阿里云出品

视觉计算开发者系列手册

图像识别-目标检测入门必读指南

第一期

阿里云 开发者社区

扫码加入阿里云
视觉计算交流群

扫码关注阿里云人工智
能社区，了解前沿动态

前言

以下技术手册内容摘自《深度学习与图像识别：原理与实践》，目标检测章节，如果读者需要系统的学习图像识别，可阅读《深度学习与图像识别：原理与实践》全书。

目标检测

本文首先会介绍目标检测的概念，然后介绍一种简化了的目标检测问题——定位 + 分类以及它存在的问题，最后由浅入深逐步进入到目标检测常用的模型及方法，如 Faster R-CNN、SSD 等。这个过程中会涉及很多细节的概念和知识点。本文针对 SSD 算法给出了 PyTorch 版的代码，大家也会在这里看到一些细节知识点的实现方法。

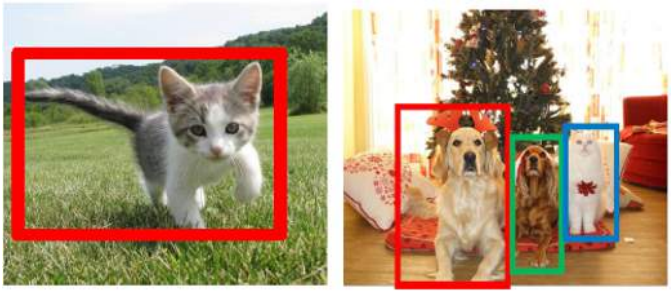
学习时，建议先根据顺序把所有检测方法浏览一遍，有一个初步的了解；然后再深入研究 SSD 部分的代码，理解其中涉及的很多细节将有助于大家深入理解一些问题。学习完 SSD 之后，再看一遍 Faster R-CNN 等方法，把第一遍阅读时缺失的细节补充上去，这样能帮助大家循序渐进地深入了解目标检测算法。当然，擅长实战的读者可以直接跳到 SSD 部分看代码，实战之后再补充理论也不迟。

开始本文内容之前，我们先明确两个定义：

（1）定位 + 分类：对于仅有一个目标的图片，检测出该目标所处的位置以及该目标的类别，如图（a，c）所示。

（2）目标检测：对于有多个目标的图片，检测出所有目标所处的位置及其类别，如图（b，d）所示。我们先从相对简单的定位 + 分类开始，而后重点介绍目标检测。学完本章之后你会发现很多图像问题就都可以解了。

（a）原图　　（b）原图



（c）定位 + 分类　　（d）目标检测

检测问题定义 [9]

目录

第一章　定位 + 分类

该问题是分类到目标检测的一个过渡问题，从单纯地图片分类到分类后给出目标所处位置，再到多目标的类别和位置。接下来，我们看一下定位 + 分类问题的解法。

（a）分类问题　　（b）定位问题

图 1-1　分类问题 vs 定位问题 [9]

分类不用多说，上一章我们以分类为例讲了卷积神经网络。在定位问题中，则需要模型返回目标所在的外接矩形框，即目标的（x,y,w,h）四元组。接下来介绍一种比较容易想到的思路，把定位当做回归问题，具体步骤如下：

（1）训练（或下载）一个分类模型，例如 AlexNet、VGGNet 或 ResNet；

（2）在分类网络最后一个卷积层的特征层（feature map）上添加“regression head”，如图 1-2 所示；补充说明：神经网络中不同的“head”通常用来训练不同的目标，每个“head”的损失函数和优化方向不同。如果想让一个网络实现多个功能，通常是在神经网络后面接多个不同功能的“head”。

（3）同时训练“classification head”和“regression head”，为了同时训练分类和定位（定位是回归问题）两个问题，最终损失函数是分类和定位两个“head”产生损失的加权和。

（4）在预测时同时使用分类和回归 head 得到分类 + 定位结果。这里强调一下，

分类预测出的结果就是 C 个类别，回归预测的结果可能有两种：一种是类别无关，输出 4 个值；一种是类别相关，输出 4*C 个值，这要看读者想要哪种结果了。

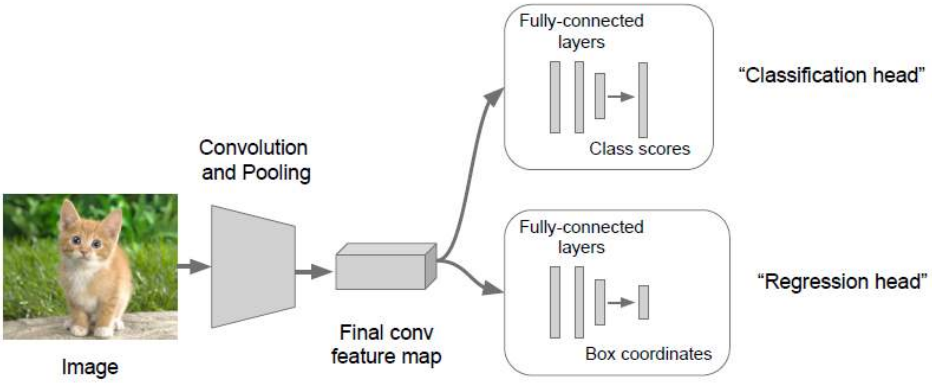


图 1-2　分类 + 定位网络结构设计 [9]

第二章　目标检测

目标检测需要获取图片中所有目标的位置及其类别，对于图 2-1 中的 3 张图而言，当图片中只有一个目标时，“regression head”预测 4 个值，当图片中有 3 个目标时，“regression head”预测 12 个值，那么当图片中有多个目标时，“regression head”要预测多少个值呢?

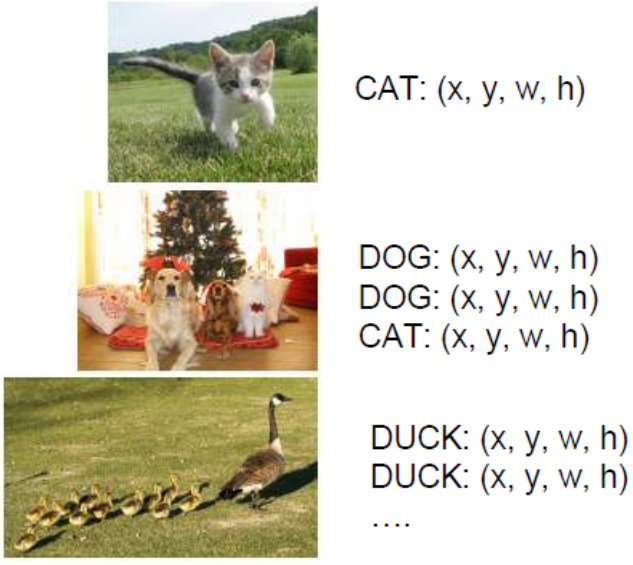


图 2-1　使用定位 + 分类解目标检测存在的问题 [9]

这时根据读者已经学过的一些知识，可能会尝试用滑窗的方法来解决，如图 2-2 所示。但是，这里有一个问题，我们需要设计大量的不同尺度和长宽比的“滑窗”使它们通过 CNN，然而这个计算量是非常巨大的。有没有什么方法能使得我们快速定位到目标的潜在区域，从而减少大量不必要的计算呢?

Dog? NO
Cat? NO
Background? YES

Dog? YES
Cat? NO
Background? NO

Dog? YES
Cat? NO
Background? NO

Dog? NO
Cat? YES
Background? NO

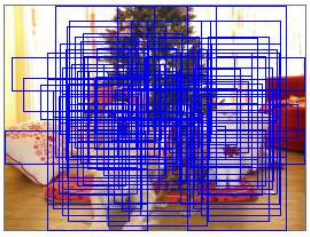


图 2-2　使用滑窗方法做目标检测存在的问题：滑窗的尺寸、大小、位置不同将产生非常大的计算量 [9]

学者们在这个方向做了很多研究，比较有名的是 selective search 方法，具体方法这里不做详细说明，感兴趣的读者可以看关于 selective search 的论文。大家只要知道这是一种从图片中选出潜在物体候选框（Regions of Interest，ROI）的方法即可。有了获取 ROI 的方法，接下来就可以通过分类和合并的方法来获取最终的目标检测结果。基于这个思路有了下面的 R-CNN 方法。

2.1 R-CNN

下面介绍 R-CNN[1] 的训练过程，整体训练流程如图 2-3 所示：

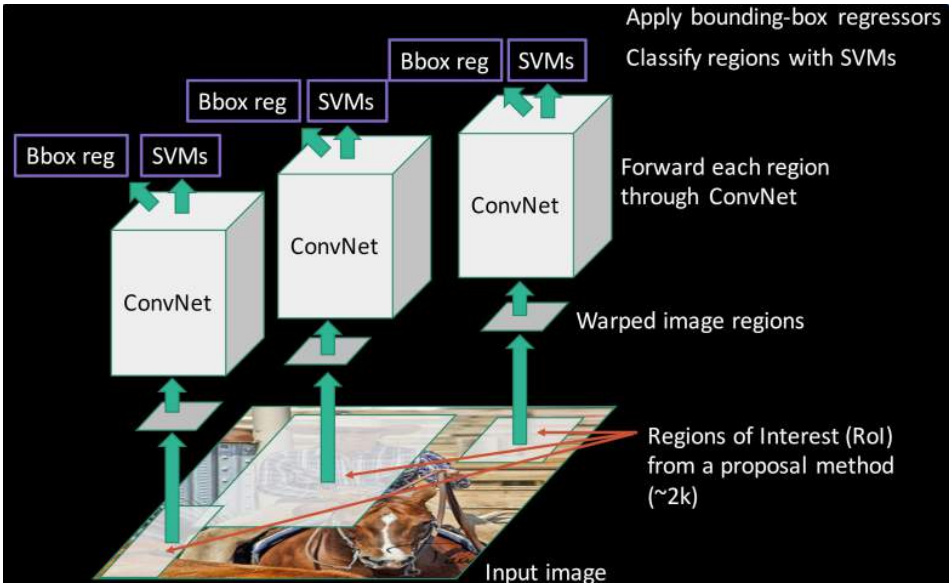


图 2-3 R-CNN 训练过程 [9]

（1）选出潜在目标候选框（ROI）

objectness[10], selective search[11], category-independent object proposals[12] 等很多论文都讲述了候选框提取的方法，R-CNN 使用 selective search 的方法选出 2000 个潜在物体候选框。

（2）训练一个好的特征提取器

R-CNN 的提出者使用卷积神经网络 AlexNet 提取 4096 维的特征向量，实际上使用 VGGNet/GoogLeNet/ResNet 等也都可以。细心的读者会发现，AlexNet 等网络要求输入的图片尺寸是固定的，而步骤（1）中的 ROI 尺寸大小不定，这就需要将每个 ROI 调整到指定尺寸，调整的方法有多种，可参见图 2-4，其中（a）是原始 ROI 图片，（b）是包含上下文的尺寸调整，（c）是不包含上下文的尺寸调整，（d）是尺度缩放。

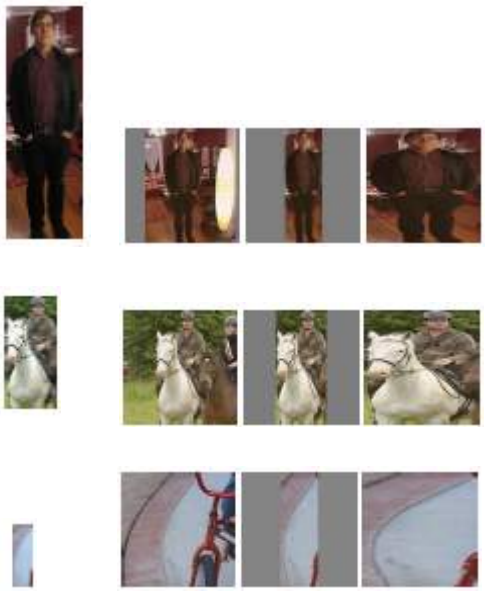


图 2-4　不同压缩方法图示

接下来，为了获得一个好的特征提取器，一般会在 ImageNet 预训练好的模型基础上做调整（因为 ImageNet 预测的种类较多，特征学习相对完善），唯一的改动就是将 ImageNet 中的 1000 个类别的输出改为（C+1）个输出，其中 C 是真实需要预测的类别个数，1 是背景类。新特征的训练方法是使用随机梯度下降（stochastic gradient descent，即 SGD），与前几章介绍的普通神经网络的训练方法相同。

提到训练，就一定要有正样本和负样本，这里先抛出一个用于衡量两个矩形交叠情况的指标：IOU（Intersection Over Union）。IOU 其实就是两个矩形面积的交集除以并集，如图 2-5 所示。一般情况下，当 IOU>=0.5 时，可以认为两个矩形基本相交，所以在这个任务中，假定两个矩形框中，1 个矩形代表 ROI，另一个代表真实的矩形框，那么当 ROI 和真实矩形框的 IOU>=0.5 时则认为是正样本，其余为负样本。

至此，R-CNN 的第二步特征提取器可以开始训练了，不过在训练过程中要注意，需要对负样本进行采样，因为训练数据中正样本太少会导致正负样本极度不平衡。最终在该步得到的是一个卷积神经网络的特征提取器，其特征是一个 4096 维特征向量。

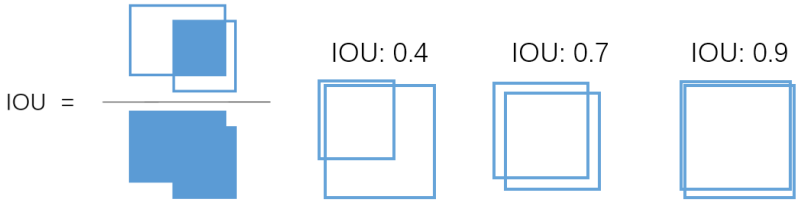


图 2-5 IOU 图示

（3）训练最终的分类器

这里为每个类别单独训练一个 SVM 分类器。这里面有个小技巧，SVM 的训练也需要选择正负样本，R-CNN 的提出者做了一个实验来选择最优 IOU 阈值，最终仅仅选择真实值的矩形框作为正样本。

注意 正负样本的选择比较有讲究，Fast R-CNN 和 Faster R-CNN 是根据 IOU 的大小选取正负样本，2.2.3 节有详细介绍。

（4）为每个类训练一个回归模型，用来微调 ROI 与真实矩形框位置和大小的偏差，如图 2-6 所示。

图 2-6　R-CNN 中的 ROI 结果微调 [9]

下面是所有检测问题都会用到的一块代码：IOU 的计算。

```
def bboxIOU (bboxA, bboxB):
    A_xmin = bboxA[0]
    A_ymin = bboxA[1]
    A_xmax = bboxA[2]
    A_ymax = bboxA[3]
    A_width = A_xmax - A_xmin
    A_height = A_ymax - A_ymin

    B_xmin = bboxB[0]
    B_ymin = bboxB[1]
    B_xmax = bboxB[2]
    B_ymax = bboxB[3]
    B_width = B_xmax - B_xmin
    B_height = B_ymax - B_ymin

    xmin = min(A_xmin, B_xmin)
    ymin = min(A_ymin, B_ymin)
    xmax = max(A_xmax, B_xmax)
    ymax = max(A_ymax, B_ymax)

    A_width_and = (A_width + B_width) - (xmax - xmin) # 宽的交集
    A_height_and = (A_height + B_height) - (ymax - ymin) # 高的交集

    if ( A_width_and <= 0.0001 or A_height_and <= 0.0001):
        return 0

    area_and = (A_width_and * A_height_and)
    area_or = (A_width * A_height) + (B_width * B_height)
    IOU = area_and / (area_or - area_and)

    return IOU
```

预测阶段有如下几个步骤：

（1）同样地，使用 selective search 方法先选出 2000 个 ROI。

（2）所有 ROI 调整为特征提取网络所需的输入大小并进行特征提取，得到 2000 个 ROI 对应的 2000 个 4096 维的特征向量。

（3）将 2000 个特征向量分别输入到 SVM 中，得到每个 ROI 预测的类别。

（4）通过回归网络微调 ROI 的位置。

（5）最终使用非极大值抑制（Non-Maximum Suppression，NMS）方法对同一个类别的 ROI 进行合并得到最终检测结果。NMS 的原理是得到每个矩形框的分数（置信度），如果两个矩形框的 IOU 超过指定阈值，则仅仅保留分数大的那个矩形框。

以上就是 R-CNN 的全部过程，我们可以从中看出，R-CNN 存在的一些问题：

- 不论是训练还是预测，都需要对 selective search 出来的 2000 个 ROI 全部通过 CNN 的 forward 过程来获取特征，这个过程非常慢。
- 卷积神经网的特征提取器和用来预测分类的 SVM 是分开的，也就是特征提取的过程不会因 SVM 和回归的调整而更新。
- R-CNN 有非常复杂的操作流程，而且每一步都是分裂的，如特征提取器通过 softmax 分类获得，最终分类结果由 SVM 获得，矩形框的位置通过回归方式获得。

2.2 Fast R-CNN

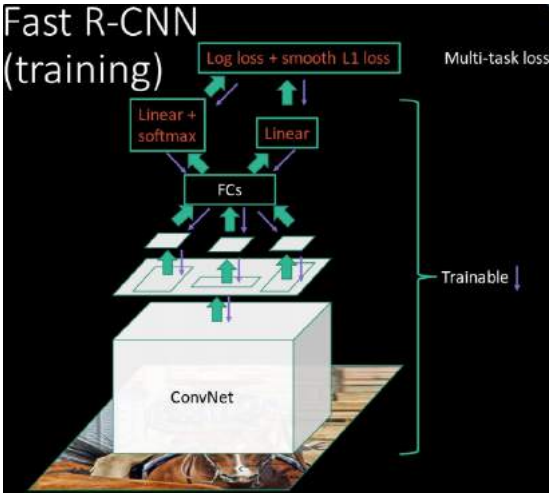
针对 R-CNN 的 3 个主要问题，我们思考一下是否有更好的解决方案。

首先是速度，2000 个 ROI 的 CNN 特征提取占用了大量的时间，是否可以用更好的方法，比如共享卷积层来同时处理所有 2000 个 ROI？

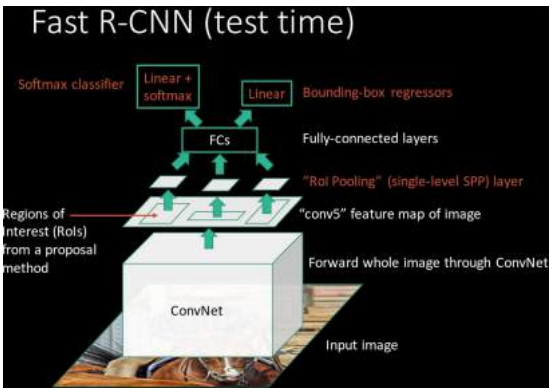
其次是 CNN 的特征不会因 SVM 和回归的调整而更新。

R-CNN 的操作流程比较复杂，能否有更好的方式使得训练过程成为端到端的?

接下来我们将介绍 Firshick 等人于 2015 年提出的 Fast R-CNN[2]，它非常巧妙地解决了 R-CNN 主要的几个问题。Fast R-CNN 的训练和预测过程如图 2-7 所示，具体训练步骤如下：



（a）Fast R-CNN 训练过程示意图



（b）Fast R-CNN 预测过程示意图

图 2-7　Fast R-CNN 训练和预测过程示意图 [9]

（1）将整张图片和 ROI 直接输入到全卷积的 CNN 中，得到特征层和对应在特征层上的 ROI（特征层的 ROI 信息可用其几何位置加卷积坐标公式推导得出）。

（2）与 R-CNN 类似，为了使不同尺寸的 ROI 可以统一进行训练，Fast R-CNN 将每块候选区域通过池化的方法调整到指定的 M*N，所以此时特征层上调整后的 ROI 作为分类器的训练数据。与 R-CNN 不同的是，这里将分类和回归任务合并到一起进行训练，这样就将整个流程串了起来。Fast R-CNN 的池化示意图如图 2-8 所示，即先将整张图通过卷积神经网络，然后在特征层上找到 ROI 对应的位置并取出，对取出的 ROI 进行池化（此处的池化方法有很多）。池化后，所有 2000 个 M*N 个训练数据通过全连接层并分别经过 2 个 head：softmax 分类以及 L2 回归，最终的损失函数为分类和回归的损失函数的加权和。通过这样的方式就实现了端到端的训练。

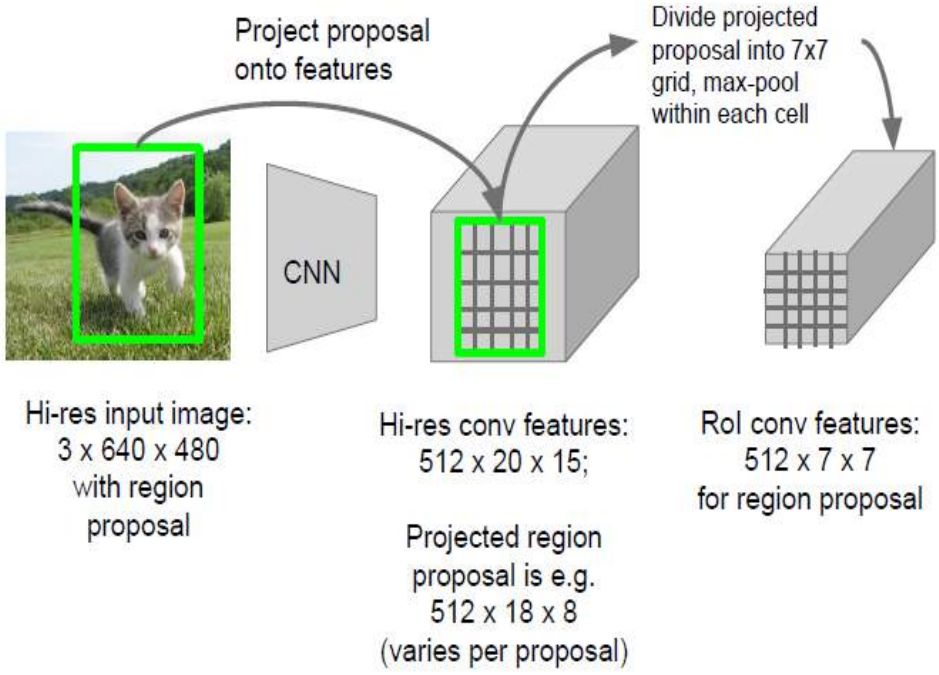


图 2-8 Fast R-CNN 中的 ROI Pooling[9]

Fast R-CNN 极大地提升了目标检测训练和预测的速度，如图 2-9 所示。从图 2-9 中我们可以看出，Fast R-CNN 把训练时长从 R-CNN 的 84 小时下降到了 8.75 小时，每张图片平均总预测时长从 49 秒降到 2.3 秒。从图 2-9 中我们还可以看出，在 Fast R-CNN 预测的这 2.3 秒中，真正的预测过程仅仅占用 0.32 秒，而 Region proposal 占用了绝大多数的时间。

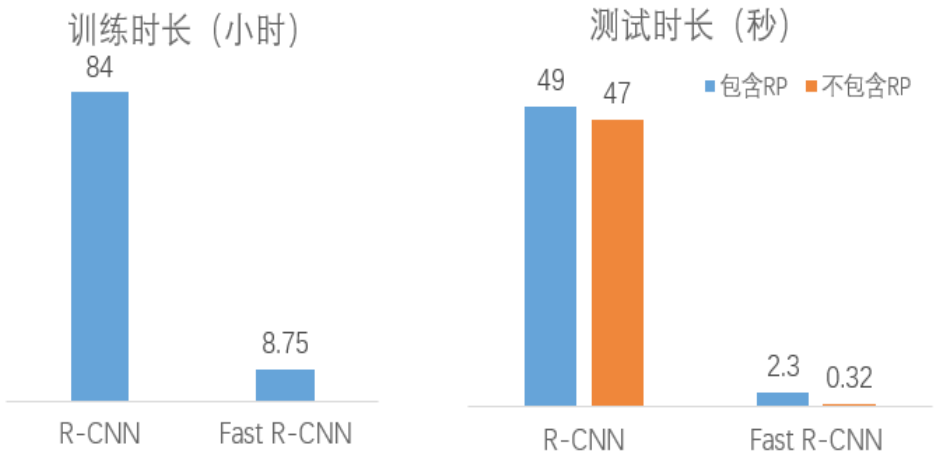


图 2-9　R-CNN 和 Fast R-CNN 训练和测试时间对比

2.3　Faster R-CNN

Faster R-CNN[3] 作为目标检测的经典方法在现今很多实战项目和比赛中频频出现。其实，Faster R-CNN 就是在 Fast R-CNN 的基础上构建一个小的网络，直接产生 region proposal 来代替通过其他方法（如 selective search）得到 ROI。这个小型的网络被称为区域预测网络（Region Proposal Network，RPN）。Faster R-CNN 的训练流程如图 2-10 所示，其中的 RPN 是关键，其余流程基本和 Fast R-CNN 一致。

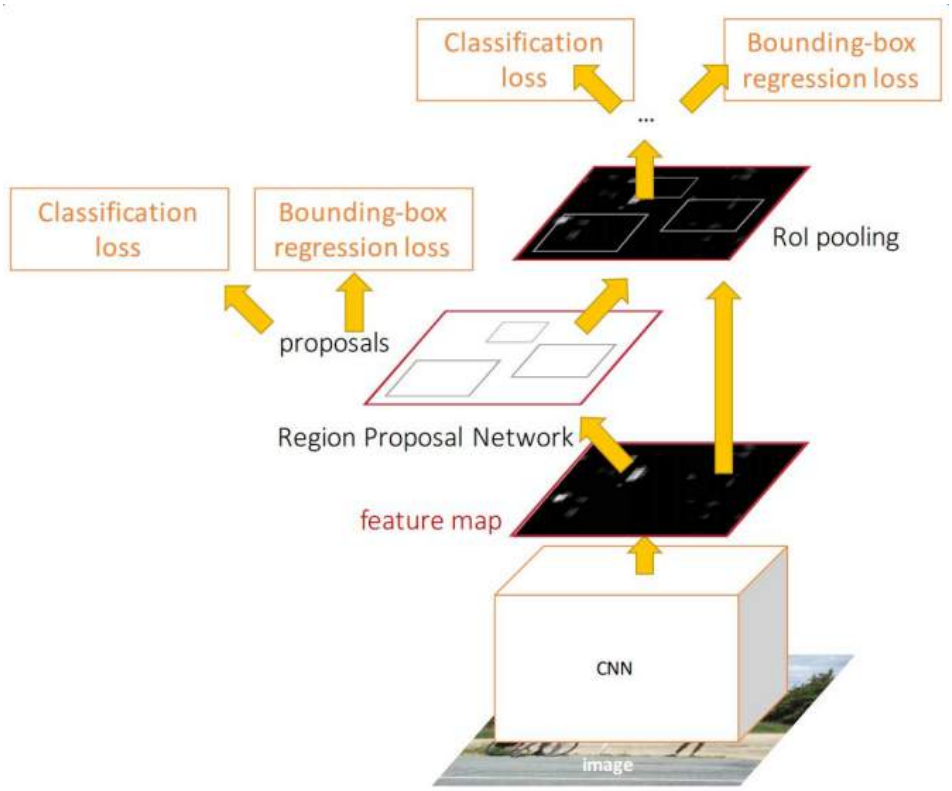
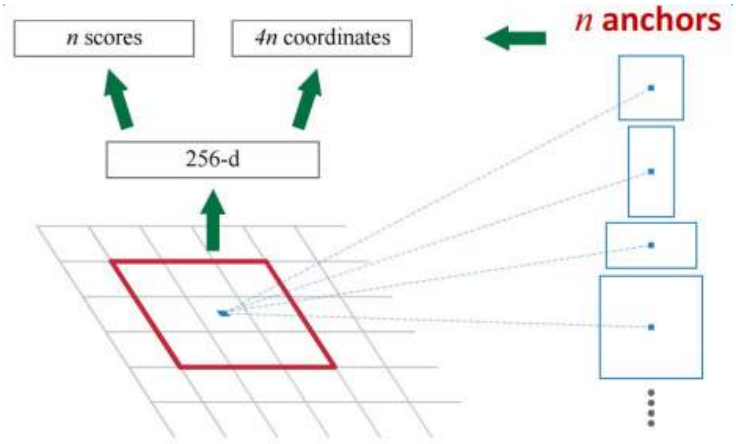


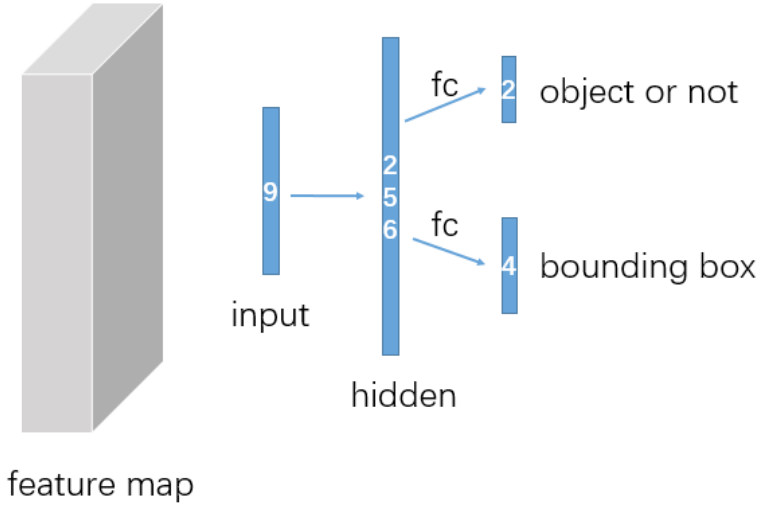
图 2-10 Faster R-CNN 训练流程 [9]

RPN 的思想是构建一个小的全卷积网络，对于任意大小的图片，输出 ROI 的具体位置以及该 ROI 是否是物体。RPN 网络在卷积神经网的最后一个特征层上滑动。

接下来我们对着图 2-11 来进一步解释 RPN 网络。图 2-11（a）中最下面灰色的网格表示卷积神经网络的特征层，红框表示 RPN 网络的输入，其大小为 3*3，而后连接到 256 维的一个低维向量。这 3*3 的窗口滑动经过整个特征层，并且每次计算都将经过这 256 维的向量并最终输出 2 个结果：该 3*3 滑动窗口位置中是否有物体以及该滑动窗口对应物体的矩形框位置。如果还是不好理解，我们将图 2-11（a）中的 RPN 网络顺时针旋转 90 度，如图 2-11（b）所示，现在可以很清晰地看出神经网络结构了，这里 input 维度是 9，即图 2-11（a）中的 3*3 大小。



（a）



（b）

图 2-11　RPN 网络原理 [3]

为了适应多种形状的物体，RPN 网络定义了 k 种不同尺度的滑窗（因为有的目标是长的，有的是扁的，有的是大的，有的是小的，统一用一个 3*3 的滑窗难以很好地拟合多种情况），这里给它一个专业的名词——anchor，每个 anchor 都是以特征层（feature map）上的像素点为中心并且根据其尺度大小进行后续计算的。在

Faster-RCNN 论文中，滑窗在特征层的每个位置上使用 3 种大小和 3 种比例共 3*3=9 种 anchor，在图 2-11（a）中 n=9。

根据上面的介绍我们知道 RPN 网络有 2 类输出：二分类网络输出是否是物体，回归网络返回矩形框位置对应的 4 个值。

接下来，我们看一下训练过程中的一些细节问题。首先，针对分类任务，对于滑窗产生的每一个 anchor 都计算该 anchor 与真实标记矩形框的 IOU。当 IOU 大于 0.7 时，便认为该 anchor 中含有物体；当 IOU 小于 0.3 时，便认为该 anchor 中不含物体；当 IOU 介于 0.3-0.7 之间时，则不参与网络训练的迭代过程。

对于回归任务，这里定义为 anchor 中心点的横、纵坐标以及 anchor 的宽高，学习目标为 anchor 与真实 bbox 在这四个值上的偏移。RPN 网络为一个全卷积网络，可以用随机梯度下降的方式端到端地进行训练。

这里需要注意，训练过程中能与真实物体矩形框相交的 IOU 大于 0.7 的 anchor 并不多，绝大多数都是负样本，因此会导致正负样本比例严重失衡，从而影响识别效果。因此，在 RPN 训练的过程，每个 batch 进行随机采样（每个 batch 中有 256 个样本）并保证正负样本的比例为 1:1，而当正样本数量小于 128 时，取全部的正样本，其余的随机使用负样本进行补全。

使用 RPN 网络产生 ROI 的好处是可以和检测网络共享卷积层，使用随机梯度下降的方式端到端地进行训练。接下来我们看下 Faster R-CNN 的训练过程：

（1）使用 ImageNet 预训练好的模型训练一个 RPN 网络。

（2）使用 ImageNet 预训练好的模型，以及第（1）步里产生的建议区域训练 Fast R-CNN 网络，得到物体实际类别以及微调的矩形框位置。

（3）使用（2）中的网络初始化 RPN，固定前面卷积层，只有调整 RPN 层的参数。

（4）固定前面的卷积层，只训练并调整 Fast R-CNN 的 FC 层。

有了 RPN 的帮助，Faster R-CNN 的速度大大提升，（如图 2-12 所示。RCNN、Fast R-CNN、Faster R-CNN 几个模型的对比如图 2-13 所示。

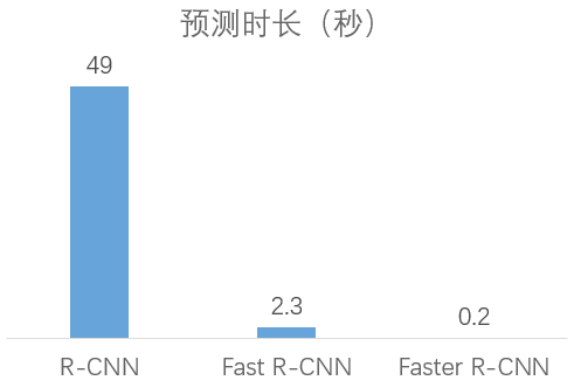


图 2-12　RCNN、Fast R-CNN、Faster R-CNN 模型耗时对比

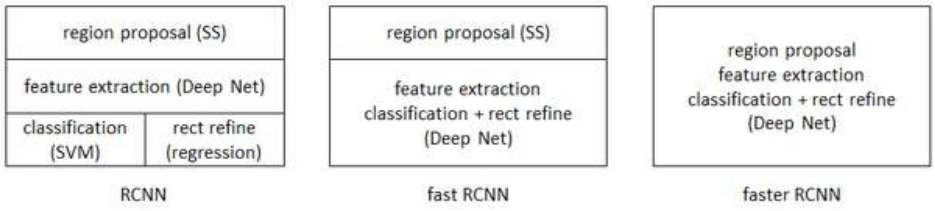


图 2-13　RCNN、Fast R-CNN、Faster R-CNN 模型对比

从 R-CNN 到 Faster R-CNN，前面讲了基于 proposal 想法做目标检测的发展史，这种思路分为产生 proposal 和检测两个步骤，可以得到相对较好的精度，但缺点是速度较慢。接下来我们介绍另外几种常用于检测的方法。

2.4　YOLO

由于在 R-CNN 的系列算法中都需要首先获取大量 proposal，但 proposal 之间有很大的重叠，会带来很多重复的工作。YOLO[5] 一改基于 proposal 的预测思路，将输入图片划分成 S*S 个小格子，在每个小格子中做预测，最终将结果合并，如图

2-14 所示。接下来我们看一下 YOLO 学习的关键步骤：

（1）YOLO 对于网络输入图片的尺寸有要求，首先需要将图片缩放到指定尺寸（448*448），再将图片划分成 S*S 的小格。

（2）每个小格里面做这几个预测：该小格是否包含物体、包含物体对应的矩形框位置以及该小格对应 C 个类别的分数是多少。因此，每个小格需要预测的的维度为 B *（1+4）+ C，其中 B 代表每个小格最多可能交叠物体的个数，1 为该小格是否包含物体的置信度，4 用来预测矩形框，C 表示任务中所有可能的类别个数（不包含背景）。因此，YOLO 网络最终特征层的大小为 S * S *（B * 5 + C），图 2-14 中特征层大小即为 7 x 7 x (2*5 + 20) =7*7*30（Pascal VOC2012 目标检测数据集共有 20 种类别）。

由于 YOLO 直接将输入图片划分为 S*S 个小格，不需要产生 proposal 的过程，所以速度比 Faster R-CNN 快很多，但是因为粒度较粗，所以精度相比 Faster R-CNN 略逊一筹。YOLO 的主要贡献是为目标检测提供了另一种思路，并使实时目标检测成为可能。近几年，YOLOv2 和 YOLOv3 接连推出，感兴趣的读者可以参考附录的 [6][7]。

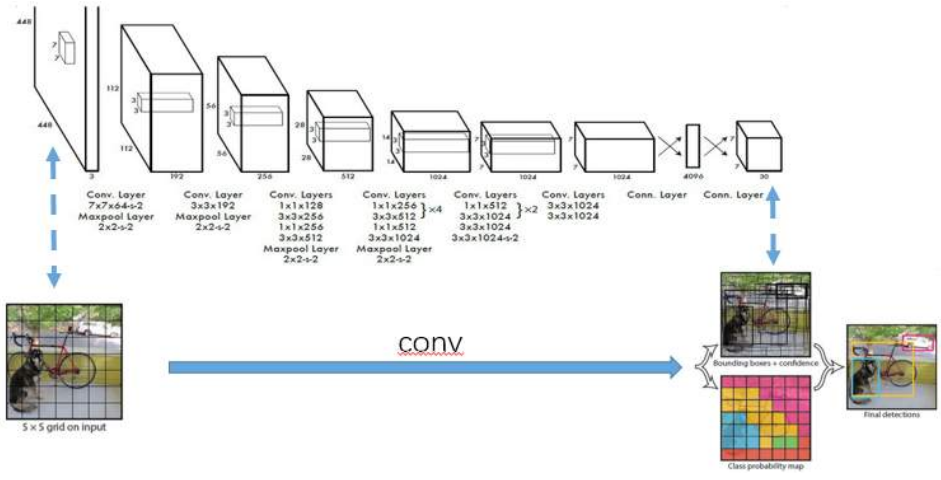


图 2-14 基于 Pascal VOC2012 目标检测数据集的 YOLO 示意图

2.5　SSD

SSD[4] 同时借鉴了 YOLO 网格的思想和 Faster R-CNN 的 anchor 机制，使得 SSD 可以快速进行预测的同时又可以相对准确地获取目标的位置。

图 2-15 中的（b）和（c）分别代表不同的特征层，（c）相对（b）离最终预测结果较近，因此跨越同样像素个数能检测的目标就越大。按照图 2-15 所示，在特征层（b）的每个点上都将产生 4 个不同大小的 anchor（1:1 两个，1:2 两个），在特征层（c）上也是如此。因此，根据真实目标矩形框与每个 anchor 的 IOU 大小计算可知，（b）中有 2 个 anchor 为正样本，（c）中 1 个 anchor 为正样本。

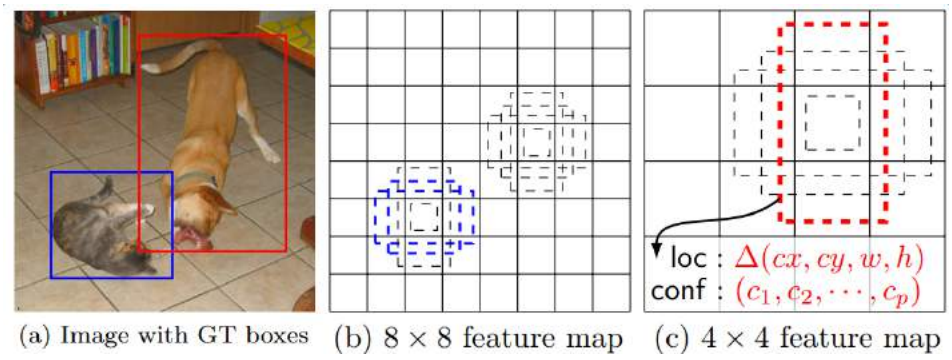


图 2-15　SSD 特征层与 anchor 示意图 [4]

对比之前学过的 Faster R-CNN，接下来我们介绍 SSD 的一些特点。

- 使用多尺度特征层进行检测。在 Faster Rcnn 的 RPN 中，anchor 是在主干网络的最后一个特征层上生成的，而在 SSD 中，anchor 不仅仅在最后一个特征层上产生，在几个高层特征层处同时也在产生 anchor。如图 2-16 所示，SSD 在 VGG16 的 conv6 开始，在 conv7、conv8、conv9、conv10 都产生 anchor。这些特征层大小依次递减，使得 SSD 可以检测不同尺度的目标。这里简单解释下，比如同样一个 3*3 的 anchor，它在 conv6 看到的目标（感受野）就要远小于 conv10 看到的目标，可以理解为靠前的特征层用于检测小目标，而靠后的特征层用来检测大目标。与 RPN 网络（2.2.3 中介绍）产生

anchor 的方法类似，SSD 也是在特征层的每个点上产生多个比例、多个尺度的 n 个 anchor。如图 2-15（b）是一个 8*8 的特征层，每个小方格子是一个特征点，每个特征点上可以产生宽高比为 1:1，1:2，1:3，大小多个尺度的 anchor。

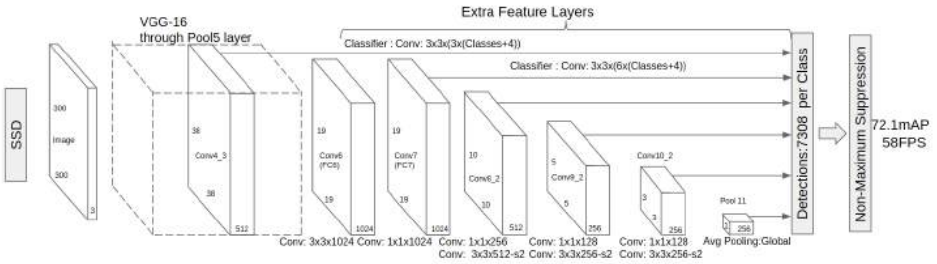


图 2-16 SSD 结构图 [4]

- SSD 中所有特征层产生的 anchor 都将经过正负样本的筛选（在 2.2.3 介绍过如何使用 IOU 进行 anchor 的筛选）后直接进行分类分数以及 bbox 位置的学习。也就是说，特征层上生成的正负样本直接进行最终的分类（ClassNum 个类别）以及 bbox 的学习，不像 Faster R-CNN 那样先在第一步学是否有物体（只有 0/1 两个类别）和 bbox 位置，然后在第二步学最终的分类（ClassNum 个类别）以及对 bbox 位置的微调。

实际应用时我们往往不仅关注精度，很多情况下也要考虑速度，比如对视频内容进行实时地检测，这时候我们就希望有方法可以很好的做速度和精度的平衡。YOLO 是第一个提出来效果很好的 1-stage 检测方法，SSD 借鉴了它的一些思想并在其基础上做了改进，做到了比较好的平衡。

第三章　目标检测的产业应用实践

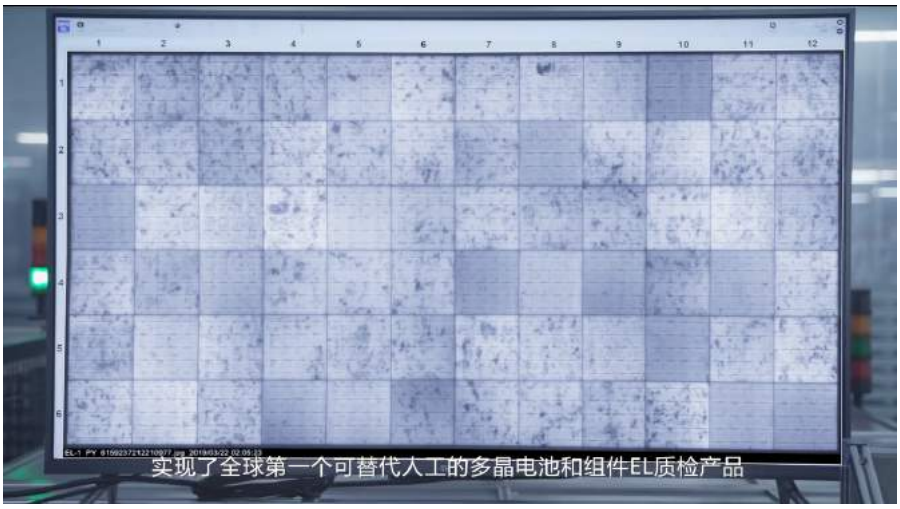
前三章节，具体讲解了目标检测的技术应用，技术如何和产业相结合，发挥出最大的价值，也是我们最为关注的。

在经济稳预期的形势下，国内制造业企业正在加快转型升级的步伐。阿里作为一家有情怀和使命感的科技公司，我们希望通过技术手段来帮助传统企业实现转型升级。正泰新能源与阿里合作引入人工智能图像技术进入生产车间，其无人质检生产线目前已使用将近一年时间，成为了生产制造的重要力量。

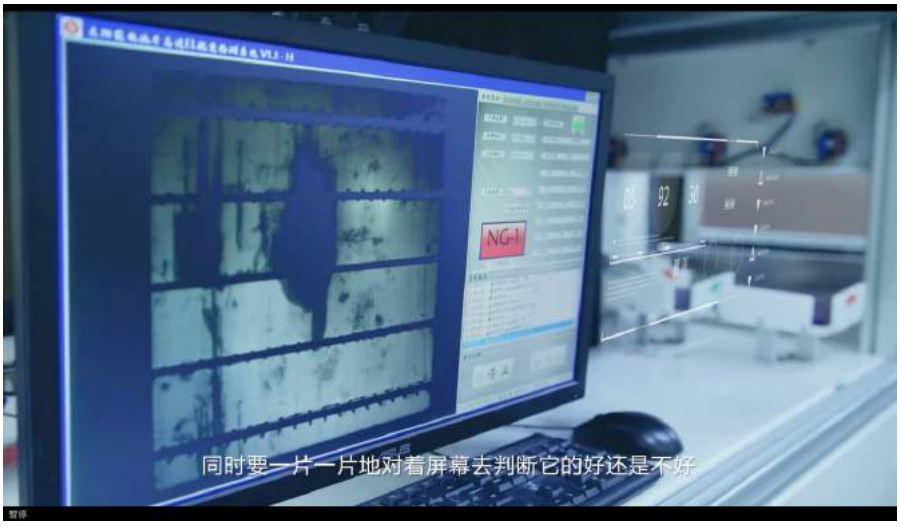
在光伏行业，质检环节长期面临**专业度高**、**招工难**、**人力不足**等问题。

工业自动化水平较高的德国曾推出过组件 EL 质检技术，但只针对典型缺陷，仅能做到辅助人工（无法替代人工）。在国内，光伏企业在智能 AI 识别技术领域做了近 10 年的尝试，但多晶电池和组件的自动质检远未达到工业生产水平。

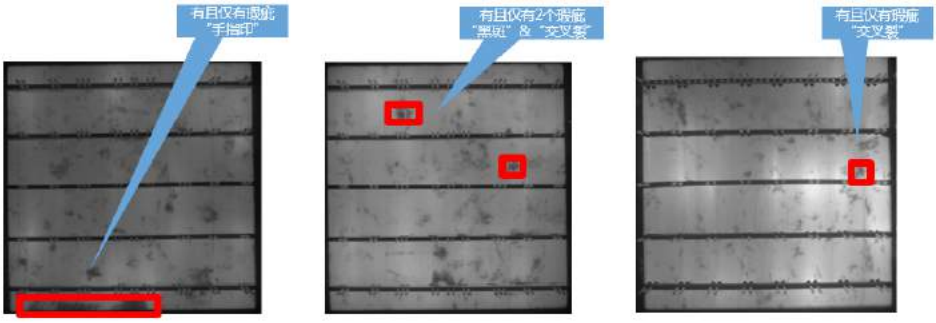
阿里巴巴利用 AI 技术，实现了**全球第一个**可代替人工的多晶电池和组件 EL 质检产品。



在应用 AI 检测之前，这个质检环节需要熟练员工一片一片地对着屏幕去判断电池片好还是不好，同时要拿手柄去确认，视觉疲劳、会导致准确率的下降，同时要培养这样一个**熟练员工**，**基本上要 3 个月**的时间。

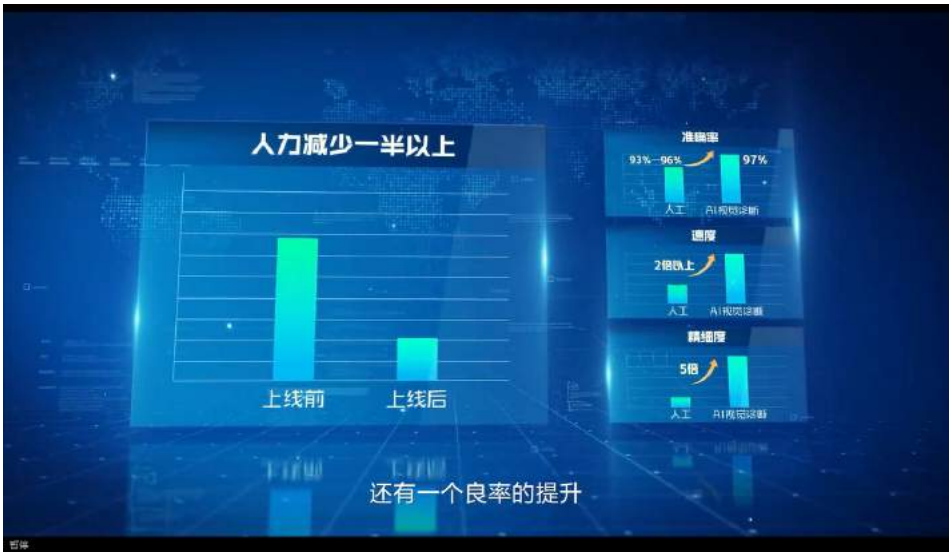


多晶电池片之所以难，是因为电池片本身有很多暗纹，这些暗纹和某些瑕疵在图像特征上比较相似，而且瑕疵本身的大小、长宽比、类间距等也很大，因此在算法上有着非常大的挑战。



在单晶、多晶电池片质检在线上稳定运行半年后，阿里推出单晶、多晶组件 EL 质检功能，目前已在产线运行且精度稳定在 95% 以上。组件由 6*10/6*12 块电池组成，因此只要有一个地方识别错误，整张组件便识别错误，因此其识别难度远大于电池片。组件 95% 以上的精度意味着单张电池片的识别精度要求远远超过 99%。

正泰新能源在应用阿里的 AI 检测之后，在“降本增效”上已经有了非常明显的优势。



阿里云未来将与更多的企业联合，书写智能制造新篇章。

附录

[1] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 580-587.

[2] Girshick R. Fast r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1440-1448.

[3] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[C]//Advances in neural information processing systems. 2015: 91-99.

[4] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]// European conference on computer vision. Springer, Cham, 2016: 21-37.

[5] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.

[6] Redmon J, Farhadi A. YOLO9000: better, faster, stronger[J]. arXiv preprint, 2017.

[7] Redmon J, Farhadi A. Yolov3: An incremental improvement[J]. arXiv preprint arXiv:1804.02767, 2018.

[8] Dai J, Li Y, He K, et al. R-fcn: Object detection via region-based fully convolutional networks[C]//Advances in neural information processing systems. 2016: 379-387.

[9] Fei-Fei Li, Justin Johnson, Serena Yeung et al. CS231n: Convolutional Neural Networks for Visual Recognition

[10] Cheng M M, Zhang Z, Lin W Y, et al. BING: Binarized normed gradients for objectness estimation at 300fps[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 3286-3293.

[11] Uijlings J R R, Van De Sande K E A, Gevers T, et al. Selective search for object recognition[J]. International journal of computer vision, 2013, 104(2): 154-171.

[12] Endres I, Hoiem D. Category independent object proposals[C]//European Conference on Computer Vision. Springer, Berlin, Heidelberg, 2010: 575-588.

[13] Law H, Deng J. Cornernet: Detecting objects as paired keypoints[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 734-750.

[14] Xin Lu, Buyu Li, Yuxin Yue, Quanquan Li, Junjie Yan Grid R-CNN arXiv preprint, 2018.

扫码加入阿里云
视觉计算交流群

扫码关注阿里云人工智
能社区，了解前沿动态