

## DLP

### Lab 4 Diabetic Retinopathy Detection

#### 1. Introduction

這次的 lab 利用 torchvision 的 models 導入 ResNet-18 和 ResNet-50 來進行糖尿病對視網膜病變的偵測

#### 2. Experiment Setups

##### A. The details of your model (ResNet)

我使用 torchvision.models.resnet18 及 torchvision.models.resnet50，  
torchvision=0.10.1+cu111，下圖是不同層數的 resnet 的架構

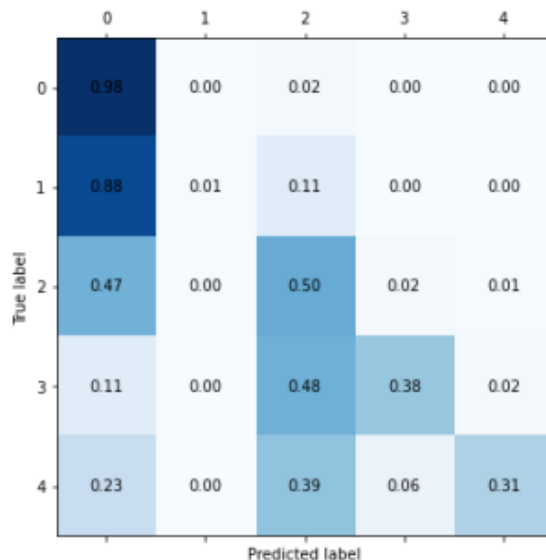
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		$1.8 \times 10^9$	$3.6 \times 10^9$	$3.8 \times 10^9$	$7.6 \times 10^9$	$11.3 \times 10^9$

##### B. The details of your Dataloader

因為我先用 datapreprocessing.py 把 images 裁切成 512\*512\*3，並且另存到 cut\_train 和 cut\_test，所以 dataloader 負責讀取這兩個資料夾，再把 images 轉成 3\*512\*512

使用 transform 增加樣本數，transform 包含旋轉、水平與垂直翻轉

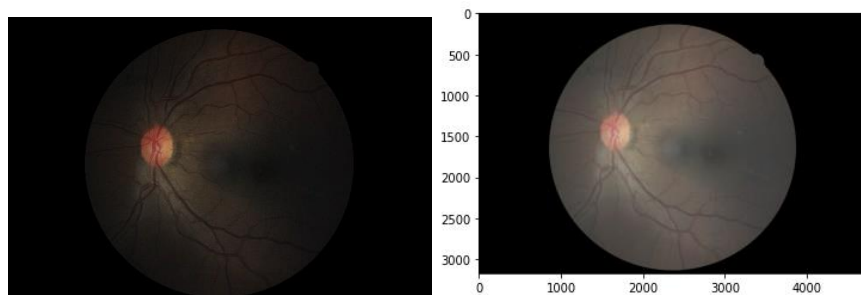
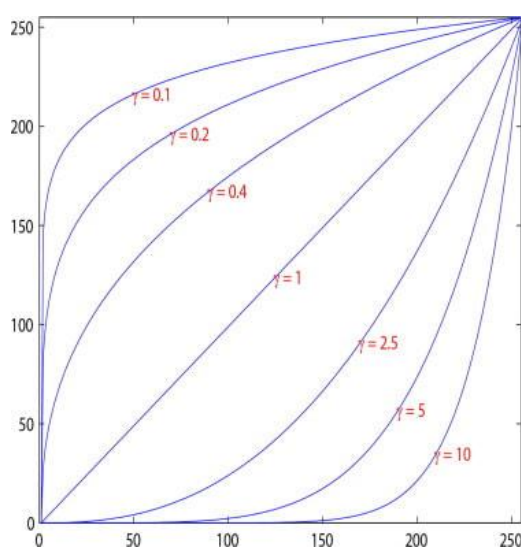
##### C. Describing your evaluation through the confusion matrix



#### 3. Data Preprocessing

##### A. How you preprocessed your data?

i. 用 gamma correction 提高 image 的亮度，用來分辨眼球的邊界



ii. 切下眼球的範圍

iii. 用 zero padding 補成正方形

iv. Resize to 512\*512\*3

v. 存到 cut\_train and cut\_test

B. What makes your method special?

用 gamma correction 把眼球的亮度提高，更方便取 threshold 把眼球的部分切割出來

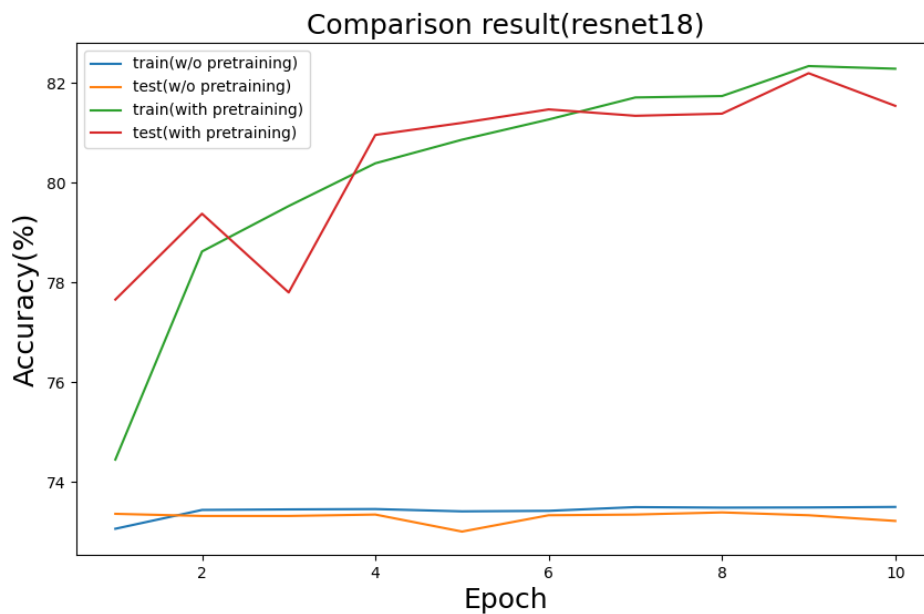
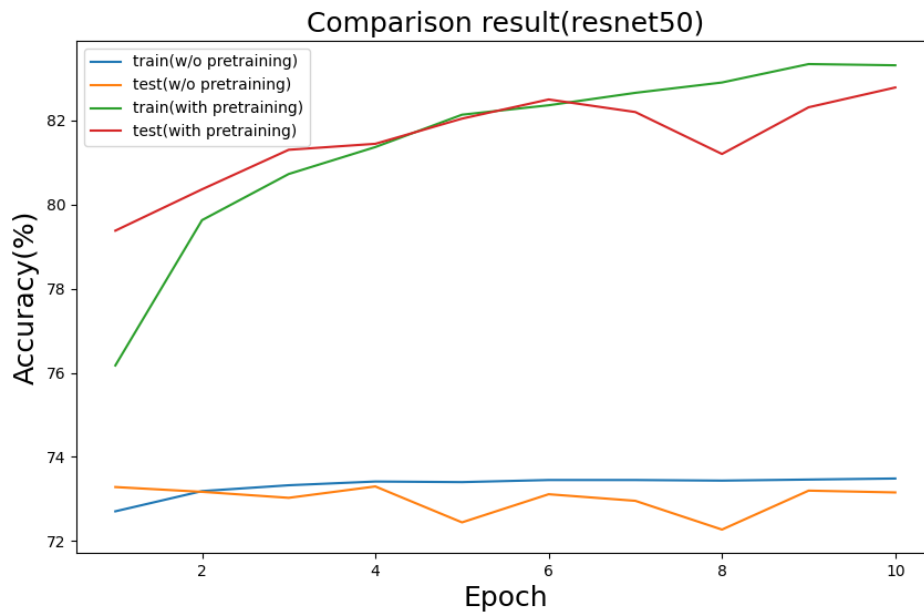
Zero padding 把圖片補齊，避免 resize 造成眼球變形

#### 4. Experimental results

A. The highest testing accuracy: 80.288%

Hyperparameters	
Batch size	8
Learning rate	0.001
Train epoch	5
Loss function	Cross Entropy
optimizer	SGD
Pretrain weight	use

## B. Comparison figures



## 5. Discussion

原本是在 `dataloader` 做 `preprocessing`，但是原始檔案太大，造成 `training` 速度緩慢，不適合做實驗，所以決定先花一些時間把影像預處理並儲存到資料夾中，`ResNet` 過程就可以加速，同時避免 `cache out of memory`。

雖然 `preprocessing` 就可以降低 `cache out of memory` 的風險，但為了以防萬一，每個 `batch` 結束後，會用 `torch.cuda.empty_cache()` 清理 `cache`