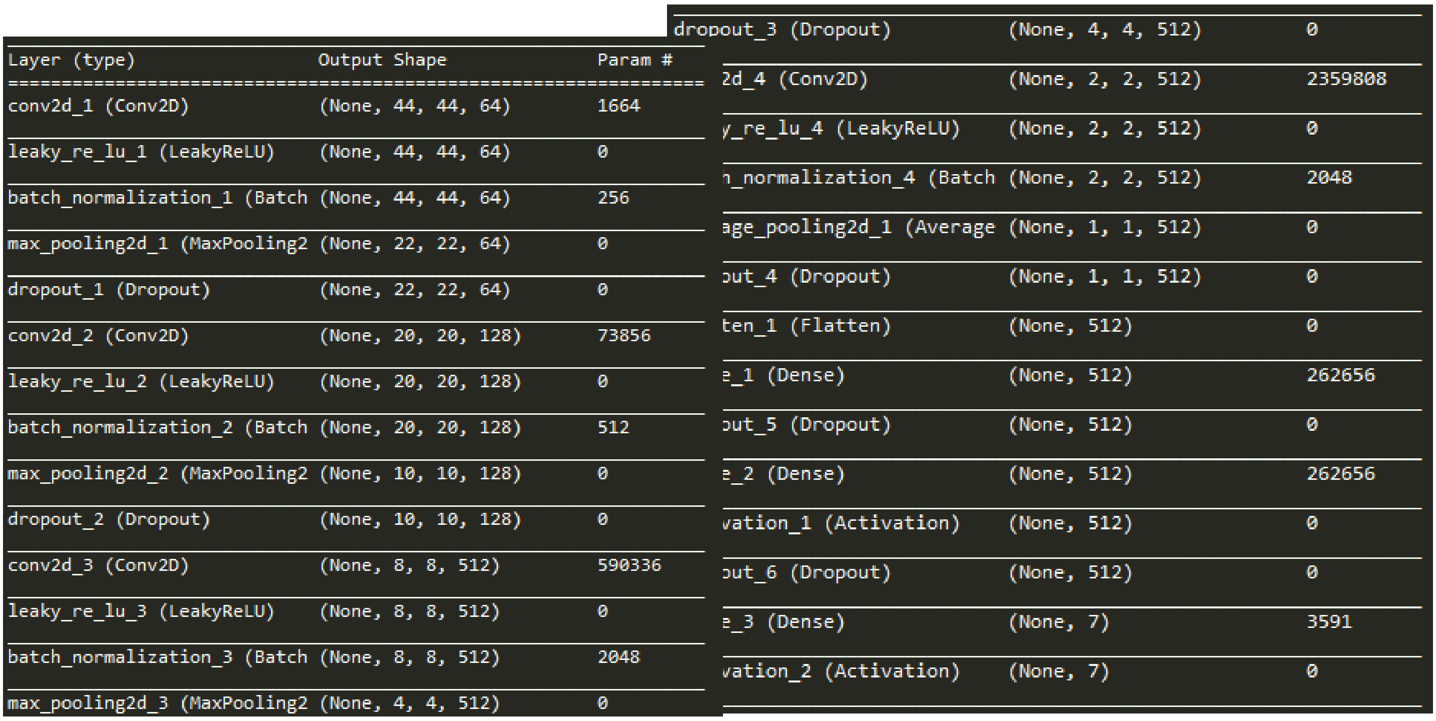
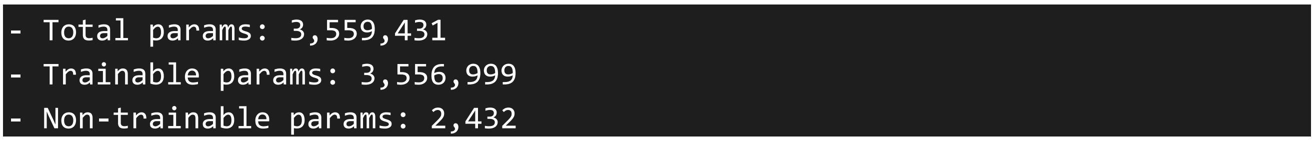
學號：R05546022 系級： 工工所碩二 姓名：謝立成

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？

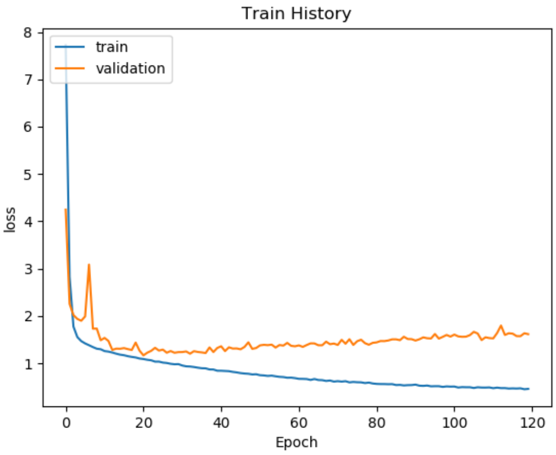


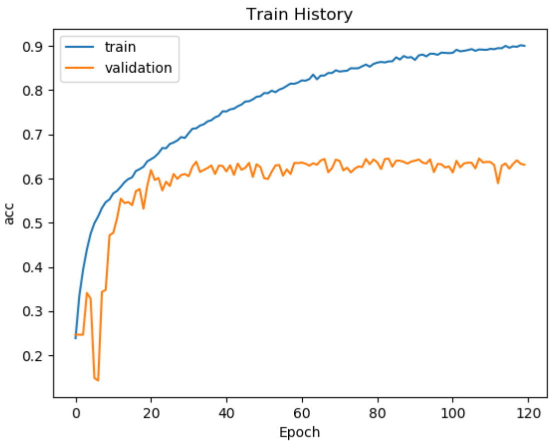
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 44, 44, 64)        1664
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 44, 44, 64)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 44, 44, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 22, 22, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 22, 22, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 20, 20, 128)       73856
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 20, 20, 128)       0
_________________________________________________________________
batch_normalization_2 (Batch (None, 20, 20, 128)       512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 10, 10, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 512)         590336
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 8, 8, 512)         0
_________________________________________________________________
batch_normalization_3 (Batch (None, 8, 8, 512)         2048
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 512)         0
```

```
dropout_3 (Dropout)          (None, 4, 4, 512)         0
_________________________________________________________________
2d_4 (Conv2D)            (None, 2, 2, 512)         2359808
_________________________________________________________________
y_re_lu_4 (LeakyReLU)    (None, 2, 2, 512)         0
_________________________________________________________________
n_normalization_4 (Batch (None, 2, 2, 512)         2048
_________________________________________________________________
age_pooling2d_1 (Average (None, 1, 1, 512)         0
_________________________________________________________________
out_4 (Dropout)          (None, 1, 1, 512)         0
_________________________________________________________________
ten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
e_1 (Dense)              (None, 512)               262656
_________________________________________________________________
out_5 (Dropout)          (None, 512)               0
_________________________________________________________________
e_2 (Dense)              (None, 512)               262656
_________________________________________________________________
vation_1 (Activation)    (None, 512)               0
_________________________________________________________________
out_6 (Dropout)          (None, 512)               0
_________________________________________________________________
e_3 (Dense)              (None, 7)                 3591
_________________________________________________________________
vation_2 (Activation)    (None, 7)                 0
_________________________________________________________________
```



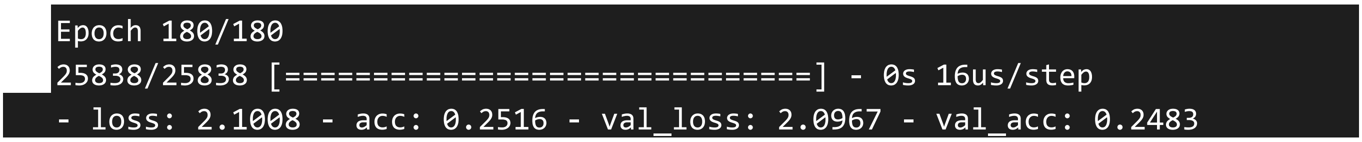
```
- Total params: 3,559,431
- Trainable params: 3,556,999
- Non-trainable params: 2,432
```





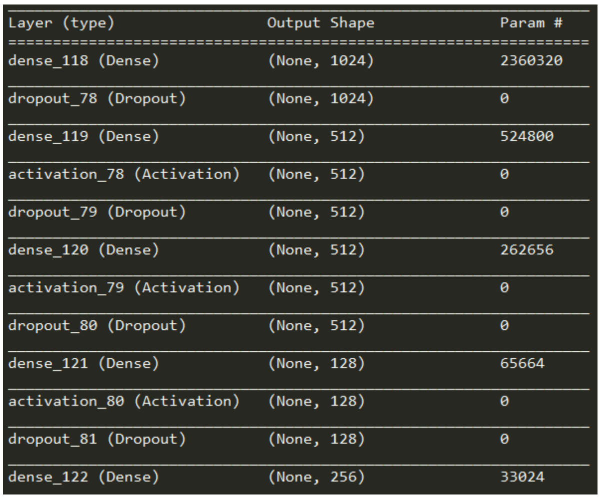
Final Valid ACC are around 65%

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？
(Collaborators: )

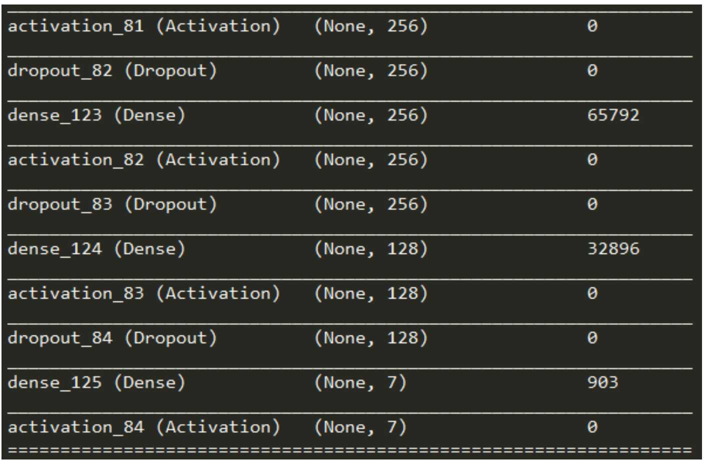


```
Epoch 180/180
25838/25838 [==============================] - 0s 16us/step
- loss: 2.1008 - acc: 0.2516 - val_loss: 2.0967 - val_acc: 0.2483
```

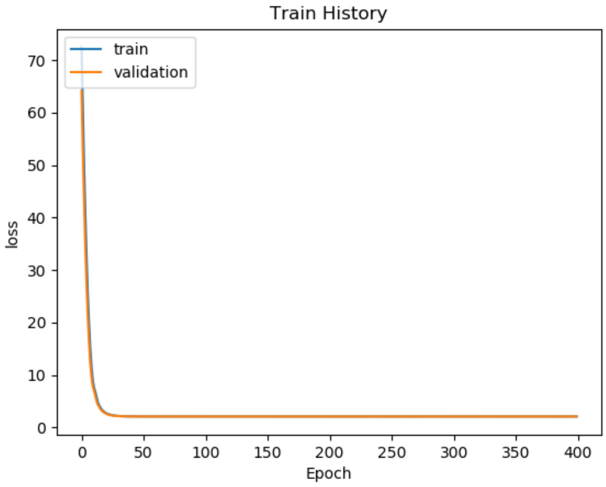
DNN sucks, it the model will be in local optimal 0.2483 even more often than CNN for image processing.

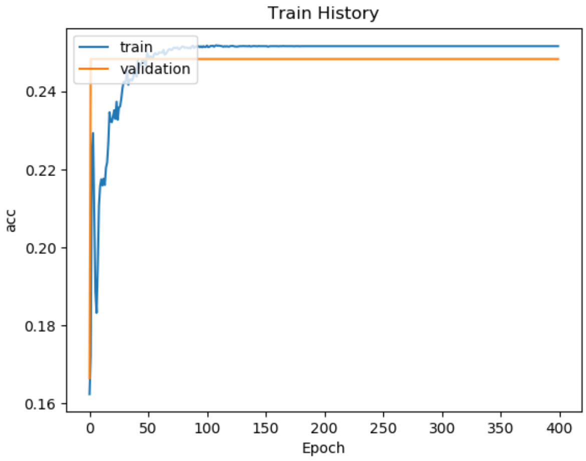


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_118 (Dense)            (None, 1024)              2360320
_________________________________________________________________
dropout_78 (Dropout)         (None, 1024)              0
_________________________________________________________________
dense_119 (Dense)            (None, 512)               524800
_________________________________________________________________
activation_78 (Activation)   (None, 512)               0
_________________________________________________________________
dropout_79 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_120 (Dense)            (None, 512)               262656
_________________________________________________________________
activation_79 (Activation)   (None, 512)               0
_________________________________________________________________
dropout_80 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_121 (Dense)            (None, 128)               65664
_________________________________________________________________
activation_80 (Activation)   (None, 128)               0
_________________________________________________________________
dropout_81 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_122 (Dense)            (None, 256)               33024
```



```
_________________________________________________________________
activation_81 (Activation)   (None, 256)               0
_________________________________________________________________
dropout_82 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_123 (Dense)            (None, 256)               65792
_________________________________________________________________
activation_82 (Activation)   (None, 256)               0
_________________________________________________________________
dropout_83 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_124 (Dense)            (None, 128)               32896
_________________________________________________________________
activation_83 (Activation)   (None, 128)               0
_________________________________________________________________
dropout_84 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_125 (Dense)            (None, 7)                 903
_________________________________________________________________
activation_84 (Activation)   (None, 7)                 0
=================================================================
```

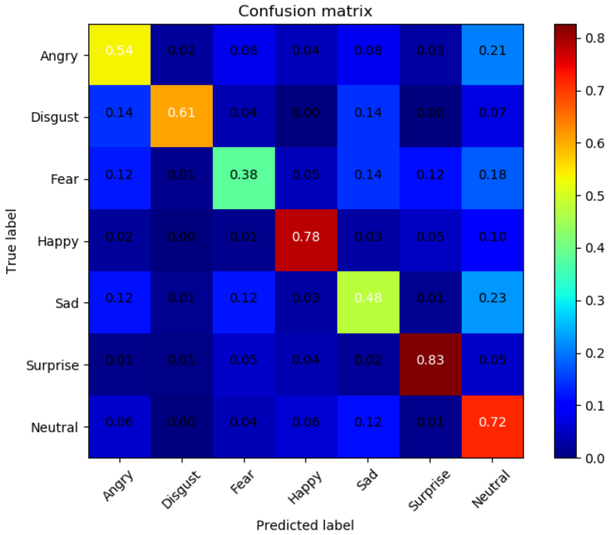




3. (1%) 觀察答錯的圖片中，哪些 class 彼此間容易用混？[繪出 confusion matrix 分析]

(Collaborators: )

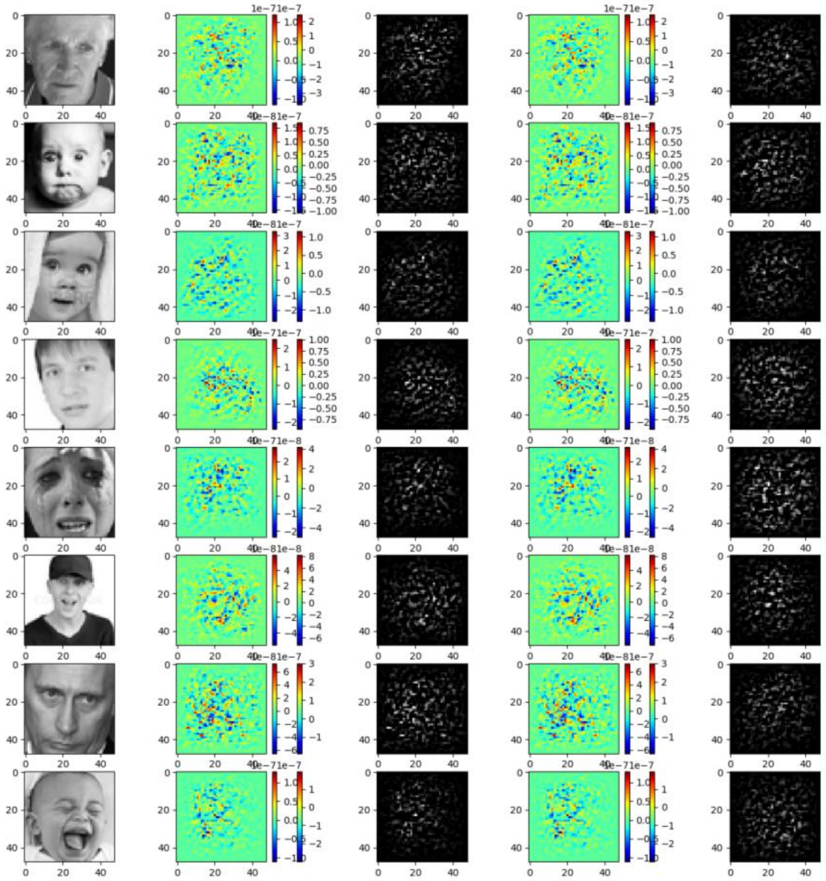
It appears that class “Neutral” are a bit easy to be classified as incorrect ; but are good Overall.



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？

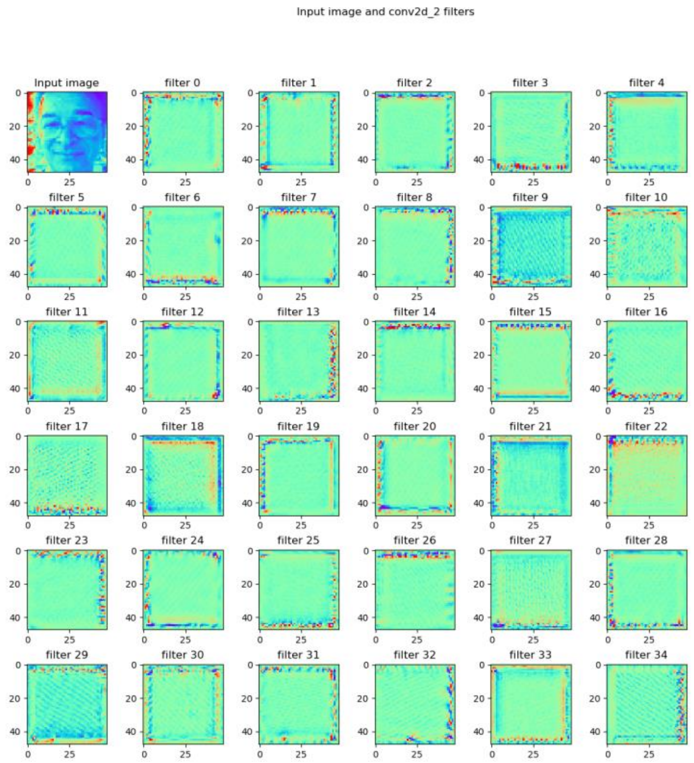
(Collaborators: r05546003 r05546024 r05546026)

If mostly focus on the central facial area, or the nose area.



答：

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。
   (Collaborators: r05546003 r05546024 r05546026 )
   答：It appears filter 9, 21, 29 are activated.


Input image and conv2d_2 filters
Input image
filter 0
filter 1
filter 2
filter 3
filter 4
filter 5
filter 6
filter 7
filter 8
filter 9
filter 10
filter 11
filter 12
filter 13
filter 14
filter 15
filter 16
filter 17
filter 18
filter 19
filter 20
filter 21
filter 22
filter 23
filter 24
filter 25
filter 26
filter 27
filter 28
filter 29
filter 30
filter 31
filter 32
filter 33
filter 34