Zero to Mastery Learn PyTorch for Deep Learning Course Summary

07. PyTorch Experiment Tracking

Nama : Al Ghifary Akmal Nasheeri

NIM : 1103201242

Kelas : TK-44-06

Modul 7 membahas tentang Pelacakan Eksperimen PyTorch. Pada perjalanan pembuatan FoodVision Mini (model klasifikasi gambar untuk mengklasifikasikan gambar pizza, steak, atau sushi), sejumlah model telah dilatih. Sejauh ini, pelacakan dilakukan melalui kamus Python atau perbandingan metrik selama pelatihan. Namun, jika kita ingin menjalankan lusinan model yang berbeda sekaligus, diperlukan cara yang lebih baik, yaitu pelacakan eksperimen.

Pentingnya pelacakan eksperimen dalam machine learning karena eksperimen sangat penting. Untuk mencari tahu apa yang berhasil dan apa yang tidak, kita perlu melacak hasil eksperimen yang berbeda.

1. **Apa itu Pelacakan Eksperimen:**
    - Machine learning dan deep learning bersifat eksperimental.
    - Diperlukan untuk menciptakan berbagai model dan melacak hasil kombinasi berbagai data, arsitektur model, dan metode pelatihan.
    - Pelacakan eksperimen membantu menentukan apa yang berhasil dan apa yang tidak.
2. **Mengapa Melacak Eksperimen:**
    - Jika hanya menjalankan beberapa model, melacak hasil dalam print out dan beberapa kamus mungkin cukup.
    - Namun, ketika jumlah eksperimen meningkat, cara ini mungkin sulit dijaga.
    - Pelacakan eksperimen menjadi sangat penting karena jumlah eksperimen yang berbeda dapat menjadi sulit dijaga.
3. **Berbagai Cara Melacak Eksperimen Machine Learning:**
    - Ada berbagai cara melacak eksperimen machine learning, dan beberapa di antaranya mencakup penggunaan kamus Python, file CSV, print out, TensorBoard, Weights & Biases Experiment Tracking, dan MLFlow.
    - Setiap metode memiliki kelebihan dan kekurangan serta biaya yang berbeda.

0. Persiapan

1. Mengunduh modul-modul yang diperlukan dengan memeriksa dan menginstal versi PyTorch dan torchvision yang sesuai.
2. Menyiapkan kode untuk penggunaan perangkat agnostik (device-agnostic) dengan menentukan perangkat (CPU atau GPU).
3. Membuat fungsi set_seeds() untuk mengatur biji acak untuk operasi-operasi torch.

1. Mendapatkan Data

1. Mengunduh dataset "pizza_steak_sushi" yang digunakan untuk eksperimen FoodVision Mini.
2. Membuat fungsi download_data() untuk mengunduh dan mengekstrak dataset jika belum ada.
3. Mengonfigurasi path untuk dataset.

2. Membuat Datasets dan DataLoaders

1. Mengonfigurasi transformasi untuk mengubah gambar menjadi tensor dan normalisasi sesuai format ImageNet.
2. Membuat DataLoader menggunakan transformasi yang dibuat secara manual.
3. Membuat DataLoader menggunakan transformasi yang dibuat secara otomatis dari pretrained weights.

2.1. Membuat DataLoaders dengan Transformasi yang Dibuat Secara Manual

1. Mengonfigurasi path direktori untuk data pelatihan dan pengujian.
2. Mengatur normalisasi ImageNet sebagai bagian dari transformasi manual.
3. Membuat pipeline transformasi manual menggunakan torchvision.transforms.Compose().
4. Membuat DataLoader menggunakan fungsi create_dataloaders() dari modul data_setup.

2.2. Membuat DataLoaders dengan Transformasi yang Dibuat Secara Otomatis

1. Mengonfigurasi path direktori untuk data pelatihan dan pengujian.
2. Memilih pretrained weights (contoh: EfficientNet_B0) dari torchvision.models.

3. Menggunakan weights tersebut untuk mendapatkan transformasi otomatis dengan memanggil metode transforms().
4. Membuat DataLoader menggunakan fungsi create_dataloaders() dari modul data_setup.

3. Mendapatkan Model Pretrained

1. Mengunduh pretrained weights untuk model EfficientNet_B0 dari torchvision.
2. Membuat model dengan weights tersebut, kemudian membekukan lapisan dasar (base layers) dan mengganti classifier head untuk sesuaikan dengan jumlah kelas.

4. Melatih Model dan Melacak Hasil

1. Menentukan fungsi loss (CrossEntropyLoss) dan optimizer (Adam) untuk pelatihan model.
2. Mengonfigurasi SummaryWriter untuk melacak hasil eksperimen menggunakan TensorBoard.
3. Menggunakan fungsi train() untuk melatih model dan melacak hasil menggunakan SummaryWriter.

5. Memantau Hasil Model pada TensorBoard

1. Menggunakan kelas SummaryWriter() untuk menyimpan hasil model dalam format TensorBoard secara default di direktori "runs/".
2. TensorBoard adalah program visualisasi yang dibuat oleh tim TensorFlow untuk melihat dan memeriksa informasi tentang model dan data.
3. Memungkinkan visualisasi hasil secara interaktif selama dan setelah pelatihan.
4. Cara mengakses TensorBoard:
    - Di VS Code: Tekan SHIFT + CMD + P, cari perintah "Python: Launch TensorBoard".
    - Di Jupyter dan Colab Notebooks: Pastikan TensorBoard terinstal, muat dengan %load_ext tensorboard, dan lihat hasil dengan %tensorboard --logdir DIR_WITH_LOGS.
    - Juga dapat mengunggah eksperimen ke tensorboard.dev untuk berbagi secara publik.

6. Membuat Fungsi Pembantu untuk Membangun Instance SummaryWriter()

1. Kelas SummaryWriter() mencatat informasi ke direktori yang ditentukan oleh parameter log_dir.
2. Membuat fungsi bantu create_writer() untuk membuat instance SummaryWriter() dengan log_dir yang disesuaikan untuk setiap eksperimen.
3. Log_dir mengandung timestamp, nama eksperimen, nama model, dan informasi tambahan (opsional).
4. Dapat menyesuaikan direktori log untuk melacak berbagai detail seperti tanggal eksperimen, nama eksperimen, nama model, dan informasi tambahan.
5. Memperbarui fungsi pelatihan (train()) untuk menerima parameter writer sehingga setiap eksperimen dapat menggunakan instance SummaryWriter() yang berbeda.

6.1. Memperbarui Fungsi train() untuk Menyertakan Parameter Writer

1. Menambahkan parameter writer ke fungsi train() untuk memberikan kemampuan menggunakan instance SummaryWriter() yang berbeda untuk setiap eksperimen.
2. Saat memanggil train(), dapat memasukkan writer yang berbeda untuk setiap eksperimen, yang menghasilkan direktori log yang berbeda.
3. Fungsi train() kini dapat aktif memperbarui instance SummaryWriter() yang digunakan setiap kali dipanggil.

7. Menyiapkan Serangkaian Eksperimen Pemodelan

Dalam langkah ini, kita melakukan serangkaian eksperimen pemodelan untuk meningkatkan model FoodVision Mini tanpa membuatnya terlalu besar. Pendekatan ini melibatkan mencoba berbagai kombinasi data, model, dan jumlah epoch. Berikut adalah langkah-langkah utamanya:

7.1. Eksperimen Pemodelan:

- Pertanyaan besar dalam pembelajaran mesin adalah jenis eksperimen apa yang sebaiknya dijalankan.
- Tidak ada batasan eksperimen yang dapat dijalankan, sehingga pembelajaran mesin menjadi menarik dan menakutkan pada saat yang sama.
- Pada langkah ini, Kita perlu mengenakan mantel ilmuwan dan mengikuti moto praktisi pembelajaran mesin: eksperimen, eksperimen, eksperimen!
- Setiap hiperparameter menjadi titik awal untuk eksperimen yang berbeda.

Beberapa eksperimen yang dapat dijalankan:

- Mengubah jumlah epoch.
- Mengubah jumlah lapisan/unit tersembunyi.

- Mengubah jumlah data.
- Mengubah tingkat pembelajaran.
- Mencoba augmentasi data yang berbeda.
- Memilih arsitektur model yang berbeda.

Dengan latihan dan menjalankan banyak eksperimen yang berbeda, Kita akan mulai membangun intuisi tentang apa yang mungkin membantu model Kita.

7.2. Eksperimen yang Akan Dijalankan:

- Tujuan adalah meningkatkan model FoodVision Mini tanpa membuatnya terlalu besar.
- Model ideal mencapai tingkat akurasi set uji tinggi (90%+), tetapi tidak memakan waktu terlalu lama untuk melatih/membuat inferensi.
- Eksperimen ini mencakup kombinasi:
    - Jumlah data yang berbeda (10% dari Pizza, Steak, Sushi vs. 20%).
    - Model yang berbeda (EfficientNetB0 vs. EfficientNetB2).
    - Waktu pelatihan yang berbeda (5 epoch vs. 10 epoch).
- Eksperimen dilakukan secara bertahap meningkatkan jumlah data, ukuran model, dan lama pelatihan.

7.3. Unduh Dataset yang Berbeda:

- Sebelum menjalankan serangkaian eksperimen, pastikan dataset sudah siap.
- Diperlukan dua bentuk set pelatihan:
    - Set pelatihan dengan 10% data gambar Pizza, Steak, Sushi dari Food101 (sudah dibuat sebelumnya).
    - Set pelatihan dengan 20% data gambar Pizza, Steak, Sushi dari Food101.
- Kedua dataset ini diunduh dari GitHub.

7.4. Transformasi Dataset dan Pembuatan DataLoader:

- Transformasi dibuat untuk menyiapkan gambar untuk model.
- DataLoader dibuat dengan batch size 32 untuk semua eksperimen.

7.5. Membuat Model Pengambil Fitur:

- Dua model pengambil fitur dibuat: EfficientNetB0 dan EfficientNetB2.
- Lapisan dasar (fitur) dibekukan, dan lapisan pengklasifikasi diubah sesuai dengan masalah klasifikasi gambar.
- EfficientNetB2 memerlukan penyesuaian pada jumlah fitur input akhir.

7.6. Membuat Eksperimen dan Menyiapkan Kode Pelatihan:

- Dua daftar dan satu kamus dibuat untuk jumlah epoch yang akan diuji, model yang akan diuji, dan DataLoader yang berbeda.
- Eksperimen dijalankan dengan mengubah model, jumlah epoch, dan DataLoader.
- Model yang dilatih disimpan setiap selesai eksperimen untuk kemudian dapat digunakan untuk prediksi.

8. Meninjau Eksperimen pada TensorBoard

- Menggunakan TensorBoard untuk visualisasi hasil eksperimen.
- Dapat dilihat bahwa model EffNetB0 yang dilatih selama 10 epoch dengan 20% data mencapai kerugian uji terendah.
- Visualisasi hasil eksperimen dapat diunggah ke tensorboard.dev untuk dibagikan secara publik.

9. Memuat Model Terbaik dan Melakukan Prediksi

- Menganalisis log TensorBoard dan menemukan bahwa eksperimen kedelapan mencapai hasil terbaik secara keseluruhan (akurasi uji tertinggi, kerugian uji kedua terendah).
- Model terbaik adalah EffNetB2 dengan parameter gkita, 20% data pelatihan pizza, steak, sushi, dan dilatih selama 10 epoch.
- Meskipun hasilnya tidak jauh lebih baik dari model lain, model yang sama pada data yang sama mencapai hasil serupa dalam setengah waktu pelatihan.
- Memuat model terbaik yang telah disimpan dan memeriksa ukuran file model.
- Melakukan prediksi pada gambar uji yang tidak pernah dilihat sebelumnya.

9.1. Memprediksi pada Gambar Kustom dengan Model Terbaik

- Menampilkan prediksi pada gambar kustom (gambar "pizza dad") menggunakan model terbaik.
- Model terbaik berhasil memprediksi "pizza" dengan tingkat kepercayaan yang tinggi (0.978).