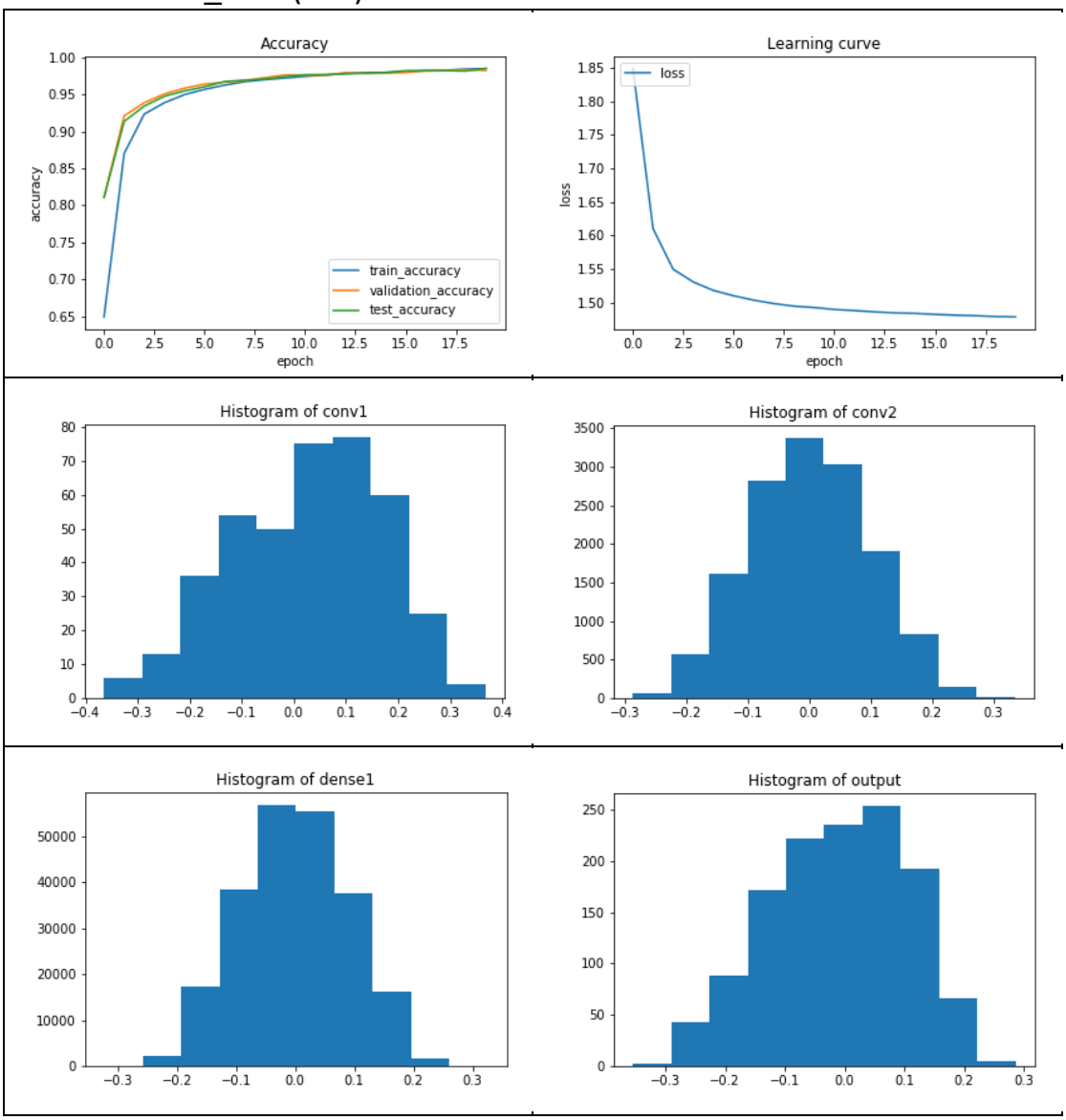
Deep Learning HW2

A061714 藍國瑞

1 Using CNN for Image Recognition

1.1

- CONV_filter(5x5)

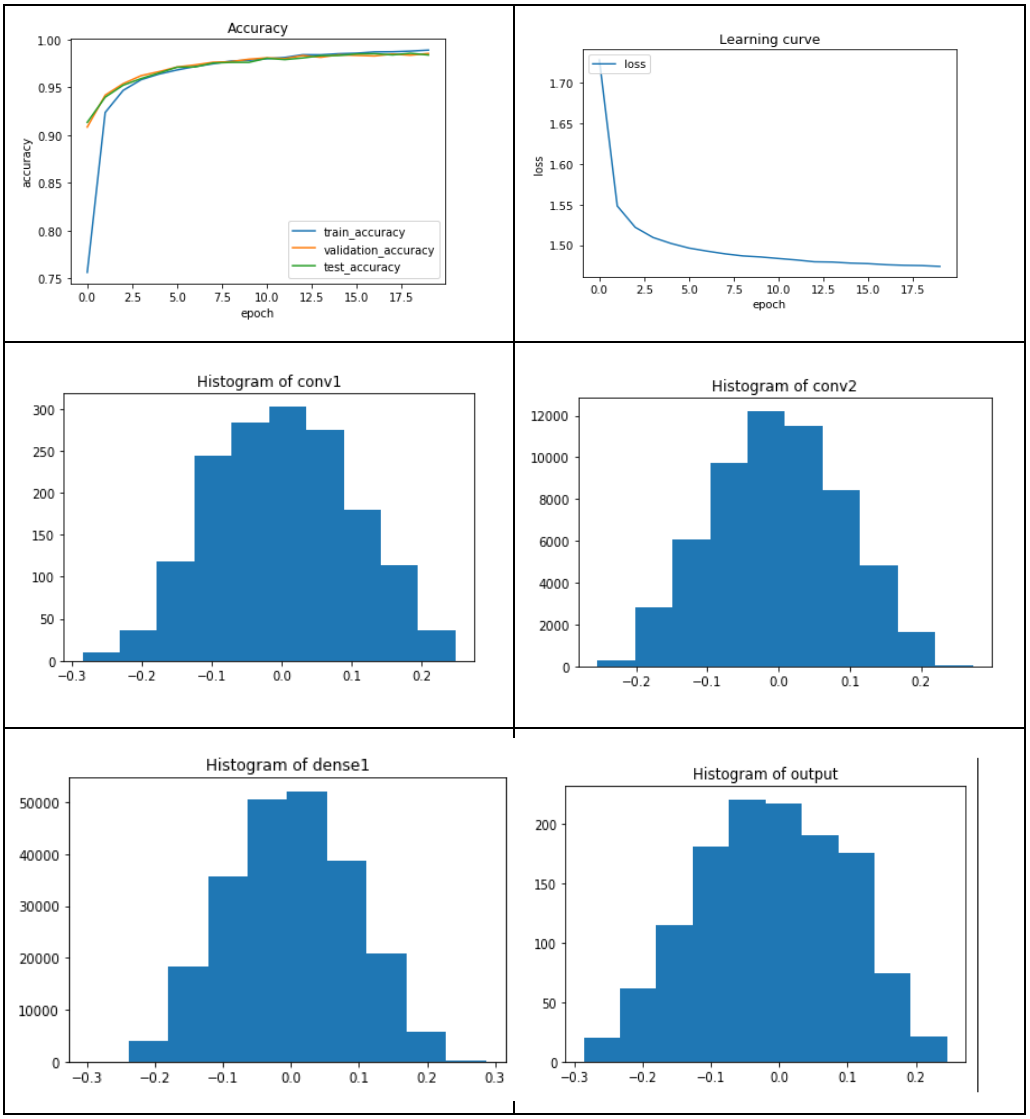


因為考慮到我筆電的性能，所以我只有加兩層捲積層和一層隱藏層，並且捲積層的 filter size 都是[5,5]，而隱藏層的 weight matrix 則是[1784,128]，epoch 的次數則是訂為 20。可以看出 weight 的 histogram 在各層的結果主要是呈現一個 mean=0 的 normal distribution。

有一點比較特別的是，因為我如果在 train 完一個 epoch，接著一次把 55000 筆

training data 傳給 model 去算 accuracy 和 loss 的話，我的電腦會因為性能不夠，所以會當機，因此我 training 的 accuracy 和 loss 的算法是，在每個 epoch 中，我就直接在每個 batch 之間去算他的 acc 和 loss，也就是假設每個 epoch 需要 550 個 batch，那我就會在這之間算出 550 個 acc 和 loss，最後再將這 550 個 acc 和 loss 取平均來當作該次 epoch 的 acc 和 loss。但這會有一個問題是，當在第一個 epoch 中前面幾次 batch 的 acc 會特別的糟糕，因為前面幾次的 model 還不是一個完整的 model，所以在一起取平均之後，第一次的 epoch 就會被拉低平均，才會造成開始前面幾次 epoch 的 training accuracy 會比 test accuracy 和 validation accuracy 還低的情況。

- CONV_filter(10*10)

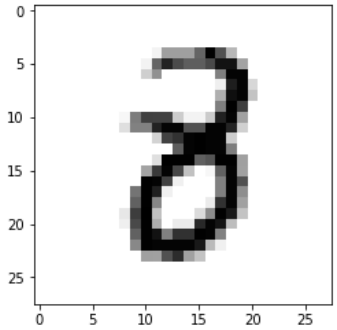


另外，我試著把捲積層的 filter size 都改為[10,10]雖然這樣所花費的時間比較長，但最後得到 weight 的 histogram 的分布看起來比較理想。

1.2

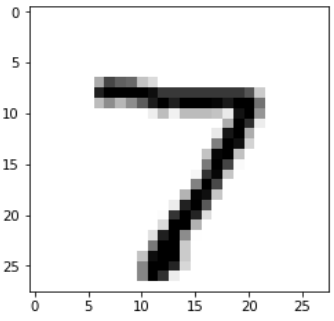
- CONV_filter(5x5)

Label=8, predict=3



可以看到以上這個圖，正確應該是 8，但 model 確判斷成 3，不過這個字的確是比較潦草，因為如果是工整的 8 的話，左上角的那個缺口應該是要連起來的才對，而這邊因為是有缺口的，所以 model 就覺得他個 3 一樣，左上角也有缺口這項特徵，所以才會造成誤判。

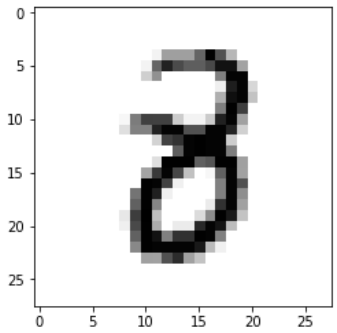
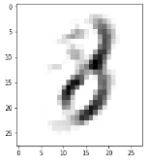
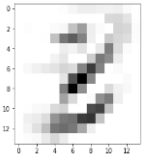
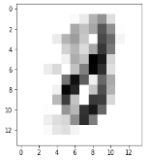
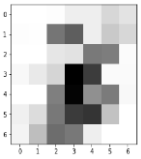
Label=7, predict=7



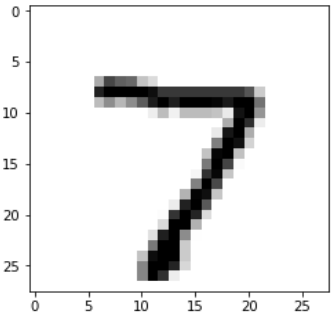
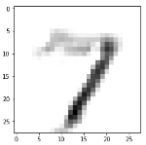
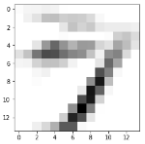
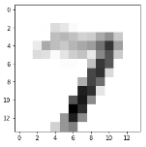
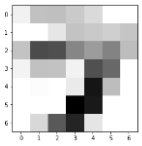
這個 7 因為特徵就比較單純，所以比較不容易和其他數字搞混。

1.3

- CONV_filter(5x5)

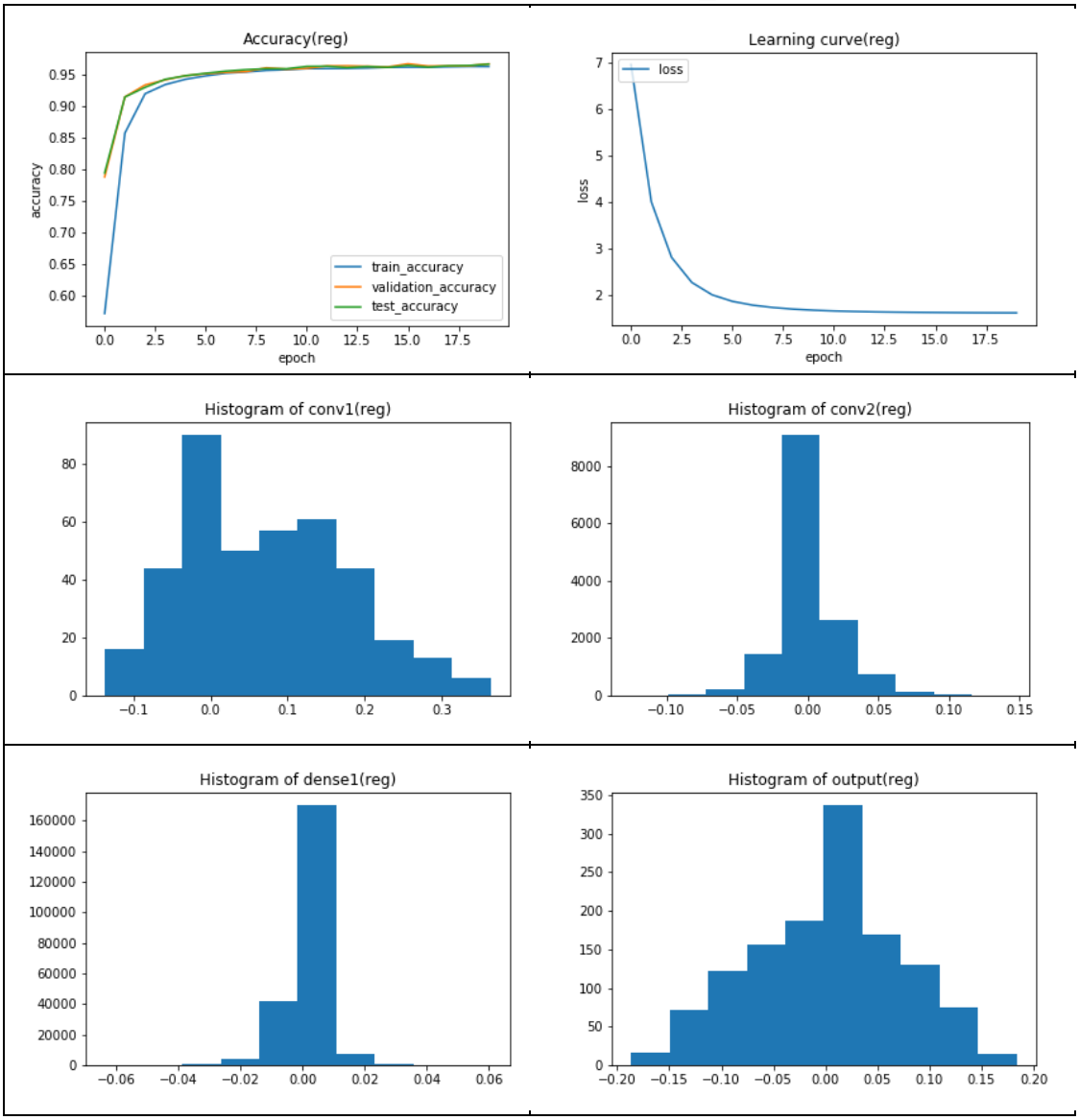
<table>
<tr><td rowspan="4">  </td><td>First CONV</td><td>Second CONV</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>First maxpooling</td><td>Second maxpooling</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

CONV 的 pixel 數目比較大，所以相較於 pooling 看起來就比較連續，而從第一層捲積到第二層捲積，可以看到第二層捲積就提取出比較重要的判斷特徵，而其他不是想要判斷的特徵的部分就變得比較不清楚。

<table>
<tr><td rowspan="4">  </td><td>First CONV</td><td>Second CONV</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>First maxpooling</td><td>Second maxpooling</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

1.4

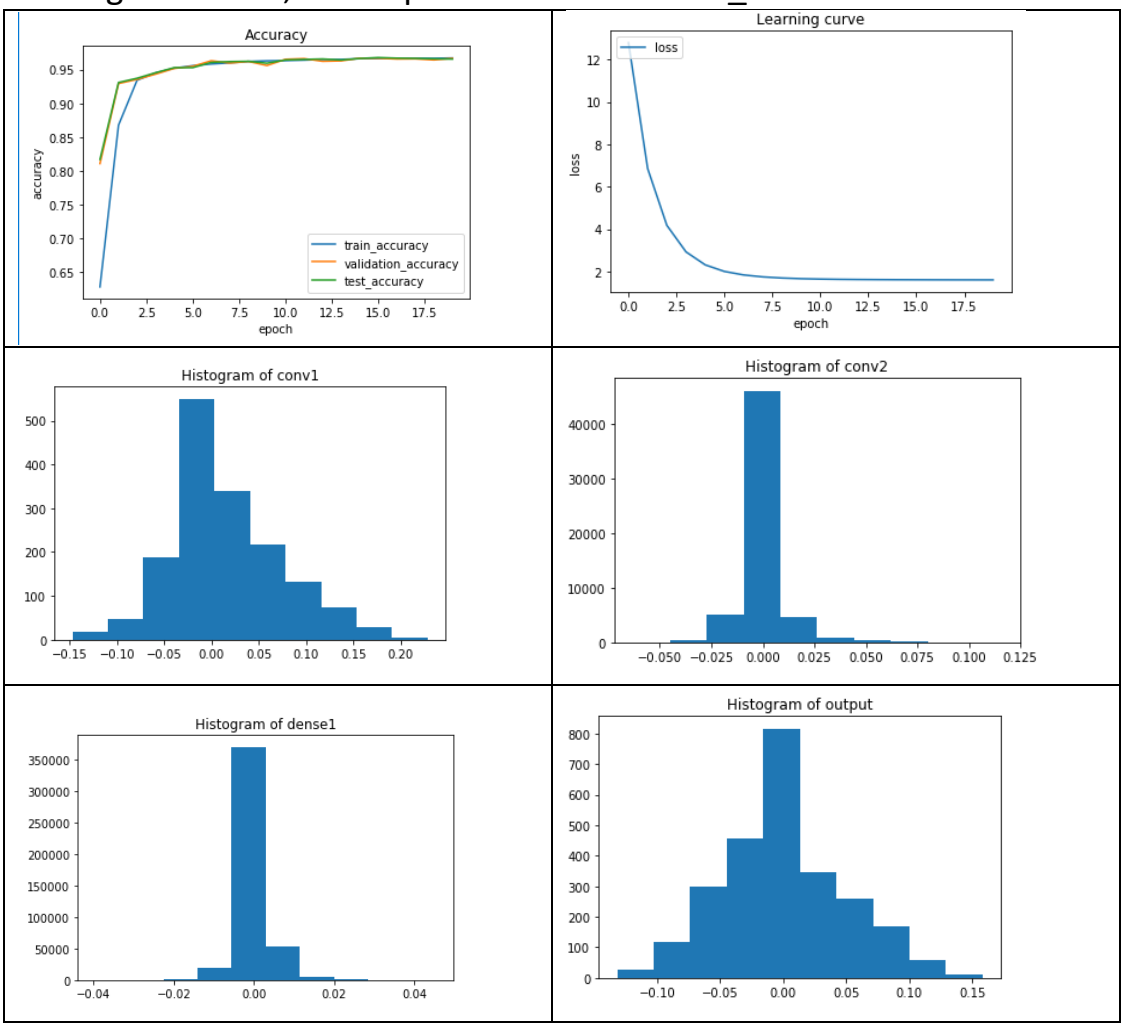
- Regularization, with alpha=0.008 and CONV_filter 5X5



因為我們在 loss_function 當中加了 L2 norm 的 regularization，所以會讓每一層中大部分的 weight 的值都是 0,也就可以看出在 weight 的 histogram 中，0 的數量會特別高。

通常做了這種 L2 norm 的 regularization 可以抑制 model 當中 overfitting 的情況，因為 weigth=0 的神經元變多，也就代表神經網路中被 deactivate 的神經元變多，有點像是 dropout 的意思，可以減少 overfitting 增加 generalization。

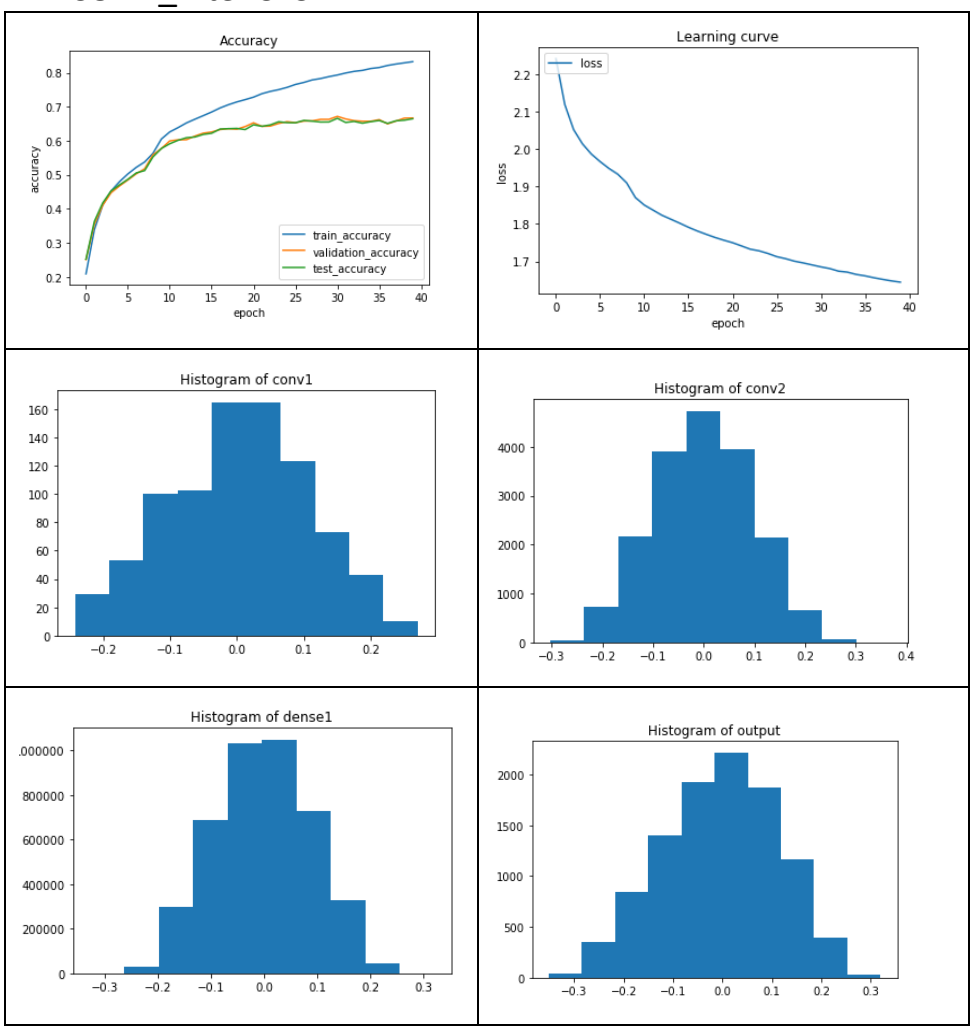
- Regularization, with alpha=0.008 and CONV_filter 10X10



2 Preprocessing before using CNN work for Image recognition

2.1

- CONV_filter 3x3

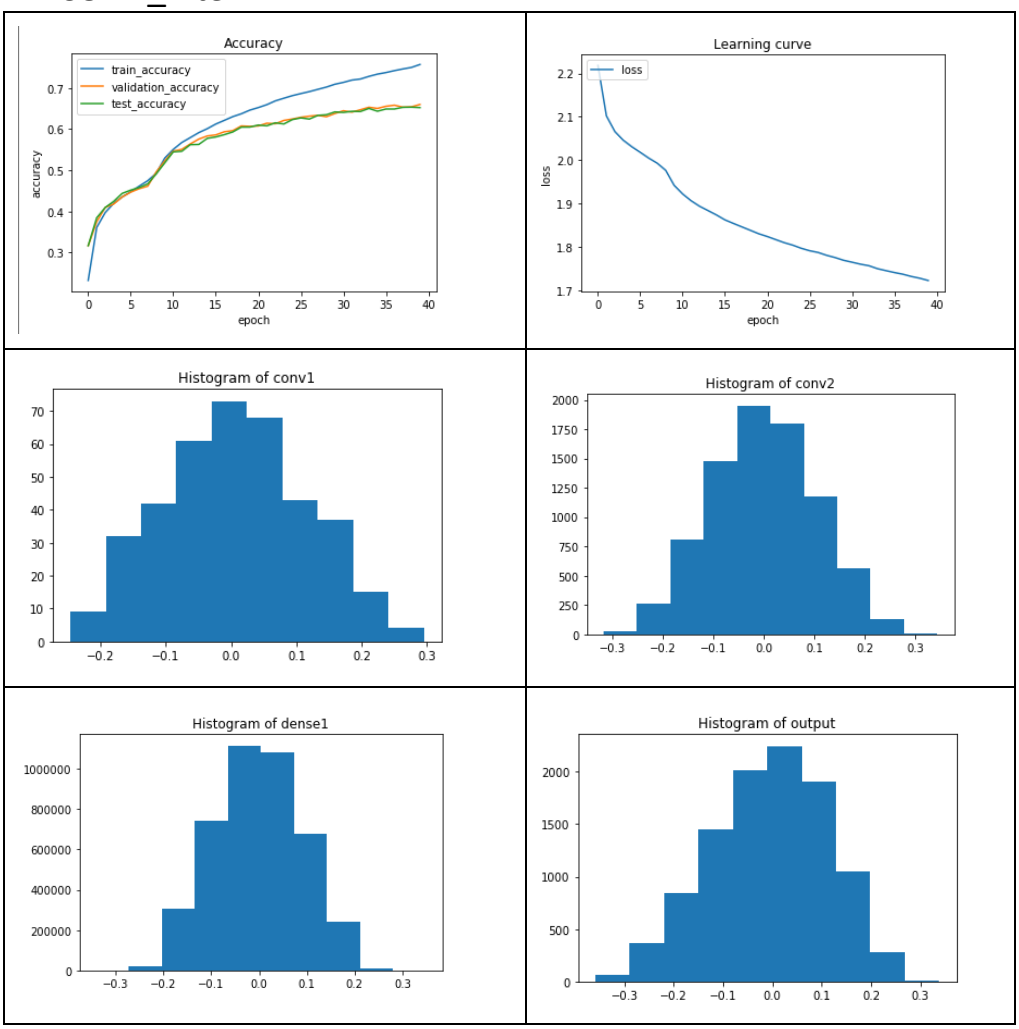


因為考慮到我筆電的性能，所以我只有加兩層捲積層和一層隱藏層，並且捲積層的 filter size 都是[3,3]，而隱藏層的 weight matrix 則是[4096,1024]，epoch 的次數則是訂為 40。可以看出 weight 的 histogram 在各層的結果主要是呈現一個 mean=0 的 normal distribution。

其中在 accuracy 的圖裡可以看到，training accuracy 到最後一值還有上升的趨勢，但是 test accuracy 和 validation accuracy 都已經不變了，所以猜測 model 是有點 overfitting 的情況產生。

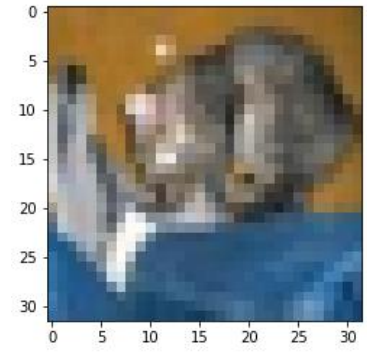
另外也因為這組 CIFAR10 的圖片相較於 MNIST 更為複雜，所以整體的 accuracy 相對來講都並不高，可能需要更複雜的 model 才可以完整表達。

- CONV_filter 2X2



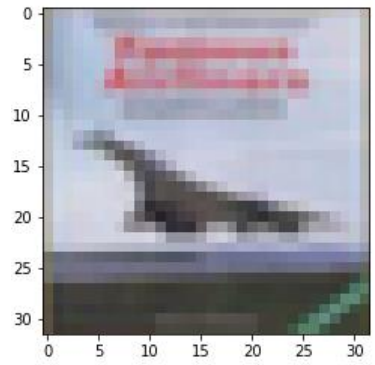
2.2

Label=cat ,predict=dog



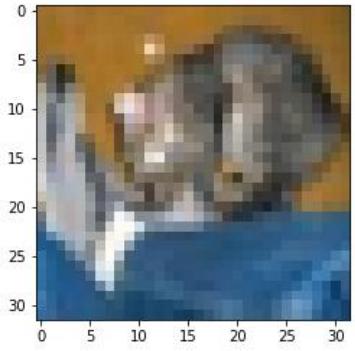
上圖正確來說應該是貓，但 model 判斷成狗，但因為我們自己有時候也會在看不清楚的情況下把貓看成是狗，所以 model 會判斷錯也是情有可原的。

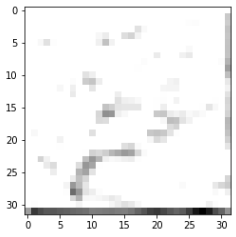
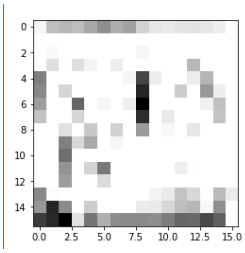
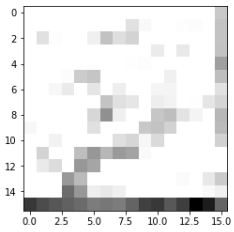
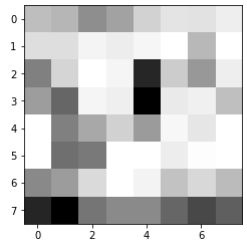
Label=airplane, predict=airplane

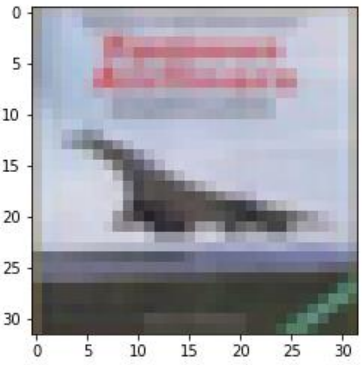


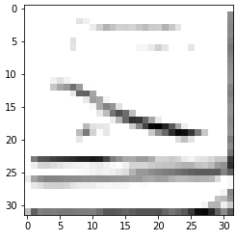
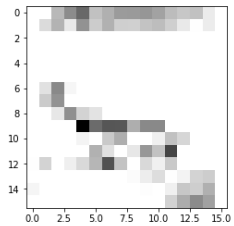
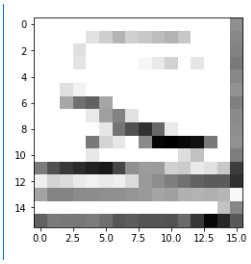
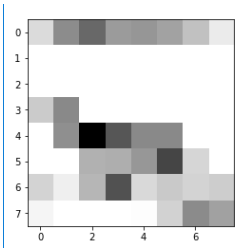
上圖實際上是架飛機，而 model 也可正確判斷，猜想主要原因可能是因為在資料庫中，並沒有其他和飛機特徵很接近的物件，因此就比較不容易搞混，不像上一張搞混的貓和狗，畢竟貓和狗的特徵相對起來非常接近。

2.3



First CONV	Second CONV
	
First maxpooling	**Second maxpooling**
	

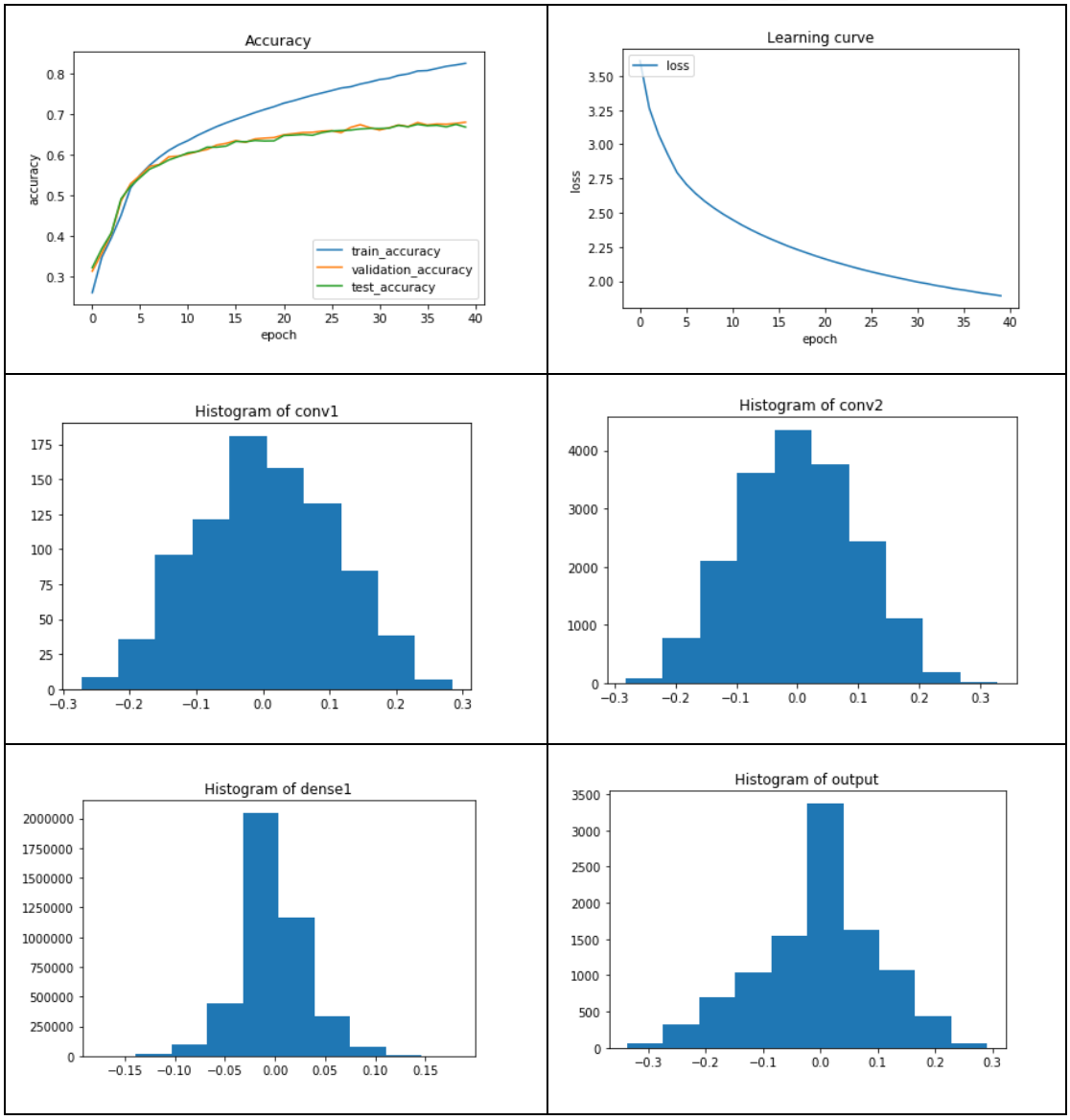


First CONV	Second CONV
	
First maxpooling	**Second maxpooling**
	

可以看到在經過捲積層後，基本上就只剩下特徵輪廓了。

2.4

- Regularization, with alpha=0.0001 and CONV_filter 3X3



經過了 L2 norm 的 regularization，可以發現 weight 的 histogram 當中的 0 的數目變多了。

2.5

- Preprocessing

首先要先對於 label 的部分做一個 onehot_encoding 這樣才能放到最後對應 output layer 的 neuron。

接著要對於 feature 作數值的 normalization，因為每張圖片中的每一個 pixel 是由 RGB 三維度所組成的，並且 value 在 0~255 之間，所以我們把這三個維度都除以 255，使得最後的 value 可以在 0~1 之間，這樣可以使 training 最後收斂的速度加快。