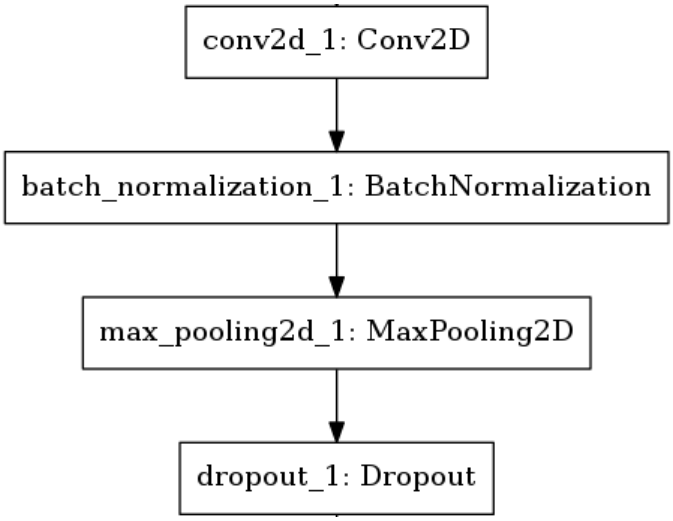
1. (1%) 請說明你實作的 CNN model，其模型架構、訓練過程和準確率為何？(Collaborators: None)

我設計每一層 Cnn 右圖所示，經過 convolution 後每個 batch 會做正規化，接這在進去 pooling layer 將維度縮小，最後做 dropout。我採的參數固定有兩個： Conv2D 的 kernel_size = (3,,3)，而 MaxPooling2D 的 pool_size = (2,2)。其他 filter, dropout 參數則隨意調整。




```
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(7))
model.add(Activation('softmax'))
```

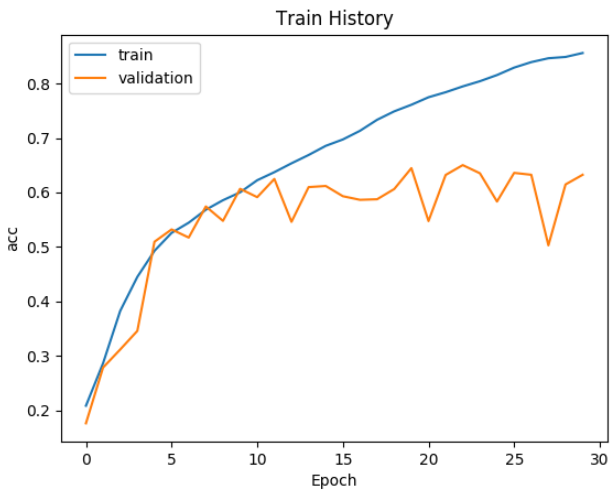
而經過 Cnn 後經過 Flatten 轉成一維，再經過兩層 dense layer ，最後接輸出層。

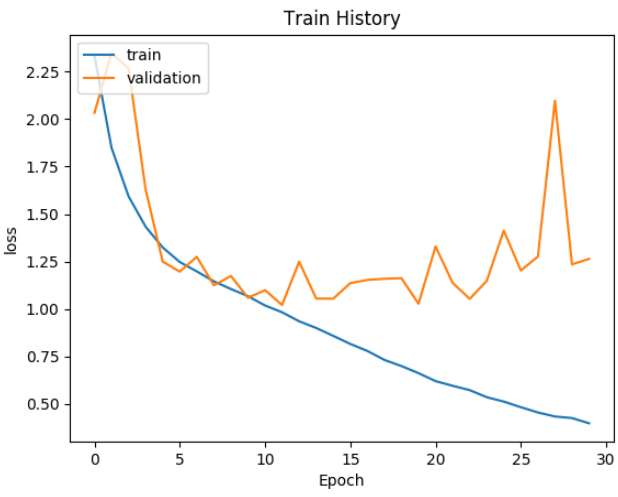
另外 training 的 epoch 我設為 30 ， batch_size = 100。

在最一開始實驗時，我疊了兩層 Cnn ，在 kaggle 的數據為 0.54639，於是我馬上再試著多疊上一層 Cnn ，分數便來到了0.58726。接著我又試了四層與五層，分數分別為0.62126、0.63471，可以看出分數已經漸漸來到了瓶頸。

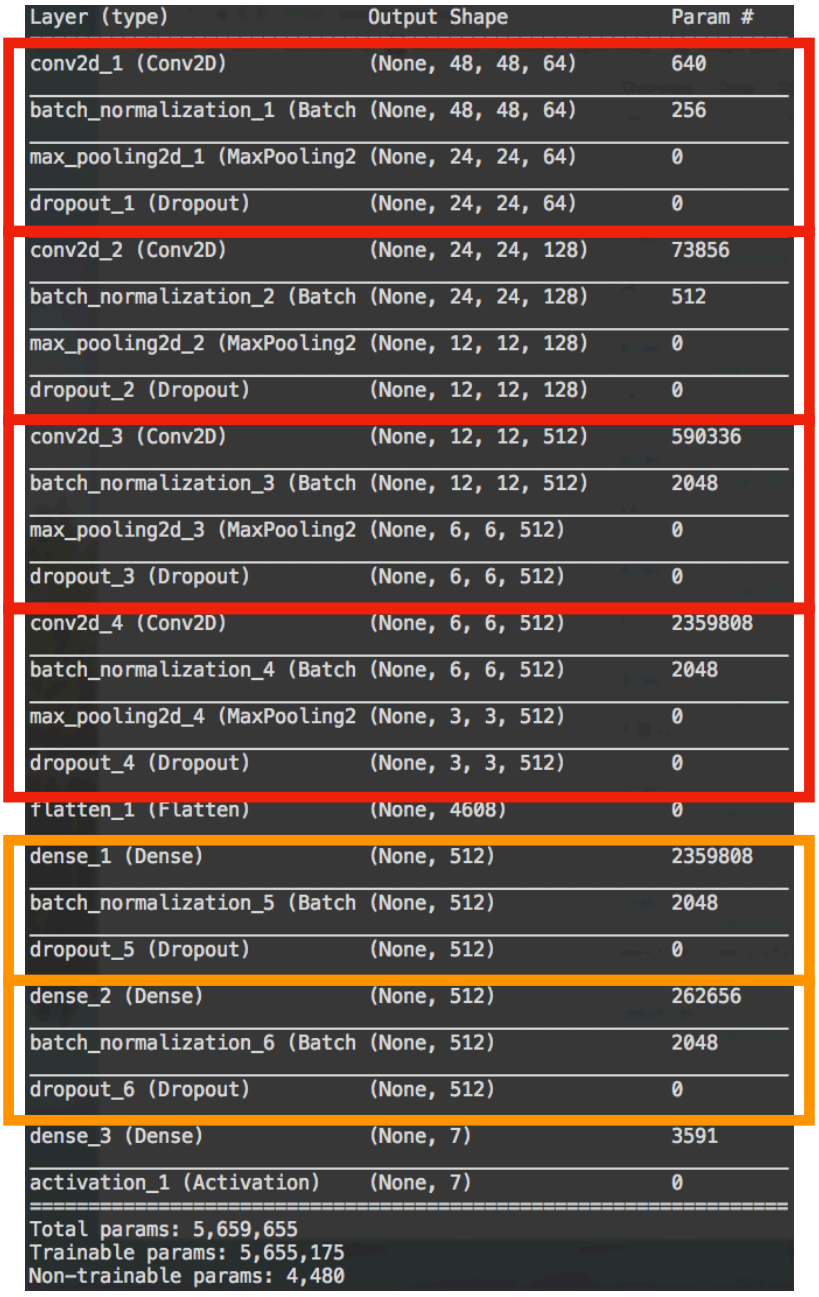
最後我選疊了四層的 Cnn ，參數量：5,660,679，要與 Dnn model 做比較。

下左圖是在訓練過程中的準確率，右圖則是 loss。





以下為整體架　　構圖。



Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 48, 48, 64)	640
batch_normalization_1 (Batch	(None, 48, 48, 64)	256
max_pooling2d_1 (MaxPooling2	(None, 24, 24, 64)	0
dropout_1 (Dropout)	(None, 24, 24, 64)	0
conv2d_2 (Conv2D)	(None, 24, 24, 128)	73856
batch_normalization_2 (Batch	(None, 24, 24, 128)	512
max_pooling2d_2 (MaxPooling2	(None, 12, 12, 128)	0
dropout_2 (Dropout)	(None, 12, 12, 128)	0
conv2d_3 (Conv2D)	(None, 12, 12, 512)	590336
batch_normalization_3 (Batch	(None, 12, 12, 512)	2048
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 512)	0
dropout_3 (Dropout)	(None, 6, 6, 512)	0
conv2d_4 (Conv2D)	(None, 6, 6, 512)	2359808
batch_normalization_4 (Batch	(None, 6, 6, 512)	2048
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, 512)	2359808
batch_normalization_5 (Batch	(None, 512)	2048
dropout_5 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 512)	262656
batch_normalization_6 (Batch	(None, 512)	2048
dropout_6 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 7)	3591
activation_1 (Activation)	(None, 7)	0

Total params: 5,659,655
Trainable params: 5,655,175
Non-trainable params: 4,480

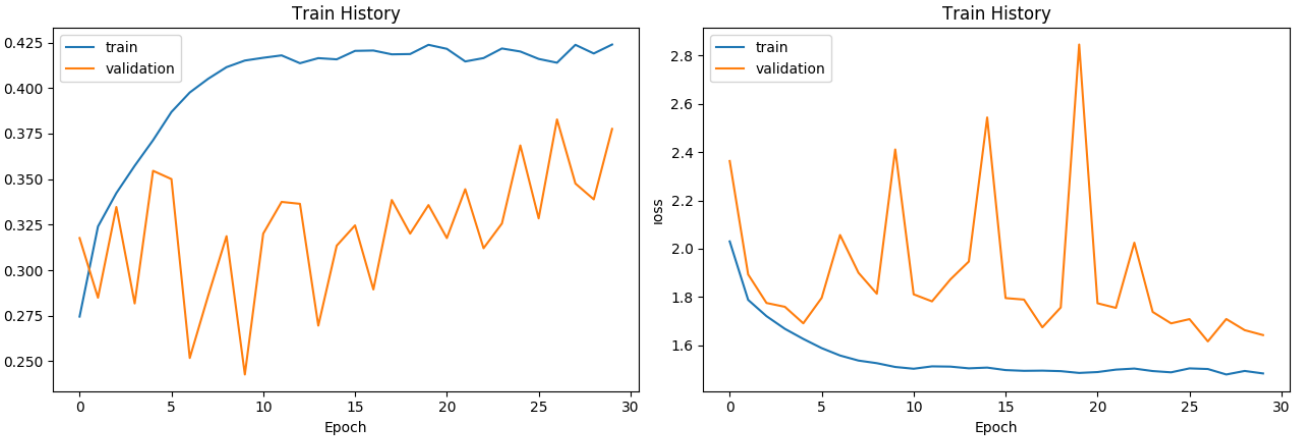
2.(1%) 承上題，請用與上述 CNN 接近的參數量，實做簡單的 DNN model。其模型架構、訓練過程和準確率為何？試與上題結果做比較，並說明你觀察到了什麼？(Collaborators: None)

在這題中，我設計中間有三層 hidden layer 第一層用 1440 unit，第二層用 1024 unit，最後用 768 unit，最後參數量為 5600295，跟上述 cnn model 的參數量非常接近。

Layer (type)	Output Shape	Param #
flatten_1 (Flatten)	(None, 2304)	0
dense_1 (Dense)	(None, 1440)	3319200
batch_normalization_1 (Batch	(None, 1440)	5760
dropout_1 (Dropout)	(None, 1440)	0
dense_2 (Dense)	(None, 1024)	1475584
batch_normalization_2 (Batch	(None, 1024)	4096
dropout_2 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 768)	787200
batch_normalization_3 (Batch	(None, 768)	3072
dropout_3 (Dropout)	(None, 768)	0
dense_4 (Dense)	(None, 7)	5383
activation_1 (Activation)	(None, 7)	0

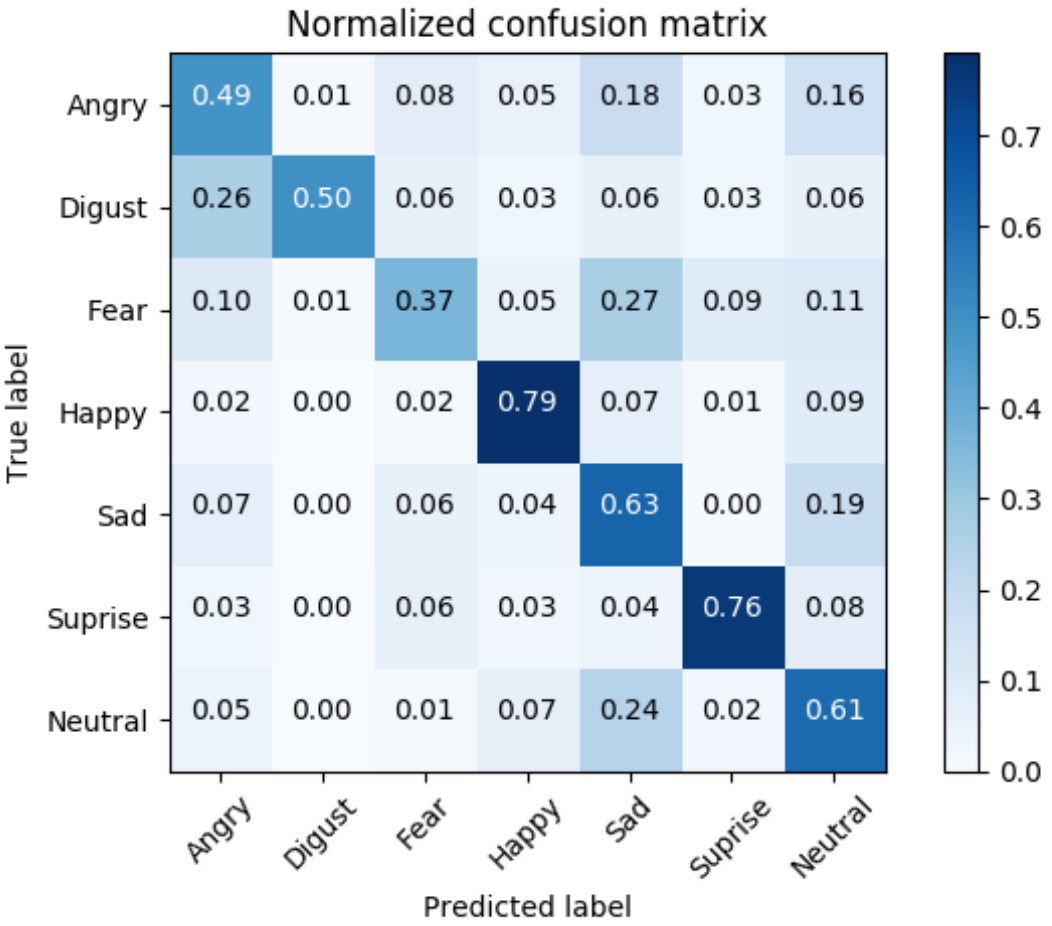
Total params: 5,600,295
Trainable params: 5,593,831
Non-trainable params: 6,464

下左圖是在訓練過程中的準確率，右圖則是 loss。波動非常的大。



我發現在 dnn model 下，每一次 epoch 的訓練時間快 cnn model 許多，
但在 validation set 上的準確度真的差滿多的，所以我也沒有丟上 kaggle 去看。
因為可想分數一定會很差XD 然後可以看出傳統的 dnn model 真的會忽略資料的形狀。
影像真的需要看周圍資訊。

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: None)

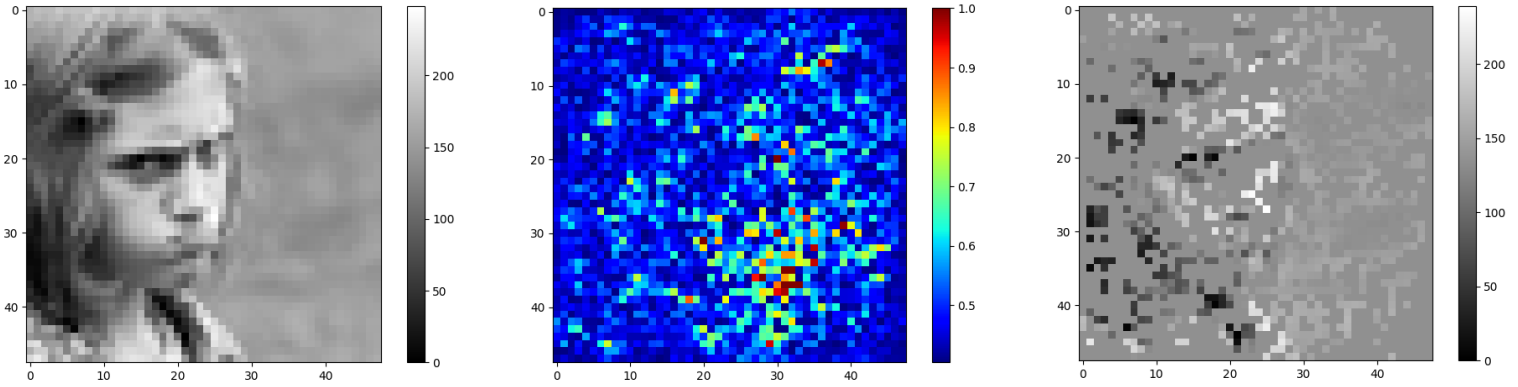


我的 model 在預測鑄要發生錯誤在：
a. 生氣時，會誤判成傷心、中立
b. 厭惡時，誤判成生氣
c. 恐懼時，誤判成傷心
d. 傷心時，誤判成中立
e. 中立時，誤判成傷心

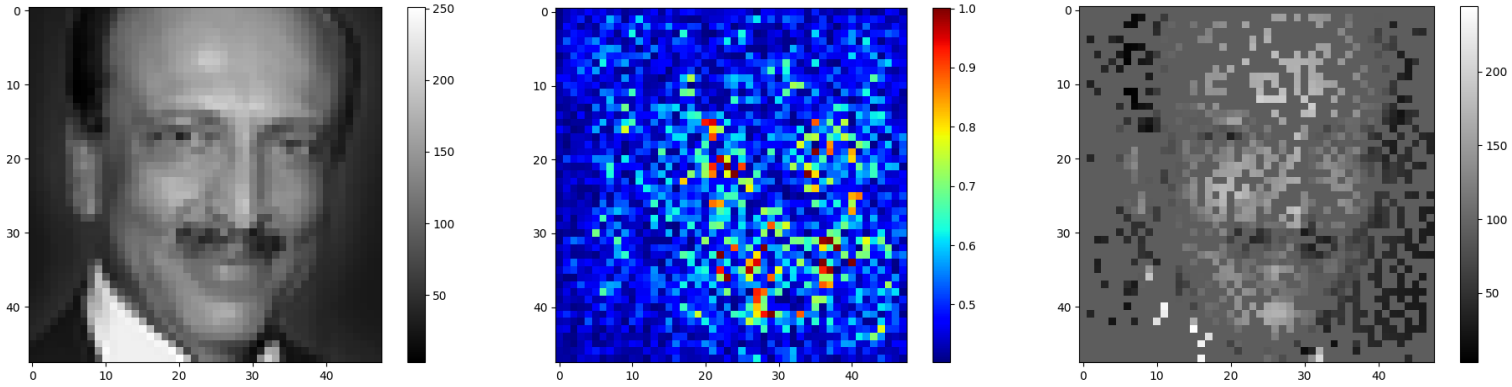
我的模型只有在開心跟驚訝時效果比較好，我覺得是因為嘴巴的特徵比較明顯，可以訓練出比較好的效果。另外我想我的模型應該還有有更多進階的做法才能預測的比較準確，單純只有疊 Cnn 不夠，應該可以試試 ensemble 的方法，訓練多種不同 model 然後把資料分別用這些 model 預測，最後把預測平均起來。若有時間研究我相信能拿到較好分數的機會就大了一點。

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: None)

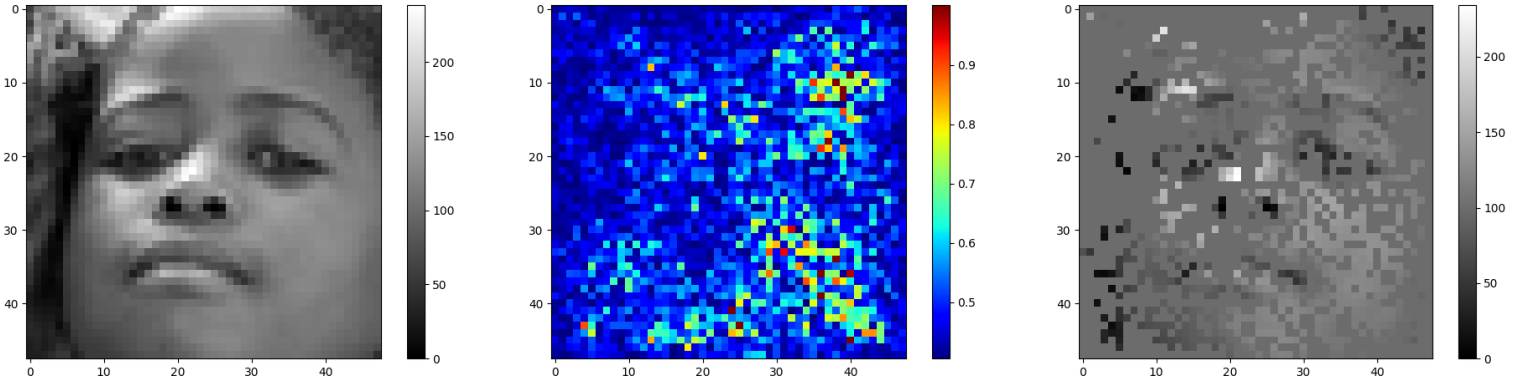
我隨機取的 image id 有 [928,8377,102]，他們 label 分別為 0,3,6 (生氣、高興、中立)



生氣



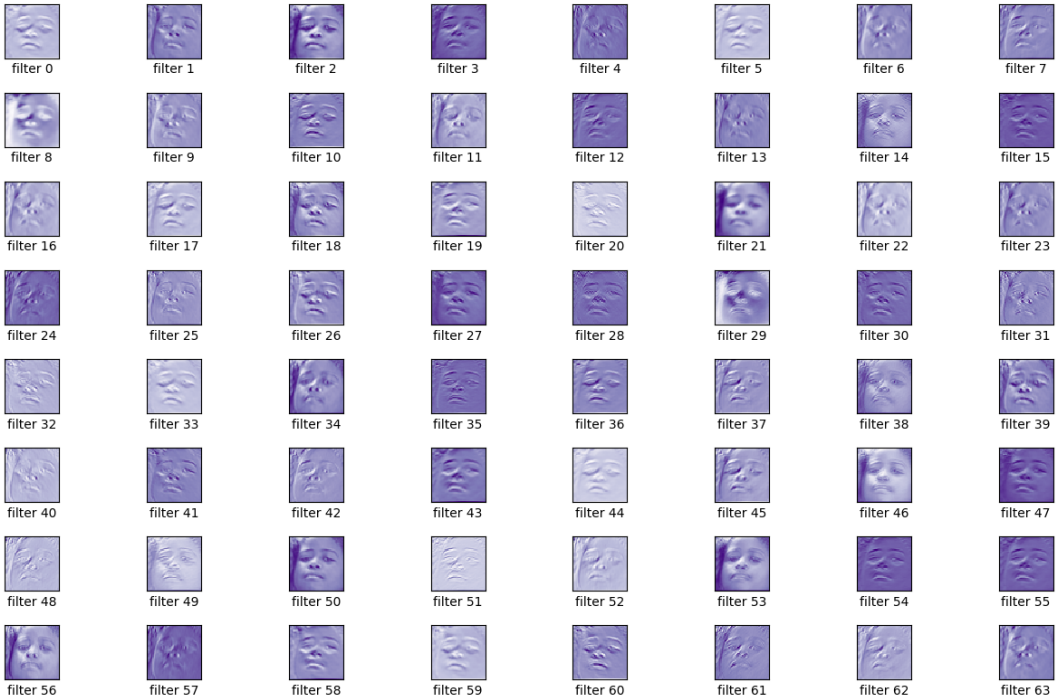
高興



中立

我覺得我隨機取的這些都滿注意嘴巴的，像生氣的那張圖，嘴巴就像是比較主要的。另外，高興跟中立臉頰部分還佔滿多的，但我覺得臉頰並不會是人在判斷這些表情的主要依據。但機器卻有拿臉頰來當作判斷依據之一。

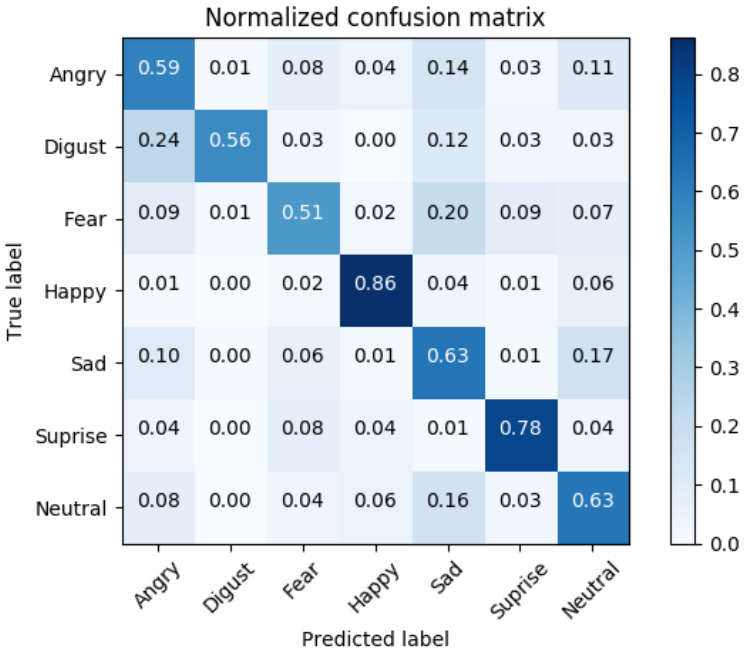
5. (1%) 承(1)(2)，利用上課所提到的 gradient ascent 方法，觀察特定層的filter最容易被哪種圖片 activate。
(Collaborators: None)
我選了 conv2d_1 來觀察。我的第一層 filter 有 64 個。

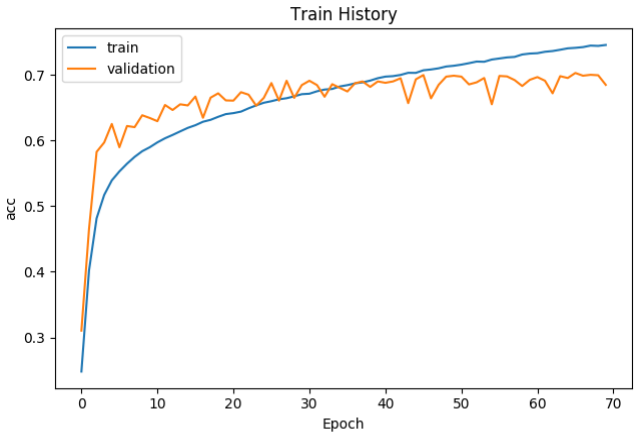


把圖片丟進去 filter 後看圖片輸出，是真的可以看出 filter 是真的有對應，只不過我覺得第一層的 texture 普遍來說滿不明顯的。

Note:

寫完這份報告後，我告訴有機器學期經驗的同學：我疊多層 Cnn 與不段試參數，但 Accuracy 就是難以上升，他告訴我說試試看先做 image processing 可以讓訓練量變大（keras ImageDataGenerator），於是我就試了這個方法，在訓練過程中就有發現 valid accuracy 的確有慢慢上升 (epoch=70)，最後在 kaggle 我得到 0.68793，以下補上 confusion matrix 與 training history。





Reference:

1. http://puremonkey2010.blogspot.tw/2017/07/toolkit-keras-mnist-cnn.html
2. https://keras.io/preprocessing/image/