DLP Lab3 Diabetic Retinopathy Detection

0711529 陳冠儒

1. Introduction (20%)

這次的 lab 要分析並預測 diabetic retinopathy (糖尿病所引發視網膜病變)，並且要 (1)搭建自己的 DataLoader (2)用 ResNet18 跟 RetNet56 架構比較 pretraining 跟 non-pretraining 的結果 (3)計算 confusion matrix 來判別這個分類 model 的好壞。

A. Dataset

這次使用的是 2015 Kaggle 比賽 Diabetic Retinopathy Detection 的資料集 [1]，他提供了許多高分辨率視網膜圖像，圖像可能包含偽像、失焦、曝光不足或曝光過度。本次競賽的一個主要目的是開發可以在存在 noise 和變化的情況下運行的穩健算法，以預測糖尿病視網膜病。

在 label 的部分，臨床醫生根據等級以 0 到 4 的等級對每張圖像中是否存在糖尿病視網膜病變進行評分：

0 - No DR

1 - Mild

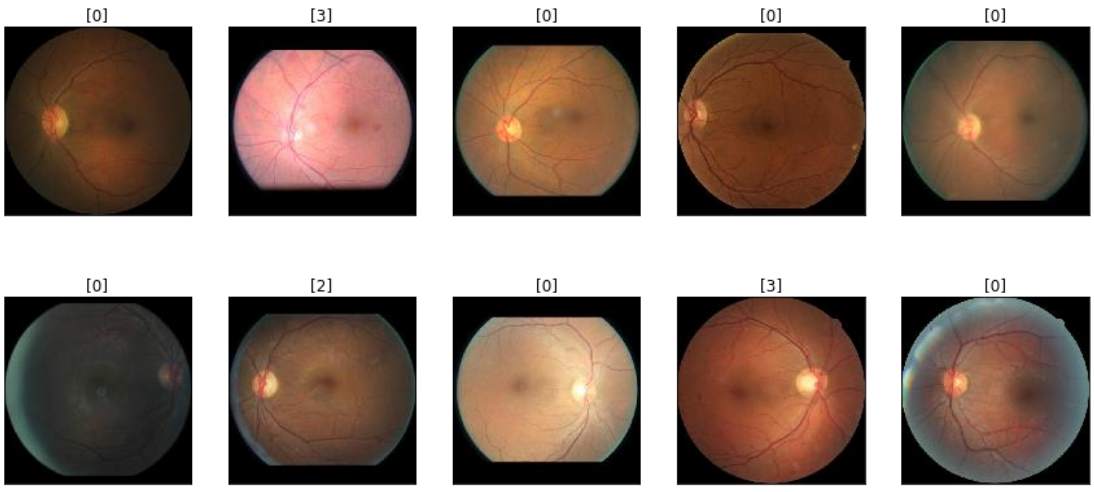
2 - Moderate

3 - Severe

4 - Proliferative DR

即 0 最輕微，而 4 最嚴重。

其中 training data 共有 28099 而 testing data 共有 7025，每一個 data 都是 (3, 512, 512)的彩色視網膜圖像，並對應到一個 label。



B. Reference

[1] Kaggle Diabetic Retinopathy Detection:
https://www.kaggle.com/c/diabetic-retinopathy-detection/data

2. Experiment setups (30%):

A. The details of your model (ResNet)

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				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. Architectures for ResNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv31, conv41, and conv51 with a stride of 2.

■ ResNet18

我使用與論文[1](即上表)相同的建構方式，首先用 conv2d、BatchNorm2d、ReLU、MaxPool2d 接下來就接 8 個 black layer，分別就是 conv2_x 兩個、conv3_x 兩個、conv4_x 兩個、conv5x 兩個，每個 black layer 都用 residual 都方式連接，最後與 paper 不同之處在於，將 fc layer 換成了 output 為 5，即 number of classes。

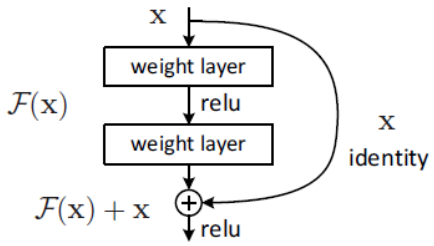


Figure. Residual learning: a building black

```
================================================================
Total params: 11,179,077
Trainable params: 11,179,077
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 3.00
Forward/backward pass size (MB): 328.00
Params size (MB): 42.64
Estimated Total Size (MB): 373.65
----------------------------------------------------------------
```

Figure. Number of parameters in the ResNet18 model.

- ResNet50

這個也是照著 paper 的方式搭建 model，最開頭的地方與 ResNet18 一樣，與 ResNet18 不同的地方有兩點，第一點 ResNet50 更深，也就是用了更多層、更多個 black layer，而第二點從上面 Table 1 可以看到他們兩個在 black layer 的設計上也不相同。

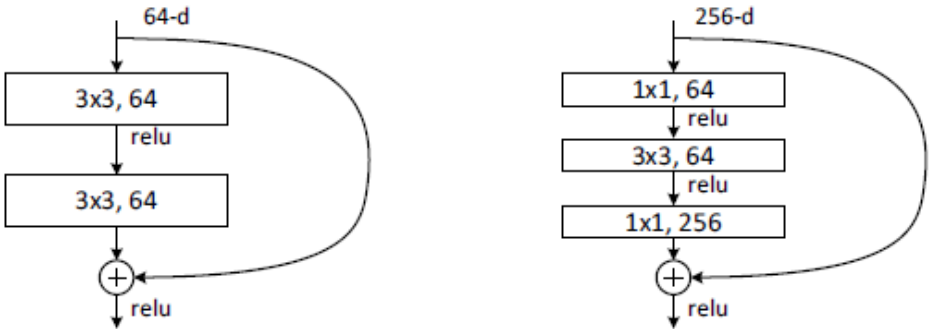


Figure. A deeper residual function F for ResNet. Left: a building block (on 56×56 feature maps). Right: a "bottleneck" building block for ResNet-50/101/152.

Bottleneck layer：使用 3 層的 layer 而不是 2 層。這三層是 1x1、3x3 和 1x1 卷積，其中 1x1 層負責減少和增加維度，使 3x3 層成為輸入/輸出維度較小的瓶頸。因為 1x1 捲積的使用，讓這兩種設計的 time complexity 相差不大，1x1 捲積能減少 trainable parameters，並加深網路層數。

```
================================================================
Total params: 23,518,277
Trainable params: 23,518,277
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 3.00
Forward/backward pass size (MB): 1497.02
Params size (MB): 89.72
Estimated Total Size (MB): 1589.73
----------------------------------------------------------------
```

Figure. Number of parameters in the ResNet50 model.

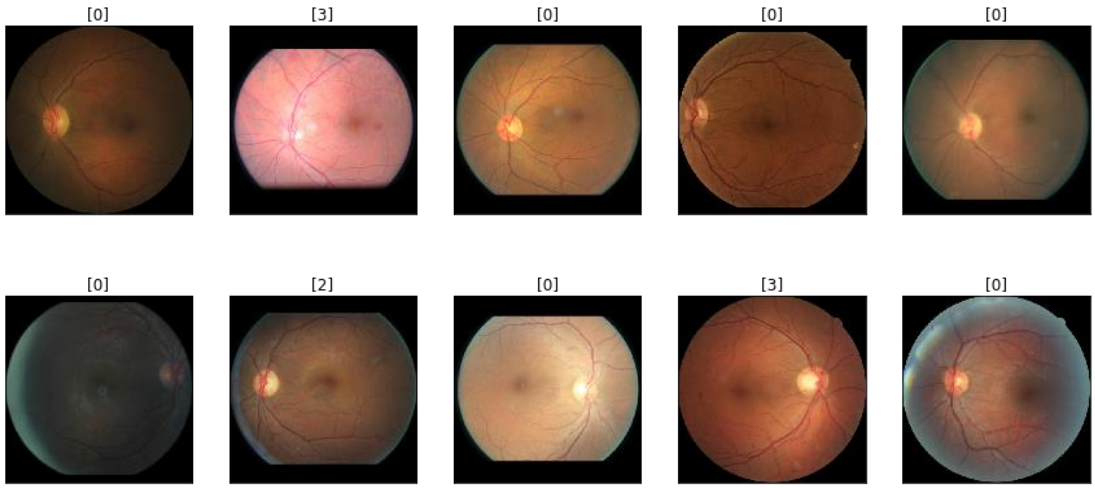
- Reference

[1] Deep Residual Learning for Image Recognition:
https://arxiv.org/pdf/1512.03385.pdf

B. The details of your Dataloader

在 dataloader 的地方除了將資料讀取外，也做了 data preprocessing 的部分。

■ Data Preprocessing



重新觀察下原始的 image 我們可以發現其中一些不一致的地方：

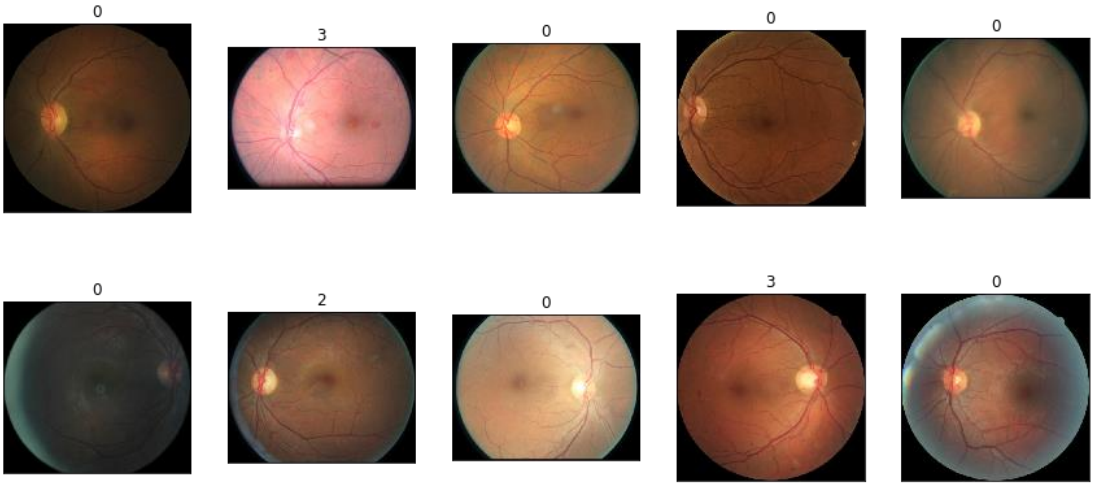
- 上下有一些不具任何資訊含量的黑色，且每個的大小都不一致。
- 有些眼睛呈圓形，而另一些則看起來像橢圓形。
- 每個的亮度跟顏色也都有些許差異

⇒ 由於位於視網膜中的線索的大小和形狀決定了疾病的嚴重程度，因此標準化眼睛形狀也至關重要。

故我對其進行了一些 preprocessing 步驟

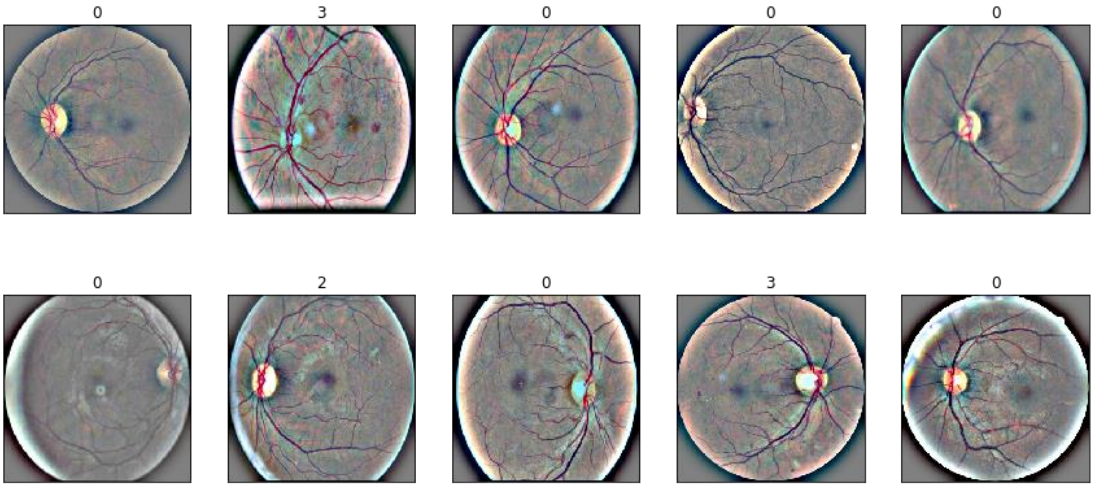
(1.) Crop Black

使用 cv2.COLOR_RGB2GRAY 得到 image 的灰階，再利用灰階製作 mask，判斷如果低過 threshold (設為 7)就將其視為黑色，將他水平的切掉。



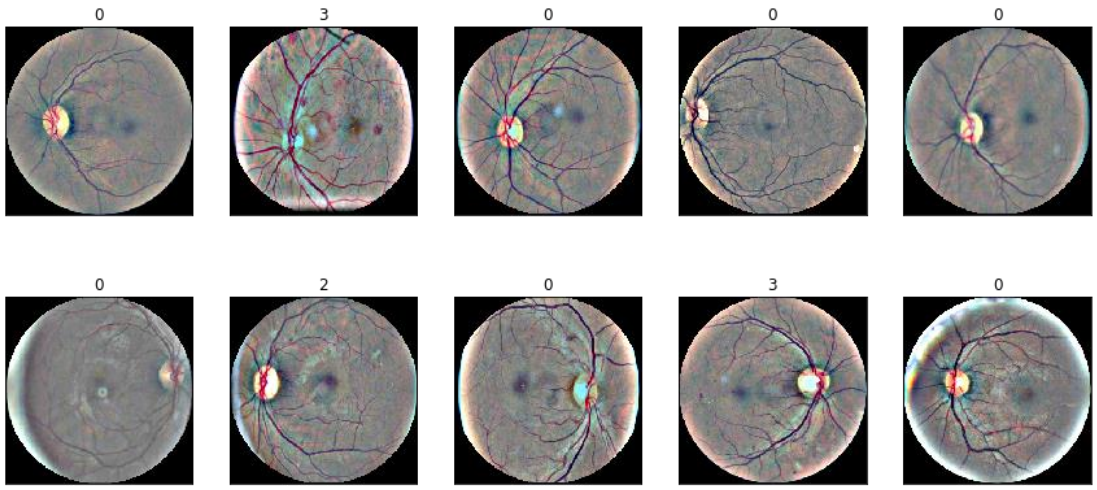
(2.) Resize and Change the Color

將 size 都調回原來的大小，並且再用 cv2.GaussianBlur 得到銳化邊緣，搭配上 cv2.addWeighted 可以將保留邊緣線條而將其他不相關的色彩去掉。



(3.) Circular Crop

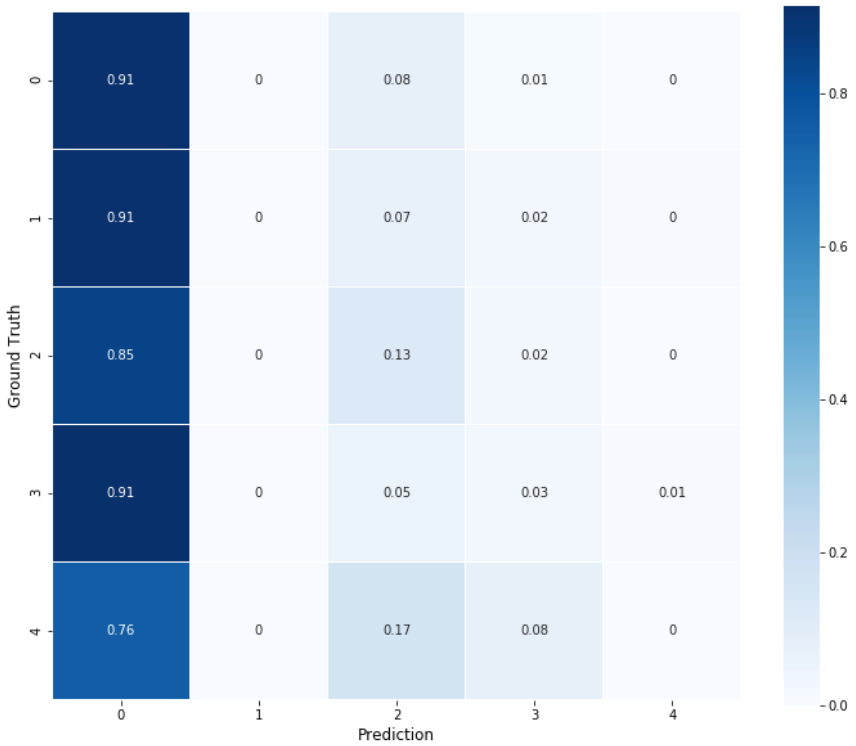
最後用 cv2. circle 將所有 image 都調成圓形。



C. Describing your evaluation through the confusion matrix

從疾病的角度上來看，應該要用「Precision」指標，寧願誤判也不可以漏掉，故下面就會分析每一個結果的 precision 為主。

■ ResNet18 without pretraining

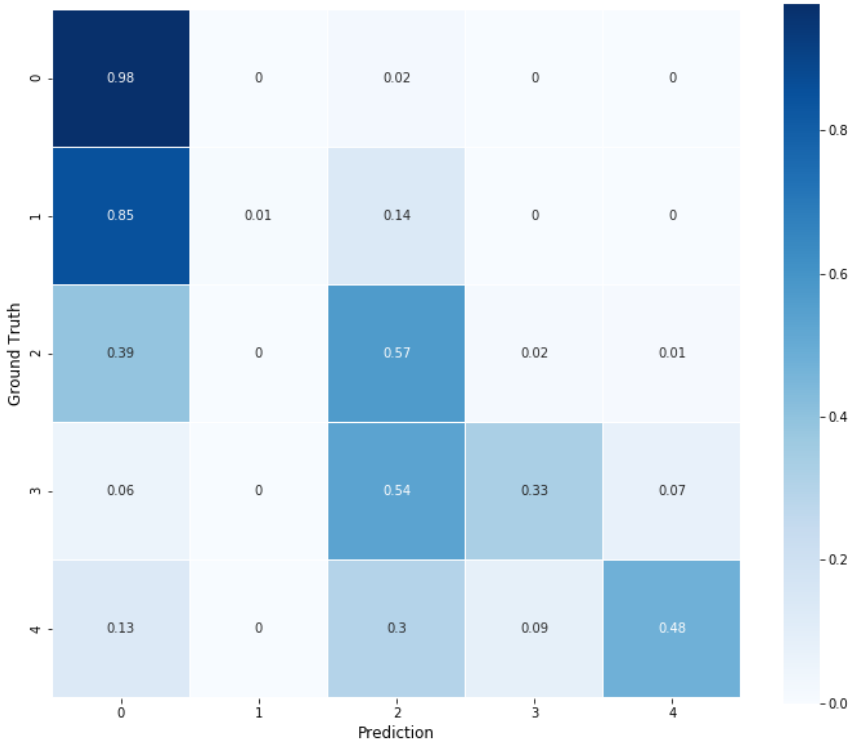


- Precision
  - ➢ label 1：0
  - ➢ label 2：14%
  - ➢ label 3：12.5%
  - ➢ label 4：0

  → 疾病的平均 precision 為 6.625%

- **結論**：雖然 Accuracy 有 76.39%，但從 precision 的結果 6.625% 上來看這個結果是很差的。

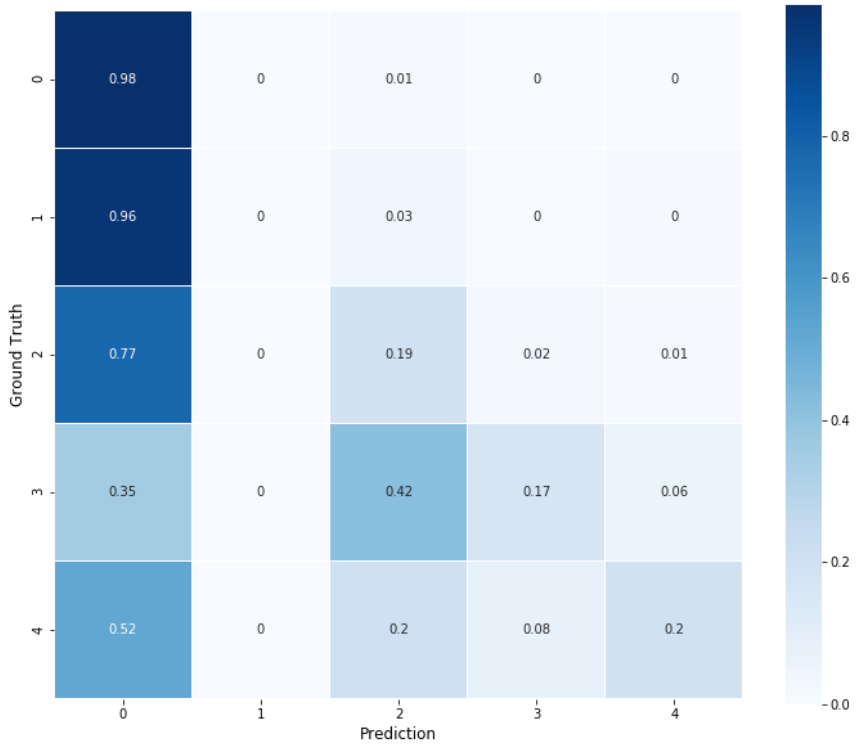
■ ResNet18 with pretraining



- Precision
  - ➢ label 1：100%
  - ➢ label 2：36.3%
  - ➢ label 3：75%
  - ➢ label 4：85.7%

  → 疾病的平均 precision 為 74.25%，而加權平均(考慮資料分布)後為 59.87%。

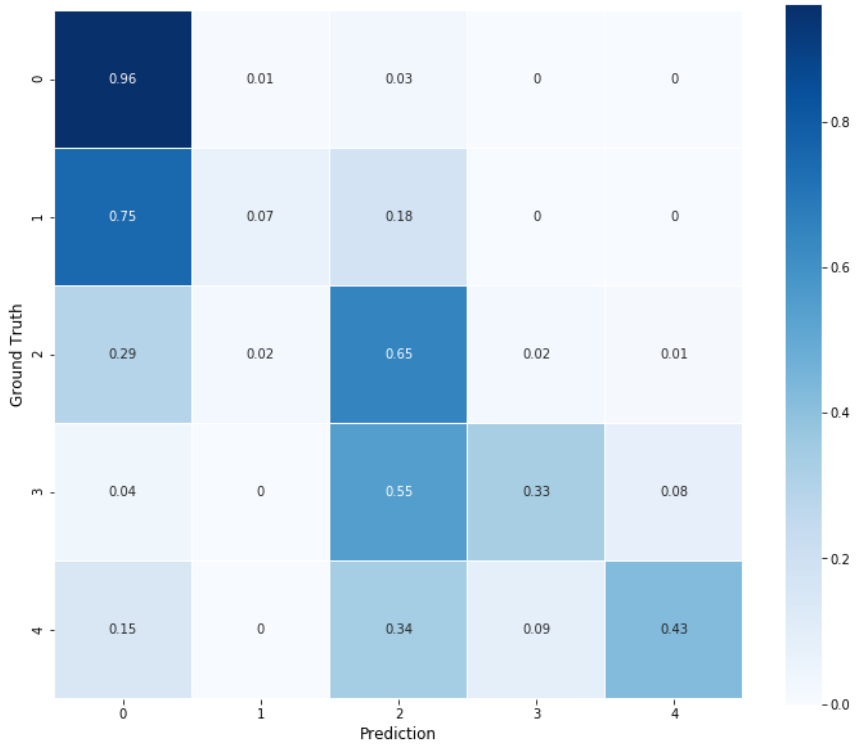
■ ResNet50 without pretraining



- Precision
  - ➢ label 1：0%
  - ➢ label 2：22.4%
  - ➢ label 3：63%
  - ➢ label 4：74.1%

  → 疾病的平均 precision 為 39.875%

■ ResNet50 with pretraining



- Precision
  - label 1：70%
  - label 2：37.1%
  - label 3：75%
  - label 4：80.8%

  → 疾病的平均 precision 為 65.725%，而加權平均(考慮資料分布)後為 52.18%。

3. Experimental results (30%)
   A. The highest testing accuracy
      - Screenshot
        - Hyperparameters of ResNet18

          - Epoch: 18 (with pretraining), 15 (without pretraining)
          - Optimizer: SGD
          - lr: 0.001/0.0005 (with pretraining), 0.001 (without pretraining)
          - momentum: 0.9
          - weight_decay: 0.0005/0.001/0.002 (with pretraining),
            0.0005 (without pretraining),
          - batch: 32
          - loss: CrossEntropy

        - Hyperparameters of ResNet50

- Epoch: 10
- Optimizer: SGD
- lr: 0.001/0.0005/0.0001 (with pretraining),
  0.001 (without pretraining)
- momentum: 0.9
- weight_decay: 0.001/0.002/0.01/0.015 (with pretraining),
  0.001 (without pretraining)
- batch: 8
- loss: CrossEntropy

- 結果

	with pretraining	without pretraining
ResNet18	82.16%	76.39%
ResNet50	82.49%	75.9%

- 結果分析

  最高的是 ResNet50 with pretraining。

  ✓ ResNet50 with pretraining 訓練過程。

  最高的 ResNet50 with pretraining testing accuracy 我是分成了很多段的訓練：

  - 前 10 epoch 用了 lr: 0.001、weight_decay: 0.001
  - 第 11 epoch 用了 lr: 0.0005、weight_decay: 0.002
  - 第 12 epoch 用了 lr: 0.0005、weight_decay: 0.01
  - 第 13 epoch 用了 lr: 0.0001、weight_decay: 0.015

  會這樣調整是因為從 loss 的結果可以看到 testing loss 會上升，所以漸漸把 weight_decay 調高以避免 overfitting，而將 lr 調低是因為看到了 training loss 也上升(不調低的情況下)，所以也將其慢慢調低，讓他以更小的步伐去接近最佳解。

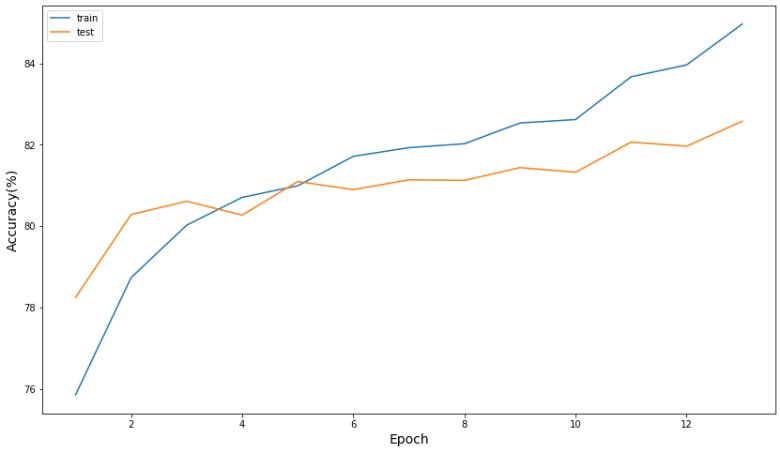


Figure. ResNet50 with pretraining accuracy-epoch

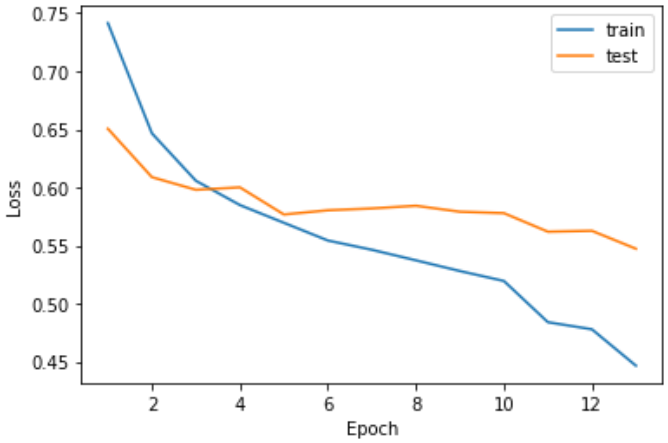
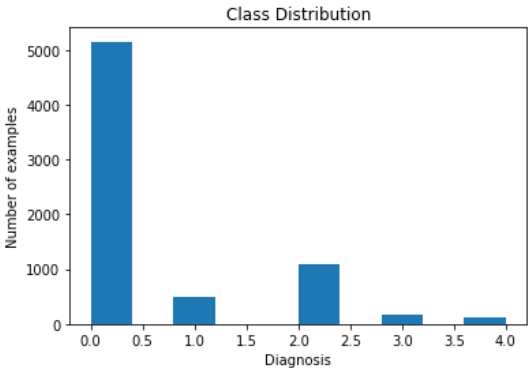


Figure. ResNet50 with pretraining loss-epoch

■ Anything you want to present

首先我先分析了一下數據的分布，以下為 testing data 的 distribution。

- 0 - No DR：5153, 73.35%
- 1 - Mild：488, 6.95%
- 2 - Moderate：1082, 15.4%
- 3 - Severe：175, 2.49%
- 4 - Proliferative DR：127, 1.81%



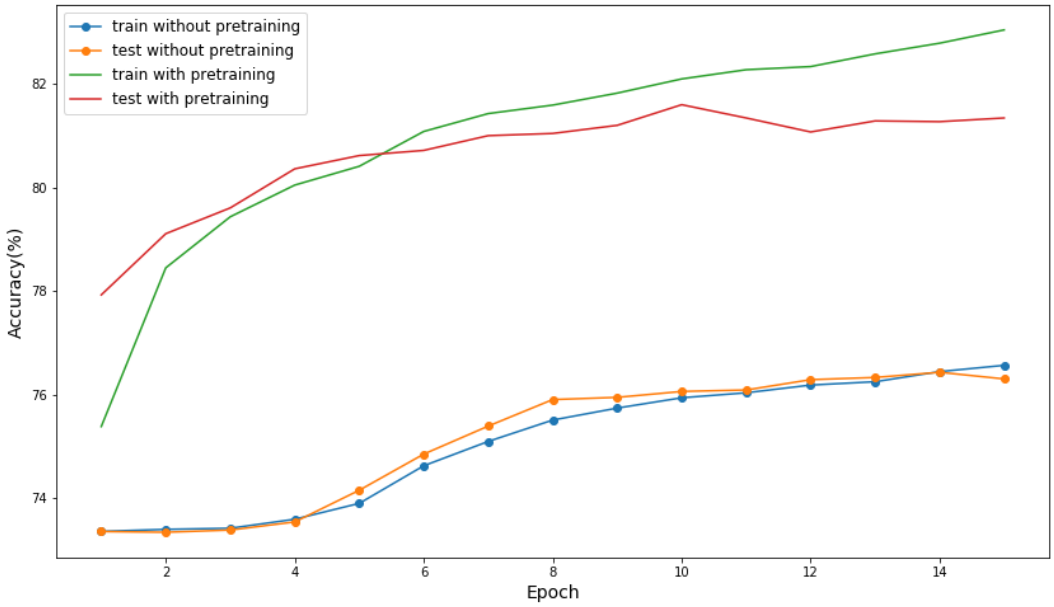
從 testing data 可以發現數據並不平衡，73% 的圖像來自健康患者。 剩下的 27% 是 DR 的不同階段。 最不常見的類別是 3 (Severe)跟 4（Proliferative DR），各

都僅佔全部的 2%。故 model 最差的情況下，預測 accuracy 應該也要有 73.35%以上，即全部猜測 label 為 0 的情況。

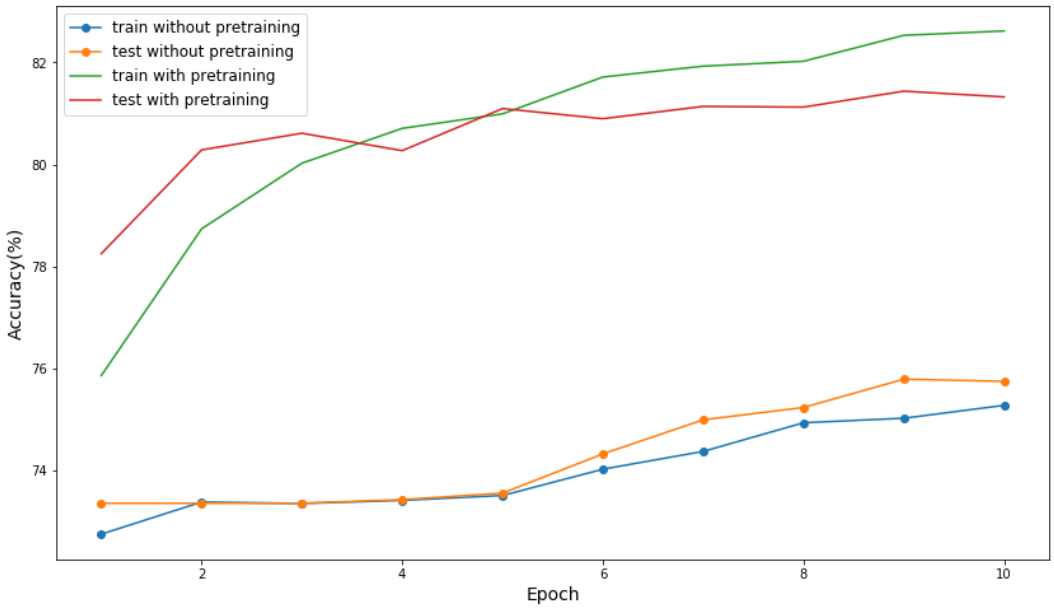
B. Comparison figures

■ Plotting the comparison figures
(RseNet18/50, with/without pretraining)

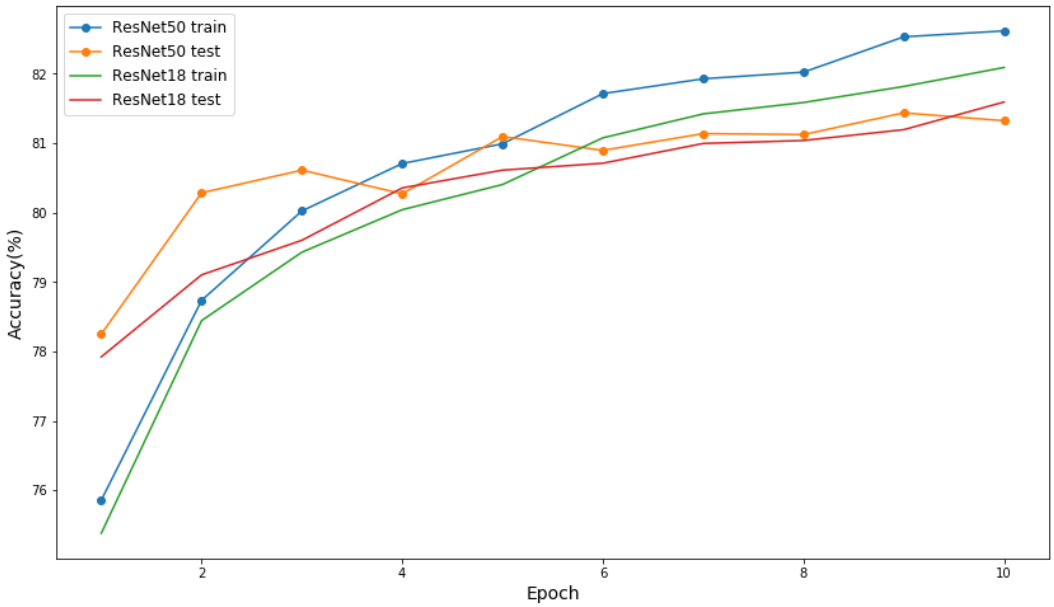
- ResNet18 with/without pretraining



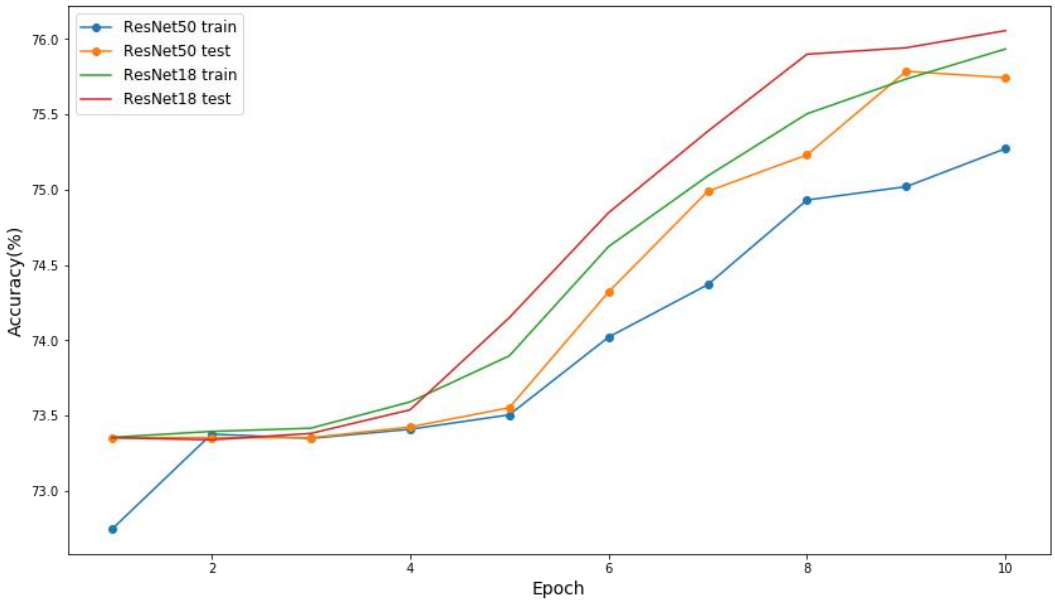
- ResNet50 with/without pretraining



- ResNet18 ResNet50 with pretraining



- ResNet18 ResNet50 without pretraining



4. Discussion (20%)
   A. Anything you want to share
      ■ 結果分析

        在分析 testing accuracy 和 confusion matrix 的 precision，從 accuracy 可以得到(同一 epoch 下) ResNet50 with pretraining > ResNet18 with pretraining > ResNet50 without pretraining > ResNet18 without pretraining，這個結果是符合預期的，但 precision 結果為 ResNet18 with pretraining > ResNet50 with pretraining > ResNet50 without pretraining > ResNet18 without pretraining，故就疾病來

說，ResNet18 with pretraining 的這個結果可能會是最佳的。

- About Paper “Deep Residual Learning for Image Recognition”
  - 為什麼 residual mapping F(x) = H(x) − x 會比較容易優化 ？
    因為在原本 H(x)→x 這樣的優化中，箭頭的兩端都含有未知數，也就是同時兩端都在更新，所以這樣的狀態下要達到平衡是比較困難的；相反的，residual mapping F(x)=H(x)−x→0 是將整個函數經過不斷調整至一個 constant，相對來說較容易達到平衡狀態。

  - 利用這樣的 residual mapping 我們可以讓網路結構趨近於 identity mapping，那不也只是代表整體網路的 performance 可以表現得跟淺層網路 「一樣好」，為何實驗結果可以更優於 「淺層網路」 ？
    原本要讓他跟淺層網路一樣好，是讓後半部的 network 趨近於 identity mapping，如此即可確保部會比淺層網路差。但我們利用 shortcut connection 後，可以讓訊息跨越任何的 network 往下傳，因此就數學的角度來看，他是將整個解的空間變大，因此更可能找到更優解，故會有比淺層網路更好的表現。