Zero to Mastery Learn PyTorch for Deep Learning Course Summary

05. PyTorch Going Modular

Nama : Al Ghifary Akmal Nasheeri

NIM : 1103201242

Kelas : TK-44-06

Modul ini membahas cara mengubah kode notebook menjadi skrip Python. Proses ini melibatkan konversi sel kode yang paling berguna dari notebook 04, "PyTorch Custom Datasets," menjadi serangkaian skrip Python yang disimpan dalam direktori bernama "going_modular."

Apa itu "going modular"? "Going modular" melibatkan transformasi kode notebook (dari Jupyter Notebook atau Google Colab notebook) menjadi serangkaian skrip Python yang menawarkan fungsionalitas serupa. Misalnya, notebook kode dapat diubah menjadi beberapa file Python, seperti **data_setup.py**, **engine.py**, **model_builder.py**, **train.py**, dan **utils.py**, sesuai dengan kebutuhan proyek.

Mengapa Anda ingin "going modular"? Meskipun notebook berguna untuk eksplorasi dan eksperimen, untuk proyek berskala besar, skrip Python lebih direkomendasikan karena lebih mudah direproduksi dan dijalankan. Dibandingkan dengan notebook, skrip Python memiliki keuntungan dalam hal versioning, penggunaan bagian tertentu, dan kemasan kode yang lebih baik.

Pros dan Cons dari Notebooks vs. Python Scripts: *Notebooks:*

- **Pros:** Mudah untuk bereksperimen, memulai, dan berbagi. Visual dan grafis.
- **Cons:** Versioning sulit, sulit menggunakan bagian tertentu, dan visual yang berlebihan.

Python Scripts:

- **Pros:** Dapat mengemas kode, menggunakan Git untuk versioning, mendukung proyek berskala besar.
- **Cons:** Bereksperimen kurang visual, harus menjalankan seluruh skrip, kurang dukungan di cloud vendors untuk notebook.

Workflow Umum: Proyek machine learning sering dimulai dengan notebook untuk eksperimen cepat, kemudian bagian-bagian kode yang paling berguna dipindahkan ke skrip Python.

PyTorch in the Wild: Banyak repositori kode proyek machine learning berbasis PyTorch menyertakan instruksi menjalankan kode PyTorch dalam bentuk skrip Python.

0. Mode Seluler vs. Mode Skrip

- *Mode Seluler:* Notebook mode seperti "05. Going Modular Part 1 (cell mode)" dijalankan seperti notebook biasa, di mana setiap sel dalam notebook dapat berisi kode atau markup.
- *Mode Skrip:* Notebook mode skrip seperti "05. Going Modular Part 2 (script mode)" mirip dengan mode seluler, tetapi banyak sel kode dapat diubah menjadi skrip Python. Meskipun tidak perlu membuat skrip Python melalui notebook, mode skrip digunakan untuk menunjukkan salah satu cara mengubah notebook menjadi skrip Python.

1. Mendapatkan Data

- Data diunduh dari GitHub menggunakan modul requests Python untuk mengunduh file .zip dan mengekstraknya.
- Data terdiri dari gambar pizza, steak, dan sushi dalam format klasifikasi gambar standar.
- Proses ini menghasilkan struktur direktori yang berisi folder "train" dan "test" untuk masing-masing kategori.

2. Membuat Datasets dan DataLoaders (data_setup.py)

- Kode pembuatan PyTorch Dataset dan DataLoader diubah menjadi fungsi bernama **create_dataloaders()** dan ditulis ke dalam file **data_setup.py** menggunakan **%%writefile**.
- Fungsi ini menerima jalur direktori untuk data pelatihan dan pengujian, transformasi, ukuran batch, dan jumlah pekerja untuk membuat DataLoaders.
- Fungsi ini mengembalikan tupel (train_dataloader, test_dataloader, class_names) yang berisi DataLoaders untuk pelatihan dan pengujian serta daftar nama kelas.

3. Membuat Model (model_builder.py)

- Model TinyVGG yang telah dibuat sebelumnya di notebook sebelumnya ditempatkan dalam file **model_builder.py** menggunakan **%%writefile**.
- File ini berisi definisi kelas **TinyVGG** yang menciptakan arsitektur TinyVGG dalam PyTorch.
- Sekarang, model TinyVGG dapat diimpor menggunakan **from going_modular import model_builder**.

4. Membuat Fungsi train_step() dan test_step() serta Fungsi train() untuk Menggabungkannya

- **train_step()**: Melatih model PyTorch untuk satu epoch dengan melakukan langkah-langkah pelatihan (forward pass, perhitungan kerugian, optimasi) pada DataLoader.
- **test_step()**: Mengevaluasi model PyTorch untuk satu epoch dengan melakukan forward pass pada dataset pengujian.
- **train()**: Menggabungkan **train_step()** dan **test_step()** untuk sejumlah epoch tertentu dan mengembalikan hasil dalam bentuk kamus.

5. Membuat Fungsi untuk Menyimpan Model (utils.py)

- **save_model()**: Menyimpan model PyTorch ke direktori target dengan fungsi utilitas. Menyimpan model ini dilakukan dengan menyertakan fungsi dalam file **utils.py** untuk menyimpan kode yang berulang.

6. Melatih, Mengevaluasi, dan Menyimpan Model (train.py)

- **train.py**: Berkumpulnya semua fungsionalitas sebelumnya (data_setup.py, engine.py, model_builder.py, utils.py) dalam satu skrip Python. Skrip ini digunakan untuk melatih model PyTorch dengan satu baris kode di baris perintah. Hyperparameter dan konfigurasi lainnya dapat diatur melalui argumen baris perintah atau secara langsung dalam skrip.