DLP HW3 李明峻 0856558

Introduction:

這次的作業利用 torchvision 的 ResNet model 來去訓練 Retinopathy Image dataset，利用自定義的 Dataloader 預先處理資料集，再送進神經網路做運算。分別有 ResNet18 及 ResNet50 兩種 model，並且去比較利用 pretrained model 和不用 pretraining 的效能差異。

Experiment set up:

A. The details of model:

若使用預設的 ResNet，最後一層的 FC 是 1000 個 class，但這個問題最後只分成 5 類，所以要更改一下最後一層的類別數，更改方式如下

```
in_features = model.fc.in_features
model.fc = torch.nn.Linear(in_features, 5)
model = model.to(device)
```

另外，可以調整 pretrained 參數決定是否要使用 pretrained weights，若使用預訓練的 model 可以加速訓練的過程。

Loss function 使用 Cross_Entropy; Optimizer 使用 SGD。

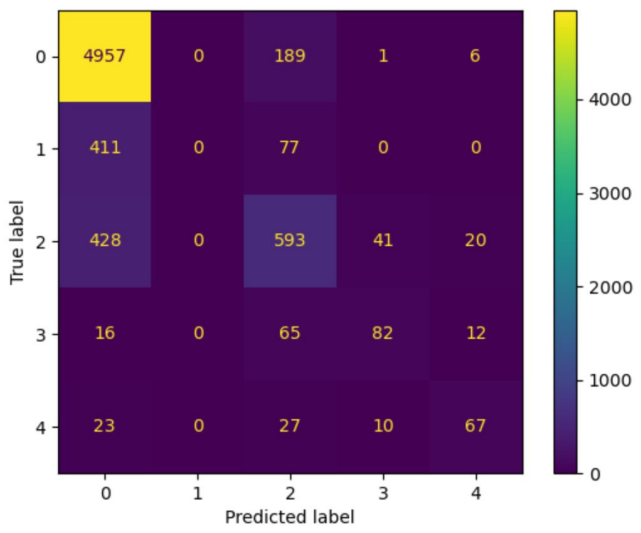
B. The details of Dataloader:

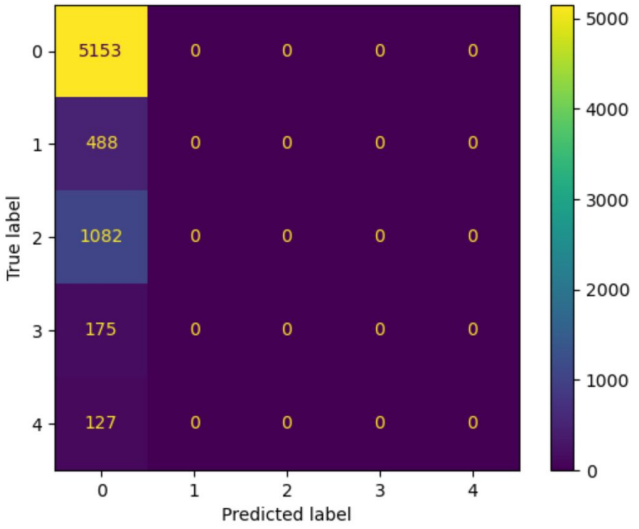
這裡我使用 torchvision 中的 transforms 對資料集做 data augmentation 的動作，利用 .Compose() 將不同的處理整合起來，我在 training data 的部分做較多的處理 (eg. RandomRotation, ColorJitter...)，希望可以增加資料的隨機性避免 overfitting，並且同時提升效率。

```
# Data
train_transform = torchvision.transforms.Compose([
        torchvision.transforms.RandomHorizontalFlip(p=0.5),
        torchvision.transforms.RandomVerticalFlip(p=0.5),
        torchvision.transforms.RandomRotation(degrees=45),
        torchvision.transforms.ColorJitter(0.05,0.05,0.05,0.05),
        torchvision.transforms.ToTensor(),  # Normalization + Transpose
    ])
test_transform = torchvision.transforms.Compose([
        torchvision.transforms.ToTensor(),
    ])
```

C. Describing evaluation through the confusion matrix:

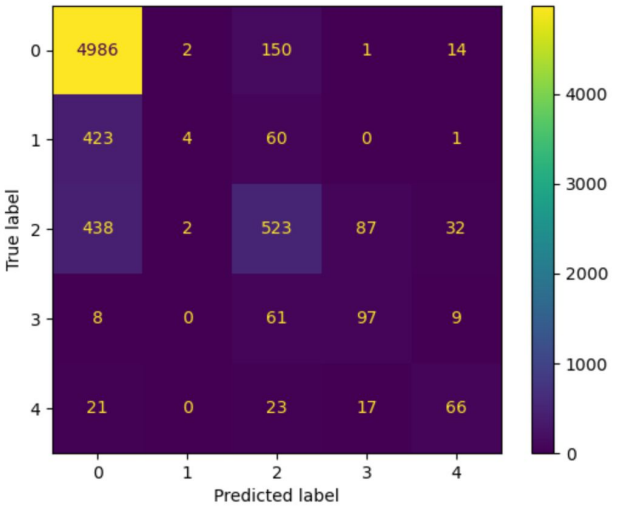
利用 sklearn 及 matplotlib 來畫 confusion matrix 及 accuracy figure，由 confusion matrix 較可以判斷模型的優劣。

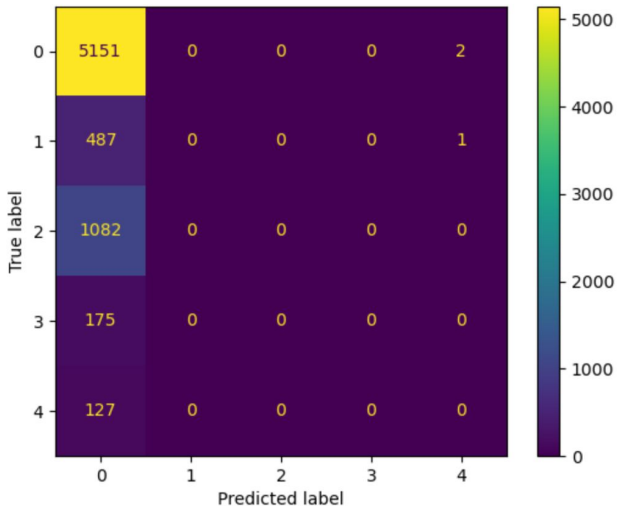




上圖左是 ResNet18 with pretraining 的 confusion matrix，圖右則是 ResNet18 without pretraining 的 confusion matrix。

對於 ResNet18 with pretraining，幾乎無法判別 label 1，而對於 ResNet18 without pretraining，是除了 label 0 以外幾乎都無法判別。





上圖左是 ResNet50 with pretraining 的 confusion matrix，圖右則是 ResNet50 without pretraining 的 confusion matrix。

ResNet50 with pretraining 一樣幾乎無法判別 label 1，而 ResNet50 without pretraining 則是除了 label 0 以外幾乎都無法判別。

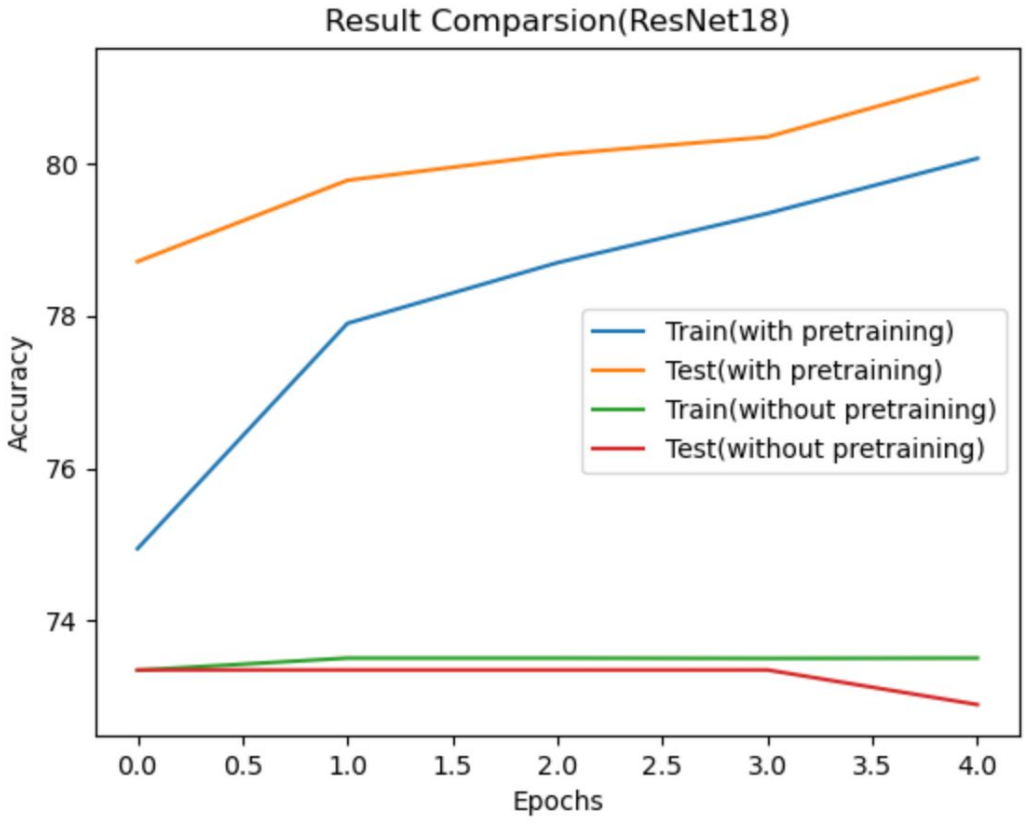
Experiment Result:

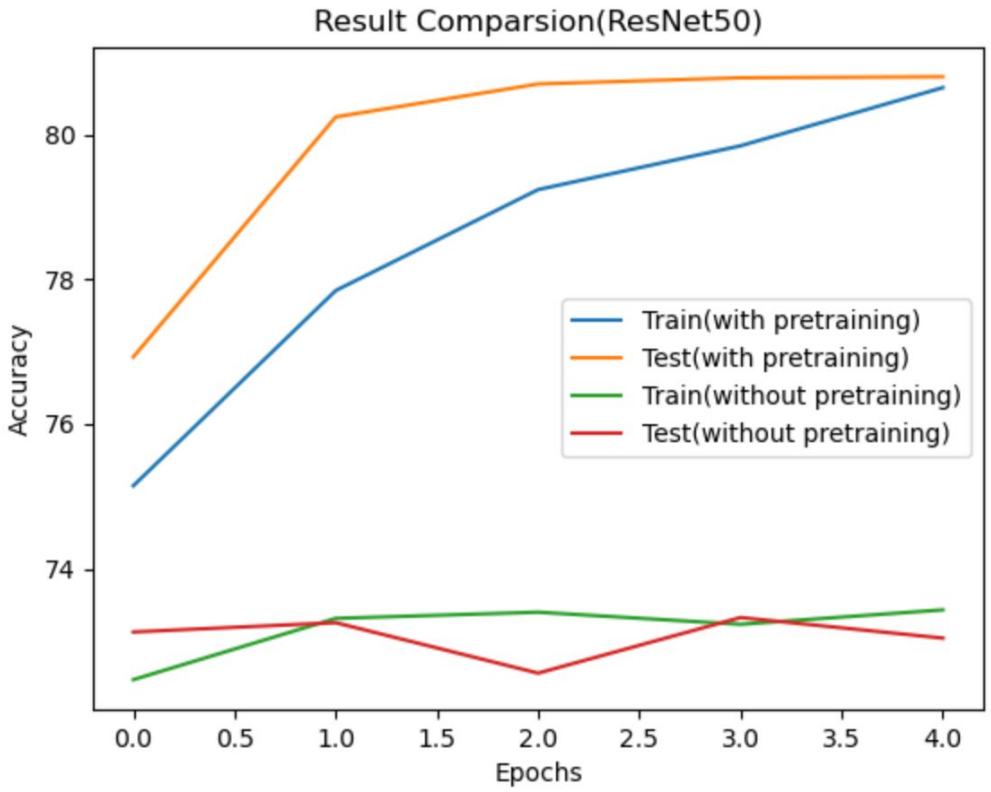
A. The highest testing accuracy:

```
> Found 7025 images...
ResNet18_with pretraining: Accuracy = 81.42%
ResNet18_without pretraining: Accuracy = 73.34%
ResNet50_with pretraining: Accuracy = 80.80%
ResNet50_without pretraining: Accuracy = 73.32%
```

ResNet18 with pretraining 可以得到最好的 accuracy

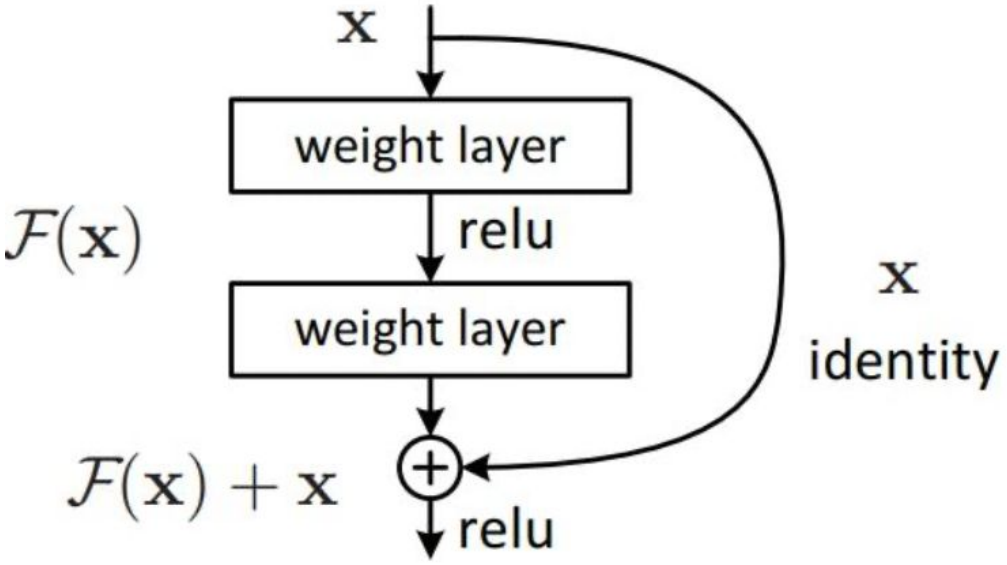
B. Comparison figure:





Discussion:

ResNet 是希望透過增加網路的 layers 數目來達到更好的效能，但是會遇到 Degradation problem，解決的方法就是輸出層改成殘差加上原本的輸入，又稱作 shortcut 的機制。



經過觀察後我發現，ResNet50 的 performance 其實並沒有 ResNet18 來得好，ResNet18 的層數其實就足夠學習到重要的 features 來去分類了。

另外，關於 pretrained model，是已經利用一些資料集預先訓練好的一組模型，網路已經學習到資料取向的大方向了，所以當我們拿預訓練好的 model 來 train 時，在大致 features 確定的情況之下，去學習更細微的部分，不僅僅加速了訓練的速度，也可以更加提升網路架構的準確率。