Deep Learning HW2

309512010 陳紀愷

1. Using Convolutional Neural Network for Image Recognition

1.Please implement a CNN for image recognition using the MNIST dataset

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 32)        160
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 32)        4128
_________________________________________________________________
flatten_1 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               1605888
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
=================================================================
Total params: 1,612,746
Trainable params: 1,612,746
Non-trainable params: 0
_________________________________________________________________
```

為了辨認 MNIST dataset，我總共使用了兩層的 CNN 和 2 層的 dense 層，其中 CNN 的厚度都是取 32，並在兩層 CNN 中加入一層的 Max pooling 來讓降低特徵數，做完 CNN 後再把它攤平並放入全連接層最後 Output 成 10 個數值，分別代表預測為 0~9 的權重。

其中我使用了 Adam 作為 optimizer，categorical crossentropy 作為 loss，做 50 個 Epoch 並分析改變 kernel size 和 strides 帶來的影響。

Strides = (1,1) 時改變 Kernel size 的影響

Kernel size	Train loss	Train acc	Val acc	Test acc
(3,3)	0.0123	0.9990	0.9895	0.9858
(5,5)	0.0342	0.9982	0.9847	0.9858
(7,7)	0.0375	0.9984	0.9851	0.9850

(3,3)時表現最好，Test accuracy 有 98.5%

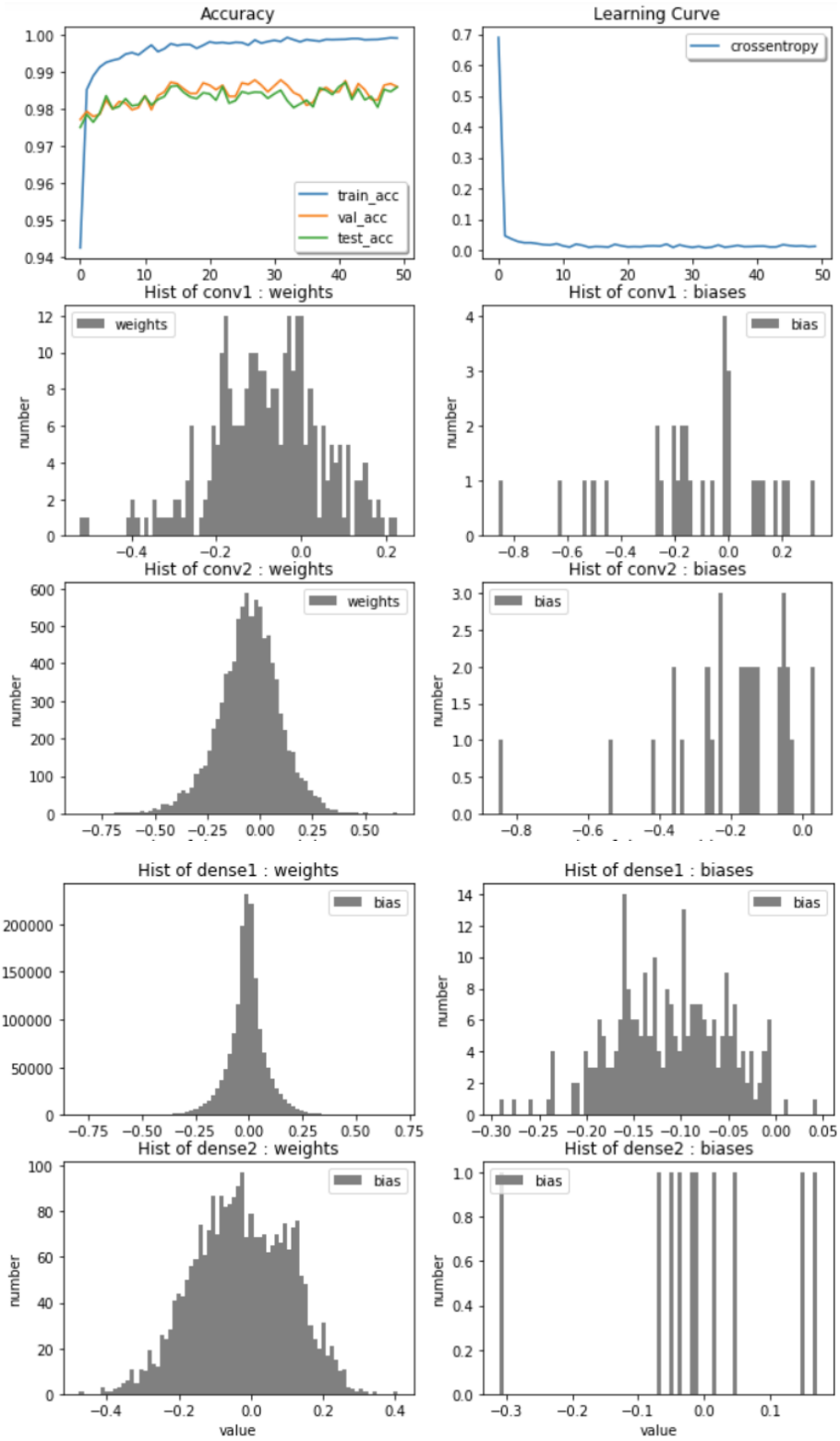
從上表可以發現在 Strides 不變的時候，在這個手寫數字辨識裡面，當 Kernel size 上升的時候會使 **Train loss 變大**，也會使訓練的準確率稍為的降低，但是對測試集的影響並不大，而且可能是因為任務比較簡單，所以其實比較起來整體的改變比較不明顯。

Kernel size=(5,5) 時改變 Strides 的影響

Strides	Train loss	Train acc	Val acc	Test acc
(1,1)	0.0289	0.9984	0.9859	0.9842
(2,2)	0.0281	0.9971	0.9853	0.9840
(3,3)	0.0216	0.9960	0.9813	0.9805

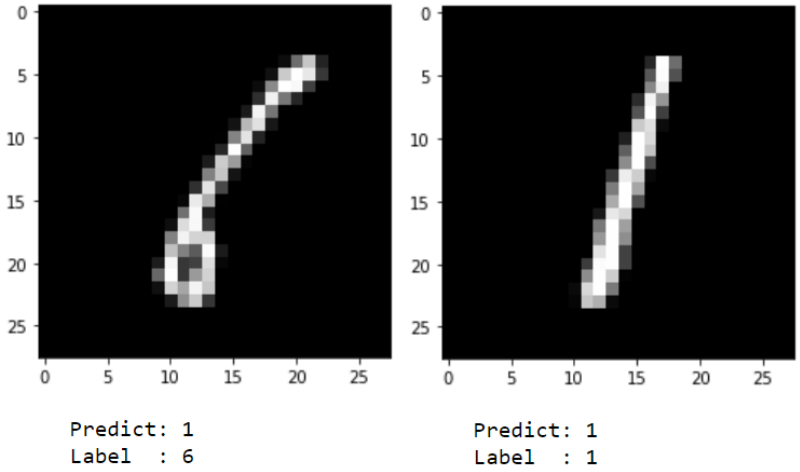
而當 Kernel size 固定而 Strides 改變時，可以發現全部的 **accuracy** 都會隨著 Strides 的上升而**變低**，同時訓練的 loss 也隨之上升，可能是因為 Strides 比較小的時候會使兩個 Kernel 的重疊部分比較多，因此取特徵的時候會比較有效果。

kernel size= (3,3) ，strides=(1,1)

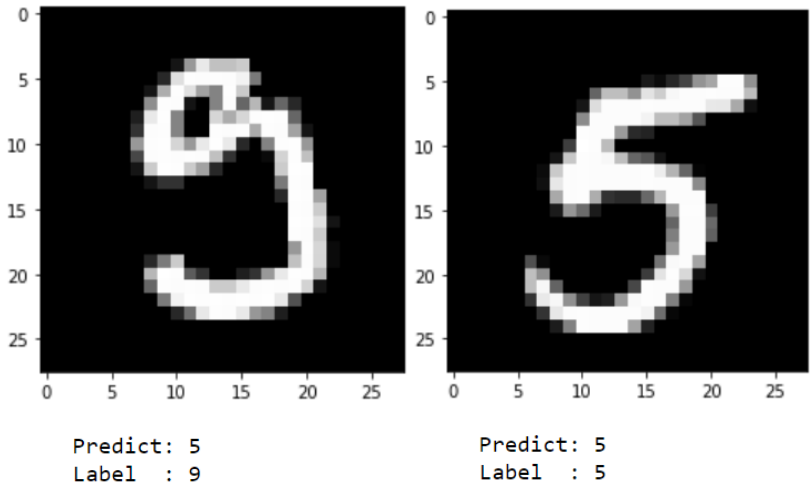


1-2

Show some examples of correctly classified and miss-classified images and discuss your results.



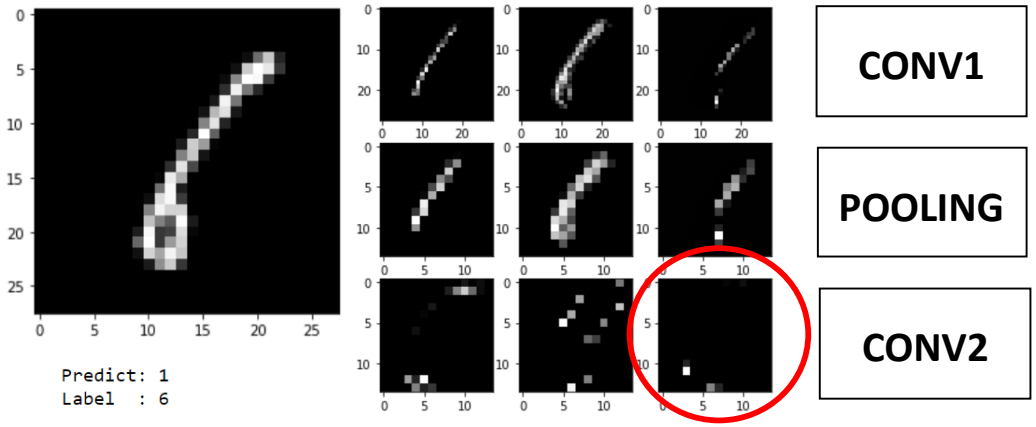
上面兩張圖分別是對數字 6 和數字 1 去做預測，可以發現右邊的圖成功預測了 1，但在同時左邊的 6 卻被誤認成 1，原因可能是因為左邊的 6 下方的圓圈比較不明顯，因此很容易和 1 搞混。

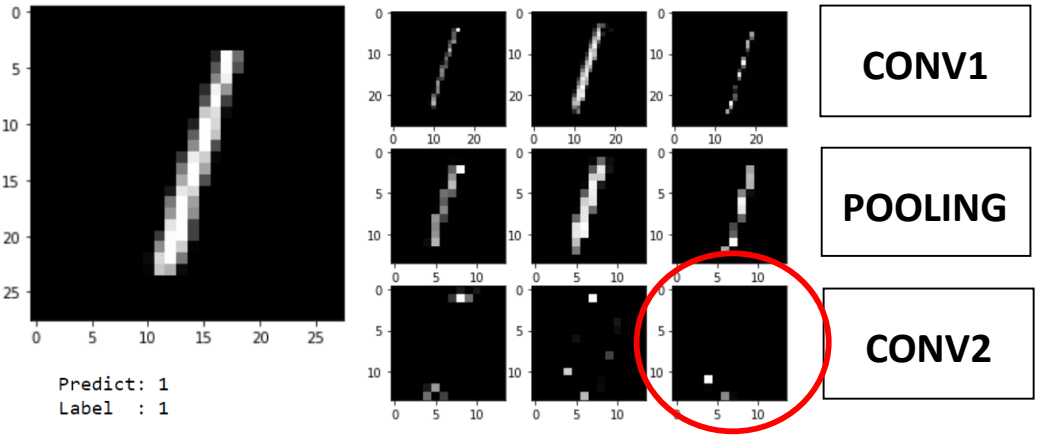


上面兩張圖分別是對數字 9 和數字 5 去做預測，可以發現右邊的圖成功預測了 5，但在同時左邊的 9 卻被誤認成 5，原因可能是因為左邊的 9 其實整體和 5 非常相似，尤其是下方凹起來的地方，因此在預測上比較有困難。

1-3

Following1-2, observe the feature maps from different convolutional layers and describe how a feature map changes with increasing depth.

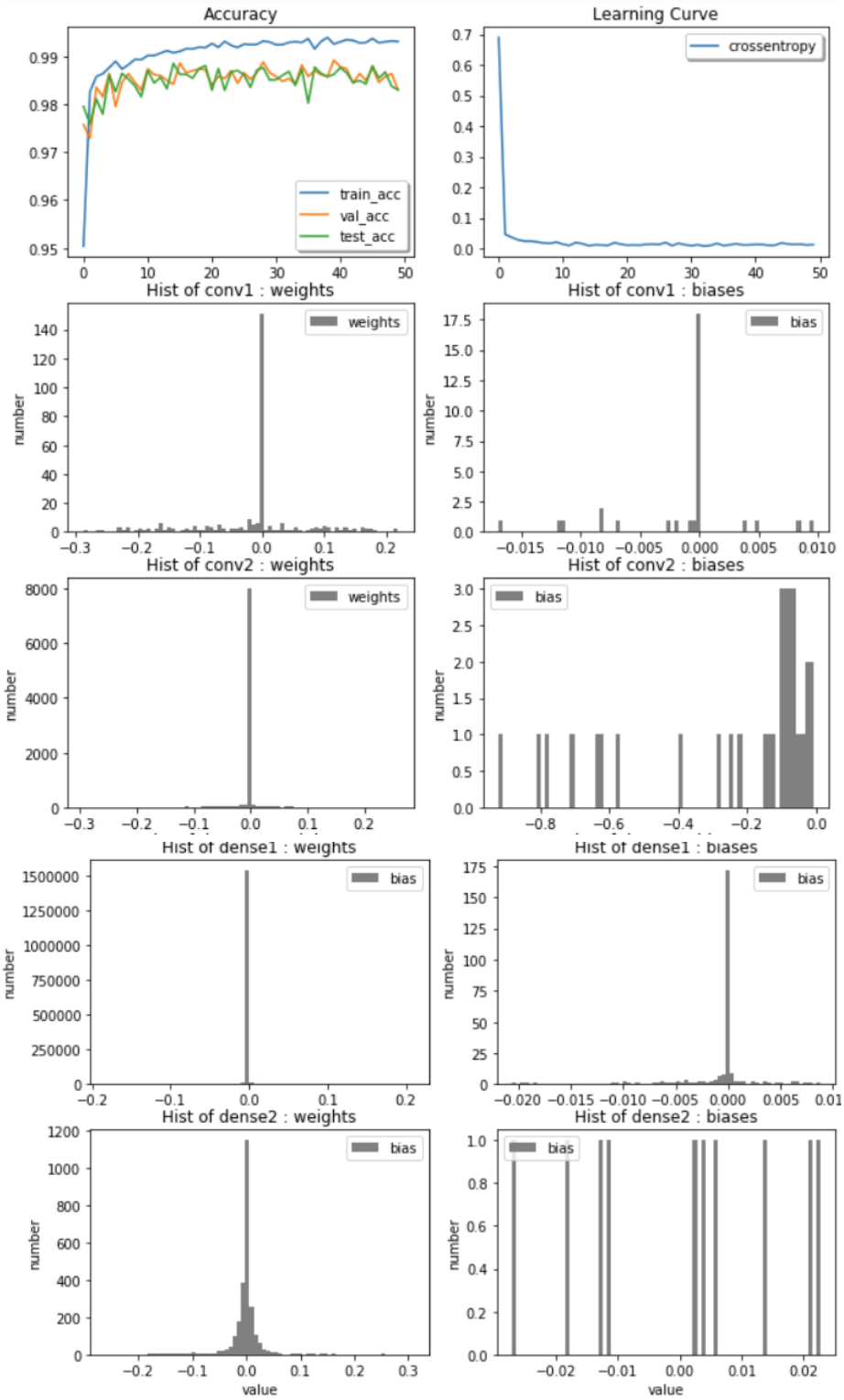




接下來對 1-2 有提到的 1 和 6 的辨識錯誤去做進一步的分析，可以發現在第一層的 CONVOLUTION 時兩張圖片還有一點點的相異，但到第二層的 CONVOLUTION 時，尤其是最左和最右邊兩張圖，他們的特徵點幾乎已經完全變成一樣了，因此才會導致辨識上出現問題。

1-4

Following 1-1, please add L2 regularization to the CNN implemented in 1-1 and discuss its effect.



L2 regularization	Train loss	Train acc	Val acc	Test acc
NO	0.0123	0.9990	0.9895	0.9858
YES	0.0635	0.9930	0.9830	0.9830

如同預期一樣，加入 L2 以後可以發現 Train loss 相較於原本的大了不小，因此導致 training 的準確率下降，因為 L2 是會把 w 和 b 盡可能的拉回 0，來避免 overfitting 的發生，w 和 b 的改變也可以從上面的圖看出來，比起原本的有點常態分佈相比起來更加的集中在原點附近。

2. Preprocessing Before Using Convolutional Neural Network for Image Recognition

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_23 (Conv2D)           (None, 32, 32, 32)        320
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 32, 32, 32)        9248
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 16, 16, 64)        36928
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 8, 8, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 8, 8, 128)         73856
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 8, 8, 128)         147584
_________________________________________________________________
batch_normalization_3 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 4, 4, 128)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten_11 (Flatten)         (None, 2048)              0
_________________________________________________________________
dense_21 (Dense)             (None, 10)                20490
=================================================================
Total params: 307,818
Trainable params: 307,370
Non-trainable params: 448
_________________________________________________________________
```

這次我使用了 6 層的 CNN 網路層，最後攤平後再加上一層的 dense 層，最後經過在 soft max 層和 Arg max 後來判斷結果。為了防止 overfitting 和提升效率，我在每兩層的 CONVOLUTION 層後加入 dropout 和 max pooling 及 batch normalization，而每次 dropout 的數

量也越來越多。

其中優化我選擇使用了”Adam”而 loss function 則是使用”categorical crossentropy”，並且對他們做了 50 次的 epoch 並分析比較 kernel size 和 strides 帶來的影響。

Strides = (1,1) 時改變 Kernel size 的影響

Kernel size	Train loss	Train acc	Val acc	Test acc
(3,3)	0.1790	0.9360	0.8356	0.8324
(5,5)	0.0987	0.9649	0.8292	0.8228
(7,7)	0.0932	0.9678	0.7876	0.7836

(3,3)時表現最好，Test accuracy 有 83%

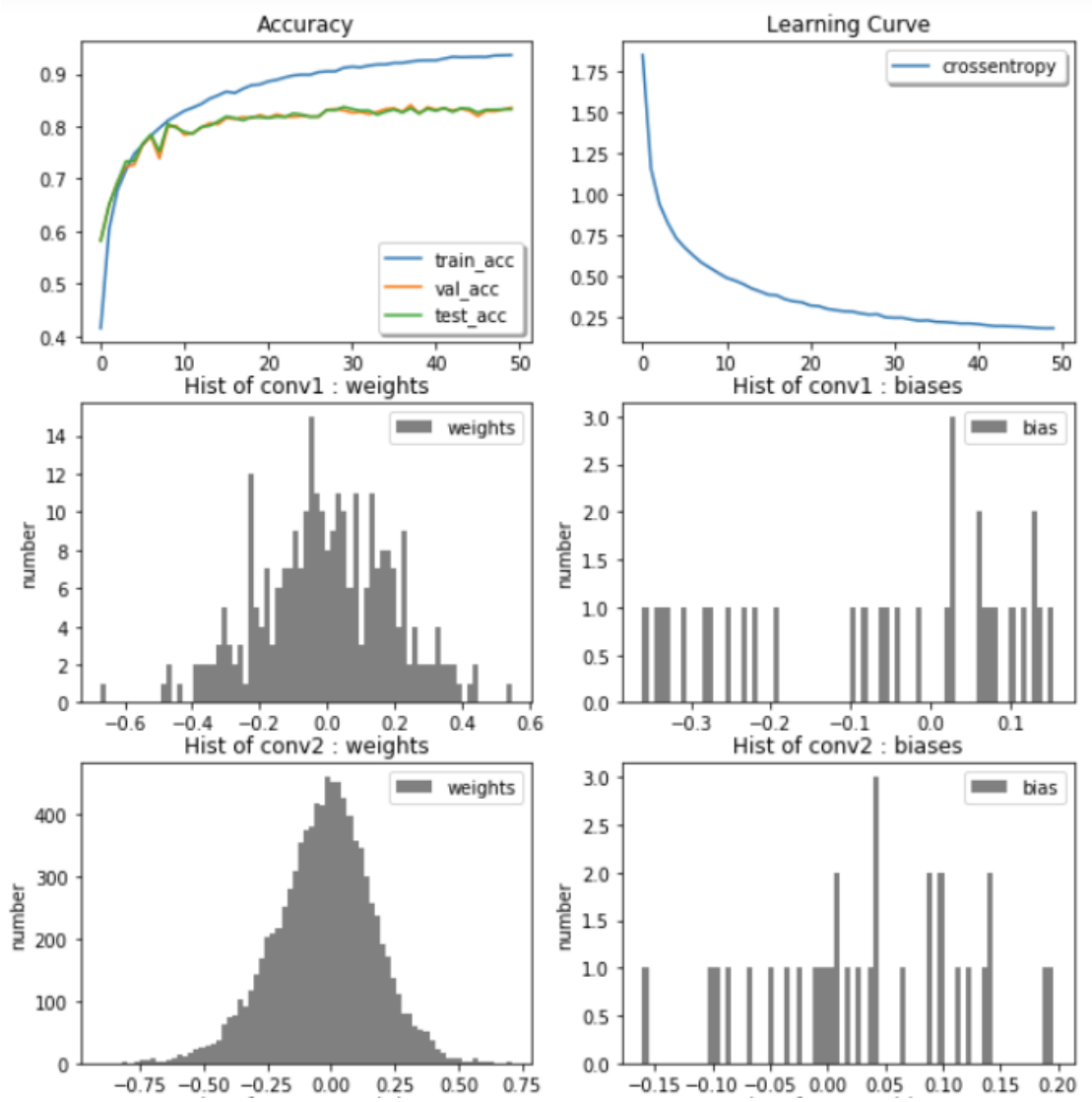
和 1-1 一樣，從上表可以發現在 Strides 不變的時候，當 Kernel size 上升的時候會使 Train loss 變大，也會使訓練的準確率稍為的降低。

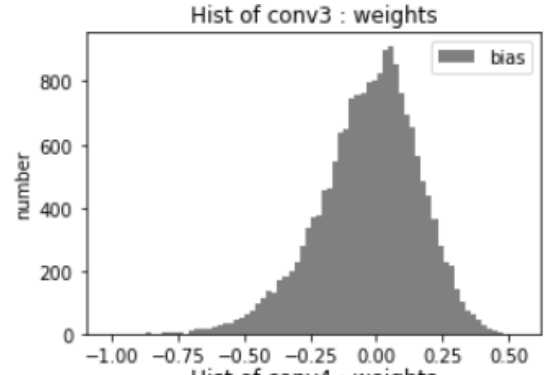
Kernel size=(5,5) 時改變 Strides 的影響

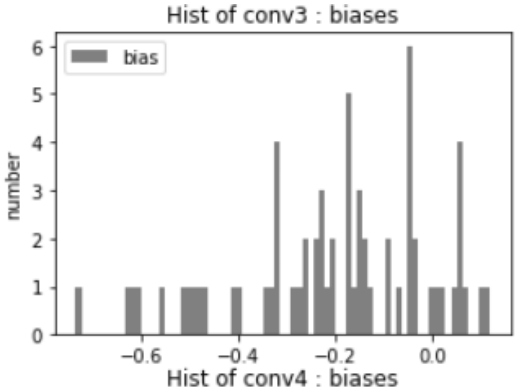
Strides	Train loss	Train acc	Val acc	Test acc
(1,1)	0.0940	0.9672	0.8240	0.8182
(2,2)	0.3322	0.8852	0.7256	0.7257

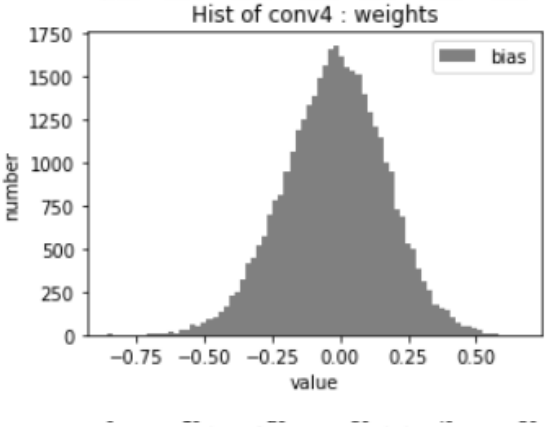
而因為我一共有六層的 conv 層，但是每筆資料都只有 32*32 的大小，所以如果對每一層的 Strides 都變大的話，後面的層數資料會沒辦法負荷，所以我只對前面兩層的 Strides 做改變，後面保持原樣，而經過實驗也可以發現增加 Strides 的大小會對效能有明顯的降低。

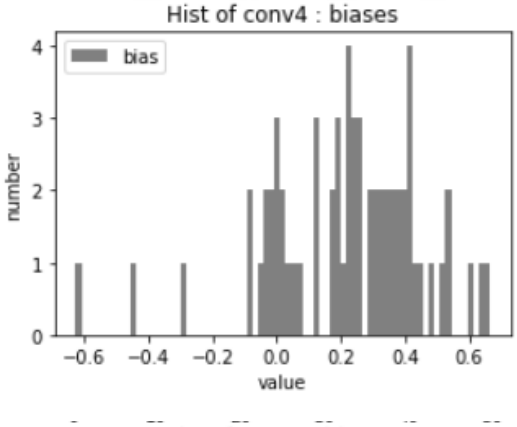
kernel size= (3,3) ，strides=(1,1)

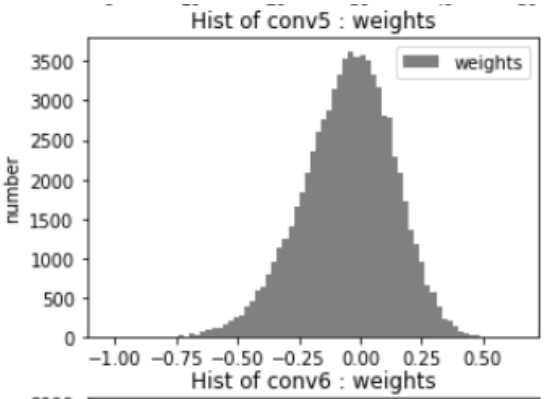


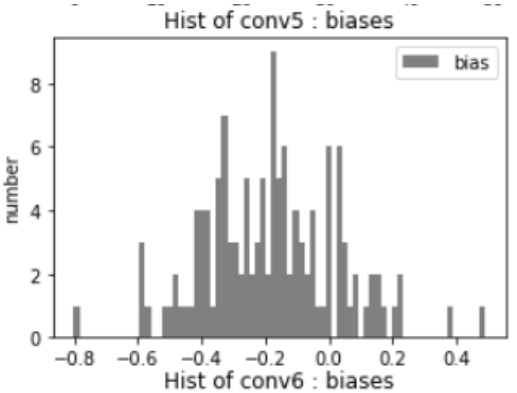

Hist of conv3 : weights
bias
number
800
600
400
200
0
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50

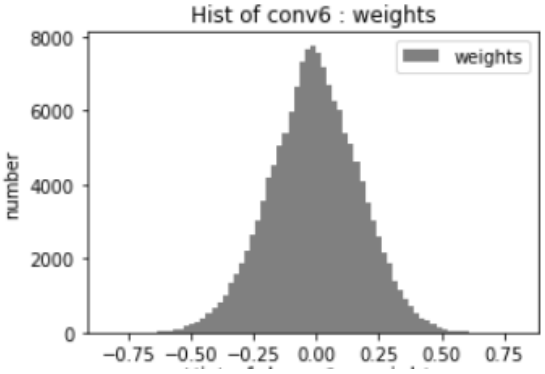

Hist of conv3 : biases
bias
number
6
5
4
3
2
1
0
−0.6 −0.4 −0.2 0.0

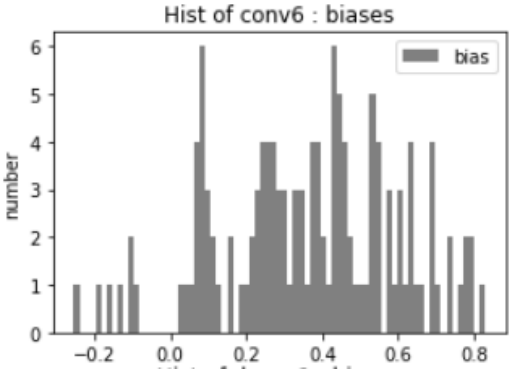

Hist of conv4 : weights
bias
number
1750
1500
1250
1000
750
500
250
0
−0.75 −0.50 −0.25 0.00 0.25 0.50
value

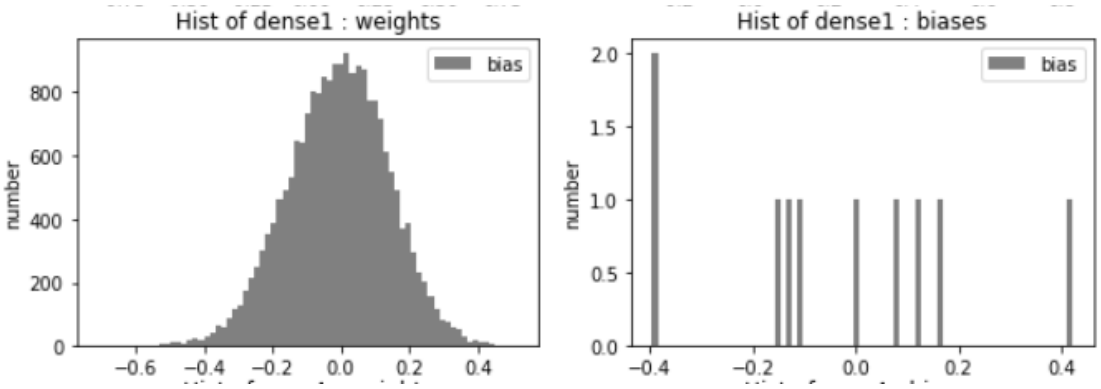

Hist of conv4 : biases
bias
number
4
3
2
1
0
−0.6 −0.4 −0.2 0.0 0.2 0.4 0.6
value


Hist of conv5 : weights
weights
number
3500
3000
2500
2000
1500
1000
500
0
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50


Hist of conv5 : biases
bias
number
8
6
4
2
0
−0.8 −0.6 −0.4 −0.2 0.0 0.2 0.4

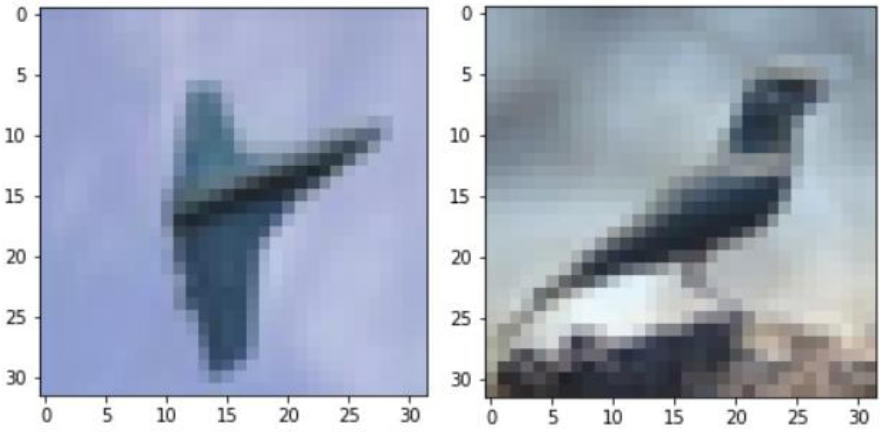

Hist of conv6 : weights
weights
number
8000
6000
4000
2000
0
−0.75 −0.50 −0.25 0.00 0.25 0.50 0.75

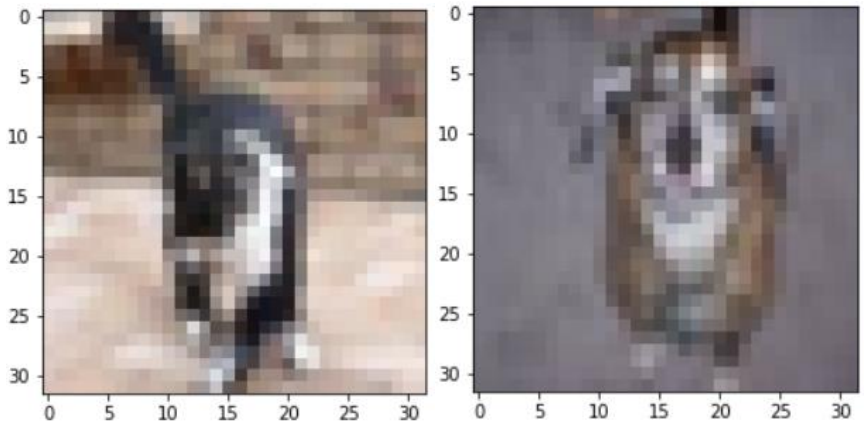

Hist of conv6 : biases
bias
number
6
5
4
3
2
1
0
−0.2 0.0 0.2 0.4 0.6 0.8



2-2

Show some examples of correctly classified and miss-classified images and discuss your results.



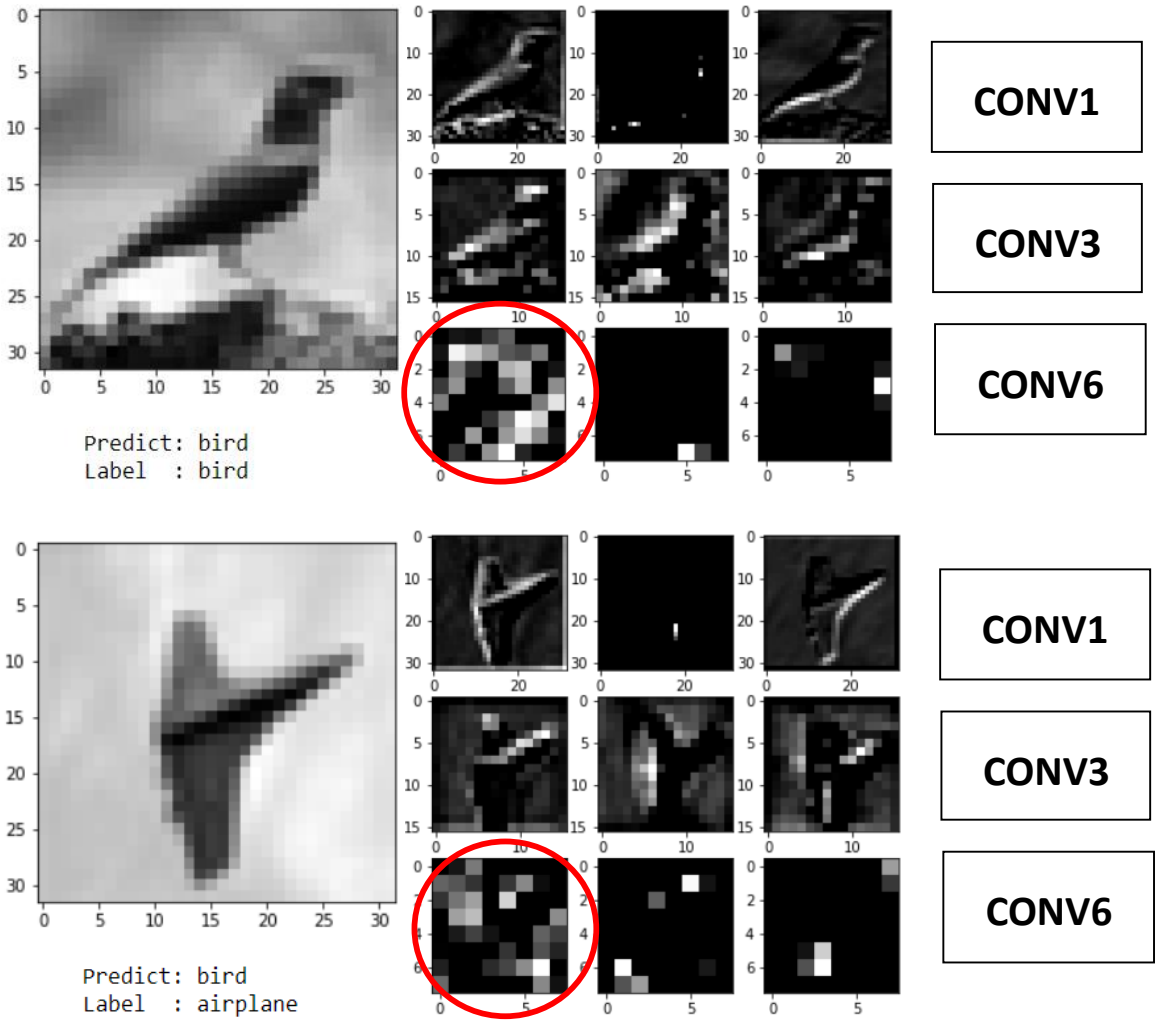


分析辨識錯誤的圖片後我發現貓和狗、鳥和飛機、車子和卡車，這幾種最容易被互相搞混，理由不外乎是因為他們其實兩兩本身就比

較接近，像是貓狗都是陸上四隻腳的動物，而鳥和飛機的身軀都比較長且會有兩個突出的翅膀，因此在辨識上也比較有難度。

2-3

Following2-2, observe the feature maps from different convolutional layers and describe how a feature map changes with increasing depth.



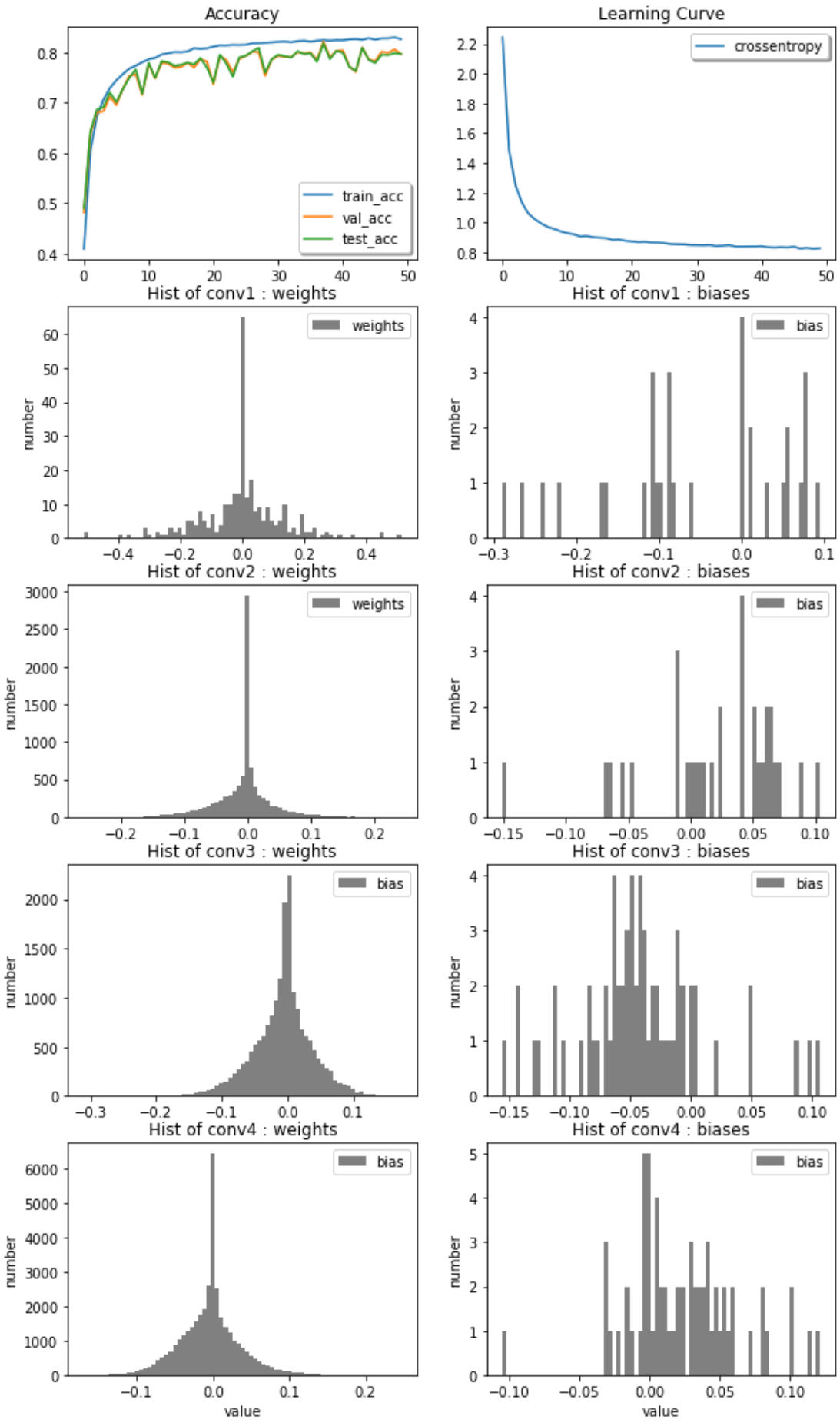
接下來對 2-2 有提到的飛機和鳥的辨識錯誤去做進一步的分析，這裡我選擇了第 1、3、6 層的 CONVOLUTION 層的結果去做比較，可以發現在第 1 層的結果的兩張圖片都大致還看的出來是甚麼東西，但是都很流線形，到第 3 層的時候就比較模糊了，而到最後第 6 層的

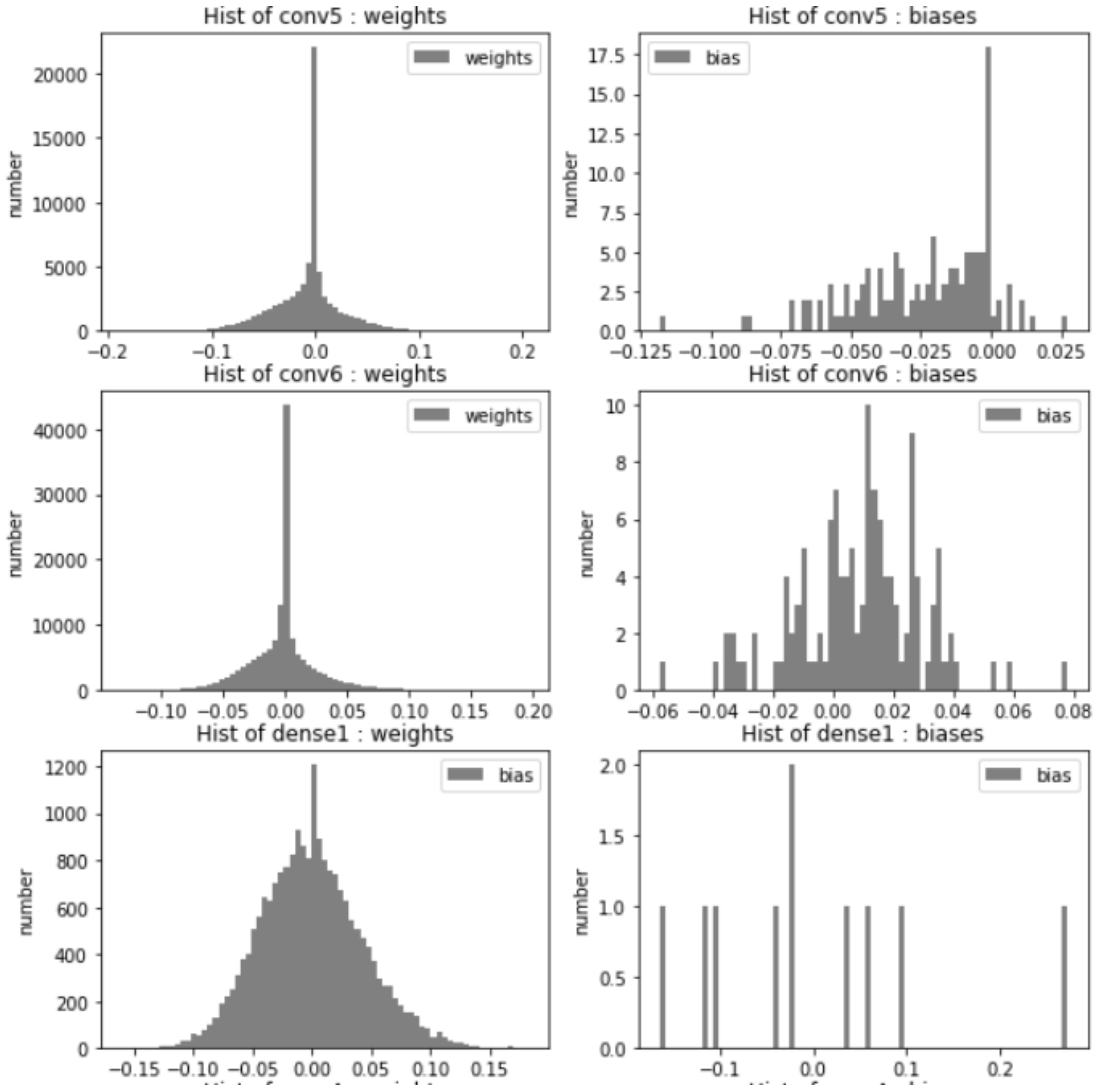
時候他們的特徵點變得有點相似，以左下的圖為例他們的點都分布在左上和右下，因此才會導致辨識上出現問題。

2-4

Following 1-1, please add L2 regularization to the CNN implemented in 1-1 and discuss its effect.

和 1-4 相同，加入 L2 以後可以發現 Train loss 相較於原本的大了不小，因此導致 training 的準確率下降， w 和 b 也比起原本的有點常態分佈相比起來更加的集中在原點附近，來避免 overfitting 的發生，但是在 Testing 和 Val Accuracy 的就會比原來還沒加入 L2 來的還要震盪。


Accuracy
train_acc
val_acc
test_acc
Learning Curve
crossentropy
Hist of conv1 : weights
weights
number
Hist of conv1 : biases
bias
number
Hist of conv2 : weights
weights
number
Hist of conv2 : biases
bias
number
Hist of conv3 : weights
bias
number
Hist of conv3 : biases
bias
number
Hist of conv4 : weights
bias
number
value
Hist of conv4 : biases
bias
number
value



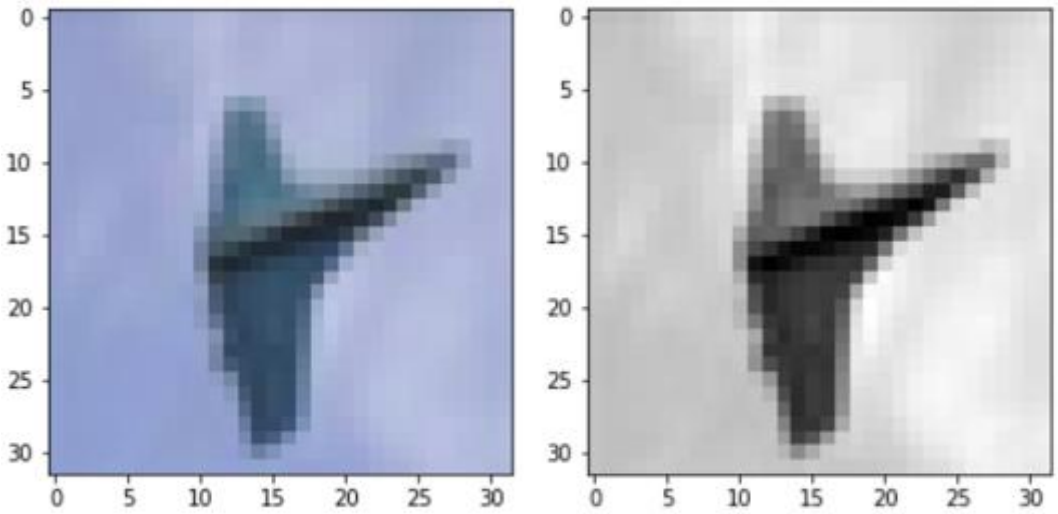
L2 regularization	Train loss	Train acc	Val acc	Test acc
NO	0.1790	0.9360	0.8356	0.8324
YES	0.8260	0.8273	0.7972	0.7975

2-5

為了統一 Cifar10 的資料，我對資料做了一些的預處理。

因為原本 Cifar10 的相片是(32,32,3)的 RGB 相片，分別代表每一個 pixel 的 RGB 值，所以我利用 RGB to gray 的公式，

Gray = R*0.299 + G*0.587 + B*0.114，利用矩陣運算先把他轉為灰階，把每筆資料的維度轉成(32,32,1)。



接著利用公式去對每筆資料去做標準化

$$z = \frac{x - \mu}{\sigma}$$

最後在對資料的標籤做 one hot encoding 把他們變成(1,10)的矩陣，分別對應他們是甚麼種類的 label。如此一來就可以開始做我們的訓練了。

參考資料：
https://www.cs.toronto.edu/~kriz/cifar.html
http://yhhuang1966.blogspot.com/2018/04/keras-cifar-10.html
http://atlaboratary.blogspot.com/2013/08/rgb-g-rey-l-gray-r0.html