Nama : Andhika Aria Pratama N

NIM : 1103202121

06 PyTorch Transfer Learning

- Menyiapkan pengaturan

Menyiapkan beberapa library yang akan dipakai.

- Mengunduh data

Mengunduh data berupa file zip yang berisikan beberapa jenis makanan.

- Membuat dataset dan dataloaders

- Membuat transformasi untuk torchvision.models (pembuatan manual)

Pada tahap ini, sebuah pipeline transformasi manual dibuat untuk digunakan dalam torchvision versi kurang dari 0.13. Transformasi ini mencakup pengubah bentuk gambar menjadi 224x224 piksel, konversi nilai piksel ke rentang antara 0 dan 1, serta normalisasi dengan mean dan standard deviation yang telah ditentukan.

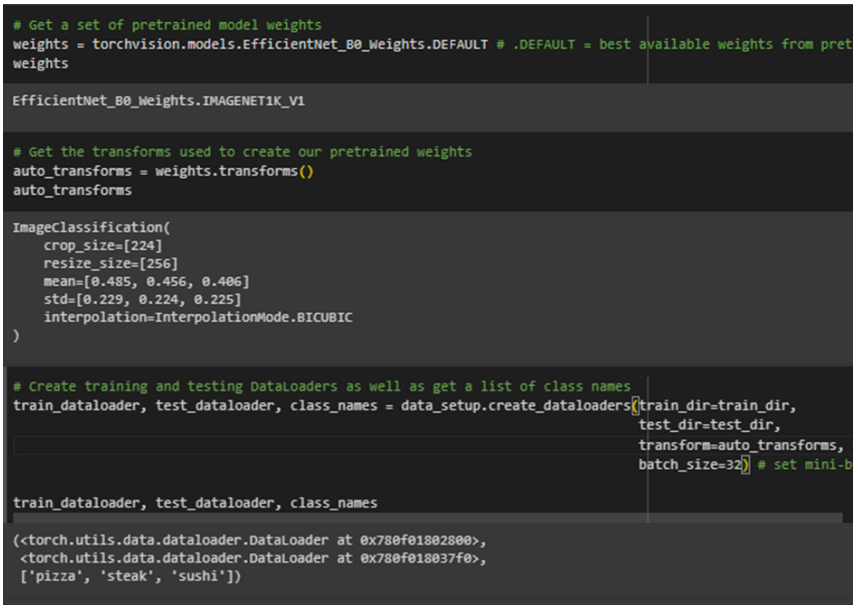
Selanjutnya, DataLoaders untuk pelatihan dan pengujian dibuat menggunakan modul `data_setup`. DataLoader tersebut menggunakan transformasi manual yang telah dibuat, dan ukuran batch diatur sebesar 32. Hasilnya, setelah menjalankan kode ini, adalah objek DataLoaders untuk pelatihan dan pengujian, serta daftar nama kelas yang diperoleh dari direktori data (pada contoh ini, kelas 'pizza', 'steak', dan 'sushi').

- Membuat transformasi untuk torchvision.models (pembuatan otomatis)

Pada tahap ini, bobot model EfficientNet-B0 yang telah dilatih sebelumnya di ImageNet diambil dengan menggunakan torchvision. Kemudian, transformasi otomatis yang digunakan untuk membuat bobot tersebut diperoleh. Transformasi ini mencakup pemangkasan gambar ke ukuran 224x224 piksel, penyesuaian ukuran ke 256x256 piksel, normalisasi menggunakan mean dan standard deviation yang telah ditentukan, serta interpolasi menggunakan metode bicubic.

Selanjutnya, DataLoaders untuk pelatihan dan pengujian dibuat menggunakan modul `data_setup`. DataLoader tersebut menggunakan transformasi otomatis yang telah diperoleh sebelumnya, dan ukuran batch diatur sebesar 32. Hasilnya adalah objek DataLoaders untuk pelatihan dan pengujian, serta daftar nama kelas yang diperoleh dari direktori data (dalam contoh ini, kelas 'pizza', 'steak', dan 'sushi').

- Mendapatkan model terlatih



Pada tahap ini, model EfficientNet-B0 dikonfigurasi dengan menggunakan bobot yang telah dilatih sebelumnya. Dengan menggunakan torchvision versi 0.13+, pengaturan bobot dilakukan dengan mendapatkan bobot dari `EfficientNet_B0_Weights.DEFAULT` dan mengirimkan model ke perangkat target (misalnya, GPU). Untuk memahami struktur dan parameter model, ringkasan model diberikan menggunakan fungsi `summary` dari torchinfo, menampilkan informasi seperti ukuran input, ukuran output, jumlah parameter, dan apakah parameter dapat diubah selama pelatihan.

Selanjutnya, lapisan "features" dari model (ekstraktor fitur) dibekukan dengan mengatur `requires_grad` menjadi `False`. Manual seed ditetapkan untuk memastikan reproduktibilitas. Panjang dari `class_names` diambil untuk menentukan jumlah unit output yang sesuai dengan jumlah kelas. Kemudian, lapisan klasifikasi model diperbarui dengan menambahkan lapisan dropout dan lapisan linear sesuai dengan jumlah kelas.

Ringkasan model kembali ditampilkan setelah membekukan lapisan fitur dan mengganti lapisan klasifikasi, memberikan gambaran struktur akhir model yang akan digunakan untuk pelatihan klasifikasi gambar.

- Pelatihan model

```python
# Define loss and optimizer
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

# Set the random seeds
torch.manual_seed(42)
torch.cuda.manual_seed(42)

# Start the timer
from timeit import default_timer as timer
start_time = timer()

# Setup training and save the results
results = engine.train(model=model,
                       train_dataloader=train_dataloader,
                       test_dataloader=test_dataloader,
                       optimizer=optimizer,
                       loss_fn=loss_fn,
                       epochs=5,
                       device=device)

# End the timer and print out how long it took
end_time = timer()
print(f"[INFO] Total training time: {end_time-start_time:.3f} seconds")
```

```
100% 5/5 [00:20<00:00, 4.21s/it]
Epoch: 1 | train_loss: 1.0924 | train_acc: 0.3984 | test_loss: 0.9133 | test_acc: 0.5398
Epoch: 2 | train_loss: 0.8717 | train_acc: 0.7773 | test_loss: 0.7912 | test_acc: 0.8153
Epoch: 3 | train_loss: 0.7648 | train_acc: 0.7930 | test_loss: 0.7463 | test_acc: 0.8561
Epoch: 4 | train_loss: 0.7109 | train_acc: 0.7539 | test_loss: 0.6373 | test_acc: 0.8655
Epoch: 5 | train_loss: 0.6255 | train_acc: 0.7852 | test_loss: 0.6261 | test_acc: 0.8561
[INFO] Total training time: 20.369 seconds
```

Pada tahap ini, fungsi `train` dari modul `engine` digunakan untuk melatih model EfficientNet-B0 dengan optimasi Adam dan fungsi kerugian CrossEntropy. Pelatihan dilakukan selama 5 epoch menggunakan data latih dari `train_dataloader` dan data uji dari `test_dataloader`. Selama pelatihan, metrik seperti loss dan akurasi dicetak untuk setiap epoch.

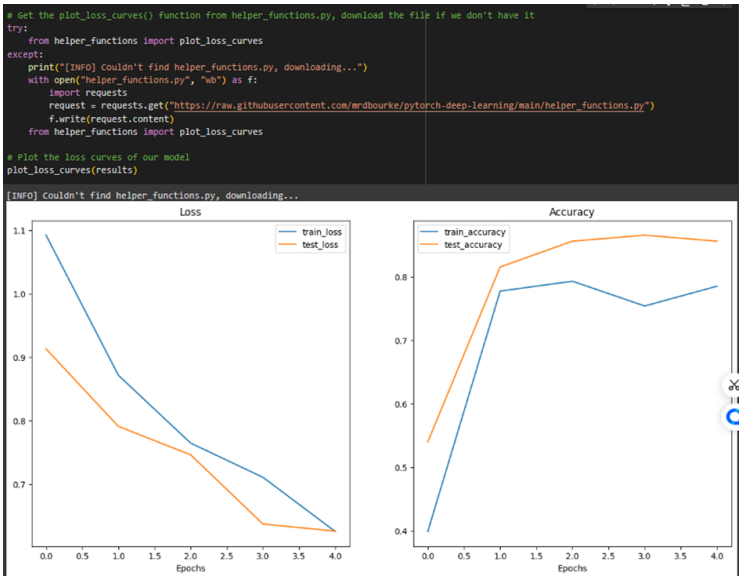
Hasil pelatihan menunjukkan penurunan loss dan peningkatan akurasi pada setiap epoch, menunjukkan bahwa model secara progresif belajar dari data latih. Waktu total pelatihan adalah sekitar 20.369 detik. Performa model didapatkan hasil train accuracy sebesar 78% dan test accuracy sebesar 85%.

- Evaluasi model dengan memplot kurva kerugian

```python
# Get the plot_loss_curves() function from helper_functions.py, download the file if we don't have it
try:
    from helper_functions import plot_loss_curves
except:
    print("[INFO] Couldn't find helper_functions.py, downloading...")
    with open("helper_functions.py", "wb") as f:
        import requests
        request = requests.get("https://raw.githubusercontent.com/mrdbourke/pytorch-deep-learning/main/helper_functions.py")
        f.write(request.content)
    from helper_functions import plot_loss_curves

# Plot the loss curves of our model
plot_loss_curves(results)
```

```
[INFO] Couldn't find helper_functions.py, downloading...
```



Dihasilkan dari bentuk graph pelatihan tiap epoch pada grap loss memiliki garis yang semakin menurun. Pelatihan pada graph loss memiliki nilai semakin lama semakin turun baik pada train dan test yang artinya model memiliki kesalahan yang semakin kecil. Pada graph accuracy nilainya makin lama makin meningkat baik pada train dan test yang artinya akurasi modelnya semakin meningkat.

- Buat prediksi pada gambar dari set pengujian

Pada tahap ini mendefinisikan sebuah fungsi `pred_and_plot_image` yang menerima model terlatih, jalur gambar, daftar nama kelas, ukuran gambar, transformasi, dan perangkat target sebagai argumen. Fungsi ini memuat gambar, melakukan transformasi (jika diperlukan), memprediksi kelas gambar menggunakan model, dan menampilkan gambar dengan label prediksi dan probabilitasnya. Selanjutnya, kode menggunakan fungsi ini untuk membuat prediksi dan menampilkan beberapa gambar dari set data uji, serta melakukan prediksi pada gambar kustom yang diunduh dari internet.