

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：

如下圖所示(由左到右)，依序經過 Conv2D、MaxPooling、BatchNormalization、Dropout，其中 activation 皆使用 selu。

Conv2D(): 第一層 Conv2D 的 kernel size 為(5,5) 其餘的皆為(3,3)，strides 為(2,2)，filter 依序為 64、128、256、512。

MaxPooling: pool\_size 與 strides 皆為(2,2)

BatchNormalization: 皆為預設參數，可以加速訓練，加快模型收斂速度。

Dense: unit 依序為 512、256、128、64、7。

Dropout: rate 依序為 0.3、0.35、0.4、0.45、0.45、0.45、0.45、0.5。

從 model.summary() 中得知，參數總量為 1,990,279。

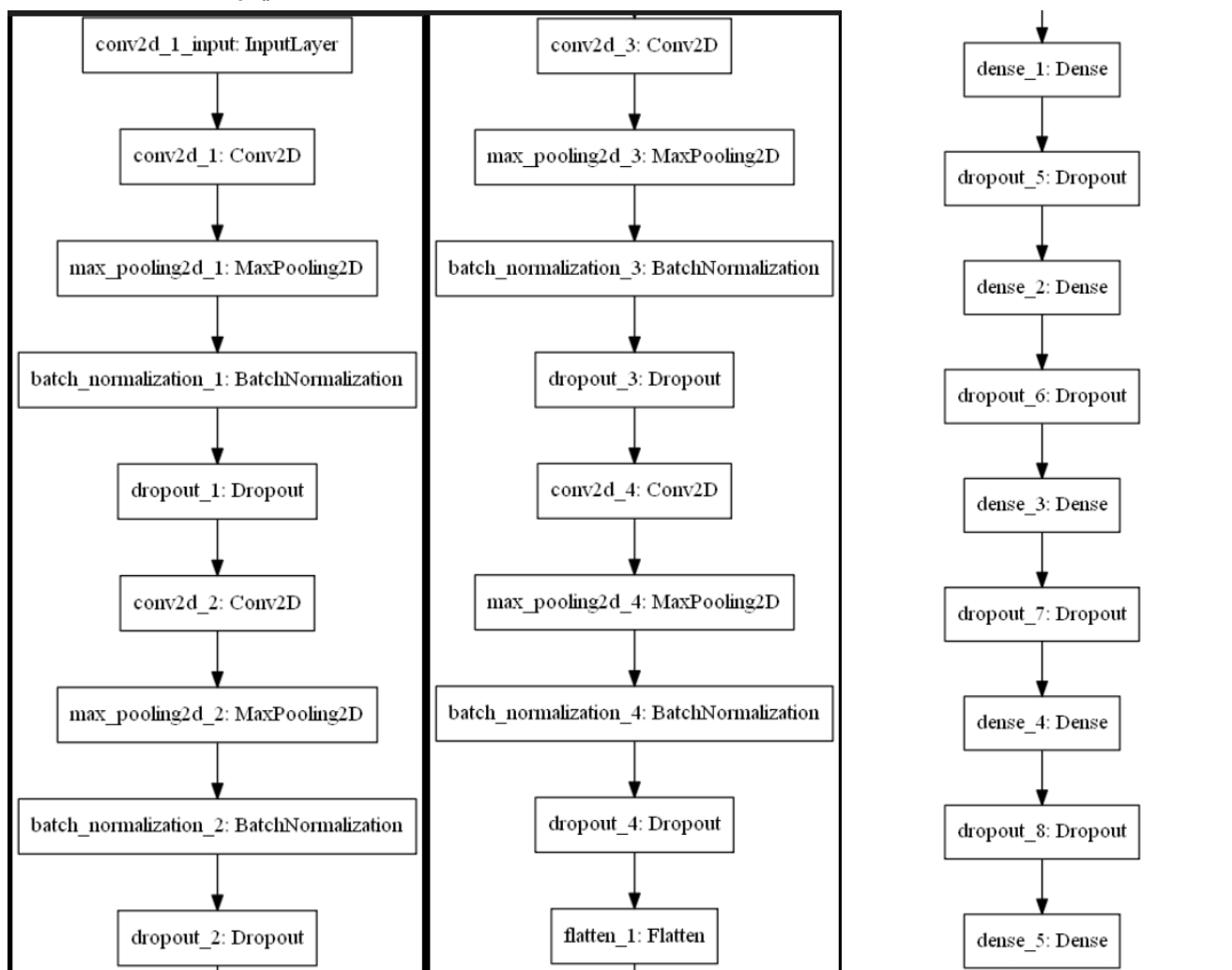


Figure 1: CNN 架構

首先我將 training data 除以 255 並做標準化，將最後的 5000 筆資料取作 Validation data，使用 keras.preprocessing.image.ImageDataGenerator 將圖片上下或左右平移、旋轉，曾經嘗試過將圖片放大、翻轉等等，但訓練效果會變差，推測是增加了太多奇怪的圖片使得 noise 增加，或只是我參數 tune 得不夠好。

配合 ImageDataGenerator，訓練時使用 fit\_generator()，batch\_size 為 128，steps\_per\_epoch 為 data 數量除以 128，loss function 為 cross entropy，optimizer 為 adam。

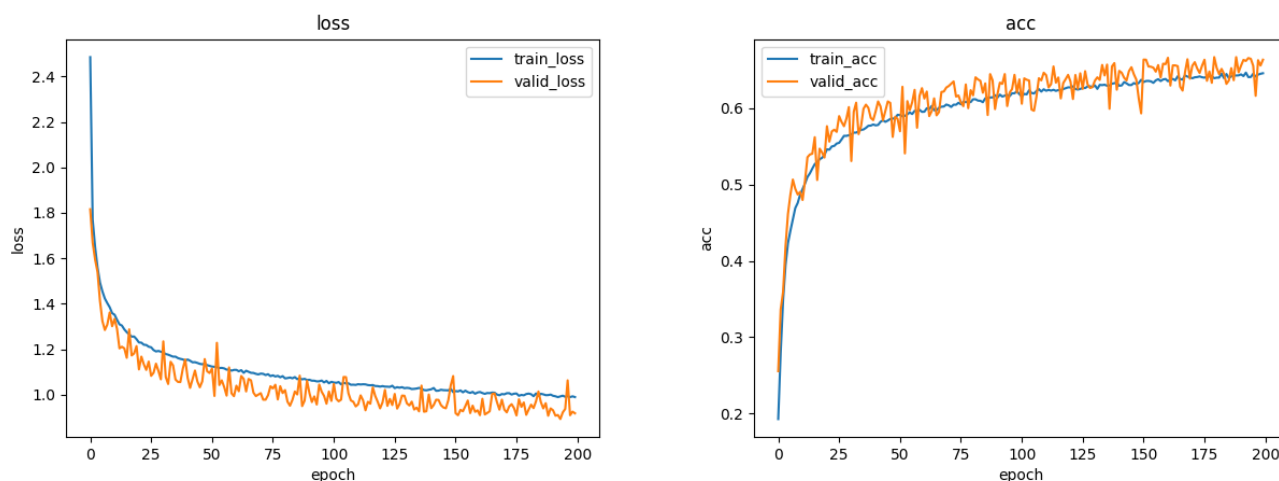


Figure 2:CNN Training 過程

從圖中可以觀察到雖然 Validation 的 loss 和 acc 在趨勢上有穩定的變好，但有很大的震盪，曾經嘗試過增加 `step_per_epoch`，增加每 epoch 訓練的資料量，雖然有使曲線平坦許多，但訓練時間變長，且效果沒有顯著提升，故不採用。

在訓練過程中 Training loss 趨勢高於 Validation loss、Training acc 趨勢低於 Validation acc，推測是因為 Training 時有 dropout，有些 neuron 沒有用到，或是因為 `ImageDataGenerator`，在 Training data 中產生 noise，使預測效果變差，但在 Validation 上沒有這個問題，所以預測效果較好。

Validation loss 最低為 0.893117，Validatoion acc 最高為 0.666600。

Kaggle Public 上最高的結果，透過 7 個 Validation acc 為 66.8~70.0 之間的 model 投票取得，訓練的 CNN 參數、activation、batch\_size、steps\_per\_epoch 皆不進相同，以此平均他們各自的 bias 得到更好的預測效果。

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: )

答：

如下圖所示。

Dense: unit 依序為 768、256、256。

Dropout:依序為 0.3、0.35、0.35。

從 `model.summary()` 中得知，參數總量為 2,034,695，與 CNN 的參數量相近。

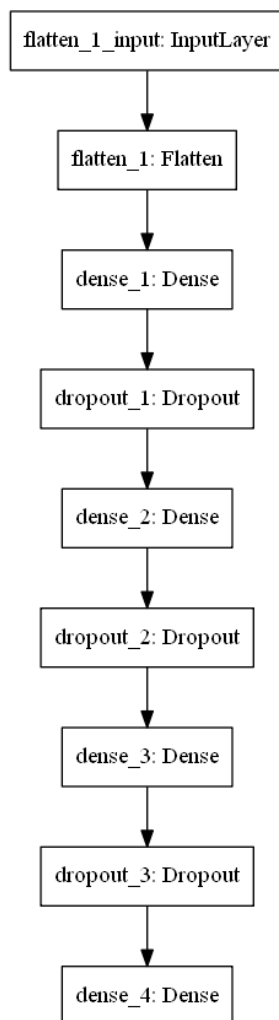


Figure 3:Dnn 架構

資料處理與 CNN 相同。

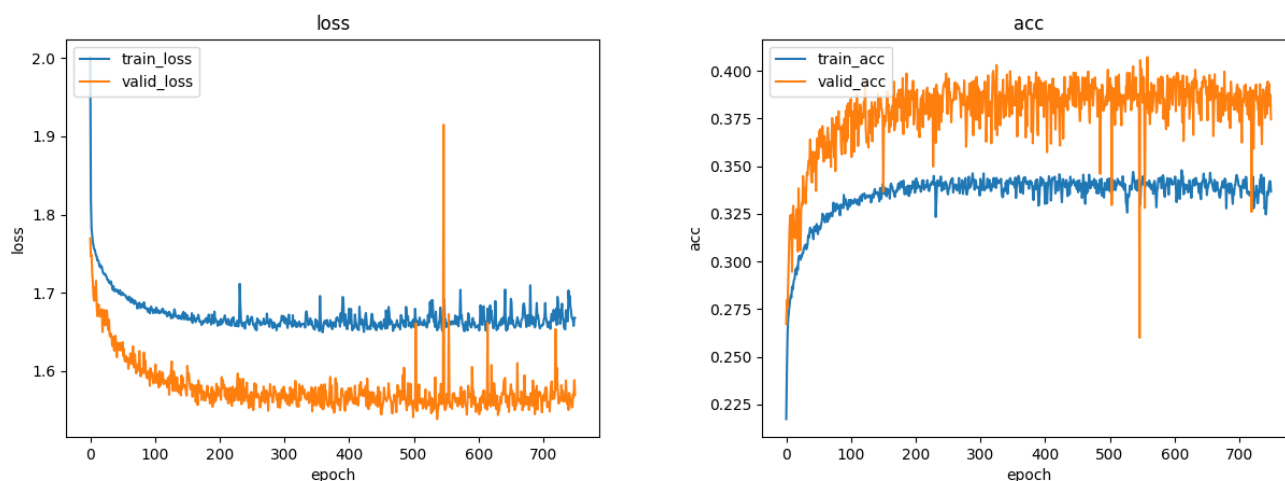


Figure 4:DNN Training 過程

DNN 的收斂速度遠不如 CNN，故我跑了 750 個 epoch，讓 DNN 收斂。

在相同 epoch 時 DNN 的 acc 或是 loss 完全比不上 CNN，且震動幅度比 CNN 大了許多，可以看出 DNN 的穩定度低了不少，且收斂到的 acc 和 loss 也很差。

儘管參數差不多，DNN 的訓練速度上比 CNN 快了好幾倍，應該是因為少了 Conv2D 取 filter 等等的複雜運算所致。

和 DNN 相同，訓練過程中 Training loss 趨勢高於 Validation loss、Training acc 趨勢低於 Validation acc，推測是因為 Training 時有 dropout，有些 neuron 沒有用到，或是因為 ImageDataGenerator，在 Training data 中產生 noise，使預測效果變差，但在 Validation 上沒有這個問題，所以預測效果較好。

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: )  
答：

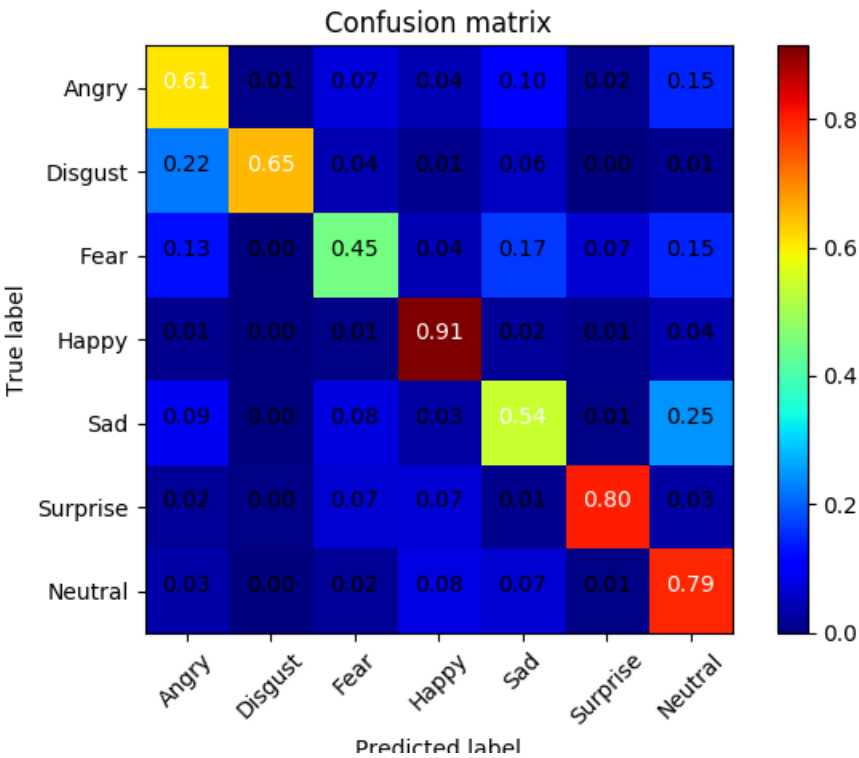


Figure 5:Confusion Matrix

從上圖中可以發現我的 model 在 predict Neutral 和 Sad 時有最大的錯誤率，故探討這兩個 Class。

下圖為我的 model predict 錯誤的兩張圖(從 Validation 取出)，這兩張圖對我來說也模稜兩可，兩者的特徵差異並不明顯，故較難區分。

下表為兩圖預測機率的表格，可以看出左圖雖然 Neutral 最高，但 Sad 的機率是次高，且和其他 Class 的機率相差一個數量級以上，可以得知 model 還是有稍微偏向 Sad。

右圖的 Neutral 和 Sad 的機率差異不大，我覺得嘴角沒上揚看起來像 Sad、眼神呆滯看起來像 Neutral，同時符合 Neutral 和 Sad 的特徵，猜測 model 抓到嘴角的特徵，才會預測為 Sad。

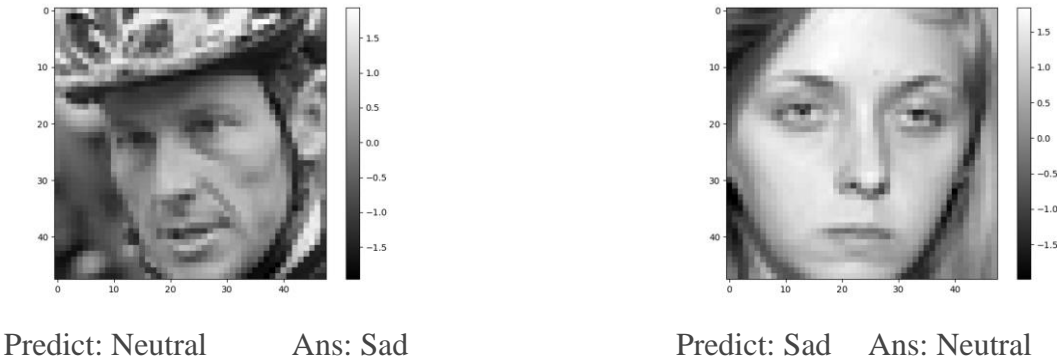


Figure 6: Mistakes

	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Left	0.0422	0.000649	0.0453	0.011	0.110	0.00895	0.782
Right	0.05	0	0.14	0.02	0.47	0	0.31

Probability of Figure 6

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:b04902002 許志軒(上學期修課學生))

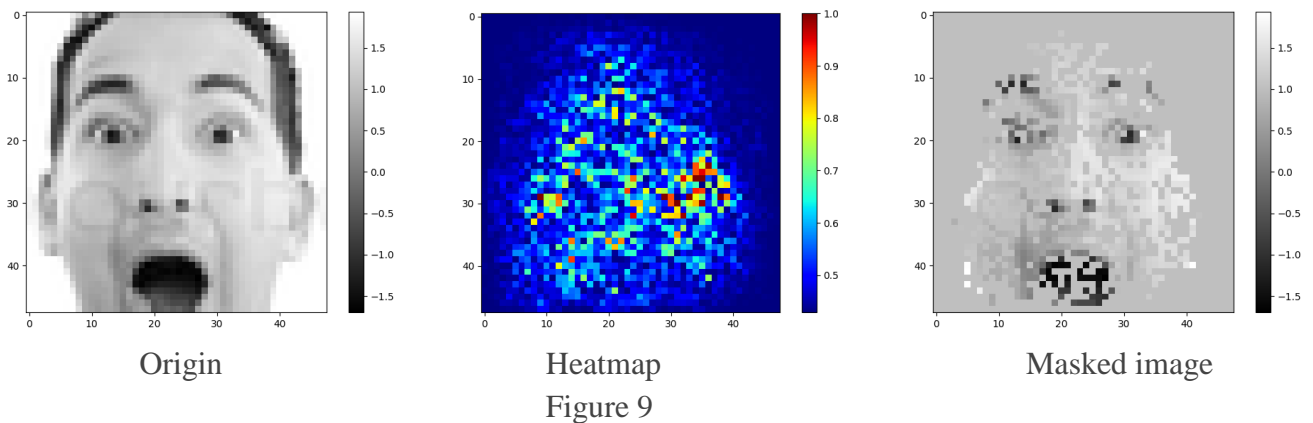
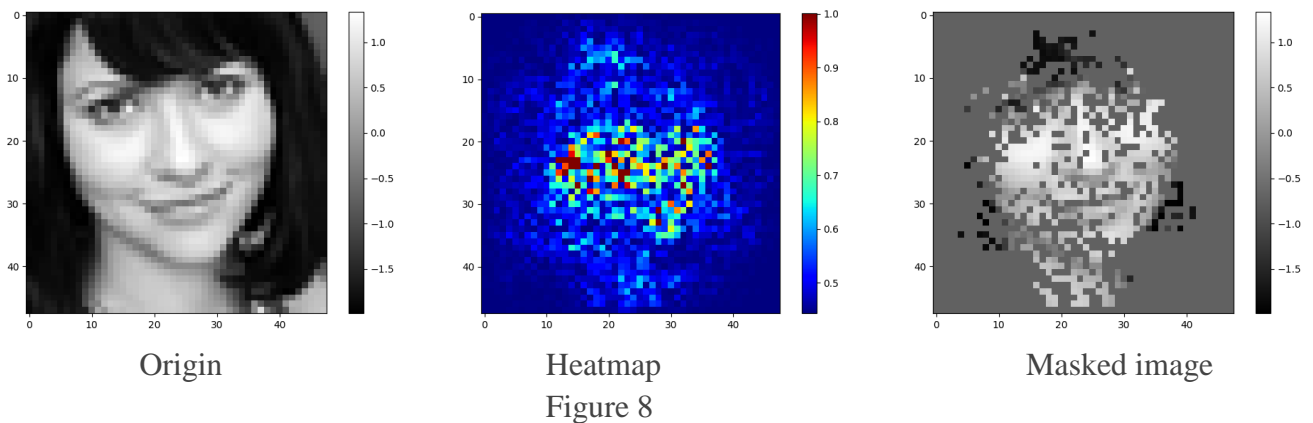
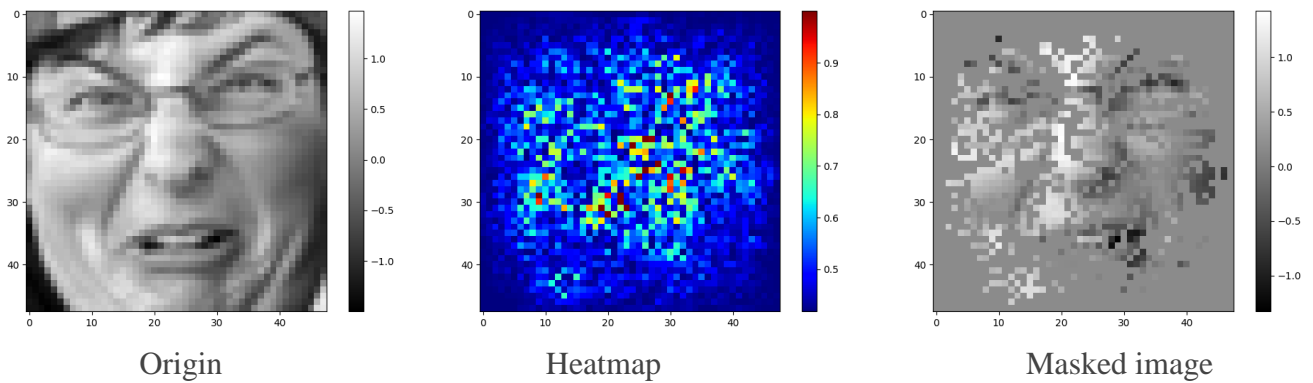
答：

Figure 7~9 是從 Validation data 中選出，Disgust、Happy、Surprise，且我的 model 的預測都正確。

Figure 7 的 Heatmap 在臉部表情及眼睛的部分有較高的值，推測 model 是由表情緊繃加上厭惡的眼神做判斷。

Figure 8 的 Heatmap 在臉頰部分有較高的值，推測 model 是由笑的時候臉頰肌肉會往上收縮作為判斷依據。

Figure 9 的 Heatmap 也是在臉頰部分有較高的值，推測 model 是以驚訝的時候嘴巴張開導致臉頰往下拉長做判斷。



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: )

答：

使用 figure 8 的原圖做為 input。

如同助教在課程網站上的 **tutorial** 所說，在同個 **layer** 之中，有許多 **filter** 都是一樣的，只是加上了旋轉，其中又以旋轉  $90^\circ$  最多。因為每次拍攝的角度都不進相同，加上每次人的五官位置會因表情等等因素而改變位置，所以為了能夠辨識圖片，**filter** 會以不同角度抓取特徵。

Figure 10，由於是經過 **MaxPooling** 的第一層，剛開始只抓取圖片的方向顏色等等，所以 **filter**，都還是比較簡單的圖形。

而 Figure 11，已經是在 **MaxPooling** 的第四層，經過層層堆疊，已經變得複雜許多，可以發現大部分都有清楚的人臉輪廓。

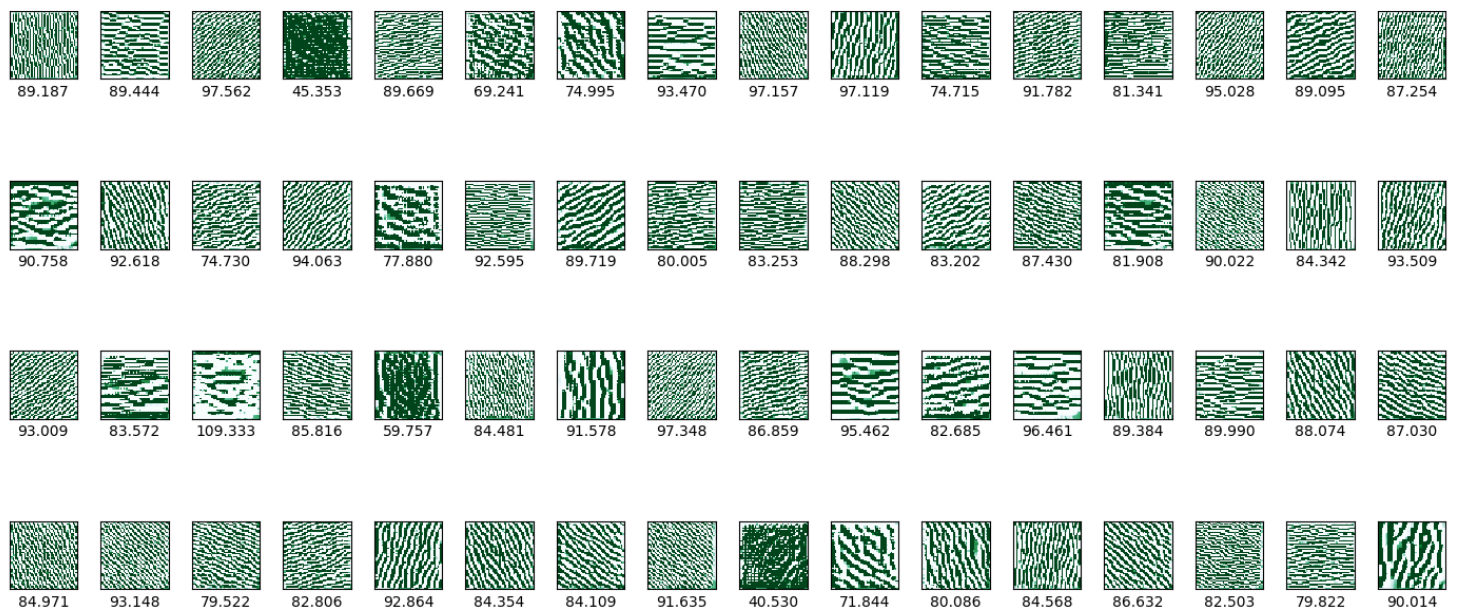


Figure 10:MaxPooling2d\_1

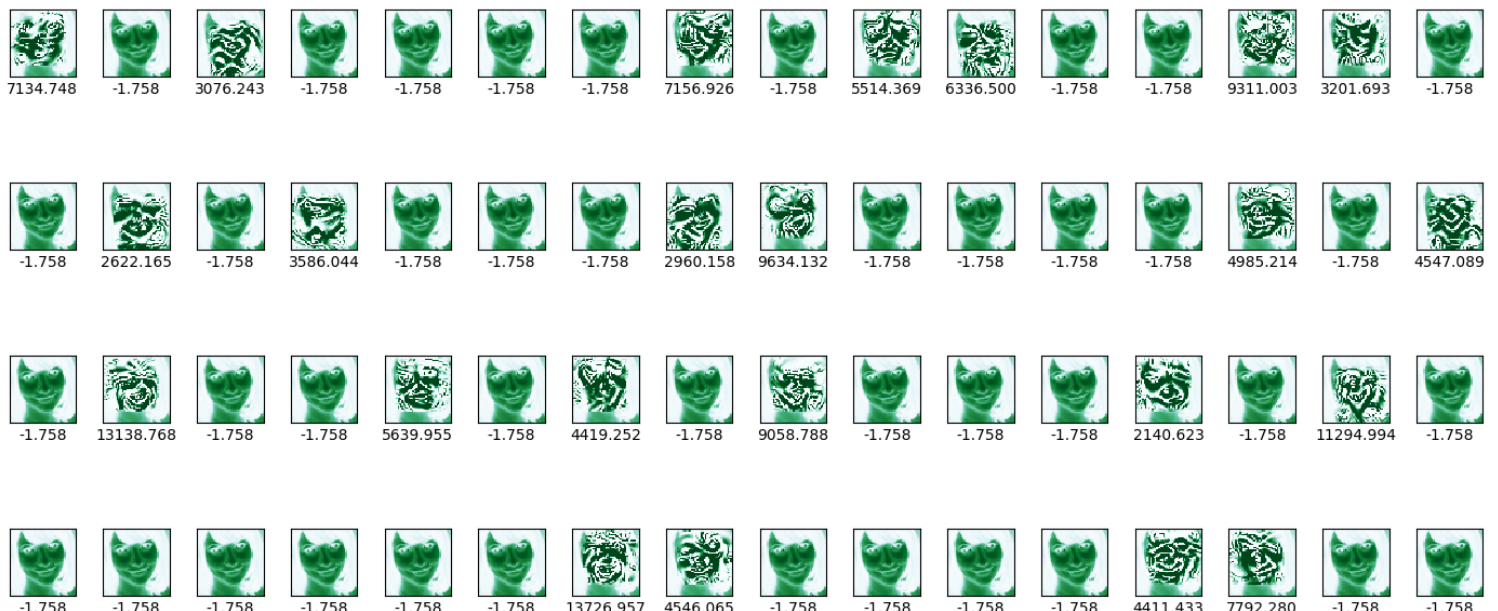


Figure 11:MaxPooling2d\_4