學號：r04921094 系級： 電機碩二 姓名：葉孟元

注：特別告知助教，由於使用 test.csv 做 self-train 所以 hw3_train.sh 沒有吃 testing data，所以無法做出來。如果助教執行發生問題請聯繫我，我一定給出合理解釋和代碼。

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：

由於 pydot 版本的關係，在 keras 中出現衝突，可能由於自己設定的關係，就算裝了 pydot 1.1.0 還是有些問題，所以只能用這種方式，請助教見諒。

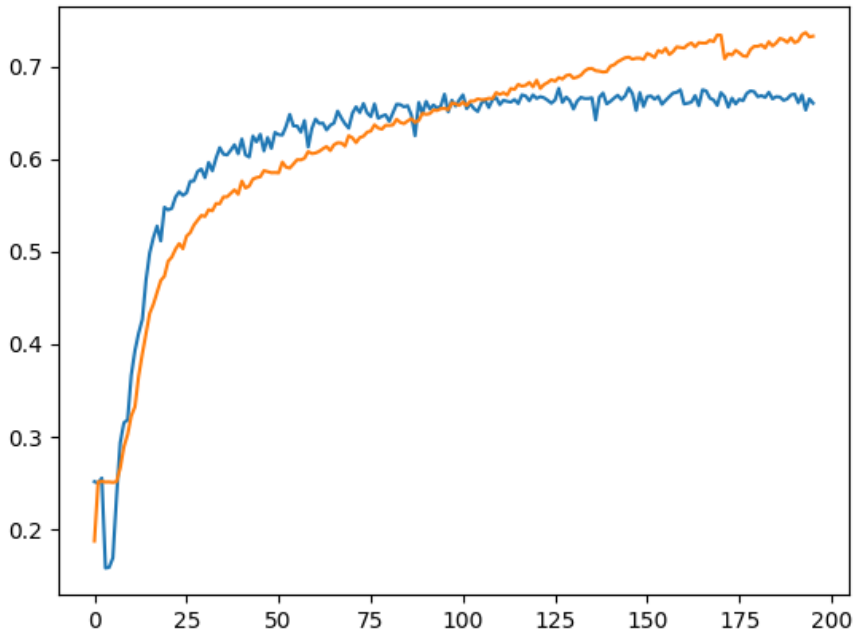
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 48, 48, 64)        640
_________________________________________________________________
activation_1 (Activation)    (None, 48, 48, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 48, 48, 64)        36928
_________________________________________________________________
activation_2 (Activation)    (None, 48, 48, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 24, 24, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 128)       73856
_________________________________________________________________
activation_3 (Activation)    (None, 24, 24, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 128)       147584
_________________________________________________________________
activation_4 (Activation)    (None, 24, 24, 128)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 12, 12, 256)       295168
_________________________________________________________________
```

activation_5 (Activation)	(None, 12, 12, 256)	0
conv2d_6 (Conv2D)	(None, 12, 12, 256)	590080
activation_6 (Activation)	(None, 12, 12, 256)	0
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 256)	0
dropout_3 (Dropout)	(None, 6, 6, 256)	0
conv2d_7 (Conv2D)	(None, 6, 6, 512)	1180160
activation_7 (Activation)	(None, 6, 6, 512)	0
conv2d_8 (Conv2D)	(None, 6, 6, 512)	2359808
activation_8 (Activation)	(None, 6, 6, 512)	0
conv2d_9 (Conv2D)	(None, 6, 6, 512)	2359808
activation_9 (Activation)	(None, 6, 6, 512)	0
max_pooling2d_4 (MaxPooling2	(None, 3, 3, 512)	0
dropout_4 (Dropout)	(None, 3, 3, 512)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_1 (Dense)	(None, 64)	294976
activation_10 (Activation)	(None, 64)	0
dense_2 (Dense)	(None, 64)	4160
activation_11 (Activation)	(None, 64)	0
dense_3 (Dense)	(None, 7)	455

```
activation_12 (Activation)   (None, 7)                 0
=================================================================
Total params: 7,343,623
```

每一層 dropout = 0.5， 在最後 DNN 部分使用了 l2 = 0.01 的 regularizer。

藍色是 validation score，黃色是 training score



在自己的 2700 筆 validation set 的成績是 0.676，.上最後在 kaggle 的 public 上是 0.66007。在訓練中使用 adam batch_size = 512。

在訓練階段使用的是 fit_generator 這個方法，因為希望可以用 generator 讓 picture 可以多一點 noise，除了 flip 以外還使用了 zome in， rotate。結果這樣也從某種程度上預防了 overfitting。

黃色的 training score 在 170 次 epoch 時發生下降，那是因為 self-train 的緣故，在 Bonus 會詳細介紹。

其實，自己的模型真的有點太大，可是這也是不斷 fine tune 的結果。一開始使用的是比較小的模型可是發現讓模型越來越大加深並且加 dropout 的方式會讓模型更好，於是就在這樣的方法下促使模型越來越大。同學有和我討論介紹我 kaggle 上 68%的模型，基本上會比我這個模型少一半的參數。我相信本應該是這個樣子才是對的。

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：

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 48, 48, 55)        110
_________________________________________________________________
flatten_1 (Flatten)          (None, 126720)            0
_________________________________________________________________
dense_2 (Dense)              (None, 55)                6969655
_________________________________________________________________
activation_1 (Activation)    (None, 55)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 55)                0
_________________________________________________________________
dense_3 (Dense)              (None, 55)                3080
_________________________________________________________________
activation_2 (Activation)    (None, 55)                0
_________________________________________________________________
dropout_2 (Dropout)          (None, 55)                0
_________________________________________________________________
dense_4 (Dense)              (None, 55)                3080
_________________________________________________________________
activation_3 (Activation)    (None, 55)                0
_________________________________________________________________
dropout_3 (Dropout)          (None, 55)                0
_________________________________________________________________
dense_5 (Dense)              (None, 55)                3080
_________________________________________________________________
activation_4 (Activation)    (None, 55)                0
_________________________________________________________________
dropout_4 (Dropout)          (None, 55)                0
_________________________________________________________________
dense_6 (Dense)              (None, 55)                3080
_________________________________________________________________
```

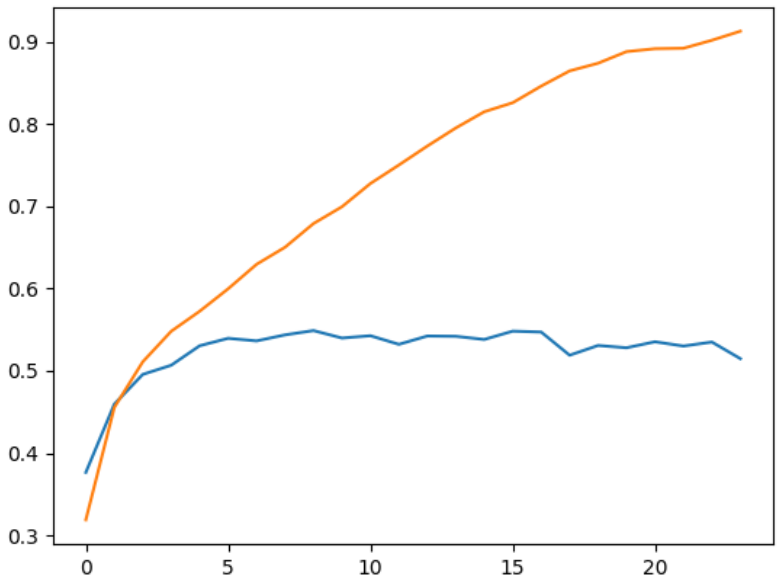
```
activation_5 (Activation)    (None, 55)                0
_________________________________________________________________
dropout_5 (Dropout)          (None, 55)                0
_________________________________________________________________
dense_7 (Dense)              (None, 7)                 392
_________________________________________________________________
activation_6 (Activation)    (None, 7)                 0
=================================================================
Total params: 6,982,477
```

dropout = 0.4 regularizer l2 = 0.001

首先對於第一層有限制 48*48*n，n 代表 neural 的數目，在經過 flatten 之後就已經有特別多的 output，這就直接導致了之後的參數量很大。沒有像之前採用 fit_generator 原因是如果採用同樣的設定根本 train 不下去，因為 generator 產生的 noise 太大了，其實 DNN 沒有學到 general 的解決方法。

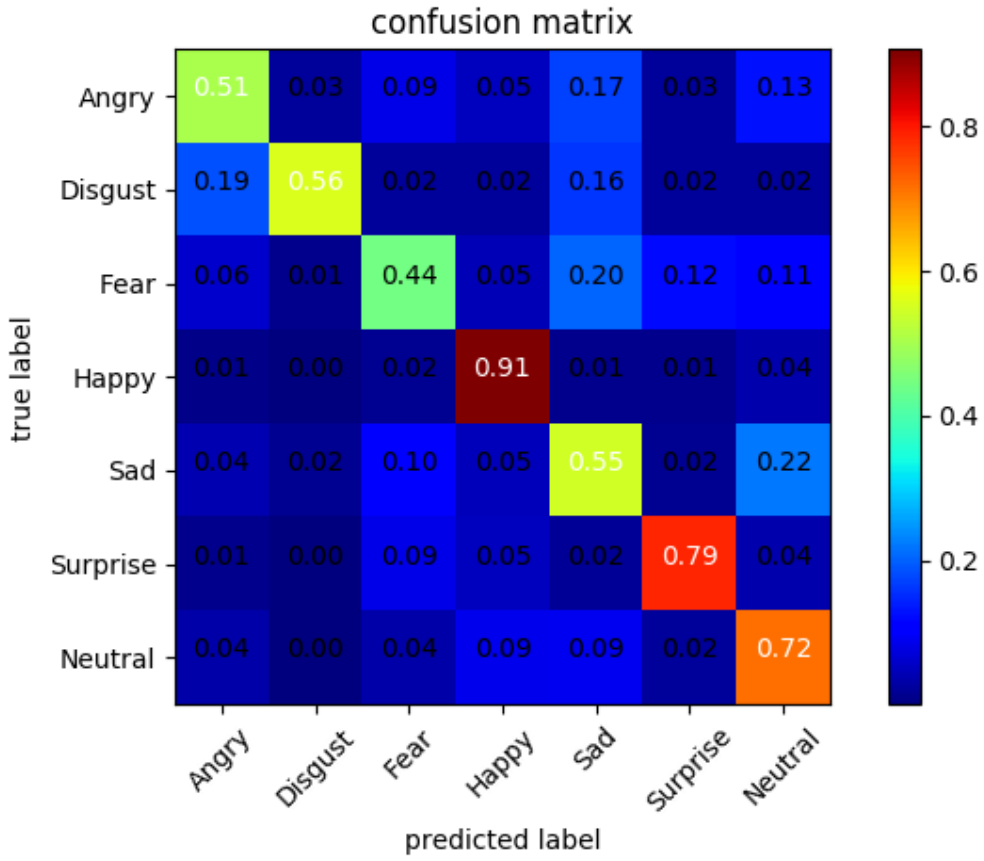
可以看到在 DNN 中很容易就發生 overfitting 我覺得就基本上是把圖片背下來而已（當然，不排除我沒有對 model 再進行 fine tune）但是，在 validation 上的表現十分不理想，最好的只有 53%而已。

藍色代表 validation score，黃色代表 training score



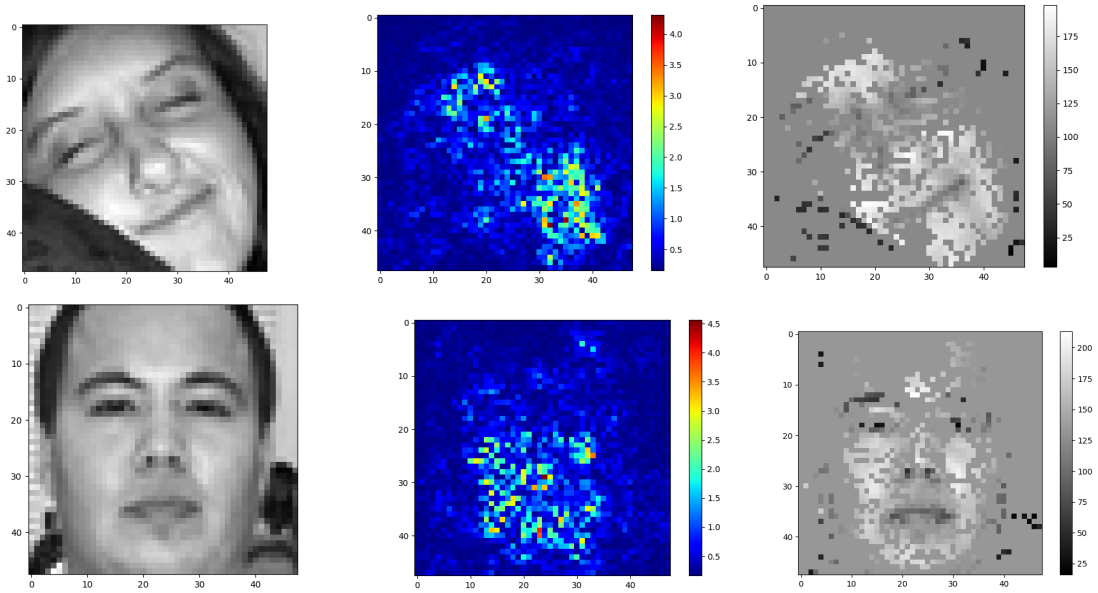
3. (1%) 觀察答錯的圖片中，哪些 class 彼此間容易用混？[繪出 confusion matrix 分析]

答：



fear 是最容易被弄錯的，並且會被認為是 sad，其實我直接打開看了之後發現者兩種類圖片對於我來說也是有點難區分。class 2 Disgust 是容易被混淆的，我認為其中一個很重要的原因是在 training data 中 class 1 只有 436 筆資料。這從某種程度上導致了這個問題的產生。

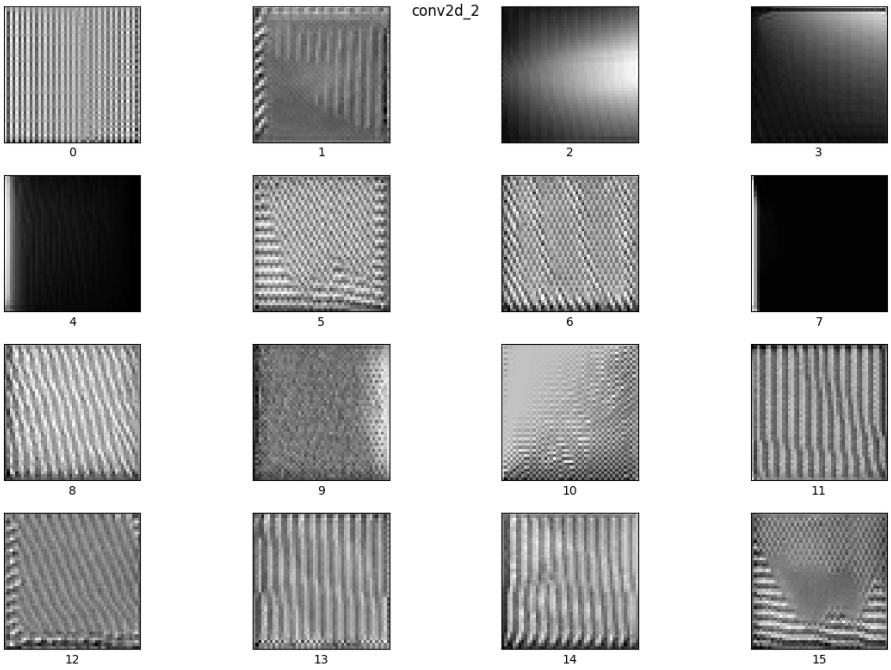
4. (1%) 從(1)(2)可以發現，使用 CNN 的確有些好處，試繪出其 saliency maps，觀察模型在做 classification 時，是 focus 在圖片的哪些部份？
答：



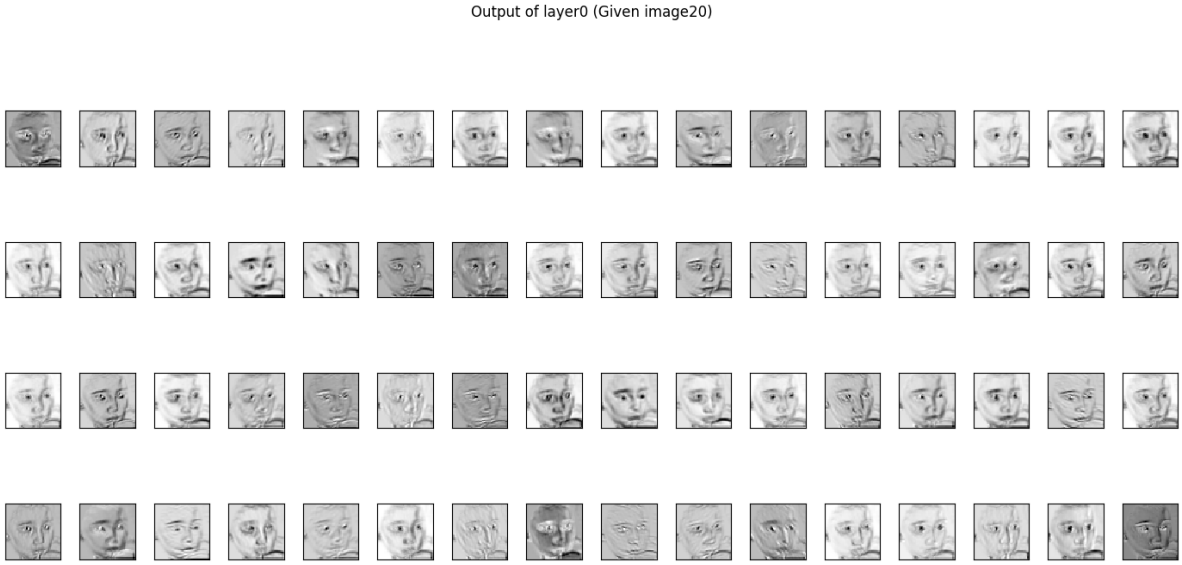
範例一，為了表示模型不只是會看正臉，有點歪斜的臉至少還是會對面部有很多的偵測，當然，沒有像助教的範例中那樣明確的看嘴巴和眼睛這可能是自己的模型並不夠好的緣故。

範例二，這個圖片就很集中在嘴巴的部分，說明在看中立的圖片的時候模型在意的是嘴巴的部分。也是有些許主角範例的感覺，當然沒有很在意到眼睛這一重點。

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。
答：



如自己在問題 1 的介紹，其實在 conv2d_2 有 64 個 filter，但是由於 GPU memory 不夠的問題，導致沒辦法一次全部畫出來，這裡主要選擇前 16 個來表現 filter 的行為。基本上在白噪聲上可以看到有明顯的紋理。



在下一圖中基本上眉毛，眼睛，嘴巴都有一些凸顯出來的情況。

[Bonus] (1%) 從 training data 中移除部份 label，實做 semi-supervised learning

自己實作的是 self-train，用的 data 是 test data。是在使用 early stop 之後，用自己 load 出來的最好的 model 去預測 test data。當預測的機率超過 0.87，就將這份資料夾到 training data 中，並確定 label。在實作先將 training data 存成圖片放在資料夾中，並且利用 fit_generator 中的 flow_from_directory 來從 directory 中取出資料。如果在 self-train 階段某些資料符合加入 training data 的需求時，就將資料存入。在這個做法下，會獲得 30000+筆資料，並且讓預測的準確度提高 2%。

[Bonus] (1%) 在 Problem 5 中，提供了 3 個 hint，可以嘗試實作及觀察 (但也可以不限於 hint 所提到的方向，也可以自己去研究更多關於 CNN 細節的資料)，並說明你做了些什麼？[完成 1 個: +0.4%, 完成 2 個: +0.7%, 完成 3 個: +1%]