Lab 4 Report
309551064 張凱翔

1. Introduction
   隨著網路越深，可以提取到更複雜的特徵，但卻會因為深的網路而造成梯度消失或爆炸的問題，因此 ResNet 被提出來解決這個問題。本次作業需要實做 Dataset、ResNet18 和 ResNet50 來判斷糖尿病所引發視網膜病變的嚴重程度(0~4)，並使用 confusion matrix 來比較有使用 pretraining model 和沒有使用 pretraining model 的差別，還需要將準確率提高到>=82%。
2. Experiment setups
   A. The detail of your model (ResNet)
      BATCHSIZE = 8
      LEARNING_RATE = 1e-3
      EPOCHS = 15
   - ResNet18
     ResNet18 會先經過一次 convolution、batch normalization、relu 和 max pooling 後，再經過 4 層 layer，最後經過 average pooling 後拉平過一個 fully connected layer 輸出 1000 維。(左圖)
     layer 都由兩個 BasicBlock 組成，包含兩次 convolution、兩次 batch normalization 和兩次 relu。比較特別的有兩個地方：
     1. 會先記錄原始的值最後再跟經過 convolution 的合起來，利用這種架構可以防止梯度消失。
     2. 當輸出和輸入維度不同時，需要使用 downsample 來讓彼此可以相加 (右圖)

```python
def forward(self, img):
    output = self.conv1(img)
    output = self.bn1(output)
    output = self.relu(output)
    output = self.maxpool(output)

    output = self.layer1(output)
    output = self.layer2(output)
    output = self.layer3(output)
    output = self.layer4(output)

    output = self.avgpool(output)
    output = output.view(output.size(0), -1)
    output = self.fc(output)
    output = self.relu(output)
    output = self.out(output)
    return output
```

```python
def forward(self, x):
    residual = x
    output = self.conv1(x)
    output = self.bn1(output)
    output = self.relu(output)

    output = self.conv2(output)
    output = self.bn2(output)

    if self.downsample is not None:
        residual = self.downsample(x)

    output += residual
    output = self.relu(output)

    return output
```

   - ResNet50
     ResNet50 的整體架構和 Resnet18 很像，差別在於它的 layer 是由[3, 4, 6, 3]個 bottleneck 組成，每次輸出為輸入的 4 倍，且 Bottleneck 比 BasicBlock 還多了一層。

```python
def forward(self, img):
    output = self.conv1(img)
    output = self.bn1(output)
    output = self.relu(output)
    output = self.maxpool(output)

    output = self.layer1(output)
    output = self.layer2(output)
    output = self.layer3(output)
    output = self.layer4(output)

    output = self.avgpool(output)
    output = output.view(output.size(0), -1)
    output = self.fc(output)
    output = self.out(output)
    return output
```

```python
def forward(self, x):
    residual = x
    output = self.conv1(x)
    output = self.bn1(output)
    output = self.relu(output)

    output = self.conv2(output)
    output = self.bn2(output)

    output = self.relu(output)

    output = self.conv3(output)
    output = self.bn3(output)

    if self.downsample is not None:
        residual = self.downsample(x)

    output += residual
    output = self.relu(output)

    return output
```

     我在最後一層加入一層 Linear layer，輸入為 1000、輸出為 5，來判斷嚴重程度。
     在訓練的時候會也判斷有沒有要使用 pretraining model 然後將 weights load 進來。

B. The details of your Dataloader

- 我把讀檔部份移到外面然後將 img 和 label 當作 Dataset 的輸入，並指定輸入為 train data 還是 test data。

```
train_img_csv = np.squeeze(pd.read_csv('train_img.csv').values)
train_label_csv = np.squeeze(pd.read_csv('train_label.csv').values)
test_img_csv = np.squeeze(pd.read_csv('test_img.csv').values)
test_label_csv = np.squeeze(pd.read_csv('test_label.csv').values)
```

```
train_dataset = RetinopathyDataset(train_img_csv, train_label_csv, 'train')
test_dataset = RetinopathyDataset(test_img_csv, test_label_csv, 'test')
```

- __init__:

  使用 self 存取輸入的參數，讓在__getitem__中也能使用並另外宣告兩種 transforms 的方法，一種是會隨機水平或垂直翻轉並 normalize 到[0, 1]之間的 **transform_for_0**，另一種只會 normalize 到[0, 1]之間的 **transform**。

```
    def __init__(self, img_name, label, mode):
        self.label = label
        self.img_name = img_name
        self.mode = mode
        self.transform_for_0 = transforms.Compose([
            transforms.RandomHorizontalFlip(),
            transforms.RandomVerticalFlip(),
            transforms.ToTensor()
        ])
        self.transform = transforms.Compose([
            transforms.ToTensor()
        ])
```

__len__:

回傳 label 的長度來代表這個 Dataset 的長度

```
def __len__(self):
    return len(self.label)
```

__getitem__:

從讀進來的參數中取得第 index 個 img 的檔案名字並讀取檔案轉成 RGB 模式。然後我有觀察到 train data 中 label 為 0 的數量很多，這可能導致就算將所有的 predict 為 0 也可以有很低的 loss，使整體的準確率上不去。

```
Counter({0: 20655, 3: 698, 2: 4210, 1: 1955, 4: 581})
```

為了避免這種情況，會去判斷如果現在用的 Dataset 是 train data 且它的 label 不為 0 時，讓它使用 **transform_for_0** 來增加其餘 label 的訓練資料量，反之則使用 **transform**，最後回傳 img 的 pixel data 和 label。

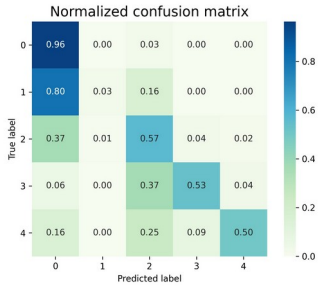
```
def __getitem__(self, index):
    image_path = './data/' + self.img_name[index] + '.jpeg'
    img = Image.open(image_path).convert('RGB')
    if self.mode == 'train' and self.label[index] != 0:
        img = self.transform_for_0(img).numpy()
    else:
        img = self.transform(img).numpy()

    return img, self.label[index]
```

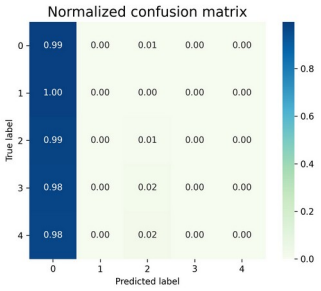
C. Describing your evaluation through the confusion matrix

- ResNet18

(1) with pretraining

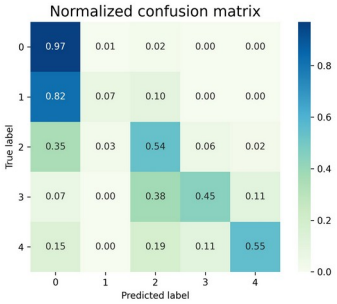


(2) without pretraining

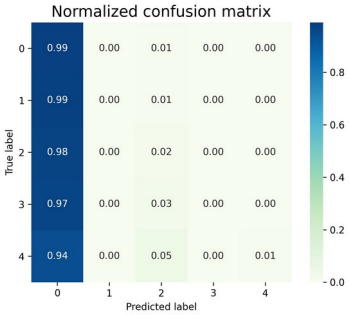


- ResNet50

(1) with pretraining



(2) without pretraining



整體看來，在同樣 epoch 的情況下，有加入 pretraining model 的表現是較佳的，因為 pretraining model 已經預先訓練過可以很好的取出特徵，相對於重頭開始訓練的 model 會比較好，但可以看出來預測出來還是 0 的結果為主，導致其餘 label 都只有 50%左右的準確率，1 的甚至只有 7%，所以雖然 model 的準確率有到 82%但卻是被 0 的 dominate 住，並不能說是很好的 model。

3. Experimental results

A. The highest testing accuracy

- Screenshot

(1) ResNet18

```
with_pretraining 0.8197864768683274
without_pretraining 0.7335231316725979
```

(2) ResNet50

```
with_pretraining 0.8216370106761566
without_pretraining 0.7326690391459074
```

	with pretraining	without pretraining
ResNet18	81.98%	73.35%
ResNet50	82.16%	73.27%

■ Anything you want to present

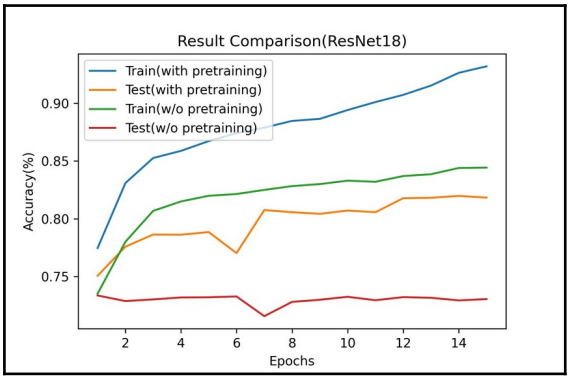
一開始手刻完 ResNet18 和 ResNet50 後就讓它開始跑，但 train 和 test 的準確率始終在 73%左右震盪一直上不去，也試過去調整 hyperparameters 像是 learning rate 和 epochs，但卻只有很小的改變。之後寫完 confusion matrix 後發現 0 的準確率很高，因此去檢查資料的分佈，發現 0 的資料是其他的很多倍，所以就開始對資料進行一些處理。一開始嘗試的方法是 downsampling，去除掉一些 0 的資料讓資料平均一點，但結果還是跟之前差不多，所以就改為 upsampling 對其餘的做翻轉來增加訓練的資料量。結果便是 train 和 test 都有穩定的上升，達到 82%。

```
Epochs 1,       0.7324815829744831 0.7335231316725979
Epochs 2,       0.73507954019716 0.7335231316725979
Epochs 3,       0.73507954019716 0.7335231316725979
Epochs 4,       0.73507954019716 0.7335231316725979
Epochs 5,       0.73507954019716 0.7335231316725979
Epochs 6,       0.73507954019716 0.7335231316725979
Epochs 7,       0.73507954019716 0.7258362989323843
Epochs 8,       0.73507954019716 0.7335231316725979
Epochs 9,       0.73507954019716 0.7335231316725979
Epochs 10,      0.7350439517420548 0.7333807829181495
```

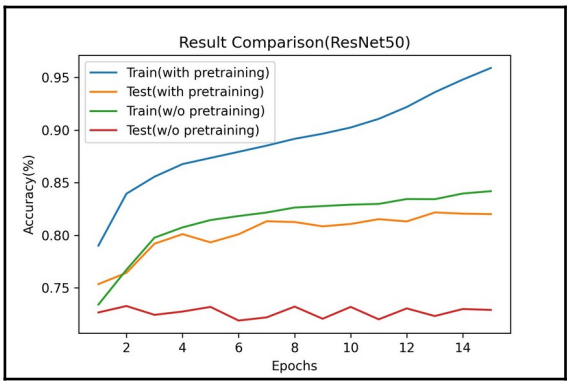
B. Comparison figures

■ Plotting the comparison figures (RseNet18/50, with/without pretraining)

(1) ResNet18



(2) ResNet50



4. Discussion

A. Anything you want to share

透過這次作業了解了在 module 中使用 module 的這種寫法，也了解 ResNet 的整體架構和 residual 的概念。這次的作業花了比較多的時間，主要是花在提高準確率上，上一個作業只加了 scheduler 就可以達到標準，但這次還要找到問題的所在並對資料進行處理，且因為 GPU 只有 8G 沒有辦法 load 很大的 batch size，所以花了很多時間 train。