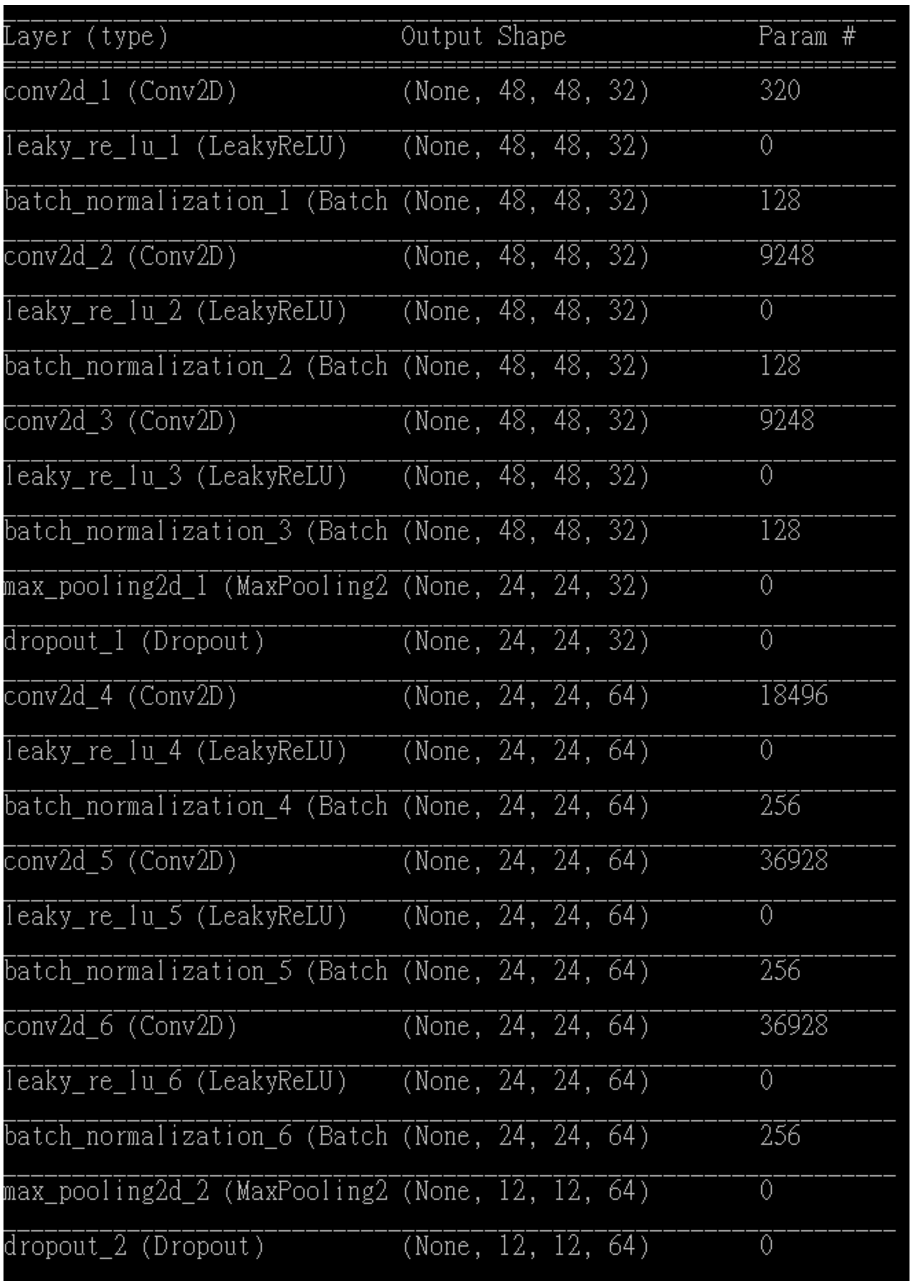
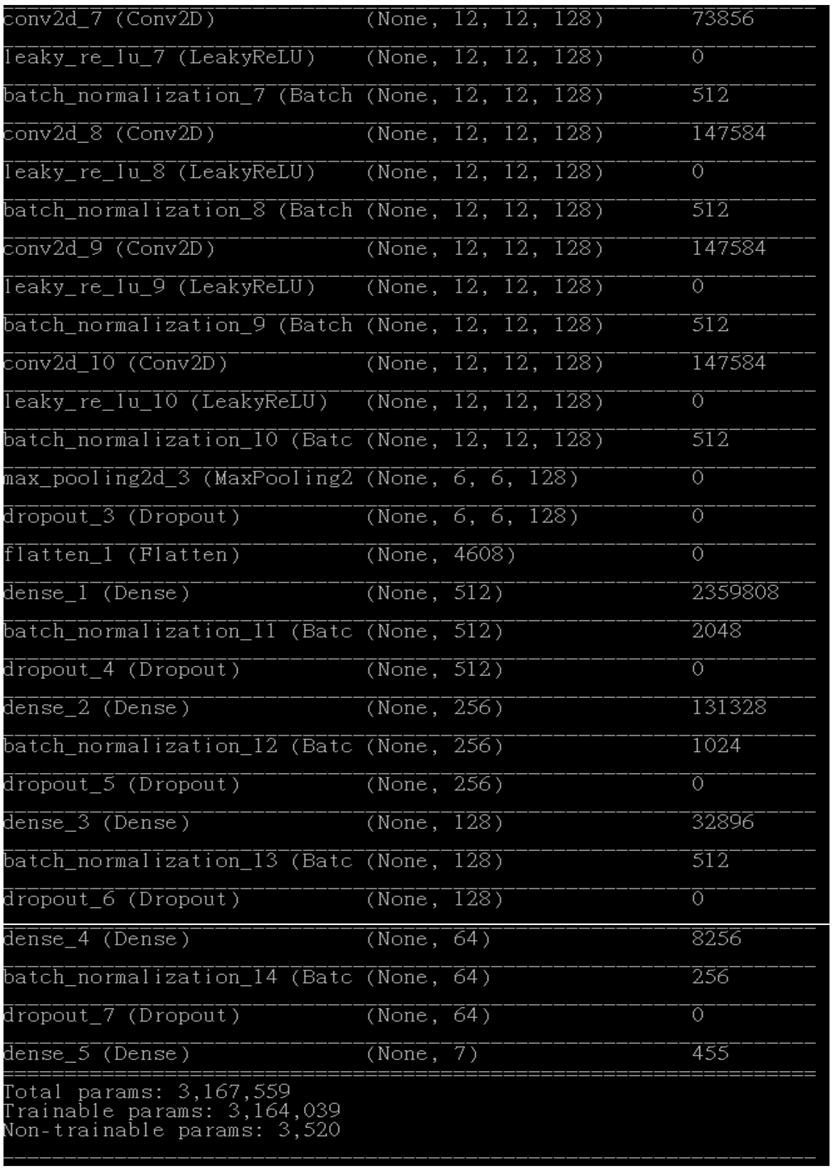
學號：R06922152 系級： 資工碩一 姓名：袁晟峻

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？
   (Collaborators: 呂承洋、陳柏堯、邵志宇)

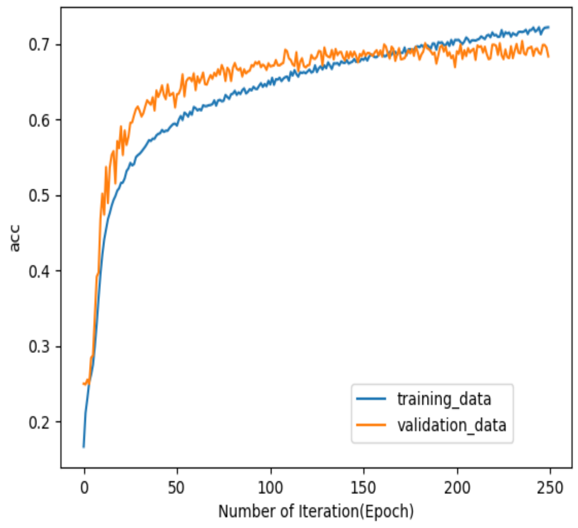
答：



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 48, 48, 32)        320
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 48, 48, 32)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 48, 48, 32)        128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 48, 48, 32)        9248
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 48, 48, 32)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 48, 48, 32)        128
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 48, 48, 32)        9248
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 48, 48, 32)        0
_________________________________________________________________
batch_normalization_3 (Batch (None, 48, 48, 32)        128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 24, 24, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 24, 24, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 24, 24, 64)        0
_________________________________________________________________
batch_normalization_4 (Batch (None, 24, 24, 64)        256
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 24, 24, 64)        36928
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 24, 24, 64)        0
_________________________________________________________________
batch_normalization_5 (Batch (None, 24, 24, 64)        256
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 24, 24, 64)        36928
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 24, 24, 64)        0
_________________________________________________________________
batch_normalization_6 (Batch (None, 24, 24, 64)        256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 12, 12, 64)        0
```

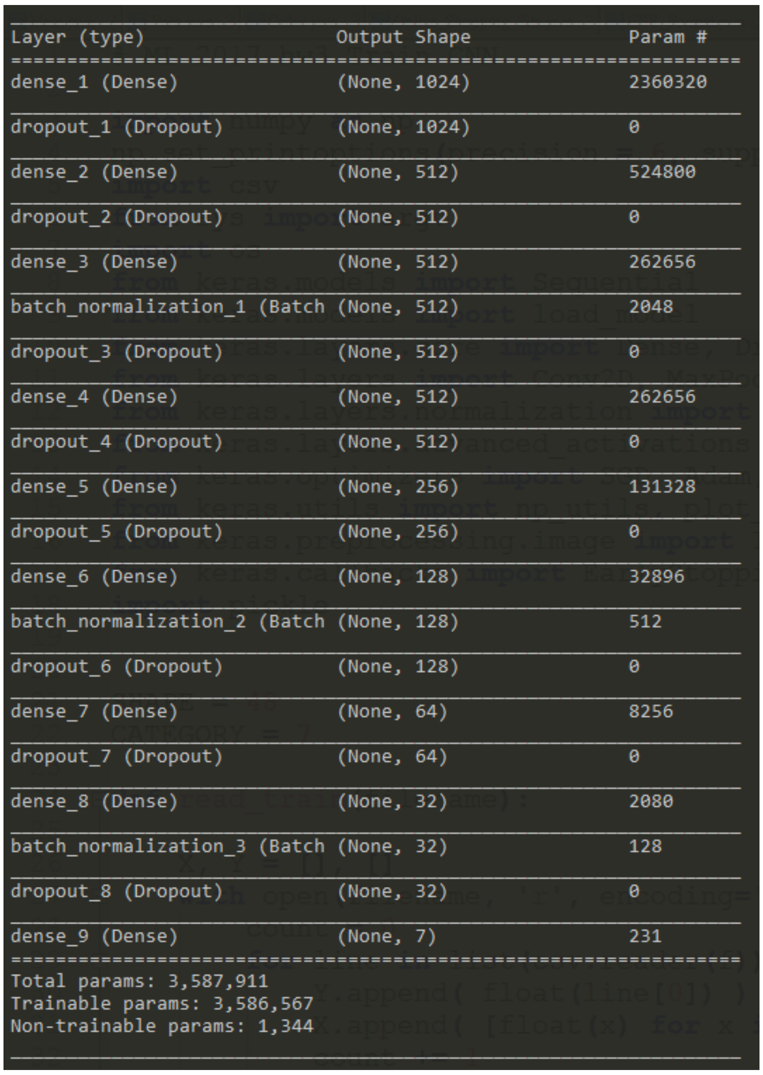


```
conv2d_7 (Conv2D)            (None, 12, 12, 128)       73856
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 12, 12, 128)       0
_________________________________________________________________
batch_normalization_7 (Batch (None, 12, 12, 128)       512
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 12, 12, 128)       147584
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 12, 12, 128)       0
_________________________________________________________________
batch_normalization_8 (Batch (None, 12, 12, 128)       512
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 12, 12, 128)       147584
_________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 12, 12, 128)       0
_________________________________________________________________
batch_normalization_9 (Batch (None, 12, 12, 128)       512
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 12, 12, 128)       147584
_________________________________________________________________
leaky_re_lu_10 (LeakyReLU)   (None, 12, 12, 128)       0
_________________________________________________________________
batch_normalization_10 (Batc (None, 12, 12, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 6, 6, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               2359808
_________________________________________________________________
batch_normalization_11 (Batc (None, 512)               2048
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
batch_normalization_12 (Batc (None, 256)               1024
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               32896
_________________________________________________________________
batch_normalization_13 (Batc (None, 128)               512
_________________________________________________________________
dropout_6 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
batch_normalization_14 (Batc (None, 64)                256
_________________________________________________________________
dropout_7 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_5 (Dense)              (None, 7)                 455
=================================================================
Total params: 3,167,559
Trainable params: 3,164,039
Non-trainable params: 3,520
_________________________________________________________________
```

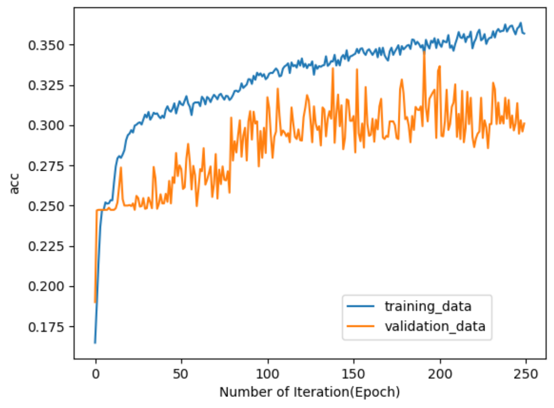


總共 10 層 conv_2d 和 5 層 dense，每層的 conv_2d 做 Leaky_Relu，dense 最後一層做 softmax，其餘做 relu，訓練過程共 250 個 epochs，batch size=128，每次跑 batch 都用 image generator 對 image 做一些處理來增加 data 數，並使用 callback 將過程中出現 validation accuracy 新高的 model 存下來，目前出現 validation 最好的是在第 233 個 epoch，validation accuracy 為 0.70533，kaggle public 為 0.68292，而 kaggle public 最高的是第 236 個 epochs 時的 model，validation accuracy 為 0.70433，kaggle public 為 0.69016(因為第一次 training，epochs 設定 1000 次太久了，後來改成 250 再 train 了一次，所以才有 236 的 validation 明明比 233 的低卻有第 236 次 model 的情況)

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: 呂承洋、陳柏堯、邵志宇)

答：

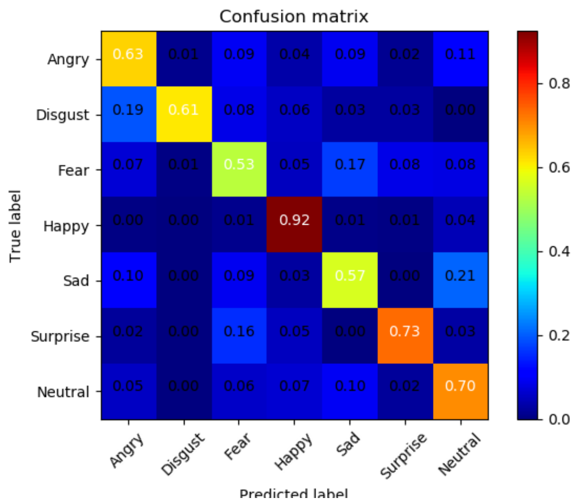
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 1024)              2360320
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               262656
_________________________________________________________________
batch_normalization_1 (Batch (None, 512)               2048
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               32896
_________________________________________________________________
batch_normalization_2 (Batch (None, 128)               512
_________________________________________________________________
dropout_6 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_7 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_7 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_8 (Dense)              (None, 32)                2080
_________________________________________________________________
batch_normalization_3 (Batch (None, 32)                128
_________________________________________________________________
dropout_8 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_9 (Dense)              (None, 7)                 231
=================================================================
Total params: 3,587,911
Trainable params: 3,586,567
Non-trainable params: 1,344
_________________________________________________________________
```



用了 9 層的 dense 建構出跟 CNN parameters 數量差不多的 DNN，每層做 relu 然後最後一層 softmax，由圖可以看出訓練結果非常不理想，完全沒有 train 成功，在圖像辨識上使用 CNN 先處理過後再丟進 dense 會比直接把所有 input 餵進去 DNN 效果好上非常多。

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]
(Collaborators: 呂承洋、陳柏堯、邵志宇)

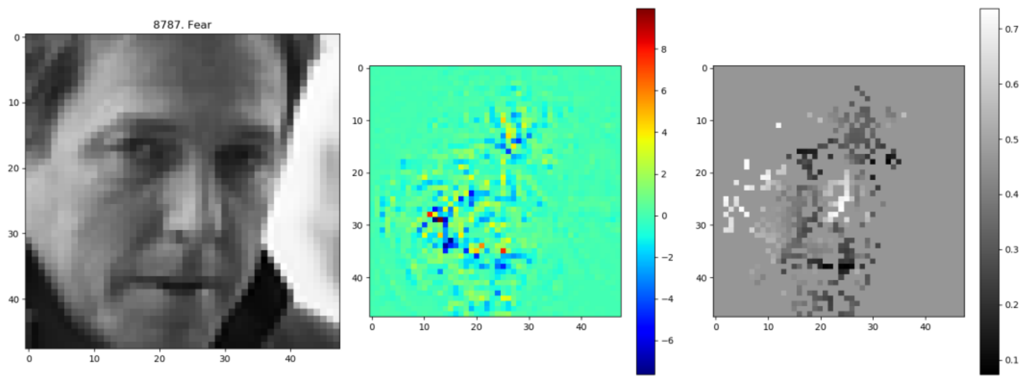
答：



True label \ Predicted label	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	0.63	0.01	0.09	0.04	0.09	0.02	0.11
Disgust	0.19	0.61	0.08	0.06	0.03	0.03	0.00
Fear	0.07	0.01	0.53	0.05	0.17	0.08	0.08
Happy	0.00	0.00	0.01	0.92	0.01	0.01	0.04
Sad	0.10	0.00	0.09	0.03	0.57	0.00	0.21
Surprise	0.02	0.00	0.16	0.05	0.00	0.73	0.03
Neutral	0.05	0.00	0.06	0.07	0.10	0.02	0.70

由圖可知，最容易預測錯的 class 是 Fear 和 Sad，其中 Fear 容易猜錯成 Sad，而 Sad 容易猜錯成 Neutral，Disgust 容易猜成 Angry

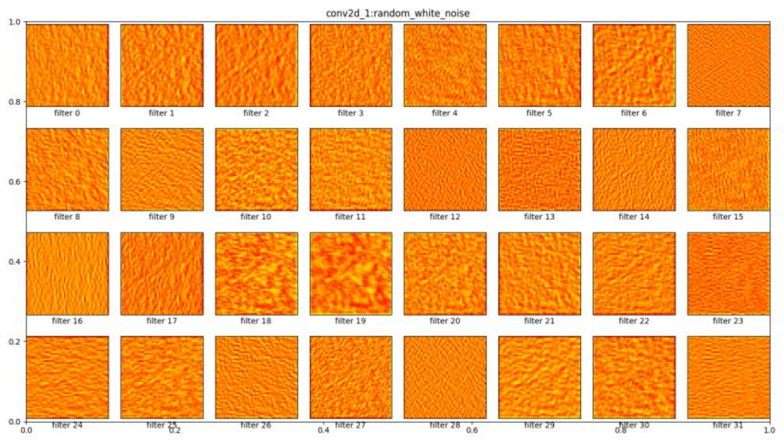
4. (1%) 從(1)(2)可以發現，使用 CNN 的確有些好處，試繪出其 saliency maps，觀察模型在做 classification 時，是 focus 在圖片的哪些部份？
(Collaborators: 呂承洋、陳柏堯、邵志宇)



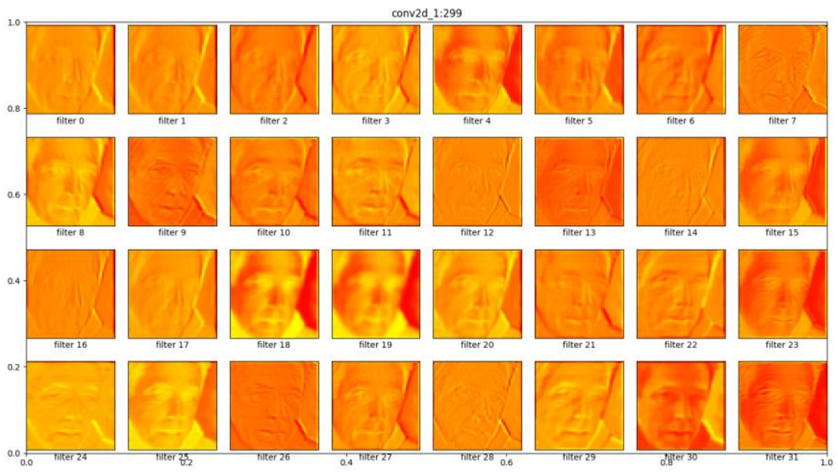
答：

可以看出是 focus 在眼睛、鼻子、嘴巴的部分，model 確實有掌握到臉部比較容易判斷表情的部分。

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: 呂承洋、陳柏堯、邵志宇)



答：



第一層的 filter 在 Fear 類的圖片顯示出的結果非常清晰，因此推測第一層的 filter 最容易被 Fear 類圖片 activate