學號：R04945008 系級： 生醫電資碩二 姓名：黃思翰

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：

以下以 C 代表 2D Convolution layer，P 代表 2D Max pooling layer，F 代表 Fully connected layer 表示，為了衝高 kaggle 排名，參考了 Pramerdorfer, C., & Kampel, M. (n.d.). Facial Expression Recognition using Convolutional Neural Networks : State of the Art.論文，以 CCPCCPCCPCCPFF (VGG) 為 model 大致的架構，於每段 CCP 前端加上 zero padding layer、並於 max pooling layer 後加上 dropout 來提高整個 model 的準確率，而每個 convolution layer 後面會接上一個 PReLU activation function、normaliztion layer 來穩定模型。在 loss function 的部分選擇了 cross entropy，optimizer 則是使用 Adam，epoch 數為 80。

訓練過程將原始 training data 切出 10% 共 2909 張圖片來做 validation data，並將剩下的 training data 共 25800 張影像每張除以 255，normalize 至 0~1 後，以 batch size = 100 丟入模型訓練，而為了提高 kaggle 分數我另外有用 keras 的 ImageDataGenerator，將 training data 做隨機的水平、垂直翻轉以及上下左右的位移。

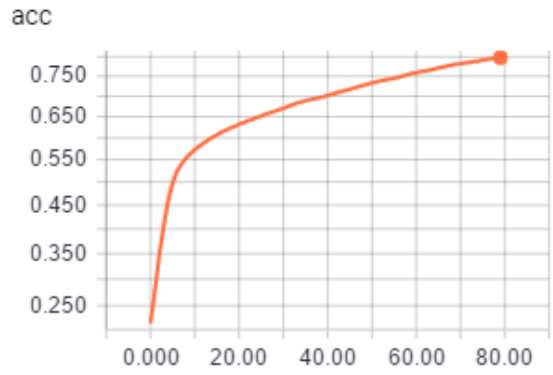
下圖為使用 tensorboard 所記錄的模型架構，因為模型過大，所以圖示會較小點，額外貼上 model.summary()的結果供助教參考，此模型總共使用的參數量為 9,976,775。

Layer (type)	Output Shape	Param #
zero_padding2d_1 (ZeroPaddin	(None, 50, 50, 1)	0
conv1_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
zero_padding2d_2 (ZeroPaddin	(None, 50, 50, 64)	0
conv1_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
dropout_1 (Dropout)	(None, 24, 24, 64)	0
zero_padding2d_3 (ZeroPaddin	(None, 26, 26, 64)	0
conv2_1 (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
zero_padding2d_4 (ZeroPaddin	(None, 26, 26, 128)	0
conv2_2 (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
dropout_2 (Dropout)	(None, 12, 12, 128)	0
zero_padding2d_5 (ZeroPaddin	(None, 14, 14, 128)	0
conv3_1 (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
zero_padding2d_6 (ZeroPaddin	(None, 14, 14, 256)	0
conv3_2 (Conv2D)	(None, 12, 12, 256)	590080
p_re_lu_6 (PReLU)	(None, 12, 12, 256)	36864

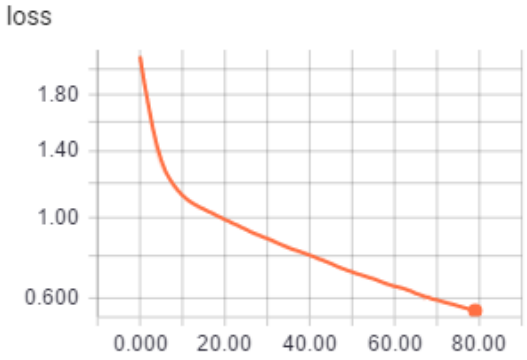
batch_normalization_6 (Batch	(None, 12, 12, 256)	1024
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 256)	0
dropout_3 (Dropout)	(None, 6, 6, 256)	0
zero_padding2d_7 (ZeroPaddin	(None, 8, 8, 256)	0
conv4_1 (Conv2D)	(None, 6, 6, 512)	1180160
p_re_lu_7 (PReLU)	(None, 6, 6, 512)	18432
batch_normalization_7 (Batch	(None, 6, 6, 512)	2048
zero_padding2d_8 (ZeroPaddin	(None, 8, 8, 512)	0
conv4_2 (Conv2D)	(None, 6, 6, 512)	2359808
p_re_lu_8 (PReLU)	(None, 6, 6, 512)	18432
batch_normalization_8 (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
fc5 (Dense)	(None, 1024)	4719616
p_re_lu_9 (PReLU)	(None, 1024)	1024
batch_normalization_9 (Batch	(None, 1024)	4096
dropout_5 (Dropout)	(None, 1024)	0
fc6 (Dense)	(None, 7)	7175

Total params: 9,976,775.0
Trainable params: 9,970,887.0
Non-trainable params: 5,888.0

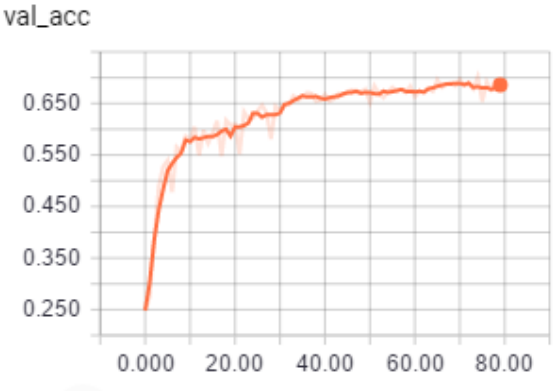
以下附上有使用 ImageDataGenerator 時的 training data、validation data 的 accuracy、loss 隨著 epoch 變化的趨勢。



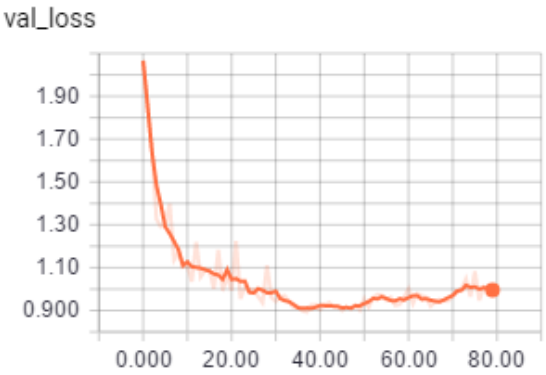
training data 的 accuracy



training data 的 loss

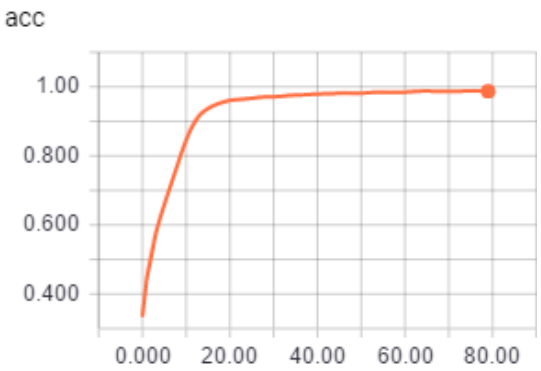


validation data 的 accuracy

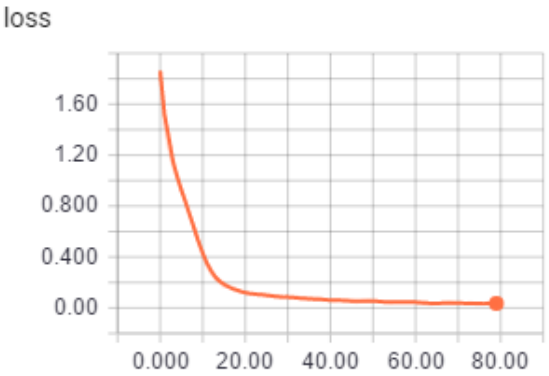


validation data 的 loss

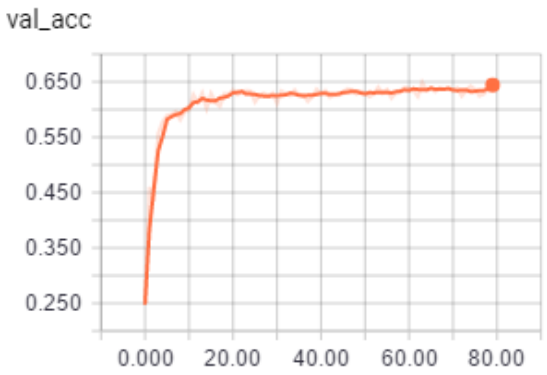
以下附上沒使用 ImageDataGenerator 時的 training data、validation data 的 accuracy、loss 隨著 epoch 變化的趨勢。



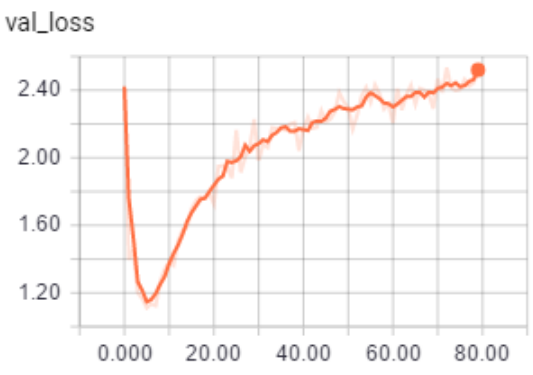
training data 的 accuracy



training data 的 loss



validation data 的 accuracy



validation data 的 loss

可以觀察到，有做 ImageDataGenerator 的因為會隨機的將圖片做上下、左右位移及翻轉，所以在 training 時增加了辨識的難度，training data 的 accuracy 比起沒做的偏低很多，但相對的整個模型在辨識的能力卻因此增強，可以觀察到在 validation data 的 accuracy 則是有做 ImageDataGenerator 的高上許多，這點也同時體現在 kaggle 的分數上，有使用 ImageDataGenerator 的模型最終在 kaggle 上拿到 0.677 的 accuracy，而沒做的則拿到 0.628 的 accuracy。此外沒做 ImageDataGenerator 的模型較早就開始收斂，因此在 train 同樣的 epoch 數時會因為 train 過久 train 壞使 loss 逐漸升高。

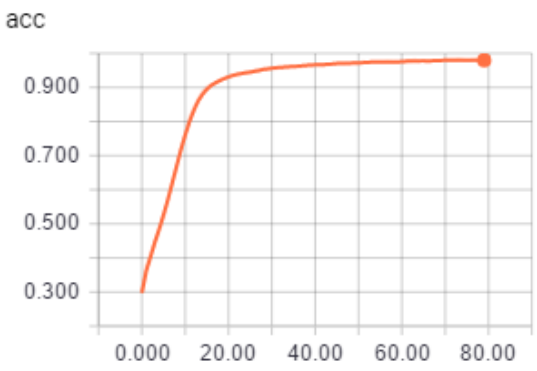
2. (1%) 承上題，請用與上述 CNN 接近的參數量，實做簡單的 DNN model。其模型架構、訓練過程和準確率為何？試與上題結果做比較，並說明你觀察到了什麼？

答：CNN 的模型總共使用了 9,976,775 個參數，仿照同樣參數量我設計了一個 DNN model 共使用 9,893,767 個參數，如下圖架構所示，由四層 full connected layer 所組成，每層的 activation function 一樣為 PReLU，而在進入下層前會先經過一個 normalization layer 並做 dropout 處理。

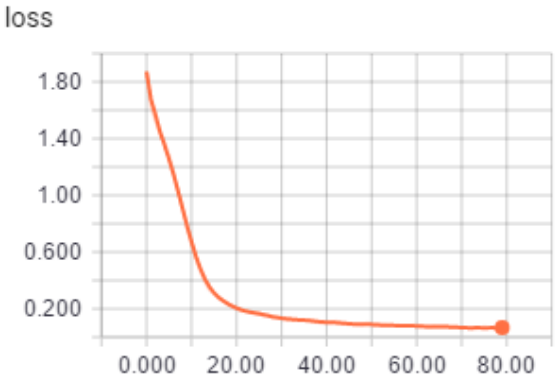
Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 2304, 64)	128
fc1 (Dense)	(None, 2304, 64)	4160
p_re_lu_1 (PReLU)	(None, 2304, 64)	147456
batch_normalization_1 (Batch	(None, 2304, 64)	256
dropout_1 (Dropout)	(None, 2304, 64)	0
fc2 (Dense)	(None, 2304, 64)	4160
p_re_lu_2 (PReLU)	(None, 2304, 64)	147456
batch_normalization_2 (Batch	(None, 2304, 64)	256
dropout_2 (Dropout)	(None, 2304, 64)	0
fc3 (Dense)	(None, 2304, 64)	4160
p_re_lu_3 (PReLU)	(None, 2304, 64)	147456
batch_normalization_3 (Batch	(None, 2304, 64)	256
dropout_3 (Dropout)	(None, 2304, 64)	0
flatten_1 (Flatten)	(None, 147456)	0
fc4 (Dense)	(None, 64)	9437248
p_re_lu_4 (PReLU)	(None, 64)	64
batch_normalization_4 (Batch	(None, 64)	256
dropout_4 (Dropout)	(None, 64)	0
fc5 (Dense)	(None, 7)	455

Total params: 9,893,767.0
Trainable params: 9,893,255.0
Non-trainable params: 512.0

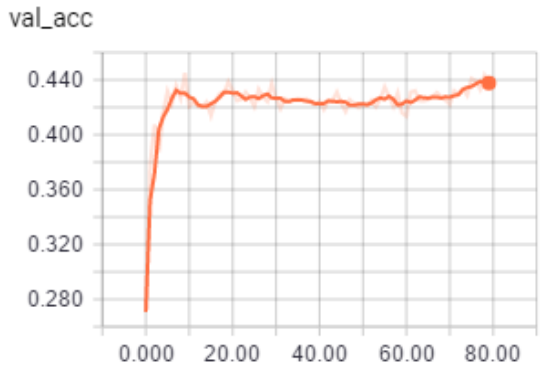
此 model 的 traning data 並未做 ImageDataGenerator 的處理，但一樣將原始 training data 切出 10% 共 2909 張圖片來做 validation data，並將剩下的 training data 共 25800 張以 batch size = 100，epoch = 80 去做訓練，以下附上此模型的的 training data、validation data 的 accuracy、loss 隨著 epoch 變化的趨勢。



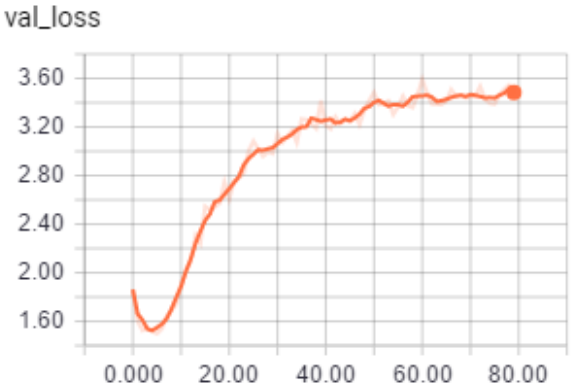
training data 的 accuracy



training data 的 loss



validation data 的 accuracy



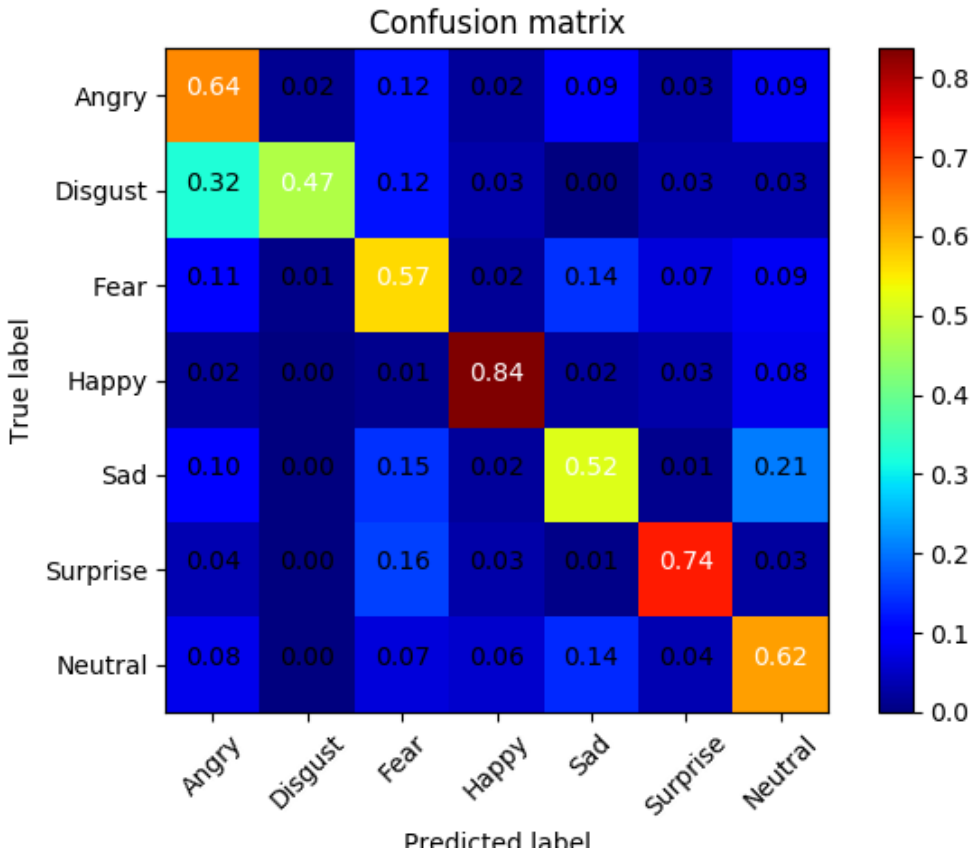
validation data 的 loss

此 DNN 模型在 validation data 的 accuracy 被 CNN 海放，可見影像辨識在相同參數下使用 CNN 會比較好，而此模型在 kaggle 上的分數為 0.431。在設計模型的時候我有觀察到，因為 DNN 參數量很容易暴增，所以為了控制一樣的參數量，我並沒有將模型疊的很深，相比之下雖然參數量 CNN、DNN 相近，但是深度卻差很多，我想這是 DNN 的 accuracy 如此低的原因之一。

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：

此題作答使用 validation data (沒拿去訓練)共 2909 張圖片來繪製 confusion matix，從下圖可以觀察到我的模型辨識快樂的表情正確率最高有 0.84，次高為辨識驚訝表情有 0.74 而厭惡表情最低僅 0.47，此外我的模型將厭惡表情誤判為生氣的機率有高達 0.32。

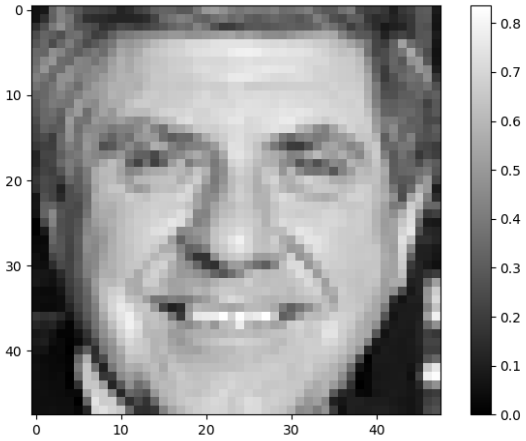


Confusion matrix

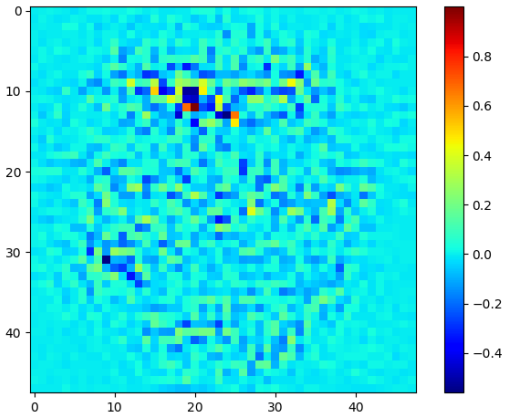
4. (1%) 從(1)(2)可以發現，使用 CNN 的確有些好處，試繪出其 saliency maps，觀察模型在做 classification 時，是 focus 在圖片的哪些部份？

答：

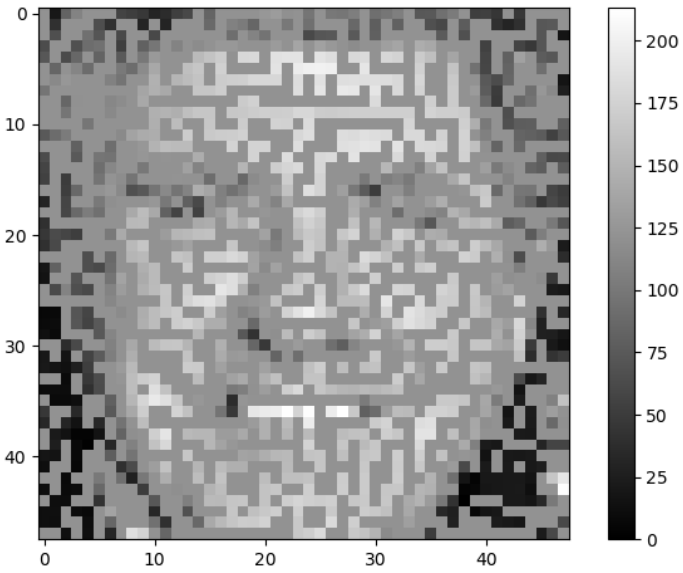
此題作答使用 validation set 的 data 來觀察我的模型在分類對跟分類錯時所觀察到的 saliency map 狀況，下圖為高興的表情，使用我的模型預測出來也為高興的分類，計算出原圖通過模型的梯度後，簡單的將梯度減去平均並除上標準差來做平滑化，並除上最大值上以方便繪出 saliency map。可以觀察到 saliency map 隱隱約約的有原圖整張臉的輪廓，而梯度變化最大的地方接近眉毛與額頭的交界處及右側的臉頰，其餘的五官在 mask 掉梯度較小的區域後也顯現出來。



原圖-高興表情

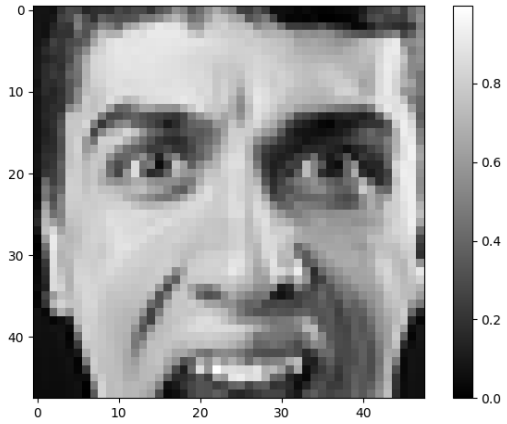


saliency map

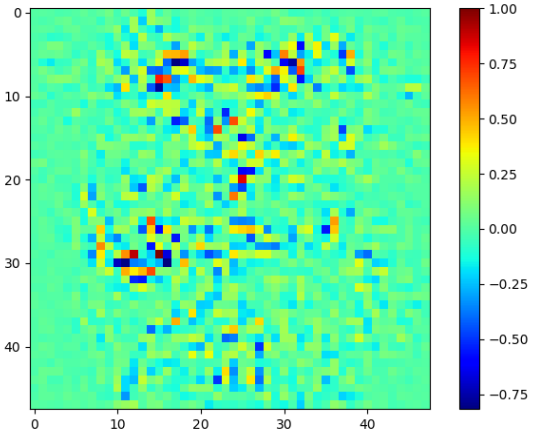


將 saliency map 值較小的地方 mask 掉

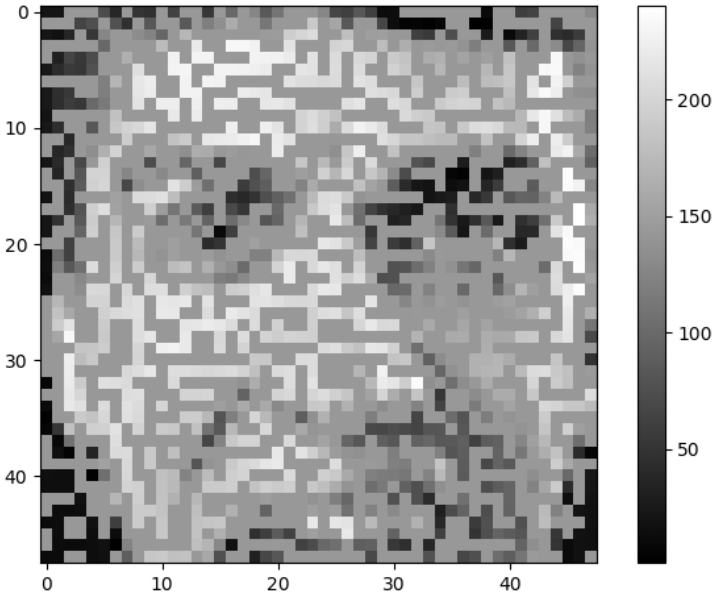
而在模型分類錯誤的情況如下，原圖為恐懼的表情，模型判斷為驚訝的表情。從 saliency map 上可以觀察到，梯度變化最明顯的地方為眉毛、眼睛四周、兩頰、嘴巴等 edge 較明顯的位置，因為人在驚訝與恐懼時臉部肌肉反應的方式相近，我個人也認為此原圖屬於模玲兩可的，模型判斷錯誤應屬正常。



原圖-恐懼表情



saliency map

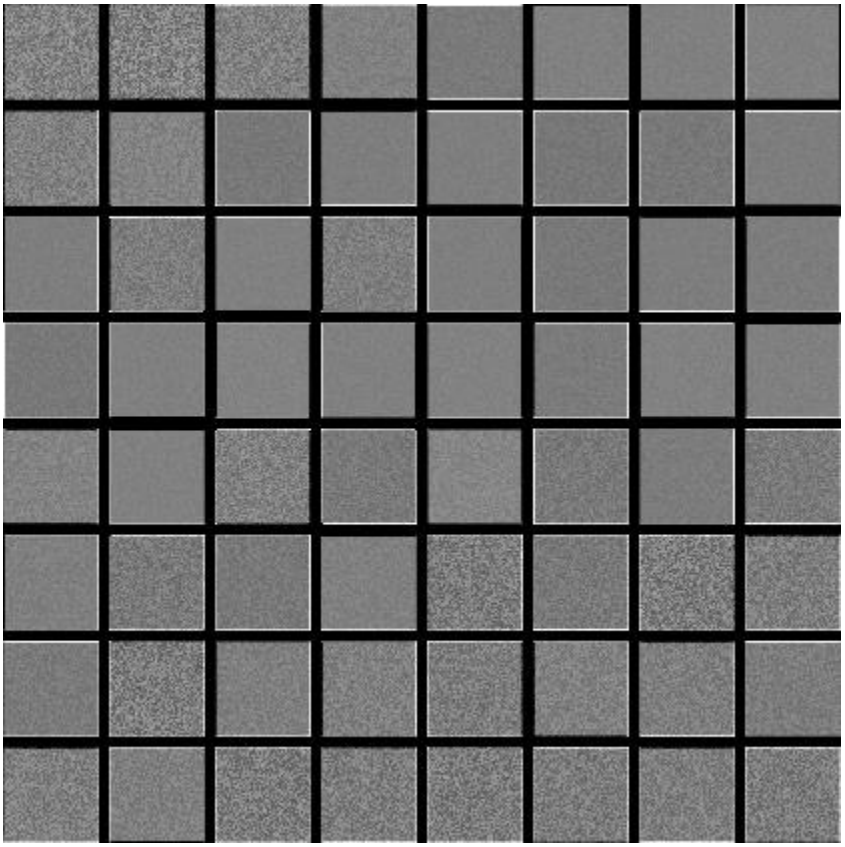


將 saliency map 值較小的地方 mask 掉

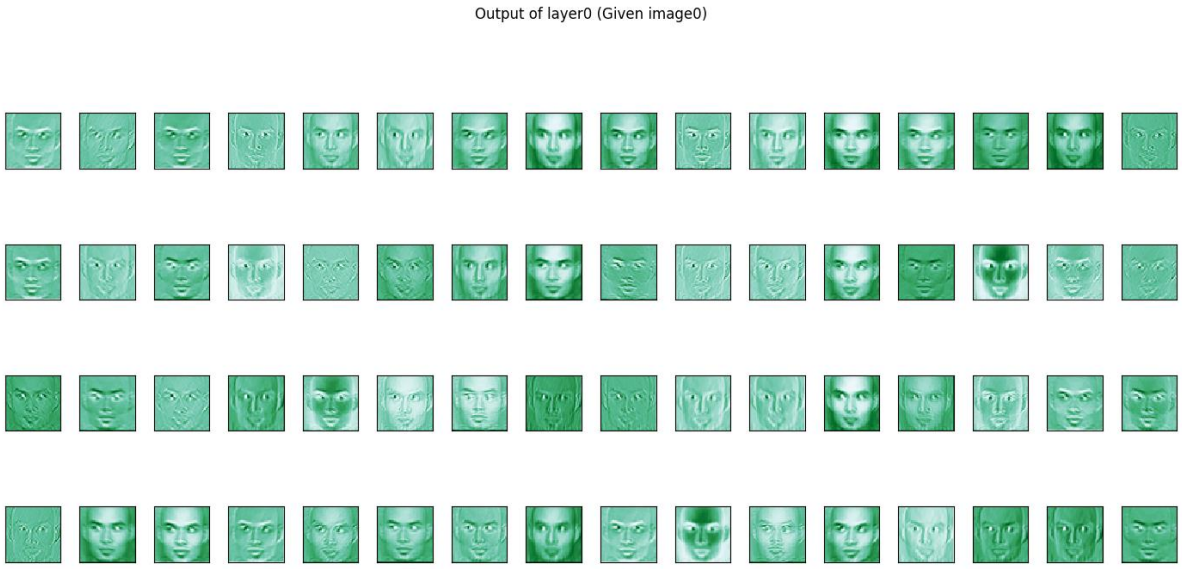
5. (1%) 承(1)(2)，利用上課所提到的 gradient ascent 方法，觀察特定層的 filter 最容易被哪種圖片 activate。
答：

此題我所觀察的為 CNN 模型最一開始的 conv1_1 layer 及位於模型正中央的 conv3_2 layer，首先先輸入隨機產生的 white noise 輸入至此兩層的 filter，並利用老師上課所教的梯度遞增法來計算出各個 filter 的輸出結果，最後透過扣掉標準差再除以平均的方式做平滑化，其中 conv1_1 共有 64 個 filter，而 conv3_2 有 256 個 filter，為了方便呈現，在 conv3_2 輸入 white noise 時僅呈現前面 64 個 filter 的結果。

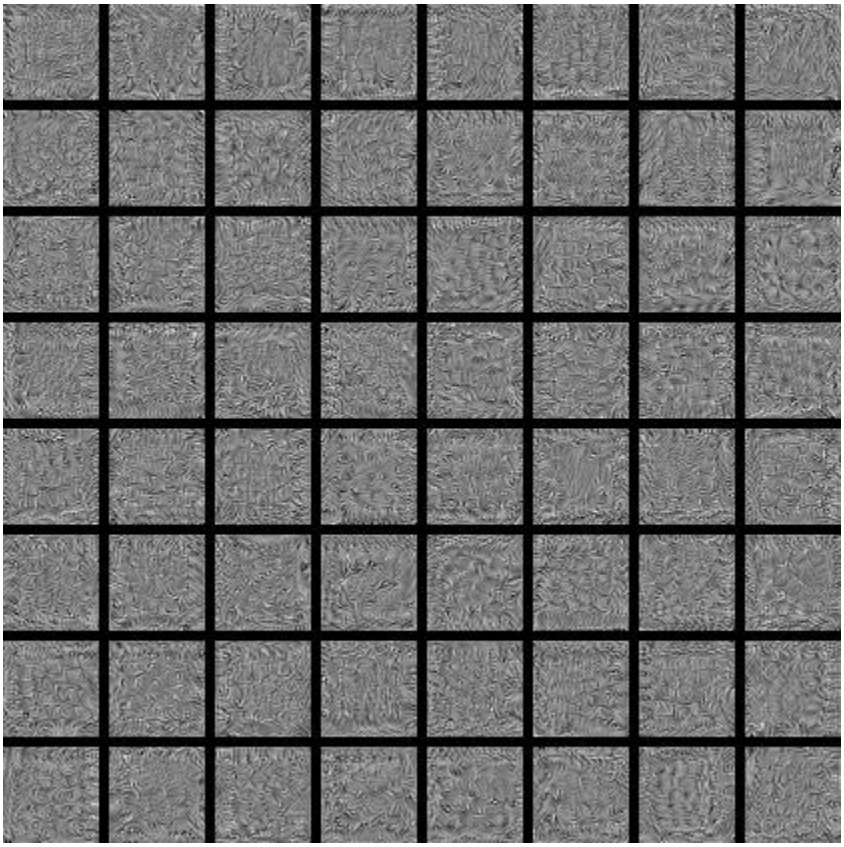
除了輸入 white noise 外，我也輸入了 validation set 的圖片，此圖片為中立表情，我的模型也分類正確。從這兩層的輸出可以觀察到，在前段的 layer，模型主要看到的以人臉的器官或是較明顯的 edge 為主，然而到了中間的 layer 就開始看到神秘的特徵，已經無法言語形容了。



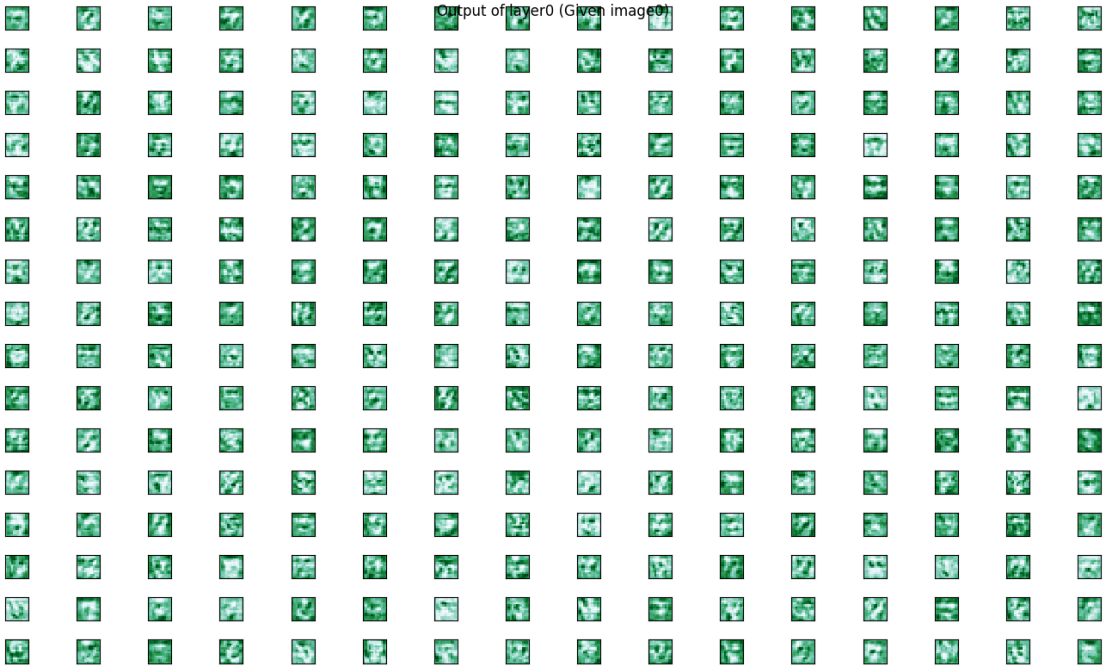
conv1_1 layer 輸入 white noise 時的結果



conv1_1 layer 輸入圖片時的結果



conv3_2 layer 輸入 white noise 時前 64 個 filter 的結果



conv3_2 layer 輸入圖片時的結果

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

[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%]