DLP_LAB4
309553012 黃建洲

1. Introduction

這次作業讓我們分別嘗試了resnet網路的手刻架構以及使用pytorch的內建函式建立model。並調整hyper parameter來對兩個model分別測試出不同的結果，整理成圖表以及分析confusion matrix。

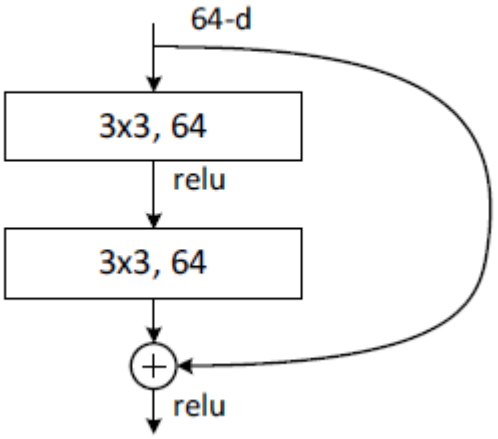
2. Experiment Setup

a. The detail of your model(ResNet)

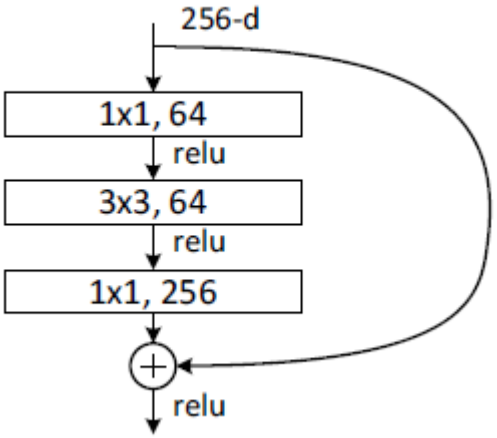
手刻ResNet的部分，參照網路上的範例以及spec給予的參考資料

(1) 首先分別定義18 layers版本以及50 layers版本的block。

basic block(for 18 layers):



Bottleneck block(for 50 layers):



(2) 定義網路架構，依照參考資料定義初始的幾層conv, activation, pooling和batch normalization層後，根據網路參數將數個block定義為一個layer，並且根據input output size的不同決定是否要在residual的部分進行downsampling。

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

The overall architecture for all network

Pretrained ResNet的部分則從torchvision下載網路架構以及pretrain好的參數後，根據api提供的layer名稱一層一層重新定義並forward，並且在最後的fully connected layer要重新定義output classes，以符合我們最後要分類的種類數量。

b. The detail of your data loader
Dataloader則根據test和train有不同的做法。test dataset由於不需要transform操作，因此直接根據csv給予的檔名以及路徑將圖片load進來之後，給予label值並回傳。
Train dataset則需要從torchvision中的transform函式庫先定義好需要的data augmentation方式，我使用的為水平與垂直翻轉的操作，總共應會讓一張圖片增強為4張圖片的量。最後同test dataset的處理方式，將增強後的資料與label一併回傳。

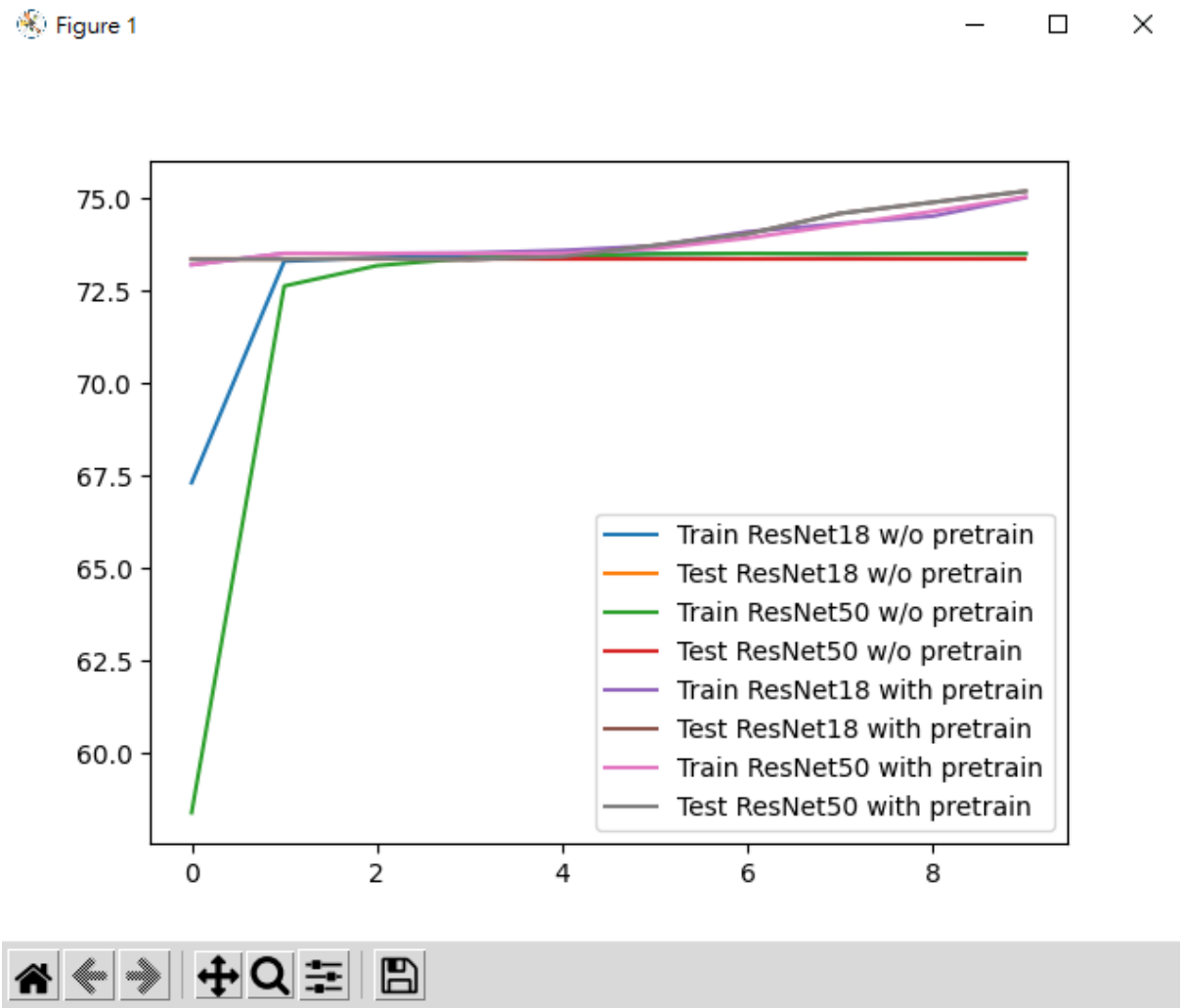
c. Describe your evaluation through the confusion matrix

3. Experiment Result

a. The highest testing accuracy

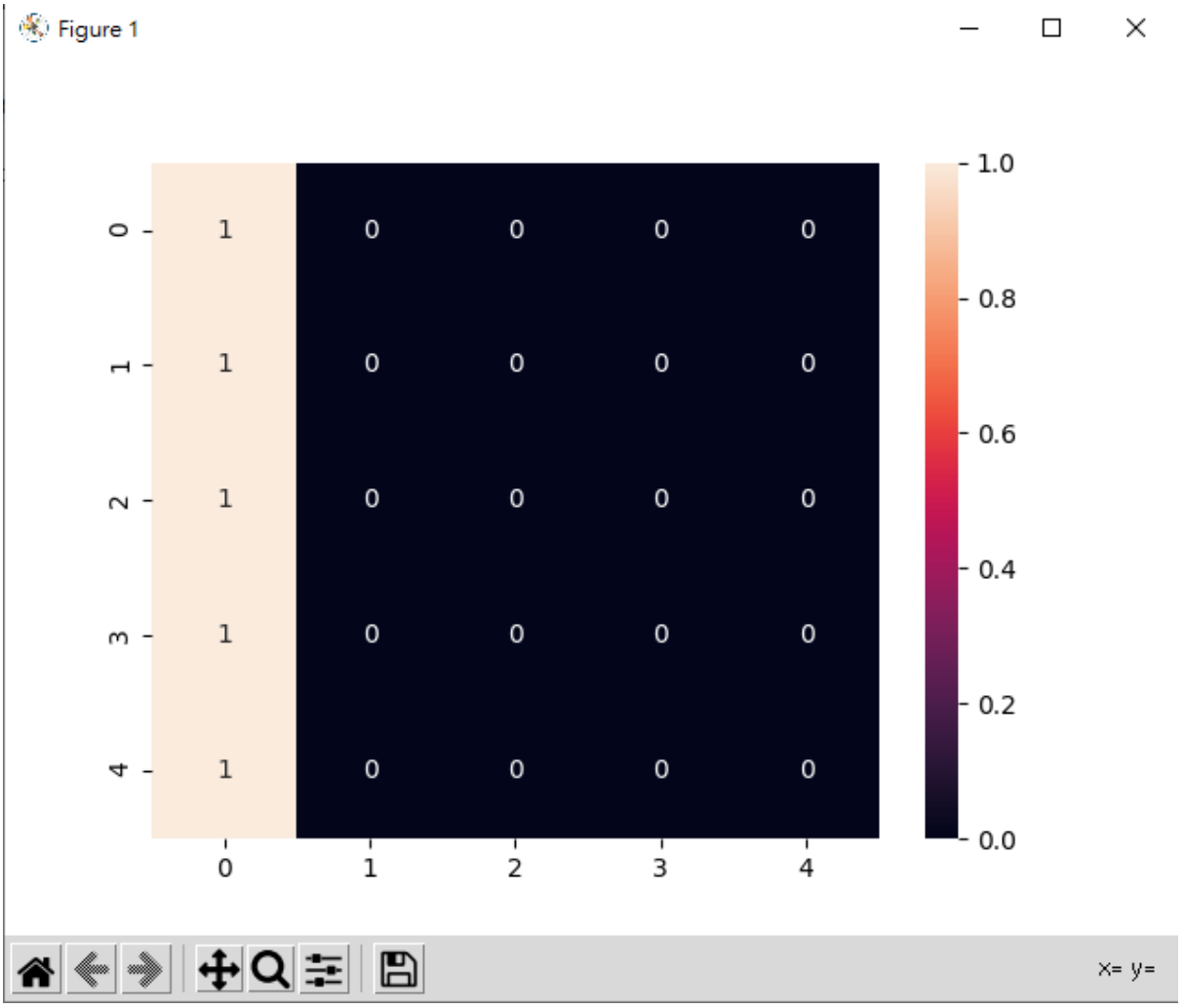
b. Comparison Figure

i. test/train accuracy plot(epoch = 0, lr = 0.0001, optimizer = SGD)

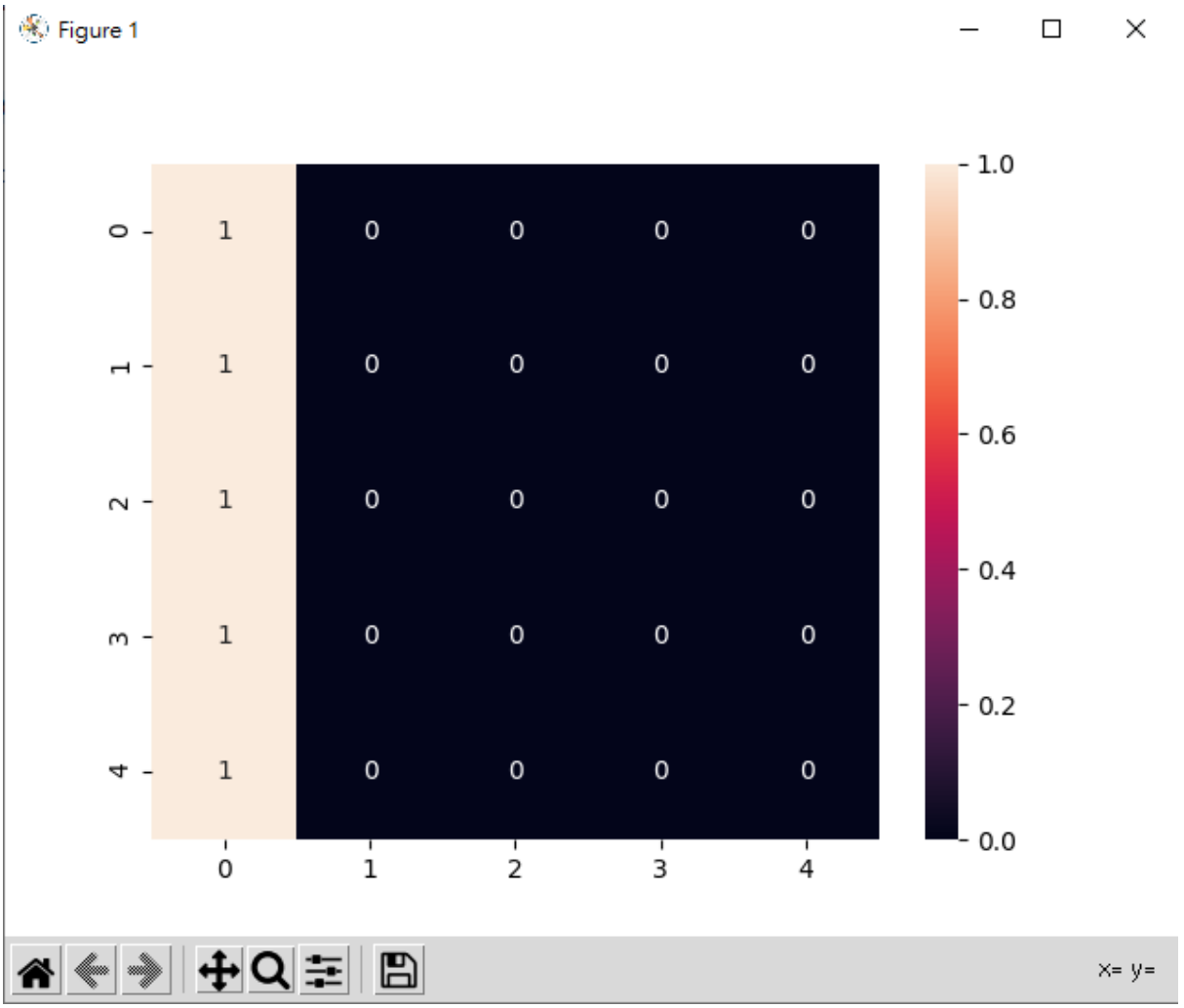


ii. confusion matrix

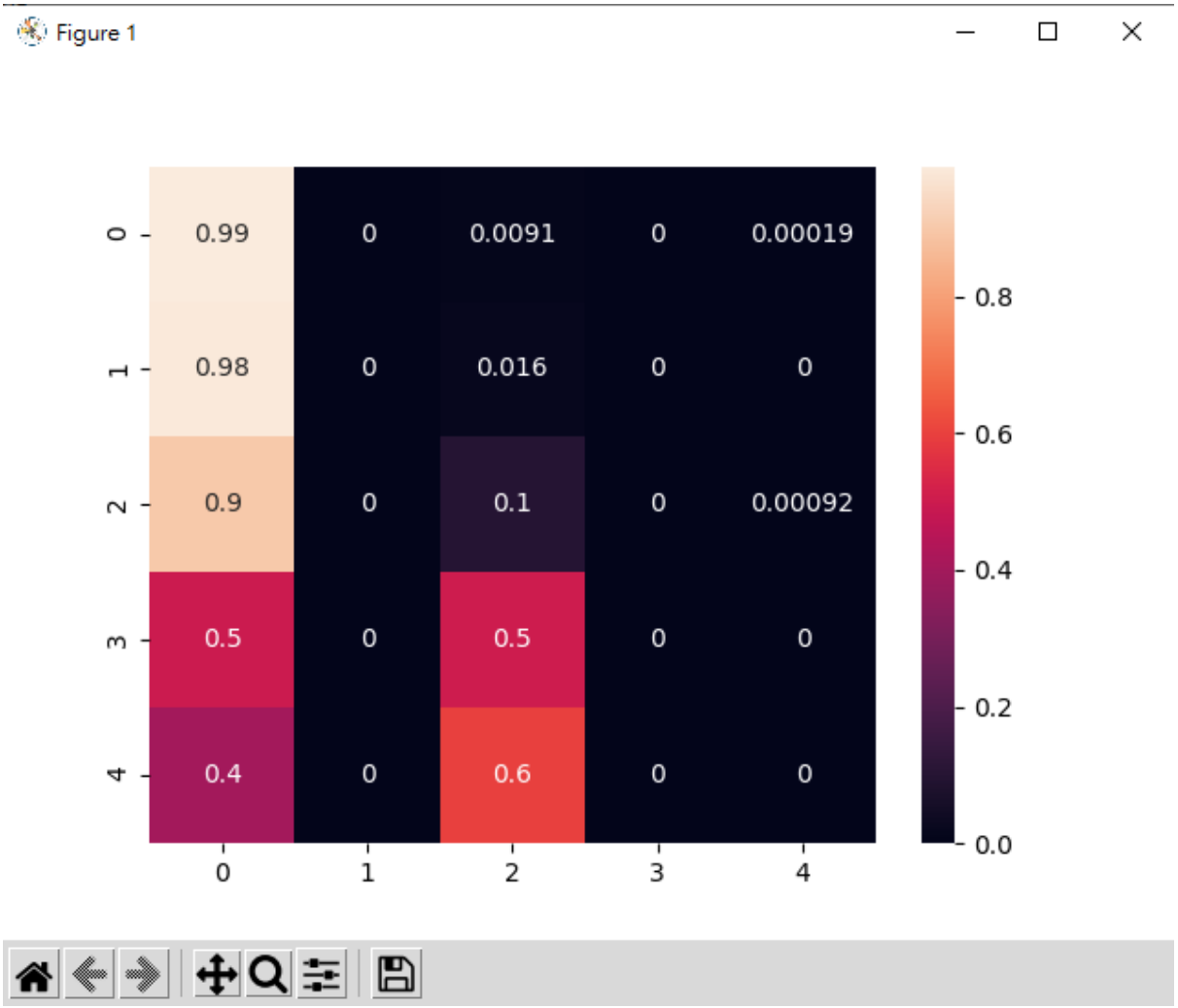
(1) ResNet 18 without pretrain:



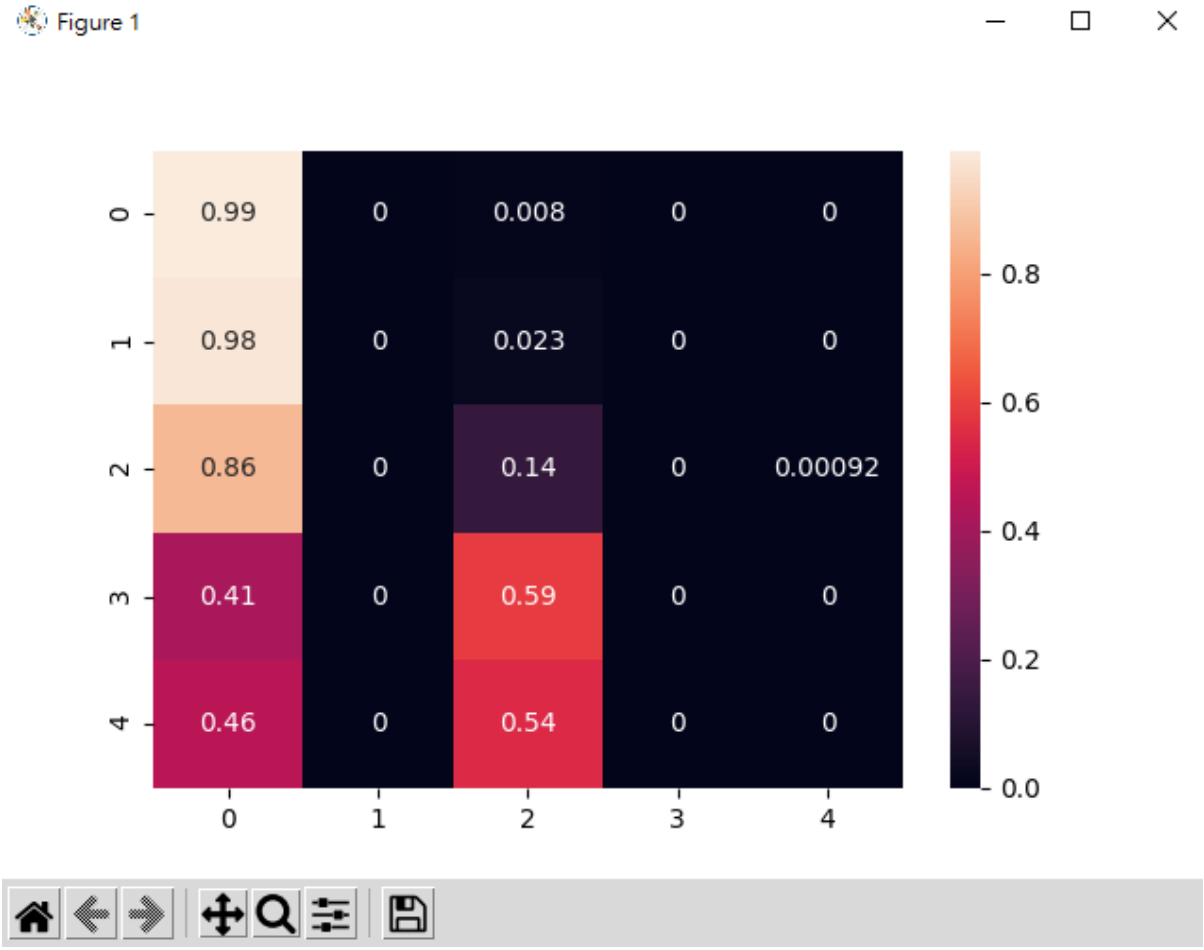
(2) ResNet 50 without pretrain:



(3) ResNet 18 with pretrain:



(4) ResNet 50 with pretrain:



4. Discussion

由於ResNet相較於上一次作業所使用的網路，在深度上增加了非常之多。因此在訓練上所需要使用的時間也大幅增加，以我們所使用的配備以及可用的時間，頂多也只能訓練到幾十個epochs，並且在batch size的使用上也有限制(因GPU的記憶體有限)。參數的配置上，我處理最久的部分是手刻ResNet的參數，在進行訓練時test accuracy都會維持一個定值，我本來推測是因為output出來的值超過crossentropy的範圍，導致在進行分類時產生錯誤。因此我嘗試對resnet的fully connected layer進行改動。但結果並沒有改變，而後我對pretrain的結果以及所有的confusion matrix觀察後發現，test accuracy會卡在同一個數字的原因應該是因為ground truth

中label為0的比率極高，因此即使網路全部都猜0也會有一定的準確率。