(q), N_o(s), N_e(s), N_m(s);
    Compute loss L_1(Θ) according to Eq. (14);
    Batch-in sample (B × N × 2N_neg) negatives for B pairs
      < q, s >;
    Compute loss L_2(Θ) according to Eq. (15);
    Compute gradients ∇L(Θ) according to Eq. (16);
    Update model: Θ = Θ − ε∇L(Θ);
end
```

Negative Sampling. To keep the computational pattern of each mini-batch fixed and more efficient, we make use of batch negative sampling technique when calculating the neighbor proximity loss. Specifically, take the query for example, suppose the batch size is B, that is, we have B pieces of queries. And we have to sample $(B \times N)$ homogeneous query neighbors within this batch. For each pair (q, q') of a piece of query q and its homogeneous neighbor $q' \in \mathcal{N}_o(q)$, we sample N_{neg} negatives from $((B-1) \times (N+1))$ queries, i.e., excluding query q itself and its homogeneous neighbors. After that we compute the mini-batch neighbor proximity loss according to Eq. (15).

Hyperparameters. The hyperparameter settings for DHGAT are as follows: (1) Embedding layers are randomly initialized with the default setting of TensorFlow xavier_initializer. For the other layers, weight parameters are initialized in the same way and bias parameters are assigned with TensorFlow zero_initializer. (2) The dimensions of identity embeddings and word embeddings are both set to 256. (3) The graph attention layer of our model is set to 1 in light of scalability and efficiency. (4) The mini-batch size is set to 512 and AdaGrad [8] is used as the optimizer, with initial learning rate is set to 0.015. (5) The number of two-tower layers L is set to 3 and hidden neurons of each layer are formulated as 512, 256, 128 from bottom to top, respectively. (6) LeakyReLU is adopted as activation function for graph attention layers and the negative slope coefficient alpha is set to 0.01. (7) The L2 regularization hyperparameter λ is set to 10^{-6}.

Hardware and Software. The proposed models are trained using Alibaba's distributed cloud platform which is based on TensorFlow 1.4 and Python 2.7. For the sake of training efficiency, we employ the batch neighbor sampling while training technology, which is supported by Behemoth, a large graph computation library used internally in Alibaba. In our experiments, the trained parameters are distributed on 40 workers (12 CPU cores for each worker) and updated asynchronously.

A.2 Implementation Details of Baselines

A.2.1 Classical Methods.

- NCF [13]. We re-implemented NCF according to the codes [3] provided by the corresponding author. Instead of inputting user and item embeddings, we feed query and shop identity embeddings into two-tower architecture and keep numbers of neurons of each layer unchanged for fair comparison.
- DSSM [15]. We adopt the latest NLP model BERT [5] as text feature extractor in DSSM setting. Note that due to characteristics of search matching, we compute inner product instead of cosine similarity as in [15]. The dimension of BERT embedding is set to 768. And parameters of BERT are pretrained using corpus in e-commerce.
- DNN. In this model, we feed into left tower concatenation of query identity embedding and BERT text embedding, right tower concatenation of shop identity embedding and BERT text embedding.

A.2.2 Two-stage Methods.

- GEPS [30]. We use DeepWalk [21] for pretraining graph embeddings in this two-stage method, which is implemented using Alibaba internal graph framework AliGraph. The dimension of node embedding is set to 512 and random walk length is set to 3. Note that DeepWalk is trained only with homogeneous relations, i.e., query to query and shop to shop.
- TGE-PS [4]. Different from GEPS, this method leverages pretrained text-driven graph embeddings. We implemented it with Bi-LSTMs over word embeddings and character embeddings as in original paper [4]. Dimension of node embeddings is same as DeepWalk in GEPS and half of it is the number of hidden neurons of Bi-LSTMs.

A.2.3 GNNs Methods.

[3] https://github.com/hexiangnan/neural_collaborative_filtering

Table 2: Basic Statistics of Dataset.

Dataset	# queries	# shops	# items	# users	# interactions
Train	20553774	7731085	19645879	61749902	3316635621
Normal Test	4179683	4750251	-	11730981	343758684
Hard Test	347550	881554	-	516419	2535040
Long Tail Test	97483	143986	-	120023	252947

Table 3: Statistics of Graph Relations.

Relation	# weighted edges	# edges
query2query	7424921084	15716555338
query2shop	99790736	381648180
query2item	83795956	207256487
shop2shop	935879056	2850329170
shop2query	99790736	381648180
shop2item	19645879	207256487

- PinSAGE [28]. We re-implemented this method on the Alibaba distributed cloud platform based on the GraphSAGE [12] codes available on GitHub [4]. The dimensions of node embeddings and two-tower structure are kept same for fair comparison. The search space of number of sampling neighbors is $\{3, 4, 5, 6, 7, 8, 9, 10\}$. Note that this method doesn't incorporate text information.
- GAT [24]. The codes [5] released by the authors could not directly generalize to our dataset as they compute attention over all neighbors. We re-implemented with attention focused only on sampled neighbors for efficiency. The sampling number is searched within same space as PinSAGE.
- HAN [26]. We refered to the author's released codes [6] and re-implemented this method in a distributed way. In our setting, the aspects of semantic attention of queries and shops are both 2.

A.3 Datasets

A.3.1 Preprocessing.

As is mentioned before, when constucting homogeneous neighbors of queries, we need the session segment data. User behavior sequences are segmented into sessions based on time interval to prevent user intention drift. In our experiments, the time interval is set to 15 minutes. The other neighbor relations are constructed from shop search and product transaction logs collected from Taobao mobile App. We sum neighbor relation (i.e., graph edge) counts over all 9 training days to build the complete large graph. Shop IDs and item IDs are designated by Taobao system universally while query IDs are derived through hashing. For words segmentation of query texts, shop names and item titles, we adopt the Alibaba internal tool AliWS.

A.3.2 Statistics.

We report basic statistics of dataset and graph relations as in Table 2 and Table 3. Table 2 shows numbers of queries, shops, items, users and interactions in train set and test set (i.e., *Normal*, *Hard* and *Long Tail*). Note that items were not involved in test set since we cannot make use of transaction records dealed at test day. Table 3 shows numbers of neighbor relationships used during training. Recall that we assign weights to edges based on the co-occurrence times of nodes, in Table 3 we give numbers of weighted edges and non-weighted edges respectively.

A.4 Details of User Features

We incorporate three groups of additional user features, which are described as follows:

- **User profile information.** This group primarily consists of user demographic features, which can help better understand user intentions and characterize user preferences in a fine-grained degree. Concretely, these features are gender, age, city, phone brand, etc. Some features (e.g., gender, city) are categorical features that could be vectorized directly via embedding look-up. And other features (e.g., age) are numerical features which will be discretized according to some thresholds and then embedded.
- **User consumption level.** This group of features mainly depicts the purchasing power and consumption habits of users, with which we can infer user's tastes and retrieve search results with high satisfaction. These features include vip type, vip score, phone price, city house price, etc. Vip level has 7 discrete feature states used in Taobao internally. Other numerical features (e.g., city house price) are collected or predicted from public markets.
- **User activity level.** The features of this group describe Taobao users activity at different granularities. For example, user's daily browsing time at Taobao mobile app, user's weekly login times, user's monthly pay amount, user's yearly luxury deals, etc. Activity level features could be utilized to capture user biases and special interests. Most of these features are continuous and need to be discretized with regard to selected threshold values.

[4] https://github.com/williamleif/GraphSAGE
[5] https://github.com/PetarV-/GAT
[6] https://github.com/Jhy1993/HAN

Applied Data Science Track Paper KDD '20, August 23–27, 2020, Virtual Event, USA

Learning to Generate Personalized Query Auto-Completions via a Multi-View Multi-Task Attentive Approach

Di Yin*
National Key Laboratory for Novel Software Technology,
Nanjing University
Nanjing, China
yindinju@gmail.com

Jiwei Tan, Zhe Zhang
Hongbo Deng
Alibaba Group
Hangzhou, China
{jiwei.tjw,fanxuan.zz,dhb167148}@alibaba-inc.com

Shujian Huang
Jiajun Chen
National Key Laboratory for Novel Software Technology,
Nanjing University
Nanjing, China
{huangsj,chenjj}@nju.edu.cn

ABSTRACT

In this paper, we study the task of Query Auto-Completion (QAC), which is a very significant feature of modern search engines. In real industrial application, there always exist two major problems of QAC - weak personalization and unseen queries. To address these problems, we propose $\mathbf{M^2A}$, a multi-view multi-task attentive framework to learn personalized query auto-completion models. We propose a new Transformer-based hierarchical encoder to model different kinds of sequential behaviors, which can be seen as multiple distinct views of the user's searching history, and then a prefix-to-history attention mechanism is used to select the most relevant information to compose the final intention representation. To learn more informative representations, we propose to incorporate multi-task learning into the model training. Two different kinds of supervisory information provided by query logs are utilized at the same time by jointly training a CTR prediction model and a query generation model.

To bridge the gap between the setting of research work and the real scenario, we release a new large-scale query log dataset - **TaobaoQAC**, which contains rich real prefix-to-query click behaviors. We conduct experiments on TaobaoQAC to demonstrate the effectiveness or our approach, and results show that M^2A achieves superior performance compared with several strong baselines in both candidate ranking and query generation. We also conduct an online A/B testing and our approach has been deployed online.

CCS CONCEPTS

• **Information systems** → **Query log analysis**; • **Computing methodologies** → **Natural language generation**; **Multi-task learning**.

KEYWORDS

query auto-completion, text generation, multi-task learning

*This work was done when Di Yin was a summer intern in Alibaba Group.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '20, August 23–27, 2020, Virtual Event, CA, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403350

ACM Reference Format:
Di Yin, Jiwei Tan, Zhe Zhang, Hongbo Deng, Shujian Huang, and Jiajun Chen. 2020. Learning to Generate Personalized Query Auto-Completions via a Multi-View Multi-Task Attentive Approach. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394486.3403350

1 INTRODUCTION

Query auto-completion (QAC) is one of the most significant features of modern search engines, and it plays a key role in assisting users to express their information need more efficiently. During the user's typing process, there is a partial query (i.e. query prefix) in the search box, and the QAC system will automatically recommend a ranked list of completed queries (i.e. query auto-completions) for the user to choose one for the follow-up searching. In real application, when the query prefix is right or error-corrected, a successful QAC system should meet the following three requirements: (1) the prefix should be contained as a part of each recommended query; (2) the online service can provide query auto-completions whenever the query prefix is valid; (3) all presented query candidates should be relevant to the user's current searching intention, which is the key to click behaviors.

To implement such a system, most online search services, such as Taobao and Google, adopt a two-stage pipeline-like architecture [3]: matching and ranking. In matching, a set of full queries are retrieved from a prepared candidate pool, which contains plenty of popular queries mined from historical searching logs. In ranking, these candidates are ranked according to the degree of fitting the user's searching intention, which can be calculated with some statistical metrics or parameterized learning-based scoring models. Finally, due to the limitation of the drop-down list space, only several top-ranked candidates will be presented to the users as the final recommended query auto-completions.

Although the task of QAC has been studied for a long time along with the evolution of search engines, there still exist two major problems which have not been well solved as follows.

- **Weak Personalization**: Most QAC systems tend to assign higher ranking scores to these queries which are most frequent in query logs, which may result in most candidates in the recommended list irrelevant to the user's specific searching intention. When the query prefix is extremely short, this phenomenon is more serious, because only a few tokens contains very little information about the search intention. To

solve the problem of weak personalization in candidate ranking, almost all previous work are making efforts to improve the ranking quality by considering diverse features into the learning-to-rank [19] framework, such as search context [2], time [25], persona [24], reformulation behaviors [13, 18] and online trends [33]. However, deep learning based ranking models, which can automatically learn feature representations, have been rarely studied for QAC. What's more, it is still not explored to model multiple modality of historical sequential behaviors (e.g. searched queries and browsed items) for QAC at the same time, while more views of persona can contribute to more accurate and comprehensive user intention modelling.

- **Unseen Queries**: The QAC system often suffers from very poor performance when the user is typing a query which doesn't appear in the candidate pool. If the entered prefix cannot match any query candidates, the system will provide no results, which may hurt the user experience. Recently, some work propose to train recurrent neural network (RNN) based language models to automatically generate query auto-completions conditioned on the query prefix and some persona-related features [7, 12, 22]. Although this approach can solve the 'unseen queries' by breaking the limitation of candidate pool with parameterized models, it faces more serious 'weak personalization' because they have not utilized enough useful personal information, such as history behavior sequence. For evaluation, no work have verified the effectiveness of generative models with online A/B testing.

In this paper, we claim that modelling the user's short-time searching intention from recent sequential behaviors is essential for the QAC system to provide more accurate prediction. However, it's challenging to do this because the information contained in the behavior sequence is diverse and dynamic. The diversity means that the user may have some different interests in the history and not all of them are relevant to the current search intention. The dynamicity means that there may exist some complex dependencies between some behaviors, such as the interest transfer process and query reformulations. Some recurrent neural network based hierarchical encoders has been proposed to tackle the tasks of query suggestion [26] and multi-turn conversation [23], but they may not perform well because RNN usually biases to the close context [11].

To address the problems of weak personalization and unseen queries at the same time, we propose $\mathbf{M^2A}$, a novel **M**ulti-view **M**ulti-task **A**ttentive framework to learn personalized QAC models. Instead of considering only one view of historical sequential behaviors like previous work [2, 13, 18], we propose to extract useful information from two distinct views, including searched queries and browsed items (e.g. products presented in Taobao), at the same time. To learn representations of behaviors in text form, such as queries or item titles, we propose to exploit the contextual information for more precise behavior understanding. Specifically, a hierarchical encoder based on Transformer [30] is designed to obtain the context-free behavior-level and context-aware context-level representations of each behavior. Then we use a prefix-to-history attention mechanism to select most relevant and useful information from each view separately. Finally, the intention representation is composed by combining the above information.

In practice, the informativeness of the intention representation is significant for the entire model to achieve superior final performance. Mainstream models in sequential recommendation or multi-turn dialogue learn this kind of representations with a discriminative classification loss or a generative MLE-based loss. We argue that using only one kind of loss restricts the model from achieving better performance because it only utilizes one kind of supervisory information provided by query logs, which is a waste of the other kind. Inspired by multi-task learning [37], we propose to jointly train a CTR prediction model and a query generation model by sharing the multi-view encoder part, which makes the learning process fetch richer supervisory information from training samples to learn better representation.

To train and evaluate QAC models, previous work constructed prefix-query pairs from AOL, a public query log dataset, by splitting full queries. We argue that it is obviously different from the real industrial scenario because AOL does not contain any real prefix-to-query click behaviors. To bridge this gap, we construct and release a new large-scale query log dataset - **TaobaoQAC**, which contains rich real prefix-to-query click behaviors, and conduct experiments on it to evaluate the effectiveness of our approach and some strong baselines. Results show that $\mathrm{M^2A}$ achieves superior performance compared with several strong baselines both in candidate ranking and query generation.

Therefore, our main contributions can be summarized as follows.

- We proposed $\mathbf{M^2A}$, a novel multi-view multi-task attentive framework to learn personalized query auto-completion models. For more comprehensive user modelling, we propose to encode multiple views of historical sequential behaviors at the same time, with several novel Transformer-based hierarchical encoders. For more informative intention representation, we incorporate multi-task learning into our framework by jointly training a discriminative CTR prediction model and a generative query generation model.
- We release **TaobaoQAC**, a new large-scale query log dataset collected from a famous Chinese e-commerce platform. To our best knowledge, it's the first query log dataset with real prefix-to-query click behaviors. What's more, it is in the domain of e-commerce, which contains much richer personalization than general web search.
- We conduct offline experiments on TaobaoQAC. Results demonstrate the effectiveness of our proposed method to address two major problems in QAC. Extensive ablation study shows that our framework can learn better representation of user searching intention. We also propose a compromised solution to online deployment by using ranking and generative models jointly, and results of online A/B testing show some obvious promotion brought by it. Our approach has already been deployed online and is serving the traffic of mobile Taobao everyday.

2 RELATED WORK

In this section, we will discuss related work to introduce some background knowledge.

2.1 Query Auto-Completion

Most existing QAC systems work in a two-stage procedure: matching and ranking [3]. The traditional strategy is *MostPopularCompletion* (MPC), which ranks query candidates according to the popularity counted based on historical query logs. However, when the query prefix is extremely short, MPC often provide poor predictions [2]. To enhance the ranking quality, many efforts are made to incorporate some learning-to-rank [19] models, which consider some persona-related features, such as context [2, 17, 34], time [25], persona [4, 24], reformulation behaviors [13] and online trends [33]. The above approaches can improve the prediction accuracy on popular queries, but most QAC systems still suffer from poor performance on unseen queries. No suggestions will be provided if the prefix doesn't match any queries from the candidate pool. Some work begin to complete the unseen queries by retrieving [20] or generating [7, 12, 22] suitable query suffixes conditioned on the prefix. In our work, we explore how to utilize more kinds of user behaviors (multi-view) and more supervised information from query logs (multi-task) in a Transformer-based framework.

2.2 Neural Text Generation

In recent years, the interest in text generation has grown substantially in the community of natural language processing (NLP) with the success of neural network based sequence-to-sequence models [27] and attention mechanism [1, 30]. A neural text generation model usually includes two parts: encoder and decoder. To produce an expected text, the first step is to compress the source input into a dense representation with the encoder, and the second step is to generate a new text conditioned on this representation token by token with some decoding strategies, such as beam search. When the source input is a sequence of texts, such as a series of utterances in multi-turn conversation, some hierarchical encoders based on recurrent neural networks (RNNs) [15, 23, 26] are employed to encode the source sequence. They often consist of a sentence-level encoder and a context-level encoder. Firstly, the sentence-level encoder will compress each sentence in the context into a vector; then the context-level encoder will encode the produced vector sequence. Different from previous work, we design a new hierarchical encoder based on Transformer, which shows better performance than RNN-based methods.

2.3 Sequential Recommendation

The task of sequential recommendation is to predict the next item or basket, which a specific user will prefer, by exploiting the information in historical sequential behaviors. According to a recent survey [6], the sequential recommendation consists of three sub-tasks: experience-based sequential recommendations, transaction-based sequential recommendations and interaction-based recommendations. Our work is more relevant to the transaction-based sequential recommendations. In this line of work, recommendation models need to model the sequential dependency relationships between different items in the given behavior sequence. Previous work can be classified into four categories: MLP-based methods [31], RNN-based methods [8], CNN-based methods [29] and attention-based methods [14, 16, 36]. In our work, we study how to utilize multiple views of historical sequential behaviors for more personalized query auto-completion, and propose a Transformer-based hierarchical encoder to model sequential behaviors with an additional attention module to select most relevant information, and also explore how to enhance the model performance with multi-task learning.

3 FRAMEWORK DESCRIPTION

In this section, we will introduce our proposed $\mathbf{M^2A}$. We will first formulate the problem of personalized query auto-completion and then describe our approach.

3.1 Overview

3.1.1 ***Problem Formulation.*** Consider a user u is typing the query q, p denotes a query prefix entered by u in the search box. Before the current query, the recent M queries searched by u from the latest to the earliest are $q_1, q_2, \ldots, q_M$, and the recent N items (e.g. documents in text retrieval or products in e-commerce search) browsed by u from the latest to the earliest are $i_1, i_2, \ldots, i_N$. We need to train a query auto-completion model, which can be a ranking model or a generative model, to predict the query q conditioned on the current prefix p and sequential behaviors of searching and browsing, which can be formulated as:

$$p(q|p; q_1, q_2, \ldots, q_M; i_1, i_2, \ldots, i_N) \tag{1}$$

3.1.2 ***Overall Framework.*** As shown in Figure 1, our framework consists of an encoder part, which contains a prefix encoder and multiple hierarchical encoders for different views of historical sequential behaviors; a query generation part, which is a Transformer decoder to generate completed queries; and a CTR prediction part, which is a multi-layer fully-connected neural network to predict each candidate's click-through rate.

To obtain the dense representation of the query prefix, we use a self-attention based encoder - **Prefix Transformer Encoder** to encode it. Intuitively, the information contained in the prefix representation is not comprehensive to model the user's search intention for more personalized successive predictions. We propose to search in the user behavior sequences for more relevant information to enhance the source representation with the query prefix as key. To encode each behavior sequence, we design a new hierarchical encoder based on Transformer. Firstly, a **Behavior-Level Transformer Encoder** with a **Multi-Head Pooling** module encodes each behavior, which is in form of text, to compressed vectors. Then, based on the obtained vector sequence, we use the **Context-Level Transformer Encoder** to get the context-aware representations, with the self-attention mechanism to model the diversity and dynamicity in the sequence. However, not all information in the history is relevant to the current intention due to the diversity of the user behavior sequence [38]. Therefore, we make use of the query prefix presentation to select the most relevant and important information from the whole history, with the aid of the **Multi-Head Context Attention** module. To utilize the information from multiple views, we conduct multiple separate such hierarchical encoding. Then we combine the query prefix representation with multiple weighted representations of different views to compose the final intention representation. At last, the enhanced source representation is fed to the **Transformer Decoder** to generated the completed query

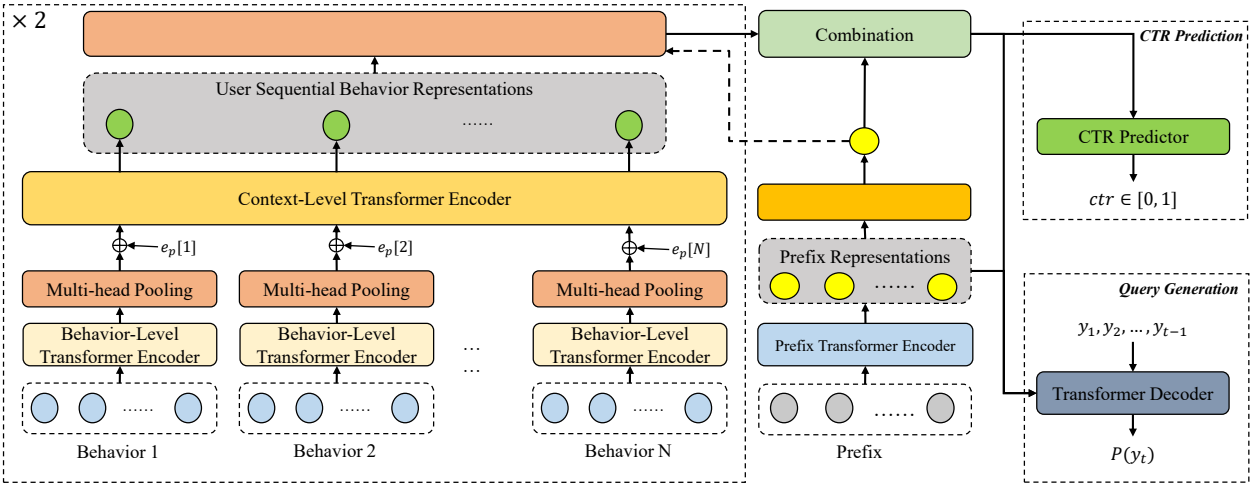


Figure 1: Model architecture of M^2A. The model input is a query prefix and two behavior sequences, and each behavior is a token sequence, which can be a searched query or a browsed item title. For each behavior, we use a behavior-level encoder and an additional multi-head pooling to obtain the context-free behavior-level representation, and then feed these vectors into the context-level encoder to get the context-aware context-level user sequential behavior representations. As shown in the left block, we conduct two independent such hierarchical encoding process for two distinct views, and then employ two multi-head context attention modules to select most relevant information from two views with the prefix representation from the middle part as key. The prefix representation are combined with the weighted view presentations as the final intention representation to feed into two specific tasks, including CTR prediction and query generation.

token by token, or to the **CTR Predictor** to predict the CTR score, ranging from 0 to 1.

We name our framework as **M^2A** because it's a **M**ulti-view **M**ulti-task **A**ttentive learning framework of discriminative and generative query auto-completion model. In the following sub-sections, we will describe some details of the model design.

3.2 Multi-View Attentive Hierarchical Encoding

In this part, we will introduce how our designed Transformer-based hierarchical encoder models historical sequential behaviors. The dominate RNN-based methods will be discussed at first, and then our proposed model will be introduced in detail.

3.2.1 ***RNN-based Achiecture.*** RNN structures have been well explored in hierarchical sequential modelling [8, 23, 26, 28]. The encoder part of their proposed models often contains a behavior-level RNN and context-level RNN. To encode sequential behaviors, the behavior-level RNN firstly encodes each behavior and uses the last hidden state as the behavior-level representation, which will later be the input of the context-level RNN to get its last hidden state the context-level representation. Assume that there are N items in the sequential behaviors, let RNN_b and RNN_c as the behavior-level and context-level encoders respectively, the encoding process can be formulated as:

$$\mathbf{h}_{n,t} = \text{RNN}_b(\mathbf{h}_{n,t-1}, \mathbf{w}_{n,t}), \ \ t = 1, \ldots, len(i_n) \tag{2}$$

$$\mathbf{h}_n = \text{RNN}_c(\mathbf{h}_{n-1}, \mathbf{h}_{m,T}), \ \ n = 1, \ldots, N \tag{3}$$

However, in practice, the performance of this approach is often limited by the model capacity of recurrent neural networks [11].

3.2.2 ***Transformer-based Architecture.*** To breakthrough the limitation of RNN-based architecture, we introduce some novel techniques from Transformer [30] to enhance the capability of hierarchical modelling.

Transformer Layers. We follow the design of Transformer layers in [30]. Given an input sequence of length t, we compute hidden representations $\mathbf{h}_i^l$ at each layer l for each position i simultaneously. Here, the hidden representations $\mathbf{h}_i^l \in \mathbb{R}^d$ are stacked together into the matrix $\mathbf{H}^l \in \mathbb{R}^{t \times d}$ to compute attention function on all positions simultaneously. As illustrated in [30], a Transformer layer often contains two sub-layers, a *Multi-Head Self-Attention* sub-layer and a *Position-wise Feed-Forward Network* sub-layer.

Attention mechanism allows capturing the dependencies between different representations regardless of their distance in the sequence. Vaswani et al. [30] found that it's beneficial to jointly attend to information from different representation subspaces and proposed the multi-head self-attention mechanism, which is the key to Transformer's outperforming RNN in hierarchical encoding. $\mathbf{H}^l$ is linearly projected into h subspaces, and then apply h attention functions in parallel to produce the output representations which concatenated and once again projected:

$$\text{MH}(\mathbf{H}^l) = [\text{head}_1; \text{head}_2; \ldots; \text{head}_h]\mathbf{W}^O \tag{4}$$

$$\text{head}_i = \text{Attention}(\mathbf{H}^l\mathbf{W}_i^Q, \mathbf{H}^l\mathbf{W}_i^K, \mathbf{H}^l\mathbf{W}_i^V) \tag{5}$$

where these projection matrices for each head $\mathbf{W}_i^Q \in \mathbb{R}^{d \times d/h}$, $\mathbf{W}_i^K \in \mathbb{R}^{d \times d/h}$, $\mathbf{W}_i^V \in \mathbb{R}^{d \times d/h}$ and $\mathbf{W}_i^O \in \mathbb{R}^{d \times d}$ are learnable

parameters. The attention function is *Scaled Dot-Product Attention*:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d/h}})\mathbf{V} \tag{6}$$

where the temperature $\sqrt{d/h}$ is introduced to produce a softer attention distribution for avoiding extremely small gradients [30].

As described above, the self-attention sub-layer is mainly based on linear projections. To endow the model with nonlinearity and interactions between different dimensions, a *Position-wise Feed-Forward Network* is applied to the outputs of the self-attention sub-layer, separately and identically at each position. It consists of two affine transformations with a ReLU activation between them.

$$\text{PFFN}(\mathbf{H}^l) = [\text{FFN}(\mathbf{h}_1^l)^T; \ldots; \text{FFN}(\mathbf{h}_t^l)^T]^T \tag{7}$$

$$\text{FFN}(\mathbf{x}) = \text{ReLU}(\mathbf{x}\mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \tag{8}$$

In practice, it is usually beneficial to learn more complex dependency by stacking the self-attention layers, but the network will become more difficult to train as it goes deeper. Therefore we add a residual connection around each of the two sublayers, followed by layer normalization. What's more, the strategy of dropout is applied to the output of each sub-layer before it is normalized. The refinement of hidden representations is formulated as follows:

$$\mathbf{H} = \text{Transformer}(\mathbf{H}^{l-1}), \forall i \in [1, \ldots, L] \tag{9}$$

$$\text{Transformer}(\mathbf{H}^{l-1}) = \text{LN}(\mathbf{A}^{l-1} + \text{Dropout}(\text{PFFN}(\mathbf{A}^{l-1}))) \tag{10}$$

$$\mathbf{A}^{l-1} = \text{LN}(\mathbf{H}^{l-1} + \text{Dropout}(\text{MH}(\mathbf{A}^{l-1}))) \tag{11}$$

Embedding Layers. Without any recurrence or convolutional module, the above Transformer layer is not aware of the order of the input sequence. In order to make use of the sequential information of the input, we inject *Positional Embeddings* into the input embeddings at the bottoms of the stacked Transformer layers. It may be important to note that the position i is the distance between the embedded behavior and the current time while encoding the behavior sequence.

Multi-Head Pooling. To obtain fixed-length representations of variable-length behavior items, we introduce a multi-head pooling mechanism, where for each item, weight distributions over tokens are calculated, allowing the model to flexibly encode items in different representation subspaces by attending to different tokens.

Let $x_{ij}^{l-1} \in \mathbb{R}^d$ denote the output vector of the last Transformer layer for token t_{ij}, which is the j-th token in the i-th item, and we use it as the input for the current layer. For each item, for head $z \in \{1, \ldots, n_{head}\}$, we first transform the input vectors into attention scores a_{ij}^z and the value vectors b_{ij}^z. Then, for each head, we calculate a probability distribution $\hat{a}_{ij}^z$ over tokens within the item based on attention scores:

$$a_{ij}^z = \mathbf{W}_a^z x_{ij}^{l-1}, \quad b_{ij}^z = \mathbf{W}_b^z x_{ij}^{l-1} \tag{12}$$

$$\hat{a}_{ij}^z = \exp(a_{ij}^z) / \sum_{j=1}^{n} \exp(a_{ij}^z) \tag{13}$$

$$\text{head}_i^z = \text{LN}(\mathbf{W}_c^z \sum_{j=1}^{n} a_{ij}^z b_{ij}^z) \tag{14}$$

where $\mathbf{W}_a^z \in \mathbb{R}^{1 \times d}$, $\mathbf{W}_b^z \in \mathbb{R}^{d_{head} \times d}$ and $\mathbf{W}_c^z \in \mathbb{R}^{d_{head} \times d_{head}}$ are learnable parameters and $d_{head} = d/n_{head}$ is the dimension of each head. The multi-head pooling mechanism can flexibly incorporate multiple heads, with each items having multiple attention distributions, thereby focusing on different views of the input.

Multi-View Information Selection. For more personalized QAC prediction, some previous work propose to infer the searching intention from the past searched queries [13, 18]. However, some users, especially new users of search engines, have limited ability to express their needs with their typed queries, which makes it difficult to infer their real intentions accurately only from historical searched queries.

Intuitively, the browsing history of users can provide additional evidences of their searching intentions, because users tend to click these items which is relevant to their searching intentions. For a more comprehensive modelling of user searching intention, we propose to utilize multiple distinct views of historical sequential behaviors, including searched queries and browsed items, in the QAC model at the same time.

Due to the diversity and dynamicity of historical sequential behaviors, not all behaviors from each view are relevant to the current searching intention, so it's necessary for the QAC model to conduct a selection of helpful information before prediction. To do this 'selection' action, we introduce a multi-head attention module (i.e. 'Multi-Head Context Attention' in Section 3.1.2) into the encoder part. An attention distribution will be calculated on the whole history with the query prefix representation as the key. Then a weighted sum of behavior representations will be produced. At last, we combine the query prefix representation with the weighted behavior representations to the searching intention representation via a simple concatenation as follows:

$$\mathbf{h}_u = [\mathbf{h}_p; \sum_m a_m \mathbf{h}_{q,m}; \sum_n b_n \mathbf{h}_{i,n}] \tag{15}$$

3.3 Multi-Task Learning

To train QAC models, some supervisory signals are required. In the query logs for training, two kinds of supervisory information are provided: the first kind is the click label of each recommended query candidate (1 for clicked queries and 0 for queries without clicks), and we can train ranking-based models with these labels; the other kind is the searched query, which is selected from recommended candidate list or completely typed by the user, and we can train generation-based models with these queries as target.

As described in Section 2.1, previous work usually use only one of them to train one QAC model, which is ranking-based or generation-based, and directly ignore another kind. We argue that it's a waste of training data and related supervisory information. To learn more informative representations of the multi-view attentive hierarchical encoder with richer supervisory signals, inspired by recent success of multi-task learning [37], we propose to jointly train a CTR prediction and a query generation model in a unified framework by sharing the encoder part.

3.3.1 ***Task 1: CTR Prediction (CP).*** To learn from the supervision of click behaviors, we introduce a CTR prediction task into our framework. Given the user intention representation $\mathbf{h}_u$ and the representation of candidate, we concatenate them and then use several fully connected layers to learn the combination of features automatically. The output of the above MLP is a probability

$p(x)$, which represents the click-through rate (CTR) of the specific candidate. For training, we construct positive and negative samples from query logs in the following way: some persona-related features, including the entered prefix, previously searched queries and browsed items, are combined with the corresponding clicked queries as positive samples; they are also combined with random selected queries which haven't been clicked as negative samples. Then we use these samples to optimize the following point-wise ranking loss function:

$$\mathcal{L}_{\mathrm{c}} = -\frac{1}{N} \sum_{(\mathbf{x},y)\in\mathcal{S}} (y \log p(\mathbf{x}) + (1-y)\log(1-p(\mathbf{x}))) \tag{16}$$

3.3.2 ***Task 2: Query Generation (QG)***. We also introduce a query generation task to exploit the supervision from full queries. Conditioned on the intention representation and the prefix token representations, we use a Transformer decoder to generate the final searched query. The query generation model is trained end-to-end, and the objective is to maximize the probability of the ground truth token sequence based on maximum likelihood estimation (MLE). Thus the following loss function is minimized:

$$\mathcal{L}_{\mathrm{g}} = -\sum_{i=1}^{N}\sum_{j=1}^{|\mathbf{y}^i|} \log P_j^i(y_j^i | y_{0:j-1}^i, p, q_{1:m}, i_{1:n}) \tag{17}$$

where N is the number of samples in the query logs for training, $|\mathbf{y}^i|$ indicates the length of the i-th ground truth query, P_j^i refers to the predicted probability distribution at the j-th step for the n-th sentence, hence $P_j^i(y_j^n | y_{0:j-1}^i, p, q_{1:m}, i_{1:n})$ is the probability of generating the ground truth token at the j-th step, conditioned on previous generated tokens, the query prefix and the user's historical behavior sequences.

For these logs with click behaviors of users, we use prefixes typed by users as source input and queries clicked by users as target output. However, the problem of 'unseen queries' still doesn't have any targeted solutions, because it lies in logs without any clicks. To utilize these logs, we construct prefix-query pairs by splitting full queries similar to most previous work [20]. The training objective is the same to Eq. 17, but we denote it as $\mathcal{L}_{\mathrm{u}}$.

In real application, the QAC system needs to recommend a ranked list of query candidates for the user to choose. For the query generation model, we take the strategy of diverse beam search to generate several top-ranked query candidates, similar to [22].

4 EXPERIMENTS

In this section, we present our experiments, including datasets, evaluation metrics, model comparison and result analysis.

4.1 Dataset Construction

To train and evaluate the QAC models, most previous work construct prefix-query pairs as positive samples from a public query log dataset (e.g. AOL) by splitting the full queries entered by users, and then randomly select some queries from the candidate pool to construct negative samples. These datasets don't contain any real prefix-to-query click behaviors, but in real industrial development, engineers usually train learning-to-rank models using prefix-query pairs, which consist of prefixes entered by users and completed queries clicked by users as positive samples and other pairs as negative samples. Therefore, it's obviously a gap between most previous research work and the real industrial deployment due to the lack of real prefix-to-query click behaviors in existing public query log datasets.

Table 1: Statistics of AOL Query Log and TaobaoQAC.

Dataset	AOL Query Log	TaobaoQAC
Number of users	489,531	49,964,279
Number of queries	9,440,731	12,998,681
Number of items	1,622,236	88,856,298
Number of samples	33,882,423	230,199,674

To bridge this gap, we collect a new large-scale query log dataset from the mobile search engine of Taobao, the largest Chinese e-commerce platform. Its advantages compared with existing query log datasets (e.g. AOL) include: (1) rich real prefix-to-query click behaviors, which consist of real typed prefixes and real clicks of recommended queries; (2) the first query log dataset in the domain of e-commerce, where personalization plays key role for better performance [38]. We name this dataset as **TaobaoQAC**[1]. Some statistics results about TaobaoQAC and AOL datasets are shown in Table 1, and we will specify some details in the following part.

4.1.1 ***Details of TaobaoQAC***. We sample parts of query logs between September 1, 2019 and September 10, 2019. This dataset contains 230,199,674 logs, which cover 49,964,279 distinct users, 12,998,681 distinct queries and 375,420,920 online products. It is partitioned into three parts: the logs between September 1, 2019 and September 7, 2019 are used as training data, 40,000 logs sampled from September 8, 2019 are used as validation data, and the other logs are used as test data.

It's necessary to note that these logs can been divided into two kinds: seen logs (154,272,712) and unseen logs (75,926,962). All samples contain the query prefix, the previous 10 searched queries and the titles of previous 20 browsed products. What's more, samples in seen logs provide some completed queries recommended by the mobile search engine of Taobao and the position of clicked query, but samples in unseen logs only provide the full query typed by users completely because users don't make any clicks in these samples.

4.2 Evaluation Metrics

4.2.1 ***Ranking Metrics***. We use the metric of mean reciprocal rank (MRR) to evaluate performance of ranking or generative models. Let rank_i to be a positive number which presents the position of the golden query of the i-th sample in the ranked candidates, the MRR score is calculated with the following equation:

$$\mathrm{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\mathrm{rank}_i} \tag{18}$$

In practice, the metric of MRR often presents serious inconsistency with some online metrics due to the existence of position bias. Therefore, we also use the metric of Unbiased MRR similar to the one proposed in [32] to evaluate the performance of each

[1]We will release the encrypted version of TaobaoQAC and more experiement details for reproduction at https://github.com/yd1996/KDD2020_TaobaoQAC.

model on Taobao Query Log, in which some real online behaviors are available for us to estimate the position bias w_i.

$$\text{Unbiased MRR} = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i \frac{1}{\text{rank}_i} \tag{19}$$

Note that we evaluate the performance on seen and unseen queries separately. To compare the performance between ranking and generative models, we use generative models as scoring function to give scores to the same candidates for ranking models, and then calculate the MRR and Unbiased MRR. To evaluate on unseen queries, we generate 10 top-ranked candidates with beam search and then use this list to calculate the two above metrics. If the golden query is not in this list, we will set its related metrics to 0.

4.2.2 ***Generation Metrics.*** To evaluate the degree of the generated query to match the ground-truth query, we calculate the score BLEU [21], a classical evaluation metric in the field of machine translation, between the top one generated query candidates and the corresponding full queries clicked or typed by users.

4.3 Comparative Methods

We compare our framework with the following strong baselines, which can be divided into ranking models and generative models. As follows, the first three models belong to ranking models, and others are neural network based generative models.

- **MPC**: *MostPopularCompletion* [2] is the traditional approach to QAC. It ranks candidates based on the popularity of each query in the whole past query logs.
- **GRM**: We represent the prefix, context and query candidate with Bag-of-Words (BoW) vectors, and then train a LambdaMART [35] model, which is a **G**BDT-based **R**anking **M**odel, with these features.
- **DRM**: We also implement a **D**eep learning based **R**anking **M**odel as a baseline. Its model architecture is similar to Deep Interest Network (DIN) [38] but we use the behavior-level Transformer encoder and the multi-head pooling introduced by us to get the representation of each behavior.
- **RNNLM**: We use **R**ecurrent **N**eural **N**etwork based **L**anguage **M**odel implemented with long short-term memory [9] to generate the query suffix conditioned on the entered prefix.
- **Seq2Seq**: Seq2Seq refers to the standard sequence-to-sequence model enhanced with attention mechansim [1]. In query auto-completion, the source input is the query prefix and the target output is the clicked or typed full query.
- **Transformer**: A standard Transformer [30] is used as a baseline. The input and output are the same as Seq2Seq.
- **HRED**: **H**ierarchical **R**ecurrent **E**ncoder **D**ecoder is firstly proposed to tackle the task of query suggestion [26]. We modify it as a personalized RNNLM [7] with two hierarchical recurrent encoders to compress two kinds of behavior sequences into conditional vectors.
- **AHRED**: HRED ignores that not all previous user behaviors are relevant to the current search intention and often suffers from the limitation of model capacity [11]. We enhance the HRED model with an attention mechanism, similar to [5], and the modified model is named as **A**ttentive **HRED**.

Table 2: Experiment results on TaobaoQAC.

Model	Seen			Unseen		
	BLEU	MRR	UMRR	BLEU	MRR	UMRR
MPC	-	0.533	0.748	-	0.000	0.000
GRM	-	0.554	0.762	-	0.000	0.000
DRM	-	0.572	0.788	-	0.000	0.000
M^2A(CP)	-	**0.579**	**0.796**	-	0.000	0.000
RNNLM	53.89	0.511	0.735	47.64	0.345	0.653
Seq2Seq	58.07	0.543	0.748	53.89	0.440	0.718
Transformer	59.23	0.549	0.754	54.40	0.443	0.721
HRED	54.38	0.515	0.742	49.54	0.426	0.705
AHRED	57.61	0.549	0.756	53.22	0.448	0.727
M^2A(QG)	**62.23**	0.573	0.785	**59.17**	**0.548**	**0.758**

4.4 Implementation Details

Our model and eight baselines are all implemented in Python. GRM is implemented using a open source learning-to-rank framework[2]. All ranking and generative models based on deep learning are implemented with Texar[3] [10], a toolkit for machine learning and text generation based on Tensorflow[4]. All experiments are all conducted on four or eight NIVIDIA 1080TI GPUs with a single batch size of 32 by training in a distributed way. Hyper-parameters are tuned with grid search on validation sets, and we report the result of each model on test sets.

For our model, we set the number of heads in all attention modules as 8, the number of Transformer blocks of all encoders and the decoder as 3. The hidden dim is 256 for each encoder, and 768 for the decoder. The input dim of CTR predictor is 1024 and the hidden dims of the following MLP layers are 512, 256 and 64. Activation functions are ReLU for previous layers and sigmoid for the final prediction. We train the model using Adam with learning rate of 1e-4, $\beta_1 = 0.9$, $\beta_2 = 0.999$, l_2 weight decay of 0.01, and linear decay of learning rate. For TaobaoQAC, we use Chinese characters to build the vocabulary of queries, and tokenize each item title using the Chinese word segmentation service provided by Aliyun[5], to build the vocabulary of items. The vocabulary size is limited to 7000 Chinese characters for queries and 50000 Chinese words for items.

4.5 Overall Performance Comparison

Table 2 summarizes the results of all models on TaobaoQAC. For seen and unseen queries, the scores of BLEU, MRR and Unbiasd MRR (UMRR) are reported separately. By utilizing the information from multiple views and multiple tasks, the CTR prediction model, termed as M^2A(CP), and the query generation model, termed as M^2A(QG), in our framework achieve the best performance in all ranking and generation models respectively.

For generative models, we have the following observations. Firstly, the encoder part of AHRED can be seen as an RNN-based multi-view hierarchical encoder, and AHRED performs much more poorly than M^2A(QG) and even not comparable to a standard Transformer without any personal features, which shows the power of our proposed Transformer-based multi-view hierarchical encoding. Secondly, by comparing HRED and AHRED, some prefix-to-history

[2]https://github.com/jma127/pyltr
[3]https://github.com/asyml/texar
[4]https://www.tensorflow.org/
[5]https://www.aliyun.com/

Table 3: Ablation study of CTR prediction models. We only report the scores of MRR and Unbiased MRR on seen logs.

Model	Seen	
	MRR	UMRR
V_q	0.552	0.762
V_i	0.533	0.751
V_q + V_i	0.564	0.768
V_q + V_i + L_g	0.574	0.789
V_q + V_i + L_g + L_u	**0.579**	**0.796**

Table 4: Ablation Study of query generation models.

Model	Seen			Unseen		
	BLEU	MRR	UMRR	BLEU	MRR	UMRR
V_q	57.69	0.562	0.777	36.07	0.221	0.646
V_i	57.17	0.554	0.773	32.72	0.186	0.642
V_q + V_i	61.49	0.573	0.783	37.41	0.229	0.645
V_q + V_i + L_c	**62.97**	**0.575**	**0.788**	37.29	0.229	0.644
V_q + V_i + L_u	60.22	0.569	0.776	58.88	0.545	0.750
V_q + V_i + L_c + L_u	62.23	0.573	0.785	**59.17**	**0.548**	**0.758**

attention modules to select most relevant information from multiple distinct views are significant for more accurate prediction. At last, most baselines perform far worse on unseen logs than seen logs, but M^{2}A(QG) can achieve a comparable result, which shows the promising prospect of our approach for solving the problem of 'unseen queries'.

4.6 Ablation Study

M^{2}A is a complex learning framework, and multiple views of historical sequential behaviors and multiple tasks with different supervisions altogether contribute to its superior performance. In this section, we conduct ablation studies to evaluate the impact of different model components on the overall performance. The studied components include: 1) the view of searched queries - V_q, 2) the view of browsed items - V_i, 3) the CTR prediction task - L_c, 4) the query generation task for clicked queries - L_g and 5) the query generation task for typed queries - L_u. Results of the CTR prediction model and the query generation model are reported in Table 3 and Table 4 respectively.

To evaluate the contribution of each view, we separately train CTR prediction models and query generation models with only V_q, only V_i or both V_q and V_i. We observe that these models with only V_q perform better than other models with only V_i, and models with both views achieve the best results. We explain that historical searched queries usually directly contain some tokens which users tend to type due to the reformulation behaviors but browsed items don't, resulting in the gap between two kinds of single-view models. But these queries sometimes cannot reflect the searching intention comprehensively, and browsed items can provide some additional information, contributing to the success of multi-view models.

To evaluate the contribution of each task, we incrementally add these tasks into the training process. It can be seen that models trained with both L_c and L_g perform much better than models with only one of them, which means that jointly optimizing multiple tasks can enhance the encoder representation. However, after we add L_u into the learning framework, the final performance of generative models will have a large rise on unseen logs, but an obvious decrease on seen logs. The cause may be the gap between real and constructed prefix-to-query click behaviors.

Table 5: Examples of query candidates recommended by different models for the same prefix and history behavior. Due to the space limitation, we present only recent five searched queries (Context$_q$) and browsed items (Context$_i$) ranked by time from recent to past, and we show only three top-ranked query candidates for each model.

Context$_q$	休闲裤女 (casual pants for girls) 九分休闲裤 (nine-cent casual pants) 超短牛仔裤 (short jeans) 秋季街装女 (autumn suit for girls) 牛仔外衣 (denim jacket)
Context$_i$	阿迪达斯裳侣运动裤 (adidas couple sports pants) 嘻哈宽松街头休闲裤 (hip-hop loose street leisure pants) 彪马运动休闲裤女 (puma sports leisure pants for girls) 新款高腰款瘦女裤 (new high waist slim pants for women) 韩版宽松运动裤(korean version loose slacks)
Prefix	运动 (sports)
Golden	运动裤女 (sports pants for girls)
MPC	运动鞋2019新款(sports shoes 2019 new style) 运动街装女 (sports suit for girls) 运动外衣女 (sports coat for girls)
DRM	运动街装女 (sports suit for girls) 运动鞋2019新款(sports shoes 2019 new style) 运动裤女 (**sports pants for girls**)
Transformer	运动鞋2019新款(sports shoes 2019 new style) 运动街装女 (sports suit for girls) 运动街装 (sports suit)
M^{2}A(QG)	运动裤女(**sports pants for girls**) 运动休闲裤女(**sports leisure pants for girls**) 运动裤女宽松(**sports pants for girls loose**)

4.7 Case Study

The job of query auto-completion systems is to assist users in formulating queries to express their search intentions. Thus, the suggested candidate list needs to be personalized because different people have various intentions when they are typing the same query prefix. An ideal auto-completion system should provide diverse queries relevant to the same or similar intentions. However, due to our observations, existing count-based or learning-based ranking models tend to assign some very popular queries with higher ranking scores than other queries, even when they are not relevant to the user's intention. As presented in Table 5, when the user has typed '运动'(sports) in the search box of Taobao search engines, some popular queries, such as '运动鞋2019新款' (sports shoes 2019 new style), always appear in the candidate list, regardless of whether the user intends to search for them, and then only a few candidates in the list is relevant to the right intention, narrowing down the space of choice. The similar phenomenon occurs on the result of Transformer, a generative model without considering any personal information. By utilizing the information from multiple views and multiple tasks for model learning, M^{2}A(QG) can predict the user's intention more accurately and then suggest some different but relevant query candidates for user to select with beam search, which brings more personalized experience to the user.

4.8 Online A/B Testing

A careful online A/B testing is conducted on the mobile search engine of Taobao from 2019-09 to 2019-10. We will describe a compromised solution to online deployment and report some gains brought by our approach to online service.

Offline experiments show the ability of generative models to solve the problem of 'unseen queries'. However, due to the slow decoding speed of beam search, the demand of quick service response becomes the obstacle to the online deployment of generative models. Previous research work [7, 12, 22] have not provide any solutions to this problem, and we will make some exploration here.

For online deployment, we propose a compromised method, in which the CTR prediction model and the query generation model learned with M^2A jointly work. Firstly, we mined about 80 millions query prefixes, which were typed by users but don't receive any recommended queries, from historical searching logs. Then, we produce 10 query candidates for each log, of which the prefix is contained in the mined set, with the query generation model. At last, we construct a prefix-to-query index with generated prefix-query pairs for online matching and use the CTR prediction model for candidate ranking.

During nearly two months of A/B testing, our approach contributes up to **3.84%** PV, **1.6%** UV and **4.12%** Click promotion compared with the previous system. What's more, it also contributes **11.06%** promotion in the online service response ratio[6], which was previously about 75%. Our approach has already been deployed online and is serving the traffic of mobile Taobao everyday.

5 CONCLUSIONS

In this paper, we study the problem of query auto-completion (QAC). To address two major problems of QAC - weak personalization and unseen queries, we propose **M^2A** - a novel multi-view multi-task attentive approach to QAC. Our framework can exploit more information to achieve superior performance than previous work, by utilizing multiple views and multiple kinds of supervision at the same time. To bridge the existing gap between research work and real scenario, we release a large-scale query log dataset - **TaobaoQAC**. To our best knowledge, it's the first dataset with real prefix-to-query click behaviors. Experiments on TaobaoQAC demonstrate the effectiveness of our approach. We also explore how to deploy our framework to improve the online service, and results of A/B testing shows the obvious promotion brought by our work.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR 2015*.
[2] Ziv Bar-Yossef and Naama Kraus. 2011. Context-Sensitive Query Auto-Completion. In *Proceedings of WWW 2011*. ACM.
[3] Fei Cai and Maarten de Rijke. 2016. A Survey of Query Auto Completion in Information Retrieval. In *Foundations and Trends in Information Retrieval*.
[4] Fei Cai, Shangsong Liang, and Maarten de Rijke. 2014. Time-sensitive Personalized Query Auto-Completion. In *Proceedings of CIKM 2014*. 1599–1608.
[5] Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. 2017. Learning to Attend, Copy, and Generate for Session-Based Query Suggestion. In *Proceedings of CIKM 2017*. 1747–1756.
[6] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2019. Deep Learning-based Sequential Recommender Systems: Concepts, Algorithms, and Evaluations. (2019).
[7] Nicolas Fiorini and Zhiyong Lu. 2018. Personalized neural language models for real-world query auto completion. (2018).
[8] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *Proceedings of ICLR 2016*.
[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[10] Zhiting Hu, Haoran Shi, Bowen Tan, Wentao Wang, Zichao Yang, Tiancheng Zhao, Junxian He, Lianhui Qin, Di Wang, Xuezhe Ma, Zhengzhong Liu, Xiaodan Liang, Wanrong Zhu, Devendra Singh Sachan, and Eric P. Xing. 2019. Texar: A Modularized, Versatile, and Extensible Toolkit for Text Generation. In *Proceedings of ACL 2019, Volume 3: System Demonstrations*. 159–164.
[11] Fakultit F/Jr Informatik, Yoshua Bengio, Paolo Frasconi, and Jfirgen Schmidhuber. 2001. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies. (2001).
[12] Aaron Jaech and Mari Ostendorf. 2018. Personalized Language Model for Query Auto-Completion. (2018).
[13] Jyun-Yu Jiang, Yen-Yu Ke, Pao-Yu Chien, and Pu-Jen Cheng. 2014. Learning User Reformulation Behavior for Query Auto-Completion. In *SIGIR 2014*.
[14] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-Attentive Sequential Recommendation. In *Proceedings of ICDM 2018*. 197–206.
[15] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A Hierarchical Neural Autoencoder for Paragraphs and Documents. In *Proceedings of ACL 2015*.
[16] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *Proceedings of CIKM 2017*.
[17] Liangda Li, Hongbo Deng, Jianhui Chen, and Yi Chang. 2017. Learning Parametric Models for Context-Aware Query Auto-Completion via Hawkes Processes. In *Proceedings of WSDM 2017*. 131–139.
[18] Liangda Li, Hongbo Deng, Anlei Dong, Yi Chang, Hongyuan Zha, and Ricardo Baeza-Yates. 2015. Analyzing User's Sequential Behavior in Query Auto-Completion via Markov Processes. In *Proceedings of SIGIR 2015*. ACM.
[19] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.
[20] Bhaskar Mitra and Nick Craswell. 2015. Query Auto-Completion for Rare Prefixes. In *Proceedings of CIKM 2015*. 1755–1758.
[21] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *ACL 2002*.
[22] Dae Hoon Park and Rikio Chiba. 2017. A Neural Language Model for Query Auto-Completion. In *Proceedings of SIGIR 2017*.
[23] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. In *Proceedings of AAAI 2016*. 3776–3784.
[24] Milad Shokouhi. 2013. Learning to Personalize Query Auto-Completion. In *Proceedings of SIGIR 2013*. ACM.
[25] Milad Shokouhi and Kira Radinsky. 2012. Time-Sensitive Query Auto-Completion. In *Proceedings of SIGIR 2012*. ACM.
[26] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion. In *CIKM 2015*.
[27] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of NIPS 2014*. 3104–3112.
[28] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. In *DLRS@RecSys 2016*. 17–22.
[29] Trinh Xuan Tuan and Tu Minh Phuong. 2017. 3D Convolutional Networks for Session-based Recommendation with Content Features. In *RecSys 2017*.
[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Proceedings of NIPS 2017*. 5998–6008.
[31] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning Hierarchical Representation Model for Next Basket Recommendation. In *Proceedings of SIGIR 2015*. 403–412.
[32] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position Bias Estimation for Unbiased Learning to Rank in Personal Search. In *Proceedings of WSDM 2018*. 610–618.
[33] Yingfei Wang, Ouyang Hua, Hongbo Deng, and Chang Yi. 2017. Learning Online Trends for Interactive Query Auto-Completion. *IEEE TKDE* (2017).
[34] Stewart Whiting and Joemon M. Jose. 2014. Recent and robust query auto-completion. In *Proceedings of WWW 2014*. 971–982.
[35] Qiang Wu, Christopher J. C. Burges, Krysta Marie Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *IR* (2010).
[36] Shuai Zhang, Yi Tay, Lina Yao, and Aixin Sun. 2018. Next Item Recommendation with Self-Attention. *CoRR* abs/1808.06414 (2018).
[37] Yu Zhang and Qiang Yang. 2017. A Survey on Multi-Task Learning. *CoRR* abs/1707.08114 (2017).
[38] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *Proceedings of KDD 2018*. 1059–1068.

[6] The ratio of prefixes with recommended completed queries in all user typed prefixes.

A APPENDIX

A.1 Calculating Unbiased MRR

In Taobao mobile search engine, the QAC module presents only 10 top-ranked recommended queries when the user is typing. In this paper, we follow the same setting and calculate the ranking metrics, including MRR and Unbiased MRR (UMRR) introduced in Section 4.2.1, with 10 top-ranked candidates. In Table 6, we list the position bias weights w_i estimated on Taobao mobile search engine for the reproduction of the Unbiased MRR metric.

Table 6: Estimated bias weights for positions from 1 to 10.

Position (i)	Bias (w_i)
1	0.321
2	0.195
3	0.130
4	0.100
5	0.080
6	0.065
7	0.050
8	0.036
9	0.020
10	0.003

信息检索领域

SIGIR 2020

ESAM: Discriminative Domain Adaptation with Non-Displayed Items to Improve Long-Tail Performance

Zhihong Chen[12‡], Rong Xiao[2⋆], Chenliang Li[3†], Gangfeng Ye[2], Haochuan Sun[2], Hongbo Deng[2]
[1]Institute of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China
[2]Alibaba Group, Hangzhou, China
[3]School of Cyber Science and Engineering, Wuhan University, Wuhan, China
[1]zhihongchen@zju.edu.cn, [2]{xiaorong.xr, gangfeng.ygf, haochuan.shc, dhb167148}@alibaba-inc.com
[3]cllee@whu.edu.cn

arXiv:2005.10545v1 [cs.IR] 21 May 2020

ABSTRACT

Most of ranking models are trained only with displayed items (most are hot items), but they are utilized to retrieve items in the entire space which consists of both displayed and non-displayed items (most are long-tail items). Due to the sample selection bias, the long-tail items lack sufficient records to learn good feature representations, *i.e.,* data sparsity and cold start problems. The resultant distribution discrepancy between displayed and non-displayed items would cause poor long-tail performance. To this end, we propose an entire space adaptation model (ESAM) to address this problem from the perspective of domain adaptation (DA). ESAM regards displayed and non-displayed items as source and target domains respectively. Specifically, we design the attribute correlation alignment that considers the correlation between high-level attributes of the item to achieve distribution alignment. Furthermore, we introduce two effective regularization strategies, *i.e., center-wise clustering* and *self-training* to improve DA process. Without requiring any auxiliary information and auxiliary domains, ESAM transfers the knowledge from displayed items to non-displayed items for alleviating the distribution inconsistency. Experiments on two public datasets and a large-scale industrial dataset collected from Taobao demonstrate that ESAM achieves state-of-the-art performance, especially in the long-tail space. Besides, we deploy ESAM to the Taobao search engine, leading to significant improvement on online performance. The code is available at https://github.com/A-bone1/ESAM.git

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**; • **Computing methodologies** → *Transfer learning*.

KEYWORDS

Domain Adaptation, Ranking Model, Non-displayed Items

‡ This work was done as a research intern in Alibaba Group.
⋆ Rong Xiao contributed equally to this research.
† Chenliang Li is the corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGIR '20, July 25–30, 2020, Virtual Event, China
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401043

ACM Reference Format:
Zhihong Chen, Rong Xiao, Chenliang Li, Gangfeng Ye, Haochuan Sun, Hongbo Deng. 2020. ESAM: Discriminative Domain Adaptation with Non-Displayed Items to Improve Long-Tail Performance. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401043

1 INTRODUCTION

A typical formulation of the ranking model is to provide a rank list of items given a query. It has a wide range of applications, including recommender systems [22, 33], search systems [4, 20], and so on. The ranking model can be formulated as: $q \xrightarrow{R} \hat{\mathbb{D}}_q$, where q is the query, e.g., user profiles and user behavior history in the recommender systems, and user profiles and keyword in the personalized search systems, depending on specific rank applications. $\hat{\mathbb{D}}_q$ represents the rank list of related items (e.g., textual documents, information items, answers) retrieved based on q, and $R = \{r_i\}_{i=1}^{n_d}$ consists of the relevance scores r_i between q and each item d_i in the entire item space, n_d is the total number of items. In short, the ranking model aims to select the top K items with the highest relevance to the query as the final ranking result.

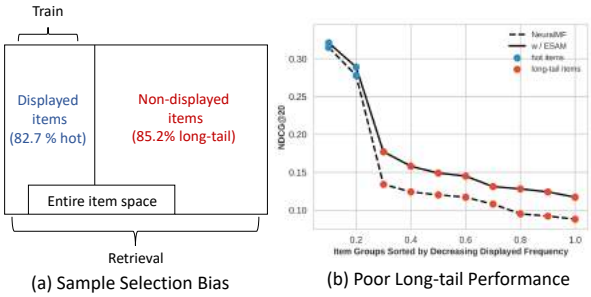
Currently, deep learning-based methods are widely used in the ranking models. For example, [11, 25, 42, 47] in the application of recommender systems, [3, 19] in the application of search systems, and [35] in the application of text retrieval. These methods show a better ranking performance than traditional algorithms [34, 36]. However, these models are mainly trained using only the implicit feedbacks (e.g., clicks and purchases) of the displayed items, but utilized to retrieval items in the entire item space including both displayed and non-displayed items when providing services. We divide the entire item space into hot items and long-tail items by the display frequency. By analyzing two public datasets (MovieLens and CIKM Cup 2016 datasets), we found that 82% of displayed items are hot items, while 85% of non-displayed items are long-tail items. Therefore, as shown in Fig. 1a, the existence of sample selection bias (SSB)[43] causes the model to overfit the displayed items (most are hot items), and cannot accurately predict long-tail items (Fig. 1b). What's worse, this training strategy makes the skew of the model towards popular items [23], which means that these models usually retrieve hot items while ignoring those long-tail items that would be more suitable, especially those that are new arrival. This phenomenon is called "Matthew Effect" [29]. We believe the reason for this phenomenon is that the SSB leads

to the lack of records (*i.e.,* feedbacks) of long-tail items to obtain good feature representations, so the features of long-tail items have an inconsistent distribution compared to that of hot items whose records are adequate. As shown in Fig. 2a, the existence of domain shift [5] means that these ranking models are difficult to retrieve long-tail items, because they always overfit the hot items.

To improve the long-tail performance of ranking models, and increase the diversity of retrieval results, existing methods exploit auxiliary information [43, 44] or auxiliary domains [15, 30], which are not easily accessible. For example, [15, 21] use two sample spaces (*e.g.,* MovieLens and Netflix) to implement knowledge transfer through shared items, and [43] uses a randomly displayed unbiased dataset to fine-tune the model. Given the diversity of model architectures and applications [18, 27], architectural solutions may not generalize well. By following some past works [24], we highlight the importance of *learning good feature representations* for non-displayed items. To achieve this, through considering poor long-tail performance is caused by the domain shift between displayed and non-displayed items and the fact that non-displayed items are unlabeled instances, we adopt unsupervised domain adaptation (DA) technology, which regards displayed and non-displayed items as source and target domains respectively. The DA method allows the application of the model trained with labeled source domain to target domain with limited or missing labels.

Previous DA-based works reduce the domain shift by minimizing some measures of distribution, such as maximum mean discrepancy (MMD) [38], or with adversarial training [23]. For the ranking task, we propose a novel DA method, named *attribute correlation alignment* (ACA). Whether an item is displayed or not, the correlation between its attributes follows the same rules (knowledge). For example, in the e-commerce, the more luxurious the brand of an item, the higher the price (brand and price are item attributes). This rule is the same for displayed and non-displayed items. In a ranking model, each item will be represented as a feature representation through feature extractor, and each dimension of the feature can be regarded as a high-level attribute of the item. Therefore, we believe that the correlation between high-level attributes should follow the same rules in both the displayed and non-displayed spaces. However, the model cannot get the features of non-displayed items well due to the lack of labels, which causes the inconsistency between displayed and non-displayed item feature distributions and makes the above paradigm untenable. Hence, we design the attribute correlation congruence (A2C) to exploit the pair-wise correlation between high-level attributes as distribution.

Although the aforementioned ACA can deal with the distribution inconsistency, there are two critical limitations: (1) learning with the point-wise cross-entropy ignores the spatial structure information, which leads to poor neighborhood relationship in the feature space [47] (Fig. 2b); (2) the target label for non-displayed item is not available, which could easily causes the negative transfer [32] (Fig. 2c) when aligning distributions blindly. Negative transfer is a dilemma that the transfer model performs even worse than non-adaptation model [32]. To address these two obstacles, we propose two novel regularization strategies, *center-wise clustering* and *self-training*, to enhance DA process. We observed that for a query, items with the same feedback are similar, and items with different feedbacks are dissimilar. For example, in the e-commence, when displaying a



(a) Sample Selection Bias (b) Poor Long-tail Performance

Figure 1: (a) The SSB problem. (b) The SSB cause poor long-tail performance on CIKM Cup 2016 dataset.

variety of mobile phones to a user, the user may click all iPhones and ignore others. Therefore, for each query, we can categorize displayed items by the feedback type. The proposed center-wise clustering is devised to make similar items cohere together while dissimilar items separate from each other. This constraint could provide further guidance for domain adaptation, leading to better ranking performance. As to the missing of target label, we assign pseudo-labels of high confidence to target items, and make the model fit these items by self-training. Moreover, when taking into account these pseudo-labels while performing alignment, the model can gradually predict more complex items correctly. In summary, the contributions of this work can be listed as follows: (1) We propose a general entire space adaptation model (ESAM) for ranking models, which exploits the domain adaptation with attribute correlation alignment to improve long-tail performance. ESAM can be easily integrated into most existing ranking frameworks; (2) Two novel yet effective regularization strategies are introduced to optimize the neighborhood relationship and handle the missing of target label for discriminative domain adaptation; (3) We realise ESAM in two typical rank applications: item recommendation and personalized search system. The results on two public datasets, and an industrial dataset collected from Taobao prove the validity of ESAM. Besides, we deploy ESAM to the Taobao search engine, which also produces better performance through an online A/B testing.

2 PROPOSED METHOD

In this section, we first briefly introduce the basic ranking framework named BaseModel. Then, ESAM, which contains the proposed A2C and two regularization strategies, is integrated into the BaseModel for better item feature representation learning in the entire space. Fig. 3 shows the overall framework of ESAM. We consider ESAM without the unlabeled non-displayed item input stream (the red flow in the Fig. 3) as the BaseModel.

2.1 Preliminaries

In this paper, the source domain (displayed items) is denoted as $\mathbb{D}^s$, the target domain (non-displayed items) is denoted as $\mathbb{D}^t$, and the entire item space is denoted as $\mathbb{D} = \mathbb{D}^s \cup \mathbb{D}^t$. Here, the source domain and target domain share the same query set $\mathbb{Q}$. The set of feedback from a query to an item is denoted as $\mathbb{O} = \{q, d, y\}$, where q is a specific query (e.g., keywords, user profiles, questions,

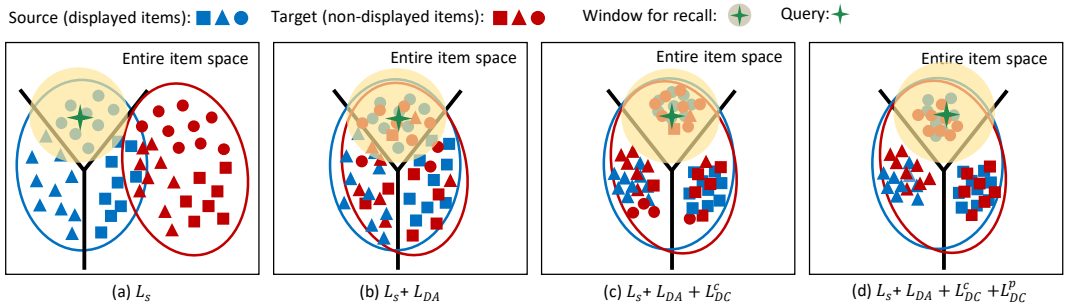


Figure 2: Three shapes represent three feedbacks, the star with a shadow represents a query feature with its retrieval range. (a) Domain shift between displayed and non-displayed item features. (b) Although the domain shift is alleviated, the poor neighborhood relationship makes the model easy to retrieve outliers. (c) The proposed center-wise clustering encourages better manifold structure. However, the class-agnostic ACA may cause negative transfer, *e.g.*, some target circles are aligned to source triangles. (d) ESAM extracts the best item feature representations.

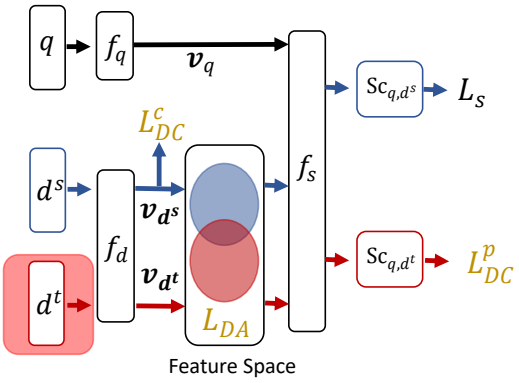


Figure 3: Overview of the ESAM. The red flow represents the non-displayed item stream.

depending on the specific ranking application), d is an item (e.g., textual documents, information items, answers), and y is an implicit feedback. For each q, we assign a labeled source item set $\mathcal{D}^s_q = \{(d^s_j, y^s_j)\}^n_{j=1}$ whose feedback is available, and an unlabeled target item set $\mathcal{D}^t_q = \{d^t_j\}^n_{j=1}$ randomly selected from non-displayed items. The goal of the ranking model is retrieving a ranked item set $\hat{\mathbb{D}}_q$ from $\mathbb{D}$ to maximize query's satisfaction.

2.2 BaseModel

As shown in Fig. 3, the ranking model mainly consists of a query side f_q and an item side f_d. The core task is to map a query q and an item d into a query feature $\boldsymbol{v}_q \in \mathbb{R}^{1\times L}$ and an item feature $\boldsymbol{v}_d \in \mathbb{R}^{1\times L}$ via f_q and f_d respectively, which can be formulated as:

$$\boldsymbol{v}_q = f_q(q), \tag{1}$$

$$\boldsymbol{v}_d = f_d(d). \tag{2}$$

When query and item features are extracted, a scoring function f_s is utilized to calculate the relevance score for a query and an item, which can be formulated as:

$$Sc_{q,d} = f_s(\boldsymbol{v}_q, \boldsymbol{v}_d). \tag{3}$$

Then, in the training stage, the ranking model can be trained with L_s (*e.g.*, point-wise objective function [26, 31], pair-wise objective function [3], list-wise objective function [4, 6]) by using the feedback. In the serving stage, the rank list of the top K items with the highest relevance score can be used as the retrieval result. Note that the implicit feedbacks are only available in most real-world applications. Thus the point-wise objection loss is widely adopted for model training.

Recently, many works design novel f_q and f_d to extract better features. For example, neural matrix factorization (NeuralMF) [19] adopts MLP as f_q and f_d to extract query and item features respectively, deep interest evolution network (DIN) [46] introduces the attention mechanism between f_q and f_d, and behavior sequence transformer (BST) [8] employs the transformer module in f_q to capture short-term interest of query. However, most methods ignore the handling of non-displayed items and these architectural solutions may not generalize well.

2.3 Entire Space Adaptation Model (ESAM)

To improve long-tail performance by suppressing inconsistent distribution between displayed and non-displayed items, ESAM utilizes discriminative domain adaptation with non-displayed items to extract better item features in the entire space. We mainly focus on the improvement on the item side f_d. Specifically, as shown in Fig. 3, ESAM additionally introduces an unlabeled non-displayed item input stream with three constraints on f_d of BaseModel.

• **Domain Adaptation with Attribute Correlation Alignment.**

Due to the existence of the domain shift between displayed items and non-displayed items. Hence, we utilize the DA technology to improve retrieval quality in the entire space. Let $\boldsymbol{D}^s = [\boldsymbol{v}_{d^s_1}; \boldsymbol{v}_{d^s_2}; ...; \boldsymbol{v}_{d^s_n}] \in \mathbb{R}^{n\times L}$ represents the source item feature matrix for $\mathcal{D}^s_q$ and $\boldsymbol{D}^t = [\boldsymbol{v}_{d^t_1}; \boldsymbol{v}_{d^t_2}; ...; \boldsymbol{v}_{d^t_n}] \in \mathbb{R}^{n\times L}$ represents the target item feature matrix for $\mathcal{D}^t_q$, where $\boldsymbol{v}_{d^s_i} \in \mathbb{R}^{1\times L}$ and $\boldsymbol{v}_{d^t_i} \in \mathbb{R}^{1\times L}$ are source item feature and target item feature respectively and

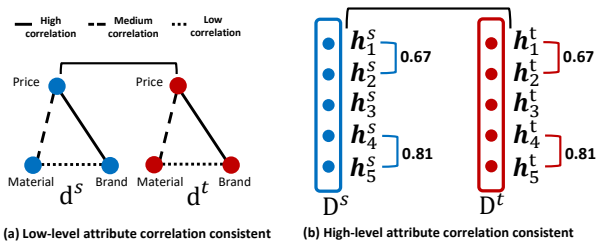


Figure 4: The correlation between low-level item attributes in the source domain is consistent with the target domain. Based on such constraint, we believe that the distributions of source and target domains are aligned when the correlation between high-level attribute vectors of item matrix in the source domain is consistent with the target domain.

are produced by item side f_d. From another perspective, these two matrices can also be seen as a source high-level attribute matrix $\boldsymbol{D}^s = [\boldsymbol{h}_1^s, \boldsymbol{h}_2^s, ..., \boldsymbol{h}_L^s] \in \mathbb{R}^{n \times L}$ and a target high-level attribute matrix $\boldsymbol{D}^t = [\boldsymbol{h}_1^t, \boldsymbol{h}_2^s, ..., \boldsymbol{h}_L^t] \in \mathbb{R}^{n \times L}$, where L is the dimension of the item feature, $\boldsymbol{h}_j^s \in \mathbb{R}^{n \times 1}$ and $\boldsymbol{h}_j^t \in \mathbb{R}^{n \times 1}$ represent the j-th high-level attribute vector in the source item matrix and the target item matrix, respectively.

We believe that the correlation between low-level attributes (*e.g.,* price, brand) of displayed items is consistent with non-displayed items, *e.g.,* as shown in Fig. 4, in the e-commerce, the brand attribute has a high correlation with the price attribute while the correlation between the brand and material attributes is low. This knowledge is consistent in the source and target domains. Therefore, we suggest that high-level attributes should also have the same correlation consistency as low-level attributes. Hence, we define the correlation matrix between high-level attribute vectors in the item matrix as the distribution. Specifically, we propose the attribute correlation congruence (A2C) as the distribution metric and reduce the distance between distributions to ensure that the correlation between high-level attributes is consistent in the source and target domains. Formally, the A2C is formulated as follows:

$$\begin{aligned} L_{DA} &= \frac{1}{L^2} \sum_{(j,k)} (\boldsymbol{h}_j^{s\top} \boldsymbol{h}_k^s - \boldsymbol{h}_j^{t\top} \boldsymbol{h}_k^t)^2 \\ &= \frac{1}{L^2} \| Cov(\boldsymbol{D}^s) - Cov(\boldsymbol{D}^t) \|_F^2, \end{aligned} \tag{4}$$

where $\| \cdot \|_F^2$ denotes squared matrix Frobenius norm. $Cov(\boldsymbol{D}^s) \in \mathbb{R}^{L \times L}$ and $Cov(\boldsymbol{D}^t) \in \mathbb{R}^{L \times L}$ represent the covariance matrices of high-level item attributes, which can be computed as $Cov(\boldsymbol{D}^s) = \boldsymbol{D}^{s\top} \boldsymbol{D}^s$, and $Cov(\boldsymbol{D}^t) = \boldsymbol{D}^{t\top} \boldsymbol{D}^t$. The value at entry (j, k) in the covariance matrix represents the correlation $\boldsymbol{h}_j^\top \boldsymbol{h}_k$ between $\boldsymbol{h}_j$ and $\boldsymbol{h}_k$ as shown in Fig. 4.

We take two batches of samples from source domain, and two batches of samples from target domain, and calculate L_{DA} between each two covariance matrices extracted by BaseModel. As shown in Table 1, the L_{DA} between source and target domains is much larger than that between the same domains, which shows that there indeed exists the distribution shift.

• Center-Wise Clustering for Source Clustering.

Table 1: Analysis of distribution distance (L_{DA}) extracted by BaseModel on the Industrial dataSet.

Source-Source	Target-Target	Source-Target
9e-4	8e-4	0.076

Most of BaseModel only optimize L_s, which is not insensitive to spatial structure in the feature space [47] and make the model cannot learn discriminative features. In our preliminary study, we observe that for a query, items with same feedback are similar, and items with different feedbacks are dissimilar. Inspired by [7], we propose a center-wise clustering to encourage the features of the items with the same feedback to be close together, and the features of the items with different feedbacks to move away from each other. The hinge-based center-wise clustering for each query q can be formulated as:

$$\begin{aligned} L_{DC}^c = &\sum_{j=1}^{n} \max(0, \| \frac{\boldsymbol{v}_{d_j^s}}{\|\boldsymbol{v}_{d_j^s}\|} - \mathbf{c}_q^{y_j^s} \|_2^2 - m_1) \\ &+ \sum_{k=1}^{n_y} \sum_{u=k+1}^{n_y} \max(0, m_2 - \|\mathbf{c}_q^k - \mathbf{c}_q^u\|_2^2) \end{aligned} \tag{5}$$

$$\mathbf{c}_q^k = \frac{\sum_{j=1}^{n} (\delta(y_j^s = Y_k) \cdot \frac{\boldsymbol{v}_{d_j^s}}{\|\boldsymbol{v}_{d_j^s}\|})}{\sum_{j=1}^{n} \delta(y_j^s = Y_k)} \tag{6}$$

where n_y indicates the number of types of feedback (e.g. non-click, click, purchase and so on), and y_j^s indicates the type of feedback on d_j^s. $\mathbf{c}_q^k$ is the class center of the features whose item take the same feedback Y_k by q. And m_1 and m_2 are two distance constraint margins. $\delta(condition) = 1$ if condition is satisfied.

In the right part of Eq. 5, the first term enforces intra-class (the items which have the same feedback) compactness, while the second term in Eq. 5 enforces the inter-class (the items which have different feedbacks) separability. Due to the existence of L_{DA}, source and target domains are highly-related. Therefore, it is reasonable to make the source item feature space more discriminative, such that the target item feature space will become discriminative also by optimizing Eq. 4. In this sense, the ranking model can extract better item features to improve domain adaptation, leading to better long-tail performance.

• Self-Training for Target Clustering.

For a query, we can assign a target pseudo-label (*i.e.,* positive or negative sample) for each non-displayed item. Right now, ESAM can be considered as a class-agnostic DA method that ignores target label information when aligning and may map the target items to the wrong positions (*e.g.,* matching target positive samples to source negative samples). The target samples with pseudo-labels will provide the model with target-discriminative information when aligning. Therefore, we use samples with pseudo-labels for self-training to increase the number of records of non-displayed items and mitigate negative transfer.

Specifically, minimizing the entropy regularization $-p \log p$ favors a low-density separation between classes and increase the

target discrimination [16]. We calculate the relevance score between query and non-displayed item as Eq. 3 and convert it to [0, 1] (e.g. the sigmoid operation). This score can be taken as the probability that the target item is a positive sample for this query. For click feedback, the score represents the probability that the query clicks the item. As shown in Fig. 5, optimizing entropy regularization by gradient descent enforces a score less than 0.4 to gradually approach 0 (negative sample), and a score greater than 0.4 gradually approaches 1 (positive sample). Therefore, this regularization can be seen as a way of self-training which increases the discriminative power between non-displayed items. However, when the regularization used directly, due to the large domain shift between displayed and non-displayed items in early training, the ranking model cannot predict non-displayed items correctly, especially for items whose score is in $[0.4, 0.6]$. Hence, the target samples are easily assigned false labels and stuck into the wrong feedback. To this end, we adopt entropy regularization with a constraint to select reliable target samples for self-training:

$$L^p_{DC} = -\frac{\sum_{j=1}^{n} \delta(Sc_{q,d^t_j} < p_1 | Sc_{q,d^t_j} > p_2) Sc_{q,d^t_j} \log Sc_{q,d^t_j}}{\sum_{j=1}^{n} \delta(Sc_{q,d^t_j} < p_1 | Sc^t_{q,j} > p_2)}, \quad (7)$$

where p_1 and p_2 are two confidence thresholds that are used to select reliable samples with pseudo-labels of high confidence. Sc_{q,d^t_j} is the relevance score between q and d^t_j, which is calculated as Eq. 3 and converted to [0, 1]. $\delta(a|b) = 1$ if condition a or condition b is satisfied. By optimizing Eq. 7, the model is self-trained with negative target items whose Sc_{q,d^t_j} are less than p_1 and positive target items whose Sc_{q,d^t_j} are greater than p_2.

This entropy regularization with a constraint guarantees the correctness of the target label to avoid the negative transfer. Moreover, based on curriculum learning [1], the model may learn target-discriminative information from reliable samples, thereby transforming more samples into reliable samples.

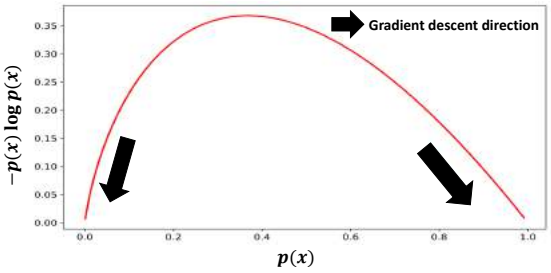


Figure 5: Entropy regularization $-p(x)\log p(x)$, which can be seen as a way to adopt self-training with pseudo-labels.

2.4 Optimization

In summary, the total loss of ESAM can be formulated as:

$$L_{all} = L_s + \lambda_1 L_{DA} + \lambda_2 L^c_{DC} + \lambda_3 L^p_{DC}, \quad (8)$$

where λ_1, λ_2 and λ_3 are hyperparameters which control the impact of corresponding terms. We define Θ as the learnable weights of ESAM, which can be updated end-to-end by back propagation:

$$\Theta^{t+1} = \Theta^t - \eta \frac{\partial(L_s + \lambda_1 L_{DA} + \lambda_2 L^c_{DC} + \lambda_3 L^p_{DC})}{\partial \Theta}, \quad (9)$$

where η is the learning rate.

3 APPLICATIONS

In this section, we apply our ESAM to two specific ranking scenarios: (i) item recommendation, and (ii) personalized search system.

3.1 Item Recommendation

Item recommendation is a typical ranking problem. One of the most important method for recommender systems is collaborative filtering which explores underlying user-item similarity [22]. The recommender system mainly includes four components, including feature composition (q and d), feature extractor (f_q and f_d), scoring function (f_s), and loss function (L_s).

Feature Composition. We mainly adopt three feature groups in item recommendation: user profile, user's behavior, and item profile. Each group is composed of some sparse features: user profile contains the user id, age, gender, etc; user's behavior contains the item ids that the user interacted recently; item profile contains the item id, brand, category, etc. Which feature groups and sparse features are used by the model depends on the design of the feature extractor.

Feature Extractor. The basic feature extractor of neural network-based recommendation systems is MLP [19]. Currently, a variety of novel modules are designed to enhance the feature extraction, such as attention mechanism [46], transformer module [8]. In our experiments, we integrate ESAM into several extractors to prove that it is a general framework to improve long-tail performance.

Scoring Function. We define the scoring function as [41]:

$$Sc_{q,d} = f_s(\boldsymbol{v}_q, \boldsymbol{v}_d) = sigmoid(b_d + \boldsymbol{v}_q \boldsymbol{v}_d^\top), \quad (10)$$

where b_d is the bias term for item d, $\boldsymbol{v}_q, \boldsymbol{v}_d \in \mathbb{R}^{1\times L}$ are the features of user q and item d extracted from f_q and f_d respectively.

Loss Function. We treat click as feedback. Hence, for a query q, whether an item d will be clicked can be regarded as a binary classification problem. We apply the point-wise cross-entropy objective function:

$$L_s = -\frac{1}{n}\sum_{i=1}^{n}(y_{q,d_i} \log Sc_{q,d_i}) + (1 - y_{q,d_i}) \log(1 - Sc_{q,d_i}), \quad (11)$$

where y_{q,d_i} is a binary variable indicating whether the query clicked on the item d_i ($y_{q,d_i} = 1$ if query q clicked on item d_i, and $y_{q,d_i} = 0$ otherwise), n is the number of displayed items for query q.

The above model is trained using only L_s, which considers only the displayed items. As shown in Sec. 2 and Fig. 3, to mitigate domain shift for performance improvement of model, we randomly assign non-displayed items to each query in every training epoch and integrate ESAM into the model by introducing additional non-display item input stream with three constraints on item side f_d.

3.2 Personalized Search System

The personalized search system investigated in this work is to identify all possible items at which the user could be interested based on the keyword issued by the user. Hence, the integration of ESAM, feature extractor, and loss function are the same as item recommendation, and only the feature composition and scoring function are different. For the feature composition, personalized search systems introduce an additional keyword feature group. The keyword is transformed into a vector by averaging word vectors of the keyword. For the scoring function, we define the cosine similarity as scoring function [41]:

$$Sc_{q,d} = f_s(\boldsymbol{v}_q, \boldsymbol{v}_d) = \frac{\boldsymbol{v}_q \boldsymbol{v}_d^\top}{\|\boldsymbol{v}_q\| \cdot \|\boldsymbol{v}_d\|}. \tag{12}$$

4 EXPERIMENTS

In this section, we conduct our experiments corresponding to two real-world applications of ESAM as discussed above, i.e., item recommendation and personalized search system.

4.1 Datasets

• Item Recommendation.

MovieLens-1M [1]**:** The dataset consists of 1 million movie rating records over thousands of movies and users. We first binarize ratings to simulate the CTR prediction task, i.e., the ratings greater than 3 are displayed positive samples, and the rest are displayed negative samples, which is common to this problem setting [40]. The features we used are the user id, age, gender, occupation of the user, user behavior in query side, and movie id, release year, genres of the movie in item side. At each training epoch, we randomly add 10 non-displayed movies similar to the displayed movies (*i.e.,* $n = 10$). The similar movies are the non-displayed movies whose genres are the same as displayed movies. We divide data randomly into the training, validation, and test sets by a ratio of 8: 1: 1.

• Personalized Search System.

CIKM Cup 2016 Personalized E-Commerce Search[2]**:** The Personalized E-commerce Search Challenge releases a dataset provided by DIGINETICA. Each item contains item id, category id, title, and description, each query consists of the user id, user's gender, age, history behavior, and keyword id (transformed by a list of words) issued by the user, and each record contains a query, 10 displayed items for this query and the items that the query clicked. In addition, we adopt the same strategy as MovieLens-1M to assign the non-displayed items for each record. For training, validation and test sets construction, we apply a 8:1:1 random splitting.

Industrial Dataset of Taobao: We collect the logs of Taobao for one week in June 2019. In this dataset, all the features used, the way of obtaining non-displayed items, and the organization of the records are the same as CIKM Cup 2016 dataset. In the experiments, we split the dataset into three parts: training set (records from the first day to the fifth day), validation set (records from the sixth day) and test set (records from the seventh day). The basic statistics of these three datasets are reported in Table 2.

[1]http://www.grouplens.org/datasets/movielens/
[2]https://competitions.codalab.org/competitions/11161/

Table 2: Statistics of experimental datasets. 'M' represents the million and 'B' represents the billion.

Dataset	#Query	#Item	#Record
MovieLens-1M dataset	6040	3883	1M
CIKM Cup 2016 dataset	5.7M	13.4M	24.5M
Taobao Industrial dataset	114.6M	223.2M	11.9B

4.2 Baselines and Evaluation Metrics

To verify that ESAM is a general framework, we integrate it to some single-domain ranking approaches based on neural network. Note that, The feature extractor (f_q and f_d) is consistent in personalized search and recommendation applications, only the feature composition (q and d), and scoring function (f_s) depend on the application. The specific differences can be seen in Sec. 3. We also compare some DA-based ranking models and "missing not random" methods to show the superiority of our proposed ESAM.

Specifically, the single-domain methods are: NeuralMF [19], RALM [29], YoutubeNet [12], BST [8]. The DA-based methods are: DARec [44], DA learning to rank framework (DALRF) [38]. A "missing not random" method is: unbiased imputation model (UIM)[43]. For methods based on DA to improve long-tail performance (e.g., DALRF, DARec), we only replace the original DA constraint with ESAM, and keep other modules unchanged for comparison. Note that the "missing not random" method requires an unbiased dataset. Here the unbiased dataset collected through deploying a uniform policy on a small percentage of our online traffic. We only replace the constraints that use the unbiased dataset with ESAM for fair comparison. Three standard evaluation metrics, NDCG (Normalized Discounted Cumulative Gain), Recall and MAP (Mean Average Precision) are used for performance comparison.

4.3 Implementation Details

All methods are implemented with TensorFlow and trained with Adam optimizer. All methods are trained five times and the average results are reported. For the hyperparameters λ_1, λ_2 and λ_3, we search the best ones from $\{0.01, 0.1, 0.3, 0.5, 0.7, 1, 10\}$ and set $\lambda_1 = 0.7$, $\lambda_2 = 0.3$ and $\lambda_3 = 0.5$. For other hyperparameters, we set learning rate $\eta = 1e-4$, batch size to be 256, feature dimension $L = 128$, distance constraints $m_1 = 0.2$ and $m_2 = 0.7$, and confidence thresholds $p_1 = 0.2$ and $p_2 = 0.8$. For the hyperparameters of comparison methods, we use a grid search based on performance on the validation set. Note that, we assign each query 10 source items $\mathcal{D}_q^s$ and 10 target items $\mathcal{D}_q^t$ (*i.e.,* $n = 10$).

4.4 Performance Comparison

We divide the entire item space into hot and long-tail items by the display frequency. Through the inflection point shown in Fig. 1b, we define the top 20% items as hot items and the rest as long-tail items.

Public Datasets: Table 3 and Table 4 show the performance of baselines without and with ESAM on the Movielens-1M dataset and CIKM Cup 2016 dataset, respectively. We make the following observations from the results. (1) The performance of each model for hot items is always much higher than that of long-tail items,

Table 3: Performance comparison between the methods without and with ESAM on MovieLens-1M dataset. "Hot" represents hot items in the test set, "Long-tail" represents long-tail items in the test set, and "Entire" represents all items in the test set. The best results are highlighted in boldface. The improvements over the baseliens are statistically significance at 0.05 level.

Group	Method	w/o ESAM									w/ ESAM								
		NDCG@20			Recall@20			MAP			NDCG@20			Recall@20			MAP		
		Hot	Long-tail	Entire	Hot	Long-tail	Entire	Hot	Long-tail	Entire	Hot	Long-tail	Entire	Hot	Long-tail	Entire	Hot	Long-tail	Entire
Single Domain	NeuralMF [19]	0.2296	0.0508	0.2017	0.2660	0.0732	0.2453	0.1953	0.0486	0.1813	**0.2431**	**0.0812**	**0.2238**	**0.2753**	**0.1005**	**0.2644**	**0.2014**	**0.0674**	**0.1962**
	YoutubeNet [12]	0.2519	0.0983	0.2385	0.2976	0.1142	0.2691	0.2138	0.0874	0.2057	**0.2671**	**0.1264**	**0.2613**	**0.3125**	**0.1369**	**0.2947**	**0.2282**	**0.1108**	**0.2270**
	RALM [29]	0.2658	0.1237	0.2584	0.3081	0.1294	0.2883	0.2259	0.1037	0.2236	**0.2784**	**0.1493**	**0.2738**	**0.3249**	**0.1455**	**0.3057**	**0.2376**	**0.1271**	**0.2412**
	BST[8]	0.2816	0.1371	0.2835	0.3292	0.1373	0.3104	0.2429	0.1230	0.2481	**0.2935**	**0.1627**	**0.2984**	**0.3453**	**0.1516**	**0.3295**	**0.2579**	**0.1451**	**0.2608**
Domain Adaptation	DALRF [38]	0.2731	0.1503	0.2467	0.3092	0.1424	0.2953	0.2376	0.1431	0.2359	**0.2816**	**0.1584**	**0.2503**	**0.3158**	**0.1507**	**0.3012**	**0.2431**	**0.1479**	**0.2418**
	DARec [44]	0.2748	0.1565	0.2497	0.3124	0.1483	0.2996	0.2418	0.1488	0.2401	**0.2824**	**0.1631**	**0.2614**	**0.3182**	**0.1549**	**0.3079**	**0.2503**	**0.1579**	**0.2495**

Table 4: Performance comparison between methods without and with ESAM on CIKM Cup 2016 dataset. The best results are highlighted in boldface. The improvements over the baseliens are statistically significance at 0.05 level.

Group	Method	w/o ESAM									w/ ESAM								
		NDCG@20			Recall@20			MAP			NDCG@20			Recall@20			MAP		
		Hot	Long-tail	Entire	Hot	Long-tail	Entire	Hot	Long-tail	Entire	Hot	Long-tail	Entire	Hot	Long-tail	Entire	Hot	Long-tail	Entire
Single Domain	NeuralMF [19]	0.1840	0.0521	0.1296	0.3145	0.0947	0.2265	0.1596	0.0559	0.1158	**0.2043**	**0.1085**	**0.1543**	**0.3420**	**0.1495**	**0.2652**	**0.1718**	**0.0857**	**0.1409**
	YoutubeNet [12]	0.2196	0.0942	0.1572	0.3481	0.1422	0.2594	0.1928	0.0974	0.1437	**0.2347**	**0.1295**	**0.1821**	**0.3702**	**0.1735**	**0.2859**	**0.2139**	**0.1266**	**0.1703**
	RALM [29]	0.2341	0.1195	0.1794	0.3592	0.1583	0.2739	0.2164	0.1219	0.1682	**0.2457**	**0.1408**	**0.2049**	**0.3796**	**0.1841**	**0.2985**	**0.2295**	**0.1451**	**0.1879**
	BST[8]	0.2483	0.1392	0.2067	0.3824	0.1793	0.3021	0.2319	0.1436	0.1928	**0.2607**	**0.1586**	**0.2241**	**0.3976**	**0.2058**	**0.3249**	**0.2465**	**0.1620**	**0.2194**
Domain Adaptation	DALRF [38]	0.2213	0.1456	0.1861	0.3684	0.1921	0.2973	0.2024	0.1583	0.1895	**0.2362**	**0.1578**	**0.2014**	**0.3805**	**0.2052**	**0.3097**	**0.2152**	**0.1735**	**0.2008**
	DARec [44]	0.2259	0.1502	0.1923	0.3705	0.1982	0.3018	0.2079	0.1604	0.1936	**0.2348**	**0.1683**	**0.2091**	**0.3859**	**0.2053**	**0.3084**	**0.2137**	**0.1762**	**0.2041**

Table 5: Cold-start performance of BST [8] w/o and w/ ESAM.

Dataset	w/o ESAM			w/ ESAM		
	NDCG@20	Recall@20	MAP	NDCG@20	Recall@20	MAP
MovieLens-1M	0.0319	0.0621	0.0485	**0.0937**	**0.1233**	**0.0949**
CIKM Cup 2016	0.0436	0.0759	0.0531	**0.0982**	**0.1349**	**0.1023**

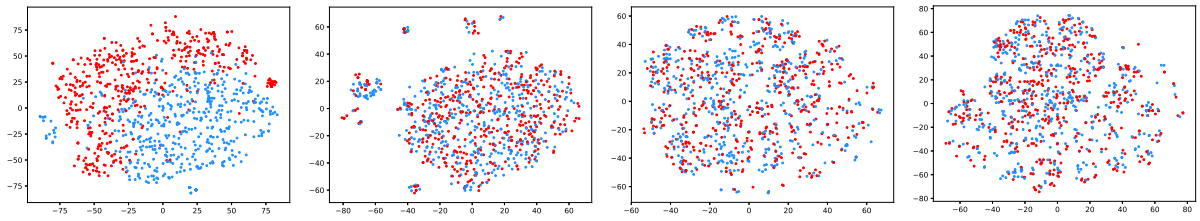
Table 6: Ablation study on Industrial dataset.

Method	Recall@1k			Recall@3k		
	Hot	Long-tail	Entire	Hot	Long-tail	Entire
BaseModel (BM)	0.1842	0.1067	0.1298	0.2776	0.1845	0.2149
BM+L_{DA}	0.1921	0.1364	0.1453	0.2805	0.2162	0.2409
ESAM w/o L_{DA}	0.1976	0.1298	0.1439	0.2873	0.2043	0.2318
ESAM w/o L^c_{DC}	0.1926	0.1392	0.1507	0.2812	0.2195	0.2490
ESAM w/o L^p_{DC}	0.1975	0.1413	0.1532	0.2886	0.2207	0.2504
ESAM	**0.1982**	**0.1567**	**0.1637**	**0.2894**	**0.2383**	**0.2621**
UIM[43]	0.1892	0.1456	0.1543	0.2831	0.2349	0.2675
UIM$_{ESAM}$	**0.1931**	**0.1521**	**0.1608**	**0.2905**	**0.2417**	**0.2759**

which indicates that the distributions of hot and long-tail items are inconsistent. (2) The single-domain baselines with ESAM outperform that without ESAM. For the personalized search application, the baselines with ESAM achieve the average absolute NDCG@20 / Recall@20 / MAP gain of 1.4% / 2.1% / 1.5% in the hot space, 3.3% / 3.4% / 2.5% in the long-tail space and 2.3% / 2.8% /2.3% in the entire space; for the item recommendation application, the baselines with ESAM achieve the average absolute NDCG@20 / Recall@20 / MAP gain of 1.3% / 1.4% / 1.2% in the hot space, 2.7% / 2.0% / 2.2% in the long-tail space and 1.8% / 1.9% / 1.7% in the entire space. The significant improvement in the long-tail space proves that the proposed ESAM is effective to alleviate the domain shift problem for long-tail items, making baselines learn better features in the entire space for different datasets and applications. (3) We find that ESAM outperforms other DA methods (*i.e.,* DALRF, DARec). The results confirm that the discriminative domain adaptation we design, which considers the correlation between high-level attributes and adopts center-wise clustering and self-training to improve source discrimination and target discrimination, can effectively transfer more discriminative knowledge.

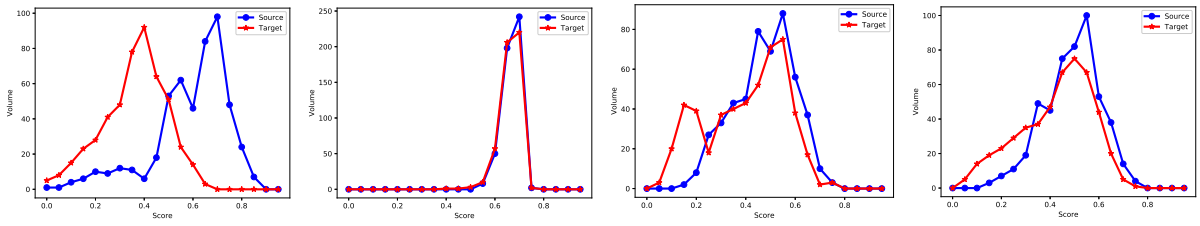
Cold-Start Performance: We randomly select 20% of the records in the test set and delete from the training set all records whose displayed items containing these selected test records. We employ BST [8] as the BaseModel since it has the best performance on the public dataset. As shown in Table 5, we find that BST is difficult to solve the cold-start problem due to the lack of these items in the displayed space. While ESAM can significantly improve the cold-start performance because it introduces non-displayed items which contain the cold-start items, to enhance the feature learning.

Industrial Dataset of Taobao: Because the Taobao Industrial dataset is very large, we adopt Recall@1k and Recall@3k as the metrics and use the BST [8] that has the best performance on the public dataset as the BaseModel for ablation study. Several observations can be made based on Table 6. (1) Comparing the method w/o L_{DA} and that w/ L_{DA} (*e.g.,* BM and BM+L_{DA} or ESAM w/o L_{DA} and ESAM), we find that the A2C can greatly improve the performance of the model in the long-tail space (*e.g.,* the average gain of +2.73% or +3.05%), which proves that poor long-tail performance can be solved by aligning the distribution of source and target domains. (2) Compared with the model w/o L^c_{DC} (*e.g.,* BM+L_{DA} or ESAM w/o L^c_{DC}), the performance of the model w/ L^c_{DC} (*e.g.,* ESAM w/o L^p_{DC} or ESAM) is particularly prominent in the hot space (*e.g.,* the average gain of +0.6% or +0.7%) by adopting L^c_{DC} to optimize the source spatial structure. Besides, L^c_{DC} makes the target spatial structure



(a) L_s (b) $L_s + L_{DA}$ (c) $L_s + L_{DA} + L^c_{DC}$ (d) $L_s + L_{DA} + L^c_{DC} + L^p_{DC}$

Figure 6: T-SNE visualization on Industrial dataset. Red and blue points represent target and source samples respectively.



(a) L_s (b) $L_s + L_{DA}$ (c) $L_s + L_{DA} + L^c_{DC}$ (d) $L_s + L_{DA} + L^c_{DC} + L^p_{DC}$

Figure 7: Score distribution on Industrial dataset. Red and blue represent the target and source score distributions respectively.

aligned with better intra-class compactness and inter-class separability, thus improving the lone-tail performance (*e.g.,* the average gain of +0.5% or +1.8%). (3) The combination of these three constraints (*i.e.,* ESAM) yields the best performance, which indicates the necessity of the three regularization terms we design. (4) The ESAM performs better than UIM, which illustrates that ESAM can suppress the "Matthew Effect" to improve long-tail performance.

4.5 Feature Distribution Visualization

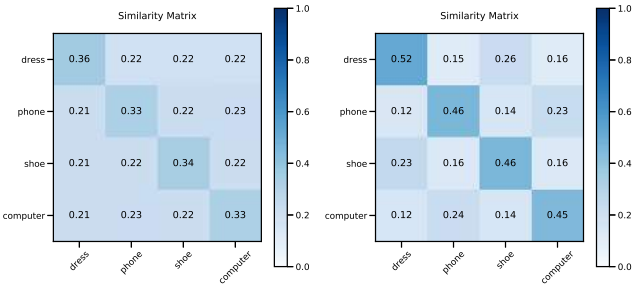
To illustrate the existence of domain shift and the validity of the proposed ESAM, we randomly select 2,000 items and visualize the features output from f_d using t-SNE [13]. As shown in Fig. 6, we can make intuitive observations. (1) As shown in Fig. 6a, there is a domain gap between distributions of source and target features. This gap makes the model that fits source items cannot be applied in the entire space where has a large number of non-displayed items. (2) We find that the integration of L_{DA} can significantly reduce the disparity between distributions (Fig. 6b), which proves that the correlation between item high-level attributes can reflect the distribution of domain well. (3) The feature space will have better manifold structure by adopting discriminative clustering. L^c_{DC} enforces the shared feature space with better intra-class compactness and inter-class separability (Fig. 6c). Also L^p_{DC} encourages further more discriminative alignment. Compared to Fig. 6c, L^p_{DC} expands the abscissa to [-60,80] and the ordinate to [-80,80] (Fig. 6d). In a word, ESAM extracts domain-invariant and discriminative features by discriminative domain adaptation, which makes the model robust to long-tail items and can retrieve more personalized and diverse items to users.

4.6 Score Distribution

To further illustrate that ESAM can effectively optimize long-tail performance. We randomly select 500 combinations $(\boldsymbol{v}_q, \boldsymbol{v}_d)$ from source and target domains respectively, to visualize the score distribution. As shown in Fig. 7, the abscissa is the score which can be calculated as Eq. 12 and the ordinate is the number of combinations whose scores fall within the interval. As can be seen in Fig. 7a, the model tends to give extremely low scores for non-displayed items, making these items difficult to be retrieved, while L_{DA} encourages the source and target domains to have the same score distribution, which increases the exposure probability of long-tail items. However, the aligned score distribution is too concentrated (Fig. 7b), which may result in retrieving irrelevant items. To solve this problem, the proposed discriminative clustering encourages the score distribution to be aligned with better discriminability (Fig. 7c and Fig. 7d). As shown in Fig. 7c, there is an abnormal spike (around score = 0.15) of the target plot, we think this is due to the negative transfer (the positive samples are aligned to negative samples) caused by ignoring the target label during alignment. As expected, self-training with target reliable pseudo-labels can solve this problem to a certain extent.

4.7 Similarity Matrix

To prove that ESAM can learn better item feature representations. We randomly selected 1, 000 items from each category (dress, phone, shoe, and computer) in the entire item space of Industrial dataset. We build two groups of items by equally spliting the item samples (*i.e.,* 500) for each category. For each group, we calculate its category center as the average of the item feature representations. Then, we can obtain the similarity matrix through the cosine similarity for each center pair. As shown in Fig. 8, compared with BaseModel,



(a) BaseModel **(b) ESAM**

Figure 8: Similarity matrix on Industrial dataset.

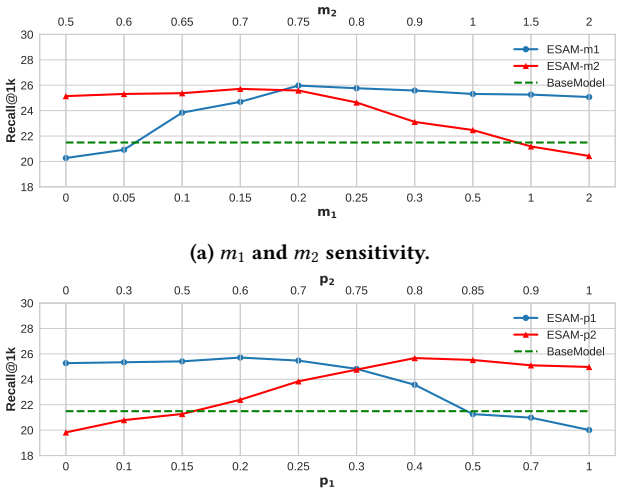
ESAM has greater intra-class similarity and smaller inter-class similarity, which suggests that ESAM has a more discriminative neighborhood relationship. Besides, the similarity matrix extracted by ESAM can better reflect the real situation, *i.e.,* the similarity between dress and shoe is higher than that of dress and phone, which cannot be displayed in the BaseModel.

4.8 Parameter Sensitivity

To study the effects of distance constraints $m1$ and $m2$ of L^c_{DC}, we vary their value as $m_1 \in \{0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.5, 1, 2\}$ and $m_2 \in \{0.5, 0.6, 0.65, 0.7, 0.75, 0.8, 0.9, 1, 1.5, 2\}$. Fig. 9a shows bell-shaped curves, which indicates that an appropriately intra-class and inter-class distance constraints can effectively optimize the neighborhood relationship to improve retrieval performance. For the confidence thresholds p_1 and p_2 of L^p_{DC}, as shown in Fig. 9b, we find that assigning pseudo-labels to samples with low confidence (p_1 is too large or p_2 is too small) for self-training would lead to collapse of the model due to a large number of labels are false ones. While increasing the confidence threshold can improve the performance of the model, which indicates that entropy regularization with a proper constraint is effective to select reliable target samples for better domain adaptation and final performance.

4.9 Online Experiments

We deploy the proposed method to Taobao search engine for personalised searching and conduct the online A/B testing experiment. To make a fair comparison, except for whether the model is integrated with ESAM, other variables are consistent, including user pool, item pool, feature composition, etc. Compared with BaseModel, ESAM achieves 0.85% CTR, 0.63% collection rate, 0.24% CVR and 0.4% GMV gain in seven days. The "collection" represents the behavior of adding the item to the cart. Note that, the performance improvement is significant enough for our business due to the search engine in Taobao serves billions of users and generates more than ten million GMV per day. We attribute this to the improved search capability for long-tail items by introducing an unlabeled non-displayed input stream to reduce the distribution discrepancy between displayed and non-displayed items. In short, the online experiments validate that ESAM can significantly advance the whole search engine.



(a) m_1 and m_2 sensitivity.

(b) p_1 and p_2 sensitivity.

Figure 9: Parameter sensitivity of L^c_{DC}, and confidence thresholds of L^p_{DC} on Industrial dataset.

5 RELATED WORK

5.1 Neural Network-Based Ranking Model

Recently, many works have been done to use deep neural networks for many ranking based applications. A significant development in the ranking model is deep learning to rank (LTR) [3, 11, 19, 25, 35, 47]. To tackle this problem, some methods utilize DA techniques, such as maximum mean discrepancy (MMD) [38], adversarial training [23], to alleviate inconsistent distributions in the source and target domains. Also, some methods [2, 43] introduce an unbiased dataset obtained by an unbiased system (*i.e.,* randomly selecting items from the entire item pool to a query) to train an unbiased model. Besides, some methods introduce auxiliary information [43, 44] or auxiliary domains [15, 30] to obtain more long-tail information. Unlike previous approaches, ESAM joints domain adaptation and non-displayed items to improve long-tail performance without any auxiliary information and auxiliary domains. Moreover, we design a novel DA technique, named attribute correlation alignment, which regards the correlation between item high-level attributes as knowledge to transfer.

5.2 Discriminative Domain Adaption

Domain adaptation, which transfers knowledge from a large number of labeled source samples to target samples with missing or limited labels for target performance improvement, has been widely studied recently. These DA methods learn the domain-invariant feature by embedding the adaptation layer for moment matching, such as maximum mean discrepancy (MMD) [10], correlation alignment (CORAL) [9, 37] and center moment discrepancy (CMD) [45], or integrating a domain discriminator for adversarial training, such as domain adversarial neural network (DANN) [14] and adversarial discriminative domain adaptation (ADDA) [39]. Some previous works pay efforts to learn more discriminative features for performance improvement, *e.g.,* the contrastive loss [17] and the center

loss [28]. These methods have been adopted in many applications, such as face recognition and person re-identification, etc. Inspired by these methods, we propose to perform domain adaptation to lean item features with better discriminative power in entire space.

6 CONCLUSION

In this paper, we propose ESAM to improve long-tail performance with discriminative domain adaptation by introducing non-displayed items. To the best of our knowledge, this is the first work to adopt domain adaptation with non-diplayed items for ranking model. It is worth mentioning that ESAM is a general framework, which can be easily integrated into many existing ranking models. The offline experiments on two public datasets and a Taobao industrial dataset prove that ESAM can be integrated into the existing SOTA baselines to improve retrieval performance, especially in the long-tail space. Online experiments further demonstrate the superiority of ESAM at Taobao search engine. Furthermore, we also verify the necessity of each constraint by ablation studies.

ACKNOWLEDGMENTS

This work was supported by Alibaba Group through Alibaba Innovative Research Program and National Natural Science Foundation of China (No. 61872278). Chenliang Li is the corresponding author.

REFERENCES

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*. 41–48.
[2] Stephen Bonner and Flavian Vasile. 2018. Causal embeddings for recommendation. In *RecSys*. 104–112.
[3] Christopher Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N Hullender. 2005. Learning to rank using gradient descent. In *ICML*. 89–96.
[4] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.
[5] J Quiñonero Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. 2009. Dataset shift in machine learning.
[6] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *ICML*. 129–136.
[7] Chao Chen, Zhihong Chen, Boyuan Jiang, and Xinyu Jin. 2019. Joint domain alignment and discriminative feature learning for unsupervised deep domain adaptation. In *AAAI*. 3296–3303.
[8] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior Sequence Transformer for E-commerce Recommendation in Alibaba. *CoRR* abs/1905.06874 (2019).
[9] Zhihong Chen, Chao Chen, Zhaowei Cheng, Ke Fang, and Xinyu Jin. 2019. Selective Transfer with Reinforced Transfer Network for Partial Domain Adaptation. *arXiv preprint arXiv:1905.10756* (2019).
[10] Zhihong Chen, Chao Chen, Xinyu Jin, Yifu Liu, and Zhaowei Cheng. 2019. Deep joint two-stream Wasserstein auto-encoder and selective attention alignment for unsupervised domain adaptation. *Neural Computing and Applications* (2019), 1–14.
[11] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *DLRS*. 7–10.
[12] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *RecSys*. 191–198.
[13] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*. 647–655.
[14] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research* 17, 1 (2016), 2096–2030.
[15] Chen Gao, Xiangning Chen, Fuli Feng, Kai Zhao, Xiangnan He, Yong Li, and Depeng Jin. 2019. Cross-domain Recommendation Without Sharing User-relevant Data. In *WWW*. 491–502.
[16] Yves Grandvalet and Yoshua Bengio. 2005. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*. 529–536.
[17] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *CVPR*. 1735–1742.
[18] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.
[19] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*. 2333–2338.
[20] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *KDD*. 133–142.
[21] Heishiro Kanagawa, Hayato Kobayashi, Nobuyuki Shimizu, Yukihiro Tagami, and Taiji Suzuki. 2019. Cross-domain recommendation via deep domain adaptation. In *European Conference on Information Retrieval*. 20–29.
[22] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.
[23] Adit Krishnan, Ashish Sharma, Aravind Sankar, and Hari Sundaram. 2018. An Adversarial Approach to Improve Long-Tail Performance in Neural Collaborative Filtering. In *CIKM*. 1491–1494.
[24] Himabindu Lakkaraju, Jon Kleinberg, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. 2017. The selective labels problem: Evaluating algorithmic predictions in the presence of unobservables. In *KDD*. 275–284.
[25] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Pipei Huang, Huan Zhao, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall. *CoRR* abs/1904.08030 (2019).
[26] Ping Li, Qiang Wu, and Christopher J Burges. 2008. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in neural information processing systems*. 897–904.
[27] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *WWW*. 689–698.
[28] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. 2016. Large-margin softmax loss for convolutional neural networks.. In *ICML*. 7.
[29] Yudan Liu, Kaikai Ge, Xu Zhang, and Leyu Lin. 2019. Real-time Attention Based Look-alike Model for Recommender System. *CoRR* abs/1906.05022 (2019).
[30] Tong Man, Huawei Shen, Xiaolong Jin, and Xueqi Cheng. 2017. Cross-Domain Recommendation: An Embedding and Mapping Approach.. In *IJCAI*. 2464–2470.
[31] Ramesh Nallapati. 2004. Discriminative models for information retrieval. In *SIGIR*. 64–71.
[32] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.
[33] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.
[34] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. 2001. Item-based collaborative filtering recommendation algorithms. *WWW* (2001), 285–295.
[35] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*. 373–382.
[36] Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *KDD*. 650–658.
[37] Baochen Sun and Kate Saenko. 2016. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*. 443–450.
[38] Brandon Tran, Maryam Karimzadehgan, Rama Kumar Pasumarthi, Michael Bendersky, and Donald Metzler. 2019. Domain Adaptation for Enterprise Email Search. *CoRR* abs/1906.07897 (2019).
[39] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *CVPR*. 7167–7176.
[40] Maksims Volkovs and Guang Wei Yu. 2015. Effective latent models for binary feedback in recommender systems. In *SIGIR*. 313–322.
[41] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *SIGIR*. 515–524.
[42] Xiangnan He Chenxu Wang Meng Wang Yan Li Yang Zhang, Fuli Feng and Yongdong Zhang. 2020. How to Retrain a Recommender System? A Sequential Meta-Learning Approach. In *SIGIR*.
[43] Bowen Yuan, Jui-Yang Hsia, Meng-Yuan Yang, Hong Zhu, Chih-Yao Chang, Zhenhua Dong, and Chih-Jen Lin. 2019. Improving Ad Click Prediction by Considering Non-displayed Events. In *CIKM*. 329–338.
[44] Feng Yuan, Lina Yao, and Boualem Benatallah. 2019. DARec: Deep Domain Adaptation for Cross-Domain Recommendation via Transferring Rating Patterns. *CoRR* abs/1905.10760 (2019).
[45] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. 2017. Central moment discrepancy (cmd) for domain-invariant representation learning. *CoRR* abs/1702.08811 (2017).
[46] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *AAAI*. 5941–5948.
[47] Ziwei Zhu, Jianling Wang, and James Caverlee. 2019. Improving Top-K Recommendation via JointCollaborative Autoencoders. In *WWW*. 3483–3482.

ATBRG: Adaptive Target-Behavior Relational Graph Network for Effective Recommendation

Yufei Feng[1*], Binbin Hu[2*], Fuyu Lv[1], Qingwen Liu[1], Zhiqiang Zhang[2], Wenwu Ou[1]
[1]Alibaba Group, Hangzhou, China
[2]Ant Financial Services Group, Hangzhou, China
fyf649435349@gmail.com,bin.hbb@antfin.com,{fuyu.lfy,xiangsheng.lqw}@alibaba-inc.com,lingyao.zzq@antfin.com,santong.oww@taobao.com

ABSTRACT

Recommender system (RS) devotes to predicting user preference to a given item and has been widely deployed in most web-scale applications. Recently, knowledge graph (KG) attracts much attention in RS due to its abundant connective information. Existing methods either explore independent meta-paths for user-item pairs over KG, or employ graph neural network (GNN) on whole KG to produce representations for users and items separately. Despite effectiveness, the former type of methods fails to fully capture structural information implied in KG, while the latter ignores the mutual effect between target user and item during the embedding propagation. In this work, we propose a new framework named **A**daptive **T**arget-**B**ehavior **R**elational **G**raph network (ATBRG for short) to effectively capture structural relations of target user-item pairs over KG. Specifically, to associate the given target item with user behaviors over KG, we propose the graph connect and graph prune techniques to construct adaptive target-behavior relational graph. To fully distill structural information from the sub-graph connected by rich relations in an end-to-end fashion, we elaborate on the model design of ATBRG, equipped with relation-aware extractor layer and representation activation layer. We perform extensive experiments on both industrial and benchmark datasets. Empirical results show that ATBRG consistently and significantly outperforms state-of-the-art methods. Moreover, ATBRG has also achieved a performance improvement of 5.1% on CTR metric after successful deployment in one popular recommendation scenario of Taobao APP.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Recommender System; Knowledge Graph; Graph Neural Network

[*] Yufei Feng and Binbin Hu contribute equally.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGIR '20, July 25–30, 2020, Virtual Event, China
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401428

ACM Reference Format:
Yufei Feng, Binbin Hu, Fuyu Lv, Qingwen Liu, Zhiqiang Zhang, Wenwu Ou. 2020. ATBRG: Adaptive Target-Behavior Relational Graph Network for Effective Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401428

1 INTRODUCTION

In the era of information overload, recommender system (RS), which aims to match diverse user interests with tremendous resource items, are widely deployed in various online services, including e-commerce [17, 27, 32], social media [3, 33] and news [4, 21]. Traditional recommendation methods, *e.g.,* matrix factorization [11], mainly learn an effective preference prediction function using historical user-item interaction records. Despite effectiveness, these methods suffer from cold-start problem due to data sparsity. With the rapid development of web services, some approaches [8, 9] are proposed to incorporate various auxiliary data for improving recommendation performance.

Recently, knowledge graph (KG), which is flexible to model comprehensive auxiliary data, has attracted increasing attention in RS [2, 20, 22–25]. Generally, KG stores external heterogeneous knowledge in the ternary form $\langle head\ entity, relation, tail\ entity \rangle$, corresponding to attribute (*e.g.,* $\langle Blouse, Category, Shirt \rangle$ or relationship (*e.g.,* $\langle Shirt, Audience, Girl \rangle$) of entities. Due to its abundant information, current recommender systems mainly aim to incorporate KG to enrich representations of users and items and promote the interpretability of recommendations.

Though with great improvements, it remains challenging to effectively integrate such heterogeneous information for recommendation. Roughly speaking, state-of-the-art KG based recommendation methods mainly fall into two groups, path based and graph neural network (GNN) based methods. Path based methods [24] infer user preference by exploring multiple meta-paths for target user-item pairs over KG, which typically requires domain knowledge. More importantly, this type of methods ignores rich structural information implied in KG, and thus cannot sufficiently characterize the underlying relationships between given target user and item. As illustrated in Fig. 1, these methods essentially overlook the strong relationships between *Blouse* and *Dress*, since each extracted path is modeled independently.

Inspired by the recently emerging graph neural networks, several GNN based methods [22, 23] have been proposed and provide strong performance by explicitly modeling high-order connectivities in KG. Nevertheless, these methods still suffer from three

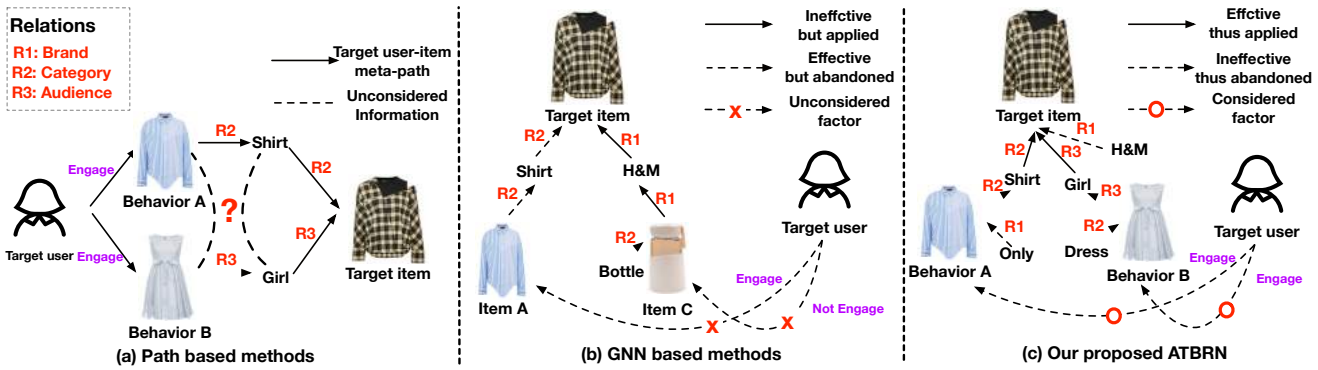


Figure 1: The comparison of our proposed framework ATBRG with previous models. (a) & (b) indicate the limitations of path based and GNN based methods, respectively, while (c) shows the superiority of ATBRG .

limitations: (**L1**) These methods mainly apply GNN to enrich the representation of target user and item separately by aggregating their own original neighbors in the KG, and thus fail to capture their mutual influence during the procedure of information aggregation. As shown in Fig. 1 (b), current GNN based methods tend to produce representation for target item through aggregating its neighbors without considering target user's interests (history behaviors). Subsequently, some unnecessary information (*i.e., Cup*) is involved in target item's refined embedding, which may harm recommendation performance; (**L2**) The KG in the real-world industrial scenario is extremely large-scale, where one entity can be linked with up to millions of items. Existing works mainly employ the random sampling on the neighbors beforehand, which may lose latent critical information for the specific target user and item. As shown in Fig. 1 (b), some neighbors (*i.e., Shirt*) are abandoned by the random sampling strategy, while they are usually informative during aggregation since the target user has engaged with them; (**L3**) Most of these methods neglect rich relations among user behaviors over KG, while some works [5, 31] have demonstrated that capturing the relations among user behaviors is also beneficial for expressing user preference.

To address above limitations, we aim to distill the original over-informative KG into recommendation in a more effective way, which is expected to satisfy the following key properties: (1) **Target-behavior**: we hang on the novel insight that an effective KG base recommendation should produce semantic sub-graph to adapt for each target user-item pair, with the aim of capturing the underlying mutual effect characterized by KG (**L1**); (2) **Adaptive**: distinct from random sampling on the whole KG, our idea is to follow the adaptive principle for the sub-graph construction, which adaptively preserves useful information connecting user behaviors and target item over the KG, driving our model to provide more effective recommendation (**L2**); (3) **Relational**: the model architecture should be designed to relation-aware in order to consider the rich relations among user behaviors and target item over KG (**L3**). For convenience, given a target user-item pair, we call the relational graphical structure bridging user behaviors (*i.e.,* historical click records) with the target item as *adaptive target-behavior relational graph* (shown in Fig. 1(c)). Propagating user preference on such a relational structure potentially takes full advantage of the mutual effect for target user-item pair, as well as comprehensively captures the structural relations derived from KG.

In this paper, by integrating above main ideas together, we propose a new framework named **A**daptive **T**arget-**B**ehavior **R**elational **G**raph **N**etwork (ATBRG), which is comprised of two main parts: (1) Graph construction part. To extract the effective relational sub-graph of for target the user-item pair over the KG adaptively, we propose the *graph connect* and *graph prune* techniques. Firstly, we explore multiple layer neighbors over KG for target item and each item in user behavior, respectively. Among these entity sets, we *connect* entities which appear in multiple entity sets and *prune* entities belonging to only one entity set. Subsequently, we construct the adaptive target-behavior relational graph, which characterizes the structural relations among user behaviors and target item over the KG. (2) Model part. Considering structural relations derived from KG, we technically design the relation-aware extractor layer, which employs relation-aware attention mechanism to aggregate structural knowledge over the relational graph for each user behavior and target item. Afterwards, we introduce the representation activation layer to activate the relative relational representations of user behavior *w.r.t.* that of target item.

The main contributions of our work are summarized as follows:

- To effectively characterize structural relations between the given target user and item, we propose to extract an adaptive target-behavior relational graph, where the graph connect and graph prune strategies are developed to adaptively build relations between user behaviors and target item over KG.
- We propose a novel framework ATBRG, a well-designed graph neural network based architecture to learn relational representations of user behaviors and target item over the extracted sub-graph. Moreover, we equip it with relation-aware extractor layer and representation activation layer for emphasizing rich relations for interaction in KG.
- We perform a series of experiments on a benchmark dataset from Yelp and an industrial dataset from Taobao App. Experimental results demonstrate that ATBRG consistently and significantly outperforms various state-of-the-art methods. Moreover, ATBRG has been successfully deployed in

one popular recommendation scenario of Taobao APP and gained a performance improvement of 5.1% on CTR metric.

2 RELATED WORK

In this section, we review the most related studies in behavior based and knowledge aware recommendation.

2.1 Behavior based recommendation

In the early stage of recommendation, researchers focus on recommending a suitable list of items based on historical user-item interaction records. In particular, a series of matrix factorization based methods [11] have been proposed to infer user preference towards items through learning latent representations of users and items. Due to the ability of modeling complex interaction between users and items, deep neural network based methods (*e.g.,* YoutubeNet [3], DeepFM [7]) are widely adopted in industrial recommender systems, and reveal the remarkable strength of incorporating various context information (*e.g.,* user profile and item attributes).

In the online e-commerce systems, we are particularly interested in user's historical behaviors, which implies rich information for inferring user preference. Hence, how to effectively characterize the relationships between user behaviors and target item remains a continuous research topic. DIN [32] adaptively learns the representation of user interests from historical behaviors w.r.t. the target item by the attention mechanism. Inspired by DIN, the majority of following up works inherit this kind of paradigm. GIN [12] mines user intention based on co-occurrence commodity graph in the end-to-end fashion. ATRANK [30] proposes an attention-based behavior modeling framework to model users' heterogeneous behaviors. DIEN [31] and SDM [14] devote to capturing users' temporal interests and modeling their sequential relations. DSIN [5] focuses on capturing the relationships of users' inter-session and intra-session behaviors. MIMN [15] and HPMN [16] apply the neural turing machine to model users' lifelong sequential behaviors. Besides these improvements, knowledge graph, consisting of various semantics and relations, emerges as an assistant to describe relationships between user behaviors and target item.

2.2 Knowledge Aware Recommendation

As a newly emerging direction, knowledge graph is widely integrated into recommender systems for enriching relationships among user behaviors and items. A research line utilizes KG aware embeddings (*e.g.,* structural embeddings [28] and semantics embeddings [21]) to enhance the quality of item representations. These methods conduct mutli-task learning within two tasks of recommendation and KG completion and share the embeddings, and thus can hardly take full advantage of high-order information over KG. On the contrary, several efforts [8, 24] have been made to explore different semantic path (meta-path) connecting target users and items over KG, and then learn prediction function through multiple path modeling. More recently, some works [26] propose to exploit reinforcement learning to explore useful path for recommendation. Despite effectiveness, the path based method ignores rich structural information implied in KG since each extracted path is modeled independently.

Recently, graph neural network has shown its potential in learning accurate node embeddings with the high-order graph topology. Taking advantages of information propagation, RippleNet [20] propagates users' potential preferences and explores their hierarchical interests over KG, while KGCN-LS [22] and KGAT [23] perform embedding propagation by stacking multiple KG aware GNN layers. Although GNN based methods have achieved performance improvement to some extent, they do not take mutual influence between target user behaviors and item into consideration in the procedure of information aggregation. Moreover, the exponential neighborhood expansion over graph extremely increases the complexity of the system.

3 PRELIMINARY

In a recommendation scenario (*e.g.,* e-commerce and news), we typically have a series of historical interaction records (*e.g.,* purchases and clicks) between users and items. Let $\mathcal{U}$ denote a set of users and $\mathcal{I}$ denote a set of items, we denote interaction records as $\mathcal{H} = \{(u, i, \mathcal{B}_{ui}, y_{ui} | u \in \mathcal{U}, i \in \mathcal{I}\}$. Here, $\mathcal{B}_{ui} \subset \mathcal{I}$ represents historical behaviors (*i.e.,* item list) for user u when recommending item i and $y_{ui} \in \{0, 1\}$ is the implicit feedback of user u *w.r.t.* item i, where $y_{ui} = 1$ when $\langle u, i \rangle$ interaction is observed, and $y_{ui} = 0$ otherwise. In the real-world industrial recommender systems, each user u is associated with a user profile x_u consisting of sparse features (*e.g.,* user id and gender) and numerical features (*e.g.,* user age), while each item i is also associated with a item profile x_i consisting of sparse features (*e.g.,* item id and brand) and numerical features (*e.g.,* price).

In order to effectively incorporate auxiliary information of items (*i.e.,* item attributes and external knowledge) into recommendation, we frame our recommendation task over knowledge graph, which can be defined as follows:

Definition 1. ***Knowledge Graph***. *A KG is defined as a directed graph $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$ with an entity set $\mathcal{E}$ and a relation set $\mathcal{R}$. Each triplet $(h, r, t) \in \mathcal{T}$ denotes a fact that there is a relationship r from head entity h to tail entity t, where $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$.*

For example, $\langle Blouse, Category, Shirt \rangle$ states the fact that *Blouse* belongs to the *Shirt Category*. To bridge knowledge graph with recommender system, we adopt a item-entity alignments function $\phi : \mathcal{I} \rightarrow \mathcal{E}$ to align items with entities in KG.

Many efforts, especially GNN based methods, have been made to leveraging KG for better recommendation. While, most of these works overlook the mutual effect between target user and item when exploiting structural information derived from KG. To effectively distill structural knowledge through KG based GNN for item recommendation, we particularly investigate into the external knowledge connecting user behaviors and target item in KG, which can reveal semantic context for user-item interactions. Formally, we define such context information as follows:

Definition 2. ***Adaptive Target-Behavior Relational Graph***. *Given a target user-item pair $\langle u, i \rangle$ and corresponding user behaviors $\mathcal{B}_{ui}$, an adaptive target-behavior relational graph* w.r.t. $\langle u, i \rangle$ *is defined as a sub-graph extracted from the original KG, connecting user behavior $\mathcal{B}_{ui}$ and target item i.*

Table 1: Notations.

Notations	Description
$\mathcal{U}, \mathcal{I}$	the set of users and items, respectively
$\mathcal{H}$	the set of historical interaction records
$y_{ui}, \hat{y}_{ui}$	the label and the predicted probability
$x_u, x_i, \mathcal{B}_{ui}$	user profile, item profile and user behaviors
i_b	the specific item in user behavior
$\mathcal{G}, \mathcal{G}_{ui}$	the knowledge graph and adaptive target-behavior relational graph, respectively
$\mathcal{E}, \mathcal{R}, \mathcal{T}$	the set of entity, relation and triples in knowledge graph, respectively
$\langle h, r, t \rangle$	the specific triple in knowledge graph
$\mathbf{x}_u, \mathbf{x}_i, \mathbf{x}_e, \mathbf{x}_r$	the embedding of user u, item i, entity e and relation r, respectively
$\mathcal{N}_h^{(l)}$	the neighbors set in l-th relation-aware extractor layer for entity h
$\widetilde{\mathbf{x}}_i, \widetilde{\mathbf{x}}_{ib}$	the relational representation for target item i and each item $i_b \in \mathcal{B}_{ui}$, respectively
$\widetilde{\mathbf{x}}_u$	the final representation of user u

Given the above preliminaries, we now formulate the recommendation task to be addressed in this paper:

Definition 3. ***Task Description***. *Given a knowledge graph $\mathcal{G}$ with historical interaction records $\mathcal{H}$, for each user-item pair $\langle u, i \rangle$, we aim to predict probability $\hat{y}_{ui}$ that user u would click item i.*

4 THE PROPOSED FRAMEWORK

In this section, we introduce our proposed framework ATBRG, which aims to take full advantage of knowledge graph for recommendation. The framework is shown in Fig. 2, which is composed of two modules: (1) To effectively extract structural relational knowledge for recommendation, we propose to construct the adaptive target-behavior relational graph for the given target user-item pair over knowledge graph, where the *graph connect* and *graph prune* techniques help mine high-order connective structure in an automatic manner; (2) To jointly distill such a relational graph and rich relations among user behaviors in an end-to-end framework, we elaborate on the model design of ATBRG, which propagates user preference on the sub-graph with relation-aware extractor layer and representation activation layer. The key notations we will use throughout the article are summarized in Table 1.

4.1 Graph Construction

A major novelty of our work is to effectively explore adaptive target-behavior relational graph for improving the modeling of the interaction. In this part, we introduce the strategy to construct the adaptive target-behavior relational graph with the proposed *graph connect* and *graph prune* techniques. To model the relationship between the given target user-item over KG, previous works either extract different paths through random walk [24], or directly leverage the neighbors of target item over the original KG [22, 23]. Unfortunately, the first strategy neglects the structural relational information of the KG, while the second ignores the mutual effect between user behaviors and target item. Hence, we argue that above two strategies only achieve the suboptimal performance for recommendation.

Intuitively, the reasons driving a user to click a target item maybe implied by his/her historical behaviors, which is expected to guide our model to adequately aggregate useful information over external KG in an automatic manner. To distill the structural relational information over the KG in a more effective way, we propose to construct the adaptive relational graph w.r.t. user behaviors and target item. The procedure of the graph construction is clearly presented in the Algorithm 1 and left part of Fig. 2. Specifically, given a target user-item pair $\langle u, i \rangle$, we firstly exhaustively search the multi-layer entity neighbors for user behaviors $\mathcal{B}_{ui}$ and target item i over the KG, and restore the paths connecting the entity and item into $\mathcal{G}_{ui}$ (lines 1-6). Through this, we connect the user behaviors and target item by multiple overlapped entities. Afterwards, for the entities in $\mathcal{G}_{ui}$, we prune the entities which do not connect different items. (lines 7-16). Finally, we get the relational graph $\mathcal{G}_{ui}$ for user u and target item i, which describes the structural relations for $\langle u, i \rangle$ over the KG.

Algorithm 1 Graph construction

Input: Target item i; User behavior $\mathcal{B}_{ui}$; Knowledge graph $\mathcal{G}$;
Output: $\mathcal{G}_{ui}$: Adaptive target-behavior relational graph for $\langle u, i \rangle$;

```
1:  for item v ∈ [i, B_ui] do:
2:      for entity e ∈ φ(v) do:
3:          Construct path p = (e, r_k, e_k, ..., e_1, r_1, v);
4:          G_ui[entity] ← G_ui[entity] ∪ p;        ▷ Graph connect.
5:      end for
6:  end for
7:  for entity e ∈ G_ui do:
8:      New item hash set s;
9:      for path p ∈ G_ui[e] do:
10:         Collect item v on the path;
11:         s ← s ∪ v;
12:     end for
13:     if s.size = 1 then:
14:         Prune e in G_ui;                        ▷ Graph prune.
15:     end if
16: end for
```

4.2 Model Architecture

After obtaining the adaptive target-behavior relational graph derived from the KG, we continue to study how to produce predictive embeddings for target user-item pairs through propagating user preference over such a sub-graph. As shown in the right part of Fig. 2, the model architecture of our proposed ATBRG is composed of four layers: 1) Embedding layer, which transforms high-dimensional sparse features into low-dimensional dense representations; 2) Relation-aware extractor layer, which produces knowledge aware embeddings for user behaviors and target item by aggregating structural relational information over adaptive target-behavior relational graph; 3) Representation activation layer, which activates the relative relational representations of user behaviors w.r.t. that

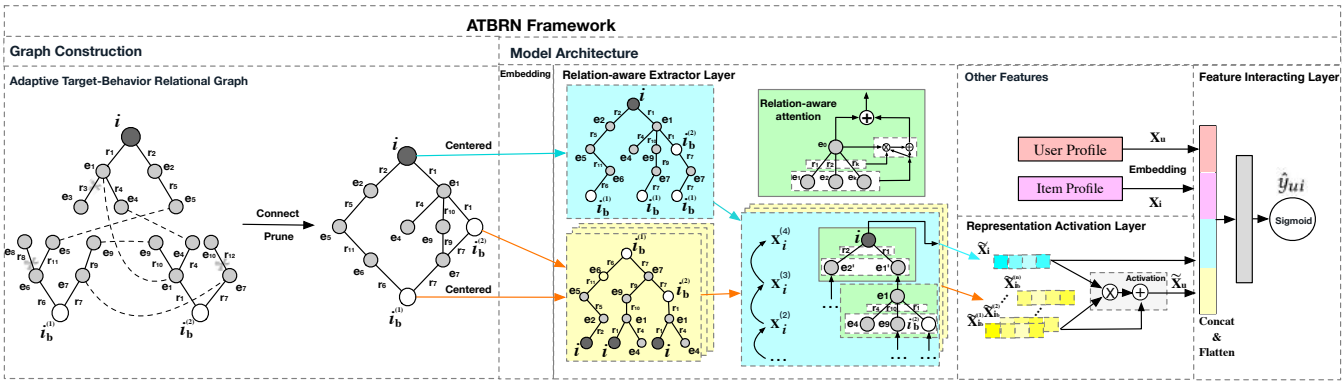


Figure 2: Overview of the proposed ATBRG framework. Overall, ATBRG consists of two parts, graph construction and model architecture.

of target item. 4) Feature interaction layer, which combines the user and item profile with the activated relational representation of user behaviors and target item for interaction.

4.2.1 Embedding Layer. As mentioned above, users and items in real-world recommendation scenario are both associated with abundant profile information in the form of sparse and dense features. Hence, we set up a embedding layer to parameterize users and items as vector representations, while preserving the above profile information. Formally, giving a user u, we have corresponding raw feature space x_u, comprised of sparse feature space x_u^s and dense feature space x_u^d. For sparse features, following [3, 5, 31, 32], we embed each feature value into d dimensional dense vector, while dense feature can be standardized or batch normalization to ensure normal distribution. Subsequently, each user u can be represented as $\mathbf{x}_u \in \mathbb{R}^{|x_u^s| \times d + |x_u^d|}$, where $|x_u^s|$ and $|x_u^d|$ denotes the size of sparse and dense feature space of user u, respectively. Similarly, we represent each item i as $\mathbf{x}_i \in \mathbb{R}^{|x_i^s| \times d + |x_i^d|}$. Moreover, each entity e and relation r in the adaptive target-behavior relational graph can also be embeded as $\mathbf{x}_e \in \mathbb{R}^d$ and $\mathbf{x}_r \in \mathbb{R}^d$ [1].

4.2.2 Relation-aware Extractor Layer. This layer is designed to effectively and comprehensively distill the structural relational information from the extracted sub-graph. Previous works [10, 19] neglect relational edges during aggregation, which play essential roles in real-world settings. In our scenario, a user u may *click* or *buy* the same item i, while the relations *click* and *buy* obviously indicate the different preference of user u towards item i. Therefore, we elaborately build the relation-aware extractor layer to adequately exploit rich structural relational information in KG in the consideration of various relation between entities.

Based on the above discussions and inspired by the study [6], we stack the relation-aware extractor layer by layer in order to recursively propagates the embeddings from an entity's neighbors to refine the entity's embedding in KG. Specifically, for each item (*i.e.,* item $i_b \in \mathcal{B}_{ui}$ or target item i), we will regard it as the center node and aggregate information over the extracted sub-graph $\mathcal{G}_{ui}$ through relation-aware aggregation. Given an entity [2] h in extracted relational sub-graph $\mathcal{G}_{ui}$, let $\mathcal{N}_h^{(l)} = \{(r,t)|(h,r,t) \in \mathcal{G}_{ui}\}$ to denote the neighbors set in l-th layer and $\mathbf{x}_h^{(l)}$ to denote the representation of entity h in the l-th layer. We implement the l-th relation-aware aggregation layer as follows,

$$\begin{aligned} \alpha^{(l)}(h,r,t) &= \frac{\exp(\mathbf{x}_r \mathbf{W}_\alpha f(\mathbf{x}_h^{(l)} \oplus \mathbf{x}_t^{(l)}))}{\sum_{(r',t') \in \mathcal{N}_h^{(l)}} \exp(\mathbf{x}_{r'} \mathbf{W}_\alpha f(\mathbf{x}_h^{(l)} \oplus \mathbf{x}_{t'}^{(l)}))}, \\ \mathbf{x}_h^{(l+1)} &= \mathbf{x}_h^{(l)} \oplus \sum_{(r,t) \in \mathcal{N}_h^{(l)}} \alpha^{(l)}(h,r,t) \mathbf{x}_t^{(l)}. \end{aligned} \tag{1}$$

Here, $f(\mathbf{x})$ and $\mathbf{W}_\alpha$ denote the single layer perceptron and attentive matrix in l-th layer, respectively. And $\oplus$ denotes the concatenation operation. Relation-aware extractor layer is stacked layer by layer to propagate user preference over KG. Subsequently, each entity e in sub-graph $\mathcal{G}_{ui}$ can be denoted as $\mathbf{x}_e^{(L)}$ after L relation-aware extractor layer.

Given a target user-item pair $\langle u, i \rangle$ and the corresponding adaptive target-behavior relational graph $\mathcal{G}_{ui}$, the knowledge aware representation of target item i can be denoted as $\widetilde{\mathbf{x}}_i = \mathbf{x}_i^{(L)}$, where $\phi(i) = e$. Similarly, we also obtain relational representation set $\{\widetilde{\mathbf{x}}_{i_b} = \mathbf{x}_{i_b}^{(L)} | \phi(i_b) = e\}_{i_b \in \mathcal{B}_{ui}}$ for user behaviors.

4.2.3 Representation Activation Layer. Intuitively, user behaviors contribute differently to the final prediction. For example, the behavior *shirt A* is more informative than *shoe B* when the target item is *shirt C*. For this purpose, we set a representation activation layer to place different importance on relational representation of user behaviors $\{\widetilde{\mathbf{x}}_{i_b}\}_{i_b \in \mathcal{B}_{ui}}$. Specifically, we apply the vanilla attention mechanism [1] to activate representations of user behaviors that are more related to target item, calculated as follows,

$$\begin{aligned} \beta(u,i,i_b) &= \frac{\exp(\widetilde{\mathbf{x}}_{i_b} \mathbf{W}_\beta \widetilde{\mathbf{x}}_i))}{\sum_{i'_b \in \mathcal{B}_{ui}} \exp(\widetilde{\mathbf{x}}_{i'_b} \mathbf{W}_\beta \widetilde{\mathbf{x}}_i)}, \\ \widetilde{\mathbf{x}}_u &= \sum_{i'_b \in \mathcal{B}_{ui}} \beta(u,i,i_b) \widetilde{\mathbf{x}}_{i'_b}, \end{aligned} \tag{2}$$

[1] Entities and relations in KG only have the sparse id features

[2] For convenience, we omit the subscript $_{ui}$ in this part.

Table 2: Statistics of datasets

	Description	Taobao	Yelp
User-Item Interaction	#Users	2.2×10^8	4.5×10^4
	#Items	1.1×10^8	4.5×10^4
	#Interactions	7.2×10^9	1.0×10^6
Knowledge Graph	#Entities	1.4×10^7	8.3×10^4
	#Relations	34	35
	#Triplets	3.8×10^{10}	1.6×10^6
	#Max neighbor depth	3	1

where $\mathbf{W}_\beta$ is the attentive matrix in representation activation layer.

4.2.4 Feature Interaction Layer. Until now, given a target user-item pair $\langle u, i \rangle$, we have the profile embeddings for user u and item i, and knowledge aware embedding from adaptive target-behavior relational graph for user behaviors and target item. We combine the four embedding vectors into a unified representation and employ Multiple Layer Perceptron (MLP) for better feature interaction [5, 15, 30–32].

$$\hat{y}_{ui} = \sigma(f(f(f(\mathbf{x}_u \oplus \mathbf{x}_i \oplus \widetilde{\mathbf{x}}_u \oplus \widetilde{\mathbf{x}}_i)))), \tag{3}$$

where $\sigma(\cdot)$ is the logistic function and $\hat{y}_{ui}$ represents the prediction probability of the user u to click on the target item i.

4.2.5 Loss Function. We reduce the task to a binary classification problem and use binary cross-entropy loss function defined as follows:

$$\mathcal{L} = -\frac{1}{N} \sum_{(u,i) \in \mathbb{D}} (y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log(1 - \hat{y}_{ui})) \tag{4}$$

where $\mathbb{D}$ is the training dataset and $y_{ui} \in \{0, 1\}$ represents whether the user u clicked on the target item i.

5 EXPERIMENTS

In this section, we perform a series of experiments on two real-world datasets, with the aims of answering the following research questions:

- **RQ1**: How does our proposed model ATBRG perform compared with state-of-the-art methods on the recommendation task?
- **RQ2**: How do different experimental settings (i.e., depth of graph, aggregator selection, etc.) influence the performance of ATBRG?
- **RQ3**: How does ATBRG provide effective recommendation intuitively?

5.1 Experimental Setup

5.1.1 Datasets. We conduct extensive experiments on two real-world datasets: industrial dataset from Taobao and benchmark dataset from Yelp.

- **Taobao** [3] dataset consists of click logs from 2019/08/22 to 2019/08/29, where the first one week's samples are used for training and samples of the last day are for testing. Moreover, Taobao dataset also contains user profile (*e.g.,* id and age), item profile (*e.g.,* id and category) and up to 10 real-time user behaviors [4].
- **Yelp** [5] dataset records interactions between users and local business and contains user profile (*e.g.,* id, review count and fans), item profile (*e.g.,* id, city and stars) and up to 10 real-time user behaviors. For each observed interaction, we randomly sample 5 items that the target user did not engage with before as negative instances. For each user, we hold the latest 30 instances as the test set and utilizes the remaining data for training.

Besides user behaviors, following [13, 18, 29], we construct item knowledge for Taobao (*e.g.,* category, parent and style). Also, for Yelp dataset, KG is organized as the local business information (e.g., location and category). The detailed descriptions of the two datasets are shown in Table 2. Note that the volume of Taobao dataset is much larger than yelp, which brings more challenges.

5.1.2 Baselines. We compare our ATBRG with three kinds of representative methods: feature based methods (*i.e.,* YoutubeNet and DeepFM) mainly utilizing raw features derived from user and item profile, behavior based methods (*i.e.,* DIN, DIEN and DSIN) capturing user's historical behaviors and knowledge graph (KG) based methods (*i.e.,* RippleNet, KGAT and KPRN) benefiting from knowledge graphs in recommendation. The comparison methods are given below in detail:

- **YoutubeNet** [3] is a standard user behavior based method in the industrial recommender system.
- **DeepFM** [7] combines factorization machine and deep neural network for recommendation.
- **DIN** [32] locates related user behaviors *w.r.t.* target itemby using attention mechanism.
- **DIEN** [31] models users temporary interests and the interest evolving process via GRU with attention update gate.
- **DSIN** [5] models user's session interests and the evolving process with self-attention mechanism and Bi-LSTM.
- **RippleNet** [20] propagates user's potential preferences over the set of knowledge entities.
- **KPRN** [24] is a typical path based recommendation method, which extracts qualified path to between a user with an item.
- **KGAT** [23] is a state-of-the-art KG-based recommendation methods, which employs GNN on KG to generate representations of users and items, respectively.

5.1.3 Evaluation Metrics. We adopt area under ROC curve (**AUC**) to evaluate the performance of all methods. Larger AUC indicates better performance. Besides, we also present the relative improvement (**RI**) *w.r.t.* AUC of our model achieves over the compared models, which can be formulated as:

$$\text{RI} = \frac{|\text{AUC}(model) - \text{AUC}(base)|}{\text{AUC}(base)} * 100\%, \tag{5}$$

where $|.|$ is the absolute value, *model* refers to our proposed framework ATBRG and *base* refers to the baseline. Note that **0.001** improvement *w.r.t.* AUC is remarkable in industrial scenario (*i.e.,* Taobao dataset).

[3] www.taobao.com.

[4] Real-time user behaviors means user behaviors before this action occurs.

[5] www.yelp.com/dataset.

Table 3: Overall performance comparison *w.r.t.* AUC (bold: best; underline: runner-up).

Model	Taobao		Yelp†	
	AUC	RI	AUC	RI
YoubtubeNet	0.6017	+2.72%	0.7109	+26.00%
DeepFM	0.6037	+2.38%	0.7334	+22.14%
DIN	0.6058	+2.03%	0.7520	+19.12%
DIEN	0.6061	+1.97%	0.7581	+18.16%
DSIN	0.6073	+1.77%	0.7774	+15.23%
RippleNet	0.5975	+3.44%	0.7324	+22.31%
KGAT	0.6062	+1.96%	0.7876	+13.73%
KPRN	<u>0.6096</u>	+1.39%	<u>0.8260</u>	+8.45%
ATBRG	**0.6181***	-	**0.8958***	-

† Note that the relative improvement on the public Yelp dataset is much higher than the industrial Taobao dataset, since the negative samples of the public Yelp dataset are generated by random sampling and thus easier to distinguish.

Table 4: Effect of the representation activation layer and relation-aware mechanism.

Model	Taobao		Yelp	
	AUC	RI	AUC	RI
$\text{ATBRG}_{w/o\ RAM}$	0.6157	+0.38%	0.8858	+1.12%
$\text{ATBRG}_{w/o\ RAL}$	0.6125	+0.91%	0.8940	+0.20%
ATBRG	**0.6181***	-	**0.8958***	-

Table 5: Effect of the depth of neighbor.

Model†	Taobao		Yelp	
	AUC	RI	AUC	RI
$\text{ATBRG}_{1/0}$	0.6054	+2.09%	0.7523	+19.07%
$\text{ATBRG}_{3/1}$	0.6143	+0.61%	**0.8958***	-
$\text{ATBRG}_{5/2}$	**0.6181***	-	-	-
$\text{ATBRG}_{7/3}$	0.6163	+0.29%	-	-

†$\text{ATBRG}_{m/n}$ means ATBRG explores m layers of neighbors over the extracted sub-graph, which corresponds to n layers of neighbors in original KG.

5.1.4 Implementation. We implement all models in Tensorflow 1.4. Moreover, for fair comparison, pre-training, batch normalization and regularization are not adopted in our experiments. For RippleNet, we set the max depth of ripple as 3. For KGAT, the max neighbour depth of target user and item is set to 4 and 3, respectively. For KPRN, the max number of extracted paths over the knowledge graph are set to 50. For ATBRG, the max neighbor depth of the item is set to 3. For all models, We employ random uniform to initialize model parameters and adopt Adagrad as optimizer using a learning rate of 0.001. Moreover, embedding size of each feature is set to 4 and the architecture of MLP is set to [512, 256, 128]. We run each model three times and reported the mean of results.

5.1.5 Significance Test. For Experimental results in Tables 4, 5, 6 and 7, we use "*" to indicate that ATBRG is significantly different from the runner-up method based on paired t-tests at the significance level of 0.01.

5.2 Performance Comparison (RQ1)

We report the AUC comparison results of ATBRG and baselines on two datasets in Table 3. The major findings from the experimental results are summarized as follows:

- Feature based methods (*i.e.,* YoutubeNet and DeepFM) achieve relatively pool performance on two datasets. It indicates that handcrafted feature engineering is insufficient to capture the complex relations between users and items, further limiting performance. Moreover, DeepFM consistently outperforms YoutubeNet across all cases, since it employs FM part for better feature interaction.
- Compared to feature based methods, the performance of behavior based methods (*i.e.,* DIN, DIEN and DSIN) verifies that incorporating historical behaviors is beneficial to infer user's preference. Among them, DSIN achieves the best performance on both datasets due to integration of user's session interests.
- Generally, KG based methods (*i.e.,* RippleNet, KGAT, KPRN) achieve better performance than behavior based methods in most cases, which indicates the effectiveness of knowledge graph for capturing underlying interaction between users and items. However, RippleNet underperforms other baselines on both datasets. One possible reason is that RippleNet ignores user's short-term interest implied in historical behaviors. Moreover, KPRN generally achieves remarkable improvements in most cases. It makes sense since reasonable and explainable target user-item paths extracted from KG are helpful to improve recommendation performance.
- ATBRG consistently yields the best performance on both datasets. In particular, ATBRG improves over the best baseline w.r.t. AUC by 1.39%, and 8.45% on Taobao and Yelp dataset, respectively. By stacking multiple GNN layers, ATBRG is capable of exploring rich structural and relational information over KG, while KPRN only models each extracted path independently. This verifies the importance of capturing both semantics and topological structures derived from KG for recommendation. Besides, compared with KGAT, which only represents target user and item separately by aggregating their own neighbors over the original KG, ATBRG achieves better performance for the following two reasons: 1) ATBRG considers the mutual effect between the given user behaviors and target item by constructing the adaptive relational sub-graph for them. Propagating on such a sub-graph can better capture the structural relations between user behaviors and target item and further explore potential reasons driving the user to click the target item; 2) ATBRG integrates relations when aggregating the entities by the relation-aware attention mechanism, and creatively produces the relational representations over the extracted sub-graph for each user behavior and target item.

Table 6: Effect of different aggregators.

Model	Taobao		Yelp	
	AUC	RI	AUC	RI
ATBRG$_{concat}$	0.6131	+0.64%	0.8901	+0.65%
ATBRG$_{sum}$	0.6133	+0.78%	0.8906	+0.58%
ATBRG$_{sa}$	0.6145	+0.58%	0.8928	+0.33%
ATBRG$_{nl}$	0.6159	+0.35%	0.8946	+0.13%
ATBRG	**0.6181***	-	**0.8958***	-

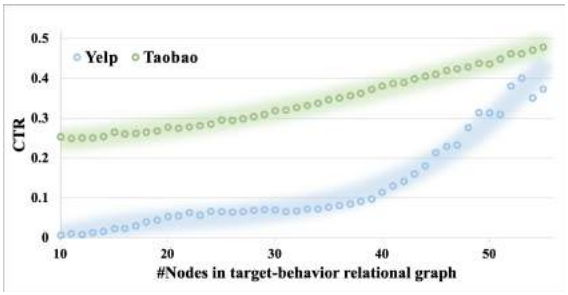


Figure 3: Impact of the number of nodes in adaptive target-behavior relational graph *w.r.t.* CTR.

5.3 Study of ATBRG (RQ2)

In this section, we perform a series of experiments to better understand the traits of ATBRG, including well-designed components (*e.g.,* relation-aware mechanism and representation activation layer) and key parameter settings (*i.e.,* neighrbor depth and aggregator).

5.3.1 Effect of Relation-aware Mechanism and Representation Activation Layer. ATBRG provides a principled way to characterize various relations in KG and user behaviors to enhance recommendation performance. To examine the effectiveness of relation-aware mechanism and representation activation layer, we prepare three variants of ATBRG:

- **ATBRG**$_{w/o\ RAM}$: The variant of ATBRG, which removes the relation-aware mechanism (Eq. 1).
- **ATBRG**$_{w/o\ RAL}$: The variant of ATBRG, which removes the representation activation layer (Eq. 2).

The AUC comparison results of ATBRG with its variants are show in Table 4. We have the following two observations:

- It is clear that the performance of ATBRG degrades without the relation-aware mechanism on both datasets (*i.e.,*ATBRG > ATBRG$_{w/o\ RAM}$). It demonstrates that different relations in KG should be distinguished, as disregarding such information leads to the worse performance.
- ATBRG without the representation activation layer performs worse consistently (*i.e.,*ATBRG > ATBRG$_{w/o\ RAL}$). It indicates that capturing the semantic relations among user behaviors over the KG can better understand user underlying preference, which is beneficial for the final prediction.

5.3.2 Effect of Neighbor Depth. The proposed ATBRG model is flexible to capture high-order structural information through recursively aggregating the embeddings from an entity's neighbors to refine the entity's embedding in KG. Here, we investigate how the neighbor depth over KG influences the model performance. Specifically, the neighbor depth of item is explored in the range of {0, 1, 2, 3} in Taobao dataset and {0, 1} for Yelp dataset. We summarize the results in Table 5 and have the following two observations:

- Overall, the model performance increases gradually when neighbor depth varies from 0 to 2 on both datasets. It demonstrates that deepening the neighbor layer helps capture the long-term structural relations of user behaviors and target item to some extent.
- The model performance of ATBRG degrades when the neighbor depth increases from 2 to 3. One possible reason is that long-term relations may include much more ineffective connectives (*i.e., Shirt - Women Clothing - Clothing - Men Clothing -Shoe*). Such relations in the graph introduce some noise and further harm the model performance.

5.3.3 Effect of Aggregator. In our model, we enrich the information of items by recursively capturing their neighbor information in KG. In order to explore the effect of different neighbor aggregator, we design different variants of ATBRG, listed as follows:

- **ATBRG**$_{concat}$: It applies concatenation operation [22].
- **ATBRG**$_{sum}$: It applies sum pooling.
- **ATBRG**$_{sa}$: It applies self-attention mechanism [19].
- **ATBRG**$_{nl}$: It applies nonlinear transformation [22].

We present the AUC comparison of ATBRG and its variants in Table 6. From the result, we have the following findings:

- Obviously, ATBRG with simple aggregators (*i.e.,*ATBRG$_{concat}$ and ATBRG$_{sum}$) performs worst on both datasets, since they ignore the different contributions of neighbors.
- Generally, ATBRG with complex aggregators (*i.e.,*ATBRG$_{sa}$ and ATBRG$_{nl}$) achieves better performance on both datasets. The reason is that ATBRG$_{sa}$ employs self-attention mechanism to place different importance on neighbors while ATBRG$_{nl}$ leverages nonlinear trasformation to characterize complex interaction.
- ATBRG consistently yields the best performance on both datasets. It illustrates that, our proposed relation-aware aggregator not only includes the nonlinear transformation in the weight calculation, but also considers the influence of relations during aggregation.

5.4 Case Study (RQ3)

To better understand the merits of our proposed ATBRG intuitively, we first make comprehensive instance-level analyses on the adaptive target-behavior relational graph. As shown in Fig. 3, we present the influence of the number of nodes over the relational graph on the click through rate (CTR). Here, CTR is calculated by averaging the real labels (1 for click and 0 otherwise). It can be clearly observed that CTR and the number of nodes are positively correlated on both datasets. This demonstrates that, the richer the relations between user behavior and target item on the extracted sub-graph, the more likely the user is to click on target item.

Moreover, with the aims of answering how ATBRG addresses the limitations (described in Section 1) existed in previous GNN based methods for knowledge-aware recommendation, we conduct

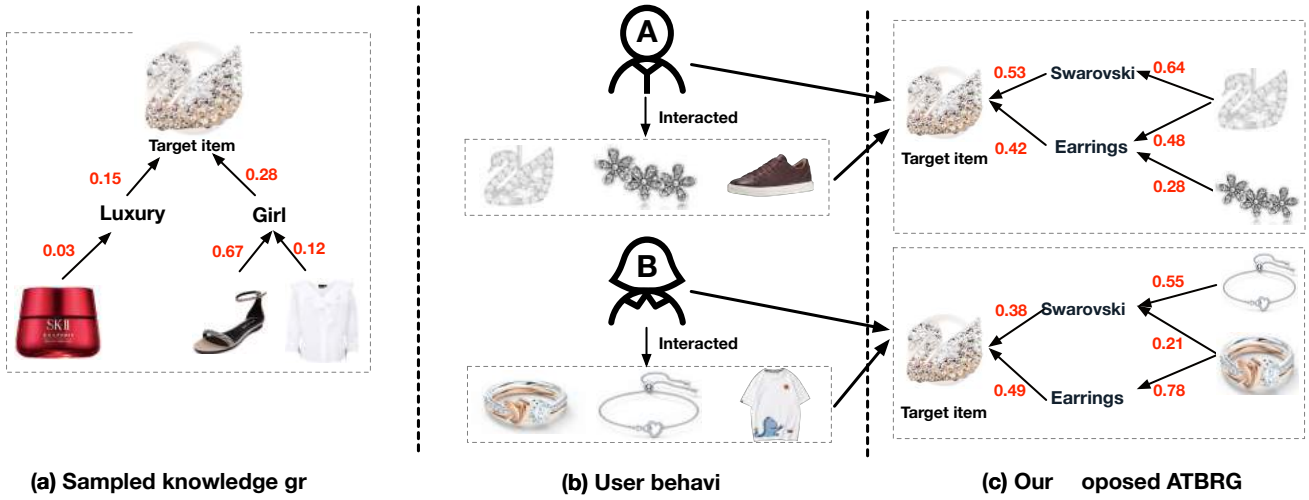


Figure 4: An illustrative example of how our proposed ATBRG works more effective than other knowledge graph based methods. (a) indicates the sampled neighbors of the target item over the knowledge graph. (b) & (c) introduce the specific user behaviors of users and the corresponding extracted sub-graph. The red number above the edge implies the calculated weights.

one case study in large-scale industrial Taobao dataset. As shown in Fig. 4, the main findings are summarized as follows:

- In part (a), due to the limitation (**L2**), the original neighbors of the target item over the knowledge graph are randomly sampled beforehand. Hence, some relevant entities (*i.e., Swarovski* and *Earring*) connecting users are discarded, while other ineffective entities (*i.e., Luxury* and *Girl*) are reserved, which inevitably introduce noises. It demonstrates that previous methods are incapable of adaptively sampling neighbors for target user-item pairs, and further harm the recommendation performance.
- In part (b), we present the users' recent behaviors, where some behaviors (*i.e., Necklace* and *Earring*) are related to target item over the knowledge graph while others (*i.e., Shoe* and *Skirt*) are not. By the graph connect and prune techniques, we adaptively preserve the effective entities and relations over knowledge graph (**L2**). Subsequently, in part (c), we construct the specific adaptive target-behavior relational graph for the given target user-item pair, which provides strong evidences for inferring user preference. Propagating embeddings on on such a relational structure can take full advantage of the mutual effect of target user-item pair for recommendation (**L1**).
- In order to consider the rich relations among user behaviors over KG, we propose the relation-aware extractor layer to weigh various underlying preferences for recommendation. Compared with part (a), we find the weights are also adaptive for different users (**L3**). Specifically, *Swarovski* is paid more attention to and scored higher than *Earring* by the user A, while it is the opposite for user B. Therefore, the final relational representation can reflect the personalized preferences of different users towards the target item.

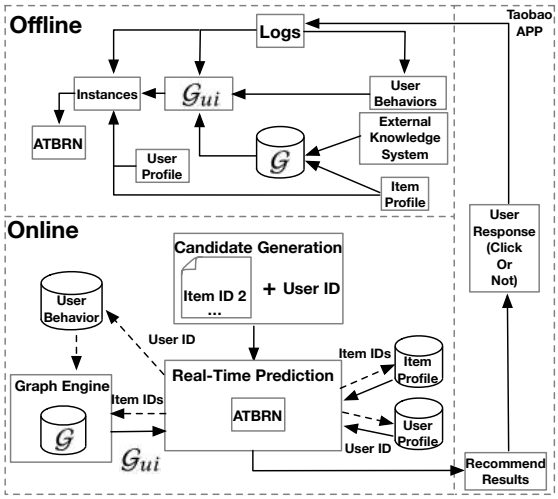


Figure 5: The deployment of ATBRG in Taobao APP.

5.5 Online A/B Testing

To verify the effectiveness of our proposed framework ATBRG in the real-world settings, ATBRG has been deployed in the popular recommendation scenario of Taobao APP. As shown in Fig. 5, the deployment pipelines consist of three parts: 1) User response. Users give implicit feedback (click or not) to the recommended items provided by the recommender system; 2) Offline training. In this procedure, we integrate the knowledge graph $\mathcal{G}$, user behaviors and target item to construct the adaptive target-behavior relational graph $\mathcal{G}_{ui}$. Afterwards, $\mathcal{G}_{ui}$, together with user profile and item profile makes up the instances, and are fed into ATBRG for training; 3) Online serving. When the user accesses Taobao APP, some candidates items are generated by the pipelines before real-time

prediction (RTP) service. The necessary components of ATBRG are achieved and organized in the same way as the offline training. At last, the candidates are ranked by the predicting scores of ATBRG, and truncated for the final recommend results.

Compared with existed deployed baseline model DIN, **6.8**% lift on click count and **5.1**% lift on CTR are observed for ATBRG, with the cost of **8** milliseconds for online inference. The promotion of recommendation performance verifies the effectiveness of our proposed framework ATBRG.

6 CONCLUSION

In this paper, we propose a novel framework ATBRG for knowledge aware recommendation. To effectively characterize the structure relations over KG, we propose the graph connect and graph prune techniques to construct adaptive target-behavior relational graph. Furthermore, we elaborate on the model design of ATBRG, equipped with relation-aware extractor layer and representation activation layer, which aims to take full advantage of structural connective knowledge for recommendation. Extensive experiments on both industrial and benchmark datasets demonstrate the effectiveness of our framework compared to several state-of-the-art methods. Moreover, ATBRG has also achieved 5.1% improvement on CTR metric in online experiments after successful deployment in one popular recommendation scenario of Taobao APP. In the future, we will consider applying causal inference in KG to improve the interpretability of recommender system.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.
[2] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *WWW*. 151–161.
[3] Covington, Paul, Adams, Jay, Sargin, and Emre. 2016. Deep neural networks for youtube recommendations. In *RecSys*. 191–198.
[4] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *WWW*. 271–280.
[5] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep Session Interest Network for Click-Through Rate Prediction. In *IJCAI*. 2301–2307.
[6] Liyu Gong and Qiang Cheng. 2019. Exploiting edge features for graph neural networks. In *CVPR*. 9211–9219.
[7] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *IJCAI*. 1725–1731.
[8] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *SIGKDD*. 1531–1540.
[9] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *SIGIR*. 505–514.
[10] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[11] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
[12] Feng Li, Zhenrui Chen, Pengjie Wang, Yi Ren, Di Zhang, and Xiaoyu Zhu. 2019. Graph Intention Network for Click-through Rate Prediction in Sponsored Search. In *SIGIR*. 961–964.
[13] Xusheng Luo, Luxin Liu, Yonghua Yang, Le Bo, Yuanpeng Cao, Jinhang Wu, Qiang Li, Keping Yang, and Kenny Q. Zhu. 2020. AliCoCo: Alibaba E-commerce Cognitive Concept Net. In *SIGMOD*.
[14] Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. 2019. SDM: Sequential deep matching model for online large-scale recommender system. In *CIKM*. 2635–2643.
[15] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Practice on Long Sequential User Behavior Modeling for Click-Through Rate Prediction. In *SIGKDD*. 2671–2679.
[16] Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, and Kun Gai. 2019. Lifelong Sequential Modeling with Personalized Memorization for User Response Prediction. In *SIGIR*.
[17] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. 285–295.
[18] Wei Shen, Jianyong Wang, and Jiawei Han. 2014. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering* 27, 2 (2014), 443–460.
[19] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.
[20] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *CIKM*. 417–426.
[21] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep knowledge-aware network for news recommendation. In *WWW*. 1835–1844.
[22] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge graph convolutional networks for recommender systems. In *WWW*. 3307–3313.
[23] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *SIGKDD*. 950–958.
[24] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *AAAI*. 5329–5336.
[25] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *AAAI*. 5329–5336.
[26] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *SIGIR*. 285–294.
[27] Fengli Xu, Jianxun Lian, Zhenyu Han, Yong Li, Yujian Xu, and Xing Xie. 2019. Relation-Aware Graph Convolutional Networks for Agent-Initiated Social E-Commerce Recommendation. In *CIKM*. 529–538.
[28] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *SIGKDD*. 353–362.
[29] Wayne Xin Zhao, Gaole He, Kunlin Yang, Hongjian Dou, Jin Huang, Siqi Ouyang, and Ji-Rong Wen. 2019. KB4Rec: A Data Set for Linking Knowledge Bases with Recommender Systems. *Data Intelligence* 1, 2 (2019), 121–136.
[30] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2018. ATRank: An attention-based user behavior modeling framework for recommendation. In *AAAI*. 4564–4571.
[31] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *AAAI*. 5941–5948.
[32] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *SIGKDD*. 1059–1068.
[33] Xiangmin Zhou, Dong Qin, Lei Chen, and Yanchun Zhang. 2019. Real-time context-aware social media recommendation. *The VLDB Journal* 28, 2 (2019), 197–219.

GMCM: Graph-based Micro-behavior Conversion Model for Post-click Conversion Rate Estimation

Wentian Bao*
wentian.bwt@alibaba-inc.com
Alibaba Group

Hong Wen*
qinggan.wh@alibaba-inc.com
Alibaba Group

Sha Li
Shal2@illinois.edu
University of Illinois
Urbana-Champaign

Xiao-Yang Liu
xl2427@columbia.edu
Columbia University

Quan Lin
tieyi.lq@alibaba-inc.com
Alibaba Group

Keping Yang
shaoyao@alibaba-inc.com
Alibaba Group

ABSTRACT

Purchase-related micro-behaviors, e.g., *favorite, add to cart, read reviews*, etc., provide implicit feedback of users' decision-making process. Such informative feedback can lead to fine-grained post-click conversion rate (CVR) modeling of the buying process. However, most existing works on CVR estimation either neglect these informative feedback, or model them as a sequential pattern with Recurrent Neural Networks. We argue such modeling could be inappropriate since different orders of micro-behaviors may represent similar user buying intention, and micro-behaviors often correlate with each other.

To this end, we propose to represent user micro-behaviors as a Purchase-related Micro-behavior Graph (PMG). Specifically, each node stands for one micro-behavior, and edge weights denote the connection strength. Based on this graph representation, we frame CVR estimation as a graph classification problem over the PMG instances. We propose a novel CVR model, namely, Graph-based Micro-behavior Conversion Model (GMCM), that utilizes Graph Convolutional networks (GCN) to enhance the conventional CVR modeling. In addition, we adopt multi-task learning and inverse propensity weighting to tackle two well-recognized issues in CVR estimation: data sparsity and sample selection bias. Extensive experiments on six large-scale production datasets demonstrate that the proposed methods outperform the state-of-the-art CVR methods under industrial setting.

CCS CONCEPTS

• **Information systems** → **Personalization**; **Information retrieval diversity**; **Recommender systems**.

*Both authors contributed equally to this work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGIR '20, July 25–30, 2020, Virtual Event, China
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401425

KEYWORDS

conversion rate estimation; graph convolutional networks; multi-task learning; recommender systems

ACM Reference Format:
Wentian Bao, Hong Wen, Sha Li, Xiao-Yang Liu, Quan Lin, and Keping Yang. 2020. GMCM: Graph-based Micro-behavior Conversion Model for Post-click Conversion Rate Estimation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401425

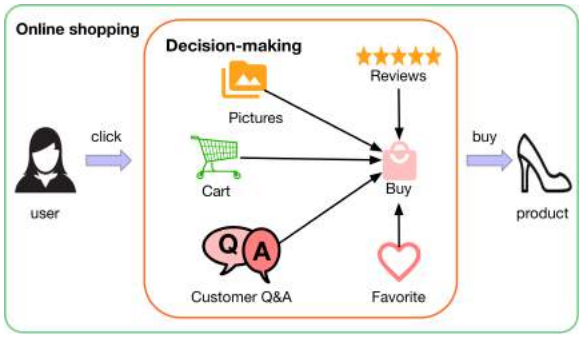


Figure 1: An illustration of the user decision-making process after clicking a product on online shopping app.

1 INTRODUCTION

E-commerce recommender systems help users find products of interest and also help business promote their products to the target costumers. Two tasks are essential for building such systems: click through rate (CTR) prediction and post-click conversion rate (CVR) prediction. CTR measures the transition probability from exposures to clicks, and CVR measures the probability of a product being purchased given it is clicked.

Despite the recent advances in CTR and CVR estimation [18, 32, 34, 35], the industrial-level CVR estimation is still challenging. In practice, the estimated CVR scores may deviate from the true values by large [17, 32], which results in unsatisfactory recommendation. Multiple reasons could lead to the performance degradation, and make the CVR estimation inaccurate. In general, we identify three

practical issues that make the industrial CVR estimation challenging: (i) limited purchase-related feedback; (ii) data sparsity; and (iii) sample selection bias.

The user decision-making process is complicated by nature, while limited purchase-related feedback are utilized to model such buying process. In practice, we notice that conversion labels alone are often too sparse to help the model accurately infer users' buying intention. For example, we have 3.6 billion exposures in our production dataset, while only 27M conversions. Besides, the users' decision-making process is complicated, and not buying a product does not necessarily mean that the user has no interest of it. Intuitively, to better understand such users' buying process, more purchase-related feedback from users should be taken into account. In practice, we find that before making a deal, users tend to have some purchase-related micro-behaviors, e.g., *add to cart, read reviews*, etc. As shown in Figure 1, these micro-behaviors often indicate users' buying intention, and reflect their decision-making process. In our systems, we find more than 70% of the purchases at least relate to one of the micro-behaviors, which emphasizes the importance of incorporating purchase-related feedback with CVR modeling.

However, very few existing works have investigated how to make use of these valuable feedback. To the best of our knowledge, [37] first proposed to utilize user micro-behaviors in e-commerce recommendation, and modeled such behaviors as a sequence. Such an attempt is meaningful but may be inappropriate in some cases. Firstly, different orders of micro-behaviors may represent similar user buying intention. For example, when the user is considering buying a product, the order of reading reviews and costumer Q&A does not make much of a difference. Secondly, micro-behaviors often correlate with each other, while sequential models fail to capture such interconnection. Therefore, we propose to represent the micro-behaviors as a purchase-related micro-behavior graph (PMG). The comparative study of graph-based and sequence-based modeling of micro-behaviors can be seen in Section 5.6.

Except for the lack of purchase-related feedback, conventional CVR modeling is also hindered by two well-recognized issues, i.e., data sparsity and sample selection bias. Data sparsity refers to the fact that training samples for CVR models may not be sufficient for fitting the large amount of model parameters. Conventional CVR models are trained using clicked samples, while clicks are relatively rare compared with exposures. Besides, for industrial CVR estimation, the embedding size of some sparse features could be millions or even billions, which makes the training require more data to converge. In our production datasets, we have 604 million clicked samples vs. over 14 billion embedding parameters (most of them are product sparse embedding). Sample selection bias comes from the fact that conventional CVR models are trained using clicked samples, while the inference is made on all exposed samples. The discrepancy between the distributions of click and exposure space biases the conventional CVR models, and makes the estimated CVR scores deviate from the true values by large [4, 18, 27].

To address the three issues above, we propose to represent the user micro-behaviors as a Purchase-related Micro-behavior Graph (PMG). Specifically, each node in graph stands for a micro-behavior type, e.g., *add to cart, read reviews*, etc, and edge weights represent the connection strength between behaviors. Based on this graph representation, we frame CVR estimation as a graph classification problem over PMG instances, and propose a novel CVR model, namely, Graph-based Micro-behaviors Conversion Model (GMCM), that utilizes the Graph Convolutional Networks (GCN) to take advantage of user micro-behaviors. To alleviate sample selection bias, GMCM adopts inverse propensity weighting, which inversely weights the CVR loss with estimated propensities (i.e., CTR scores), and leverage multi-task learning framework by jointly train CTR and CVR tasks together to address the data sparsity problem.

To summarize, the key contributions of this work are as follows:

- We propose to represent the user micro-behaviors as a Purchase-related Micro-behavior Graph (PMG). Based on this graph representation, we take a novel perspective to frame the CVR estimation as a graph classification problem.
- We propose a novel CVR model, i.e., Graph-based Micro-behavior Conversion Model (GMCM), to take advantage of the PMG, and capture the correlation between user micro-behaviors. Mutli-task learning frame work is leveraged to mitigate the severe data sparsity problem, and inverse propensity weighting technique is adopted, to alleviate the sample selection bias.
- We conduct extensive experiments on six production datasets that collected from the e-commerce platform Taobao. On all datasets, the proposed method GMCM consistently outperforms the baselines and the state-of-the-art models, in terms of several widely-adopted metrics.

2 PRELIMINARIES

In this section, we briefly review the basics of Graph Convolutional Networks (GCN) and inverse propensity weighting (IPW) technique.

2.1 Graph Convolutional Networks

GCN works with a spectral representation of the graphs and has been successfully applied in many applications such as node classification, social networks, and recommender systems [25, 30, 31]. GCN defines the graph convolution operation in Fourier domain by computing the eigen-decomposition of graph Laplacian [36]. Let $x \in \mathbb{R}^N$ with a filter $g_\theta = \text{diag}(\theta)$ parameterized by $\theta \in \mathbb{R}^N$:

$$g_\theta \star x = \mathbf{U} g_\theta(\mathbf{\Lambda}) \mathbf{U}^T x, \tag{1}$$

where $\star$ denotes the graph convolution operation, $\mathbf{U}$ and $\mathbf{\Lambda}$ denote the eigenvectors and eigenvalues' matrix of the normalized graph Laplacian $\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, and $\mathbf{I}_N$, $\mathbf{D}$, $\mathbf{A}$ are the identity matrix, degree matrix, adjacency matrix, respectively. [13] generalizes the operation to signal $\mathbf{X} \in \mathbb{R}^{N \times C}$ with C input channels and F filter as follow,

$$\mathbf{Z} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{\Theta}, \tag{2}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$, $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$, $\mathbf{\Theta} \in \mathbb{R}^{C \times F}$ denotes the matrix of filter parameters and $\mathbf{Z} \in \mathbf{R}^{N \times F}$ is the convolved signal matrix.

2.2 Inverse Propensity Weighting

Inverse propensity weighting (IPW) inversely weights the prediction error for each observed sample with the propensity of observing that sample [22]. Let $\mathcal{U} = (u_1, u_2, u_3, ..., u_N)$ be a set of users

and $\mathcal{I} = (i_1, i_2, i_3, ..., i_M)$ be a set of items. $\mathcal{D} = \mathcal{U} \times \mathcal{I}$ denotes the user-item pairs. Let $\mathbf{R} \in \mathbb{R}^{N \times M}$ be the true rating matrix and $\hat{\mathbf{R}} \in \mathbb{R}^{N \times M}$ be the prediction matrix where each entry $\hat{r}_{u,i}$ is a predicted rating computed by the prediction model. If the true rating matrix $\mathbf{R}$ is fully observed, the prediction inaccuracy, $\mathcal{P}$, over all user-item pairs can be measured as follow,

$$\mathcal{P} = \mathcal{P}(\mathbf{R}, \hat{\mathbf{R}}) = \frac{1}{|\mathcal{D}|} \sum_{(u,i) \in \mathcal{D}} e(r_{u,i}, \hat{r}_{u,i}), \tag{3}$$

where $e(r_{u,i}, \hat{r}_{u,i})$ denotes the prediction error. However, in practice, the true rating matrix $\mathbf{R}$ could only be partially observed. Let $\mathbf{O} \in \{0, 1\}^{N \times M}$ be the indicator matrix where each entry $o_{u,i}$ is an observation indicator, and let $\mathbf{R}^{obs}$ and $\mathbf{R}^{mis}$ be the set of observed and missing entries in the true rating matrix $\mathbf{R}$. The estimated prediction inaccuracy could be calculated as follow,

$$\begin{aligned} \mathcal{E}^{obs} &= \mathcal{E}(\mathbf{R}^{obs}, \hat{\mathbf{R}}) \\ &= \frac{1}{|\mathbf{O}|} \sum_{(u,i) \in \mathcal{D}} o_{u,i} e(r_{u,i}, \hat{r}_{u,i}), \end{aligned} \tag{4}$$

The IPW-based methods use inverse propensity to weight the prediction error, which leads to,

$$\begin{aligned} \mathcal{E}^{IPW} &= \mathcal{E}(\mathbf{R}, \hat{\mathbf{R}}) \\ &= \frac{1}{|\mathcal{D}|} \sum_{(u,i) \in \mathcal{D}} \frac{o_{u,i} e(r_{u,i}, \hat{r}_{u,i})}{\hat{p}_{u,i}}. \end{aligned} \tag{5}$$

It can be proved that given the propensity is accurate, the IPW-based estimations are unbiased, i.e., $|\mathbb{E}_{\mathbf{O}}[\mathcal{E}] - \mathcal{P}| = 0$. In this work, we adopt IPW technique to address selection bias issue in CVR estimation. More details are discussed in Section 4.

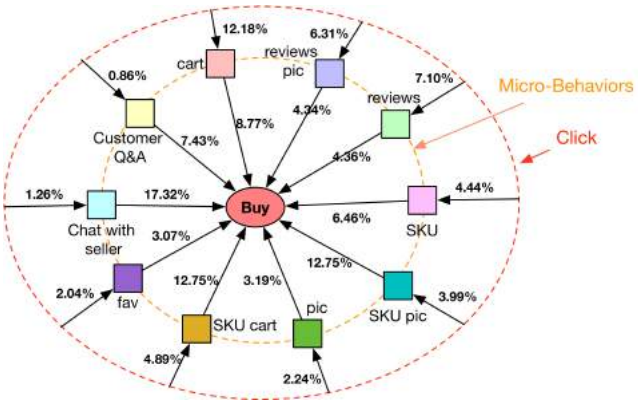


Figure 2: Transition probability among clicks, micro-behaviors and purchases. the outer circle denotes the click event, and inner circle denotes 10 types of micro-behaviors. The arrow stands for transition probability. Probabilities are calculated based on over 400 million transaction logs in our production systems.

3 PURCHASE-RELATED MICRO-BEHAVIOR GRAPH

In this section, we define the purchase-related micro-behavior graph (PMG) over user micro-behaviors after clicking a product. we start with discussion of what purchase-related micro-behaviors are, and their high relevance with the user purchase decision. Then we give a formal definition of PMG. Lastly, we discuss how to frame CVR estimations as a graph classification problem over PMG instances.

Purchase-related micro-behaviors denote the users' purchase-related actions after clicking a product, e.g., *add to cart, read reviews*, etc. Such implicit feedback provides us with fine-grained understanding of the user decision-making process, and helps CVR models better infer the users' buying preference. In Figure 2, we show 10 kinds of micro-behaviors and their transition probability to the final purchase. Figure 2 reflects the strong connection between micro-behaviors and the user purchase decision, and highlights the importance of micro-behaviors modeling in the CVR estimation.

Then we formally define the purchase-related micro-behavior graph (PMG). Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denotes the undirected graph PMG with N nodes $v_i \in \mathcal{V}$, M edges $(v_i, v_j) \in \mathcal{E}$. Define the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, and degree matrix $\mathbf{D}$, where each entry A_{ij} stands for the edge weight of (v_i, v_j), and each entry $\mathbf{D}_{ii} = \sum_j A_{ij}$. Let $\mathcal{M}$ be the set of all micro-behavior events collected by the log engine. Then we have $\mathcal{V} \subseteq \mathcal{M}$, indicating that every node in PMG stands for one type of micro-behavior events. To determine the edge weight, let concurrency matrix of micro-behaviors be $\mathbf{C} \in \mathbb{R}^{N \times N}$ and concurrent probability matrix be $\mathbf{P} \in \mathbb{R}^{N \times N}$, where each entry c_{ij} = #(concurrency of micro-behavior i and j in logs), and each entry $p_{ij} = \frac{c_{ij}}{\sum_j c_{ij}}$. Then for each edge $(v_i, v_j) \in \mathcal{E}$, $p_{ij} \in \mathbf{P}$ denotes the initial weight of the edge (v_i, v_j). To further enable the model adjust edge weights during the training process, we define a learnable edge bias matrix $\mathbf{B} \in \mathbb{R}^{N \times N}$, and let $\mathbf{A} = \mathbf{P} + \mathbf{B}$. Figure 3 shows an instance of PMG. Note that the buy button in Figure 3 only denotes the micro-behavior of clicking the buy button, which does not necessarily lead to the final conversion. SKU stands for stock keeping unit, which represents the different type of a product, e.g., *size, color*, etc.

Next we frame CVR estimation as a graph classification problem over the PMG instance. Let $\mathcal{Y}^{\text{CVR}}$ and $\mathcal{Y}^{\text{node}}$ be the label set of conversions and micro-behaviors in PMG. Suppose the input feature matrix is $\mathbf{X}$, and a PMG instance is $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Then the CVR prediction matrix over the PMG instance is defined as,

$$\hat{\mathbf{R}} = f(\mathcal{E}, \mathcal{V}, \mathbf{X}), \tag{6}$$

where f is a differentiable predicting function, $\hat{\mathbf{R}} = [\hat{\mathbf{R}}^{\text{node}} | \hat{\mathbf{R}}^{\text{CVR}}]$, and $|$ denotes the matrix concatenation. Given the prediction matrix, the loss of estimated node scores and CVR are,

$$\begin{aligned} L^{\text{node}} &= l(\hat{\mathbf{R}}^{\text{node}}, \mathcal{Y}^{\text{node}}) \\ L^{\text{CVR}} &= l(\hat{\mathbf{R}}^{\text{CVR}}, \mathcal{Y}^{\text{CVR}}), \end{aligned} \tag{7}$$

where l denotes the loss function, and the loss of CVR estimation over the PMG instance is then defined as,

$$L^{\text{PMG}} = h(L^{\text{node}}, L^{\text{CVR}}), \tag{8}$$

where h is the merging function of two losses.

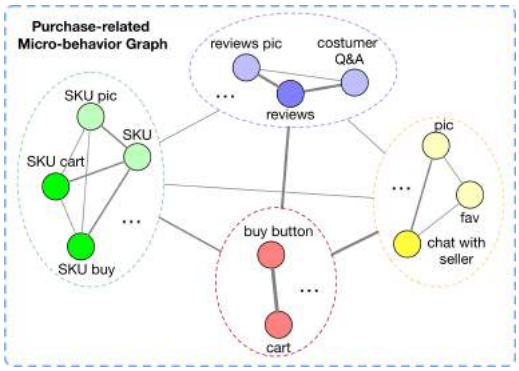


Figure 3: Illustration of the Purchase-related Micro-behavior Graph (PMG). Each node denotes one micro-behavior type, and thickness of edge represents the connection strength. SKU stands for stock keeping unit. Note that the clustering of nodes is only for drawing convenience.

4 GRAPH-BASED MICRO-BEHAVIORS CONVERSION MODEL

In this section, we propose the graph-based micro-behavior conversion model (GMCM). We firstly present the motivation, then discuss the key components of GMCM in detail. Figure 4 depicts the architecture of the proposed method.

4.1 Motivation

GMCM is devised to address three practical issues in conventional CVR modeling, i.e., limited purchase-related feedback, data sparsity and sample selection bias.

To address the first issue, GMCM takes advantage of user micro-behaviors by modeling them over PMG instances, and adopts GCN to capture their correlations. In practice, we find that these micro-behaviors are highly-related to the user purchase decision, thus may serve as a clue of the user buying intention. Besides, we observe that micro-behaviors often correlate with each other, and different orders of micro-behaviors may indicates similar user buying intention. Therefore, we represent the micro-behaviors as a graph rather than a sequence. To address data sparsity and sample selection bias issues, we leverage mutli-task learning framework, and adopt inverse propensity weighting technique. Specifically, GMCM trains CTR and CVR task together, and shares embedding layer between the two tasks. Therefore, the large amount of ID embedding parameters could be trained with both exposed and clicked samples. Besides, GMCM inversely weights the CVR prediction error with estimated propensities (i.e., CTR scores), to alleviate sample selection bias. Note that IPW-based methods are theoretically unbiased contingent on the propensities are accurately estimated. In practice, such constraint is often too restricted. However, existing works [32, 33] show that even if the propensities are accurately estimated, IPW-based method still can mitigate sample selection bias issue to some degree.

4.2 Multi-task Learning Module

GMCM chains CTR and CVR tasks together and trains them in parallel. Concretely, the two tasks have independent input layer, and a shared embedding layer. Let the shared embedding matrix be $\mathbf{W}_e \in \mathbb{R}^{d_h \times d_e}$, where d_h denotes the hush bucket size and d_e denotes the embedding dimension. Define the ID feature matrices $\mathbf{X}_s^{\text{CTR}} \in \mathbb{R}^{B \times F_s^{\text{CTR}}}$, $\mathbf{X}_s^{\text{CVR}} \in \mathbb{R}^{B \times F_s^{\text{CVR}}}$, where B is the mini batch size, F_s is the input dimension of ID features. Then the embedded ID features are represented as,

$$\begin{aligned} \mathbf{X}_e^{\text{CTR}} &= f_h(X_s^{\text{CTR}}, W_e) \\ \mathbf{X}_e^{\text{CVR}} &= f_h(X_s^{\text{CVR}}, W_e) \end{aligned} \tag{9}$$

where $X_e \in \mathbb{R}^{B \times F_e}$, f_h denotes the hash function, and F_e is the embedding dimension of ID feature matrix X_s. Then define dense feature matrices $\mathbf{X}_d^{\text{CTR}} \in \mathbb{R}^{B \times F_d^{\text{CTR}}}$, $\mathbf{X}_d^{\text{CVR}} \in \mathbb{R}^{B \times F_d^{\text{CVR}}}$, where F_d denotes the input dimension of all dense features. Then we have the concatenated feature representation,

$$\begin{aligned} X_{\text{concat}}^{\text{CTR}} &= [X_d^{\text{CTR}} | \mathbf{X}_e^{\text{CTR}}] \\ X_{\text{concat}}^{\text{CVR}} &= [X_d^{\text{CVR}} | \mathbf{X}_e^{\text{CVR}}] \end{aligned} \tag{10}$$

where | denotes the matrix concatenation, and $X_{\text{concat}} \in \mathbb{R}^{B \times (F_d + F_e)}$ is concatenated input features after the shared embedding layer.

The remaining part of CTR networks is several layers of MLP with the leaky Relu activation function. Suppose that the logits of CTR output layer is $\mathbf{o} \in \mathbb{R}^B$, then for each sample the predicted CTR is given by,

$$\hat{r_i}^{\text{CTR}} = \text{sigmoid}(\mathbf{o_i}), \tag{11}$$

and the loss of CTR networks is,

$$L^{\text{CTR}} = \frac{1}{B} \sum_{i=1}^{B} l(y_i^{\text{CTR}}, r_i^{\text{CTR}}), \tag{12}$$

where $y_i \in \mathcal{Y}^{\text{CTR}}$ is the CTR label, and l stands for cross entropy loss.

4.3 Graph-based CVR Networks

The CVR networks consist of multiple MLP layers, a node embedding layer, and GCN.

4.3.1 MLP Layers. Let $H_k \in \mathbb{R}^{B \times h_k}$ and h_k denote the hidden representation and the dimension of k^{th} layer. Then we can define a single layer of MLP as,

$$\mathbf{H}_k^{\text{MLP}} = \sigma(\mathbf{H}_{k-1}^{\text{MLP}} \Theta_k^{\text{MLP}}) \tag{13}$$

where σ is leaky relu activation function, and Θ_k is parameters of k^{th} layer. By stacking multiple MLP layers, GMCM produces common input representation for the node embedding layer.

4.3.2 Node Embedding Layer. GMCM makes use of user micro-behaviors to enhance conventional CVR modeling. For each node v_i, a single layer of MLP is utilized to generate the node embedding. Suppose that there are N nodes in PMG, and $\mathbf{H}^{\text{MLP}}$ denotes the output of final layer of MLP with hidden dimention h_K. For each node v_i, the node embedding is formulated as,

$$\mathbf{e}_i = \sigma(\mathbf{H}^{\text{MLP}} \Theta_i^{\text{node-MLP}}), \mathbf{e}_i \in \mathbb{R}^{d_{\text{node}}} \tag{14}$$

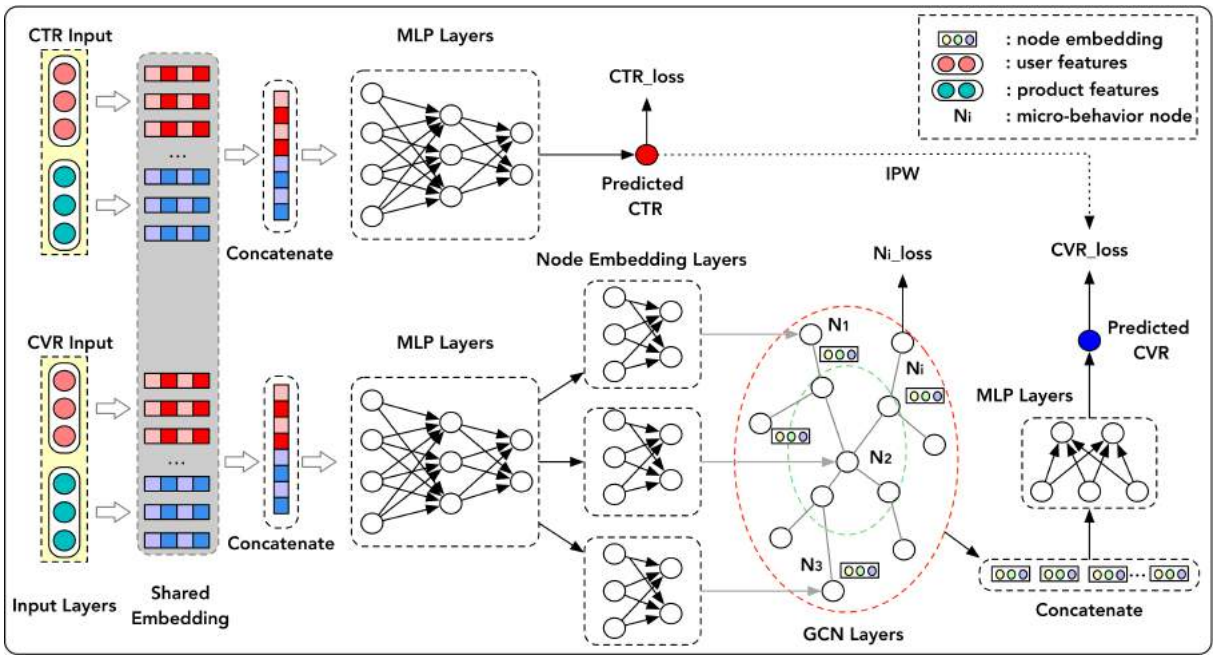


Figure 4: The architecture of Graph-based Micro-behaviors Conversion Model (GMCM).

where d_{node} is the embedding dimension, and $\Theta_i^{\text{node-MLP}} \in \mathbb{R}^{h_K \times d_{\text{node}}}$ denotes parameters of node embedding layer for node v_i.

4.3.3 Graph Convolutional Networks. GMCM exploits GCN to convolve the node embedding and capture their correlations. Define an instance of undirected graph PMG as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes $v_i \in \mathcal{V}$ and M edges $(v_i, v_j) \in \mathcal{E}$. Define the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, and degree matrix $\mathbf{D}$, where each entry A_{ij} denote the edge weight of (v_i, v_j), and each entry $\mathbf{D}_{ii} = \sum_j A_{ij}$. Recall that by introducing the concurrent probability matrix $\mathbf{P}$ and edge bias matrix $\mathbf{B}$, the adjacency matrix $\mathbf{A} = \mathbf{P} + \mathbf{B}$. thus a single layer of GCN can be defined as,

$$\mathbf{H}_l^{\text{GCN}} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}_{l-1}^{\text{GCN}}\mathbf{\Theta}_l^{\text{GCN}}), \tag{15}$$

where $\mathbf{H}_l^{\text{GCN}} \in \mathbb{R}^{B \times N \times h_l}$ and $\Theta_l^{\text{GCN}} \in \mathbb{R}^{h_{l-1} \times h_l}$ denotes the hidden representation and parameters of the l^{th} GCN layer, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$, and $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$.

For each node v_i, GMCM outputs a predicted score $\hat{r}_i^{\text{node}}$, indicating the probability of the user having that micro-behavior after clicking the product. Besides, the hidden representations of all nodes are aggreated to generate the predicted CVR $\hat{r}^{\text{CVR}}$. Formally, suppose that the convolved feature tensor of the last GCN layer is $\mathbf{H}^{\text{GCN}} \in \mathbb{R}^{B \times N \times h^{\text{GCN}}}$. For each sample, let $\mathbf{h}_i^{\text{GCN}} \in \mathbb{R}^{h^{\text{GCN}}}$ be the representation of node v_i. Then the predicted micro-behavior score $\hat{r}_i^{\text{node}}$ is calculated as,

$$\hat{r}_i^{\text{node}} = \text{sigmoid}(\mathbf{h}_i^{\text{GCN}}\Theta_i^{\text{GCN-logit}}), \tag{16}$$

where $\Theta_i^{\text{GCN-logit}} \in \mathbb{R}^{h^{\text{GCN}}}$ is the parameters of logit layer for node v_i. And for predicted CVR $\hat{r}^{\text{CVR}}$ we have,

$$\hat{r}^{\text{CVR}} = \text{sigmoid}(f_{\text{output}}(\mathbf{H}^{\text{GCN}})\Theta_{\text{CVR}}^{\text{GCN-logit}}), \tag{17}$$

where $\Theta_{\text{CVR}}^{\text{GCN-logit}} \in \mathbb{R}^{h^{\text{GCN}}}$ is the parameters of CVR logit layer, and f_{output} denotes the output function, e.g., *concatenation, sum pooling, mean pooling*, etc.

4.3.4 Loss Layer. Given the estimated CVR scores $\hat{\mathbf{r}}^{\text{CVR}} \in \mathbb{R}^B$, micro-behaviors scores $\hat{\mathbf{R}}^{\text{node}} \in \mathbb{R}^{B \times N}$, and estimated CTR scores $\hat{\mathbf{r}}^{\text{CTR}} \in \mathbb{R}^B$, the loss of graph-based CVR networks over the PMG instance is defined as,

$$L^{\text{PMG}} = \frac{1}{B}\sum_{i=1}^{B}\frac{1}{\hat{r}_i^{\text{CTR}}}(l(y_i^{\text{CVR}}, \hat{r}_i^{\text{CVR}}) + \frac{1}{N}\sum_{j=1}^{N} l(y_{ij}^{\text{node}}, \hat{r}_{ij}^{\text{node}})), \tag{18}$$

where l denotes the cross entropy loss, $y_i^{\text{CVR}} \in \mathcal{Y}^{\text{CVR}}$ and $y_{ij}^{\text{node}} \in \mathcal{Y}^{\text{node}}$ denote the label of CVR and micro-behaviors. Note that the CVR loss is weighted by inverse predicted CTR scores, to alleviate sample selection bias. In practice, the inverse CTR scores may lead to numerical stability problem. Thus we self-normalize the predicted CTR scores when weighting CVR loss [24]. Lastly, the loss of GMCM can be defined as,

$$L^{\text{GMCM}} = L^{\text{CTR}} + L^{\text{PMG}}. \tag{19}$$

5 EXPERIMENTS

In this section, we evaluate the performance of GMCM with six production datasets collected from Mobile Taobao, the leading e-commerce platform in China. The experiments are intended to answer the following questions:

- **Q1**: Does GMCM outperform other state-of-art CVR estimation methods?
- **Q2**: Does GCN help to better utilize the micro-behaviors feedback?

Table 1: Statistics of six production datasets.

Dataset	total	# clicks	# buys	# users	# products	# training	# testing
SM-A	1.3B	215M	10M	41M	99M	1.1B	200M
SM-B	2.5B	412M	19M	63M	124M	2.2B	200M
SM-C	3.6B	604M	27M	80M	143M	3.4B	200M
GUL-A	390M	77M	7M	29M	44M	330M	59M
GUL-B	744M	147M	13M	47M	58M	685M	59M
GUL-C	1.1B	220M	20M	63M	68M	1B	59M

- **Q3**: How is the performance of GMCM affected by hyper-parameters of GCN?

5.1 Datasets

We collect six production datasets from the Mobile Taobao. Table 1 summarizes the statistics of all 6 subsets. All datasets contain 258 features, including 230 dense features, e.g., product price, click times, conversion times, etc., and 28 sparse features, e.g., user age, user gender, etc. The datasets contain 3-week consecutive logs (2020-02-15 to 2020-03-07) from two different recommendation applications: Guess You Like inShop(GUL), and Shopping More inShop(SM). For SM datasets, there are totally 3.6 billion data samples, including 604 million clicks, 27 million conversions, and 80 million users. For GUL datasets, there are totally 1.1 billion data samples, including 220 million clicks, 20 million conversions and 63 million users. We further split these 3-week logs of SM and GUL into 6 subsets:

- **Set SM-A** contains data from 2020-03-01 to 2020-03-06 as training set, and data in 2020-03-07 as testing set. Set SM-A contains approximately 30% of the SM data samples.
- **Set SM-B** contains data from 2020-02-23 to 2020-03-06 as training set, and data in 2020-03-07 as testing set. Set SM-B contains approximately 70% of the SM data samples.
- **Set SM-C** contains data from 2020-02-15 to 2020-03-06 as training set, and data in 2020-03-07 as testing set. Set SM-C contains all SM data samples.
- **Set GUL-A** contains data from 2020-03-01 to 2020-03-06 as training set, and data in 2020-03-07 as testing set. Set GUL-A contains approximately 30% of the GUL data samples.
- **Set GUL-B** contains data from 2020-02-23 to 2020-03-06 as training set, and data in 2020-03-07 as testing set. Set GUL-B contains approximately 70% of the GUL data samples.
- **Set GUL-C** contains data from 2020-02-15 to 2020-03-06 as training set, and data in 2020-03-07 as testing set. Set GUL-C contains all GUL data samples.

5.2 Compared Methods

To evaluate the performance of proposed method, We compare GMCM with the following models. Note that all compared models are based on multi-task learning framework and share the embedding layer. Hence, the data sparsity issue is mitigated to some degree.

- **Base** is a co-trained CTR and CVR networks. It has separate input layer and a shared embedding layer.
- **Division** [18] estimates CVR by CTCVR = CTR × CVR. Different from [18], we adapt Division in our experiments with co-trained CTR and CVR networks and shared embedding layer, rather than individually trained networks.
- **ESMM** [18] reduces the CVR estimation to two auxiliary tasks, i.e., CTR task and CTCVR task. By successfully adopting the multi-task learning and entire space modelling, ESMM could tackle both data sparsity and sample selection bias issues, thus is deemed as the sate-of-the-art CVR model in practice.
- **ResNet** [7] Residual Neural Networks (ResNet) introduces identity shortcut connection that allows one or more layers being skipped. ResNet has achieved great success in areas like computer vision. We adapt ResNet for CVR estimation by co-training CTR and CVR tasks with ResNet.
- **DCN** [26] Deep and Cross Networks (DCN) introduces a novel cross networks that is more efficient in learning certain bounded-degree feature interactions. We adapt DCN for CVR estimation by co-training CTR and CVR tasks with DCN.

5.3 Evaluation Metrics

We adopt three widely used metrics in our experiments: AUC, GAUC and MSE. Generally, AUC and GAUC indicates the ranking performance, and MSE reflects the prediction inaccuracy.

In CTR and CVR estimations, area under receiver operator curve (AUC) is a widely used metric [5]. One interpretation of AUC in the context of ranking system is that it denotes the probability of ranking a random positive sample higher than a negative sample. Meanwhile, we also adopt Group AUC (GAUC) in our assessments [35]. GAUC extends AUC by calculating the weighted average of AUC grouped by a certain property, e.g., user, page, etc. GAUC can be calcuated as,

$$\text{GAUC} = \frac{\sum_i w_i \times \text{AUC}_i}{\sum_i w_i}, \tag{20}$$

where w_i is the weight of certain group. GAUC is commonly recognized as a more indicative metric for ranking performance [35]. Besides, we utilize MSE to measure the prediction inaccuracy. MSE is defined as,

$$\text{MSE} = \frac{1}{|O|} \sum_{(u,i) \in O} (r_{u,i} - \hat{r}_{u,i})^2 \tag{21}$$

where O is the observed user-item rating pairs.

5.4 Hyper-parameter Settings

5.4.1 Common Settings. all competitors in the experiments use batch size of 2048, Adam Optimizer [12] with $\beta_1 = 0.9$,$\beta_2 = 0.999$. Learning rate is tuned and set to be 0.0001, and same embedding dimension of sparse features is used for all competitors. All models except for GMCM shares the network architecture of $(512, 256, 128, 64)$ for both CTR and CVR tasks. To stabilize the training process, all models use batch normalization [8].

5.4.2 Individual Settings. For DCN, five cross layers are used besides the MLP, following the setting in [26]. For GMCM, we set the architecture of CTR networks as $(512, 256, 128, 64)$, same with other baselines. For CVR networks, we adopt a three layers of MLP with the shape of $(512, 256, 128)$, and a node embedding layer with dimension of 64. We search for the number of GCN layers in $(2, 3, 4)$, and the dimension of hidden units in $(16, 32, 64, 128, 160)$. The best performance is reported with 2 GCN layers and 64 hidden units for each layer.

Table 2: Model performance evaluated by AUC/GAUC on six production datasets. The best scores are in boldface, and the second best scores are underlined. RI indicates the relative improvement of GMCM compared with baseline models in permillage. Note that to save space, lower case for AUC, GAUC, CVR, CTCVR are used in this table.

Model	SM-A(1.3B)				SM-B(2.5B)				SM-C(3.6B)			
	cvr auc	ctcvr auc	ctcvr gauc	RI	cvr auc	ctcvr auc	ctcvr gauc	RI	cvr auc	ctcvr auc	ctcvr gauc	RI
Base	0.8151	0.8278	0.7032	9.71‰	0.8249	0.8357	0.7063	7.28‰	0.8255	0.8374	0.7103	8.83‰
ESMM	0.8163	0.8283	0.7051	8.11‰	0.8263	0.8359	0.7077	5.94‰	0.8305	0.8412	0.7119	4.53‰
Division	0.8155	0.8271	0.6987	12.05‰	0.8208	0.8317	0.7072	10.15‰	0.8243	0.8344	0.7117	9.91‰
ResNet	0.8146	0.8230	0.6959	15.46‰	0.8233	0.8322	0.7002	12.27‰	0.8272	0.8364	0.7037	11.69‰
DCN	0.8169	0.8304	0.7021	8.47‰	0.8250	0.8349	0.7047	8.29‰	0.8313	0.8420	0.7115	4.08‰
GMCM	**0.8183**	**0.8332**	**0.7164**	-	**0.8300**	**0.8417**	**0.7123**	-	**0.8340**	**0.8446**	**0.7157**	-
Model	GUL-A(390M)				GUL-B(744M)				GUL-C(1.1B)			
	cvr auc	ctcvr auc	ctcvr gauc	RI	cvr auc	ctcvr auc	ctcvr gauc	RI	cvr auc	ctcvr auc	ctcvr gauc	RI
Base	0.7982	0.8335	0.7620	9.98‰	0.8059	0.8447	0.7746	8.83‰	0.8110	0.8435	0.7723	10.37‰
ESMM	0.7973	0.8298	0.7634	11.22‰	0.8066	0.8421	0.7728	10.32‰	0.8080	0.8441	0.7767	9.46‰
Division	0.7885	0.8268	0.7591	18.08‰	0.8030	0.8405	0.7692	14.04‰	0.8047	0.8423	0.7686	15.07‰
ResNet	0.7934	0.8321	0.7609	13.02‰	0.8050	0.8402	0.7683	13.76‰	0.8112	0.8491	0.7738	7.39‰
DCN	0.7969	0.8337	0.7639	9.59‰	0.8080	0.8451	0.7732	8.41‰	0.8088	0.8433	0.7721	11.47‰
GMCM	**0.8067**	**0.8422**	**0.7687**	-	**0.8180**	**0.8521**	**0.7767**	-	**0.8183**	**0.8529**	**0.7807**	-

Table 3: Model performance evaluated by MSE on six production datasets. The best scores are in boldface, and the second best scores are underlined. RI indicates the relative improvement of GMCM compared with baseline models in permillage.

Model	SM-A(1.3B)			SM-B(2.5B)			SM-C(3.6B)		
	CVR MSE	CTCVR MSE	RI	CVR MSE	CTCVR MSE	RI	CVR MSE	CTCVR MSE	RI
Base	0.04496	0.008125	11.02‰	0.04423	0.008097	4.25‰	0.04395	0.008094	7.89‰
ESMM	0.04513	0.008143	13.98‰	0.04417	0.008103	3.94‰	0.04394	0.008090	7.485‰
Division	0.04470	0.008111	7.32‰	0.04454	0.008156	11.35‰	0.04400	0.008103	8.96‰
ResNet	0.04482	0.008141	10.49‰	0.04431	0.008108	5.84‰	0.04381	0.008081	5.37‰
DCN	0.04434	0.008101	2.65‰	0.04444	0.008102	6.88‰	0.04345	**0.008057**	-0.12‰
GMCM	**0.04411**	**0.008099**	-	**0.04395**	**0.008079**	-	**0.04340**	0.008068	-
Model	GUL-A(390M)			GUL-B(744M)			GUL-C(1.1B)		
	CVR MSE	CTCVR MSE	RI	CVR MSE	CTCVR MSE	RI	CVR MSE	CTCVR MSE	RI
Base	0.0778	0.01739	13.41‰	0.0770	0.01731	21.98‰	0.0758	0.01717	9.73‰
ESMM	0.0774	0.01745	12.17‰	0.0774	0.01737	26.44‰	0.0762	0.01721	13.06‰
Division	0.0811	0.01764	40.51‰	0.0787	0.01745	36.30‰	0.0783	0.01740	32.08‰
ResNet	0.0783	0.01749	18.97‰	0.0765	0.01727	17.73‰	**0.0745**	**0.01706**	-2.16‰
DCN	0.0782	0.01745	17.84‰	0.0768	0.01732	21.11‰	0.0764	0.01739	19.89‰
GMCM	**0.0763**	**0.01726**	-	**0.0746**	**0.01708**	-	0.0748	0.01708	-

5.5 Analysis on Overall Performance (Q1)

In this section, we analyze the overall performance of the proposed method and other compared models. The experiment results are summarized in table 2 and table 3. Based on the results, we have following observations:

- GMCM consistently outperforms other baseline models across all six datasets in term of AUC/GAUC, and achieves the lowest MSE in most of the experiments. Recall that GMCM is devised to better tackle three practical issues in conventional CVR estimations, i.e., limited purchase-related feedback, data sparsity and sample selection bias. In the experiments, all models are based on multi-task learning framework and share embedding layers between CTR and CVR tasks, thus the data sparsity issue is mitigated to some degree. Compared with baseline models, however, GMCM makes use of users' micro-behaviors feedback, and leverages GCN to better model the user decision-making process. Besides, the IPW technique is adopted to alleviate sample selection bias. Therefore, we believe the performance boost of GMCM owing to the better modeling of micro-behaviors with GCN, and also the adoption of IPW method.
- ESMM is a strong competitor, and deemed as the state-of-the-art CVR model in practice. ESMM proposed a novel method of entire space modeling to mitigate the sample selection bias issue. For GMCM, we adopt the IPW technique, which is proved theoretically unbiased, and empirically tested better than ESMM in [32].
- DCN is another strong baseline and achieves decent performance in most of the experiments. For dataset SM-C, DCN even outperforms GMCM in term of CVR MSE, and has a pretty close CTCVR MSE to GMCM. Recall that DCN utilizes

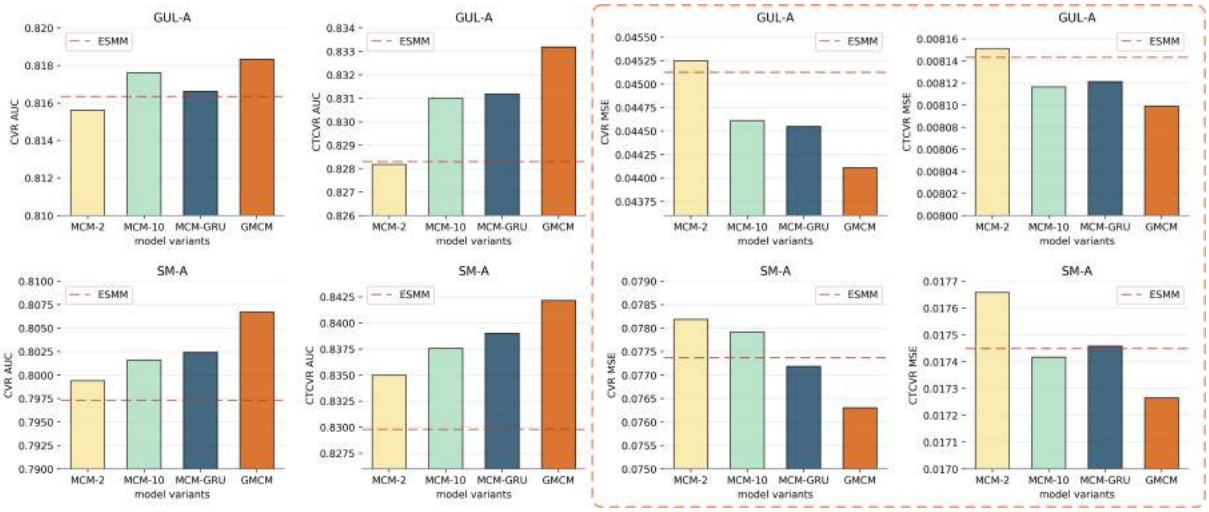


Figure 5: Ablation study of GMCM and two variant models on GUL-A and SM-A. The red dashed line denotes the performance of ESMM. Note that figures in the orange dashed box are evaluated by MSE, which means the best performance is represented by the shortest bar.

cross networks to automatically capture the feature interactions and enhance the representation ability of vanilla MLP. Generally, the good performance of DCN in our experiments indicates that changing vanilla MLP with other types of networks such as DCN, may further boost the performance of a co-trained CVR networks.

- We observe that in nearly all experiments, the model performance is improved as more training data is used. Moreover, the performance boost achieved by using more data is sometimes greater than that of using different models. We reason that in production datasets hundreds of features are used, and the samples contain millions of users and products. Therefore, the training generally requires more data to converge. Such observation suggests that in industry-level CVR estimation, the size of training data plays a key role in enhancing the model performance, and collecting more training data is as important as choosing the right model.

5.6 Ablation Study (Q2)

To further verify the benefits of utilizing user micro-behaviors, and the effectiveness of graph-based modeling, we conduct ablation experiments to compare GMCM with the following variant models. Note that MCM stands for Micro-behaviors Conversion Model, a simple variant of GMCM without using the GCN module. In MCM, all the node embedding is simply concatenated to predict the CVR, hence neither sequential information nor the correlation of nodes are taken into account.

- MCM-2: Only the two most easily accessible micro-behaviors are used, i.e., add to cart and add to favorite list.
- MCM-10: 10 micro-behaviors are used, same as GMCM.
- MCM-GRU: It replaces GCN with a sequence model. Specifically, after the node embedding layer, it adopts GRU to capture the sequential information of micro-behaviors, and uses hidden representation of the last GRU layer to predict CVR. Note that MCM-GRU is used to compare graph-based and sequence-based modeling of micro-behaviors.

Figure 5 shows the performance of GMCM and different variants. the state-of-the-art CVR model ESMM is also included as a baseline. Based on the experiment results, we have the following observations:

- Compare MCM-10 with MCM-2, we find that the model performance in terms of all three metrics is improved as more micro-behaviors are taken into account. MCM-2 only utilizes the feedback of adding to cart and favorite list, while MCM-10 incorporates more feedback such as reading reviews, zooming in pictures, etc. The results of MCM-10 and MCM-2 indicates that providing the model with more purchase-related feedback from users will generally boost the model performance.
- GMCM outperforms MCM-10 and MCM-GRU in all the experiments. Compared with MCM-10 and MCM-GRU, GMCM adopts GCN to model the micro-behaviors and capture their correlation. In practice, we find that different orders of micro-behaviors may represent similar user buying intention. Besides, the behaviors often correlate with each other. For example, the user who reads more reviews has a high probability also reads the customer Q&A. Hence, the micro-behaviors are better represented as a graph rather than a sequence.

5.7 Hyper-parameter Sensitivity (Q3)

We conduct experiments on how the performance of GMCM is affected by two important hyper-parameters of GCN, i.e., the number of GCN layers and hidden units. We search for the number of GCN layers in (2, 3, 4), and the number of hidden units in (16, 32, 64, 128, 160). The experiment results are summarized in Figure 6. From the figures we could easily observe that GMCM has the best performance when GCN has two layers, and hidden units number is between

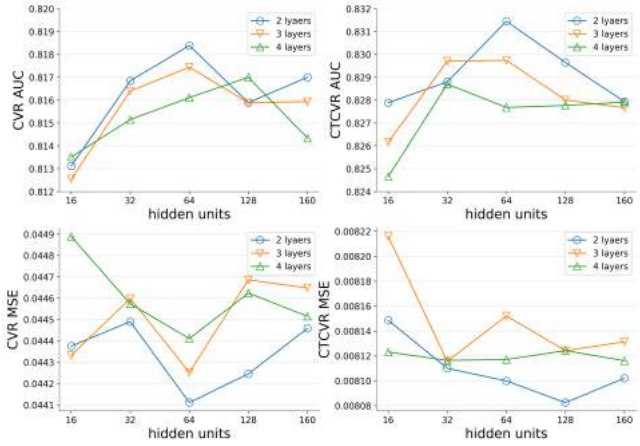


Figure 6: Experiment results of hyper-parameter sensitivity on SM-A. Note that the lower two figures are evaluated by MSE, which indicates the bottom plots have the best performance.

64 and 128. We could clearly see the performance drops when the GCN layer number increases, and when too many or too few hidden units are used. We reason that since the edge weights are learned during the training phase, too many GCN layers may make the representation of each node aggregate too much information from the uncorrelated nodes, hence introducing noise during training.

6 RELATED WORK

GMCM is devised to make use of purchase-related micro-behaviors, and tackle the practical issues of data sparsity and sample selection bias in CVR estimation. In what follows, we briefly review the existing works that most relates to these aspects.

6.1 General Approach to CVR Estimation

Due to the inherent similarity, many CTR models are adopted for CVR estimation in practice, such as logistic regression [11, 21], factorization machines (FM) [9, 10, 20], deep neural networks (DNN) [1–3, 6], methods that combines DNN with FM [15, 16], and recently, sequence models that utilize user historical behaviors [34, 35]. Few literatures directly study on the CVR estimation task. Lee *et al.* [14] proposed a hierarchical CVR model with separate binomial distributions for different feature levels and integrated these individual estimators via logistic regression. Shan *et al.* [23] proposed combined regression and triplet-wise ranking method (CRT) to jointly consider regression loss and triplet-wise ranking loss for CVR estimation. Wen *et al.* [29] proposed a deep cascade tree structure to ensemble Gradient Boost Decision Tree (GBDT) and improve the feature representation ability.

6.2 Data Sparsity and Sample Selctioin Bias

Two practical issues, i.e., data sparsity and sample selection bias, may degrade CVR estimators under industrial setting. Several studies have been carried out to tackle these challenges [18, 19, 28, 32, 33]. For example, Gary *et al.* [28] proposed the oversampling method that copied rare class samples to mitigate the data sparsity issue. Pan *et al.* [19] proposed All Missing As Negtive(AMAN) which applied random sampling on un-clicked samples. Zhang *et al.* [33] proposed to achieve unbiased CTR estimation via reject sampling on the underlying distribution of the observation. Recently, Ma *et al.* proposed Entire Space Multi-task Model (ESMM) [18], which aimed for addressing both data sparsity and sample selection bias issues. Specifically, ESMM exploited the sequential pattern of online shopping, "exposure → click → purchase", and reduced the CVR estimation to two auxiliary tasks, i.e., CTR estimation and click-through and conversion rate (CTCVR) estimation. By sharing the embedding layer between tasks, data sparsity issue is mitigated. It also alleviated the sample selection bias via entire space modeling. Zhang *et al.* [32] provided a causal perspective of CVR estimation. Two causal estimators, i.e., Multi-IPW and Multi-DR were proposed to tackle the data sparsity and sample selection bias in concert.

6.3 Micro-behaviors Based Methods

Very few existing works study on how to utilize user micro-behaviors in recommender systems. To our knowledge, [37] is the very first attempt to make use of such implicit feedback, and RNN-based method was adopted to model micro-behaviors. However, we argue that the sequence-based modeling could be inappropriate in some cases. Firstly, different orders of micro-behaviors may represent similar user buying intention. For example, when the user considering buying a product, the order of reading reviews and costumer Q&A does not make much difference. Besides, we find micro-behaviors are often inter-connected, while the sequence model could fail to capture such correlations. In this work, we propose to represent the micro-behaviors as a purchase-related micro-behavior graph (PMG), and adopt GCN to take advantage of such graph. The comparative study of graph-based and sequence-based modeling of micro-behaviors can be found in Section 5.6.

7 CONCLUSION AND FUTURE WORK

In this work, we propose GMCM, a novel graph-based CVR estimator, to tackle three challenging issues in industrial CVR estimation: limited purchase-related feedback, data sparsity and sample selection bias. We propose to represent the user micro-behaviors as a purchase-related micro-behavior graph (PMG), and based on this graph representation, we frame the CVR estimation as a graph classification problem over PMG instances. Mutli-task learning and inverse propensity weighting are also adopted to mitigate the data sparsity and sample selection bias. Experiments on six production datasets demonstrate that GMCM consistently outperforms other strong competitors including the state-of-the-art CVR methods.

Two meaningful directions deserve more attention in the future works. Firstly, what kind of micro-behaviors should be utilized. In our experiments, top 10 behaviors that are most related to the purchase are selected. However, more than a hundred are observed in our logs. How to choose the right micro-behaviors deserves further study. Secondly, the losses of micro-behaviors, CTR and CVR are simply added for optimization in this work. More sophisticated fusion strategy may further improve the model performance.

REFERENCES

[1] Junxuan Chen, Baigui Sun, Hao Li, Hongtao Lu, and Xian-Sheng Hua. 2016. Deep ctr prediction in display advertising. In *Proceedings of the 24th ACM International Conference on Multimedia*. 811–820.

[2] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. 7–10.

[3] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 191–198.

[4] Arnaud De Myttenaere, Bénédicte Le Grand, Boris Golden, and Fabrice Rossi. 2014. Reducing offline evaluation bias in recommendation systems. *23rd annual Belgian-Dutch Conference on Machine Learning* (2014).

[5] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27, 8 (2006), 861–874.

[6] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence* (2017).

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[8] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15)*. JMLR.org, 448–456.

[9] Yuchin Juan, Damien Lefortier, and Olivier Chapelle. 2017. Field-aware factorization machines in a real-world online advertising system. In *Proceedings of the 26th International Conference on World Wide Web Companion*. 680–688.

[10] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 43–50.

[11] Muhammad Junaid Effendi and Syed Abbas Ali. 2017. Click Through Rate Prediction for Contextual Advertisment Using Linear Regression. *International Journal of Computer Science and Information Security (IJCSIS)* (2017).

[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations* (2014).

[13] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *Fivth International Conference on Learning Representations* (2017).

[14] Kuang-chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. 2012. Estimating conversion rate in display advertising from past erformance data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 768–776.

[15] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1754–1763.

[16] Weiwen Liu, Ruiming Tang, Jiajin Li, Jinkai Yu, Huifeng Guo, Xiuqiang He, and Shengyu Zhang. 2018. Field-aware probabilistic embedding neural network for ctr prediction. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 412–416.

[17] Quan Lu, Shengjun Pan, Liang Wang, Junwei Pan, Fengdan Wan, and Hongxia Yang. 2017. A practical framework of conversion rate prediction for online display advertising. In *Proceedings of the ADKDD'17*. 1–9.

[18] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1137–1140.

[19] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 502–511.

[20] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[21] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th International Conference on World Wide Web*. 521–530.

[22] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments: debiasing learning and evaluation. *Proceedings of the 33rd International Conference on Machine Learning* (2016).

[23] Lili Shan, Lei Lin, and Chengjie Sun. 2018. Combined Regression and Tripletwise Learning for Conversion Rate Prediction in Real-Time Bidding Advertising. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 115–123.

[24] Adith Swaminathan and Thorsten Joachims. 2015. The self-normalized estimator for counterfactual learning. In *Advances in Neural Information Processing Systems*. 3231–3239.

[25] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. *Sixth International Conference on Learning Representations* (2018).

[26] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.

[27] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. 2019. Doubly Robust Joint Learning for Recommendation on Data Missing Not at Random. *Proceedings of Machine Learning Research* (2019).

[28] Gary M. Weiss. 2004. Mining with Rarity: A Unifying Framework. *SIGKDD Explor. Newsl.* 6, 1 (June 2004), 7–19. https://doi.org/10.1145/1007730.1007734

[29] Hong Wen, Jing Zhang, Quan Lin, Keping Yang, and Pipei Huang. 2019. Multi-Level Deep Cascade Trees for Conversion Rate Prediction in Recommendation System. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 338–345.

[30] Le Wu, Peijie Sun, Richang Hong, Yanjie Fu, Xiting Wang, and Meng Wang. 2018. Socialgcn: An efficient graph convolutional network based model for social recommendation. *arXiv preprint arXiv:1811.02815* (2018).

[31] Jiani Zhang, Xingjian Shi, Shenglin Zhao, and Irwin King. 2019. STAR-GCN: stacked and reconstructed graph convolutional networks for recommender systems. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 4264–4270.

[32] Wenhao Zhang, Wentian Bao, Xiao-Yang Liu, Keping Yang, Quan Lin, Hong Wen, and Ramin Ramezani. 2020. A Causal Perspective to Unbiased Conversion Rate Estimation on Data Missing Not at Random. *the Web Conference* (2020).

[33] Weinan Zhang, Tianxiong Zhou, Jun Wang, and Jian Xu. 2016. Bid-aware gradient descent for unbiased learning with censored data in display advertising. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 665–674.

[34] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5941–5948.

[35] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.

[36] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434* (2018).

[37] Meizi Zhou, Zhuoye Ding, Jiliang Tang, and Dawei Yin. 2018. Micro behaviors: A new Perspective in E-commerce Recommender Systems. In *Proceedings of the eleventh ACM International Conference on Web Search and Data Mining*. 727–735.

Entire Space Multi-Task Modeling via Post-Click Behavior Decomposition for Conversion Rate Prediction

Hong Wen*
Alibaba Group
Hangzhou, Zhejiang, China 311121
qinggan.wh@alibaba-inc.com

Jing Zhang*
The University of Sydney
Darlington NSW 2008, Australia
jing.zhang1@sydney.edu.au

Yuan Wang
Alibaba Group
Hangzhou, Zhejiang, China 311121
wy175696@alibaba-inc.com

Fuyu Lv
Alibaba Group
Hangzhou, Zhejiang, China 311121
fuyu.lfy@alibaba-inc.com

Wentian Bao
Alibaba Group
Hangzhou, Zhejiang, China 311121
wentian.bwt@alibaba-inc.com

Quan Lin, Keping Yang
Alibaba Group
Hangzhou, Zhejiang, China 311121
{tieyi.lq,shaoyao}@taobao.com

ABSTRACT

Recommender system, as an essential part of modern e-commerce, consists of two fundamental modules, namely Click-Through Rate (CTR) and Conversion Rate (CVR) prediction. While CVR has a direct impact on the purchasing volume, its prediction is well-known challenging due to the Sample Selection Bias (SSB) and Data Sparsity (DS) issues. Although existing methods, typically built on the user sequential behavior path "impression→click→purchase", is effective for dealing with SSB issue, they still struggle to address the DS issue due to rare purchase training samples. Observing that users always take several purchase-related actions after clicking, we propose a novel idea of post-click behavior decomposition. Specifically, disjoint purchase-related Deterministic Action (DAction) and Other Action (OAction) are inserted between click and purchase in parallel, forming a novel user sequential behavior graph "impression→click→D(O)Action→purchase". Defining model on this graph enables to leverage all the impression samples over the entire space and extra abundant supervised signals from D(O)Action, which will effectively address the SSB and DS issues together. To this end, we devise a novel deep recommendation model named Elaborated Entire Space Supervised Multi-task Model (ESM^2). According to the conditional probability rule defined on the graph, it employs multi-task learning to predict some decomposed sub-targets in parallel and compose them sequentially to formulate the final CVR. Extensive experiments on both offline and online environments demonstrate the superiority of ESM^2 over state-of-the-art models. The source code and dataset will be released.

CCS CONCEPTS

• **Computer systems organization** → **Neural networks**; • **Information systems** → **Recommender systems**.

*Both authors contributed equally to this paper.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGIR '20, July 25–30, 2020, Virtual Event, China
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401443

KEYWORDS

Recommender System, Entire Space Multi-Task Learning, Post-Click Behavior Decomposition, Conversion Rate Prediction

ACM Reference Format:
Hong Wen, Jing Zhang, Yuan Wang, Fuyu Lv, Wentian Bao, and Quan Lin, Keping Yang. 2020. Entire Space Multi-Task Modeling via Post-Click Behavior Decomposition for Conversion Rate Prediction. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401443

1 INTRODUCTION

Discovering valuable products or services from massive available options on the Internet for users has become a fundamental functionality in modern online applications such as e-commerce [1, 21, 26, 28], social networking [7, 20], advertising [34, 35], *etc*. Recommender System (RS) can serve this role to provides accurate, timely, and personalized services to users [4, 17, 37]. Figure 1 shows the architecture of online recommendation in e-commerce platform. It consists of two phases, *i.e*., system recommendation and user feedback. After analyzing users' long and short-term behaviors, RS first recalls a large number of related items. Then, they are ranked and exposed to users according to several ranking metrics, *e.g*., Click-Through Rate (CTR) [34, 35], Conversion Rate (CVR) [19, 28] , *etc*. Next, when going through the recommended items, users may click on and eventually purchase the interested ones, indicating a typical user sequential behavior path "impression→click→purchase" for e-commerce transaction [19]. These feedback from users are collected by RS and used to estimate more accurate ranking metrics, which are indeed very crucial for generating high-quality recommendations in turn. In this paper, we focus on the post-click CVR estimation task.

However, two critical issues in the CVR estimation makes the task quite challenging, *i.e*., Sample Selection Bias (SSB) [31] and Data Sparsity (DS) [15]. SSB refers to the systematic difference of data distributions between training space and inference space, *i.e*., conventional CVR models are trained only on clicked samples while being used for inference on all impression samples. Intuitively, clicked samples are only a very small portion of the impression samples and are biased by user self-selection (such as clicking). Therefore, when the CVR model serving online, the SSB issue will degrade its performance. Besides, due to the relatively rare clicking

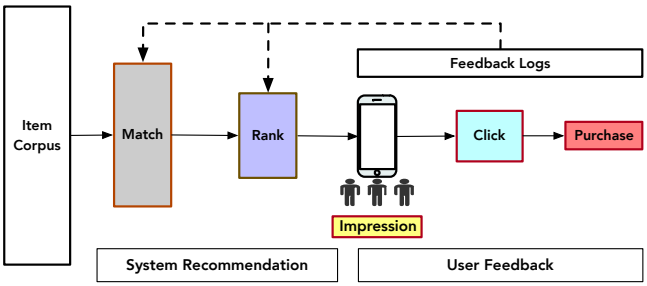


Figure 1: A diagram of online recommendation in e-commerce platform, comprising of two fundamental components, *i.e.*, system recommendation and user feedback.

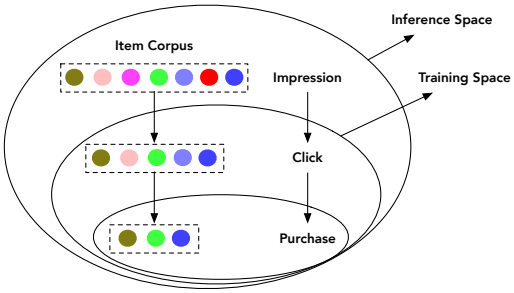


Figure 2: Illustration of sample selection bias problem in conventional CVR prediction, where training space only composes of clicked samples, while inference space is the entire space for all impression samples. And data volume gradually decreased from impression to purchase.

samples compared with impressions, the number of training samples from the sequential behavior path "click→purchase" is insufficient to fit the large parameter space of CVR task, which results in the DS problem. As illustrated in Figure 2, how to deal with the SSB and DS problems is crucial for developing an efficient industrial-level recommender system.

Several studies have been carried out to tackle these challenges [15, 19, 22, 27, 32]. For example, Ma *et al*. propose a new model named Entire Space Multi-Task Model (ESMM) [19], which defines CVR task on the user sequential behavior path "impression→click→purchase" via multi-task learning framework. It is trained with all impression samples over the entire space for two auxiliary tasks namely post-view CTR and post-view click-through conversion rate (CTCVR). Therefore, the derived CVR from CTR and CTCVR is also applicable in the same entire space when inferring online, thus addressing the SSB issue effectively. Besides, an auxiliary CTR network with rich labeled samples shares the same feature representation with the CVR network, helping to alleviate the DS issue. Although ESMM achieves better performance than conventional methods by dealing with the SSB and DS issues simultaneously, it still struggles to alleviate the DS issue due to the rare purchase training samples, *i.e.*, less than 0.1% of impression behaviors converts to purchase according to the large scale real transaction logs from our e-commerce platform.

After a detailed analysis of the logs, we observe that users always take some purchase-related actions after clicking. For example, users may add the preferred items to their shopping cart (or wish list) instead of immediately purchases due to some reasons (*i.e.*, waiting for a discount). Besides, these actions are indeed more abundant than purchase actions. Motivated by this, we propose a novel idea of post-click behavior decomposition. Specifically, disjoint purchase-related Deterministic Action (DAction) and Other Action (OAction) are inserted between click and purchase in parallel, forming a novel user sequential behavior graph "impression→click→D(O)Action→purchase", where the task relationship is explicitly defined by the conditional probability. Besides, defining model on this graph enables to leverage all impression samples over the entire space and extra abundant supervisory signals from post-click behaviors, efficiently addressing the SSB and DS issues.

In this paper, we resort to deep neural networks to embody the above idea. Specifically, we propose a novel deep neural recommendation model named Elaborated Entire Space Supervised Multi-task Model (ESM^2), which consists of three modules: 1) a shared embedding module (SEM), 2) a decomposed prediction module (DPM), and 3) a sequential composition module (SCM). First, SEM embeds a one-hot feature vector of ID features into dense representation through a linear fully connected layer. Then, these embeddings are fed into the subsequent DPM, where individual prediction network estimates the probabilities of decomposed sub-targets in parallel by employing multi-task learning on all the impression samples over the entire space. Finally, SCM composes the final CVR as well as some auxiliary probabilities sequentially according to the conditional probability rule defined on the graph. Multiple losses defined on some sub-paths of the graph are used to supervise the training of ESM^2.

The main contributions of this paper are summarized as follows:

- To the extent of our knowledge, we are the first to introduce the idea of post-click behavior decomposition to model CVR over the entire space. The explicit decomposition results in a novel user sequential behavior graph "impression→click→D(O)Action→purchase".
- We propose a novel deep neural recommendation method named ESM^2, which models CVR prediction and auxiliary tasks simultaneously in a multi-task learning framework according to the conditional probability rule defined on the user behavior graph. ESM^2 can address the SSB and DS issues efficiently by harvesting the abundant post-click action data with labels.
- Our model achieves better performance on the real-world offline dataset than representative state-of-the-art methods. We also deploy it in our online recommender system and achieve significant improvement, confirming its value in industrial applications.

The rest of this paper is organized as follows. Section 2 presents a brief survey of related work, followed by the details of the proposed model in Section 3. Experiment results and analysis are presented in Section 4. Finally, we conclude the paper in Section 5.

2 RELATED WORK

Our proposed method specifically tackles the conversion rate prediction problem by employing the multi-task learning framework over the entire space. Therefore, we briefly review the most related

work from the following two aspects: 1) conversion rate prediction and 2) multi-task learning.

Conversion Rate Prediction: Conversion rate prediction is a key component of many online applications, such as search engines [2, 33], recommender systems [10, 23] and online advertising [8, 12]. However, there are few literatures directly proposed for CVR tasks [16, 28, 30], regardless of recent prosperous development of CTR methods [3, 29, 34, 35]. Indeed, CVR modeling is very challenging since conversions are extremely rare events that only a very small portion of impression items are eventually being clicked and bought. Recently, the deep neural network has achieved significant progress in many areas including recommender systems due to its remarkable ability in feature representation and end-to-end modeling [4, 9, 13, 14, 17, 24]. In this paper, we also adopt a deep neural network to model the conversion rate prediction task. In contrast to the above methods, we derive a new user sequential behavior graph "impression→click→D(O)Action→purchase" based on a novel idea of post-click behavior decomposition. According to the conditional probability rule defined on the graph, our network structure is specifically devised to predict several decomposed sub-targets in parallel and compose them sequentially to formulate the final CVR.

Multi-Task Learning: Due to the temporal multi-stage nature of users' purchasing behavior, *e.g.*, impression, click, and purchase, prior work attempts to formulate the conversion rate prediction task by a multi-task learning framework. For example, Hadash *et al*. propose a multi-task learning-based recommender system by modeling the ranking and rating prediction tasks simultaneously [11]. Ma *et al*. propose a multi-task learning approach named multi-gate mixture-of-experts to explicitly learn the task relationship from data [18]. Gao *et al*. propose a neural multi-task recommendation model to learn the cascading relationship among different types of behaviors [6]. In contrast, we model the CTR and CVR tasks simultaneously by associating with users' sequential behavior graph, where the task relationship is explicitly defined by the conditional probability (See Section 3). Ni *et al*. propose to learn universal user representations across multiple tasks for more effective personalization [21]. We also explore such an idea by sharing embedded features across different tasks. Recently, Ma *et al*. propose an entire space multi-task model (ESMM) for CVR prediction [19]. It adds the CTR task and CTCVR task as an auxiliary to the main CVR task. Our method is partially inspired by ESMM but has the following significant difference: we propose a novel idea of post-click behavior decomposition to reformulate a novel user sequential behavior graph "impression→click→D(O)Action→purchase". Defining model on this graph enables to formulate the final CVR as well as some auxiliary tasks together. It can leverage all the impression samples over the entire space and the abundant supervisory signals from users' post-click behaviors, which are highly relevant to the purchase behaviors, consequently addressing the SSB and DS issue simultaneously.

3 PROPOSED METHOD

3.1 Motivation

In practice, from an item being displayed to it being purchased successfully, we identify that there may exist multiple kinds of sequential actions a user could choose to do. For example, after clicking one interested item, a user may directly purchase it without any hesitation, or add it to the shopping cart and then make the purchase eventually. These behavior paths are shown in Figure 3(a). We can simplify and group these paths according to several predefined specific purchase-related post-click actions, *i.e.*, adding to Shopping Cart (SCart) and adding to Wish list (Wish), as shown in Figure 3(b). Based on our data analysis of online real-world logs, we found that only 1% of clicked behaviors are converted to purchase eventually, indicating rare purchase training samples. However, the data volume of several post-click actions like SCart and Wish are much larger than purchase. For example, 10% will be added to the shopping cart given clicked behaviors. Besides, these post-click actions are highly relevant to the final purchase action, *e.g.*, 12% (or 31%) will be bought eventually after they have been added to the shopping cart (or wish list). How can we leverage the larger volume of post-click behaviors to benefit CVR prediction in some manner, regarding their high relevance to purchase?

Intuitively, one solution is to model these purchase-related post-click actions along with purchase into a multi-task prediction framework. The key is how to formulate them properly since they have explicit sequential correlations, *e.g.*, the purchase action probably conditioned on the SCart or Wish action. To this end, we define a single node named Deterministic Action (DAction) to merge these predefined specific purchase-related post-click actions, such as SCart and Wish, as shown in Figure 3(c). DAction has two properties: 1) it is highly relevant to the purchase action and 2) it has abundant deterministic supervisory signals from users' feedback, *e.g.*, 1 for taking some specific actions (*i.e.*, adding to shopping cart or wish list after clicking) and 0 for none. We also add a node named Other Action(OAction) between click and purchase to deal with other post-click behaviors except DAction. In this way, the conventional behavior path "impression→click→purchase" becomes to a novel elaborated user sequential behavior graph "impression→click→D(O)Action→purchase", as shown in Figure 3(c). Defining model on this graph enables to leverage all the impression samples over the entire space and extra abundant supervisory signals from D(O)Action, which will circumvent the SSB and DS issues efficiently. We call this novel idea as post-click behavior decomposition.

3.2 Conditional probability decomposition

In this section, we present the conditional probability decomposition of CVR as well as related auxiliary tasks according to the digraph defined in Figure 3(c). First, the probability of post view click-through rate of an item x_i, denoted as p_i^{ctr}, is defined as the conditional probability of being clicked given that it has been viewed, which depicts the path "impression→click" in the digraph. Mathematically, it can be written as:

$$p_i^{ctr} = p\left(c_i = 1 \,|v_i = 1\right) \triangleq y_{1i}, \tag{1}$$

where $c_i \in C$ denotes whether the i^{th} item x_i is being clicked, $c_i \in \{0, 1\}$, C is the label spaces of all the items being clicked or not, $i \in [1, N]$ and N is the number of items. Similarly, $v_i \in V$ denotes whether the i^{th} item x_i is being viewed (*i.e.*, impression), $v_i \in \{0, 1\}$, V is the label spaces of all the items being viewed or not. y_{1i} is a surrogate symbol for simplicity.

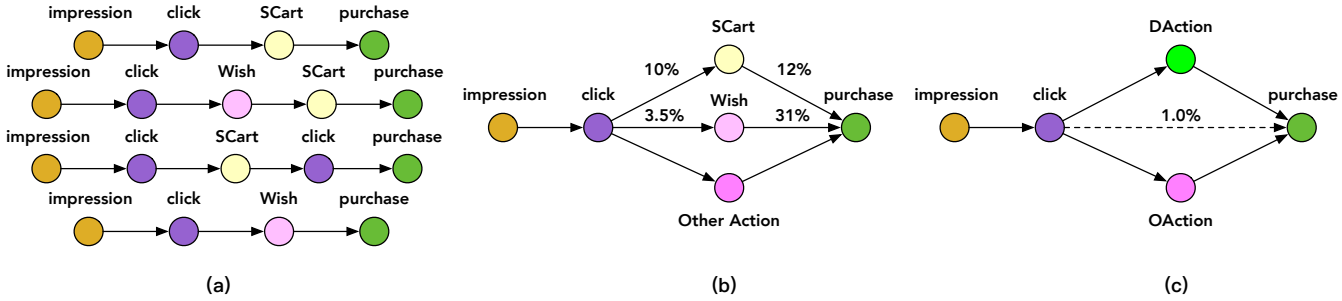


Figure 3: Illustration of the proposed user sequential behavior graph based on post-click behavior decomposition. (a) The multiple paths from impression to purchase after distinguishing post-click behaviors, such as "impression→click→SCart→purchase". (b) A digraph is used to describe the simplified purchasing process, where the numbers above edges represent the sparsity of different paths. (c) Several specific purchase-related post-click actions are merged into a single node, *i.e.*, DAction, which also inherits their supervisory signals. OAction represents other cases except DAction.

Then, the probability of click-through DAction conversion rate of an item x_i, denoted as p_i^{ctavr}, is defined as the conditional probability of being taken DAction given that it has been viewed, which depicts the path "impression→click→DAction" in the digraph. Mathematically, it can be written as:

$$\begin{aligned} p_i^{ctavr} &= p\left(a_i = 1 \,|v_i = 1\right) \\ &= \sum_{c_i \in \{0,1\}} p\left(a_i = 1 \,|v_i = 1, c_i\right) p\left(c_i \,|v_i = 1\right) \\ &= p\left(a_i = 1 \,|v_i = 1, c_i = 0\right) p\left(c_i = 0 \,|v_i = 1\right) \\ &\quad + p\left(a_i = 1 \,|v_i = 1, c_i = 1\right) p\left(c_i = 1 \,|v_i = 1\right) \\ &= y_{2i} y_{1i} \end{aligned}, \tag{2}$$

where $a_i \in A$ denotes whether the i^{th} item x_i is being taken some specific actions defined in Section 3.1, $a_i \in \{0,1\}$, A is the label spaces of all the items being taken some specific actions or not. $y_{2i} = p\left(a_i = 1 \,|v_i = 1, c_i = 1\right)$, depicting the path "click→DAction", is a surrogate symbol for simplicity as y_{1i}. It is trivial that $y_{2i} = p\left(a_i = 1 \,|c_i = 1\right)$ since all the samples are impression samples (*i.e.*, $v_i = 1$). It is noteworthy that Eq. (2) holds due to the fact that no action occurs without being clicked, *i.e.*, $p\left(a_i = 1 \,|v_i = 1, c_i = 0\right) = 0$.

Next, the probability of conversion rate of an item x_i, denoted as p_i^{cvr}, is defined as the conditional probability of being bought given that it has been clicked, which depicts the paths "click→D(O)Action→purchase" in the digraph. Mathematically, it can be written as:

$$\begin{aligned} p_i^{cvr} &= p\left(b_i = 1 \,|c_i = 1\right) \\ &= \sum_{a_i \in \{0,1\}} p\left(b_i = 1 \,|c_i = 1, a_i\right) p\left(a_i \,|c_i = 1\right) \\ &= p\left(b_i = 1 \,|c_i = 1, a_i = 0\right) p\left(a_i = 0 \,|c_i = 1\right) \\ &\quad + p\left(b_i = 1 \,|c_i = 1, a_i = 1\right) p\left(a_i = 1 \,|c_i = 1\right) \\ &\triangleq y_{4i}\left(1 - y_{2i}\right) + y_{2i} y_{3i} \end{aligned}, \tag{3}$$

where $b_i \in B$ denotes whether the i^{th} item x_i is being bought, $b_i \in \{0,1\}$, B is the label spaces of all the items being bought or not. $y_{3i} = p\left(b_i = 1 \,|c_i = 1, a_i = 1\right)$, $y_{4i} = p\left(b_i = 1 \,|c_i = 1, a_i = 0\right)$ are some surrogate symbols for simplicity as y_{1i}. y_{3i} or y_{4i} depicts the path "DAction→purchase" or "OAction→purchase" in the digraph, respectively.

The probability of click-through conversion rate of an item x_i, denoted as p_i^{ctcvr}, is defined as the conditional probability of being bought given that it has been viewed, which depicts the complete graph "impression→click→D(O)Action→purchase" in the digraph. Mathematically, it can be written as:

$$\begin{aligned} p_i^{ctcvr} &= p\left(b_i = 1 \,|v_i = 1\right) \\ &= \sum_{c_i} p\left(b_i = 1 \,|v_i = 1, c_i\right) p\left(c_i \,|v_i = 1\right) \\ &= \sum_{c_i} \sum_{a_i} p\left(b_i = 1 \,|v_i, c_i, a_i\right) p\left(a_i \,|v_i, c_i\right) p\left(c_i \,|v_i\right) \cdot \\ &= y_{4i}\left(1 - y_{2i}\right) y_{1i} + y_{3i} y_{2i} y_{1i} \\ &= y_{1i}\left(y_{4i}\left(1 - y_{2i}\right) + y_{3i} y_{2i}\right) \end{aligned} \tag{4}$$

Here, we use v_i to replace $v_i = 1$ in the third equality for simplicity without causing any ambiguity. It is noteworthy that the fourth equality holds due to the fact that no items will be bought without being clicked, *i.e.*, $p\left(b_i = 1 \,|v_i = 1, c_i = 0, a_i\right)$ equals to zero, $\forall a_i \in \{0,1\}$. Indeed, Eq. (4) can be derived by decomposing the graph "impression→click→D(O)Action→purchase" into "impression→click" and "click→D(O)Action→purchase", and integrating Eq. (1) and Eq. (3) together according to the chain rule, *i.e.*, $p_i^{ctcvr} = p_i^{ctr} * p_i^{cvr}$.

3.3 Elaborated entire space supervised multi-task model

From Eq. (1)~ Eq. (4), we can see that p_i^{ctr}, p_i^{ctavr}, and p_i^{ctcvr} can be derived from four hidden probability variables y_{1i}, y_{2i}, y_{3i}, and y_{4i}, which represents the conditional probabilities over some sub-paths in the graph, *i.e.*, "impression→click", "click→DAction", "DAction→purchase" and "OAction→purchase". On the one hand, these four sub-targets are defined over the entire space and can be predicted using all the impression samples. Taking y_{2i} as an example, training y_{2i} directly with only clicked samples suffers from the SSB issue. Indeed, y_{2i} is an intermediate variable derived

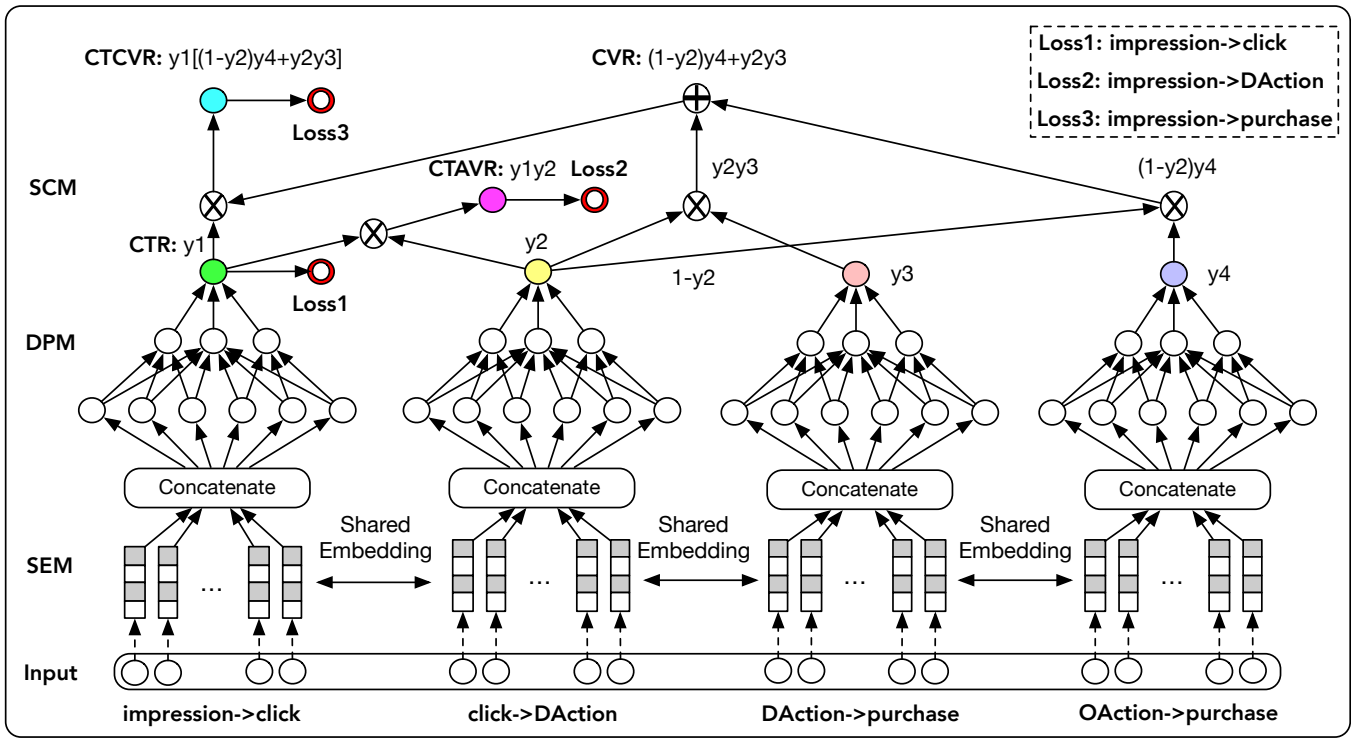


Figure 4: The diagram of ESM^2 model over the entire space, which consists of three key modules: SEM, DPM and SCM. SEM embeds sparse features into dense representation. DPM predicts the probabilities of decomposed targets. SCM integrates them together sequentially to calculate the final CVR as well as other related auxiliary tasks namely CTR, CTAVR, and CTCVR.

from p_i^{ctr} and p_i^{ctavr} according to Eq. (2). Since both p_i^{ctr} and p_i^{ctavr} are modeled over the entire space with all impression samples, the derived y_{2i} is also applicable over the entire space, therefore, no SSB in our model. On the other hand, ground truth labels of p_i^{ctr}, p_i^{ctavr}, and p_i^{ctcvr} are available given users' logs, which can be used to supervise these sub-targets. Therefore, an intuitive way is to model them simultaneously by employing a multi-task learning framework. To this end, we propose a novel deep neural recommendation model named Elaborated Entire Space Supervised Multi-task Model (ESM^2) for CVR prediction. ESM^2 gets its name since 1) p_i^{ctr}, p_i^{ctavr}, and p_i^{ctcvr} are modeled over the entire space and predicted using all the impression samples; 2) the derived p_i^{cvr} from Eq. (3) also benefits from the entire space multi-task modeling, which will be validated in the experiment part. ESM^2 consists of three key modules: 1) a shared embedding module, 2) a decomposed prediction module, and 3) a sequential composition module. We present each of them in detail as follows.

Shared Embedding Module (SEM): First, we devise a shared embedding module to embed all the sparse ID features and dense numerical features coming from user field, item field, and user-item cross field. The user features include users' ID, ages, genders and purchasing powers, *etc*. The item features include items' ID, prices, accumulated CTR and CVR from historical logs, *etc*. The user-item features include users' historical preference scores on items, *etc*. Dense numerical features are first discretized based on their boundary values and then represented as one-hot vectors. Here, we use $f_i = \{f_{ij}, \forall j \in \Lambda_f\}$ to denote the one-hot features of the i^{th} training sample, where Λ_f denotes the index set of all kinds of features. Due to the sparseness nature of one-hot encoding, we employ linear fully connected layers to embed them into dense representation, which can be formulated as:

$$g_{ij} = P_{\theta_j}^T f_{ij}, \tag{5}$$

where P_{θ_j} denotes the embedding matrix for the j^{th} kind of features, θ_j represents the network parameters.

Decomposed Prediction Module (DPM): Then, once all the feature embeddings are obtained, they are concatenated together, fed into several decomposed prediction modules, and shared by each of them. Each prediction network in DPM estimates the probability of the decomposed target on the path "impression→click", "click→DAction", "DAction→purchase", "OAction→purchase", respectively. In this paper, we employ Multi-Layer Perception (*MLP*) as the prediction network. All the non-linear activation function is *ReLU* except the output layer, where we use a *Sigmoid* function to map the output into a probability taking real value from 0 to 1. Mathematically, it can be formulated as:

$$y_{ki} = \sigma\left(\varphi_{\vartheta_k}^k(g_i)\right), \tag{6}$$

where σ denotes the *Sigmoid* function, $\varphi_{\vartheta_k}^k$ denotes the mapping function learned by the kth MLP, ϑ_k denotes its network parameters. For example, as shown in the first *MLP* in Figure 4, it output

the estimated probability y_1, which is indeed the post-view click-through rate.

Sequential Composition Module (SCM): Finally, we devise a sequential composition module to compose the above predicted probabilities sequentially according to Eq. (1) ~ Eq.(4) to calculate the conversion rate p^{cvr} and some auxiliary targets including the post-view click-through rate p^{ctr}, click-through DAction conversation rate p^{ctavr}, and click-through conversion rate p^{ctcvr}, respectively. As shown in the top part of Figure 4, SCM is a parameter-free feed forward neural network which represents the underlying conditional probabilities defined by the purchasing decision digraph in Figure 3.

Remarks: 1) All the tasks share the same embedding, making them be trained with all impression samples, *i.e.*, they are modeled over the entire space, resulting in no SSB issue during the inference phase; 2) the lightweight DPM is strictly regularized by the shared embedding module, which makes up the majority of the trainable parameters; and 3) our model suggests an efficient network design, where SEM can run in parallel, leading to low latency when deployed online.

3.4 Training objective

We use $S = \{(c_i, a_i, b_i; f_i)\}|_{i=1}^{N}$ to denote the training set, where c_i, a_i, b_i, represent the ground truth label whether the i^{th} impression sample is being clicked, taken deterministic actions, and bought. Then, we can define the joint post-view click-through probability of all training samples as follows:

$$p^{ctr} = \prod_{i \in C_+} p_i^{ctr} \prod_{j \in C_-} \left(1 - p_j^{ctr}\right), \tag{7}$$

where C_+ and C_- denote the positive and negative samples in the label space C, respectively. After taking negative logarithm on Eq.(7), we obtain the *logloss* of p^{ctr}, which is widely used in recommender systems, *i.e.*,

$$L_{ctr} = -\sum_{i \in C_+} log p_i^{ctr} - \sum_{j \in C_-} log\left(1 - p_j^{ctr}\right). \tag{8}$$

Similarly, we can obtain the loss function of p^{ctavr} and p^{ctcvr}as follows:

$$L_{ctavr} = -\sum_{i \in A_+} log p_i^{ctavr} - \sum_{j \in A_-} log\left(1 - p_j^{ctavr}\right), \tag{9}$$

and

$$L_{ctcvr} = -\sum_{i \in B_+} log p_i^{ctcvr} - \sum_{j \in B_-} log\left(1 - p_j^{ctcvr}\right). \tag{10}$$

The final training objective to be minimized is defined as:

$$L\left(\Theta\right) = w_{ctr} \times L_{ctr} + w_{ctavr} \times L_{ctavr} + w_{ctcvr} \times L_{ctcvr}, \tag{11}$$

where $\Theta = \left\{\theta_j, \forall j \in \Lambda_f\right\} \cup \{\vartheta_i, i = 1, 2, 3, 4\}$ denotes all the network parameters in ESM^2. w_{ctr}, w_{ctavr}, w_{ctcvr} are loss weights of L_{ctr}, L_{ctavr}, L_{ctcvr}, which are set to 1 in this paper, respectively.

Remarks: 1) Adding intermediate losses to supervise the decomposed sub-tasks can efficiently leverage the abundant labeled data from post-click behaviors, making the model less affected by the DS issue; and 2) all the losses are computed from the view of entire space modeling, effectively addressing the SSB issue.

4 EXPERIMENTS

To evaluate the effectiveness of the proposed ESM^2 model, we conducted extensive experiments on both the offline dataset collected from real-world e-commerce scenarios and online deployment. ESM^2 is compared with some representative state-of-the-art (SOTA) methods including GBDT [5], DNN [13], DNN using over-sampling idea [22] and ESMM [18]. First, we present the evaluation settings including the dataset preparation, evaluation metrics, a brief description of these SOTA methods, and the implementation details. Then, we present the comparison results and analysis. Ablation studies are presented next, followed by the performance analysis on different post-click behaviors.

4.1 Evaluation settings

4.1.1 Dataset preparation. We make the offline dataset by collecting the users' sequential behaviors and feedback logs [1] from our online e-commerce platform, which is one of the largest third-party retail platforms in the world. More than 300 million instances with user/item/user-item features and sequential feedback labels (*e.g.*, whether click, or DAction, or purchase) are filtered out. The statistics of this offline dataset are listed in Table 1. They are further divided into the disjoint training set, validation set, and test set.

Table 1: Statistics of the offline dataset.

Category	#User	#Item	#Impression
Number	13,383,415	10,399,095	326,325,042
Category	#Click	#Purchase	#DAction
Number	20,637,192	226,918	2,501,776

4.1.2 Evaluation metrics. To comprehensively evaluate the effectiveness of the proposed model and compare it with SOTA methods, we adopt three widely used metrics in recommendation and advertising system, *i.e.*, Area Under Curve (AUC), GAUC [35, 36] and F_1 score, where AUC reflecting the ranking ability.

$$AUC = \frac{1}{|S_+|\,|S_-|} \sum_{x^+ \in S_+} \sum_{x^- \in S_-} I\left(\phi\left(x^+\right) > \phi\left(x^-\right)\right), \tag{12}$$

where S_+ and S_- denote the set of positive/negative samples, respectively, $|S_+|$ and $|S_-|$ denote the number of samples in S_+ and S_-, $\phi\left(\cdot\right)$ is the prediction function, $I\left(\cdot\right)$ is the indicator function.

GAUC [36] is calculated as follows. First, all the test data are partitioned into different groups according to the individual user ID. Then, AUC is calculated in every single group. Finally, we average the weighted AUC. Mathematically, GAUC is defined as:

$$GAUC = \frac{\sum_u w_u \times AUC_u}{\sum_u w_u}, \tag{13}$$

where w_u denotes the weight for user u (set as 1 for our offline evaluations). AUC_u denotes the AUC for user u.

Moreover, F_1 score is defined as:

$$F_1 = \frac{2 \times P \times R}{P + R}, \tag{14}$$

[1]To the extent of our knowledge, there are no public datasets suited for this entire space modeling task, we will release our dataset for reproducibility and future research.

where P and R denote the precision and recall, *i.e.*,:

$$P = \frac{TP}{TP + FP}, \tag{15}$$

$$R = \frac{TP}{TP + FN}, \tag{16}$$

where TP, FP, and FN denote the number of true positive, false positive, and false negative predictions, respectively.

4.1.3 Brief description of comparison methods. The representative state-of-the-art methods are described as follows.

• **GBDT** [5]: The gradient boosting decision tree (GBDT) model follows the idea of gradient boosting machine (GBM), is able to produce competitive, highly robust, and interpretable procedures for regression and classification tasks [28]. In this paper, we use it as the representative of non-deep learning-based methods.

• **DNN** [13]: We also implement a deep neural network baseline model, which has the same structure and hyper-parameters with the single branch in ESM^2. Different from ESM^2, it is trained with samples on the path "click→purchase" or "impression→click" to predict conversion rate p^{cvr} or click-through rate p^{ctr}, respectively.

• **DNN-OS** [22]: Due to the data sparsity on the paths "impression→purchase" and "click→purchase", it is hard to train a deep neural network with good generalization. To address this issue, we leverage the *over-sampling* strategy to augment positive samples during training the deep model, named DNN-OS. It has the same structure and hyper-parameters with the above DNN model.

• **ESMM** [19]: For a fair comparison, we use the same backbone structure as the above deep models for ESMM. It directly models the conversion rate on the user sequential path "impression→click→purchase" without considering the purchase-related post-click behaviors.

In a nutshell, the first three methods learn to predict p^{ctr} and p^{cvr} using samples on the path "impression→click" and "click→purchase", respectively, then multiply them together to derive the click-through conversion rate p^{ctcvr}. As for ESMM and our ESM^2, they directly predict p^{ctcvr} and p^{cvr} by modeling them over the entire space.

4.1.4 Hyper-parameters settings. For the GBDT model, the number of trees, depth, minimum instance numbers for splitting a node, the sampling rate of the training set for each iteration, the sampling rate of features for each iteration, and the type of loss function, are set as 150, 8, 20, 0.6, 0.6, and *logistic loss*, respectively, which are chosen according to the AUC score on the validation set. For the deep neural network-based models, they are implemented in TensorFlow using the Adam optimizer. The learning rate is set to 0.0005 and the mini-batch size is set to 1000. Logistic loss is used as the loss function for each prediction task in all models. There are 5 layers in the *MLP*, where the dimension of each layer is set to 512, 256, 128, 32, and 2, respectively, as summarized in Table 2.

4.2 Main results

4.2.1 Comparison on the offline dataset. In this subsection, we report the AUC, GAUC, and F_1 scores of all the competitors on the offline test set. Table 3 summarizes the results of AUC and GAUC. It can be seen that the DNN method achieves gains of 0.0242, 0.0102, 0.0117 for CVR AUC, CTCVR AUC, and CTCVR GAUC over the baseline GBDT model, respectively. It demonstrates the strong representation ability of deep neural networks. Different from the vanilla DNN, DNN-OS utilizes an over-sampling strategy to address the DS issue, achieving a better performance than DNN. As for ESMM, it is modeled on the path "impression→click→purchase", which tries to address the SSB and DS issues simultaneously. Benefiting from modeling over the entire space and the abundant training samples, it outperforms DNN-OS. Nevertheless, ESMM, neglecting the impact of post-click behaviors while being further exploited by the proposed ESM^2, still struggles to address the DS issue due to rare purchase training samples. After predicting some decomposed sub-targets in parallel under a multi-task learning framework, ESM^2 composes them sequentially to formulate the final CVR. As can be seen, it obtains the best scores among all the methods. For example, the gains over ESMM are 0.0088, 0.0101, 0.0145 for CVR AUC, CTCVR AUC, and CTCVR GAUC, respectively. It is worth mentioning that a gain of 0.01 in offline AUC always means a significant increment in revenue for online RS [19, 28].

Table 2: Hyper-parameters of deep neural network-based models including DNN, DNN-OS, ESMM, and ESM^2.

Hyper-parameter	Choice
Loss function	Logistic Loss
Optimizer	Adam
Number of layers in MLP	5
Dimensions of layers in MLP	[512,256,128,32,2]
Batch size	1000
Learning rate	0.0005
Dropout ratio	0.5

Table 3: The AUC and GAUC scores of all methods.

Method	CVR AUC	CTCVR AUC	CTCVR GAUC
GBDT	0.7823	0.8059	0.7747
DNN	0.8065	0.8161	0.7864
DNN-OS	0.8124	0.8192	0.7893
ESMM	0.8398	0.8270	0.7906
ESM^2	**0.8486**	**0.8371**	**0.8051**

As for the F_1 score, we report several values by setting different thresholds for CVR and CTCVR, respectively. First, we sort all the instances in descending order according to the predicted CVR or CTCVR score. Then, due to the sparsity of CVR task (about 1% of the predicted samples are positive), we choose three thresholds namely top@0.1%, top@0.6%, and top@1% to split the predictions into positive and negative groups accordingly. Finally, we calculate the precision, recall, and F_1 scores of these predictions at these different thresholds. Results are summarized in Table 4 and Table 5. A similar trend to Table 3 can be observed. Again, the proposed method ESM^2 achieves the best performance in different settings.

4.2.2 Comparison on online deployment. It is not an easy job to deploy deep network models in our recommender system since it servers hundreds of millions of users every day, *e.g.*, more than 100 million users per second at a traffic peak. Therefore,

Table 4: The Precision, Recall and F_1 scores of all methods for CVR.

	CVR@top0.1%			CVR@top0.6%			CVR@top1%		
Method	Recall	Precision	F1-Score	Recall	Precision	F1-Score	Recall	Precision	F1-Score
GBDT	4.382%	14.348%	6.714%	16.328%	9.894%	12.322%	27.384%	7.384%	11.631%
DNN	4.938%	15.117%	7.445%	17.150%	10.495%	13.021%	28.481%	8.196%	12.729%
DNN-OS	5.383%	15.837%	8.034%	17.381%	10.839%	13.353%	29.032%	8.423%	13.058%
ESMM	5.813%	16.295%	8.570%	18.585%	11.577%	14.267%	29.789%	8.961%	13.777%
ESM^2	**6.117%**	**17.145%**	**9.017%**	**23.492%**	**10.574%**	**14.584%**	**30.032%**	**9.034%**	**13.890%**

Table 5: The Precision, Recall and F_1 scores of all methods for CTCVR.

	CTCVR@top0.1%			CTCVR@top0.6%			CTCVR@top1%		
Method	Recall	Precision	F1-Score	Recall	Precision	F1-Score	Recall	Precision	F1-Score
GBDT	2.937%	0.701%	1.132%	4.870%	0.649%	1.145%	8.894%	0.531%	1.002%
DNN	3.168%	0.851%	1.341%	5.269%	0.768%	1.340%	9.461%	0.643%	1.204%
DNN-OS	3.382%	0.871%	1.385%	5.369%	0.801%	1.395%	9.863%	0.673%	1.260%
ESMM	3.858%	0.915%	1.479%	5.504%	0.828%	1.439%	10.088%	0.691%	1.294%
ESM^2	**4.219%**	**1.001%**	**1.618%**	**5.987%**	**0.900%**	**1.566%**	**10.991%**	**0.753%**	**1.410%**

a practical model is required to make real-time CVR prediction with high throughput and low latency. For example, hundreds of recommendation items for each visitor should be predicted in less than 100 milliseconds in our system. Thanks to the parallel network structure, our model is computationally efficient and can respond to each online request within 20 milliseconds. To make the online evaluation fair, confident, and comparable, each deployed method for an A/B test has involved the same number of users, *i.e.*, millions of users. The results are listed in Figure 5, where we use the GBDT model as the baseline. As can be seen, DNN, DNN-OS, and ESMM achieve comparable performance and outperform the baseline model significantly, while ESMM performs slightly better. As for the proposed ESM^2, the significant margins between it and the above methods demonstrate its superiority. Besides, it contributes to a 3% CVR promotion compared with ESMM, indicating a significant business value for the e-commercial platform.

Remarks: 1) The deep neural network has stronger representation ability than the decision tree-based GBDT; 2) the multi-task learning framework over the entire sample space serves as an efficient tool to address the SSB and DS issues; and 3) based on the idea of post-click behaviors decomposition, ESM^2 efficiently addresses the SSB and DS issues by modeling CVR over the entire space and leveraging abundant supervisory signals from deterministic actions and achieves the best performance.

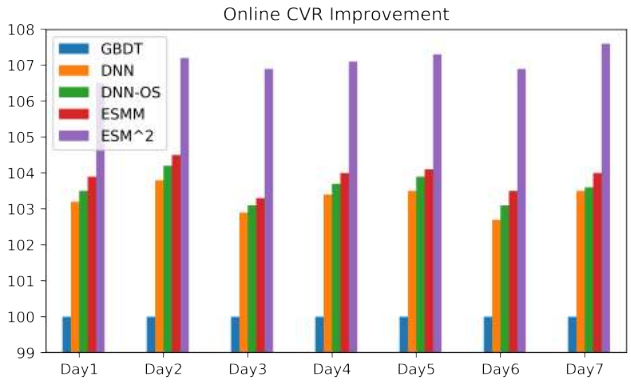


Figure 5: The results of A/B test for CVR by deploying different models in our recommender system.

4.3 Ablation studies

In this part, we present the detailed ablation studies including hyper-parameter settings of the deep neural network, effectiveness of embedding dense numerical features, and the choice of decomposing post-click behaviors, respectively.

4.3.1 Hyper-parameters of deep neural network. Here, we take three critical parameters, namely *dropout ratio*, *the number of hidden layers*, and *the dimension of item feature embeddings* as example to illustrate the process of parameter selection in our ESM^2 model.

Dropout [25] refers to the regularization technique by randomly deactivating some neural nodes during training. It can increase the generalization of the deep neural network by introducing randomness. We try different choices of the dropout ratio from 0.2 to 0.7 in our model. As shown in Figure 6(a), a dropout ratio of 0.5 leads to the best performance. Therefore, we set it to 0.5 in all the experiments if not specified.

Increasing the depth of network layers can enhance the model capacity but also potentially leads to over-fitting. Therefore, we carefully set this hyper-parameter according to the AUC scores on the validation set. As can be seen from Figure 6(b), at the beginning stage, *i.e.*, from two layers to five layers, increasing the number of hidden layers consistently improves the model's performance. However, it saturates at five layers that increasing more layers even marginally decreases the AUC scores, where the model may overfit the training set. Therefore, we use five hidden layers in all experiments if not specified.

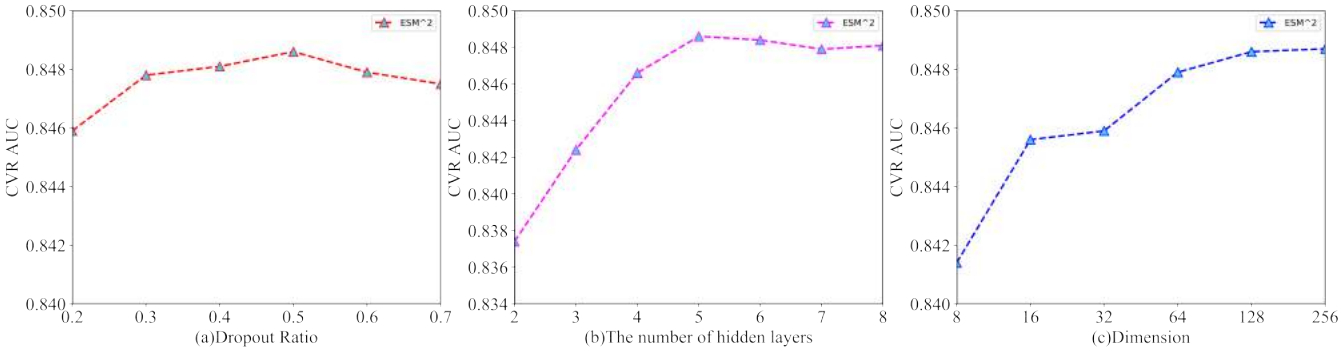


Figure 6: The results of different hyper-parameter settings in ESM^2.

The dimension of item feature embeddings is a critical parameter that high-dimension features reserve more information but also contains noise and leads to higher model complexity. We try different settings of the parameter and plot the results in Figure 6(c). As can be seen, increasing the dimension generally improves performance. It finally saturates at 128 while doubling it leads no more gains. Therefore, to make a trade-off between model capacity and complexity, we set the dimension of item feature embeddings to 128 in all the experiments if not specified.

4.3.2 Effectiveness of embedding dense numerical features. In our task, there are several numerical features. A common practice is to discretize them into one-hot vectors first and then concatenate them with ID features together, which are then embedded into dense features through a linear projection layer as described in Section 3.3. However, we hypothesize that the one-hot vector representation of numerical features may degrade the precision during discretization. In contrast, we try another solution by normalizing the numerical features and embedding them using the Tanh activation function, *i.e.*,

$$g_{ij} = \tanh\left(\frac{f_{ij} - \mu_{f_j}}{\sigma_{f_j}}\right), \tag{17}$$

where μ_{f_j} and σ_{f_j} denotes the mean and standard deviation of the j^{th} kind of features. Then, we concatenate the embedded features with the ID features together as the input of our ESM^2 model. It achieves a gain of 0.004 AUC over the discretization-based method. Therefore, we use the normalization-based embedding method for dense numerical features in all the experiments if not specified.

4.3.3 Effectiveness of decomposing post-click behaviors. When decomposing the post-click behaviors, we can integrate different behaviors into the DAction node, *e.g.*, only SCart, only Wish, or both SCart and Wish (SCart and Wish). Here, we evaluate the effectiveness of different choices. The results are summarized in Table 6. As can be seen, the combination of both SCart and Wish achieves the best AUC scores. It is reasonable since there are more purchase-related labeled data than the other two cases to address the DS issue.

Table 6: The results of choices on post-click behaviors.

	CVR AUC	CTCVR AUC	CTCVR GAUC
SCart	0.8457	0.8359	0.7996
Wish	0.8403	0.8319	0.7962
SCart and Wish	**0.8486**	**0.8371**	**0.8051**

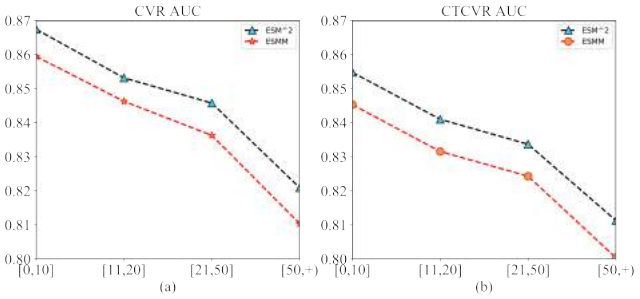


Figure 7: The AUC scores of CVR and CTCVR for ESMM and ESM^2 at different groups regarding the number of purchasing behaviors. Please refer to Section 4.4.

4.4 Performance analysis of user behaviors

To understand the performance of ESM^2 and its difference with ESMM, we further partition the test set into four groups according to the number of users' purchasing behaviors, *i.e.*, [0,10], [11,20], [21,50], [50,+). We report AUC scores of CVR and CTCVR for both methods at each group, and the results are plotted in Figure 7. As can be seen, the CVR AUC(CTCVR AUC) of both methods decreases with the increase of the number of purchasing behaviors. However, we observe that the relative gain of ESM^2 over ESMM in each group increases, *i.e.*, 0.72%, 0.81%, 1.13%, 1.30%. Generally, users having more purchasing behaviors always have more active post-click behaviors such as SCart and Wish. Our ESM^2 model deals with such post-click behaviors by adding a DAction node supervised by deterministic signals from users' feedback. Therefore, it has better representation ability on those samples than ESMM and achieves better performances on the users with high-frequency purchasing behaviors.

5 CONCLUSION

In this paper, we introduce a novel idea of post-click behavior decomposition for modeling CVR task in the context of e-commerce recommender system. A novel user sequential behavior graph "impression→click→D(O)Action→purchase" is constructed, which is used to model CVR over the entire space. Based on the conditional probability rule, we disentangle CVR and some related auxiliary tasks including the post-view click-through rate, click-through DAction conversation rate, and click-through conversion rate into four hidden probability variables, which are defined on explicit sub-paths of the graph. Consequently, we propose a novel deep neural recommendation model named ESM^2 by employing a multi-task learning framework to predict CVR as well as related auxiliary tasks simultaneously. By training with all impression samples and leveraging the abundant labels of deterministic post-click behaviors, our ESM^2 model efficiently addresses the SSB and DS issues. Extensive experiments on both offline and online environments demonstrate the superiority of ESM^2 over state-of-the-art models.

ACKNOWLEDGMENT

This work was partly supported by the National Natural Science Foundation of China (NSFC) under Grant 61806062.

REFERENCES

[1] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior Sequence Transformer for E-commerce Recommendation in Alibaba. *arXiv preprint arXiv:1905.06874* (2019).

[2] Georges E Dupret and Benjamin Piwowarski. 2008. A user browsing model to predict search engine click data from past observations.. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 331–338.

[3] Muhammad Junaid Effendi and Syed Abbas Ali. 2017. Click through rate prediction for contextual advertisment using linear regression. *arXiv preprint arXiv:1701.08744* (2017).

[4] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep Session Interest Network for Click-Through Rate Prediction. *arXiv preprint arXiv:1905.06482* (2019).

[5] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[6] Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, and Depeng Jin. 2019. Neural Multi-Task Recommendation from Multi-Behavior Data. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1554–1557.

[7] Jennifer Golbeck, James Hendler, et al. 2006. Filmtrust: Movie recommendations using trust in web-based social networks. In *Proceedings of the IEEE Consumer communications and networking conference*, Vol. 96. Citeseer, 282–286.

[8] Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, and Ralf Herbrich. 2010. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. Omnipress.

[9] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 6645–6649.

[10] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[11] Guy Hadash, Oren Sar Shalom, and Rita Osadchy. 2018. Rank and rate: multi-task learning for recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 451–454.

[12] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. ACM, 1–9.

[13] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation* 18, 7 (2006), 1527–1554.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[15] Kuang-chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. 2012. Estimating conversion rate in display advertising from past erformance data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 768–776.

[16] Quan Lu, Shengjun Pan, Liang Wang, Junwei Pan, Fengdan Wan, and Hongxia Yang. 2017. A practical framework of conversion rate prediction for online display advertising. In *Proceedings of the ADKDD'17*. 1–9.

[17] Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. 2019. SDM: Sequential deep matching model for online large-scale recommender system. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2635–2643.

[18] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1930–1939.

[19] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 1137–1140.

[20] Jeff Naruchitparames, Mehmet Hadi Güneş, and Sushil J Louis. 2011. Friend recommendations in social networks using genetic algorithms and network topology. In *2011 IEEE Congress of Evolutionary Computation (CEC)*. 2207–2214.

[21] Yabo Ni, Dan Ou, Shichen Liu, Xiang Li, Wenwu Ou, Anxiang Zeng, and Luo Si. 2018. Perceive your users in depth: Learning universal user representations from multiple e-commerce tasks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 596–605.

[22] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 502–511.

[23] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1149–1154.

[24] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*. ACM, 101–110.

[25] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[26] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. *arXiv preprint arXiv:1904.06690* (2019).

[27] Gary M Weiss. 2004. Mining with rarity: a unifying framework. *ACM Sigkdd Explorations Newsletter* 6, 1 (2004), 7–19.

[28] Hong Wen, Jing Zhang, Quan Lin, Keping Yang, and Pipei Huang. 2019. Multi-Level Deep Cascade Trees for Conversion Rate Prediction in Recommendation System. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[29] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617* (2017).

[30] Hongxia Yang, Quan Lu, Angus Xianen Qiu, and Chun Han. 2016. Large scale cvr prediction through dynamic transfer learning of global and local features. In *Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*. 103–119.

[31] Bianca Zadrozny. 2004. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the international conference on Machine learning*.

[32] Weinan Zhang, Tianxiong Zhou, Jun Wang, and Jian Xu. 2016. Bid-aware gradient descent for unbiased learning with censored data in display advertising. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 665–674.

[33] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential click prediction for sponsored search with recurrent neural networks. In *AAAI Conference on Artificial Intelligence*.

[34] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[35] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1059–1068.

[36] Han Zhu, Junqi Jin, Chang Tan, Fei Pan, Yifan Zeng, Han Li, and Kun Gai. 2017. Optimized cost per click in taobao display advertising. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[37] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning Tree-based Deep Model for Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1079–1088.

CIKM 2020

Full Paper Track

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

Deep Time-Aware Item Evolution Network for Click-Through Rate Prediction

Xiang Li, Chao Wang, Bin Tong, Jiwei Tan, Xiaoyi Zeng, Tao Zhuang
Alibaba Group, Hangzhou & Beijing, China
{leo.lx,xiaoxuan.wc,tongbin.tb,jiwei.tjw,yuanhan,zhuangtao.zt}@alibaba-inc.com

ABSTRACT

For better user satisfaction and business effectiveness, Click-Through Rate (CTR) prediction is one of the most important tasks in E-commerce. It is often the case that users' interests different from their past routines may emerge or impressions such as promotional items may burst in a very short period. In essence, such changes relate to item evolution problem, which has not been investigated by previous studies. The state-of-the-art methods in the sequential recommendation, which use simple user behaviors, are incapable of modeling these changes sufficiently. It is because, in the user behaviors, outdated interests may exist and the popularity of an item over time is not well represented. To address these limitations, we introduce time-aware item behaviors for addressing the recommendation of emerging preference. The time-aware item behavior for an item is a set of users who interact with this item with timestamps. The rich interaction information of users for an item may help to model its evolution. In this work, we propose a CTR prediction model TIEN based on the time-aware item behavior. In TIEN, by leveraging the interaction time intervals, information of similar users in a short time interval helps identify the emerging user interest of the target user. By using the sequential time intervals, the item's popularity over time can be captured in evolutionary item dynamics. Noisy users who interact with items accidentally are further eliminated thus learning robust personalized item dynamics. To the best of our knowledge, this is the first study to the item evolution problem for E-commerce CTR prediction. We conduct extensive experiments on five real-world CTR prediction datasets. The results show that the TIEN model consistently achieves remarkable improvements to the state-of-the-art methods.

CCS CONCEPTS

• **Information systems** → **Content ranking**; **Online shopping**; **Recommender systems**; **Content analysis and feature selection**; **Data encoding and canonicalization**.

KEYWORDS

sequential recommendation, information dissemination, recurrent neural network, attention, e-commerce recommendation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CIKM '20, October 19–23, 2020, Virtual Event, Ireland
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3411952

ACM Reference Format:
Xiang Li, Chao Wang, Bin Tong, Jiwei Tan, Xiaoyi Zeng, Tao Zhuang. 2020. Deep Time-Aware Item Evolution Network for Click-Through Rate Prediction. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411952

1 INTRODUCTION

Large E-commerce portals such as Amazon and Taobao are serving hundreds of millions of users nowadays with billions of items. For better user satisfaction and business effectiveness, Click-Through Rate (CTR) prediction has been one of the most important tasks in E-commerce. With the rapid progress of deep neural models, most of advanced CTR prediction models [8, 17, 26, 32–34] achieve great success by leveraging sequential user behaviors. For a given user, the user behavior consists of items with which the user interacts, which allows models to predict his/her potential interest according to the historical behaviors. However, users' preferences may change or emerge rapidly, for example breaking news, tweets or photos may occasionally attract users' interests. There are also cases in E-commerce that a user's interest may drift or some items may burst suddenly. For example, during the shopping festival, such as Amazon's Black Friday and Taobao's Double 11, users' preferences different from their daily routines may emerge, and items in promotions can become popular in a very short period. In such situations, using typical user behavior usually fails to predict users' emerging interests. The reason is not only outdated interests may exist in the user behaviors, but also the prediction of user's emerging preference heavily depends on the evolution of items. This problem is substantial but has rarely been studied by previous works. We name it as item evolution problem. Therefore, in this paper we focus on how to recommend emerging preference in E-commerce CTR prediction.

In order to tackle this problem, we introduce item behaviors to the CTR prediction task besides user behaviors. The item behavior for an item is a set of users who interact with this item. It can be viewed as *global* interactive behavior. In contrast, user behavior is viewed as *local* interactive behavior, since interaction information of the other users is unavailable. Item behavior was proposed in social network recommendation [5, 19, 24, 31] to find a group of users similar to the interacted users in the item behaviors. Item behaviors can be applied to our problem, because the rich interaction information of users may help to model the evolution of an item. Previous works either integrate the item behaviors through a simple aggregation, or take the item behavior as a behavioral sequence to capture the item dynamics in a temporal evolution [29, 30]. However, these methods either treat users in item behaviors equally or preserve only the order of users in an item behavior sequence, which ignores

two possible time-sensitive characteristics in recommending the emerging preferences.

First, the interaction time in item behavior is critical to predicting the potential audience of the item. Users may have interacted with an item far from the current, and now they may have changed their interests. Methods that ignore interaction time in item behaviors may diffuse users' outdated interests to their current similar users. For example, a school uniform was purchased by a girl student a few years ago who may now prefer ladies wear or urban fashion dress due to the drifting of her interests, therefore it is not appropriate to recommend a school uniform for her similar users. Instead, the school uniform should be recommended to similar users of ones who have recently purchased it. Second, the interaction time in item behavior is critical to reflect the items' attractiveness to users varying over time. The items even with same item behaviors can have different popular trends and life cycles. For example, item A and item B have the same interactive users, but the interaction time was one day ago and one year ago, respectively. This implies that item A is a more popular item than item B at this time. Thus, an item clicked or purchased far before the current time may have less impact to the current recommendation. Unfortunately, modeling the item behavior as a sequence simply ignores the interaction timestamps, which results in the same temporal dynamics for the two items. The above limitations suggest that an evolutionary item dynamics is important for the emerging preference recommendation. In this paper, we propose time-aware item behavior to address the limitations. Unlike traditional item behavior, the time-aware item behavior for an item is a set of users who interact with this item with timestamps. The time-aware item behaviors are helpful for recommending the emerging preferences since they may imply items' potential preference evolution and impact over time.

Time-awareness in the item behaviors helps to pay attention to users who interact with a specific item in a recent period. However, the emerging preference recommendation can be affected by noisy interactions. It is because noisy behaviors of a similar user to the target one make the recommendation unreliable. For example, in a very recent period, user A who prefers political news accidentally clicks a piece of entertainment news. User B, who is similar to user A, prefers political news as well. Even if time awareness takes effect, entertainment news will still be unnecessarily recommended to user B due to that both user A and B share the same interest in the recent period. Moreover, Guo et al. [6] pointed out that even the same item should be viewed differently for different users. To achieve this, attention mechanism was used to activate the most similar users in item behaviors concerning certain target users [6, 14]. However, the noisy interactions may result in unreliable personalized item dynamics if considering the most similar users only. In order to obtain robust personalized item dynamics, how to eliminate the noisy users in the time-aware item behaviors remains a challenge.

Based on the above, we propose TIEN (Deep **T**ime-Aware **I**tem **E**volution **N**etwork) to deal with the item evolution problem. In our TIEN, a Time-Interval Attention Layer is proposed to calculate the importance weight for each user in the item behaviors according to the intervals between the historical interaction and the recommendation time. This allows the model to pay attention to valuable users in the item behaviors. Moreover, a Robust Personalized Attention Layer is proposed to eliminate noisy users in the item behaviors thereby achieving robust personalized item dynamics. Last, a Time-Aware Evolution Layer is proposed to model the sequence of time intervals in an item behavior to capture the popularity of the items, such that two items with the same item behaviors can be featured in different popularity. Therefore, more accurate recommendation of emerging preferences is achieved by fully leveraging the global item behaviors, that is modelling both personalized item dynamics and popularity varied over time. The contributions of the paper are summarized as:

- To the best of our knowledge, this is the first study to the item evolution problem for E-commerce CTR prediction. To tackle this problem, we introduce time-aware item behavior, an extension of traditional user behavior which has not been used in recommending emerging preference.
- We propose the following two techniques to fully utilize the rich interaction information in the time-aware item behavior. The first is to pay attention to up-to-date interaction in the item behaviors and capture evolutionary item dynamics by using sequential time intervals. The second is to eliminate noisy interaction, which helps realize robust personalized item dynamics.
- We conduct extensive experiments on five real-world CTR prediction datasets. Experimental results verify the proposed model consistently brings remarkable improvements to previous methods and achieves new state-of-the-art results in most the cases.

2 RELATED WORK

2.1 Sequential Recommendations

In recent years, there have been growing numbers of researches on personalized recommendation for music [27], news [18], videos [4], and jobs [2] based on deep neural networks. Personalized search and recommendation is typically based on sequential user behaviors. The RNN model is used in GRU4Rec [8] for inferring users' future intentions based on their historical click behaviors. Several extensions to the RNN model are proposed in [25] for enhancing the performance. In CASER [26], several historical items are regarded as an 'image', and horizontal and vertical convolutional filters are used to capture behavior patterns at different scales. However, these methods can only obtain a fixed trajectory of interest evolution for any candidate item, so they may be disturbed by interest drifting. To resolve the above problem, an attention mechanism is proposed in DIN [34] to capture relative interests to the candidate item and obtain adaptive interest representation. Inspired by Transformer [28] for machine translation, an attention-based framework is invented in ATRANK [32] for the CTR prediction. A two-layer RNN structure with an attention mechanism is proposed in DIEN [33], which uses the attention weights to control the second RNN layer to activate the most relative interests to the candidate item.

Compared to sequence-independent recommendation approaches, such as LR [16] and FM [23], methods based on sequential user behaviors improve the performance of CTR due to users' potential interest being predicted. However, as mentioned in Introduction, personalized recommendation based on the user behavior may fail to predict users' emerging interest, because users' preferences may change or emerge rapidly.

2.2 Information Dissemination

In many social network scenarios, information like news, events, pictures or some fashion elements may spread in a predictable pattern, in which future evolution and potential impact of an item can be predicted with an early-stage observation on it. Item behavior was proposed in social network recommendation [3, 19, 24] to predict users who will be affected in the future, which essentially aims at finding a group of users similar to the interacted users in item behaviors. Early studies [9, 10] use matrix factorization to predict who will be influenced in the future. For further mining the sequential pattern in item behaviors, Du et al. [5] uses RNN model to capture the hidden dynamics in the temporal evolution of the item's affected sequence. By extending this idea, Typo-LSTM [29] uses RNN model to predict the next affected user. Recently, DIB is proposed in [6] to capture personalized item dynamics concerning certain target users through the attention mechanism, so that even the same item may be viewed distinctively for different users. Also, Liu et al. [14] activates users in item behaviors most similar to target users for learning a personalized item representation.

However, previous RNN-based methods ignore timestamps and preserve the order of items only. In other words, these methods implicitly assume that adjacent users in the sequence have the same time intervals. Moreover, previous works, which use the most similar users to learn personalized item representations, may unnecessarily have noisy interactions involved in the item behaviors, which results in an unreliable personalized item dynamics.

3 PRELIMINARY

In a recommender system, the user-item interaction is typically formulated as a matrix $\mathcal{Y} = \{y_{ui}\}_{M\times N}$, where M and N denote the numbers of users and items, respectively. The interaction y is either implicit feedback [1], e.g., click, or explicit user rating [13]. In this work, we focus on the CTR prediction, which implies that the matrix $\mathcal{Y}$ consists of 0 and 1. Specifically, $y_{ui} = 1$ means that u has clicked i, otherwise $y_{ui} = 0$. Moreover, each interaction is associated with a timestamp t that records the time of interaction. Therefore, the data in the recommender system are denoted by a set of quadruplets $\mathcal{T} = \{(u, i, t, y)\}$, each of which includes the user $u \in \mathcal{U}$ interacts with an item $i \in \mathcal{I}$ at a recommendation time t. As to a target triple (u, i, t) for which an interaction probability will be predicted, two historical interaction behaviors can be created for the target user u and candidate item i at the given time t as follows.

Definition 1. (User Behavior): Given a user u, the user behavior is a set of items that the user u interacts in chronological order, as $\mathcal{I}_u^t = \{i \mid (u, i, t') \in \mathcal{T}, t' < t\} = \{i_1, i_2, \cdots\}$.

Definition 2. (Item Behavior): Symmetrically, given an item i, the item behavior is a sequence of users who interacts with the candidate item i, as $\mathcal{U}_i^t = \{u \mid (u, i, t') \in \mathcal{T}, t' < t\} = \{u_1, u_2, \cdots\}$.

In this work, the final CTR prediction aims to estimate the probability of interaction $\hat{y}$ between the target user $u \in \mathcal{U}$ and the candidate item $i \in \mathcal{I}$, with the consideration of both the user's interaction behavior $\mathcal{I}_u$ and the item's interaction behavior $\mathcal{U}_i$, as:

$$\hat{y} = \Pr\left(y \mid u, i, \mathcal{I}_u^t, \mathcal{U}_i^t, t\right) = \mathcal{F}\left(u, i, \mathcal{I}_u^t, \mathcal{U}_i^t, t; \Theta\right) \tag{1}$$

The notations and the corresponding descriptions have been summarized in Table 1.

Notation	Description
$\mathcal{U}$,$\mathcal{I}$	User and Item set.
u, i, t	The target user and the candidate item at the recommendation time.
$\mathcal{I}_u^t$,$\mathcal{U}_i^t$	Corresponding User behavior and Item behavior of u and i at time t.
$y, \hat{y}$	The indicator and the predicted probability of the user-item interaction.
$\boldsymbol{e}_u, \boldsymbol{e}_i$	Embeddings of a user u and an item i.
$\mathcal{S}_u^t, \mathcal{S}_i^t$	Embeddings of the user behavior $\mathcal{I}_u^t$ and item behavior $\mathcal{U}_i^t$.
$\boldsymbol{e}_{tiv}$	Embeddings of the time interval for each interaction in item behaviors.
$\mathcal{S}_{tiv}^t$	Embeddings of the time interval sequence of item behaviors.
Θ	The parameters of the function $\mathcal{F}()$.

Table 1: Notations and descriptions.

4 PROPOSED MODEL

In this section, we present our model TIEN as shown in Figure 1. TIEN leverages both user behaviors and item behaviors. The item behavior for an item is a set of users who interact with this item. Time-Interval Attention Layer is proposed to calculate the importance weight for each user in the item behaviors according to the intervals between the historical interaction and the recommendation time. This allows the model to pay attention to valuable users in the item behaviors. Then, a Robust Personalized Attention Layer is proposed to eliminate noisy users in the item behaviors thereby achieving the robust personalized item dynamics. Finally, a Time-Aware Evolution Layer is proposed to model the sequence of time intervals in an item behavior to capture popularity of the items, such that two items with the same item behaviors can be featured different popularity. Besides the above three components, the Embedding Layer and Downstream Application Network of our model are similar to related CTR prediction models.

4.1 Embedding Layer

To better encode user's and item's features, the embedding layer is applied upon the sparse features to obtain low-dimensional vectors. In our proposed TIEN, there are two kinds of features: user/item feature and user/item sequence feature. The user/item feature is composed of a certain number of fields. For example, an item feature is represented by multiple fields, such as item ID, shop ID and category ID; a user feature represented by user ID, gender, age and so on. Each field is encoded by a one-hot vector. For example, the gender field of user feature is encoded as $[0, 1]$. The one-hot vectors of all the fields in the user and item features are concatenated to obtain multi-hot vectors $\boldsymbol{x}_u$ and $\boldsymbol{x}_i$, respectively. The user/item sequence feature can be represented by collecting user/item features ($\boldsymbol{x}_u$ or $\boldsymbol{x}_i$) into an ordered set.

In the embedding layer, each field of feature is corresponding to one embedding matrix. For example, the embedding matrix of item ID can be represented by $\mathbf{W}_{emb}^{item_id} \in \mathbf{R}^{d_{item_id} \times v_{item_id}}$, where d_{item_id} is the dimension and v_{item_id} is the vocabulary size. We perform embedding lookups to transform the multi-hot vectors $\boldsymbol{x}_u$ and $\boldsymbol{x}_i$ into low-dimensional embeddings $\boldsymbol{e}_u$ and $\boldsymbol{e}_i$, according to:

$$\begin{aligned} \boldsymbol{e}_u &= [\mathbf{W}_{emb_u}^1 \boldsymbol{x}_u^1, \ldots, \mathbf{W}_{emb_u}^F \boldsymbol{x}_u^F], \mathbf{W}_{emb_u}^f \in \mathbf{R}^{d_f \times v_f} \\ \boldsymbol{e}_i &= [\mathbf{W}_{emb_i}^1 \boldsymbol{x}_i^1, \ldots, \mathbf{W}_{emb_i}^F \boldsymbol{x}_i^F], \mathbf{W}_{emb_i}^f \in \mathbf{R}^{d_f \times v_f} \end{aligned} \tag{2}$$

where F is the number of user/item feature, and $\boldsymbol{x}_u^f / \boldsymbol{x}_i^f$ is the f^{th} feature. Then the user behavior embeddings is formed with item embeddings in the sequence, as $\mathcal{S}_u^t = \left\{\boldsymbol{e}_{i_1}, \cdots, \boldsymbol{e}_{i_{|\mathcal{S}_u|}}\right\}$. Similarly, the item behavior embeddings is formed with user embeddings,

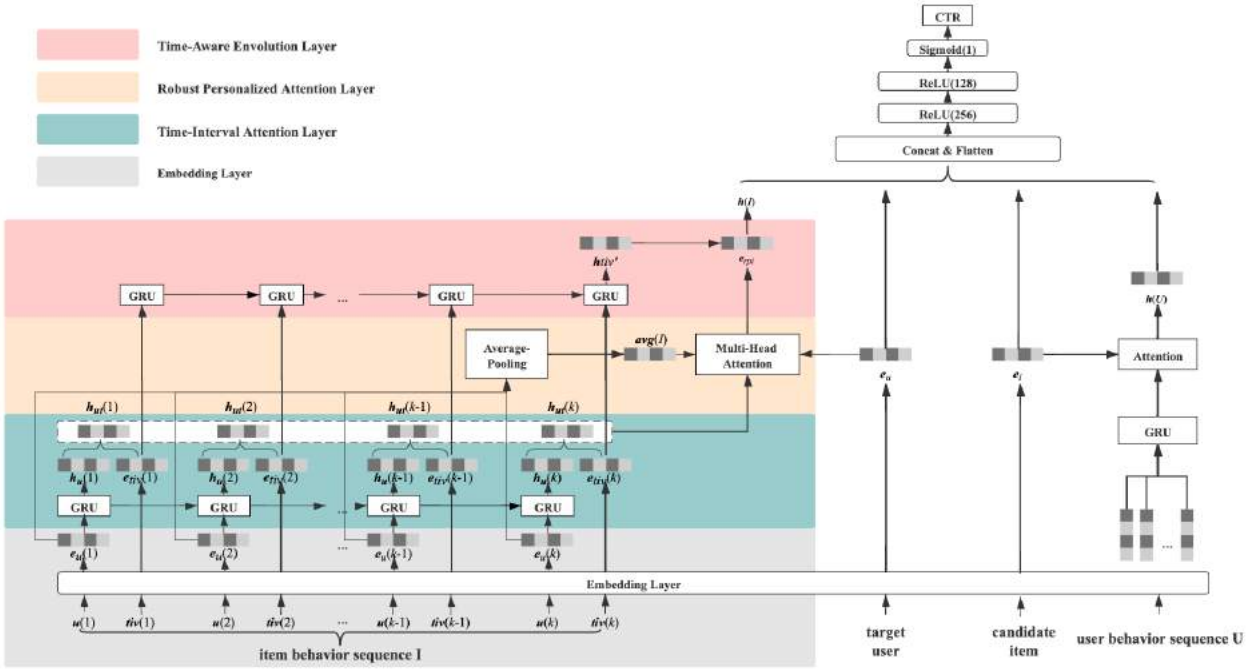


Figure 1: The architecture of TIEN that predicts the probability of a user clicks an item. Besides the target user and candidate item, TIEN uses the information from both user and item behaviors. Embedding Layer transforms user feature $u[k]$ and time interval feature $tiv[k]$ to embedding vectors $\boldsymbol{e}_u[k]$ and $\boldsymbol{e}_{tiv}[k]$, respectively. The Time-Interval Attention Layer leverages the time intervals between the historical interaction and the recommendation time to strength information of certain users in item behaviors. The Robust Personalized Attention Layer uses both the target user and the representative information shared by all users in the item behaviors to build the query of the attention mechanism, thereby eliminating noisy users in the item behaviors. The Time-Aware Evolution Layer models the sequence of interaction time intervals for each item to capture the evolutionary item dynamics. Finally, the evolutionary item dynamics $\boldsymbol{h}(I)$ and embedding vectors of remaining features including user embedding $\boldsymbol{e}_u$, item embedding $\boldsymbol{e}_i$ and user behavior embeddings $\boldsymbol{h}(U)$ are concatenated together and fed into a point-wise fully connected neural network for the CTR prediction.

as $\mathcal{S}_i^t = \left\{\boldsymbol{e}_{u_1}, \cdots, \boldsymbol{e}_{u_{|\mathcal{S}_i|}}\right\}$, where $|\mathcal{S}_u|$ and $|\mathcal{S}_i|$ are the sequence lengths of user and item behaviors, respectively.

4.2 Time-Interval Attention Layer

Item behaviors can be applied to recommend emerging preference in E-commerce CTR prediction, because the rich interaction information of users may help to model the evolution of an item. Previous works either integrate the item behaviors through a simple aggregation, or take the item behaviors as a behavioral sequence to capture the item dynamics in a temporal evolution [29, 30]. However, these methods either treat users in item behaviors equally or preserve only the order of users in an item behavior sequence.

Due to the above reason, the Time-Interval Attention Layer is proposed to calculate the importance weight of each user in the item behaviors using the time intervals between the historical interaction and the recommendation time. Given an item behavior $\mathcal{U}_i^t$ of item i at the timestamp t, each interactive user $u_k \in \mathcal{U}_i^t$ corresponds to a time interval tiv_k, which can be calculated by $tiv_k = |t - t_k|$, where t_k is the interaction timestamp between u_k and i. Since the elapsed time interval tiv follows an exponential distribution, the time intervals are mapped in the ranges $[0, 1), [1, 2), [2, 4), \cdots, [2^j, 2^{j+1})$ to categorical feature $\boldsymbol{x}_{tiv}$ of $0, 1, 2, \cdots, j+1$. We then perform embedding lookups to obtain the time interval embedding $\boldsymbol{e}_{tiv}$ to capture impacts of different time intervals, according to:

$$\boldsymbol{e}_{tiv} = [\mathbf{W}_{emb_{tiv}} \boldsymbol{x}_{tiv}], \mathbf{W}_{emb_{tiv}} \in \mathbf{R}^{d_{tiv} \times v_{tiv}} \tag{3}$$

where d_{tiv} is the dimension and v_{tiv} is the vocabulary size. Then the time interval sequence embedding is formed with each time interval embedding in the item behaviors, as $\mathcal{S}_{tiv}^t = \left\{\boldsymbol{e}_{tiv_1}, \cdots, \boldsymbol{e}_{tiv_{|\mathcal{S}_i|}}\right\}$, where$|\mathcal{S}_i|$ is the sequence length of item behaviors.

In the Time-Interval Attention Layer, we first use GRU to model the dependencies of item behavior embeddings and obtain the behavior hidden state $\boldsymbol{h}_{u_k}$ at each step k. Then, to calculate the importance weight for each user in the item behaviors thus strengthening information of certain users, we build an attention *Query* of the attention mechanism [28] by achieving the time-interval attentive user state. It is produced by the element-wise summation of behavior hidden state and item-interval embedding, according to $\boldsymbol{h}_{ut_k} = \boldsymbol{h}_{u_k} + \boldsymbol{e}_{tiv_k}$.

4.3 Robust Personalized Attention Layer

Time-awareness in the item behaviors helps to pay attention to users who interact with a specific item in a recent period. However, the emerging preference recommendation may be affected by noisy interactions. Moreover, to achieve the personalized item dynamics, attention mechanism was used to activate the most similar users in item behaviors concerning certain target users [6, 14]. However, the noisy interactions may result in an unreliable personalized item dynamics if considering the most similar users only.

Due to the above reason, the Robust Personalized Attention Layer is proposed for eliminating the noisy users in the time-aware item behaviors, thereby improving the robustness of personalized item dynamics. In the Robust Personalized Attention Layer, we represent the user embeddings in an item behavior using average pooling. The average of the user embeddings encodes the most representative information shared by all users in the item behaviors. We use both the target user and the averaged item behavior embeddings to build an attention *Query* of the attention mechanism, according to:

$$\boldsymbol{e}_{u_{smooth}} = \boldsymbol{e}_u + \sum_{k=1}^{K} \boldsymbol{e}_{u_k} \tag{4}$$

where $\boldsymbol{e}_u$ is the embedding of target user, and $u_k \in \mathcal{U}_i^t$. Then, an attentive pooling method is used for generating robust personalized item dynamics according to the target user. It should be noted that Eq. 4 smooths the representation of the target user in an intuitive way, which eliminates the side-effect from the target user. The attention query can be regarded as a trade-off between personalization and noisy elimination.

Inspired by previous works [28, 32], we apply scaled dot-product attention as the attentive pooling method, according to:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}})\mathbf{V} \tag{5}$$

where $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ represent *Query*, *Key* and *Value*, respectively and d_k is the dimension of $\mathbf{K}$. Here, *Query* is the smooth embeddings of the target user $\boldsymbol{e}_{u_{smooth}}$, *Key* is the time-interval attentive user state $\boldsymbol{h}_{ut_k}$, and *Value* is the behavior hidden state $\boldsymbol{h}_{u_k}$. Eq. 5 calculates the weight of each user in the item behaviors by considering both the time interval embedding and the smooth target user embedding, and applies the weight to the hidden state of the item behaviors, thereby achieving robust personalized item dynamics through the weighted summation. Moreover, in order to enable the model to jointly attend to information from different representation subspaces at different positions, we adopts multi-head attention, following [28]. The multi-head attention is defined as follows:

$$\begin{aligned} MultiHead(\mathbf{Q}, \mathbf{K}, \mathbf{V}) =& Concat(head_1, head_2, \cdots, head_h)\mathbf{V}^O \\ head_i =& Attention(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \end{aligned} \tag{6}$$

where $\mathbf{V}^O$, $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$ and $\mathbf{W}_i^V$ are parameters and h is the number of heads. Then, the multi-head attention network employs a feed-forward network for further strengthening the performance. Finally, the robust personalized item dynamics $\boldsymbol{e}_{rpi}$ can be achieved by:

$$\boldsymbol{e}_{rpi} = ReLU((MultiHead(\mathbf{Q}, \mathbf{K}, \mathbf{V})\mathbf{W_1} + \boldsymbol{b_1})\mathbf{W_2} + \boldsymbol{b_2} \tag{7}$$

4.4 Time-Aware Evolution Layer

In many scenarios, each item has a different fashion trend and life cycle, which is also the most important characteristic of an item in E-commerce. However, previous methods [6, 14, 29, 30] that model the item behavior as a sequence simply ignore the interaction timestamps of item behaviors, which results in the same temporal dynamics for the two items. Therefore, it is beneficial to capture the evolutionary item dynamics for the emerging preference recommendation.

Due to the above reason, the Time-Aware Evolution Layer is proposed to model the sequence of time intervals in an item behavior to capture the popularity of the items, such that two items with the same item behaviors can be featured in different popularity. We use GRU to learn the relevant time-interval embedding forward step by step to achieve most representative information of the whole time-interval sequence. Each GRU unit takes the corresponding time-interval embedding $\boldsymbol{e}_{tiv_k}$ at each time step and the hidden state from the last time step. We use the output final state as the evolutionary time state of item behaviors, as:

$$\boldsymbol{h}_{tiv}{}' = GRU(\boldsymbol{e}_{tiv_1}, \boldsymbol{e}_{tiv_2}, \cdots, \boldsymbol{e}_{tiv_{|S_i|}}) \tag{8}$$

Finally, the embedding of evolutionary item dynamics $\boldsymbol{h}_{item}$ is obtained by using the combination of evolutionary time state $\boldsymbol{h}_{tiv}{}'$ and robust personalized item dynamics $\boldsymbol{e}_{rpi}$. Note that the combination function can be in a flexible form, such as element-wise summation, multiplication or nonlinear projection.

4.5 Downstream Application Network

The learned evolutionary item dynamics can be applied to various kinds of neural networks according to the downstream task requirement. In this paper, we perform the CTR prediction task and set the downstream application network to be a point-wise fully connected neural network. For this CTR prediction model, besides the information from a target user u and a candidate item i, we use both the user behaviors $\mathcal{I}_u$ and item behaviors $\mathcal{U}_i$ as inputs. A probability is produced that user u clicks item i. The predicted probability of interaction between u and i is calculated as:

$$\hat{y} = \mathcal{F}\left(u, i, \mathcal{I}_u^t, \mathcal{U}_i^t, t; \Theta\right) \tag{9}$$

where $\mathcal{F}$ is a ranking function implemented as a multi-layer deep network with three layers, whose widths are 256, 128 and 1, respectively. The first and second layers use ReLU as activation function while the third layer uses Sigmoid function.

We adopt the widely-used cross entropy [17, 32, 33] as the loss function. The ultimate loss function is formulated as:

$$\arg\min_{\Theta} = -\sum_{k=1}^{N} \left[y_k \log \hat{y}_k + (1 - y_k) \log(1 - \hat{y}_k)\right] \tag{10}$$

where $\boldsymbol{\Theta}$ is the parameters and N is the size of training dataset.

5 EXPERIMENTS

To comprehensively evaluate the proposed model TIEN, we conduct experiments to answer the following research questions:

RQ1 How does TIEN perform, compared to the state-of-the-art models for the CTR prediction task?

RQ2 Does the essential components in TIEN necessarily contribute to the improvement of the performance? Does the incorporation of proposed evolutionary item dynamics contribute to the performance of the state-of-the-art CTR prediction models based on the sequential user behaviors?
RQ3 How do different truncation length K of item behaviors in TIEN and other baselines affect the performances? Is long-term item behaviors well handled by all baselines?
RQ4 How do hyper-parameters in TIEN affect the performance?

5.1 Experimental Setups

5.1.1 Datasets. We evaluate the state-of-the-art methods over popular real-world datasets of the Amazon product data[1] [7, 15].
Amazon Dataset. The dataset contains the temporal users' purchasing behaviors ranging from the May 1996 to July 2014, with 24 categories of different sizes. We choose five categories including Clothing, Beauty, Grocery, Phones and Sports to conduct our experiments. Each category is a set of tuples, and each tuple consists of the information of user, item, rating and timestamp. Since we focus on the CTR prediction tasks, we treat all existing ratings as positive interactions with a label of 1. Following other works [32–34], we filter out users and items whose interactions are less than five. The statistics of each dataset is shown in Table 2.

Datasets	# Users	# Items	# Category	# Interactions	Density	# samples
Clothing	39388	23034	720	278677	0.0003	478580
Beauty	22364	12102	221	198502	0.0007	352278
Grocery	14682	8714	129	151254	0.0011	273146
Phones	27880	10430	51	194439	0.0006	333120
Sports	35599	18358	1073	296337	0.0004	521478

Table 2: Statistics of the datasets.

Dataset Preprocessing. We sort the interactions of each user and items by the timestamp to build the user behaviors and item behaviors. Assuming there are user behaviors $\{i_1, \ldots, i_{j-1}, i_j, \ldots, i_{|\mathcal{S}_u|}\}$ of the user u, where $|\mathcal{S}_u|$ is the sequence length. Let u be the target user and i_j be the candidate item, which is the j^{th} behavior of u. For the item i_j, there exist item behaviors $\{u_1, \ldots, u_{t-1}, u_t, \ldots, u_{|\mathcal{S}_i|}\}$, where $|\mathcal{S}_i|$ is the sequence length. Let u_t be the target user u who interacted with item i_j at timestamp t. In the CTR prediction task, we predict the user response probability between u and i_j according to the first $j-1$ user behaviors, i.e., $\{i_1, \ldots, i_{j-1}\}$, and the item behaviors before the timestamp t, i.e., $\{u_1, \ldots, u_{t-1}\}$. Note that, in order to build the negative samples, 50% candidate items at the prediction time in each dataset have been replaced with another item from the non-clicked item set for each user. This replacement is often used in related works [32–34]. The sample statistics of each dataset is shown in Table 2.
Training & Test Splitting. Following works [17, 21], we split the entire dataset into the training and test parts according to the timestamp of the prediction behavior, effectively avoiding feature leakage. We set a cut time within the time range covered by the full dataset. If the prediction behavior of a sequence took place before the cut time, the sequence is put into the training set. Otherwise it would be put into the test set. In this way, the training set is about 85% of the whole dataset and the left 15% of data is the test set.

[1] http://jmcauley.ucsd.edu/data/amazon/

5.1.2 Evaluation Metrics. We adopt three widely used metrics for the CTR prediction task [17, 20, 22, 32–34], which show a model's performance from different perspectives. The first metric is area under ROC curve (**AUC**) which assesses the pairwise ranking performance of the classification results between the clicked and non-clicked samples. AUC is formulated as:

$$\text{AUC} = \frac{1}{\left|U^{Test}\right|} \sum_{u \in U^{Test}} \frac{1}{|I_u^+|\,|I_u^-|} \sum_{i \in I_u^+} \sum_{j \in I_u^-} \delta(\hat{p_{u,i}} > \hat{p_{u,j}}) \quad (11)$$

where $\hat{p_{u,i}}$ is the predicted probability that a user $u \in U^{Test}$ may click on the item i in the test set and $\delta(\cdot)$ is the indicator function. I_u^+ is the item set clicked by the user, and I_u^- is the item set that the user does not click. The second metric is **Logloss** (cross entropy), which is to measure the overall likelihood of the whole test data. Different from AUC which measures a pairwise relation, Logloss measures predicted probability that can be used to estimate the benefit of a ranking strategy. For example, in the Cost-Per-Click (CPC) advertising system, advertisements are ranked by the effective Cost Per Mille (eCPM), which is the product of the bid price and CTR. The last metric is **F1-score** which is the harmonic mean of precision and recall. It reaches its best value with 1, which means perfect precision and recall, and with 0 at the worst case.

5.1.3 Baselines. To illustrate the effectiveness of TIEN, we compare it with the state-of-the-art methods, which can be grouped into non-sequential and sequential categories.

For non-sequential methods, three models are compared. For these three models, we label them as Group 1.

SVD++ [12] is a hybrid method of latent factor model and neighbor based model. It regards all the sequential behaviors as a whole and ignores the temporal dynamics.
YoutubeNet [4] is proposed to recommend videos in YouTube, which gets user representations by simply averaging the item embeddings in the user behavior sequence.
PNN [20] uses a product layer to capture interactive patterns between inter-field categories on the basis of **YoutubeNet**.

For sequential methods, five models that use user behaviors only are compared. For these five models, we label them as Group 2.

GRU4Rec [8] uses RNN and it is the first work using recurrent cell to model sequential user behaviors.
CASER [26] is a CNN-based model, using horizontal and vertical convolutional filters to capture behavior patterns at different scales.
ATRANK [32] is an attention-based framework modeling the dependencies between user behaviors.
DIEN [33] is a two-layer RNN structure with an attention mechanism. It uses the calculated attention values to control the second RNN layer to activate the most relative interests to the candidate item.
UB-GRUA is a sub-model of **TIEN**, which uses GRU to model the dependencies of user behaviors and uses multi-head attention to activate the most interest hidden state with respect to the candidate item.

For sequential methods, another three models are also compared, which use both user and item behaviors to capture the temporal dependencies in user's states and item's states. Note that these

Group	Method	Clothing			Beauty			Grocery			Phones			Sports		
		AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score
1	SVD++	0.7332	0.6271	0.6964	0.7410	0.6343	0.7112	0.6379	0.6992	0.6707	0.7014	0.6664	0.6758	0.7842	0.5988	0.7445
	YoutubeNet	0.7363	0.6061	0.6763	0.7081	0.6773	0.6472	0.5805	0.9637	0.5761	0.6367	0.7110	0.5881	0.7717	0.5758	0.7133
	PNN	0.7435	0.5972	0.6656	0.7838	0.6598	0.7140	0.6364	0.8058	0.6071	0.7094	0.7765	0.6495	0.7967	0.5514	0.7346
2	GRU4Rec	0.7554	0.5885	0.6764	0.7973	0.6418	0.7179	0.6134	0.9850	0.6007	0.6980	0.8135	0.6342	0.8021	0.5492	0.7329
	ATRANK	0.7521	0.5883	0.6858	0.7834	0.6744	0.7129	0.6562	0.8152	0.5732	0.6824	0.7757	0.6222	0.8026	0.5451	0.7301
	CASER	0.7504	0.5935	0.6803	0.7930	0.7068	0.7170	0.6183	0.9499	0.5031	0.7222	0.7638	0.6507	0.7989	0.5489	0.7347
	DIEN	0.7564	0.5876	0.6792	0.7933	0.6966	0.7047	0.6402	0.8579	0.6027	0.7005	0.8276	0.6420	0.8054	0.5427	0.7318
	UB-GRUA	0.7590	0.5828	0.6791	0.7975	0.6498	0.7154	0.6442	0.8078	0.5413	0.6982	0.8248	0.6378	0.8039	0.5466	0.7328
3	Topo-LSTM	0.7682	0.5748	0.6740	0.8458	0.5111	0.7657	0.7927	0.5299	0.7313	0.8089	0.5287	0.7392	0.8156	0.5267	0.7523
	DIB	0.7577	0.5820	0.6816	0.8464	0.4848	0.7680	0.7717	0.5751	0.7114	0.7949	0.6797	0.7317	0.8120	0.5316	0.7474
	IB-GRUA	0.7657	0.5763	0.6817	0.8478	0.4842	0.7703	0.7762	0.5768	0.7036	0.8043	0.6360	0.7401	0.8139	0.5275	0.7492
	TIEN	**0.7962***	**0.5476***	**0.6980***	**0.8701***	**0.4479***	**0.7840***	**0.8252***	**0.5019***	**0.7524***	**0.8389***	**0.4949***	**0.7427***	**0.8266***	**0.5101***	**0.7543***

Table 3: Performance comparisons of TIEN with baseline models on AUC, Logloss and F1-score metrics. The bold value marks the best one in one column, while the underlined value corresponds to the best one among all baselines in each group. Here, * indicates statistical significance improvement compared to the best baseline measured by t-test at p-value of 0.05.

methods model user behaviors in the same way as TIEN. For these three models, we label them as Group 3.

Topo-LSTM [29] considers the item behaviors as a sequence, and adopts LSTM to capture the item dynamics.

DIB [6] proposes a dynamic item block, which activates users in item behaviors most similar to target users for learning personalized item dynamics.

IB-GRUA integrates the technique in Topo-LSTM and DIB, which models the dependencies of sequential item behaviors and further learns personalized item dynamics.

TIEN is our proposed model described in Sec. 4.

5.1.4 Reproducibility. All datasets and code of our approach can be found here[2]. We implement TIEN with Tensorflow 1.4. The code of ATRANK[3], CASER[4], DIEN[5], DIB[6], and Typo-LSTM[7] are from the authors. For GRU4Rec[8] and SVD++[9], we also use code from public resources. The code of YoutubeNet and PNN are from DIEN[10]. All neural network models were trained using NVIDIA Tesla P4 GPU with 8 GB memory.

Parameter Settings. For our TIEN and baseline methods, all hyper-parameters are tuned on the validation dataset, where early stopping strategy is applied such that the training is terminated if validation performance does not improve over 10 iterations. Batch size is fixed on 128 for all methods. The dimensions of the item ID, user ID and category ID are set to 32, which refers to the settings in the work [33]. Adam [11] is used as the optimizer with the learning rate of 0.001. Since we mainly discuss sequential models based on user and item behaviors in this paper, the hidden units of the prediction layer is fixed for all models with [256, 128]. For all methods, the truncation length of user behaviors is 100. For dual behavior models, the truncation length K of item behaviors is chosen from $\{5, 10, 20, 30, 40, 50\}$. We report the results of each method under its optimal hyper-parameters settings.

Significance Test. The experiments are repeated 10 times by changing the random seed for TIEN and the baseline models. The two-tailed pairwise t-test is performed to detect significant differences between TIEN and the baseline models.

5.2 Experimental Results: RQ1

All experiments are repeated 10 times and averaged metrics are reported in Table 3. The influence of random initialization to results is less than 0.0002. From Table 3, we can see that TIEN improves the performance significantly against all the baselines and achieves the state-of-the-art performance.

In most cases, the models in Group 1, which do not consider the user behaviors as a sequence, have degraded performances compared with models in Group 2 and 3. Moreover, in all cases, models in Group 3 outperform models in Group 2, which verifies the effectiveness of both user- and item-side information.

Compared with models in Group 3, i.e., Topo-LSTM, DIB and IB-GRUA, which use both user behaviors and item behaviors, our TIEN shows its superior performance. The possible reasons are shown below: Topo-LSTM can capture the item dynamics in a temporal evolution, but the item dynamics is fixed for all users, lacking personalization; DIB activates the most similar users in item behaviors concerning certain target users for learning personalized item representations, but it ignores the dependency between adjacent item behaviors; Though IBGRUA allows the integration of sequential and personalized characteristics of item behaviors, it does not take the crucial time-sensitive characteristics of item behaviors into consideration and thus cannot capture the evolutionary item dynamics. Moreover, it cannot eliminate the noisy users in the item behaviors to achieve robust personalized item dynamics.

Finally, the improvement of TIEN over the best baseline model are 2.80%, 2.23%, 3.25%, 3.00% and 1.10% AUC scores. As reported in [34], compared with YoutubeNet, DIN improves AUC scores by 1.13% and the improvement of online CTR is 10.0%, which means a small improvement in offline AUC is likely to lead to a significant increase in online CTR. In our practice, for cutting-edge CTR prediction models, even one point of AUC improvement is substantial and achieves significant online promotion.

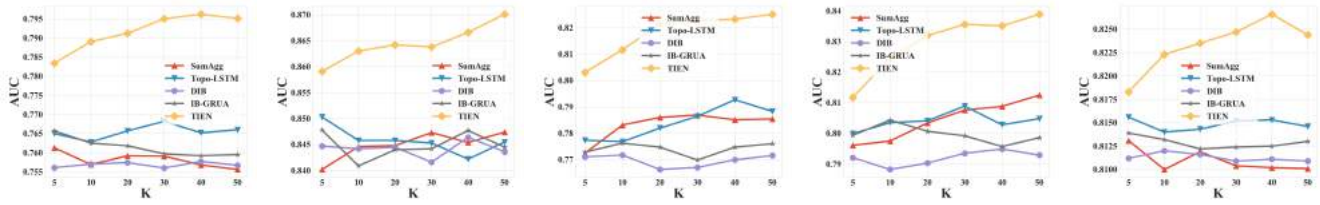
[2]https://github.com/itemevolutionnet/ItemEvolutionNet
[3]https://github.com/jinze1994/ATRank
[4]https://github.com/graytowne/caser_pytorch
[5]https://github.com/mouna99/dien
[6]https://github.com/ououuououuou/DIB-PEB-Sequential-RS
[7]https://github.com/vwz/topolstm/
[8]https://github.com/Songweiping/GRU4Rec_TensorFlow
[9]https://github.com/WindQAQ/tf-recsys
[10]https://github.com/mouna99/dien

Method	Clothes			Beauty			Grocery			Phones			Sports		
	AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score
BaseModel	0.7590	0.5828	0.6791	0.7975	0.6498	0.7154	0.6442	0.8078	0.5413	0.6982	0.8248	0.6378	0.8039	0.5466	0.7328
BaseModel+SumAGG	0.7613↑	0.5794	0.6787	0.8474↑	0.4932	0.7687	0.7869↑	0.5787	0.7123	0.8125↑	0.6100	0.7416	0.8131↑	0.5274	0.7469
BaseModel+TAL	0.7824↑	0.5613	0.6801	0.8499↑	0.4792	0.7557	0.8026↑	0.5225	0.7294	0.8303↑	0.5070	0.7318	0.8203↑	0.5189	0.7490
BaseModel+TAL+RAL	0.7864↑	0.5589	0.6798	0.8598↑	0.4656	0.7790	0.8040↑	0.5241	0.7206	0.8301↓	0.5083	0.7349	0.8200↓	0.5215	0.7420
TIEN (TAL+RAL+TEL)	**0.7962**	**0.5476**	**0.6980**	**0.8701**	**0.4479**	**0.7840**	**0.8252**	**0.5019**	**0.7524**	**0.8389**	**0.4949**	**0.7427**	**0.8266**	**0.5101**	**0.7543**

Table 4: Ablation study of TIEN on AUC, Logloss and F1-score metrics on Amazon datasets. ↑ and ↓ indicates whether adding key components of TIEN boost or degrade the AUC performance.

Model	Clothing			Beauty			Grocery			Phones			Sports		
	AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score
GRU4Rec	0.7554	0.5885	0.6764	0.7973	0.6418	0.7179	0.6134	0.9850	0.6007	0.6980	0.8135	0.6342	0.8021	0.5492	0.7329
GRU4Rec+TIEN	0.7966	0.5528	0.6811	0.8668	0.4537	0.7686	0.8261	0.5040	0.7357	0.8378	0.4986	0.7322	0.8266	0.5113	0.7562
ATRANK	0.7521	0.5883	0.6858	0.7834	0.6744	0.7129	0.6562	0.8152	0.5732	0.6824	0.7757	0.6222	0.8026	0.5451	0.7301
ATRANK +TIEN	0.7902	0.5619	0.6668	0.8675	0.4498	0.7737	0.8246	0.5037	0.7437	0.8307	0.5110	0.7222	0.8234	0.5154	0.7406
CASER	0.7504	0.5935	0.6803	0.7930	0.7068	0.7170	0.6183	0.9499	0.5031	0.7222	0.7638	0.6507	0.7989	0.5489	0.7347
CASER+TIEN	0.7920	0.5486	0.7089	0.8638	0.4581	0.7653	0.8235	0.5059	0.7445	0.8307	0.5110	0.7222	0.8206	0.5217	0.7366
DIEN	0.7564	0.5876	0.6792	0.7933	0.6966	0.7047	0.6402	0.8579	0.6027	0.7005	0.8276	0.6420	0.8054	0.5427	0.7318
DIEN +TIEN	0.7956	0.5516	0.6851	0.8675	0.4523	0.7846	0.8273	0.5023	0.7400	0.8379	0.4974	0.7399	0.8255	0.5115	0.7526

Table 5: Practicability Study of TIEN.



(a) Amazon Clothing (b) Amazon Beauty (c) Amazon Grocery (d) Amazon Phones (e) Amazon Sports

Figure 2: Performance comparisons of TIEN with baseline models on AUC with respect to different item behavior length K.

5.3 Ablation Study and Practicability Study: RQ2

In this section, we design ablation experiments to study how each component in TIEN contributes to the final performance.

BaseModel: A sub-model of TIEN that uses user behaviors only.

BaseModel+SumAGG: A simple aggregation model which uses sum-pooling to integrate the item behavior embeddings.

BaseModel+TAL: A sub-model of TIEN with the Time-Interval Attention Layer (TAL).

BaseModel+TAL+RAL: A sub-model of TIEN with both TAL and Robust Personalized Attention Layer (RAL).

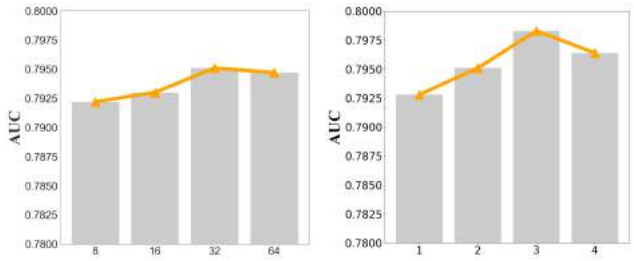
TIEN: The entire proposed model that includes TAL, RAL and Time-Aware Evolution Layer (TEL).

As shown in Table 4, removing any component in TIEN leads to a drop in performance. Moreover, some further observations can be made. First, by leveraging item behaviors, SumAGG alleviates the emerging preference issues and outperforms the baseline model. Second, TAL outperforms SumAGG about 2.11%, 0.25%, 1.57%, 1.78% and 0.72% AUC scores. It shows that leveraging the time intervals between the historical interaction and the recommendation time to strengthen information of certain users in an item behavior can lead to better performance. Third, in many cases, RAL further improves performance, which implies that eliminating the side effects of target users by smoothing the target user's representation can effectively improve the robustness of personalized item dynamics. Last, TEL contributes to the improvements of 0.98%, 1.03%, 1.20%, 0.88%, and 0.66% in AUC. It shows that modeling the sequence of time intervals for each item to capture the evolutionary item dynamics can effectively resolve the item evolution problem.

To verify the utility of evolutionary item dynamics proposed by TIEN, we select several models using user behaviors as base models, including GRU4Rec, ATRANK, CASER, and DIEN. As shown in Table 5, the performance of all based models are improved with the help of the generated evolutionary item dynamics, which proves the effectiveness of the evolutionary item dynamics and shows the practicability of TIEN. Thus, our TIEN can be viewed as a general framework to enhance existing user behavior based CTR prediction models.

5.4 Influence of Truncation Length of Item Behaviors: RQ4

The truncation length K of item behaviors is changed in a range of $\{5, 10, 20, 30, 40, 50\}$ for further investigating the performance of our TIEN and other dual behavior baselines. The results on five Amazon datasets are shown in Figure 2. It can be observed that TIEN improves the performance of the AUC when K increases. In



(a) dimension (b) hidden layer

Figure 3: Performance of TIEN on Amazon Clothing dataset with respect to different hyper-parameters.

most cases, the absolute improvement of the AUC scores is greater than 1%. This implies that TIEN is capable of handling long-term item behavior thereby achieving remarkable improvements. On the contrary, the performances of the baseline models decrease when K increases. This indicates that, without the time-interval attention layer and robust personalized layer, the baseline models cannot handle long-term item behaviors since noisy and useless information is unnecessarily introduced.

5.5 Parameter Sensitivity: RQ4

Impact of Embedding Size: We let the embedding size in TIEN varies in a range of $\{8, 16, 32, 64\}$. The results on Amazon Clothing datasets are shown in Figure 3(a). It can be observed that the AUC scores improve at first and decays latter. The possible reason is that small embedding size may deprive the model of enough expressiveness, and too large embedding size would make the representation vector too sparse, which leads to performance decline.

Impact of Network Depth: We let the number of fully connected layers follow the settings: 1 layer-$[256]$; 2 layers-$[256, 128]$; 3 layers-$[512, 256, 128]$; 4 layers-$[1024, 512, 256, 128]$. The results on Amazon Clothing dataset are shown in Figure 3(b). It is observed that increasing the number of fully connected layers can improve the AUC in the beginning, but the benefit diminishes when more layers are added. Adding more layers may even result in slight performance degradation, possibly due to more model parameters and increased difficulty of training deeper neural networks.

6 CONCLUSION

In this paper, we study how to recommend emerging preferences in E-commerce CTR prediction. We define it as the item evolution problem that has rarely been studied by previous works. In order to tackle this problem, we propose time-aware item behaviors for the CTR prediction task. The time-awareness helps to represent the drift of user interest and the item popularity over time. With the time-aware item behaviors, we propose TIEN (Deep **T**ime-Aware **I**tem **E**volution **N**etwork). Time-Interval Attention Layer is used to identify the emerging user interest by leveraging the interaction time intervals. This allows the model to pay attention to the information of similar users in a short time interval. By using the sequential time intervals, Time-Aware Evolution Layer captures the item's popularity over time, which is represented in evolutionary item dynamics. Robust Personalized Attention Layer helps to eliminate the noisy interaction in the item behaviors, thereby producing robust personalized item dynamics. The extensive experiments on the real-world datasets show the superiority of TIEN over the state-of-the-art methods.

REFERENCES

[1] Deepak Agarwal, Bee-Chung Chen, and Pradheep Elango. 2009. Spatio-temporal models for estimating click-through rate. In *WWW*. ACM, 21–30.
[2] Fedor Borisyuk, Liang Zhang, and Krishnaram Kenthapadi. 2017. LiJAR: A system for job application redistribution towards efficient career marketplace. In *SIGKDD*. 1397–1406.
[3] Justin Cheng, Lada Adamic, P Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. 2014. Can cascades be predicted?. In *WWW*. 925–936.
[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Recsys*. 191–198.
[5] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*. ACM, 1555–1564.
[6] Guibing Guo, Shichang Ouyang, Xiaodong He, Fajie Yuan, and Xiaohua Liu. 2019. Dynamic item block and prediction enhancing block for sequential recommendation. In *IJCAI*. AAAI Press, 1373–1379.
[7] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*. 507–517.
[8] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. *ICLR* (2016).
[9] Bo Jiang, Jiguang Liang, Ying Sha, Rui Li, Wei Liu, Hongyuan Ma, and Lihong Wang. 2016. Retweeting behavior prediction based on one-class collaborative filtering in social networks. In *SIGIR*. 977–980.
[10] Bo Jiang, Jiguang Liang, Ying Sha, and Lihong Wang. 2015. Message clustering based matrix factorization model for retweeting behavior prediction. In *CIKM*. 1843–1846.
[11] Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.
[12] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*. 426–434.
[13] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *KDD*. ACM, 447–456.
[14] Yudan Liu, Kaikai Ge, Xu Zhang, and Leyu Lin. 2019. Real-time Attention Based Look-alike Model for Recommender System. In *KDD*. ACM.
[15] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. 43–52.
[16] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: a view from the trenches. In *KDD*. 1222–1230.
[17] Yabo Ni, Dan Ou, Shichen Liu, Xiang Li, Wenwu Ou, Anxiang Zeng, and Luo Si. 2018. Perceive Your Users in Depth: Learning Universal User Representations from Multiple E-commerce Tasks. In *KDD*. 596–605.
[18] Kyo-Joong Oh, Won-Jo Lee, Chae-Gyun Lim, and Ho-Jin Choi. 2014. Personalized news recommendation using classified keywords to capture user preference. In *ICACT*. 1283–1287.
[19] Sasa Petrovic, Miles Osborne, and Victor Lavrenko. 2011. RT to Win! Predicting Message Propagation in Twitter. In *ICWSM*.
[20] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *ICDM*. IEEE, 1149–1154.
[21] Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, et al. 2019. Lifelong Sequential Modeling with Personalized Memorization for User Response Prediction. In *SIGIR*. 565–574.
[22] Kan Ren, Weinan Zhang, Yifei Rong, Haifeng Zhang, Yong Yu, and Jun Wang. 2016. User response learning for directly optimizing campaign performance in display advertising. In *CIKM*. ACM, 679–688.
[23] Steffen Rendle. 2010. Factorization machines. In *ICDM*. IEEE, 995–1000.
[24] Bongwon Suh, Lichan Hong, Peter Pirolli, and Ed H Chi. 2010. Want to be retweeted? large scale analytics on factors impacting retweet in twitter network. In *ICSC*. IEEE, 177–184.
[25] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Recsys*. 17–22.
[26] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*. 565–573.
[27] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *NIPS*. 2643–2651.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.
[29] Jia Wang, Vincent W Zheng, Zemin Liu, and Kevin Chen-Chuan Chang. 2017. Topological recurrent neural network for diffusion prediction. In *ICDM*. IEEE, 475–484.
[30] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *WSDM*. ACM, 495–503.
[31] Qingyuan Zhao, Murat A Erdogdu, Hera Y He, Anand Rajaraman, and Jure Leskovec. 2015. Seismic: A self-exciting point process model for predicting tweet popularity. In *KDD*. ACM, 1513–1522.
[32] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2018. ATRank: An attention-based user behavior modeling framework for recommendation. In *AAAI*.
[33] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *AAAI*, Vol. 33. 5941–5948.
[34] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *KDD*. ACM, 1059–1068.

移动计算领域

MobiCom 2020

OnRL: Improving Mobile Video Telephony via Online Reinforcement Learning

Huanhuan Zhang[+], Anfu Zhou[+], Jiamin Lu[+], Ruoxuan Ma[+], Yuhan Hu[+], Cong Li[+], Xinyu Zhang[◇], Huadong Ma[+], Xiaojiang Chen[†]

[+]Beijing Univ. of Posts and Telecom. {zhanghuanhuan, zhouanfu,mhd}@bupt.edu.cn
[◇] University of California San Diego xyzhang@ucsd.edu
[†] Taobao Inc. zhongsheng.cxj@taobao.com

ABSTRACT

Machine learning models, particularly reinforcement learning (RL), have demonstrated great potential in optimizing video streaming applications. However, the state-of-the-art solutions are limited to an *"offline learning"* paradigm, *i.e.*, the RL models are trained in simulators and then are operated in real networks. As a result, they inevitably suffer from the simulation-to-reality gap, showing far less satisfactory performance under real conditions compared with simulated environment. In this work, we close the gap by proposing OnRL, an online RL framework for real-time mobile video telephony. OnRL puts many individual RL agents directly into the video telephony system, which make video bitrate decisions in real-time and evolve their models over time. OnRL then aggregates these agents to form a high-level RL model that can help each individual to react to unseen network conditions. Moreover, OnRL incorporates novel mechanisms to handle the adverse impacts of inherent video traffic dynamics, and to eliminate risks of quality degradation caused by the RL model's exploration attempts. We implement OnRL on a mainstream operational video telephony system, Alibaba Taobao-live. In a month-long evaluation with 543 hours of video sessions from 151 real-world mobile users, OnRL outperforms the prior algorithms significantly, reducing video stalling rate by 14.22% while maintaining similar video quality.

CCS CONCEPTS

• **Networks** → **Transport protocols**; **Mobile networks**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

Video Telephony; Online Learning; Reinforcement Learning

ACM Reference Format:

Huanhuan Zhang, Anfu Zhou, Jiamin Lu, Ruoxuan Ma, Yuhan Hu, Cong Li, Xinyu Zhang, Huadong Ma, Xiaojiang Chen. 2020. OnRL: Improving Mobile Telephony via Online Reinforcement Learning. In *The 26th Annual International Conference on Mobile Computing and Networking (MobiCom'20), September 21-25, 2020, London, United Kingdom.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3372224.3419186

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

MobiCom '20, September 21–25, 2020, London, United Kingdom

© 2020 Association for Computing Machinery.

ACM ISBN 978-1-4503-7085-1/20/09…$15.00
https://doi.org/10.1145/3372224.3419186

1 INTRODUCTION

Real-time interactive video communication carries a dominant fraction of today's Internet traffic [5], largely due to many mainstream video telephony applications, such as Facetime, Skype, Google Hangouts, WeChat, *etc.* With the upgrade of the LTE-Advanced and 5G network infrastructures, new use cases are also rapidly emerging, such as live video/VR broadcasting [23, 31], cloud gaming [53, 58], tele-operation of surgery robots or vehicles [39, 59]. Such interactive video applications impose the toughest demand on the network in terms of bandwidth and latency. Although the telecom infrastructure strives to match the demands, it only promises best effort service, and has to rely on the application itself to adapt to the highly dynamic network conditions.

To maintain high quality of experience (QoE), traditional interactive video applications adopt rule based protocols, *e.g.*, congestion control at transport layer and video bitrate adaption on the application. However, the pre-programmed rules fall short of accommodating the highly heterogeneous modern Internet consisting of cellular/WiFi wireless links, cross-continent fiber links, intra-cloud data-center links, *etc.*, all with diverse bandwidth, latency, and buffer capacities. In recent years, data-driven approaches [25, 33, 42, 60, 69] have been explored to improve video QoE. For instance, Pensieve [42] employs reinforcement learning (RL) in video streaming systems to adapt the video bitrate, aiming to mitigate the risk of frame stalling while maximizing bandwidth utilization. Concerto [69] customizes imitation learning to harmonize the transport layer and video codec, so as to reduce stalling in mobile video telephony.

While showing potential, these solutions commonly adopt a *"learning offline, running online"* strategy. The learning models are trained in simulators or emulators, and then the trained models are directly deployed and tested in real applications. Unfortunately, such offline learning models lead to far less satisfactory performance in the real world, despite the high performance achieved in the simulator/emulator [40]. Moreover, they may exhibit even the opposite performance characteristics from those observed in simulations, as validated in [69]. The root cause lies in two interrelated factors: *(i)* It is very challenging to faithfully simulate the complicated real-world Internet dynamics [66]. Individual routers along a network path can bear various capabilities and states, *e.g.*, multiple-flow competition, packet drop strategy, load-induced quality fluctuation, let alone the complicated interaction among multiple protocol layers on end devices. *(ii)* In essence, the capacity of a data-driven algorithm is strictly bounded by its learning environment. Its experience acquired via trial-and-error in a simulator may become stale when coping with the real-world Internet.

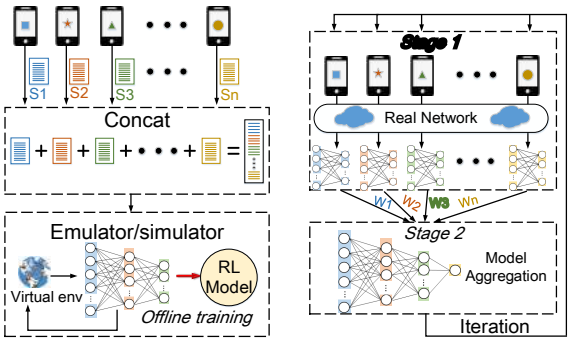


Figure 1: Learning architecture comparison: (a) offline learning (b) online learning.

In this work, we seek to close the simulation-to-reality gap [1, 20] by proposing OnRL, an *online reinforcement learning based adaptation framework for mobile video telephony*. OnRL runs and keeps training directly on an operational video telephony system (*i.e.*, Alibaba Taobao-Live [6, 7, 54]), so as to learn and respond to the real network conditions. OnRL is not merely a straightforward learning environment shift. Instead, it raises three unique design challenges.

(i) Learning from massive concurrent video telephony sessions. In conventional offline learning, one can collect and concatenate network traces from individual user, and feed them into a simulator/emulator to train an RL model [42, 69]. In this way, the RL model can explore diverse environments to enrich its experience, and converge to a universal model with swarm intelligence learned from all users. However, for online learning, a massive number of video telephony sessions run simultaneously, during which the learning algorithm needs to evolve with each session in real time. Therefore, the key challenge lies in how to transform from serial offline learning to parallel online learning while still harnessing the swarm intelligence. To meet the challenge, we propose a two-stage online learning architecture. *Firstly*, we design a new deep reinforcement learning model based on the state-of-the-art PPO algorithm [52], and associate one individualized RL model instance with each user. In this way, each user leads to a different RL model with its own learning experience. *Secondly*, we *aggregate* all users' experiences following a *federated learning* principle, so as to form a high-level model that can react to any network dynamics unseen by individuals. These two stages are coupled iteratively, so as to strike a balance between individualized experience and swarm intelligence.

(ii) Enforcing the RL algorithm under real-time video dynamics. A basic requirement for effective learning is that the learning algorithm's action should be faithfully executed. In our case, once the RL algorithm decides on a sending bitrate x, the video encoder should ideally produce the video stream at exactly the same rate. Though this is straightforward for offline simulation or VoD streaming, the video codec in real-time telephony cannot generate a perfect bitrate of x particularly in short time-scales [28]. Instead, the video codecs have their own control logic dependent on image scene dynamics, compression strategy and even device computing capacities, which results in inherent and substantial video bitrate fluctuation, and a gap between the RL's bitrate decision and the actual video traffic output rate (Sec. 3). To handle the problem, we adapt OnRL's learning algorithm by feeding the gap into the RL neural model. In this way, OnRL can learn the dynamics of the gap and then remedy the impacts by tuning its award operation, automatically. For instance, once it detects a large gap, OnRL will deem the previous RL action as contaminated, and reduce the action's impact by imposing a low weighting factor when computing the corresponding reward from the action.

(iii) Robust hybrid learning. In essence, an RL algorithm learns by following a trial-and-error principle, which risks disrupting the system when applied directly to online learning. Specifically, the algorithm may take incorrect exploitation actions (*e.g.*, selecting very high video bitrates under poor network conditions) that lead to catastrophic effects (*e.g.*, severe congestion and thus low QoE), especially during the early learning phase when the RL model is not well trained yet. To handle the problem, we design a *hybrid learning* mechanism, in which OnRL falls back to the legacy rule based video bitrate algorithm once the RL algorithm is regarded as operating abnormally (*e.g.*, continuously experiencing high loss rate and high latency), and switches back to RL otherwise. To manage such switching, we design an adaptive trend prediction algorithm to discriminate how well one control algorithm performs. Moreover, we augment the RL learning by mapping each such switching event as a penalty into the RL's reward function. In this way, the RL algorithm will learn that it should avoid invoking the legacy protocol as much as possible, and evolve to be a self-dependent and robust video bitrate adaptation algorithm.

We have implemented and deployed OnRL in Alibaba Taobao Live, a mainstream video telephony system[1]. We evaluate OnRL with 543 hours of Taobao Live sessions involving 151 beta users worldwide during one-month tests. The results confirm the advantages of online learning, which brings remarkable QoE improvement, *i.e.*, reducing video stalling time by 14.22%, while maintaining nearly equal video bitrate. Furthermore, our controlled experiments validate the individual design choices in OnRL and explain their effectiveness through micro-benchmarks.

Contributions: To our knowledge, OnRL represents the first online learning driven video telephony solution deployed in a large commercial system. OnRL resolves general problems that arise when RL meets real-time applications over the highly dynamic and heterogeneous mobile Internet. The specific contributions of OnRL include:

(i) We propose a two-stage iterative RL algorithm that enables online learning directly from massive concurrent video telephony sessions, instead of conventional simulation/emulation based offline learning (Sec. 2).

(ii) We design novel mechanisms to mitigate the inaccrate execution of actions when applying RL to real-time video applications (Sec. 3), and to boost the robustness of online learning (Sec. 4).

(iii) We implement and deploy OnRL on a mainstream commercial video telephony platform (Sec. 5) and validate its remarkable performance gain over the state-of-the-art solutions (Sec. 6).

[1]Taobao-live is part of the Taobao app–the 4th most popular mobile app in China with 666 million monthly active users [6, 7]. Over 1 million new users joined Taobao-Live during the single month of February 2020 [55].

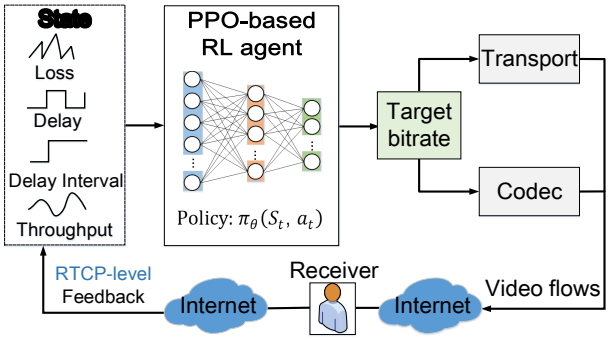


Figure 2: The workflow of OnRL individual learning.

2 PARALLEL ONLINE LEARNING: ARCHITECTURE AND ALGORITHM

2.1 Iterative Two-Stage Architecture

Existing machine learning driven video transport systems all adopt the "offline learning" approach [42, 60, 69] as illustrated in Fig. 1 (a). Such a system needs to first collect plethora of simulated [42, 60, 69] or real network traces [40, 65] from different users. Then these traces are serialized, *i.e.*, concatenated regardless of their inherent personal/temporal characteristics, and then replayed by a custom built simulator/emulator to train a neural network model (usually an RL model). Finally, the trained model is distributed to each user, who executes the learned bitrate adaptation strategy at run time.

However, such offline learning cannot deliver expected performance improvement due to the simulation-to-reality gap as reported in [40, 65, 66] and also corroborated in Sec. 6.2. Moreover, the serialized learning aggravates the gap: *(i)* The uniform trace serialization blurs the users' individual characteristics, *i.e.*, each user will get the same training experience, which loses the possibility of personalized optimization. *(ii)* Once the RL model is offline trained, it freezes and stops learning in the real-world. Therefore, it is unable to deal with new network conditions unseen before.

In contrast, OnRL adopts a fundamentally different design principle of learning at runtime. To realize the principle, we design *an iterative two-stage learning architecture* (Fig. 1(b)), which operates as follows, *(i) Stage-1: individualized learning.* Each user separately learns its own RL model at run time, based on a state-of-the-art PPO RL algorithm [52] with customized design for real-time video transport (Sec. 2.2). In this way, each user possesses an individualized model incorporating its personalized learning experiences. *(ii) Stage-2: Learning aggregation.* The individualized models are then fed to a backend server, and aggregated to generate a universal model. The aggregated model employs a weighted averaging over each model's neural model parameters following the principle of federated learning [44] (details in Sec. 2.3). From a high level, the aggregated model has experienced enough network variants from different users, and thus acquires the desired capability of coping with network dynamics.

The two stages operate iteratively. Specifically, the aggregated model will be distributed to all users, and each user, upon a new video telephony session, will start out with the latest aggregated model, and keep on learning from the new session. The procedure leads to a new individual model, which is fused at the back-end server for a new round of learning aggregation. The continuous iterations aim to strike a balance between each user's personalized experience and the swarm experience from all users. It is noteworthy that a mature aggregated model can be applied to a new user immediately, so as to eliminate the risk of low QoE due to random exploration at the bootstrapping phase.

We now proceed to elaborate on the individual learning algorithm and the learning aggregation, respectively.

2.2 Individualized Online Learning

We design a PPO-based RL algorithm for individualized online learning, as shown in Fig. 2. Specifically, the RL agent keeps observing the instantaneous network *state* S_t (*i.e.*, packet loss, delay, delay interval and throughput) at any time t, and deciding an *action* a_t (*i.e.*, a bitrate), which is expected to match currently available network bandwidth. Then the action a_t will be enforced on both the video codec and the transport layer protocol, which generates and consumes the video traffic at the rate a_t, respectively. The traffic, after going through the network path, will produce a new *state* S_{t+1}, which initiates a new round of RL *action*.

The mapping of $S_t \rightarrow a_t$, the key function of RL agent, is decided by the RL's control policy π_{S_t,a_t}, which is learned in the above process. Intuitively, if the RL agent produces a bitrate exceeding the available bandwidth, it will incur network congestion. The consequence will be reflected in the next state S_{t+1}, most likely with high packet loss or large delay. By observing the change from S_t to S_{t+1}, the RL agent can realize that it has made an inappropriate action a_t, so it needs to update the policy π to generate a more conservative bitrate when observing S_t or similar state next time.

Underlying the workflow above, we identify two requirements: *(i)* The RL agent should be very agile, *i.e.*, responding to network dynamics at video frame-level granularity, corresponding to a timescale of dozens of milliseconds. *(ii)* The RL agent should refrain from jumpy actions which hurt video perception quality and QoE [35]. To meet the requirements, OnRL first adopts a batch-level policy update running in the background at second level granularity, while executing agile response using the latest model at millisecond-level granularity. Secondly, OnRL guarantees smooth bitrate adaptation by utilizing a relatively stable policy adjustment mechanism. We now present OnRL's RL model and training methodology.

2.2.1 OnRL's PPO model. Fig. 3 depicts OnRL's RL model, including state, action, and the neural network architecture.

State. The state, also the input of the RL agent, at any time t, is denoted with $S_t = (\vec{l}_t, \vec{d}_t, \vec{i}_t, t_t)$, which represents the packet loss, delay, delay interval (*i.e.*, the difference between the arriving interval of two consecutive RTP packets at the receiver side and the corresponding departure interval at the sender side [22].), and the receiver-side throughput. These inputs can be easily derived at the sender side at a fine-grained timescale of around 50 ms, based on the periodic RTCP ACK message in WebRTC [3]– the transport/application layer protocol stack used in mainstream interactive video services.

Action. In each RTCP interval, OnRL maps S_t to a compact action space $\mathcal{A}: \{0.1Mbps, 0.2Mbps, \cdots, 2.5Mbps\}$, representing the target output bitrate of video codec. After the video traffic is transmitted through the network, OnRL will immediately transform to a new state S_{t+1}, and generate a new action a_{t+1}. Through such continuous iterations, OnRL will learn to cope with network dynamics.

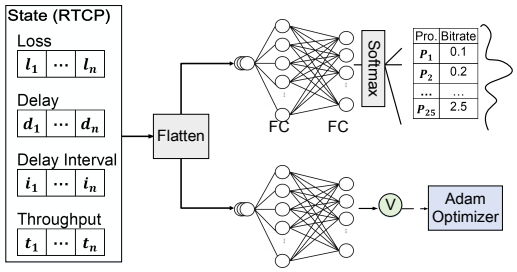


Figure 3: OnRL's RL model.

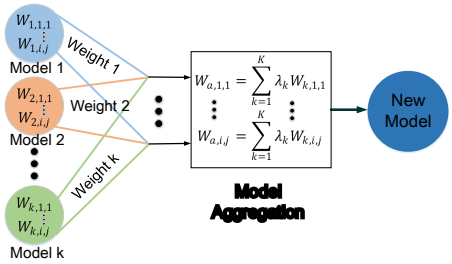


Figure 4: The workflow of learning aggregation.

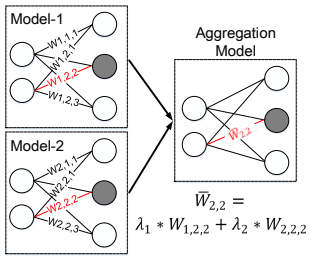


Figure 5: A showcase of the weighted models.

Reward. To ensure that OnRL can learn from past experience, each action is associated with a reward, which is the video QoE of previous time interval defined as follows,

$$r_t = \alpha \times \sum_{n=1}^{N} q_n - \beta \times \sum_{n=1}^{N} l_n - \eta \times \sum_{n=1}^{N} d_n - \varphi \times \sum_{n=1}^{N-1} \left| q_n - q_{n-1} \right| \quad (1)$$

where N represents the number of RTP packets in a state, q_n is the throughput measured at the receiver side which is directly related to the video quality, l_n and d_n are the packet loss rate and delay at the transport layer. The final term is used to enforce video smoothness by penalizing large bitrate fluctuations. These metrics are weighted by different impacting factors $\alpha, \beta, \eta, \varphi$, and then summed together. These hyper-parameters have significant impacts on training effectiveness. Clearly, the four metrics have different magnitudes. The value of q_n and the final item usually falls within 0.1~4.0 Mbps. l_n is commonly less than 10% in practical video telephony systems, while d_n ranges from dozens of milliseconds to hundreds of milliseconds depending on the end-to-end path length. To obtain a meaningful r_t, these metrics need to be normalized to a consistent range by adjusting values of the parameters $\alpha, \beta, \eta, \varphi$. In OnRL, the scope of the parameters is limited to 30~100, 50~100, 5~25, and 10~50, respectively. In the actual deployment of OnRL, they are empirically set to 50, 50, 10, 30, respectively.

Neural network (NN) architecture. OnRL uses a NN architecture to represent the policy π with a set of parameters θ. It adopts the simple fully-connected (FC) layer structures to distill implicit features hidden in different input elements. More specifically, OnRL first flattens the sequences $(\vec{l_t}, \vec{d_t}, \vec{i_t}, t_t)$, and then feeds them into two similar NNs, one is used for feature extraction and then outputs the bitrate action, and the other is served to judge the overall objectives like a critic. Each NN contains two FC layers, with 64 and 32 elements, respectively. We have empirically verified that flattening the input sequences performs better in terms of bandwidth prediction accuracy than feeding each state to a separate NN structure. In addition, each layer employs the tanh activation function [24], which shows more accurate bandwidth prediction capabilities than traditional relu or softmax activation. Besides FC, we have also tried CNN and long-short-term-memory (LSTM) for feature extraction, but their performance is not as good. Further analysis shows that the CNN architecture specializes in extracting image features that consist of complex spatial information, which does not exist in OnRL's state space. As for LSTM, it is more useful for reasoning the time-series data that takes into account long historic impact, but the performance of video telephony is more dependent on instantaneous network conditions.

2.2.2 Training Methodology. The objective of OnRL's training is to maximize the total cumulative rewards: $\mathbb{R}_\tau = \sum_{t'=t}^{T_\vartheta} \gamma^{t'-t} r_{t'}^{\vartheta}$, where γ servers as a discounting factor, usually customized as 0.99 or 0.9. The gradient policy updates for maximizing $\mathbb{R}_\tau$ can be formulated as follows:

$$\nabla \bar{\mathbb{R}}_\tau = \frac{1}{\Theta} \sum_{\vartheta=1}^{\Theta} \sum_{t=1}^{T_\vartheta} (\hat{A}(a_t^\vartheta, s_t^\vartheta)) \nabla log p_\theta(a_t^\vartheta, s_t^\vartheta) \quad (2)$$

where T_ϑ stands for the batch size for gradient policy updates, and Θ represents the number of batches. $\hat{A}(a_t^\vartheta, s_t^\vartheta) = \mathbb{R}_\tau - b$, (short as $\hat{A}$, $b \approx E[\mathbb{R}_\tau]$), represents the *advantage function* that expresses the difference in expected reward decided by practical action from state, compared with the averaged expected reward from policy π. Intuitively, the advantage function can show the extra benefit of a certain action over the average action. Recall that traditional policy gradient [48] may have large or little adjustment between contiguous policy, due to the uncertain learning rate value. This uncertainty can bring catastrophic QoE and difficult convergence in real-time video especially for the real-world learning of OnRL. To avoid this, we design two customized modules to maintain the online learning performance:

(i) Batch-level updates instead of instance-level (a single input) updates. As noted before, OnRL needs to respond to each input instance to adapt to the real-time bandwidth variation. Usually, the response also leads to an update of neural network parameters. However, such instance-level updates will largely increase the learning time and in turn slow down the response. We thus customize a *batch update policy*. In particular, the learning agent buffers the recent ⟨state, action, reward⟩ records. Only when the buffer is more than a batch-size, the agent will feed the buffer to the policy network to update the neural network parameters. In this way, the agent can execute fine-grained response in real time, while simultaneously running the online learning.

(ii) Smoothing-level updates. OnRL utilizes the loss function about gradient updates as follows,

$$L(\theta) = \hat{\mathbb{E}}[min(\frac{p_\theta(s,a)}{p_{\theta_{old}}(s,a)}\hat{A}, clip(\frac{p_\theta(s,a)}{p_{\theta_{old}}(s,a)}, 1-\varepsilon, 1+\varepsilon)\hat{A})] \quad (3)$$

where we bound the difference between a new policy and an old one, by taking into account their probability ratio $\frac{p_\theta(s,a)}{p_{\theta_{old}}(s,a)}$. Specifically, we clip the ratio and thus remove the incentive from of the loss function $L(\theta)$, when $\frac{p_\theta(s,a)}{p_{\theta_{old}}(s,a)}$ is beyond the interval [1-ε, 1+ε] (here ε is a hyper-parameter set to 0.2). In this way, OnRL will update a learning policy smoothly, and in consequence avoids too jumpy bitrate decisions, so as to enhance bitrate smoothness for better QoE.

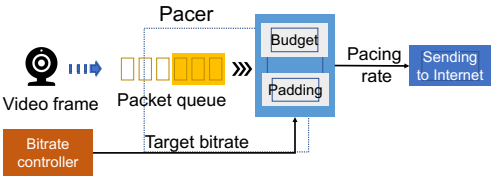


Figure 6: The pacing mechanism.

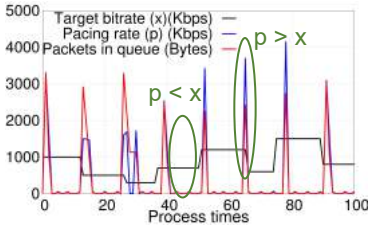


Figure 7: Action deviation.

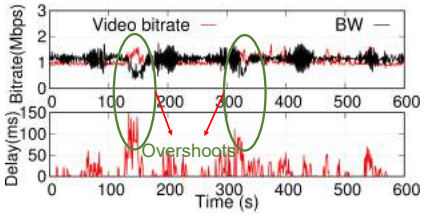


Figure 8: RL's bitrate overshoots.

Neural network implementation parameters. We use TensorFlow [13] to implement the training workflow and TFlearn [14] to construct the NN architecture. OnRL takes the fine-grained exploration in RTCP-level, and the batch size for batch training is 32 in our design. It is noteworthy that we have tried the common practice of batch size adjustment, *e.g.*, 16, 32 and 64, and did not observe substantial change on the QoE-related metric. We employ Adam optimizer [37] to update the stochastic gradient descent.

2.3 Learning Aggregation

The learning aggregation is inspired by two key observations from our analysis of the Taobao-Live traces: *(i)* Different users exhibit different levels of network dynamics. For instance, some users typically initialize video sessions at home (*e.g.*, make-up tutors or in-store shopping guides), with stable WiFi connections. Correspondingly, the PPO learning algorithm tends to generate stable video bitrate decisions. In contrast, other client users (*e.g.*, outdoor travelers) usually undergo fluctuant cellular network conditions, due to mobility or handoff. Accordingly, the learning algorithm will learn to change its video bitrates frequently to match the instantaneous network variations. *(ii)* While a user usually has relatively consistent network conditions, she may change the daily routines, thus encountering new network dynamics. In this case, an RL model purely trained from her own previous network conditions will react inappropriately.

To strike a balance between individual and swarm experience, we propose a *weighted model aggregation* method. Recent works [29, 44] have shown that, averaging the parameters of the same neuron across many NN models can achieve similar effect of aggregating these models' experience. We thus utilize the strong feature representation ability of NN parameters to realize the model aggregation, as shown in Fig. 4.

Specifically, suppose we have a total number of K users, and each user k's neural model can be denoted with a matrix $W_{k,i,j}$, where each element of $W_{k,i,j}$ is the parameter of a neuron in the neural model (i represents the i-th layer of the NN, and j is the j_{th} neural cell in each layer). Note that the dimensions of all matrices $W_{k,i,j}(k \in [1,K])$ are the same, since the NN architecture across all users is the same. To fuse the individual models, we perform a weighted averaging operation following the principle of federated learning [44]:

$$\overline{W} = \sum_{k=1}^{K} \sum_{i=1}^{I} \sum_{j=1}^{J} \lambda_k W_{k,i,j} \tag{4}$$

where $\overline{W}$ represents the aggregated model, and λ_k is the weight of user k's model. We provide an example in Fig. 5 to illustrate the model averaging process. In particular, the weights $\overline{W}_{2,2}$ of the dark neural cell is computed from the corresponding neuron position of two individual models: model-1 and model-2.

We have two ways to determine the value of each λ_k:

(i) Average aggregation. For a newly joined user (*i.e.*, a user without any prior experience), we generate and apply a model with average experience from all users, *i.e.*, we let $\lambda_k = \frac{1}{K}$ for $k \in [1,K]$.

(ii) Prioritized aggregation. For other users, we prioritize a user's own weight thus it can achieve optimal performance under its typical network conditions, while reacting appropriately when confronting unusual conditions. Specifically, for each user k, we let $\lambda_k = p$ ($p \in [0,1]$), and $\lambda_m = \frac{1-p}{K-1}, \forall m \neq k$. In Sec. 6, we experimentally evaluate the impact of weight parameter of p.

3 ENFORCING RL ACTIONS IN REAL-TIME VIDEO TELEPHONY

Understanding action deviation. In a practical video telephony system such as WebRTC, a single video frame often needs to instantly burst through the network. To avoid transient congestion, a *pacer* mechanism is introduced to evenly distribute the video data across continuous time slices. A simplified pacing workflow is illustrated in Fig. 6. When a video frame is produced, it is fed into a pacer queue. The pacer splits the video frame into groups of packets and progressively injects them into the network path every Δ interval (usually 5ms). The pacer has its own control logic (*e.g.*, budget and padding) to send these packets based on the instantaneous target bitrate (denoted as x), which is generated by congestion control algorithms (*e.g.*, GCC in WebRTC). Surprisingly, by analyzing the behavior of the default WebRTC pacer used in Taobao-Live, we find that the pacer's actual sending rate still deviates wildly from the target x. Our analysis reveals two reasons that inevitably cause the gaps.

(i) The pacer occasionally needs to increase the sending bitrate to accelerate its buffer flushing. The pacer queue has its own control logic which depends on image scene dynamics, compression strategy and even end devices' computing capacities. On the other hand, the video traffic is quite bursty, *e.g.*, a key-frame may comprise excessive data when the video scene changes quickly. In such a case, if the pacing rate strictly executes x, it will cause long latency in the pacer queue, which may lead to a detrimental freezing effect for real-time video. Hence, the pacing rate will be recalculated by multiplying a factor (typically around 2.5 in WebRTC) to accelerate the buffer clearance. This means that *RL's action (x) cannot be strictly executed by the interactive video system, which compromises its effectiveness and may eventually mislead its policy update.*

(ii) The pacer sometimes has insufficient video data to meet the target bitrate x, *e.g.*, given relatively static video scenes. Whereas the padding logic inside the pacer can artificially add some dummy data packets, we find that such a mechanism seldom works for real-time video. The root cause is that the dummy packets bring more

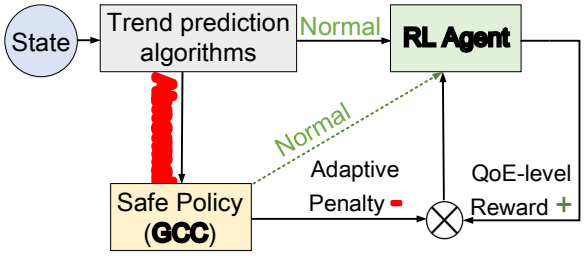


Figure 9: Hybrid learning mechanism of OnRL.

traffic and worsen the bandwidth cost. Besides, the padding data tend to increase the latency of actual video packets and hence lower the QoE. As a result, the operational video telephony services like Taobao-Live are usually reluctant to use padding, and the target bitrate of being commanded by RL model cannot be accurately enforced.

We showcase the action deviation by logging and plotting a segment of the target bitrate, pacing rate, and queue size in the pacing buffer sampled from a video session, in Fig. 7. We can find that, when the pacer queue accumulates many packets (*e.g.*, larger than 1000 bytes whenever key-frames appear), the pacer will execute a sending bitrate p higher than the target bitrate x to accelerate the buffer flushing. In contrast, when the queue is almost empty, the actual sending rates usually drop to zero due to the absence of padding.

Learning to tolerate the action deviation. To overcome the aforementioned problem, we further customize OnRL's RL model to implicitly learn the gap between the target bitrate and actual video traffic. We realize this by simply feeding the gap into the RL neural model. Specially, we define the gap as $g_t = x_t - p_t$, where x_t stands for the target rate of OnRL's RL action at time t, and p_t is the pacing rate. Then the state/input in Sec. 2.2 is updated to $S'_t = (\vec{l_t}, \vec{d_t}, \vec{i_t}, t_t, g_t)$. In this way, OnRL can learn the dynamics of the gap and then remedy the impacts by tuning its reward operation automatically. Once it detects a large gap (*e.g.*, $|g_t| > 0.5$ Mbps), OnRL will deem the previous RL action as corrupted, and reduce the action's impact by imposing a low weighting factor (default 0.5 in our design) in the cumulative reward (Sec. 2.2.2). Sec. 6.4 verifies the effectiveness of this mechanism compared with the gap-oblivious RL model.

4 ROBUST HYBRID LEARNING

In this section, we describe the hybrid learning policy to ensure the reliability of RL when it is trained online subject to real network dynamics. Typical RL models learn in a trial-and-error manner, whereas at run time, any erroneous action may lead to severe QoE degradation for the interactive video. We provide a showcase in Fig. 8. The video bitrate, packet delay, and ground truth network bandwidth are sampled from a controlled experiment on a local testbed (the testbed implementation will be introduced in Sec. 5). Within a period of 10 minutes, there exist 8 instances where the RL-generated bitrate overshoots the available bandwidth, causing latency spikes. Ideally, the RL algorithm should learn from these mistakes. But unlike simulation based training, accumulating experience through failures is unacceptable for real-time video. This is likely to happen particularly at the beginning of a video session, where the RL model may need to keep exploiting and failing, so as to materialize its policy.

On the other hand, traditional rule-based video transport control algorithms, such as GCC [22], tend to make conservative decisions. So a straightforward solution that ensures the reliability of online learning is to combine the rule-based algorithm and the RL algorithm, *i.e.*, switching to a conservative algorithm once RL's decision becomes too aggressive. However, this compromises RL's learning capacity, *i.e.*, the RL's learning experience is interrupted and it becomes oblivious of the network state changes during its absence period. As a result, RL cannot react appropriately in the future. Therefore, the key question is: *How to suppress the RL algorithm's catastrophic exploration actions while preserving its learning ability at the same time?*

To meet this challenge, we design a *hybrid leaning mechanism* that integrates the RL and conservative algorithm in a coupled close-loop. Intuitively, we do not merely regard the rule-based algorithm (*i.e.*, GCC in our design) as a protective backup, but treat it as a "teacher" for the RL algorithm and provide useful feedback to prevent inappropriate actions. We believe that the hybrid complexity is critical for ensuring robustness. In particular, since OnRL runs in an operational video telephony system, any erroneous action caused by trial-and-error policies may lead to catastrophic degradation of the system performance at run time.

Fig. 9 illustrates the workflow of hybrid learning scheme, with the following key elements. *(i)* Different from the pure RL learning (Sec. 2.2), we introduce a safety condition detector to evaluate whether the RL algorithm functions properly, by examining the current input state. *(ii)* If so, the input state goes to the RL agent, who will be in charge of bitrate adaptation following the process in Sec. 2.2. *(iii)* Otherwise, GCC will take over. Remarkably, there will be a penalty whenever GCC is invoked, which is incorporated into the RL's reward function. In this way, OnRL can still learn implicitly even when GCC is running. *(iv)* OnRL will roll back to RL once the detector decides that the possibility of RL causing QoE damage is low. Note that the hybrid switches will mostly happen in the early stage of online learning. After the penalty feedback becomes effective beyond the early stage, the OnRL agent will learn to converge to appropriate actions and switch to GCC as less as possible. We now proceed to detail the design of the safety condition detector and penalty feedback in the hybrid learning framework.

Safety condition detector. Due to the elusive network conditions, it is non-trivial to detect whether the learning agent has made an inaccurate action. To enable early detection while suppressing false positives, we design a delay-based filter to predict video QoE damage, which is inspired by the delay based congestion control mechanism in [22, 61].

The basic idea is to detect whether recently received latency sequence is showing a rising trend. If so, the filter will command OnRL to switch from RL to GCC. Specifically, we employ the inter-packet interval $\Delta d(t_i)$ sequences to estimate the queuing build-up trend on the network path, defined as follows:

$$\Delta d(t_i) = (a_i - a_{i-1}) - (s_i - s_{i-1}) \tag{5}$$

where s_i and a_i denote the sending and receiving timestamp of the i-th RTP packet. At time T, the possibility of QoE degradation in the near future can be expressed by the historical sequences: $\{\Delta d(t_0), \Delta d(t_1), ..., \Delta d(t_T)\}$. We define the *possibility metric* by an

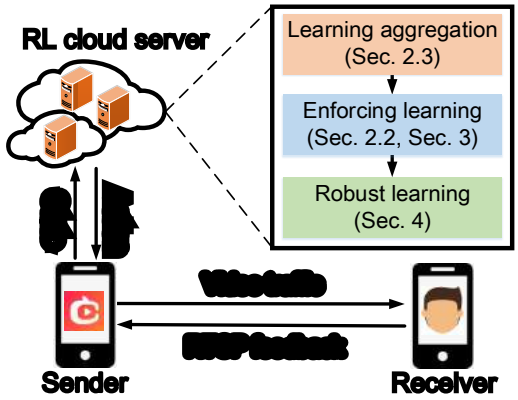


Figure 10: System implementation of OnRL.

exponential weighted average value $D(t_i)$, as follows,

$$D(t_i) = \sum_{i=1}^{T} 2^{-i} \times \Delta d(t_{T-i}) \tag{6}$$

where the closer time a delay condition, the more important weight it has in $D(t_i)$. Note that the filter computes $D(t)$ using a sliding window along t, and once $D(t_i)$ exceeds a threshold $\gamma(t)$, the filter predicts a high risk of QoE degradation, and then switches to GCC. Since GCC is latency-sensitive, its control policy will immediately decrease current sending bitrate, thereby mitigating the risky situation. On the other hand, once the condition returns to be safe (*i.e.*, no trend of latency growth), the RL policy will take control.

One challenge here is to determine the threshold $\gamma(t)$. We note that a constant threshold cannot handle inherent network heterogeneity and dynamics. So we design an adaptive threshold $\gamma(t_i)$ which evolves with the latency sequences:

$$\gamma(t_i) = \gamma(t_{i-1}) + k_\gamma \times (|D(t_i)| - \gamma(t_{i-1})) \tag{7}$$

Eq. (7) is a function of the adaptive threshold $\gamma(t_i)$, which evolves with the sliding window of latency. Specifically, k_γ is a constant to determine the ratio of $\gamma(t_i)$'s increase/decrease. Intuitively, once the exponential weighted average value $D(t_i)$ becomes larger within last threshold, the dynamic threshold $\gamma(t_i)$ tends to be decreased, and consequently OnRL has a larger possibility to evoke GCC.

Switching penalty for RL. We leverage the switch between RL and GCC as a feedback to augment the RL model. Each switching event is configured as an extra penalty into the agent's reward function. In this way, it will learn to act appropriately and thus switch to GCC as less as possible in the future. Specifically, we design an adaptive penalty parameter η' to replace the default η in the standard RL award in Eq. (1):

$$\eta' = \eta \times 2^{\epsilon}, subject\ to\ \epsilon = latency/10 \tag{8}$$

Under the η' regulation, the larger latency that the RL action triggers, the larger penalty will get in the overall process. We will validate the design of the entire hybrid learning scheme in Sec. 6.

5 IMPLEMENTATION

Implementation on Taobao-Live. We implement OnRL based on the operational Taobao-Live video telephony system and release it as a beta app to users. Taobao-Live builds on WebRTC [3], a real-time video communication framework with built-in support of video codec and transport-layer protocol (*i.e.*, GCC). WebRTC allows flexible re-implementation of the video control algorithms, and has been used in the state-of-the-art transport and video application studies, such as BBR [21], Salsify [28] and Concerto [69], *etc.*

Our OnRL implementation essentially replaces the existing bitrate control module in Taobao-Live. Ideally, OnRL's components should be implemented inside the Taobao-Live app. However, due to the lack of API support for training RL neural networks on mobile devices [57], we implement OnRL via a cloud-assisted framework, as shown in Fig. 10. Besides the pair of video telephony's sender and receiver, we introduce an RL server, on which we implement the three design components of OnRL (*i.e.*, the two-stage iterative learning, coordination learning and robust learning) based on Tensorflow [13]. During each telephony session, the sender maintains a connection and exchanges information with the RL server. It collects RTCP feedbacks (*i.e.*, packet loss, delay, throughput, *etc.*) from the receiver and sends them to the RL server as the input of OnRL. Then OnRL processes the input and returns an action (*i.e.*, the target video bitrate) to the sender, which is then executed by Taobao-Live. Meanwhile, the OnRL module on the server periodically updates its control policy to realize online learning. We found that the information exchange latency from the sender to the server is only about 10 ms in most cases, which has negligible impacts on video telephony as will be validated in Sec. 6.3.

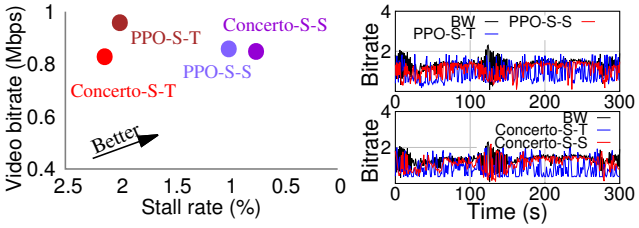
To handle some extreme events (*e.g.*, possible connection failure between the sender and RL server, or when OnRL consistently leads to low QoE), we also implement a fallback mechanism on the sender, *i.e.*, the sender automatically downgrades to the default WebRTC controller. In addition, we deploy a separate back-end server to perform learning aggregation at the frequency of once per day (typically in early morning with the least user activity).

RL server. Each RL server is a COTS PC equipped with 32 kernels, 2.5-GHz CPU and 65-GB memory, and runs in RedHat Linux 4.8.5-11. We adopt the Tensorflow version 1.15.2 to host OnRL's neural network, and each server can accommodate at least 50 concurrent users, due to the efficient neural network structure of OnRL with a small size of *32KB* and runtime memory requirement of 0.5 GB. Currently, we have deployed 5 RL servers for our beta testing and OnRL's high *scalability* is guaranteed using the Alibaba cloud management system Apsara [4], which has the capacity of efficient resource scheduling and load balancing across millions of servers [8].

Simulator and local testbed. Besides the operational app, we have also developed a trace-driven simulator similar to the one used in existing work [69]. We also follow [69] to build a local video telephony testbed, comprised of a client and server, whose end-to-end network bandwidth follows the traces of which we collect from Taobao-Live sessions and is enforced by Linux tc [11]. Unlike the operational app, these platforms allow for controlled experiments with known bandwidth ground-truth, which facilitates deeper diagnosis of the results.

6 EVALUATION

In this section, we evaluate OnRL from three aspects: *(i)* We validate the necessity of online learning by comparing the performance of models under different algorithms and operating environments including simulation, testbed and real networks (Sec. 6.2). *(ii)* We deploy and evaluate OnRL in the operational Taobao-Live with a real-world user base (Sec. 6.3). *(iii)* We evaluate each design module



(a) QoE metrics. (b) A showcase.

Figure 11: Performance of simulation-trained models under simulator and testbed, respectively.

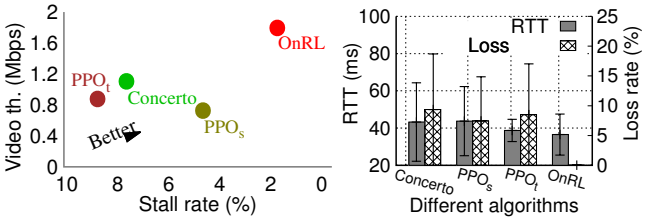


Figure 12: Performance comparison of different models running over Taobao-Live.

inside OnRL separately to gain an in-depth microscopic understanding of online learning (Sec. 6.4).

6.1 Methodology

Evaluation Metrics. We employ both application-layer and transport-layer metrics for a comprehensive evaluation, including: *(i)* Video throughput and Peak Signal to Noise Ratio (PSRN) that characterize the perceived quality of the video frames [32]. Note that our experiments are performed in both controlled testbed and the operational Taobao-Live system. Whereas PSNR can be easily measured on the testbed, it is unavailable in the Taobao-Live system. As a remedy, we use the receiver-side video throughput as a metric to reflect the video quality. *(ii)* Stall rate to measure the fluency of the video telephony session. In the Taobao-Live system, a stalling happens whenever the packet RTT exceeds 300ms [2]. Here we remark that the stall rate indicates how frequent video freezing occurs, which is different from stalling duration. In the simulator/testbed in-lab environments, the RTT hardly rises to 300ms, so a stalling event is considered to occur when the instantaneous video frame rate drops below 12 fps, as suggested by DevOps engineers of Taobao-Live. *(iii)* We also examine the transport layer metrics including packet loss rate and RTT during video telephony, which help to understand OnRL's behaviors in-depth and explain the application-layer performance.

Baseline methods. We compare OnRL against the following algorithms which represent the state-of-the-art in video telephony: *(i)* Concerto [69], which utilizes deep imitation learning [50] to optimize video QoE. As a semi-supervised algorithm, Concerto is trained in a simulator with known ground-truth network bandwidth. The trained model is then deployed and tested in real applications. In this work, we run Concerto using its original model [69] which is trained over 1 million real-world video telephony sessions unless otherwise stated. *(ii)* WebRTC [3], which is the most popular rule-based video telephony framework. WebRTC has been incorporated into Google Chrome [9] and mainstream real-time video apps, including Google Hangout, Facebook messenger, Amazon Chime and Taobao Live.

6.2 The Need for Online Learning

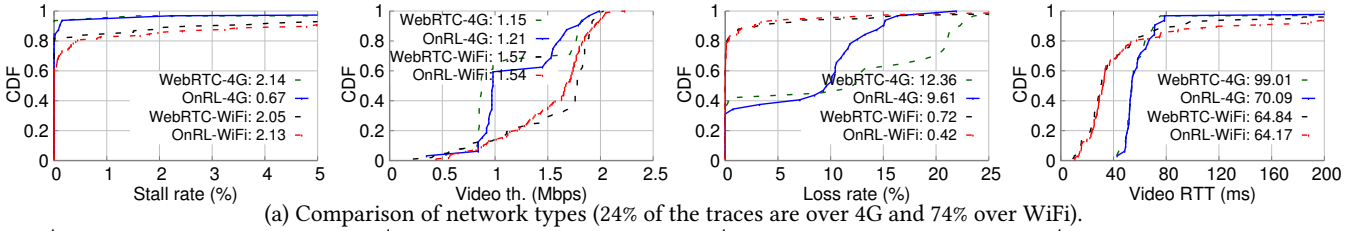
The gap between simulation and testbed. In order to validate the limitations of "offline learning", we first run a micro-benchmark to compare the performance of simulation-trained models, *i.e.*, Concerto and the basic PPO model described in Sec. 2.2. To ensure fairness, we use the same 3-hour trace in two different testing environments: the simulator and local testbed. The resulting video bitrates and stall rates are presented in Fig. 11(a), which confirms that Concerto and PPO work well when the testing is done in the same environment as the training (*i.e.*, the cases marked as Concerto-S-S and PPO-S-S). However, the performance drops significantly after the testing environment is changed to be the testbed (*i.e.*, Concerto-S-T and PPO-S-T). Specifically, their stall rates increase by 64.9% and 51.2% respectively, with minor video bitrate deviation (between 0.02 Mbps and 0.1 Mbps). These results clearly validate the necessity of online learning. To understand the performance drop, we plot a 300-second segment of the models' actions (*i.e.*, video bitrates) against the ground-truth bandwidth in Fig. 11(b). We observe that both Concerto and PPO in simulation environment can follow the bandwidth very closely despite high dynamics, with average deviations of only 17.4% and 13.6%, respectively. In contrast, the deviation increases to 30.3% and 37.12%, when coping with the same traces in the testbed environment. *To summarize, once the deployment environment differs from the training environment, the offline RL models' experience becomes stale, leading to unsatisfactory performance.*

The gap between simulation/testbed and real-world network conditions. We further examine the limitations of "offline learning" under real-world network dynamics. In addition to OnRL, we intentionally integrate three offline-trained models into the Taobao-Live system, including the Concerto and PPO models trained in simulator (referred to as PPO_s), and the PPO model trained in testbed (referred to as PPO_t). We then run the models on a randomly selected Taobao-Live app user. Note that the offline-trained models all showed satisfactory video QoE in their own training environment as validated above. We plot the average video throughput, stall rate, packet rtt and loss rate in Fig. 12, over a random set of video sessions lasting 10 hours in total. We find that OnRL achieves the most compelling video QoE. In particular, it outperforms the most competent offline-trained scheme Concerto by 31.9% in terms of video throughput, and leads to a remarkable 78.3% reduction in stall rate, which again corroborates the benefits of online learning mechanism. On the other hand, PPO_t and PPO_s exhibit large QoE gaps on both metrics. Accordingly, OnRL also shows the smallest packet delays and loss rates, which confirms its ability to cope with network dynamics. *The result further validates the necessity of online learning for optimizing real-time interactive video applications under real network conditions.*

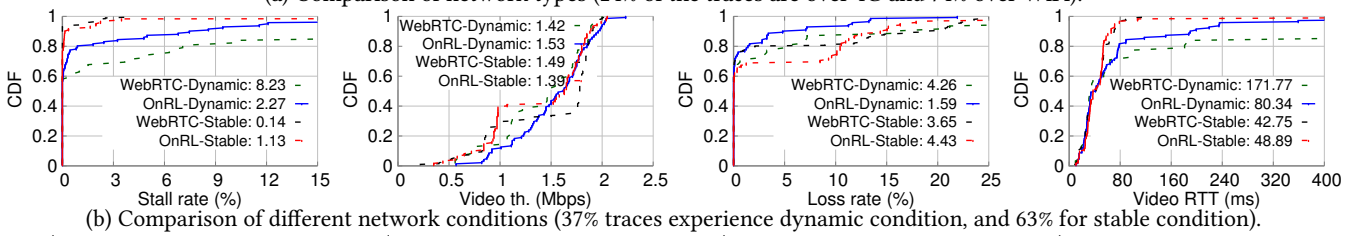
In particular, we find that PPO_t, which is equipped with the same pacing mechanism with OnRL in Taobao-Live, performs worse (*e.g.*, 6.88% lower stall rate and 0.75 Mbps lower throughput) than OnRL, which demonstrates that the online learning, instead of the pacing mechanism, is the reason for performance improvement.

6.3 System-level evaluation in the wild

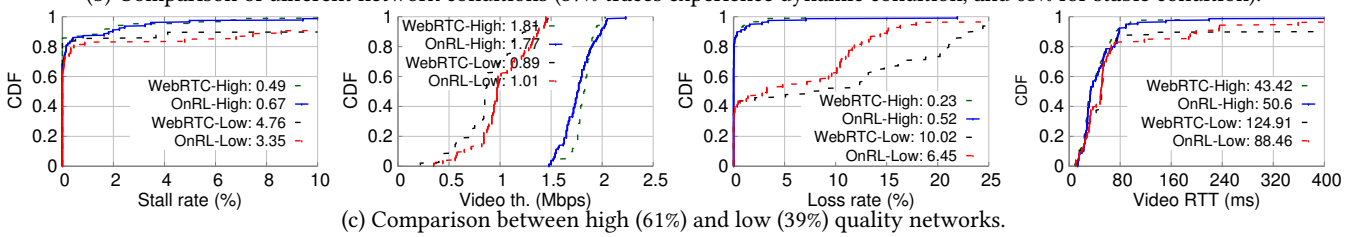
We distribute a beta version of Taobao-Live equipped with OnRL



(a) Comparison of network types (24% of the traces are over 4G and 74% over WiFi).



(b) Comparison of different network conditions (37% traces experience dynamic condition, and 63% for stable condition).



(c) Comparison between high (61%) and low (39%) quality networks.

Figure 13: OnRL's performance while running in Taobao-Live: a breakdown analysis.

Table 1: Average statistics from Taobao-Live.

Metrics	Stall rate (%)	Video th. (Mbps)	Pkt loss (%)	Rtt (ms)
WebRTC	2.04	**1.48**	3.79	73.05
OnRL	**1.75**	1.46	**2.90**	**65.8**

to real-world users for a large-scale system-level evaluation. In particular, we have recruited 151 users from 30 cities (3 abroad, 27 domestic), and collected 543 hours of video telephony sessions from them. In each session trace, we log the network/application performance metrics (Sec. 6.1) at second-level granularity, which form a dataset of 4.55 GB in total. Currently, OnRL only supports iOS, and the end user devices range from iPhone 7 to iPhone 11 Pro. It is infeasible to implement all the ML based algorithms directly on the operational Taobao-Live app, so we mainly compare OnRL against the default WebRTC algorithm for an in-depth analysis. For a fair comparison, we command each session to use the default WebRTC for the first 10 minutes, and then switch to OnRL afterwards. In this way, we can roughly regard the two algorithms undergoing nearly the same network conditions.

Overall performance. Table 1 summarizes the QoE metrics averaged over all sessions. We observe that, *(i)* OnRL achieves remarkable improvement in terms of transport-layer metrics: 23.5% less packet loss, and 9.92% RTT reduction. *(ii)* The transport-layer advantages transform into application-layer performance gains, *i.e.*, reducing stall rate by 14.22% while maintaining almost the same video throughput. It is noteworthy that the stalling rate here differs from the stalling time metric in Concerto [69] and the user base also differs (151 vs. 6). Thus, the real-world evaluation results are not directly comparable (a direct comparison under the same conditions has been presented in Sec. 6.2). Furthermore, the performance gain may seem modest compared with RL-based VoD systems driven by a simulator/emulator [42], primarily because the operational video telephony system involves real-world users with more heterogeneous network conditions. In addition, for the operational Taobao-Live system, even a small improvement is significant, considering that it can benefit millions of sessions each day. Below we provide a more in-depth analysis of the results.

Performance breakdown. We group the users according to various session attributes, and replot the evaluation results in Fig. 13. Here the number following each legend label stands for the average value of the performance metric.

(i) 4G vs. WiFi. OnRL significantly reduces stall rate by 68.7% and slightly increases the video throughput over 4G, while has very close performance (difference within 3%) over WiFi networks. Accordingly, the reductions in terms of packet loss and RTT are more remarkable over 4G networks. For Taobao-Live use cases, most WiFi users stay indoor with relatively stationary setup, in comparison to 4G which tends to serve more dynamic outdoor scenarios. Therefore, the experiment implies that *OnRL's advantages become more prominent when handling dynamic network conditions.*

(ii) Dynamic vs. stable networks. We thus divide all sessions into stable/dynamic categories according to the variance of video throughput (separated by a threshold set to be the mean std. of 0.197 Mbps). We find that, OnRL improves QoE significantly over dynamics networks, while is not as good (sometimes slightly worse) over stable networks. For instance, the average video RTT is decreased from 171.77ms to 80.34ms for dynamic networks, but is increased from 42.75ms to 48.89ms for stable networks. The other metrics have the similar trend. Still, we conjecture that the main reason behind lies in OnRL's abilities to explore and closely track the available bandwidth under network dynamics. If the network bandwidth is stable, the exploring mechanism may cause overshoots, which result in relatively lower video quality.

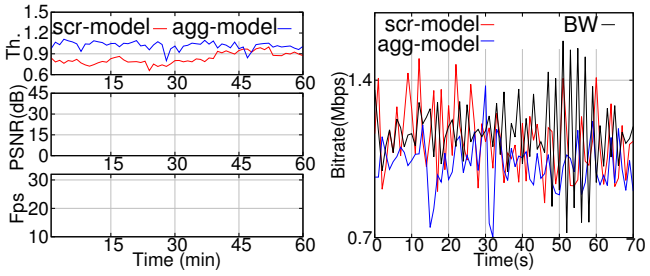
(iii) High vs. low network quality. We further categorize the video sessions based on the empirically observed network condition. If one session's throughput is larger than the mean of all the sessions,

Table 2: The effect of handling RL action deviation.

Metrics	Video th. (Mbps)	Stall rate (%)	Loss rate (%)	PSNR (dB)
Ignoring deviation	0.81	2.56	5.11	35.89
Considering deviation	0.79	1.14	4.58	34.21

Table 3: The effect of robust learning.

Metrics	Video th. (Mbps)	Stall rate (%)	Loss rate (%)	PSNR (dB)
RL-only	0.96	2.30	5.05	38.71
Robust-RL	0.93	0.99	1.84	42.28



(a) QoE metrics. (b) A showcase of bitrate.

Figure 14: Video performance with and without learning aggregation.

we regard the corresponding network quality as high, and low otherwise. We find that OnRL performs significantly better in low-quality network conditions.

6.4 Detailed Analysis of OnRL Pipeline

We now study the impact of each design module inside OnRL separately. Here we conduct controlled experiments on our local testbed since: *(i)* The operational Taobao-Live does not allow testing of separate modules on the end users. *(ii)* We need ground-truth bandwidth for an in-depth understanding of each module's behaviors, which is not available when running Taobao-Live in real networks.

The gain from learning aggregation. We first demonstrate the gain of exploiting prior learning experience, by comparing the performance of an aggregated model (*agg-model*) against the one that starts from scratch (*scr-model*). More specifically, we train 8 separate models from randomly selected Taobao-Live session traces, each lasting longer than 2 hours. Then we aggregate them through averaging as specified in Sec. 2.3. Fig. 14(a) plots the video throughput, PSNR and fps, when running agg-model and scr-model over the same set of new traces lasting 1 hour. We find that, agg-model outperforms scr-model on all important QoE-related metrics: throughput, PSNR, and fps gains are 18.2%, 41.1%, and 47.6%, respectively. Moreover, we can observe that the lower PSNR (< 15dB) under scr-model indicating extremely low quality picture happens at the beginning phase, which demonstrates the cost of starting without any experience.

To further understand above results, we analyze the two models' actions (*i.e.*, the video sending bitrates) in Fig. 14(b). We see that *scr-model* usually sends more traffic than the available bandwidth, which incurs congestion and thus low QoE. Moreover, the actions of *scr-model* exhibit 47.62% higher variance than that of the agg-model, indicating it struggles to explore the available bandwidth at the cost of overshooting/underuse.

The impact of the number of models involved in aggregation. We train 15 different models using randomly 10 hour traces and select 0 (*i.e.*, starting from scratch), 5, 10, and 15 out of them for aggregation. We examine the performance of the resulting 4 aggregated models under a same 1-hour test trace, as depicted in Fig. 15. We observe that: *(i)* Compared with the case with 0 aggregation, the aggregation of 5, 10, 15 models reduce the stall rate by 77.1%, 80.7%, 83.7%, respectively. *(ii)* While model aggregation also improves video throughput, the gain is relatively lower, *i.e.*, 10.8%, 21.27% and 21.3%, respectively. Overall, *the more models involved in aggregation, the more swarm intelligence the aggregated model will have, and the better it can cope with the complex network dynamics.*

The effect of prioritized aggregation. We aggregate 8 random individual models (Each model is trained for 1-hour) in 3 ways: averaged-model, a single model without averaging (named model-1), and prioritized-model (assigning a controlled weight of $p = \lambda_k$ to model-1, $(1-\lambda_k)/7$ to the remaining models, and we vary the range of λ_k from 0.1 to 0.9.). Here we intentionally choose 3 traces from model-1's user, over which the eleven models run. We make an average over 3 traces to mitigate occasionality, and depict the results in Fig. 16. We observe that, *(i)* Even the simplest learning aggregation (averaged-model, stage-2 of OnRL) has 44.9%, 7.3% gains on stall rate and throughput than model-1 (stage-1 of individualized learning). The result indicates that OnRL's learning aggregation stage indeed helps to improve QoE after leveraging swarm intelligence. *(ii)* While the throughput is relatively similar (the largest difference occurs under the two settings of $p = 0.3$ and $p = 0.7$, with a gap of 0.13 Mbps), the prioritized-models have much lower stall rate than model-1 and averaged-model. In particular, $p = 0.4$ shows the lowest stall of 0.27%, while that of model-1 and averaged-model is 4.09% and 2.25%, respectively. *(iii)* Different weighted value of prioritized-models has certain impacts on the QoE metrics. Specially, we find that, the benefits of boundary weights (*e.g.*, $p = 0.1, 0.2, 0.7, 0.9$) are not as good as the middle values (*e.g.*, $p = 0.3, 0.5$). These findings indicate that *for an existing user who already has a trained model, it is better to balance its individual experience with others', by using medium aggregation weights (i.e., close to 0.5), so as to achieve the optimal QoE.*

Effect of handling RL action deviation. We compare two models with and without handling the action deviation, by running them under the same randomly selected network trace lasting 2 hours, respectively. From the result in Table 2, we find that by taking the deviation as an extra input to the RL, the stall rate decreases significantly by 55.4%, loss rate degradation by 10.3%, with slight sacrifice on throughput (0.02Mbps), and PSNR (1.68dB). We thus conclude that *the performance of online training benefits from learning the action deviation explicitly, and the model can therefore achieve high performance even though its action is not strictly executed.*

Effect of robust hybrid learning. We now run OnRL with and without the robust hybrid learning mechanism, labeled as RL-only and Robust-RL, respectively. Table 3 summarizes the performance when the two schemes run on the same 1-hour trace. We observe that *Robust-RL can significantly improve the robustness*: the stall rate and loss rate are reduced by 56.9% and 63.5%, respectively. From the video definition's perspective, robust-RL maintains nearly the same throughput, while improving PSNR by 3.57 dB. To understand the effect of Robust-RL more intuitively, Fig. 17 showcases a 150s

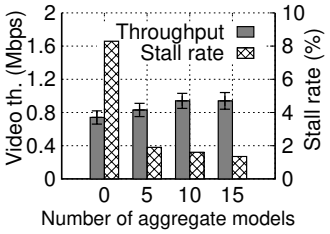


Figure 15: QoE vs. number of models in aggregation.

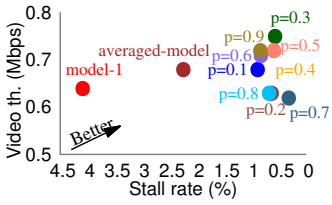


Figure 16: Effect of prioritized aggregation.

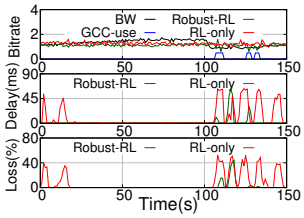


Figure 17: A showcase of Robust-RL's effect.

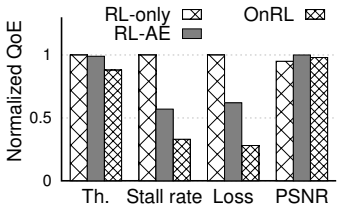


Figure 18: Contribution of separate modules in OnRL.

segment of the experimental traces. We find that during 25 to 100 seconds, the loss rate and delay are relatively low, which shows that the video playback is smooth and the three methods all perform relatively conservatively. Then, we can observe that when RL performs poorly (*e.g.*, during 100s–150s), the loss and delay performance degrades sharply. Robust-RL manages to promptly switch to the conservative GCC algorithm, so as to avoid QoE damage.

In addition, within the 1-hour testing period, we found that Robust-RL evokes GCC for 17 times, and GCC lasts only 28 seconds in total. In particular, at around 10 minutes, Robust-RL has a more frequent switching of 7 times, with GCC lasting 13 seconds in total and taking up around half of the GCC time across the whole session. The results imply that hybrid learning mostly happens in the early stage of online learning. Afterwards, the OnRL agent will learn to converge to appropriate actions and rarely evoke GCC.

Contribution of separate modules in OnRL. We further run a progressive experiment to demonstrate the benefit of OnRL's three modules: *(a)* Pure online RL-only, *(b)* Online RL with action enforcement (RL-AE), and *(c)* Three modules putting together (OnRL). We examine the performance of above three methods under the same 1-hour test trace, and plot the normalized results in Fig. 18. We observe that: *(i)* OnRL's action enforcement and robust learning mechanism have significant impacts on stall rate and loss rate. In particular, the action enforcement mechanism (RL-AE) decreases the stall rate by 42.8%, and the robust learning mechanism further decreases the stall rate by 66.6%. Similarly, the loss rate is reduced by 38.1% and 72.0%, respectively by the two mechanisms. The reason lies in that OnRL's RL-AE is beneficial due to learning the action deviation explicitly, and robust learning can further decrease stalls by mitigating RL's trial-and-error impacts. *(ii)* OnRL's action enforcement and robust learning modules lead to slightly lower throughput and higher PSNR than RL-only, which indicates that they reduce stall rate and packet loss without harm on other performance metrics.

7 RELATED WORK

Low-latency video transport. Real-time Internet video applications impose the toughest requirements on data transport protocols [5, 34]. Although the transport layer has been studied for decades, new network characteristics and applications are constantly emerging that motivate new designs. In contrast to the packet loss metrics commonly used in traditional congestion control protocols [30, 36], many emerging systems target low latency and have explored customized congestion indicators, including a mix of delay and loss [22, 61, 64, 68] as well low-layer KPIs of cellular networks [62, 63]. While these methods can better balance the tradeoff between delay and throughput, they rely on a set of hand-crafted controlling rules, and fall short of handling the increasing heterogeneity and dynamics especially for mobile networks [60].

Starting from Remy [60], machine learning models and particularly RL are utilized to generate more adaptive controlling rules automatically beyond the hand-crafted ones. Remy [60] explores a Markov model (*i.e.*, a kind of tabular RL) to optimize congestion control algorithms. A recent system, Concerto [69], designs an imitation learning (IL, a type of RL) based algorithm to better coordinate the codec and transport layers of video telephony. While making remarkable progress, these models are all trained offline on simulators or emulators. Due to the simulation-to-reality gap, as reported in [20] and corroborated by our experiments (Sec. 6), these approaches often deliver marginal performance gain when tested at large-scale under real network conditions [40].

Learning-based transport for Video-on-Demand (VoD). Learning-based optimization has also been used in VoD services. To name a few, Pensieve [42] adopts the A3C RL algorithm [45] to predict the optimal VoD bitrate that fits the instantaneous network condition. Indigo [66] proposes an IL algorithm to improve VoD QoE. The most recent works [17, 40, 65] move forward to adapt and evaluate ML-based video streaming algorithms in real-world environment. In particular, ABRL [40] makes customized designs based on Pensieve algorithm to accommodate the unique challenges when running it on Facebook VoD systems. QFLow [17] introduces a learning based network reconfiguration framework to achieve high QoE for YouTube streaming. Puffer [65] builds and operates a VoD open platform, and trains a supervised learning model "in situ", *i.e.*, using the traces collected from the real deployment.

Compared with these systems, OnRL differs in two major aspects: *(i)* Essentially, the ML models in these works are still trained offline (though using real-world traces), and thus detour unique challenges arising from online training, *e.g.*, parallel learning and robust learning. *(ii)* They focus on VoD applications, which differ from the interactive video telephony that imposes much harsher requirements atop today's best-effort Internet service [69]. In essence, VoD and video telephony applications are quite different for two reasons: *(i)* The VoD clients commonly maintain a playback buffer of dozens of seconds [15], so they are insensitive to short term (*e.g.*, sub-second level) network dynamics. In contrast, video telephony is more sensitive to instantaneous network traffic variation, which is hard to reproduce in simulation. *(ii)* Most information of a VoD session is known in advance,*e.g.*, the size of video chunks and buffers. In contrast, the video telephony content is always generated and consumed instantaneously. The algorithms need to responsively react to video dynamics under very tough low-latency intervals.

Online learning. Classical online learning advocates training with data streams on the fly [51]. The salient feature is that it keeps refining its prediction model constantly in the presence of a new

environment [47], in contrast to using a stationary post-trained model. Much research effort has been made on theoretical aspects of online learning in the AI community, ranging from the earliest online gradient update [18, 19, 27, 56] to the recent meta-learning [49]. Online learning has rarely been explored for practical network optimization. PCC [25] and PCC-Vivace [26] recently propose congestion control protocols sharing an online learning flavor. They leverage probing packets to continuously optimize a network utility objective. Park [41] proposes an open platform to facilitate experimenting with online RL for solving computer system problems, beyond RL's conventional application domains (*e.g.*, gaming [46]). In this work, OnRL adapts online learning to address the unique challenges in real-time video telephony.

In OnRL, we also propose a hybrid learning mechanism, which has shown its capability of preventing the catastrophic effect caused by RL's inherent but risky trial-and-error exploration. The hybrid learning component is inspired by the concept of safe learning, which has been adapted in other RL-based domains. For example, Shielding [16] detours RL's certain actions when a safety condition is triggered. A more recent work [43] proposes a guaranteed-safe policy to dynamically solve load balancing problem. To our knowledge, OnRL is the first framework to customize hybrid learning to improve the robustness of real-time video telephony.

Federated Learning is a distributed machine learning architecture, in which data is stored on distributed local devices instead of central servers [44]. Generally, federated learning aims to aggregate multiple users' experiences while preserving privacy, and also to reduce communication overhead [38, 67]. The learning aggregation component of OnRL is inspired by the concept of federated learning. But OnRL goes much further to solve practical problems in optimizing video telephony, including designing the specific RL neural network suitable with video telephony, enforcing RL's action despite video traffic dynamics and enhancing video robustness by designing the hybrid learning mechanism.

8 DISCUSSION

On-device online learning. Our current deployment of OnRL adopts a cloud-assisted architecture. The key design modules are located in remote cloud servers instead of the mobile devices, to accommodate the lack of mobile platform for RL training. Note that the RL cloud-server is unlikely to cause any privacy issue, as it only collects transport layer performance statistics, *i.e.*, loss rate, RTT, without any personal information. In addition, it is also noteworthy that the mainstream Tensorflow-Lite [12] and Core-ML [10] only allow executing pre-trained RL models, *i.e.*, they support the inference phase but not the training phase. The only RL-training-support platform in a recent work [57], as far as we know, is developed in Java and hard to be integrated with Taobao-Live. However, as neural processing units (NPUs) rapidly become available on mobile devices, neural network training on mobile devices will be feasible. On-device training will eliminate the overhead and cost of deploying RL servers, which shall facilitate the wide use of online learning algorithms like OnRL.

Adaptive learning aggregation. In OnRL, we have verified the significance of learning aggregation, by designing the weighted linear aggregation algorithms. Besides, we have tried more fine-grained aggregation methods, *i.e.*, aggregating a separate model for a group of users belonging to a certain network type (*e.g.*, WiFi or 4G) or ISP, but observed little gain. The results hint that the available bandwidth dynamics (particularly the short-term bandwidth variation that is important for OnRL) of an Internet path may not have strong correlation to its network type or ISP. On the other hand, we conjecture that other fine-grained aggregation methods using content-related attributes, such as grouping users by telephony scenarios (indoor, outdoor walking, or driving), may further improve video QoE. These approaches involve non-trivial design (*e.g.*, addressing user privacy), and is left for the next-phase of OnRL.

Closer integration of RL and rule-based algorithms. The evaluation in Sec. 6.3 demonstrates that RL exhibits significant advantages in handling network dynamics, but sometimes underperforms rule-based protocols under stable network conditions. It is possible to integrate OnRL and the rule-based algorithms according to the level of network dynamics, just as done in the hybrid learning scheme. We leave this for future work.

OnRL's scalability. We note that OnRL does not aim to improve reinforcement learning in general, but focuses on addressing the system-level challenges when adapting learning algorithms to optimize video telephony transport. The novelty of OnRL lies in the two-stage learning framework (including the neural network architecture and training methodology), the RL action enforcement and robust hybrid learning mechanisms, as well as in implementing, deploying and evaluating online RL on a mainstream operational video telephony system. We believe these customized mechanisms of OnRL may benefit future work on optimizing video transport and even other computer system design, such as resource scheduling and load balancing in practical networks. To extend the usage of OnRL to other applications, one can reuse the architecture and mechanisms, but need to re-train RL models using the applications' own data.

9 CONCLUSION

In this work, we designed an online reinforcement learning based real-time video telephony system named OnRL. We solve three unique challenges: learning from concurrent telephony sessions; enforcing RL actions despite of inherent video traffic variations; avoiding the QoE damage from reckless exploitation of RL. The real-world evaluation on a mainstream operational video telephony system demonstrates that OnRL outperforms the state-of-the-art solutions. We believe OnRL hints on a new direction that embraces online learning into more video communication applications, such as VoD, 360 panoramic video, virtual reality, *etc.*

ACKNOWLEDGMENTS

We appreciate the insightful feedback from the anonymous reviewers and our shepherd who helped improve this work. Anfu Zhou and Huadong Ma are the corresponding authors. This project was supported by the Innovation Research Group Project of NSFC (61921003), NSFC (61772084, 61720106007, 61832010), the 111 Project (B18008), the Fundamental Research Funds for the Central Universities (2019XD-A13) and Alibaba Innovation Research Program.

REFERENCES

[1] 2017. Closing the Simulation-to-Reality Gap for Deep Robotic Learning. https://ai.googleblog.com/2017/10/closing-simulation-to-reality-gap-for.html.

[2] 2017. Video Quality of Service (QOS) Tutorial . https://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-video/212134-Video-Quality-of-Service-QOS-Tutorial.html.

[3] 2018. WebRTC Homepage. https://webrtc.org/.

[4] 2019. Alibaba Cloud Overview. https://www.assistanz.com/alibaba-cloud-overview/.

[5] 2019. Cisco Visual Networking Index: Forecast and Trends. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html?dtid=osscdc000283.

[6] 2019. Taobao statistics, market share, trends, insights . https://www.chinainternetwatch.com/tag/taobao/.

[7] 2020. 30 amazing taobao statistics and facts (2020). By the numbers. https://expandedramblings.com/index.php/taobao-statistics/.

[8] 2020. Alibaba cloud, Server Load Balancer. https://www.alibabacloud.com/help/doc-detail/27544.htm?spm=a2c63.p38356.b99.5.4bc42299yeUcWm.

[9] 2020. Chromium in webrtc. https://chromium.googlesource.com/external/webrtc/.

[10] 2020. Core ML framework. https://developer.apple.com/documentation/coreml.

[11] 2020. Linux Traffic Control. https://events.static.linuxfound.org/sites/events/files/slides/Linux_traffic_control.pdf.

[12] 2020. Tensorflow Lite . https://www.tensorflow.org/lite.

[13] 2020. Tensorflow source code. https://github.com/tensorflow/tensorflow/tree/master/tensorflow/tools..

[14] 2020. TFLearn: Deep learning library featuring a higher-level API for TensorFlow. http://tflearn.org/.

[15] Saamer Akhshabi, Sethumadhavan Narayanaswamy, Ali C. Begen, and Constantine Dovrolis. 2012. An experimental evaluation of rate-adaptive video players over HTTP. *Signal Process. Image Commun.* 27, 4 (2012), 271–287. https://doi.org/10.1016/j.image.2011.10.003

[16] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe Reinforcement Learning via Shielding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018.* 2669–2678. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17211

[17] Rajarshi Bhattacharyya, Archana Bura, Desik Rengarajan, Mason Rumuly, Srinivas Shakkottai, Dileep M. Kalathil, Ricky K. P. Mok, and Amogh Dhamdhere. 2019. QFlow: A Reinforcement Learning Approach to High QoE Video Streaming over Wireless Networks. In *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing, Mobihoc 2019, Catania, Italy, July 2-5, 2019.* 251–260. https://doi.org/10.1145/3323679.3326523

[18] Léon Bottou and Yann LeCun. 2003. Large Scale Online Learning. In *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada].* 217–224.

[19] Haitham Bou-Ammar, Eric Eaton, Paul Ruvolo, and Matthew E. Taylor. 2014. Online Multi-Task Learning for Policy Gradient Methods. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014.* 1206–1214.

[20] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, Sergey Levine, and Vincent Vanhoucke. 2018. Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018.* 4243–4250.

[21] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. BBR: Congestion-Based Congestion Control. *ACM Queue* 14, 5 (2016), 20–53.

[22] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2017. Congestion Control for Web Real-Time Communication. *IEEE/ACM Trans. Netw.* 25, 5 (2017), 2629–2642.

[23] Jason Clements, Teodros Gessesse, Darshan Sedani, and Jerry Klein. 2016. Live video broadcasting mobile application for social sharing. US Patent App. 14/821,519.

[24] Yann Le Cun, Ido Kanter, and Sara A. Solla. [n.d.]. Eigenvalues of covariance matrices: Application to neural-network learning. *Physical Review Letters* 66, 18 ([n. d.]), 2396–2399.

[25] Mo Dong, Qingxi Li, Doron Zarchy, P. Brighten Godfrey, and Michael Schapira. 2015. PCC: Re-architecting Congestion Control for Consistent High Performance. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15).* USENIX Association, Oakland, CA, 395–408. https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/dong

[26] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. 2018. PCC Vivace: Online-Learning Congestion Control. In *15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9-11, 2018.* 343–356.

[27] John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.* 12 (2011), 2121–2159.

[28] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S. Wahby, and Keith Winstein. 2018. Salsify: Low-Latency Network Video through Tighter Integration between a Video Codec and a Transport Protocol. In *15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9-11, 2018.* 267–282.

[29] Ian J. Goodfellow and Oriol Vinyals. 2015. Qualitatively characterizing neural network optimization problems. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* http://arxiv.org/abs/1412.6544

[30] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. *Operating Systems Review* 42, 5 (2008), 64–74.

[31] Marie Hopkins. 2017. Live sports virtual reality broadcasts: Copyright and other protections. *Duke L. & Tech. Rev.* 16 (2017), 141.

[32] Quan Huynh-Thu and Mohammed Ghanbari. 2012. The accuracy of PSNR in predicting video quality for different video scenes and frame rates. *Telecommunication Systems* 49, 1 (2012), 35–48. https://doi.org/10.1007/s11235-010-9351-x

[33] Nathan Jay, Noga H. Rotman, Brighten Godfrey, Michael Schapira, and Aviv Tamar. 2019. A Deep Reinforcement Learning Perspective on Internet Congestion Control. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA.* 3050–3059.

[34] Junchen Jiang, Rajdeep Das, Ganesh Ananthanarayanan, Philip A. Chou, Venkat N. Padmanabhan, Vyas Sekar, Esbjorn Dominique, Marcin Goliszewski, Dalibor Kukoleca, Renat Vafin, and Hui Zhang. 2016. Via: Improving Internet Telephony Call Quality Using Predictive Relay Selection. In *Proceedings of the ACM SIGCOMM 2016 Conference, Florianopolis, Brazil, August 22-26, 2016.* 286–299.

[35] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming with FESTIVE. In *Proc. of International Conference on Emerging Networking Experiments and Technologies (CoNEXT).*

[36] Yueqiu Jiang, Junkun Zhang, and Qixue Guan. 2014. Improvement of TCP Reno Congestion Control Protocol. *Sensors & Transducers* 163, 1 (2014), 308.

[37] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* http://arxiv.org/abs/1412.6980

[38] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. 2018. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings.*

[39] Amanda Lee Kelly Lockwood, Ravi Gogna, Gary Linscott, Timothy Caldwell, Marin Kobilarov, Paul Orecchio, Dan Xie, Ashutosh Gajanan Rege, and Jesse Sol Levinson. 2019. Interactions between vehicle and teleoperations system. US Patent App. 15/644,310.

[40] Hongzi Mao, Shannon Chen, Drew Dimmery, Shaun Singh, Drew Blaisdell, Yuandong Tian, Mohammad Alizadeh, and Eytan Bakshy. 2019. Real-world Video Adaptation with Reinforcement Learning. In *Proceedings of the 2019 Reinforcement Learning for Real Life Workshop.*

[41] Hongzi Mao, Parimarjan Negi, Akshay Narayan, Hanrui Wang, Jiacheng Yang, Haonan Wang, Ryan Marcus, Ravichandra Addanki, Mehrdad Khani Shirkoohi, Songtao He, Vikram Nathan, Frank Cangialosi, Shaileshh Bojja Venkatakrishnan, Wei-Hung Weng, Song Han, Tim Kraska, and Mohammad Alizadeh. 2019. Park: An Open Platform for Learning-Augmented Computer Systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada.* 2490–2502. http://papers.nips.cc/paper/8519-park-an-open-platform-for-learning-augmented-computer-systems

[42] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2017, Los Angeles, CA, USA, August 21-25, 2017.* 197–210.

[43] Hongzi Mao, Malte Schwarzkopf, Hao He, and Mohammad Alizadeh. 2019. Towards Safe Online Reinforcement Learning in Computer Systems. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019).*

[44] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA.* 1273–1282. http://proceedings.mlr.press/v54/mcmahan17a.html

[45] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016.* 1928–1937.

[46] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis

Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013). arXiv:1312.5602 http://arxiv.org/abs/1312.5602

[47] Anusha Nagabandi, Chelsea Finn, and Sergey Levine. 2019. Deep Online Learning Via Meta-Learning: Continual Adaptation for Model-Based RL. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

[48] Matteo Pirotta, Marcello Restelli, and Luca Bascetta. 2013. Adaptive Step-Size for Policy Gradient Methods. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. 1394–1402. http://papers.nips.cc/paper/5186-adaptive-step-size-for-policy-gradient-methods

[49] Sachin Ravi and Hugo Larochelle. 2017. Optimization as a Model for Few-Shot Learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

[50] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*. 627–635.

[51] Doyen Sahoo, Quang Pham, Jing Lu, and Steven C. H. Hoi. 2018. Online Deep Learning: Learning Deep Neural Networks on the Fly. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*. 2660–2666.

[52] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017).

[53] Ivan Slivar, Mirko Suznjevic, and Lea Skorin-Kapov. 2018. Game Categorization for Deriving QoE-Driven Video Encoding Configuration Strategies for Cloud Gaming. *TOMM* 14, 3s (2018), 56:1–56:24.

[54] Taobao. 2020. Taobao Live APP. https://apps.apple.com/cn/app/E6B798E5AE9DE79BB4E692AD/id1448831879/.

[55] Taobao. 2020. Taobao-live monthly report, 2020 February. https://mp.weixin.qq.com/s/wNMfYAYcTI5T3O10x6hA2w/.

[56] Kyriakos G. Vamvoudakis and Frank L. Lewis. 2009. Online actor critic algorithm to solve the continuous-time infinite horizon optimal control problem. In *International Joint Conference on Neural Networks, IJCNN 2009, Atlanta, Georgia, USA, 14-19 June 2009*. 3180–3187. https://doi.org/10.1109/IJCNN.2009.5178586

[57] Cong Wang, Yanru Xiao, Xing Gao, Li Li, and Jun Wang. 2019. Close the Gap between Deep Learning and Mobile Intelligence by Incorporating Training in the Loop. In *Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, October 21-25, 2019*. 1419–1427. https://doi.org/10.1145/3343031.3350904

[58] Haiyang Wang, Tong Li, Ryan Shea, Xiaoqiang Ma, Feng Wang, Jiangchuan Liu, and Ke Xu. 2018. Toward Cloud-Based Distributed Interactive Applications: Measurement, Modeling, and Analysis. *IEEE/ACM Trans. Netw.* 26, 1 (2018), 3–16.

[59] Ziheng Wang, Isabella Reed, and Ann Majewicz Fey. 2018. Toward Intuitive Teleoperation in Surgery: Human-Centric Evaluation of Teleoperation Algorithms for Robotic Needle Steering. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. 1–8.

[60] Keith Winstein and Hari Balakrishnan. 2013. TCP ex machina: computer-generated congestion control. In *ACM SIGCOMM 2013 Conference, SIGCOMM'13, Hong Kong, China, August 12-16, 2013*. 123–134.

[61] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. 2013. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (nsdi'13)*.

[62] Xiufeng Xie, Xinyu Zhang, Swarun Kumar, and Li Erran Li. 2015. PiStream: Physical Layer Informed Adaptive Video Streaming over LTE. In *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*.

[63] Xiufeng Xie, Xinyu Zhang, and Shilin Zhu. 2017. Accelerating Mobile Web Loading Using Cellular Link Information. In *Proceedings of the Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*.

[64] Qiang Xu, Sanjeev Mehrotra, Zhuoqing Mao, and Jin Li. 2013. PROTEUS: Network Performance Forecast for Real-time, Interactive Mobile Applications. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '13)*. 347–360.

[65] Francis Y. Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. 2020. Learning in situ: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020, Santa Clara, CA, USA, February 25-27, 2020*. 495–511. https://www.usenix.org/conference/nsdi20/presentation/yan

[66] Francis Y. Yan, Jestin Ma, Greg D. Hill, Deepti Raghavan, Riad S. Wahby, Philip Levis, and Keith Winstein. 2018. Pantheon: the training ground for Internet congestion-control research. In *2018 USENIX Annual Technical Conference, USENIX ATC 2018, Boston, MA, USA, July 11-13, 2018*. 731–743.

[67] Xin Yao, Tianchi Huang, Chenglei Wu, Rui-Xiao Zhang, and Lifeng Sun. 2019. Federated Learning with Additional Mechanisms on Clients to Reduce Communication Costs. *CoRR* abs/1908.05891 (2019).

[68] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. 2015. Adaptive Congestion Control for Unpredictable Cellular Networks. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM '15)*. ACM, New York, NY, USA, 509–522. https://doi.org/10.1145/2785956.2787498

[69] Anfu Zhou, Huanhuan Zhang, Guangyuan Su, Leilei Wu, Ruoxuan Ma, Zhen Meng, Xinyu Zhang, Xiufeng Xie, Huadong Ma, and Xiaojiang Chen. 2019. Learning to Coordinate Video Codec with Transport Protocol for Mobile Video Telephony. In *The 25th Annual International Conference on Mobile Computing and Networking, MobiCom 2019, Los Cabos, Mexico, October 21-25, 2019*. 29:1–29:16.

Billion-Scale Federated Learning on Mobile Clients: A Submodel Design with Tunable Privacy

Chaoyue Niu, Fan Wu
Shanghai Jiao Tong University, China

Shaojie Tang
University of Texas at Dallas, USA

Lifeng Hua, Rongfei Jia, Chengfei Lv, Zhihua Wu
Alibaba Group, China

Guihai Chen
Shanghai Jiao Tong University, China

ABSTRACT

Federated learning was proposed with an intriguing vision of achieving collaborative machine learning among numerous clients without uploading their private data to a cloud server. However, the conventional framework requires each client to leverage the full model for learning, which can be prohibitively inefficient for large-scale learning tasks and resource-constrained mobile devices. Thus, we proposed a submodel framework, where clients download only the needed parts of the full model, namely, submodels, and then upload the submodel updates. Nevertheless, the "position" of a client's truly required submodel corresponds to its private data, while the disclosure of the true position to the cloud server during interactions inevitably breaks the tenet of federated learning. To integrate efficiency and privacy, we designed a secure federated submodel learning scheme coupled with a private set union protocol as a cornerstone. The secure scheme features the properties of randomized response, secure aggregation, and Bloom filter, and endows each client with customized plausible deniability (in terms of local differential privacy) against the position of its desired submodel, thereby protecting private data. We further instantiated the scheme with Alibaba's e-commerce recommendation, implemented a prototype system, and extensively evaluated over 30-day Taobao user data. Empirical results demonstrate the feasibility and scalability of the proposed scheme as well as its remarkable advantages over the conventional federated learning framework, from model accuracy and convergency, practical communication, computation, and storage overhead.

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing**; • **Computing methodologies** → **Machine learning**; • **Security and privacy** → **Distributed systems security**.

KEYWORDS

federated submodel learning, recommendation systems, local differential privacy, private set union, randomized response, secure aggregation, Bloom filter

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MobiCom '20, September 21–25, 2020, London, United Kingdom
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7085-1/20/09…$15.00
https://doi.org/10.1145/3372224.3419188

ACM Reference Format:
Chaoyue Niu, Fan Wu, Shaojie Tang, Lifeng Hua, Rongfei Jia, Chengfei Lv, Zhihua Wu, and Guihai Chen. 2020. Billion-Scale Federated Learning on Mobile Clients: A Submodel Design with Tunable Privacy. In *MobiCom 2020 (MobiCom '20), September 21–25, 2020, London, United Kingdom.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3372224.3419188

1 INTRODUCTION

1.1 Motivating Industrial Scenario

The industrial scenario in Alibaba that drives *federated submodel learning (FSL)* is the desire to provide accurate, customized, and real-time e-commerce recommendations for billion-scale mobile clients while keeping user data on local devices.

Currently, Alibaba's recommendation systems are cloud based (e.g., two-stage models with a matching stage [32, 49] for retrieving candidates and a ranking stage [14, 55, 56] for generating final recommendations, or unified models [57, 58]) and require the server cluster to collect, store, and process massive amounts of user data. Typical data fields involved in recommendations include user profile (e.g., user ID, gender, and age), user behavior (e.g., the list of visited goods IDs and relevant information, such as category IDs and shop IDs), and context (e.g., time, page number, and display position). More or less, these data fields are sensitive, and some clients who value privacy highly may refuse to share their data. In addition, according to the General Data Protection Regulation (GDPR), which took effect on May 25, 2018, any institution or company is prohibited from uploading user data (e.g., collected by apps on local devices) without the consent of European Union users [20]. Under such circumstances, refining the recommendation models and further providing accurate recommendations become practical demands.

Federated learning (FL), which decouples the ability to do machine learning from the need to upload and store data in the cloud, is a potential solution. However, the original framework of FL, proposed by Google in [34], requires each client to download the full model for training and then to upload the update of the full model, which is impractical for complex learning tasks and resource-constrained clients. Specifically, deep learning with a huge and sparse input space (e.g., e-commerce goods IDs, natural language texts, and locations) requires an embedding layer to transform inputs into a lower-dimensional space where similar inputs are close [11, 36]. In addition, the full embedding matrix tends to occupy a large proportion of the whole model parameters (e.g., 98.22% in our evaluated recommendation model and more than two-thirds in the language model of Google's Android keyboard, called Gboard [24]). Furthermore, as the largest online consumer-to-consumer platform in China, Taobao (owned by Alibaba) has roughly two billion goods

in total [49], which is far larger than the 10,000-word vocabulary in Gboard's natural language scenario [13, 24, 45]. This implies that the full embedding matrix of goods IDs in Taobao's deployed recommendation models roughly has two billion rows and occupies 134GB space when the embedding dimension is 18, and each element adopts 32-bit representation. If each client directly uses the full model for learning, it inevitably incurs unacceptable and unaffordable overhead for one billion Taobao users with smart devices. To improve efficiency, we observe that the input data of a certain client normally involve a small subspace of the full feature space. Thus, the client tends to need a tailored model rather than the full model, especially in the practical applications that require personalization, such as recommendation systems and multi-task learning. For example, if a Taobao user's historical data contain 300 goods, he/she requires only the corresponding 300 rows, rather than the entire two billion rows, of the full embedding matrix. Based on this key observation, we propose a general and scalable FSL framework.

1.2 Federated Submodel Learning Framework

In each communication round of FSL, a cloud server first selects some eligible clients to participate. Then, each chosen client downloads part of the global model as required, namely, a submodel, from the cloud server. For example, in the e-commerce recommendation, a client's submodel mainly contains the embedding vectors for the goods IDs in its historical data, as well as the parameters of the other network layers. Then, the client trains the submodel over its local data and uploads the submodel update. The cloud server aggregates the submodel updates from live clients and forms a consensus update to the global model. The process above is iterated continuously in practice so that the global model on the cloud server and the submodels on the clients are kept up-to-date.

If each client leverages the full model rather than its required submodel for learning, FSL will degenerate to conventional FL. FSL further decouples the ability to accomplish FL from the need to use the prohibitively large full model, dramatically improving efficiency. For example, in our evaluation, the size of a client's desired submodel is only 1.99% of the full model's size. The generalization of FSL also implies that almost all existing methods to improve the efficiency and scalability of FL can still apply to FSL. For example, the model compression algorithms not only can compress the full model (update) but also can compress the submodel (update) to reduce overhead. Therefore, FSL is more practical for distributing industrial-strength learning tasks on ubiquitous mobile devices.

1.3 Newly Introduced Privacy Risks

Just as every coin has two sides, FSL not only provides efficiency but also introduces two extra privacy risks.

First, compared with using the public full model in FL, the download of a submodel and the upload of the submodel update would require each client to provide an index set as auxiliary information, specifying the "position" of its submodel. However, the real index set normally corresponds to the client's private data. For example, to specify the required rows of the embedding matrix in the e-commerce scenario, a client mainly needs to provide the goods IDs in its user data as the real index set. Similarly, in the natural language scenario, a client's real index set to locate the desired word embedding vectors is actually the vocabulary extracted from the client's typed texts. Thus, the disclosure of a client's real index set to the cloud server can still be regarded as the leakage of the client's private data, breaking the tenet of FL.

Second, compared with the aligned full model in FL, each client chosen in one round of FSL submits only the update of its customized submodel, which tends to be highly differentiated from the other chosen clients' submodels. As a result, the aggregation of updates w.r.t. a certain index can come from a unique client out of the chosen ones (e.g., with probability 86.7% for 100 clients randomly chosen in our Taobao dataset). This implies that from the aggregate update, the cloud server not only can ascertain that the client has the index but also can learn its detailed update. Besides the real index revealing a client's private data, the client's individual update can still memorize or even allow reconstruction of its private data, namely, the model inversion attack [22, 59] on the individual client. Further, as the goods IDs of different Taobao users are more heterogeneous than the vocabularies of different Gboard users, the misalignment of desired submodels and the privacy risk in e-commerce are more severe than those in natural language.

1.4 Fundamental Problems and Challenges

To mitigate the privacy risks above, we need to jointly solve two fundamental problems abstracted from the download and upload phases of FSL: (1) How a client can download a row of a matrix, which represents the global/full model and is maintained by an untrusted cloud server, without revealing which row or the row index to the cloud server; and (2) how a client can modify a row of the matrix, still without revealing which row was modified and the altered content. We analyze two problems in detail.

We start with the first problem. One naïve method is that the client downloads the full matrix, as in conventional FL, and then extracts the required row locally. This method perfectly hides the fetched row index, but incurs immense communication overhead. To avoid downloading the full matrix, *private information retrieval (PIR)* [4, 12, 44] can be applied, which exactly matches the problem settings, including the read-only mode and the concealment of the retrieved elements. Therefore, if we consider the first problem independently, PIR may be a good choice.

We next deconstruct the second problem. For a concrete row of the full matrix, if clients modify the row one by one, the cloud server definitely knows the clients who modified this row and their detailed modifications. Thus, one feasible way is to first securely aggregate (i.e., add up) all the modifications without revealing any individual modification and then apply the aggregate modification (i.e., the sum of the modifications) to the row of the full matrix once. Such a primitive of oblivious addition can be provided by many cryptographic schemes (e.g., the protocol specific to the FL setting in [8] and additively homomorphic encryption [9, 43]). With the secure aggregation guarantee, if more than one client participates in aggregation and at least one of their modifications is nonzero, then the cloud server cannot know any individual modification and thus cannot reveal which client(s) truly intend to modify this row. Further, a larger number of involved clients imply better privacy. One extreme case is in conventional FL with secure aggregation, also called *secure federated learning (SFL)*, which harshly lets all

chosen clients in one round be involved for each row of the full matrix, no matter whether they truly intend to modify this row or not. Thus, SFL offers the best privacy but the worst efficiency. Another extreme case is directly combining FSL with secure aggregation, which simply lets those clients who really intend to modify a row be involved. Thus, each chosen client needs to be involved only for those rows that it truly intends to modify, implying the best efficiency. However, different clients tend to modify highly differentiated or even mutually exclusive rows. For joint modification w.r.t. some row, chances are high that only one client is involved. Under this circumstance, the secure aggregation guarantee no longer works, while the client's real intention and detailed modification are disclosed. In a nutshell, trivial solutions to the second problem cannot achieve a tunable balance between privacy and efficiency.

1.5 Solution Overview and Contributions

Considering the two fundamental problems above and several practical issues together, we propose *secure federated submodel learning (SFSL)*. We first determine the scope of joint modification in each round of FSL, thereby aligning differentiated submodels. The key obstacle is the need to conceal the position of any chosen client's desired submodel (alternatively, its real index set or private data). The baseline SFL trivially uses the prohibitively large full index set (i.e., the position of the full model) for alignment. In contrast, our SFSL identifies a necessary and sufficient scope for alignment, namely, the union of the chosen clients' real index sets. The union is efficiently obtained through our proposed *private set union (PSU)* protocol without revealing any individual real index set. In addition, because the union is (far in our application scenario) smaller than the full index set, SFSL can significantly outperform SFL without sacrificing privacy. Based on the union, each chosen client generates a randomized index set to replace and protect its real index set for interactions in the download and upload phases. Specifically, the randomized index set is generated by applying randomized response twice with one memoization step between, and the parameter setting in randomized response is customized by the client. Such a design with secure aggregation allows the client to hold self-controllable deniability against whether it really intends or does not intend to download some row and to upload the modification of this row, even if the client participates in multiple rounds. The strength of deniability is rigorously quantified using *local differential privacy (LDP)*. Meanwhile, the probability of the cloud server inferring the client's real intention from the aggregate modification is also thoroughly analyzed. Furthermore, the cardinality of the randomized index set dominates the client's overhead, and the intersection of the client's real and randomized index sets controls its local training. Therefore, each client can fine-tune the tension among privacy, efficiency, and effectiveness when using SFSL.

We summarize the key contributions of this work as follows:

- To the best of our knowledge, we are the first to propose the FSL framework for industrial-strength, on-device intelligence, and further to identify and remedy privacy risks.
- The proposed secure scheme SFSL empowers each client with finely tunable deniability against its real intention of downloading the desired submodel and uploading the submodel update, thereby protecting private data.
- As a moat, we designed an efficient and scalable PSU protocol using Bloom filter, secure aggregation, and randomization, which can have independent and significant value in practice.
- We instantiated SFSL with Alibaba's e-commerce scenario, adopted the *deep interest network (DIN)* for recommendation, and implemented a prototype system. We also conducted extensive evaluations using the data of Taobao users collected over a period of 30 days. When the number of chosen clients per round is 100, the major experimental results are presented as follows: (1) Compared with conventional FL, which diverges in the end, SFSL improves the highest *area under the curve (AUC)* by 7.22%; (2) at the same security and privacy levels as SFL, SFSL reduces 80.05% of the communication overhead on both sides of the client and the cloud server, and reduces 85.02% (resp., 45.43%) and 72.51% (resp., 63.77%) of the computation (resp., memory) overhead on the sides of the client and the cloud server, respectively; and (3) for the proposed PSU protocol, the communication overhead per client is less than 1MB. The computation overhead of the client and the cloud server is each less than 40s, even if the dropout ratio of the chosen clients reaches 20%.

2 RELATED WORK

In recent few years, FL has become an active topic in academic and industrial fields. In this section, we focus on the related work about security and privacy. For other focuses, we direct interested readers to comprehensive surveys [26, 31, 54].

Bonawitz et al. [8] designed a secure aggregation protocol in both honest-but-curious and actively adversarial settings, which will be used as a building block of our SFSL. In particular, their design was specific to the FL framework, where each resource-constrained client cannot establish direct communication channels with other clients and needs to rely on an untrusted cloud server as a relay, while part of clients may drop out during the aggregation process. To bound the leakage of each individual client's training data from the model updates, several differentially private mechanisms were proposed. McMahan et al. [35] considered recurrent language models, let the trusted cloud server perturb the aggregate model update in a single round with a Gaussian mechanism, and relied on the celebrated moments account scheme [1] for multi-round privacy composition. Agarwal et al. [2] studied how to integrate *differential privacy (DP)* with model compression and proposed a Binomial mechanism for each client to perturb its quantized model update. The quantization [30, 48, 53] is to encode original float-type and continuous parameters into integer-type and discrete values with a few bits. In contrast to these defense mechanisms, Bagdasaryan et al. [6] developed a model replacement attack launched by malicious clients to backdoor the global model on the cloud server. Nasr et al. [40] considered the membership inference attack in pure FL without secure aggregation and leveraged a client's individual model update to uncover the membership of its training data.

Parallel to existing work in FL, where each client uses the same global model for learning, we propose a novel FSL framework for scalability. Under FSL, we focus on new privacy issues due to the dependence between the position of a client's desired submodel and its private data, as well as the misalignment of the submodel updates from heterogeneous clients in secure aggregation.

3 PRELIMINARIES

In this section, we define security requirements corresponding to the privacy risks introduced in Section 1.3. We also review a primitive building block, called randomized response.

We first introduce some frequently used notations. We use a two-dimensional matrix with m rows and d columns to represent the global/full model, denoted as $\mathbf{W}$. We also let $\mathcal{S} = \{1, 2, \ldots, m\}$ denote the full (row) index set of $\mathbf{W}$. In addition, we let C denote n clients who are selected by the cloud server to participate in any round of FSL. For a chosen client $i \in C$, we let $\mathcal{S}^{(i)} \subset \mathcal{S}$ denote its real index set and let $\mathbf{W}_{\mathcal{S}^{(i)}}$ denote its desired submodel, which imply that the user data of client i involve (intuitively, after local training may update) the rows in $\mathbf{W}$ with indices $\mathcal{S}^{(i)}$.

3.1 Security Requirements

First, for the disclosure of real index sets when clients interact with the untrusted cloud server, we consider that each client should have plausible deniability of whether a certain index is or is not in its real index set. To measure the strength of plausible deniability, we adopt LDP, which is a variant of standard DP in the local setting. Specifically, the randomization in LDP is performed by clients in a distributed manner, rather than relying on a data curator as a trusted authority to conduct centralized perturbation in DP. Thus, the privacy of an individual client's data is protected not only from external attackers but also from the untrusted data curator (e.g., the cloud server in FSL). LDP for various population statistics has received industrial deployments (e.g., by Google [19, 21], Apple [5], and Microsoft [16]) as well as extensive academic attention [18, 27, 50, 51]. We present the formal definition of LDP as follows:

Definition 3.1. A randomized mechanism M satisfies ϵ-LDP, if for any pair of inputs from a client, denoted as x and y, and for any possible output of M, denoted as z, we have $\frac{\Pr(M(x)=z)}{\Pr(M(y)=z)} \leq \exp(\epsilon)$. A smaller privacy budget ϵ offers a better LDP guarantee.

Intuitively, LDP says that the output distribution of the randomized mechanism does not change too much, given distinct inputs from the client. Thus, LDP formalizes a sort of plausible deniability: No matter what output is revealed, it is approximately equally as likely to have come from one input as any other input. When LDP applies to obscure the membership of a certain index in FSL, the inputs and outputs are boolean values, where possible inputs (resp., outputs) are two states: a certain index "in" or "not in" a client's real (resp., revealed) index set. We can check that FL provides the strongest deniability, where the level of LDP is $\epsilon = \ln(1/1) = 0$ for each client. The reason is that no matter whether an index is or is not in a client's real index set (different inputs), this index will definitely be revealed (the same output "in"). In contrast, FSL provides the weakest deniability, where the level of LDP is $\epsilon = \ln(1/0) = \infty$ for each client, because if an index is in (resp., not in) a client's real index set, this index will definitely (resp., definitely not) be revealed, namely, output "in" with probability 1 (resp., 0).

Second, for the secure aggregation of misaligned submodel updates in any round, we consider that the clients chosen in the round should be enabled to tune the privacy level, rather than trivially choosing extreme SFL or FSL with secure aggregation. We define a client-tunable privacy protection mechanism as follows:

Definition 3.2. A privacy protection mechanism for aggregating submodel updates is client tunable, if participating clients can control the probabilities of the following two events from the securely aggregated submodel update.

- *Event 1:* The cloud server ascertains that an index belongs to some client and learns its detailed update w.r.t. this index.
- *Event 2:* The cloud server ascertains that an index does not belong to some client.

If we view the disclosure of a client's real intention w.r.t. an index (i.e., the occurrence of either Event 1 or Event 2) as a breach of deniability about the index, then Definition 3.2 complements Definition 3.1 for secure aggregation of submodel updates. If we view the disclosure of a client's individual submodel update as the premise of the model inversion attack (resp., the membership inference attack) to reveal the content (resp., the membership) of the client's private data, Definition 3.2 can also be interpreted from mitigating the attack on each individual client. We first examine SFL. If at least two clients participate, the probability of Event 1 is 0, and the probability of Event 2 is still 0 for those indices in the union of the chosen clients' real index sets. For an index outside the union, the probability of Event 2 approaches 1. Although each chosen client submits a zero vector w.r.t. this index, from the aggregate zero vector, the cloud server almost ascertains that all the chosen clients do not have the index, despite of some rare cases. Regarding directly combining FSL with secure aggregation, the probability of Event 1 depends entirely on the heterogeneity of the user data on the clients, and the probability of Event 2 is the same as that in SFL.

3.2 Randomized Response

Randomized response, due to Warner in 1965 [52], is a survey technique in the social sciences for collecting statistics about illegal, embarrassing, or sensitive topics, where the respondents want to preserve the privacy of their answers. A classic example for illustration is the "Are you a member of the Communist Party?" question. For this question, each respondent flips a fair coin in secret and tells the truth if it comes up tails; otherwise, he/she flips a second coin and responds "Yes" if heads and "No" if tails. Thus, a communist (resp., non-communist) will answer "Yes" with probability 75% (resp., 25%) and "No" with probability 25% (resp., 75%).

Randomized response can provide plausible deniability for both "Yes" and "No" answers. A communist can contribute his/her response of "Yes" to the event that the first and second coin flips were both heads, which occurs with probability 25%. Meanwhile, a non-communist can also contribute his/her response of "No" to the event that the first coin was heads and the second coin was tails, which still occurs with probability 25%. Furthermore, the plausible deniability of a one-time randomized response can be rigorously quantified by LDP, particularly at the level $\epsilon = \ln(75\%/25\%) = \ln 3$, irrespective of any attacker's prior knowledge [17, 19].

4 DESIGN OF SFSL

4.1 Design Rationale

We illustrate key design principles mainly through demonstrating how to handle the two fundamental problems raised in Section 1.4 and how to resolve several practical issues.

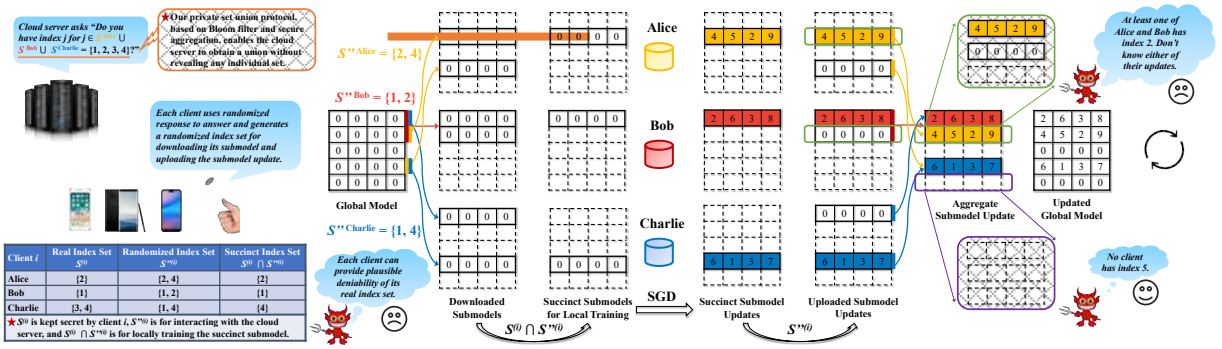


Figure 1: An illustration of SFSL. The gray rounded rectangle denotes secure aggregation. The randomized index sets of Alice, Bob, and Charlie are $\{2, 4\}$, $\{1, 2\}$, and $\{1, 4\}$, respectively. For index 2, Alice submits the update $(4, 5, 2, 9)$, Bob pretends to submit a zero vector, while Charlie does not submit. Secure aggregation ensures that the adversarial cloud server obtains only the sum of Alice's and Bob's updates $(4, 5, 2, 9)$, but does not know either update. Thus, the cloud server can only infer that at least one of Alice and Bob has index 2. Due to plausible deniability, the cloud server also cannot ascertain the state of Charlie.

As shown in Figure 1, we handle two fundamental problems in a unified manner rather than in separate ways. During the download and upload phases, a client consistently uses a randomized index set in place of its real index set. In contrast, during the local training phase, the client leverages the intersection of its real and randomized index sets (called the client's succinct index set) to prepare the succinct submodel and the user data involved. With the blind of the randomized index set to interact with the outside world, the client holds plausible deniability of some index being in or not in its real index set. Specifically, the client generates its randomized index set locally with randomized response as follows. The sensitive question asked by the cloud server is "Do you have a certain index?". Then, the client answers "Yes" with two customized probabilities, conditional on whether the index is or is not in the client's real index set, and puts the index into the randomized index set if it receives a "Yes" answer. The two probabilities allow the client to fine-tune the balance between deniability and utility.

We further examine the feasibility of index set randomization in handling two problems. For the first problem in the download phase, if a client intends (resp., does not intend) to download a certain row, and it actually downloads (resp., does not download), it can blame its action to randomization, i.e., the occurrence of the event that the index not in (resp., in) a client's real index set returns a "Yes" (resp., "No") answer. Regarding the second problem in the upload phase, the usage of the randomized index set still empowers a client to deny its real intention of modifying or not modifying some row, even if the cloud server observes its binary action of modifying or not modifying. Additionally, for a concrete row, there are two different groups of clients involved in the joint modification: (1) One group consists of those clients who intend to modify the row and contribute nonzero modifications; and (2) the other group comprises those clients who do not intend to modify the row and pretend to modify by submitting zero modifications. With the secure aggregation guarantee, even though the adversarial cloud server observes the aggregate modification, it is hard to identify any individual modification and further to infer whether some client originally intends to perform a modification or not. The hardness is controlled by the sizes of two groups, alternatively, the probabilities of an index in and not in the real index set returning a "Yes" answer, which are fully tunable by clients as expected in Definition 3.2.

Besides two fundamental problems, there still exist two practical issues to be solved before the method of index set randomization can apply to FSL. The first issue regards whether it is practical and necessary for the cloud server to ask "Do you have a certain index?" for each index in the full index set, which is intrinsically adopted by FL. For example, the matrix, representing Taobao's full recommendation model, has billions of rows. Thus, it is impractical for a client to answer billion-scale questions and further to download and securely upload those rows with "Yes" answers. We turn to narrowing down the scope to the union of n chosen clients' real index sets, which is normally far smaller than the full index set (i.e., the union of the whole clients' real index sets). Our optimization is inspired by an observation: If a client's real index set is of size 300, and the full index set is of size 2 billion, using the probability parameters in the survey of party membership, the expected number of "Yes" answers is $300 \times 75\% + (2 \times 10^9 - 300) \times 25\% \approx 5 \times 10^8$. Such a calculation implies that the dominant "Yes" answers are those with "No" in reality but "Yes" due to randomness. Nevertheless, most of the "No"-to-"Yes" answers are useless. Specifically, for those indices that do not belong to any chosen client's real index set (e.g., index 5 in Figure 1), although part (25% in expectation) of the chosen clients upload zero vectors for randomization, the cloud server can still infer from the aggregate zero vectors that these clients do not actually have the indices. Therefore, it is unnecessary to cover any index outside the union of n chosen clients' real index sets. By using the union, we set $n = 100$ and recalculate the expected number of "Yes" answers as $300 \times 75\% + (100 \times 300 - 300) \times 25\% \approx 8 \times 10^3$. This implies that a prohibitively large number of unnecessary "No"-to-"Yes" answers are avoided, significantly improving efficiency and without degrading privacy. Now, a basic and thorny problem that accompanies is how multiple clients can obtain the union of their real index sets under the mediation of an untrusted cloud server without revealing any client's real index set, namely, the need of a PSU protocol. Considering existing schemes cannot satisfy the atypical setting of FSL and high scalability, we propose a novel design of PSU based on Bloom filter and secure aggregation.

The second issue regards the longitudinal privacy when a client is chosen to participate in multiple rounds of FSL. The initial version of randomized response provides a rigorous privacy guarantee only when an audience answers the same question once, facilitating a one-time response to "Do you have a certain index?". Thus, we need to extend the initial version to allow repeated responses from the same client to those already answered indices. Our extension leverages key principles from *randomized aggregatable privacy-preserving ordinal response (RAPPOR)* [19, 21] and plays randomized response twice with a memoization step between. In particular, the noisy answers generated by the inner (permanent) randomized response will be memoized and permanently replace the real answers in the outer (instantaneous) randomized response. This ensures that even though a client responds to the membership of a concrete index for infinite times, the client can still hold plausible deniability of its real answer, where the level of deniability is lower bounded by the privacy level of the memoized noisy answer.

4.2 Design Details

We introduce the design details of SFSL in a top-down manner. We first give an overview of its top-level architecture and then show two underlying modules, namely, index set randomization and PSU.

4.2.1 Secure Federated Submodel Learning (SFSL). Before getting into SFSL, we first review federated averaging [34], which is the default distributed optimization algorithm of FL. In each round, the cloud server restarts a collaborative training process and randomly chooses some eligible clients to locally train the latest global model. Then, the cloud server takes a weighted average of the full model updates, one chosen client's weight being proportional to the size of its training set, and adds the aggregate update to the global model.

We now present SFSL in Algorithm 1, which generalizes federated averaging to support effective and efficient FSL. SFSL also preserves desired security and privacy properties while incorporating the unstable and limited network connections of mobile devices. At the initial stage, the cloud server randomly initializes the global model (Line 1). In each communication round, the cloud server selects n clients to participate (Lines 2 and 3) and also maintains an up-to-date set of the chosen clients who are alive throughout the whole round, denoted by $\hat{C}$. A chosen client determines its real index set based on its local data, which specifies the "position" of its truly required submodel (Line 10). For example, if the goods IDs of a Taobao user include $\{1, 2, 4\}$, then he/she requires the first, second, and fourth rows of the embedding matrix for goods IDs, further implying that the real index set should contain $\{1, 2, 4\}$. Then, the cloud server launches PSU to obtain the union of the chosen clients' real index sets while keeping each individual client's real index set in secret (Lines 4 and 11). The union is delivered to live clients for them to generate randomized index sets with customized LDP guarantees against the cloud server (Line 12). Each live client will use its randomized index set, rather than its real index set, to download the submodel and to securely upload the submodel update (Lines 13 and 19). Upon receiving the randomized index set from a client, the cloud server stores it for later usage and returns the corresponding submodel and training hyperparameters to the client (Line 6).

Depending on the intersection of the real and randomized index sets (i.e., the succinct index set), the client extracts a succinct submodel and prepares involved data as the succinct training set (Line 14). For example, if a Taobao user's real index set is $\{1, 2, 4\}$, and his/her randomized index set is $\{2, 4, 6, 9\}$, he/she receives a submodel with the row indices $\{2, 4, 6, 9\}$ from the cloud server, but just needs to train the succinct submodel with the row indices $\{2, 4\}$ over his/her local data involving the goods IDs $\{2, 4\}$. After training under the preset hyperparameters, the client obtains the update of the succinct submodel (Line 15). Then, it prepares the submodel update to be uploaded with the randomized index set by adding the update of the succinct submodel to the rows with the succinct indices and padding zero vectors to the other rows (Line 16). To help the cloud server average multiple submodel updates according to the size of relevant local training data, each chosen client also needs to count the number of its training samples involving every

Algorithm 1: Secure Federated Submodel Learning (SFSL)

```
/* Cloud server's process */
1  Initialize the global model W;
2  foreach communication round do
3      Randomly select n clients, denoted as C;
4      Launch private set union (Algorithm 3), obtain the union of n
         clients' real index sets, namely, ⋃_{i∈C} S^(i), and deliver the
         union result to the up-to-date set of the clients who are alive,
         denoted as Ĉ ⊂ C;
5      foreach client i ∈ Ĉ do
6          Receive and store the randomized index set S''^(i) from
             client i, and return the submodel W_{S''^(i)} and training
             hyperparameters to i;
7      foreach j ∈ ⋃_{i∈C} S^(i) do
8          Determine the live clients involving index j, denoted as
             Ĉ_j = {i | i ∈ Ĉ ∧ j ∈ S''^(i)}, let them submit materials
             for secure aggregation, and obtain the sum of (weighted)
             updates Σ_{i∈Ĉ_j} Δw_j^(i) and the total count number of
             relevant samples Σ_{i∈Ĉ_j} v_j^(i);
9          Update the j-th row of the global model W by adding
             Σ_{i∈Ĉ_j} Δw_j^(i) / Σ_{i∈Ĉ_j} v_j^(i);
/* Client i's process */
10 Determine its real index set S^(i) based on local data;
11 Participate in private set union (Algorithm 3);
12 Generate a randomized index set S''^(i) (Algorithm 2);
13 Use S''^(i) to download a submodel, denoted as W_{S''^(i)};
14 Depending on the succinct index set S''^(i) ∩ S^(i), locally extract
     the succinct submodel W_{S''^(i) ∩ S^(i)} from W_{S''^(i)} and prepare
     involved data as the succinct training set;
15 Train W_{S''^(i) ∩ S^(i)} using the hyperparameters and obtain the
     update of the succinct submodel ΔW_{S''^(i) ∩ S^(i)};
16 Initialize the submodel update to be uploaded with S''^(i), denoted
     as ΔW_{S''^(i)}, all to 0, and add ΔW_{S''^(i) ∩ S^(i)};
17 Count the number of training samples involving each index
     j ∈ S''^(i) and store the results in the vector v_{S''^(i)};
18 Weight ΔW_{S''^(i)} in advance by multiplying each row with the
     corresponding count number in v_{S''^(i)};
19 Upload materials for securely aggregating ΔW_{S''^(i)}, v_{S''^(i)}.
```

index in the randomized index set (Line 17). In particular, the numbers of samples involving the indices outside the succinct index set are all zeros. Furthermore, each client weights the submodel update to be uploaded in advance by multiplying each row with the corresponding count number, namely, the weight (Line 18).

The weighted submodel updates and the count vectors from live clients are securely aggregated side by side under the coordination of the cloud server (Lines 7–9 and 19). Specifically, the cloud server guides the secure aggregation by enumerating every index in the union of real index sets as follows. It first determines the set of live clients whose randomized index sets contain this index and then lets these clients submit the materials for securely adding up the weighted updates and the count numbers w.r.t. the index (Line 8). The cloud server finally applies an aggregate update to the global model in this row by adding the quotient between the sum of the weighted updates and the total count number, namely, the weighted average (Line 9). To reduce the interactions between the cloud server and any client, they can package all the materials supporting secure aggregation, rather than exchanging for one index each time (Lines 7–9). In particular, the cloud server executes Lines 7 and 8 for each live client $i \in \hat{C}$ in parallel and then executes Line 9 for each index in the union $j \in \bigcup_{i \in C} \mathcal{S}^{(i)}$.

4.2.2 Index Set Randomization. We next show how a client can generate a randomized index set in each participating round.

We start with the basic design. The sensitive question, asked by the cloud server, is "Do you have a certain index?". All the clients chosen in one round of FSL make up the population to be surveyed, while the union of their real index sets works as the necessary and sufficient scope of the questionnaire. Given the questionnaire, client i leverages randomized response to answer "Yes" or "No" and puts the indices with "Yes" answers into its randomized index set. To fine-tune the tension among effectiveness, efficiency, and privacy, client i basically adjusts two probability parameters $p_1^{(i)}, p_2^{(i)}$ in randomized response. In particular, $p_1^{(i)}$ denotes the probability that an index in client i's real index set will return a "Yes" answer and controls the factual size of client i's data contributed to FSL. A larger $p_1^{(i)}$ generally implies better effectiveness from convergence rate. In addition, $p_2^{(i)}$ denotes the probability that an index outside client i's real index set will return a "Yes" answer and determines the number of redundant rows to be downloaded and the number of padded zero vectors to be uploaded through secure aggregation. Hence, given a fixed $p_1^{(i)}$, a smaller $p_2^{(i)}$ indicates better efficiency. Furthermore, $p_1^{(i)}, p_2^{(i)}$ jointly adjust the level of LDP, where a pair of closer $p_1^{(i)}, p_2^{(i)}$ provide better LDP. We examine three typical instances: (1) The party membership survey takes $p_1^{(i)} = 75\%, p_2^{(i)} = 25\%$ for each respondent; (2) FL essentially uses the full index set as the scope of the cloud server's questionnaire, takes $p_1^{(i)} = p_2^{(i)} = 1$ for each client, and offers the best LDP and effectiveness but the worst efficiency; and (3) FSL adopts $p_1^{(i)} = 1, p_2^{(i)} = 0$ for each client and provides the best effectiveness and efficiency but the worst LDP.

We further extend the basic design to allow multi-round participation of client i and its repeated responses to any index. As shown in Algorithm 2, we let client i maintain two index sets with "Yes" and "No" answers in the permanent randomized response,

Algorithm 2: Client i's Index Set Randomization

Input: Client i's real index set $\mathcal{S}^{(i)}$, Union of the chosen clients' real index sets $\bigcup_{i \in C} \mathcal{S}^{(i)}$, Client i's memoized index set $\mathcal{Y}^{(i)}$ (resp., $\mathcal{N}^{(i)}$) initialized to $\emptyset$ at very beginning with permanent "Yes" (resp., "No") answers to the question "Do you have a certain index?", Client i's customized probability parameters $0 \le p_1^{(i)}, p_2^{(i)}, p_3^{(i)}, p_4^{(i)} \le 1$.

Output: Client i's doubly randomized index set $\mathcal{S}''^{(i)}$

1 $\mathcal{S}'^{(i)} = \emptyset, \mathcal{S}''^{(i)} = \emptyset$;

// Permanent randomized response

2 **foreach** $j \in \bigcup_{i \in C} \mathcal{S}^{(i)} \wedge j \notin \mathcal{Y}^{(i)} \cup \mathcal{N}^{(i)}$ **do**

3 **if** $j \in \mathcal{S}^{(i)}$ **then**

4 Add j to $\mathcal{S}'^{(i)}$ with probability $p_1^{(i)}$;

5 **else**

6 Add j to $\mathcal{S}'^{(i)}$ with probability $p_2^{(i)}$;

// Memoization of permanent answers

7 **if** $j \in \mathcal{S}'^{(i)}$ **then**

8 $\mathcal{Y}^{(i)} = \mathcal{Y}^{(i)} \cup j$;

9 **else**

10 $\mathcal{N}^{(i)} = \mathcal{N}^{(i)} \cup j$;

// Instantaneous randomized response

11 **foreach** $j \in \bigcup_{i \in C} \mathcal{S}^{(i)}$ **do**

12 **if** $j \in \mathcal{Y}^{(i)}$ **then**

13 Add j to $\mathcal{S}''^{(i)}$ with probability $p_3^{(i)}$;

14 **else**

15 Add j to $\mathcal{S}''^{(i)}$ with probability $p_4^{(i)}$;

16 **return** $\mathcal{S}''^{(i)}$

respectively (Input). The permanent randomized response is parameterized by $p_1^{(i)}, p_2^{(i)}$ as illustrated above. Given that a client can be grouped with distinct clients in different rounds while the union of the chosen clients' real index sets varies from one round to another, the client needs to handle new indices. For a certain round, as any new index comes (Line 2), client i first generates a permanent noisy answer for it, conditional on whether the new index is in or is not in the real index set (Lines 3–6). Then, client i updates two memoized index sets (Lines 7–10). Based on the memoized noisy answers to the union of real index sets, client i obtains its final randomized index set by performing an instantaneous randomized response over the union (Lines 11–16). The instantaneous randomized response is parameterized with another two probabilities $p_3^{(i)}, p_4^{(i)}$, similar to $p_1^{(i)}, p_2^{(i)}$ in the permanent randomized response. Now, these four probability parameters jointly support tuning the tension among privacy, effectiveness, and efficiency. In particular, $p_5^{(i)} = p_1^{(i)}(p_3^{(i)} - p_4^{(i)}) + p_4^{(i)}$ (resp., $p_6^{(i)} = p_2^{(i)}(p_3^{(i)} - p_4^{(i)}) + p_4^{(i)}$), denoting the probability that an index in (resp., not in) client i's real index set finally returns a "Yes" answer, now plays the same role as $p_1^{(i)}$ (resp., $p_2^{(i)}$) in the basic design. Detailed derivations of $p_5^{(i)}, p_6^{(i)}$ can be found in the complete manuscript [42].

4.2.3 Private Set Union (PSU). We introduce the last PSU module. We first briefly review existing PSU protocols and illustrate their practical infeasibility in FSL. We then present our new design.

Algorithm 3: Private Set Union (PSU)

1 Cloud server partitions the full index set $\mathcal{S}$;
2 **foreach** client $i \in C$ **do**
3 Represent its real index set $\mathcal{S}^{(i)}$ as a Bloom filter $\mathbf{b}^{(i)}$;
4 Randomize $\mathbf{b}^{(i)}$ to an integer vector $\mathbf{b}'^{(i)}$ by replacing each bit 1 in $\mathbf{b}^{(i)}$ with a random integer from $\mathbb{Z}_R$;
5 Use a bit vector $\mathbf{a}^{(i)}$ to indicate whether there exists an element in $\mathcal{S}^{(i)}$ falling into the partitions of $\mathcal{S}$;
6 Randomize $\mathbf{a}^{(i)}$ to an integer vector $\mathbf{a}'^{(i)}$ by replacing each bit 1 in $\mathbf{a}^{(i)}$ with a random integer from $\mathbb{Z}_R$;
7 Submit materials for securely aggregating $\mathbf{b}'^{(i)}, \mathbf{a}'^{(i)}$;
8 Cloud server obtains $\sum_{i\in C} \mathbf{b}'^{(i)}$ and $\sum_{i\in C} \mathbf{a}'^{(i)}$ with secure aggregation, reconstructs the union $\bigcup_{i\in C} \mathcal{S}^{(i)}$, and delivers the union to each live client $i \in \hat{C}$.

PSU promises a wide range of applications in practice, including union queries over several databases, and, more generally, integration or sharing of datasets from multiple private sources. According to the representation format of a set, existing PSU protocols can be generally divided into two categories: polynomial-based [23, 25, 28, 29, 46] and Bloom filter-based [15, 33, 38]. For the polynomial-based protocols, elements of a set are represented as the roots of a polynomial, and the union of two sets is converted to the multiplication of two polynomials. For the protocols based on Bloom filter, the union operation over sets is normally transformed to the element-wise OR operation over the Bloom filters, and the logic OR operation can be further converted to bit addition and bit multiplication. To obliviously perform addition and multiplication operations, the two kinds of protocols mainly turn to generic secure two-party/multi-party computation, or outsource secure computation to multiple noncolluding servers. Due to unaffordable overhead, none of the existing PSU protocols have been deployed in practice. Besides inefficiency, the basic setting of these protocols significantly differs from that of FSL, where clients cannot directly communicate or natively authenticate with each other and should mediate through an untrusted cloud server. In addition, the set elements here can come from a billion-scale domain, which has not been touched in previous work as of yet.

Given the infeasibility of existing work and the atypical setting of FSL, we propose a new PSU scheme. As shown in Algorithm 3, each chosen client in one round of FSL first represents its real index set as a Bloom filter (Line 3). The details about how to set the parameters of the Bloom filter can be found in [7, 10, 47]. Then, each chosen client randomizes its Bloom filter by replacing each bit 1 with a random integer from $\mathbb{Z}_R = \{0, 1, \ldots, R-1\}$, while keeping each bit 0 unchanged (Line 4). All the chosen clients use their randomized Bloom filters rather than original ones to participate in secure aggregation (Line 7). With the secure aggregation guarantee, the untrusted cloud server obtains only the sum of the randomized Bloom filters (Line 8), but cannot know any individual randomized Bloom filter, which protects the original Bloom filter and the underlying real index set of each chosen client during the aggregation process. Further, the randomization process guarantees that the aggregate Bloom filter contains only the necessary membership information of the elements in the union and obscures unnecessary count numbers. In other words, the untrusted cloud server knows only whether there exists a client having a certain index or not, but cannot know how many clients have the index, which exactly reaches the goal of PSU. Suppose the original Bloom filters are securely aggregated without randomization. The aggregate Bloom filter is actually a counting Bloom filter, equivalent to constructing it from scratch by sequentially inserting each chosen client's real index set. Besides the desired membership information, the counting Bloom filter also contains the count numbers, which are undesired leakage in PSU. Overall, our PSU is structured internally in a new way of first local randomization and then oblivious addition, which uses the counting Bloom filter as a springboard, but conceals the undesired count numbers. Thus, our PSU avoids the costly oblivious multiplications in the common practice to derive the union of sets by performing bit-wise OR operations over the Bloom filters.

After obtaining the sum of the randomized Bloom filters, the cloud server can recover the union by doing membership tests for the full index set. It is prohibitively time-consuming if the full domain of index is huge (e.g., in the magnitude of billions in Taobao's e-commerce scenario). Even worse, the direct enumeration method can also introduce a large number of false positives to the union (i.e., those indices that are not in the union but falsely judged to be in), further leading to unnecessary redundancy in the download and upload phases. To handle these problems, we incorporate a private partition union to narrow down the scope of index for the union reconstruction above, thereby circumventing the full index set. We let the cloud server divide the full domain of index into a certain number of partitions ahead of time (Line 1). A good partition scheme needs to well balance the pros in the union reconstruction phase and the cons of additional cost. One trivial partition strategy is interval-based, where the full index set is sequentially and evenly split into some intervals. Another partition strategy is hierarchical and is often application-oriented. According to the size of partitions, we can determine the retrieval level or granularity. For example, (1) in the e-commerce recommendation, a tree structure can be used to partition the goods IDs, where the tree is initialized with the category information and further can be optimized using learning methods [57, 58]; and (2) in the natural language processing, a word semantic hierarchy can be built with the expert knowledge from WordNet [37, 39]. Given the partitions, each client first uses a bit vector to indicate whether there exists an index in its real index set falling into the partitions (Line 5). Just like hiding the concrete count numbers for the Bloom filter, the client replaces each bit 1 in its partition indicator vector with a random integer (Line 6). Then, the cloud server obtains the sum of the randomized partition indicator vectors through secure aggregation and reveals those partitions with nonzero integers in the corresponding positions (Lines 7 and 8). By simply doing membership tests for the indices falling into these partitions, the cloud server efficiently reconstructs the union and delivers it to all live clients (Line 8).

5 THEORETICAL ANALYSIS

In this section, we first analyze the security and privacy of SFSL. We then analyze its complexity and compare with SFL at the same security and privacy levels. Due to space limitations, detailed proofs and analyses are reserved in the complete manuscript [42].

5.1 Security and Privacy Analyses

Under Definition 3.1, we analyze the LDP of our index set randomization in Algorithm 2. We first prove that the union of the chosen clients' real index sets is necessary and sufficient to be the input. Then, as two stepping stones, we analyze the LDP guarantees of the permanent randomized response and a one-time instantaneous randomized response, which impose an upper bound and a lower bound on the LDP of the index set randomization, respectively.

Lemma 5.1. *The union of the chosen clients' real index sets in a certain communication round is necessary and sufficient for the index set randomization of each chosen client in the round.*

Lemma 5.2. *Permanent randomized response achieves LDP of* $\epsilon_\infty^{(i)} = \ln\left(\max\left(\frac{p_1^{(i)}}{p_2^{(i)}}, \frac{p_2^{(i)}}{p_1^{(i)}}, \frac{1-p_1^{(i)}}{1-p_2^{(i)}}, \frac{1-p_2^{(i)}}{1-p_1^{(i)}}\right)\right)$.

Lemma 5.3. *A one-time instantaneous randomized response achieves LDP of* $\epsilon_1^{(i)} = \ln\left(\max\left(\frac{p_5^{(i)}}{p_6^{(i)}}, \frac{p_6^{(i)}}{p_5^{(i)}}, \frac{1-p_5^{(i)}}{1-p_6^{(i)}}, \frac{1-p_6^{(i)}}{1-p_5^{(i)}}\right)\right)$.

Theorem 5.4. *When client i participates in any number of rounds, index set randomization satisfies $\epsilon^{(i)}$-LDP, where $\epsilon_1^{(i)} \le \epsilon^{(i)} \le \epsilon_\infty^{(i)}$.*

If client i sets $p_1^{(i)} = p_2^{(i)} = p_3^{(i)} = p_4^{(i)} = 1$, then $p_5^{(i)} = p_6^{(i)} = 1, \epsilon_1^{(i)} = 0, \epsilon_\infty^{(i)} = 0$, and $\epsilon^{(i)} = 0$. This corresponds to the case that any index in the union will always receive a permanent "Yes" answer and an instantaneous "Yes" answer. In other words, if client i takes the union as its randomized index set, it can achieve the strongest 0-LDP as in conventional FL.

We next analyze the stage of securely aggregating submodel updates in SFSL under Definition 3.2. We consider any index j from the union. We let $n_{j,0}$ and $n_{j,1}$ denote the numbers of live chosen clients who do not have and have j in reality, respectively. For feasibility and clarity in analysis, we let each chosen client use the same probability parameters, namely, $\forall i \in C, p_1^{(i)} = p_1, p_2^{(i)} = p_2, p_3^{(i)} = p_3, p_4^{(i)} = p_4$, which result in the same $p_5^{(i)} = p_1(p_3 - p_4) + p_4 = p_5$ and $p_6^{(i)} = p_2(p_3 - p_4) + p_4 = p_6$. Then, we have Theorem 5.5 as follows:

Theorem 5.5. *SFSL is a client-tunable privacy protection mechanism for aggregating submodel updates and guarantees that: Event 1 happens with probability $p_7 = n_{j,1}p_5(1-p_5)^{n_{j,1}-1}(1-p_6)^{n_{j,0}}$, and Event 2 happens with probability $p_8 = (1-p_5)^{n_{j,1}}(1-(1-p_6)^{n_{j,0}})$.*

We still examine the case that each chosen client uses the union as its randomized index set to securely upload the submodel update by setting $p_1 = p_2 = p_3 = p_4 = 1$, resulting in $p_5 = p_6 = 1$ and $p_7 = p_8 = 0$. This is the strongest privacy for aggregating submodel updates as in SFL. Combining with the aforementioned 0-LDP for each chosen client, we can find that SFSL with the setting $p_1 = p_2 = p_3 = p_4 = 1$ (essentially, $p_5 = p_6 = 1$) is as secure as SFL under Definition 3.1 and Definition 3.2. We further generalize this observation in Theorem 5.6, which is free of any privacy definition.

Theorem 5.6. *If each chosen client uses the union of real index sets as its randomized index set, then SFSL is as secure as SFL.*

We finally analyze the security of our PSU scheme in Algorithm 3.

Theorem 5.7. *Only the union of the chosen clients' real index sets is revealed in the proposed PSU protocol.*

Table 1: Complexities of SFSL (and PSU in it) at the same security and privacy levels as SFL.

		Communication	Time	Space
Client	SFSL	$O(nsd)$	$O(n^2sd)$	$O(nsd)$
	PSU	$O(ns)$	$O(n^2s)$	$O(ns)$
	SFL	$O(n+md)$	$O(n^2+nmd)$	$O(n+md)$
Server	SFSL	$O(n^2sd)$	$O(n^3sd)$	$O(n^2+nsd)$
	PSU	$O(n^2s)$	$O(n^3s)$	$O(n^2+ns)$
	SFL	$O(n^2+nmd)$	$O(n^2md)$	$O(n^2+md)$

*$|\bigcup_{i\in C}\mathcal{S}^{(i)}| \ll |\mathcal{S}| \Rightarrow ns \ll m$.

5.2 Complexity Analysis and Comparison

We analyze the communication, time, and space complexities of the client and the cloud server in one round of SFSL. We introduce SFL as a benchmark for comparison. To be consistent with the analysis of the secure aggregation protocol [8], we adopt the honest-but-curious setting and consider the worst-case dropouts of clients.

We first analyze SFSL. For feasibility and clarity, we let each client use the same p_1, p_2, p_3, p_4 and the resulting p_5, p_6. We assume that the expected cardinality of each client's real index set is s, which indicates that the expected cardinality of the union of n chosen clients' real index sets $\bigcup_{i\in C}\mathcal{S}^{(i)}$ is upper bounded by ns. We note that ns is normally much less than the cardinality of the full index set m (i.e., $ns \ll m$). In addition, the expected cardinality of each client's randomized index set $\mathcal{S}'^{(i)}$ is upper bounded by $sp_5+(n-1)sp_6$, and the expected cardinality of each client's succinct index set $\mathcal{S}^{(i)} \bigcap \mathcal{S}'^{(i)}$ is sp_5. Moreover, according to [7, 10, 47], the optimal length of the Bloom filter in the proposed PSU protocol is proportional to the cardinality of the set to be filtered, namely, the cardinality of the union. In what follows, we directly show the complexity formulas containing p_5, p_6. First, the overall communication complexities of the client and the cloud server are $O(ns+(sp_5+(n-1)sp_6)(2d+1))$ and $O(n^2s+n(sp_5+(n-1)sp_6)(2d+1))$, respectively. Second, the overall time complexities of the client and the cloud server are $O(n^2s+n(sp_5+(n-1)sp_6)(d+1))$ and $O(n^3s+n^2(sp_5+(n-1)sp_6)(d+1))$, respectively. Third, the overall space complexities of the client and the cloud server are $O(ns+(sp_5+(n-1)sp_6)(d+1))$ and $O(n^2+ns+(sp_5+(n-1)sp_6)(n+d+1))$, respectively. In fact, considering secure aggregation is the most costly primitive underlying SFSL, the complexity formulas above can be roughly estimated by leveraging the complexity analysis results of the secure aggregation protocol reported in [8]. More specifically, we let the dimension of vector there take $O(ns+(sp_5+(n-1)sp_6)(d+1))$ here, where $O(ns)$ corresponds to the optimal length of the Bloom filter and the preset constant size of the partition indicator vector in our PSU, and $(sp_5+(n-1)sp_6)(d+1)$ accounts for the total size of the weighted submodel update and corresponding count vector with the randomized index set. From the analysis above, we can derive that our SFSL is quite scalable because its complexity depends on the cardinality of the union of n chosen clients' real index sets ns, but is independent of the cardinality of the full index set m that controls the size of the full model. In other words, SFSL completely relieves the dependence on the full model.

Table 2: Statistics of Taobao dataset.

Type	#User(s)	#Goods	#Categories	#Samples
Test (Full)	24,790	138,829	4,758	1,010,284
Train (Full)	49,023	143,534	4,815	15,854,357
Train (Per Client)	1	301	117	323

We next compare our SFSL with SFL at the same security and privacy levels by imposing $p_5 = p_6 = 1$ in SFSL. Table 1 shows their complexities along with the complexity of our PSU. The complexities of SFL and PSU are derived using the complexity analysis of the secure aggregation protocol, where the dimension of vector takes the size of the full model md and $O(ns)$, respectively. From Table 1, we can draw that given $ns \ll m$ in our application scenario, the complexities of the client and the cloud server in SFSL are both far lower than those in SFL. We can also draw that only if $ns < m$, our SFSL outperforms SFL. In the worst case where $ns = m$ (i.e., the union of the chosen clients' real index sets is the full index set), the complexity of SFSL is the same as that of SFL. An intuitive explanation is that if all the clients participate in each round of FSL, the union of the chosen clients' real index sets is the full index set, and SFSL degenerates to SFL. However, the full participation is ideal and impractical because of the unavailability of some clients in each round. Thus, the worst case may never happen. We note that ns cannot be larger than m, because the full index set is actually the union of all the clients' real index sets. In summary, SFSL outperforms SFL in any scale of FSL applications, where ubiquitous data heterogeneity results in model differentiation for the clients.

6 EVALUATION

Dataset: We use an industrial dataset[1] built from 30-day impression and click logs of Taobao users from June 15, 2019 to July 15, 2019. For a certain Taobao user, we leverage his/her click behaviors in previous 14 days as historical data to predict his/her click and non-click behaviors in the following 1 day. We leave out the behaviors within the last 1 day as the target items of the test set while putting the other samples into the training set. Specifically, the test set is located on the cloud server to judge the accuracy and convergency of the global model. For the full training set, we further cluster each Taobao user's data as a training set located on a client. Table 2 shows some statistics about the dataset.

Model, Hyperparameters, and Metric: We take the deployed DIN in Alibaba [56] as the recommendation model for FSL, where the number of columns in the embedding matrix is set to 18. Except the embedding layer for user IDs, goods IDs, and category IDs, the parameters of the other network layers in DIN, including the attention layer and the fully connected layer, are of size 64,327. Hence, the global model on the cloud server is of size 3,617,023, whereas a desired submodel on the client is of size 71,869 in average, which is only 1.99% of the global model's size and roughly requires 0.27MB space using 32-bit representation. For each client's local training, we choose mini-batch *stochastic gradient descent (SGD)* as the optimization algorithm, set the batch size to 2, and set the local epoch number to 1. In addition, we initially set the learning rate to 1 and further apply exponential decay with the decay rate of 0.999 per communication round. For the cloud server's global testing, we adopt a golden metric in the task of *click-through rate (CTR)* prediction, called AUC, and set the batch size to 1,024.

[1] A similar Taobao dataset is online available from Tianchi [3].

Table 3: Choices of probability parameters (CPPs) and resulting privacy levels. CPP1 corresponds to pure FSL with secure aggregation. CPP5 is as secure as SFL.

	p_1, p_3	p_2, p_4	p_5	p_6	ϵ_1	ϵ_∞	p_7	p_8
CPP1	1	0	1	0	∞	∞	86.7%	0
CPP2	15/16	1/16	88.3%	11.7%	2.02	2.71	0	10.3%
CPP3	7/8	1/8	78.1%	21.9%	1.27	1.95	0	19.5%
CPP4	3/4	1/4	62.5%	37.5%	0.51	1.10	0	34.2%
CPP5	1	1	1	1	0	0	0	0

* Smaller ϵ_1, ϵ_∞, p_7, p_8 indicate better privacy for each chosen client i.

Prototypes and Configurations: We implemented the prototypes of our SFSL and the baseline SFL in Python 2.7.16, and the source code is publicly available from [41]. Figure 2 provides an overview of SFSL from the perspective of a client in a certain round. We took a synchronous architecture on top and implemented a communication module between the cloud server and each client with standard socket programming. We used TensorFlow 1.12.0 to implement DIN. We mainly used PyCryptodome 3.7.3 to implement the secure aggregation protocol in [8]. To support modular addition underlying secure aggregation, we performed float-to-integer conversion for each client's submodel or full model update with stochastic quantization [48], where the quantization level is set to 2^{15}. We let each client use the same *choice of probability parameters (CPP)* in SFSL. Table 3 lists 5 CPPs in the evaluation and their resulting privacy levels of index set randomization and aggregating submodel updates when the number of chosen clients per round n is 100. Specifically, CPP1 corresponds to directly combining FSL with secure aggregation, which lets each client reveal its real index set and provides the worst LDP. The resulting $p_7 = 86.7\%$ at CPP1 further implies that 86.7% of the goods IDs in the union of 100 randomly chosen clients' real goods IDs involve a single client. Therefore, the user data in Taobao's e-commerce context are highly heterogeneous. In contrast, CPP5 lets each client use the union of the chosen clients' real index sets as its randomized index set. By Theorem 5.6, SFSL with CPP5 is as secure as SFL.

Our running environment is a Linux workstation with 64-bit Ubuntu 18.04.2 OS. The processor is Intel(R) Core(TM) i9-9900K with 8 cores, the base frequency is 3.60GHz, the memory size is 64GB, and the cache size is 16MB. The workstation is also equipped with 2 NVIDIA's GeForce RTX 2080 Ti graphics cards. To manifest the difference between the roles of client and cloud server, (1) from hardware, we ran all the clients only on CPU, but allowed the cloud server to accelerate its operations using GPU; and (2) from parallelism and concurrency, we optimized the cloud server's hotspot functions with Python's multiprocessing library. For more implementation details, please refer to the complete manuscript [42].

6.1 Model Accuracy and Convergency

We bring in centralized training and conventional FL as two baselines. We plot the AUCs of our SFSL and two baselines in Figure 3.

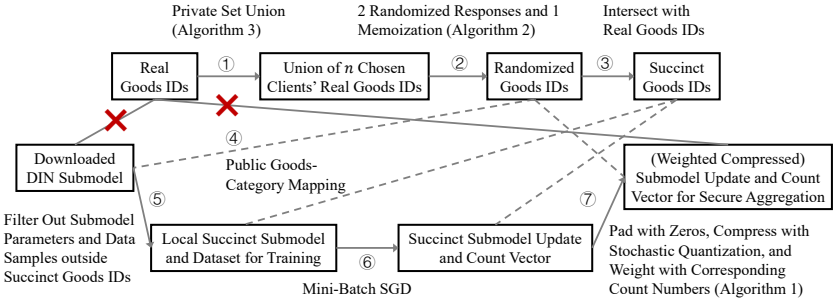


Figure 2: An implementation overview of SFSL from a client (i.e., a Taobao user) in a certain round for DIN-based CTR prediction.

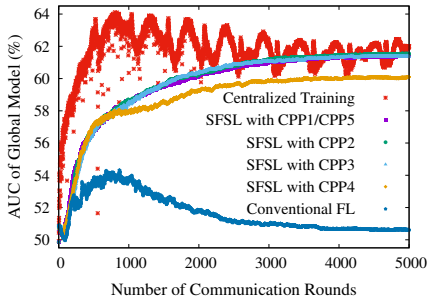
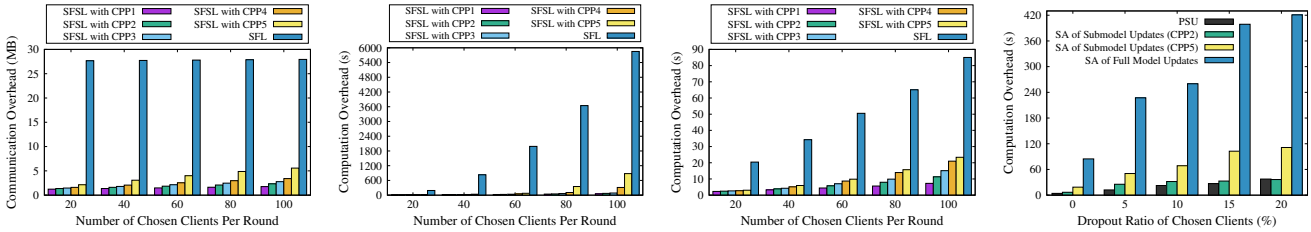


Figure 3: AUCs of the global model in centralized training, SFSL, and FL.



(a) Per Client (Overall) (b) Per Client (Overall) (c) Cloud Server (Overall) (d) Cloud Server (SA)

Figure 4: Communication and computation overhead of the client and the cloud server per round in our SFSL and the baseline SFL. In Figure 4(d), SA denotes secure aggregation, and 100 clients are chosen per round.

The number of clients randomly chosen in each round n is set to 100. In addition, centralized training refers to the traditional case that the cloud server first collects data from all the clients, then trains the DIN model, and tests the global model once training over the samples with a size similar to the total size of n clients' datasets.

From Figure 3, we can see that compared with centralized training, which reaches the best AUC of 64.11% in 803 rounds, our SFSL with CPP2 achieves the highest AUC of 61.54% in 4,908 rounds, decreasing by 2.57% of AUC. In contrast, conventional FL performs worst, achieving the highest AUC of 54.32% in 867 rounds and diverging in the end. The major reason is that FL coarsely computes the weighted average of the chosen clients' full model updates proportional to their training set sizes, no matter whether one client's whole training set actually involves some model parameters (the full model excluding its desired submodel, e.g., some embedding vectors in DIN) or not, thereby inaccurately counting in the weights (i.e., the training set sizes) of those clients who contribute zero/no updates for these model parameters. With a higher heterogeneity of user data and a higher differentiation of submodels, the roughness and inaccuracy of FL will be exposed more completely, which clarifies why FL can work in the natural language context with a 10,000-word vocabulary considered by Google, but does not work well in Alibaba's e-commerce context with billion-scale goods IDs.

From Figure 3, we can also observe that CPP4 is the worst among all CPPs and achieves the highest AUC of 60.11%, because p_5, which controls the size of each client's succinct training set as well as the length of historical goods and category IDs in each training sample, in CPP4 is the smallest among all CPPs. This still explains another observation that CPP1 and CPP5 with the same $p_5 = 1$ have identical model performance.

6.2 Communication Overhead

We show the communication overhead of our SFSL and introduce SFL as a baseline. Figure 4(a) plots the overall communication overhead per client per round. We do not plot the communication overhead of the cloud server, since it is equal to the communication overhead per client multiplying by n. In more detail, the incoming data of the cloud server are exactly the total outgoing data of all n chosen clients, and vice verse. We also do not plot for different dropout ratios because this factor has little impact on the communication overhead.

One key observation from Figure 4(a) is that compared with SFL, our SFSL can sharply reduce the communication overhead. In particular, when $n = 100$, the overall communication overhead per client per round is 1.76MB, 2.33MB, 2.78MB, 3.40MB, and 5.57MB in SFSL with CPP1, CPP2, CPP3, CPP4, and CPP5, respectively, reducing 93.72%, 91.65%, 90.06%, 87.81%, and 80.05% than SFL, which incurs 27.94MB. Considering SFSL with CPP5 is as secure as SFL, we can draw that our SFSL can reduce communication overhead even with not sacrificing any security or privacy. These results coincide with the complexity analysis in Section 5.2 and Table 1.

The second key observation from Figure 4(a) is that the communication overhead per client in SFSL increases with n for a certain CPP and also increases with the serial number of CPP for a certain n. We clarify the reasons by the client's communication complexity $O(ns + (sp_5 + (n-1)sp_6)(2d+1))$ in Section 5.2. On the one hand, the communication complexity grows linearly with n. On the other hand, it is increasing with p_5 and p_6, and thus, CPP5 is the most communication expensive. Additionally, given $p_5 + p_6 = 1$ for CPPs from CPP1 to CPP4, we can simplify the communication complexity to $O(ns + (s + (n-2)sp_6)(2d+1))$, which increases with p_6 as $n > 2$.

From Table 3, we can see that p_6 increases with the serial number of CPP, implying higher communication overhead as depicted in Figure 4(a). Intuitively, p_6 controls the size of the redundant/zero parameters to be downloaded and securely uploaded and dominates the holistic trend of the communication overhead.

We finally present the communication overhead per client per round incurred by our PSU, which increases linearly with n, roughly with an increase of 0.07MB per 20 clients. In addition, our PSU is quite communication efficient and incurs 0.91MB as n reaches 100.

6.3 Computation Overhead

We now report the practical computation overhead, mainly by investigating the effects of n, the CPP, and the ratio of dropout. To be consistent with the time complexity analysis, the computation overhead includes only the run time of the client or the cloud server in executing the protocol, but ignores synchronization delay and the time overhead of testing global model. Given mobile devices are highly parallel in practice, the total run time per communication round can be estimated by adding up the computation overhead of the client and the cloud server reported here. In addition, testing the global model per round costs the cloud server 32.12s.

We first show the overall computation overhead of the client and the cloud server per round in our SFSL with different CPPs in Figure 4(b) and Figure 4(c). We still introduce SFL as a baseline. First, we can see that our SFSL significantly outperforms SFL on both sides of the client and the cloud server. In particular, when $n = 100$, at the same security and privacy levels, SFSL with CPP5 reduces 85.02% and 72.51% of the computation overhead than SFL on the client and the cloud server, respectively. When security and privacy become weaker, the advantages of our SFSL are more evident under CPP1 to CPP4. For example, CPP2 reduces 98.77% and 86.70% of the run time than SFL on the client and the cloud server, respectively. Second, we focus on a certain side, either the client or the cloud server, and can observe that its computation overhead grows with n or the serial number of CPP. The reason can be explained by the detailed time complexity in Section 5.2, while the intuition is analogous to that behind the communication overhead.

We next investigate the effect of the dropout ratio and depict the evaluation results per round in Figure 4(d) as $n = 100$. We mainly focus on the secure aggregation-based stages while ignoring the other stages irrelevant with dropout. We consider that clients are randomly selected to go offline at random points. We report only the computation overhead of the cloud sever because dropped clients do not introduce additional operation cost for live clients. From Figure 4(d), we can see that with the growth of the dropout ratio, the computation overhead of the cloud server increases. This is because the cloud server needs to remove the mutual masks between dropped and live clients in secure aggregation. We also compare our SFSL with SFL at the stage of securely aggregating submodel or full model updates. We can find that SFSL significantly outperforms SFL at this stage for any dropout ratio. Specifically, when the dropout ratio is 20%, CPP2 and CPP5 reduce 91.33% and 73.55% of the computation overhead, respectively. We finally examine our PSU and can see that it is quite efficient, even when the dropout ratio is high. In particular, the computation overhead of the cloud server is 37.66s as the dropout ratio reaches 20%.

6.4 Memory and Disk Loads

We present the practical storage overhead of our SFSL and the baseline SFL. First is about the memory overhead. The cloud server requires the video memory of 551MB, mainly for testing the global model at the end of each communication round, which is the same for all schemes. In addition, when $n = 100$, and there is no dropout, the memory overhead per client is 209MB and 281MB in SFSL with CPP2 and CPP5, respectively, reducing 59.40% and 45.43% than SFL. Correspondingly, the memory overhead of the cloud server is 1.58GB and 3.15GB, respectively, reducing 81.88% and 63.77% than SFL. Regarding the disk load, only the client in SFSL with CPP2, CPP3, and CPP4 needs to store its permanent noisy answers in the index set randomization, which roughly occupies the disk space of 280KB within the total 5,000 rounds.

6.5 Discussion on Billion-Scale Issues

We finally discuss the issues as the size of the full index set (which controls the size of the full model) and the total number of clients scale further to billions in practice. First, the baseline SFL, depending on the full model, will be prohibitively inefficient to be applicable. Second, we analyze our SFSL. SFSL relives the dependence on the full model, executes in communication rounds, and involves only a constant number of (e.g., $n = 100$) clients in each round. Thus, no additional overhead will be incurred in SFSL due to the scale of billions. We further consider how to shorten the periods of building/polishing the full model and covering the whole clients without affecting any client's efficiency and privacy. It is trivial to scale up SFSL at the level of the cloud server. In particular, a master server manages multiple child servers. For every round, each child server selects n clients, lets them use SFSL to download submodels, and securely aggregates their submodel updates with SFSL. The master server further aggregates the aggregated submodel updates from the child servers to update the global model.

7 CONCLUSION

In this paper, we have proposed a submodel design SFSL for mobile clients to effectively and efficiently train large-scale models under the coordination of an untrusted cloud server while keeping user data private. We further have applied SFSL to the e-commerce recommendation scenario of Alibaba, implemented a prototype system, and validated its practical feasibility over a Taobao dataset.

ACKNOWLEDGMENTS

We sincerely thank our anonymous shepherd and reviewers for their insightful feedback. We also thank Renjie Gu, Hongtao Lv, Hejun Xiao, and Zhenzhe Zheng, as well as many colleagues in Alibaba, for meaningful discussions and great engineering support. This work was supported in part by National Key R&D Program of China No. 2019YFB2102200, China NSF grant No. 61972252, 61972254, 61672348, and 61672353, Joint Scientific Research Foundation of the State Education Ministry No. 6141A02033702, and Alibaba Innovation Research (AIR) Program. The opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies or the government. Fan Wu (fwu@cs.sjtu.edu.cn) is the corresponding author of this paper.

REFERENCES

[1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *Proc. of CCS*. ACM, 308–318.

[2] Naman Agarwal, Ananda Theertha Suresh, Felix Yu, Sanjiv Kumar, and H. Brendan McMahan. 2018. cpSGD: Communication-efficient and differentially-private distributed SGD. In *Proc. of NeurIPS*. 7575–7586.

[3] Alimama. 2017. Ad Display/Click Data on Taobao.com. https://tianchi.aliyun.com/dataset/dataDetail?dataId=56.

[4] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. 2018. PIR with Compressed Queries and Amortized Query Processing. In *Proc. of S&P*. IEEE, 962–979.

[5] Apple's Differential Privacy Team. 2017. Learning with Privacy at Scale. *Apple Machine Learning Journal* 1, 8 (2017).

[6] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How To Backdoor Federated Learning. In *Proc. of AISTATS*. PMLR, 2938–2948.

[7] Burton H. Bloom. 1970. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM* 13, 7 (1970), 422–426.

[8] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proc. of CCS*. ACM, 1175–1191.

[9] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. 2005. Evaluating 2-DNF Formulas on Ciphertexts. In *Proc. of TCC*. Springer, 325–341.

[10] Andrei Broder and Michael Mitzenmacher. 2004. Network Applications of Bloom Filters: A Survey. *Internet Mathematics* 1, 4 (2004), 485–509.

[11] Qingqing Cao, Noah Weber, Niranjan Balasubramanian, and Aruna Balasubramanian. 2019. DeQA: On-Device Question Answering. In *Proc. of MobiSys*. ACM, 27–40.

[12] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. 2018. Labeled PSI from Fully Homomorphic Encryption with Malicious Security. In *Proc. of CCS*. ACM, 1223–1237.

[13] Mingqing Chen, Rajiv Mathews, Tom Ouyang, and Françoise Beaufays. 2019. Federated Learning Of Out-Of-Vocabulary Words. arXiv: 1903.10635. http://arxiv.org/abs/1903.10635.

[14] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior Sequence Transformer for E-commerce Recommendation in Alibaba. arXiv: 1905.06874. http://arxiv.org/abs/1905.06874.

[15] Alex Davidson and Carlos Cid. 2017. An Efficient Toolkit for Computing Private Set Operations. In *Proc. of ACISP*. Springer, 261–278.

[16] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately. In *Proc. of NeurIPS*. 3574–3583.

[17] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407.

[18] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2019. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. In *Proc. of SODA*. ACM-SIAM, 2468–2479.

[19] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proc. of CCS*. ACM, 1054–1067.

[20] European Parliament and Council of the European Union. 2016. The General Data Protection Regulation (EU) 2016/679 (GDPR). https://eur-lex.europa.eu/eli/reg/2016/679/oj. Took effect from May 25, 2018.

[21] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. 2016. Building a RAPPOR with the Unknown: Privacy-Preserving Learning of Associations and Data Dictionaries. *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2016, 3 (2016), 41–61.

[22] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. 2014. Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing. In *Proc. of USENIX Security*. 17–32.

[23] Keith Frikken. 2007. Privacy-Preserving Set Union. In *Proc. of ACNS*. Springer, 237–252.

[24] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated Learning for Mobile Keyboard Prediction. arXiv: 1811.03604. http://arxiv.org/abs/1811.03604.

[25] Jeongdae Hong, Jung Woo Kim, Jihye Kim, Kunsoo Park, and Jung Hee Cheon. 2013. Constant-round privacy preserving multiset union. *Bulletin of the Korean Mathematical Society* 50, 6 (2013), 1799–1816.

[26] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2019. Advances and Open Problems in Federated Learning. arXiv: 1912.04977. http://arxiv.org/abs/1912.04977.

[27] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2008. What Can We Learn Privately?. In *Proc. of FOCS*. IEEE, 531–540.

[28] Lea Kissner and Dawn Xiaodong Song. 2005. Privacy-Preserving Set Operations. In *Proc. of CRYPTO*. Springer, 241–257.

[29] Vladimir Kolesnikov, Mike Rosulek, Ni Trieu, and Xiao Wang. 2019. Scalable Private Set Union from Symmetric-Key Techniques. IACR Cryptology ePrint Archive, Report 2019/776. https://eprint.iacr.org/2019/776.

[30] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated Learning: Strategies for Improving Communication Efficiency. arXiv: 1610.05492. http://arxiv.org/abs/1610.05492.

[31] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60.

[32] Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. 2019. SDM: Sequential Deep Matching Model for Online Large-scale Recommender System. In *Proc. of CIKM*. ACM, 2635–2643.

[33] Dilip Many, Martin Burkhart, and Xenofontas Dimitropoulos. 2012. *Fast private set operations with sepia.* Technical Report TIK-Report No. 345. Communication Systems Group, ETH Zürich, Switzerland.

[34] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proc. of AISTATS*. PMLR, 1273–1282.

[35] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *Proc. of ICLR*. OpenReview.net.

[36] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. 2019. Exploiting Unintended Feature Leakage in Collaborative Learning. In *Proc. of S&P*. IEEE, 497–512.

[37] George A. Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM* 38, 11 (1995), 39–41.

[38] Atsuko Miyaji and Katsunari Shishido. 2018. Efficient and Quasi-accurate Multiparty Private Set Union. In *Proc. of SMARTCOMP*. IEEE, 309–314.

[39] Frederic Morin and Yoshua Bengio. 2005. Hierarchical Probabilistic Neural Network Language Model. In *Proc. of AISTATS*. Society for Artificial Intelligence and Statistics, 246–252.

[40] M. Nasr, R. Shokri, and A. Houmansadr. 2019. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *Proc. of S&P*. IEEE, 1021–1035.

[41] Chaoyue Niu, Renjie Gu, Hongtao Lv, and Hejun Xiao. 2020. Source code for secure federated submodel learning. https://github.com/NiuChaoyue/Secure-Federated-Submodel-Learning.

[42] Chaoyue Niu, Fan Wu, Shaojie Tang, Lifeng Hua, Rongfei Jia, Chengfei Lv, Zhihua Wu, and Guihai Chen. 2019. Secure Federated Submodel Learning. arXiv: 1911.02254. http://arxiv.org/abs/1911.02254.

[43] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Proc. of EUROCRYPT*. Springer, 223–238.

[44] Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. 2018. Private Stateful Information Retrieval. In *Proc. of CCS*. ACM, 1002–1019.

[45] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. 2019. Federated Learning for Emoji Prediction in a Mobile Keyboard. arXiv: 1906.04329. http://arxiv.org/abs/1906.04329.

[46] Jae Hong Seo, Jung Hee Cheon, and Jonathan Katz. 2012. Constant-Round Multiparty Private Set Union Using Reversed Laurent Series. In *Proc. of PKC*. Springer, 398–412.

[47] David Starobinski, Ari Trachtenberg, and Sachin Agarwal. 2003. Efficient PDA Synchronization. *IEEE Transactions on Mobile Computing* 2, 1 (2003), 40–51.

[48] Ananda Theertha Suresh, Felix X. Yu, Sanjiv Kumar, and H. Brendan McMahan. 2017. Distributed Mean Estimation with Limited Communication. In *Proc. of ICML*. PMLR, 3329–3337.

[49] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale Commodity Embedding for E-commerce Recommendation in Alibaba. In *Proc. of KDD*. ACM, 839–848.

[50] Tianhao Wang, Bolin Ding, Jingren Zhou, Cheng Hong, Zhicong Huang, Ninghui Li, and Somesh Jha. 2019. Answering Multi-Dimensional Analytical Queries under Local Differential Privacy. In *Proc. of SIGMOD*. ACM, 159–176.

[51] Tianhao Wang, Ninghui Li, and Somesh Jha. 2018. Locally Differentially Private Frequent Itemset Mining. In *Proc. of S&P*. IEEE, 127–143.

[52] Stanley L. Warner. 1965. Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. *J. Amer. Statist. Assoc.* 60, 309 (1965), 63–69.

[53] Xiufeng Xie and Kyu-Han Kim. 2019. Source Compression with Bounded DNN Perception Loss for IoT Edge Computer Vision. In *Proc. of MobiCom*. ACM, 47:1–47:16.
[54] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. *ACM Transactions on Intelligent Systems and Technology* 10, 2 (2019), 12:1–12:19.
[55] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep Interest Evolution Network for Click-Through Rate Prediction. In *Proc. of AAAI*. AAAI Press, 5941–5948.
[56] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *Proc. of KDD*. ACM, 1059–1068.
[57] Han Zhu, Daqing Chang, Ziru Xu, Pengye Zhang, Xiang Li, Jie He, Han Li, Jian Xu, and Kun Gai. 2019. Joint Optimization of Tree-based Index and Deep Model for Recommender Systems. In *Proc. of NeurIPS*. 3973–3982.
[58] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning Tree-based Deep Model for Recommender Systems. In *Proc. of KDD*. ACM, 1079–1088.
[59] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep Leakage from Gradients. In *Proc. of NeurIPS*. 14774–14784.

数据库领域

VLDB 2020

EdgeDIPN: a Unified Deep Intent Prediction Network Deployed at the Edge

Long Guo, Lifeng Hua, Rongfei Jia
Fei Fang, Binqiang Zhao
Alibaba Inc
{leo.gl,issac.hlf,rongfei.jrf,mingyi.ff,binqiang.zhao}@alibaba-inc.com

Bin Cui
School of EECS & Key Laboratory of High Confidence Software Technologies (MOE), Peking University
bin.cui@pku.edu.cn

ABSTRACT

With the rapid growth of e-commerce in recent years, e-commerce platforms are becoming a primary place for people to find, compare and ultimately purchase products. To improve online shopping experience for consumers and increase sales for sellers, it is important to understand user intent accurately and be notified of its change timely. In this way, the right information could be offered to the right person at the right time. To achieve this goal, we propose a unified deep intent prediction network, named EdgeDIPN, which is deployed at the edge, i.e., mobile device, and able to monitor multiple user intent with different granularity simultaneously in real-time. We propose to train EdgeDIPN with multi-task learning, by which EdgeDIPN can share representations between different tasks for better performance and saving edge resources in the meantime. In particular, we propose a novel task-specific attention mechanism which enables different tasks to pick out the most relevant features from different data sources. To extract the shared representations more effectively, we utilize two kinds of attention mechanisms, where the multi-level attention mechanism tries to identify the important actions within each data source and the inter-view attention mechanism learns the interactions between different data sources. In the experiments conducted on a large-scale industrial dataset, EdgeDIPN significantly outperforms the baseline solutions. Moreover, EdgeDIPN has been deployed in the operational system of Alibaba. Online A/B testing results in several business scenarios reveal the potential of monitoring user intent in real-time. To the best of our knowledge, EdgeDIPN is the first full-fledged real-time user intent understanding center deployed at the edge and serving hundreds of millions of users in a large-scale e-commerce platform.

PVLDB Reference Format:
Long Guo, Lifeng Hua, Rongfei Jia, Fei Fang, Binqiang Zhao, and Bin Cui. EdgeDIPN: a Unified Deep Intent Prediction Network Deployed at the Edge. PVLDB, 14(3): 320-328, 2021.
doi:10.14778/3430915.3430922

1 INTRODUCTION

E-commerce platforms are becoming a primary place for people to find, compare and ultimately purchase products. One of the fundamental questions that arises in e-commerce is to understand user intent, which allows for providing better services for both sellers and customers. In reality, a user's intent is changing all the time when surfing the e-commerce app. For example, a user's intent may change from browsing to buying when she finds some goods of interest. It is important to be notified of the change of user intent timely, which can provide users with brand new proactive experiences by offering the the right information at the right time. However, previous work only deals with the offline user intent prediction [7, 18, 23, 24, 27] and cannot monitor user intent in real-time. The reasons are twofold. First, previous work cannot predict users' real-time intent with a high accuracy limited by the representation capability of traditional coarse-grained behavior data. Second, it is difficult to monitor user intent in real-time in traditional cloud-based computing architecture due to the high communication cost.

In order to overcome the aforementioned difficulty when monitoring user intent in real-time, we propose to deploy our user intent understanding system at the edge, i.e., mobile devices, following the idea of edge computing. There are several advantages to deploy the system at the edge. First, it offers us the opportunity to unlock the potential of the vast untapped data created by the connected devices. In our system, we collect a new type of user interactive behavior data, which we call touch-interactive behavior and contributes a lot when predicting user's real-time intent. Second, it can reduce the communication cost between the cloud and the devices significantly. During the Alibaba 2019 Double 11 Shopping Festival, EdgeDIPN serves more than 0.5 billion customers without suffering from the traditional peak traffic problem of the platform on that day. Third, it can greatly increase the response speed of EdgeDIPN by moving the model to the data instead of the data to the model. The response time of EdgeDIPN is between 20 ~ 50 ms on different devices, which is immune to the influence of the network traffic and can achieve the real-time requirement. Last but not least, it enables us to create new business scenarios in the e-commerce platform based on the real-time user intent.

However, monitoring user intent in real-time at the edge is challenging. First, it is necessary to build a computationally lightweight and memory efficient system, because edge resources are typically resource-constrained. Second, the touch-interactive behavior is more fine-grained and contains less semantic information compared with traditional browse-interactive behavior. Therefore, it is challenging to extract useful features from these data to improve the prediction performance. To solve these challenges, we propose a unified deep intent prediction network, named EdgeDIPN, which is able to monitor multiple user intent with different granularity simultaneously in real-time. To save edge resources, we propose

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 14, No. 3 ISSN 2150-8097.
doi:10.14778/3430915.3430922

to train EdgeDIPN with multi-task learning. In more details, we extract a universal shared representation based on users' behavior data from several views, based on which multiple user intent with different granularity could be derived. In this way, edge resources can be reduced dramatically, because we do not need to build a model for each intent separately. We improve the performance of multi-task learning from two aspects. First, we utilize two kinds of attention mechanisms to extract the shared representation more effectively, where the multi-level attention mechanism tries to identify the important actions within each data source and the inter-view attention mechanism learns the interactions between different data sources. Second, we propose a novel task-specific attention mechanism which enables different tasks to pick out the most relevant features from different data sources. The contribution of the paper can be summarized as follows:

- We propose a unified deep intent prediction network, named EdgeDIPN, which is deployed at the edge and able to monitor multiple user intent with different granularity simultaneously in real-time.
- We propose to train EdgeDIPN with multi-task learning by sharing representations among several intent, which can save edge resources dramatically. In addition, several attention mechanisms are proposed to improve the performance.
- We evaluate the performance of EdgeDIPN by conducting extensive experiments on a large-scale industrial dataset. The results show the superiority of EdgeDIPN in predicting user intent. In particular, EdgeDIPN has been deployed in the operational system of Alibaba. Online A/B testing results show the benefits of monitoring user intent in real-time.

2 PROBLEM STATEMENT

2.1 Dataset

We build two types of user interactive behavior dataset which depict users from different views, i.e., the novel touch-interactive behavior collected from the edge and the traditional browse-interactive behavior. In particular, the touch-interactive behavior is further divided into the swipe and tap interactive behavior. The touch-interactive behavior records a user's fine-grained behavior when surfing in the app and thus contains rich user behavior patterns.

The swipe-interactive behavior includes four types of basic actions, i.e., *Open Page*, *Leave Page*, *Swipe* and *Tap*. A user's swipe-interactive track is a time sequence of actions, consisting of these four basic types of actions. Each action has a timestamp and a page index to identify when and where the action occurs. In addition, the positional coordinates of the action on the touch screen are also recorded (i.e., we record the tap position for the *Tap* action and the start and end positions for the *Swipe* action.). The duration presents how long the action lasts. For each action at a data point, we extract 14 features. Among these features, the time duration of a swipe, time gap between two actions and positional coordinates of actions (i.e., tap position X/Y, swipe start position X/Y, swipe end position X/Y and swipe length on X/Y) are continuous variables. Page indices, action indices and swipe directions (i.e., left/right and up/down) are categorical variables. We conduct discretization on all the raw features to ensure unified inputs for EdgeDIPN. The discretization of the continuous variables is described as follows:

- **Position.** The positional coordinates of actions are discretized according to the resolution of the touch screen. We divide the width and length of the screen into 17 and 25 uniform segments corresponding to X and Y, respectively, for one-hot vectors encoding.
- **Swipe Length.** The length of a swipe is encoded into a one-hot vector, the length of which is as twice as the length of the one-hot vectors of position encoding. The reason of applying twice length is that, for a swipe track, we consider the direction of the swipe.
- **Time Gap and Duration.** We apply a step function to encode time gaps between actions and swipe duration as follow:

$$y = \begin{cases} x/f_s, & x < f_b \\ x/f_b + 9, & f_b \le x < 10 \times f_b \\ 19, & x \ge 10 \times f_b \end{cases}$$

  where $\{f_s = 100, f_b = 1000\}$ are used for time gap, and $\{f_s = 25, f_b = 250\}$ are used for time duration.

The tap-interactive behavior records the information associated with the tap actions. A user's tap-interactive track is a time sequence of tap actions. Each action has a timestamp and page index to identify when and where the action occurs. There is also an event id to identify whether a user taps on a page or a button. If a button is tapped, the button name is also recorded. We extract 3 raw features (i.e., event index, page index and button index) for each tap action, all of which are categorical variables.

The browse-interactive behavior represents the typical behavior users conduct on products. It includes five types of actions, i.e., *Browse*, *Search*, *collect*, *Add to cart* and *Purchase*. A user's browse-interactive track is a time sequence of these actions. We extract 6 raw features for each action (i.e., type index, top category index, leaf category index, page index, page stay time and timestamp), where the type index represents the type of an action.

2.2 Problem Formulation

EdgeDIPN is used to monitor multiple user intent with different granularity in real-time, which enables us to send the right information to the right user at the right time. The typical predicted user intent is listed as follows.

- **Real-time Purchasing Intent**: whether a user would buy some goods within one hour.
- **Long-time Purchasing Intent**: whether a user would buy some goods within the current day.
- **Real-time Cart Intent**: whether a user would add some goods to cart within one hour.
- **Retention Intent**: whether a user would visit the app the next day.

Besides the above global intent, EdgeDIPN also predicts some fine-grained intent such as category preference. Note that a user's category preference is measured by the probability of whether the user would buy some goods belonging to the category. We also set the category preference as real-time and long-time preference, which means the probability of the order being placed within one hour and one week, respectively. In addition, the category is further divided into the top category and leaf category, following the settings of the category label system in Taobao app.

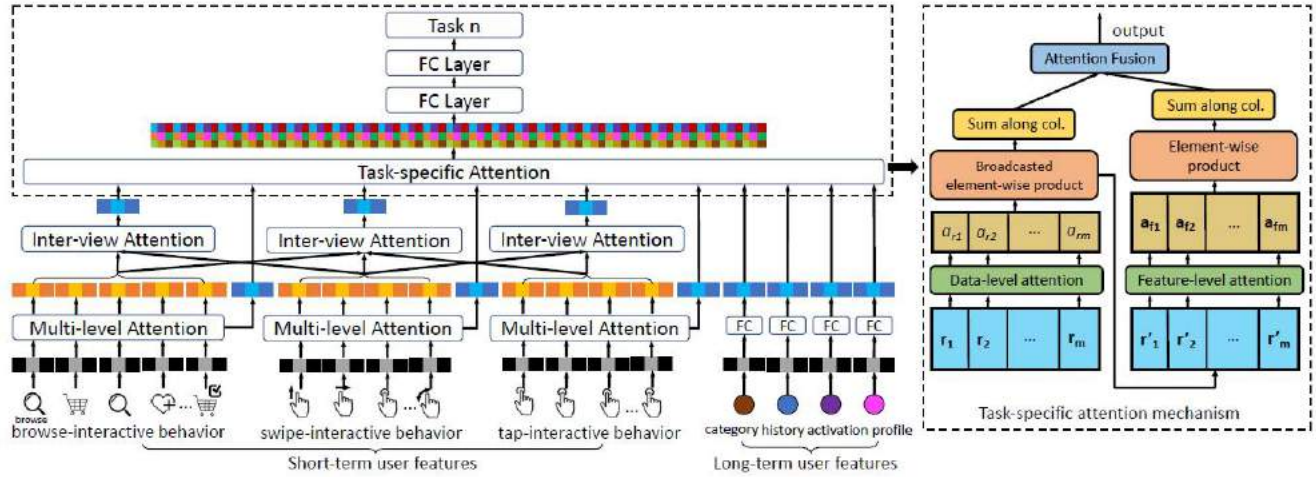


Figure 1: The model architecture of EdgeDIPN.

In our work, each intent is treated as a separate task and we adopt the binary classification for each task. Note that we do not use multi-label classification for the category preference, because the softmax operation would make the model too big to fit at the edge due to the large number of categories to be predicted.

3 MODEL ARCHITECTURE

We propose a unified deep intent prediction network deployed at the edge, named EdgeDIPN, to predict multiple user intent with different granularity simultaneously, as shown in Figure 1.

3.1 Embedding Layer

In the discretization process, the raw values of every feature are encoded into one-hot vectors. Then the one-hot vectors are used as the inputs of EdgeDIPN. As the inputs are high dimensional binary vectors with limited representation capacity, we use the embedding layer to transform them into low dimensional dense representations. The embedding operation follows the table lookup mechanism. In more details, each feature is corresponding to one embedding matrix. For example, the embedding matrix of the *Button Index* feature in the tap-interactive behavior is represented as $\mathrm{E}_{button} = [e_1; e_2; ...; e_{n_b}] \in \mathbb{R}^{n_e \times n_b}$, where $e_i \in \mathbb{R}^{n_e}$ represents an embedding vector with dimension n_e (i.e., 10), and n_b (i.e., 500) represents the number of buttons that a user can tap. The embedding vector of the *Button Index* feature can then be obtained as $\mathrm{e}_{\mathrm{button}} = \mathrm{E}_{button} \cdot \mathrm{b}_{\mathrm{button}} \in \mathbb{R}^{n_e}$, where $b_{button} \in \mathbb{R}^{n_b}$ is the one-hot vector of the *Button Index* feature. In addition, to capture the temporal order information in users' behavior sequence used by the following multi-level attention mechanism, we add the time feature to each action defined as the elapsed time (i.e., seconds) between the moment the action happens and the current moment. However, the model size would become too large after the embedding operation if we directly use the original elapsed time as the input feature. We notice that the impact of the time feature to the current intent decays as the elapsed time increases. Therefore, we propose a new discretization method which would not influence the model performance and can save model space significantly at the same time. Specifically, we split the time feature into multiple granularity, i.e., the continuous time feature in range of $[2^k, 2^{(k+1)})$ is mapped to the discrete feature $(k+1)$ while $[0, 1)$ is mapped to 0. At last, for each action, all the embedded features are concatenated into a vector and fed into a fully-connected layer for reshape. Specifically, each behavior sequence $\mathbf{S}$ can be encoded as a matrix $\mathbf{E}$, i.e., $\mathbf{E} = \{\mathbf{e_1}, \mathbf{e_2}, ..., \mathbf{e_n}\} \in \mathbb{R}^{n \times d_e}$, where d_e is the dimension of $\mathbf{e_i}$ and n is the number of actions contained in $\mathbf{S}$. Moreover, as shown in Figure 1, except for the behavior sequences which can capture the short-term user features, we also collect some statistics for long-term user features.

3.2 Multi-level Attention Layer

The user interactive behaviors are all time sequence of actions. A traditional method is to use RNN to model the long-term dependencies between actions. The adoption of RNN can eliminate the need for extensive feature engineering. However, there are two limitations which hinder us from adopting RNN in EdgeDIPN. First, edge resources are typically resource-constrained and thus places a high demand on the memory cost and inference time of the model. Under this context, RNN is not suitable for use at the edge due to the complex data dependencies and limited parallelism. Second, RNN cannot achieve the best performance because of the unique characteristics of the touch-interactive behavior sequence with plenty of noisy actions operated by users inadvertently.

Therefore, we propose to use the attention mechanism to identify the important actions in the behavior sequence and reduce the interference from the noisy actions. Specifically, we use a softmax layer to determine the weights of different behavior actions. In this case, those actions contributing more to the intent understanding are given larger weights. The weights $\mathbf{a} = \{a_1, a_2, ..., a_n\}$ are computed as follows:

$$a_t = \mathrm{softmax}(\mathbf{h_t}), \mathbf{h_t} = \boldsymbol{w}_{s2}\tanh(\boldsymbol{W}_{s1}\mathbf{e_t}), \quad (1)$$

where $\mathbf{e_t}$ is embedding vector of the t-th action, $\boldsymbol{W}_{s1} \in \mathbb{R}^{d_e \times d_e}$, $\boldsymbol{w}_{s2} \in \mathbb{R}^{d_e}$, and a_t is the weight calculated for $\mathbf{e_t}$. Then we can sum up the behavior sequence $\mathbf{E}$ according to the calculated weights to get a representation vector $\mathbf{r} \in \mathbb{R}^{d_e}$, i.e., $\mathbf{r} = \sum_{i=1}^{n} a_i \mathbf{e_i}$.

However, there is a limitation in the representation vector $\mathbf{r}$, which usually focuses on a specific component of the sequence, like a special set of actions. In reality, there can be multiple components in a sequence that together form the overall preferences of a user. More importantly, EdgeDIPN is proposed to predict multiple user intent and different intent may correspond to different component of the behavior sequence. Therefore, we perform multiple hops of attention and get multiple representation vectors that focus on different component of the behavior sequence, following the idea of the sentence embedding task in [15]. Suppose we want h different components to be extracted from the sequence, we just need to extend $\boldsymbol{w}_{s2}$ into a matrix $\boldsymbol{W}_{s2} \in \mathbb{R}^{h \times d_e}$. Formally, we have

$$\mathbf{A} = \text{softmax}(\boldsymbol{W}_{s2}\tanh(\boldsymbol{W}_{s1}\mathbf{E}^T)),\ \mathbf{R} = \mathbf{AE}, \tag{2}$$

where the softmax() is performed along the second dimension of its input. As a result, the weight vector $\mathbf{a} \in \mathbb{R}^n$ becomes a weight matrix $\mathbf{A} \in \mathbb{R}^{h \times n}$, and the representation vector $\mathbf{r} \in \mathbb{R}^{d_e}$ becomes a embedding matrix $\mathbf{R} \in \mathbb{R}^{h \times d_e}$. After the multi-level attention layer, for each behavior sequence, we can get two outputs, i.e., the representation vector $\mathbf{r_f} \in \mathbb{R}^{d_e}$ by flattening and reshaping $\mathbf{R}$, and the reweighed sequence by applying the element-wise product on $\mathbf{E}$ and $\mathbf{R}$ as follows:

$$\mathbf{r_m} = \text{reduce_mean}(\mathbf{R}),\ \mathbf{V} = \mathbf{E} \odot \mathbf{r_m}. \tag{3}$$

The reweighed sequences are then sent to the following layer.

3.3 Inter-view Attention Layer

To better fuse the views extracted from different behavior sequences, we adopt the inter-view attention mechanism [9] that learns the inter-view relations between different views, as shown in Figure 1. The inter-view attention takes two views as inputs, and emphasizes the important interactions between the two views using attention scores. In particular, for each action in one view, we calculate its relevance with all the actions in the other view. In this way, the asynchronous interactions between actions in these two sequences can be captured effectively. The inter-view attention $\text{IA}(\mathbf{V_s}, \mathbf{V_b})$ is formulated as follows, taking the swipe-interactive view $\mathbf{V_s}$ and the browse-interactive view $\mathbf{V_b}$ for example:

$$\begin{aligned}
\text{IA}(\mathbf{V_s}, \mathbf{V_b}) &= \text{A}_\text{s}(\mathbf{V_b}, \mathbf{V_s}, \mathbf{V_s}) \odot \text{A}_\text{b}(\mathbf{V_s}, \mathbf{V_b}, \mathbf{V_b}), \\
\text{A}_\text{s}(\mathbf{V_b}, \mathbf{V_s}, \mathbf{V_s}) &= \text{softmax}(\frac{\mathbf{V_b}\mathbf{V_s}^T}{\sqrt{2d}})\mathbf{V_s}, \\
\text{A}_\text{b}(\mathbf{V_s}, \mathbf{V_b}, \mathbf{V_b}) &= \text{softmax}(\frac{\mathbf{V_s}\mathbf{V_b}^T}{\sqrt{2d}})\mathbf{V_b},
\end{aligned} \tag{4}$$

where $\text{A}_\text{s} \in \mathbb{R}^{n \times d_e}$ and $\text{A}_\text{b} \in \mathbb{R}^{n \times d_e}$ are attentive representations of $\mathbf{V_s}$ and $\mathbf{V_b}$, respectively, and the element-wise product $\odot$ is used to model the interactions between A_s and A_b. Note that $\text{IA}(\mathbf{V_s}, \mathbf{V_b})$ is a symmetric operation and returns the representation $\mathbf{R_{sb}}$ between $\mathbf{V_s}$ and $\mathbf{V_b}$. We can calculate $\mathbf{R_{st}}$ and $\mathbf{R_{tb}}$ following the same procedure. At last, we can get three representations $\mathbf{r_{sb}} \in \mathbb{R}^{d_e}$, $\mathbf{r_{st}} \in \mathbb{R}^{d_e}$ and $\mathbf{r_{tb}} \in \mathbb{R}^{d_e}$ by applying the reduce_mean operation.

3.4 Multi-task Layer

In this work, we need to monitor multiple user intent simultaneously. Considering the limited computational and memory resources at the edge, it is not reasonable to build a model for each intent separately. Therefore, we propose to train EdgeDIPN with multi-task learning, by which a universal shared representation could be learned and used to derive multiple intent simultaneously In this way, edge resources can be reduced dramatically.

Prior approaches to simultaneously learning multiple tasks commonly concatenate the learned representations from several data sources and send the concatenated representation to the following task-specific output layers [2, 8, 18]. However, we find that different data representations have different contributions in different tasks. Therefore, the performance of each task is highly dependent on an appropriate choice of weighting for each representation. Searching for an optimal weighting is prohibitively expensive and difficult to resolve with manual tuning. Therefore, we propose a novel task-specific attention mechanism to learn the weights for each task automatically and enable each task to pick out the most relevant features from different data representations.

The task-specific attention consists of two components, i.e., data-level attention and feature-level attention, as shown in Figure 1. These two components allow each task the selection of effective data representations and latent features, respectively.

Data-level Attention. The data-level attention is used to learn the weights for each data representation given a task. Suppose we have m representations extracted from different data sources shown as the blue rectangles in Figure 1, i.e., $\mathbf{D} = \{\mathbf{r_1}, \mathbf{r_2}, ..., \mathbf{r_m}\}$, the weight vector $\mathbf{a_r}$ for the given task on different data representations is:

$$\begin{aligned}
\mathbf{a_r} &= \{a_{r1}, a_{r2}, ..., a_{rm}\}, \\
a_{ri} &= \frac{\exp(e(\mathbf{r_i}))}{\sum_{j=1}^{m}\exp(e(\mathbf{r_j}))}, \\
e(\mathbf{r_i}) &= \boldsymbol{w}_{r2}\tanh(\boldsymbol{W}_{r1}\mathbf{r_i} + b_{r1}),
\end{aligned} \tag{5}$$

where $\boldsymbol{W}_{r1} \in \mathbb{R}^{d_e \times d_e}$, $\boldsymbol{w}_{r2} \in \mathbb{R}^{d_e}$, and a_{ri} is the weight calculated for the data representation $\mathbf{r_i}$. After getting the weight vector $\mathbf{a_r} \in \mathbb{R}^m$, we extract a weighed data representation matrix $\mathbf{M}$ by applying the broadcasted element-wise product on $\mathbf{D}$ and $\mathbf{a_r}$.

Feature-level Attention. After getting the weighed data representation $\mathbf{M}$, we go one step further and let each task to pick out the relevant features from $\mathbf{M}$. The intuition behind this idea is that different features in the same data representation may have different impact on different tasks. The feature-level attention is a natural extension of the data-level attention at the feature level. Specifically, instead of computing a single scalar weight for each weighed data representation $\mathbf{r'_i}$, the feature-level attention computes a feature-wise weight vector for $\mathbf{r'_i}$, formalized as follows:

$$\begin{aligned}
\mathbf{a_f} &= \{\mathbf{a_{f1}}, \mathbf{a_{f2}}, ..., \mathbf{a_{fm}}\}, \\
\mathbf{a_{fi}} &= \frac{\exp(e(\mathbf{r'_i}))}{\sum_{j=1}^{m}\exp(e(\mathbf{r'_j}))}, \\
e(\mathbf{r'_i}) &= \boldsymbol{W}_{f2}\tanh(\boldsymbol{W}_{f1}\mathbf{r'_i} + b_{f1}),
\end{aligned} \tag{6}$$

where $\boldsymbol{W}_{f1} \in \mathbb{R}^{d_e \times d_e}$, the vector $\boldsymbol{w}_{r2} \in \mathbb{R}^{d_e}$ in Eq. 5 is replaced with the matrix $\boldsymbol{W}_{f2} \in \mathbb{R}^{d_e \times d_e}$ in Eq. 6, and the softmax() is performed along the second dimension of its input. As a result, we can get a feature-wise weight matrix $\mathbf{a_f} \in \mathbb{R}^{m \times d_e}$. Then the new feature-wise weighted data representation $\mathbf{M'}$ is obtained by applying the element-wise product on $\mathbf{M}$ and $\mathbf{a_f}$, i.e., $\mathbf{M'} = \mathbf{M} \odot \mathbf{a_f}$.

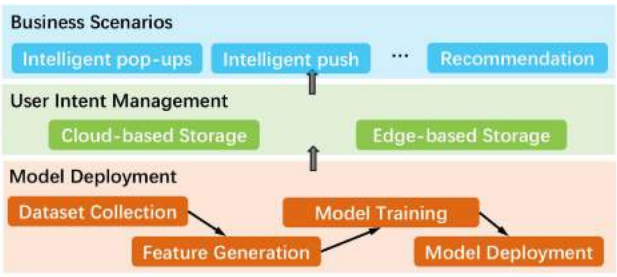


Figure 2: System overview under edge computing.

Attention Fusion. This component is used to combine the two weighted data representation $\mathbf{M}$ and $\mathbf{M}'$, as shown in Figure 1. The combination is accomplished by a dimension-wise fusion gate:

$$\begin{aligned} \mathbf{F} &= \text{sigmoid}(\boldsymbol{W}\mathbf{M} + \boldsymbol{W}'\mathbf{M}' + b), \\ \mathbf{O} &= \mathbf{F} \odot \mathbf{M} + (1 - \mathbf{F}) \odot \mathbf{M}', \end{aligned} \tag{7}$$

where $\boldsymbol{W}, \boldsymbol{W}' \in \mathbb{R}^{d_e \times d_e}$ and $b \in \mathbb{R}^{d_e}$.

For each task, the extracted data representation $\mathbf{O}$ is then fed into two fully connected layers to further learn the combination of task-specific features automatically. Given T binary classification tasks, the loss function of task t_i is:

$$\mathcal{L}_i = -\frac{1}{N} \sum_{(x,y)\in\mathcal{D}}^{N} (y \log p_i(x) + (1-y)\log(1-p_i(x))) \tag{8}$$

where $\mathcal{D}$ is the training set with size D, x is the input of the network and y is the label, and $p_i(x)$ represents the predicted probability of sample x for task t_i. The global loss is:

$$\mathcal{L} = \sum_{i=1}^{T} \mathcal{L}_i + \lambda ||(\mathbf{A}\mathbf{A}^T - \mathbf{I})||_F^2 \tag{9}$$

where the term $||(\mathbf{A}\mathbf{A}^T - \mathbf{I})||_F^2$ is used to punish redundancy between different vectors in the weight matrix $\mathbf{A}$ mentioned in Section 3.2, and thus the multi-level attention mechanism can focus on different component of the behavior sequence.

4 SYSTEM OVERVIEW

The structure of our user intent prediction system is illustrated in Figure 2. For model deployment, the procedure is as follows. First, user behavior data is collected from the mobile app usage logs to construct the dataset. Second, features are generated to build the training samples. Third, EdgeDIPN is trained in the cloud based on the samples. Finally, the trained model is deployed to each user's mobile device. In addition, we build a platform where the model deployment procedure is automatically conducted. As a result, we have built a user intent understanding system in several apps owned by Alibaba for more than a year, such as Taobao, Tmall, Xianyu and Freshippo with little manual work. For user intent management, EdgeDIPN automatically predicts multiple user intent using the real-time features collected from the mobile device at a predefined time interval. The predicted user intent is stored in two forms, i.e., the cloud-based storage where only the predicted intent is sent to the cloud and the edge-based storage where the intent is directly stored at the edge, and used to support the above business scenarios.

5 EXPERIMENTS

5.1 Experimental Setup

Dataset Statistics. We conduct the experiments on a large-scale industrial dataset collected from Taobao. The dataset contains normal users' daily interaction information when using our app. We collect 800,000 users' behaviors for two weeks. For each user, we randomly truncate about 400 groups of samples on average. In total, we obtain 300 million groups of samples. Each group contains the user profile feature, the user statistic feature, and the three behavior sequences (the default length is 256 and padding 0 for short ones). The labels of the multiple tasks are then tagged for each group based on the timestamp of the last action. We use samples in the first 13 days for training and samples in the last day for evaluation.

Compared Methods. We compare EdgeDIPN with the following baseline methods. In particular, we conduct experiments to verify the effect of each component in EdgeDIPN. In the following, we introduce the compared methods briefly.

- **GBDT**, which is the competitive gradient boosting model widely used in the industrial environment.
- **EdgeDIPN-tradition**, which does not use the touch interactive behavior data. Our goal is to see the benefit of using the new touch-interactive behavior.
- **EdgeDIPN-single**, which does not use multi-task learning and train a separate model for each intent.
- **EdgeDIPN-v0**, which replaces the multi-level attention mechanism with the GRU architecture and removes the inter-view attention and task-specific attention mechanism from EdgeDIPN. This method is used to verify the effect of each component in EdgeDIPN.
- **EdgeDIPN-fastgrnn**, which replaces the GRU architecture in EdgeDIPN-v0 with a more advanced RNN architecture named Fastgrnn [13] with smaller model size and lower prediction costs suited to be used at the edge. In addition, we also test the low rank version of Fastgrnn which can further reduce the mode size, denoted as EdgeDIPN-fastgrnnlr.
- **EdgeDIPN-v1**, which replaces GRU in EdgeDIPN-v0 with the multi-level attention mechanism.
- **EdgeDIPN-v2**, which adds the inter-view attention mechanism to EdgeDIPN-v1.
- **EdgeDIPN-v3**, which adds the data-level attention mechanism to EdgeDIPN-v2.
- **EdgeDIPN-v4**, which adds the feature-level attention mechanism to EdgeDIPN-v3 but the feature-level attention mechanism picks out relevant features from the original data representations $\mathbf{D}$ instead of $\mathbf{M}$.
- **EdgeDIPN**, which is the complete model.

Evaluation Metrics. The metric used in our experiments is Area Under the Curve (AUC), which is insensitive to class imbalance and suitable to our experiments. Since EdgeDIPN is used to predict multiple intent simultaneously, we define a metric name for each intent, as shown in Table 1.

Experimental Details. The dimension of embedding features is 10 and the dimension of the reshaped embedding vector is set to $d_e = 64$. The parameters h and λ corresponding to the multi-level attention mechanism are set to 10 and 0.01, respectively. Layers of MLP is set by $128 \times 64 \times 2$. In our experiments, we test 8 binary

Table 1: The metric name of each user intent.

User Intent	Metric Name
Real-time Purchasing Intent	RPAUC
Long-time Purchasing intent	LPAUC
Real-time Cart Intent	RCAUC
Retention Intent	RAUC
Real-time Top Category Preference	RTCAUC
Long-time Top Category Preference	LTCAUC
Real-time Leaf Category Preference	RLCAUC
Long-time Leaf Category Preference	LLCAUC

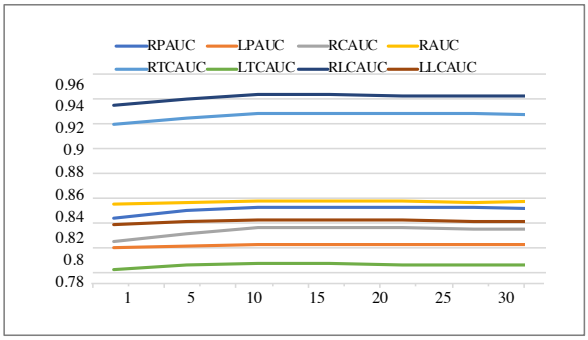


Figure 3: Impact of multiple components.

classification tasks, i.e., $T = 8$. EdgeDIPN is trained with SGD, using the Adam optimizer [12] with initial hyper-parameters of $\epsilon = 10^{-3}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We set the mini-batch size to be 1024 and apply exponential decay, in which the learning rate starts at 0.001 and decay rate is set to 0.9. We train EdgeDIPN using a distributed TensorFlow with 1 parameter server and 200 workers.

5.2 Experimental Results

Results of different models. Table 2 shows the performance of the evaluated models. We have the following observations. (1) EdgeDIPN outperforms the baseline methods EdgeDIPN-tradition on all the tasks by a significant margin between 0.9% and 5.3% in terms of AUC. The improvement of EdgeDIPN over EdgeDIPN-tradition reveals the value of adopting the touch-interactive behavior to depict users from different views. (2) The multi-level attention mechanism is effective in identifying the important actions and reduce the interfence from the noisy actions. We can see that EdgeDIPN-v1 performs better than EdgeDIPN-v0, EdgeDIPN-fastgrnn and EdgeDIPN-fastgrnnlr for all the tasks. (3) The interview attention mechanism plays an important role in EdgeDIPN. As shown, EdgeDIPN-v2 outperforms EdgeDIPN-v1 for all the tasks. (4) With the help of multi-task learning, EdgeDIPN can increase AUC by about 0.85% on average compared with EdgeDIPN-single. This shows the superiority of adopting multi-task learning in EdgeDIPN, which can not only save edge resources dramatically but also improve the performance greatly. In particular, the comparison of EdgeDIPN, EdgeDIPN-v4, EdgeDIPN-v3 with EdgeDIPN-v2 demonstrate the effectiveness of our proposed task-specific attention mechanism. The reason is that different data representations as well as their features play different role in different tasks. By adopting the task-specific attention mechanism, we can let each task choose the most relevant data representations and features, and thus reduce the mutual influence between tasks. We can also see that EdgeDIPN outperforms EdgeDIPN-v4 for all the tasks. This is because the feature-level attention cannot consider the relationship between data representations in EdgeDIPN-v4, which can be solved by adopting the data-level attention first in EdgeDIPN.

Impact of sequence length (Scalability). In this section, we evaluate the impact of different sequence lengths on the performance of the models. The results are shown in Table 3. We evaluate four different models, where Att, RNN, FG and FG-lr denote the model with the the multi-level attention mechanism, GRU, Fastgrnn and Fastgrnnlr structure, repectively. As shown, the prediction performance [1] of each model increases when the sequence length increases. This is because more useful features can be extracted from longer sequences. However, longer sequences result in larger memory cost and longer inference time, which creates a heavy burden for the deployment at the edge. Compared to RNN, Att scales better when the sequence length increases. As shown, when the sequence length is 512, Att can reduce the model size and inference time by 43.9% and 56.2% , respectively. Although FG and FG-lr can reduce the model size and inference time significantly compared with RNN, they perform much worse in terms of AUC in our scenario. Compared with FG and FG-lr, Att not only performs much better in terms of AUC, but also has comparable and even better performance in model size and inference time, which further demonstrates its the effectiveness.

Impact of multiple components. In this section, we evaluate the impact of the parameter h in the multi-level attention mechanism. We vary h from 1 to 30 for each task and the results are shown in Figure 3. We can see the best performance appears after h reaches 10. This demonstrates that there can be multiple components in a sequence that together form the overall preferences of a user. However, it is not always true that a larger h results in better performance. As shown, there is no more benefit when h is larger than 10. Therefore, an appropriate h should be selected.

5.3 Online A/B Testing

5.3.1 Intelligent Pop-ups. Pop-ups are one of the most powerful tools on e-commerce platforms which can reach most of the traffic the platforms receive. If used intelligently, they can play an important role in turning the visitors into sales. In this section, we introduce a novel pop-up strategy based on the real-time category preference predicted by EdgeDIPN in online traffic of Taobao. One important task in Taobao is to promote the conversions of the inactive customers. Since these customers are hard to place an order, we design a special promotion strategy, introduced as follows. We first use the pop-ups to pop some product from the candidate products to the inactive customers. If one customer taps the pop-ups, she would be directed into a shopping page where the products are specially designed for the inactive customers and can promote the conversions effectively. Therefore, one of the key tasks in this strategy is to improve the click-through rate of the pop-ups.

[1] Due to the page space limitation, we only show the AUC for the real-time purchasing intent. The performance is simlar for other intent.

Table 2: Comparison of different models.

Model	RPAUC	LPAUC	RCAUC	RAUC	RTCAUC	LTCAUC	RLCAUC	LLCAUC
GBDT	0.7911	0.7747	0.8057	0.8421	0.8933	0.7826	0.9102	0.8203
EdgeDIPN-tradition	0.7960	0.7856	0.8159	0.8457	0.9069	0.7883	0.9267	0.8263
EdgeDIPN-single	0.8396	0.8068	0.8208	0.8512	0.9241	0.7889	0.9407	0.8286
EdgeDIPN-fastgrnn	0.8367	0.8049	0.8181	0.8496	0.9153	0.7834	0.9291	0.8214
EdgeDIPN-fastgrnnlr	0.8342	0.8037	0.8160	0.8490	0.9105	0.7807	0.9262	0.8185
EdgeDIPN-v0	0.8413	0.8093	0.8191	0.8519	0.9208	0.7895	0.9397	0.8278
EdgeDIPN-v1	0.8435	0.8097	0.8221	0.8522	0.9231	0.7901	0.9418	0.8283
EdgeDIPN-v2	0.8458	0.8106	0.8239	0.8527	0.9266	0.7909	0.9433	0.8292
EdgeDIPN-v3	0.8471	0.8126	0.8271	0.8534	0.9317	0.7947	0.9471	0.8341
EdgeDIPN-v4	0.8482	0.8138	0.8291	0.8538	0.9335	0.7955	0.9499	0.8357
EdgeDIPN	**0.8489**	**0.8152**	**0.8298**	**0.8543**	**0.9346**	**0.7969**	**0.9514**	**0.8368**

Table 3: Impact of sequence length.

Sequence Length	64	128	256	512
RPAUC (RNN)	0.8395	0.8418	0.8463	0.8499
RPAUC (FG)	0.8371	0.8384	0.8419	0.8432
RPAUC (FG-lr)	0.8353	0.8359	0.8396	0.8389
RPAUC (Att)	0.8413	0.8457	0.8489	0.8508
Model Size (RNN)	1.38MB	1.73MB	2.42MB	3.83MB
Model Size (FG)	1.06MB	1.38MB	1.69MB	2.45MB
Model Size (FG-lr)	0.88MB	1.19MB	1.53MB	2.21MB
Model Size (Att)	1.03MB	1.30MB	1.57MB	2.15MB
Infer Time (RNN)	19.5ms	32.9ms	59.8ms	111.6ms
Infer Time (FG)	15.6ms	24.5ms	41.3ms	81.7ms
Infer Time (FG-lr)	14.2ms	23.6ms	35.5ms	59.8ms
Infer Time (Att)	13.7ms	22.6ms	31.2ms	48.9ms

To improve the CTR of the pop-ups, one common method is to pop the hot products to the customers, which is usually a good strategy. A more effective method is to pop the products based on customers' preference. However, the inactive customers have very few history behaviors and thus traditional recommendation methods are not suited to be used in this scenario. The advantage of our EdgeDIPN is that we can utilize users' rich real-time behaviors to predict users' intent. Therefore, we propose to pop the products based on customers' real-time leaf category preference.

We set four pop-ups strategies as follows.

- **Hot Strategy** where the popped product is selected based on the purchase frequency of the products.
- **Category-1 Strategy** where the leaf category ranking first in the category preference is selected and then the hot product belonging to this category is selected.
- **Category-2** and **Category-3 Strategy** are used to compare with **Category-1**, where the leaf category ranking second and third are selected, respectively.

The online A/B testing results are shown in Figure 4, where the x-axis and y-axis represents the date time and the CTR, respectively. It is notable that the category-1 strategy contributes up to **50.1%** CTR promotion in average compared with the hot strategy in the online

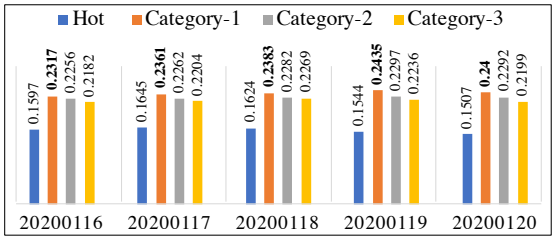


Figure 4: The results of different pop-up strategies.

traffic. The reason is that EdgeDIPN can understand a user's real-time preference by utilizing the rich real-time behaviors conducted by the user, which helps the pop-ups to select more appropriate product. In addition, the category-1 strategy increases the CTR by **4.4%** and **7.3%** compared with the category-2 and category-3 strategy, respectively. This indicates the effectiveness of EdgeDIPN in ranking user category preference.

5.3.2 Intelligent Push. Push notifications are another powerful tool to boost the user engagement on e-commerce platforms. Compared with the pop-ups, push notifications cause less disturbance to users and thus can be used at any moment when a user is visiting the app. In this section, we introduce a novel push notification strategy for the soon-to-expire coupons based on the real-time purchasing intent predicted by EdgeDIPN in online traffic of Taobao.

We choose two different types of coupons (denoted as C_1 and C_2) and set two push strategies to compare the performance.

- **Random Strategy** where the system chooses the moment randomly to push the notification of the soon-to-expire coupons to a user when she is visiting.
- **Model Strategy** where the system uses the score given by EdgeDIPN and a fixed threshold to decide the push moment.

The users selected to get this notification will be pushed a message in Taobao's mobile application. We use the CTR improvement I_C as the evaluation metric, defined as follows:

$$I_C = \frac{CTR_{model} - CTR_{rand}}{CTR_{rand}}, \tag{10}$$

where CTR_{model} and CTR_{rand} are the CTR of the push notifications with model and random strategy, respectively.

Table 4: The results of different push notification strategies.

	Num. of Users	I_{C_1}	I_{C_2}
Random Strategy	18.3M	/	/
Model Strategy	18.3M	**+9.02%**	**+5.32%**

We hypothesize that the users with a low purchasing intent may not be interest in these coupons and would be disturbed by these push notifications. On the contrary, the users with a high purchasing intent may be attracted by these coupons and click the related push notifications. Therefore, we set the threshold $t_l = 0.3$ which has the best online performance. The users whose real-time purchasing intent score given by EdgeDIPN is large than t_l are selected to get this push notification in the model strategy bucket.

As shown in Table 4, the model strategy contributes up to **9.02%** and **5.32%** *CTR* promotion compared with random strategy in the two different types of coupons in the large scale online traffic. The reason is that EdgeDIPN can help the system to understand users' real-time purchasing intent and push the coupon notification to the right person at the right time. Compared with random strategy, a reasonable model strategy relied on EdgeDIPN would result in a significant CTR improvement.

6 RELATED WORK

6.1 Multi-task Learning

Multi-task learning has been used successfully across various applications of machine learning. Ruder [20] presents an overview of multi-task learning in deep learning, where multi-task learning is typically done with either hard or soft parameter sharing of hidden layers. The hard parameter sharing method is the most commonly used multi-task learning approach, which shares the hidden layers between all tasks and keeps several task-specific output layers [4, 5, 8, 18]. The shared-bottom model structure suffers from optimization conflicts caused by task differences. To overcome this issue, Ma et al. [16] propose the MMoE method by sharing the expert submodels across all tasks with a gating network trained to optimize each task. However, MMoE is originally designed for DNNs where the shared-bottom model can be easily partitioned into small expert submodels to reduce the parameter size. This is not suited for EdgeDIPN with a complicated bottom model, which would result in a model with a large size and cannot be deployed at the edge. In soft parameter sharing, each task has its own model with its own parameters where the distance between the parameters is regularized [6, 17, 25]. However, compared to the hard parameter sharing methods, the soft parameter sharing methods have much more task-specific parameters and thus are not suited to be adopted at the edge. Recently, some work [3, 21] try to optimize multi-task learning from the perspective of loss functions. Previous methods usually use a weighed sum of losses, where the loss weights are uniform or manually tuned, and is hard to achieve the best performance. To overcome this issue, Kendall et al. [3] propose to automatically learn the weights by using the homoscedastic task uncertainty, while Sener et al. [21] cast multi-task learning as multi-objective optimization and aim to find a Pareto optimal solution. These methods are orthogonal to our work and can be adopted in EdgeDIPN seamlessly. Compared with previous work, EdgeDIPN proposes a task-specific attention mechanism to let each task pick out the most relevant features from the extracted data representations and thus reduce the conflict between different tasks. Moreover, the task-specific attention mechanism is light-weight and very friendly to be used at the edge.

6.2 Mobile User State Prediction

Another related work is user state prediction based on the mobile phone usage history. In these studies, the usage history is used as features for machine learning models and the targeted user state varies for each study, such as emotion state [26], user trait [22], and mental health [1]. However, none of these studies implement an on-device inference for a real-time prediction [14]. Current on-device applications are mainly image recognition [10] and natural language processing such as smart reply [11]. Recently, Ochiai et al. [19] implement a MLP model on Android smartphone for the smartphone operation troubleshooting recommendation. We are the first to monitor multiple user intent on a large-scale e-commerce platform at the edge. In [9], we make a preliminary attempt at predicting user's real-time purchasing intent with a model named DIPN. EdgeDIPN distinguishes from DIPN in several aspects. First, EdgeDIPN is a full-fledged real-time user intent understanding center and demonstrates its effectiveness in several business scenarios while DIPN just focuses on the purchasing intent prediction. Second, EdgeDIPN replaces the RNN component with the multi-level attention mechanism, which can not only improve prediction performance but also save edge resources. Last but not least, EdgeDIPN proposes a novel task-specific attention mechanism for the multi-task learning.

7 CONCLUSION

In this paper, we propose a unified deep intent prediction network, named EdgeDIPN, which is deployed at the edge and able to monitor multiple user intent with different granularity simultaneously in real-time. To save edge resources, we propose to train EdgeDIPN with multi-task learning by sharing representations among several intent. In addition, several attention mechanisms are proposed to improve the performance of EdgeDIPN. Experimental results on a large-scale industrial dataset shows the superiority of EdgeDIPN. In particular, EdgeDIPN has been deployed in the operational system of Alibaba. Online A/B testing on several business scenarios shows the benefits of monitoring users' intent in real-time.

Note that EdgeDIPN is a universal and scalable architecture which extracts an effective user embedding from various user behaviors and thus new user intent can be easily integrated. In addition, EdgeDIPN is the first full-fledged on-device attempt at the domain of e-commerce, which can offer the opportunity to create more novel business scenarios in the e-commerce platforms.

ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61702016 and 61832001, the National Key Research and Development Program of China (No.2018YFB1004403),Beijing Academy of Artificial Intelligence (BAAI), and Okawa Research Grant.

REFERENCES

[1] B. Cao, L. Zheng, C. Zhang, P. S. Yu, A. Piscitello, J. Zulueta, O. Ajilore, K. Ryan, and A. D. Leow. Deepmood: Modeling mobile phone typing dynamics for mood detection. In *KDD*, page 747–755, 2017.

[2] Y. Chen, Y. Ma, X. Mao, and Q. Li. Multi-task learning for abstractive and extractive summarization. *Data Science and Engineering*, 4(1):14–23, 2019.

[3] R. Cipolla, Y. Gal, and A. Kendall. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, pages 7482–7491, June 2018.

[4] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008.

[5] L. Deng, G. Hinton, and B. Kingsbury. New types of deep neural network learning for speech recognition and related applications: an overview. In *ICASSP*, 2013.

[6] L. Duong, T. Cohn, S. Bird, and P. Cook. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *ACL-IJCNLP*, pages 845–850, 2015.

[7] Z. Gharibshah, X. Zhu, A. Hainline, and M. Conway. Deep learning for user interest and response prediction in online display advertising. *Data Science and Engineering*, 5(1):12–26, 2020.

[8] R. Girshick. Fast r-cnn. In *ICCV*, pages 1440–1448, 2015.

[9] L. Guo, L. Hua, R. Jia, B. Zhao, X. Wang, and B. Cui. Buying or browsing?: Predicting real-time purchasing intent using attention-based deep network with multiple behavior. In *KDD*, page 1984–1992, 2019.

[10] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, and L. V. Gool. Ai benchmark: Running deep neural networks on android smartphones. In *ECCV Workshops*, 2018.

[11] A. Kannan, K. Kurach, S. Ravi, T. Kaufmann, A. Tomkins, B. Miklos, G. Corrado, L. Lukacs, M. Ganea, P. Young, and et al. Smart reply: Automated response suggestion for email. In *KDD*, page 955–964, 2016.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[13] A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma. Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. In *NIPS*, pages 9017–9028. 2018.

[14] S. Li, D. Zhai, P. Du, and T. Han. Energy-efficient task offloading, load balancing, and resource allocation in mobile edge computing enabled iot networks. *Science China Information Sciences*, 62(2):29307, 2018.

[15] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. In *ICLR*, 2017.

[16] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *KDD*, page 1930–1939, 2018.

[17] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. *CoRR*, abs/1604.03539, 2016.

[18] Y. Ni, D. Ou, S. Liu, X. Li, W. Ou, A. Zeng, and L. Si. Perceive your users in depth: Learning universal user representations from multiple e-commerce tasks. In *KDD*, pages 596–605, 2018.

[19] K. Ochiai, K. Senkawa, N. Yamamoto, Y. Tanaka, and Y. Fukazawa. Real-time on-device troubleshooting recommendation for smartphones. In *KDD*, page 2783–2791, 2019.

[20] S. Ruder. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, 2017.

[21] O. Sener and V. Koltun. Multi-task learning as multi-objective optimization. *CoRR*, abs/1810.04650, 2018.

[22] S. Seneviratne, A. Seneviratne, P. Mohapatra, and A. Mahanti. Predicting user traits from a snapshot of apps installed on a smartphone. *SIGMOBILE Mob. Comput. Commun. Rev.*, 18(2):1–8, 2014.

[23] A. Toth, L. Tan, G. D. Fabbrizio, and A. Datta. Predicting shopping behavior with mixture of rnns. In *ACM SIGIR Forum*, 2017.

[24] R. Xu, J. Du, Z. Zhao, Y. He, Q. Gao, and L. Gui. Inferring user profiles in social media by joint modeling of text and networks. *Science China Information Sciences*, 62(11):219104, 2019.

[25] Y. Yang and T. M. Hospedales. Trace norm regularised deep multi-task learning. *CoRR*, abs/1606.04038, 2016.

[26] X. Zhang, W. Li, X. Chen, and S. Lu. Moodexplorer: Towards compound emotion detection via smartphone sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(4), 2018.

[27] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai. Deep interest network for click-through rate prediction. In *KDD*, pages 1059–1068, 2018.

机器学习和计算神经科学领域

NIPS 2020

Hard Example Generation by Texture Synthesis for Cross-domain Shape Similarity Learning

Huan Fu[1*] **Shunming Li**[1*] **Rongfei Jia**[1] **Mingming Gong**[2]

Binqiang Zhao[1] **Dacheng Tao**[3]

[1]TaoXi Technology Department, Alibaba Group, China
[2]The University of Melbourne, Australia
[3]The University of Sydney, Australia

{fuhuan.fh, shunming.lsm, rongfei.jrf, binqiang.zhao}@alibaba-inc.com
mingming.gong@unimelb.edu.au dacheng.tao@sydney.edu.au

Abstract

Image-based 3D shape retrieval (IBSR) aims to find the corresponding 3D shape of a given 2D image from a large 3D shape database. The common routine is to map 2D images and 3D shapes into an embedding space and define (or learn) a shape similarity measure. While metric learning with some adaptation techniques seems to be a natural solution to shape similarity learning, the performance is often unsatisfactory for fine-grained shape retrieval. In the paper, we identify the source of the poor performance and propose a practical solution to this problem. We find that the shape difference between a negative pair is entangled with the texture gap, making metric learning ineffective in pushing away negative pairs. To tackle this issue, we develop a geometry-focused multi-view metric learning framework empowered by texture synthesis. The synthesis of textures for 3D shape models creates hard triplets, which suppress the adverse effects of rich texture in 2D images, thereby push the network to focus more on discovering geometric characteristics. Our approach shows state-of-the-art performance on a recently released large-scale 3D-FUTURE [1] repository, as well as three widely studied benchmarks, including Pix3D [2], Stanford Cars [3], and Comp Cars [4]. Codes will be made publicly available at: https://github.com/3D-FRONT-FUTURE/IBSR-texture.

1 Introduction

Cross-domain image-based 3D shape retrieval (IBSR) is to identify the matched 3D CAD model of the object in a query image. With the growing number of 3D shapes, the establishment of reliable IBSR platforms is significant. For example, it can help to build 3D virtual scenes for real-world houses from 2D images. High-performing IBSR systems may also inspire and benefit studies in shape collection based 3D object reconstruction. Unluckily, compared to content-based image retrieval [5, 6, 7, 8, 9], IBSR is challenging due to the large appearance gap between 3D shapes and 2D images.

There are mainly two directions to alleviate the issue, as shown in Fig. 1. For the first one, researchers take efforts to predict 2.5D sketches from images, such as surface normal, depth, and location filed, to bridge the gaps between 3D and 2D domains [2, 10, 11, 12, 13, 14, 15]. These methods achieve

[*]Equal contribution.

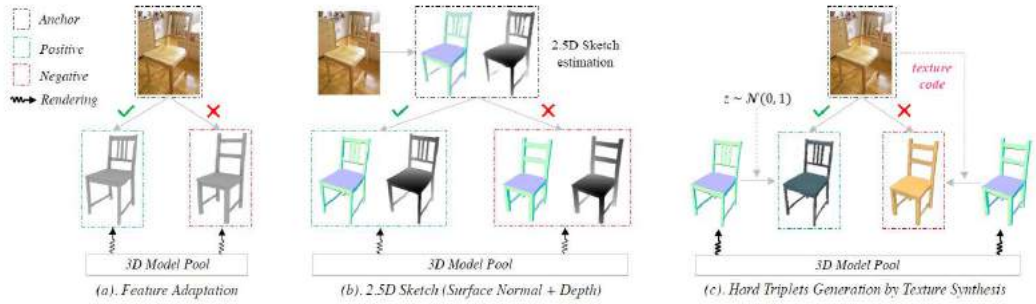


Figure 1: **Motivation.** We show the differences (based on metric learning) between our solution (c) and two widely studied routines ((a) and (b)) for cross-domain image-based 3D shape retrieval. z is a random latent code. Our method assigns the "yellow" texture to the negative 3D chair, and synthesis a random texture ("blue") to the positive 3D chair. Since the anchor image also contain a "yellow" texture, the learning procedure may more easily suppress the adverse effects of creamy texture in 2D images, thereby push the network to focus more on discovering geometric characteristics.

state-of-the-art (SOTA) retrieval accuracy and sequences. However, while 2.5D representations for 3D CAD models can be conveniently rendered, accurately estimating 2.5D sketches from 2D images is itself an open problem [16, 17, 18, 19, 20, 21]. The second routine is to map cross-domain representations into a unified constrained embedding space and define (or learn) a shape similarity measure [22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33]. Standard metric learning with feature adaptation techniques seems to be a natural solution to shape similarity learning. However, their performance is often unsatisfactory for fine-grained shape retrieval. One of the possible reasons is that the shape difference between a negative pair is entangled with the texture gap, making metric learning ineffective in pushing away negative pairs.

As the analysis, assuming that we have many hard triplets like (x, x^+, x^-), where the positive example x^+ and the anchor image x are identical in shape, but differ in texture, the negative example x^- and x are similar in texture, but differ in shape. The metric learning procedure thus should suppress the adverse impact of texture in 2D images, thereby push the network to focus more on discovering geometric characteristics. Towards the goal, we develop a geometry-focused multi-view metric learning framework empowered by texture synthesis. The synthesis of textures for 3D models creates hard triplets. More specifically, we learn a conditional generative adversarial network that can assign a texture to a 3D model based on a texture code encoded from an example image. Then, we can generate hard samples in an online manner to improve the cross-domain shape similarity learning, as shown in Fig 1. Note that, in contrast to category-driven metric learning for image retrieval, each 3D shape is an individual instance (category). It's not easy to apply hard sample mining strategies in image retrieval to IBSR. We are the first to consider texture information to generate hard triplets based on the special properties of the IBSR subject.

Furthermore, the geometry-focused multi-view metric learning framework also contains: (1) a mask attention mechanism to enhance the feature from saliency regions, and (2) a simple but effective viewpoint attention policy to guide the multi-view embedding aggregation. With these guidances, we find that the network can easily focus more on the learning of shape characteristics. Our approach shows state-of-the-art performance on a recently released large-scale 3D-FUTURE [1] repository, as well as three widely studied benchmarks, including Pix3D [2], Stanford Cars [3], and Comp Cars [4]. We also conduct various ablation studies to comprehensively discuss our method.

2 Proposed Method

Our main goal is to learn an encoder that can map both natural query images and 3D shapes to an embedding space. We compute the distances between a query image and the 3D shapes to perform cross-domain retrieval. To match the 3D shapes with the 2D images, we render multiple views (12 views in our paper) of surface normal S for each 3D model using Blender [34]. During the learning procedure, we have N query image and multi-view surface normal pairs $\{(X^i, S^i)\}_{i=1}^N$. Each query image contains a ground-truth mask M, a viewpoint label V, and a category label C (*e.g.,* sofa, chair, and table). Each 3D shape is coupled with a UV atlas (texture map), so that we can also render

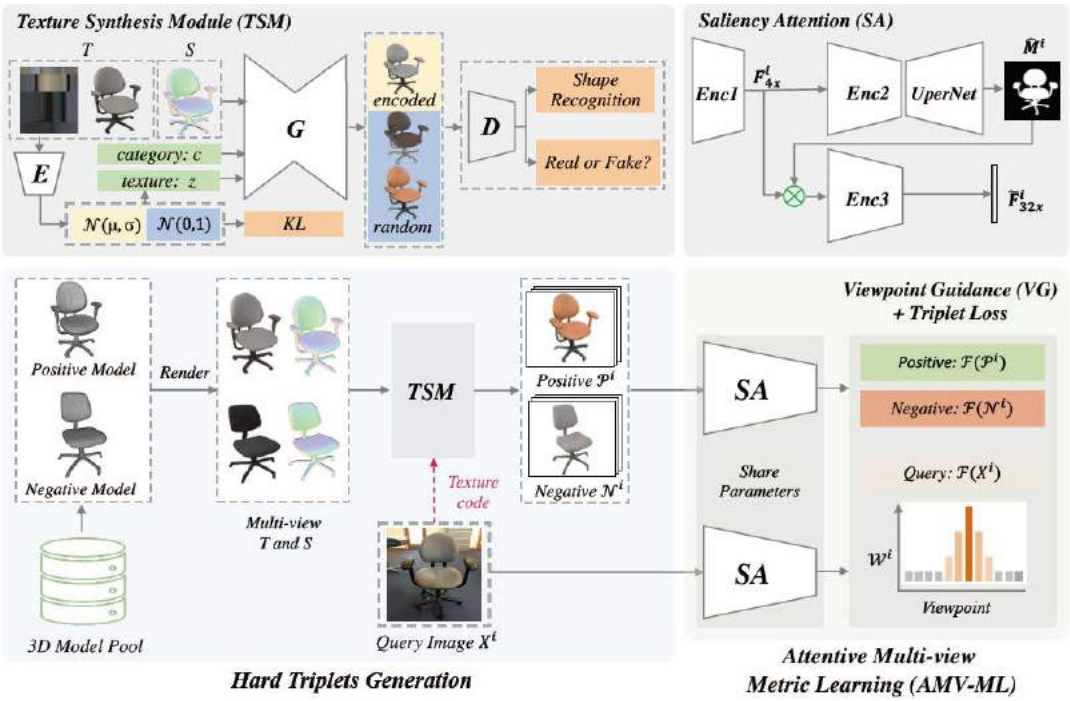


Figure 2: Bottom: The main pipeline of our method. Top: The architectures of some components in our framework. We generate hard triplets in an online manner to empower our shape similarity learning network. See Sec. 2 and Fig. 1 for more details.

synthetic 2D image T (with the corresponding texture) from a random viewpoint. Note that, though public benchmarks [35] provide Shape-UV pairs, there are some reasons that we can not directly recast IBSR as an standard image retrieval problem. Firstly, large amounts of shapes in practice do not have the corresponding UV atlases. For example, while ShapeNetCore [35] contains 51,300 3D models, only several thousands shapes contain UV atlases. Secondly, we concern shape similarities rather than texture similarities in IBSR. However, texture information is a core feature for typical image retrieval subjects. These special properties motivate us to generate hard triplets as shown in Figure 1. Our full framework mainly contains a texture synthesis module (TSM) for hard examples generation and an attentive multi-view metric learning (AMV-ML) process, as shown in Figure 2. In the following, we will first introduce how do we generate hard examples by texture synthesis in an online manner. Then, we will present the attention mechanisms in the metric learning framework.

2.1 Hard Example Generation

Our hard example generation policy relies on texture synthesis. If ignoring the texture consistency on 3D surfaces, texture synthesis can be recast as a special image-to-image (I2I) translation task [36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49]. For instance, VON [50] demonstrated that integrating 2D texture priors into CycleGAN [36] could make controllable texture generation possible. [51, 52, 53, 54, 55] studied conditional GANs for sketch-to-image synthesis according to example texture and color.

Specific to the particular properties of the IBSR subject, we are the first to consider texture information for hard triplets generation to improve the shape similarity learning. Thus, we do not focus on investigating the high-quality UV mapping on a 2D texture atlas and a parameterized 3D mesh. This is out of the scope of the paper. Instead, we intuitively study the objectives of Variational Autoencoder (VAE) and Generative Adversarial Network (GAN) in a hybrid model, and extend it to a conditional scenario (cVAE-GAN) [50, 53] for our purpose. Further, we improve cVAE-GAN to remedy the category-specific training issue and enhance the controlment of the example texture.

Beyond cVAE-GAN. The baseline cVAE is to firstly capture a latent code z from a textured image T by learning a encoder E, then adopt a generator G to obtain $\widetilde{T}$ (a reconstruction of T) from (S, z):

$$z \sim E(T) = Q(z|T), \quad \widetilde{T} \sim G(S, z) = P(T|S, z). \tag{1}$$

Here, S is the rendered surface normal of a specific 3D shape from a random camera pose (6DoF), T is the corresponding 2D image (with texture) under the same pose, and $z \sim E(T)$ is sampled using a re-parameterization trick. GAN [56] performs an minimax game to ensure $\widetilde{T}$ and T distinguishable in both style and content. As an extension, we first render a T' that differs from T in the camera pose, and additional enforce G to reconstruct T from (S, z'). We find that this will intensify the texture guidance of the reference (or example) texture image.

Another issue in cVAE-GAN for texture synthesis is that it may generate textures with unreasonable part segmentation for the objects when we have mixture categories of 3D shapes in our training set. Some cases are shown in Figure 3. It seems like previous works ignore the problem by training category-specific texture synthesis models [50, 53]. We provide a partially solution by injecting a semantic latent code c when learning G, such that $\widetilde{T} \sim G(S, z, c) = P(T|S, z, c)$. We estimate c from S via a pre-trained ResNet-18 for shape classification. The network is fixed during the training of our TSM. Furthermore, we incorporate an auxiliary supervision (shape recognition) on top of the discriminator D to impose the category explainable texture synthesis.

Objectives. As the analysis, the adversarial loss of our TSM takes the form as:

$$\begin{aligned}\mathcal{L}_{adv} = \mathbb{E}_T[\log(D(T))] + 0.5 * \mathbb{E}_{(S,z\sim E(T),c)}[\log(1 - D(G(S, z, c))] \\ + 0.5 * \mathbb{E}_{(S,z'\sim E(T'),c)}[\log(1 - D(G(S, z', c))].\end{aligned} \tag{2}$$

We additionally apply the l_1 loss to enforce the reconstruction process:

$$\mathcal{L}_{rec} = \mathbb{E}_{(T,S,z\sim E(T),c)}[\|G(S, z, c) - T\|_1] + \mathbb{E}_{(T,S,z'\sim E(T'),c)}[\|G(S, z', c) - T\|_1]. \tag{3}$$

To allow texture synthesis from a random latent code at the inference stage, we utilize a Kullback–Leibler (KL) loss to push $E(T)$ to be close to a Gaussian distribution:

$$\mathcal{L}_{KL} = \mathbb{E}_T[\mathcal{D}_{KL}(E(T)\|\mathcal{N}(0, 1))] + \mathbb{E}_{T'}[\mathcal{D}_{KL}(E(T')\|\mathcal{N}(0, 1))], \tag{4}$$

where $\mathcal{D}_{KL}(p\|q) = -\int p(z) \log \frac{p(z)}{q(z)} dz$. We further adopt a cross-entropy loss to mitigate the unexpected cross-category synthesis issue:

$$\begin{aligned}\mathcal{L}_{cls} = \mathbb{E}_T[-\log(D_{cls}(C|T))] + 0.5 * \mathbb{E}_{(S,z,c)}[-\log(D_{cls}(C|G(S, z, c)))] \\ + 0.5 * \mathbb{E}_{(S,z',c)}[-\log(D_{cls}(C|G(S, z', c)))]),\end{aligned} \tag{5}$$

where D_{cls} shares the same parameters with the discriminator D in the convolutional part, and C is the category label of the 3D shape.

The full objective for the texture synthesis module is written as:

$$\mathcal{L}_{TSM} = \mathcal{L}_{adv} + \lambda_{rec}\mathcal{L}_{rec} + \lambda_{KL}\mathcal{L}_{KL} + \lambda_{cls}\mathcal{L}_{cls}. \tag{6}$$

The hyperparameters λ_{rec}, λ_{KL}, and λ_{cls} are set to 10.0, 0.01, and 1.0 in all the experiments.

2.2 Attentive Multi-View Metric Learning (AMV-ML)

For conventional presentation, assuming that we have N hard triplets $\{(X^i, \mathcal{P}^i, \mathcal{N}^i)\}_{i=1}^N$, where $\mathcal{P}^i = G(S^i, z, c^i)$, and $\mathcal{N}^i = G(S^j, E(X^i), c^j)$. Here, S^i pairs up with X^i in shape, and S^j $(j \neq i)$ is a random negative sample from $\{S_k\}_{k=1}^N$. See Sec. 2.1 for details of the texture generator G and other variables. Directly learning a multi-view metric learning based embedding network would be a straightforward solution to IBSR. We go a further step by introducing attention mechanisms to address the background noises and viewpoint variation issues of query images.

Saliency Attention (SA). We introduce a saliency attention (SA) policy to suppress the surrounding noises of objects in natural images. Specifically, we take the ResNet backbone as an encoder, and split it into two parts $Enc1$ and $Enc2$. Here, $Enc1$ is the beginning convolutional blocks until the 4x output layer (*e.g.*, *layer*2 of ResNet serials), and $Enc2$ represents the reminder convolutional blocks. Given a query image X^i, we can obtain several side feature maps $\{F^i_{4x}, F^i_{8x}, F^i_{16x}, F^i_{32x}\} = Enc2(Enc1(X^i))$, where F^i_{4x} is the output of $Enc1(X^i)$. We add a light version of UperNet [57, 58] decoder (Dec) on top of $Enc1$ to predict saliency regions $\hat{M}^i = Dec(F^i_{4x}, F^i_{8x}, F^i_{16x}, F^i_{32x})$ in a supervised manner. The objective is expressed as:

$$\mathcal{L}_{SA} = \frac{-1}{N * H * W} \sum_{i=1}^{N} \sum_{h,w} m^i_{hw} \log(\hat{m}^i_{hw}) + (1 - m^i_{hw}) \log(1 - \hat{m}^i_{hw}), \tag{7}$$

where $H * W$ is the spatial size of $\hat{M}^i$, $\hat{m}^i_{hw} \in \hat{M}^i$ is the probabilistic estimation in position (h, w), and $m^i_{hw} \in M^i$ is the ground truth. All the images are resized to 224×244 in our experiments.

Then, we operate element-wise multiplication for each channels between $F^i_{4\text{x}}$ and $\hat{M}^i$ to capture the attention feature, and feed it to another encoder $Enc3$ to secure the enhanced feature $F^i_{32\text{x}}$:

$$\widetilde{F}^i_{32\text{x}} = Enc3(F^i_{4\text{x}} \otimes \hat{M}^i), \tag{8}$$

where $Enc3$ has the same network structure with $Enc2$. The saliency attention mechanism is simple but effective to enforce the network taking care of region of interest (RoI) for natural query images.

Viewpoint Guidance (VG). We incorporate soft viewpoint guidance into the multi-view metric learning procedure to mitigate the impact of unconstrained viewpoints. Specifically, we simply add an azimuth (viewpoint) classifier (driven by a cross-entropy loss $\mathcal{L}_{view}$) on top of $Enc2$ to capture the probability attention vector $\mathcal{W}^i \in R^V$ from $F^i_{32\text{x}}$. Here, V denotes the predefined bin number of azimuth. Simultaneously, we use an additional convolutional layer followed by an average pooling operation to obtain the embeddings ($\mathcal{F}(X^i) \in R^d, \mathcal{F}(\mathcal{P}^i) \in R^{V \times d}, \mathcal{F}(\mathcal{N}^i) \in R^{V \times d}$) from their corresponding SA features, where d is the feature dimension ($V = 12, d = 256$ in our paper).

Then, we operate $\mathcal{W}^i$ on $\mathcal{F}(\mathcal{P}^i)$ to capture the guided embedding $\widetilde{\mathcal{F}}(\mathcal{P}^i)$ of the positive sample:

$$\widetilde{\mathcal{F}}(\mathcal{P}^i) = \mathcal{F}(\mathcal{P}^i) \otimes \mathcal{W}^i = \sum_{v=1}^{V} \mathcal{F}(p^i_v) * w^i_v, \tag{9}$$

where w^i_v is the vth attention score of $\mathcal{W}^i$. $\mathcal{P}^i = G(S^i, z, c^i)$ contains V view of image $\{q^i_v\}^V_{v=1}$, as S^i is the multi-view surface normal maps of shape i. $\widetilde{\mathcal{F}}(\mathcal{N}^i)$ can be produced in the same manner.

We now have a query image embedding $\mathcal{F}(X^i)$, the positive 3D model embedding $\widetilde{\mathcal{F}}(\mathcal{P}^i)$, and the negative 3D model embedding $\widetilde{\mathcal{F}}(\mathcal{N}^i)$. The objective for the viewpoint guided metric learning is:

$$\mathcal{L}_{VG} = \frac{1}{N} \sum_{i=1}^{N} max(\| f(\mathcal{F}(\mathcal{X}^i)) - f(\widetilde{\mathcal{F}}(\mathcal{P}^i)) \|^2 - \| f(\mathcal{F}(\mathcal{X}^i)) - f(\widetilde{\mathcal{F}}(\mathcal{N}^i)) \|^2 + \alpha,\ 0), \tag{10}$$

where $f(\cdot)$ is a $l2$ norm function as discussed in [5], and $\alpha = 0.1$ is the margin.

Full Objective. Apart from the loss terms mentioned above, we utilize an instance classification loss $\mathcal{L}_{inst}$ to capture shape similarity among 3D CAD instances, as previously [5, 15]. The full objective of our attentive multi-view metric learning takes the form as:

$$\mathcal{L}_{AMV-ML} = \mathcal{L}_{inst} + \lambda_1 \mathcal{L}_{view} + \lambda_2 \mathcal{L}_{SA} + \lambda_3 \mathcal{L}_{VG}, \tag{11}$$

where $\lambda_1 \sim \lambda_3$ are the trade-off hyperparameters. According to the difficulty level of optimizing each task, we set $\lambda_1 = 0.5$, $\lambda_1 = 1.0$, $\lambda_3 = 3.0$ in all of our experiments.

3 Experiments

In the section, we conduct extensive experiments to demonstrate the superiority of our method. We first present the implementation details, then make comparisons against previous works on four challenging benchmarks including 3D-FUTURE [1], Pix3D [2], Stanford Cars [3], and Comp Cars [4]. We do not evaluate our method on PASCAL3D+ [59] and ObjectNet3D [60] since the 3D models are not matched the objects in the images as stated in [2]. Finally, we examine various ablations to comprehensively discuss our method.

3.1 Implementation Details

We refer the supplemental materials for more implementation details. Codes will be shared.

Network Architecture. For the texture synthesis module (TSM), the encoder E consists of five residual blocks followed by a liner layer. The generator G follows the same architecture as U-Net [61]. We take PatchGANs [37] as our discriminator D to identify in two scales (70×70 and 140×140). The shape classifier D_{cls} shares the same parameters with D in the convolutional part. We add the

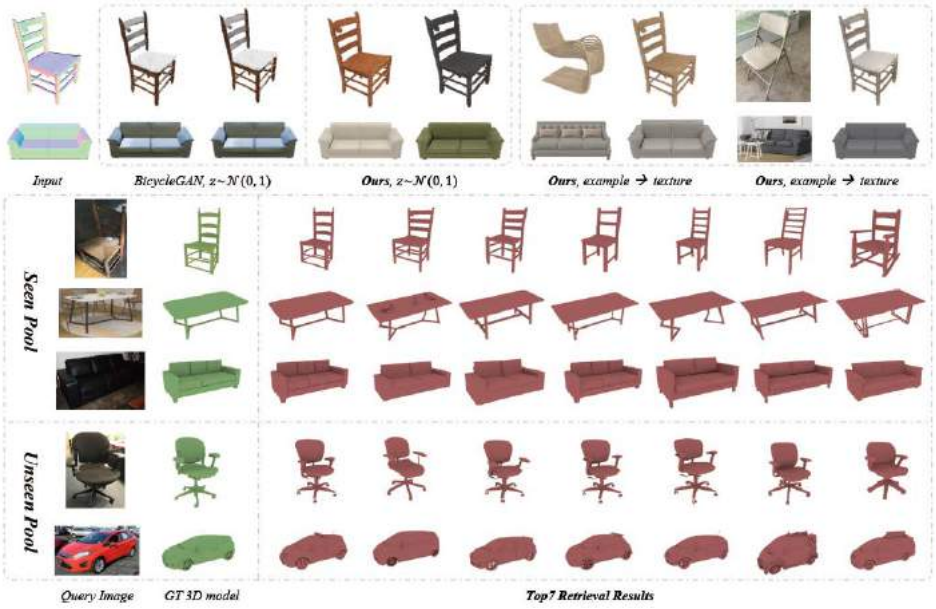


Figure 3: **Qualitative Results.** Top: Texture synthesis from random codes or example images. Top Right: Texture synthesis from example images. Bottom: Shape retrieval. BicycleGAN [41] is a SOTA method for supervised multimodal I2I translation. Unseen Pool: ShapeNet [35] Pool. Zoom in for better view.

latent code z and c to each intermediate layer of G following [62]. For the attentive multi-view metric learning (AMV-ML) part, we take the convolutional part of ResNet-34 as the backbone.

Training Details. We first train TSM, and utilize the learned G to generate hard triplets in an online manner. We use the Adam solver [27] with a learning rate of 0.0002 and coefficients of (0.5, 0.999). The latent dimension $|z|$ is set to 8, and $|c|$ is the number of shape categories. We train the network with a image size of 256×256 and with a batch size of 16. For the AMV-ML procedure, we train the network in two stages, *i.e.,* a warm-up stage with $\mathcal{L}_{inst}$ for 20 epochs and a fine-tuning stage with the objective $\mathcal{L}_{AMV-ML}$ for another 60 epochs. We use the SGD solver with a beginning learning rate of 0.002, the momentum of 0.9, and the weight decay of 0.0004. The learning rate is fixed in the initial $10 \setminus 20$ epochs, and linearly decays to zero over the next $10 \setminus 40$ epochs. The model is trained with a batch size of 24 and with a image size of 224×224.

Evaluation. For evaluation, we render multi-view images without texture $\hat{S}$ for each 3D model contained in the 3D pool. The distance between a query image X^i and a 3D shape j can be computed as $\sum_{i=1}^{V} w_v^i * (\mathcal{F}(X^i) \odot \mathcal{F}(\hat{s}_v^j))$, where $\hat{s}_v^j$ is the vth view image of $\hat{S}^j$. See Sec. 2.2 (VG) for details. We use TopK Recall as our primary quantitative measure, while also report HAU (mean modified Hausdorff Distance) and IoU (Intersection Over Union) as suggested in [15].

3.2 Benchmark Performance

Pix3D. Pix3D [2] offers 10,069 images with 395 3D models. Following [15], we conduct experiments on categories that contain more than 300 non-occluded and non-truncated samples, *i.e.,* bed, chair, sofa, and table. This results in 5,118 images and 322 shapes, with 2,648 for training and 2,470 for test. We introduce 18,648 3D models from ShapeNet [35] as the 3D pool for more challenging evaluation.

Stanford and Comp Cars. Stanford Cars [3] focuses on more challenging fine-grained retrieval, where 134 3D car models with 16,185 images are provided. The images are officially split into the train (8,144) and test (8,041) sets. Comp Cars [4] shows similar characteristics as Stanford Cars. There are 94 3D car models with 5,696 images (3,798 for training and 1,898 for test). The 7,497 3D car models in ShapeNet [35] are used to demonstrate the robustness of our method.

3D-FUTURE. 3D-FUTURE [1] is a recently released 3D furniture repository. We study the IBSR subject on the six furniture categories, *i.e.,* sofa, bed, table, chair, cabinet, and stool. We have 25,913

Table 1: **Benchmark Performance.** For all the benchmarks, we train a single model for all categories (not category-specifc training). For Pix3D, We report both the category-specific scores and the mean scores as [15]. Here, "mean" represents the average scores of each category. For 3D-FUTURE, we conduct the sketch (Fig. 1) based method as our baseline. "mean*" shows the general performance over all the shapes (without considering the categories). Unseen Pool represents the ShapeNet [35] Pool.

Method	Category	Seen Pool				Unseen Pool	
		Top1@R	Top10@R	HAU	IoU	HAU	IoU
Pix3D							
UDF-CGI [31]	bed	19.4%	46.6%	0.0821	0.3397	0.0960	0.2487
Grabner *et al.* [27]		35.1%	83.2%	0.0385	0.5598	0.0577	0.3013
LFD [15]		64.4%	89.0%	0.0152	0.8074	0.0448	0.3490
Ours		**65.3%**	**95.4%**	**0.0122**	**0.8213**	**0.0425**	**0.3684**
UDF-CGI [31]	chair	17.3%	49.1%	0.0559	0.3027	0.0843	0.1334
Grabner *et al.* [27]		41.3%	73.9%	0.0305	0.5469	0.0502	0.1965
LFD [15]		58.1%	81.8%	0.0170	0.7169	0.0375	0.2843
Ours		**87.9%**	**97.9%**	**0.0041**	**0.9063**	**0.0152**	**0.7482**
UDF-CGI [31]	sofa	21.7%	52.2%	0.0503	0.3824	0.0590	0.3493
Grabner *et al.* [27]		44.1%	89.8%	0.0197	0.7762	0.0294	0.6178
LFD [15]		67.0%	94.4%	0.0075	0.9028	0.0178	0.7472
Ours		**72.8%**	**97.7%**	**0.0047**	**0.9070**	**0.0156**	**0.7963**
UDF-CGI [31]	table	12.0%	34.2%	0.1003	0.1715	0.1239	0.1047
Grabner *et al.* [27]		33.9%	66.1%	0.0607	0.4500	0.0753	0.1730
LFD [15]		53.3%	80.1%	0.0288	0.6383	0.0482	0.2573
Ours		**73.7%**	**92.4%**	**0.0170**	**0.7667**	**0.0228**	**0.4391**
UDF-CGI [31]	mean	17.6%	45.5%	0.0722	0.2991	0.0908	0.2090
Grabner *et al.* [27]		38.6%	78.3%	0.0374	0.5832	0.0531	0.3222
LFD [15]		60.7%	86.3%	0.0171	0.7663	0.0370	0.4095
Ours		**74.9%**	**95.8%**	**0.0095**	**0.8503**	**0.0240**	**0.6081**
Stanford Cars							
UDF-CGI [31]	car	3.7%	20.1%	0.0198	0.7169	0.0242	0.6526
Grabner *et al.* [27]		11.3%	42.2%	0.0153	0.7721	0.0183	0.7201
LFD [15]		29.5%	69.4%	0.0110	0.8352	0.0150	0.7744
Ours		**68.4%**	**92.1%**	**0.0034**	**0.9210**	**0.0074**	**0.8735**
Comp Cars							
UDF-CGI [31]	car	2.4%	18.2%	0.0207	0.7224	0.0271	0.6344
Grabner *et al.* [27]		10.2%	36.9%	0.0158	0.7805	0.0194	0.7230
LFD [15]		20.5%	58.0%	0.0133	0.8142	0.0165	0.7707
Ours		**67.1%**	**93.7%**	**0.0035**	**0.9256**	**0.0092**	**0.8591**
3D-FUTURE (Mixed Pool)							
Baseline	mean*	31.8%	65.4%	0.05160	0.4915	-	-
Ours		**69.1%**	**91.7%**	**0.0198**	**0.7890**	-	-

images and 4,662 3D shapes for our train set, and 5,865 images and 886 3D shapes for the validation set. In contrast to other datasets, the CAD models in 3D-FUTURE's validation set are unseen in the training stage. In the evaluation stage, we perform retrieval for the 5,865 images from the whole 5,548 (4,662 + 886) model pool. Comparisons on 3D-FUTURE are convincing due to its large 3D pool and the challenging setting.

Performance. The scores are reported in Table 1. Generally, our method acquires significantly higher accuracy compared against baselines and SOTA methods over all the metrics. For Pix3D, it (74.9%) outperforms the SOTA (60.7%) by a large margin. We notice that our approach captures similar scores as LFD [15] on the bed and sofa categories. The reason is that there are only 19 3D beds and 20 3D sofas in the pool. For Car benchmarks, our method can accurately identify the best-matched cars with an impressive Top1@R of 67.1% $\sim$ 68.4%, while previous SOTA scores are less than 30.0%. For the challenging scenarios on 3D-FUTURE, our method still yields a promising Top1 retrieval accuracy of 69.1%. Moreover, we evaluate our trained models on a large-scale unseen pool (ShapeNet). Our approach captures preferable IoU and HAU compared with previous works,

Table 2: **Hard Example Generation by Texture Synthesis.** The two baselines (Adaptation and 2.5D-Sketches) are illustrated in Fig. 1. For a fair comparison, all the experiments here are developed based on our AMV-ML in Sec. 2.2. Benefiting from AMV-ML, all the compared methods here performs much better than SOTA methods.

Method	Top1@R	Top10@R	HAU	IoU
Pix3D (322 3D CAD models)				
Adaptation	68.5%	94.5%	0.0118	0.8050
2.5D-Sketches	69.8%	96.4%	0.0104	0.8225
Ours	**74.9%**	**95.8%**	**0.0095**	**0.8503**
Comp and Stanford Cars (210 3D Cars)				
Adaptation	56.5%	87.1%	0.0035	0.9020
2.5D-Sketches	36.9%	69.3%	0.0078	0.8161
Ours	**68.3%**	**93.2%**	**0.0028**	**0.9273**
3D-FUTURE (5,548 3D CAD models)				
Adaptation	63.1%	90.4%	0.0242	0.7471
2.5D-Sketches	31.8%	65.4%	0.0516	0.4915
Ours	**69.1%**	**91.7%**	**0.0198**	**0.7890**

Table 3: **Performance Gains of AMV-ML.** $\surd$: Optimizing the network via the selected loss. Gain: Top1@R gains compared with the baseline. We take $\mathcal{L}_{inst}$ as the baseline. $\mathcal{L}_{VG}$ and $\mathcal{L}_{view}$ are coupled, because $\mathcal{L}_{VG}$ relies on $\mathcal{L}_{view}$. See Sec. 2.2 for details of these loss items.

$\mathcal{L}_{inst}$	$\mathcal{L}_{SA}$	$\mathcal{L}_{view}$ + $\mathcal{L}_{VG}$	Top1@R	HAU	IoU	Gain
$\surd$			56.3%	0.0270	0.7077	-
$\surd$	$\surd$		63.4%	0.0234	0.7456	+7.1%
$\surd$		$\surd$	61.4%	0.0238	0.7354	+5.1%
$\surd$	$\surd$	$\surd$	**69.1%**	**0.0198**	**0.7890**	+12.8%

which shows the generalization (or robustness) of our models. We can conclude from the surprising performance that hard sample generation by texture synthesis may be a potential venue for IBSR.

3.3 Ablation Studies

Hard Example Generation. We are the first to study hard triplets generation by texture synthesis for IBSR specific to its particular properties. We take Adaptation and 2.5D-Sketches in Fig. 1 as our compared methods. They can represent the two widely studied frameworks for IBSR. Adaptation here can be seen as a weakened version of our approach, where the positive and negative samples are textureless renderings. For 2.5D-Sketches, we firstly predict 2.5D sketches (*i.e.*, surface normal and depth) for anchor images via UperNet-50 [63]. Then, we render multi-view 2.5D sketches for each 3D CAD model. Finally, the positive and negative examples are 2.5D sketches. We combine Stanford Cars and Comp Cars to obtain an enlarged fine-grain car dataset with 210 (134+94-22) 3D cars. There are 22 identical 3D cars.

From Table 2, our method ranks 1st over all the metrics. In detail, our approach improves Adaptation by promising margins on all the scenarios (6.0%∼11.8%). The observation greatly supports our motivation that exploiting hard example generation by texture synthesis would improve the shape similarity learning for IBSR. Besides, 2.5D-Sketches may suffer from inaccurate dense predictions. It shows a performance gap compared with Adaptation and our methods in fine-grained retrieval. Finally, we find that L_{cls} can help generate more visually appealing translations, but only improve the Top1 retrieval accuracy by 2.0% on 3D-FUTURE. One of the possible reasons is that when considering hard triplets or suppressing the adverse impact of texture like our method, the color information may be more important than the semantics of the texture.

Attentive Multi-view Metric Learning (AMV-ML). We discuss several baselines in Table 3 and Table 4 to study our AMV-ML on 3D-FUTURE. From the scores in Table 3, we can generally conclude that the proposed attention mechanisms are potent in remedying the background and viewpoint variation issues. From the top part of Table 4, the simple saliency attention mechanism can effectively improve the baseline by 7.7%. Coming to the bottom part, the proposed Viewpoint Guidance (VG) (69.1%) is a better option compared with Mean-Pooling (66.1%) and Max-Pooling

Table 4: **Attention Mechanisms in AMV-ML.** Our hard triplets generation policy is used in all the experiments. †: We take a mean pooling operation on the multi-view embedding of a 3D shape as its shape embedding (only in the evaluation phase). Mean-Pooling \ Max-Pooling: The embedding for a 3D shape is obtained by the mean \ max pooling operation in both training and evaluation stages.

Method	Top1@R	HAU	IoU	Gain
Saliency Attention ($\mathcal{L}_{SA}$)				
AMV-ML w/o L_{SA}	61.4%	0.0238	0.7354	-
AMV-ML	**69.1**%	**0.0198**	**0.7980**	+7.7%
Viwepoint Guidance ($\mathcal{L}_{view} + \mathcal{L}_{VG}$)				
AMV-ML† w/o VG	63.4%	0.0234	0.7456	-
Mean-Pooling	66.1%	0.0218	0.7685	+2.7%
Max-Pooling	64.7%	0.0224	0.7572	+1.3%
AMV-ML	**69.1**%	**0.0198**	**0.7980**	**+5.7**%
AMV-ML†	67.2%	0.0214	0.7758	+3.7%

(64.7%) in mitigating the influence of unconstrained viewpoints. Note that, [2] have demonstrated that directly adding pose supervisions to their network would largely degrade their retrieval performance. Here, we partially rectify the issue via our viewpoint guidance policy. Furthermore, we find that AMV-ML† (a degraded version) has a promising Top1@R of 67.2%. This observation shows that our framework is robust in enforcing the embedding network to capture the highlight shape characteristics.

Moreover, we discuss the impact of viewpoint distributions for the viewpoint guidance mechanism. For the indoor benchmarks (Pix3D and 3D-FUTURE), most of images (70.3%) are under the front viewpoints. The elevations mainly range from 20 to 45 degrees. The viewpoint distribution makes sense since most people prefer to captured indoor images from front views. For the outdoor Car benchmarks [3,4], the azimuth distribution follows a uniform distribution. We evaluate our trained models on the “Front" (5 views) and “Back" (5 views) viewpoints, while remove the east and west views. We find that our models are not sensitive to the azimuth distribution, though the azimuths of the training images are biased towards “Front" (Pix3D: 79.5% vs. 75.3%, 3D-FUTURE: 69.4% vs. 69.4%). Besides, some extremely unpreferable evaluates may decrease the retrieval performance, but our models show robustness in most cases.

4 Conclusion

In the paper, we studied metric learning for cross-domain image-based shape retrieval (IBSR). We found that the shape difference between a negative pair is entangled with the texture gap, making metric learning ineffective in pushing away negative pairs. We thus proposed to create hard triplets by texture synthesis, and developed a geometry-focused multi-view metric learning framework to tackle the issue. Our method effectively suppresses the impact of texture, thereby push the network to focus more on discovering geometric characteristics. Retrieval performances on different challenging benchmarks, including 3D-FUTURE, Pix3D, Stanford Cars, and Comp Cars, demonstrate the superiority of our approach. We also conduct various ablations to discuss our method more comprehensively. Our solution may be a potent venue for high-performing IBSR studies.

Broader Impact Statement

Based on our knowledge, our work may not have an adverse impact on ethical aspects and future societal consequences. With the growing number of 3D shapes, the studies of cross-domain image-based shape retrieval (IBSR) is significant. We thus believe our work in this paper may have a positive impact on related subjects and techniques. For example, it can help to build 3D virtual scenes for real-world houses by accurately identify the exact 3D shapes contained in the captured 2D scene images. Furthermore, designers may be able to develop their required 3D CAD models based on the retrieved highly similar 3D shapes instead of drawing 3D shapes from scratches. High-performing IBSR systems may also inspire and benefit studies in 3D object reconstruction from large-scale shape collections.

Acknowledgments

Dacheng Tao is supported by Australian Research Council Project FL-170100117.

References

[1] H. Fu, R. Jia, L. Gao, M. Gong, B. Zhao, S. Maybank, and D. Tao, "3d-future: 3d furniture shape with texture," *arXiv preprint arXiv:2009.09633*, 2020.

[2] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman, "Pix3d: Dataset and methods for single-image 3d shape modeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2974–2983, 2018.

[3] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, (Sydney, Australia), 2013.

[4] Y. Wang, X. Tan, Y. Yang, X. Liu, E. Ding, F. Zhou, and L. S. Davis, "3d pose estimation for fine-grained object categories," in *European Conference on Computer Vision*, pp. 619–632, Springer, 2018.

[5] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742, 2018.

[6] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in *European conference on computer vision*, pp. 584–599, Springer, 2014.

[7] F. Radenović, G. Tolias, and O. Chum, "Fine-tuning cnn image retrieval with no human annotation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 7, pp. 1655–1668, 2018.

[8] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2064–2072, 2016.

[9] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 132–149, 2018.

[10] S. Huang, S. Qi, Y. Zhu, Y. Xiao, Y. Xu, and S.-C. Zhu, "Holistic 3d scene parsing and reconstruction from a single rgb image," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 187–203, 2018.

[11] J. Wu, Y. Wang, T. Xue, X. Sun, B. Freeman, and J. Tenenbaum, "Marrnet: 3d shape reconstruction via 2.5 d sketches," in *Advances in neural information processing systems*, pp. 540–550, 2017.

[12] A. Bansal, B. Russell, and A. Gupta, "Marr revisited: 2d-3d alignment via surface normal prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5965–5974, 2016.

[13] C. W. Bachman, "Data processing system utilizing data field descriptors for processing data files," Jan. 10 1978. US Patent 4,068,300.

[14] A. Grabner, P. M. Roth, and V. Lepetit, "3d pose estimation and 3d model retrieval for objects in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3022–3031, 2018.

[15] A. Grabner, P. M. Roth, and V. Lepetit, "Location field descriptors: Single image 3d model retrieval in the wild," in *2019 International Conference on 3D Vision (3DV)*, pp. 583–593, IEEE, 2019.

[16] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE international conference on computer vision*, pp. 2650–2658, 2015.

[17] X. Wang, D. Fouhey, and A. Gupta, "Designing deep networks for surface normal estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 539–547, 2015.

[18] W. Chen, Z. Fu, D. Yang, and J. Deng, "Single-image depth perception in the wild," in *Advances in neural information processing systems*, pp. 730–738, 2016.

[19] X. Qi, R. Liao, Z. Liu, R. Urtasun, and J. Jia, "Geonet: Geometric neural network for joint depth and surface normal estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 283–291, 2018.

[20] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2002–2011, 2018.

[21] P. Wang, X. Shen, B. Russell, S. Cohen, B. Price, and A. L. Yuille, "Surge: Surface regularized geometry estimation from a single image," in *Advances in Neural Information Processing Systems*, pp. 172–180, 2016.

[22] Y. Li, H. Su, C. R. Qi, N. Fish, D. Cohen-Or, and L. J. Guibas, "Joint embeddings of shapes and images via cnn image purification," *ACM transactions on graphics (TOG)*, vol. 34, no. 6, p. 234, 2015.

[23] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic, "Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3762–3769, 2014.

[24] T. Lee, Y.-L. Lin, H. Chiang, M.-W. Chiu, W. Hsu, and P. Huang, "Cross-domain image-based 3d shape retrieval by view sequence learning," in *2018 International Conference on 3D Vision (3DV)*, pp. 258–266, IEEE, 2018.

[25] F. Massa, B. C. Russell, and M. Aubry, "Deep exemplar 2d-3d detection by adapting from real to rendered views," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6024–6033, 2016.

[26] F. P. Tasse and N. Dodgson, "Shape2vec: semantic-based descriptors for 3d shapes, sketches and images," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, p. 208, 2016.

[27] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta, "Learning a predictable and generative vector representation for objects," in *European Conference on Computer Vision*, pp. 484–499, Springer, 2016.

[28] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893, IEEE, 2005.

[29] H. Izadinia, Q. Shan, and S. M. Seitz, "Im2cad," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5134–5143, 2017.

[30] S. Tulsiani, S. Gupta, D. F. Fouhey, A. A. Efros, and J. Malik, "Factoring shape, pose, and layout from the 2d image of a 3d scene," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 302–310, 2018.

[31] M. Aubry and B. C. Russell, "Understanding deep features with computer-generated imagery," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2875–2883, 2015.

[32] K. Z. Lin, W. Xu, Q. Sun, C. Theobalt, and T.-S. Chua, "Learning a disentangled embedding for monocular 3d shape retrieval and pose estimation," *arXiv preprint arXiv:1812.09899*, 2018.

[33] B.-S. Hua, Q.-T. Truong, M.-K. Tran, Q.-H. Pham, A. Kanezaki, T. Lee, H. Chiang, W. Hsu, B. Li, Y. Lu, H. Johan, S. Tashiro, M. Aono, M.-T. Tran, V.-K. Pham, H.-D. Nguyen, V.-T. Nguyen, Q.-T. Tran, T. V. Phan, B. Truong, M. N. Do, A.-D. Duong, L.-F. Yu, D. T. Nguyen, and S.-K. Yeung, "RGB-D to CAD Retrieval with ObjectNN Dataset," in *Eurographics Workshop on 3D Object Retrieval* (I. Pratikakis, F. Dupont, and M. Ovsjanikov, eds.), The Eurographics Association, 2017.

[34] "Blender." `https://www.blender.org`.

[35] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[36] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[37] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[38] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in neural information processing systems*, pp. 700–708, 2017.

[39] S. Benaim and L. Wolf, "One-sided unsupervised domain mapping," in *Advances in neural information processing systems*, pp. 752–762, 2017.

[40] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 172–189, 2018.

[41] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, "Toward multimodal image-to-image translation," in *Advances in Neural Information Processing Systems*, pp. 465–476, 2017.

[42] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8789–8797, 2018.

[43] A. Palazzi, L. Bergamini, S. Calderara, and R. Cucchiara, "Warp and learn: Novel views generation for vehicles and other objects," *arXiv preprint arXiv:1907.10634*, 2019.

[44] W. Xian, P. Sangkloy, V. Agrawal, A. Raj, J. Lu, C. Fang, F. Yu, and J. Hays, "Texturegan: Controlling deep image synthesis with texture patches," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[45] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros, "Real-time user-guided image colorization with learned deep priors," *arXiv preprint arXiv:1705.02999*, 2017.

[46] Y. Liu, Z. Qin, Z. Luo, and H. Wang, "Auto-painter: Cartoon image generation from sketch by using conditional generative adversarial networks," *arXiv preprint arXiv:1705.01908*, 2017.

[47] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. S. Kweon, "Pixel-level domain transfer," in *European Conference on Computer Vision*, pp. 517–532, Springer, 2016.

[48] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *European conference on computer vision*, pp. 702–716, Springer, 2016.

[49] H. Fu, M. Gong, C. Wang, K. Batmanghelich, K. Zhang, and D. Tao, "Geometry-consistent generative adversarial networks for one-sided unsupervised domain mapping," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2427–2436, 2019.

[50] J.-Y. Zhu, Z. Zhang, C. Zhang, J. Wu, A. Torralba, J. B. Tenenbaum, and W. T. Freeman, "Visual object networks: Image generation with disentangled 3D representations," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[51] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays, "Scribbler: Controlling deep image synthesis with sketch and color," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[52] W. Xian, P. Sangkloy, V. Agrawal, A. Raj, J. Lu, C. Fang, F. Yu, and J. Hays, "Texturegan: Controlling deep image synthesis with texture patches," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8456–8465, 2018.

[53] M. Oechsle, L. Mescheder, M. Niemeyer, T. Strauss, and A. Geiger, "Texture fields: Learning texture representations in function space," in *Proceedings IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[54] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt, and Y.-L. Yang, "Hologan: Unsupervised learning of 3d representations from natural images," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7588–7597, 2019.

[55] J. Kossaifi, L. Tran, Y. Panagakis, and M. Pantic, "Gagan: Geometry-aware generative adversarial networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[56] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[57] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," *International Journal on Computer Vision*, 2018.

[58] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[59] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond pascal: A benchmark for 3d object detection in the wild," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014.

[60] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese, "Objectnet3d: A large scale database for 3d object recognition," in *European Conference on Computer Vision*, pp. 160–176, Springer, 2016.

[61] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[62] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[63] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 418–434, 2018.

Neuron-level Structured Pruning using Polarization Regularizer

Tao Zhuang[1], **Zhixuan Zhang**[*1], **Yuheng Huang**[2], **Xiaoyi Zeng**[1], **Kai Shuang**[2], **Xiang Li**[1]
[1]Alibaba Group
[2]Beijing University of Posts and Telecommunications
{zhuangtao.zt, zhibing.zzx}@alibaba-inc.com, hyhlryf@bupt.edu.cn,
yuanhan@taobao.com, shuangk@bupt.edu.cn, leo.lx@alibaba-inc.com

Abstract

Neuron-level structured pruning is a very effective technique to reduce the computation of neural networks without compromising prediction accuracy. In previous works, structured pruning is usually achieved by imposing L1 regularization on the scaling factors of neurons, and pruning the neurons whose scaling factors are below a certain threshold. The reasoning is that neurons with smaller scaling factors have weaker influence on network output. A scaling factor close to 0 actually suppresses a neuron. However, L1 regularization lacks discrimination between neurons because it pushes all scaling factors towards 0. A more reasonable pruning method is to only suppress unimportant neurons (with 0 scaling factors), and simultaneously keep important neurons intact (with larger scaling factor). To achieve this goal, we propose a new regularizer on scaling factors, namely *polarization* regularizer. Theoretically, we prove that polarization regularizer pushes some scaling factors to 0 and others to a value $a > 0$. Experimentally, we show that structured pruning using polarization regularizer achieves much better results than using L1 regularizer. Experiments on CIFAR and ImageNet datasets show that polarization pruning achieves the state-of-the-art result.

1 Introduction

Network pruning is proved to effectively reduce the computational cost of inference without significantly compromising accuracy [Liu et al., 2019a]. There are two major branches of network pruning. One branch is *unstructured* pruning, which prunes at the level of individual weights [LeCun et al., 1989, Han et al., 2015, Zhou et al., 2019, Ding et al., 2019, Frankle and Carbin, 2019]. The other branch is *structured* pruning, which prunes at the level of neurons (or channels) [Wen et al., 2016, Liu et al., 2017, Ye et al., 2018]. Although unstructured pruning usually reduces more weights than structured pruning [Liu et al., 2019a], it has the drawback that the resulting weight matrices are sparse, which cannot lead to speedup without dedicated hardware or libraries [Han et al., 2016]. In this paper, we focus on neuron-level structured pruning, which does not require additional hardware or libraries to reduce computation on common GPU/CPU devices.

For structured pruning, one promising method is to associate each neuron with a scaling factor, and regularize these scaling factors in training. Then neurons with scaling factors below a certain threshold are pruned [Huang and Wang, 2018]. The regularizer on scaling factors is usually chosen to be L1 [Liu et al., 2017]. However, L1 regularizer tries to push all scaling factors to 0. It is often difficult to find a reasonable pruning threshold. For example, the distribution of scaling factors under L1 regularization for VGG-16 network trained on CIFAR-10 dataset is shown in Figure 1 (a), where

[*]Corresponding author

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

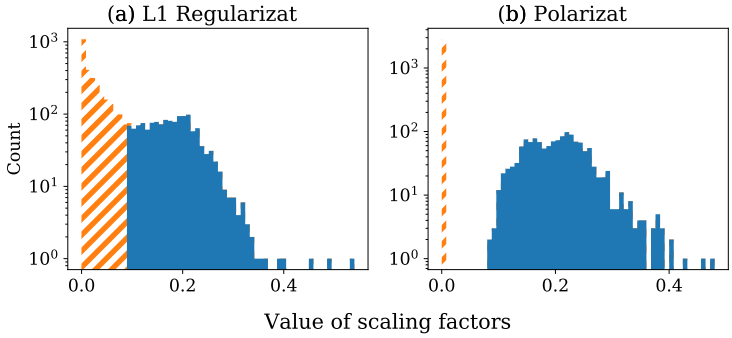


Figure 1: The distributions of scaling factors in VGG-16 trained on CIFAR-10 dataset, with L1 and polarization regularizers respectively. Under the same pruning ratio for both regularizers, the orange part are pruned.

the scaling factors distribute densely around the threshold value. A cut at the threshold value is not very reasonable because there is no margin around the threshold to separate the pruned neurons from the preserved ones. Pruning with this threshold will lead to severe accuracy drop. A more reasonable regularizer should separate the pruned and the preserved neurons more obviously, with a larger margin between them. To attain this goal, we propose a novel regularizer named *polarization*. Different from L1 regularizer that pushes all scaling factors to 0, polarization simultaneously pushes a proportion of scaling factors to 0 (thus pruning these neurons), and the rest scaling factors to a value larger than 0 (thus preserving these neurons). Intuitively, instead of suppressing all neurons in pruning, polarization tries to suppress only a proportion of neurons while keeps others intact. Polarization regularizer naturally makes a distinction between pruned and preserved neurons. And the resulted scaling factors are more separable. As shown in Figure 1 (b), polarization leads to an obvious margin between scaling factors of the pruned neurons (the orange part) and the preserved ones (the blue part). Pruning using polarization is more reasonable because the pruned neurons has much smaller influence on network output than the preserved neurons. Our code is available at `https://github.com/polarizationpruning/PolarizationPruning`.

We summarize our contributions as follows:

- We propose a novel regularizer, namely polarization, for structured pruning of neural networks. We theoretically analyzed the properties of polarization regularizer and proved that it simultaneously pushes a proportion of scaling factors to 0 and others to values larger than 0.
- We verify the effectiveness of polarization pruning on the widely-used CIFAR and ImageNet datasets, and achieve state-of-the-art pruning results.

2 Related Work

Neuron-level structured pruning is also called channel/filter pruning in works focusing on convolutional neural network (CNN) structures. It is equivalent to unstructured pruning with the constraint that all weights connected to one neuron must be pruned or preserved together. There are usually three stages for structured pruning [Liu et al., 2019a]: 1) train an over-parameterized model, 2) prune the trained model according to some criteria, and 3) fine-tune the pruned model to boost performance. Structured pruning methods are characterized by how to handle these 3 stages, especially the training stage. Some methods [Li et al., 2017, He et al., 2018a, 2019] do not add extra sparsity regularizer on the weights of networks in the training stage, while other works [Wen et al., 2016, Zhou et al., 2016, Alvarez and Salzmann, 2016, Lebedev and Lempitsky, 2016, He et al., 2017] impose group sparsity regularization on network weights during training. Yang et al. [2020] propose to use the Hoyer regularizer for pruning. Peng et al. [2019] exploit the inter-channel dependency to determine the optimal combination of preserved channels in the pre-trained model. Luo and Wu [2020] insert binary mask layers after convolution layers during training and removes filters corresponding to zero mask values. Zhuang et al. [2018] fine-tune the pre-trained model with discrimination-aware losses to choose the channels of each layer in the training stage. A group of sparsity regularization pruning

methods introduce a scaling factor for each neuron and add sparsity regularization on the scaling factors in training. Then neurons are pruned based on the values of their scaling factors. Liu et al. [2017] and Ye et al. [2018] use the scale factor in Batch Normalization (BN) as the scaling factor for each neuron, whereas Huang and Wang [2018] add a new scaling factor for each neuron and thus does not depend on BN anymore. During training, L1 regularizer is imposed on the scaling factors of neurons to induce sparsity [Liu et al., 2017, Ye et al., 2018]. After training, neurons with the smallest scaling factors are pruned. Our method falls into this group. And the work most related to ours is [Liu et al., 2017]. We also use the scale factor in BN as the scaling factor of a neuron. But different from all previous works, we propose a novel polarization regularizer to make the pruned and preserved neurons more separable.

Another remotely related area is Neural Architecture Search (NAS), which is employed to find computationally efficient architectures [He et al., 2018b, Liu et al., 2019b]. And some works [Cai et al., 2020] requires much more computation in the training and search processes. Our method is a sparsity regularization pruning method. So we do not focus on comparisons with these NAS methods in this paper.

3 Structured Pruning through Polarization

3.1 Preliminaries

Given a training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$, where $\boldsymbol{x}_i, \boldsymbol{y}_i$ denote the input feature and the label of sample i respectively, we need to train a neural network $\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ denotes the parameters of the network. We introduce a scaling factor for each neuron and represent the scaling factors as a vector $\boldsymbol{\gamma} \in \mathbf{R}^n$, where n is the number of neurons in the network. As in [Liu et al., 2017], we use the scale factor in BN as the scaling factor of each neuron, because of the wide adoption of BN in modern neural networks. The objective function for network training with regularization on scaling factors is:

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} L(\boldsymbol{f}(\boldsymbol{x_i}; \boldsymbol{\theta}), \boldsymbol{y}_i) + R(\boldsymbol{\theta}) + \lambda R_s(\boldsymbol{\gamma}) \tag{1}$$

where $L(\cdot)$ is the loss function, $R(\cdot)$ is usually the L2 regularization on weights of the network, and $R_s(\cdot)$ is the sparsity regularizer on scaling factors of neurons. In pruning, a threshold is chosen and neurons with scaling factors below the threshold are pruned. In [Liu et al., 2017], the sparsity regularizer is chosen to be L1, i.e. $R_s(\boldsymbol{\gamma}) = \|\boldsymbol{\gamma}\|_1$. The effect of L1 regularization is to push all scaling parameters to 0. Therefore, L1 regularization lacks discrimination between pruned and preserved neurons. A more reasonable pruning method is to only suppress unimportant neurons (with 0 scaling factor) and simultaneously keep important neurons intact (with larger scaling factor). To achieve this goal, we propose a new regularizer on scaling factors, namely *polarization* regularizer.

3.2 Polarization Regularizer

Let $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_n)$, and $\mathbf{1}_n = (1, 1, \cdots, 1) \in \mathbf{R}^n$. Let $\bar{\gamma}$ denote the mean of $\gamma_1, \ldots, \gamma_n$:

$$\bar{\gamma} := \frac{1}{n} \mathbf{1}_n^\top \boldsymbol{\gamma} = \frac{1}{n} \sum_{i=1}^{n} \gamma_i$$

γ_i is the scaling factor for neuron i. Obviously, a scaling factor should be positive and bounded. So it is reasonable to constrain each scaling factor to a range: $\gamma_i \in [0, a]$, where $a > 0$. L1 regularizer pushes all scaling factors to 0, because the optimal solution of $\min_{\boldsymbol{\gamma} \in [0,a]^n} \|\boldsymbol{\gamma}\|_1$ is $\mathbf{0}$. To obtain the polarization effect, we need to prevent the scaling factors from converging to one value. So we define the polarization regularizer to be:

$$\begin{aligned} R_s(\boldsymbol{\gamma}) &= t\|\boldsymbol{\gamma}\|_1 - \|\boldsymbol{\gamma} - \bar{\gamma}\mathbf{1}_n\|_1 \\ &= \sum_{i=1}^{n} t|\gamma_i| - |\gamma_i - \bar{\gamma}|, (t \in \mathbb{R}, \gamma_i \in [0, a]) \end{aligned} \tag{2}$$

In Equation (2), we add the new term $-\|\boldsymbol{\gamma} - \bar{\gamma}\mathbf{1}_n\|_1$ to the L1 term $\|\boldsymbol{\gamma}\|_1$. The effect of $-\|\boldsymbol{\gamma} - \bar{\gamma}\mathbf{1}_n\|_1$ is to separate $\gamma_i, 1 \leq i \leq n$ from their average as far as possible. Actually, $-\|\boldsymbol{\gamma} - \bar{\gamma}\mathbf{1}_n\|_1$ reaches its

maximum value when all γ_i are equal, and reaches its minimum value when half of $\gamma_i, 1 \leq i \leq n$ equals 0, while the other half equals a. We also use a hyper-parameter t to control the weight of $\|\boldsymbol{\gamma}\|_1$ relative to $-\|\boldsymbol{\gamma} - \bar{\gamma}\mathbf{1}_n\|_1$. As we will show later, t also controls the proportion of scaling factors that equal 0 under polarization regularizer.

The polarization regularizer has several nice properties. The first property is *permutation invariance*: Let $\pi(\boldsymbol{\gamma})$ denote the vector obtained by an arbitrary permutation of $\boldsymbol{\gamma}$ on its n dimensions. Then it is obvious that $R_s(\pi(\boldsymbol{\gamma})) = R_s(\boldsymbol{\gamma})$, which means that the polarization regularizer is permutation invariant. This property ensures all neurons be equally treated in pruning: no prior pruning bias for any neuron. The second property is concavity, as stated in Lemma 3.1:

Lemma 3.1. *$R_s(\boldsymbol{\gamma})$ defined in Equation (2) is concave.*

To see the effect of polarization regularizer more clearly, we need to solve the following minimization problem:

$$\min_{\boldsymbol{\gamma} \in [0,a]^n} R_s(\boldsymbol{\gamma}) = \sum_{i=1}^{n} t|\gamma_i| - |\gamma_i - \bar{\gamma}|, (t \in \mathbb{R}) \tag{3}$$

This is a minimization of $R_s(\boldsymbol{\gamma})$ on an n-dimensional cube. Utilizing the concavity of $R_s(\boldsymbol{\gamma})$, we prove that the optimal value of Equation (3) is attained on vertices of the n-dimensional cube $[0, a]^n$. It is well known that an n-dimensional cube has 2^n vertices. Utilizing the permutation invariance property of $R_s(\boldsymbol{\gamma})$, we finally prove the following theorem:

Theorem 3.2. *A class of optimal solutions of Equation (3) is that either $\lfloor n\rho \rfloor$ or $\lfloor n\rho \rfloor + 1$ number of $\gamma_i, (1 \leq i \leq n)$ are a, and the rest are 0, where:*

$$\rho = \begin{cases} -t/4 + 1/2, & -2 \leq t \leq 2 \\ 0, & t > 2 \\ 1, & t < -2 \end{cases} \tag{4}$$

The detailed proof of Lemma 3.1 and Theorem 3.2 is in Appendix A. Theorem 3.2 states that the effect of polarization regularizer is to push a proportion of scaling factors to 0 and the rest scaling factors to a. The proportion ρ depends piecewise linearly on the hyper-parameter t as in Equation (4). Therefore, Theorem 3.2 precisely shows the effect of polarization regularizer, and tells that the proportion of 0s is determined by t.

As for the choice of a scaling factor for a neuron, it is convenient to reuse the scale factor in BN of this neuron if it has BN. It is also feasible to introduce an extra scaling factor on the output of a neuron as in [Huang and Wang, 2018]. Our polarization regularizer is applicable for both choices of scaling factors above. To fully separate the two poles, the upper bound a of scaling factors should not be too small, especially when reusing the scale factors in BN. The reason is as follows. Let $z_{in}^{(i)}$ and $z_{out}^{(i)}$ be the input and output of BN on neuron i, then BN performs the following transformation:

$$\hat{z} = \frac{z_{in}^{(i)} - \mathrm{E}[z_{in}^{(i)}]}{\sqrt{\mathrm{Var}[z_{in}^{(i)}]}}$$
$$z_{out}^{(i)} = \gamma_i \hat{z} + \beta_i$$

where $\mathrm{E}[z_{in}^{(i)}]$ and $\mathrm{Var}[z_{in}^{(i)}]$ are the mean and variance of $z_{in}^{(i)}$ respectively. As pointed out in [Ioffe and Szegedy, 2015], it is important for BN to include the *identity transform* by setting $\gamma_i = \sqrt{\mathrm{Var}[z_{in}^{(i)}]}$ and $\beta_i = \mathrm{E}[z_{in}^{(i)}]$. Therefore, a should be sufficiently large such that $\forall 1 \leq i \leq n$, $a > \sqrt{\mathrm{Var}[z_{in}^{(i)}]}$.

Implementation Details In the training stage, we optimize Equation (1) using Stochastic Gradient Descent (SGD). Due to the L1 norm in $R_s(\cdot)$, there exist nondifferentiable points in Equation (1). We use subgradients on these nondifferentiable points. To implement the constraint that $\gamma_i \in [0, a]$, we simply clamp the value of γ_i to a when it is larger than a, and clamp it to 0 when it is smaller than 0 during the training process. All scaling factors are initialized to be 0.5 in image classification tasks as in [Liu et al., 2017]. We find that in our neural networks, the standard deviation of input to BN is mostly smaller than 1. Therefore, we set the upper bound of scaling factor a to 1 in this paper. Empirically, we find that the pruning result is insensitive to the exact value of a.

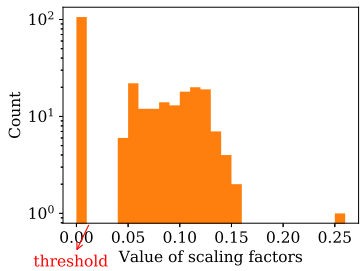


Figure 2: Our strategy to determine the pruning threshold.

3.3 Pruning Strategy

After training with the polarization regularizer, we obtain the values and distribution of scaling factors. We still need a threshold value to prune away neurons with small scaling factors. In [Liu et al., 2017], the threshold is determined by the percentile of neurons to be pruned. Although this method precisely controls the percentile of neurons to be pruned, it often leads to an unreasonable threshold, as shown in Figure 1 (a). We utilize the polarization effect and propose a more reasonable strategy to automatically set the threshold. Because of the polarization effect, the distribution graph always has at least two local maximums (peaks): one is centered near 0, and others are centered around larger values, as is shown in Figure 2. Our strategy is to only prune the neurons that belong to the peak nearest to 0. So the threshold is located at the tail of the peak nearest to 0, as shown by the example in Figure 2. More specifically, when we draw the distribution histogram, we set the bin width to 0.01. Then we scan the bins in the histogram from left to right and find the first local minimum bin. Then the horizontal coordinate of this bin is the threshold for pruning. Using this strategy, we automatically prune the neurons with scaling factors nearest to 0. And empirically we find that this strategy produces good pruning results very robustly.

In our method, we control the number of reduced Floating-point Operations (FLOPs) by adjusting two hyper-parameters: λ in Equation (1) and t in Equation (2). Given a targeted number of reduced FLOPs, our aim is to choose values of λ and t so that polarization pruning will actually reduce the targeted number of FLOPs. Note that the number of reduced FLOPs is positively related to the number of pruned neurons. Theorem 3.2 gives the equation on how the ratio of 0 scaling factors, which equals $1-\rho$, is determined by hyper-parameter t under polarization regularizer alone. However, when minimizing the training objective in Equation (1), where polarization regularizer is just one part of the objective, there is no theory to quantitatively describe the relation between ρ and t. It is obvious that the hyper-parameter λ in Equation (1) also affects the number of reduced FLOPs. We will present experiment results on the effects of the hyper-parameters λ and t in Section 4.4.

Given a targeted reduced FLOPs, we use a search strategy to determine the hyper-parameters λ and t. We first empirically set λ and t to a range and a value, and then proceed by line search until we are close to the targeted FLOPs. The search algorithm aims to find hyper-parameters that lead to a FLOPs reduction $F \in (F^* - \delta, F^* + \delta)$, where F^* is the target reduction and δ is a permissible range. Our algorithm first performs a coarse search on λ, then carries out a fine search on t, as described in Algorithm 1. In the image classification experiments in this paper, the permissible range δ is set to 5%. Empirically, our algorithm needs to try out about 5 groups of hyper-parameters to stop. For each group of hyper-parameters we only need to train 30 epochs on ImageNet dataset for the FLOPs reduction to be stable, rather than the 120 epochs in full training. So the computational overhead of our search is not too much in practice. Note that Algorithm 1 is empirically designed for the image classification tasks in this paper. Strictly speaking, lines 11-18 in Algorithm 1 have a risk of being an infinite loop. However, empirically Algorithm 1 works well and efficiently in our experiments.

After pruning, we fine-tune the pruned network on the training data.

4 Experiments

In this section, we carried out extensive experiments to evaluate our approach on the image classification task, which is the most widely-used task to test pruning methods. We presents the our main

Algorithm 1 Search for hyper-parameters of Polarization Pruning

Input: Target FLOPs reduction F^* with a permissible range δ.
Output: Hyper-parameter values (λ, t), such that, after pruning, the FLOPs reduction $F \in (F^* - \delta, F^* + \delta)$.

Initialize $(\lambda_l, \lambda_u, t) = (0, 2.0 \times 10^{-4}, 1.2)$
// Fix t and search for λ such that $|F - F^*| < 2\delta$ using binary search.
do
  $\lambda \leftarrow (\lambda_l + \lambda_u)/2$;
  Train network with (λ, t) until the corresponding FLOPs reduction F is stable;
  if $F - F^* \geq 2\delta$ **then**
    $\lambda_u \leftarrow \lambda$;
  else if $F - F^* \leq -2\delta$ **then**
    $\lambda_l \leftarrow \lambda$;
  end
while $|F - F^*| \geq 2\delta$;
// Fix λ and search for t such that $|F - F^*| < \delta$.
while $|F - F^*| \geq \delta$ **do**
  if $F - F^* \geq \delta$ **then**
    $t \leftarrow t - 0.1$
  else
    $t \leftarrow t + 0.1$
  end
  Train network with (λ, t) until the corresponding FLOPs reduction F is stable;
end

experiment results in Section 4.3. In Section 4.4, we discuss the effect of hyper-parameters in our approach. To compare the effects of L1 and polarization regularizers, we visualize the distributions of scaling factors in the training process. We also compare the distributions of the scale factors in BN induced by different regularizers with that of the baseline model. At last, we experiment on a wider spectrum of FLOPs reduction.

4.1 Datasets and Compared Methods

As pointed out in [Gale et al., 2019], the same pruning method may behave differently on small datasets and large datasets. Therefore, we evaluate our method on both small datasets (CIFAR 10/100 [Krizhevsky et al., 2009]) and a large dataset (ImageNet [Russakovsky et al., 2015]). We also experiment with three widely-used deep CNN structures: the VGG-Net [Simonyan and Zisserman, 2015], ResNet [He et al., 2016], and MobileNet V2 [Sandler et al., 2018]. This also makes difficulties for us to compare with previous works, because many works only performed experiment in one dataset, or on one network structure. Therefore, on each dataset and network structure, we only compare with those works that have published experiment results on this dataset and network structure. For a method that released codes, we also run it in our experimental setting. If we get better results for this method than the originally published ones, we will use the better results for this method in our comparisons, and append a "(Our-impl.)" suffix for this method in our result tables. Otherwise we will use the originally published results for this method. We have reviewed all the compared methods in Section 2. Some methods, such as CCP-AC in [Peng et al., 2019], insert additional auxiliary classifiers into the network. As pointed out in [Peng et al., 2019], it is unfair to directly compare these methods with other methods. So we do not compare with these methods in our experiments.

4.2 Experimental Setup

The base VGG model is implemented following [Liu et al., 2017], which add BN to the fully connected layers of VGG. The base ResNet model is implemented following [He et al., 2016]. Specifically, our code implementation is based on PyTorch and Torchvision [Paszke et al., 2019]. We list the detailed parameters in our training in Appendix B. We adjust the hyper-parameters λ and t in our polarization regularizer to control the reduced FLOPs as described in Section 3.3. For ResNet and MobileNet V2, we prune blocks with residual connections as [He et al., 2019], where the last

Table 1: Results on CIFAR-10 and CIFAR-100. Best results are bolded.

Dataset	Model	Approach	Baseline	Pruned		
			Acc. (%)	Acc. (%)	Acc. Drop (%)	FLOPs Reduction
CIFAR-10	ResNet-56	NS [Liu et al., 2017] (Our-impl.)	93.80	93.27	0.53	48%
		CP [He et al., 2017]	92.80	91.80	1.00	50%
		AMC [He et al., 2018b]	92.80	91.90	0.90	50%
		DCP [Zhuang et al., 2018]	93.80	93.49	0.31	50%
		DCP-adapt [Zhuang et al., 2018]	93.80	93.81	-0.01	47%
		SFP [He et al., 2018a]	93.59	93.35	0.24	51%
		FPGM [He et al., 2019]	93.59	93.49	0.10	53%
		CCP [Peng et al., 2019]	93.50	93.46	0.04	47%
		DeepHoyer [Yang et al., 2020] (Our-impl.)	93.80	93.54	0.26	48%
		Ours	93.80	93.83	**-0.03**	47%
	VGG-16	FPGM [He et al., 2019]	93.58	93.54	0.04	34%
		NS [Liu et al., 2017] (Our-impl.)	93.88	93.62	0.26	51%
		Ours	93.88	93.92	**-0.04**	54%
CIFAR-100	ResNet-56	NS [Liu et al., 2017] (Our-impl.)	72.49	71.40	1.09	24%
		Ours	72.49	72.46	**0.06**	25%
	VGG-16	NS [Liu et al., 2017] (Our-impl.)	73.83	74.20	-0.37	38%
		COP [Wang et al., 2019]	72.59	71.77	0.82	43%
		Ours	73.83	74.25	**-0.42**	43%

convolution layer in each residual block is padded with zeros to align with the input to the block and the details on pruning ResNet-50 are shown in Appendix C.

Table 2: Results on ImageNet. Best results are bolded.

Model	Approach	Baseline	Pruned		
		Acc. (%)	Acc. (%)	Acc. Drop (%)	FLOPs Reduction
ResNet-50	NS [Liu et al., 2017] (Our-impl.)	76.15	74.88	1.27	53%
	SSS [Huang and Wang, 2018]	76.12	71.82	4.30	43%
	DCP [Zhuang et al., 2018]	76.01	74.95	1.06	56%
	FPGM [He et al., 2019]	76.15	74.13	2.02	53%
	CCP [Peng et al., 2019]	76.15	75.21	0.94	54%
	MetaPruning [Liu et al., 2019b]	76.6	75.4	1.2	50%
	SFP [He et al., 2018a]	76.15	62.14	14.01	42%
	PFP [Liebenwein et al., 2020]	76.13	75.21	0.92	30%
	AutoPruner [Luo and Wu, 2020]	76.15	74.76	1.39	49%
	Ours	76.15	75.63	**0.52**	54%
MobileNet v2	AMC [He et al., 2018b]	71.8	70.8	1.0	27%
	MetaPruning [Liu et al., 2019b]	72.0	71.2	0.8	27%
	DeepHoyer [Yang et al., 2020] (Our-impl.)	72.0	71.7	0.3	25%
	Ours	72.0	71.8	**0.2**	28%

4.3 Experiment Results

We compare the performances of different pruning methods in terms of the accuracy and FLOPs reduction. For each method, we report its baseline model accuracy (Baseline Acc.), its fine-tuned accuracy after pruning (Pruned Acc.), and the accuracy drop (Acc. Drop) between baseline and pruned accuracy. The FLOPs are reported as the pruning ratio, i.e. the ratio of the FLOPs reduced by pruning to the FLOPs of the baseline model.

The CIFAR dataset is a relatively small dataset for image classification. It is the most widely used dataset to compare neural network pruning methods. The experiment results on CIFAR datasets are shown in Table 1. Note that a negative accuracy drop in Table 1 means that the pruned model gets better accuracy than its unpruned baseline model. On CIFAR-10, ResNet-56 task, our method achieves the smallest accuracy drop (-0.03%) and the best pruned accuracy while pruning 47%

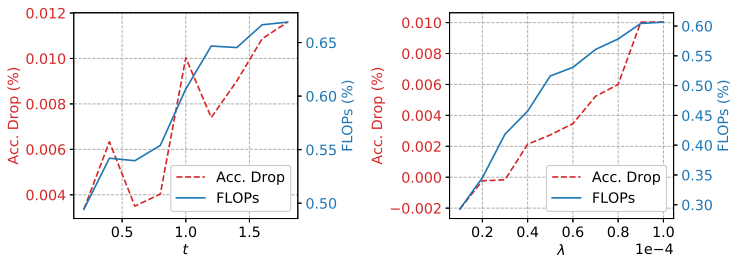


Figure 3: The effect of the hyper-parameters t and λ on the reduced FLOPs and accuracy drop for polarization pruning. When we draw the figure of t, the value of λ is fixed to 1e-4. When we draw the figure of λ, the value of t is fixed to 1.0. We conduct each experiment for 3 times and report the mean value of the accuracy drop and the FLOPs.

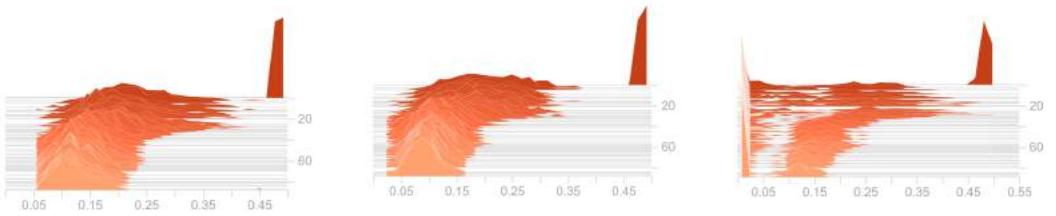
FLOPs, which is much better than other methods. On CIFAR-10, VGG-16 task, our method also achieves the smallest accuracy drop and the best pruned accuracy, with FLOPs reduction similar to NS. On the CIFAR-100 dataset, few previous works on channel pruning have experimented and reported their results. So we only compare with NS and COP on CIFAR-100. And our method still achieves the best results as shown in Table 1.

The ImageNet dataset is a much larger dataset compared to CIFAR. The experiment results on ImageNet dataset are shown in Table 2. On ImageNet, ResNet-50 task, our method achieves the smallest accuracy drop and the best pruned accuracy with a FLOPs very close to other methods. MobileNet V2 is optimized for computation efficiency on mobile devices and contains less redundant computations. Therefore, it is more difficult to prune compared to ResNet-50. On ImageNet, MobileNet task, our method still has the smallest accuracy drop with similar FLOPs reduction to other methods. In summary, our method achieves state-of-the-art channel pruning results on both CIFAR and ImageNet datasets.

4.4 Analysis

The Effect of Hyper-parameters In Section 3.3, we point out that there is no theory to quantitatively describe how the number of reduced FLOPs depends on the hyper-parameters λ in Equation (1) and t in Equation (2). As shown in Figure 3, we empirically studied the effect of the hyper-parameters t and λ on the number of reduced FLOPs as well as the accuracy drop between baseline and the pruned model. Figure 3 shows that when t gets larger, the reduced FLOPs also gets larger. This is reasonable because according to Theorem 3.2, when t gets larger, polarization regularizer will push more scaling factors to 0, and thus more neurons/FLOPs will be pruned. At the same time, the overall trend of the accuracy drop is to gets larger, but with more fluctuations in the curve. This shows that the accuracy drop does not increase monotonically with respect to t. A larger t, and thus more FLOPs reduction, does not always lead to a larger accuracy drop. When λ gets larger, both of the FLOPs reduction and accuracy drop get larger monotonically. This is also reasonable because λ is the overall weight for polarization regularizer in Equation (1). As λ gets larger, polarization regularizer has more influence in training, causing more neurons being pruned and larger accuracy drop.

Visualizing the Effects of Regularizers in Pruning The effect of L1 and polarization regularizers can be visualized through the distribution/histogram of scaling factors. Although Figure 1 already shows an example, we visualize them in more details in Figure 4. Figure 4 shows the evolution of the distributions of scaling factors in the training process. In Figure 4, subfigures (a, b, c) correspond to the training process of baseline ResNet-50, pruning ResNet-50 using L1 regularizer, and pruning ResNet-50 using polarization regularizer on the ImageNet dataset respectively. In subfigure (c), we can clearly see that polarization regularizer gradually pushes the scaling factors to two clusters, with one cluster located at 0, and the other cluster located at a larger value. There is a clear margin between these two clusters. On the other hand, subfigure (b) shows that L1 regularizer pushes the scaling factors to one cluster located near 0.09. Visualization in this subsection verified our conclusion in Section 3.2 that polarization regularizer has the effect of pushing scaling factors to different poles, thus making the pruned and preserved neurons more separable.



(a) Baseline ResNet-50.

(b) Pruning ResNet-50 using L1 regularizer.

(c) Pruning ResNet-50 using polarization regularizer.

Figure 4: The histograms of scaling factors for neurons in a layer of ResNet-50 during the training process. The width, height, and depth of a histogram corresponds to the value of scaling factors, the number of scaling factors, and the training epochs.

Experiments on Wider Spectrum of FLOPs Reduction Previous works on channel pruning usually experiments with a FLOPs reduction no more than 50%. A larger FLOPs reduction may not favor a regularization-based pruning method like ours, due to the huge regularization imposed in the training loss. So we add experiments on larger FLOPs reduction. We compare with a simple channel pruning method that is not based on regularization: the Uniform Channel Scaling (UCS) method, which uses a width-multiplier to uniformly increases or decreases the channel counts across all layers with the same proportion. We also compare with two regularization based methods: NS and DeepHoyer. The original DeepHoyer uses pre-trained models before pruning with the regularizer, which requires extra pre-training epochs before pruning. For fair comparisons, we run all methods using the same experimental setup in our paper, which means exactly the same number of training epochs are used for all methods. The results are in Table 3, which shows that our method performs consistently better under a wide spectrum of FLOPs reductions.

Table 3: Results on large FLOPs reduction.

Dataset / Model	Approach	Baseline	Pruned		
		Acc. (%)	Acc. (%)	Acc. Drop (%)	FLOPs Reduction
CIFAR10 / ResNet56	UCS	93.80	92.25	1.55	70%
	NS [Liu et al., 2017] (Our-impl.)	93.80	91.20	2.60	68%
	DeepHoyer [Yang et al., 2020]	93.80	91.26	2.54	71%
	Ours	93.80	92.63	**1.17**	71%
ImageNet / ResNet50	UCS	76.15	73.78	2.37	70%
	NS [Liu et al., 2017] (Our-impl.)	76.15	70.61	5.54	70%
	Ours	76.15	74.15	**2.00**	70%
ImageNet / MobileNetv2	UCS	72.0	63.71	8.29	61%
	NS [Liu et al., 2017] (Our-impl.)	72.0	65.65	6.35	56%
	Ours	72.0	67.49	**4.51**	57%

5 Conclusion

In this paper, we propose a novel polarization regularizer and perform structured pruning by applying it on the scaling factors of neurons. Different from existing regularizers which push parameters to one pole (usually 0), polarization regularizer pushes parameters to two poles. We theoretically analyzed the properties of our polarization regularizer and empirically validate our analysis by examining the distribution of scaling factors in pruning. Extensive experiments on image classification tasks show that polarization regularizer produces the state-of-the-art pruning results, much better than the commonly used L1 regularizer. Our method is easy to implement and computationally efficient, requiring only one pass of network "pruning+fine-tuning". The polarization regularizer can be conveniently added to neural network training, suppressing some neurons and activating others simultaneously. In the future, we will find more applications where the polarization effect is needed.

6 Broader Impact

This work has the following potential positive impact in the society: because our work saves the computational cost of neural network inference, we can apply it to online inference services to reduce energy consumption, which is beneficial for environmental protection. We have not noticed any obvious negative consequences of our work.

Acknowledgments and Disclosure of Funding

This work is supported by Alibaba Group. The authors would like to thank the Search and Recommendation Division of Alibaba Group for support of this work. The authors are also grateful to their colleagues in Alibaba who helped to provide the computing resources: Tao Lan, Pengfei Fan, Xiaochuan Tang, and Xiaowei Lu.

References

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019a.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 598–605. Morgan Kaufmann, 1989.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1135–1143, 2015.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 3592–3602, 2019.

Xiaohan Ding, Guiguang Ding, Xiangxin Zhou, Yuchen Guo, Jungong Han, and Ji Liu. Global sparse momentum SGD for pruning very deep neural networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 6379–6391, 2019.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2074–2082, 2016.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2755–2763. IEEE Computer Society, 2017. doi: 10.1109/ICCV.2017.298.

Jianbo Ye, Xin Lu, Zhe Lin, and James Z. Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. EIE: efficient inference engine on compressed deep neural network. In *43rd ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2016, Seoul, South Korea, June 18-22, 2016*, pages 243–254. IEEE Computer Society, 2016. doi: 10.1109/ISCA.2016.30.

Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI*, volume 11220 of *Lecture Notes in Computer Science*, pages 317–334. Springer, 2018. doi: 10.1007/978-3-030-01270-0_19.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2234–2240. International Joint Conferences on Artificial Intelligence Organization, 7 2018a. doi: 10.24963/ijcai.2018/309.

Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4340–4349. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00447.

Hao Zhou, Jose M. Alvarez, and Fatih Porikli. Less is more: Towards compact cnns. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, pages 662–677, 2016. doi: 10.1007/978-3-319-46493-0_40.

Jose M. Alvarez and Mathieu Salzmann. Learning the number of neurons in deep networks. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2262–2270, 2016.

Vadim Lebedev and Victor S. Lempitsky. Fast convnets using group-wise brain damage. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2554–2564. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.280.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1398–1406. IEEE Computer Society, 2017. doi: 10.1109/ICCV.2017.155.

Huanrui Yang, Wei Wen, and Hai Li. Deephoyer: Learning sparser neural network with differentiable scale-invariant sparsity measures. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Hanyu Peng, Jiaxiang Wu, Shifeng Chen, and Junzhou Huang. Collaborative channel pruning for deep networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5113–5122. PMLR, 2019.

Jian-Hao Luo and Jianxin Wu. Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. *Pattern Recognition*, page 107461, 2020. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2020.107461.

Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jin-Hui Zhu. Discrimination-aware channel pruning for deep neural networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 883–894, 2018.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. AMC: automl for model compression and acceleration on mobile devices. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, pages 815–832, 2018b. doi: 10.1007/978-3-030-01234-2_48.

Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 3295–3304. IEEE, 2019b. doi: 10.1109/ICCV.2019.00339.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *CoRR*, abs/1902.09574, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Y. He, X. Dong, G. Kang, Y. Fu, C. Yan, and Y. Yang. Asymptotic soft filter pruning for deep convolutional neural networks. *IEEE Transactions on Cybernetics*, pages 1–11, 2019. ISSN 2168-2275. doi: 10.1109/TCYB.2019.2933477.

Wenxiao Wang, Cong Fu, Jishun Guo, Deng Cai, and Xiaofei He. COP: customized deep model compression via regularized correlation-based filter-level pruning. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 3785–3791. ijcai.org, 2019. doi: 10.24963/ijcai.2019/525.

Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*, 2016.

Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Appendices

A Proofs

This part contains the proofs of Lemma 3.1 and Theorem 3.2. For notational convenience, we use $f(\cdot)$ to substitute the $R_s(\cdot)$, and use $\boldsymbol{x}$ to substitute the $\boldsymbol{\gamma}$, in Equations (2) and (3) of Section 3.2. We also restate Lemma 3.1 and Theorem 3.2 using the new notations here, so that this part can be self-contained. Under the new notations here, Equation (2) in Section 3.2 becomes:

$$f(\boldsymbol{x}) = t\|\boldsymbol{x}\|_1 - \|\boldsymbol{x} - \bar{x}\mathbf{1}_n\|_1 = \sum_{i=1}^{n} t|x_i| - |x_i - \bar{x}|, (t \in \mathbb{R}, x_i \in [0, a]) \tag{5}$$

And Lemma 3.1 becomes:

Lemma A.1. *$f(\boldsymbol{x})$ defined in (5) is concave.*

Proof. For any $0 < \theta < 1$ and $x_i, y_i \in [0, a]$, we have:

$$\begin{aligned} &|\theta x_i + (1-\theta)y_i - [\theta\bar{x} + (1-\theta)\bar{y}]| \\ =&|\theta(x_i - \bar{x}) + (1-\theta)(y_i - \bar{y})| \\ \leq&\theta|x_i - \bar{x}| + (1-\theta)|y_i - \bar{y}| \end{aligned} \tag{6}$$

Using $t(\theta x_i + (1-\theta)y_i)$ to subtract both sides of inequality (6), we have:

$$\begin{aligned} &t(\theta x_i + (1-\theta)y_i) - |\theta x_i + (1-\theta)y_i - [\theta\bar{x} + (1-\theta)\bar{y}]| \\ \geq&\theta\left(tx_i - |x_i - \bar{x}|\right) + (1-\theta)\left(ty_i - |y_i - \bar{y}|\right) \end{aligned}$$

Therefore,

$$\begin{aligned} &\sum_{i=1}^{n} t(\theta x_i + (1-\theta)y_i) - |\theta x_i + (1-\theta)y_i - [\theta\bar{x} + (1-\theta)\bar{y}]| \\ \geq&\theta\sum_{i=1}^{n} tx_i - |x_i - \bar{x}| + (1-\theta)\sum_{i=1}^{n} ty_i - |y_i - \bar{y}| \end{aligned}$$

Note that when $x_i, y_i \in [0, a]$,

$$\begin{aligned} f(\boldsymbol{x}) &= \sum_{i=1}^{n} tx_i - |x_i - \bar{x}| \\ f(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) &= \sum_{i=1}^{n} t(\theta x_i + (1-\theta)y_i) - |\theta x_i + (1-\theta)y_i - [\theta\bar{x} + (1-\theta)\bar{y}]| \end{aligned}$$

So we have

$$f(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) \geq \theta f(\boldsymbol{x}) + (1-\theta)f(\boldsymbol{y})$$

which shows that $f(\boldsymbol{x})$ is concave on $\boldsymbol{x} \in [0, a]^n$.

□

Under the new notations here, Equation (3) in Subection 3.2 becomes:

$$\min_{\boldsymbol{x}\in[0,a]^n} f(\boldsymbol{x}) = \sum_{i=1}^{n} t|x_i| - |x_i - \bar{x}|, (t \in \mathbb{R}) \tag{7}$$

Given k indices: $1 \leq i_1, \cdots, i_k \leq n$, and k values: $v_1, \cdots, v_k \in \{0, a\}$, we define the notation:

$$\mathbb{F}_{i_{1,\cdots,k}, v_{1,\cdots,k}} = \{\boldsymbol{x} \in [0, a]^n | x_{i_1} = v_1, \cdots, x_{i_k} = v_k\} \tag{8}$$

Then we prove the following lemma:

Lemma A.2. *The optimal value of (7) is attained on the vertices of* $[0,a]^n$.

Proof. $[0,a]^n$ is an n-dimensional cube, and its boundary consists of $2n$ facets:

$$\begin{aligned}\partial[0,a]^n &= \bigcup_{i=1}^{n}(\{\boldsymbol{x}\in[0,a]^n|x_i=0\}\cup\{\boldsymbol{x}\in[0,a]^n|x_i=a\})\\ &= \bigcup_{i=1}^{n}(\mathbb{F}_{i,0}\cup\mathbb{F}_{i,a})\end{aligned}$$

where the notation of (8) is used. Obviously, $\mathbb{F}_{i,v}(v=0 \text{ or } a)$ is a facet of $[0,a]^n$. Note that $\mathbb{F}_{i,v}=\{\boldsymbol{x}|x_i=v,\forall j\neq i,x_j\in[0,a]\}$. So $\mathbb{F}_{i,v}$ is a $n-1$ dimensional cube on the dimensions $1\leq j\leq n, j\neq i$. Actually, $\mathbb{F}_{i_{1,\cdots,k},v_{1,\cdots,k}}$ is a $n-k$ dimensional cube on dimensions $1\leq j\leq n, j\notin\{i_{1,\cdots,k}\}$.

First step, we prove that the optimal value of (7) is attained on one facet of $[0,a]^n$. Assume $\boldsymbol{x}^*\in[0,a]^n$ is an optimal solution of (7). Because $[0,a]^n$ is a bounded and closed convex set on $\mathbb{R}^n$, then if $\boldsymbol{x}^*\notin\partial[0,a]^n$, there exists a line segment in $[0,a]^n$ passing through $\boldsymbol{x}^*$ and intersects with the boundary of $[0,a]^n$ on two points: $\boldsymbol{x}_1^{(n)},\boldsymbol{x}_2^{(n)}$, i.e. $\exists\boldsymbol{x}_1^{(n)},\boldsymbol{x}_2^{(n)}\in\partial[0,a]^n$, and $\exists\theta\in(0,1)$, such that $\boldsymbol{x}^*=\theta\boldsymbol{x}_1^{(n)}+(1-\theta)\boldsymbol{x}_2^{(n)}$. According to Lemma A.1, $f(\boldsymbol{x})$ is concave on $[0,a]^n$. Therefore $f(\boldsymbol{x}^*)\geq\theta f(\boldsymbol{x}_1^{(n)})+(1-\theta)f(\boldsymbol{x}_2^{(n)})\geq\min\{f(\boldsymbol{x}_1^{(n)}),f(\boldsymbol{x}_2^{(n)})\}$. Without loss of generality, suppose $f(\boldsymbol{x}_1^{(n)})\leq f(\boldsymbol{x}_2^{(n)})$, then $f(\boldsymbol{x}^*)\geq f(\boldsymbol{x}_1^{(n)})$, which means $\boldsymbol{x}_1^{(n)}$ is also an optimal solution. Because $\boldsymbol{x}_1^{(n)}\in\partial[0,a]^n$, $\boldsymbol{x}_1^{(n)}$ must lie on one facet of $[0,a]^n$. Let $\mathbb{F}_{i_1,v_1}$ denote the facet that $\boldsymbol{x}_1^{(n)}$ lies on.

Second step, note that $\mathbb{F}_{i_1,v_1}$ is an $n-1$ dimensional cube, $\boldsymbol{x}_1^{(n)}\in\mathbb{F}_{i_1,v_1}$ is an optimal solution of (7), and $f(\boldsymbol{x})$ is concave on $\mathbb{F}_{i_1,v_1}$. For the same reasons as in the first step, the optimal value of (7) is attained on one facet of $\mathbb{F}_{i_1,v_1}$. Without loss of generality, suppose $\mathbb{F}_{i_{1,2},v_{1,2}}$ is the facet that the optimal solution lies on. Obviously, $\mathbb{F}_{i_{1,2},v_{1,2}}$ is an $n-2$ dimensional cube.

We can iteratively apply the same reasoning as the 2 steps above until we reach the n-th step, in which case the optimal solution $\boldsymbol{y}^*=(y_1^*,y_2^*,\cdots,y_n^*)$ lies on a 1-dimensional cube $\mathbb{F}_{i_{1,\cdots,n-1},v_{1,\cdots,n-1}}$. Let $\{j\}=\{1,\cdots,n\}\setminus\{i_{1,\cdots,n-1}\}$, then $y_j^*\in[0,a]$, and $\forall i\neq j(1\leq i\leq n), y_i^*\in\{0,a\}$. Suppose $y_j^*=\xi a,(0\leq\xi\leq1)$, and let $\boldsymbol{y}_1=(y_1^*,\cdots,y_{j-1}^*,a,y_{j+1}^*,\cdots,y_n^*)$ and $\boldsymbol{y}_2=(y_1^*,\cdots,y_{j-1}^*,0,y_{j+1}^*,\cdots,y_n^*)$, then both $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ are vertices of $[0,a]^n$, and $\boldsymbol{y}^*=\xi\boldsymbol{y}_1+(1-\xi)\boldsymbol{y}_2$. Because $f(\boldsymbol{x})$ is concave, we have $f(\boldsymbol{y}^*)\geq\xi f(\boldsymbol{y}_1)+(1-\xi)f(\boldsymbol{y}_2)\geq\min\{f(\boldsymbol{y}_1),f(\boldsymbol{y}_2)\}$, i.e. $f(\boldsymbol{y}^*)\geq f(\boldsymbol{y}_1)$ or $f(\boldsymbol{y}^*)\geq f(\boldsymbol{y}_2)$. Because both $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ are vertices of $[0,a]^n$, this shows that the optimal value is attained on vertices of $[0,a]^n$. □

Under the new notations here, Theorem 3.2 in Subection 3.2 becomes:

Theorem A.3. *A class of optimal solutions of (7) is that either* $\lfloor n\rho\rfloor$ *or* $\lfloor n\rho\rfloor+1$ *number of* $x_i,(1\leq i\leq n)$ *are* a*, and the rest are 0, where:*

$$\rho=\begin{cases}-t/4+1/2, & -2\leq t\leq 2\\ 0, & t>2\\ 1, & t<-2\end{cases}\tag{9}$$

Proof. By Lemma A.2, the optimal solution of (7) can be found on the vertices of $[0,a]^n$: $\{\boldsymbol{x}|x_i\in\{0,a\},1\leq i\leq n\}$. So (7) is solved by:

$$\min_{x_i\in\{0,a\}}f(\boldsymbol{x})=\sum_{i=1}^{n}t|x_i|-|x_i-\bar{x}|\tag{10}$$

Suppose k variables of $x_i,(1\leq i\leq n)$ equal a, and the rest $n-k$ variables equal 0. Then

$$\begin{aligned}f(\boldsymbol{x}) &= a\left(tk-\left[k|1-\frac{k}{n}|+(n-k)|0-\frac{k}{n}|\right]\right)\\ &= a\left(k(t-1+\frac{k}{n})-(n-k)\frac{k}{n}\right)\end{aligned}$$

This shows that k determines the value of $f(\boldsymbol{x})$, which is permutation invariant. And because $a > 0$, solving (10) is equivalent to solving the following problem:

$$\min_{1\le k\le n} g(k) = k(t-1+\frac{k}{n}) - (n-k)\frac{k}{n} \tag{11}$$

Let $\rho = k/n$, and define

$$\begin{aligned} h(\rho) = \frac{g(k)}{n} &= \rho(t-1+\rho) - (1-\rho)\rho \\ &= 2\rho^2 + (t-2)\rho \end{aligned}$$

Because k can only be integers, the values of ρ are discrete. We first relax this discreteness constraint and assume ρ is a continuous variable in $[0, 1]$. Then (11) is simplified to:

$$\min_{0\le\rho\le 1} h(\rho) = 2\rho^2 + (t-2)\rho \tag{12}$$

Solving (12), we get the optimal solution:

$$\rho^* = \begin{cases} -t/4 + 1/2, & -2 \le t \le 2 \\ 0, & t > 2 \\ 1, & t < -2 \end{cases}$$

Now we consider the discreteness constraint. When $t > 2$, the optimal solution of (11) is 0. When $t < -2$, the optimal solution of (11) is n. When $t \in [-2, 2]$, $\rho^* \in [0, 1]$. If $\rho^* = m/n$, where $m \in \{0, 1, \cdots, n\}$, then the optimal solution of (11) is m. Otherwise, ρ^* must lie in some interval $(m/n, (m+1)/n)$, where $m \in \{0, 1, \cdots, n-1\}$. Because $\rho = \rho^*$ is the symmetric axis of the quadratic function (12), the minimal value of (12) on the discrete points $\{0, 1/n, 2/n, \cdots, 1\}$ must be attained either on m/n or on $(m+1)/n$. So the optimal solution of (11) is either m or $m+1$. □

B Detailed Parameters in Training

The detailed parameters in the training and pruning stages of our method are listed in Table 4. We use the most commonly used batch size, training epochs and learning rate decay scheme as in [Zagoruyko and Komodakis, 2016, Liu et al., 2017]. For example, in the training stage on ImageNet, the learning rate decay scheme "0.1,0.01,0.001@30,60" means that the learning rate in the first 30 epochs is 0.1, in epochs 31 to 60 is 0.01, and in epochs after 60 is 0.001. And "0.05 cosine lr decay" means using cosine learning rate decay strategy with the initial learning rate 0.05.

Table 4: Training hyper-parameters for sparsity training and fine-tuning. Learning rate denotes the learning rates and corresponding decay milestones.

Dataset	Model	Batch size	Training		Fine-tuning	
			Epoch	Learning rate	Epoch	Learning rate
ImageNet	ResNet	512	120	{0.1, 0.01, 0.001} @{40,80}	128	0.05 cosine lr decay [Loshchilov and Hutter, 2017]
ImageNet	MobileNet v2	1024	256	0.4 cosine lr decay with warmup	256	0.05 cosine lr decay
CIFAR	ResNet & VGG	128	200	{0.01, 0.1, 0.02, 0.004, 0.0008} @{1,60,120,160}	200	{0.001, 0.0005, 0.00025, 0.0001} @{20,60,150}

C Details on Pruning ResNet-50

The main building block of ResNet-50 is the bottleneck block [He et al., 2016], as shown in Figure 5. The polarization regularizer is applied to the scale factors of the first two BN layers in each bottleneck block, i.e. the "`bn1`" and "`bn2`" in Figure 5. In the baseline ResNet-50 model, many scaling factors in the "`bn3`" of a bottleneck block are already 0 or very close to 0. So we do not apply any extra regularization on "`bn3`", either in our polarization pruning method or in our implementation of the NS method [Liu et al., 2017]. We visualize the layer-wise distribution of scaling factors in Figure 6. Figure 6 compares the layer-wise distributions of scaling factors between the baseline ResNet-50 model and the model trained with our polarization regularizer on ImageNet dataset. Due to space limit, we only visualize the distributions in the "conv2" and "conv3" layers as defined in [He et al., 2016], where "conv2" and "conv3" contains 3 and 4 bottleneck blocks respectively. Figure 6 shows that our polarization regularizer also makes the scaling factors more separable in layer-level.

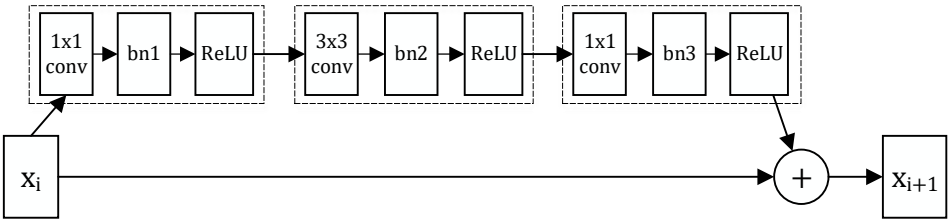


Figure 5: A “bottleneck” block in ResNet-50.

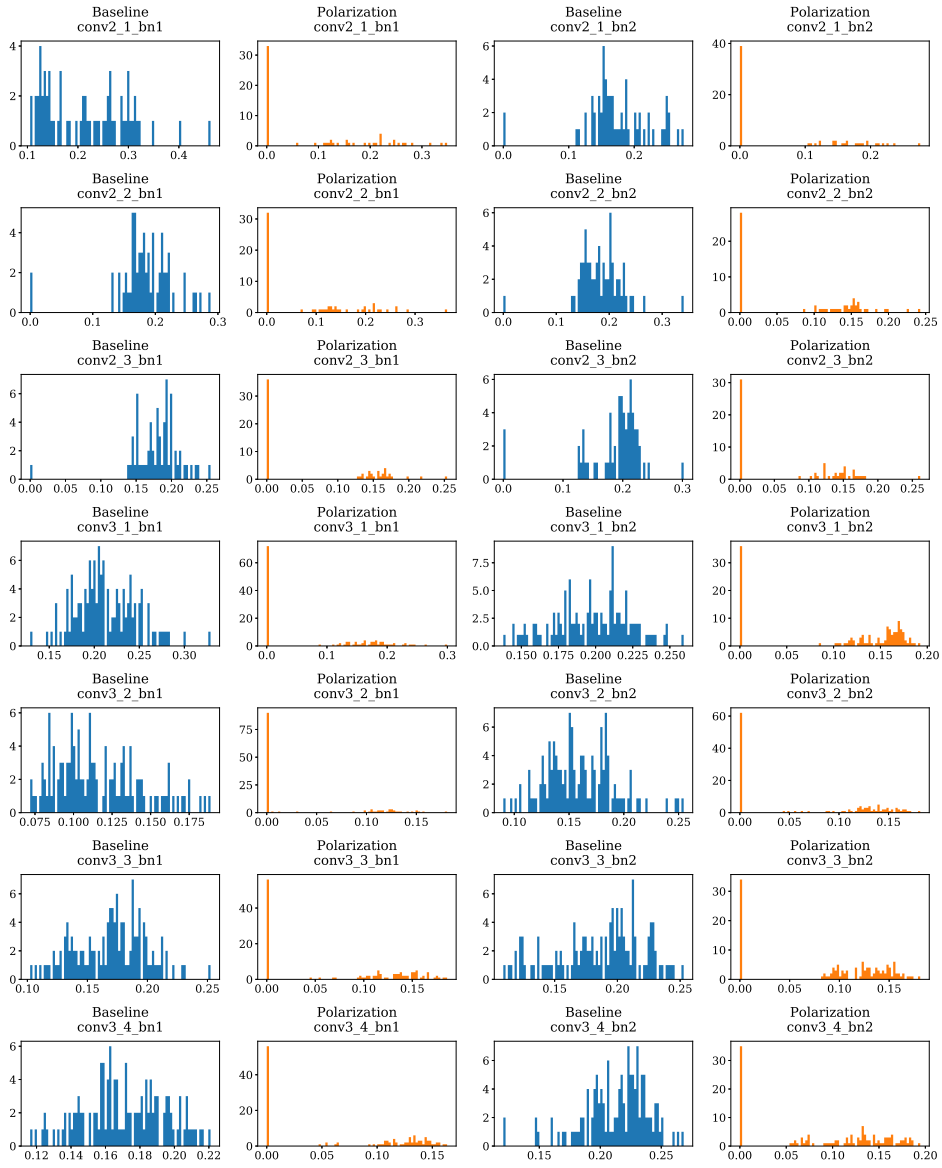


Figure 6: Comparison of the layer-wise scaling factor distributions between baseline ResNet-50 model and the model trained with our polarization regularizer on ImageNet dataset. The left two columns show the distributions of scaling factors in “bn1” of the bottleneck block. The right two columns show the distributions of scaling factors in “bn2” of the bottleneck block. The top 3 rows show scaling factor distributions in each bottleneck block of the “conv2” layer. The bottom 4 rows show scaling factor distributions in each bottleneck block of the “conv3” layer.

人工智能综合领域

AAAI2020

Multi-Question Learning for Visual Question Answering

Chenyi Lei,[1,2*] Lei Wu,[3*] Dong Liu,[1†] Zhao Li,[2] Guoxin Wang,[2] Haihong Tang,[2] Houqiang Li[1]
[1]University of Science and Technology of China
[2]Alibaba Group, [3]Zhejiang University
chenyi.lcy@alibaba-inc.com, shenhai1895@zju.edu.cn, dongeliu@ustc.edu.cn
lizhao.lz@alibaba-inc.com, {xiaogong.wgx, piaoxue}@taobao.com, lihq@ustc.edu.cn

Abstract

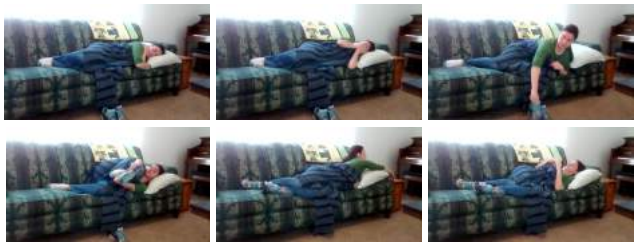
Visual Question Answering (VQA) raises a great challenge for computer vision and natural language processing communities. Most of the existing approaches consider video-question pairs individually during training. However, we observe that there are usually multiple (either sequentially generated or not) questions for the target video in a VQA task, and the questions themselves have abundant semantic relations. To explore these relations, we propose a new paradigm for VQA termed Multi-Question Learning (MQL). Inspired by the multi-task learning, MQL learns from multiple questions jointly together with their corresponding answers for a target video sequence. The learned representations of video-question pairs are then more general to be transferred for new questions. We further propose an effective VQA framework and design a training procedure for MQL, where the specifically designed attention network models the relation between input video and corresponding questions, enabling multiple video-question pairs to be co-trained. Experimental results on public datasets show the favorable performance of the proposed MQL-VQA framework compared to state-of-the-arts.

Introduction

Visual Question Answering (VQA), which is considered as one of the highest goals of artificial intelligence, has raised a great challenge for both computer vision and natural language processing communities. Specifically, VQA is a task about automatically returning the relevant answer to the given question with respect to the reference visual content. Currently, for ease of evaluation, most of the researches (including this paper) focus on selecting answers from given multiple choices. Producing answers in natural language from scratch is deemed a much more difficult task.

Most of the existing works considered the problem of VQA as a multimodal understanding task, that is to combine the visual cues based on image/video understanding with the question based on natural language understanding and reasoning. With the development of attention mechanism, many studies explored the attention-based fusion of image/video representations learned from deep convolutional neural network and question representations learned via the sequential model (Anderson et al. 2018; Nguyen and Okatani 2018; Yang et al. 2016; Yu et al. 2019; 2017b; Li et al. 2019; Gao et al. 2019b), which made significant progress in the last few years. For example, (Anderson et al. 2018) proposed to combine bottom-up and top-down attention mechanism that enables attention to be estimated at the level of objects and other salient image/video regions. Meanwhile, some recent works also considered the modeling of answer representations to enrich the multimodal understanding (Hu, Chao, and Sha 2018; Bai et al. 2018). For instance, (Bai et al. 2018) proposed a deep attention neural tensor network to fuse the information of image-question-answer triplets. Although the existing works have achieved promising performance in VQA, it is still not clear whether they do actually reason, or they reason in a human-comprehensible way (Bai et al. 2018; Cadene et al. 2019).

One issue that was superficially ignored in prior arts is about the multiple questions for one video sequence. Indeed, we observe that there are more and more VQA datasets (Jang et al. 2017; Zhu et al. 2016; Ranjay et al. 2016) containing



Q1: What is in the living room?
Q2: What is the color of the wall?
Q3: What is the person in the video doing?
Q4: Where is the pillow?
Q5: Is the person in the video sitting or lying?

Figure 1: An example of a video sequence with multiple questions. There are abundant relations between the multiple questions themselves, either explicitly *e.g.* between Q3 and Q5, or more implicitly *e.g.* between Q1 and Q4.

*Equal contribution. This work was done when Lei Wu was an intern at Alibaba.
†Dong Liu is the corresponding author.

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

multiple questions for a video sequence. This is also a common phenomenon in applications such as education for children (Calhoun 1999) and multi-turn video question answering (Zhao et al. 2018). It is worth noting that the semantic relational information among multiple video-question pairs plays an important role for human performing actually reasoning in VQA tasks. Actually, multiple questions complement information and provide clues for each other. Taking the question Q3 in Fig.1 as an example, existing VQA methods are subject to bias in Q3 itself to recognize the action of the person in the video sequence, with the underlying models probably not truly understanding the relevance of textual and visual concepts. The visual and textual concepts in Q1 and Q4 (*e.g*. "pillow", "living room" and corresponding potential answers) are strongly related to Q3 (*e.g*. "sleeping"). Furthermore, being able to better exploit the textual and visual concepts from the heavy tails of the question and answer distributions (*e.g*. the concept "sleeping" in Q3) would enable more accurate question answering. It is therefore of great importance to learn these heavy tails answers with related questions and visual information jointly. Nevertheless, almost all of the previous works considered and learned from each video-question pair independently.

The success of multi-task learning (Caruana 1997) implies that inductive transfer may improve generalization by using the domain information contained in the training signal of related tasks as an inductive bias. Analogous to multi-task learning, we propose a new modeling paradigm for VQA tasks termed Multi-Question Learning (MQL) mechanism to address the challenges mentioned above. Our main idea is to co-train multiple video-question pairs jointly to generate a generic video-question representation, which can produce better answers for the questions and can be transferred and utilized in new questions. The representation can extract general and effective features from complex visual and textual information source, which is learned across several related semantic concepts of different questions.

It is worth noting that our MQL is different from the VQA in a visual dialog scenario (Teney et al. 2018; Zhao et al. 2018), where for a new question, the previous questions and answers can be utilized as context. Instead, in our setting, we have multiple questions only without any answer, and we consider the questions in no order. In addition, even when there is only one question for testing, our trained MQL still works and performs well, which indicates that the improvements of trained MQL are not only from the information gain comparing to Single Question Learning (SQL) but also from the better understanding and reasoning capability studied by MQL mechanisms.

To fulfill the MQL, we propose an effective multi-head attention-based deep neural network for video question answering task. As noted video QA is a more difficult problem than image QA (Zhao et al. 2017). Our framework integrates the techniques of convolutional neural network (CNN), gated recurrent unit (GRU), attention mechanism, and multi-question supervision. Specifically, video representations are modeled by CNN; GRU as recurrent neural network (RNN) is used as the building block to learning desired representations from multiple questions. A multi-head attention network is designed on top of the GRU and CNN, which learns attention-weighed representations within questions, video frames and related video regions. Then, all of these attention-weighed representations are integrated as a general representation, which is made more robust and reliable by sharing representations and learning within the multi-question setting. Finally, the general representation is co-trained with different question-answer pairs jointly.

To summary, the main contributions of this paper are as follows:

- Different from the previous studies, we propose the Multi-Question Learning (MQL) mechanism as a new formulation of the VQA problem, which learns visual content with corresponding questions jointly to increase the capacity of modeling latent relation among textual and visual information. Our proposed MQL mechanism provides a new paradigm for VQA tasks.
- We propose effective multi-head attention-based deep learning framework for the video question answering task, in which the video and corresponding questions are co-trained jointly to capture the complex semantic relations among visual and textual information source.
- We validate the proposed MQL mechanism and the Video QA framework on two public datasets. Experimental results demonstrate that the proposed MQL mechanism and the Video QA framework achieve superior performance than the state-of-the-art Video QA methods significantly. Furthermore, we also perform several in-depth analyses to give insights into our proposed approach.

Related Work

In this section, we briefly review some related works about visual question answering and multi-task learning.

Visual Question Answering. The visual question answering task is to provide an accurate answer for the question given in natural language about the given visual content (Antol et al. 2015). Most of the previous works formulated the visual question answering task as a classification problem and solved it with a deep neural network that implements the joint representation of image and question (Gao et al. 2019a; Anderson et al. 2018; Nguyen and Okatani 2018; Yang et al. 2016; Lu et al. 2016; Yu et al. 2019; 2017b; Shih, Singh, and Hoiem 2016; Liang et al. 2018). With the advancement of visual attention, (Shih, Singh, and Hoiem 2016) introduced the spatial attention that selects the relevant image regions to the text-based questions. (Yang et al. 2016) developed the stacked attention network, and (Liang et al. 2018) proposed the focal visual-text attention network for image question answering, where both visual and text sequence information is presented.

To better exploit the reasoning for visual question answering, some works tried to introduce the answer representation into consideration (Hu, Chao, and Sha 2018; Bai et al. 2018; Teney et al. 2018). The researches in (Hu, Chao, and Sha 2018; Teney et al. 2018) tried to project the joint image-question representation into the answer embedding space learned from a text corpus. Meanwhile, multimodal feature

fusion is crucial for visual question answering, which has been studied in many works (Fukui et al. 2016a; Gao et al. 2016). (Fukui et al. 2016a) showed that a more complicated fusion method does improve the performance. However, the reasoning of visual question answering may still be ineffective due to the complexity of visual and textual information (Bai et al. 2018; Jabri, Joulin, and VanderMaaten 2016; Nguyen and Okatani 2018).

Recently, with the development of datasets (Jang et al. 2017; Zhu et al. 2016; Ranjay et al. 2016) and applications (Zhao et al. 2018; Teney et al. 2018) with multiple questions for a piece of visual content, a few works tried to introduce the information of multiple questions into the reasoning for visual question answering. (Zhao et al. 2018) proposed the hierarchical attention context network for context-aware question understanding, which takes a series of questions and answers as sequential input to train and predict for the target question. Different from these works, in this paper we propose to co-train multiple video-question pairs and to answer them simultaneously. Our considered scenario is that multiple questions are to be answered for a video sequence, instead of visual dialog or multi-turn visual question answering.

Multi-Task Learning. The success of deep learning in general depends on data representation, as different representations can entangle and hid more or less different properties of variation behind the data (Ni et al. 2018; Caruana 1997). Deep network learned representations for a single task can only capture information required by the task and other information of the data may be discarded. In contrast, multi-task learning allows statistical strength sharing and knowledge transfer, thus the representations can capture more underlying factors. This hypothesis has been confirmed by a number of previous works (Misra et al. 2016; Collobert and Weston 2008; Hashimoto et al. 2016), demonstrating the effectiveness of representation learning in scenarios of multi-task learning and transfer learning. However, most of traditional methods of visual question answering treat each video-question pair independently, lack of considering the information of and relations to other questions. Therefore, we are keen to re-frame visual question answering as a multi-question learning problem.

Multi-Question Learning

Setup, Notations, and Main Idea

In this paper, we focus on the choice-based visual question answering task, which is to select an answer a from an answer set $\mathcal{A}$ for the given visual content v and the target question q. The answer set $\mathcal{A}$ is usually defined as all possible answers in the related task (Anderson et al. 2018; Benyounes et al. 2017), *i.e.* the same for all (v, q) pairs. We also define the set $\mathcal{C}$ that contains all the correct answers for (v, q). $\mathcal{C}$ is usually a singleton, but may also contain multiple correct answers depending on the task. Classically, the training dataset is denoted by a set of distinctive triplets $\mathcal{D} = \{(v_n, q_n, \mathcal{C}_n \subseteq \mathcal{A})\}$, which is treated independently in the training procedure. However, this traditional paradigm cannot capture all the information encoded in the visual and textual information to derive better models. This is mainly attributed to two challenges.

First, by separating two different questions q_i and q_j for the same visual content v, with the ordinal numbers i and j, we lose the semantic kinship between the two. Specifically, if there are two triplets $(v, q_i, a_i \in \mathcal{C}_i)$ and $(v, q_j, a_j \in \mathcal{C}_j)$ having semantic relations between q_i and q_j, we would expect the reasoning of a_i and a_j to have some degrees of semantic relations, too. In the traditional setup, two triplets are considered independently such that the learning procedure focuses on the part of the visual and textual information corresponding to the specific triplet, which does not preserve and takes full advantage of related semantic relevance information in the questions. This semantic relevance information from different questions is beneficial and adds robustness for the more general multimodal representation, which can be transferred for answering and reasoning other questions.

Second, traditional methods that treat triplets $\mathcal{D}$ as for an independent classification problem, are severely influenced by the bias of answers of the training data. Class imbalance has been a longstanding challenge for classification, thus it is difficult to learn the visual and textual information for low-frequency questions and answers (Chao, Hu, and Sha 2018). In contrast, learning multiple questions jointly provides a new possibility to narrow the bias of answers of the training data by the transferred information learned from other questions.

Therefore, we propose a new paradigm for studying visual question answering called Multi-Question Learning (MQL) in this work. Specifically, we further consider the question set $\mathcal{Q}$, that is all questions for the specific visual content v. For the specific visual content v_n, we assume there is a set of questions $\mathcal{Q}_n$, and the question number is Q_n, then the corresponding training dataset is re-defined as $\{(v_n, q_k \in \mathcal{Q}_n, \mathcal{Q}_n, \mathcal{C}_k \subseteq \mathcal{A})\}$, which will be co-trained.

Problem Formulation of Multi-Question Learning

For the specific visual content v_n with Q_n questions $q_k \in \mathcal{Q}_n$, we first define a joint embedding function $f_\theta(v_n, \mathcal{Q}_n)$ to generate the general embedding representation r_n. Then, we learn multiple question-answer pairs simultaneously. For each question, the other questions are viewed as regularization. By sharing representations with MQL, we can enable joint visual and textual representations to be general and reliable. We also define a function $g_\phi(r_n, q_k)$ for the target question to emphasize its uniqueness. The related functions are parameterized by θ and ϕ, respectively. In the MQL mechanism, any multimodal representation network can be used, such as multi-layer perceptron (MLP) and Stacked Attention Network (SAN) (Yang et al. 2016). In our work, we also design effective multi-head attention-based network to implement the function f_θ and g_ϕ.

Formally, given the data $(v_n, q_k \in \mathcal{Q}_n, \mathcal{Q}_n, a \in \mathcal{C}_k)$ corresponding to visual content v_n and target question $q_k \in \mathcal{Q}_n$, we define the following probabilistic model,

$$p(a|v_n, q_k, \mathcal{Q}_n) = \frac{\exp(g_\phi(r_n, q_k)^\top W_a)}{\sum_{a' \in \mathcal{A}} \exp(g_\phi(r_n, q_k)^\top W_{a'})} \quad (1)$$

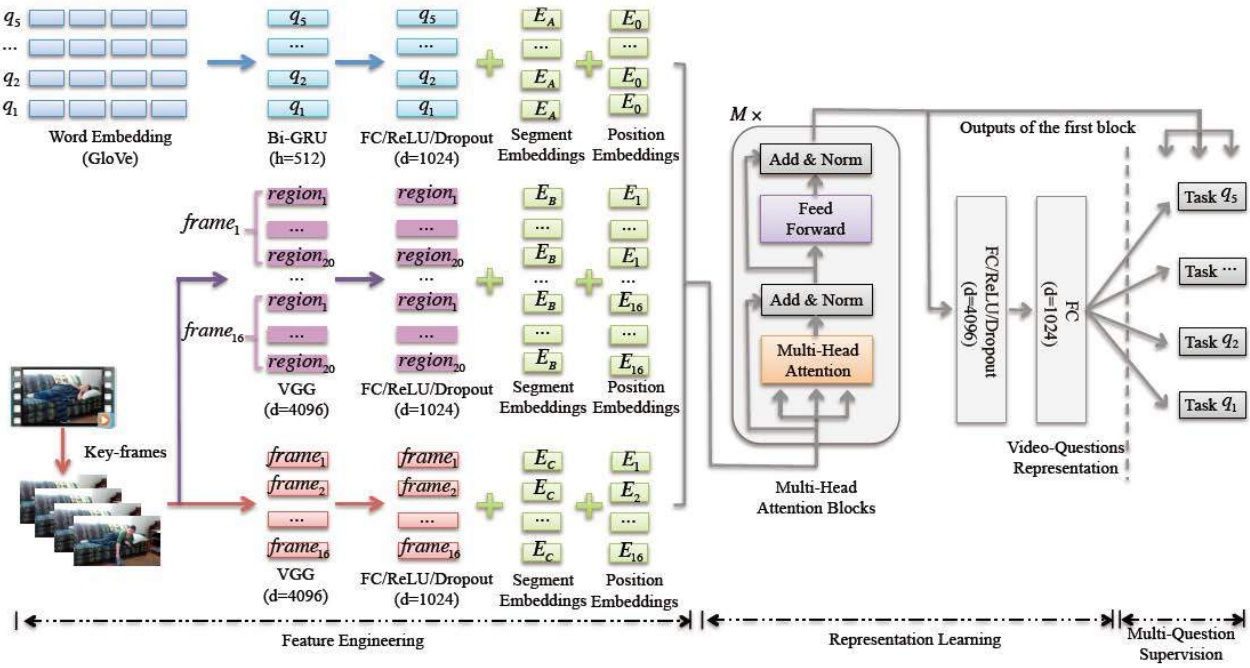


Figure 2: The proposed video question answering framework with the Multi-Question Learning (MQL) mechanism. The framework has three parts including *Feature Engineering*, *Representation Learning*, and *Multi-Question Supervision*.

where $r_n = f_\theta(v_n, \mathcal{Q}_n)$ and W_a is the learned representation for the answer $a \in \mathcal{A}$.

In practice, we need to predefine a constant denoted as M_Q to represent the number of the questions used for a single visual content. There are three different cases regarding the constant for each question. First, if $Q_n = M_Q$, there is nothing to do. Second, if $Q_n < M_Q$, we augment $\mathcal{Q}_n$ with either all-zero vector or duplication. Third, if $Q_n > M_Q$, then for each question q_k, we construct a set $\mathcal{Q}_n^k \subset \mathcal{Q}_n$ such that the subset $\mathcal{Q}_n^k$ has exactly M_Q questions; here we select the top M_Q relevant questions to q_k into the subset, according to some pre-trained models such as BERT (Devlin et al. 2018). In our work, we set $M_Q = 5$ by default. Note that if we set $M_Q = 1$, then the MQL degenerates to single-question learning (SQL), *i.e.* the traditional visual question answering setting.

Given Eq. (1), it is natural to learn the parameters to maximize the likelihood of the probabilistic model. As there might be different numbers of questions for different visual content items, we introduce the weighted likelihood in our work, which has also been exploited in several previous works (Goyal et al. 2017; Hu, Chao, and Sha 2018). Specifically, we have

$$\mathcal{L} = -\sum_{n}^{N} \sum_{k}^{|\mathcal{Q}_n^k|} \frac{|\mathcal{Q}_n^k|}{M_Q} \log p(a|v_n, q_k, \mathcal{Q}_n^k) \qquad (2)$$

whose intuition is the hypothesis that the v_n with more related questions contributes more to the MQL mechanism.

During prediction, given the learned parameters mentioned above, we can apply the following decision rule

$$a^* = \arg\max_{a \in \mathcal{A}} g_\phi(r_n, q_k)^\top W_a \qquad (3)$$

to identify the answer to the question q_k of visual content v_n, where $r_n = f_\theta(v_n, \mathcal{Q}_n^k)$.

Video Question Answering Framework

The proposed MQL mechanism is generic for different visual question answering tasks as long as there are available multi-question-per-content data for training. In this paper, we focus on the Video QA task as it is considered the most challenging problem among peers (Zhao et al. 2017). Motivated by recent attention mechanisms in NLP (Devlin et al. 2018) and CV (Gao et al. 2019b), as illustrated in Fig. 2, we design a multi-head attention-based neural network to better modeling the correlations and interactions between the multi-questions and videos, which is divided into three parts for a better description. We first extract visual and textual features in the *Feature Engineering* part. Then, the multi-head attention-based question and video representations are generated in the *Representation Learning* part. Finally, there is a *Multi-Question Supervision* part for co-training different questions.

Feature Engineering. From an input video sequence, we first select 16 key-frames and resize each frame to the resolution of 224×224. Then, there are two streams for visual information. One is global information for each frame. In this stream, the initial visual representation of each frame is a 4096-dimensional vector that is extracted by the pre-trained VGG model (Simonyan and Zisserman 2014). The other is region information for each region inside a frame, which is motivated by the claim that the global representation of the frame may fail to capture all necessary information for answering region-specific questions (Li and Jia 2016). We generate 20 candidate regions per frame using the method in (Li and Jia 2016). For each region, we also em-

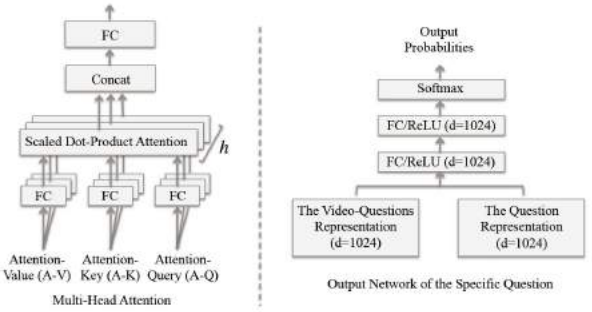


Figure 3: More details of the proposed Video QA framework shown in Fig. 2. Left: the multi-head attention mechanism; Right: the *Multi-Question Supervision* part.

Dataset and Category		QA Pairs			Video Sequences		
		Train	Test	Total	Train	Test	Total
TGIF-QA	Object	16755	5586	22341	15584	3209	18793
	Number	8096	3148	11244	8033	1903	9936
	Color	11939	3904	15843	10872	3190	14062
	Location	2602	1053	3655	2600	917	3517
Total		84124	19795	103919	48198	8522	56720
CPT-QA	Object	7431	769	8200	7288	751	8039
	Number	1435	160	1595	1159	159	1593
	Color	7938	875	8813	1434	868	8749
	Location	7276	795	8071	7242	788	8030
	Action	7108	762	7870	7105	760	7865
	Either-OR	6150	681	6831	5834	644	6478
	Other	2542	358	2900	2115	279	2394
Total		39880	4400	44280	7976	880	8856

Table 1: Statistics of the two datasets, in which we divide questions into several different categories, such as "Object" and "Number."

ploy the pre-trained VGG to generate a 4096-dimensional vector. For textual information, the GloVe model (Pennington, Socher, and Manning 2014) is employed to extract the semantic word embeddings for each question, and a single-layer bidirectional GRU (Bi-GRU) with the hidden dimension of 512 is used on top of each question. All of these Bi-GRUs are sharing parameters. After the extraction of initial representations, we employ three full-connection layers to change dimension to 1024 for each kind of features, respectively. To distinguish the different types of features, three kinds of segment embeddings are added to corresponding features, known as E_A, E_B and E_C, respectively. Since the temporal information is believed more important for video, we further design a position embedding layer like in recent works (Devlin et al. 2018). Note that we allow questions to be out of order, so all questions for a video sequence have the same position embedding E_0, and the frame and its related regions are assigned position embedding E_i for the i-th key-frame. Finally, we have a 1024-dimensional feature for each question, region, and key-frame.

Representation Learning. Motivated by the success of multi-head attention mechanism in recent works (Devlin et al. 2018; Vaswani et al. 2017), we design a multi-head attention-based representation learning approach to capture the complex semantic relations among questions, regions, and key-frames. There are M multi-head attention blocks where M is a hyper-parameter and equals 6 in our work. Each block refers to the structures in (Vaswani et al. 2017), contains two sub-layers. The first is a multi-head self-attention, and the second is a simple input-wise fully connected feed-forward. In this way, the model could exploit a series of important semantic information for representing and reasoning the questions and the video, including the relations among questions, the temporal information among key-frames, and the spatial information and the overall semantic relations among questions and frames.

The left part of Fig. 3 illustrates the multi-head attention mechanism. We first linearly project an input vector h times (h-heads) with different, learned linear projections to d_q, d_k and d_v dimensions, respectively, known as attention-query (A-Q), attention-key (A-K) and attention-value (A-V). In our work, A-Q, A-K, and A-V are the same, and particularly, they are the extracted feature vectors from the Feature Engineering part or the previous multi-head attention block. Next, we perform scaled dot-product attention for each head, and we concatenate all scaled dot-product attention output vectors along each head. A full-connection layer is then employed to get the output. In our work, we set $h = 8$ and $d_q = d_k = d_v = 1024/h = 128$. The feed-forward network includes two full-connection layers, which is applied to each input vector separately and identically.

After multi-head attention blocks, we concatenate all output vectors and utilize two full-connection layers to generate the final question/video representations. Note that the output vectors of the first block corresponding to the input question vector are employed to represent each question, respectively, which will be used in the following *Multi-Question Supervision* part. Our motivation is that the output vectors of the first multi-head attention block preserve and distinguish the information of each question to the largest extent. At the same time, the relations within questions, regions, and frames can be integrated to a certain extent.

Multi-Question Supervision. As mentioned above, we believe that jointly train the multiple questions could make the question/video representations more general and reliable. Thus we define a network for co-training, which is to share parameters for each question and is illustrated in the right part of Fig. 3. We concatenate the representations of questions/video and the target question, in order to emphasize the uniqueness of the target question on the basis of the shared semantic expression of multiple questions and the video. Both of them are generated from the previous *Representation Learning* part. Then, there are two full-connection layers to generate the final feature vector for prediction.

Experiments

Datasets and Metrics

TGIF-QA dataset[1]. It is a large-scale public dataset introduced by (Jang et al. 2017). Since we focus on the problem of selecting appropriate answers from a pool of candidate answers in this paper, we perform our experiments on the "Frame QA" task described in TGIF-QA. In this dataset,

[1]https://github.com/YunseokJANG/tgif-qa

Method	Accuracy(%)
VIS+LSTM(avg) (Ren, Kiros, and Zemel 2015)	34.97
Yu *et al*. (Fukui et al. 2016b)	39.64
ST-TP(R+F) (Jang et al. 2017)	49.50
Co-memory (full) (Gao et al. 2018)	51.50
PSAC(R) (Li et al. 2019)	55.7
Ours (SQL)	57.03
Ours (M-S)	58.20
Ours (MQL)	**59.83**

Table 2: Results on the Frame QA task of TGIF-QA dataset. "Ours (M-S)" stands for using MQL for training but testing with single question.

Method	Accuracy(%)
SAN (SQL)	53.14
SAN (MQL)	58.08
r-STAN (SQL)	55.62
r-STAN (MQL)	59.82
MLAN (SQL)	56.11
MLAN (MQL)	61.54
Ours (SQL)	58.26
Ours (M-S)	59.00
Ours (MQL)	**62.85**

Table 3: Results on the CPT-QA dataset.

each video sequence is accompanied with one or more questions, and each question is associated with only one answer. We follow the training and test dataset of TGIF-QA.

CPT-QA dataset[2]. CPT-QA dataset is a recent Video QA dataset released by Zhejiang Lab. For this dataset, the providers pre-define several types of questions and invite volunteers to ask and answer questions for video sequences. For each video sequence there are up to five questions, and for each question there are three persons providing "ground truth" answers, thus, there can be up to three correct answers for one question. Compared with the TGIF-QA dataset, CPT-QA is smaller but has more variety in the category of questions, as can be observed in Table 1. As this dataset is prepared for a competition, we do not have the ground truth of its test set. In our experiments, we divide its training set into two parts: randomly selecting 90% video sequences for training and using the rest for the test.

Metrics. We use the classification accuracy (ACC) as the metric to evaluate the performance of different methods. For the CPT-QA dataset, it is worth noting that the true positive is defined as the predicted answer hits any one of the correct answers for each question.

Compared Approaches

Our full approach is denoted by Ours (MQL). To identify the general effect of MQL mechanism, we also test a single question learning (SQL) version of our approach, *i.e*. we set $M_Q = 1$ in Eq. (2). Furthermore, we test the performance of our model that is trained based on MQL mechanism but deals with a single question in the test, which is denoted by Ours (M-S). This is to cope with scenarios where only one question is given for a video sequence in practice.

Since there are several state-of-the-art works reported their results on the TGIF-QA dataset, we directly compare our result with these results and mark the related work to each method in Table 2. However, for the CPT-QA dataset, there is no published result to our best knowledge. So we design several baselines following (Zhao et al. 2017):

SAN (SQL) is the incremental algorithm based on stacked attention networks (Yang et al. 2016). Following (Zhao et al. 2017), we add the GRU network to fuse the sequential representations of spatially attended frames for Video QA.

MLAN (SQL) is modified from the MLAN algorithm (Yu et al. 2017a), where we add the Bi-GRU network for obtaining the representations of the video and the target question.

r-STAN (SQL) is a state-of-the-art Video QA method (Zhao et al. 2017), which proposed a hierarchal spatio-temporal attentional encoder-decoder learning method with multi-step reasoning process for Video QA.

SAN (MQL), MLAN (MQL) and r-STAN (MQL) are the MQL versions of SAN, MLAN, and r-STAN, respectively, which are compared with Ours (MQL) to reveal the effects of different architectures for modeling the relationship between the questions and videos in MQL. They perform a concatenation and add a full-connection layer on top of the corresponding SQL version.

In summary, we design the comparisons between SQL and MQL to demonstrate the advantages of MQL. Considering that the MQL setting provides more information than the SQL setting during prediction, and it is intuitive that MQL performs better than SQL. For fairer comparisons, we also design and implement several MQL baselines, and compare Ours (MQL) with the other MQL methods. Besides, we also compare all SQL-based methods with Ours (M-S), which only involves a single question during prediction.

All approaches employ common initial textual and visual features. Configurations of our method can be found in Fig. 2 and related sections. All approaches are implemented on the TensorFlow, utilizing 100 parameter servers and 2000 workers, each of with runs with 15 CPU cores. The batch size of Ours (MQL) is set to be 16. All the batches are trained for 30 epochs. We use the Adam optimizer for all the approaches. The initial learning rate is set to 1e-4.

Overall Results

Table 2 and 3 give a comprehensive evaluation for baselines on TGIF-QA and CPT-QA datasets, respectively. Specifically, in all cases in Table 2 and 3, Ours (MQL) achieves the best performance, which shows that the MQL mechanism with an effective multimodal features fusion model can further improve the performance of visual question answering. Meanwhile, these experimental results also reveal a number of interesting points:

- In Table 3, all the methods based on MQL mechanism outperform the corresponding SQL version by a noticeable margin, which suggests the MQL mechanism is critical for the performance of VQA tasks. We attribute the reasons for two points. First, multiple questions contribute to the better description and supplement for the target question. Second, co-training multiple questions jointly

[2]https://tianchi.aliyun.com/competition/entrance/231676/information

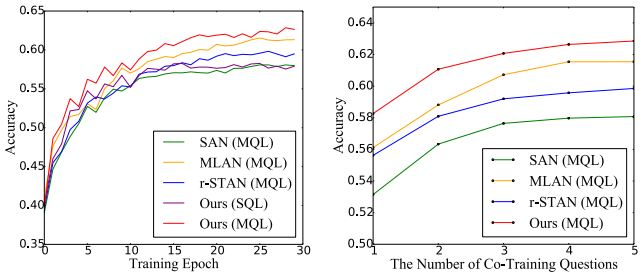


Figure 4: Left: the learning curves of different approaches. Right: the performance with respect to different numbers of co-trained questions (*i.e.* M_Q in our MQL).

makes the model better understanding the semantic relationships among questions and the visual content for reasoning, and leads to a more general multimodal representation. Furthermore, it indicates that, by better modeling the correlations and interactions between the questions and videos, Ours (MQL) is in favor of outperforming all other MQL-based methods. We make more in-depth discussions of MQL in the next sections.

- Ours (SQL) outperforms the state-of-the-art approaches in Table 2, and Ours (SQL) surpasses the other single question learning baselines in Table 3. It indicates the effectiveness of our proposed video question answering framework, especially the network design. Most probably, the multi-head attention mechanism has a strong ability to correlate and integrate three different types of features, *i.e.* question, region, and frame.
- The accuracy of Ours (M-S) drops a lot compared to that of Ours (MQL). It emphasizes the importance to identify the relations between multiple questions for answer prediction. However, it still outperforms Ours (SQL), which indicates that the model trained with MQL setting is good enough to cope with different test settings: it is applicable when there is only one question for the video sequence during prediction.

Discussion and Analysis

For better understanding our proposed MQL mechanism and video question answering framework, we perform further in-depth discussion and analysis in this section.

We observed the convergence of training with different approaches on the CPT-QA dataset, as shown in the left part of Fig. 4. The learning curves of Ours (SQL) and SAN (MQL) converge quickly after about 15 epochs, and the accuracy on validation set of Ours (SQL) even begins to decline after 15 epochs. In contrast, the learning curves of other MQL based methods grow continually. This phenomenon indicates that multi-question learning mechanism with an effective multimodal fusion model could learn more semantical information from the training data continually, which exploits the training data better.

In our approach, the maximal number of questions for each video sequence during training–M_Q in Eq. (2) is an essential parameter. Since most of the video sequences in the two datasets have up to 5 questions, we test different M_Q values from 1 to 5. Note that when $M_Q = 1$ it is indeed single question learning. The related accuracy results are displayed in the right part of Fig. 4, in which we can observe that the performance increases continually as the number of co-training questions increases. This result suggests the advantage of MQL. Note that the gain becomes marginal when M_Q is greater than 3. One reason is that most of the video sequences in the training data contain no more than 3 questions. The other reason is due to the diversity of questions, *i.e.* introducing more irrelevant questions is not helpful.

Method	Accuracy						
	Object	Color	Number	Location	Action	Either-OR	Other
SAN (SQL)	0.4859	0.5859	0.7379	0.6849	0.2029	0.7667	0.1487
SAN (MQL)	0.5051	0.6332	0.7711	0.7321	0.2414	0.8212	0.1789
r-STAN (SQL)	0.5098	0.6020	0.7524	0.6973	0.2156	0.8001	0.1712
r-STAN (MQL)	0.5386	0.6640	0.8006	0.7468	0.2564	0.8611	0.1953
MLAN (SQL)	0.5244	0.6151	0.8087	0.7149	0.2196	0.8096	0.1856
MLAN (MQL)	0.5537	0.6873	0.8422	0.7644	0.2604	0.8756	0.2093
Ours (SQL)	0.5329	0.6332	0.8310	0.7273	0.2223	0.8240	0.1937
Ours (MQL)	**0.5722**	**0.7033**	**0.8865**	**0.7768**	**0.2770**	**0.8928**	**0.2150**

Table 4: Results of accuracy per each category on the CPT-QA dataset.

Table 4 summarizes the evaluation results of accuracy for different types of questions on the CPT-QA dataset. All MQL based models outperform their corresponding single question learning versions on all kinds of questions. Furthermore, comparing Ours (MQL) with Ours (SQL), we observe that for questions about "Actions," such as "What is the person in the video doing?" the relative improvement is the highest among different types of questions, reaches as much as 24.60%. This can be attributed to the characteristics of these questions, to answer which we need the global information of the video and multiple questions can help here. It also implies that multi-task learning can be helpful for video action recognition.

Conclusion

In this paper, we propose the multi-question learning mechanism as a new paradigm for the visual question answering tasks, analogous to multi-task learning. Multi-question learning mechanism helps generate more robust and generic multimodal representations for visual and textual information, which can not only benefit for the target question, but also be transferred for answering and reasoning other questions. We also propose a multi-head attention-based video question answering framework to implement the multi-question learning mechanism, which captures the complex semantic relations among questions, spatially variant visual information, and temporal information. The empirical evaluations on two large-scale public datasets demonstrate that our proposed approach builds a new state-of-the-art for video question answering.

Acknowledgements

This work was supported in part to Dr. Houqiang Li by NSFC under contract No. 61836011, and was partly supported by Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies.

References

Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; and Zhang, L. 2018. Bottom-up and top-down attention for image captioning and visual question answering. *In CVPR*.

Antol, S.; Agrawal, A.; Lu, J.; Mitchell, M.; Batra, D.; C.Zitnick; and Parikh, D. 2015. Vqa: Visual question answering. *In ICCV*.

Bai, Y.; Fu, J.; Zhao, T.; and Mei, T. 2018. Deep attention neural tensor network for visual question answering. *In ECCV*.

Benyounes, H.; Cadene, R.; Cord, M.; and Thome, N. 2017. Mutan: Multimodal tucker fusion for visual question answering. *In ICCV*.

Cadene, R.; Ben-younes, H.; Cord, M.; and Thome, N. 2019. Murel: Multimodal relational reasoning for visual question answering. *In CVPR*.

Calhoun, E. 1999. Teaching beginning reading and writing with the picture word inductive model. *Association for Supervision Curriculum Development* 138.

Caruana, R. 1997. Multitask learning. *Machine Learning* 28(1):41–75.

Chao, W.; Hu, H.; and Sha, F. 2018. Cross-dataset adaptation for visual question answering. *In CVPR*.

Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. *In ICML*.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*.

Fukui, A.; D.Park; Yang, D.; Rohrbach, A.; Darrell, T.; and Rohrbach, M. 2016a. Multimodal compact bilinear pooling for visual question answering and visual grounding. *In ECCV*.

Fukui, A.; Park, D. H.; Yang, D.; Rohrbach, A.; Darrell, T.; and Rohrbach, M. 2016b. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*.

Gao, Y.; Beijbom, O.; Zhang, N.; and Darrell, T. 2016. Compact bilinear pooling. *In CVPR*.

Gao, J.; Ge, R.; Chen, K.; and Nevatia, R. 2018. Motion-appearance co-memory networks for video question answering. *In CVPR*.

Gao, L.; Zeng, P.; Song, J.; Li, Y.; Liu, W.; Mei, T.; and Shen, H. 2019a. Structured two-stream attention network for video question answering. *In AAAI*.

Gao, P.; Jiang, Z.; You, H.; Lu, P.; Hoi, S.; Wang, X.; and Li, H. 2019b. Dynamic fusion with intra- and inter-modality attention flow for visual question answering. *In CVPR*.

Goyal, Y.; Khot, T.; Summers-Stay, D.; Batra, D.; and Parikh, D. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. *In CVPR*.

Hashimoto, K.; Xiong, C.; Tsuruoka, Y.; and Socher, R. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*.

Hu, H.; Chao, W.; and Sha, F. 2018. Learning answer embeddings for visual question answering. *In CVPR*.

Jabri, A.; Joulin, A.; and VanderMaaten, L. 2016. Revisiting visual question answering baselines. *In ECCV*.

Jang, Y.; Song, Y.; Yu, Y.; Kim, Y.; and Kim, G. 2017. Tgif-qa: To- ward spatio-temporal reasoning in visual question answering. *In CVPR*.

Li, R., and Jia, J. 2016. Visual question answering with question representation update (qru). *In NIPS*.

Li, X.; Song, J.; Gao, L.; Huang, W.; He, X.; and Gan, C. 2019. Beyond rnns: Positional self-attention with co-attention for video question answering. *In AAAI*.

Liang, J.; Jiang, L.; Cao, L.; Li, L.; and Hauptmann, A. 2018. Focal visual-text attention for visual question answering. *In CVPR*.

Lu, J.; Yang, J.; Batra, D.; and Parikh, D. 2016. Hierarchical question-image co-attention for visual question answering. *In Neural Information Processing Systems (NIPS)*.

Misra, I.; Shrivastava, A.; Gupta, A.; and Hebert, M. 2016. Cross-stitch networks for multi-task learning. *In CVPR*.

Nguyen, D., and Okatani, T. 2018. Improved fusion of visual and language representations by dense symmetric co-attention for visual question answering. *In CVPR*.

Ni, Y.; Ou, D.; Liu, S.; Li, X.; Ou, W.; Zeng, A.; and Si, L. 2018. Perceive your users in depth: Learning universal user representations from multiple e-commerce tasks. *In KDD*.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. *https://nlp.stanford.edu/projects/glove/*.

Ranjay, K.; Yuke, Z.; Oliver, G.; Justin, J.; Kenji, H.; Joshua, K.; Stephanie, C.; Yannis, K.; Jia, L.; David, S.; Michael, B.; and Li, F. 2016. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *https://arxiv.org/abs/1602.07332*.

Ren, M.; Kiros, R.; and Zemel, R. 2015. Exploring models and data for image question answering. *Advances in Neural Information Processing Systems* 2953–2961.

Shih, K.; Singh, S.; and Hoiem, D. 2016. Where to look: Focus regions for visual question answering. *In CVPR*.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Teney, D.; Anderson, P.; He, X.; and Hengel, A. 2018. Tips and tricks for visual question answering: Learnings from the 2017 challenge. *In CVPR*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *arXiv:1706.03762*.

Yang, Z.; He, X.; Gao, J.; Deng, L.; and Smola, A. 2016. Stacked attention networks for image question answering. *In CVPR*.

Yu, D.; Fu, J.; Mei, T.; and Rui, Y. 2017a. Multi-level attention networks for visual question answering. *In CVPR*.

Yu, Y.; Ko, H.; Choi, J.; and Kim, G. 2017b. End-to-end concept word detection for video captioning, retrieval, and question answering. *In CVPR*.

Yu, Z.; Yu, J.; Cui, Y.; Tao, D.; and Tian, Q. 2019. Deep modular co-attention networks for visual question answering. *In CVPR*.

Zhao, Z.; Yang, Q.; Cai, D.; He, X.; and Zhuang, Y. 2017. Video question answering via hierarchical spatio-temporal attention networks. *In IJCAI*.

Zhao, Z.; Jiang, X.; Cai, D.; Xiao, J.; He, X.; and Pu, S. 2018. Multi-turn video question answering via multi-stream hierarchical attention context network. *In IJCAI*.

Zhu, Y.; Groth, O.; Bernstein, M.; and Li, F. 2016. Visual7w: Grounded question answering in images. *In CVPR*.

交叉/综合/新兴领域

WWW 2020

Adversarial Multimodal Representation Learning for Click-Through Rate Prediction

Xiang Li, Chao Wang, Jiwei Tan, Xiaoyi Zeng, Dan Ou, Bo Zheng
Alibaba Group, Hangzhou & Beijing, China
{leo.lx,xiaoxuan.wc,jiwei.tjw,yuanhan,oudan.od,bozheng}@alibaba-inc.com

ABSTRACT

For better user experience and business effectiveness, Click-Through Rate (CTR) prediction has been one of the most important tasks in E-commerce. Although extensive CTR prediction models have been proposed, learning good representation of items from multimodal features is still less investigated, considering an item in E-commerce usually contains multiple heterogeneous modalities. Previous works either concatenate the multiple modality features, that is equivalent to giving a fixed importance weight to each modality; or learn dynamic weights of different modalities for different items through technique like attention mechanism. However, a problem is that there usually exists common redundant information across multiple modalities. The dynamic weights of different modalities computed by using the redundant information may not correctly reflect the different importance of each modality. To address this, we explore the complementarity and redundancy of modalities by considering modality-specific and modality-invariant features differently. We propose a novel Multimodal Adversarial Representation Network (MARN) for the CTR prediction task. A multimodal attention network first calculates the weights of multiple modalities for each item according to its modality-specific features. Then a multimodal adversarial network learns modality-invariant representations where a double-discriminators strategy is introduced. Finally, we achieve the multimodal item representations by combining both modality-specific and modality-invariant representations. We conduct extensive experiments on both public and industrial datasets, and the proposed method consistently achieves remarkable improvements to the state-of-the-art methods. Moreover, the approach has been deployed in an operational E-commerce system and online A/B testing further demonstrates the effectiveness.

CCS CONCEPTS

• **Information systems** → **Content ranking**; **Online shopping**; **Recommender systems**; **Content analysis and feature selection**; **Data encoding and canonicalization**.

KEYWORDS

multimodal learning, adversarial learning, recurrent neural network, attention, representation learning, e-commerce search

This paper is published under the Creative Commons Attribution 4.0 International (CC-BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.
WWW '20, April 20–24, 2020, Taipei, Taiwan
© 2020 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC-BY 4.0 License.
ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380163

ACM Reference Format:
Xiang Li, Chao Wang, Jiwei Tan, Xiaoyi Zeng, Dan Ou, Bo Zheng. 2020. Adversarial Multimodal Representation Learning for Click-Through Rate Prediction. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3366423.3380163

1 INTRODUCTION

Large E-commerce portals such as Taobao and Amazon are serving hundreds of millions of users with billions of items. For better user experience and business effectiveness, Click-Through Rate (CTR) prediction has been one of the most important tasks in E-commerce. As widely studied by both academia and industry, extensive CTR prediction models have been proposed. Nevertheless, it is also an effective way to improve the CTR prediction accuracy through better mining and leveraging the multimodal features of items, since an item in E-commerce usually contains multiple heterogeneous modalities including IDs, image, title and statistic. Therefore, in this paper we make our effort to improve the CTR prediction accuracy by learning better representations for items of multiple modalities.

To leverage the multiple modalities for better item representations, a straightforward way [20, 38, 39] is to concatenate the multiple modality features, which is equivalent to giving a fixed importance weight to each modality regardless of different items. However, the importance of a modality may be different according to different items, and ideal item representations should be able to weigh different modalities dynamically so that emphasis on more useful signals is possible. For example, in the clothing category, whether users will click an item or not is highly affected by observing the images, so greater importance should be given to the image feature. On the contrary, in the cell phone and grocery food categories, the statistic feature of items reflects the popularity of items, while there is little difference between the images of items. A conceivable improvement [14, 31] is to learn dynamic weights of different modalities and emphasis on more useful signals through technique like attention mechanism. However, a problem is that there usually exists common redundant information across multiple modalities. The dynamic weights of different modalities computed by using the redundant information may not correctly reflect the different importance of each modality.

Due to the above reason, our motivation is that the weight of completely independent information across modalities should be dynamic, and the weight of information shared by different modalities should be fixed. To address this, we divide multiple modality features into modality-specific (exist in one modality, and should have dynamic weights) and modality-invariant features (redundant in different modalities, and should have fixed weights). Take a dress which is displayed by an image with a title *Girls Ballet Tutu*

Zebra Hot Pink for example. The item involves a latent semantic feature of the material, such as yarn, which can be expressed by its image while not involved in its title, so the material feature is considered as the modality-specific (latent) feature for this example. The item also involves a common latent semantic feature of color (hot pink) in the subspace of both its image and title features, so the color feature is considered as the modality-invariant (latent) feature for this example. The key idea is that modality-specific features provide an effective way to explore dynamic contributions of different modalities, while modality-invariant features should have a fixed contribution and can be used as supplementary knowledge for comprehensive item representations.

To the best of our knowledge, this is the first work that learns multimodal item representations by exploiting modality-specific and modality-invariant features differently. To achieve this, we propose a Multimodal Adversarial Representation Network (MARN) to deal with this challenging task. In MARN, a modality embedding layer extracts embedding vectors from multiple modalities and decomposes each embedding vector into modality-specific and modality-invariant features. Then, a multimodal attention network calculates the weights of multiple modalities for each item according to its modality-specific features. Also, a multimodal adversarial network learns modality-invariant representations where a double-discriminators strategy is introduced. The double-discriminators strategy is designed to identify the potential modalities involving common features across modalities and drive knowledge transfer between modalities. Finally, we achieve multimodal item representations by combining both modality-specific and modality-invariant representations. The contributions of the paper are summarized as:

- The proposed MARN introduces a novel multimodal representation learning method for multimodal items, which can improve the CTR prediction accuracy in E-commerce.
- We explore the complementarity and redundancy of modalities by considering modality-specific and modality-invariant features differently. To achieve discriminative representations, we propose a multimodal attention fusion network. Moreover, to achieve common representations across modalities, we propose a double-discriminators multimodal adversarial network.
- We perform extensive experiments on both public and industrial datasets. MARN significantly outperforms the state-of-the-art methods. Moreover, the approach has been deployed in an operational E-commerce system and online A/B testing further demonstrates the effectiveness.

2 RELATED WORK

2.1 Multimodal Learning

Representing raw data in a format that a computational model can work with has always been a big challenge in machine learning [1]. Multimodal representation learning methods aim to represent data using information from multiple modalities. Neural networks have become a very popular method for unimodal representations [2, 7]. They can represent visual or textual data and are increasingly used in the multimodal domain [19, 22]. To construct multimodal representations using neural networks, each modality starts with several individual neural layers followed by a hidden layer that projects the modalities into a joint space [33]. The joint multimodal representations are then passed through extra multiple hidden layers or used directly for prediction.

The above multimodal learning methods typically treat different modalities equally by concatenation, which is equivalent to giving the fixed weight for each modality. In practical, item representation learning methods should be capable of assigning dynamic importance weights to each modality according to different items. Therefore, we propose a multimodal attention network to learn different weights of multiple modalities for each item, so that better item representations can be achieved by the weighted combination.

2.2 Adversarial Transfer Learning

Features of multiple modalities may contain redundant information, which should be eliminated when computing the different contributions of modalities. To exploit the redundant information, the common subspace across different modalities should be exploited. Adversarial transfer learning [4, 8] is inspired by Generative Adversarial Nets [9], which enables domain adaptation in deep networks that can be trained on a large amount of labeled data from the source domain and unlabeled data from the target domain. Wang et al. [30] propose an adversarial cross-modal retrieval method, which seeks an effective common subspace based on adversarial learning. Pei et al. [23] capture multimodal structures to enable fine-grained alignment of different data distributions based on multiple domain discriminators. Unlike previous adversarial transfer methods that solely match distribution at domain-level, Xie et al. [34] propose to match distribution at class-level and align features semantically without any target labels. Yu et al. [36] propose a novel Dynamic Adversarial Adaptation Network to learn domain-invariant representations while quantitatively evaluate the relative importance of the marginal and conditional domain distributions.

The above adversarial learning methods are still a one-to-one adversarial paradigm, although the items in E-commerce involve multiple modalities. Moreover, different modality features involve different degrees of common features. Therefore, we propose a novel double-discriminators multimodal adversarial network to learn common latent subspace across multiple modalities.

2.3 Context Aware Personalization Model

In recent years, there have been growing numbers of researches on the personalization based on deep neural networks for recommending music [29], news [21], videos [5], and jobs [3].

Personalization for search and recommendation is typically based on user behaviors, where the RNN-based model is a good choice due to the sequential characteristic of user behaviors. Hidasi et al. [11] apply the RNN model to infer users' future intentions based on their previous click behavior sequence. Tan et al. [28] present several extensions to the basic RNN model to enhance the performance of recurrent models. Ni et al. [20] adopt LSTM and the attention mechanism to model the user behavior sequence. Compared to sequence-independent approaches, these methods can significantly improve the CTR prediction accuracy and most of these techniques have been deployed in real-world applications [20, 32, 38, 39].

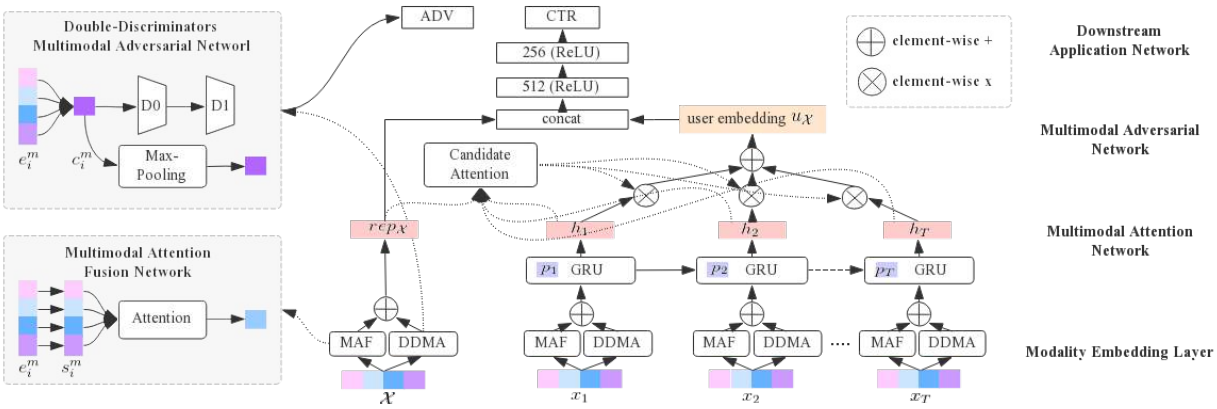


Figure 1: The architecture of MARN.

3 OUR PROPOSED METHOD

The typical framework of CTR prediction is to take a user behavior sequence $u = \{x_1, x_2, \ldots, x_T\}$ and the candidate item $\mathcal{X}$ as inputs, and aims to learn the probability that u clicks $\mathcal{X}$. Items are mostly represented by ID to learn its representation through embedding. When considering multiple modalities, previous works either first extract multimodal embeddings and fuse these embeddings through concatenation [20, 38, 39], or dynamically give different weights for each modality through an attention mechanism [14, 31]. Differently, we propose to divide multiple modality features into modality-specific and modality-invariant features. The overall architecture of the proposed MARN is presented in Figure 1. In our work, *Multimodal Attention Network* learns the dynamic weights of multiple modalities for each item according to its modality-specific features. *Multimodal Adversarial Network* learns common representations across multiple modalities for more comprehensive item representations. Besides the above two components, the *Modality Embedding Layer* and *Downstream Application Network* of our model are similar to related CTR prediction models.

3.1 Modality Embedding Layer

Each item in the user behavior sequence is associated with a behavior property and forms a user-item interaction pair $< x_i, p_i >$. The modality embedding layer is applied upon the sparse features of each item in the user behavior sequence and its behavior property to compress them into low-dimensional vectors.

3.1.1 Item Modality Embedding. An item x_i is represented by the multimodal information: i) **IDs**, unordered discrete feature including item ID, shop ID, brand ID and category ID; ii) **image**, pixel-level visual information; iii) **title**, word sequence; iv) **statistic**, historical exposure, click, transaction order/amount.

IDs: The IDs feature is represented as $[x_i^{id(1)}, \ldots, x_i^{id(F)}]$, which is a multi-hot vector ($[\cdot, \cdot]$ denotes vector concatenation). F is the number of IDs feature, and $x_i^{id(f)}$ is the f^{th} feature which is a one-hot vector representing an ID like item ID or shop ID. The embedding layer transforms the multi-hot vector into a low-dimensional vector e_i^{id} with an embedding lookup table, as shown in Equation 1:

$$e_i^{id} = [W_{emb}^1 x_i^{id(1)}, \ldots, W_{emb}^F x_i^{id(F)}], W_{emb}^f \in R^{d_{emb}^f \times V_f} \quad (1)$$

where d_{emb}^f is the dimension and V_f is the vocabulary size.

Image: Recent progress in computer vision shows that the learned semantic embeddings from the pixel-level visual information for classification tasks have good generalization ability [10, 27]. Thus for an input image, the image embedding is the output of pre-trained VGG16 [27] model with the last two layers for classification purpose removed, which result in a 4096-dimensional vector.

Title: The title containing h words (padded where necessary) is represented as an $h \times d_{emb}^{term}$ matrix, where each word is represented as a d_{emb}^{term}-dimensional vector. As suggested by Word2Vec [18], we set $d_{emb}^{term} = 300$. We design a fast convolutional network for title embeddings which uses multiple filters with varying window sizes $n = 2, 3, 4$ to obtain the n-gram features, following Kim [13].

Statistic: We found it difficult in learning a good embedding directly on continuous statistic feature. Following Yuanfei et al. [37], we adopt multi-granularity discretization, which discretizes each numerical feature into two, rather than only one, categorical features, each with a different granularity, ranging from 0 to 9 and 0 to 99 according to the number of items. We then perform categorial feature lookups to obtain two 8-dimensional vectors.

3.1.2 Behavior Property Embedding. The behavior property p_i describes the type and time of items in the user behavior sequence. Behavior type is a one-hot vector representing click, add-to-cart or purchase. Behavior time is a statistic feature indicating the seconds from the time it happens to the current recommendation time. We transform each behavior property into embedding $p_i = e_i^p$ and treat the the behavior property embedding p_i as a strong signal to reflect the importance of each behavior.

3.2 Multimodal Attention Network

Utilizing multiple modality features is often effective to improve the performance of CTR tasks. A straightforward way [20, 38, 39] is to concatenate the multiple modality features, which is equivalent to giving a fixed importance weight to each modality regardless of

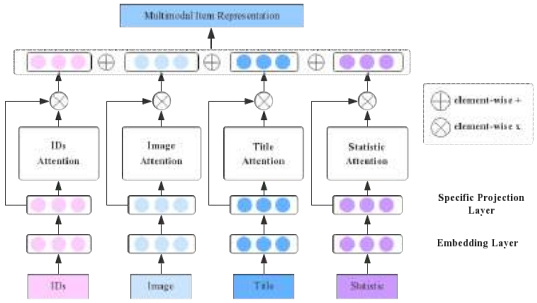


Figure 2: Illustration of multimodal attention fusion.

different items. A conceivable improvement [14, 31] is to dynamically distinguish the contributions of different modalities through an attention mechanism. However, features of multiple modalities may contain redundant information, which should be eliminated when computing the different contributions of modalities.

Due to the above reason, we propose to explore the complementarity and redundancy of modalities by considering modality-specific and modality-invariant features differently. More precisely, the distributions of modality-specific features are separated, whereas the distributions of modality-invariant features are close to each other. To address this, we decompose each item embedding vector into modality-specific and modality-invariant features, i.e., s_i^m, c_i^m. More specifically, we project each item modality embedding into vector e_i^m of the same dimension, and then apply two nonlinear projection layers to obtain two 256-dimensional vectors according to $s_i^m, c_i^m = \mathcal{S}_m(e_i^m), \mathcal{I}(e_i^m)$. $\mathcal{S}_m(\cdot)$ is an independent projection matrix of each modality m for extracting modality-specific features, while the projection matrix $\mathcal{I}(\cdot)$ is shared across modalities for learning modality-invariant features.

Therefore, we propose a multimodal attention fusion (MAF) network shown in Figure 2, to learn dynamic weights according to its modality-specific features s_i^m and then obtain the modality-specific representation s_i by the weighted summation:

$$s_i = \sum_{m=1}^{M} atten_i^m \odot s_i^m \tag{2}$$

$$atten_i^m = tanh(W_m^\top \cdot s_i^m + b_m) \tag{3}$$

where $atten_i^m$ controls the attention weight of modality m, and $\odot$ denotes element-wise multiplication. The parameters $W_m^\top$ is matrix with the size of $d \times d$, and b_m is a vector with the size of $d \times 1$, where $d = 256$. The vector attention adjusts the importance weights of each dimension for the modality-specific features.

3.3 Multimodal Adversarial Network

Since the weights of the $\mathcal{S}_m(\cdot)$ and the $\mathcal{I}(\cdot)$ are only supervised by the label of the CTR task, we cannot expect that MARN has successfully learned the modality-specific and modality-invariant features. The distributions of modality-specific features extracted by $\mathcal{S}_m(\cdot)$ should be kept as far as possible, meanwhile, the distributions of modality-invariant features extracted by $\mathcal{I}(\cdot)$ should be drawn as close as possible. To address this, we design a multimodal adversarial network, which makes the modality-invariant feature extractor compete with a modality discriminator, in order to learn a common subspace across multiple modalities. Moreover, we propose a modality-specific discriminator to supervise the modality-specific feature extractor, so that the learned multimodal information has very good separability, leading to the modality-specific features.

3.3.1 Cross-Modal Adversarial. The general cross-modal adversarial method [30] seeks an effective common subspace across modalities, which is built around a minimax game, as:

$$\begin{aligned} \min_{\mathcal{I}} \max_{\mathcal{D}} \mathcal{L}_D = & \mathbb{E}_{x \sim px^1} [log(\mathcal{D}(\mathcal{I}(x)))] \\ & + \mathbb{E}_{x \sim px^2} [log(1 - \mathcal{D}(\mathcal{I}(x)))] \end{aligned} \tag{4}$$

where px^1 and px^2 are distributions of two modalities.

For $\mathcal{I}$ fixed, the optimal discriminator D is:

$$D^*_{\mathcal{I}}(x) = \frac{px^1(x)}{px^1(x) + px^2(x)} \tag{5}$$

Given the optimum D^*, the minimax game of Equation 4 is:

$$\mathcal{L}_D = -log(4) + 2 \cdot JSD(px^1 || px^2) \tag{6}$$

Since the Jensen-Shannon divergence (JSD) between two distributions is always non-negative and achieves zero only when they are equal, $\mathcal{L}_D$ obtains the global minimum on $px^1 = px^2$ [9], which is the goal of cross-modal adversarial learning.

3.3.2 Double-Discriminators Multimodal Adversarial. The above cross-modal adversarial method focuses on a one-to-one adversarial paradigm, while the items in E-commerce involve multiple modalities. Moreover, different modality features involve different degrees of common latent features, and the degrees can be treated as guidance to confuse the modality discriminator towards better discrimination performance. For example, the item ID embedding should be assigned relatively small weights due to the uniqueness. On the contrary, the image and title embeddings that involve much potential common subspace should be emphasized for further confusing the modality discriminator.

To address this, we propose a novel double-discriminators multimodal adversarial (DDMA) network. More precisely, we bring two contributions. First, we expand the original one-to-one adversarial paradigm to multimodal scenarios. Second, we introduce a double-discriminators strategy. The first discriminator identifies modality-invariant features that are potentially from the common latent subspace across multiple modalities and also emphasizes the identified modality-invariant features for further confusing the second discriminator. Meanwhile, the second discriminator drives knowledge transfer between modalities so as to learn common latent subspace across multiple modalities.

The first discriminator D_0 is an M-class classifier and is given by $D_0(x) = softmax(\mathcal{D}_0(x))$, where $x = c = \mathcal{I}(e)$, e is the modality embedding, $\mathcal{I}$ is the invariant projection layer and c is the projected modality-invariant feature. Suppose that D_0 has converged to its optimum D_0^* and x belongs to modality i, $D_0^i(x)$ gives the likelihood of x in terms of modality i. If $D_0^{i*}(x) \approx 1$, then x highly involves few common features across modalities, since it can be perfectly discriminated from other modalities. Thus, the degree of x as $w^i(x)$

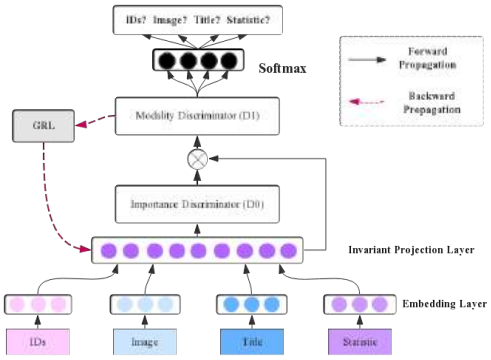


Figure 3: Illustration of double-discriminators multimodal adversarial network.

contributing to the modality-invariant feature should be inversely related to $D_0^i(x)$ according to $w^i(x) = 1 - D_0^i(x)$.

The second discriminator D_1 is also an M-class classifier, which is applied to reduce the JSD between modalities. After adding the degrees to the modality features for the discriminator D_1, the objective function of weighted multimodal adversarial network is:

$$\min_{\mathcal{I}} \max_{\mathcal{D}_1} \mathcal{L}_{D_1} = \mathbb{E}_{e \sim pe^m} \sum_{m=1}^{M} \left[w^m(x) log(\mathcal{D}_1(\mathcal{I}(e))) \right] \tag{7}$$

where $w^m(x) = w^m(\mathcal{I}(e))$ is a constant and independent of $\mathcal{D}_1$.

For fixed $\mathcal{I}$ and $\mathcal{D}_0$, the optimal output for D_1^i is:

$$D_1^{i*}(x) = \frac{w^i(x) px^i(x)}{\sum_{m=1}^{M} w^m(x) px^m(x)} \tag{8}$$

Given the optimum $D_1^*(x)$, the minimax game of Equation 7 is:

$$\begin{aligned}
\mathcal{L}_{D_1} &= \sum_{i=1}^{M} \int_x w^i(x) px^i(x) log(\frac{w^i(x) px^i(x)}{\sum_{m=1}^{M} w^m(x) px^m(x)}) dx \\
&= \sum_{i=1}^{M} \int_x w^i(x) px^i(x) log(\frac{w_i(x) px^i(x)}{\sum_{m=1}^{M} \frac{w^m(x) px^m(x)}{M}}) dx - M log(M) \\
&= \sum_{i=1}^{M} KL\left(w^i(x) px^i(x) \parallel \sum_{m=1}^{M} \frac{w^m(x) px^m(x)}{M} \right) - M log(M) \\
&= M \cdot JSD_{\frac{1}{M}, \ldots, \frac{1}{M}} \left(w^1(x) px^1(x), \cdots, w^M(x) px^M(x) \right) - M log(M)
\end{aligned} \tag{9}$$

As deduced in Equation 9, the minimax game of Equation 7 is essentially reducing JSD between M weighted modality features and obtains the optimum on $w^1(x)px^1(x) = \cdots = w^M(x)px^M(x)$.

Since $w^m(x) = w^m(\mathcal{I}(e^m))$ and $px^m(x) = c^m = \mathcal{I}(e^m)$, we can achieve modality-invariant representation through a max-pooling operation over the weighted modality-invariant features and take the maximum value, as $c_i = max\{w^1 c^1, \ldots, w^M c^M\}$.

3.3.3 Modality-Specific Discriminator. To encourage the modality-specific features s_i^m to be discriminated between multiple modalities, we propose a modality-specific modality discriminator D_s on the modality-specific features,where the discriminating matrix of D_s is shared with the second discriminator D_1.

Finally, we achieve the complementarity and redundancy of multiple modalities, resulting in the modality-specific representations s_i and modality-invariant representations c_i. Then, the multimodal item representation rep_i is produced by the element-wise summation of modality-specific and modality-invariant representations.

3.4 Downstream Application Network

3.4.1 Behaviors Integrating Layer. With the proposed item representation learning model, we can integrate the user behaviors with various kinds of RNN-based model. In this paper, we choose GRU [11] to model the dependencies between the user behaviors. However, GRU lacks of capturing two characteristics: i) different behavior type and time contribute differently to extracting users' interests; ii) whether to click a candidate item or not highly depends on the most related behaviors with respect to the candidate item. Moreover, extracting the key information in the complex behaviors further contributes towards learning better item representations.

Therefore, we propose an Attentional Property GRU (APGRU) layer. First, we modify the update gate of GRU to extract the most interested behavior state according to $u_t = \sigma(W^u rep_t + P^u p_t + U^u h_{t-1} + b^u)$, where p_t is the behavior property, rep_t is the t^{th} item representation, and σ is the sigmoid function.

Second, we concatenate the hidden state h_t and the candidate item $rep_{\mathcal{X}}$ and apply a Multilayer Perceptron as the candidate attention network. Then we calculate the normalized attention weights, and the user embedding $u_{\mathcal{X}}$ can be calculated as the weighted summation of the output hidden states.

3.4.2 CTR Prediction Layer. In this paper, we set the prediction layer to be a deep network with point-wise loss for the CTR task. The overall objective functions are:

$$\begin{aligned}
&\min_{\mathcal{F}, \mathcal{I}, \mathcal{S}_m, \mathcal{E}} \mathcal{L}_{CTR} = -\frac{1}{N} \sum_{n=1}^{N} [y_n log(\mathcal{F}(u_n, rep_n)) \\
&\qquad + (1 - y_n) log(1 - \mathcal{F}(u_n, rep_n))] \\
&\min_{\mathcal{D}_s, \mathcal{S}_m} \mathcal{L}_{D_s} = \lambda \left(-\mathbb{E}_{e, y \sim pe^{e,y}} \sum_{m=1}^{M} \mathbb{I}_{[m=y]} log \mathcal{D}_s(\mathcal{S}_m(e)) \right) \\
&\min_{\mathcal{D}_0} \mathcal{L}_{D_0} = -\mathbb{E}_{c, y \sim pc^{c,y}} \sum_{m=1}^{M} \mathbb{I}_{[m=y]} log \mathcal{D}_0(c) \\
&\min_{\mathcal{I}} \max_{\mathcal{D}_1} \mathcal{L}_{ADV} = \lambda \left(\mathbb{E}_{e \sim pe^m} \sum_{m=1}^{M} \left[w^m(e) log(\mathcal{D}_1(\mathcal{I}(e))) \right] \right)
\end{aligned} \tag{10}$$

where u_n denotes the user embedding of the n^{th} instance. y_n denotes the label, which is set to 1 if user clicked item and 0 otherwise. λ is a tradeoff hyper-parameter that controls the learning procedure of modality-specific and modality-invariant representations.

We optimize the four objective functions, i.e, $\mathcal{L}_{CTR}$, $\mathcal{L}_{D_s}$, $\mathcal{L}_{D_0}$, and $\mathcal{L}_{ADV}$ to update different layers in MARN simultaneously. First, we minimize $\mathcal{L}_{CTR}$ to optimize the overall prediction performance. Second, we minimize $\mathcal{L}_{D_s}$ to learn the modality-specific projection layers $\mathcal{S}_m$, while the gradient of $\mathcal{D}_s$ will not be back-propagated for updating the embedding layer $\mathcal{E}$. Then, we minimize the first discriminator $\mathcal{L}_{D_0}$ to identify the potential common subspace across

multiple modalities. Since $\mathcal{D}_0$ are learned on un-weighted modality-invariant features and would not be a good indicator, the gradient of $\mathcal{D}_0$ will not be back-propagated for updating invariant projection layer $\mathcal{I}$. Finally, the second discriminator $\mathcal{D}_1$ plays the minimax game with the modality-invariant feature for updating $\mathcal{I}$. To solve the minimax game between $\mathcal{I}$ and $\mathcal{D}_1$, we adopt an end-to-end architecture through a gradient reversal layer (GRL) [8], so the sub-networks are trained jointly and boost each other.

4 EXPERIMENTS

In this section, we present our experiments in detail, including dataset, evaluation metric, experimental setup, model comparison, and the corresponding analysis. Experiments on a public dataset with user behaviors and a dataset collected from Taobao Search system demonstrate MARN achieves remarkable improvements to the state-of-the-art methods on the CTR task.

4.1 Datasets and Experimental Setup

Amazon Dataset: We use the Amazon dataset in Zhou et al. [39], which comprises several subsets of the amazon product data [16] and have sequential user behavior sequence. For a behavior sequence $\{x_1, x_2, \ldots, x_j, \ldots, x_T\}$, in the training phase, we predict the $j{+}1^{th}$ behavior with the first j behaviors, where $j = 1, 2, \ldots, T-2$. In the test phase, we use the first $T-1$ behaviors to predict the last one. The feature set we use contains item ID, category ID, image feature (extracted using pre-trained VGG16 model), statistic, together with 300-dimensional GloVe vectors [24] for the terms of the title. The statistics of Amazon dataset is shown in Table 1.

Taobao Dataset: We collect 8×10^9 samples from the daily exposure and click logs in Taobao Search system, which consist of the user behaviors and the labels (i.e., exposure or click) for the CTR task. The user behavior type consists of click, add-to-cart and purchase. The feature set of an item contains item ID, shop ID, brand ID, category ID, image feature (extracted using pre-trained VGG16 model), statistic and title, also with a behavior type and time. For the production scenario, we use the samples across 7 days for training and evaluate on the samples of the next day. The statistics of Taobao dataset is shown in Table 1.

We set the hyper-parameter configuration as follows. For Amazon and Taobao datasets, the dimensions of the item ID, shop ID, brand ID and category ID are set to be the same as in [20], which are 32, 24, 24 and 16, respectively. The vocabulary sizes of the embeddings are set according to the statistics of the datasets. Since we focus on multimodal item representation learning in this paper, we make the hidden units of the prediction layer as fixed values for all models, with [128,64] for the Amazon dataset and [512,256] for the Taobao dataset. The activation functions are ReLU. The model is trained on 32-sized batches for Amazon dataset and 1024-sized batches for Taobao dataset. We use AdaGrad [6] as the optimizer with the learning rate of 0.1. We gradually change λ from 0 to λ_0 following the schedule $\lambda = \lambda_0(\frac{2}{1+\exp(-\gamma \cdot p)} - 1)$, where γ is set to 10 in all experiments (the schedule was not optimized/tweaked), and p linearly changes from 0 to 1 in the training progress. The hyper-parameter λ_0 is tuned ranging from 0.01 to 1.8 and MARN achieves the best performance at $\lambda_0 = 0.05$.

4.2 Offline Comparison of Different Methods

To show the effectiveness of our method, we compare our method with three groups of nine baselines. The first group consists of models before deep networks for the CTR prediction task.

***LR* [17]:** Logistic regression (LR), which is a widely used shallow model before deep networks for CTR prediction task.

***FM* [25]:** Factorization Machine, which models both first-order feature importance and second-order feature interactions.

The second group contains concatenation-based mutlimodal fusion methods for mutlple modalities of items, which are the mainstream CTR prediction models.

***YoutubeNet* [5]:** A deep model proposed to recommend videos in YouTube, which gets user representations by simply averaging the item embeddings in the user behavior sequence.

***xDeepFM* [15]:** A compressed interaction network, which aims to automatically learn high-order feature interactions in both explicit and implicit fashions.

***DUPN* [20]:** A general user representation learning method for E-commerce scenarios, which adopts LSTM and attention mechanism to model the user behavior sequence.

***DIEN* [38]:** A two-layer RNN structure with attention mechanism. It uses the calculated attention values to control the second RNN layer to model drifting user interests.

The third group is formed of the state-of-the-art multimodal methods, which learn unified representations from multiple modalities and utilize variety of fusion techniques to improve the performance of the CTR prediction task.

***DMF* [12]:** A hierarchically multimodal joint network, which densely correlates the representations of different modalities layer-by-layer, where the shared layer not only models the correlation in the current level but also depends on the lower one.

***MMSS* [14]:** A modality-based attention mechanism model with image filters to selectively use visual information.

***NAML* [31]:** An attentive multi-view learning model, which incorporates titles, bodies and categories as different views of news, and applies attention mechanism to news encoder to select important views for learning informative news representations.

Note that, the inputs of LR and FM contains the IDs feature (item ID, shop ID, brand ID and category ID). All compared neural network models are fed with the same features as MARN for fair comparison. Finally, we conduct the significance test to verify the statistical significance of the performance improvement of our model against the baseline models.

4.3 Evaluation Metrics

Following [15, 20, 35, 38], we adopt AUC as the evaluation metric, which is widely used in CTR prediction tasks. Average AUC is evaluated as following:

$$AUC = \frac{1}{|U^{Test}|} \sum_{u \in U^{Test}} \frac{1}{|I_u^+|\,|I_u^-|} \sum_{i \in I_u^+} \sum_{j \in I_u^-} \delta(\hat{p_{u,i}} > \hat{p_{u,j}}) \quad (11)$$

where $\hat{p_{u,i}}$ is the predicted probability that a user $u \in U^{Test}$ may click on the item i in the test set and $\delta(\cdot)$ is the indicator function. I_u^+ is the item set clicked by the user, and I_u^- is the item set that the user does not click.

Dataset	# Users	# Items	# Shops	# Brands	# Categories	# Images	# Titles	# Samples
Electro. (Amazon)	192403	63001	-	-	801	63001	63001	2993570
Clothe. (Amazon)	39387	23033	-	-	484	23033	23033	477862
Taobao	0.25 billion	0.8 billion	17 million	1.0 million	20 thousand	0.8 billion	0.8 billion	64 billion

Table 1: Statistics of Amazon and Taobao datasets.

Group	Method	Electro. AUC	Clothe. AUC	Taobao AUC
1	*LR*	0.7272	0.7143	0.7011
	FM	0.7369	0.7224	0.7091
2	*YoutubeNet*	0.7675	0.7593	0.7241
	xDeepFM	0.7762	0.7691	0.7287
	DUPN	0.7904	0.7774	0.7348
	DIEN	0.7923	0.7793	0.7363
3	*DMF*	0.7924	0.7797	0.7371
	MMSS	0.7934	0.7806	0.7380
	NAML	0.7958	0.7825	0.7398
	MARN	**0.8034***	**0.7909***	**0.7486***

Table 2: AUC on Amazon and Taobao datasets. Here, * indicates statistical significance improvement compared to the best baseline (NAML) measured by t-test at p-value of 0.05.

4.4 Results on Amazon and Taobao Datasets

All experiments are repeated 5 times and averaged metrics are reported in Table 2. The influence of random initialization on AUC is less than 0.0002. From Table 2, we can tell that MARN improves the performance significantly against all the baselines and achieves state-of-the-art performance.

In the majority of the cases, non-neural network models in Group 1, i.e., LR and FM perform worse than neural network models, which demonstrates the power of deep learning.

Comparing with the concatenation-based methods of Group 2, multimodal item learning methods of Group 3 outperforms all of them, which reflects the benefits of well designed multimodal fusion methods for learning good representation of items.

By comparing MARN with the models in Group 3, i.e., DMF, MMSS and NAML, although all the baselines are proposed to deal with multimodal information, MARN has better performance on the multimodal items in E-commerce. Since DMF was proposed to hierarchically joint multimodal information, it has the same issue as the concatenation-based methods which may ignore the different contributions of multiple modalites. MMSS was proposed to selectively use visual information with image filters, which may not utilize all image information effectively. Though the attention mechanism of NAML improves the performance, it has not explored the complementarity and redundancy of modalities by considering modality-specific and modality-invariant features differently.

Finally, the improvement of MARN over the best baseline model NAML are 0.76%, 0.84% and 0.88% AUC scores. As reported in [39], compared with YoutubeNet, DIN improves AUC scores by 1.13% and the improvement of online CTR is 10.0%, which means a small improvement in offline AUC is likely to lead to a significant increase in online CTR. Considering the massive size of Taobao Search system, the lift of AUC scores by MARN may bring considerable CTR improvement and additional income each year.

4.5 Ablation Study

In this section, we design ablation experiments to study how each component in MARN contributes to the final performance.

BaseModel*:** A model with unimodal item features which uses GRU to integrate the behavior embeddings. We conduct ***BaseModel+IDs with IDs feature (item ID, shop ID, brand ID and category ID) and ***BaseModel+IMAGE*** with image feature.

***BaseModel+CONC*:** A concatenation model, where concatenation is performed in the fusion layer by element-wise summing up all the projected modality features (IDs, image, title and statistic).

***BaseModel+MAF*:** A sub-model of MARN with the multimodal attention fusion network (MAF).

***BaseModel+MAF+ADV*:** A sub-model of MARN with both MAF and the original adversarial transfer network (ADV)[30].

***BaseModel+MAF+DDMA*:** A sub-model of MARN with both MAF and the proposed double-discriminators multimodal adversarial network (DDMA).

***MARN*:** The entire proposed model with MAF, DDMA and Attentional Property GRU (APGRU).

As shown in Table 3, removing any component in MARN leads to a drop in performance. Further than that, we have the following observations. First, with the help of bringing in more modalities (i.e., image, title, and statistic), CONC outperforms the unimodal baselines by 0.35%, 0.63% and 0.65% in AUC. Second, MAF outperforms CONC about 0.32%, 0.35% and 0.39% in terms of AUC, which reflects that learning dynamic contributions of different modalities for multimodal item representations further leads to better performance. Third, both the original adversarial network and the proposed double-discriminators multimodal adversarial network can boost the CTR task performance, which reveals the benefits of identifying the potential common subspace across multiple modalities by multimodal adversarial learning. Moreover, the double-discriminators strategy (DDMA) outperforms ADV by 0.39%, 0.51% and 0.52% AUC scores. It demonstrates that emphasizing the identified modality-invariant features by the first discriminator and further confusing the second discriminator are both crucial for learning better common latent subspace across multiple modalities. APGRU further provides AUC gain of 0.2%, which results in the final performance.

4.6 Item Representations Validation

4.6.1 Generalization on Unpopular Items. To verify what kinds of items benefit more from MARN, we uniformly divide all items in the test set of Taobao dataset into 10 equal frequency popularity levels according to their frequencies of interactions. The higher thepopularity level, the more popular items. We then compute the

Method	Electro. AUC	Clothe. AUC	Taobao AUC
BaseModel+IDs	0.7897	0.7729	0.7295
BaseModel+IMAGE	0.7631	0.7564	0.6833
BaseModel+CONC	0.7929	0.7792	0.7360
BaseModel+MAF	0.7961	0.7827	0.7399
BaseModel+MAF+ADV	0.7974	0.7838	0.7411
BaseModel+MAF+DDMA	0.8013	0.7889	0.7463
MARN (*MAF+DDMA+APGRU*)	**0.8034**	**0.7909**	**0.7486**

Table 3: AUC on Amazon and Taobao datasets.

AUC at each level and show the results in Table 4. It shows that both methods perform very well at high popularity levels from 7 to 9, while the performances decrease at low popularity levels from 0 to 3. For those less popular items, the unimodal model may not have enough training instances to capture the co-occurrence of the items. Fortunately, with the help of representing items with multimodal information, such as image and title features, MARN achieves extremely remarkable improvements at low popularity levels from 0 to 3. It demonstrates that *BaseModel+IDs* cannot deal with unpopular items well, while *MARN* boosts the generalization of the item representations and achieves better results.

4.6.2 Transfer from Seen Items to Unseen Items. New items usually cause the cold-start problem, and many methods cannot deal with them. The item-based CF (Collaborative Filtering) [26] cannot perceive the items with no historical records, so the result are nearly random items (*RANDOM*). *BaseModel+IDs* can relieve the cold-start items using some generic IDs feature such as brand ID and shop ID. MARN can further address the cold-start problem by constructing an item embedding vector from not only generic IDs feature, but also image and title features. To study item transferability on unseen items, we evaluate the click recall@top-N of the generated candidate set in the next day. The results are shown in Table 5. The classical item-based CF cannot deal with new items. Compared to *BaseModel+IDs*, MARN achieves the improvement of about 2% when N is less than 70 and about 1% when N is beyond 70. Although the effectiveness decreases as N increases, this is rare case because the number of items browsed by users is usually limited.

4.7 Case Study of Multimodal Attention

To verify the effectiveness of the multimodal attention in our model, we visualize several categories in Taobao dataset in Figure 4. We randomly select ten thousand items for each category and each item is represented by multiple modalities including IDs, image, title and statistic. We then calculate the average $l2$-norm of the vector attention of multiple modalities for each category. The heat map matrix represents the attention intensity of each modality with respect to different categories.

We can find that the image feature is highly weighted under categories such as clothing, shoes, and jewelry, while the statistic feature gains more weight in the cell phone and grocery food categories than that of the image feature. As a result, the multimodal attention mechanism in MARN can effectively learn dynamic contributions of different modalities for multimodal item representations.

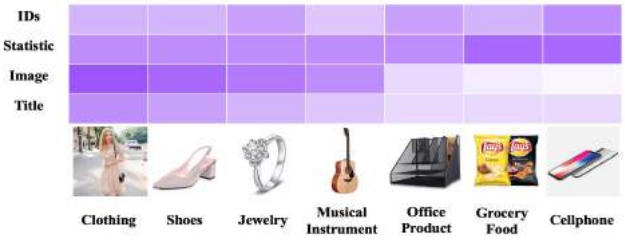


Figure 4: The heat map of attention weights with different item modalities of different categories.

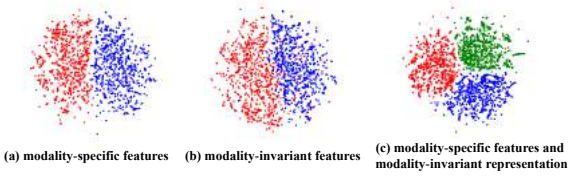
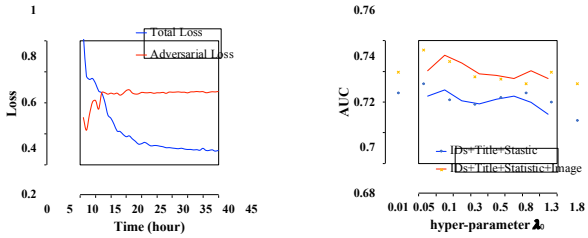


Figure 5: The t-SNE modality visualization. Red points denote the image modality feature, while blue points denote title modality feature. Green points denote the modality-invariant representation obtained by max-pooling.



(a) Convergence. (b) Parameter Sensitivity.

Figure 6: Convergence and parameter sensitivity study.

4.8 Empirical Analysis of Adversarial Network

4.8.1 Modality Visualization. To verify the effectiveness of the proposed multimodal adversarial network, we visualize the t-SNE features for ten thousand randomly selected items in Taobao dataset. The visualized features are modality-specific features (s_i^{image} and s_i^{title}), modality-invariant features (c_i^{image} and c_i^{title}) and modality-invariant representation (c_i). As shown in Figure5(a), s_i^{image} and s_i^{title} are almost completely separated, which reflects that the auxiliary modality discriminator can reduce the redundancy and increase the distinctiveness of the modality-specific features. As shown in Figure5(b), c_i^{image} and c_i^{title} are drawn close to each other, which reveals that the multimodal adversarial mechanism has the ability to decrease the modality gap and align distributions of different modalities. Figure5(c) illustrates that s_i^{image}, s_i^{title} and c_i are completely separated, which means MARN can effectively exploit the complementarity and redundancy of multiple modalities and achieve universal item representations by combining both modality-specific and modality-invariant representations.

Popularity-level	0	1	2	3	4	5	6	7	8	9
BaseModel+IDs	0.672	0.687	0.694	0.717	0.725	0.732	0.736	0.739	0.750	0.752
MARN	0.716	0.712	0.716	0.738	0.743	0.745	0.742	0.741	0.752	0.753
GAP	+4.4%	+2.5%	+2.2%	+2.1%	+1.8%	+1.3%	+0.6%	+0.2%	+0.2%	+0.1%

Table 4: AUC at different popularity levels on Taobao dataset.

Methods (top-*N*)	10	20	30	40	50	60	70	80	90	100	1000
RANDOM	0.01%	0.01%	0.02%	0.02%	0.03%	0.04%	0.05%	0.06%	0.07%	0.07%	0.59%
BaseModel+IDs	5.82%	6.45%	10.13%	11.21%	12.16%	12.81%	13.42%	14.24%	14.43%	14.54%	21.28%
MARN	6.95%	7.97%	12.31%	13.63%	14.86%	15.34%	15.59%	15.75%	15.86%	15.99%	21.99%

Table 5: The click recall @top-N on Taobao dataset.

4.8.2 Convergence and Parameter Sensitivity. As shown in Figure 6(a), along with the training procedure, the total loss of MARN on Taobao dataset converges to a stable state, while the adversarial loss gradually increases, which means the modality discriminator becomes incapable of discriminating between different modalities based on the learned modality-invariant features. Figure 6(b) shows the AUC of MARN on Taobao dataset with the hyper-parameter λ_0 ranging from 0.01 to 1.8. MARN achieves the best performance at $\lambda_0 = 0.05$, which is an appropriate balance.

4.9 Online Evaluation

4.9.1 Online Metrics. Careful online A/B testing in Taobao Search system was conducted for one month. The evaluation metrics are computed over all exposure items returned by the ranking system based on the CTR prediction models (i.e., YoutubeNet, DUPN and MARN). We report two metrics: CTR and GMV (Gross Merchandise Volume). CTR is computed as the ratio between the number of clicked items and the total number of exposure items. GMV is computed by the transaction number of items multiplied by the price of each item.

Method	CTR Improve	GMV Improve
YoutubeNet	0%	0%
DUPN	5.23%	3.17%
MARN	**10.46%**	**5.43%**

Table 6: Online A/B testing.

Due to the policy restrictions, we do not report the absolute numbers for the online metrics. Instead, we report the relative numbers with respect to *YoutubeNet*. As shown in Table 6, compared to *DUPN*, the last version of our online serving model, *MARN* has improved CTR by 5.23% and GMV by 2.26%. Considering the massive size of Taobao Search system, such consistent online improvements are significant. MARN has already been deployed online, serving the main traffic for hundreds of million users with billions of items.

4.9.2 Online Serving. Online serving of industrial deep networks is not an easy job with hundreds of millions of users visiting our system everyday. Even worse, at traffic peak our system serves more than 200,000 users per second. It is required to make real-time CTR predictions with high throughput and low latency. Though MARN is complex, it has additional advantage that the multimodal item representations can be pre-extracted during the stage of model deploying. Since MARN automatically learns the weights of different modalities only according to the item itself, we can perform the subnetwork of MARN to extract the multimodal representation of each item and store the representation vector in the index of the search engine, thereby reducing the latency of online serving. Moreover, several important techniques are deployed for accelerating online serving of industrial deep networks: i) quantization for recurrent neural networks, which adopts multi-bit quantization strategies to accelerate online inference; ii) heterogeneous calculations including GPUs and ALI-FPGAs, which can accelerate matrix computations. In conclusion, optimization of these techniques doubles the QPS (Query Per Second) capacity of a single machine practically. Online serving of MARN also benefits from this.

5 CONCLUSIONS

This paper proposes a novel multimodal representation learning method for multimodal items, which can improve CTR and further boost GMV in E-commerce. We explore the complementarity and redundancy of multiple modalities by considering modality-specific and modality-invariant features differently. To achieve discriminative representations, we propose a multimodal attention fusion network. Moreover, to achieve a common latent subspace across modalities, we propose a double-discriminators multimodal adversarial network. We perform extensive offline experiments on Amazon and Taobao datasets to verify the effectiveness of the proposed method. MARN consistently achieves remarkable improvements to the state-of-the-art methods. Moreover, the approach has been deployed in an operational E-commerce system and online A/B testing further demonstrates the effectiveness.

ACKNOWLEDGMENTS

We thank colleagues of our team - Zhirong Wang, Tao Zhuang, Heng Li, Ling Yu, Lei Yang and Bo Wang for valuable discussions and suggestions on this work. We thank our search engineering team for the large scale distributed machine learning platform of both training and serving. We also thank scholars of prior works on multimodal representation learning and recommender system. We finally thank the anonymous reviewers for their valuable comments.

REFERENCES

[1] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2018. Multimodal machine learning: A survey and taxonomy. *TPAMI* (2018).
[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *TPAMI* 35, 8 (2013), 1798–1828.
[3] Fedor Borisyuk, Liang Zhang, and Krishnaram Kenthapadi. 2017. LiJAR: A system for job application redistribution towards efficient career marketplace. In *SIGKDD*. 1397–1406.
[4] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. 2017. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, Vol. 1. 7.
[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Recsys*. 191–198.
[6] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.
[7] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *WWW*. International World Wide Web Conferences Steering Committee, 278–288.
[8] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised Domain Adaptation by Backpropagation. In *ICML*. 1180–1189.
[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*. 2672–2680.
[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*. 1026–1034.
[11] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *ICLR*.
[12] Di Hu, Chengze Wang, Feiping Nie, and Xuelong Li. 2019. Dense Multimodal Fusion for Hierarchically Joint Representation. In *ICASSP*. IEEE, 3941–3945.
[13] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.
[14] Haoran Li, Junnan Zhu, Tianshang Liu, Jiajun Zhang, Chengqing Zong, et al. 2018. Multi-modal sentence summarization with modality attention and image filtering. In *IJCAI*.
[15] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *KDD*. ACM, 1754–1763.
[16] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. 43–52.
[17] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: a view from the trenches. In *SIGKDD*. 1222–1230.
[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.
[19] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal deep learning. In *ICML*. 689–696.
[20] Yabo Ni, Dan Ou, Shichen Liu, Xiang Li, Wenwu Ou, Anxiang Zeng, and Luo Si. 2018. Perceive Your Users in Depth: Learning Universal User Representations from Multiple E-commerce Tasks. In *SIGKDD*. 596–605.
[21] Kyo-Joong Oh, Won-Jo Lee, Chae-Gyun Lim, and Ho-Jin Choi. 2014. Personalized news recommendation using classified keywords to capture user preference. In *ICACT*. 1283–1287.
[22] Wanli Ouyang, Xiao Chu, and Xiaogang Wang. 2014. Multi-source deep learning for human pose estimation. In *CVPR*. 2329–2336.
[23] Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. 2018. Multi-adversarial domain adaptation. In *AAAI*.
[24] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*. 1532–1543.
[25] Steffen Rendle. 2010. Factorization machines. In *ICDM*. IEEE, 995–1000.
[26] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. 285–295.
[27] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. In *ICLR*.
[28] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Recsys*. 17–22.
[29] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *NIPS*. 2643–2651.
[30] Bokun Wang, Yang Yang, Xing Xu, Alan Hanjalic, and Heng Tao Shen. 2017. Adversarial cross-modal retrieval. In *ACMMM*. 154–162.
[31] Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. 2019. Neural News Recommendation with Attentive Multi-View Learning. In *IJCAI*.
[32] Sai Wu, Weichao Ren, Chengchao Yu, Gang Chen, Dongxiang Zhang, and Jingbo Zhu. 2016. Personal recommendation using deep recurrent neural networks in NetEase. In *ICDE*. 1218–1229.
[33] Zuxuan Wu, Yu-Gang Jiang, Jun Wang, Jian Pu, and Xiangyang Xue. 2014. Exploring inter-feature and inter-class relationships with deep neural networks for video classification. In *ACMMM*. 167–176.
[34] Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. 2018. Learning semantic representations for unsupervised domain adaptation. In *ICML*. 5419–5428.
[35] Ling Yan, Wu-jun Li, Gui-Rong Xue, and Dingyi Han. 2014. Coupled group lasso for web-scale ctr prediction in display advertising. In *ICML*. 802–810.
[36] Chaohui Yu, Jindong Wang, Yiqiang Chen, and Meiyu Huang. 2019. Transfer Learning with Dynamic Adversarial Adaptation Network. In *ICDM*.
[37] Luo Yuanfei, Wang Mengshuo, Zhou Hao, Yao Quanming, Tu WeiWei, Chen Yuqiang, Yang Qiang, and Dai Wenyuan. 2019. AutoCross: Automatic Feature Crossing for Tabular Data in Real-World Applications. In *KDD*.
[38] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *AAAI*, Vol. 33. 5941–5948.
[39] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *SIGKDD*. 1059–1068.

Large-scale Causal Approaches to Debiasing Post-click Conversion Rate Estimation with Multi-task Learning

Wenhao Zhang*†
University of California, Los Angeles
wenhaoz@ucla.edu

Wentian Bao*
Alibaba Group
wentian.bwt@alibaba-inc.com

Xiao-Yang Liu
Columbia University
xl2427@columbia.edu

Keping Yang
Alibaba Group
shaoyao@alibaba-inc.com

Quan Lin
Alibaba Group
tieyi.lq@alibaba-inc.com

Hong Wen
Alibaba Group
qinggan.wh@alibaba-inc.com

Ramin Ramezani
University of California, Los Angeles
raminr@ucla.edu

ABSTRACT

Post-click conversion rate (CVR) estimation is a critical task in e-commerce recommender systems. This task is deemed quite challenging under industrial setting with two major issues: 1) selection bias caused by user self-selection, and 2) data sparsity due to the rare click events. A successful conversion typically has the following sequential events: "exposure → click → conversion". Conventional CVR estimators are trained in the click space, but inference is done in the entire exposure space. They fail to account for the causes of the missing data and treat them as missing at random. Hence, their estimations are highly likely to deviate from the real values by large. In addition, the data sparsity issue can also handicap many industrial CVR estimators which usually have large parameter spaces.

In this paper, we propose two principled, efficient and highly effective CVR estimators for industrial CVR estimation, namely, **Multi-IPW** and **Multi-DR**. The proposed models approach the CVR estimation from a causal perspective and account for the causes of missing not at random. In addition, our methods are based on the multi-task learning framework and mitigate the data sparsity issue. Extensive experiments on industrial-level datasets show that our methods outperform the state-of-the-art CVR models.

CCS CONCEPTS

• **Information systems** **Recommender systems**; **Information retrieval diversity**; **Personalization**.

KEYWORDS

conversion rate estimation; causal inference; selection bias; multi-task learning; recommender systems

*Both authors contributed equally. Authorship order determined by coin flip.
†Work done while interning at Alibaba Group.

This paper is published under the Creative Commons Attribution 4.0 International (CC-BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.
WWW '20, April 20–24, 2020, Taipei, Taiwan
© 2020 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC-BY 4.0 License.
ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380037

ACM Reference Format:
Wenhao Zhang, Wentian Bao, Xiao-Yang Liu, Keping Yang, Quan Lin, Hong Wen, and Ramin Ramezani. 2020. Large-scale Causal Approaches to Debiasing Post-click Conversion Rate Estimation with Multi-task Learning. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3366423.3380037

1 INTRODUCTION

Selection bias is a widely-recognized issue in recommender systems [4, 17, 21]. For example, music stream services usually suggest genres that have positive user feedbacks (e.g., favorite, share, and buy, etc.), and selectively ignore the ones that are rarely exposed to users [23]. In this paper, we study the selection bias that exists in the post-click conversion rate (CVR) estimation.

Problem Statement. Post-click conversion rate (CVR) estimation is a critical task in e-commerce recommender systems [30, 33]. A typical e-commerce transaction has the following sequential events: "exposure → click → conversion" [17]. Post-click conversion rate indicates the probability of transitions from click to conversion. Typically, when training CVR models, we only include the items that customers clicked on as we are unaware of the conversion feedback of the items that are not clicked by customers [12]. Bear in mind, not clicking on an item does not necessarily indicate the customer is not interested in purchasing it. Customers may unconsciously skip certain items that might be interesting to them. Fig. 1 reveals that the exposure space $\mathcal{D}$ is a super set of the click space $\mathcal{O}$. Selection bias occurs when conventional CVR models are trained in the click space, and the predictions are made in the entire exposure space (see Fig. 1) [17]. Intuitively, data in the click space is drawn from the entire exposure space and is biased by the user self-selection. Therefore, the data distribution in the click space is systematically different from the one in the exposure space. This inherent discrepancy leads to data that is missing not at random (MNAR), and selection bias in the conventional CVR models [3, 4, 15, 16].

In summary, we identify two practical issues that make CVR estimation quite challenging in industrial-level recommender systems:

- **Selection bias**: The systematic difference between data distributions in training space $\mathcal{O}$ and inference space $\mathcal{D}$ biases

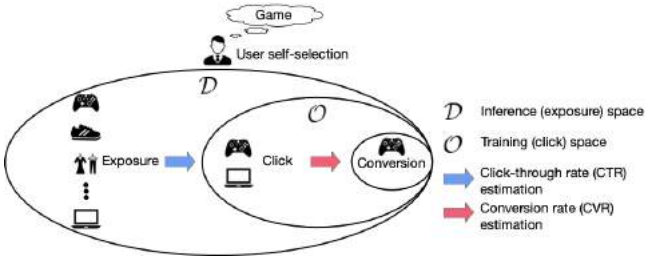


Figure 1: Illustration of the selection bias issue in conventional conversion rate (CVR) estimation. The training space of conventional CVR models is the click space $\mathcal{O}$, whereas the inference space is the entire exposure space $\mathcal{D}$. The discrepancy of data distribution between $\mathcal{O}$ and $\mathcal{D}$ leads to selection bias in conventional CVR models.

conventional CVR models [1, 3, 12, 22]. This bias usually causes the performance degradation.

- **Data sparsity**: This issue occurs since clicks are relatively rare events (we have a CTR of 5.2% in the production dataset and 4% in the public dataset). Conventional CVR models are typically trained only using data in the click space. Therefore, the number of training samples may not be sufficient for the large parameter space. In the experiments, we have 0.6 billion data samples vs. 5.3 billion parameters for production dataset, and 4.3 million samples vs. 2.6 billion parameters for public dataset (see Section 4.1) [14, 26].

To simplify the debiasing task of CVR estimation, we assume the exposure space is the entire item space we are interested in (see Fig. 1) [17]. Such a relaxation is also made based on the postulation that most items are exposed at least once. Table 1 shows that our dataset contains 81.5 million items and 11.5 billion exposures, i.e., each item is exposed, on average, about 150 times.

To address the critical issues of selection bias and data sparsity in the CVR estimation, we take a causal perspective and develop causal methods in a multi-task learning framework. We propose two principled, efficient and highly effective CVR estimators, namely, **Multi**-task **I**nverse **P**ropensity **W**eighting estimator (**Multi-IPW**) and **Multi**-task **D**oubly **R**obust estimator (**Multi-DR**). Our methods are designed for unbiased CVR estimation. They also account for the data sparsity issue.

The main contributions of this paper are summarized as follows:

- To the best of our knowledge, this is the first paper that combines IPW-based and DR-based methods with multi-task learning. From a causal perspective, we aim to tackle the well-recognized issues (i.e., selection bias and data sparsity) in CVR estimation in concert.
- We highlight that the state-of-the-art CVR model, ESMM [17], is biased. Different from existing works, our methods adjust for MNAR data, and deal with the selection bias in a principled way. Meanwhile, we give mathematical proofs that the proposed methods are theoretically unbiased. The empirical study shows our approaches outperform ESMM and several state-of-the-art causal models, and demonstrates the efficiency of our methods in real industrial setting.

2 RELATED WORKS

Ma *et al.* [17] proposed the Entire Space Multi-task Model (ESMM) to remedy selection bias and data sparsity issues in the conversion rate (CVR) estimation. ESMM is trained in the entire exposure space, and it formulates CVR task as two auxiliary tasks, i.e., click-through rate (CTR) and click-through & conversion rate (CTCVR) estimations. However, we argue that ESMM is biased. The details of our argument are presented in Section 3.2.

Causal inference offers a way to adapt for the data generation process when we attempt to restore the information from MNAR data [15]. Schnabel *et al.* [21] proposed an IPW-based estimator for training and evaluating recommender systems from biased data. IPW-based models may still be biased if the propensities are not accurately estimated. Wang *et al.* [27] proposed a doubly robust (DR) joint learning approach for estimating item ratings that are MNAR. Doubly robust estimator combines the IPW-based methods with an imputation model that estimates the prediction error for the missing data. When the propensities are not accurately learned, DR estimator can still enjoy unbiasedness as long as its imputation model is accurate. However, the existing DR-based methods are not devised for CVR estimation, hence fail to account for the severe data sparsity issue that widely exists in the CVR estimation. In addition, such a joint learning approach is not efficient in industrial setting (see Fig. 4).

3 CAUSAL CVR ESTIMATORS WITH MULTI-TASK LEARNING

3.1 Preliminary

Let $\mathcal{U} = (u_1, u_2, ..., u_N)$ be a set of N users and $\mathcal{I} = (i_1, i_2, ..., i_M)$ be a set of M items, $\mathcal{D} = \mathcal{U} \times \mathcal{I}$ be the user-item pairs, $\mathbf{R} \in \mathbb{R}^{N \times M}$ be the true conversion label matrix where each entry $r_{u,i} \in \{0, 1\}$, and $\hat{\mathbf{R}} \in \mathbb{R}^{N \times M}$ be the predicted conversion score matrix where each entry $\hat{r}_{u,i} \in [0, 1]$. Then, the *Prediction inaccuracy* $\mathcal{P}$ over all user-item pairs can be formulated as follows,

$$\mathcal{P} = \mathcal{P}(\mathbf{R}, \hat{\mathbf{R}}) = \frac{1}{|\mathcal{D}|} \sum_{(u,i) \in \mathcal{D}} e(r_{u,i}, \hat{r}_{u,i}), \tag{1}$$

where $e(r_{u,i}, \hat{r}_{u,i}) = -r_{u,i} \log(\hat{r}_{u,i}) - (1 - r_{u,i}) \log(1 - \hat{r}_{u,i})$.

Let $\mathbf{O} \in \{0, 1\}^{\mathcal{U} \times \mathcal{I}}$ be the *indicator matrix* where each entry $o_{u,i}$ is an observation indicator: $o_{u,i} = 1$ if a user u clicks on item i, $o_{u,i} = 0$ otherwise. Since the clicks are subjective to certain unobserved factors (e.g., users latent interests), such user self-selection process generates MNAR data [6, 20]. Naive CVR estimators are trained only in the click space $\mathcal{O} = \{(u, i) | o_{u,i} = 1, (u, i) \in \mathcal{D}\}$. Let $\mathbf{R}^{\text{obs}}$ and $\mathbf{R}^{\text{mis}}$ be the set of conversion labels that are present and absent in $\mathcal{D}$. We evaluate these naive CVR models by averaging the cross-entropy loss over the observed data [21, 27],

$$\begin{aligned} \mathcal{E}^{\text{Naive}} &= \mathcal{E}(\mathbf{R}^{\text{obs}}, \hat{\mathbf{R}}) \\ &= \frac{1}{|\mathcal{O}|} \sum_{(u,i) \in \mathcal{O}} e(r_{u,i}, \hat{r}_{u,i}) \\ &= \frac{1}{|\mathcal{O}|} \sum_{(u,i) \in \mathcal{D}} o_{u,i} e(r_{u,i}, \hat{r}_{u,i}), \end{aligned} \tag{2}$$

where $|\mathcal{O}| = \sum_{(u,i) \in \mathcal{D}} o_{u,i}$.

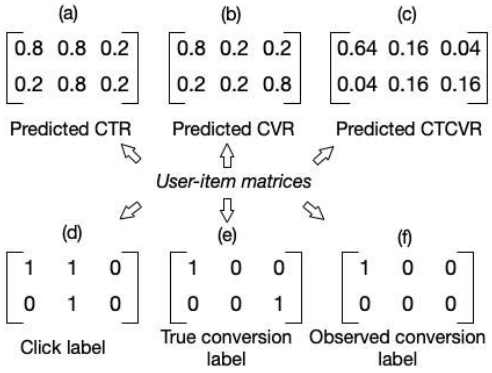


Figure 2: A toy example that demonstrates ESMM is biased.

We say a CVR estimator $\mathcal{M}$ is *unbiased* when the expectation of the estimated prediction inaccuracy over $\mathbf{O}$ equals to the prediction inaccuracy $\mathcal{P}$, i.e., $\text{Bias}^{\mathcal{M}} = |\mathbb{E}_{\mathbf{O}}[\mathcal{E}^{\mathcal{M}}] - \mathcal{P}| = 0$, otherwise it is *biased*. If data is MNAR, $|\mathbb{E}_{\mathbf{O}}[\mathcal{E}_{\mathcal{M}}] - \mathcal{P}| \gg 0$ [18].

3.2 Is ESMM an Unbiased CVR Estimator?

In this section, we demonstrate that ESMM, the state-of-the-art CVR estimator in practice, is essentially biased, though the author claim in the paper that the model eliminates the selection bias [17]. We formulate the estimation bias of ESMM, and prove it is not theoretically unbiased by giving a counter example.

Let $e^{\text{CTR}}_{u,i}, e^{\text{CVR}}_{u,i}, e^{\text{CTCVR}}_{u,i}, (u,i) \in \mathcal{D}$, be the cross-entropy losses of CTR, CVR, and CTCVR tasks. Then we have,

$$\begin{aligned}
&\text{Bias}^{\text{ESMM}} = |\mathbb{E}_{\mathbf{O}}[\mathcal{E}^{\text{ESMM}}] - \mathcal{P}| \\
&= \left| \mathbb{E}_{\mathbf{O}} \left[\frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} (e^{\text{CTR}}_{u,i} + e^{\text{CTCVR}}_{u,i}) \right] - \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} e^{\text{CVR}}_{u,i} \right| \\
&= \left| \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} (e^{\text{CTR}}_{u,i} + e^{\text{CTCVR}}_{u,i}) - \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} e^{\text{CVR}}_{u,i} \right| \\
&= \frac{1}{|\mathcal{D}|} \left| \sum_{(u,i)\in\mathcal{D}} (e^{\text{CTR}}_{u,i} + e^{\text{CTCVR}}_{u,i} - e^{\text{CVR}}_{u,i}) \right|.
\end{aligned} \tag{3}$$

We can easily verify that $\text{Bias}^{\text{ESMM}} > 0$ using the counter example in Fig. 2. Note that to be theoretically unbiased, ESMM should satisfy $|\mathbb{E}_{\mathcal{D}}[\mathcal{E}^{\text{ESMM}}] - \mathcal{P}| = 0, \forall \mathcal{D}$. Therefore, we conclude that ESMM cannot ensure unbiased CVR estimation.

3.3 Multi-task Learning Module

To address the data sparsity issue, we adopt the philosophy of multi-task learning and introduce an auxiliary CTR task [25]. The multi-task learning module exploits the typical sequential events in e-commerce recommender system, i.e., "exposure → click → conversion", and chains the main CVR task with the auxiliary CTR task. The amount of training data in CTR task is generally larger than that in CVR task by $1 \sim 2$ order of magnitudes (see Table 1), thus CTR task trains the large volume of model parameters more sufficiently. Besides, the feature representation learned in the CTR task is shared with the CVR task, which makes the CVR model benefit from the extra information via parameter sharing. Hence, the data sparsity issue is remedied [10, 17, 19].

Meanwhile, multi-task learning is also perceived as being cost-effective in training phase [17]. Specifically, multi-task learning co-trains multiple tasks simultaneously as if they were one task. This mechanism can potentially reduce storage space for saving duplicate copies of embedding matrix. In addition, the parallel training mechanism generally reduces the training time by large. The Multi-IPW and Multi-DR models inherit aforementioned merits by incorporating a multi-task learning module.

3.4 Multi-task Inverse Propensity Weighting CVR Estimator

Let the marginal probability $P(o_{u,i} = 1)$ denote the propensity score, $p_{u,i}$, of observing an entry in $\mathbf{R}$. In practice, the real $p_{u,i}$ can not be obtained directly. Instead, we estimate the real propensity with $\hat{p}_{u,i}$. The IPW-based estimator uses $\hat{p}_{u,i}$ to inversely weight prediction loss [11, 13, 15, 21],

$$\mathcal{E}^{\text{IPW}} = \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} \frac{o_{u,i} e(r_{u,i}, \hat{r}_{u,i})}{\hat{p}_{u,i}}. \tag{4}$$

Typically, $\hat{p}_{u,i}$ is learned via an independent logistic regression model [2]. In Fig. 3, we propose the Multi-IPW model which leverages the multi-task learning framework to simultaneously learn the propensity score (i.e., CTR in Multi-IPW) with CVR. Hence, the loss function of Multi-IPW estimator can be written as follows,

$$\begin{aligned}
&\mathcal{E}^{\text{Multi-IPW}}(\mathcal{X}_O; \theta_{\text{CTR}}, \theta_{\text{CVR}}, \Phi) \\
&= \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} \frac{o_{u,i} e(r_{u,i}, \hat{r}_{u,i}(\vec{x}_{u,i}; \theta_{\text{CVR}}, \Phi))}{\hat{p}_{u,i}(\vec{x}_{u,i}; \theta_{\text{CTR}}, \Phi)},
\end{aligned} \tag{5}$$

where Φ represents the shared embedding parameters. θ_{CVR} and θ_{CTR} are neural network parameters of CVR task and CTR task, respectively. $e(r_{u,i}, \hat{r}_{u,i})$, parameterized by θ_{CVR} and Φ, is the cross entropy loss of true CVR label $r_{u,i}$ and predicted CVR score $\hat{r}_{u,i}$. We use the predicted CTR score $\hat{p}_{u,i}$, parameterized by θ_{CTR} and Φ, as propensities. $\mathcal{D}$ denotes all the data in the exposure space. $\mathcal{X}_O$ is the input feature vectors in O.

we formally derive the bias of Multi-IPW and prove it is unbiased given the propensities are accurately estimated.

Theorem 3.1. *Given the true propensities* $\mathbf{P}$ *and the true conversion label matrix* $\mathbf{R}$*, the Multi-IPW CVR estimator gives unbiased CVR prediction when estimated propensity scores are accurate* $\hat{p}_{u,i} = p_{u,i}$*,*

$$|\mathbb{E}_{\mathbf{O}}[\mathcal{E}^{\textit{Multi-IPW}}] - \mathcal{P}| = 0. \tag{6}$$

Proof.

$$\begin{aligned}
&|\mathbb{E}_{\mathbf{O}}[\mathcal{E}^{\text{Multi-IPW}}] - \mathcal{P}| \\
&= \left| \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} \mathbb{E}_{\mathbf{O}} \left[\frac{o_{u,i} e(r_{u,i}, \hat{r}_{u,i})}{\hat{p}_{u,i}} \right] - \mathcal{P} \right| \\
&= \left| \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} \frac{p_{u,i} e(r_{u,i}, \hat{r}_{u,i})}{\hat{p}_{u,i}} - \mathcal{P} \right| \\
&= \left| \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} e(r_{u,i}, \hat{r}_{u,i}) - \mathcal{P} \right| = 0.
\end{aligned} \tag{7}$$

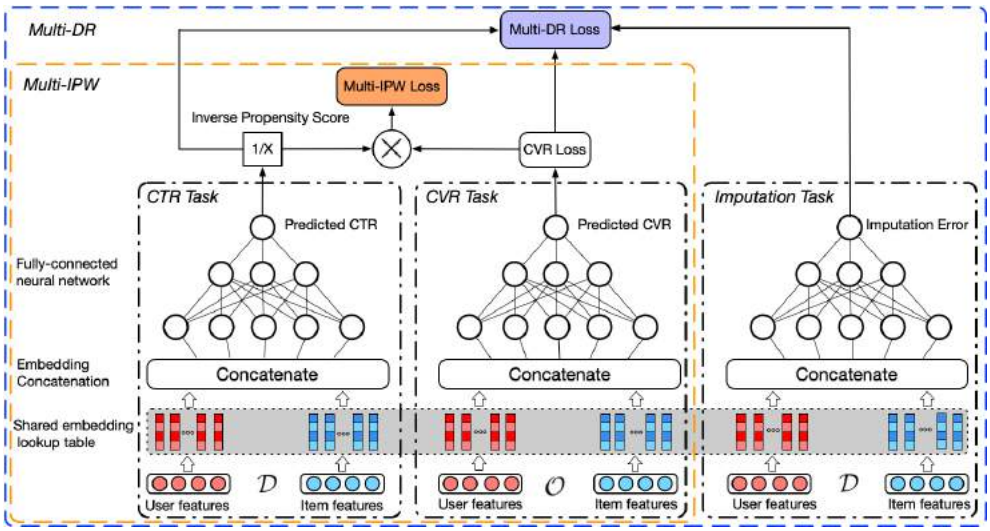


Figure 3: Multi-Inverse Propensity Weighting estimator and Multi-Doubly Robust estimator. The Multi-DR estimator augments Multi-IPW with an imputation model. We use predicted CTR as propensity scores in the Multi-IPW estimator. In the multi-task learning module, the CTR task, CVR task, and Imputation task are chained together via parameter sharing.

□

Multi-IPW estimator inherits the merits of multi-task learning: 1) better CVR prediction due to parameter sharing, and 2) reduced training time and parameter storage. These are clear advantages over conventional IPW-based estimators.

3.5 Multi-task Doubly Robust CVR Estimator

The IPW-based models are unbiased contingent on accurately estimated propensities (i.e., $\hat{p}_{u,i} = p_{u,i}$). In practice, this condition is too restricted. To address this issue, doubly robust estimator is introduced by previous works [5, 7, 24, 27].

Wang *et al.* [27] proposed a joint learning approach for training a doubly robust estimator, and introduced two models: 1) a prediction model $\hat{r}_{u,i} = f_\theta(\vec{x}_{u,i})$, and 2) an imputation model $\hat{e}_{u,i} = g_\phi(\vec{x}_{u,i})$. The prediction model, parameterized by θ, aims to predict the ratings, and its performance is evaluated by $e_{u,i} = e(r_{u,i}, \hat{r}_{u,i}), (u,i) \in \mathcal{D}$. The imputation model, parameterized by ϕ, aims to estimate the prediction error $e_{u,i}$ with $\hat{e}_{u,i}$. Its performance is assessed by $\delta_{u,i} = e_{u,i} - \hat{e}_{u,i}$. The feature vector $\vec{x}_{u,i}$ encodes all the information about the user u and the item i, $(u,i) \in \mathcal{D}$. Then, we can formulate the loss of doubly robust estimator as,

$$\mathcal{E}^{\text{DR}} = \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} \left(\hat{e}_{u,i} + \frac{o_{u,i}\delta_{u,i}}{\hat{p}_{u,i}} \right), \tag{8}$$

Similarly, we propose the Multi-DR estimator which augments Multi-IPW estimator by including an imputation model estimating the prediction error $e_{u,i}$. Multi-DR optimizes the following loss,

$$\begin{aligned}&\mathcal{E}^{\text{Multi-DR}}(\mathcal{X}; \theta_{\text{CTR}}, \theta_{\text{CVR}}, \theta_{\text{Imp}}, \Phi) \\ &= \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} \left(\hat{e}_{u,i}(\vec{x}_{u,i}; \theta_{\text{Imp}}, \Phi) + \frac{o_{u,i}\delta_{u,i}(\vec{x}_{u,i}; \theta_{\text{CVR}}, \theta_{\text{Imp}}, \Phi)}{\hat{p}_{u,i}(\vec{x}_{u,i}; \theta_{\text{CTR}}, \Phi)} \right),\end{aligned} \tag{9}$$

where Φ represents the shared embedding parameters among CTR task, CVR task, and imputation task. θ_{CTR}, θ_{CVR}, θ_{Imp} are neural network parameters of CTR task, CVR task, imputation task, respectively. $\hat{p}_{u,i}$ is the propensity (i.e., predicted CTR score) given by CTR task. Estimated prediction error $\hat{e}$, parameterized by θ_{Imp} and Φ, is given by imputation task. $\delta_{u,i} = e_{u,i} - \hat{e}_{u,i}$ is the error deviation.

We can formally derive the bias of Multi-DR and prove it is unbiased if either true propensity scores or true prediction errors are accurately estimated (i.e., $\Delta_{u,i} = 0$ or $\delta_{u,i} = 0$).

Theorem 3.2. *Given the true propensities* $\mathbf{P}$ *and the true conversion label matrix* $\mathbf{R}$*, the Multi-DR CVR estimator gives unbiased CVR prediction when either estimated propensity scores are accurate* $\Delta_{u,i} = \frac{p_{u,i} - \hat{p}_{u,i}}{\hat{p}_{u,i}} = 0$ *or the estimated prediction errors are accurate* $\delta_{u,i} = e_{u,i} - \hat{e}_{u,i} = 0$*,*

$$|\mathbb{E}_{\mathbf{O}}[\mathcal{E}^{\textit{Multi-DR}}] - \mathcal{P}| = 0. \tag{10}$$

Proof.

$$\begin{aligned}&|\mathbb{E}_{\mathbf{O}}[\mathcal{E}^{\text{Multi-DR}}] - \mathcal{P}| \\ &= \frac{1}{|\mathcal{D}|} \left| \sum_{(u,i)\in\mathcal{D}} \left(\hat{e}_{u,i} + \mathbb{E}_{\mathbf{O}} \left[\frac{o_{u,i}\delta_{u,i}}{\hat{p}_{u,i}} \right] \right) - \mathcal{P} \right| \\ &= \frac{1}{|\mathcal{D}|} \left| \sum_{(u,i)\in\mathcal{D}} \left(\hat{e}_{u,i} + \frac{p_{u,i}\delta_{u,i}}{\hat{p}_{u,i}} \right) - \mathcal{P} \right| \\ &= \frac{1}{|\mathcal{D}|} \left| \sum_{(u,i)\in\mathcal{D}} \frac{(p_{u,i} - \hat{p}_{u,i})\delta_{u,i}}{\hat{p}_{u,i}} \right| \\ &= \frac{1}{|\mathcal{D}|} \left| \sum_{(u,i)\in\mathcal{D}} \Delta_{u,i}\delta_{u,i} \right| = 0.\end{aligned} \tag{11}$$

□

Table 1: Statistics of experimental datasets.

Dataset	# Exposure	# Click	# Conversion	# User	# Item
Ali-CCP	84M	3.4M	18k	0.4M	4.3M
Set A	1.1B	54.5M	0.6M	-	22.5M
Set B	2.7B	0.2B	1.9M	-	39.1M
Set C	6.0B	0.4B	4.3M	-	62.6M
Set D	11.5B	0.6B	8.3M	-	81.5M

4 EXPERIMENTATION

In this section, we evaluate the performance of the proposed models with a public dataset and a large-scale production dataset collected from Mobile Taobao, the leading e-commerce platform in China. The experiments are intended to answer the following questions:

- **Q1**: Do our proposed approaches outperform the state-of-the-art CVR estimation methods?
- **Q2**: Are our proposed models more efficient in industrial setting than other baseline models?

4.1 Datasets

Ali-CCP[1] *[17].* Alibaba Click and Conversion Prediction (Ali-CCP) dataset is collected from real-world traffic logs of the recommender systems in the Taobao platform. See the statistics in Table 1.

Production sets. This industrial production dataset is collected from the Mobile Taobao e-commerce platform. It contains 3-week transactional data (see Table 1). Our production dataset includes 109 features, which are primarily categorized into: 1) user features, 2) item features, and 3) combination features. We further divide this dataset into 4 subsets: Set A, Set B, Set C, and Set D, which contain the first two days (5%), the first five days (20%), the first twelve days (50%), and the 3-week of data (100%), respectively. We use the data of the last day in each set as testing set and the remaining data as training set.

4.2 Baseline Models

We compare the proposed models with the following baselines.

4.2.1 Non-causal estimators.

- **Base** is a naive post-click CVR model, which is a multi-layer perceptron (see the CVR task in Fig. 3).
- **Oversampling** [28, 31] Duplicates the minority data samples (conversion=1) in training set with an oversampling rate k=5.
- **ESMM** [17] learns CVR using a CTR task and a CTCVR task.
- **Naive Imputation** uses unclicked data as negative samples.

4.2.2 Causal estimators.

- **Naive IPW** [21] is the naive IPW estimator (see Section 3.4).
- **Joint Learning DR** [27] is tailored for CVR estimation.
- **Heuristic DR** is designed as a baseline for Multi-DR. It assumes that the unclicked items are negative samples with probability $1-\eta$, where η is smoothing rate and it denotes the probability of having a positive label. In the experiments, we explore η in $\{0.0005, 0.001, 0.002, 0.005, 0.01\}$.

[1] https://tianchi.aliyun.com/dataset/dataDetail?dataId=408

4.3 Metrics

In CVR prediction task, ROC AUC is a widely used metric [8]. One interpretation of AUC in the context of ranking system is that it denotes the probability of ranking a random positive sample higher than a negative sample. Meanwhile, we also adopt Group AUC (GAUC) [32]. GAUC extends AUC by calculating the weighted average of AUC grouped by page views or users,

$$\text{GAUC} = \frac{\sum_{i \in U} w_i \times \text{AUC}_i}{\sum_{i \in U} w_i}, \tag{12}$$

where w_i is exposures. GAUC is commonly recognized as a more indicative metric in real practice [32]. In the public dataset, models are only assessed with AUC as the dataset is missing the information for computing GAUC.

4.4 Unbiased Evaluation

In this work, we use CTCVR-AUC/GAUC to evaluate the unbiasedness of CVR estimators [17]. We need to point out that testing with an unbiased dataset or randomization is generally a golden standard for unbiasedness assessment [9, 29]. However, the unbiased training/testing dataset for CVR estimation is rather unobtainable in real practice.The real-world system can randomly expose items to users and generate unbiased evaluation sets for CTR estimation. But they cannot force users to randomly click on items to generate unbiased data for CVR estimations. This limitation may be further investigated in the future work.

4.5 Experiments Setup

Ali-CCP experiment. The experiment setup on Ali-CCP follows the prior work [17].

Production set experiment. In the production set experiment, we vary the dimensions of feature embedding vectors according to each feature's real size in order to minimize the memory usage. In order to have a fair comparison study, all the models in this experiment share $|batch| = 10000$, MLP architecture 1024×512×256×128×32×2, adam optimizer with learning rate $lr = 0.0005$. We also added $l2$ regularization to imputation model in Multi-DR, and the coefficient is $v = 0.0001$.

5 RESULTS AND DISCUSSION

5.1 Model Assessments (Q1)

In this section, we report the experiment results in Table 2, 3. Multi-IPW and Multi-DR are clear winners over other baselines across all experiments. Meanwhile, we have the following observations:

- In production dataset, Multi-IPW and Multi-DR consistently outperform Joint Learning DR [27]. We reason that the performance improvement benefits from multi-task learning module, which remedies the data sparsity issue.
- We notice that Multi-DR mostly has better performance than Multi-IPW. Recall that Multi-DR augments Multi-IPW by introducing an imputation model. Provided that $0 \le \hat{e} \le 2e$, the tail bound of Multi-DR is proven to be lower than that of Multi-IPW for any learned propensity score $\hat{p}_{u,i}, (u,i) \in O$ [27]. Therefore, Multi-DR is expected to perform better than Multi-IPW when the imputation model is well-trained.

Table 2: Results of comparison study on Production datasets. The best scores are bold-faced in each column.

Model	Set A (1.1B)			Set B (2.7B)			Set C (6.0B)			Set D (11.5B)		
	CVR AUC	CTCVR AUC	CTCVR GAUC	CTR AUC	CTCVR AUC	CTCVR GAUC	CTR AUC	CTCVR AUC	CTCVR GAUC	CTR AUC	CTCVR AUC	CTCVR GAUC
Base	78.24	73.12	59.69	78.67	73.86	60.16	79.62	74.70	60.58	81.66	76.28	61.27
Oversampling [28]	78.63	73.53	60.17	78.72	74.09	60.28	79.69	74.82	60.59	81.77	76.30	61.30
ESMM [17]	79.29	73.86	60.53	79.74	74.33	60.90	80.11	74.97	61.13	82.17	76.55	61.76
Naive Imputation	78.12	73.21	60.14	78.44	73.50	60.39	79.32	73.81	60.56	81.56	76.39	61.39
Naive IPW [21]	79.23	73.82	60.51	79.73	74.34	60.95	80.14	74.92	61.09	82.13	76.45	61.77
Heuristic DR	78.45	73.45	60.01	78.84	73.99	60.30	79.52	74.18	60.65	81.74	76.40	61.35
Joint Learning [27]	79.09	73.67	60.43	79.53	74.51	60.83	80.01	74.90	60.97	82.09	76.61	61.67
Multi-IPW	79.51	73.99	60.70	**79.85**	74.81	**61.09**	80.21	75.01	61.25	82.57	76.89	61.98
Multi-DR	**79.72**	**74.45**	**60.90**	79.80	**74.91**	60.99	**80.50**	**75.39**	**61.52**	**82.72**	**77.23**	**62.28**

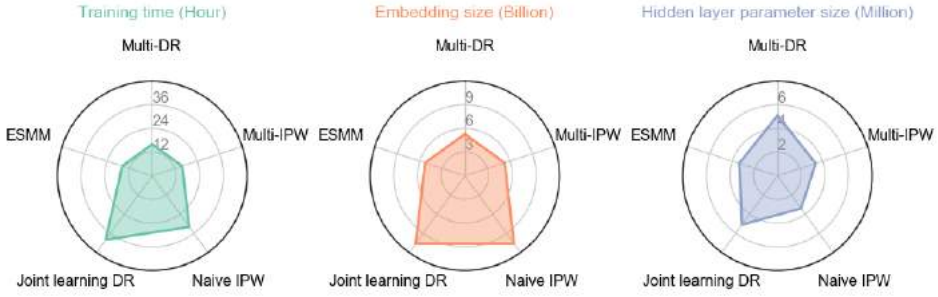


Figure 4: Computational cost comparison radar diagrams. The left subplot reveals the hours needed to complete one epoch of training. The middle subplot shows the size of embedding parameters. The right subplot shows the size of hidden layer parameters.

Table 3: Results of comparison study on Ali-CCP dataset. Experiments are repeated 10 times and mean ± 1 std of AUC scores are reported. The best scores are bold-faced.

Model	CVR AUC	CTCVR AUC
Base	66.00 ± 0.37	62.07 ± 0.45
Oversampling [28]	67.18 ± 0.32	63.05 ± 0.48
ESMM-NS [17]	68.25 ± 0.44	64.44 ± 0.62
ESMM [17]	68.56 ± 0.37	65.32 ± 0.49
Multi-IPW	69.21 ± 0.42	65.30 ± 0.50
Multi-DR	**69.29 ± 0.31**	**65.43 ± 0.34**

- We observe that Multi-IPW/Naive IPW estimator are superior to Base in all experiments. Compared with the Base, both IPW-based models introduce estimated propensities to correct the selection bias. Recall that Theorem3.1. ensures the CVR estimators are unbiased if the propensities are accurately estimated. While in practice the estimated propensities may deviate from the real values, this control experiment attests to "enough accuracy" of the propensity model.

5.2 Computational Efficiency (Q2)

In this section, we study the computational efficiency of the proposed models against the baselines under industrial setting. We summarize the records of training time and parameter space size of each model in Fig. 4, and the cluster configuration in Table 4 [2] .

We observe that Multi-IPW and Multi-DR require less or equivalent training time compared with other baselines. Recall that multi-task learning method co-trains multiple tasks simultaneously as if they were one task. We can expect the training time being greatly shortened. Meanwhile, our methods are also economical in memory usage due to parameter sharing in the multi-task learning module.

[2]GPU specs: Tesla P100-PCIE-16G

Table 4: Distributed cluster configuration.

Cluster configuration	Parameter Server	Worker
# Instances	4	100
# CPU	28 cores	440 cores
# GPU	-	25 cards
Memory (GB)	40	1000

6 CONCLUSION

In this paper, we proposed Multi-IPW and Multi-DR CVR estimators for industrial recommender system. Both CVR estimators aim to counter the inherent issues in practice: 1) selection bias, and 2) data sparsity. Extensive experiments with billions of data samples demonstrate that our methods outperform the state-of-the-art CVR predictive models, and handle CVR estimation task in a principled, highly effective and efficient way. Although our methods are devised for CVR estimation, the idea can be generalized to debiasing CTR estimation by exploiting the sequential pattern "item pool → exposure → click".

REFERENCES

[1] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. 2018. Unbiased learning to rank with unbiased propensity estimation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 385–394.

[2] Peter C Austin. 2011. An introduction to propensity score methods for reducing the effects of confounding in observational studies. *Multivariate Behavioral Research* 46, 3 (2011), 399–424.

[3] Elias Bareinboim, Jin Tian, and Judea Pearl. 2014. Recovering from selection bias in causal and statistical inference. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.
[4] Arnaud De Myttenaere, Bénédicte Le Grand, Boris Golden, and Fabrice Rossi. 2014. Reducing offline evaluation bias in recommendation systems. In *23rd annual Belgian-Dutch Conference on Machine Learning (Benelearn 2014)*. Bruxelles, Belgium, 55–62. https://hal.archives-ouvertes.fr/hal-01017734
[5] Miroslav Dudík, John Langford, and Lihong Li. 2011. Doubly robust policy evaluation and learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML'11)*. Omnipress, Madison, WI, USA, 1097–1104.
[6] Craig K Enders. 2010. *Applied missing data analysis*. Guilford press.
[7] Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. 2018. More robust doubly robust off-policy evaluation. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Jennifer Dy and Andreas Krause (Eds.), Vol. 80. PMLR, Stockholmsmässan, Stockholm Sweden, 1447–1456. http://proceedings.mlr.press/v80/farajtabar18a.html
[8] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27, 8 (2006), 861–874.
[9] Ronald A Fisher. 1956. *Statistical methods and scientific inference*. Hafner Publishing Co.
[10] Guy Hadash, Oren Sar Shalom, and Rita Osadchy. 2018. Rank and rate: multi-task learning for recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 451–454.
[11] Keisuke Hirano, Guido W Imbens, and Geert Ridder. 2003. Efficient estimation of average treatment effects using the estimated propensity score. *Econometrica* 71, 4 (2003), 1161–1189.
[12] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex J Smola. 2007. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems*. 601–608.
[13] Guido W Imbens and Donald B Rubin. 2015. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press.
[14] Kuang-chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. 2012. Estimating conversion rate in display advertising from past erformance data. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 768–776.
[15] RJA Little and DB Rubin. 2002. Statistical analysis with missing data. Wiley. *New York* (2002).
[16] Roderick JA Little and Donald B Rubin. 2019. *Statistical analysis with missing data*. Vol. 793. John Wiley & Sons.
[17] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 1137–1140. https://doi.org/10.1145/3209978.3210104
[18] Karthika Mohan, Judea Pearl, and Jin Tian. 2013. Graphical models for inference with missing data. In *Advances in Neural Information Processing Systems*. 1277–1285.
[19] Yabo Ni, Dan Ou, Shichen Liu, Xiang Li, Wenwu Ou, Anxiang Zeng, and Luo Si. 2018. Perceive your users in depth: Learning universal user representations from multiple e-commerce tasks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 596–605.
[20] Donald B Rubin. 1976. Inference and missing data. *Biometrika* 63, 3 (1976), 581–592.
[21] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments: Debiasing learning and evaluation. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16)*. JMLR.org, 1670–1679.
[22] Harald Steck. 2010. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 713–722.
[23] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*. 2643–2651.
[24] Karel Vermeulen and Stijn Vansteelandt. 2015. Bias-reduced doubly robust estimation. *J. Amer. Statist. Assoc.* 110, 511 (2015), 1024–1036.
[25] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-task feature learning for knowledge graph enhanced recommendation. In *The World Wide Web Conference*. ACM, 2000–2010.
[26] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 839–848.
[27] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. 2019. Doubly robust joint learning for recommendation on data missing not at random. *Proceedings of Machine Learning Research* (2019).
[28] Gary M Weiss. 2004. Mining with rarity: a unifying framework. *ACM Sigkdd Explorations Newsletter* 6, 1 (2004), 7–19.
[29] Anne Whitehead and John Whitehead. 1991. A general parametric approach to the meta-analysis of randomized clinical trials. *Statistics in Medicine* 10, 11 (1991), 1665–1677.
[30] Yuriko Yamaguchi, Mimpei Morishita, Youichi Inagaki, Reyn Nakamoto, Jianwei Zhang, Junichi Aoi, and Shinsuke Nakajima. 2016. Web advertising recommender system based on estimating users' latent interests. In *Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services*. 42–49.
[31] Wenhao Zhang, Ramin Ramezani, and Arash Naeim. 2019. WOTBoost: Weighted oversampling technique in boosting for imbalanced learning. *arXiv e-prints*, Article arXiv:1910.07892 (Oct 2019), arXiv:1910.07892 pages. arXiv:cs.LG/1910.07892
[32] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM.
[33] Han Zhu, Junqi Jin, Chang Tan, Fei Pan, Yifan Zeng, Han Li, and Kun Gai. 2017. Optimized cost per click in taobao display advertising. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2191–2200.

出品单位

阿里巴巴淘系技术部
PMO & 技术战略发展

Alibaba Group 阿里巴巴集团 淘系技术部 TAO TECHNOLOGY

2021 01 05