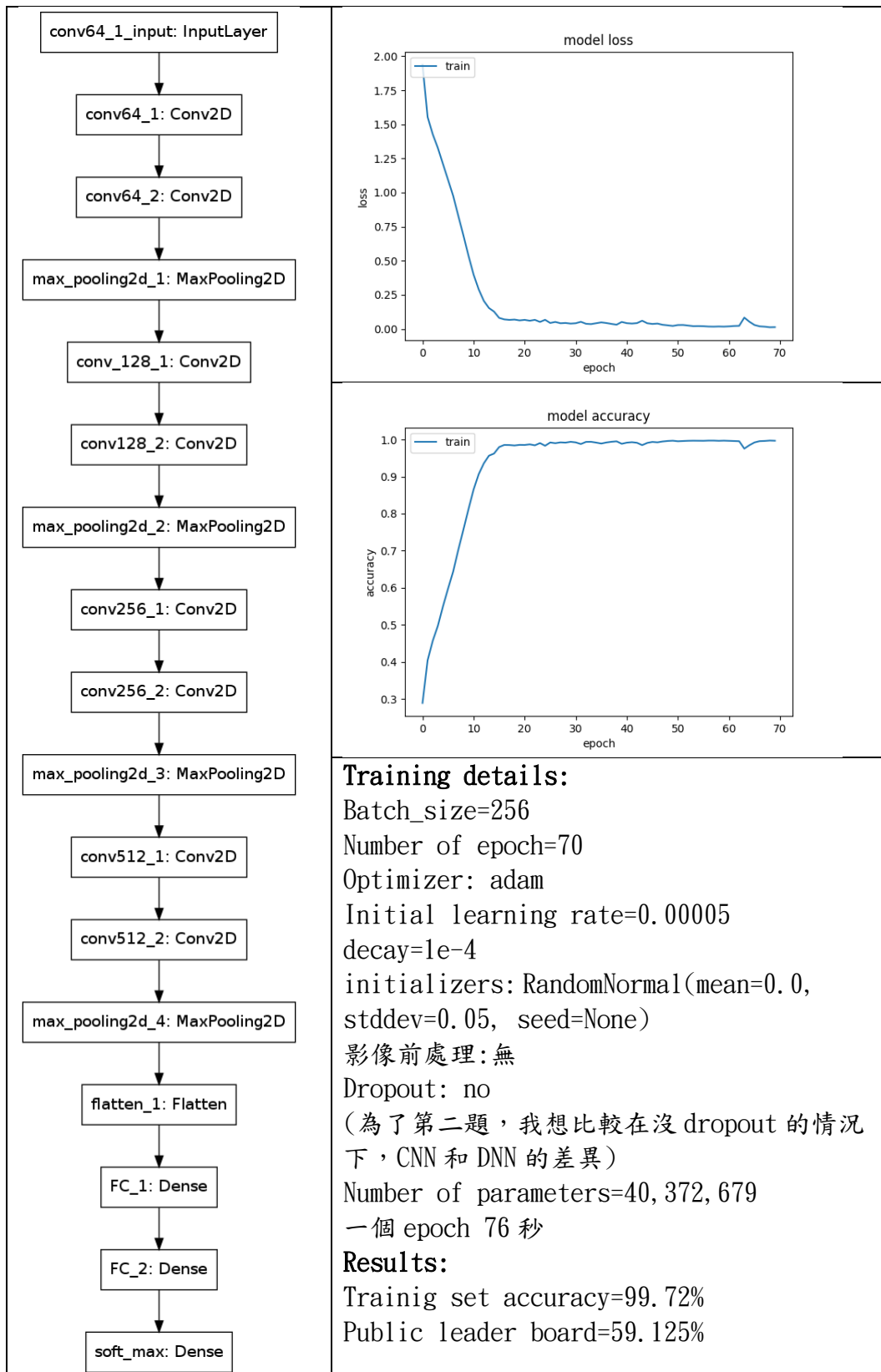


學號：B03901096 系級：電機三 姓名：周晁德

1. (1%) 請說明你實作的 CNN model，其模型架構、訓練過程和準確率為何？

答：



這個 network 是我參考 Vgg net 設計出來的，所有 hidden layer 都有搭配 relu，每個 convolution layer 的 filter 皆為  $3 \times 3$ ，有做 padding，input 進來之後，convolution layer 的 filter 數目，從最少 64 個以 2 的倍數遞增，到最多 512 個，每兩個 convolution layer 就有一個  $2 \times 2$  的 max pooling layer，convolution layer 和 max pooling 結束後是三層 fully connected layer，前面兩層每層都有 4096 個 units，最後一層為 soft max。

我嘗試過以相同的 max pooling layer 數，在每個 max pooling 中間增加 convolution layer 的數目，但反而 overfitting 得更嚴重，也嘗試改變 activation function，例如使用 elu，雖然 training 時，收斂更快，但結果並沒有使用 relu 來得好，也使用過 batch normalization 但結果也不好，因此這是我設計出來最好的 network 架構。

這個 network 若每一層都搭配 dropout=0.5，仍然可以在 training set 上達到 99.~% 的 accuracy，在 public leader board 也可以達到 65.~% (超過 strong baseline)，但在 train dropout=0.5 時，如果直接 random initialize network 中所有的 weight，這麼做 dropout=0.5 train 不起來，必須先 train 沒有 dropout 的 network，再用這些 train 好的參數來 initialize network，dropout=0.5 才 train 得起來。

此外若使用 ImageDataGenerator，不用使用 dropout 這個 network 在 training set 上雖然只有 96.~% 的 accuracy，但在 public leader board 上可以達到 69.~%。

```
datagen = ImageDataGenerator(
```

```
    featurewise_center=False,
```

```
    featurewise_std_normalization=False,
```

```
    rotation_range=20,
```

```
    shear_range=0.2,
```

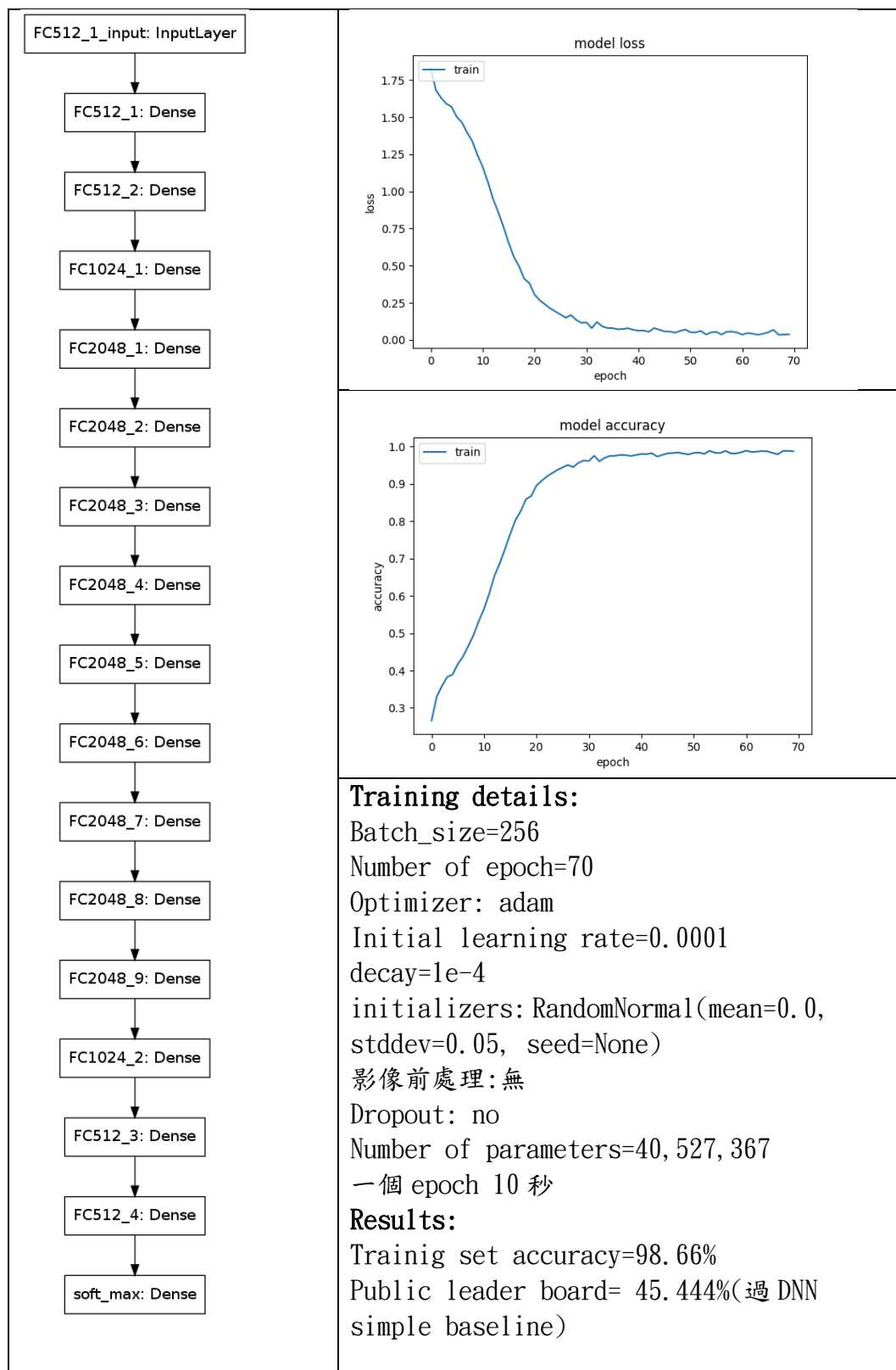
```
    width_shift_range=0.2,
```

```
    height_shift_range=0.2,
```

```
    horizontal_flip=True)
```

2. (1%) 承上題，請用與上述 CNN 接近的參數量，實做簡單的 DNN model。其模型架構、訓練過程和準確率為何？試與上題結果做比較，並說明你觀察到了什麼？

答：

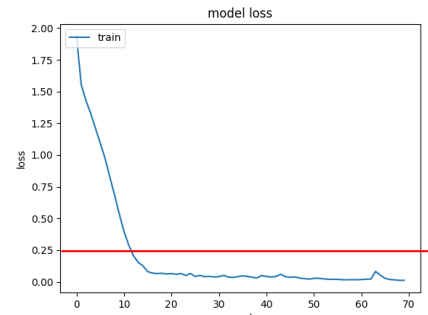
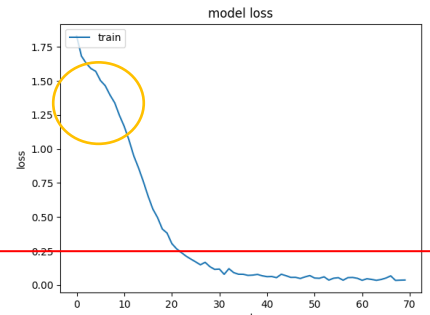


這個 DNN network 是我自己想出來的，每層 hidden layer 都搭配 relu，最後一層為 soft max，因為一般 DNN 的結果會比 CNN 來得差，加上要使 problem1 和 problem2 的參數數量很接近，所以我調了很久，想讓參數接近又希望 DNN 的參數稍微多一點點，並且想達到 network 的對稱性，前面的 layer 由兩層 units=512 和一層 units=1024 所組成，中間有連續 9 層 units=2048 的 layer，會在中間選擇使用 units=2048，是因為如果使用其它 units 較少的 layer，例如:units=512 和 units=1024，network 會太深，擔心 train 不起來，經過 units=2048 的 layers 之後，是一層 units=1024 和兩層 units=512 的 layer，會在 network 中加入 units=512 的 layers 純粹是為了要讓 DNN 的參數和 CNN 接近。

### CNN v. s. DNN

表(一)	Time per epoch	Training acc	Kaggle acc	Initial lr
CNN	76s	99.72%	59.125%	0.00005
DNN	10s	98.66%	45.444%	0.0001

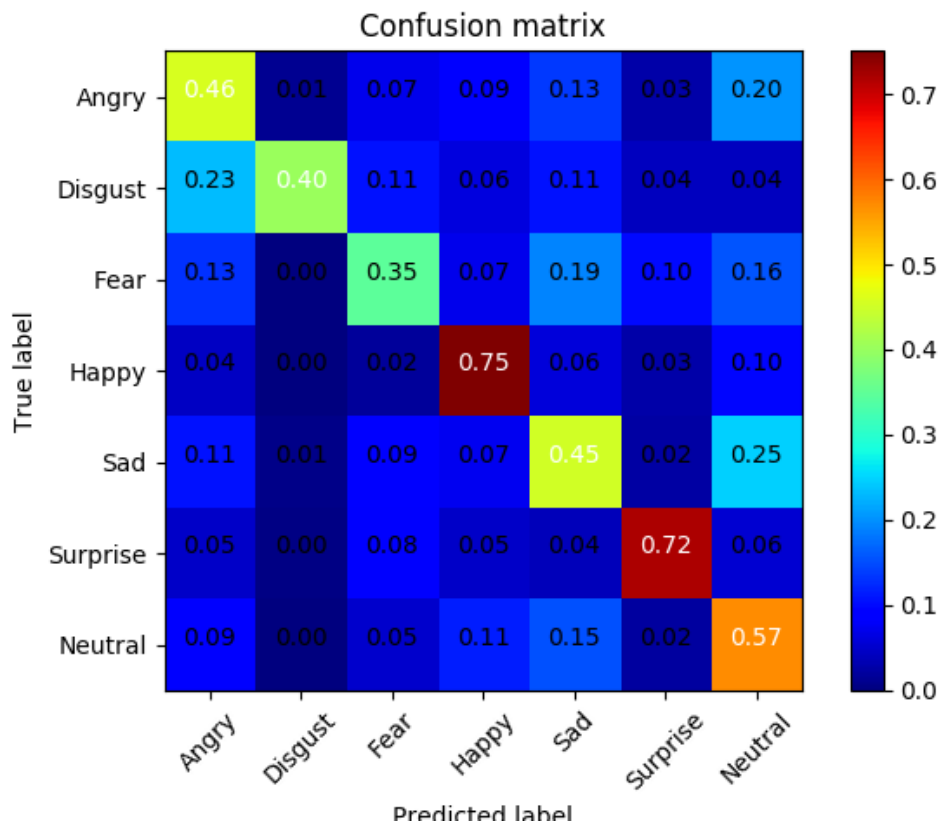
  

表(二)	CNN	DNN
Loss		

由上表(一)可知，CNN 不論是在 training set 或者是 public leader 的表現都比 DNN 來得好(我試過把 DNN training 的 epoch 數增加到 100 但 loss 並沒有再繼續下降)，CNN 的缺點大概就是 training 要花較長的時間，每個 epoch 所需時間大約為 DNN 的 7 倍，此外 CNN 的 initial learning rate 比較難調，一開始我使用 initial lr=0.0001，但很快 loss 就爆掉了，因此才調為 0.00005，但由表(二)中 training loss 的變化可以發現，即使 DNN 的 initial learning rate 較 CNN 來的大，但 CNN loss 下降的速度較 DNN 來得快(同樣降到 loss=0.25，CNN 大約只花 10 個 epoch，DNN 卻花大約 20 個 epoch)且穩定(即使是剛開始 train DNN 在黃圈中的部分 loss 下降突然變慢)，此外當快要收斂的時候 DNN loss 的波動比較多，CNN 只有一個較明顯的波動，其他時候幾乎已經呈現一條直線。

3. (1%) 觀察答錯的圖片中，哪些 class 彼此間容易用混？[繪出 confusion matrix 分析]

答：



我用 problem1 的 network(同樣沒有做 dropout)，把 training set 中前 25000 筆 data 當作 training data，剩下的 3000 多筆當作 validation set，所得到的結果。

由圖中可以發現每種圖片被歸類為正確的機率都是最高的，但其中只有 Happy, Surprise 和 Neutral 被歸類正確的機率超過 5 成，其他種表情被歸類錯誤的前三種的可能情況，我整理如下表：

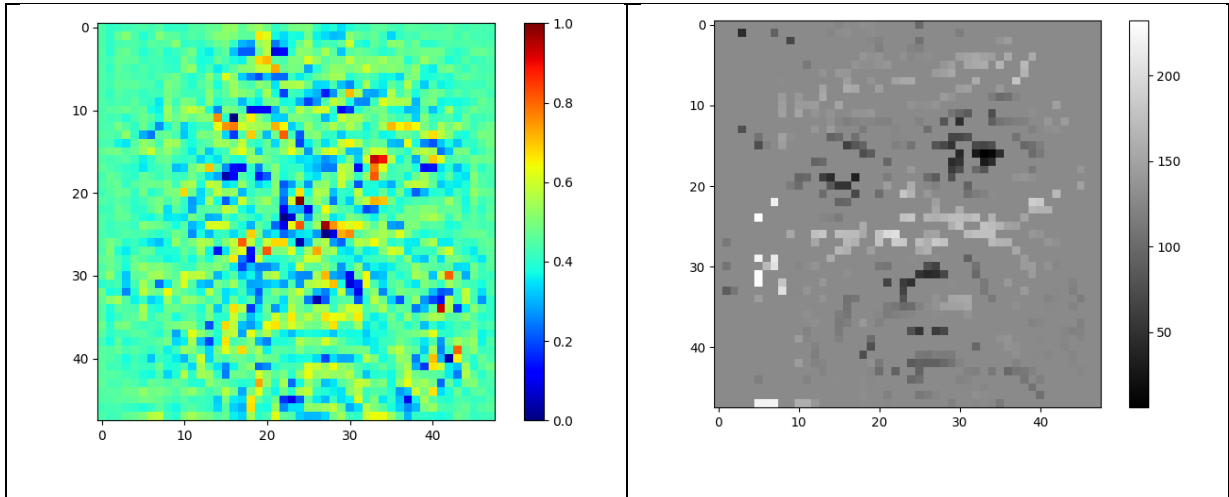
正確表情	最高錯誤	次高錯誤	第三高錯誤
Angry(0.46)	Neutral(0.2)	Sad(0.13)	Happy(0.09)
Disgust(0.40)	Angry(0.23)	Fear(0.11) Sad(0.11)	
Fear(0.35)	Sad(0.19)	Neutral(0.16)	Angry(0.13)
Sad(0.45)	Neutral(0.25)	Angry(0.11)	Fear(0.09)

由表中及圖中我們可以發現其實並沒有哪兩種表情特別相似，例如:Disgust 最容易被誤判成 Angry，但 Angry 最容易被誤判成 Neutral，而被誤判成 Disgust 的機率卻只有 0.01(是 Angry 被誤判的所有可能性中最低的)。此外圖中我們可以發現，很少表情符號會被誤判成 Disgust。

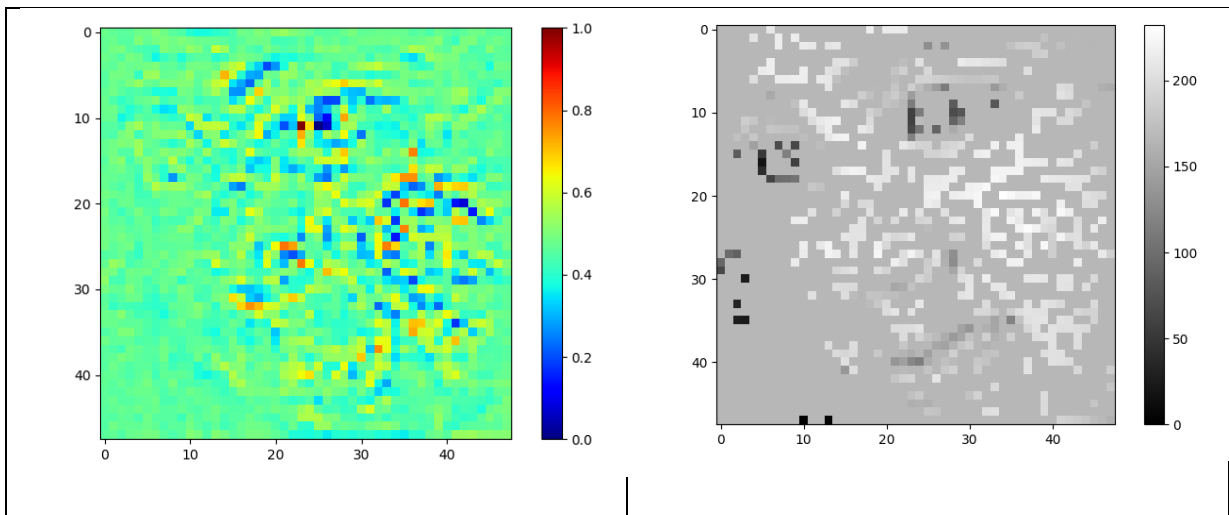
4. (1%) 從(1)(2)可以發現，使用 CNN 的確有些好處，試繪出其 saliency maps，觀察模型在做 classification 時，是 focus 在圖片的哪些部份？

答：

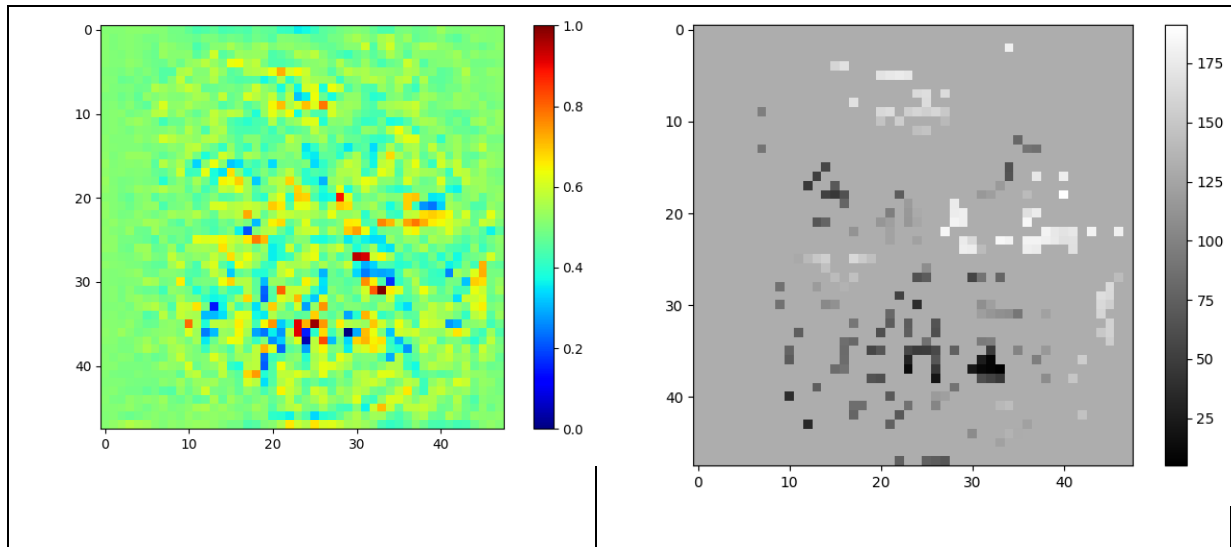
Class0: Angry



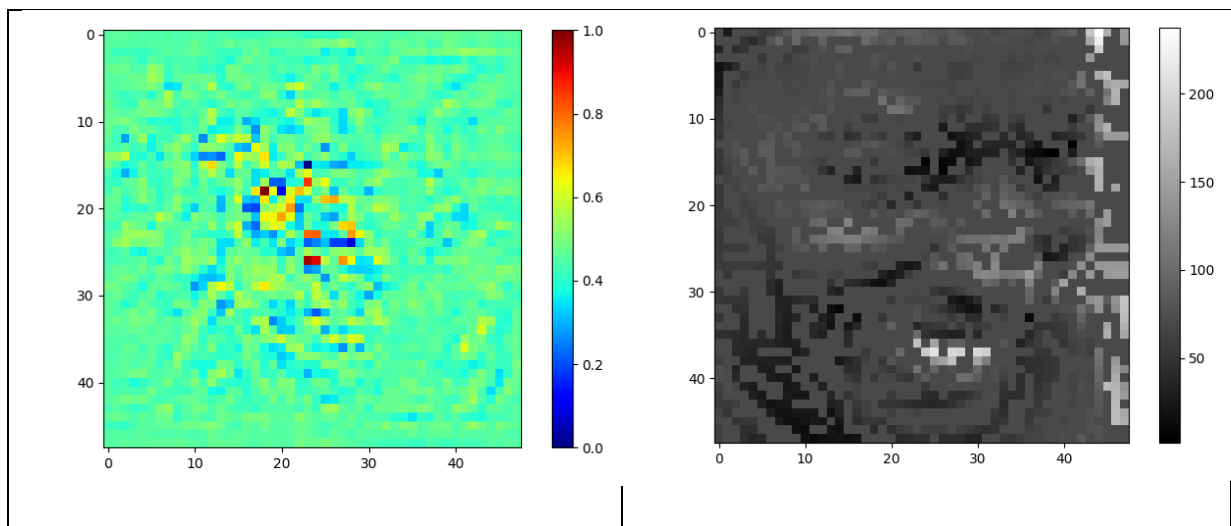
Class1: Disgust



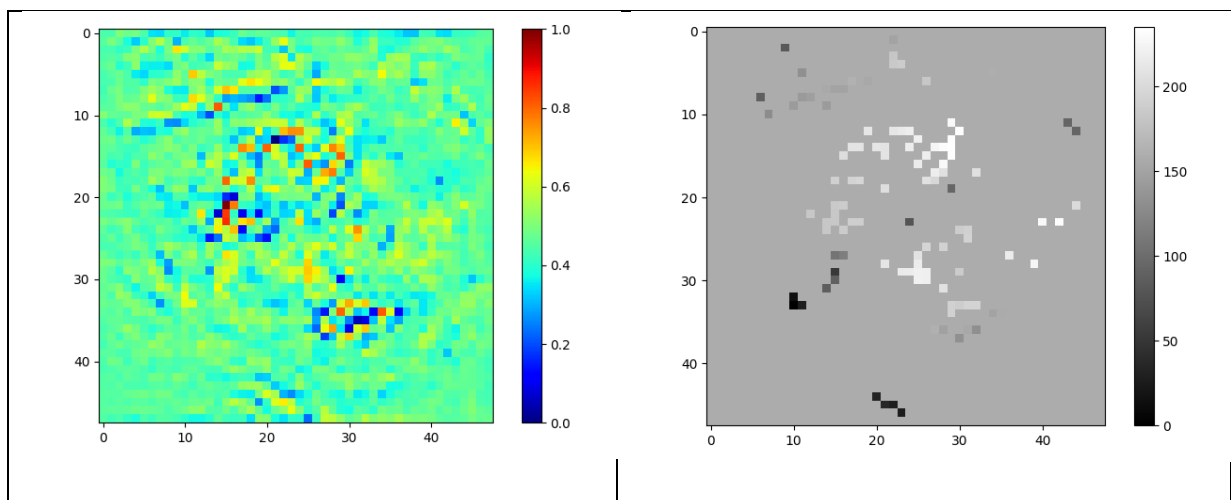
Class2: Fear



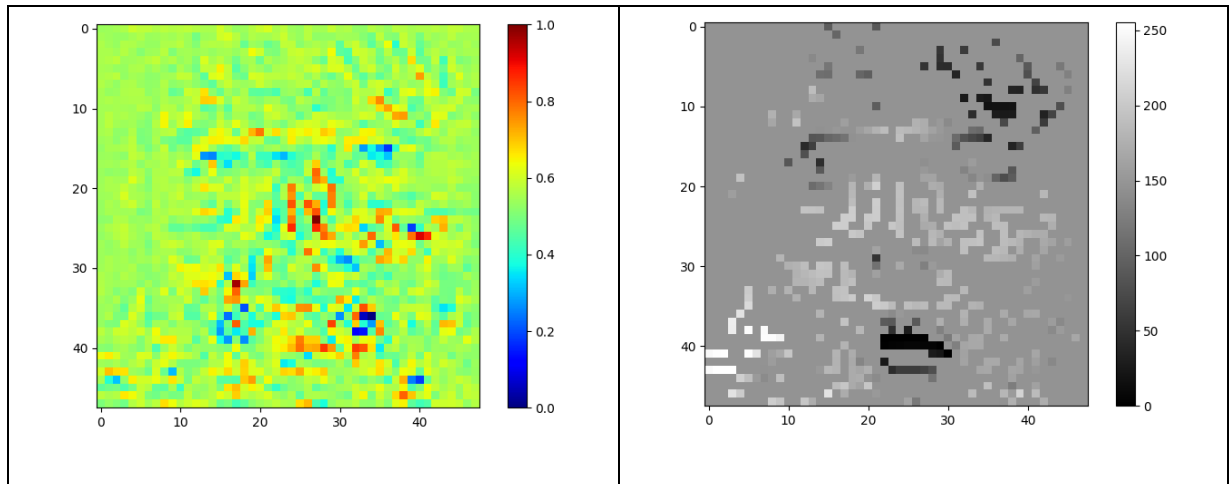
Class3 : Happy



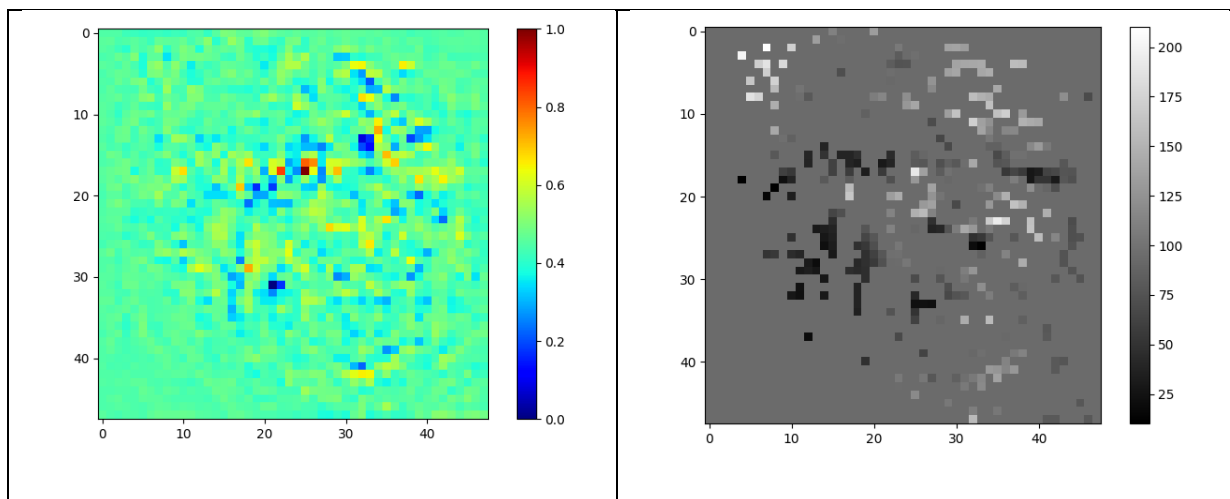
Class4: Sad



Class5: Surprise



Class6: Neutral



由圖中其實可以發現這些 saliency map 主要觀察集中在眼睛、鼻子和嘴巴附近的區域，尤其是眼睛和嘴巴的位置、形狀，例如: class3 中很明顯就是由嘴巴的形狀來判斷是否為 Happy。



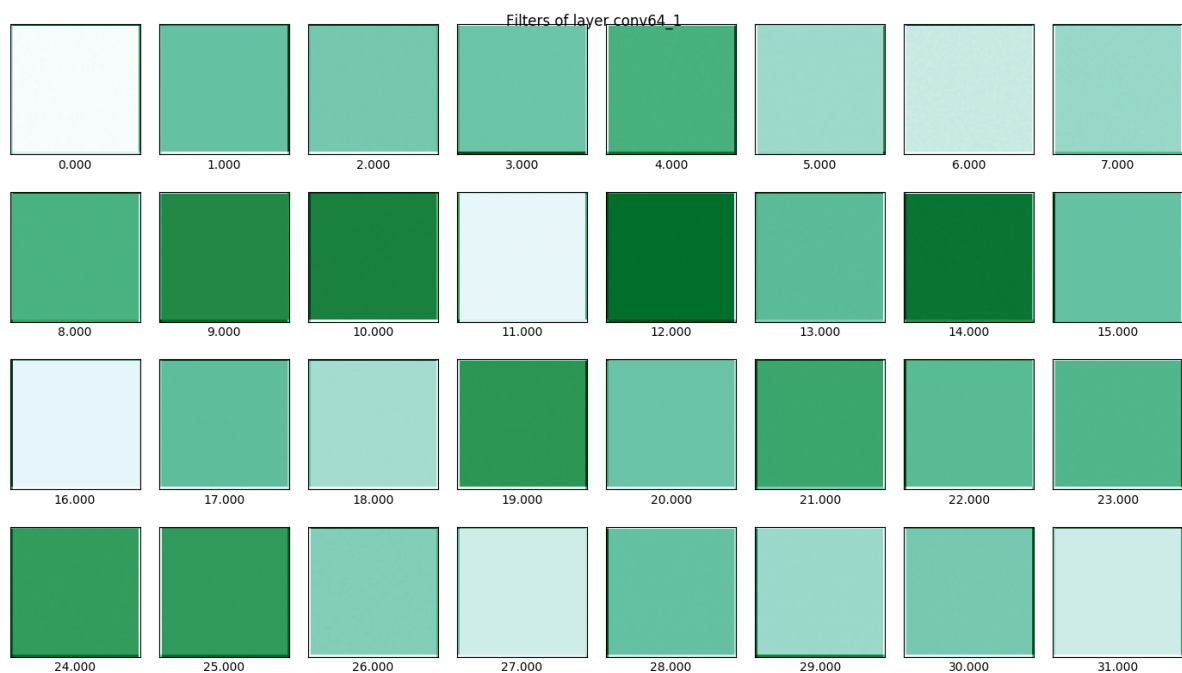
5. (1%) 承(1)(2)，利用上課所提到的 **gradient ascent** 方法，觀察特定層的 **filter** 最容易被哪種圖片 **activate**。

答：

我使用我結果最好的 model，又因為我 layer 有點多，filter 數也非常多，所以我只選取每個 layer 中前 32 個 filter(我試過在報告中放全部的 filter，但是因為這些照片真的很版佔面，加上如果這片太小張又看不清楚，因此只選取前 32 個 filter)，圖片下方的數字指的是 filter 的編號。

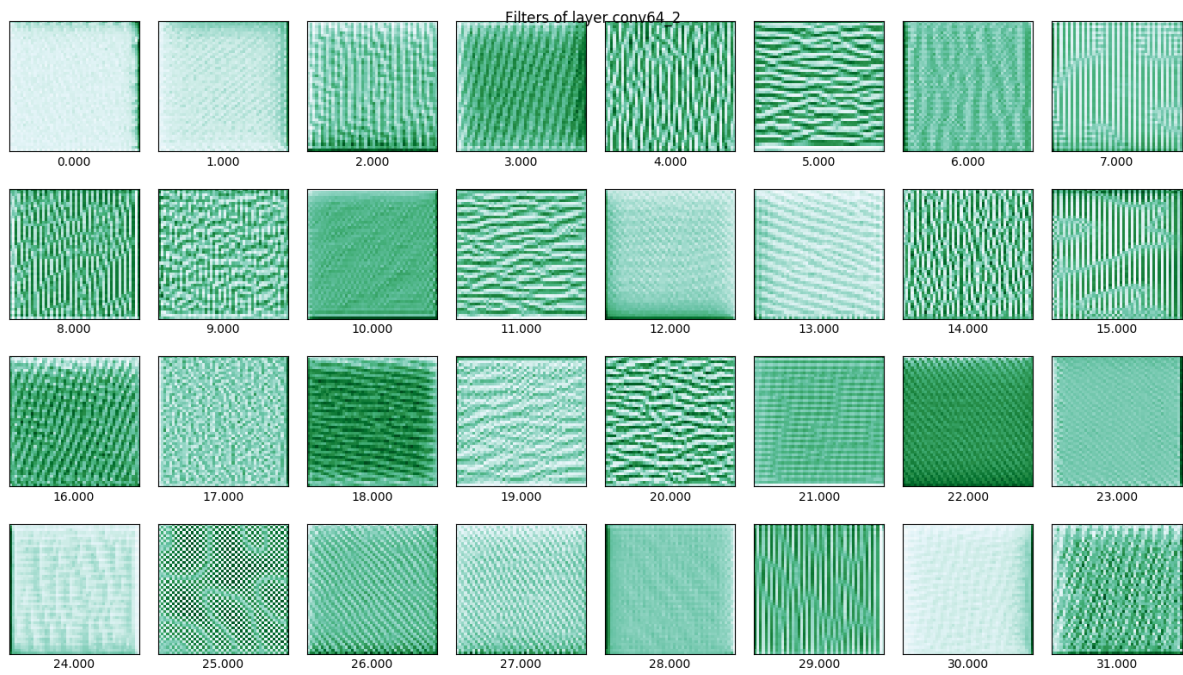
(1)

Conv64\_1



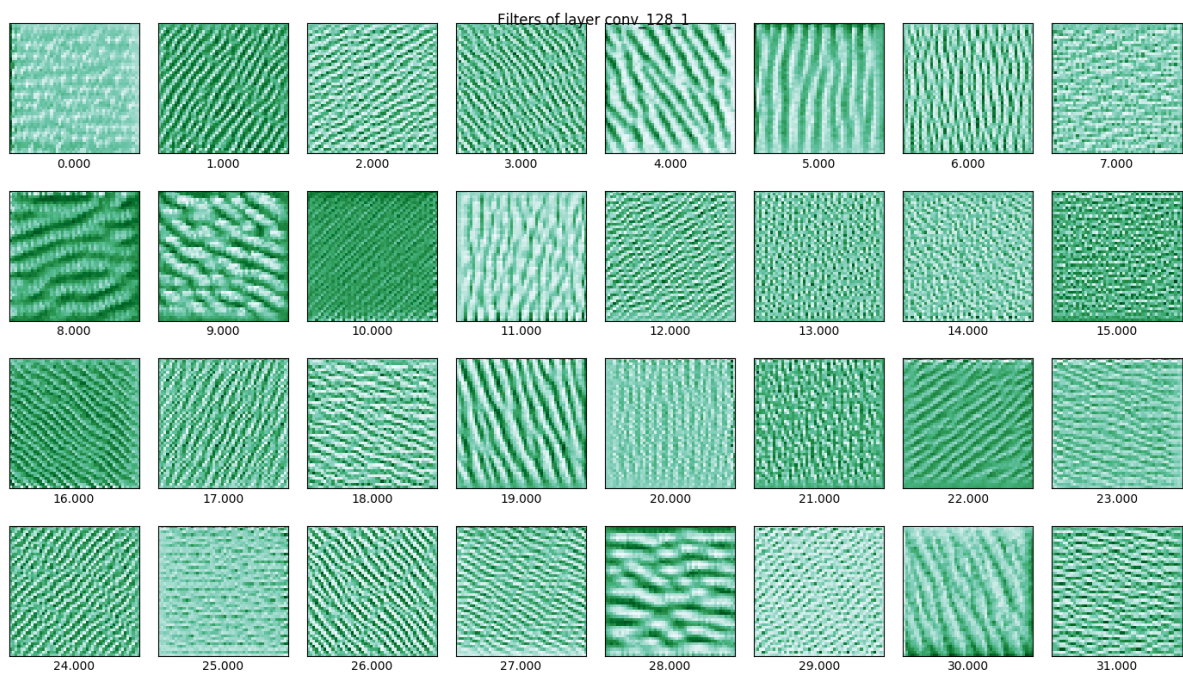
第一層 filter 好像只是看 image 的 intensity。

## Conv64\_2

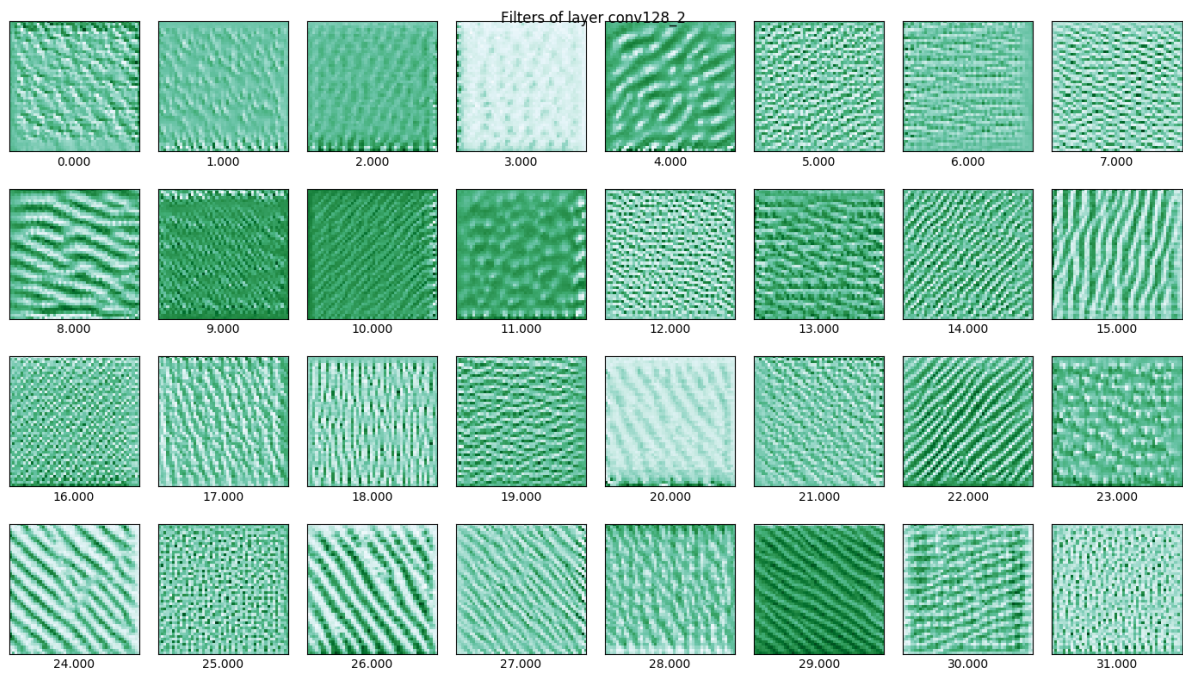


這層好像是在偵測一些線條(filter#29)和點(filter#25)。

## conv128\_1

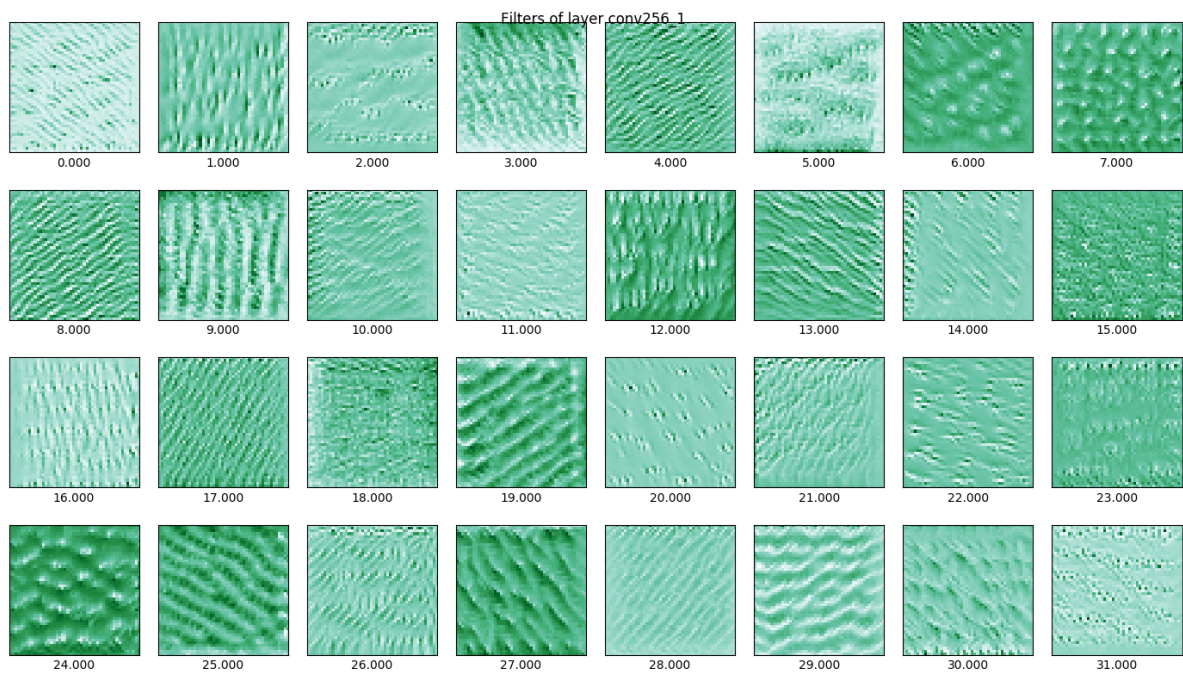


## Conv128\_2



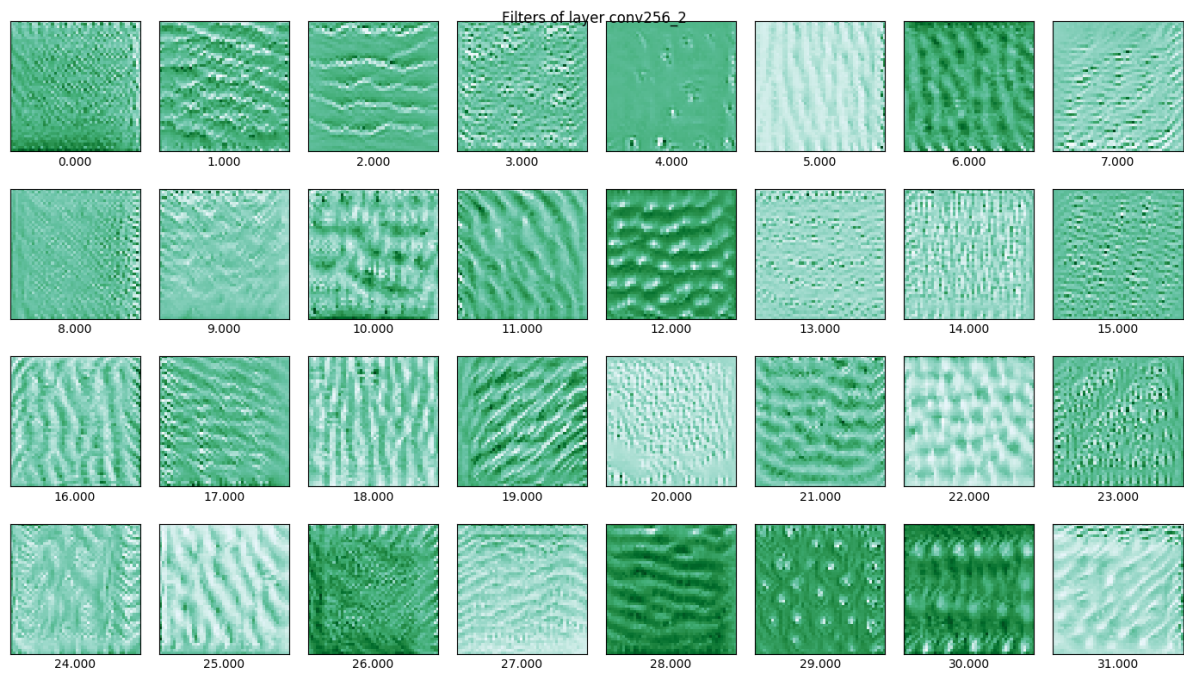
Conv128 相對 conv64 所偵測得圖案又更細緻(conv128\_1 filter#26)，有些部分點也更密了(conv128\_2 filter#25)。

## Conv256\_1



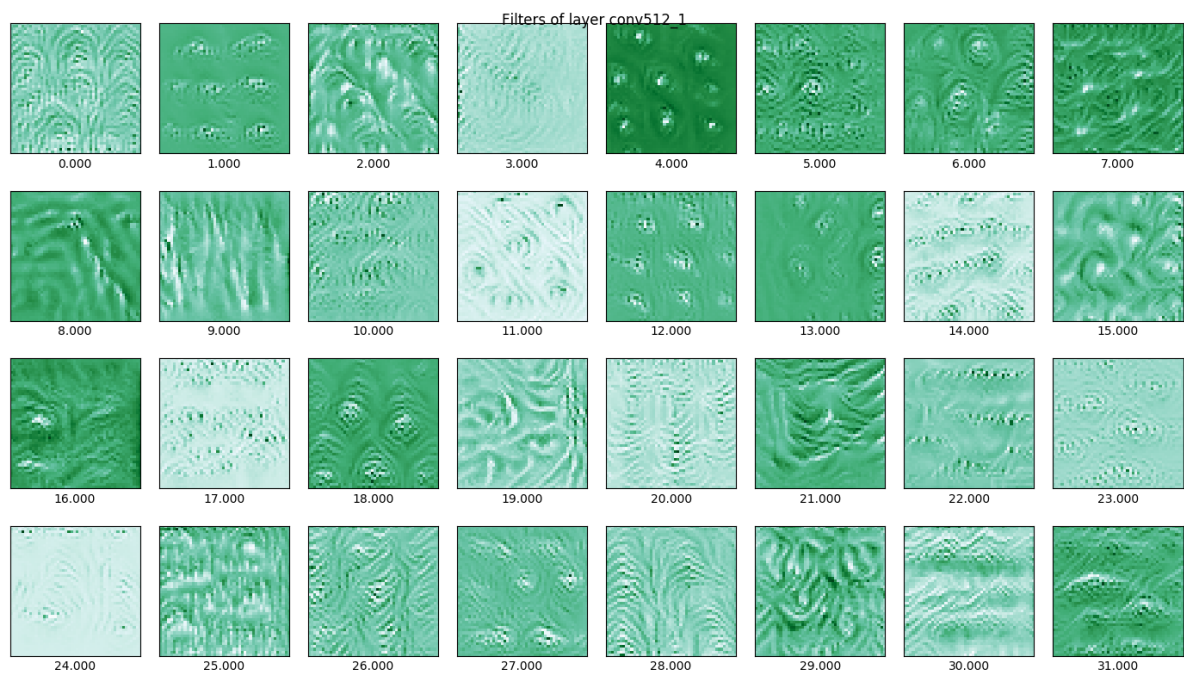


## Conv256\_2

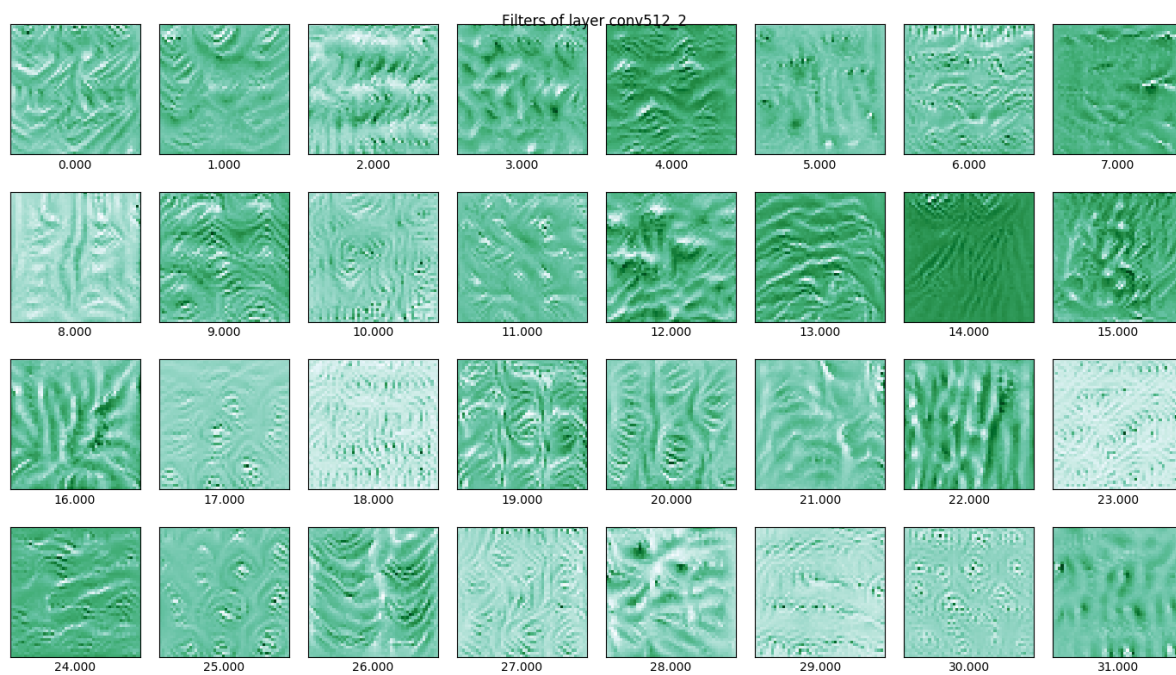


Conv256 所偵測到的圖案更大(conv256\_1 filter#27)而且已經略有一點形狀出來了(conv256\_2 filter#29)。

## Conv512\_1



## Conv512\_2



Conv512 看起來就有很多完整的形狀了，例如 conv512\_1 filter#31 感覺是在偵測肌肉線條，conv512\_2 filter#28 其實看起來也蠻像一個人臉了。

(2)

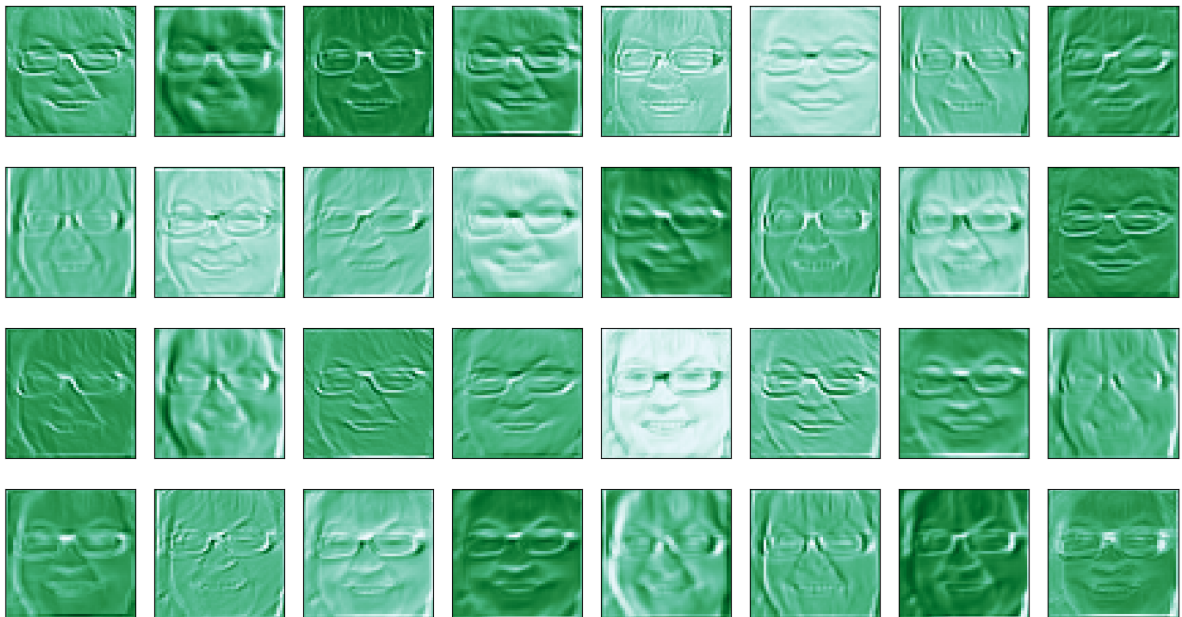
Conv64\_1

Output of layer0 (Given image100)



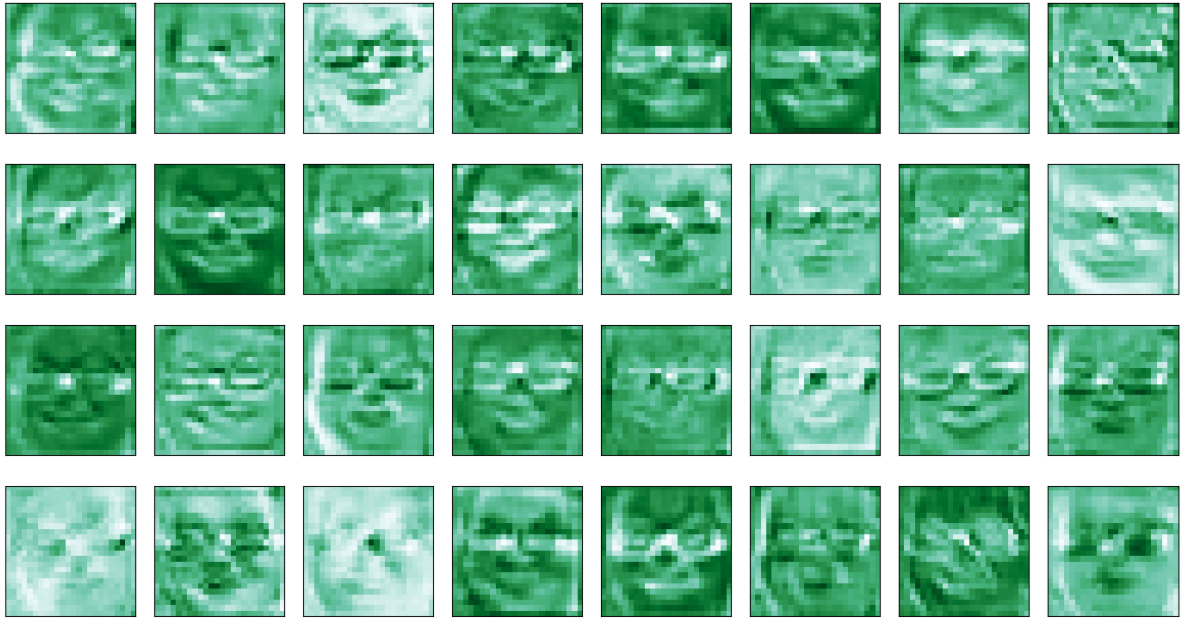
Ccconv64\_2

Output of layer1 (Given image100)



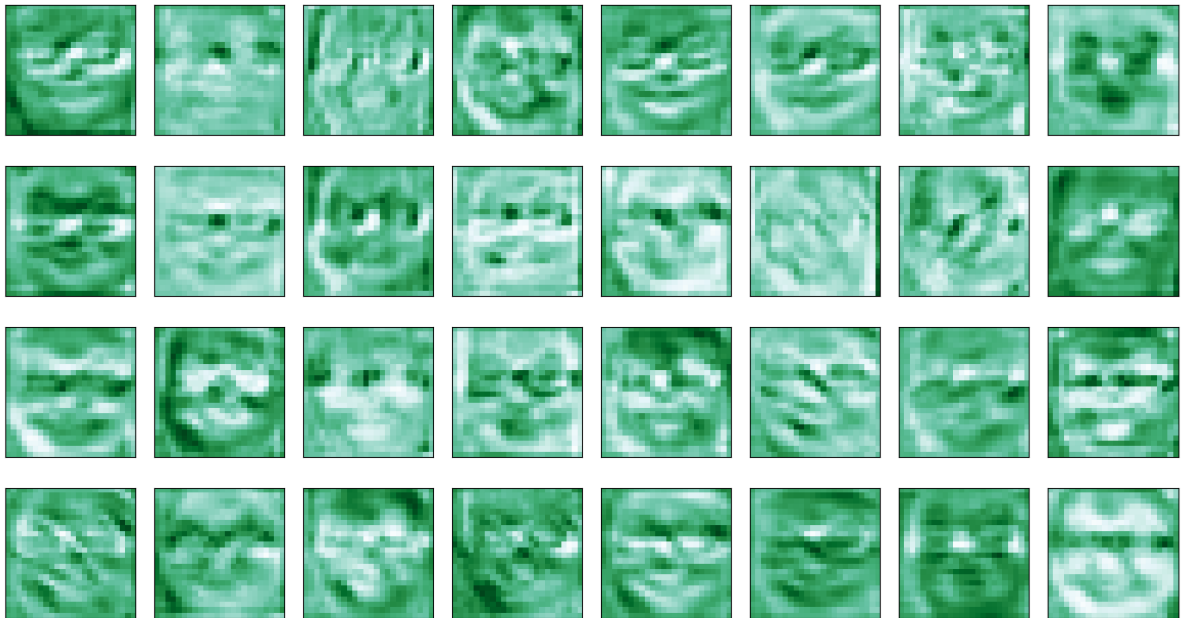
Conv128\_1

Output of layer2 (Given image100)



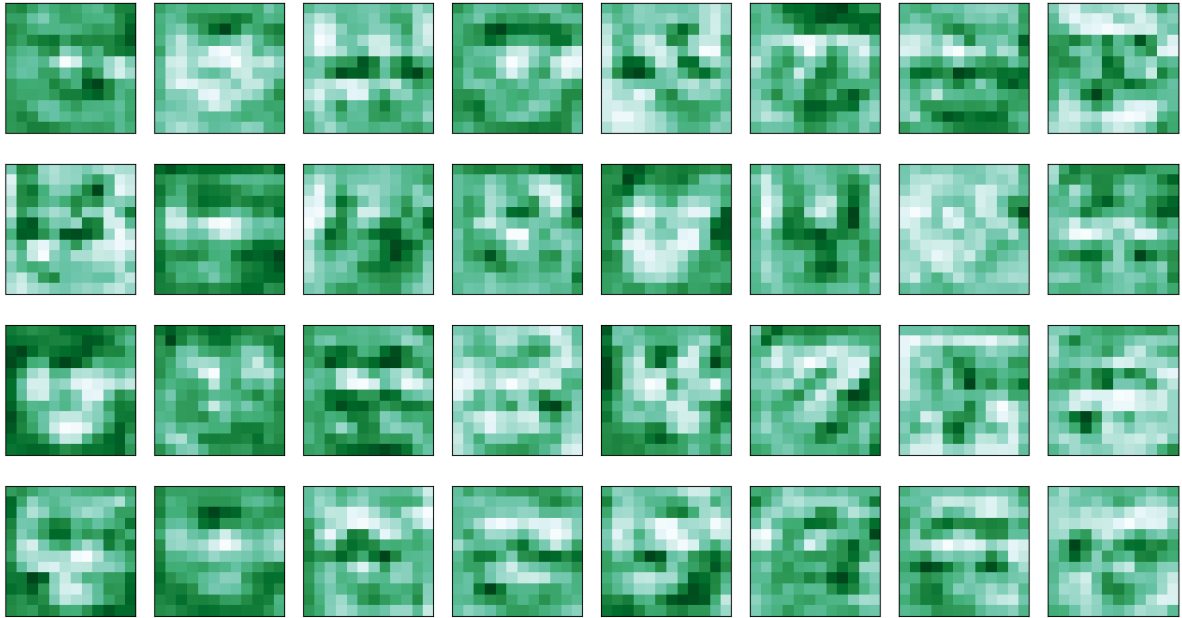
Conv128\_2

Output of layer3 (Given image100)



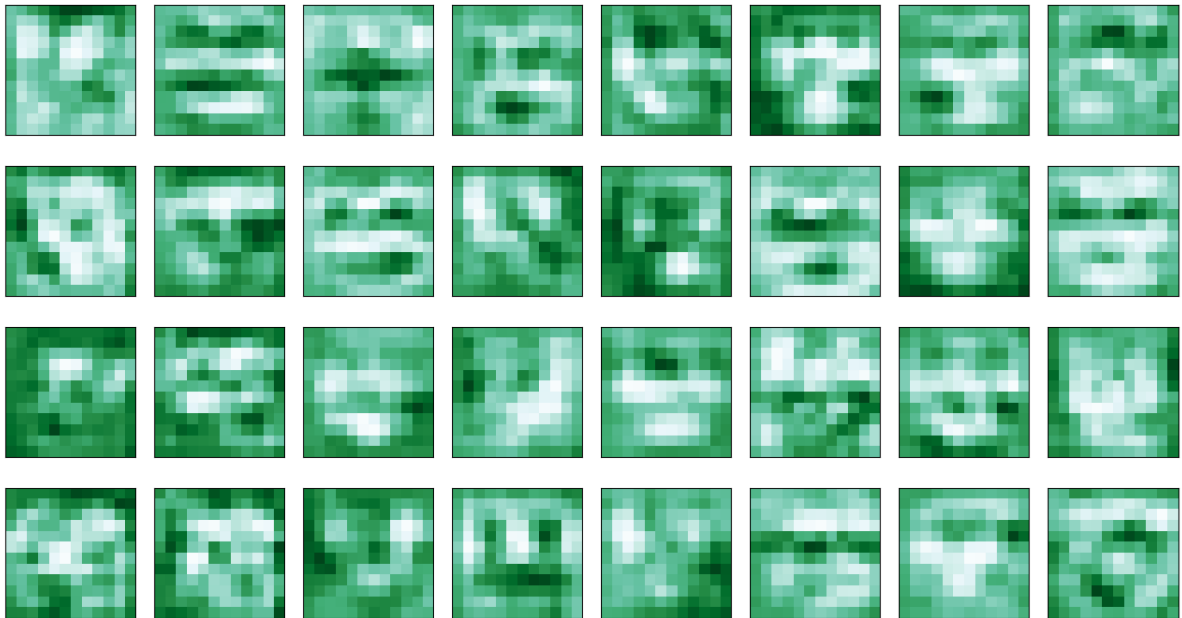
Conv256\_1

Output of layer4 (Given image100)



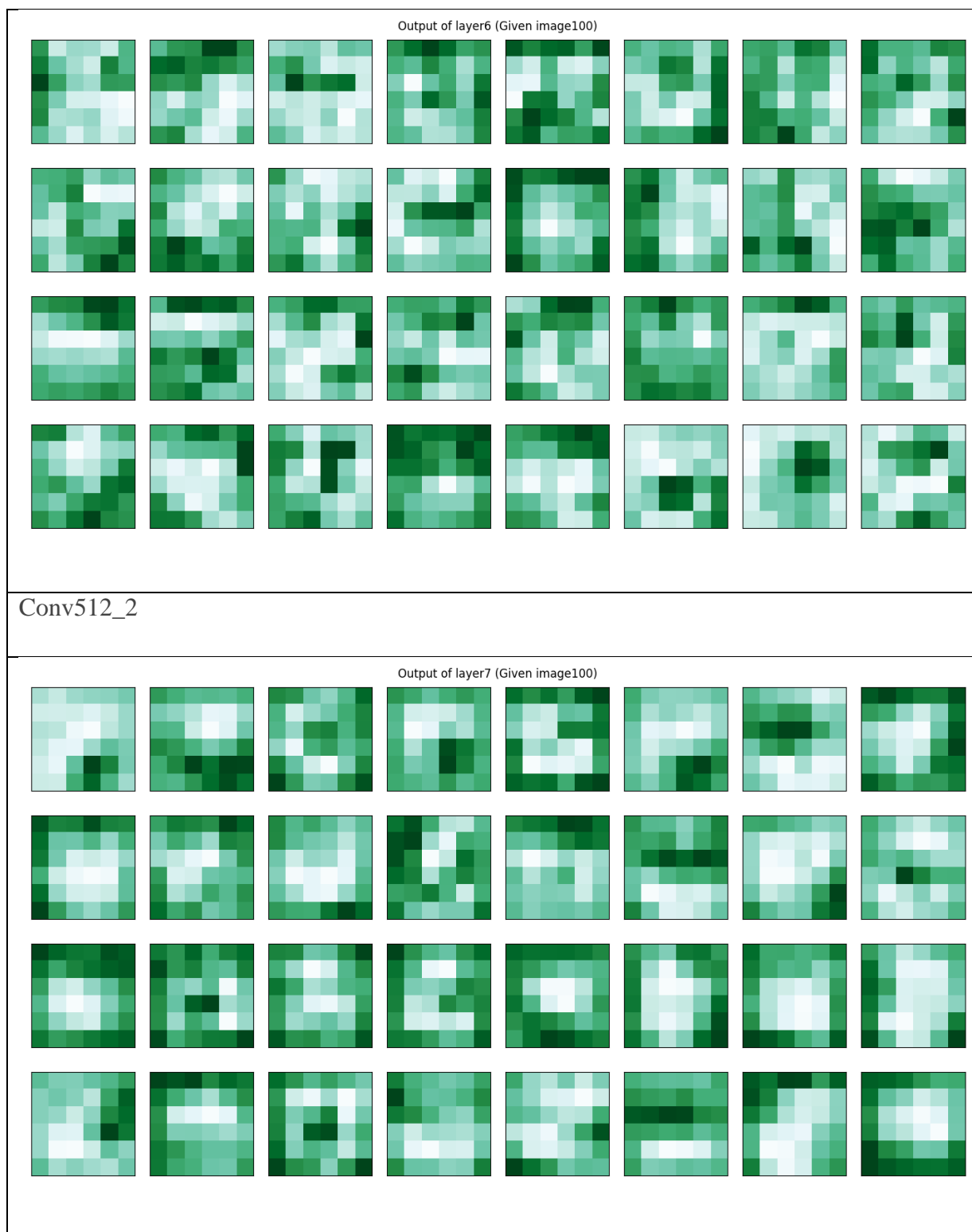
Conv256\_2

Output of layer5 (Given image100)



Conv512\_1





Conv\_64 幾乎和原圖一樣只是做了不同 edge 的 detection，但隨著 layer 數的增加和 maxpooling 的加入，圖片越來越模糊，也代表每個 filter 的結果越來越接近某些 feature。

[Bonus] (1%) 從 training data 中移除部份 label，實做 semi-supervised learning

我採用 self-training 的方式，因為教授投影片中說到做 semi-supervised learning，labeled data 量  $\gg$  unlabeled data 量，所以我把 training set 中，25000 筆 data 當作 unlabeled data，分成 10 份，剩下的當作 labeled data，搭配 problem1 的 network，training 方式如下：

先用 Labeled data 做 training  
Batch\_size=256  
Number of epoch=800(為了確保收斂)  
Optimizer: adam  
Initial learning rate= 0.000001  
decay=  $1e-3$   
initializers: RandomNormal(mean=0.0, stddev=0.05, seed=None)  
影像前處理:  
ImageDataGenerator(featurewise\_center=False, featurewise\_std\_normalization=False, rotation\_range=10, shear\_range=0.1, width\_shift\_range=0.1, height\_shift\_range=0.1, horizontal\_flip=True)  
(盡力使用所有 labeled 的 data)  
Dropout: no  
Number of parameters=40,372,679  
一個 epoch 9 秒



10 份 unlabeled data 中的一份，用 train 好的 model 對這份 data 做預測，並把這份 data 加入 labeled training data，training 方式如下：

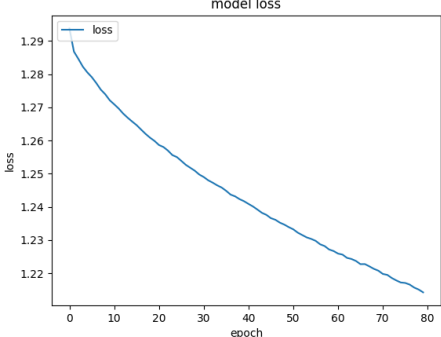
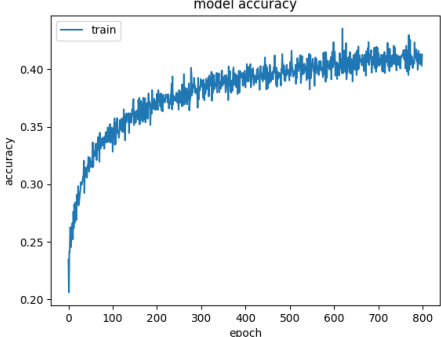
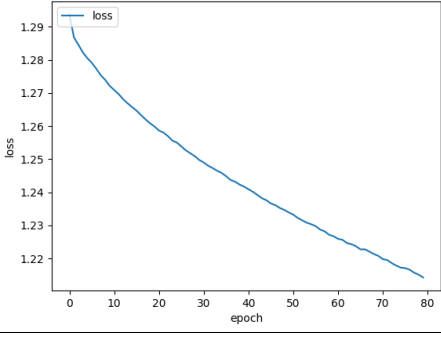
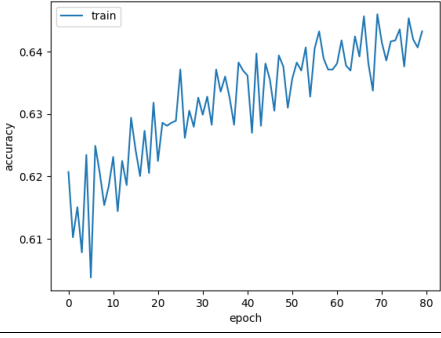
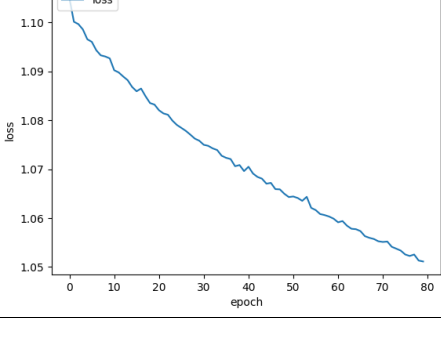
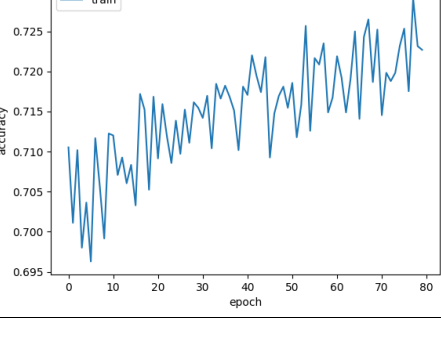
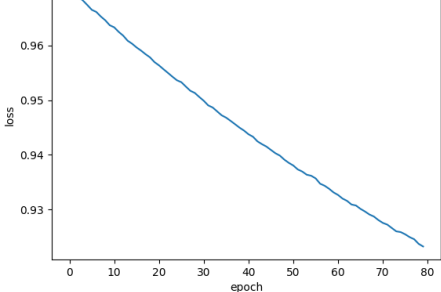
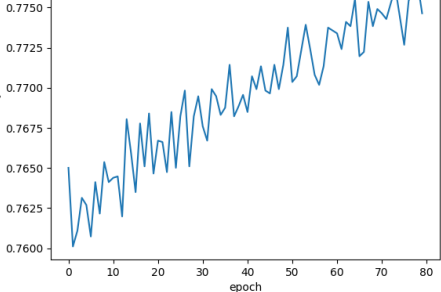
Batch\_size=256  
Number of epoch=80  
optimizer: adam  
Initial learning rate= 0.000001  
decay=  $1e-3$   
initializers: RandomNormal(mean=0.0, stddev=0.05, seed=None)  
影像前處理:no  
Dropout: no  
Number of parameters=40,372,679  
重複數步驟，直到 10 份 unlabeled data 都被 label 後加入 training data

結果:

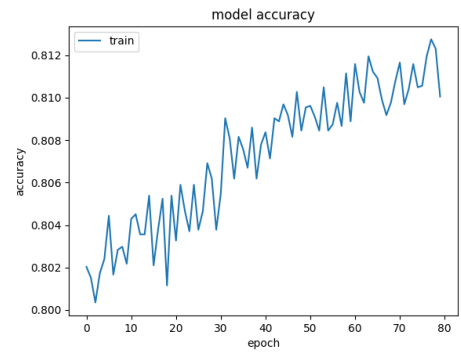
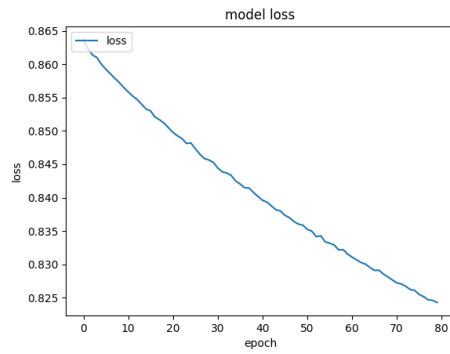
Training set=79.82863

Public leader board= 37.392%

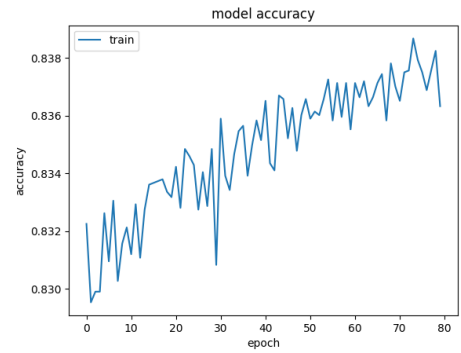
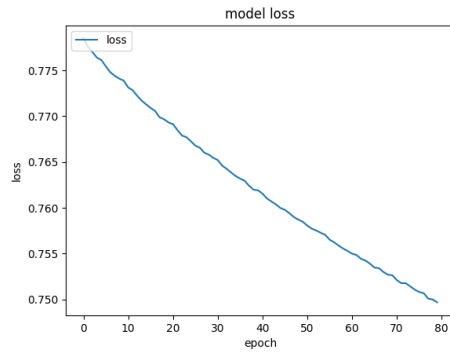
由於 training set 中的 data 真的太少，再加上那些被標錯 label 的 unlabel data 將會把我的 model 往錯誤的方向影響，導致我雖然在 training set 上表現還不錯(我在 model train 完之後，與正確 label 比較所得之結果)看起來還不錯，但在 public leader board 上卻爛掉了！

	loss	Accuracy
Label data		
Unlabel0		
Unlabel1		
Unlabel12		

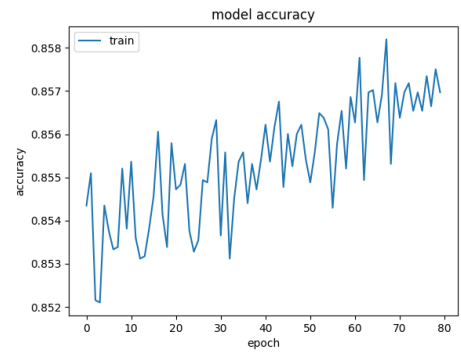
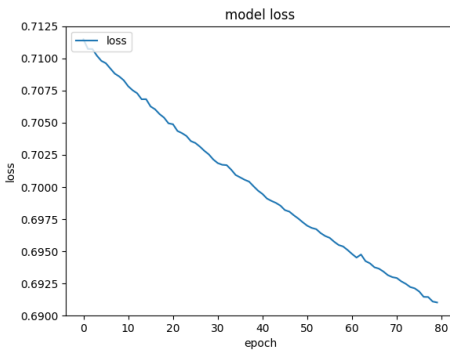
Unlabel13



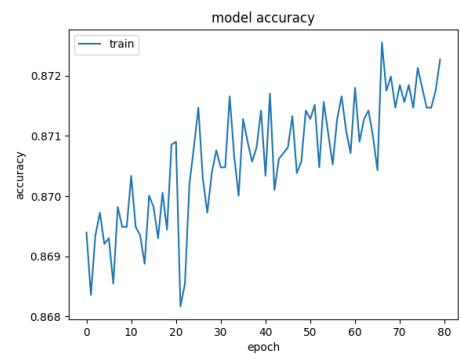
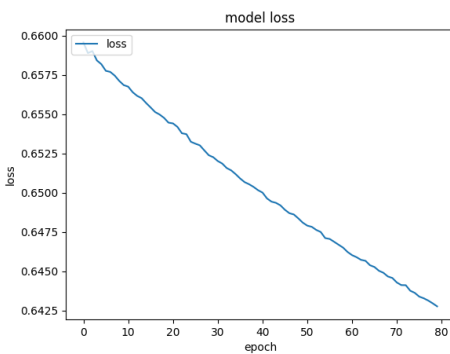
Unlabel14

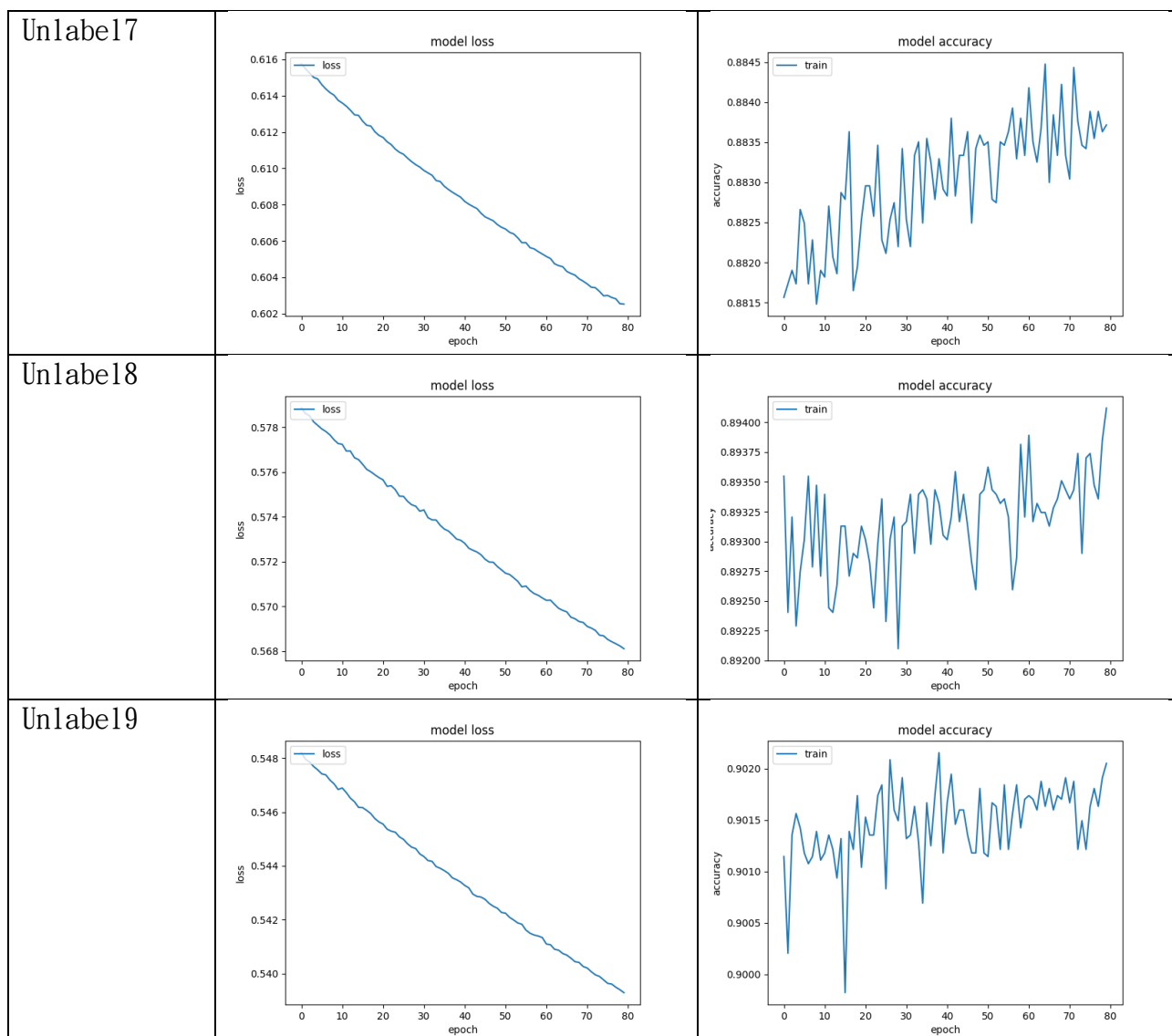


Unlabel15



Unlabel16





[Bonus] (1%) 在 Problem 5 中，提供了 3 個 hint，可以嘗試實作及觀察 (但也可以不限於 hint 所提到的方向，也可以自己去研究更多關於 CNN 細節的資料)，並說明你做了些什麼？ [完成 1 個: +0.4%, 完成 2 個: +0.7%, 完成 3 個: +1%]