

Nama : Andhika Aria Pratama N

NIM : 1103202121

## 09 PyTorch Model Deployment

- Membuat ekstraktor fitur EffNetB2

```
# 1. Setup pretrained EffNetB2 weights
effnetb2_weights = torchvision.models.EfficientNet_B2_Weights.DEFAULT

# 2. Get EffNetB2 transforms
effnetb2_transforms = effnetb2_weights.transforms()

# 3. Setup pretrained model
effnetb2 = torchvision.models.efficientnet_b2(weights=effnetb2_weights)

# 4. Freeze the base layers in the model (this will freeze all layers to
for param in effnetb2.parameters():
    param.requires_grad = False

Downloading: "https://download.pytorch.org/models/efficientnet_b2_rwigh
100%|██████████| 35.2M/35.2M [00:00<00:00, 98.1MB/s]

# Check out EffNetB2 classifier head
effnetb2.classifier

Sequential(
  (0): Dropout(p=0.3, inplace=True)
  (1): Linear(in_features=1408, out_features=1000, bias=True)
)

# 5. Update the classifier head
effnetb2.classifier = nn.Sequential(
    nn.Dropout(p=0.3, inplace=True), # keep dropout layer same
    nn.Linear(in_features=1408, # keep in_features same
              out_features=3)) # change out_features to suit our number
```

Pada tahap `effnetb2` mengambil data pre-trained wight yang nantinya digunakan. Lalu mentransformasi datanya setelahnya model akan menggunakan data pre-trained tersebut.

- Membuat fungsi untuk membuat ekstraktor fitur EffNetB2

```
def create_effnetb2_model(num_classes:int=3,
                        seed:int=42):
    """Creates an EfficientNetB2 feature extractor model and transform

    Args:
        num_classes (int, optional): number of classes in the classification task. Defaults to 3.
        seed (int, optional): random seed value. Defaults to 42.

    Returns:
        model (torch.nn.Module): EffNetB2 feature extractor model.
        transforms (torchvision.transforms): EffNetB2 image transforms
    """
    # 1, 2, 3. Create EffNetB2 pretrained weights, transforms and model
    weights = torchvision.models.EfficientNet_B2_Weights.DEFAULT
    transforms = weights.transforms()
    model = torchvision.models.efficientnet_b2(weights=weights)

    # 4. Freeze all layers in base model
    for param in model.parameters():
        param.requires_grad = False

    # 5. Change classifier head with random seed for reproducibility
    torch.manual_seed(seed)
    model.classifier = nn.Sequential(
        nn.Dropout(p=0.3, inplace=True),
        nn.Linear(in_features=1408, out_features=num_classes),
    )

    return model, transforms

effnetb2, effnetb2_transforms = create_effnetb2_model(num_classes=3,
                                                    seed=42)

from torchinfo import summary
```

Pada tahap dilakukan implementasi untuk menciptakan model efficientnetb2 sebagai ekstraktor fitur dengan classifier head yang telah disesuaikan. Setelahnya classifier head datanya diganti.


- Melatih ekstraktor fitur EffNetB2

```
from going_modular.going_modular import engine

# Setup optimizer
optimizer = torch.optim.Adam(params=effnetb2.parameters(),
                              lr=1e-3)

# Setup loss function
loss_fn = torch.nn.CrossEntropyLoss()

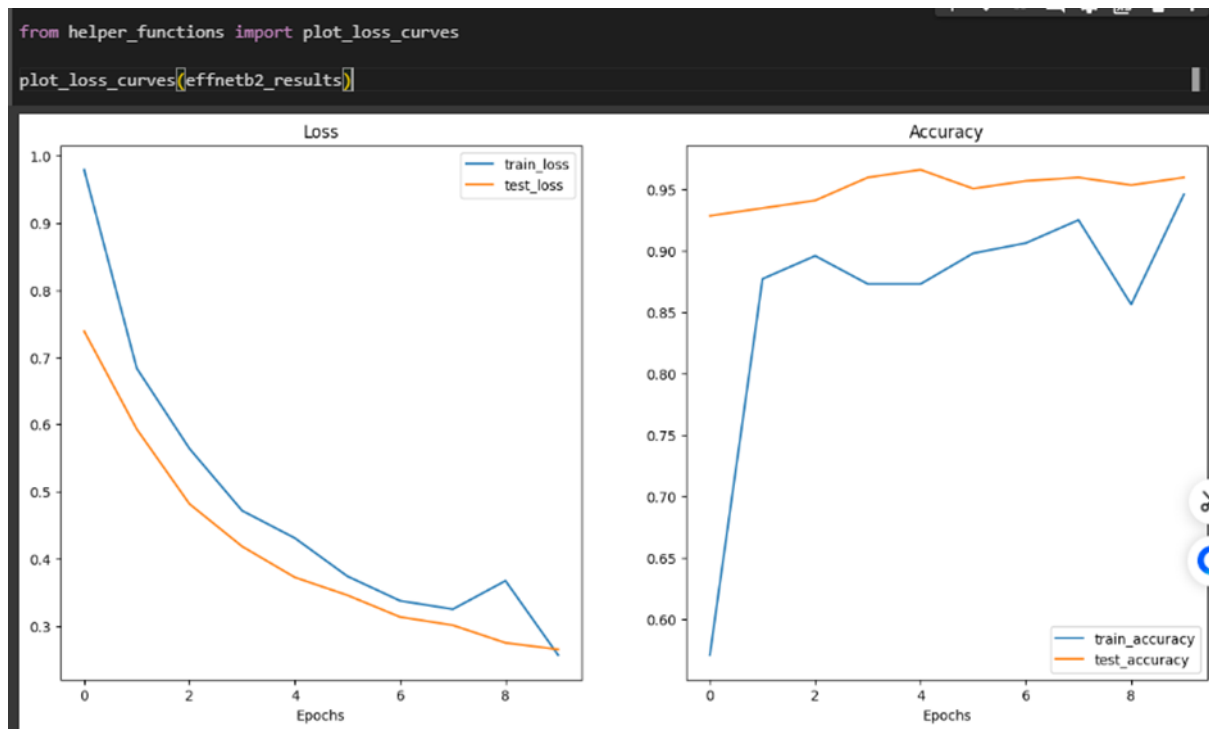
# Set seeds for reproducibility and train the model
set_seeds()
effnetb2_results = engine.train(model=effnetb2,
                                train_dataloader=train_dataloader_effnetb2,
                                test_dataloader=test_dataloader_effnetb2,
                                epochs=10,
                                optimizer=optimizer,
                                loss_fn=loss_fn,
                                device=device)
```

100%  10/10 [31:23<00:00, 188.22s/it]

|           |                    |                   |                   |                  |
|-----------|--------------------|-------------------|-------------------|------------------|
| Epoch: 1  | train_loss: 0.9794 | train_acc: 0.5708 | test_loss: 0.7390 | test_acc: 0.9284 |
| Epoch: 2  | train_loss: 0.6835 | train_acc: 0.8771 | test_loss: 0.5929 | test_acc: 0.9347 |
| Epoch: 3  | train_loss: 0.5640 | train_acc: 0.8958 | test_loss: 0.4816 | test_acc: 0.9409 |
| Epoch: 4  | train_loss: 0.4716 | train_acc: 0.8729 | test_loss: 0.4183 | test_acc: 0.9597 |
| Epoch: 5  | train_loss: 0.4308 | train_acc: 0.8729 | test_loss: 0.3723 | test_acc: 0.9659 |
| Epoch: 6  | train_loss: 0.3738 | train_acc: 0.8979 | test_loss: 0.3456 | test_acc: 0.9506 |
| Epoch: 7  | train_loss: 0.3373 | train_acc: 0.9062 | test_loss: 0.3132 | test_acc: 0.9568 |
| Epoch: 8  | train_loss: 0.3248 | train_acc: 0.9250 | test_loss: 0.3009 | test_acc: 0.9597 |
| Epoch: 9  | train_loss: 0.3671 | train_acc: 0.8562 | test_loss: 0.2747 | test_acc: 0.9534 |
| Epoch: 10 | train_loss: 0.2564 | train_acc: 0.9458 | test_loss: 0.2649 | test_acc: 0.9597 |

Melakukan pelatihan yang mana datasetnya sudah dibagi menjadi train dan test data. Pada tahap ini dilakukan optimizer dengan adam sebesar  $1e-3$  sebagai parameter. Model dilakukan train sebanyak 10 epoch dihasilkan akhirnya pada presentase : train\_loss = 25%, train\_acc = 94%, test\_loss = 26%, test\_acc = 95%.

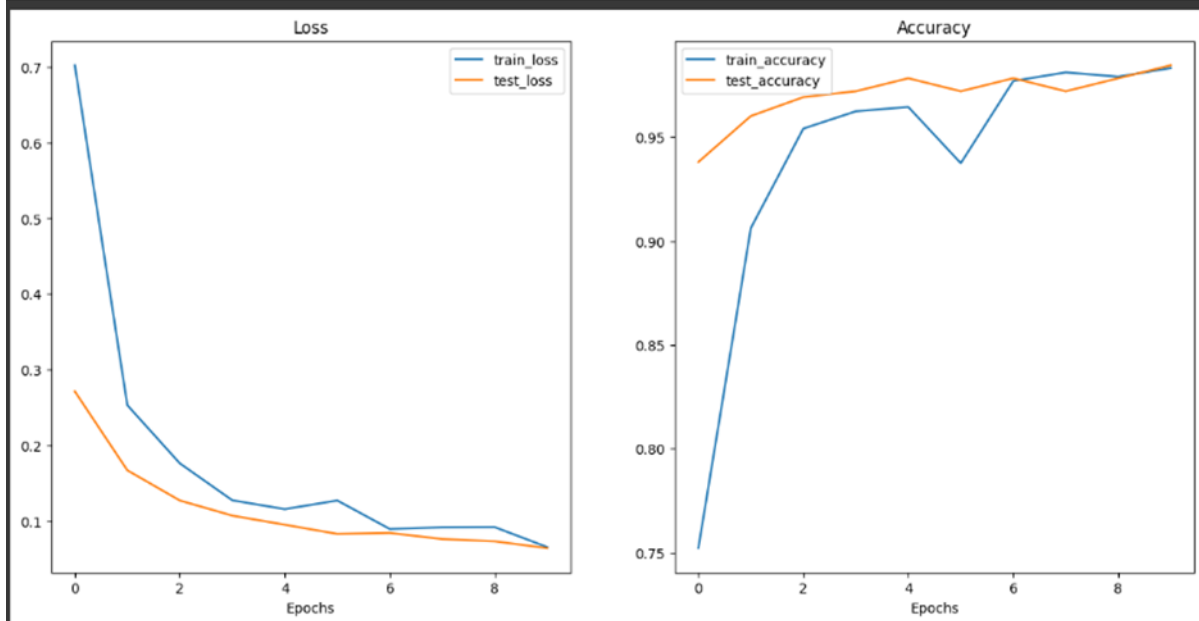
- Memeriksa kurva kerugian EffNetB2



Hasil dari train model sebelumnya yang dibentuk graph. Dari graph loss hasil nilai loss semakin lama semakin kecil yang artinya nilai error semakin lama semakin sedikit yang error. Pada graph akurasi dihasilkan nilai semakin meningkat setiap pengujiannya yang artinya nilai tersebut akurasi bagus.

- Membuat ekstraktor fitur ViT

```
from helper_functions import plot_loss_curves
plot_loss_curves(vit_results)
```



Pada tahap ini membuat model vision transformer (ViT) dengan arsitektur menggunakan pretrained weights, memodifikasi classifier head sesuai dengan jumlah kelas yang diinginkan, dan mencetak ringkasan model, memberikan fleksibilitas untuk konfigurasi tugas klasifikasi. Serta melakukan train pada fitur extractor yang dihasilkan untuk nilai loss semakin lama lossnya semakin kecil dan nilai pada akurasi yang selalu meningkat setiap trainnya. Setelah dilakukan train model tersebut disimpan.

- Membuat prediksi dengan model terlatih dan mengatur waktunya

```
# Turn the test_pred_dicts into a DataFrame
import pandas as pd
effnetb2_test_pred_df = pd.DataFrame(effnetb2_test_pred_dicts)
effnetb2_test_pred_df.head()
```

|   | image_path  | class_name | pred_prob | pred_class | time_for_pred | correct |
|---|---|------------|-----------|------------|---------------|---------|
| 0 | data/pizza_steak_sushi_20_percent/test/pizza/2... | pizza      | 0.9226    | pizza      | 0.2810        | True    |
| 1 | data/pizza_steak_sushi_20_percent/test/pizza/4... | pizza      | 0.5292    | steak      | 0.1529        | False   |
| 2 | data/pizza_steak_sushi_20_percent/test/pizza/9... | pizza      | 0.7431    | pizza      | 0.1412        | True    |
| 3 | data/pizza_steak_sushi_20_percent/test/pizza/7... | pizza      | 0.7909    | pizza      | 0.1528        | True    |
| 4 | data/pizza_steak_sushi_20_percent/test/pizza/1... | pizza      | 0.9352    | pizza      | 0.1365        | True    |

```
# Turn vit_test_pred_dicts into a DataFrame
import pandas as pd
vit_test_pred_df = pd.DataFrame(vit_test_pred_dicts)
vit_test_pred_df.head()
```

|   | image_path  | class_name | pred_prob | pred_class | time_for_pred | correct |
|---|---|------------|-----------|------------|---------------|---------|
| 0 | data/pizza_steak_sushi_20_percent/test/pizza/2... | pizza      | 0.9977    | pizza      | 0.7434        | True    |
| 1 | data/pizza_steak_sushi_20_percent/test/pizza/4... | pizza      | 0.9977    | pizza      | 0.7491        | True    |
| 2 | data/pizza_steak_sushi_20_percent/test/pizza/9... | pizza      | 0.9984    | pizza      | 0.7000        | True    |
| 3 | data/pizza_steak_sushi_20_percent/test/pizza/7... | pizza      | 0.9980    | pizza      | 0.7244        | True    |
| 4 | data/pizza_steak_sushi_20_percent/test/pizza/1... | pizza      | 0.9981    | pizza      | 0.7007        | True    |

Pada tahap ini dilakukan prediksi pada model yang telah dilatih. Hasil dari model EffNetB2 dan model vit ekstraktor keduanya memiliki nilai akurasi yang baik, namun model vit ekstraktor memiliki akurasi yang pada saat di uji dibanding dengan effnetb2.

- Membandingkan hasil

```
# Turn stat dictionaries into DataFrame
df = pd.DataFrame([effnetb2_stats, vit_stats])

# Add column for model names
df["model"] = ["EffNetB2", "ViT"]

# Convert accuracy to percentages
df["test_acc"] = round(df["test_acc"] * 100, 2)

df
```

|   | test_loss | test_acc | number_of_parameters | model_size (MB) | time_per_pred_cpu | model    |
|---|-----------|----------|----------------------|-----------------|-------------------|----------|
| 0 | 0.264900  | 95.97    | 7705221              | 29              | 0.1569            | EffNetB2 |
| 1 | 0.064434  | 98.47    | 85800963             | 327             | 0.8156            | ViT      |

Walaupun model vit memiliki akurasi yang baik Ketika diuji tetapi penggunaan komputasinya yang sangat besar berbeda dengan effnetb2 yang memakai komputasi cukup kecil.

- Mengubah Demo FoodVision Mini Gradio menjadi aplikasi yang dapat diterapkan

Pada tahap melakukan transformasi dan prediksi menggunakan model EfficientNetB2 pada gambar input, mengimplementasikan demo Gradio yang memanfaatkan fungsi untuk menampilkan prediksi dan waktu inferensi model pada gambar-gambar uji, dan akhirnya meluncurkan antarmuka demo Gradio yang memungkinkan pengguna untuk mengunggah gambar dan menerima prediksi serta waktu inferensi dari model klasifikasi makanan yang telah dilatih.

- Menciptakan FoodVision Besar

Pada tahap ini, sebuah model EfficientNetB2 untuk klasifikasi 101 kelas pada dataset Food101 dibuat dan diatur untuk melatih dengan transformasi data. Selanjutnya, dilakukan pembagian dataset

Food101 menjadi data latih dan uji dengan proporsi 20%, serta dibuat DataLoader untuk kedua dataset tersebut. Model tersebut kemudian dilatih dengan optimizer Adam, fungsi loss CrossEntropy dengan label smoothing, dan hasil pelatihan dievaluasi dengan melakukan plotting terhadap kurva kerugian. Setelah mencapai hasil yang memuaskan, model tersebut disimpan, kemudian di-load kembali untuk memverifikasi proses penyimpanan dan pembacaan model. Ukuran file model yang disimpan juga dihitung dan ditampilkan.

- Mengubah model FoodVision Big menjadi aplikasi yang dapat diterapkan

Dalam tahap ini, dilakukan beberapa tindakan:

1. Direktori dan struktur file untuk demo FoodVision Big diatur, termasuk pengunduhan contoh gambar, pemindahan model yang telah dilatih, serta pembuatan dan penulisan class names ke file.
2. File Python dibuat untuk model, aplikasi Gradio, dan file requirements.txt untuk dependencies.
3. Aplikasi Gradio untuk FoodVision Big dibuat, termasuk fungsi prediksi, interface Gradio, dan pengaturan untuk meluncurkan aplikasi. Selain itu, direktori demo dan file terkait dikompres dalam zip dan disediakan untuk diunduh sebagai paket aplikasi FoodVision Big.