DEEP LEARNING

Homework 2

309511045 黃麟凱

1. MNIST

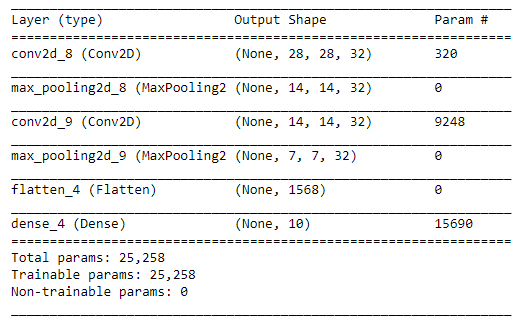
Training data : 50000

Validation data : 5000

Testing data : 10000

Network architecture：

這次使用的架構如下圖所示，首先圖片會經過一層convolution的計算，接著再對其做max\_pooling，讓feature變少，並重複一次。當做完convolution之後將feature攤平，並經過dense層把數量降成10，且activation function選擇softmax，即完成這次的Network architecture

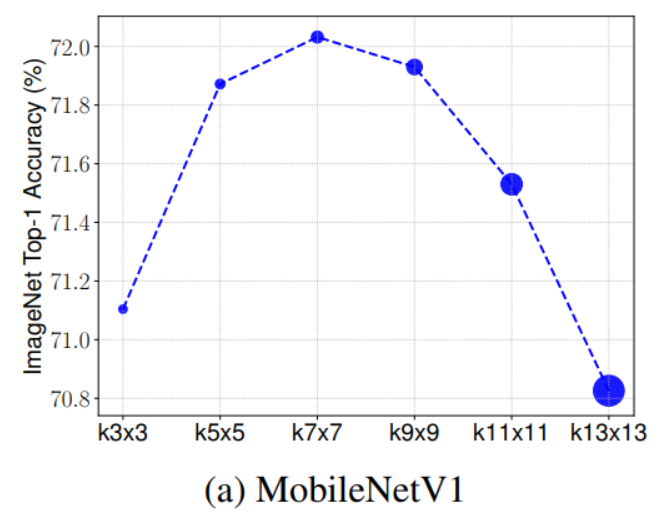


Analysis with different stride size and filter size (all train for 100 epochs)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Kernel size | stride | Train acc | Test acc | Val acc | Epochs Train acc > 95% |
| (3,3) | (1,1) | 99.99% | 99.02% | 99.04% | 2nd |
| (5,5) | (1,1) | 99.99% | 99.22% | 99.30% | 2nd |
| (7,7) | (1,1) | 99.99% | 99.28% | 99.18% | 2nd |

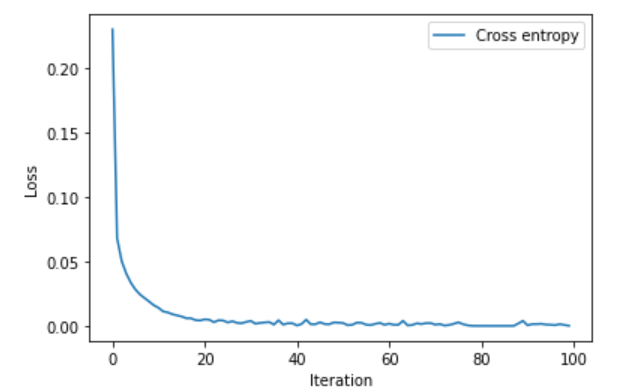
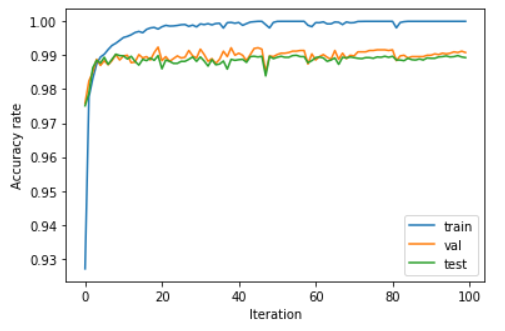
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Kernel size | stride | Train acc | Test acc | Val acc | Epochs Train acc > 95% |
| (3,3) | (1,1) | 99.99% | 99.02% | 99.04% | 2nd |
| (3,3) | (2,2) | 99.99% | 98.49% | 98.66% | 2nd |
| (3,3) | (3,3) | 99.63% | 96.65% | 96.38% | 7th |

若調整kernel size 的大小可發現，當size變大整體的performance會變好一點。但若再更大的話，反而會有反效果。上網找了一下data發現kernel size的選擇的確如上述提到那樣，大過一個狀況後，效果就變得不好。從下圖可發現，MobileNet這個網路的kernel size若從3x3 -> 7x7，整體的accuracy有上升，但到9x9，甚至是13x13時，整體的accuracy就一直下降。由此可見kernel size的選擇是很重要的

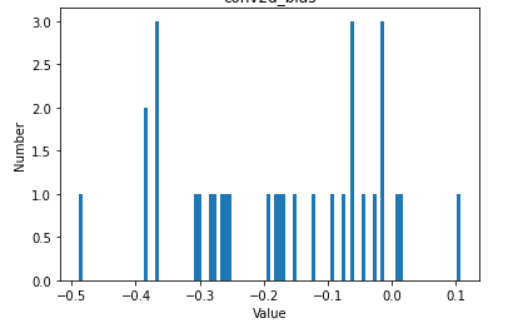
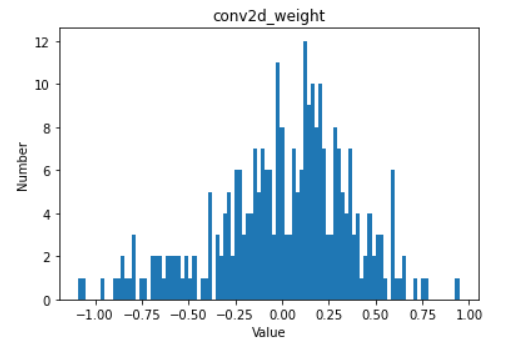


另一方面，若調整strides的大小，也會對整體的accuracy有影響，當stride上升時，accuracy也開始下降。原因可能在於我的kernel size只設定3x3，因此當strides太大時，導致model沒辦法有效的把feature提取出來。

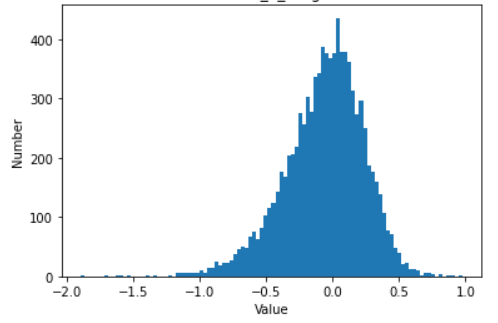
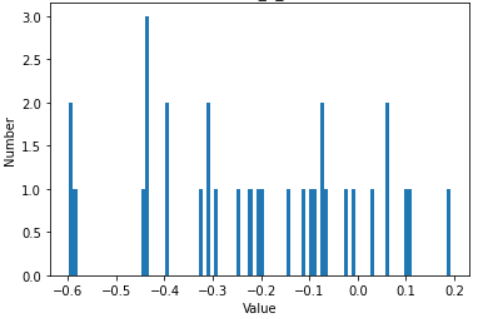
Training Accuracy Learning Curve



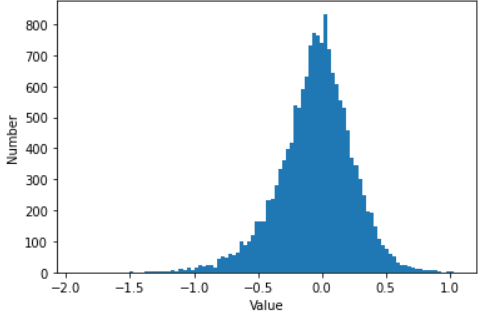
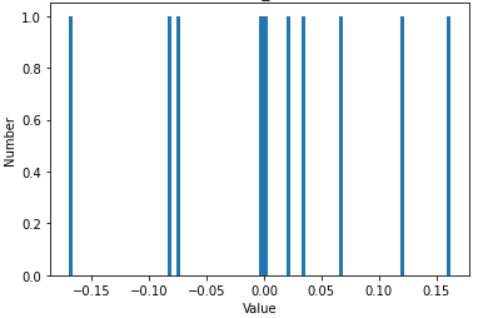
Histogram of Conv1：weights & bias

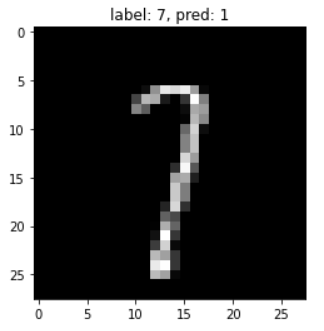
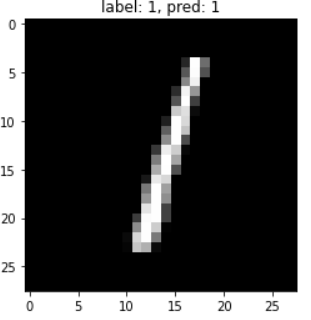


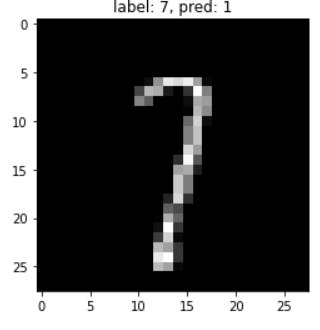
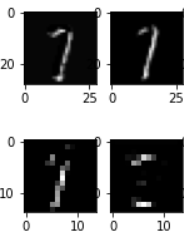
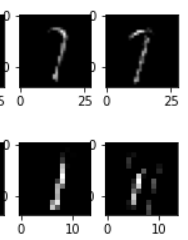
Histogram of Conv2：weights & bias

Histogram of Dense：weights & bias

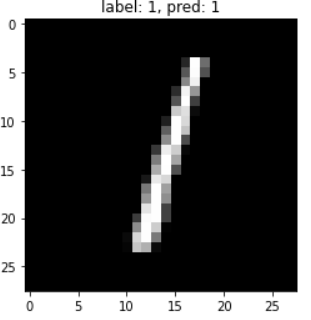
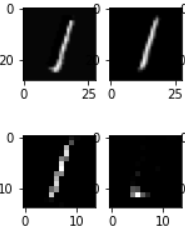
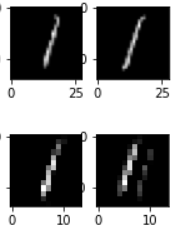
 



Conv2

Conv1

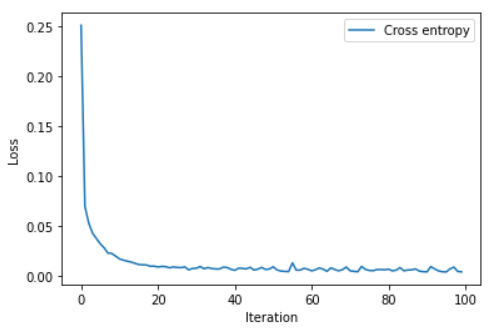
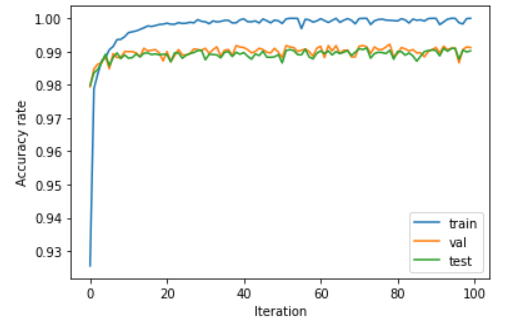


Conv1

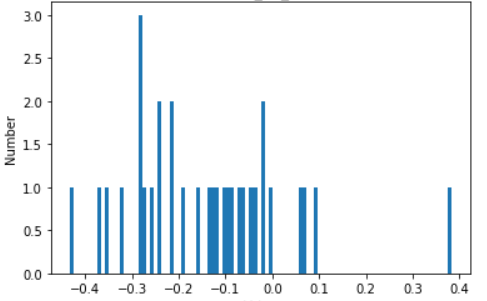
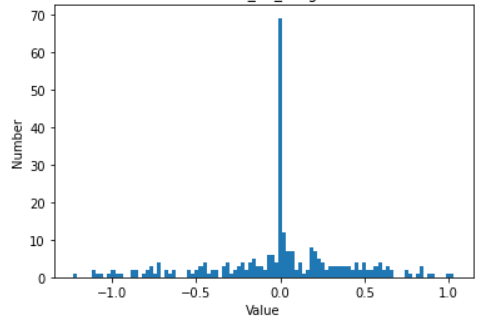
Conv2

在辨識1跟7時，的確有部分是很相似的，從feature map 也可以看出經過convolution之後的結果有部份也很相似(如紅框的地方)，但還是有差異的，所以整體的錯誤辨識結果並沒有很高

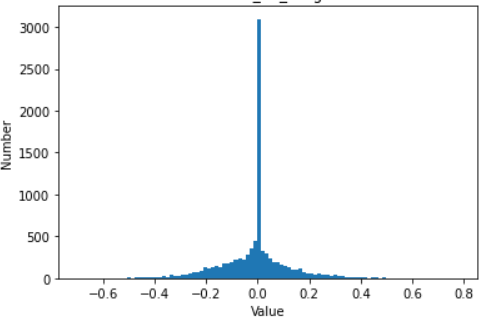
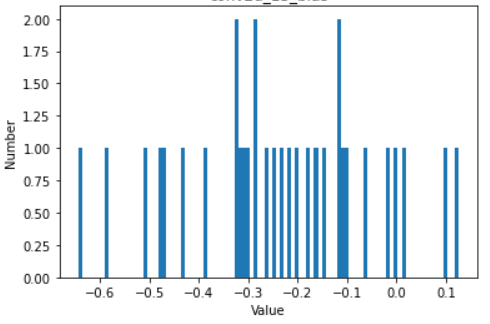
Training Accuracy Learning Curve



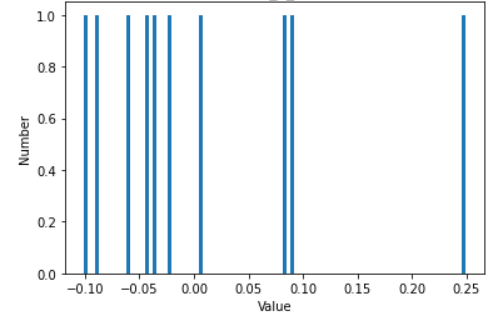
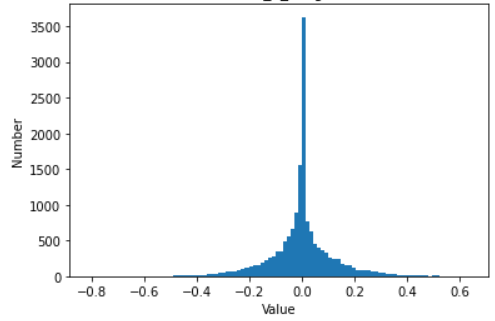
Histogram of Conv1：weights & bias



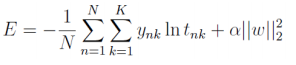
Histogram of Conv2：weights & bias

Histogram of Dense：weights & bias



從結果可以看出，經過L2 regularization後，整體的weight有明顯被限縮的趨勢。原因可以從公式看出，在原本的loss function上，加上了一個權重的參數，如此一來為了讓Error變小，不外乎除了需要讓predict的結果越接近answer之外，還要讓weight的大小變小才行。



1. CIFAR-10

Training data : 50000

Validation data : 5000

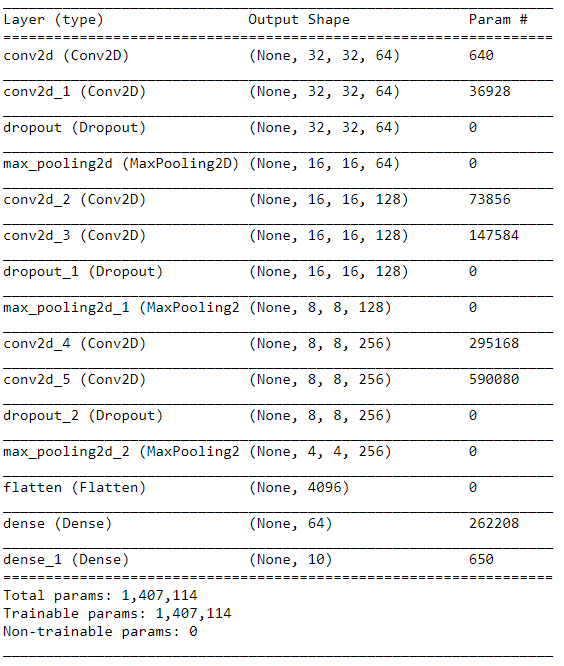
Testing data : 10000

Network architecture：

這次所使用的model架構是參考VVG16的model架構。該架構的特色是，以連續使用兩個convolution加上maxpooling為一組，並這樣一組一組的接下去，如下圖，其中當model越深，convolution 的kernel depth就會越大。



而這次建構的model如下圖：



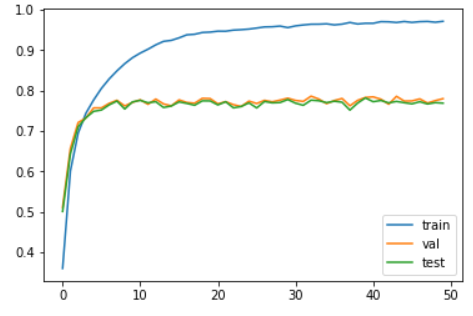
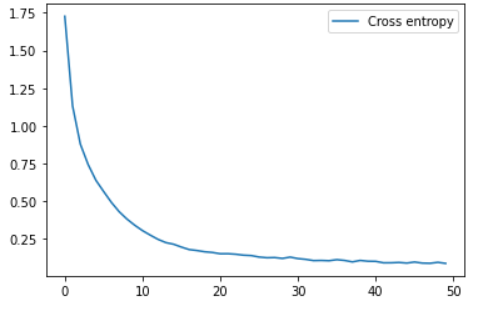
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Kernel size | stride | Train acc | Test acc | Val acc | Epochs Train acc > 95% | s/epochs |
| (3,3) | (1,1) | 97.11% | 76.89% | 78% | 24th | 18s |
| (5,5) | (1,1) | 97.26% | 72.64% | 73.12% | 23nd | 33s |
| (7,7) | (1,1) | 11.02% | 10.16% | 10.22% | XX | 47s |

Analysis with different filter size (all train for 50 epochs)

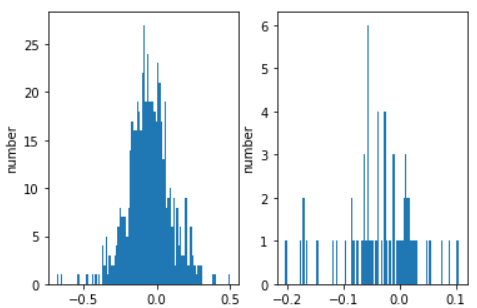
若調整kernel size的大小，發現3x3 -> 5x5的時候，雖然train的accuracy有上升，但Test以及validation的都下降不少，而且整體的訓練時間變長很多，而5x5 -> 7x7的時候，除了訓練時間更長以外，整個model連收斂都沒辦法，一直處與震盪狀態。

而調整stride的大小時發現，由於本身的網路比較深，且輸入圖片的size並沒有很大，導致光model都沒辦法成功地建立起來，因此就沒有附上實驗的結果圖。其原因在於，假設一開始的圖片是32x32，那經過stride為2的kernel後，其輸出會為16\*16的數值。導致做個幾次就讓feature size變為1，而沒辦法繼續做下去。

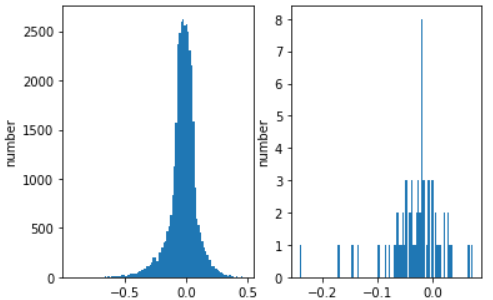
Training Accuracy Learning Curve

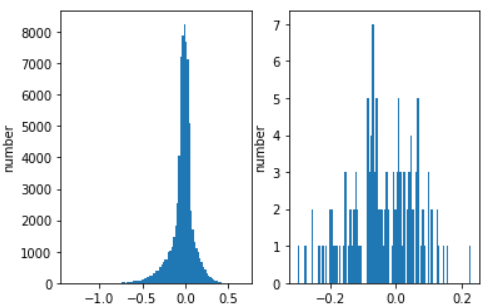
Histogram of Conv1：weights & bias



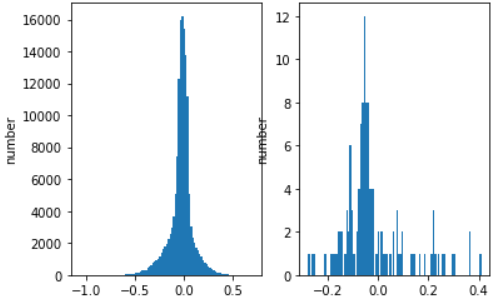
Histogram of Conv2：weights & bias



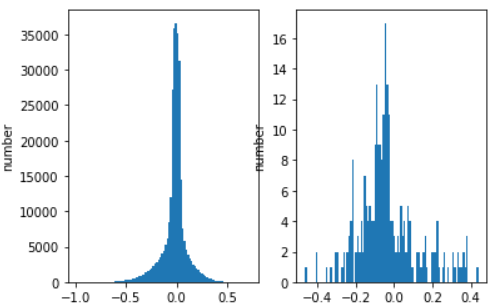
Histogram of Conv3：weights & bias



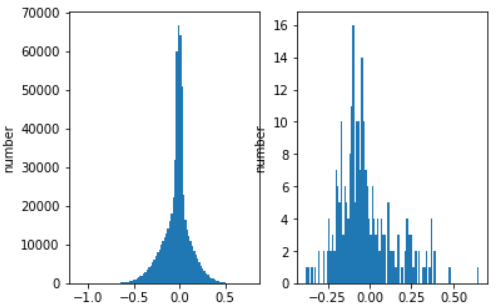
Histogram of Conv4：weights & bias



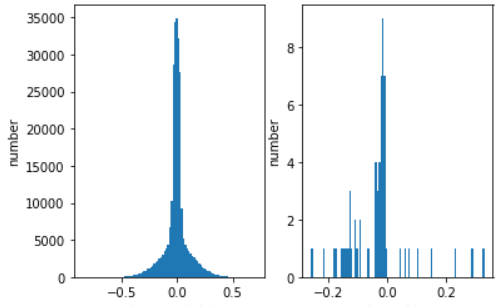
Histogram of Conv5：weights & bias



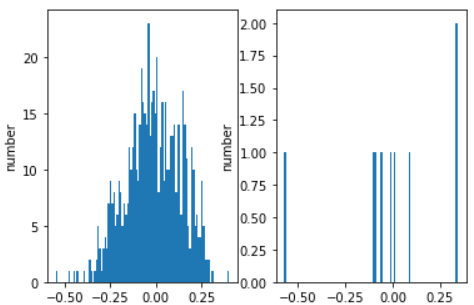
Histogram of Conv6：weights & bias

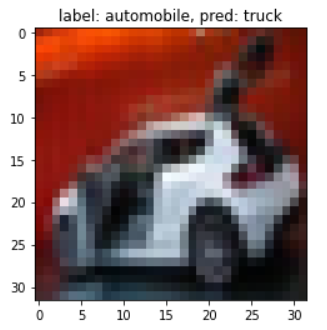
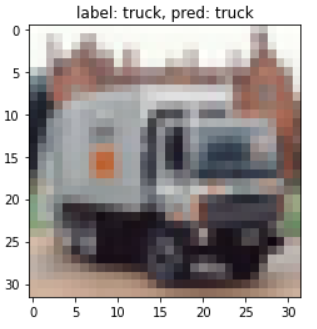


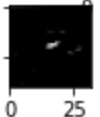
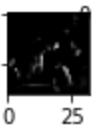
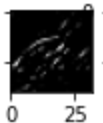
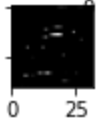
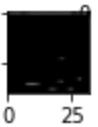
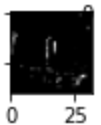
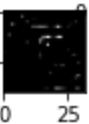
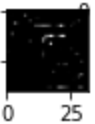
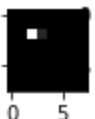
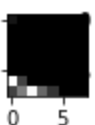
Histogram of Dense1：weights & bias



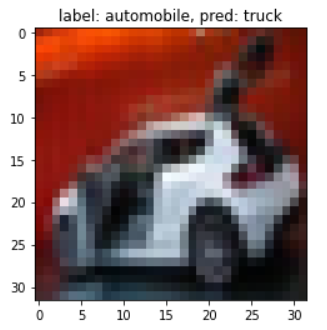
Histogram of Dense2：weights & bias

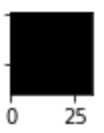


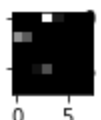
* 1. 

Conv1





Conv2

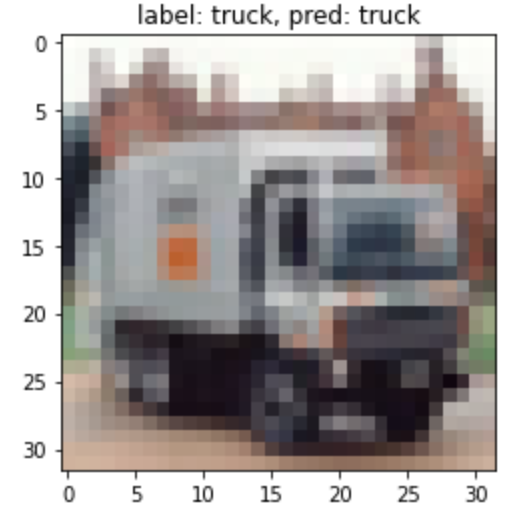


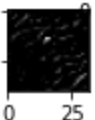
Conv6

Conv2

Conv1

Conv6

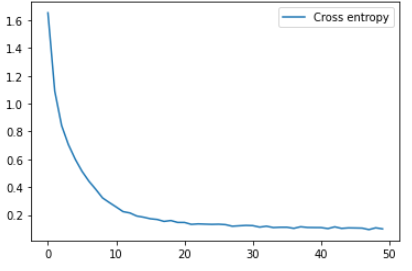
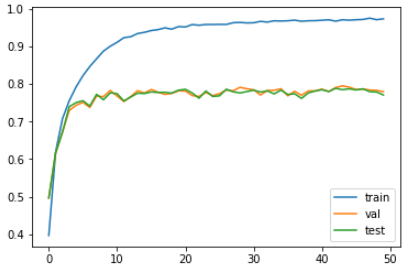




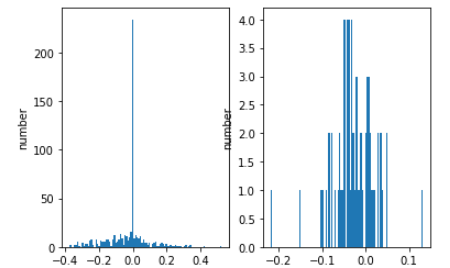


當圖片經過第一層的convolution之後，還能稍微看出圖片中的內容是甚麼。但越往後面做，就越來越模糊，有點難比較出其中的差異，沒辦法像MNIST那樣仍保有能分辨的feature map，所以這題我沒辦法藉由feature map比較出為甚麼model會辨識錯誤。但就label來看是有辦法分析的，因為automobile 跟truck在本質上本來就很類似，所以辨識錯很是很正常的。

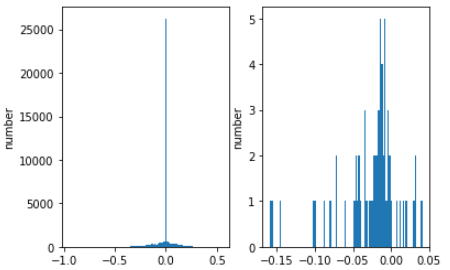
Training Accuracy Learning Curve



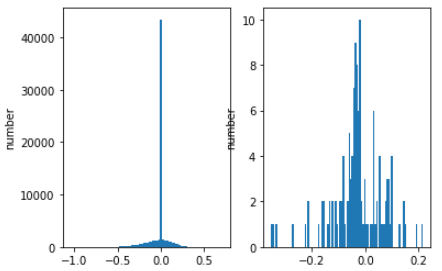
Histogram of Conv1：weights & bias



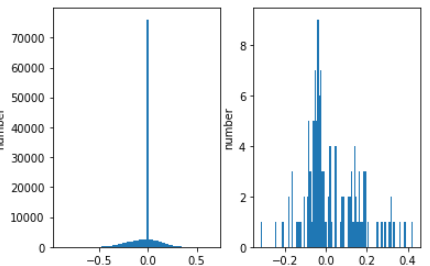
Histogram of Conv2：weights & bias



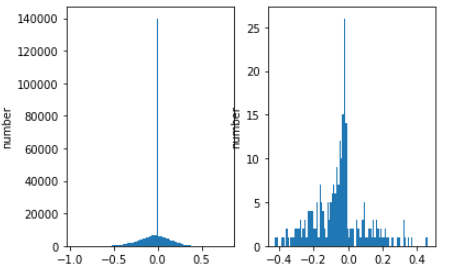
Histogram of Conv3：weights & bias



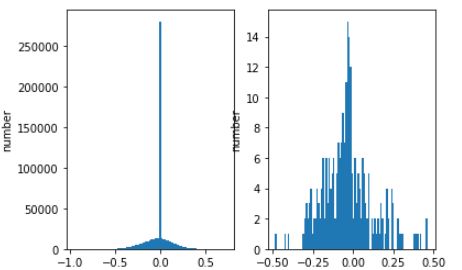
Histogram of Conv4：weights & bias



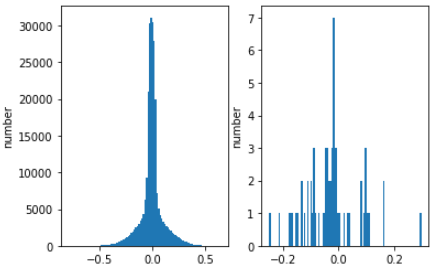
Histogram of Conv5：weights & bias



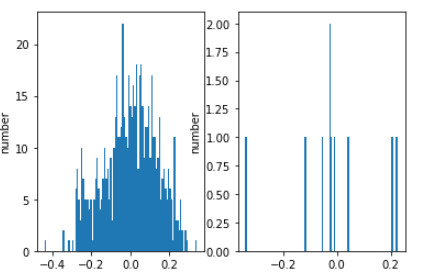
Histogram of Conv6：weights & bias



Histogram of Dense1：weights & bias



Histogram of Dense2：weights & bias

****

結果跟1-4的時候差不多，因為有限制了weight的大小，因此weight的分布比較不會那麼大。

關於preprocess，首先我先將圖片從彩色的轉為灰階，其作法是將原本圖片的矩陣，乘上[0.2989, 0.5870, 0.1140]的矩陣，這數字的由來是，參考網路上的。做完灰階處理後，為了能讓神經網路能方便訓練，需要將數值normalization。且由於灰階的圖片其數值範圍為0~255，因此將整個矩陣的數值再除以255即可。如此一來就為乘了預處理的動作。

Reference：

<https://medium.com/ai-academy-taiwan/cnn%E8%AB%96%E6%96%87%E5%B0%8E%E8%AE%80-mixconv-mixed-depthwise-convolutional-kernels-4357ccdbfe6c>

<https://neurohive.io/en/popular-networks/vgg16/>

<https://stackoverflow.com/questions/12201577/how-can-i-convert-an-rgb-image-into-grayscale-in-python>