collaborators: b04901096 蔡昕宇(未修課)、b04902039 羅際禎(上學期修課)、b04901060 黃文璁、b04901061 蔡忠紘

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：模型架構用助教給的 sample code 下去改，跑了 70 個 epoch。

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 64, 44, 44)        1664
_________________________________________________________________
batch_normalization_1 (Batch (None, 64, 44, 44)        176
_________________________________________________________________
zero_padding2d_1 (ZeroPaddin (None, 64, 48, 48)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 64, 23, 23)        0
_________________________________________________________________
zero_padding2d_2 (ZeroPaddin (None, 64, 25, 25)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 23, 23)        36928
_________________________________________________________________
batch_normalization_2 (Batch (None, 64, 23, 23)        92
_________________________________________________________________
zero_padding2d_3 (ZeroPaddin (None, 64, 25, 25)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 23, 23)        36928
_________________________________________________________________
batch_normalization_3 (Batch (None, 64, 23, 23)        92
_________________________________________________________________
average_pooling2d_1 (Average (None, 64, 11, 11)        0
_________________________________________________________________
zero_padding2d_4 (ZeroPaddin (None, 64, 13, 13)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 128, 12, 12)       32896
_________________________________________________________________
batch_normalization_4 (Batch (None, 128, 12, 12)       48
_________________________________________________________________
zero_padding2d_5 (ZeroPaddin (None, 128, 14, 14)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 128, 12, 12)       147584
_________________________________________________________________
batch_normalization_5 (Batch (None, 128, 12, 12)       48
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 128, 5, 5)         0
_________________________________________________________________
zero_padding2d_6 (ZeroPaddin (None, 128, 9, 9)         0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 128, 5, 5)         0
_________________________________________________________________
zero_padding2d_6 (ZeroPaddin (None, 128, 9, 9)         0
_________________________________________________________________
average_pooling2d_2 (Average (None, 128, 9, 9)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 256, 7, 7)         295168
_________________________________________________________________
batch_normalization_6 (Batch (None, 256, 7, 7)         28
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 256, 2, 2)         0
_________________________________________________________________
zero_padding2d_7 (ZeroPaddin (None, 256, 4, 4)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 2048)              8390656
_________________________________________________________________
batch_normalization_7 (Batch (None, 2048)              8192
_________________________________________________________________
dropout_1 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              2098176
_________________________________________________________________
batch_normalization_8 (Batch (None, 1024)              4096
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 7175
=================================================================
Total params: 11,059,947
Trainable params: 11,053,561
Non-trainable params: 6,386
```

epoch	10	20	30	40	50	60	70
acc	0.454	0.523	0.592	0.641	0.672	0.683	0.701

Kaggle 準確率：0.67456

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：同樣跑 70 個 epoch

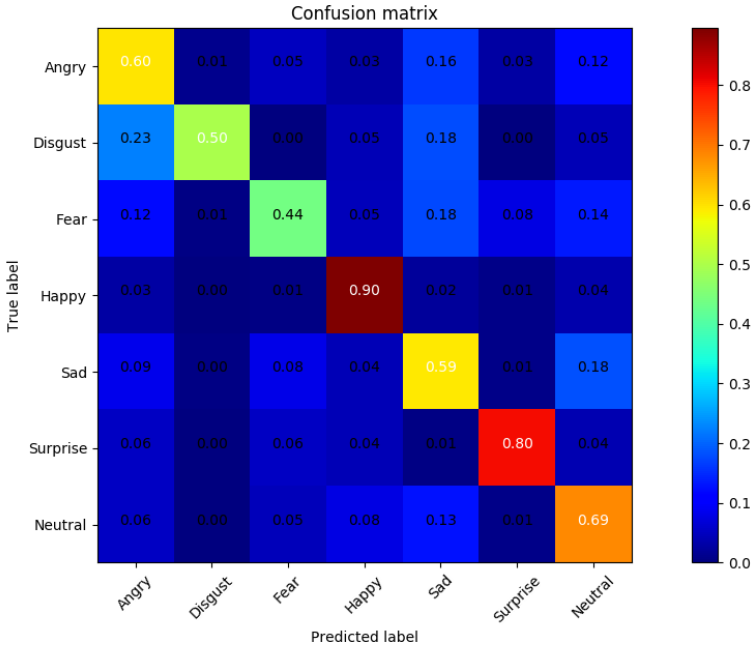
```
Layer (type)                 Output Shape              Param #
=================================================================
batch_normalization_1 (Batch (None, 48, 48, 1)         4
_________________________________________________________________
flatten_1 (Flatten)          (None, 2304)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 2304)              0
_________________________________________________________________
dense_1 (Dense)              (None, 824)               1899320
_________________________________________________________________
dropout_2 (Dropout)          (None, 824)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              844800
_________________________________________________________________
dropout_3 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_3 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dropout_4 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_4 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dropout_5 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_5 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dropout_6 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_6 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dropout_7 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_7 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dropout_8 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_8 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dropout_9 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_9 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dropout_10 (Dropout)         (None, 1024)              0
_________________________________________________________________
dense_10 (Dense)             (None, 1024)              1049600
_________________________________________________________________
dropout_11 (Dropout)         (None, 1024)              0
_________________________________________________________________
dense_11 (Dense)             (None, 7)                 7175
=================================================================
Total params: 11,148,099
Trainable params: 11,148,097
Non-trainable params: 2
```

epoch	10	20	30	40	50	60	70
acc	0.443	0.463	0.481	0.507	0.527	0.551	0.548

Kaggle 準確率：0.51011

在 DeepQ 平台上跑，DNN 比 CNN 快很多，只用了約 1/6 的時間，即使把平台上因使用者數量不同造成的影響去除，DNN 應該還是會比較快。不過 DNN 的準確率到了 0.55 左右就差不多到瓶頸了。

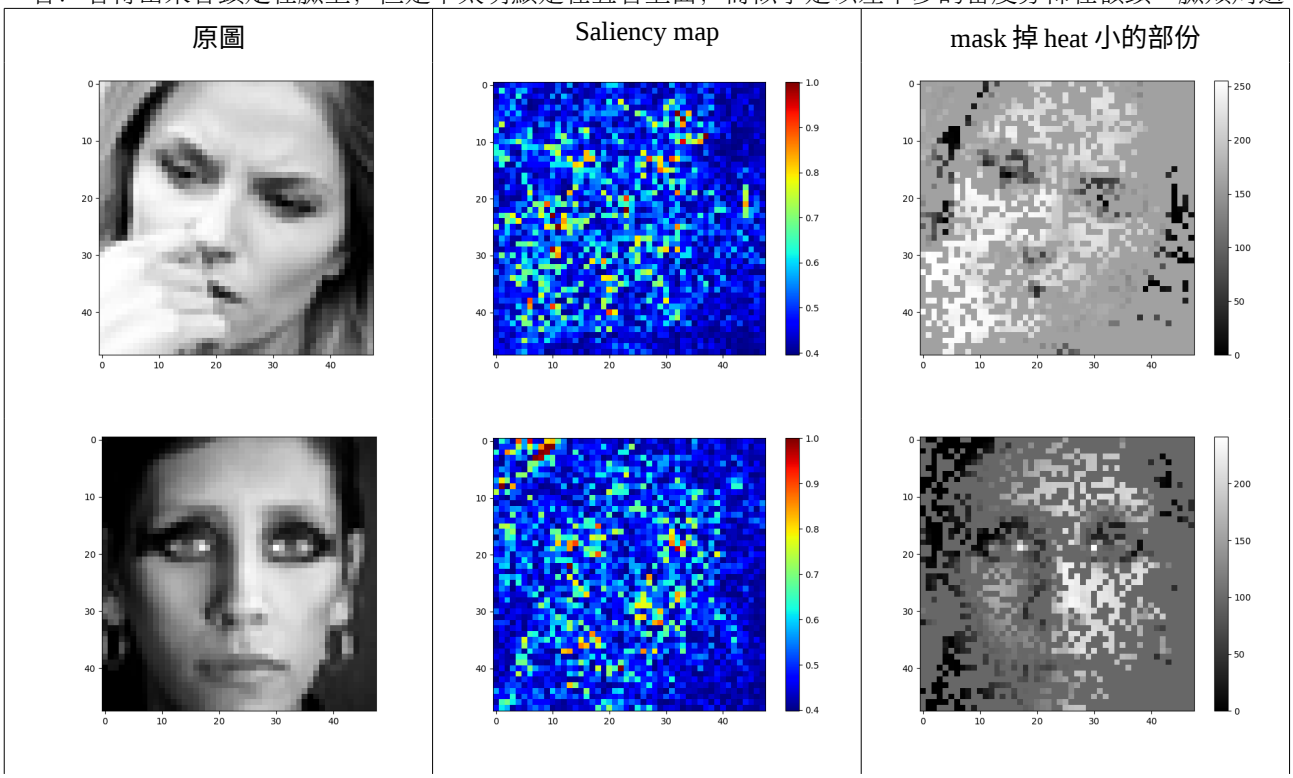
3. (1%) 觀察答錯的圖片中，哪些 class 彼此間容易用混？[繪出 confusion matrix 分析]

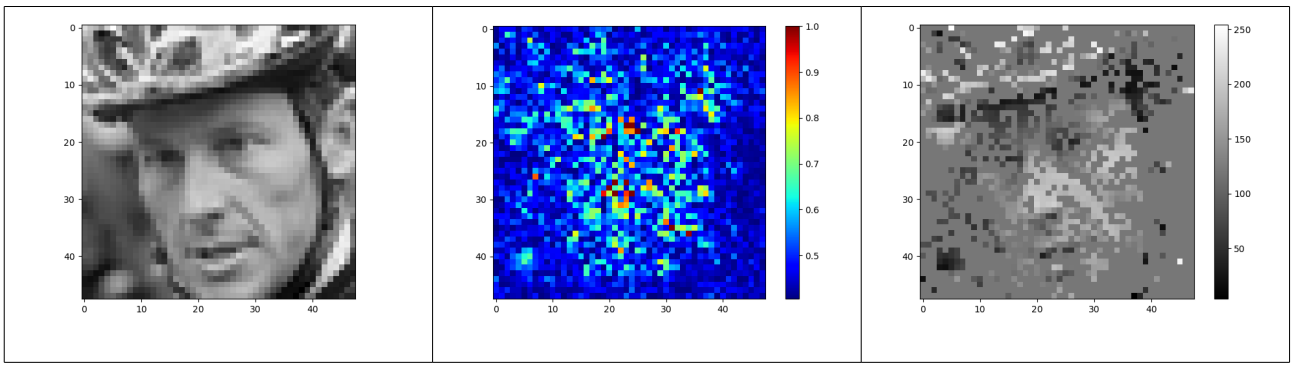


答：由 confusion matrix 可以看到不好分的是 Angry, Disgust, Fear, Sad, Neutral 四種情緒。彼此之間分錯的都是分成另外一類，較不容易與 Happy, Surprise 搞混。

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：看得出來會鎖定在臉上，但是不太明顯是在五官上面，而似乎是以差不多的密度分佈在額頭、臉頰周遭。





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：可以看出不同的 filter 就會對圖片的不同特性或是某個方向有比較明顯的感應。filter 對特定一張圖片(id699)的反應顯示在下圖，每個圖片的「筆觸」都符合上圖的 filter 看起來的特性。