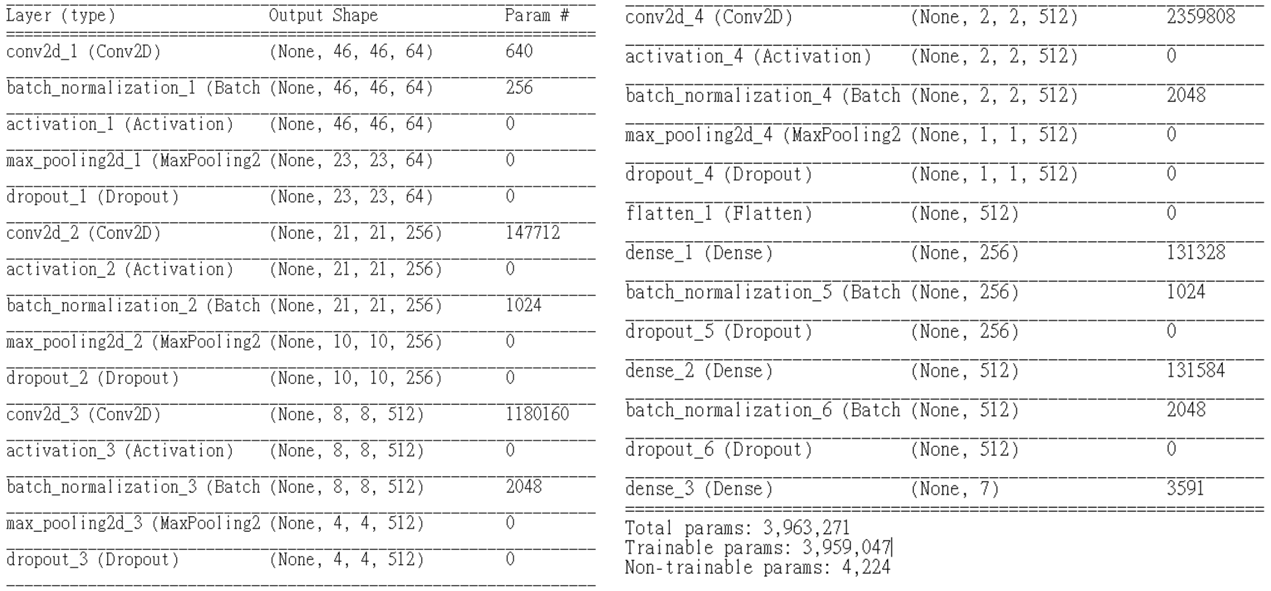
學號：R06725005 系級： 資管碩一 姓名：郝思喬

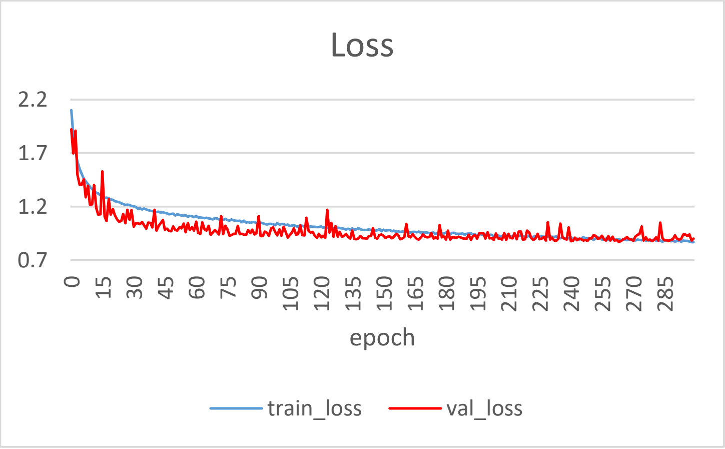
1. (1%) 請說明你實作的 CNN model，其模型架構、訓練過程和準確率為何？
答：
4 層 Convolution layers，每層都使用 BatchNormalization、MaxPooling2D、Dropout(0.3)、並且使用 selu 當 activation，filter 數量分別為 64,256,212,512。外加 2 層 Dense layers，每層使用 BatchNormalization、Dropout(0.3)、並且使用 selu 當 activation，神經元數量分別為 256,512。最後再加一層 Dense 使用 softmax 當 output。訓練過程有使用 ImageDataGenerator 前處理圖片產生更多資料，batch_size 設為 64，並且跑 300 個 epochs，過程中儲存 loss 最低的 model。最終準確率為 validation 以及 Kaggle 都接近 0.68。
架構圖：

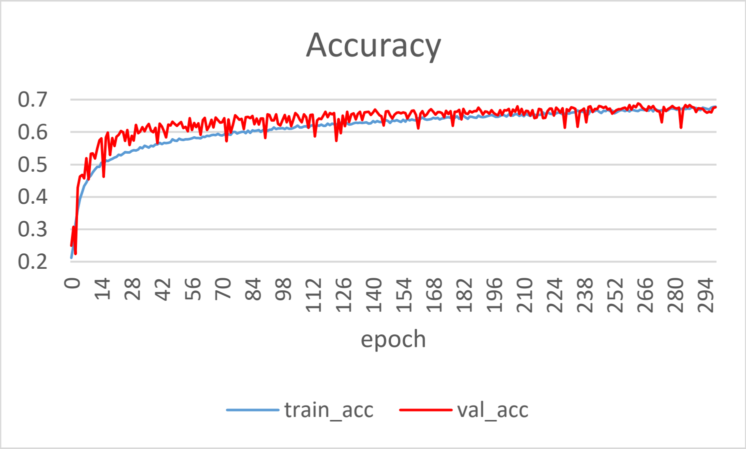


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 46, 46, 64)        640
_________________________________________________________________
batch_normalization_1 (Batch (None, 46, 46, 64)        256
_________________________________________________________________
activation_1 (Activation)    (None, 46, 46, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 23, 23, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 23, 23, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 21, 21, 256)       147712
_________________________________________________________________
activation_2 (Activation)    (None, 21, 21, 256)       0
_________________________________________________________________
batch_normalization_2 (Batch (None, 21, 21, 256)       1024
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 256)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 10, 10, 256)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 512)         1180160
_________________________________________________________________
activation_3 (Activation)    (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
_________________________________________________________________
```

```
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 2, 2, 512)         2359808
_________________________________________________________________
activation_4 (Activation)    (None, 2, 2, 512)         0
_________________________________________________________________
batch_normalization_4 (Batch (None, 2, 2, 512)         2048
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 1, 1, 512)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 1, 1, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
batch_normalization_5 (Batch (None, 256)               1024
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               131584
_________________________________________________________________
batch_normalization_6 (Batch (None, 512)               2048
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 3591
=================================================================
Total params: 3,963,271
Trainable params: 3,959,047
Non-trainable params: 4,224
```

Loss 以及 Accuracy 變化：





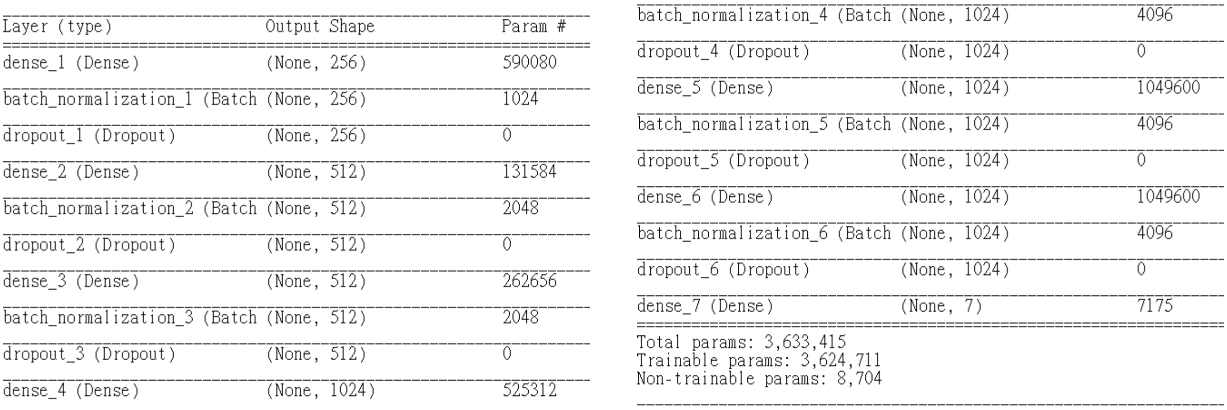
2. (1%) 承上題，請用與上述 CNN 接近的參數量，實做簡單的 DNN model。其模型架構、訓練過程和準確率為何？試與上題結果做比較，並說明你觀察到了什麼？

答：

為 6 層 Dense layers，每層都加 BatchNormalization、Dropout(0.3)，使用 selu 當 activation，參數量分別為 256,512,512,1024,1024,1024，最後再加一層 Dense 使用 softmax 當 output。訓練過程與 CNN 相同 batch_size=64 跑 300 個 epochs，紀錄 loss 最低的 model，準確率為 0.43。

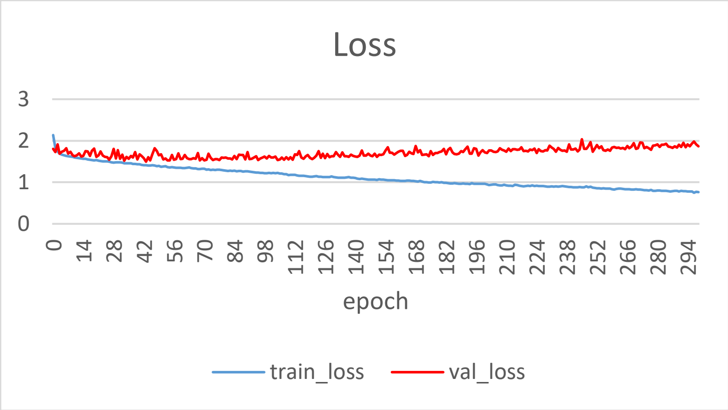
DNN 用相同的參數量得到的準確率遠遠比 CNN 訓練出來的結果還要差很多。

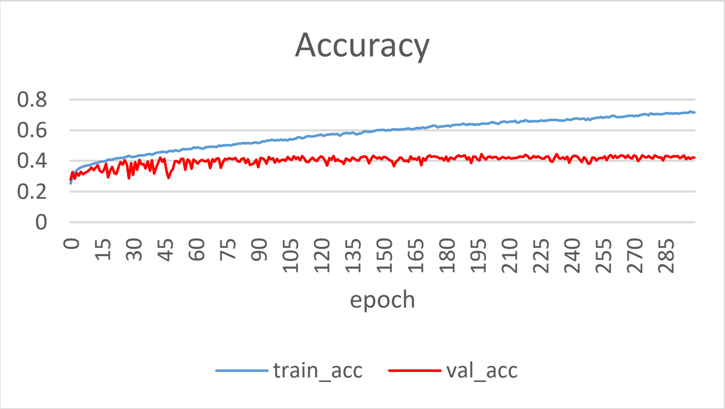
架構圖：



```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 256)               590080
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               131584
_________________________________________________________________
batch_normalization_2 (Batch (None, 512)               2048
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               262656
_________________________________________________________________
batch_normalization_3 (Batch (None, 512)               2048
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 1024)              525312
_________________________________________________________________
batch_normalization_4 (Batch (None, 1024)              4096
_________________________________________________________________
dropout_4 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_5 (Dense)              (None, 1024)              1049600
_________________________________________________________________
batch_normalization_5 (Batch (None, 1024)              4096
_________________________________________________________________
dropout_5 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_6 (Dense)              (None, 1024)              1049600
_________________________________________________________________
batch_normalization_6 (Batch (None, 1024)              4096
_________________________________________________________________
dropout_6 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_7 (Dense)              (None, 7)                 7175
=================================================================
Total params: 3,633,415
Trainable params: 3,624,711
Non-trainable params: 8,704
_________________________________________________________________
```

Loss 以及 Accuracy 變化：



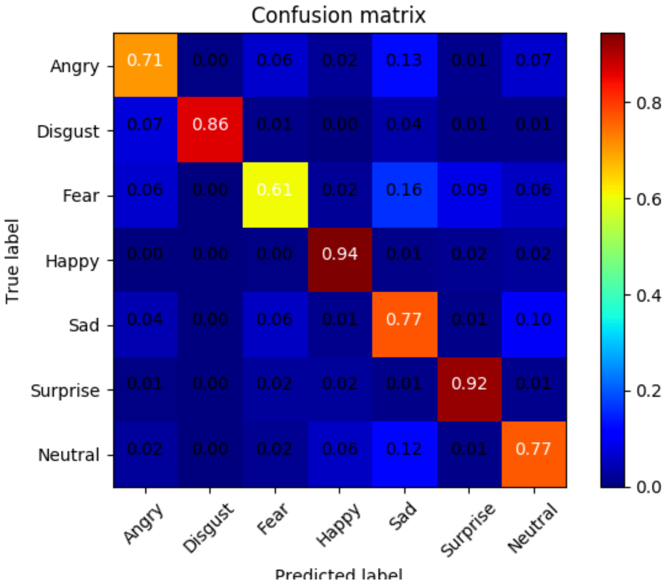


3. (1%) 觀察答錯的圖片中，哪些 class 彼此間容易用混？[繪出 confusion matrix 分析]

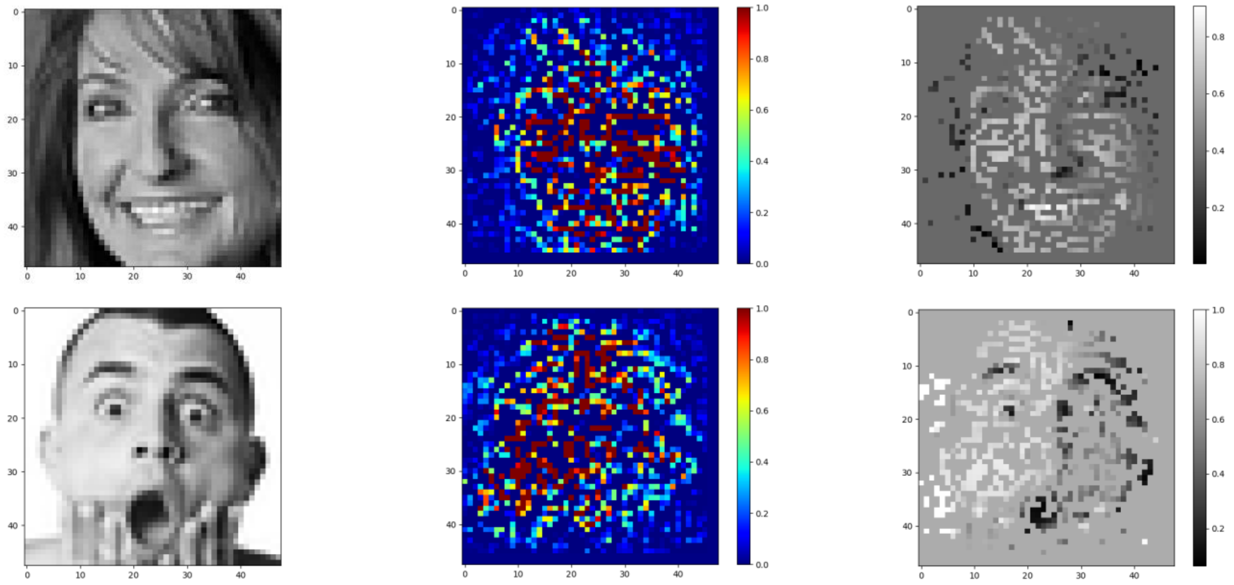
答：

各個類別的準確率皆有到達 0.6。

Fear、Angry、Neutral 常會與 Sad 類別用混，機率皆高於 0.1



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：
   舉兩張圖為例



由這兩張圖可知 model 是處要 focus 在眼睛以及嘴巴輪廓的部分。

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。
   答：
   以第一層 convolution 為例，這 64 個 filters 最容易被 happy 這類別 active，下圖顯示出這張 happy 的輪廓接清晰可見。而前面的 confusion matrix 也顯示此 CNN 架構 happy 類別的準確率最高。