1. Introduction

這次作業就是用resnet18，resnet50，用圖片去預測class。

這份作業讓大家學習:

(1)如何製作customized dataset，還有實作data augmentation。

(2)使用別人pretrained/unpretrained的model，並且學習如何train pretrained model。

(3)學習如何繪畫confusion matrix。

2. Experiment setups

A. The details of your model (ResNet)

我的model是參考data.zip中的model architecture實作的，也就是將torchvision.models.Resnet18/50引入model的不同layer中，最後再用flatten攤平後，才做classify。

其中Resnet本體分成9層，分別是

conv1

bn1

relu

maxpool

layer1

layer2

layer3

layer4

classify

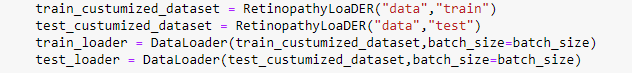
然後basic block/bottleneck block就包在layer1,2,3,4內，可以解決gradiant vanish。

至於說如何知道torchvision.models.Resnet18/50有幾層，可以使用model.modules()查看。



B. The details of your Dataloader

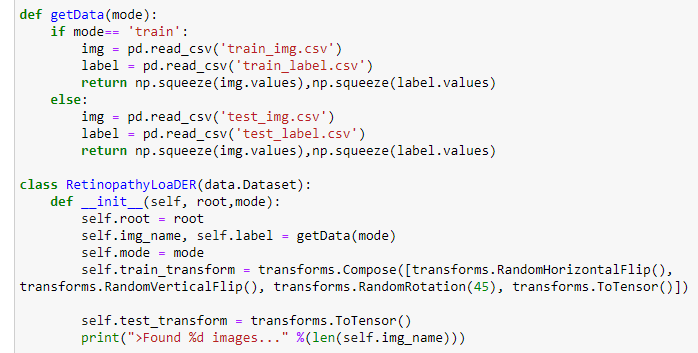
Dataloader也是跟上次一樣，直接呼叫torch的Dataloader。



這次不太一樣的地方，是dataset要自己寫一個custumized\_dataset，不能直接拉TensorDataset用。

至於custumized\_dataset，我的寫法如下

首先是\_\_init\_\_

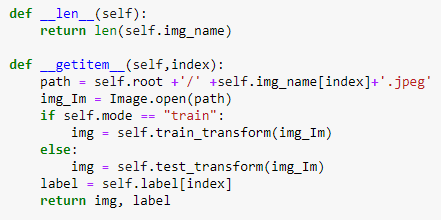


先將dataset的相關資訊存起來，像說位置，檔名，train/test，還有data\_augmentation都先存起來。

data\_augmentation的話，如果是train\_dataset，我就對他做任意水平/垂直翻轉和旋轉後，再對其做nomalize和把dimension變成[C,H,W]

如果是test\_dataset，我則不對他做data\_augmentation，只做nomalize和把dimension變成[C,H,W]。

接下來是\_\_len\_\_和\_\_getitem\_\_，這兩個函式是為了讓Dataloader能正常使用dataset而生的。



\_\_len\_\_就回傳有多少筆data，\_\_getitem\_\_則是讀取照片，對照片做/不做augmentation，再將照片和label return出去。

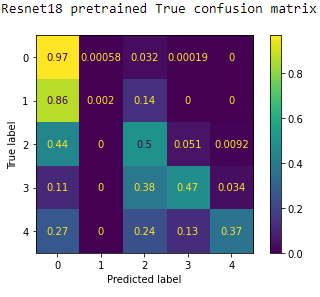
C. Describing your evaluation through the confusion matrix

confusion matrix就是看說ground\_truth和對應prediction分別分佈在哪裡，能更有效知道分佈狀況。

而有時ground truth分佈非常不均時，直接看每格對應的數字可能會失真，應該要做normalization。

舉個例子，假設一個村落正常人10000人，該年正常人死了20人，癌症病患2人，該年癌症病患死了1人。看起來死亡人數正常人比癌症病患多很多，但實際上致死率的話，癌症病患死亡率顯然比較高的。

而這個dataset也有分布不均的特性，True label=0的data特別多，佔了7成，所以最後我對每個ground\_truth做了normalize。

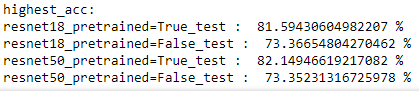


像這張是有pretrained過的resnet18最高test\_acc的confusion matrix，可以看到true label=0預測的特別準，而其他的資料比較少，預測的不太準。尤其是true label=1的，大多都預測成了true label=0。

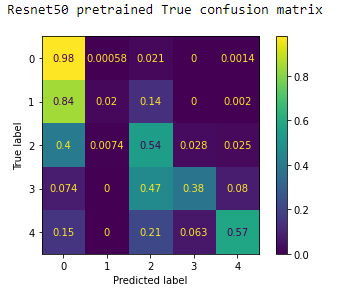
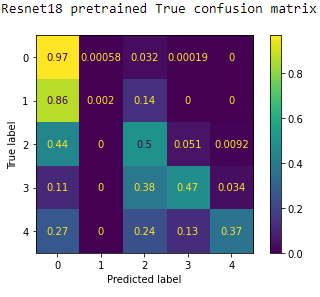
3. Experimental results

A. The highest testing accuracy

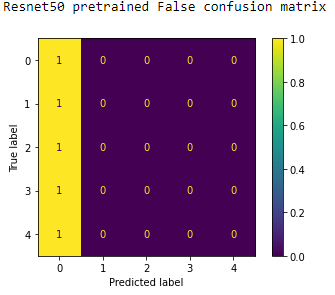
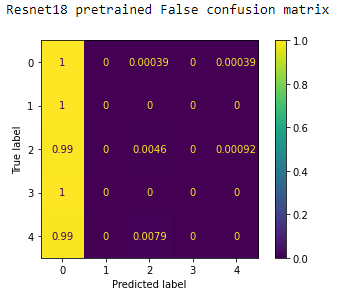
Screenshot



Anything you want to present



從confusion matrix中可以看見，pretrained model在 true label=0時都預測得特別好，反之true label=1時預測的比較差，我猜是因為true label=0占了快7成，而true label=1又跟true label=0時很像。

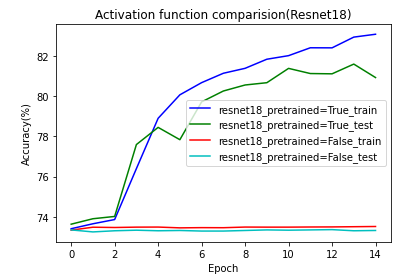


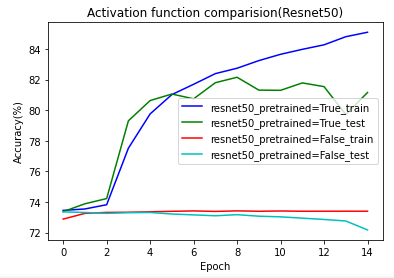
可以看到unpretrained model似乎有偏好只猜0的狀況，畢竟true label=0占了快7成，無腦猜0也會有73%左右，認真猜還不一定能猜贏無腦猜。

B. Comparison figures

Plotting the comparison figures

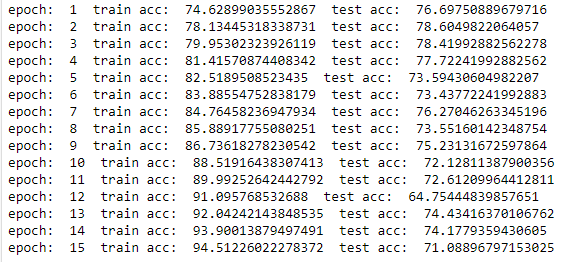
(RseNet18/50, with/without pretraining)





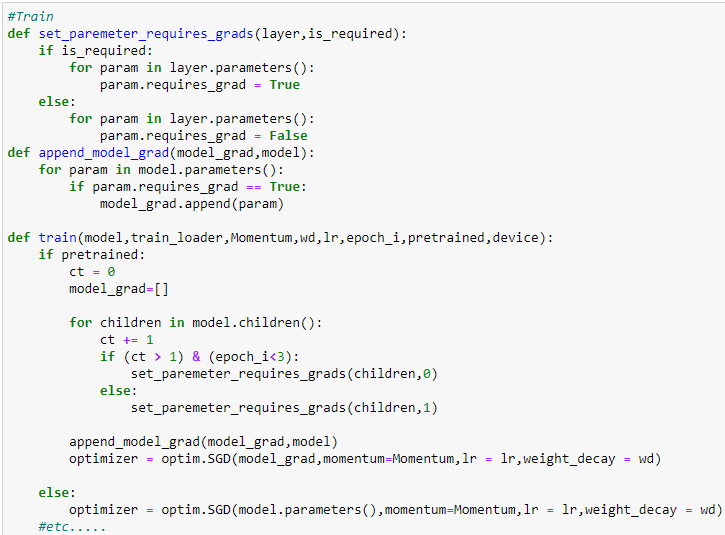
4. Discussion

A. Anything you want to share

1.一開始我完全沒做data augmentation，也沒讓pretrained model的前幾個epoch凍結pretrained backbone，導致我的resnet18的test accuracy在前幾次都有overfitting的現象。

為了解決這個現象，所以我對dataset做了data augmentation。

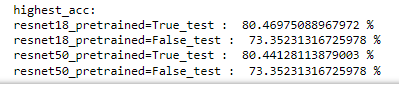
並且在前3epoch只train network的最後一層(flatten)。



作法也就是把model的layer用for迴圈跑過

選擇將要凍結的model，requires\_grad =False，並且參數不要加入optimizer中。

2.一開始做batch size = 4，最高%數圖如下。



為了讓模型能到82%，所以我就調成batch size=32，也就是之前提到的結果。