DLP HW4

數據所 311554019 宋沛潔

1. Introduction:

這次的作業是使用 pytorch 來設計神經網路架構，對於糖尿病所引發視網膜病變的照片進行嚴重分類，類別共有五種(0:最輕微~4:最嚴重)。神經網路模型規定要實作 Resnet18 & Resnet50，其中分別對 pretrained model 及沒有 pretrained model 的結果進行比較，最後要呈現 confusion matrix 及 training/testing accuracy 的結果。

2. Experiment setups

A. The details of your model (ResNet):

ResNet18 & ResNet50 是多了 residual block，去防止模型在神經網路傳遞時梯度消失或爆炸的問題。ResNet18 包含 basic block、ResNet50 包含 Bottleneck block。

使用 torchvision.models 中寫好的 ResNet18 & ResNet50 架構，版本是 v1.5，至於 pretrained model 的部分，torchvision 已經設計好，可以直接將 model pretrained 參數設成 True。不管是有 pretrained model 還是沒有，在訓練過程都會更新 weight。那在最後一層的後面加了 fully connection，主要的用意是要將照片分類。類別總共是 5 類，所以最後的輸出會是 5。ResNet 架構中 v1.0 與 v1.5 的差別是: v1.5 bottleneck blocks 第一個 convolution 1x1 的 stride 由 2 改 1，第二個 convolution 3x3 stride 由 1 改 2。Downsampling 的工作由第一個 convolution 1x1 到第二個 convolution 3x3。那這個好處使高維特徵保留，第二個 convolution 3x3 可以得到較高維度的特徵抽取，訓練穩定，讓模型的泛化能力變高，不會使模型的特徵表示能力下降。

```
class ResNet50(torch.nn.Module):
    def __init__(self,numclass, pretrained):

        super(ResNet50,self).__init__()
        self.model=resnet50(pretrained=pretrained)
        self.model.fc = torch.nn.Sequential(
                torch.nn.Flatten(),
                torch.nn.Linear(in_features=self.model.fc.in_features, out_features=100),
                torch.nn.ReLU(inplace=True),
                torch.nn.Linear(in_features=100, out_features=5)
            )
    def forward(self, X):
        out = self.model(X)
        return out
```

B. The details of your Dataloader:

RetinopathyLoader 的部分是使用助教給的，將照片進行轉換後，在將照片裝成一個一個 batch。Train data 會進行 shuffle。

```
train_dataset = RetinopathyLoader("./data/new_train", "train")
test_dataset = RetinopathyLoader("./data/new_test", "test")
train_loader = DataLoader(train_dataset, batch_size = batch_size, shuffle=True, num_workers=8)
test_loader = DataLoader(test_dataset, batch_size = batch_size, shuffle=False, num_workers=8)
```

在 RetinopathyLoader 中，__init__ function 中，root 會先取存放資料的位置。Mode 有兩種，train data 或是 test data。getData function 會透過 csv 去讀取 train/teat 對應資料。__len__會回傳 dataset 的大小。在__getitem__中，首先將 self.img_name 取取圖像路徑。再來，從 self.label 獲取 ground truth label 及進行 Transform 轉換。最後回傳處理後的 image 和 label。

```
def getData(mode):
    if mode == 'train':
        img = pd.read_csv('train_img.csv', header=None)
        label = pd.read_csv('train_label.csv', header=None)
        return np.squeeze(img.values), np.squeeze(label.values)
    else:
        img = pd.read_csv('test_img.csv', header=None)
        label = pd.read_csv('test_label.csv', header=None)
        return np.squeeze(img.values), np.squeeze(label.values)


class RetinopathyLoader(data.Dataset):
    def __init__(self, root, mode):
        """
        Args:
            root (string): Root path of the dataset.
            mode : Indicate procedure status(training or testing)

            self.img_name (string list): String list that store all image names.
            self.label (int or float list): Numerical list that store all ground truth label values.
        """
        self.root = root
        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 __getitem__(self, index):
        img = Image.open(os.path.join(self.root, self.img_name[index]) + '.jpeg')
        label = self.label[index]
        img = transformer(self.mode, img)

        return img, label
```

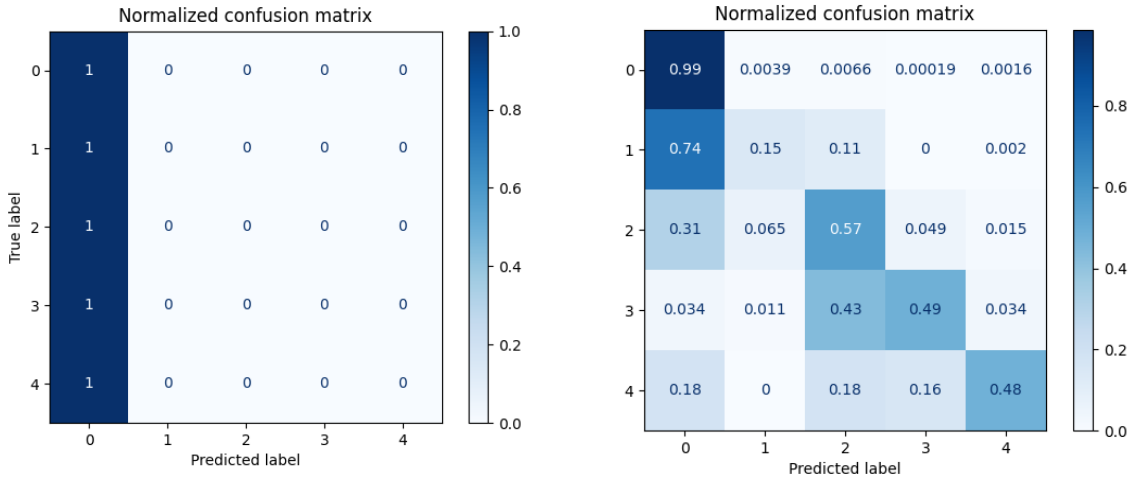
C. Describing your evaluation through the confusion matrix

利用 sklearn.metrics 中的 confusion_matrix 及 ConfusionMatrixDisplay 來呈現 confusion matrix。將 testing predict 的結果與 ground truth label 來進行比較。confusion matrix 中，x 軸代表 predict label，y 軸代表 ground truth label。進行 row normalization，顏色越淺代表機率越低。

```python
def plot_confusion(ground_truth, prediction, model):
    matrix = confusion_matrix(
        ground_truth,
        prediction,
        labels= [0, 1, 2, 3, 4],
        normalize='true',
    )
    disp = ConfusionMatrixDisplay(confusion_matrix = matrix,
                                  display_labels = [0, 1, 2, 3, 4],)
    disp.plot(cmap=plt.cm.Blues)

    plt.title('Normalized confusion matrix')
    plt.xlabel('Predicted label')
    plt.ylabel('True label')
    plt.savefig(model + ' Normalized confusion matrix.png')
```

下方的圖中，左邊是沒有 pretrained model 的 confusion matrix 結果，右邊是 pretrained model 的 confusion matrix 結果。可以看到沒有 pretrained model 會將照片全部歸類於 0，表示模型的泛化能力不好，都猜同一種類別。但是 pretrained model 的 confusion matrix 結果看起來不錯。



3. Data Preprocessing

A. How you preprocessed your data?

使用 torchvision 的 transformer，torchvision 包含了一些對於圖像進行預處理和數據增強的功能。首先我選擇將所有照片 resize 成 512x512，這樣轉向量後每一張照片的長度才會一致。至於 training data 對照片進行隨機選轉 180 度、每張照片都有 0.5 機率被水平翻轉、使用隨機 crop，來增加照片的多樣性，讓模型預測效果變好，可以提高模型的泛化能力。

```python
def transformer(mode, img):
    if mode == "train":
        transformer = transforms.Compose([
                                transforms.RandomRotation(180),
                                transforms.RandomHorizontalFlip(p=0.5),
                                transforms.Resize([512,512]),
                                transforms.RandomCrop(512, padding=16, padding_mode='reflect'),
                                transforms.ToTensor(),
                            ])

        return(transformer(img))
    else:
        transformer = transforms.Compose([
                                    transforms.Resize([512,512]),
                                    transforms.ToTensor(),
                                    ])
        return(transformer(img))
```

B. What makes your method special?

經過將照片進行隨機選轉 180 度、每張照片都有 0.5 機率被水平翻轉、使用隨機 crop，來增加照片的多樣性。原本有使用多種轉換，但發現一次用太多，效果不好。

4. Experimental results

A. The highest testing accuracy

在最高的 accuracy 是用 Resnet50，參數為 epoch = 8、momentum = 0.9、Learning rate = 1e-3、weight decay = 5e-4、batch size = 16。

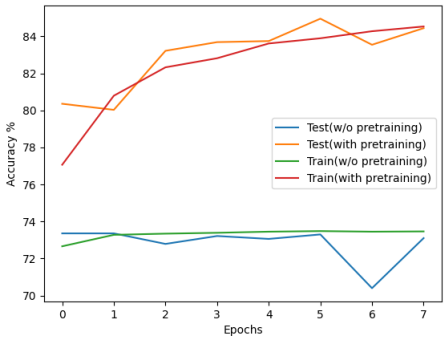
```
100%| 440/440 [01:43<00:00,  4.25it/s]
Test's accuracy is: 84.258%
```

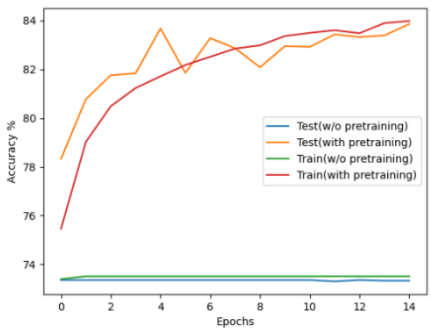
B. Comparison figures

下方的圖中，左邊是 Resnet50 model 的結果，右邊是 Resnet18 model 的結果。Resnet50 在 epoch = 8 的時候就可以達到比 Resnet18 在 epoch = 15 還要好的結果。

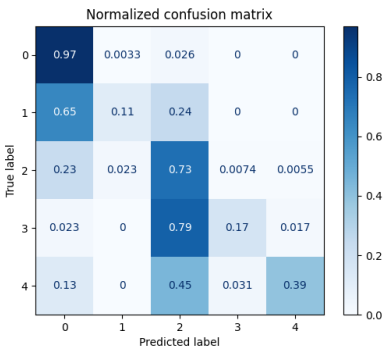
Resnet50: 參數為 epoch = 8、momentum = 0.9、Learning rate = 1e-3、weight decay = 5e-4、batch size = 16

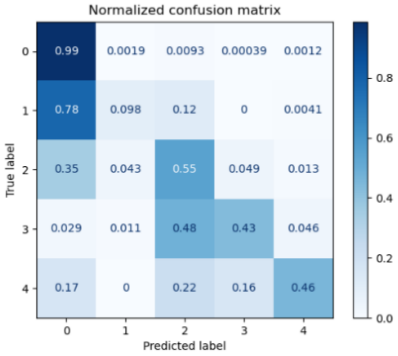
Resnet18: 參數為 epoch = 15、momentum = 0.9、Learning rate = 1e-3、weight decay = 5e-4、batch size = 8



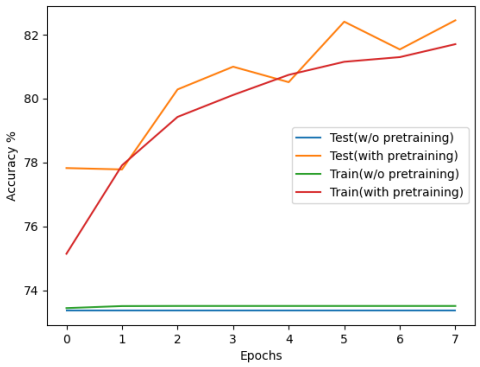


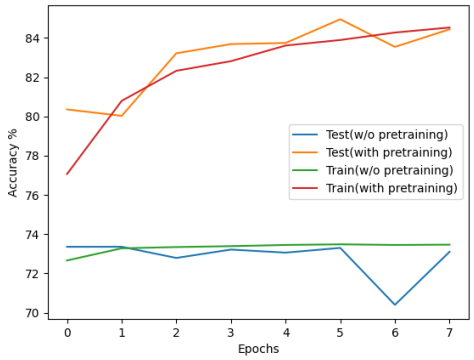
下方的圖中，左邊是 Resnet50 model 最好的 confusion matrix 結果，右邊是 Resnet18 model 最好的 confusion matrix 結果。



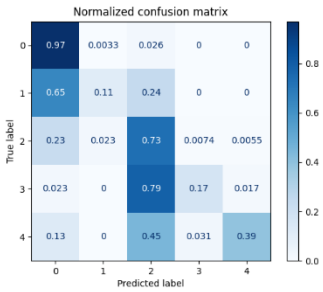


5. Discussion





1. 上圖左方是 Resnet50 batch size=4; 右方是 Resnet50 batch size=16。對於 batch size 的大小，只要是 batch size 越大，收斂較快，而且準確度就會越好。
2. 對於照片 transformer 的處理中，如果套用太多效果不好。
3. 有另外加入 scheduler 去調整 learning rate，但也是效果沒有比較好。



4. 對於最好的 confusion matrix 結果來看，除了 0 跟 2 的預測結果有達 70%以上，剩餘的 label 結果都相當不好，可見泛化性還可以再提升。