學號：B04901074 系級： 電機三 姓名：吳倉永

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:)

Conv 架構 **PReLU as activation**	**Dense 架構** **LeakyReLU as activation with alpha= 0.05**
Conv2D(64, (3, 3)) batch-normalization	Dense(512) batch-normalization dropout(0.3)
Conv2D(64, (3, 3)) batch-normalization, dropout(0.2)	Dense(512) batch-normalization dropout(0.3)
Conv2D(64, (3, 3)) batch-normalization Maxpooling(2, 2) dropout(0.25)	Dense(128) batch-normalization dropout(0.5)
Conv2D(128, (3, 3)) batch-normalization	Dense(128) batch-normalization dropout(0.6)
Conv2D(128, (3, 3)) batch-normalization Maxpooling(2, 2) dropout(0.3)	
Conv2D(256, (3, 3)) batch-normalization dropout(0.4)	
Conv2D(512, (5, 5)) batch-normalization Maxpooling(2, 2) dropout(0.4)	

• **CNN 架構**: 左表格是我的架構。
a. **Conv2D()**：除了最後一個的 kernel size 為 (5,5)，其餘皆為 (3,3)，padding 都是 “same”。
b. **Activation**：如左圖示
c. **Max-pooling**：如左圖
d. **Dropout**：如左圖，採取漸進式 dropout。
e. **Batch-normalization**: 加快收斂速度，讓 model 不 overfitting。

• **訓練過程：**
a. Data preprocessing: 先取 10%當做 validation set 使用 np.flip 鏡像圖片，產生 2 倍 data 量，大約 51000 張 train data 及 2500 張 validation data。之後使用 Keras 的 image-generator 產生平移(shift)、tilting、縮放的各種圖片，讓模型的抗噪能力增強，避免 overfitting。
b. 訓練設定: batch-size 設 128 且 1 個 epoch step 為 403，因此共 51584 筆資料。Loss 是 cross-entropy，使用 adam 作為 optimizer。使用 check-point 跟 early-stopping(patience 70，想讓他多看一點，硬體許可)，並使用 validation 最好的 model 進行 predict。
c. #參數: 14,357,255 個可 train 參數。

- **準確率分析:**
    a. **漸進式 Dropout:** 一開始不 dropout 太多，否則從剛開始的靠近 input layer 的參數很容易壞掉，造成後面也壞掉。實驗結果顯示漸進 dropout acc: 70.6%; 非漸進 dropout acc: 69.8%。並可發現 dropout 多時，val_loss 會比較好，但結果沒有比較好(如圖 1 及圖 2)
- **Ensemble 實驗**： 我還利用很多以前 train 出來的不同 model(架構不一樣共 3 個 70 % 的 model)，用 ensemble 再 Kaggle 得到 72.4% 的準確率。

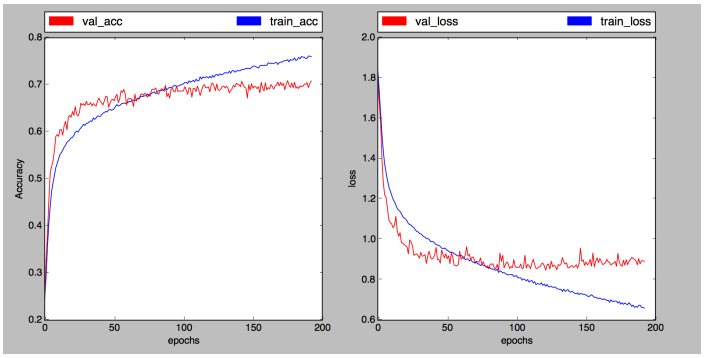


圖 1(model dropout a lot 的走勢圖)

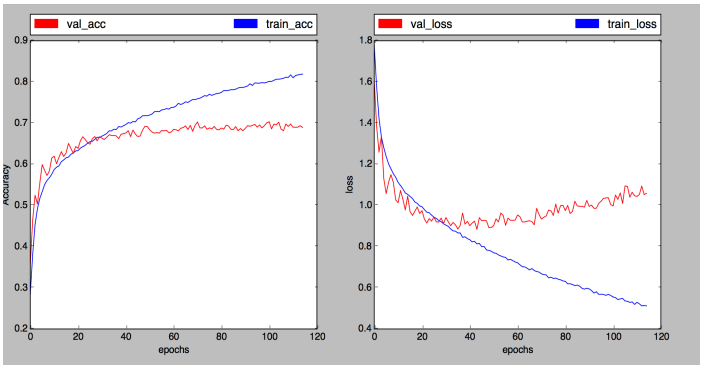


圖 2(model 正常的走勢圖)

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:)

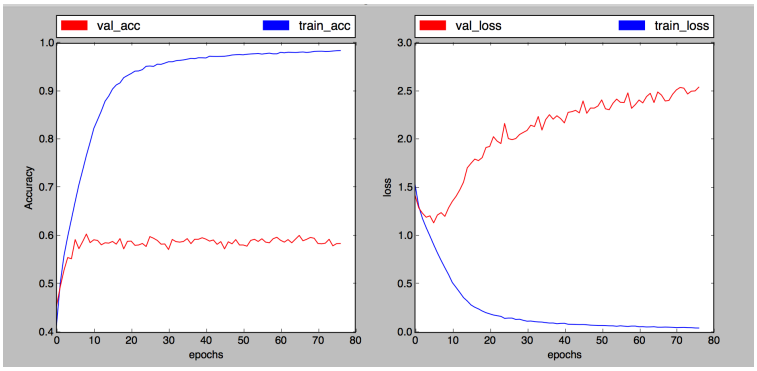
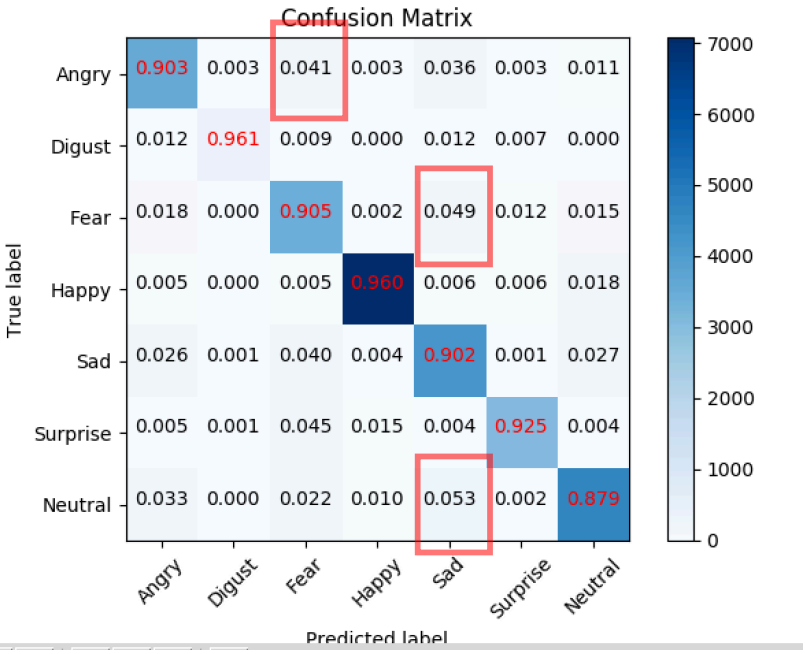


圖 3(未使用 augmentation 的走勢圖)

a. **data augmentation:** 使用 Keras 的 image-generator 產生平移(shift)、tilting、縮放的各種圖片。圖 3 可發現在未使用前，epoch=10 的時候，val_loss 已經爛掉(overfitting 了)，train_acc 也是很早就到 100%，不像圖 2 到 epoch=100 之後還在穩定上升。可判斷，model 是可以的(train_acc 100%)，因此我們如果給他更多 data，準確率是會提昇的。使用後，Kaggle 準確率從 58.6%提升到 70.6%

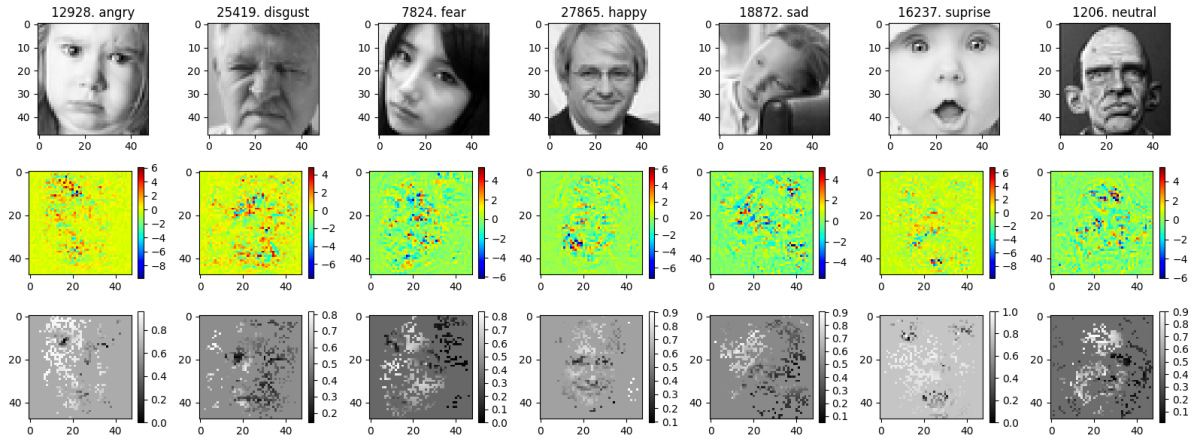
b. **data normalization:** 原本我是將所有的 batch normalization 拿掉，結果根本 train 不起來，其可見重要性。後來我是將在 IO 處，用 center 做 norm(圖 4)。在 Kaggle 上是有略小一點只有 69.8%準確率(比正常的少 1%)，model 竟然因為 normalization 而變爛，我把標準差印出來，發現標準差很大，可能間接讓 data 數量級變小，實際上，兩個不同圖之間背景等等是沒有什麼關連性的，因此差異都會很大，所以使用這樣的方法沒辦法讓準確率提昇 。

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:)



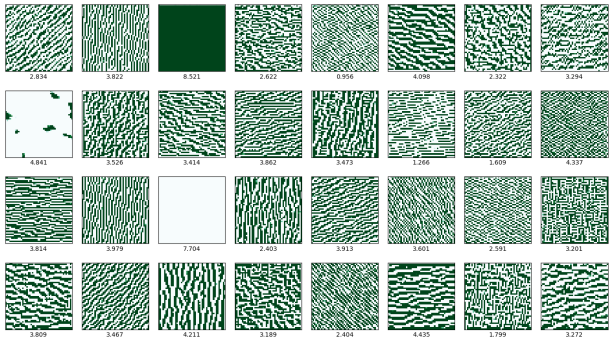
不難發現，對角線是預測正確的部份，都有達到 0.9 左右，而最容易搞混的是 sad 跟 neutral 這兩個 class，sad 跟 angry 以及 angry 跟 fear，其實可以想見，因為這些比較偏向負面的情緒，人的臉都是有類似的情況，光人我們也很難分辨出來。

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: 張問寬 b04901109)

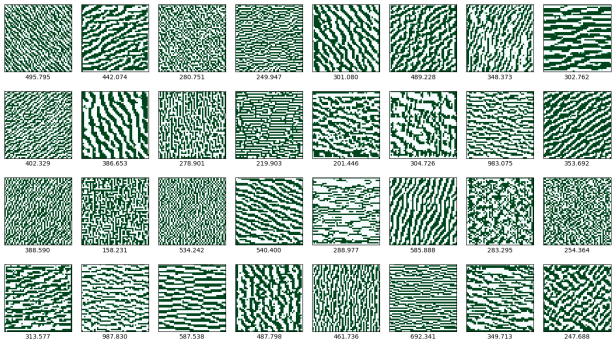


可以看到，熱點都是出現在眼睛、眉毛、嘴巴的部份。我們看 disgust 那張圖，他集中在眼睛跟嘴巴，跟我們人去分辨差不多，以及 happy 的部份熱點圖則集中在嘴巴那附近。機器真的有學到特定的特徵精隨，抓取出重點部分去做分析，而不是瞎猜或者是去記憶 data。

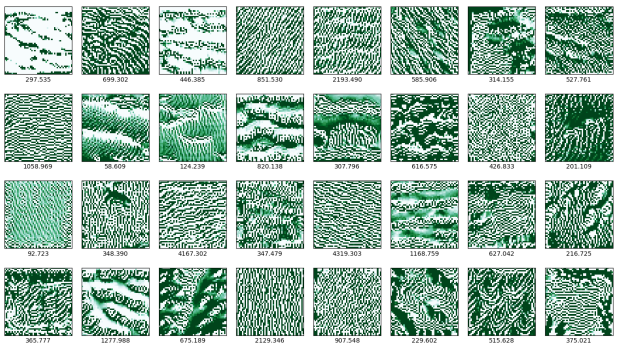
5. (1%) 承(4) 利用上課所提到的 gradient ascent 方法，觀察特定層的 filter 最容易被哪種圖片 activate 與觀察 filter 的 output。(Collaborators:)



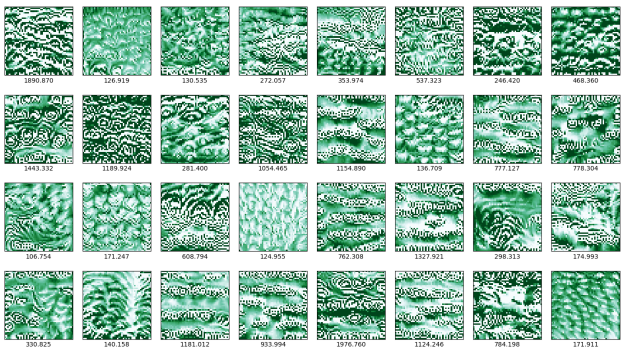
p_re_lu_1(a)



p_re_lu_2(b)



p_re_lu_5(c)



p_re_lu_7(d)

可以很明顯的看到，在淺層部分容易被粗線條紋(抓取臉部輪廓)給激化。且同層的不同 filter 差別在角度而已(如圖 a, b)，有旋轉 90 度的，或者 45 度的。當往深層，較細節比較不規則的東西呈現越明顯，如人的臉的細節(鼻子、嘴巴等)。其實人的五官會有很多角度變化，因此會需要有偵測不同角度的 filter。可以大概觀察出，同層的 filter 數量越多，角度變化可以較多變，而疊越深細節可以越來越明顯。