Deep Learning and Practice Lab 3

311605011

黃品振

1. Introduction:

這次的作業是要用 ResNet 來分類視網膜照片，分類出因為糖尿病造成的視網膜病變程度。另外，還要利用自訂的 dataloader 來讀取資料，還有用 confusion matrix 來評估模型的好壞。

2. Experiment set up:

A. Detail of my model:

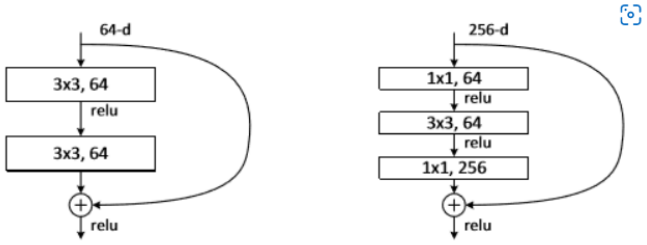
- ResNet:
  隨著神經網路層數的加深，degradation problem 可能會出現，可能是因為 gradient vanishing 造成資訊無法向前傳達，因此 ResNet 透過 shortcut connection 的方式，來改善這個問題。

不同深度 ResNet 架構:

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

從上圖可以看到 ResNet 內會有很多重複的 Convolutional layer，因此在實作的時候會利用不同的 block 去堆疊而成。

下圖是 ResNet18 的 basic block，以及 ResNet50 的 bottleneck block 架構。



以下是我的 model 架構:

```python
def Resnet(type = 18, class_num = 5, pretrained = False):
    if type ==18:
        model = models.resnet18(pretrained)
        model.fc = nn.Linear(model.fc.in_features, class_num)

    if type ==50:
        model = models.resnet50(pretrained)
        model.fc = nn.Linear(model.fc.in_features, class_num)

    return model
```

我直接利用 PyTorch 內的套件建立 ResNet18 和 ResNet50

B. The details of dataloader:

```python
def __init__(self, root, mode, augmentation = NONE):
    """
    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
    trans = []
    #augmentation 是要擴增資料種類的list
    if augmentation == True:
        trans += augmentation
    trans.append(transforms.ToTensor())
    self.transform = transforms.Compose(trans)
    print("> Found %d images..." % (len(self.img_name)))
```

先將助教給的檔案讀取進來，然後將想要擴增的種類打加入在 augmentation 的 list 裡面，然後建立 transform 將資料轉換成 pytorch 讀取的方式。

```
def __getitem__(self, index):
    path = os.path.join(self.root, self.img_name[index] + '.jpeg')
    label = self.label[index]
    img = PIL.Image.open(path)
    img = self.transform(img)
    return img, label
```

__getitem__會根據取得的 index 回傳對應的圖片以及其 label。

C. Describing evaluation through the confusion matrix

```
def plot_confusion_matrix(y_true, y_pred):
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)
    cf_matrix = confusion_matrix(y_true, y_pred, normalize = 'true')
    class_names = ['No DR', 'Mild', 'Moderate', 'severe', 'Proliferative DR']
    df_cm = pd.DataFrame(cf_matrix, class_names, class_names)
    plt.figure(figsize = (9,6))
    sns.heatmap(df_cm, annot=True, fmt=".3g", cmap=plt.cm.Blues)
    plt.xlabel("predicted label")
    plt.ylabel("True lable")
    plt.title('Normolized confusion matrix')
    plt.show()
```

上圖是我畫 confusion matrix 的 code，我利用 sklearns 的函式庫來製作 confusion matrix，再利用 seaborn 繪製熱力圖，將 confusion matrix 繪製出來。

3. Experimental results:

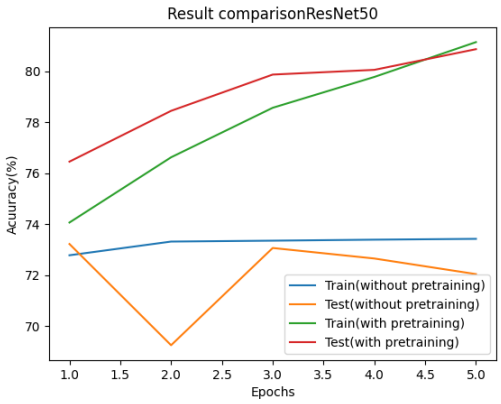
A. The highest testing accuracy:

- Screenshot

```
Highest testing accuracy(without pretraining):  73.22%
Highest testing accuracy(with pretraining):  80.87%
```

上圖是我利用 ResNet50 跑出來的最佳測試結果，參數如下:
Epoch=5、learning rate=0.001、batch size=4

- Anything want to present:

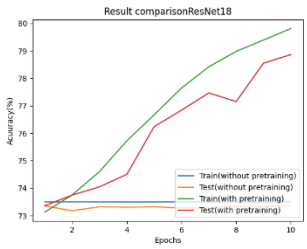
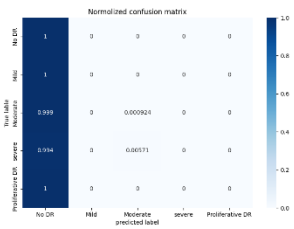
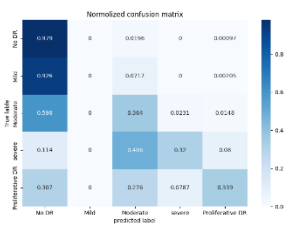


從上圖可以看出，有 pretrain 的訓練效果遠比沒有 pretrain 得來的好，推測是因為 pretrain 過的 model 可能有學到一些其他特徵的辨識方法，可以幫助辨識視網膜的病變，而自己的資料因為長得比較相似所以學習成效不佳。

B. Comparison figures:

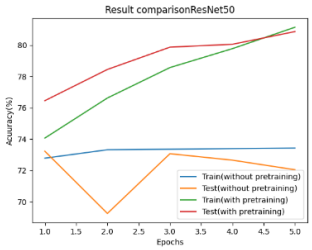
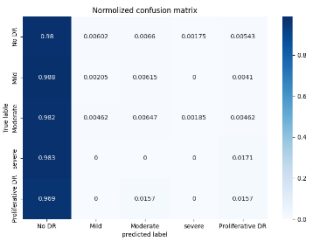
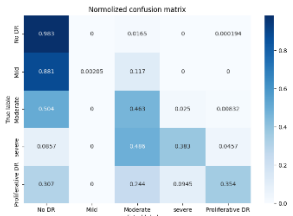
- ResNet18:

以下對比圖的參數為:epoch =10、batch size=32、learning rate=0.001

Comparison figure	Without pretraining (cf matrix)	With pretraining (cf matrix)
		

- ResNet50:

  以下對比圖的參數為:epoch =5、batch size=4、learning rate=0.001

Comparison figure	Without pretraining (cf matrix)	With pretraining (cf matrix)
		

4. Discussion:

ResNet18(epoch=10、learning rate=0.001、batch size=32)測試最高準確率:

```
Highest testing accuracy(without pretraining):  73.37%
Highest testing accuracy(with pretraining):  78.65%
```

ResNet50(epoch=5、learning rate=0.001、batch size=4)測試最高準確率:

```
Highest testing accuracy(without pretraining):  73.22%
Highest testing accuracy(with pretraining):  80.87%
```

上圖發現:ResNet50 的訓練效果在有 pretraining 的狀況下，比沒有 pretraining 的效果好，推測是因為學到一些圖像的通用特徵，因此可以讓視網膜病變的辨別更有效。而 ResNet50 與 ResNet18 在同樣有 pretraining 的狀況下，ResNet50 表現比較好的原因推測是因為神經網路的層數較多，或者是 ResNet18 的 epoch 太少導致。