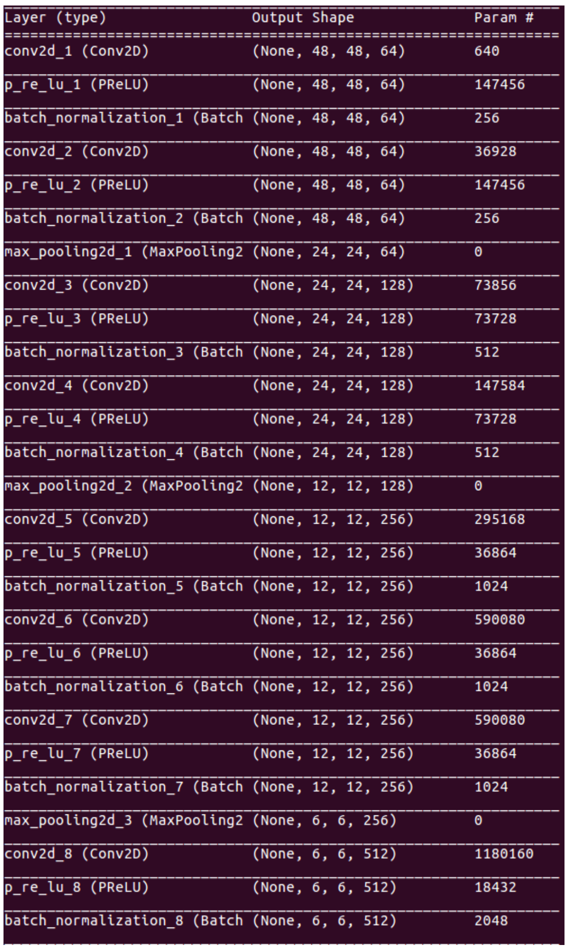
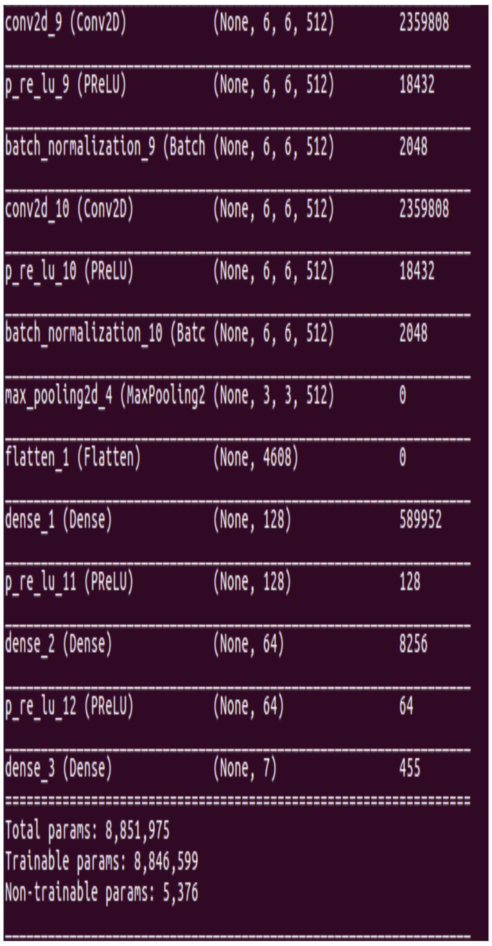
學號：r06921095 系級： 電機碩一 姓名：吳睿哲

1. (1%) 請說明你實作的 CNN model，其模型架構、訓練參數和準確率為何？
(Collaborators:)
答:

我總共實作了 3 種 model，以下為第一種



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 48, 48, 64)        640
_________________________________________________________________
p_re_lu_1 (PReLU)            (None, 48, 48, 64)        147456
_________________________________________________________________
batch_normalization_1 (Batch (None, 48, 48, 64)        256
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 48, 48, 64)        36928
_________________________________________________________________
p_re_lu_2 (PReLU)            (None, 48, 48, 64)        147456
_________________________________________________________________
batch_normalization_2 (Batch (None, 48, 48, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 128)       73856
_________________________________________________________________
p_re_lu_3 (PReLU)            (None, 24, 24, 128)       73728
_________________________________________________________________
batch_normalization_3 (Batch (None, 24, 24, 128)       512
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 128)       147584
_________________________________________________________________
p_re_lu_4 (PReLU)            (None, 24, 24, 128)       73728
_________________________________________________________________
batch_normalization_4 (Batch (None, 24, 24, 128)       512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 12, 12, 256)       295168
_________________________________________________________________
p_re_lu_5 (PReLU)            (None, 12, 12, 256)       36864
_________________________________________________________________
batch_normalization_5 (Batch (None, 12, 12, 256)       1024
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 12, 12, 256)       590080
_________________________________________________________________
p_re_lu_6 (PReLU)            (None, 12, 12, 256)       36864
_________________________________________________________________
batch_normalization_6 (Batch (None, 12, 12, 256)       1024
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 12, 12, 256)       590080
_________________________________________________________________
p_re_lu_7 (PReLU)            (None, 12, 12, 256)       36864
_________________________________________________________________
batch_normalization_7 (Batch (None, 12, 12, 256)       1024
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 256)         0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 6, 6, 512)         1180160
_________________________________________________________________
p_re_lu_8 (PReLU)            (None, 6, 6, 512)         18432
_________________________________________________________________
batch_normalization_8 (Batch (None, 6, 6, 512)         2048
```



```
conv2d_9 (Conv2D)            (None, 6, 6, 512)         2359808
_________________________________________________________________
p_re_lu_9 (PReLU)            (None, 6, 6, 512)         18432
_________________________________________________________________
batch_normalization_9 (Batch (None, 6, 6, 512)         2048
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 6, 6, 512)         2359808
_________________________________________________________________
p_re_lu_10 (PReLU)           (None, 6, 6, 512)         18432
_________________________________________________________________
batch_normalization_10 (Batc (None, 6, 6, 512)         2048
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 3, 3, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               589952
_________________________________________________________________
p_re_lu_11 (PReLU)           (None, 128)               128
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
p_re_lu_12 (PReLU)           (None, 64)                64
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 455
=================================================================
Total params: 8,851,975
Trainable params: 8,846,599
Non-trainable params: 5,376
_________________________________________________________________
```

Model 1

這個 model 我總共接了 12 層,其中包括了 10 層的 convolution 以及 max pooling,我總共跑了 200 個 epoch , batch size 設定為 30(原本開到 200,但是發現小一點的 batch size 收斂速度較快,較最後收斂結果的準確率較高),validation data 為資料最後的 1/10,訓練參數由上圖所示,這個 model 是我在 validation 得過最好一次的 model,可以達到 0.68793 的 accuracy,但是在 training error 嚴重的 over fitting,我總共跑了 200 個 epoch,最後 training accuracy 來到了 0.96713 但是 validation 只有 0.68 左右。

Model 2

第二個 model 跟第一個的主要是差在增加了 drop out 以及減少 convolution 的層數,由於第一個 model overfitting 的問題太嚴重,所以第二個 model 我減少了 2 層 convolution,且在最後 fully connected 的地方,加上 0.2 的 drop out rate,雖然 keras 的官方文件表示在做 batch normalization 之後可以達到等同於 drop out 的效果,但我仍然加了 0.2 的 drop out,這樣的作法卻是在一開始能很有效的抑制 training data overfitting 的問題,但是在做了 60 個 epoch 之後,還是會慢慢地產生 overfitting 的問題,且 validation accuracy 只有 0.67572。

Modle 3

第三個 model 跟第二個主要是差在 flatten 之後拔掉一層 128 的 neural network,且把 drop out rate 增加到 0.25,這麼做的目的是為了解決 model2 沒有改善 overfitting 的問題,實作之後,overfiting 的問題又比 model2 改善很多,training accuracy 上升的速度也降低很多,最後得到的 validation accuracy 來到 0.68626,但是還是沒有第一個 model 來的好。

2. (1%) 請嘗試 data normalization, data augmentation,說明實行方法並且說明對準確率有什麼樣的影響？

(Collaborators:)
答：

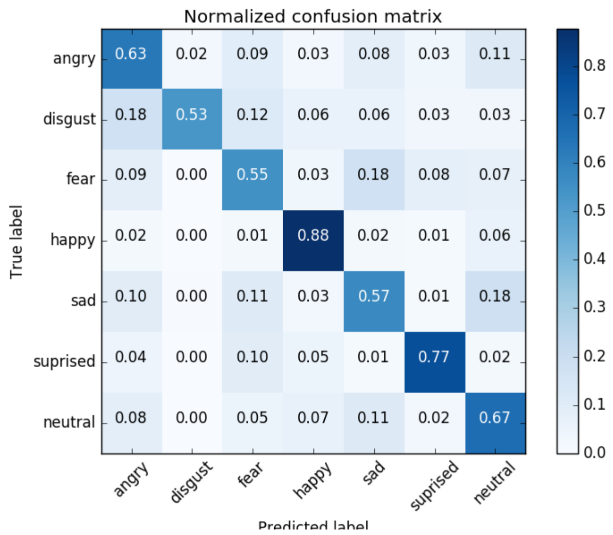
Data normalization: 我 data normalize 的作法是先減去一個照片的平均再除於標準差,這樣的作法再 validation accuracy 會差 2%左右的準確率,在同樣都跑 100 個 epoch 的情況下,validation accuracy 只有 0.64873,但是在 normalize 之後可以達到 0.67372

Data augmentation:data augmentation 的影響應該是所有裡面最大的,我使用了 keras 內部包好的套件,imagedatagenerator 來做操作,我主要更改了 raotation range 改為 10 度,width_shift_range 以及 height_shift_range 設定為 0.2,平行翻轉設定為 true,垂直翻轉設定為 false,這樣的作法可以讓準確率提升 4%左右, 在同樣都跑 100 個 epoch 的情況下,validation accuracy 只有 0.63173,但是在增加了 data 數量之後可以達到 0.67152

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:)

答：

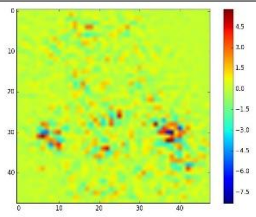
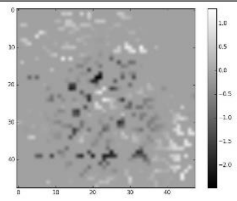


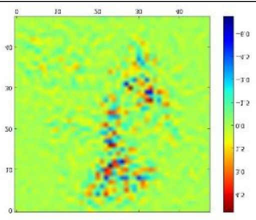
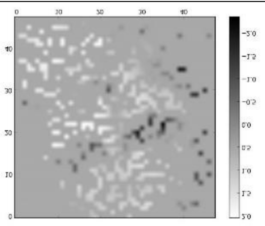
由此圖可發現,預測最精準的是 happy 這個表清,最容易被混用的是 sad 以及 fear 這兩個表情,這個結果也是蠻合理的,因為他們同樣都是屬於負面的情緒,被混用的機率可能比較大。

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:)

答：

Original	saliency maps	Mask heat
Label:0,Predict:0		

Label:2,Predict:4		

我用的圖片是在 training data 中的第二個人跟第三個人的照片，其中第二個人的 predictr 結果是正確的,他的表情是生氣的,我本來預測 saliency map 抓到的特徵應該要是眼睛跟嘴巴的部分,但是主要抓到的部分確是臉頰,這我覺得蠻奇怪的,我猜測可能的原因是因為生氣時臉部會變得猙獰,雖然用肉眼看起來他看起來沒有很猙獰,但是如果是用一個 pixel 一個 pixel 去看的話,卻是有機會讓類神經網路判斷為生氣的特徵。

第二個人判斷的結果是錯誤的,label 的表情為恐懼,但是預測的結果為難過,雖然預測的結果不對,但是我覺得他有蠻大的機會是有抓到特徵的,因為就算是我用肉眼去看,我眼分辨不出來這個人的情緒為何,且在 saliency map 上,類神經網路抓到的部分為眉心以及鼻子及嘴巴的部分,並歸類其為負面的情緒,雖然最後判斷錯誤,但是特徵是有掌握的。

5. (1%) 承(4) 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 與觀察 filter 的 output。

   (Collaborators:)

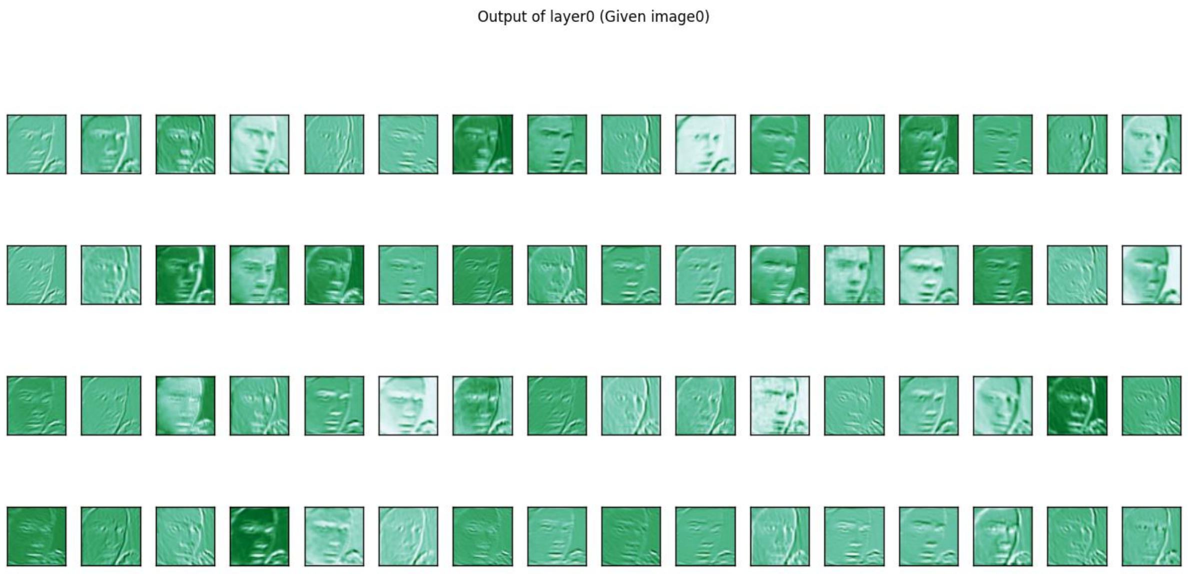
   答：

Original

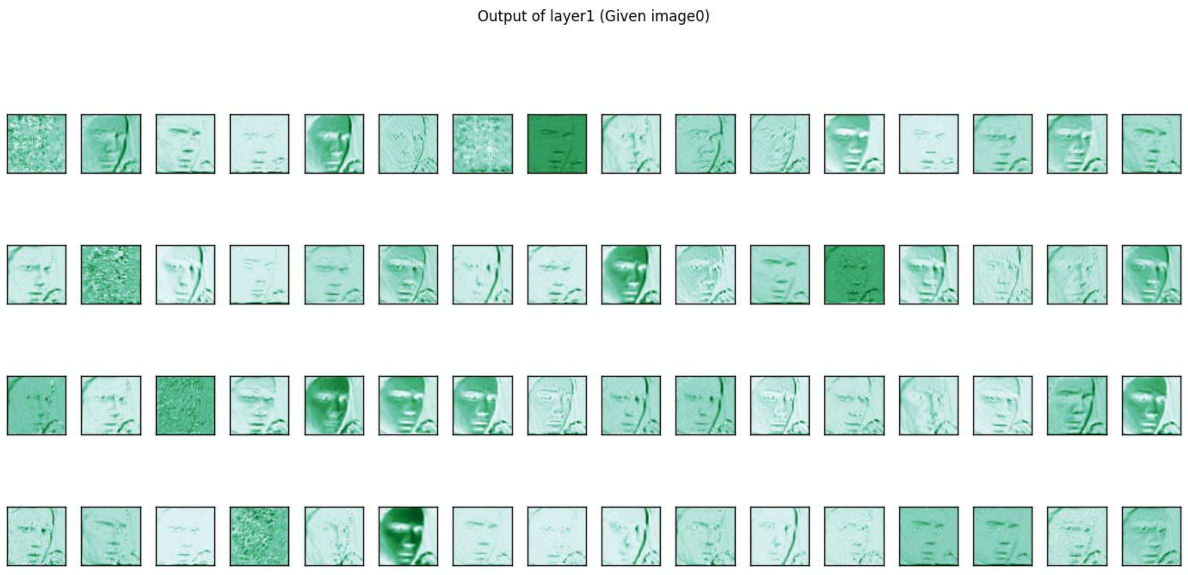
Training data 的第一個人在經過第二層 convolution 以及第一層 prelu 的 Layer output

第二層 covolution

由觀察到的圖案可以看到每個 filter 最被 activate 的樣子,以第 4 個 filter 為例,他可能就會抓有類似紋路的特徵,讓他被 activate, 第 6 個 filter 可能就沒有做什麼事。,



第一層 prelu



Training data 的第一個人在經過第二層 convolution 之後每一個 filter 最被 activate 的樣子,
由觀察到的圖案可以看到每個 filter 最被 activate 的樣子,以第 4 個 filter 為例,他可能就會抓有類似紋路的特徵,讓他被 activate, 第 6 個 filter 可能就沒有做什麼事。,

Filters of layer