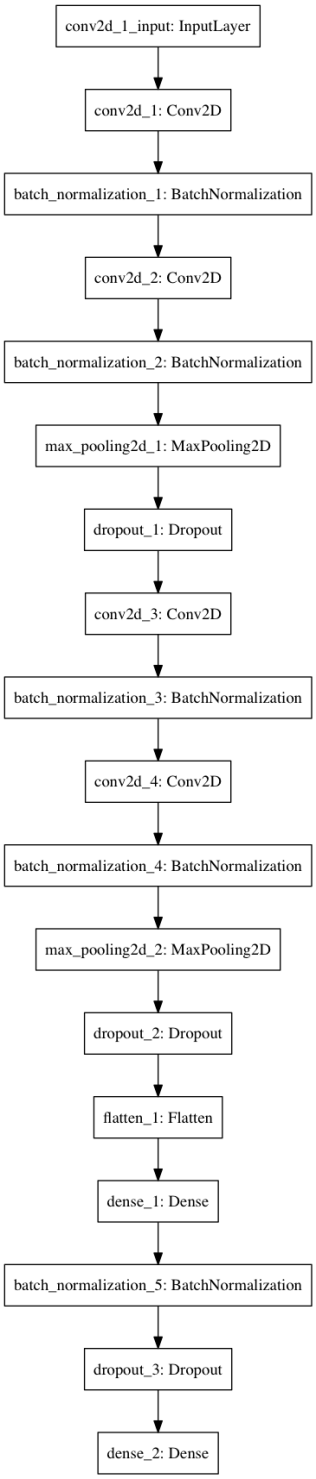
學號：B03902090 系級： 資工三 姓名：邵楚荏

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：

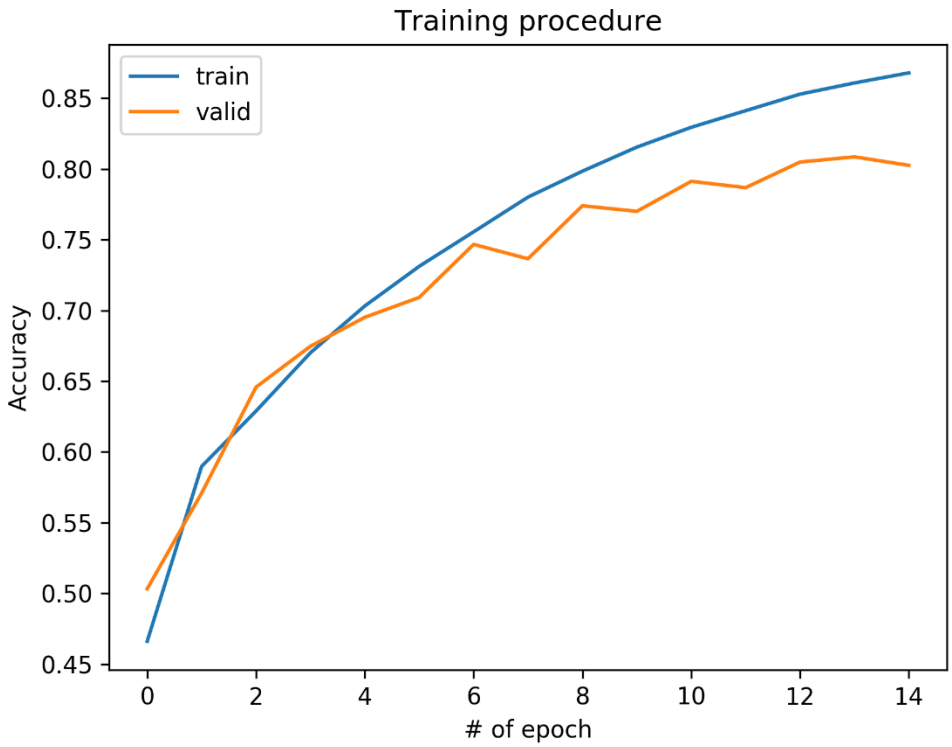
Model Structure:



模型架構上，我用了四層的 convolution2D 以及兩層的 Dense，其中在每兩層的 convolution2D 以及第一層的 Dense 底下加了 BatchNormalization，這可以幫助在 training 的時候更快達到收斂的效果。另外，除了最後一層的 softmax 之外，美一層的 activation function 我都是使用”relu”，optimizer 則是使用”adam”。

除此之外，因為原本的 data 數量太少，所以我使用了 keras 提供可以自動生成 image data 的 API: “Imagegenerator”，將原本的 image 做了橫移以及縱移還有水平翻轉之後，將原本將近三萬筆的 training data，擴增到將近 20 萬的 training data，將 data 擴張成 20 萬後，可以輕鬆解決原本 training data 數量不足的問題。

Training procedure:



訓練過程中，我將 batch_size 設為 200，也就是說跑 200 個 data 之後更新一次參數，另外我在 training 的時候設了 checkpoint，將 validation accuracy 表現最好的 model 存起來，在 testing 的時候 load 近來使用，作了很多次之後發現，大約將 epoch 設為 15 左右就可以在 training 當中得到最好的結果，上圖為 15 個 epoch 中，所得到的 training accuracy 以及 validation accuracy，validation 在 15 個 epoch 之後，都不會再有進步。

這個 model 傳上 Kaggle 所得到的 public accuracy 為 0.67595，我觀察到這比我的 validation accuracy 低了不少，我想這原因應該是我在生成新的 image data 的時候，

生成的參數不夠多，因此生成的圖片出來都是比較大同小異的，所以在 validation 上可以有很好的表現，但 Kaggle 上的 testing data 有些圖片可能有經過小小的翻轉，甚至是黑白像素不一這些比較特別的圖片，所以這些在 training 中比較沒辦法做到，這是我想為甚麼我在 Kaggle 上的表現比不上我自己做的 validation 的原因。

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:

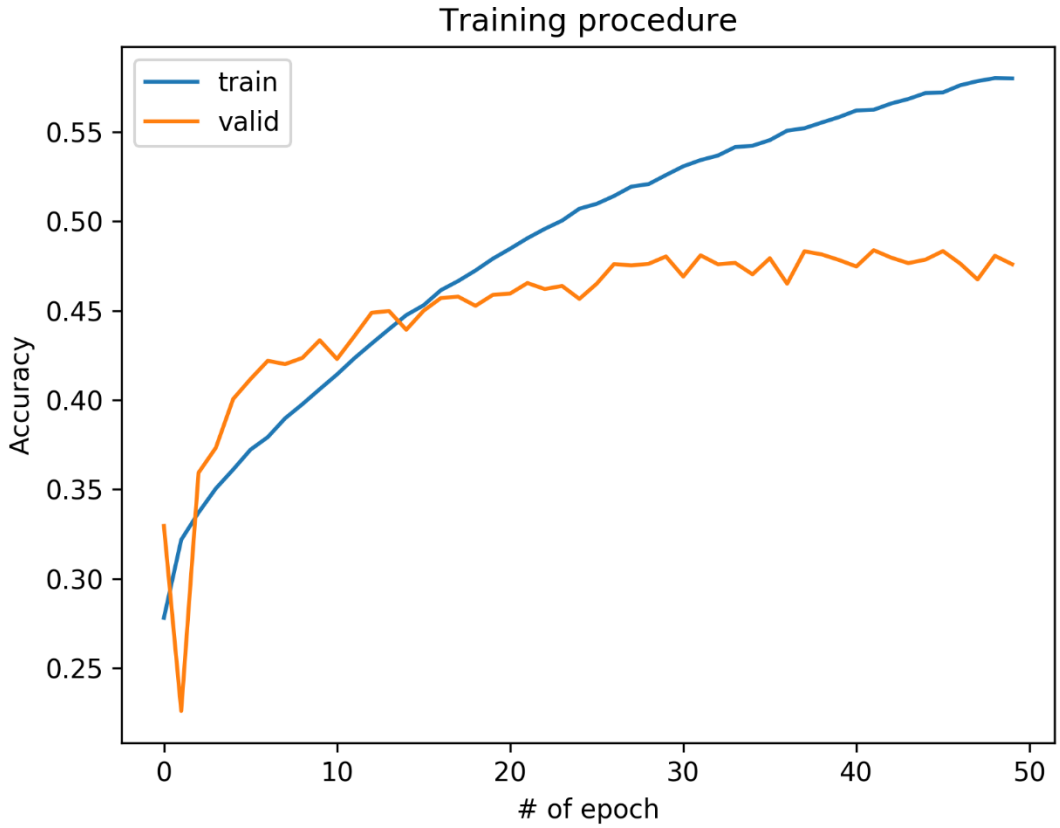
在上題的 CNN model，整個模型的參數量為 5575110 個參數(如下圖):

```
Total params: 5,575,111.0
Trainable params: 5,573,319.0
Non-trainable params: 1,792.0
```

因此，在這題為了與 CNN 參數相近，以下為我設計的 DNN 模型架構，主要是疊了四層的 hidden layer 以及最後 7 個 neuron 的 output layer，另外前四層的 hidden layer 的 neuron 皆為 1024 個。這樣的參數量為下圖所示:5532679 個，與上題的 CNN 相近!

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 1024)              2360320
_________________________________________________________________
batch_normalization_1 (Batch (None, 1024)              4096
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              1049600
_________________________________________________________________
batch_normalization_2 (Batch (None, 1024)              4096
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_3 (Dense)              (None, 1024)              1049600
_________________________________________________________________
batch_normalization_3 (Batch (None, 1024)              4096
_________________________________________________________________
dropout_3 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_4 (Dense)              (None, 1024)              1049600
_________________________________________________________________
batch_normalization_4 (Batch (None, 1024)              4096
_________________________________________________________________
dropout_4 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_5 (Dense)              (None, 7)                 7175
=================================================================
Total params: 5,532,679.0
Trainable params: 5,524,487.0
Non-trainable params: 8,192.0
```

Training procedure:



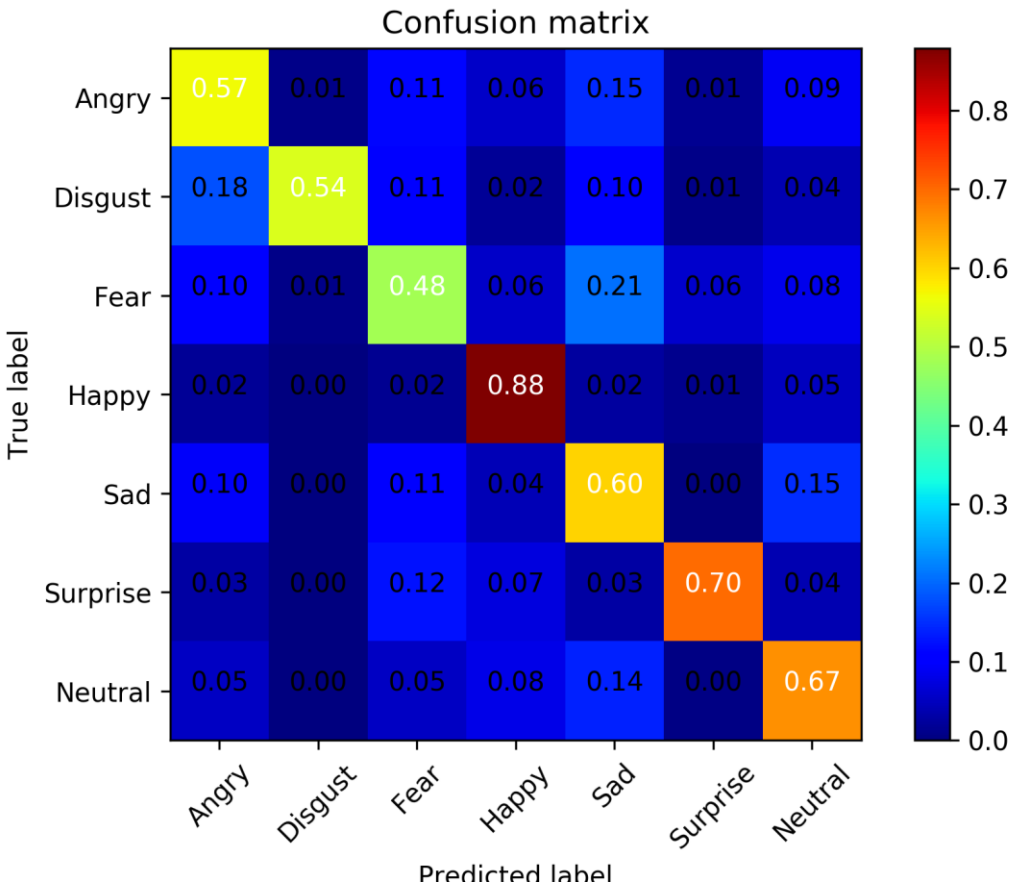
在這題中，我將原本的 28709 筆 training data 後面的 8709 筆當成 validation data 使用，然後利用剩下的 training data 去做 Image generator，生成 20 萬筆新的 training data，訓練過程中，batch_size 一樣跟第一題為 200，另外 epoch 數在經過觀察之後決定設為 50。由上圖可以發現，validation accuracy 大概到 0.50 左右就不會在上升了。

與上題的 CNN 比較，我想 CNN 比較好的原因是因為進行 convolution 的時候，比較能記住圖片中每個 pixel 或者是區域的資訊，所以在每層的 convolution，都可以將特定區域的特徵強化或者弱化，而在這次的 task 中，表情的分類利用 CNN 來實作比較容易在 convolution layer 中就集中於圖片的某些區域來做分類，DNN 則是比較難有此特性，這是我想為甚麼 CNN 的表現比 DNN 來的好的原因。

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：

Confusion Matrix:

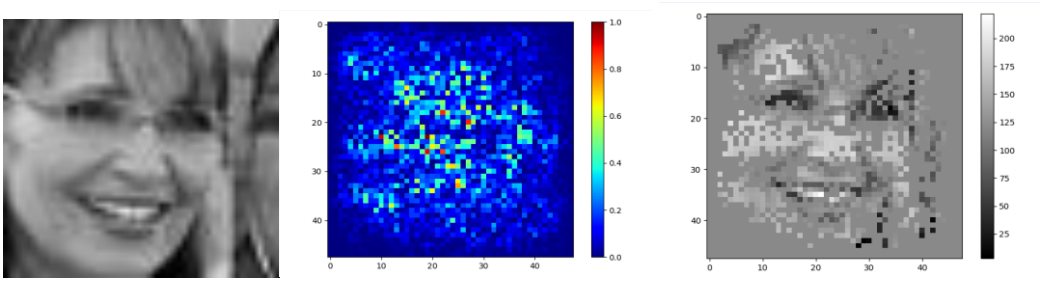


上圖是在 validation set 裡面，所做出來的 confusion matrix，由圖可知，”happy”這個 class 的分辨度最高，最低的則是”fear”，由圖可知，”sad”跟”neutral”兩個 class 容易被混亂，”angry”跟”fear”也容易被搞混，觀察這張圖之後，也可以發現 CNN 對於同樣為”負面”表情(例如 fear, sad, disgust)的辨識度較低，對於”正面”的表情(happy, surprise)的辨識度較高!

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：

下面最左邊的圖是我從 validation set 中，挑出的一張由 48*48 個 pixel 轉成的原圖，而這張 圖片應該被預測成”Happy”的 class。

在經過最後一層 output layer 對 gradient 的計算之後，能印出原圖的 saliency maps(下圖中)以及 Mask 掉 heat 低於 threshold 的部份的圖片(下圖右):

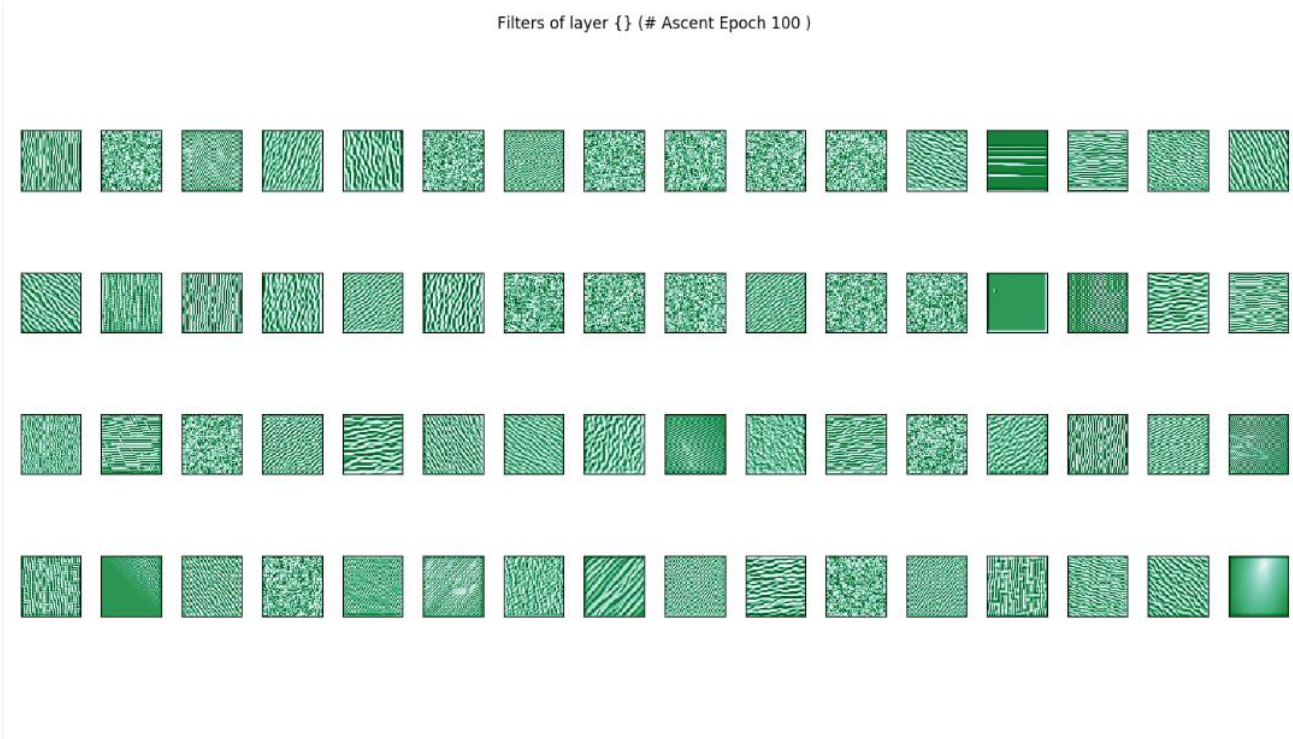


由第二張 heatmap 以及第三張 mask 掉 heat 低於 threshold 的圖片可以知道，模型在做 classification 的時候，會主要集中在人的五官上，其中眼睛以及嘴巴笑容的位置為整張圖片 heat 較高的部分。而最後這張圖片成功被預測為”Happy”的 class!

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：

(Part I):

承第一小題，在這題中，我分析的是 model 裡的第一層 convolution，總共 64 個 filter，每個 filter 大小為(3,3)，所以經過這層之後，原本(48,48)大小的圖片，會變成 64 個(46,46)大小的圖，在第一個部分利用 gradient ascent 得到下列的圖表，可以發現有些 filter 較能被直線性的圖片所 activate，有些則是橫線，另外有些也是散布的點。



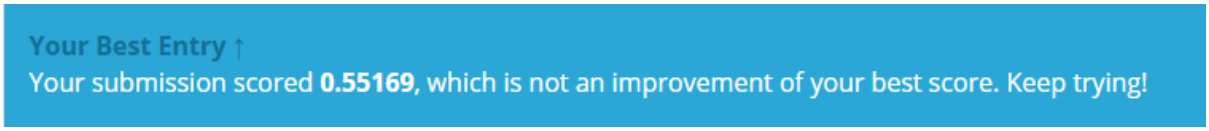
(Part II):

而在第二個部分，我實際從 validation set 中隨機挑選一張圖片，看看他經過第一層 convolution 之後，64 個 filter 分別對這個圖片所做出的影響，觀察到若是 part 1 圖表中越分散點的 filter，對此張圖片的 output 越不清楚(接近白色)，而有些 filter 則是可以成功 focus 在圖片的五官上，特別是眼睛和嘴巴的部位!

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

為了在這題看出 semi-supervised learning 的效果，所以我只使用了最原本 28709 筆 training data 去做觀察，在這題中，我先將最原初的 training data 切成 21000 筆以及剩下的 7709 筆當作 unlabeled 的 data，並且用最剛開始的 21000 筆跑出一個最原本的 model，然後再 predict7709 筆 data，如果在這 7709 筆 data 中機率分布最高的大於 0.8 的話，將這個 predict 出的 label 以及整筆資料加進 training data，之後再 train 一個較少次的 model，當作最後預測 testing data 的 model。最後做完 semi-supervised learning，在 Kaggle 上 public set 進步了將近 0.01 的 accuracy!

尚未做 semi-supervised learning 在 Kaggle 上的結果:



做完 semi-supervised learning 後在 Kaggle 上的結果: