Zero to Mastery Learn PyTorch for Deep Learning Course Summary

06. PyTorch Transfer Learning

Nama : Al Ghifary Akmal Nasheeri

NIM : 1103201242

Kelas : TK-44-06

Modul ini memperkenalkan penggunaan teknik transfer learning dalam deep learning dengan PyTorch.

- **Pentingnya Transfer Learning:**
  - Beberapa model yang telah dibangun sebelumnya memiliki kinerja yang buruk.
  - Transfer learning memungkinkan penggunaan pola (bobot) yang telah dipelajari oleh model lain untuk masalah kita sendiri.
  - Dengan transfer learning, kita dapat mengambil pola dari model computer vision pada dataset ImageNet dan menggunakannya untuk FoodVision Mini kita.
- **Apa itu Transfer Learning:**
  - Transfer learning memungkinkan kita menggunakan pola yang telah dipelajari oleh model pada masalah lain untuk masalah kita sendiri.
- **Manfaat Transfer Learning:**
  - Dapat memanfaatkan model yang sudah ada yang terbukti berhasil pada masalah serupa.
  - Dapat memanfaatkan model yang sudah memiliki pola pada data serupa dengan data kita sendiri, seringkali menghasilkan hasil yang baik dengan data kustom yang lebih sedikit.
- **Aplikasi Transfer Learning pada FoodVision Mini:**
  - Transfer learning akan diuji pada FoodVision Mini dengan menggunakan model computer vision yang telah dilatih pada ImageNet.

0. Persiapan Awal

- Langkah ini mencakup langkah-langkah untuk mengimpor/mengunduh modul yang diperlukan.
- Kode menggunakan beberapa skrip Python yang telah dibuat sebelumnya, seperti data_setup.py dan engine.py dari Modul 05: PyTorch Going Modular.

- Mengunduh direktori going_modular dari repositori pytorch-deep-learning jika belum ada.
- Menginstal torchinfo jika belum tersedia untuk memberikan representasi visual dari model kita.
- Menggunakan versi nightly dari torch dan torchvision (versi v0.13+) untuk menjalankan notebook ini.
- Setelah itu, dilakukan import dan setup reguler serta menentukan perangkat (device) yang akan digunakan (cuda jika tersedia, jika tidak, menggunakan cpu).

1. Dapatkan Data

- Sebelum menggunakan transfer learning, kita membutuhkan dataset.
- Kode ini mendownload dataset pizza_steak_sushi.zip dari GitHub kursus dan mengekstraknya jika belum ada.
- Dataset ini berisi gambar pizza, steak, dan sushi.

2. Buat Dataset dan DataLoader

- Karena sudah mengunduh direktori going_modular, kita dapat menggunakan skrip data_setup.py untuk menyiapkan DataLoaders.
- Terlepas dari itu, karena akan menggunakan model terlatih dari torchvision.models, perlu menyiapkan transformasi khusus untuk gambar.

2.1. Manual Creation

Membuat pipeline transformasi secara manual dengan menggabungkan beberapa transformasi menggunakan **transforms.Compose**. Transformasi ini mencakup resize, konversi nilai piksel ke rentang [0, 1], dan normalisasi dengan mean dan std tertentu.

2.2. Auto Creation

Menggunakan fitur transformasi otomatis yang diperkenalkan di torchvision v0.13+. Fitur ini memungkinkan kita mendapatkan transformasi yang sesuai dengan model terlatih yang dipilih tanpa harus membuatnya secara manual. Sebagai contoh, transformasi untuk model EfficientNet_B0 diambil dari bobot terlatih tersebut.

Selanjutnya, DataLoaders dibuat dengan menggunakan fungsi **create_dataloaders** dari data_setup.py, baik dengan transformasi manual maupun otomatis.

3. Mendapatkan Model yang Sudah Dipretraining

Dalam langkah ini, kita akan menggunakan model yang sudah dipretraining untuk menyelesaikan tugas klasifikasi gambar makanan dengan FoodVision Mini. Meskipun kita telah membangun jaringan saraf PyTorch dari awal dalam beberapa notebook sebelumnya, model-model tersebut belum memberikan performa yang sesuai dengan harapan.

Oleh karena itu, kita akan menerapkan transfer learning. Ide dasar dari transfer learning adalah mengambil model yang sudah baik kinerjanya dalam ruang masalah yang mirip dengan milik kita, lalu menyesuaikannya dengan kasus penggunaan kita.

Dalam konteks ini, karena kita bekerja pada masalah computer vision (klasifikasi gambar dengan FoodVision Mini), kita dapat menemukan model klasifikasi yang sudah dipretraining di torchvision.models.

3.1 Pemilihan Model yang Sudah Dipretraining

Pemilihan model yang sesuai tergantung pada masalah dan perangkat yang Kita gunakan. Umumnya, angka yang lebih tinggi pada nama model (contoh: **efficientnet_b0()** hingga **efficientnet_b7()**) menunjukkan performa yang lebih baik tetapi dengan ukuran model yang lebih besar.

Sebagai contoh, jika Kita ingin menjalankan model pada perangkat mobile, Kita perlu memperhatikan sumber daya komputasi yang terbatas pada perangkat tersebut dan mungkin memilih model yang lebih kecil.

Namun, jika Kita memiliki daya komputasi tanpa batas, Kita dapat memilih model yang lebih besar. Pemahaman tentang pertukaran performa vs. kecepatan vs. ukuran ini akan berkembang dengan waktu dan praktik.

3.2 Persiapan Model yang Sudah Dipretraining

Dalam contoh ini, kita akan menggunakan model **efficientnet_b0()** yang sudah dipretraining. Model ini memiliki tiga bagian utama: **features** (kumpulan lapisan konvolusional dan lapisan aktivasi lainnya untuk mempelajari representasi dasar data visual), **avgpool** (mengambil rata-rata keluaran dari lapisan fitur dan mengubahnya menjadi vektor fitur), dan **classifier** (mengubah vektor fitur menjadi vektor dengan dimensi sesuai dengan jumlah kelas keluaran yang dibutuhkan).

Untuk memasukkan berat ImageNet yang sudah dipretraining, kita dapat menggunakan kode yang sama dengan yang kita gunakan untuk membuat transformasi.

Penting untuk dicatat bahwa arsitektur model dapat berubah seiring waktu dengan rilis penelitian baru. Oleh karena itu, disarankan untuk bereksperimen dengan beberapa arsitektur dan menyesuaikannya dengan masalah Kita.

3.3 Ringkasan Model dengan torchinfo.summary()

Untuk memahami lebih lanjut tentang model, kita dapat menggunakan metode **torchinfo.summary()**. Ini membutuhkan model, ukuran input, kolom informasi yang diinginkan, lebar kolom, dan pengaturan baris.

3.4 Membekukan Bagian Dasar Model dan Mengubah Layer Output

Transfer learning melibatkan membekukan beberapa lapisan dasar dari model yang sudah dipretraining (biasanya bagian **features**) dan kemudian menyesuaikan lapisan keluaran (juga disebut lapisan kepala/klasifikasi) agar sesuai dengan kebutuhan kita.

Dalam contoh ini, kita membekukan semua lapisan dasar dengan mengatur **requires_grad=False** untuk setiap parameter dalam **model.features**. Selanjutnya, kita mengubah lapisan keluaran atau bagian klasifikasi model untuk sesuai dengan jumlah kelas yang kita miliki.

4. Melatih Model:

- Model yang telah di-pretrain sekarang semi-frozen dan memiliki klasifikasi yang disesuaikan. Kita akan menerapkan transfer learning.
- Untuk melatih, kita perlu membuat fungsi loss dan optimizer.
- Menggunakan **nn.CrossEntropyLoss()** untuk fungsi loss karena ini adalah masalah klasifikasi multi-kelas.
- Menggunakan **torch.optim.Adam()** sebagai optimizer dengan lr=0.001.
- Melatih model menggunakan fungsi **train()** yang telah didefinisikan sebelumnya.
- Melatih hanya parameter classifier, sementara parameter lain sudah difreeze.
- Menetapkan random seeds dan menghitung waktu total pelatihan.

5. Evaluasi Model dengan Plotting Kurva Loss:

- Model tampaknya berperforma dengan baik. Untuk mengevaluasi lebih lanjut, kita akan plotting loss curves dari hasil pelatihan.
- Menggunakan fungsi **plot_loss_curves()** yang dibuat sebelumnya.
- Loss curves menunjukkan bahwa baik loss pada dataset pelatihan maupun uji mengarah ke arah yang benar, demikian juga dengan nilai akurasi.

6. Prediksi pada Gambar dari Dataset Uji:

- Model telah melatih dengan baik secara kuantitatif, sekarang mari kita lihat secara kualitatif dengan membuat prediksi pada gambar dari dataset uji.

- Membuat fungsi **pred_and_plot_image()** untuk membuat prediksi pada gambar uji.
- Fungsi ini memastikan bahwa gambar yang akan diprediksi memiliki format yang sesuai dengan gambar yang digunakan untuk melatih model.
- Menggunakan beberapa gambar acak dari dataset uji dan memvisualisasikan prediksinya
- Hasil prediksi menunjukkan kualitas yang jauh lebih baik dibandingkan dengan model TinyVGG sebelumnya.

6.1 **Prediksi pada Gambar Kustom:**

- Melakukan prediksi pada gambar kustom ("pizza-dad.jpeg").
- Menggunakan fungsi **pred_and_plot_image()** untuk membuat prediksi pada gambar tersebut.