學號：B03505052 系級： 工海四 姓名：李吉昌

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: b03505040 林後維 / b03902130 楊書文)
答：說明模型架構、訓練參數和準確率。

filter 的大小在考慮 image size 後一律採用 3x3，並參考 VGGNet 後座一些參數上的微調，並在每一層加上 batch normalization 使其 loss 更容易下降到最低點，並在每一層做 drop out，至於每一層 drop out 的大小則是經過幾次實驗得來。

DNN 的部分架構是先用兩排決定最適合的參數量，在將兩排改成更深層的架構(因為上課有提到越 DEEP 效果越好)

```
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(48, 48, 1), activation='relu', padding='same'))
model.add(BatchNormalization(axis=-1))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization(axis=-1))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization(axis=-1))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.5))


model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization(axis=-1))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization(axis=-1))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization(axis=-1))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.5))


model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization(axis=-1))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization(axis=-1))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization(axis=-1))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.5))


model.add(Flatten())
model.add(Dense(units = 256, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(units = 128, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(units = 64, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(units = 7, activation='softmax'))
```

至於 optimizer 則選用兩種分別是 adam(defalut)以及 sgd (lr=0.005, decay=0.00001, momentum=0.9)

adam 得 0.70047 / sgd 得 0.69908

另外再將資料作處理(x = x – 127.5)再用兩種 optimizer 得出另外兩個 model

Kaggle 的 best acc: 0.71524 則是將四種 model 做 ensemble 得出的結果。

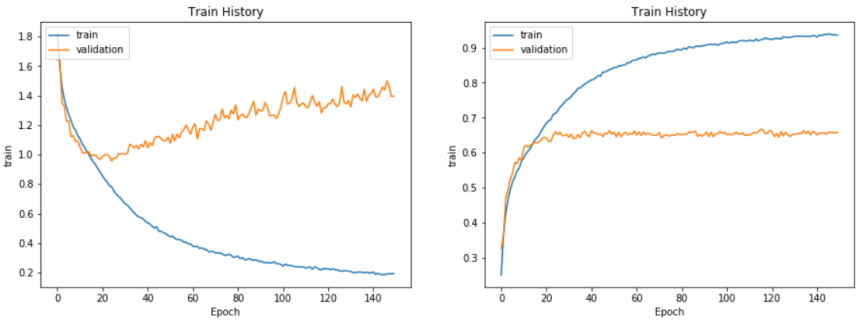
2. (1%) 請嘗試 data normalization, data augmentation,說明實行方法並且說明對準確率有什麼樣的影響？
data normalization:
一開始將所有資料除以 255,在 CNN 架構上每一層有使用 batchnormalization，另外在 train 之前則將每一張照片做 standard normalization 的處理，而此題實驗上則將這兩項控制項作更動。

MODEL	PUBLIC	PRIVATE
w data normalization	0.64558	0.64669
w/o data normalization	0.65645	0.65366
w/o batch normalization	0.58539	0.58735

有沒有在一開始做 standard normalization 在有使用 bathnormalization 條件下其實沒有差很多甚至比本來效果差，但如果沒有 batchnormalization 的話 loss 下降速度會明顯變慢。

data augmentation:

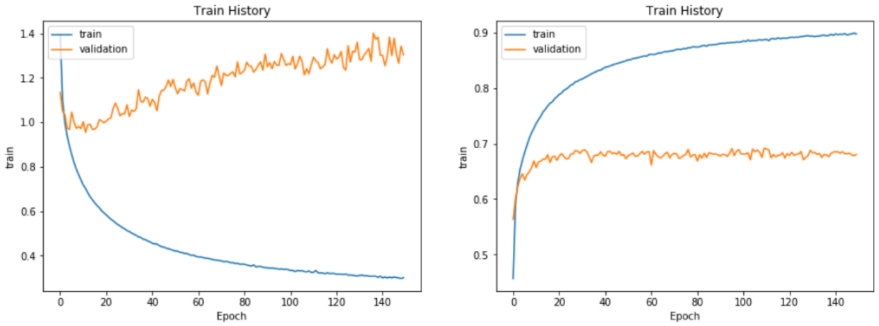
原來資料曲線圖:



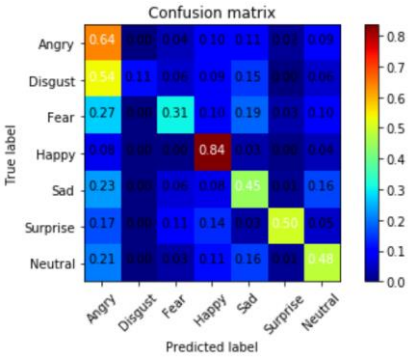
agumentation 的部分採上下左右平移以及左右翻轉(資料量變成 12 倍)

還沒做資料強化以前 valid 分數約略在 0.65 左右，kaggle 的 acc 最高為 0.65645

強化後 valid acc 可以到 0.6915，在 kaggle 上表現為 0.70047

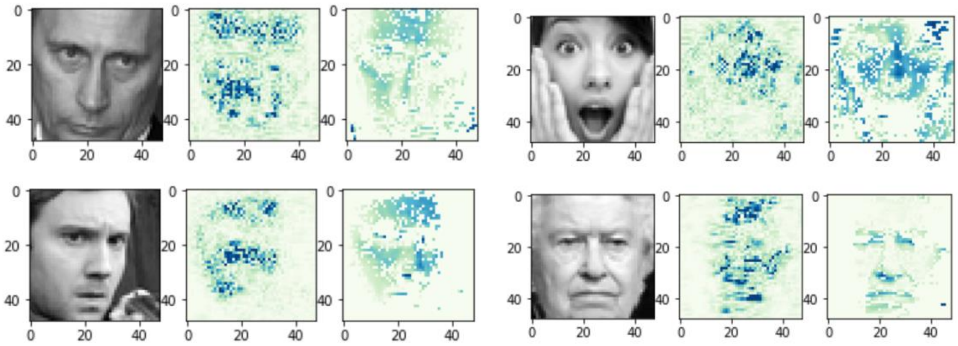


3. (1%) 觀察答錯的圖片中，哪些 class 彼此間容易用混？[繪出 confusion matrix 分析]



由圖可見，生氣容易誤判成尷尬。

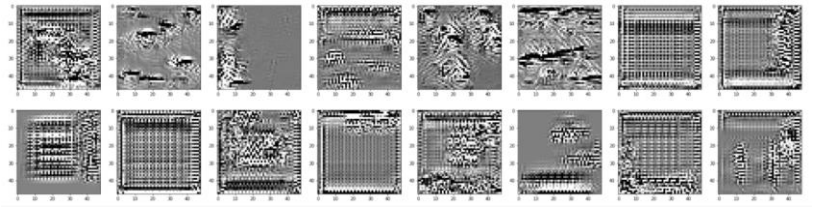
4. (1%) 從(1)(2)可以發現，使用 CNN 的確有些好處，試繪出其 saliency maps，觀察模型在做 classification 時，是 focus 在圖片的哪些部份？



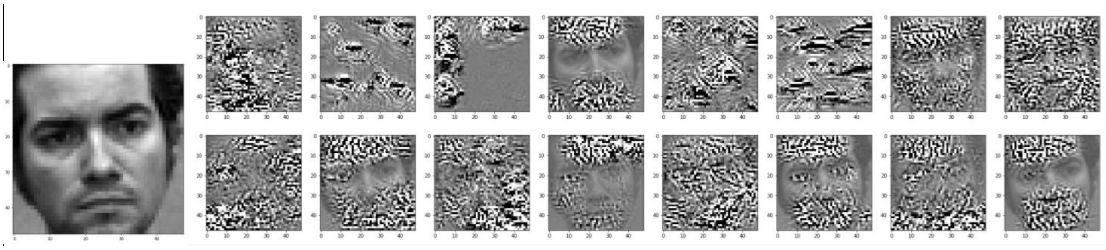
由圖可知，model 大部分會注意的地方除了五官以外還會注意臉部肌肉的變化以及臉部光強度高的地方，個人推斷之所以眉毛以及嘴型或是眼睛聚焦程度較低的原因是因為人在表達情緒的時候這些看似很重要的地方反而滿不規律的，而臉頰肌肉反而比較容易辨別。

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。

取第七層 filter:



Input: **Output:**



由 output 推測額頭或是鼻形或是臉頰以及下巴的部分比較容易被 activate，
如第四題的結果雷同。