Lab3: Diabetic Retinopathy Detection

311581005 智能碩一 吳佳豪

1 INTRODCTION

本次作業主要使用 ResNet18 與 ResNet50 針對因糖尿病引發視網膜病變的眼睛圖片進行辨別，與實作自己的 Dataloader 並用 Confusion matrix 來分析辨別表現。另外也分析使用有無預訓練(pretrained)的差別，發現使用預處理的 ResNet18 可以達到 **83.06%**的精確度，而 ResNet50 也可達到 81.5%的精確度。

2 EXPERIMENT SETUPS

2.1 THE DETAILS OF YOUR MODEL (RESNET)

宣告一個 Retinopathy_Resnet 的 class 來包裝 ResNet18 與 ResNet50。因 ResNet 最後一層的輸出為 1000 個類別，而此次作業只有 5 個類別，並 Resnet18 與 ResNet50 最後一層的輸入數量分別為 512 與 2048 個 input，所以分別對兩個模型的 fully connected layer 進行改寫。

```python
class Retinopathy_Resnet(nn.Module):
    def __init__(self,layer_size,pretrained=False):
        super(Retinopathy_Resnet, self).__init__()
        self.layer_size = layer_size if layer_size == 18 else 50
        self.pretrained = pretrained
        self.model = models.resnet18(pretrained=self.pretrained) if self.layer_size == 18 else models.resnet50(pretrained=self.pretrained)
        self.model_name = f"resnet{self.layer_size}_pretrained" if self.pretrained else f"resnet{self.layer_size}"
        if self.layer_size == 18 :
            self.model.fc = nn.Linear(512,5)
        else:
            self.model.fc = nn.Linear(2048,5)

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

Figure 1. Retinopathy_Resnet model implement

2.2 THE DETAILS OF YOUR DATALOADER

可以自行傳入 transform 的 function 來增強自己的數據集，如果沒有傳入則會將資料都轉換為 tensor 的資料型態，並將圖片與 label 依照順序回傳。

```python
class RetinopathyLoader(data.Dataset):
    def __init__(self, root, mode, transform=None):
        """
        Args:
            root (string): Root path of the dataset.
            mode : Indicate procedure status("train" or "test")

            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
        self.transform = transform
        self.transform_toTensor = transforms.Compose([
            transforms.ToTensor()
        ])
        print("> Found %d images..." % (len(self.img_name)))

    def __len__(self):
        """'return the size of dataset"""
        return len(self.img_name)

    def __getitem__(self, index):
        """
         Return processed image and label
        """
        img_path = os.path.join(self.root, self.img_name[index] + ".jpeg")
        image = Image.open(img_path).convert('RGB')
        label = torch.tensor(self.label[index])
        if self.transform is not None:
            img = self.transform(image)
        else:
            img = self.transform_toTensor(image)
        return img, label
```

Figure 2. Dataloader implement

同時也檢測資料分布，可以發現 label 0 的圖片幾乎占了七成以上，所以會發生 data imbalance 的問題，這也是訓練時需要克服的問題。

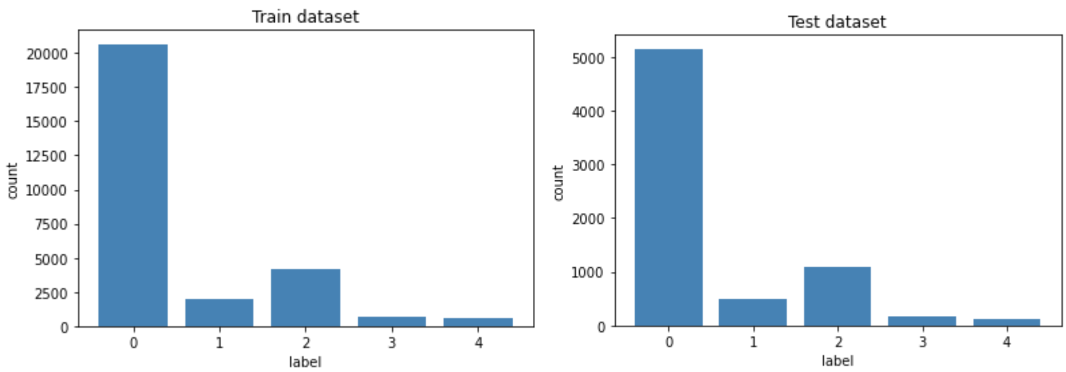


Figure 3. Train dataset and test dataset distribution

2.3 Describing your evaluation through the confusion matrix

下圖使用 pretrained 過的 ResNet18 model 經過訓練後並在 testing data 中預測出的 Confusion matrix。Matrix 的對角所代表的為分別預測每個 class 的

Accuracy，可以發現將對角線的數值 0.7127 + 0.003132 + 0.09609 + 0.00911 + 0.009537 相加大約等於 0.8306，驗證 test accuracy 沒有算錯。

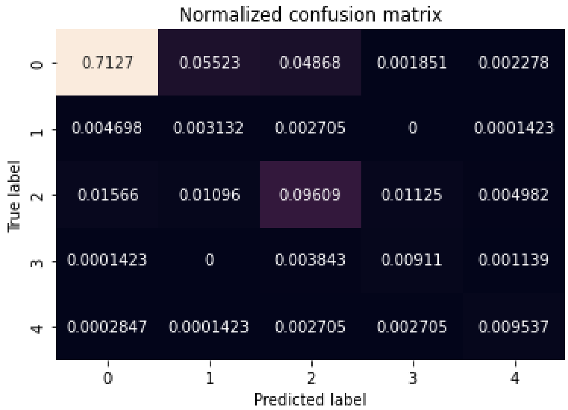


Figure 4. Confusion matrix

3 EXPERIMENTAL RESULTS

超參數設定：

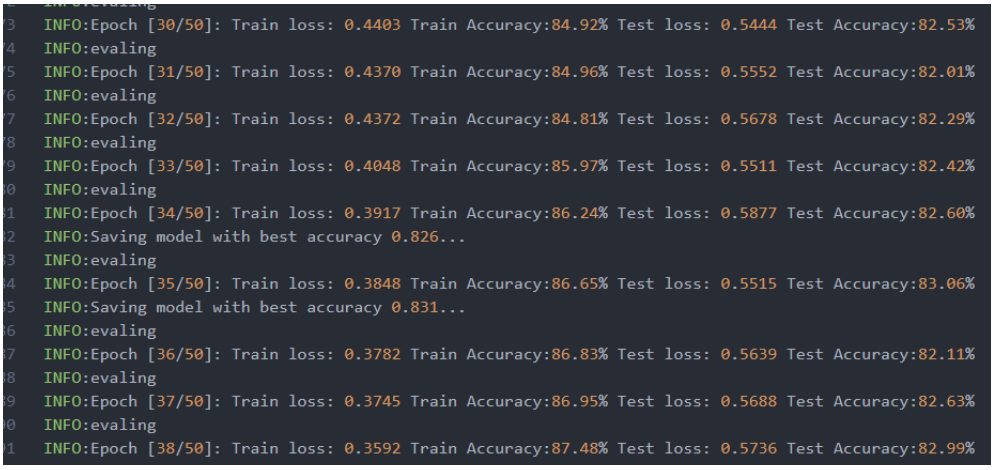
Table 1. Model hyperparameter settings

Hyperparameters	
Epochs	50
Batch_size	ResNet18: 64 / ResNet50: 16
Learning_rate	1e-2
Lr_scheduler	ReduceLROnPlateau(optimizer,mode="min",factor=0.5 ,patience=4,verbose=True,min_lr=1e-3)
Optimizer	SGD(momentum=0.9)
Weight_deacy	1e-3
Early_stop	20
Random seed	123456
Data augmentation	RandomHorizontalFlip(p=0.5), RandomVerticalFlip(p=0.5), ColorJitter(brightness=0.2,contrast=0.2,saturation=0.2), RandomAffine(degrees=30, scale=(0.9, 1))

3.1 THE HIGHEST TESTING ACCURACY

3.1.1 Screenshot

下方實驗結果皆依照 Table 1.參數進行設定。當 Pretrained-ResNet18 在訓練到 35 epoch 時達到 **83.06%**的精確度。



```
INFO:Epoch [30/50]: Train loss: 0.4403 Train Accuracy:84.92% Test loss: 0.5444 Test Accuracy:82.53%
INFO:evaling
INFO:Epoch [31/50]: Train loss: 0.4370 Train Accuracy:84.96% Test loss: 0.5552 Test Accuracy:82.01%
INFO:evaling
INFO:Epoch [32/50]: Train loss: 0.4372 Train Accuracy:84.81% Test loss: 0.5678 Test Accuracy:82.29%
INFO:evaling
INFO:Epoch [33/50]: Train loss: 0.4048 Train Accuracy:85.97% Test loss: 0.5511 Test Accuracy:82.42%
INFO:evaling
INFO:Epoch [34/50]: Train loss: 0.3917 Train Accuracy:86.24% Test loss: 0.5877 Test Accuracy:82.60%
INFO:Saving model with best accuracy 0.826...
INFO:evaling
INFO:Epoch [35/50]: Train loss: 0.3848 Train Accuracy:86.65% Test loss: 0.5515 Test Accuracy:83.06%
INFO:Saving model with best accuracy 0.831...
INFO:evaling
INFO:Epoch [36/50]: Train loss: 0.3782 Train Accuracy:86.83% Test loss: 0.5639 Test Accuracy:82.11%
INFO:evaling
INFO:Epoch [37/50]: Train loss: 0.3745 Train Accuracy:86.95% Test loss: 0.5688 Test Accuracy:82.63%
INFO:evaling
INFO:Epoch [38/50]: Train loss: 0.3592 Train Accuracy:87.48% Test loss: 0.5736 Test Accuracy:82.99%
```

Figure 5. Train log

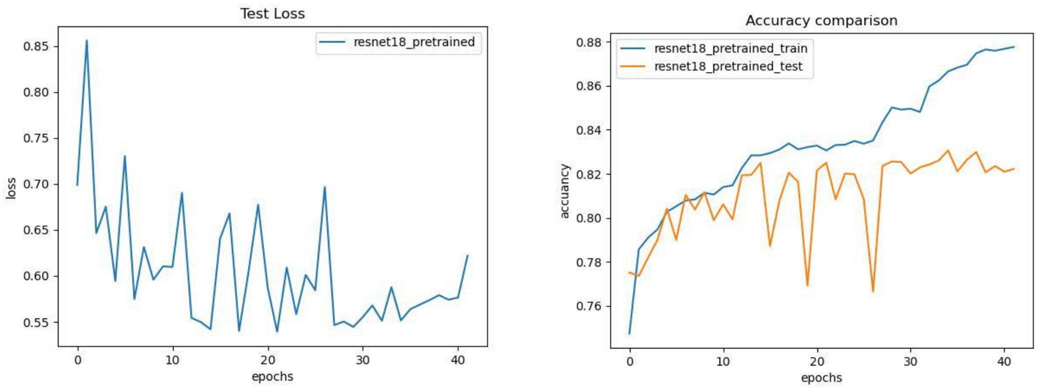


Figure 6. Loss and Accuracy with Pretrained ResNet18

3.1.2 Anything you want to present

下方實驗結果皆依照 Table 1.參數進行設定。一開始為了 training 速度較為快速，所以都將訓練圖片 resize 到 256 * 256，但由下面 Table 2.可以發現就算是使用 data augmentation 增加資料量，還是難以突破 82%的精確度，因此最後還是將訓練圖片變回 512 * 512 進行 training，雖然時間增加兩倍以上，但可以看到 Table 3.的結果相較 Table 2.好。

Table 2. Accuracies of all models is trained with 256 * 256 image size

Results	ResNet18	ResNet50
pretrained	**81.17%**	**81.41%**
no pretrained	76.24%	73.35%

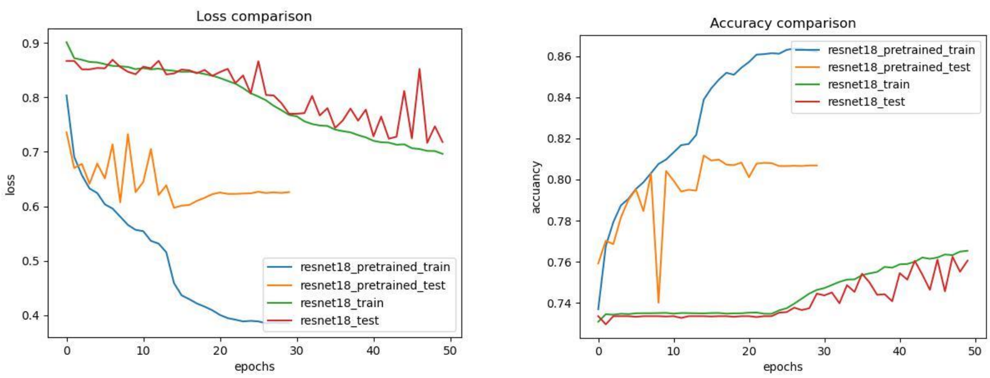


Figure 7. Loss and Accuracy comparison with ResNet18 which is trained by

256 * 256 image size

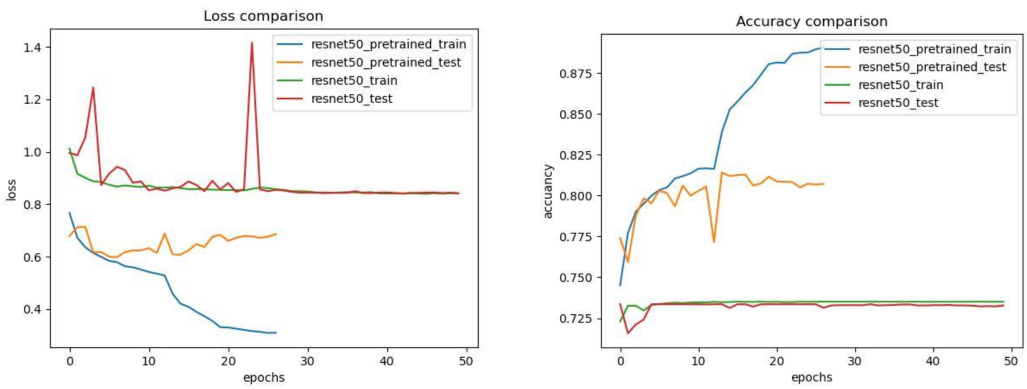


Figure 8. Loss and Accuracy comparison with ResNet50 which is trained by

256 * 256 image size

3.2 COMPARISON FIGURES

下方 Table 2.為 ResNet18 與 ResNet50 的實驗結果皆依照 Table1.參數進行設定，並且圖片 size 為 512 * 512。

Table 3. Accuracies of all models is trained with 512 * 512 image size

Results	ResNet18	ResNet50
pretrained	**83.06%**	**81.47%**
no pretrained	77.89%	75.76%

下面為 ResNet18 與 ResNet50 的 loss 和 accuracy 的比較圖。可以發現兩個架構都在使用 pretrained 的 model 時有較佳的表現。

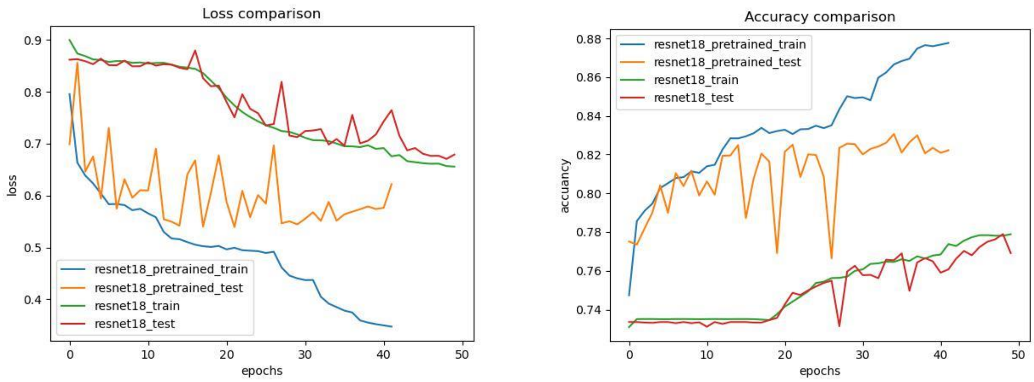


Figure 9. Loss and Accuracy comparison with ResNet18

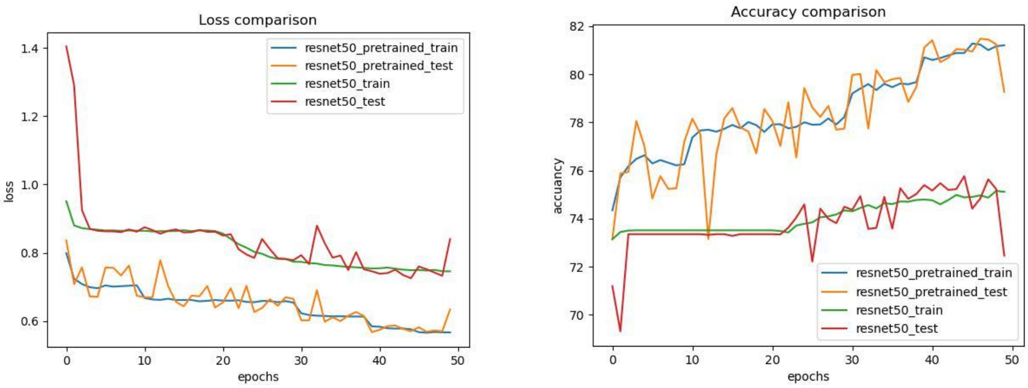


Figure 10. Loss and Accuracy comparison with ResNet50

4 DISCUSSION

DATA AUGMENTATION

為了可以獲得更佳的精確度，因此使用下面四種方法增加訓練資料，分別為 RandomHorizontalFlip、RandomVerticalFlip、ColorJitter 和 RandomAffine，並且之後使用 Normalize，提升訓練效率與精確度。

```
train_transforms_func = transforms.Compose([
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.RandomVerticalFlip(p=0.5),
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2),
    transforms.RandomAffine(degrees=30, scale=(0.9, 1)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])
```

Figure 11. Transforms function

Different method :

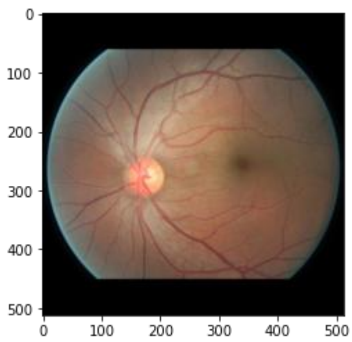


Figure 12. Origin

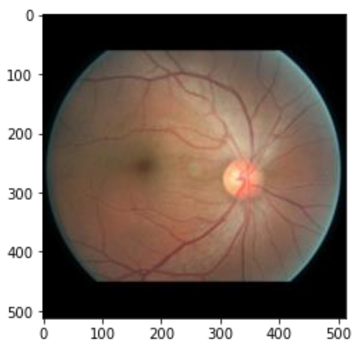


Figure13.RandomHorizontalFlip

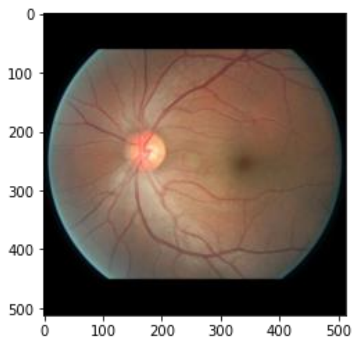


Figure 14.RandomVerticalFlip

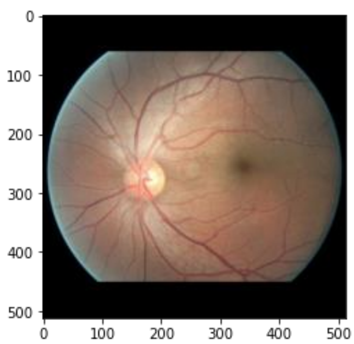


Figure 15. ColorJitter

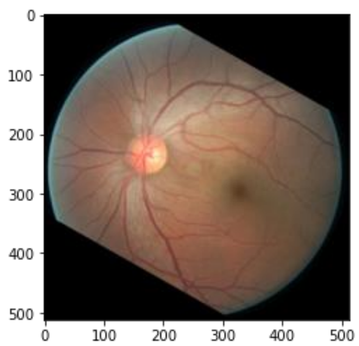


Figure 16. RandomAffine

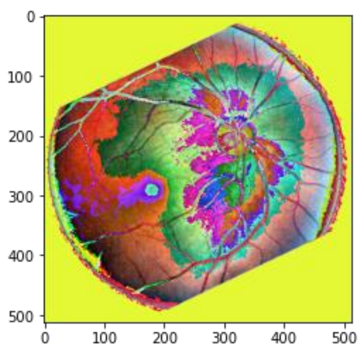


Figure 17. Add all