ML hw3 Report

學號：R06921038 系級： 電機所碩一 姓名：謝宗宏

1. (1%) 請說明你實作的 CNN model, 其模型架構、訓練參數和準確率為何？ (Collaborators:)

答：說明模型架構、訓練參數和準確率。寫出一個 -> 0.3 分 兩個 -> 0.6 分 三個 -> 1 分

模型架構：

Input(48, 48, 1)->

2*Conv2D(filter=64)->BatchNormalization->MaxPooling2D->Dropout(0.5)->

2*Conv2D(filter=128)->BatchNormalization->MaxPooling2D->Dropout(0.5)->

2*Conv2D(filter=256)->BatchNormalization->MaxPooling2D->Dropout(0.5)->

Flatten->3*[Dense(unit=512)->Dropout(0.5)]->Dense(unit=7, softmax)

除了 output layer 以外，每個 Conv2D 跟 Dense 都是用 selu 當作 activation。

每個 Conv2D 的 kernel size=(3, 3), stride=(1, 1), padding=same。

每個 MaxPooling2D 的 pool size=(2, 2), stride=(2, 2), padding=same。

使用 nadam optimizer, loss 使用 categorical_crossentropy。

訓練參數：

總共 6400199 個參數，可訓練的有 6396231 個。Epoch: 200, early stopping: 若 validation loss 在 30 個 epoch 內沒有減少則停止. Learning rate=0.002。batch size=128，用百分之十的資料做 validation。

準確率：public: 0.69239 private: 0.69044

2. (1%) 請嘗試 data normalization, data augmentation,說明實行方法並且說明對準確率有什麼樣的影響？ (Collaborators:)

答： 寫出實作 data normalization 過程、與實作前、實作後準確率。 ->0.5 分 寫出實作 data augmentation 過程、與實作前、實作後準確率。 -> 0.5 分

實作過程：

augmentation: 使用 Keras 的 ImageDataGenerator，rotation=20, width shift=0.1, height shift=0.1, random horizontal flip。

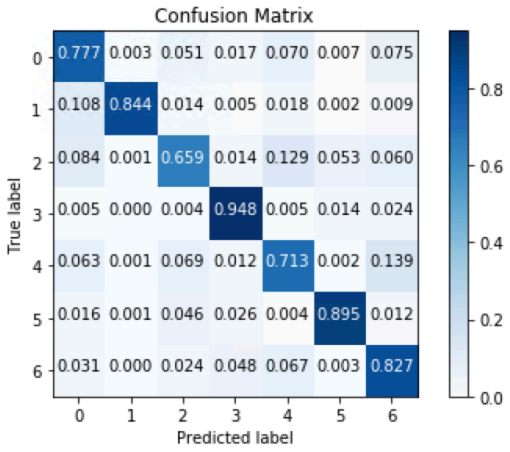
normalization: 分成 featurewise 跟 samplewise 實作，我將每個 pixel 當作一個 feature，因此 featurewise 就是把整個 training set 沿著每個 pixel 位置標準化成 mean=0, stddev=1。而 samplewise 則是將每張圖片都標準化成 mean=0, stddev=1。(以上都可用 np.mean, np.std 指定 axis 完成)，對於整張都是零的圖片則全部設成 0。

準確率：

baseline: val_acc:0.6322 val_loss:1.3009 public: 0.65617 private: 0.65338

data augmentation: val_acc:0.6632 val_loss:0.9030 public: 0.66118 private: 0.66174
featurewise normalization: val_acc: 0.6437 val_loss: 1.2757 public: 0.64809 private: 0.64641
samplewise normalization: val_acc:0.6559 val_loss:1.2500 public: 0.65004 private: 0.65840
samplewise+featurewise: val_acc:0.6461 val_loss:1.2830 public: 0.65394 private: 0.65059
baseline 方法是沒有使用任何 normalization 與 augmentation 的版本。
以 baseline 作為每個方法實作前的基準來比較：
使用 data augmentation 明顯可以提高 performance ，public 跟 private 都從 0.65 提高到 0.66。
使用 featurewise normalization 的分數反而下降，有可能是因為受到 training data 中為數不少的 outliers 影響，實際用 outlier detection 時發現有許多全 0 的圖片且 label 不一致，也有許多不是人臉的圖片混在其中。這些 noise 可能透過 featurewise normalization 影響到整批資料的 normalization 結果。
使用 samplewise normalization 看起來影響不大，(但實際使用上似乎有一點幫助)，在 public 的分數比實作前低，但在 private 上較高。

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：貼出 confusion matrix -> 1 分

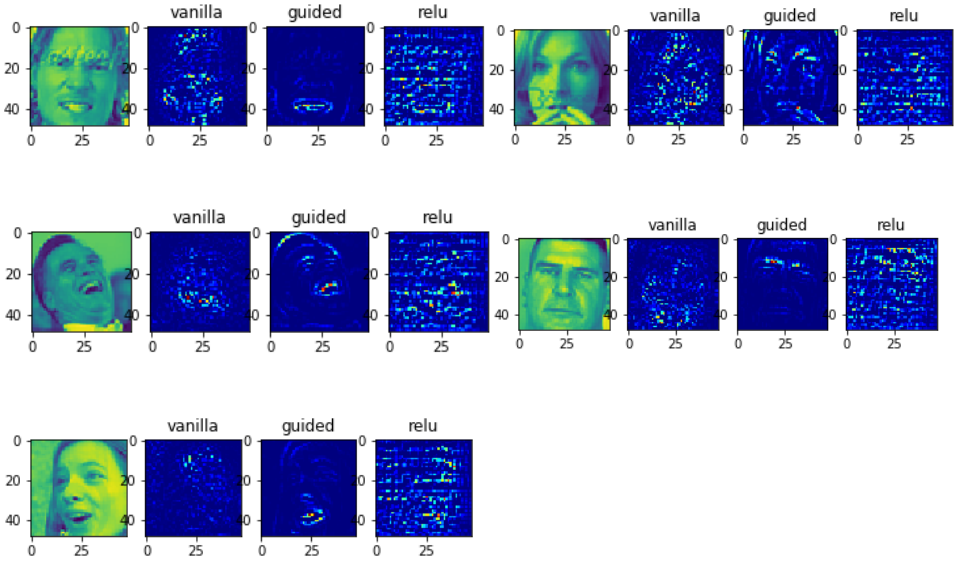


圖為 normalized confusion matrix。可以看出主對角線上的數值都在 0.65 以上，顯示出 model 確實有一定程度的分辨能力，其中以 class=3 的判斷正確率最高，class=2 較差。較容易混淆的狀況為：將 class=4 判斷成 6、class=1 判斷成 0、class=2 判斷成 4。

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:)

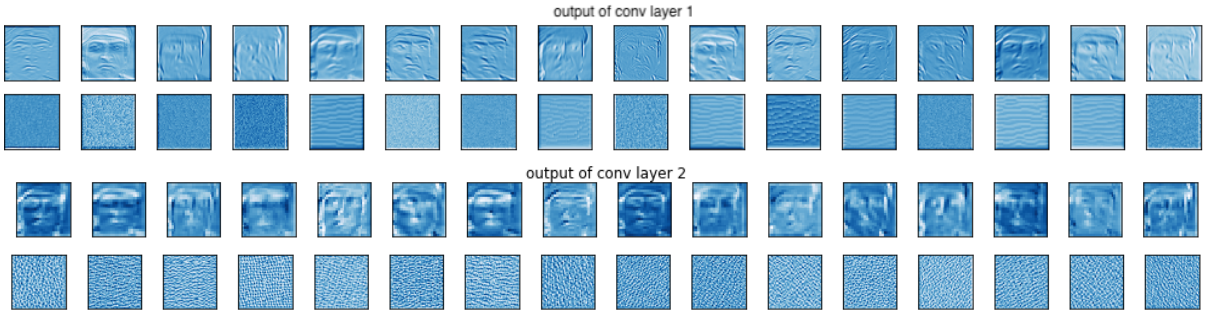
答：合理說明 test 的圖片和觀察到的東西 -> 0.5 分 貼出 saliency 圖片 -> 0.5 分



圖為每張 image (scaled to [0, 1]) 及其對應由三種不同的 backprop modifier 得出的 saliency map。僅管不是很明顯，但是可隱約看出 model 主要 focus 的部分大概就是人的五官，尤其是眉毛、眼睛、和嘴的部分，這確實和人類判斷表情時關注的部分類似。

5. (1%) 承(4) 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 與觀察 filter 的 output。
(Collaborators:)

答：合理說明 test 的層數和觀察到的東西 -> 0.5 分 貼出 filter input and output 的圖片 -> 0.5 分



上圖分別為第一、二層的 filter output，與對應層數的 filter 經過 gradient ascent 後的樣子。
layer1 可以較明顯看出人臉的形狀跟表情，其對應到的 filter 則是偏向單純的線條一類的 texture。而 layer2 可以發現人物只剩下比較隱約的輪廓，其對應到的 filter 所取的也變成比較複雜一點的 texture。