

Tutorial Library Dearning

Library Dearning adalah pustaka AI berbasis NumPy yang lahir pada tanggal 26 April 2025 dan dibuat oleh Oriza. Dirancang untuk support berbagai device dari yang low sampai end. Ia mendukung autograd sederhana, training modular, dan berbagai jenis neuron canggih. Terdiri dari beberapa modul seperti model, utils, dan GPUD, library ini cocok untuk riset dan eksperimen AI ringan.

Modul: model

Modul model menyimpan class utama CustomAIModel.

Fungsi utamanya adalah membangun struktur model AI dari layer-layer sederhana dan neuron eksperimental.

Ia mendukung penambahan Dense layer, fungsi aktivasi, memory neuron, attention, spiking, NALU, dan Mixture of Experts.

Contoh 1:

```
model = CustomAIModel()  
model.add(Dense(4, 8))  
model.add(Activation("relu"))
```

Contoh 2:

```
model.addneuron(kind="attention")  
y = model.forward(data)
```

Modul: utils

Modul utils menyediakan fungsi bantu penting seperti preprocessing data dan evaluasi model.

Fungsi `preprocess_data` melakukan normalisasi otomatis dan paralel jika data besar.

Fungsi `evaluate_model` menilai performa model AI untuk klasifikasi atau regresi, lalu mengembalikan metrik akurat.

Contoh 1:

```
X = preprocess_data(X_raw)
```

```
hasil = evaluate_model(model, X, y)
```

Contoh 2:

```
optim = Optimizer(method="adam")
```

```
w2, b2 = optim.update(w, b, grad_w, grad_b, layer_idx=1)
```

Modul: training

Modul training menyediakan fungsi utama untuk proses pelatihan model.

Dilengkapi dengan loader dataset, loader gambar, evaluator otomatis, dan fitur autograd (jika tersedia).

Training dapat dijalankan dengan threading agar responsif dan bisa divisualisasikan loss-nya.

Contoh 1:

```
model = CustomAIModel(...)
```

```
train_model(model, task="regression", visualize=True)
```

Contoh 2:

```
X, y = load_dataset("classification")
```

```
model.train(X, y, epochs=20)
```

Modul: pipeline

Modul pipeline menyediakan integrasi lengkap dari berbagai komponen Dearning. Tersedia fungsi untuk membangun model, menjalankan pipeline pelatihan, reinforcement learning, analisis teks, text-to-speech, dan pelacakan kode.

Fungsi `build_basic_model()` membuat arsitektur neural network sederhana. Fungsi `run_pipeline()` dapat menjalankan tugas berbasis teks seperti analisis, translasi, tts, dan konversi biner.

`train_model_pipeline()` memungkinkan training model menggunakan DOnsotensor atau NumPy biasa. Juga mendukung visualisasi loss dan pelacakan trace autograd.

Contoh 1:

```
model = build_basic_model()
model, eval = train_model_pipeline(model)
```

Contoh 2:

```
run_pipeline(task="translate", input_data="halo dunia")
```

Modul: AI_tools

Modul AI_tools adalah kumpulan fitur AI praktis seperti.

Tersedia tools untuk pemrosesan teks, pengolahan gambar dan audio, navigasi GPS, reinforcement learning, hingga manajemen memori sederhana.

Untuk teks, digunakan TextBlob untuk analisis sentimen, ekstraksi kata benda, POS tagging, dan terjemahan otomatis.

Untuk RL, tersedia RLTools yang mendukung Q-learning dan random agent berbasis simple_rl.

Contoh 1:

```
dlp = DLP()
```

```
result = dlp.process("saya senang sekali", translate_to="en")
```

Contoh 2:

```
rl = RLTools()
```

```
rl.add_q_agent()
```

```
rl.run(epsisodes=10)
```

Modul: GPUD

Modul GPUD merupakan bagian dari libraries Dearnig. Sebuah utilitas untuk mempercepat perhitungan numerik menggunakan GPU (via ArrayFire) atau TPU . Jika perangkat keras akselerator tidak tersedia, fungsi akan otomatis menggunakan fallback CPU (NumPy) agar tetap kompatibel.

`use_gpu_conditionally(data_size)`: Mengembalikan True jika GPU aktif dan data cukup besar.

- `af_array(x)`: Mengonversi array NumPy ke array ArrayFire (jika GPU tersedia).

- `gpu_dot(x, y)`: Melakukan operasi dot product dengan GPU jika tersedia.

- `gpu_add(x, y)`: Menjumlahkan dua array menggunakan GPU jika tersedia.

- `gpu_mean(x)`: Menghitung nilai rata-rata array menggunakan GPU jika tersedia.

- `gpu_info()`: Mengembalikan informasi perangkat GPU (atau pesan fallback).

- `is_gpu_on()`: Mengembalikan True jika GPU tersedia.

- `TPUChecker`: Kelas untuk mendeteksi dan menginisialisasi TPU menggunakan TensorFlow.

Contoh 1:

```
from dearning.GPUD import gpu_dot, af_array
import numpy as np
```

```
a = np.random.rand(1024, 512)
```

```
b = np.random.rand(512, 1)
```

```
a_gpu = af_array(a)
```

```
b_gpu = af_array(b)
```

```
hasil = gpu_dot(a_gpu, b_gpu)
```

```
print("Dot product:", hasil)
```

Contoh 2:

```
from dearning.GPUD import gpu_info, is_gpu_on, TPUChecker
```

```
print("Apakah GPU aktif?", is_gpu_on())  
print("Info GPU:", gpu_info())
```

```
tpu = TPUChecker()  
print("Status TPU:", tpu.status())
```


Modul: multimodel

Modul multimodel.py berisi dua model AI pretrained dalam libraries Dearnig:

1. AIModel

Merupakan model AI.Model yang bernama "simpleAI" menggunakan arsitektur feedforward neural network dan dapat:

- Memprediksi keluaran dari fitur numerik teks.
- Belajar dari kesalahan dan memperbarui model lewat train_from_memory().
- Menganalisis teks menggunakan neural network atau fallback ke TextBlob jika diperlukan (analyze_text).
- Menyimpan dan memuat model AI dari file.

Fitur unik:

- learn_from_mistake() dan train_from_memory() memungkinkan pembelajaran berbasis kesalahan secara interaktif.
- analyze_text() menggabungkan AI internal dan fallback NLP (DLP).

2. locAler

Model AI yang menganalisis arah atau lokasi dari teks perintah pengguna seperti "go to hospital" dan mencarikan jalur terpendek ke tujuan tersebut.

- Menggunakan modul NavigationHelper.
- Menyediakan integrasi GPS (lokasi langsung) dan mode simulasi rute.
- Cocok untuk sistem berbasis arah, lokasi, atau perintah natural language sederhana.

peringatan:

Model AI ini mempunyai libraries native java dan memerlukan izin untuk bisa menggunakannya

Contoh 1:

```
from dearning.multymodel import AImodel
```

```
ai = AImodel()
```

```
hasil = ai.analyze_text("CEK ANGKA 123")
```

```
print("Teks:", hasil["text"])
```

```
print("Output:", hasil["output"])
```

```
print("Confidence:", hasil["confidence"])
```

Contoh 2:

```
from dearning.multymodel import locAler
```

```
lokal = locAler()
```

```
arah = lokal.analyze_direction_from_text("navigate to school")
```

```
print("Rute:", arah)
```

Modul: libD

Modul libD hanya lah paduan libraries NumPy, Scipy, dan math. Dan ketiga libraries itu hanya berganti nama saja seperti scipy menjadi sciD.

Contoh 1:

```
from libD import nd, Domath
```

```
x = nd.array([1, 2, 3])
```

```
y = nd.array([4, 5, 6])
```

```
dot_product = nd.dot(x, y)
```

```
akar = Domath.sqrt(16)
```

```
sinus_pi = Domath.sin(Domath.pi)
```

```
print("Hasil dot product:", dot_product)    # 1*4 + 2*5 + 3*6 = 32
```

```
print("Akar dari 16:", akar)                # 4.0
```

```
print("Sinus  $\pi$ :", sinus_pi)          # Hampir 0
```

Contoh 2:

```
from libD import sciD
```

```
import numpy as np
```

```
if sciD.stats:
```

```
    data = np.random.normal(loc=0, scale=1, size=1000)
```

```
    mean = sciD.stats.tmean(data)
```

```
    mode = sciD.stats.mode(data)
```

```
    print("Rata-rata:", mean)
```

```
    print("Modus:", mode.mode[0])
```

```
if sciD.optimize:
```

```
    f = lambda x: (x - 3)**2 + 4
```

```
    res = sciD.optimize.minimize(f, x0=0)
```

```
    print("Minimum ditemukan di:", res.x[0])
```

Modul: AI_core

Modul AI_core merupakan

1. CodeTracker

Kelas ini digunakan untuk melacak pemanggilan fungsi. Setiap kali fungsi yang diberi dekorator @DOtensor dijalankan, akan tercatat:

- Kapan fungsi dipanggil.
- Hasil kembalian dari fungsi.

Berikut fungsinya:

- __call__: Dekorator untuk membungkus fungsi.
- get(): Mengambil seluruh log dalam bentuk list.
- clear(): Menghapus semua log.

2. BinaryConverter

Kelas ini digunakan untuk mengonversi kode Python ke representasi biner (dan sebaliknya).berikut fungsinya:

- code_to_binary(string): Mengubah string ke biner (berbasis ASCII).
- file_to_binary(path): Membaca isi file dan mengubahnya ke biner.
- binary_to_code(binary_string): Mengubah biner ASCII kembali ke teks.

Contoh 1:

```
from AI_core import CodeTracker
```

```
@DOtensor
```

```
def tambah(a, b):
```

```
    return a + b
```

```
tambah(5, 3)
```

```
tambah(2, 7)
```

```
for log in tracker.get():
```

```
    print(log)
```

Contoh 2:

```
from AI_core import BinaryConverter
```

```
kode = "halo"
```

```
biner = BinaryConverter.code_to_binary(kode)
```

```
print("Hasil biner:", biner)
```

```
asli = BinaryConverter.binary_to_code(biner)
```

```
print("Kembali ke kode:", asli)
```

Terima kasih

Terima kasih telah mengikuti tutorial lengkap penggunaan library dearning!

Dalam tutorial ini, kita telah mempelajari:

- Cara membangun model AI menggunakan modul model dan training.
- Mengelola dan memproses data dengan utils.
- Menjalankan pipeline otomatis dengan pipeline.

Memanfaatkan fitur-fitur canggih seperti analisis teks, navigasi AI, audio, gambar, reinforcement learning, dan GPS lewat AI_tools. Menggunakan alat bantu seperti AI_core untuk debugging dan konversi kode.

Serta tambahan performa lewat GPUD dan utilitas libD.

Dengan struktur modular dan fleksibel, dearning dapat digunakan untuk keperluan edukasi, eksperimen, hingga pengembangan prototipe AI ringan di perangkat lokal, termasuk di Android melalui Pydroid 3.

Semoga library ini membantu kamu belajar dan bereksperimen dengan AI lebih mudah dan menyenangkan.

Jangan ragu untuk mencoba, mengubah, dan mengembangkan!