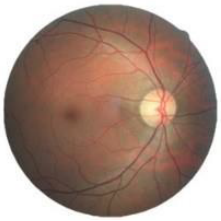
Diabetic Retinopathy Detection

1. Introduction

糖尿病視網膜病變是發達國家工作年齡人口失明的主要原因，在這個實驗中，主要是利用 pytorch實作出 ResNet來分類視網膜照片，分類出糖尿病視網膜病變的嚴重程度，根據病變的嚴重程度總共分做5類(label 0~4)。



數據集:包含 35124 張圖像，我們將數據集分為 28,099 個訓練數據和 7025 個測試數據。 圖像分辨率為 512x512 並經過預處理

Download link:

https://drive.google.com/open?id=1RTmrk7Qu9IBjQYLczaYKOvXaHWBS0o72

ResNet全名為Residual Neural Network。

有時較深的神經網路不容易訓練起來，其可能會出現的度消失、梯度爆炸等問題，使得更深的網路有時反而帶來更差的效果。而ResNet提出的residual learning使得深層網路更容易訓練。

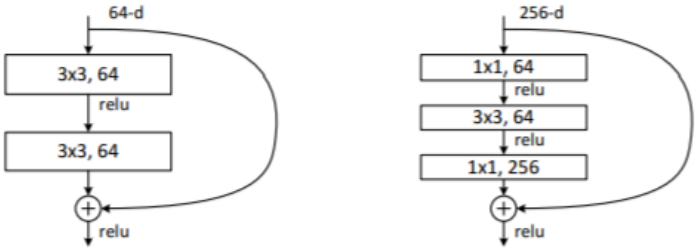


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

2. Experiment setups

A. The details of your model (ResNet)

ResNet18:

```
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
```

ResNet50:

```
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

將實做出RetinopathyLoader類別，放進torch.utils.data 提供的DataLoader裡，便可以透過DataLoader獲得所需要的Data。

```
train_dataset = RetinopathyLoader('./data', mode="train")
train_loader = DataLoader(train_dataset, batch_size=batch_size_50, shuffle=True, num_workers=4)
test_dataset = RetinopathyLoader('./data', mode="test")
test_loader = DataLoader(test_dataset, batch_size=batch_size_50, shuffle=True, num_workers=4)
```

RetinopathyLoader:

1.繼承了torch.utils.data 的Dataset。

2.覆寫__getitem__，讓DataLoader能從路徑找到input和target。

3.在建構函式__init__中對圖片做正規化處理(transforms.Normalize)。

```
class RetinopathyLoader(data.Dataset):
    def __init__(self, img_path, mode):
        self.img_path = img_path
        self.img_name, self.label = getData(mode)
        self.mode = mode
        #使用 transforms.Compose將一系列的transforms操作連結起來
        self.transformation = transforms.Compose([transforms.ToTensor(),transforms.Normalize((0.37,
0.26, 0.18),(0.25, 0.17, 0.12))])
        print("> Found %d images..." % (len(self.img_name)))
    def __len__(self):
        return len(self.img_name)
    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
```

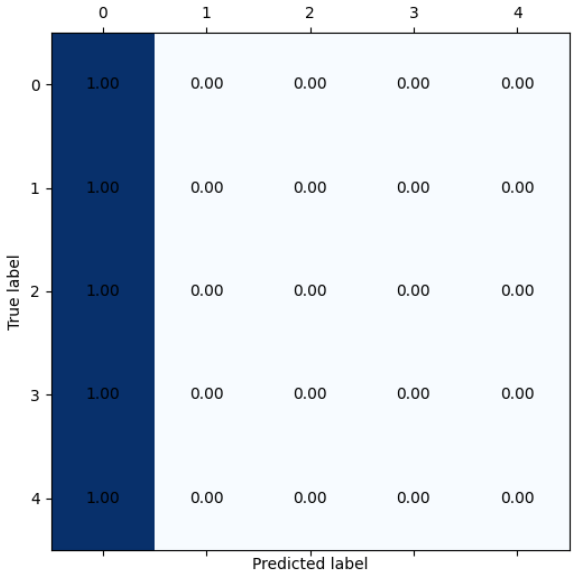
C. Describing your evaluation through the confusion matrix

建立一個5*5的矩陣在evaluate時統計各個結果的數量，視覺化label和predict結果。並做正規化處理，使每一行加起來為1。

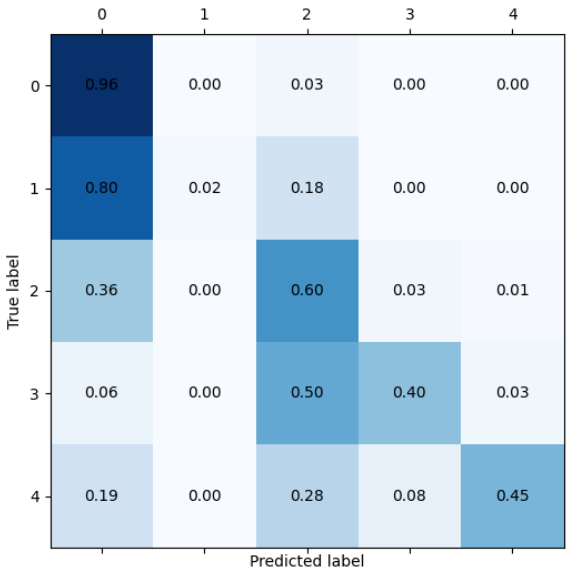
```
def evaulate(model, test_loader, device, num_class):
    confusion_matrix=np.zeros((num_class,num_class))
    with torch.set_grad_enabled(False):
        model.eval()
        correct = 0
        for _, (images, label) in enumerate(test_loader):
            images = images.to(device, dtype=torch.float)
            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)
#normalize
confusion_matrix=confusion_matrix/confusion_matrix.sum(axis=1).reshape(num_class,1)
return acc, confusion_matrix
```

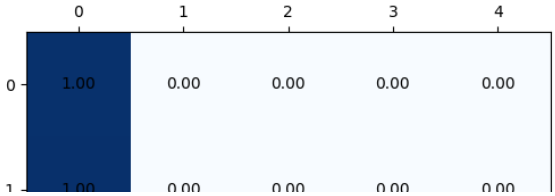
ResNet18 without pretrain weights

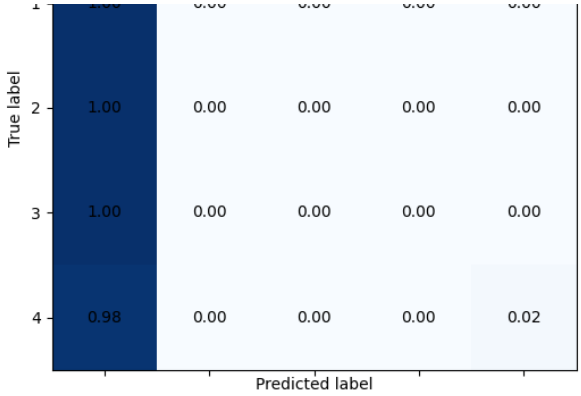


ResNet18 with pretrain weights

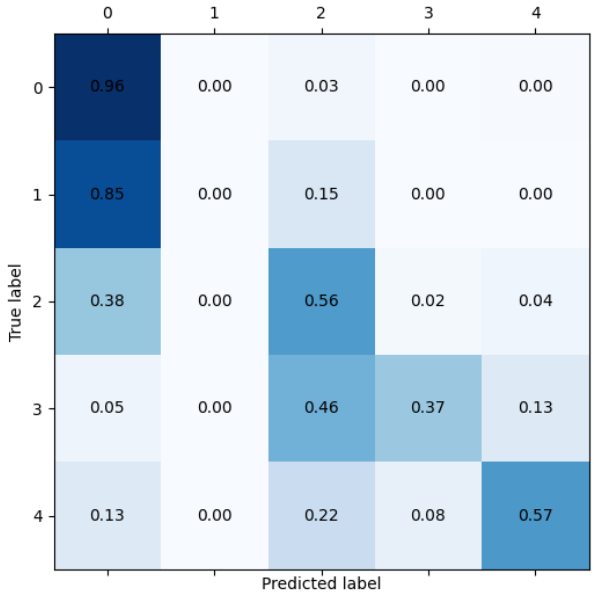


ResNet50 without pretrain weights





ResNet50 with pretrain weights



3. Experimental results

A. The highest testing accuracy

◆Screenshot

	ResNet18	ResNet50
without pretrain	73.35%	73.02%
with pretrain	81.70%	81.33%

ResNet18 without pretrain

```
epoch 8  acc_test:72.740%
best acc  resnet18_wo_pretrain :  73.35231316725978
```

ResNet18 with pretrain

```
epoch 8  acc_test:81.352%
best acc  resnet18_with_pretrain :  81.70818505338079
```

ResNet50 without pretrain

```
epoch 5  acc_test:73.025%
best acc  resnet50_wo_pretrain :  73.02491103202847
```

ResNet50 with pretrain

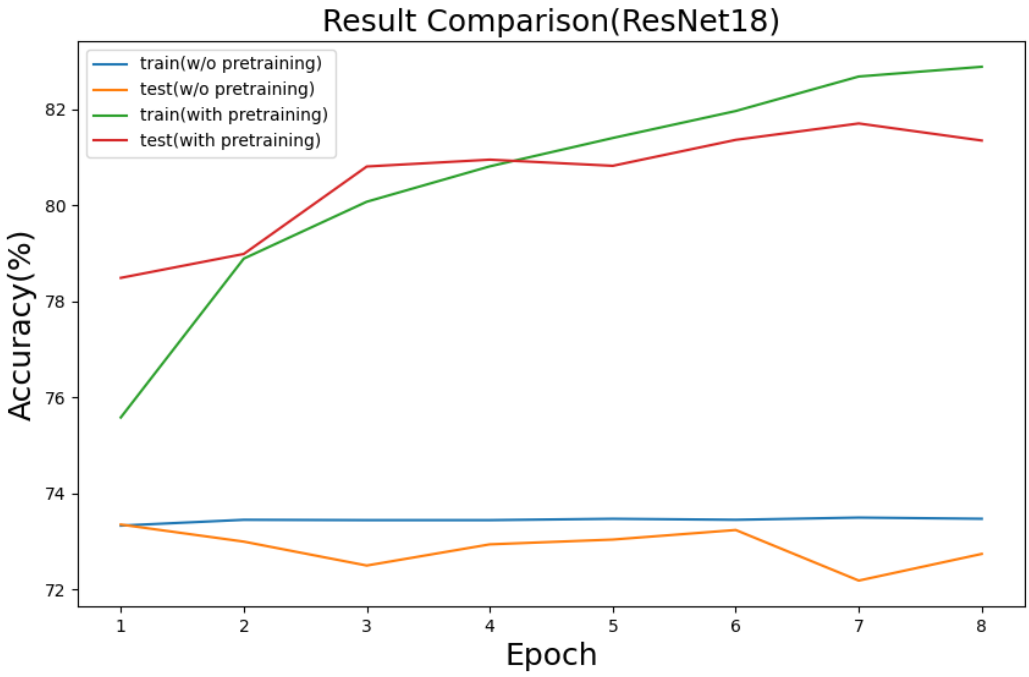
```
epoch 5  acc_test:81.338%
best acc  resnet50_with_pretrain :  81.33807829181495
```

B. Comparison figures

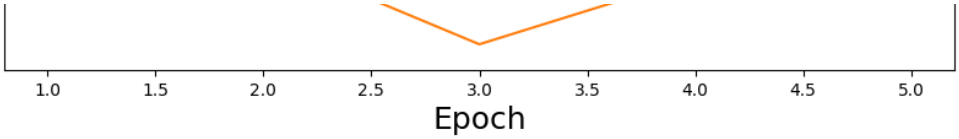
◆ Plotting the comparison figures

(RseNet18/50, with/without pretraining)

ResNet18 Compare



ResNet50 Compare



4. Discussion

A. Anything you want to share

◆訓練中，龐大的參數數量可能會造成cudaout of memory的問題(GPU空間不足)，在training的過程中可以調整batch的大小來解決。也可以resize image的大小，不過這個方法容易導致準確率下降。

◆實驗結果可以看出，有pretrained的model明顯表現得較佳。也許是因為這次實驗中的資料集過度不平衡，使得沒有pretrained的model難以進步。

因此可發現官方提供的ResNet pretrained model在提取低維特徵的表現上還是相當不錯的，更是省去了model重新從頭訓練所需要的工作 。