Zero to Mastery Learn PyTorch for Deep Learning Course Summary

04. PyTorch Custom Datasets

Nama : Al Ghifary Akmal Nasheeri

NIM : 1103201242

Kelas : TK-44-06

Modul ini fokus pada pembuatan dan penggunaan dataset kustom dalam PyTorch untuk pemodelan visi komputer. Pembahasan mencakup langkah-langkah untuk mengimpor dan mempersiapkan data, transformasi data, pembuatan dataset kustom, dan pengembangan model menggunakan dataset kustom tersebut.

1. Mendapatkan Data

- Langkah awal adalah memerlukan data.
- Dataset Food101 telah disiapkan, berisi 101.000 gambar dari 101 jenis makanan.
- Untuk latihan, digunakan subset dataset tersebut: 10% dari 3 kelas (pizza, steak, sushi).
- Data disiapkan dalam format klasifikasi gambar stkitar, terstruktur dalam direktori masing-masing kelas.
- Data diunduh dari GitHub dan disiapkan dengan Python.

2. Memahami Data (Persiapan Data)

- Melibatkan langkah penting sebelum membangun model, termasuk memahami struktur dan karakteristik data.
- Data terdiri dari gambar pizza, steak, dan sushi, dengan masing-masing kelas disimpan dalam direktori terpisah.
- Membuat fungsi untuk mengeksplorasi jumlah direktori dan gambar di setiap direktori.

3. Transformasi Data

- Langkah ini diperlukan untuk mempersiapkan data gambar sebelum digunakan oleh PyTorch.
- Menggunakan torchvision.transforms untuk mengubah gambar menjadi tensor dan memberikan beberapa transformasi seperti meresize dan flip horizontal.

- Dibuat fungsi untuk memvisualisasikan transformasi yang diterapkan pada beberapa gambar.

4. Memuat Data Gambar Menggunakan ImageFolder

- Menggunakan torchvision.datasets.ImageFolder untuk mengonversi data gambar menjadi Dataset PyTorch.
- Data transformasi diaplikasikan saat membuat Dataset.
- Dataset pelatihan dan pengujian diinisialisasi dengan menggunakan fungsi ImageFolder dan dicetak informasinya.
- Kelas-kelas dan panjang setiap Dataset diambil untuk referensi selanjutnya.
- Contoh gambar dan label dari Dataset ditampilkan.

5. Opsi 2 - Memuat Data Gambar dengan Dataset Kustom

Jika pembuat Dataset pra-dibangun seperti torchvision.datasets.ImageFolder() tidak ada, atau jika tidak ada yang sesuai dengan masalah khusus Kita, Kita dapat membuat Dataset kustom sendiri. Namun, ini memiliki kelebihan dan kekurangan.

Pro dari membuat Dataset kustom:

- Dapat membuat Dataset dari hampir segala sesuatu.
- Tidak terbatas pada fungsi Dataset pra-dibangun di PyTorch.

Kontra dari membuat Dataset kustom:

- Meskipun Kita dapat membuat Dataset dari hampir segala sesuatu, tidak berarti itu akan berhasil.
- Penggunaan Dataset kustom seringkali menghasilkan penulisan kode yang lebih banyak, yang dapat rentan terhadap kesalahan atau masalah kinerja.

Untuk melihatnya beraksi, kita akan bekerja menuju mereplikasi torchvision.datasets.ImageFolder() dengan mensubkelasinya menggunakan torch.utils.data.Dataset sebagai kelas dasar untuk semua Dataset di PyTorch.

Langkah-langkahnya melibatkan:

1. Mengimpor modul yang dibutuhkan.
2. Membuat fungsi pembantu untuk mendapatkan nama kelas.
3. Membuat kelas Dataset kustom kita sendiri untuk mereplikasi ImageFolder.
4. Membuat transformasi data untuk persiapan gambar.

5.1. Membuat Fungsi Pembantu untuk Mendapatkan Nama Kelas

Dalam langkah ini, kita membuat fungsi find_classes() yang dapat membuat daftar nama kelas dan kamus nama kelas dan indeks mereka berdasarkan path direktori target.

5.2. Membuat Dataset Kustom untuk Mereplikasi ImageFolder

Langkah ini melibatkan pembuatan kelas Dataset kustom, ImageFolderCustom, untuk mereplikasi fungsionalitas torchvision.datasets.ImageFolder(). Ini melibatkan mensubkelas torch.utils.data.Dataset, menginisialisasi dengan parameter target direktori dan transformasi opsional, dan mengimplementasikan metode **len** dan **getitem**.

5.3. Membuat Fungsi untuk Menampilkan Gambar Acak

Kita membuat fungsi display_random_images() untuk memvisualisasikan gambar-gambar dalam Dataset kita. Fungsi ini mengambil Dataset, jumlah gambar yang akan ditampilkan, kelas (opsional), dan seed acak untuk membuat gambar secara reproduktif.

5.4. Mengubah Gambar yang Dimuat Sendiri Menjadi DataLoader

Setelah kita membuat Dataset kustom kita, langkah selanjutnya adalah mengonversinya menjadi DataLoader menggunakan torch.utils.data.DataLoader(). DataLoader memungkinkan kita untuk memuat data dalam bentuk batch dan menyederhanakan proses pelatihan model.

6. Bentuk Lain dari Transformasi (Augmentasi Data)

Transformasi data dapat digunakan untuk mengubah gambar dengan berbagai cara, termasuk augmentasi data untuk meningkatkan keberagaman set pelatihan. Salah satu bentuk augmentasi data yang dijelaskan adalah transforms.TrivialAugmentWide() yang memanfaatkan kekuatan randomness untuk meningkatkan kinerja model.

Terakhir, kita menciptakan transformasi data untuk pelatihan dan pengujian yang mencakup augmentasi data pada set pelatihan dan konversi gambar menjadi tensor. Transformasi data ini digunakan untuk membuat DataLoader dari Dataset kustom kita.

Sekarang, kita telah memahami dan menerapkan langkah-langkah ini dalam konteks membuat Dataset kustom dan melakukan augmentasi data pada gambar.

7. Model 0: TinyVGG tanpa augmentasi data

7.1. Membuat Transformasi dan Memuat Data untuk Model 0

- Membuat transformasi sederhana, misalnya, meresize gambar menjadi (64, 64) dan mengubahnya menjadi tensor.
- Memuat data, mengonversi folder pelatihan dan pengujian ke dalam Dataset menggunakan torchvision.datasets.ImageFolder(), kemudian menjadi DataLoader menggunakan torch.utils.data.DataLoader().

7.2. Membuat Kelas Model TinyVGG

- Membuat model TinyVGG dengan lapisan konvolusi dan penggkitaan.
- Model ini digunakan untuk tugas klasifikasi gambar.

7.3. Melakukan Forward Pass pada Satu Gambar (Untuk Pengujian Model)

- Melakukan forward pass pada satu gambar untuk menguji model.
- Menggunakan torch.softmax() untuk mengonversi logits menjadi probabilitas prediksi.

7.4. Menggunakan torchinfo untuk Mendapatkan Gambaran Bentuk yang Dilalui oleh Model

- Menggunakan torchinfo untuk mendapatkan informasi rinci tentang model, termasuk jumlah parameter dan ukuran estimasi total.

7.5. Membuat Fungsi Loop Pelatihan dan Pengujian

- Membuat fungsi train_step() untuk pelatihan model pada DataLoader.
- Membuat fungsi test_step() untuk mengevaluasi model pada DataLoader.
- Membuat fungsi train() yang menggabungkan train_step() dan test_step() untuk pelatihan model selama beberapa epoch.

8. Evaluasi Model 0 dan Strategi untuk Memperbaikinya:

- Model 0 memberikan hasil yang kurang memuaskan.
- Beberapa strategi untuk meningkatkan kinerja model melibatkan penanganan overfitting dan underfitting.
- Strategi untuk mengatasi overfitting:
    - Mendapatkan lebih banyak data.
    - Menyederhanakan model.
    - Menggunakan augmentasi data.
    - Menggunakan transfer learning.
    - Menggunakan dropout layers, learning rate decay, dan early stopping.
- Strategi untuk mengatasi underfitting:
    - Menambahkan lebih banyak lapisan atau unit ke model.
    - Penyesuaian learning rate.
    - Menggunakan transfer learning.
    - Melatih model lebih lama.

- Mengurangi regularisasi.

8.1. Cara Mengatasi Overfitting:

- Menggunakan lebih banyak data.
- Menyederhanakan model.
- Menggunakan augmentasi data.
- Menggunakan transfer learning.
- Menggunakan lapisan dropout.
- Menggunakan penurunan tingkat pembelajaran.
- Menggunakan early stopping.

8.2. Cara Mengatasi Underfitting:

- Menambahkan lebih banyak lapisan/unit ke model.
- Menyesuaikan tingkat pembelajaran.
- Menggunakan transfer learning.
- Melatih model untuk lebih lama.
- Mengurangi penggunaan regularisasi.

8.3. Keseimbangan Antara Overfitting dan Underfitting:

- Tidak ada metode yang sempurna, dan keseimbangan antara overfitting dan underfitting adalah tantangan utama dalam pembelajaran mesin.
- Transfer learning sering digunakan sebagai solusi karena dapat mengambil model yang sudah bekerja di ruang masalah serupa dan menggunakannya pada dataset yang berbeda.

9. Model 1 - TinyVGG dengan Augmentasi Data

9.1. Transformasi Data dengan Augmentasi

- Membuat transformasi pelatihan dengan TrivialAugmentWide, meresize gambar, dan mengubahnya menjadi tensor.
- Membuat transformasi pengujian tanpa augmentasi data.

9.2. Membuat Dataset dan DataLoader

- Menggunakan **torchvision.datasets.ImageFolder()** untuk mengonversi folder gambar menjadi Dataset.
- Membuat DataLoader dengan batch_size=32 dan num_workers sesuai jumlah CPU pada mesin.

9.3. Membangun dan Melatih Model 1 (TinyVGG)

- Membuat model TinyVGG (model_1) dan mengirimkannya ke perangkat target.
- Melatih model dengan menggunakan fungsi pelatihan sebelumnya dengan parameter yang sesuai.
- Menggunakan CrossEntropyLoss sebagai fungsi kerugian dan Adam sebagai pengoptimal dengan lr=0.001.
- Melakukan pelatihan selama 5 epoch dan mencetak hasil pelatihan.

9.4. Plot Kurva Kerugian Model 1

- Menampilkan kurva kerugian model_1 menggunakan fungsi plot_loss_curves.

10. Membandingkan Hasil Model

1. **Mengubah Hasil Model ke DataFrame Pkitas**
    - Mengonversi hasil model_0 dan model_1 menjadi DataFrames.
2. **Membandingkan Hasil Model dengan Grafik**
    - Menggunakan matplotlib, membuat grafik untuk membandingkan model_0 dan model_1 dalam hal kerugian dan akurasi pada set pelatihan dan pengujian.

11. Membuat Prediksi pada Gambar Kustom

Langkah ini membahas cara membuat prediksi pada gambar kustom setelah melatih model. Dalam hal ini, model telah dilatih pada dataset gambar pizza, steak, dan sushi. Langkah-langkahnya adalah sebagai berikut:

11.1. Memuat gambar kustom dengan PyTorch

- Mendownload gambar kustom dari URL tertentu menggunakan Python's requests module.
- Membaca gambar dengan torchvision.io.read_image(), menghasilkan tensor uint8.
- Memastikan bahwa format gambar tersebut kompatibel dengan model yang telah dilatih.

11.2. Memprediksi gambar kustom dengan model PyTorch yang dilatih

- Melakukan transformasi pada gambar kustom agar memiliki format yang sama dengan data yang digunakan untuk melatih model.
- Menggunakan torchvision.transforms.Resize() untuk mengubah ukuran gambar.
- Membuat prediksi pada gambar kustom menggunakan model yang telah dilatih, dengan memperhatikan format, perangkat (device), dan ukuran gambar yang sesuai.

- Mengonversi output model dari logits ke probabilitas prediksi.
- Menampilkan gambar bersama dengan prediksi dan probabilitas prediksi.

11.3. Membuat fungsi untuk prediksi dan plot gambar

- Membuat fungsi yang menggabungkan langkah-langkah sebelumnya agar lebih mudah digunakan.
- Fungsi ini menerima model, path gambar target, nama kelas (opsional), transformasi (opsional), dan perangkat sebagai input.
- Fungsi menghasilkan prediksi pada gambar target dan menampilkannya bersama dengan prediksi dan probabilitas prediksi.