學號：B03901161 系級： 電機四 姓名：楊耀程

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:謝世暐，廖宜倫，周晁德，童寬)

答：因參照網路上 VGG 的設定，架了一個 8 層 convolution + 2 層 fully connect + 一層 softmax 的 CNN。為了避免 overfitting，將 model 改寫成有 dropout，train 的時候用 keras 內建的 data augmentation。

參數設定:

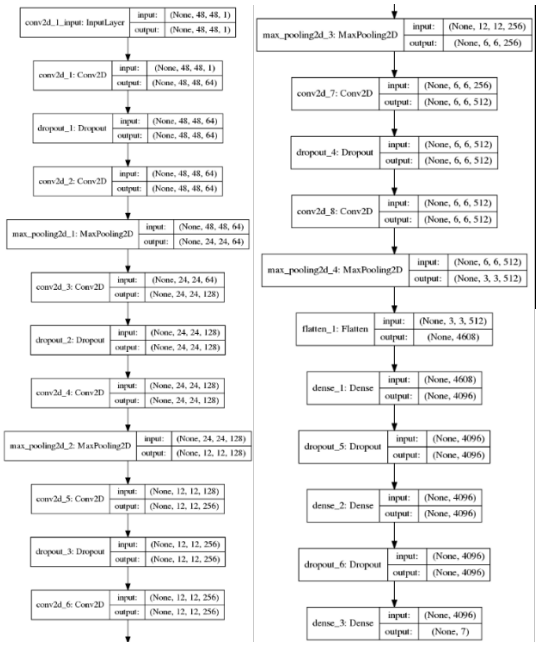
batchsize = 256

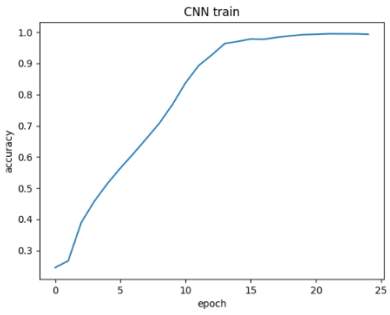
adam optimizer (lr = 0.00003, decay = 1e-5)，有時會微調。

ImageDataGenerator　rotation_range=20, shear_range=0.1, width_shift_range=0.2, height_shift_range, horizontal_flip=True

Dropout = 0.1~0.4

CNN model 架構如下(左圖):





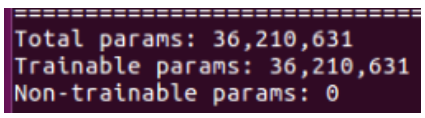
訓練過程與準確率: 上圖(右)是 CNN 的 training 過程(此圖是 no dropout, no data augmentation 的 CNN)，由下圖可以看到 CNN 的 model 在 15 個 epoch training 準確度就達到了 95%以上，幾乎收斂了。另外有 dropout 和 data augmentation 的 model 是分段 train 了多次，才得到較高的 training 準確率(分段 train 的過程在額外討論會提到)。Testing 準確率的部分，在 kaggle public 上準確率= 59.041% (有 dropout 和 data-augmentation 則為 70.660%)，

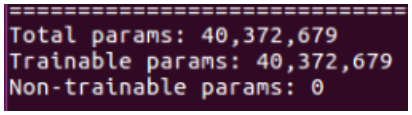
額外討論: 這次我觀察到幾個現象: batchsize 太小，dropout 太多以及 data augmentation 變化太大，會 train 不起來。跟同學討論和參考 paper，某些 paper 遇到類似的情況時，會用之前 train 過的 model 初始化，重新跑 gradient descent，得新的 model，我是用自己 train 的沒 dropout 沒 data augmentation 的 model 做初始化，效果還不錯，就連有 dropout 和 data augmentation training 準確度也達到 94%。

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:謝世暐，廖宜倫，周晁德，童寬)

答：dropout 和 data augmentation 是為了希望在 kaggle 上表現好，此處為了方便比較，就沒有使用這兩者，因此 CNN model 圖可以想成是，第一題扣掉所有的 dropout layer。DNN 的 model 圖如下，基本上是拿掉 conv layer 的同時盡量維持一樣的參數數量，**參數數量比較**表如下:

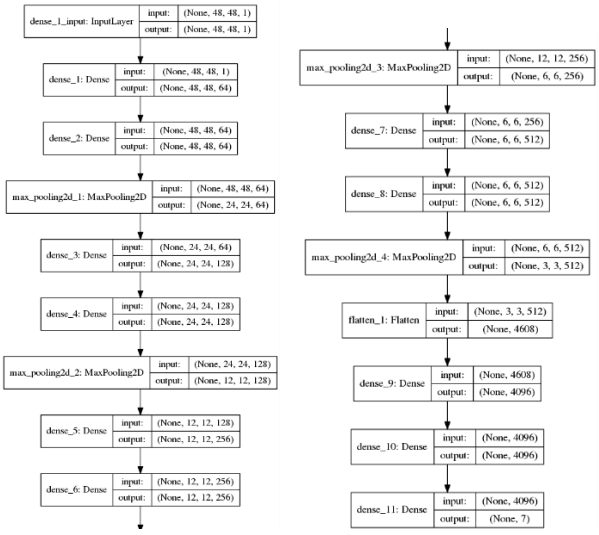
DNN CNN

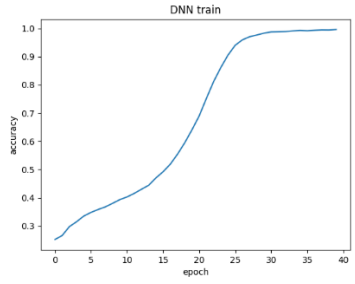




參數設定: 與題 1.的類似，只是沒有 dropout 與 data augmentation

DNN model 架構如下(左圖):





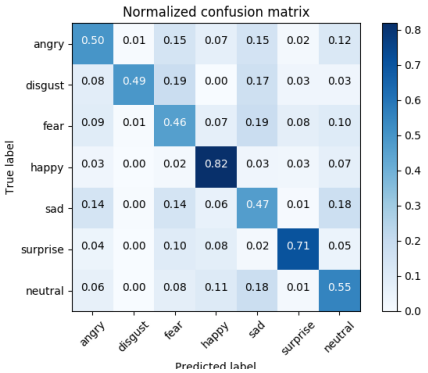
訓練過程與準確率: 上圖(右)是 DNN 的 training 過程，training 準確率收斂在 99%，可是放到 kaggle 上 public 只有 40.707%，應該是 overfitting 嚴重。

比較 CNN 與觀察 ：DNN train 的過程較 CNN 慢，DNN 在前幾個 epoch 準確率上升幅度較慢，在 30 epoch 達到快 99%，而 CNN 前幾個 epoch training 準確率上升快很多，在 epoch 17 左右就達到快 99%

另外 CNN(no dropout, no data augmentation)在 kaggle 上準確率= 59.041%，DNN 在 kaggle 上準確率為=40.707%，可以發現 CNN 有較好的表現，推測是 conv layer 擷取 image feature 的策略很成功，能較 DNN 更有效的找出有效的 feature，也減少了 model 在 testing data 上 overfitting 的現象(因為 conv layer 可以使得在圖片不同處的相同特徵共享同一個 weight)。

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:謝世暐，廖宜倫，周晁德，童寬)

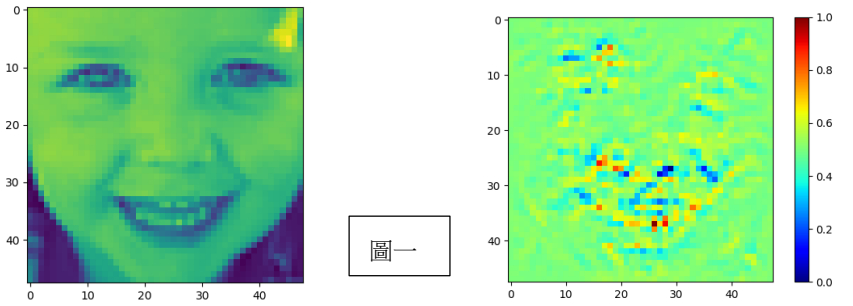
答：利用 10%左右的 validation set 得到的 **confusion matrix** 如下

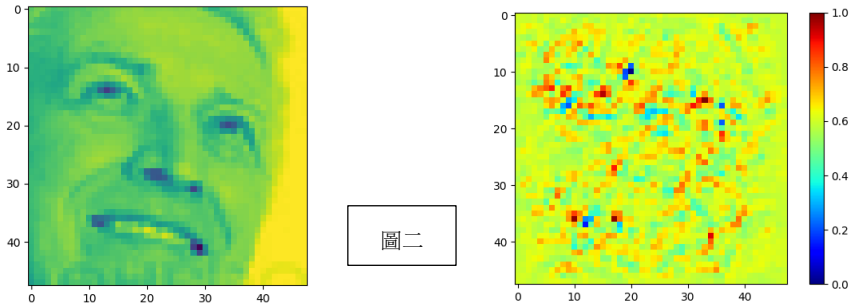


分析與討論: 可以發現 happy, surprise 這兩種情緒被判斷正確的比率是最高的，達到了 82% 和 71%。Angry 最容易被誤判為 fear, sad。Disgust 最容易被誤判為 fear。Fear 最容易被誤判為 sad。Happy 相當不容易被誤判。Sad 最容易被誤判為 neutral，surprise 也不太容易被誤判。Neutral 最容易被誤判為 sad。**推測可能造成這樣現象的原因**為: Angry, fear, sad 這三者臉部皺紋較相近，而且 fear 跟 sad 連人眼看都覺得有點難區分。Happy 跟 surprise 準確率高可能是因為他們的嘴巴特徵較明顯，上揚或是大開。Neutral 和 sad 容易搞混，可能的原因是 sad 的表情不夠明顯時，皺紋沒很多，就被判定為 neutral。

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:謝世暐，廖宜倫，周晁德)

答：從 10%左右的 validation set 中挑照片，以下列兩組範例，左圖為 original image，右圖為 saliency map。因此可以觀察到: 主要是 focus 在眼睛跟嘴巴(圖一較明顯)、皺紋上(圖二較明顯)。

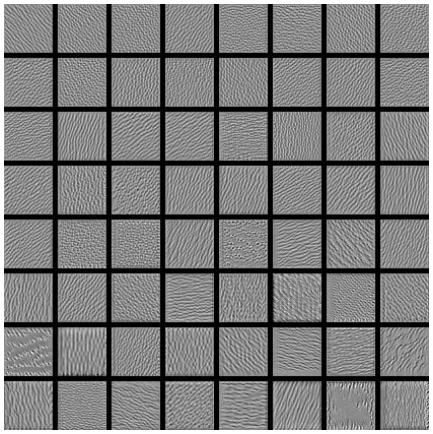




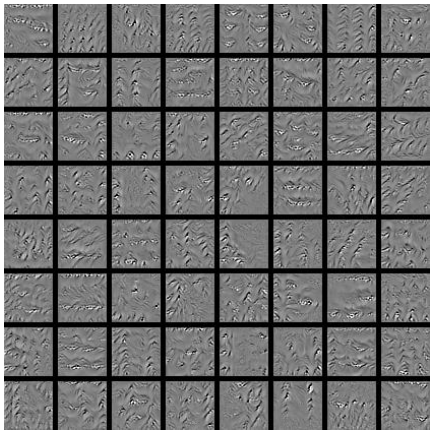
5. (1%) 承(1)(2)，利用上課所提到的 gradient ascent 方法，觀察特定層的 filter 最容易被哪種圖片 activate。
(Collaborators:謝世暐，廖宜倫，周晁德)

答：使用 random noise 做 initialize，將 tutorial 裡面的 code 做一些修改，就可以跑了。利用 gradient ascend 得到某一特定層 loss(gradient ascend 的 loss)前 64 大的 filter，將其拼接為 8x8。我跑了很多層 convolution layer，也嘗試過不同 iteration，篇幅有限，以下只列兩張圖。

Conv2d_3



Conv2d_7



結論: 經驗上來說， 30 個與 150 個 iteration 得到的結果沒差很多。此外，我觀察到用這樣方法 generate 最能 activate CNN filter 的圖片都看不太出人臉的樣子，通常是沿某些方向的波紋跟紋路。另外看了多層 layer 的感覺是: 較接近 input 端的 conv 都是一些細小的紋路，而較接近 output 端紋路則有大的花紋(感覺比較有實體的)，上兩張圖也顯現了這樣的情況，推測這就是不同層 layer 之間有 Maxpooling，造成兩者 scale 不同，因此 activate 不同層的圖片會有上述的差別。