lab3

資工碩一 0856105 吳承翰

1. Introduction

分別使用有pretrain weight/無pretrain weight的Resnet18/Resnet50訓練黃斑部病變的classification problem，有五種class label:0~4，表示病變的嚴重程度。

2. Experiment Setup

(A)Details of my model

用torchvision import resnet18/resnet50。
如果為需要訓練model with pretrain weight就把pretrained參數設True，並把最後一層layer的output feature設為number of class:5。
我會先feature extraction model(只訓練最後一層layer)幾個epoch，再finetuning整個model(訓練所有layer)數個epoch。

```
class ResNet50(nn.Module):
    def __init__(self,num_class,pretrained=False):
        """
        Args:
            num_class: #target class
            pretrained:
                True: the model will have pretrained weights, and only the last layer's 'requires_gr
ad' is True(trainable)
                False: random initialize weights, and all layer's 'require_grad' is True
        """
        super(ResNet50,self).__init__()
        self.model=models.resnet50(pretrained=pretrained)
        if pretrained:
            for param in self.model.parameters():
                param.requires_grad=False
        num_neurons=self.model.fc.in_features
        self.model.fc=nn.Linear(num_neurons,num_class)

    def forward(self,X):
        out=self.model(X)
        return out
```

(B)The details of my Dataloader

要實作自己的dataset，再把dataset放到Dataloader裡。
要定義好__getitem__()，Dataloader才可以依照指定的mini-batch取得input與target。
每一張圖片可以透過transforms來實現data augmentation或是normalization。
我發現無論有沒有normalize，正確率都幾乎一樣。

```python
class RetinopathyDataSet(Dataset):
    def __init__(self, img_path, mode):
        """
        Args:
            img_path: Root path of the dataset.
            mode: training/testing

            self.img_names (string list): String list that store all image names.
            self.labels (int or float list): Numerical list that store all ground truth label value
s.
        """
        self.img_path = img_path
        self.mode = mode

        self.img_names=np.squeeze(pd.read_csv('train_img.csv' if mode=='train' else 'test_img.csv').
values)
        self.labels=np.squeeze(pd.read_csv('train_label.csv' if mode=='train' else 'test_label.cs
v').values)
        assert len(self.img_names)==len(self.labels),'length not the same'
        self.data_len=len(self.img_names)

        self.transformations=transforms.Compose([transforms.RandomHorizontalFlip(),transforms.Random
VerticalFlip(),transforms.ToTensor(),
                                                 transforms.Normalize((0.3749,0.2602,0.1857),(0.252
6, 0.1780, 0.1291))])
        print(f'>> Found {self.data_len} images...')

    def __len__(self):
        return self.data_len

    def __getitem__(self, index):
        single_img_name=os.path.join(self.img_path,self.img_names[index]+'.jpeg')
        single_img=Image.open(single_img_name)  # read an PIL image
        img=self.transformations(single_img)
        label=self.labels[index]

        return img, label
```

(C)Describe my evaluation through the confusion matrix

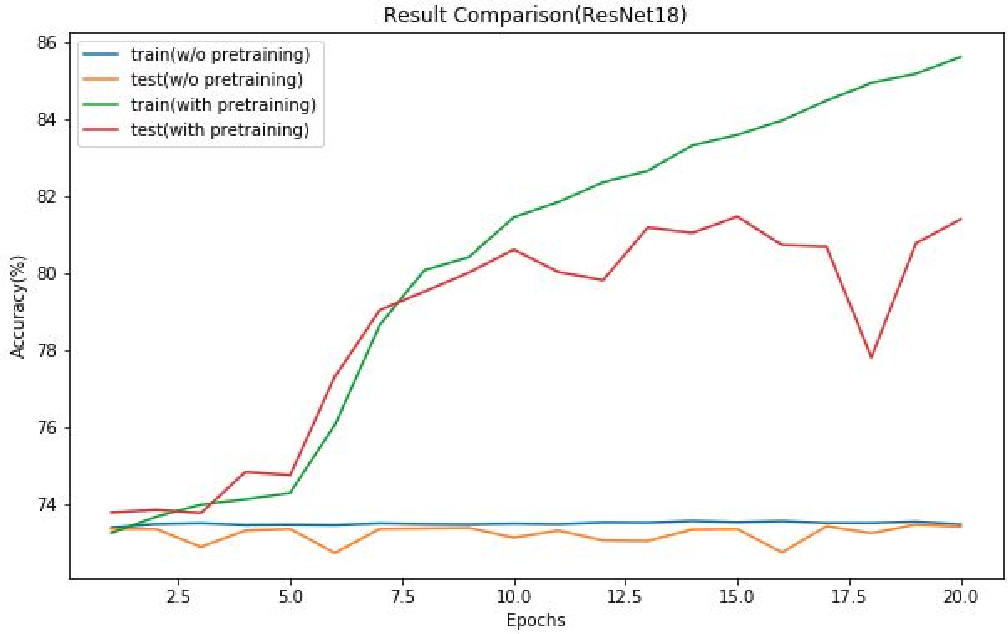
用一個5*5的矩陣在testing data時統計數量，最後再依照每一列做normalize。

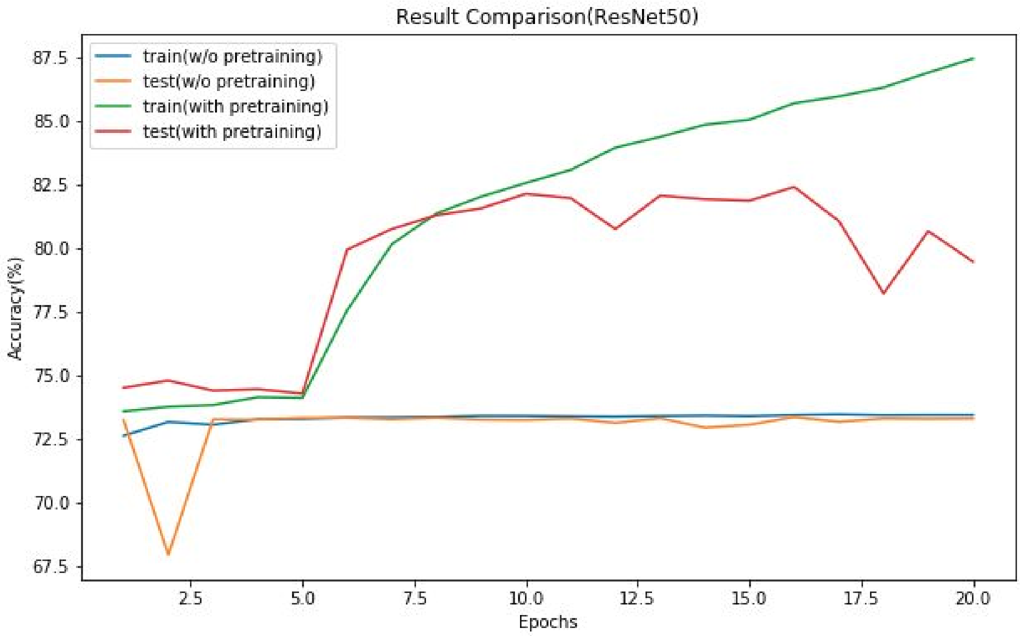
```python
confusion_matrix=np.zeros((num_class,num_class))

with torch.set_grad_enabled(False):
    model.eval()
    correct=0
    for images,targets in loader_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)

# normalize confusion_matrix
confusion_matrix=confusion_matrix/confusion_matrix.sum(axis=1).reshape(num_class,1)
```

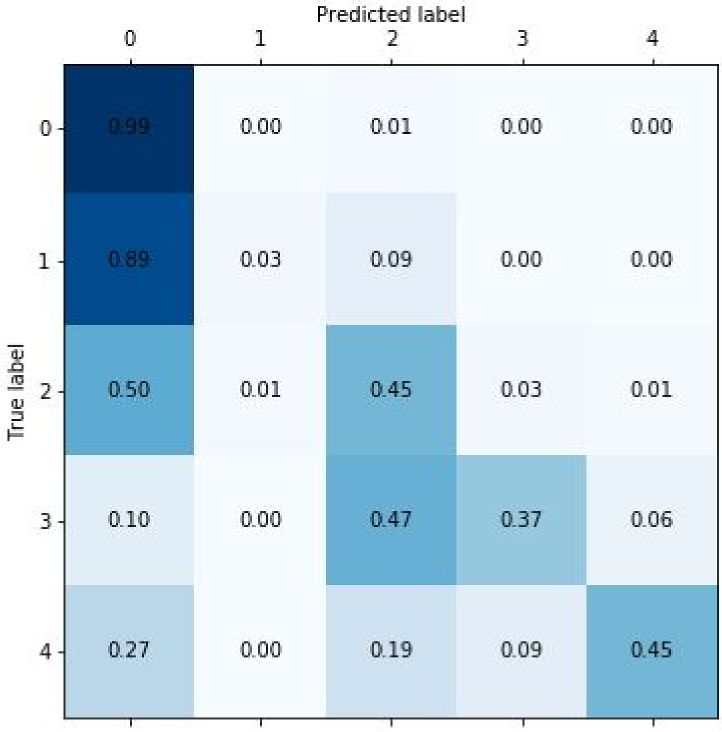
3. Experimental results



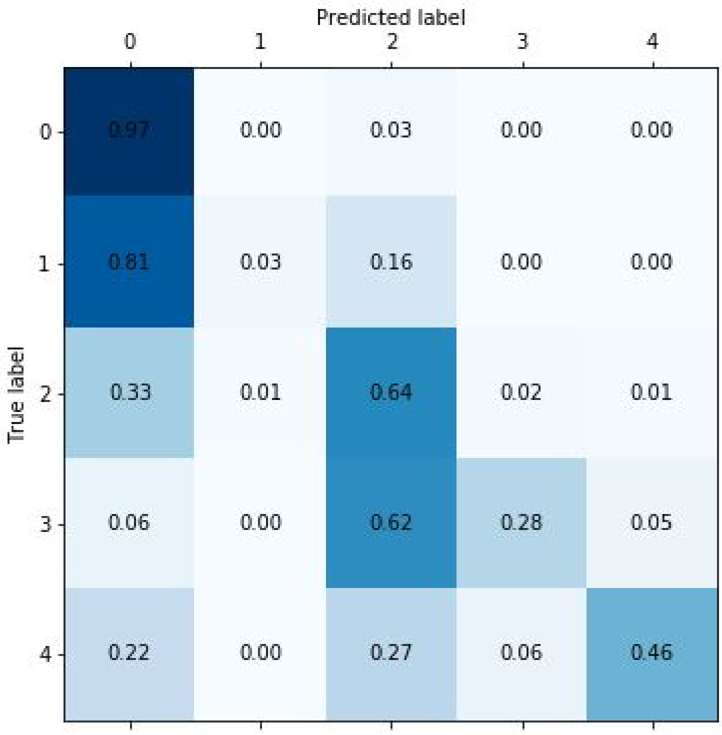


acc可以達到82.4%左右

ResNet18 with pretrained weights 的 Confusion matrix:



ResNet50 with pretrained weights 的 Confusion matrix:



4.Discuss

雖然正確率82%，但是這並不實用

因為這種imbalance data只要不論看到什麼都一律輸出predict class=0的話正確率都有70%。

正確作法是要加入weighted loss,

我也train了一個有weighted loss的model,

(weight=[1.0,10.565217391304348,4.906175771971497,29.591690544412607,35.55077452667814])

結果如下,

可以發現confusion matrix比之前的好很多，acc可以到80.7%。