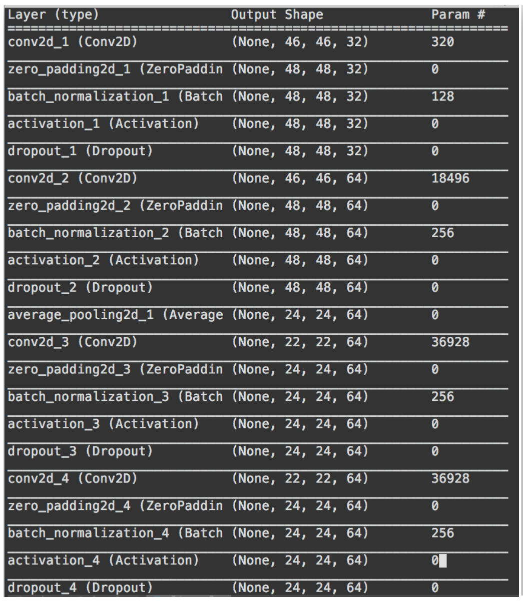
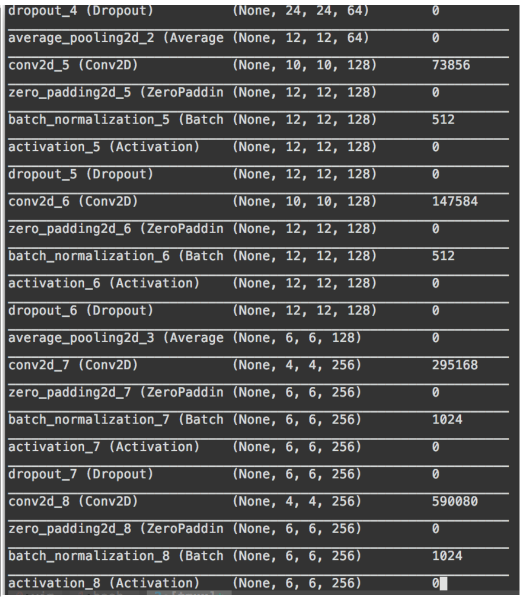
學號：B03902084 系級： 資工四　姓名：王藝霖

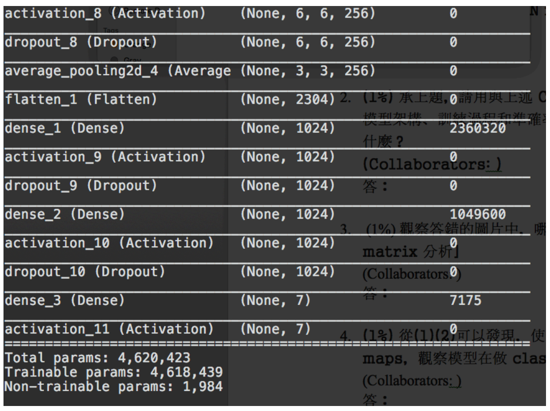
1. (1%) 請說明你實作的 CNN model，其模型架構、訓練過程和準確率為何？



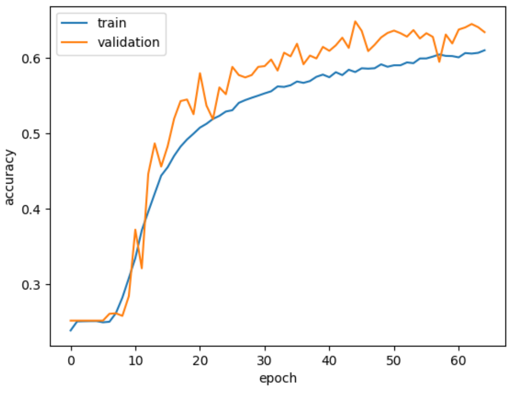
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 46, 46, 32)        320
_________________________________________________________________
zero_padding2d_1 (ZeroPaddin (None, 48, 48, 32)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 48, 48, 32)        128
_________________________________________________________________
activation_1 (Activation)    (None, 48, 48, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 48, 48, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 46, 46, 64)        18496
_________________________________________________________________
zero_padding2d_2 (ZeroPaddin (None, 48, 48, 64)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 48, 48, 64)        256
_________________________________________________________________
activation_2 (Activation)    (None, 48, 48, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 48, 48, 64)        0
_________________________________________________________________
average_pooling2d_1 (Average (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 22, 22, 64)        36928
_________________________________________________________________
zero_padding2d_3 (ZeroPaddin (None, 24, 24, 64)        0
_________________________________________________________________
batch_normalization_3 (Batch (None, 24, 24, 64)        256
_________________________________________________________________
activation_3 (Activation)    (None, 24, 24, 64)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 22, 22, 64)        36928
_________________________________________________________________
zero_padding2d_4 (ZeroPaddin (None, 24, 24, 64)        0
_________________________________________________________________
batch_normalization_4 (Batch (None, 24, 24, 64)        256
_________________________________________________________________
activation_4 (Activation)    (None, 24, 24, 64)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 24, 24, 64)        0
```



```
dropout_4 (Dropout)          (None, 24, 24, 64)        0
_________________________________________________________________
average_pooling2d_2 (Average (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 10, 10, 128)       73856
_________________________________________________________________
zero_padding2d_5 (ZeroPaddin (None, 12, 12, 128)       0
_________________________________________________________________
batch_normalization_5 (Batch (None, 12, 12, 128)       512
_________________________________________________________________
activation_5 (Activation)    (None, 12, 12, 128)       0
_________________________________________________________________
dropout_5 (Dropout)          (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 10, 10, 128)       147584
_________________________________________________________________
zero_padding2d_6 (ZeroPaddin (None, 12, 12, 128)       0
_________________________________________________________________
batch_normalization_6 (Batch (None, 12, 12, 128)       512
_________________________________________________________________
activation_6 (Activation)    (None, 12, 12, 128)       0
_________________________________________________________________
dropout_6 (Dropout)          (None, 12, 12, 128)       0
_________________________________________________________________
average_pooling2d_3 (Average (None, 6, 6, 128)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 4, 4, 256)         295168
_________________________________________________________________
zero_padding2d_7 (ZeroPaddin (None, 6, 6, 256)         0
_________________________________________________________________
batch_normalization_7 (Batch (None, 6, 6, 256)         1024
_________________________________________________________________
activation_7 (Activation)    (None, 6, 6, 256)         0
_________________________________________________________________
dropout_7 (Dropout)          (None, 6, 6, 256)         0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 4, 4, 256)         590080
_________________________________________________________________
zero_padding2d_8 (ZeroPaddin (None, 6, 6, 256)         0
_________________________________________________________________
batch_normalization_8 (Batch (None, 6, 6, 256)         1024
_________________________________________________________________
activation_8 (Activation)    (None, 6, 6, 256)         0
```

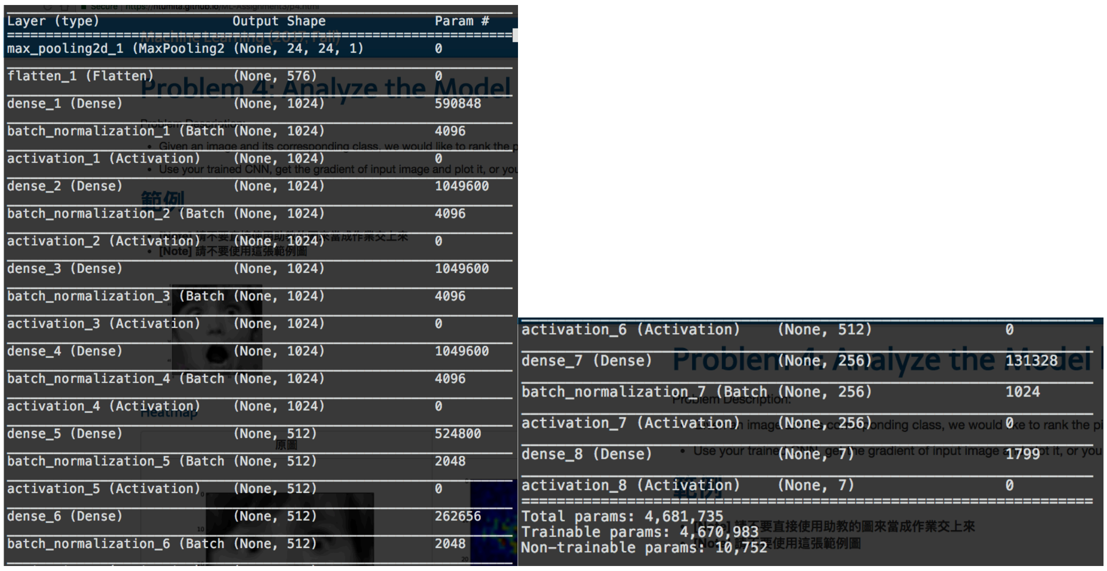


```
activation_8 (Activation)    (None, 6, 6, 256)         0
_________________________________________________________________
dropout_8 (Dropout)          (None, 6, 6, 256)         0
_________________________________________________________________
average_pooling2d_4 (Average (None, 3, 3, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              2360320
_________________________________________________________________
activation_9 (Activation)    (None, 1024)              0
_________________________________________________________________
dropout_9 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              1049600
_________________________________________________________________
activation_10 (Activation)   (None, 1024)              0
_________________________________________________________________
dropout_10 (Dropout)         (None, 1024)              0
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 7175
_________________________________________________________________
activation_11 (Activation)   (None, 7)                 0
=================================================================
Total params: 4,620,423
Trainable params: 4,618,439
Non-trainable params: 1,984
```

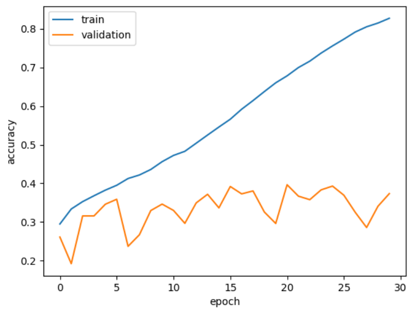


大致上是 ccpccpccpccpff 的架構（參考 vgg 架構），其中 pooling 使用 average pooling，activation funcation 使用 relu，有加上 batch normalization。訓練時有使用 keras image generator 來翻轉、旋轉、位移圖片。訓練過程的**圖如上，validation 取原始資料的 10%，可能是因為 validation 資料數量較小，因此有較大的波動。Kaggle 上的最高分數是取多次 epoch ensemble 後的結果，分別是第 62,63,71,74 次 epoch，選取的方式為在 validation 上表現最好的。Kaggle 上的**accuracy 為：0.6883

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？

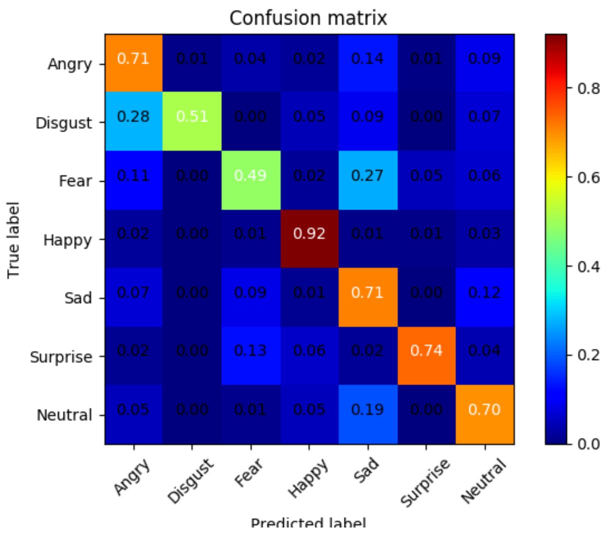


```
Layer (type)                 Output Shape              Param #
=================================================================
max_pooling2d_1 (MaxPooling2 (None, 24, 24, 1)         0
flatten_1 (Flatten)          (None, 576)               0
dense_1 (Dense)              (None, 1024)              590848
batch_normalization_1 (Batch (None, 1024)              4096
activation_1 (Activation)    (None, 1024)              0
dense_2 (Dense)              (None, 1024)              1049600
batch_normalization_2 (Batch (None, 1024)              4096
activation_2 (Activation)    (None, 1024)              0
dense_3 (Dense)              (None, 1024)              1049600
batch_normalization_3 (Batch (None, 1024)              4096
activation_3 (Activation)    (None, 1024)              0
dense_4 (Dense)              (None, 1024)              1049600
batch_normalization_4 (Batch (None, 1024)              4096
activation_4 (Activation)    (None, 1024)              0
dense_5 (Dense)              (None, 512)               524800
batch_normalization_5 (Batch (None, 512)               2048
activation_5 (Activation)    (None, 512)               0
dense_6 (Dense)              (None, 512)               262656
batch_normalization_6 (Batch (None, 512)               2048
activation_6 (Activation)    (None, 512)               0
dense_7 (Dense)              (None, 256)               131328
batch_normalization_7 (Batch (None, 256)               1024
activation_7 (Activation)    (None, 256)               0
dense_8 (Dense)              (None, 7)                 1799
activation_8 (Activation)    (None, 7)                 0
=================================================================
Total params: 4,681,735
Trainable params: 4,670,983
Non-trainable params: 10,752
```



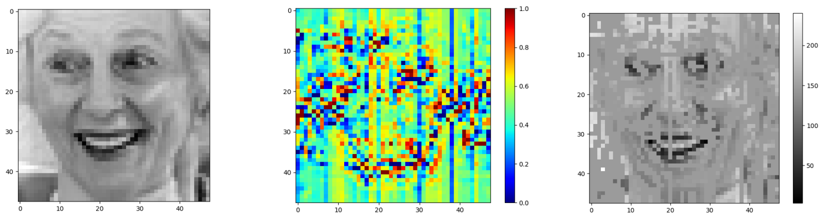
雖然參數差不多多，但結果準確率卻差非常多。模型架構大概就是用了 6 層 fully connected network，最後參數 4,670,983 和 cnn 的 4,618,439 差不多。Kaggle 上的準確率：0.365975，相較 cnn 差了非常多。

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]



其中，可發現錯誤其實算是平均的，之間差距沒有非常大。其中較大的部分有兩個：1. 把 disgust 判斷成 angry 2. 把 fear 判斷成 sad。可以想像容易混淆的 class 之間確實是有一定相似度，也造成辨識上的困難。

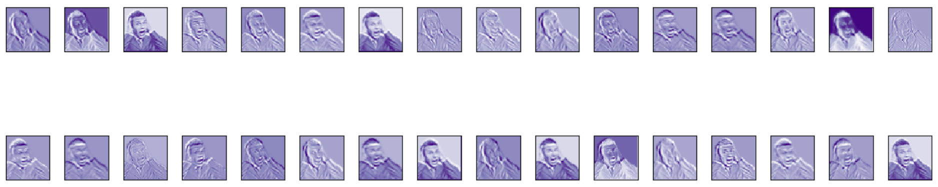
4. (1%) 從(1)(2)可以發現，使用 CNN 的確有些好處，試繪出其 saliency maps，觀察模型在做 classification 時，是 focus 在圖片的哪些部份



大致上是 focus 在眼睛和嘴巴的部分，對於表情來說的確是重要的，看來 cnn 有抓到重點。

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。

Output of layer0 (Given image17)



因為圖片資料的處理比較不熟悉，再加上不知道助教手把手教學的 input output 是怎麼樣的形式，和一點 shape 的問題，導致沒辦法做出 filter。Gradient ascent 如果是用數值資料應該能夠處理。猜測這些圖片的 filter 例如右上第二張，應該是比較偏向平面的 texture