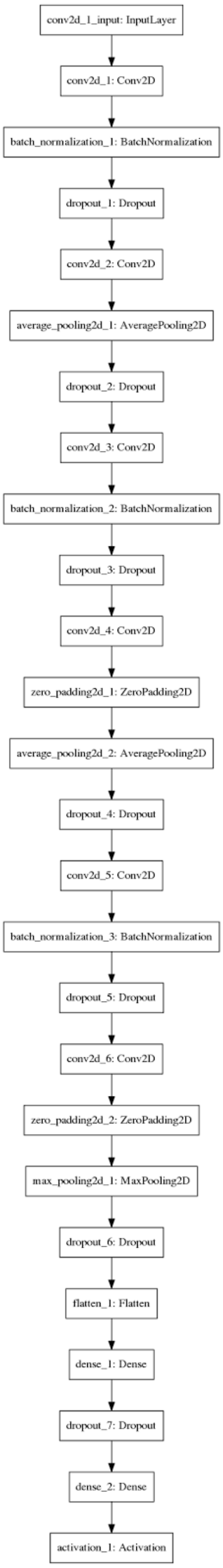
學號：B04902112 系級： 資工二 姓名：張凱捷

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？

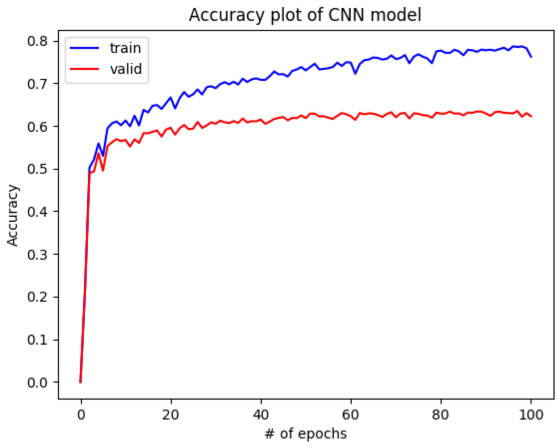
- 模型架構：
  - ❖ plot_model(model, to_file='model.png')




```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 46, 46, 32)        320
_________________________________________________________________
batch_normalization_1 (Batch (None, 46, 46, 32)        128
_________________________________________________________________
dropout_1 (Dropout)          (None, 46, 46, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 44, 44, 32)        9248
_________________________________________________________________
average_pooling2d_1 (Average (None, 22, 22, 32)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 22, 22, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 20, 20, 64)        18496
_________________________________________________________________
batch_normalization_2 (Batch (None, 20, 20, 64)        256
_________________________________________________________________
dropout_3 (Dropout)          (None, 20, 20, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 18, 18, 64)        36928
_________________________________________________________________
zero_padding2d_1 (ZeroPaddin (None, 20, 20, 64)        0
_________________________________________________________________
average_pooling2d_2 (Average (None, 10, 10, 64)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
batch_normalization_3 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
dropout_5 (Dropout)          (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 6, 6, 128)         147584
_________________________________________________________________
zero_padding2d_2 (ZeroPaddin (None, 8, 8, 128)         0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_6 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 712)               1458888
_________________________________________________________________
dropout_7 (Dropout)          (None, 712)               0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 4991
_________________________________________________________________
activation_1 (Activation)    (None, 7)                 0
=================================================================
Total params: 1,751,207
```

- 訓練方法：
    - ❖ training data左右翻轉
    - ❖ 對data做histogram equalization
    - ❖ 使用keras和上述的模型做100個epoch

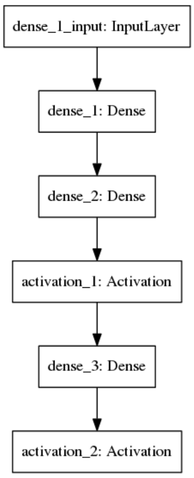
- 訓練過程:
    - ❖ model.fit(train_in, train_out, epochs=100, batch_size=128, validation_split=0.1)



- 準確度：Public test data: 66.70%

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？

- 模型架構
    - ❖ plot_model(model, to_file='model.png')

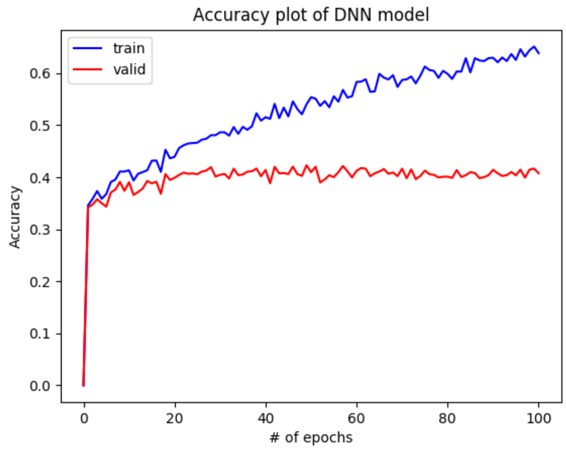


Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 689)	1588145
dense_2 (Dense)	(None, 500)	345000
activation_1 (Activation)	(None, 500)	0
dense_3 (Dense)	(None, 7)	3507
activation_2 (Activation)	(None, 7)	0

Total params: 1,936,652

- 訓練方法：
    - ❖ training data左右翻轉
    - ❖ 對data做histogram equalization
    - ❖ 使用keras和上述的模型做100個epoch

- 訓練過程
    - ❖ model.fit(train_in, train_out, epochs=100, batch_size=128, validation_split=0.1)

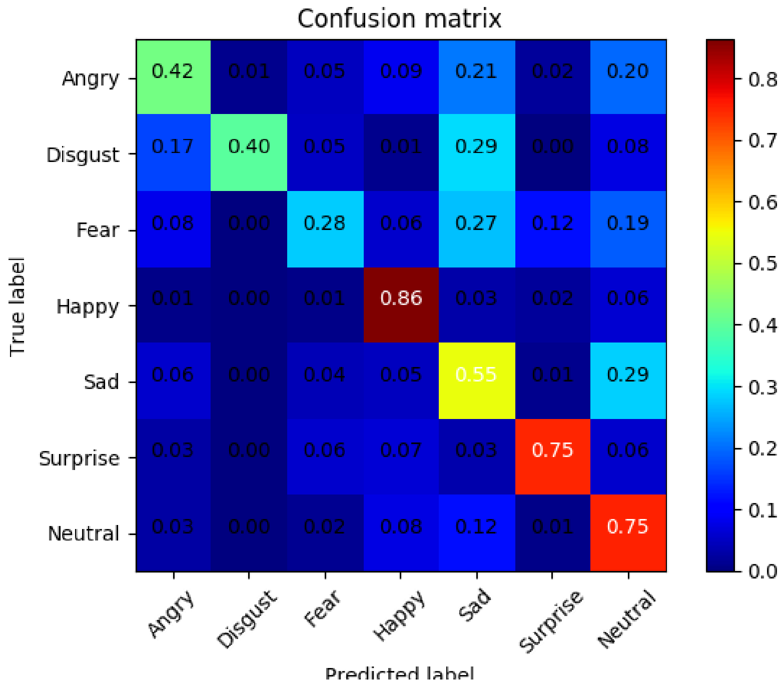


- 準確度：Public test data: 42.29%

- 比較
  - ❖ DNN一個epoch花的時間比較短，可能是因為model的結構比較簡單
  - ❖ 這個DNN model雖然沒有Dropout，但是loss卻下降的比較慢，CNN若沒有Dropout，一下子準確度就飆到9X%造成overfit
  - ❖ 整體的準確度比較低，因為他不像CNN有把data當作一張圖片

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]

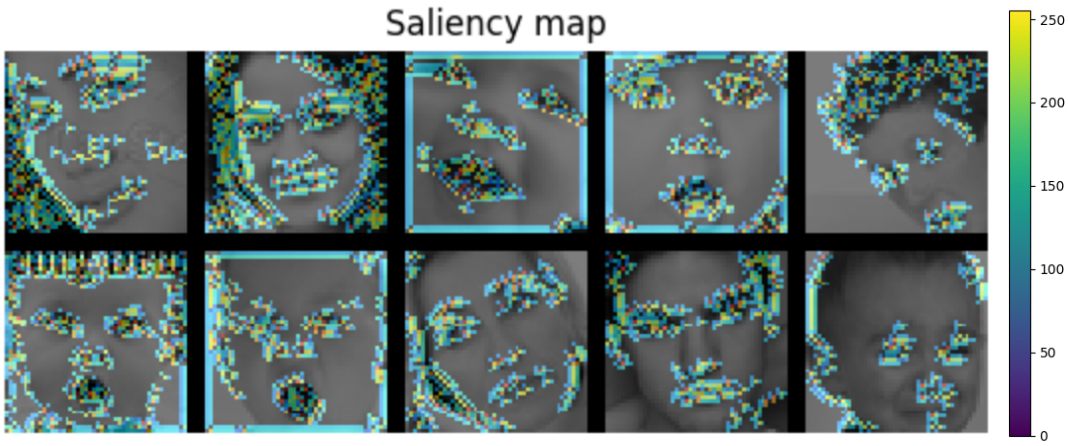
- 使用第1題的model，並將後10%的data來來做validation

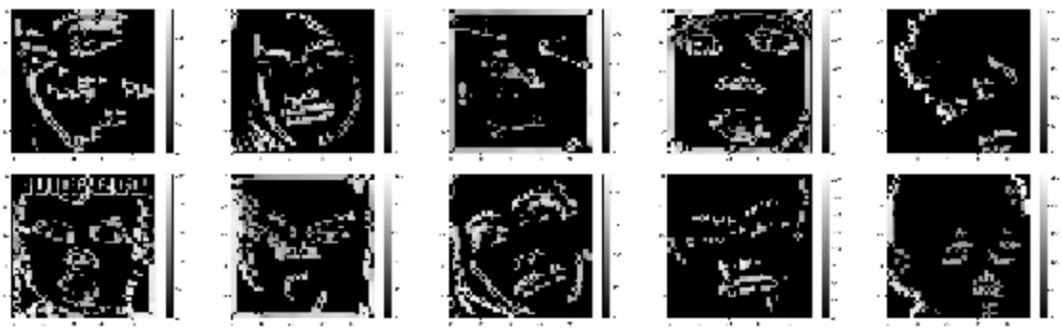


  - ❖ 由上表可知Fear的正確率最低，28%正確率就有27%的資料被誤判成Sad
  - ❖ 正確率低於50%的Angry, Disgust, Fear，皆有超過20%的data被判為Sad，而Sad有29%的資料被誤認為Neutral
  - ❖ Model對Happy的辨識度最高
- Code : *confuse.py*

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？

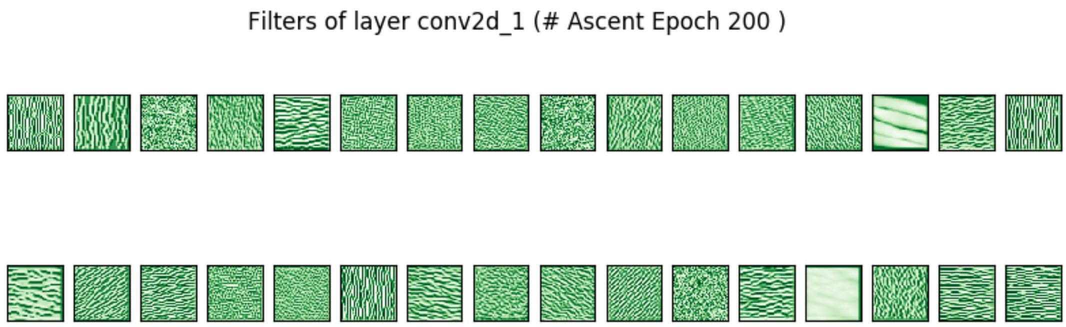
- 使用keras_vis中的visualize_saliency
  - ❖ visualize_saliency(model, 0, [pred_class], test_in[NUM])
  - ❖ 使用CNN model的第0層、pred_class為預測的答案、test_in[NUM]為原始圖片
  - ❖ gradient愈大愈偏紅色，gradient低於一定程度不會顯示
  - ❖ 取testing data中id為以下的圖片
  - ❖ ids = [1, 5, 10, 18, 20, 21 ,28, 43, 50, 78]

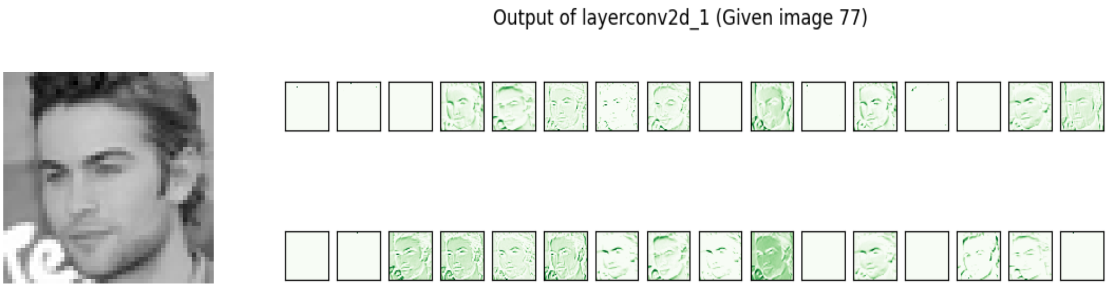


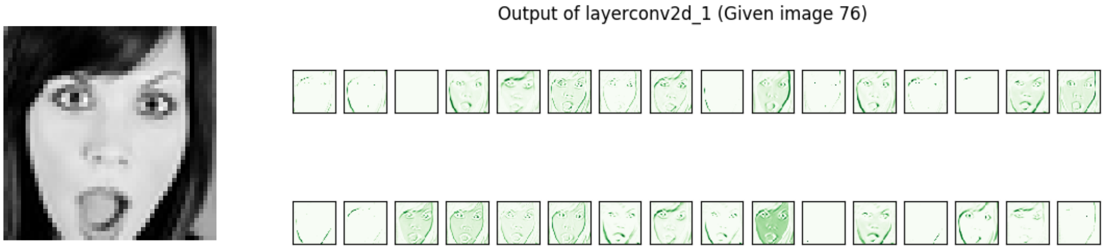


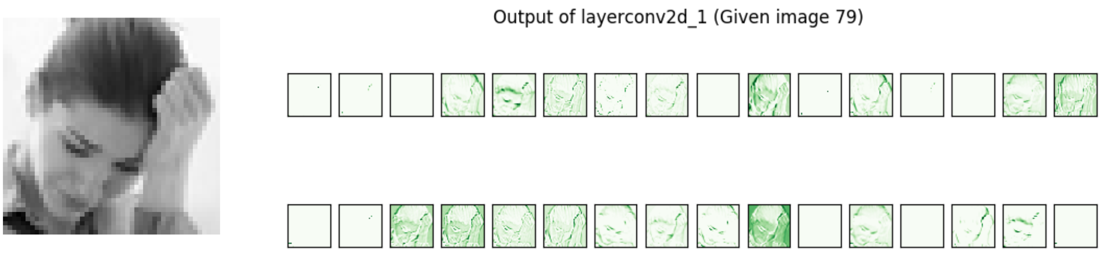
- 分析
  - ❖ 由上圖可發現，model會focus在圖片的眼睛鼻子嘴巴
- Code : *saliency.py*

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。









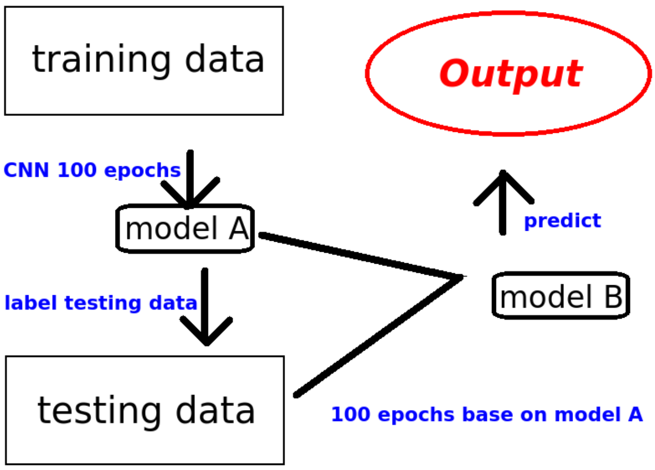
- 觀察
    - 不同的filter會有不同的篩選標準
    - 由上面幾張圖可以發現第26個filter個別容易被activate
    - 經過嘗試很多張照片，對於我的model，每一種類型的照片似乎activate相同的filter
    - 某些filter會將五官凸顯出來，不同的filter凸顯不一樣的部位，例如第31個會找出眉毛
- Gradient Ascent

```
for i in range(200):
    loss_value, grads_value = iterate([output])
    output += grads_value
```

- Code : *layer.py*

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

- 使用第1題的model-A將test data標上label，並使用所有得資料一起產生model-B
- 使用model-B預測test data (self training)



- 準確度：Public test data: 66.06%

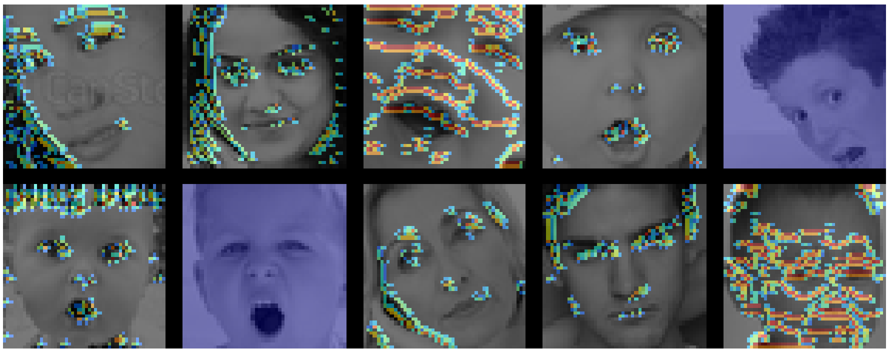
    - ❖ 此model做semi-supervised learning對於準確度的提升沒有幫助
- Code : *semi.py*

[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%]

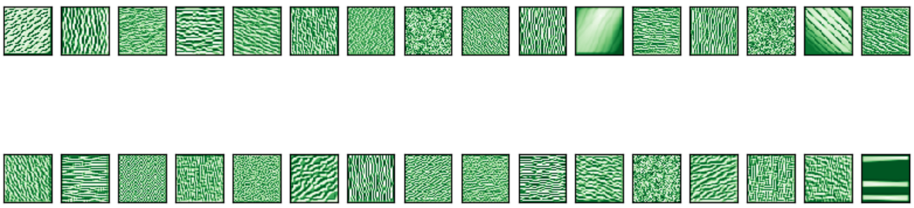
- **You can use other model with poor performance to see what is the difference**

- 使用public正確率59%的model

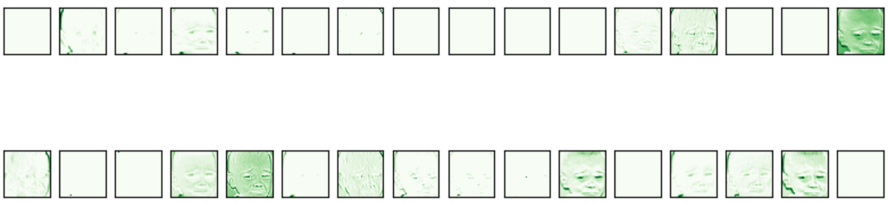
Saliency map



Filters of layer conv2d_1 (# Ascent Epoch 200)



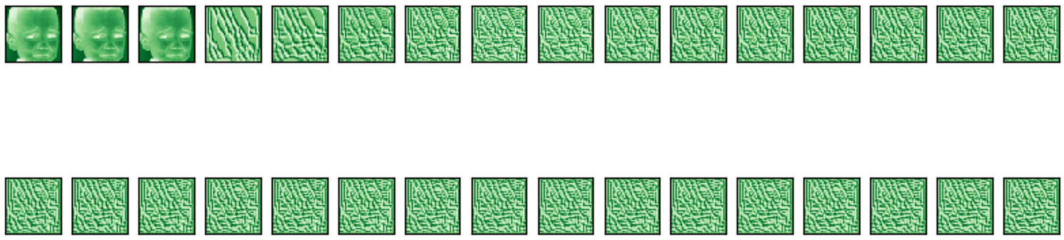
Output of layerconv2d_1 (Given image 78)



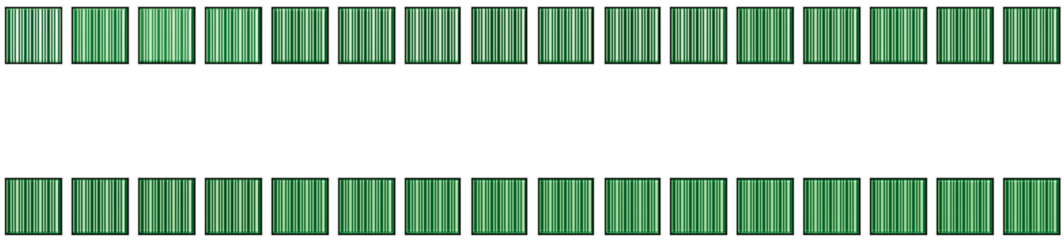
- 比較
    - ❖ 透過Saliency map，可以發現這兩個model focus的點不同，59%的model比較無法抓出五官
    - ❖ 透過output第一層的比較可以發現兩個model被activate的filter不同

- **You can start from natural image (not white noise), and try to create the adversial image**

Filters of layer conv2d_1 (# Ascent Epoch 200) (Given image 78)



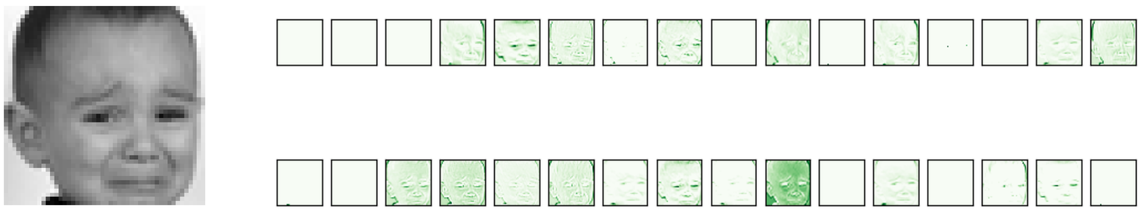
Filters of layer conv2d_1 (# Ascent Epoch 200) (Given image 79)



- 觀察
  - ❖ image78的左上角3個filter皆可以看出人臉，估計是那幾個filter沒有被activate

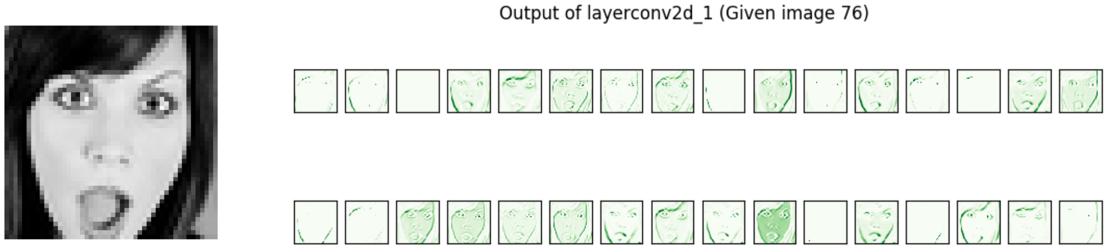
- **you can also try to find which image will activate the specific class the most**

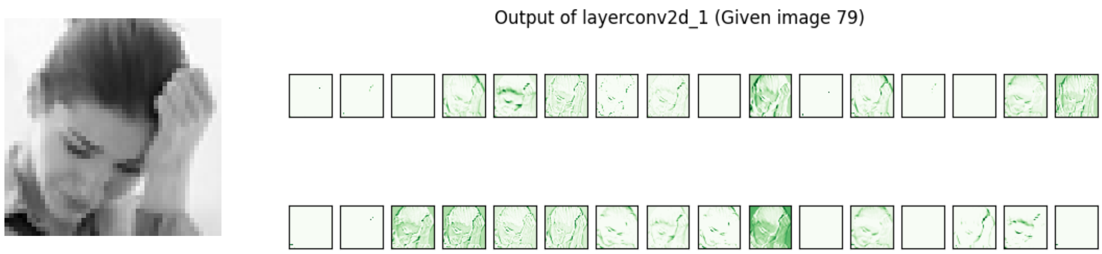
Output of layerconv2d_1 (Given image 78)

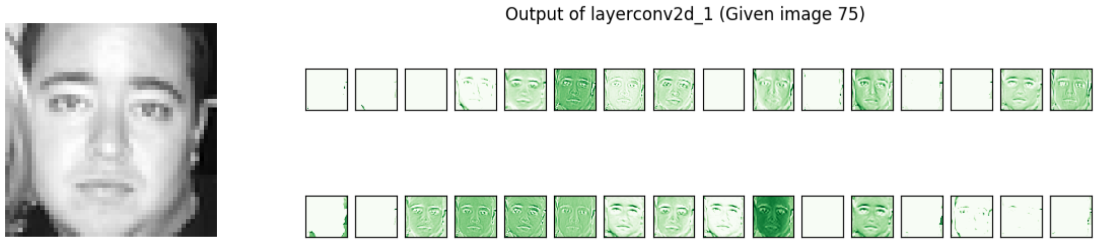


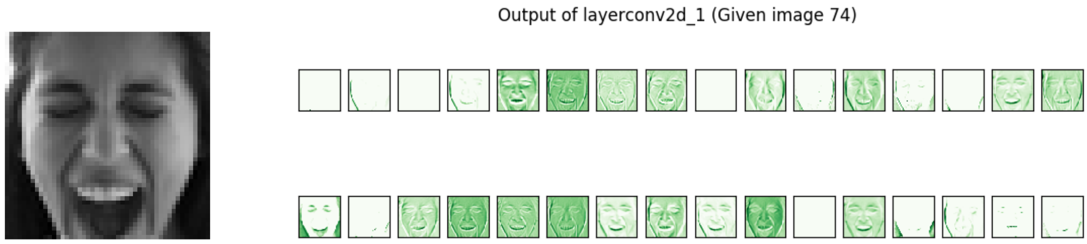
Output of layerconv2d_1 (Given image 77)

Output of layerconv2d_1 (Given image 76)









- 觀察
  - ❖ 左下角的filter感覺最容易被驚嚇類型的照片activate