LAB 4 Diabetic Retinopathy Detection

謝宇恆 411551022

1. Introduction

本實驗藉由 ResNet18 以及 ResNet50 進行糖尿病視網膜病變的分類問題，將照片進行 5 種嚴重性的分類。

2. Experiment Setup

a) The Details of your model(ResNet)

Model 中我利用 pytorch 函示庫中 models 的 ResNet18 以及 ResNet50 進行訓練。Pretrained 的部分我使用了 IMAGENET1K_V1 這個 weights 進行模型的訓練。而在 Unpretrained 的部分我則是將模型的 input feature 設定為 512，輸出則是為 5 種輸出對應到 5 種嚴重性。

```
def getmodels(modelType,pretrained=False):
    if pretrained:
        if modelType=='resnet18':
            model = models.resnet18(weights=ResNet18_Weights.IMAGENET1K_V1)
            model.fc = nn.Linear(in_features=512, out_features = 5)
        else:
            model = models.resnet50(weights=ResNet50_Weights.IMAGENET1K_V1)
            model.fc = nn.Linear(in_features=512, out_features = 5)
    else:
        if modelType=='resnet18':
            model = models.resnet18()
            model.fc = nn.Linear(in_features=512, out_features = 5)
        else:
            model = models.resnet50()
            model.fc = nn.Linear(in_features=512, out_features = 5)
    return model
```

b) The details of your Dataloader:

普遍的 Dataloader 中，都有三個 function，__init__、__len__以及__getitem__。在 init 中我們將檔案路徑放置在 root 變數裡面並將 Train mode 以及 test mode 放在 mode 變數裡面，再拿取到 mode 以及 root 之後我們會呼叫 getData(mode)將含有所有圖片檔名以及 label 的 csv 讀取進去 img_name 以及 label 這兩個變數中。

在要取得圖片時，我們會呼叫 getitem 這個函式，在這個函示中，我們會依據預設的路徑/date/new_test 或者/data/new_train 拿取該資料夾內的圖片，並將該圖片 load 進去模型中，並且也會將圖片對應的 label 讀入至 label 這個變數裡面。

```python
class RetinopathyLoader(data.Dataset):
    def __init__(self, root, mode):
        self.root = root       You, last week • add Lab04 …
        self.img_name, self.label = getData(mode)
        self.mode = mode

        print("> Found %d images..." % (len(self.img_name)))

    def __len__(self):
        """'return the size of dataset"""
        return len(self.img_name)
    def transform(self,img):
        shortSize=min(img.size)
        script = transforms.Compose([
            transforms.RandomHorizontalFlip(),
            transforms.CenterCrop((shortSize,shortSize)),
            transforms.Resize((512, 512)),
            transforms.ToTensor()
        ])

        return script(img)
    def __getitem__(self, index):

        single_img_name = os.path.join(self.root,self.img_name[index]+'.jpeg')
        img=Image.open(single_img_name)
        label=self.label[index]
        try:
            return self.transform(img), label
        except:
            single_img_name = os.path.join(self.root,self.img_name[index+1]+'.jpeg')
            img=Image.open(single_img_name)
            label=self.label[index+1]
            return self.transform(img), label
```

c) Decribing your evaluation through the confusion matrix

我們用一個 5*5 的矩陣(變數:confusion_matrix)儲存最後一次 testing 的結果。

```python
confusion_matrix=np.zeros((num_class,num_class))

with torch.set_grad_enabled(False):
    model.eval()
    correct=0
    for images,targets in tqdm(loader_test,desc=f'(test)'):
        images,targets=images.to(device),targets.to(device,dtype=torch.long)
        predict=model(images)
        predict_class=predict.max(dim=1)[1]
        correct+=predict_class.eq(targets).sum().item()
        for i in range(len(targets)):
            confusion_matrix[int(targets[i])][int(predict_class[i])]+=1
    acc=100.*correct/len(loader_test.dataset)
print(f'acc:{acc:.2f}%')
# normalize confusion_matrix
confusion_matrix=confusion_matrix/confusion_matrix.sum(axis=1).reshape(num_class,1)
```

3. Data Preprocessing

a) How you preprocessed your data

在 getitem 裡面，我針對檔案遺失或是毀損做了 try exception，讓其不要因為檔案的問題造成整個程式 shut down。

```
def __getitem__(self, index):

    single_img_name = os.path.join(self.root,self.img_name[index]+'.jpeg')
    img=Image.open(single_img_name)
    label=self.label[index]
    try:
        return self.transform(img), label
    except:
        single_img_name = os.path.join(self.root,self.img_name[index+1]+'.jpeg')
        img=Image.open(single_img_name)
        label=self.label[index+1]
        return self.transform(img), label
```

b) What makes your method special

原始的圖片除中間部分其餘部分皆是黑色，因此我先用 CenterCrop 抓取圖片正中間的部分並將原始圖片縮成正方形(以短邊為邊長)，之後再將原始圖片縮成 512*512 的大小，在每一次訓練時，我都會隨機上下翻轉進行訓練，提高模型的隨機性。

```
def transform(self,img):
    shortSize=min(img.size)
    script = transforms.Compose([
        transforms.RandomHorizontalFlip(),
        transforms.CenterCrop((shortSize,shortSize)),
        transforms.Resize((512, 512)),
        transforms.ToTensor()
    ])

    return script(img)
```

我嘗試了 RandomHorizontalFlip 在 pretrined 以及 unpretrained 的 model 中，其中利用 HorizontalFlip 圖片的 Pretrained model 可以達到比較好的結果，我們可以在接下來的實驗結果中看到。

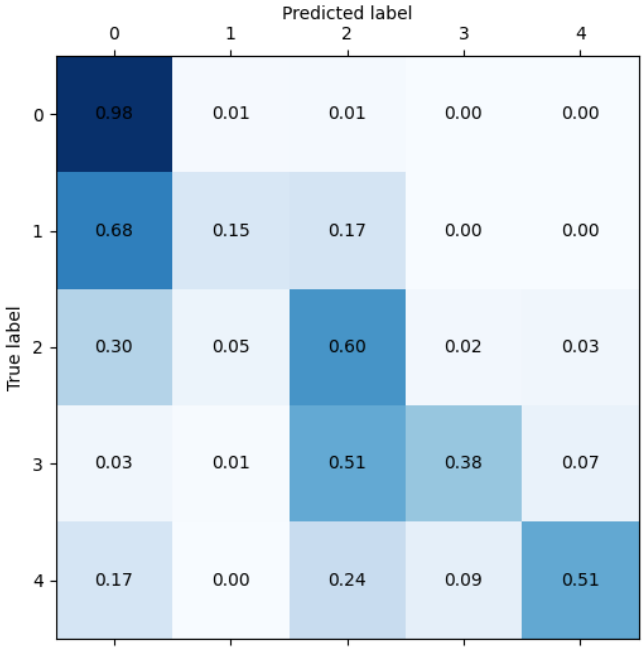
4. Experiment Results:

a) The highest Test Result

- ResNet18:

  Testing Accuracy: 84.21%

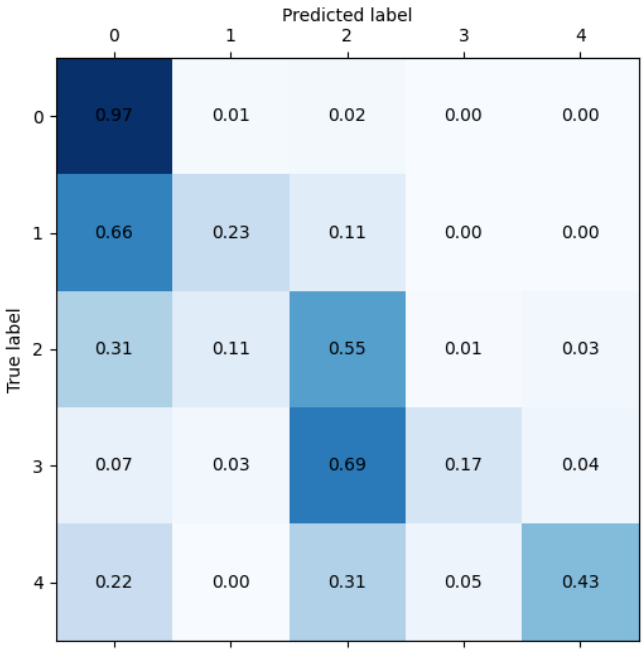
  ResNet confusion Matrix



- ResNet50

  Testing Accuracy: 82.88 %

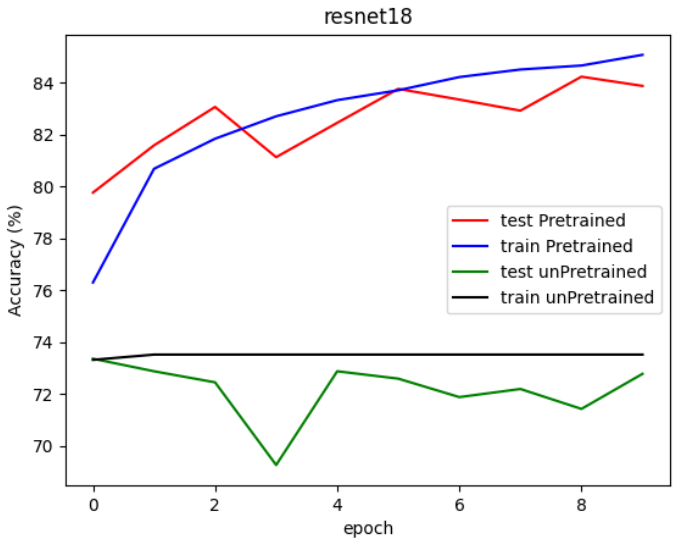
  Confusion Matrix



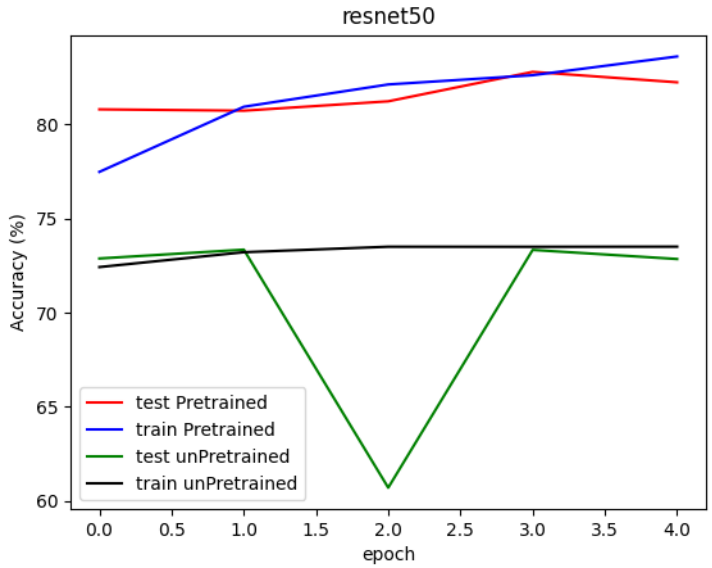
b) Comparison figures:

在比較圖中，我們比較了 resnet 有 pretrained 以及沒有 pretrined 的結果在比較圖中，黑線與綠線為 unpretrained 的結果，而藍線與紅線為有 pretrained 的結果。

- ResNet18

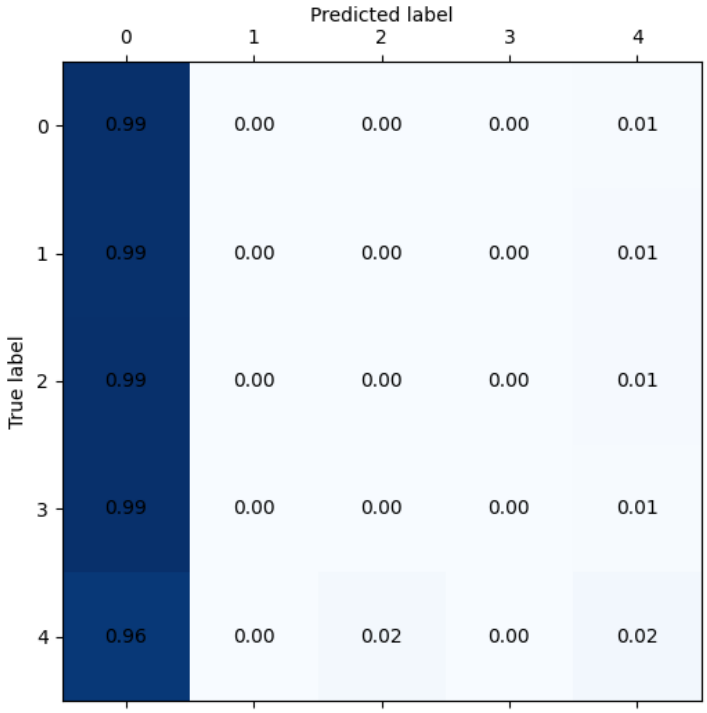


- ResNet50



5. Discussion:

本次實驗使用的是 unbalanced 的 dataset，雖然測試資料集以及訓練資料集的分布長的差不多，但是 label 1 對應的圖片高達 70%，在 unpretrained model 裡面，模型會自動將所有圖片分類至第一類，反而是錯誤的結果。儘管進行了 HorizontalFlip 也無法有效分類這些圖片。下圖為 unpretrained 的 confusion matrix，也可以看到全部的圖片都被分到第一類。


Predicted label
0
1
2
3
4
True label
0
1
2
3
4
0.99
0.00
0.00
0.00
0.01
0.99
0.00
0.00
0.00
0.01
0.99
0.00
0.00
0.00
0.01
0.99
0.00
0.00
0.00
0.01
0.96
0.00
0.02
0.00
0.02