學號：R06725035 系級： 資管碩一 姓名：陳廷易

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: 小老師教學相長)

答：

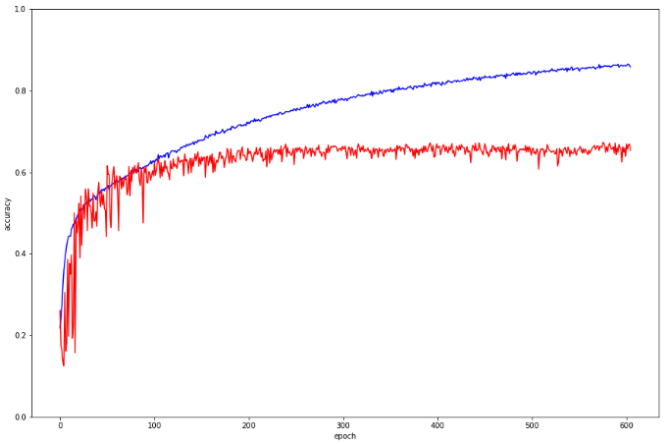
模型架構有四層 CNN block+兩層 fully-connected。每一層 initializer 皆採用 glorot_normal，且都設置 dropout 與 batch normalization。在 CNN block 中，皆採用 same padding 方式，除第一層使用 PReLU(alpha_initializer='zeros', alpha_regularizer=None, alpha_constraint=None, shared_axes=None)外，其餘皆使用 LeakyReLU(alpha=0.05)，經過 activation function 後會進行 MaxPooling，再經過 ZeroPadding。而前三層 CNN block 中，kernel size 皆為(4,4)，最後一層則為(3,3)。於 fully-connected 則是採用 softplus 的 activation function，並進行 L2 regularization。其他訓練參數如下：

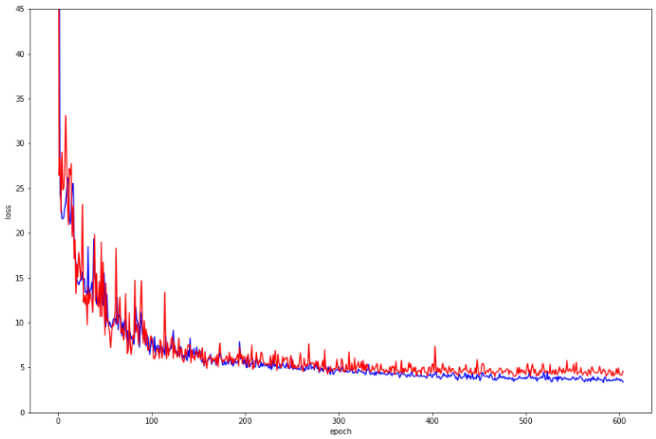
- Optimizer: Adam，decay=1e-6。Learning rate=0.015
- Batch size: 128。train_data : valid_data = 9 : 1
- Epoch: 1500，patience=100 monitor on validation accuracy
- Loss function: categorical_crossentropy

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 48, 48, 256)	4352
p_re_lu_1 (PReLU)	(None, 48, 48, 256)	589824
zero_padding2d_1 (ZeroPaddin	(None, 52, 52, 256)	0
batch_normalization_1 (Batch	(None, 52, 52, 256)	1024
max_pooling2d_1 (MaxPooling2	(None, 26, 26, 256)	0
zero_padding2d_2 (ZeroPaddin	(None, 28, 28, 256)	0
dropout_1 (Dropout)	(None, 28, 28, 256)	0
conv2d_2 (Conv2D)	(None, 28, 28, 512)	2097664
leaky_re_lu_1 (LeakyReLU)	(None, 28, 28, 512)	0
batch_normalization_2 (Batch	(None, 28, 28, 512)	2048
max_pooling2d_2 (MaxPooling2	(None, 14, 14, 512)	0
zero_padding2d_3 (ZeroPaddin	(None, 16, 16, 512)	0
dropout_2 (Dropout)	(None, 16, 16, 512)	0
conv2d_3 (Conv2D)	(None, 16, 16, 1024)	8389632
leaky_re_lu_2 (LeakyReLU)	(None, 16, 16, 1024)	0
batch_normalization_3 (Batch	(None, 16, 16, 1024)	4096
max_pooling2d_3 (MaxPooling2	(None, 8, 8, 1024)	0
zero_padding2d_4 (ZeroPaddin	(None, 10, 10, 1024)	0
dropout_3 (Dropout)	(None, 10, 10, 1024)	0
conv2d_4 (Conv2D)	(None, 10, 10, 2048)	18876416
leaky_re_lu_3 (LeakyReLU)	(None, 10, 10, 2048)	0
zero_padding2d_5 (ZeroPaddin	(None, 12, 12, 2048)	0
batch_normalization_4 (Batch	(None, 12, 12, 2048)	8192
max_pooling2d_4 (MaxPooling2	(None, 6, 6, 2048)	0
dropout_4 (Dropout)	(None, 6, 6, 2048)	0
flatten_1 (Flatten)	(None, 73728)	0
dense_1 (Dense)	(None, 1024)	75498496
batch_normalization_5 (Batch	(None, 1024)	4096
dropout_5 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 1024)	1049600
batch_normalization_6 (Batch	(None, 1024)	4096
activation_1 (Activation)	(None, 1024)	0
dropout_6 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 7)	7175
activation_2 (Activation)	(None, 7)	0

Total params: 106,536,711
Trainable params: 106,524,935
Non-trainable params: 11,776

最終於 public 的準確率為 0.70130，而 private 準確率為 0.69155。詳細訓練過程的 accuracy 與 loss 如下圖所示(紅色為 valid 藍色為 training)。





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: 助教講解)

答： 在 data normalization 部分，為將所有 train.csv 的值與 test.csv 的值全部進行運算，求出全部資料個點的平均值與標準差，將各筆數值減去平均值再除掉標準差。在 public 與 private 實作前的準確率為 0.65645 / 0.64697，實作後的準確率為 0.67205 / 0.66174。可看出在我的 model 中，透過資料標準化令資料間的跨度能統一，有助於使 model 學得更好。

在 data augmentation 部分，為使用 keras 所內建的 ImageDataGenerator 與 flow 方式來達成，旋轉的範圍為 20 度、水平與垂直位移皆 20%、允許隨機水平翻轉，詳細參數如右所示。

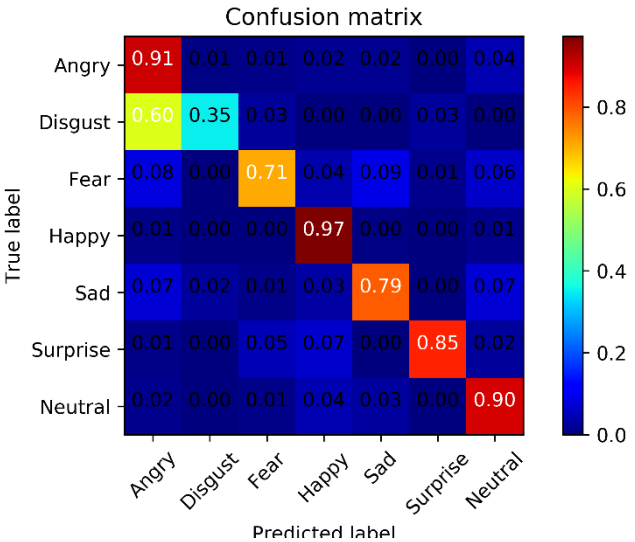
```
featurewise_center=True,
featurewise_std_normalization=True,
samplewise_center=True,
samplewise_std_normalization=True,
rotation_range=20,
width_shift_range=0.2,
height_shift_range=0.2,
horizontal_flip=True,
shear_range=0.001,
zoom_ranqge=0.001,
data_format='channels_last'
```

實作前的準確率為 0.67205 / 0.66174，實作後的準確率為 0.69378 / 0.68626。而在訓練過程中也發現，若未進行 augmentation，比較容易會有 overfitting 的情況出現。而若透過 data augmentation 增加一些圖片的 jitter、noise，使資料量增加，能讓 model 更為 robust。

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: 上網查資料，主要依據 google 與 github 搜尋到的內容進行修改)

答：

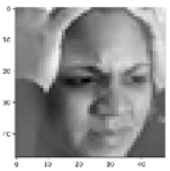
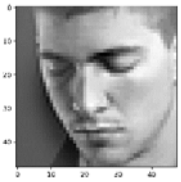


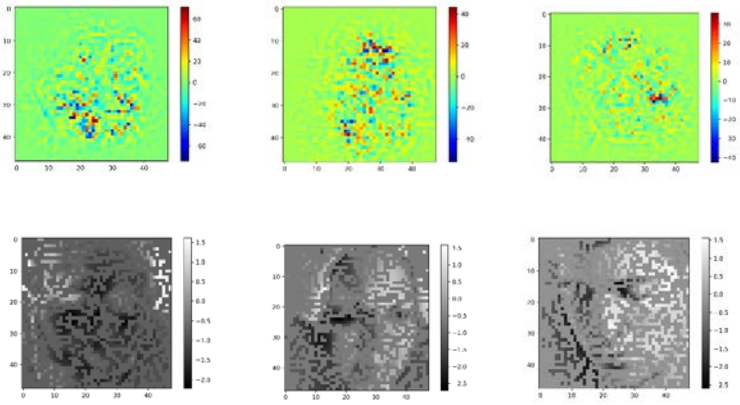
從 confusion matrix 中可看出，或許是因為 Angry 與 Disgust 表情較為相像的關係，對我的 model 而言較不易區分，而相對容易造成混淆。

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: 上網查資料，主要依據 google 與 github 搜尋到的內容進行修改)

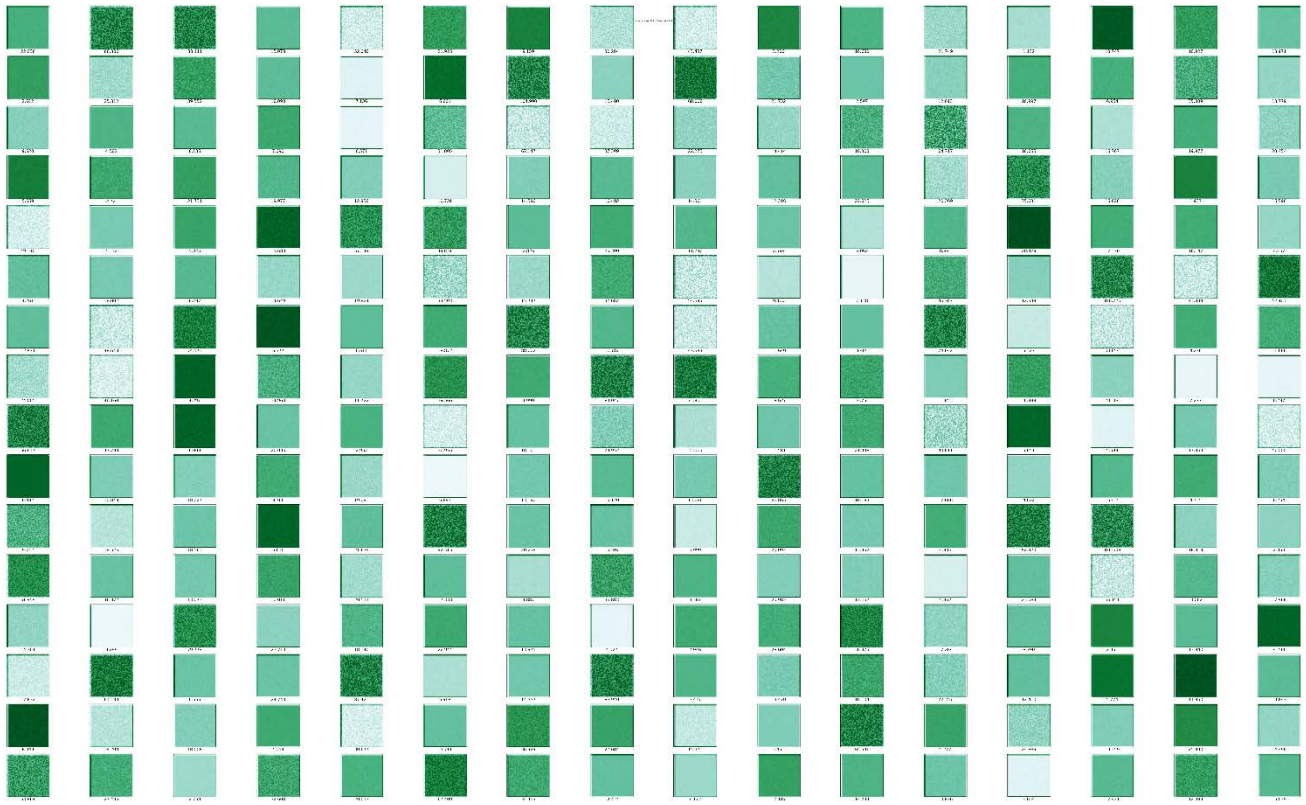
答：



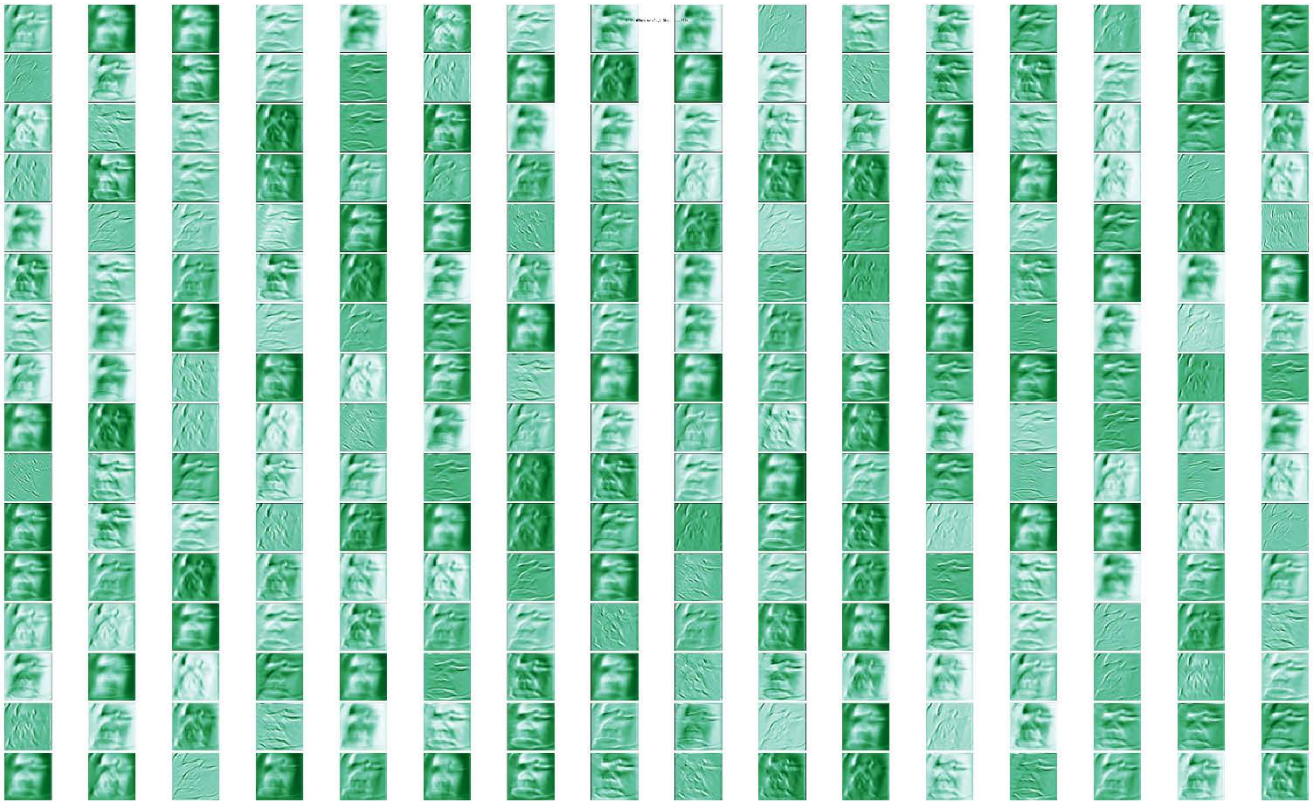
從 data 中隨機選出三張，分別畫出影響 CNN 分類比較大的 saliency map，並將對 CNN 影響不大的部分遮蓋掉。從 saliency map 圖中可發現我的 model 主要是 focus 在人臉的眼睛與嘴巴來分類情緒，這也與一般對表情的認知一致。此外，有些比較誇張表情的圖片，會有手遮住眼睛或嘴巴的情況，因此較容易令 model 判斷錯誤。

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: 上網查資料，主要依據 google 與 github 搜尋到的內容進行修改)
答：



上圖為第一層 Convolution filters 的 input(ascent epoch=150，256 filters)，下圖為其所對應的 output。

hw3-Image Sentiment Classification



利用 gradient ascent 方式，每次更新影像的 step=0.01*gradient，經過 150 個 epoch 後顯示出各 filter 最容易被哪種影像 activate，而各影像下排 的數字為其最後產生的 loss。因我的 model 之 filters 數量較多，礙於篇幅未將全部各層貼上來，但從第一層可看到所產生的影像多為類似霧狀的雜訊，需要至第二層才會較有明顯的紋理圖案，或許是因為第一層 CNN 主要是在偵測較為通俗一般化的特徵。在 output 方面，從圖中可看出有一點人臉的輪廓，此外眼睛與嘴巴也是著重的部分，因此可更加確定所訓練出來的模型，會從特徵中抽取嘴巴及眼睛部分進行分類。而在第二層 output 紋理則更為明顯、輪廓更深，越後面層之 output 越為複雜。