1. Introduction

   這次的 Lab 使用 Residual Network 來達成分類視網膜照片的目的，分類出病變嚴重程度，實作 ResNet18 以及 ResNet50 架構並從 pretrain model 載入參數，並且比較有使用 pretrain model 以及沒有使用 pretrain model 的差異，並且要根據助教提供之 testloader 為基礎修改程式碼處理圖像資料，最後要計算 confusion matrix 並且 plot 出結果

2. Experiment setup

   A. The details of your model(ResNet)

      a. Basic block(按照 spec 上面的描述 implement)

```python
class BasicBlock(nn.Module):
    expansion = 1

    def __init__(self, in_channels, out_channels, stride = 1, down_sample = None):
        super(BasicBlock, self).__init__()

        self.activation = nn.ReLU(inplace = True)
        self.block = nn.Sequential(nn.Conv2d(in_channels, out_channels, kernel_size = 3, stride = stride, padding =1 , bias = False),
                                   nn.BatchNorm2d(out_channels), self.activation,
                                   nn.Conv2d(out_channels, out_channels,kernel_size = 3,padding =1 ,bias = False),
                                   nn.BatchNorm2d(out_channels),
                                   )
        self.down_sample = down_sample
    def forward(self, inputs):
        residual = inputs
        outputs = self.block(inputs)
        if self.down_sample is not None:
            residual = self.down_sample(inputs)

        outputs = self.activation(outputs + residual)

        return outputs
```

      b. BottleneckBlock(按照 spec 上面的描述 implement)

```python
class BottleneckBlock(nn.Module):
    expansion = 4

    def __init__(self, in_channels, out_channels, stride, down_sample = None):
        super(BottleneckBlock, self).__init__()

        external_channels = out_channels * self.expansion
        self.activation = nn.ReLU(inplace=True)
        self.block = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=1,bias=False),
            nn.BatchNorm2d(out_channels),
            self.activation,
            nn.Conv2d(out_channels ,out_channels, kernel_size=3, stride=stride, padding=1, bias=False),
            nn.BatchNorm2d(out_channels),
            self.activation,
            nn.Conv2d(out_channels, external_channels, kernel_size=1,bias=False),
            nn.BatchNorm2d(external_channels),
        )
        self.down_sample = down_sample

    def forward(self, inputs):
        residual = inputs
        outputs = self.block(inputs)
        if self.down_sample is not None:
            residual = self.down_sample(inputs)

        outputs = self.activation(outputs + residual)

        return outputs
```

      其中 down sample 為了處理 ResNet 中跨 convolutional layer 時 channel 數變化而需要使用的，要讓 input output channel 數以及 height width

相等

c. ResNet(in code line 67~119)

使用 basic block 或使用 bottleneck block 來組成 convolutional layer2~5, 如果是 pretrained model, 會從 pytorch 載入 pretrained ResNet 來建構, 只有 fully connected layer 是非 pretrained, 目前使用兩層, make_layer 則是負責建立 convolutional layer.

B. The details of your DataLoader

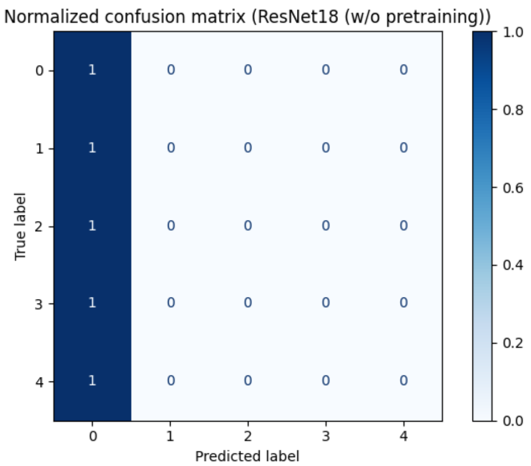
```
def __getitem__(self, index):
    """something you should implement here"""
    image_path = os.path.join(self.root, f'{self.img_name[index]}.jpeg')
    label = self.label[index]
    img = PIL.Image.open(fp = image_path)
    trans = [torchvision.transforms.ToTensor()]
    transform = torchvision.transforms.Compose(trans)
    img = transform(img)
```

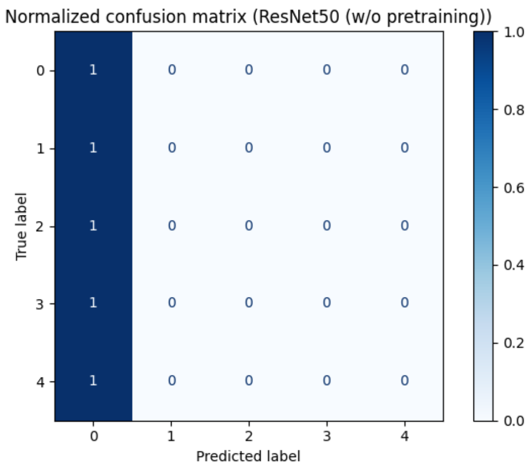
```
return img, label
```

先使用 PIL.Image.open 讀取圖片

再來使用 torchvision 中的 method 轉換圖片

C. Describing your evaluation through the confusion matrix




由 without pretraining 的 confusion matrix 可以看出 without pretraining 的都會將所有圖片歸為 label 0，可見 model 不夠 general。

所以會使用 with pretrain model 的實驗結果來找最高的 testing accuracy

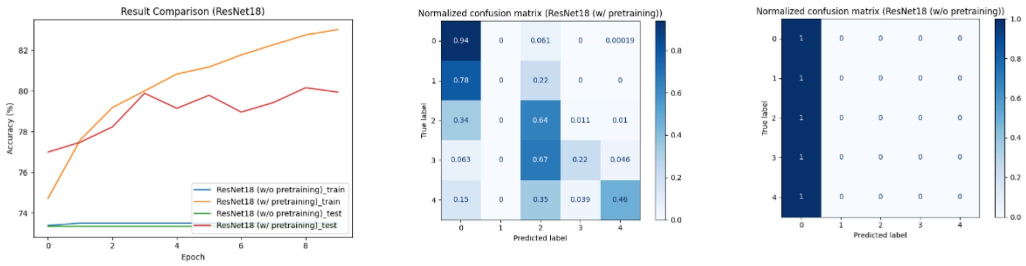
3. Experimental result

A. The highest testing accuracy

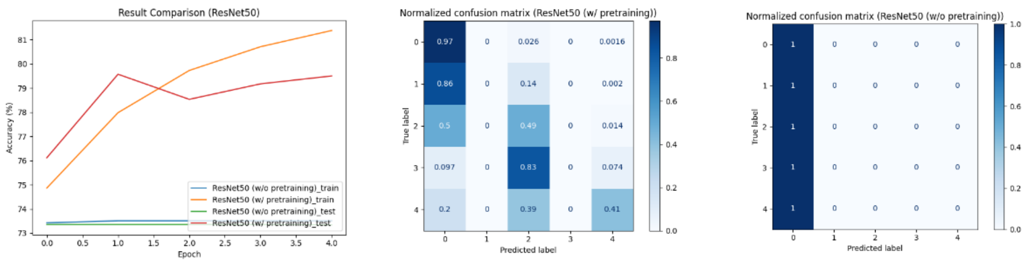
```
ResNet18 (w/o pretraining)_test:   73.35 %
ResNet18 (w/ pretraining)_test:    80.16 %
```

B. Comparison figure

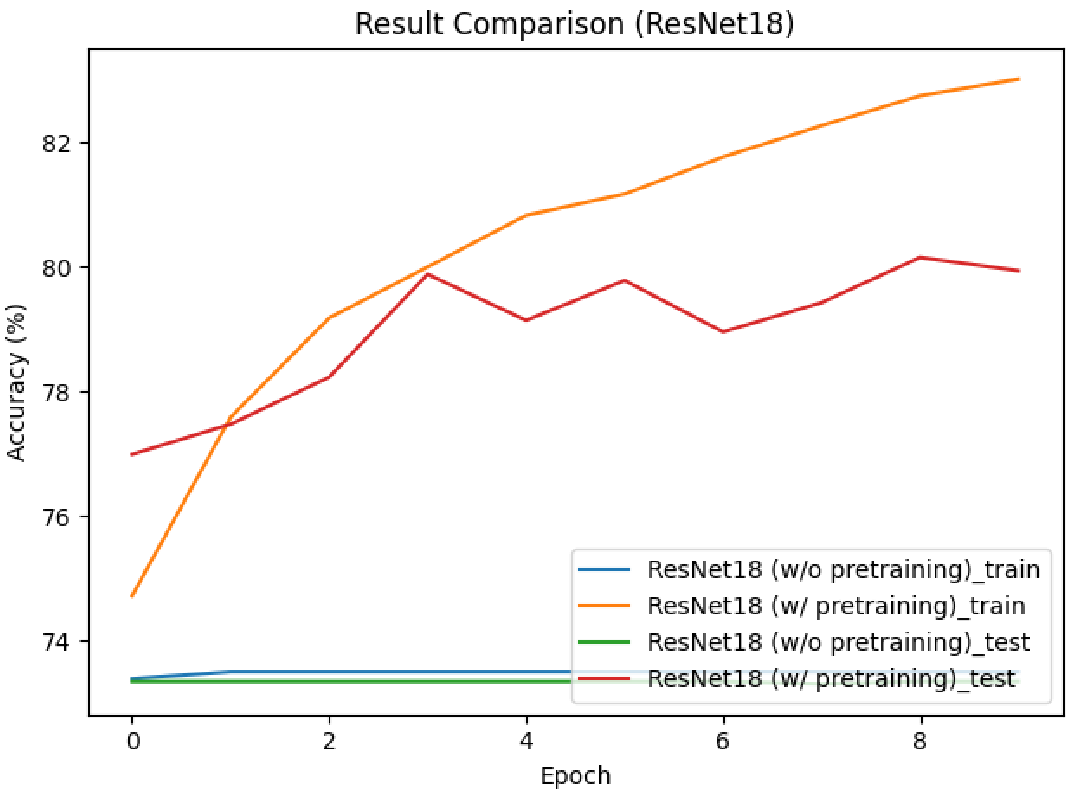
(1) ResNet18



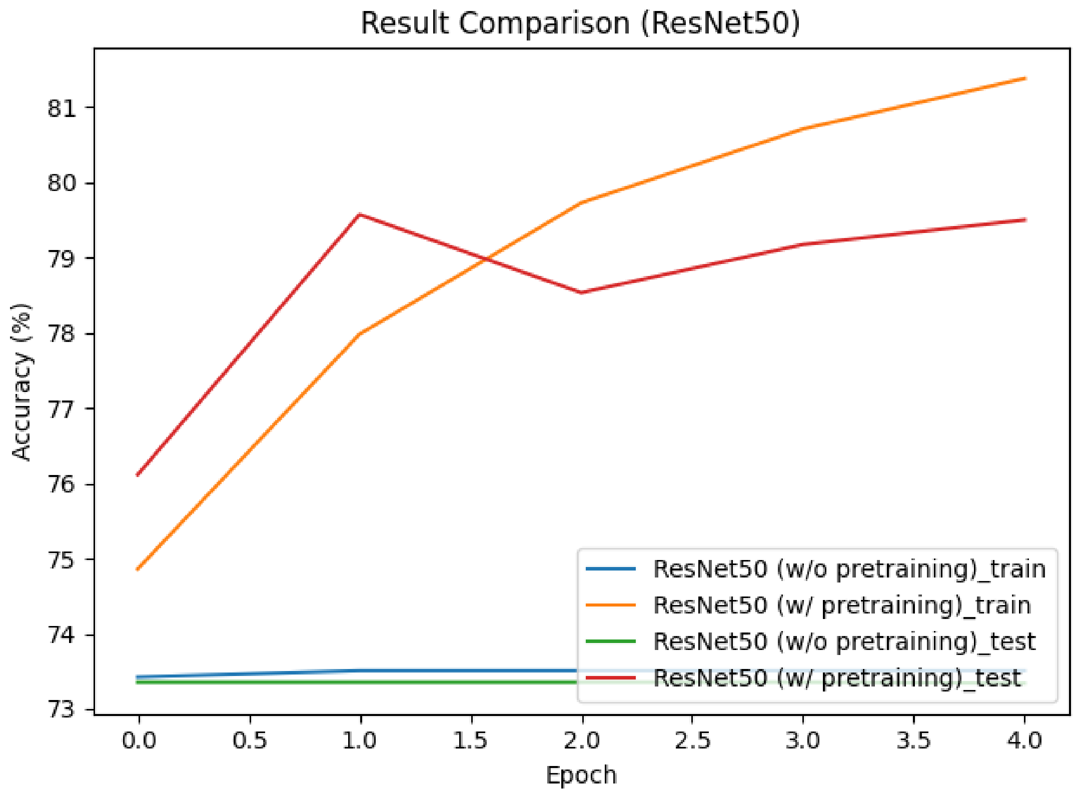

(2) ResNet50




4. Discussion



一開始做 ResNet18 的時候設 batch size = 4 w/ pretrain model 只能得到 76 左右的 testing accuracy，後來將 batch size 改成 12 才能得到 80 左右的 accuarcy，不過受限於記憶體，很難再調升至 12 以上的 batch size。以上結果為 batch size =12 , lr = 1e-3, epochs = 10, opt = sgd, momentum = 0.9, weight_decay=5e-4, Loss = CrossEntropy




後來在做 ResNet50 的時候用 batch size =12 epoch = 5 記憶體就佔到 8G 非常可觀，比較難繼續增加下去，而 hyperparameter 除了 batch size 都符合預設，或許 epoch 再增加會有更好的效果，畢竟只全部看過五次還是有點少