HW4 Report Diabetic Retinopathy Detection

tags: `DL and Practice`

電控碩一 黃柏叡 309512074

Report

Introduction

這次實驗，我使用ResNet來訓練黃斑部病變的classification problem，有五種label(0~4)，並根據病變的嚴重程度做分類，這次lab的主要目標:

- Implement ResNet18, ResNet50
- Implement ResNet18, ResNet50 with pretrained model
- Compare and visualize the accuracy trend
- Custom DataLoader
- Implement Confusion matrix

Experimental Setup

A. Detail of your model(ResNet)

用torchvision import ResNet18跟ResNet50，並將最後一層的fully connected layer的output feature設為5(number of class)

```
class ResNet18(nn.Module):
    def __init__(self, num_class, pretrained):
        super(ResNet18,self).__init__()
        self.model = models.resnet18(pretrained=pretrained)
        fc_num_neurons = self.model.fc.in_features
        self.model.fc = nn.Linear(fc_num_neurons, num_class)

    def forward(self, x):
        x = self.model(x)
        return x

class ResNet50(nn.Module):
    def __init__(self, num_class, pretrained):
        super(ResNet50,self).__init__()
        self.model = models.resnet50(pretrained=pretrained)
        fc_num_neurons = self.model.fc.in_features
        self.model.fc = nn.Linear(fc_num_neurons, num_class)

    def forward(self, x):
```

```
        x = self.model(x)
        return x
```

B. The details of your Dataloader

定義自己的Dataset，並定義好**getitem**(如何取得input及label)，Dataloader才能依照指定的mini-batch取得input與label

```
def __getitem__(self, index):
    single_img_name = os.path.join(self.img_path, self.img_name[index]+ '.jpeg')
    single_img =  Image.open(single_img_name)
    img = self.transformation(single_img)
    label = self.label[index]
    return img, label
```

也有對img做transformation，因為訓練集的照片差異並不是很大，所以對資料集做RandomFlip及Normalization來避免overfitting problem，同時也將資料轉成Tensor型態，以放入GPU加速運算

```
self.transformation = transforms.Compose([transforms.RandomHorizontalFlip(p=0.5),trans
                                    transforms.Normalize((0.37, 0.26, 0.18),(0.25, 0
```

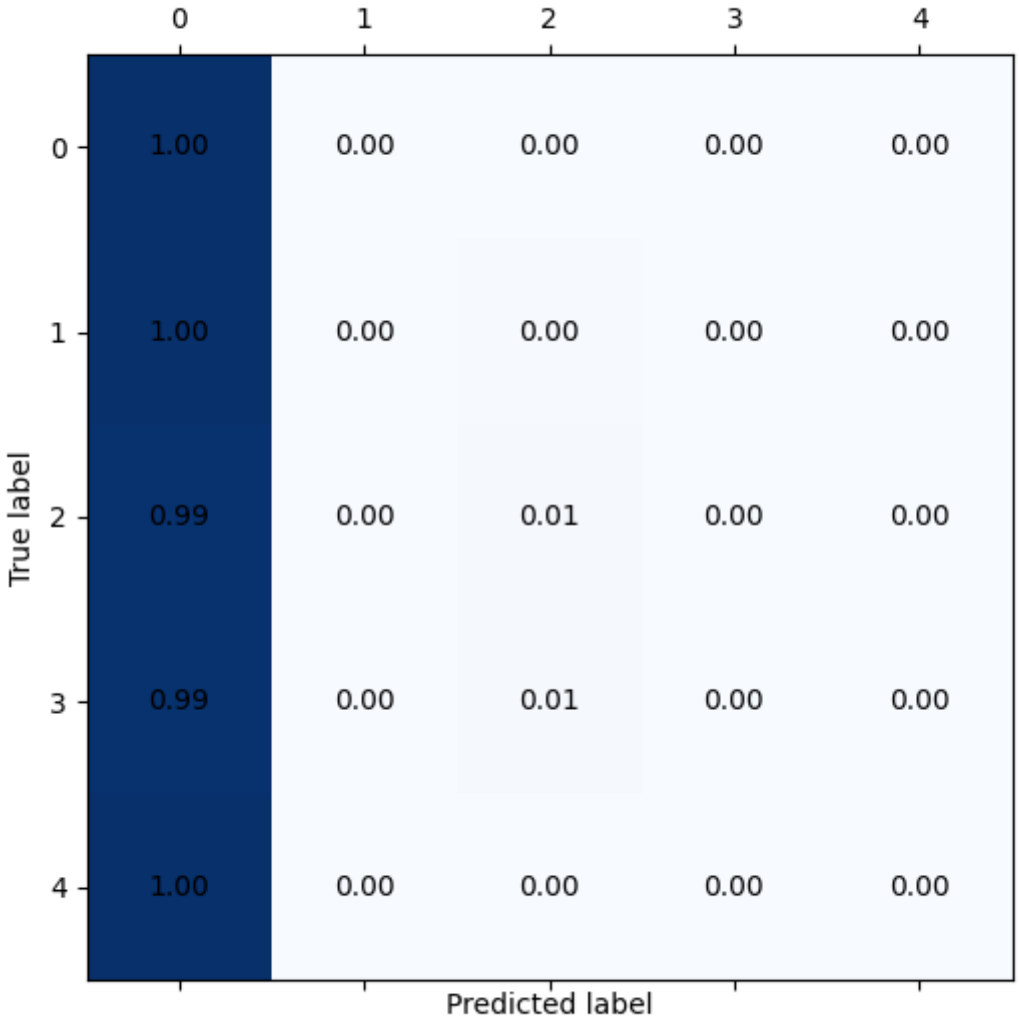
C. Describing your evaluation through the confusion matrix

建立一個5*5的矩陣在evaluate時統計各個結果的數量，最後在對每一列做normalize轉換成機率值

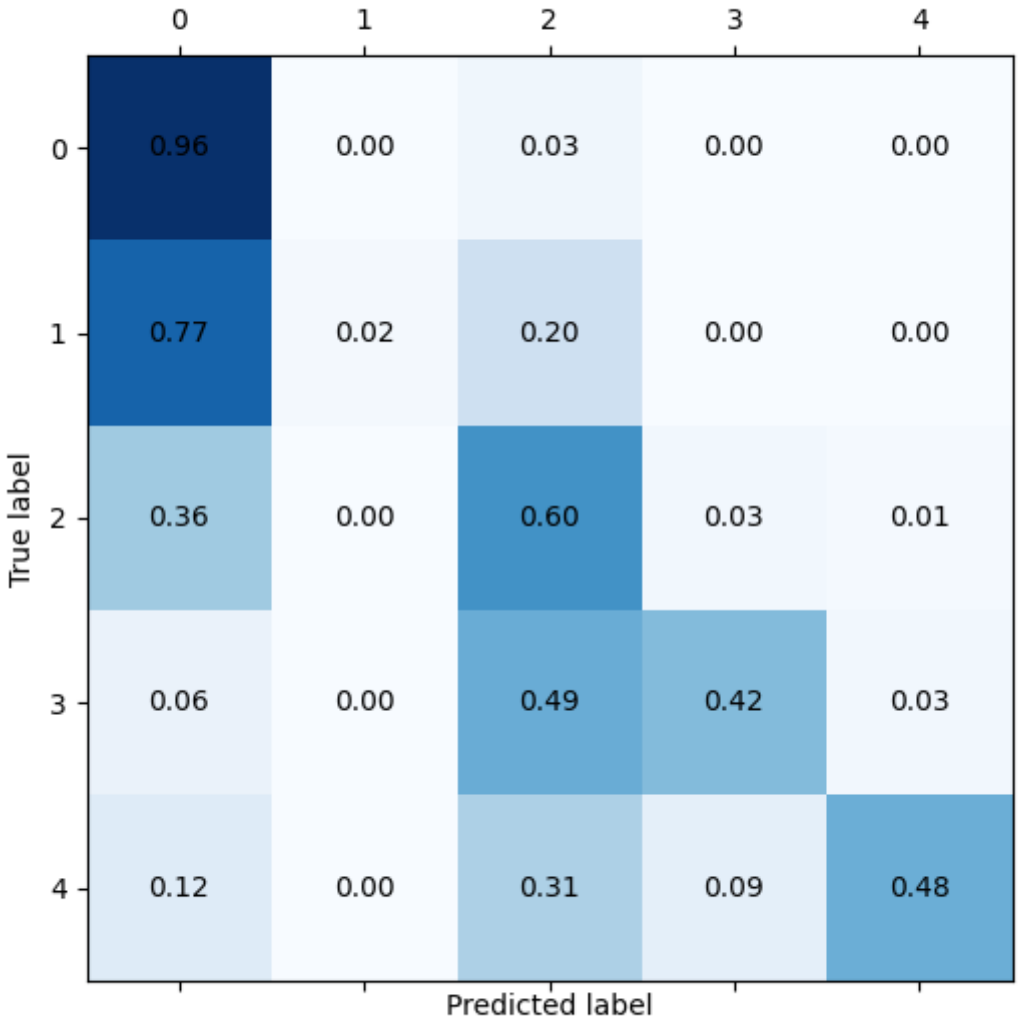
```
confusion_matrix=np.zeros((num_class,num_class))
with torch.set_grad_enabled(False):
    model.eval()
    correct = 0
    for _, (images, label) in enumerate(test_loader):
        images = images.to(device, dtype=torch.float)
        label = label.to(device, dtype=torch.long)
        predict = model(images)
        pred = predict.argmax(dim=1)
        for i in range(len(label)):
            confusion_matrix[int(label[i])][int(pred[i])]+=1
            if pred[i] == label[i]:
                correct +=1
    acc = 100. * correct / len(test_loader.dataset)
confusion_matrix=confusion_matrix/confusion_matrix.sum(axis=1).reshape(num_class,1)
```

Confusion matrix figure: Confusion matrix shows that the accuracy for class0 is high, but for model without pretrained weight, almost all kinds of classes will be predicted to class0(ResNet18). For model with pretrained weights, the accuracy is better as a whole. However, my model predict many class1 to class0, besides, seldom predict output is class1. It may represent that no strong features exist to classify class0 and class1.

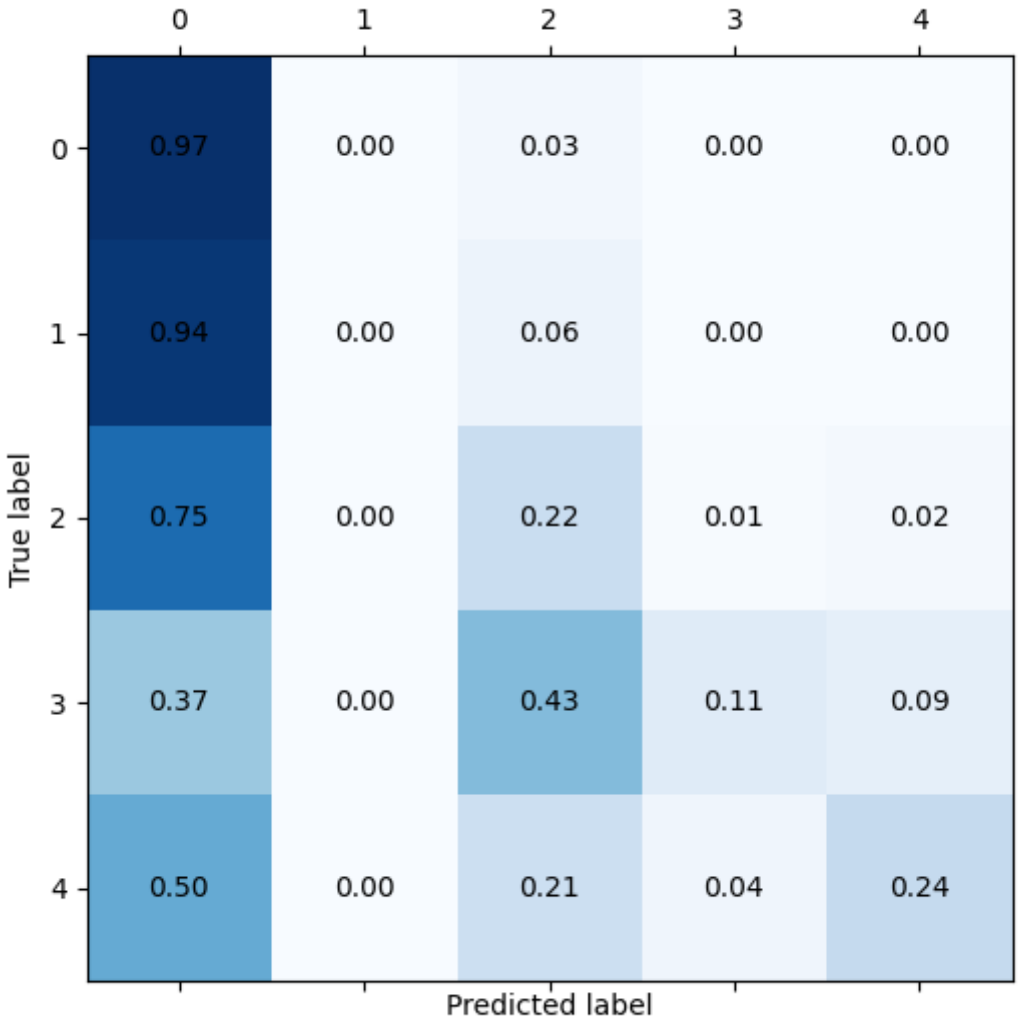
- ResNet18 w/o pretrained weights:



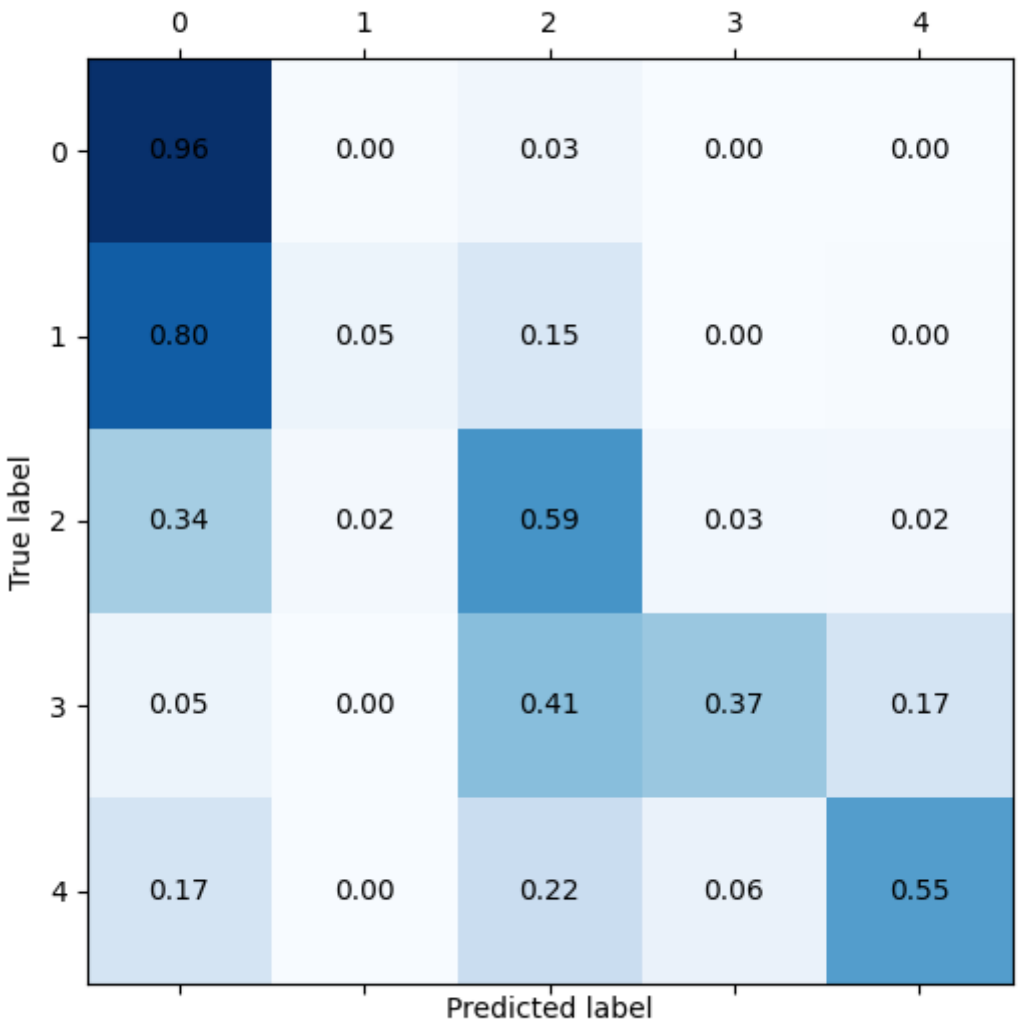
- ResNet18 with pretrained weights:



- ResNet50 w/o pretrained weights:



- ResNet50 with pretrained weights:



Experimental results

A. The highest testing accuracy

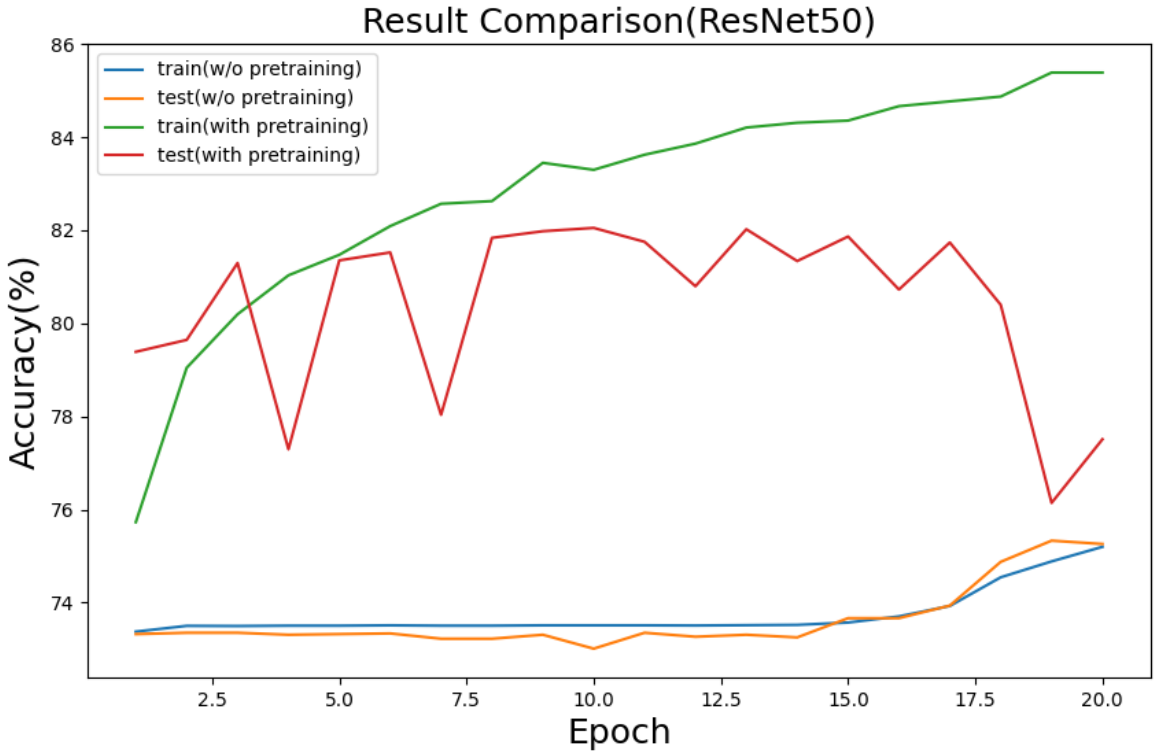
	ResNet18	ResNet50
w/o pretrained	73.367%	75.33%
with pretrained	82.26%	82.05%

- anything you want to present
  - epochs: 20
  - optimizer: SGD
  - weight_decay: 5e-4
  - criterion: CrossEntropy
  - batch size for ResNet18: 16
  - batch size for ResNet50: 8

- learning rate: 0.001

B. Comparison figures

- Plotting the comparison figures(ResNet18, ResNet50)



Model without pretrained can't increase accuracy. The accuracy of model with pretrain in training will increase, however, the accuaracy in testing drop in the last few epoch. This means model with pretrain occur overfitting.

Discussion

A. Anything you want to share

- 在training的過程中需要調整batch的大小或resize image的大小，不然龐大的參數數量會造成cuda out of memory，一開始我resize image的大小，但這會使accuracy降低，最後我降低batch的size來避免cuda out of memory的問題
- 我覺得這次的資料集class0的種類太多了，這種imbalance data不論看到什麼都輸出0的話正確律都有70%，如果dataset分佈比較平均的話在class(1,2,3,4)的辨別上應該會有更好的表現