IMPORT LIBRARIES

In [ ]:
```
!pip install torchinfo
```

In [ ]:
```
import matplotlib.pyplot as plt
import torch
import torchvision

from torch import nn
from torchvision import transforms
from torchinfo import summary
import os

from torchvision import datasets, transforms
from torch.utils.data import DataLoader
```

In [ ]:
```
device = "cuda" if torch.cuda.is_available() else "cpu"
device
```

Out[2]:
```
'cuda'
```

In [ ]:
```
def set_seeds(seed: int=42):
    """Sets random sets for torch operations.

    Args:
        seed (int, optional): Random seed to set. Defaults to 42.
    """

    torch.manual_seed(seed)

    torch.cuda.manual_seed(seed)



pretrained_vit_weights = torchvision.models.ViT_B_16_Weights.DEFAULT

pretrained_vit = torchvision.models.vit_b_16(weights=pretrained_vit_weights

for parameter in pretrained_vit.parameters():
    parameter.requires_grad = False

class_names = ['HL','UL']

set_seeds()
pretrained_vit.heads = nn.Linear(in_features=768, out_features=len(class_na
```

```
Downloading: "https://download.pytorch.org/models/vit_b_16-c867db91.pth"
to /root/.cache/torch/hub/checkpoints/vit_b_16-c867db91.pth
100%|██████████| 330M/330M [00:01<00:00, 196MB/s]
```

In [ ]:
```
!pip install torchinfo
```

```
Collecting torchinfo
  Downloading torchinfo-1.8.0-py3-none-any.whl (23 kB)
Installing collected packages: torchinfo
Successfully installed torchinfo-1.8.0
```

In [ ]:
```
summary(model=pretrained_vit,
        input_size=(32, 3, 224, 224),
        col_names=["input_size", "output_size", "num_params", "trainable"],
        col_width=20,
        row_settings=["var_names"]
)
```

Out[5]:

```
==========================================================================
==========================================================================
Layer (type (var_name))                                  Input Shape
Output Shape         Param #              Trainable
==========================================================================
==========================================================================
VisionTransformer (VisionTransformer)                    [32, 3, 224,
224]    [32, 2]              768                 Partial
├─Conv2d (conv_proj)                                     [32, 3, 224,
224]    [32, 768, 14, 14]    (590,592)           False
├─Encoder (encoder)                                      [32, 197, 76
8]       [32, 197, 768]       151,296             False
│    └─Dropout (dropout)                                 [32, 197, 76
8]       [32, 197, 768]       --                  --
│    └─Sequential (layers)                               [32, 197, 76
8]       [32, 197, 768]       --                  False
│    │    └─EncoderBlock (encoder_layer_0)               [32, 197, 76
8]       [32, 197, 768]       (7,087,872)         False
│    │    └─EncoderBlock (encoder_layer_1)               [32, 197, 76
8]       [32, 197, 768]       (7,087,872)         False
│    │    └─EncoderBlock (encoder_layer_2)               [32, 197, 76
8]       [32, 197, 768]       (7,087,872)         False
│    │    └─EncoderBlock (encoder_layer_3)               [32, 197, 76
8]       [32, 197, 768]       (7,087,872)         False
│    │    └─EncoderBlock (encoder_layer_4)               [32, 197, 76
8]       [32, 197, 768]       (7,087,872)         False
│    │    └─EncoderBlock (encoder_layer_5)               [32, 197, 76
8]       [32, 197, 768]       (7,087,872)         False
│    │    └─EncoderBlock (encoder_layer_6)               [32, 197, 76
8]       [32, 197, 768]       (7,087,872)         False
│    │    └─EncoderBlock (encoder_layer_7)               [32, 197, 76
8]       [32, 197, 768]       (7,087,872)         False
│    │    └─EncoderBlock (encoder_layer_8)               [32, 197, 76
8]       [32, 197, 768]       (7,087,872)         False
│    │    └─EncoderBlock (encoder_layer_9)               [32, 197, 76
8]       [32, 197, 768]       (7,087,872)         False
│    │    └─EncoderBlock (encoder_layer_10)              [32, 197, 76
8]       [32, 197, 768]       (7,087,872)         False
│    │    └─EncoderBlock (encoder_layer_11)              [32, 197, 76
8]       [32, 197, 768]       (7,087,872)         False
│    └─LayerNorm (ln)                                    [32, 197, 76
8]       [32, 197, 768]       (1,536)             False
├─Linear (heads)                                         [32, 768]
[32, 2]              1,538               True
==========================================================================
==========================================================================
Total params: 85,800,194
Trainable params: 1,538
Non-trainable params: 85,798,656
Total mult-adds (G): 5.52
==========================================================================
==========================================================================
Input size (MB): 19.27
Forward/backward pass size (MB): 3330.74
Params size (MB): 229.20
Estimated Total Size (MB): 3579.20
==========================================================================
==========================================================================
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')

train_dir = '/content/drive/MyDrive/VISION_TRANSFORMER/TRAIN/'
test_dir = '/content/drive/MyDrive/VISION_TRANSFORMER/TEST/'
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount,
call drive.mount("/content/drive", force_remount=True).
```

In [ ]:
```
pretrained_vit_transforms = pretrained_vit_weights.transforms()
print(pretrained_vit_transforms)
```

```
ImageClassification(
    crop_size=[224]
    resize_size=[256]
    mean=[0.485, 0.456, 0.406]
    std=[0.229, 0.224, 0.225]
    interpolation=InterpolationMode.BILINEAR
)
```

In [ ]:
```

NUM_WORKERS = os.cpu_count()

def create_dataloaders(
    train_dir: str,
    test_dir: str,
    transform: transforms.Compose,
    batch_size: int,
    num_workers: int=NUM_WORKERS
):


  train_data = datasets.ImageFolder(train_dir, transform=transform)
  test_data = datasets.ImageFolder(test_dir, transform=transform)


  class_names = train_data.classes


  train_dataloader = DataLoader(
      train_data,
      batch_size=batch_size,
      shuffle=True,
      num_workers=num_workers,
      pin_memory=True,
  )
  test_dataloader = DataLoader(
      test_data,
      batch_size=batch_size,
      shuffle=False,
      num_workers=num_workers,
      pin_memory=True,
  )

  return train_dataloader, test_dataloader, class_names
```

In [ ]:
```
train_dataloader_pretrained, test_dataloader_pretrained, class_names = crea
```

In [ ]:
```
try:
    from going_modular.going_modular import data_setup, engine
except:

    print("[INFO] Couldn't find going_modular scripts... downloading them f
    !git clone https://github.com/mrdbourke/pytorch-deep-learning
    !mv pytorch-deep-learning/going_modular .
    !rm -rf pytorch-deep-learning
```

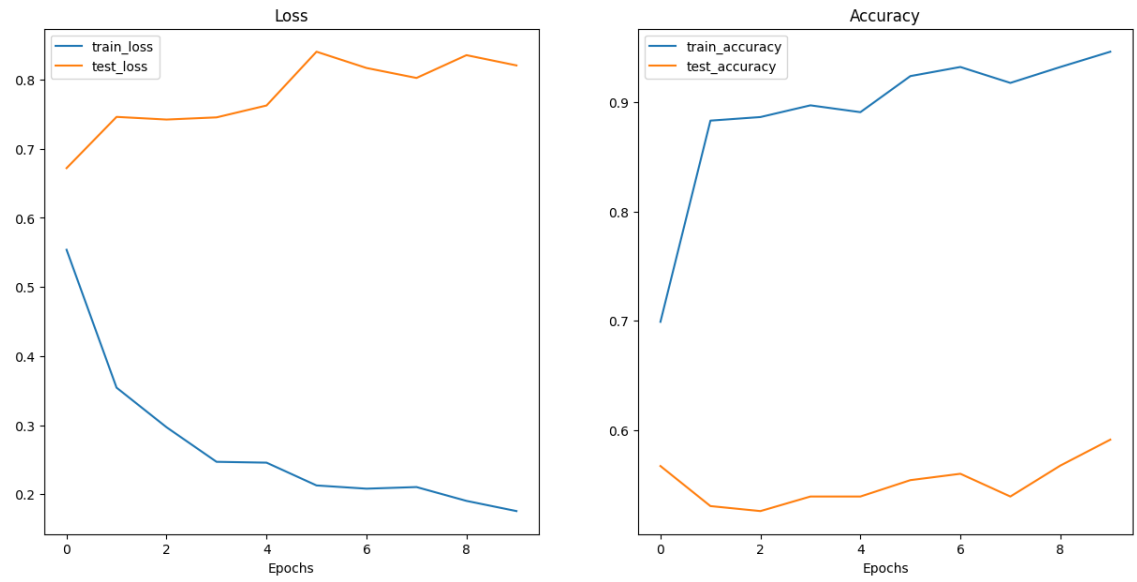
```
[INFO] Couldn't find going_modular scripts... downloading them from GitHu
b.
Cloning into 'pytorch-deep-learning'...
remote: Enumerating objects: 3884, done.
remote: Counting objects: 100% (1075/1075), done.
remote: Compressing objects: 100% (141/141), done.
remote: Total 3884 (delta 977), reused 980 (delta 933), pack-reused 2809
Receiving objects: 100% (3884/3884), 648.04 MiB | 36.87 MiB/s, done.
Resolving deltas: 100% (2268/2268), done.
Updating files: 100% (248/248), done.
```

In [ ]:
```
from going_modular.going_modular import engine

optimizer = torch.optim.Adam(params=pretrained_vit.parameters(),
                             lr=1e-3)
loss_fn = torch.nn.CrossEntropyLoss()

set_seeds()
pretrained_vit_results = engine.train(model=pretrained_vit,
                                      train_dataloader=train_dataloader_pre
                                      test_dataloader=test_dataloader_pretr
                                      optimizer=optimizer,
                                      loss_fn=loss_fn,
                                      epochs=10,
                                      device=device)
```

```
  0%|          | 0/10 [00:00<?, ?it/s]

Epoch: 1 | train_loss: 0.5539 | train_acc: 0.6991 | test_loss: 0.6720 | t
est_acc: 0.5671
Epoch: 2 | train_loss: 0.3543 | train_acc: 0.8833 | test_loss: 0.7462 | t
est_acc: 0.5304
Epoch: 3 | train_loss: 0.2970 | train_acc: 0.8866 | test_loss: 0.7423 | t
est_acc: 0.5258
Epoch: 4 | train_loss: 0.2469 | train_acc: 0.8973 | test_loss: 0.7455 | t
est_acc: 0.5392
Epoch: 5 | train_loss: 0.2457 | train_acc: 0.8911 | test_loss: 0.7627 | t
est_acc: 0.5392
Epoch: 6 | train_loss: 0.2127 | train_acc: 0.9241 | test_loss: 0.8405 | t
est_acc: 0.5542
Epoch: 7 | train_loss: 0.2080 | train_acc: 0.9324 | test_loss: 0.8170 | t
est_acc: 0.5600
Epoch: 8 | train_loss: 0.2104 | train_acc: 0.9179 | test_loss: 0.8025 | t
est_acc: 0.5392
Epoch: 9 | train_loss: 0.1904 | train_acc: 0.9324 | test_loss: 0.8355 | t
est_acc: 0.5675
Epoch: 10 | train_loss: 0.1756 | train_acc: 0.9464 | test_loss: 0.8207 |
test_acc: 0.5913
```

pretrained ViT performed far better than our custom ViT model trained from scratch (in the same amount of time).

In [ ]:
```
def plot_loss_curves(results):
    """Plots training curves of a results dictionary.

    Args:
        results (dict): dictionary containing list of values, e.g.
            {"train_loss": [...],
             "train_acc": [...],
             "test_loss": [...],
             "test_acc": [...]}
    """
    loss = results["train_loss"]
    test_loss = results["test_loss"]

    accuracy = results["train_acc"]
    test_accuracy = results["test_acc"]

    epochs = range(len(results["train_loss"]))

    plt.figure(figsize=(15, 7))


    plt.subplot(1, 2, 1)
    plt.plot(epochs, loss, label="train_loss")
    plt.plot(epochs, test_loss, label="test_loss")
    plt.title("Loss")
    plt.xlabel("Epochs")
    plt.legend()


    plt.subplot(1, 2, 2)
    plt.plot(epochs, accuracy, label="train_accuracy")
    plt.plot(epochs, test_accuracy, label="test_accuracy")
    plt.title("Accuracy")
    plt.xlabel("Epochs")
    plt.legend()


plot_loss_curves(pretrained_vit_results)
```



Let's make Prediction:

In [ ]:
```
import requests

from going_modular.going_modular.predictions import pred_and_plot_image


custom_image_path = "/content/drive/MyDrive/VISION_TRANSFORMER/TEST/UL/90a.

pred_and_plot_image(model=pretrained_vit,
                    image_path=custom_image_path,
                    class_names=class_names)
```