

參數連接：https://drive.google.com/file/d/1Q_94krmQFJiOa2ePcgWqQvQlc9IE2yTN/view?usp=sharing

1. (1%) 請以block diagram或是文字的方式說明這次表現最好的model使用哪些layer module(如 Conv/Linear 和各類 normalization layer) 及連接方式(如一般forward 或是使用 skip/residual connection), 並概念性逐項說明選用該 layer module 的理由。

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 64, 64]	640
BatchNorm2d-2	[-1, 64, 64, 64]	128
RReLU-3	[-1, 64, 64, 64]	0
MaxPool2d-4	[-1, 64, 32, 32]	0
Dropout-5	[-1, 64, 32, 32]	0
Conv2d-6	[-1, 128, 32, 32]	73,856
BatchNorm2d-7	[-1, 128, 32, 32]	256
ReLU-8	[-1, 128, 32, 32]	0
MaxPool2d-9	[-1, 128, 16, 16]	0
Dropout-10	[-1, 128, 16, 16]	0
Conv2d-11	[-1, 256, 16, 16]	295,168
BatchNorm2d-12	[-1, 256, 16, 16]	512
ReLU-13	[-1, 256, 16, 16]	0
MaxPool2d-14	[-1, 256, 8, 8]	0
Dropout-15	[-1, 256, 8, 8]	0
Conv2d-16	[-1, 512, 8, 8]	1,180,160
BatchNorm2d-17	[-1, 512, 8, 8]	1,024
ReLU-18	[-1, 512, 8, 8]	0
MaxPool2d-19	[-1, 512, 4, 4]	0
Dropout-20	[-1, 512, 4, 4]	0
Dropout-21	[-1, 8192]	0
Linear-22	[-1, 4096]	33,558,528
RReLU-23	[-1, 4096]	0
Dropout-24	[-1, 4096]	0
Linear-25	[-1, 1024]	4,195,328
RReLU-26	[-1, 1024]	0
Linear-27	[-1, 256]	262,400
RReLU-28	[-1, 256]	0
Linear-29	[-1, 7]	1,799
=====		
Total params: 39,569,799		
Trainable params: 39,569,799		
Non-trainable params: 0		
=====		
Input size (MB): 0.02		
Forward/backward pass size (MB): 13.30		
Params size (MB): 150.95		
Estimated Total Size (MB): 164.26		
=====		

本次使用的CNN模型如上圖所示。主要有四層卷積層，每層卷積層都有Batch Normalization, 激活函數RReLU(隨機Leaky ReLU), Max Pooling, 還有Dropout。使用Batch Normalization的原因是可以加快模型收斂速度，以及可以緩解訓練過程中梯度鬆散的問題，令神經網絡更加穩定。池化層(Max Pooling)的作用是減少計算量，加快模型訓練速度。Dropout的作用是防止Overfit，這點在後面實驗的時候發現可以大量增加validation accuracy。

輸入的圖片經過這六層卷積層之後，就會作扁平化(Flatten)處理，把input換成一維，最後輸入到全連接層裡面，最後用Softmax函數進行分類。連接方式主要是一般的forward。總參數量是39,569,799。訓練的時候使用了隨機12%作為驗證集，使用了torchvision.transforms做數據增強，包括把圖片水平翻轉，旋轉圖片，高斯模糊等等。訓練過程以批大小128，300個迭代進行訓練，以CrossEntropy作為Loss function。

ion,透過SGD進行參數更新,最後在Kaggle上的Private leaderboard分數為0.65285%, 超過了strong baseline。

2. (1%) 嘗試使用 **augmentation/early-stopping/ensemble** 三種訓練 **trick** 中的兩種,說明實作細節並比較有無該 **trick** 對結果表現的影響(validation 或是 testing 擇一即可)。

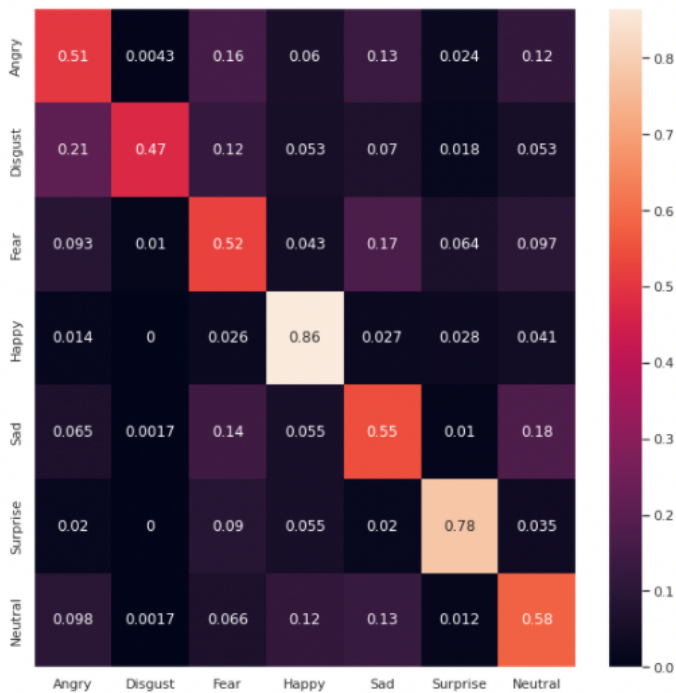
我使用了**early-stopping**和**augmentation**。在沒有使用**augmentation**的情況下, **validation accuracy**最高只有55%, 怎麼train也過不了simple baseline。但是使用了**augmentation**情況下, 就可以達到70%的**validation accuracy**, 能過strong baseline了。實作的細節包括使用了隨機將圖片水平翻轉, 隨機旋轉圖片, 高斯噪音, 以及改變perspective。

Early-stopping則是能讓我設置一個比較大的epoch size,然後如果validation accuracy超過我設置的threshold, 就可以提前停止, 不需要運行足1000個epoch, 節省時間。而且, 由於overfit的問題, 可能training accuracy一直在上升, 但是validation accuracy在某個epoch後會下降, 所以提前停止就可以防止這個問題。由於public leaderboard上的strong baseline是0.64114, 而第一名是0.69914的成績, 所以我選擇了0.66作為閾值。結果是在300 epoch左右就已經達到0.66了, 因此就自動停下來。

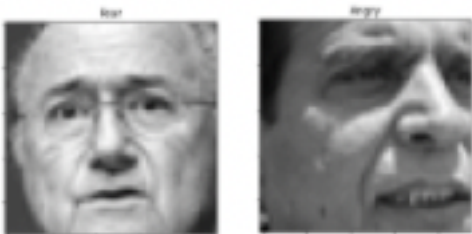
3. (1%) 畫出 **confusion matrix** 分析哪些類別的圖片容易使 **model** 搞混, 並簡單說明。

(ref: https://en.wikipedia.org/wiki/Confusion_matrix)

Confusion matrix如下:



可以看出當Y是Happy, Surprise的時候，分類的最好，而當Y是Disgust, Angry和Fear, Neutral的時候，分類準確度只有~50%. 推測可能是由於Happy和Surprise的圖片同質性比較強，可能微笑，驚恐這些表情都有明顯的feature(如嘴角上揚等等)，加上Happy在training dataset的frequency最多，但是Disgust, Angry, Fear等等都有共通性，比如眉頭緊皺等等

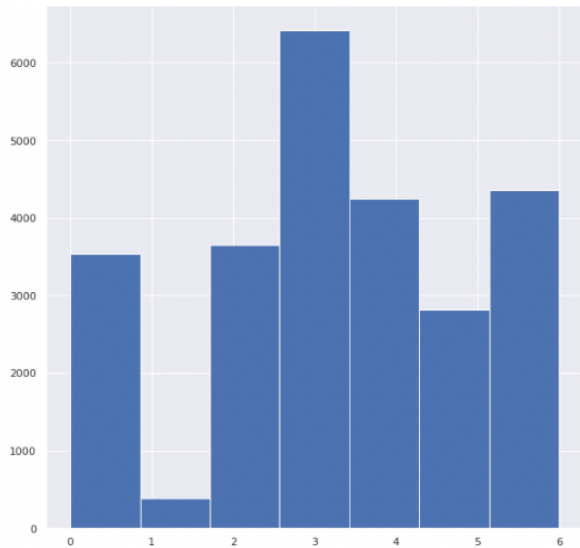


(左圖為Fear而右圖為Angry), 這樣就需要CNN抓去臉部更加細微的feature, 而這是比較困難的，因此accuracy比較低。

4. (1%) 請統計訓練資料中不同類別的數量比例，並說明：

對 **testing** 或是 **validation** 來說，不針對特定類別，直接選擇機率最大的類別會是最好的結果嗎？針對上述內容，是否存在更好的方式來提升表現？例如設置不同條件來選擇預測結果/變更訓練資料抽樣的方式，或是直接回答「否」(但需要給出支持你論點的論述)

訓練資料中不同類別的數量比例如下：



分別為：[3542, 393, 3651, 6419, 4245, 2820, 4351]

選擇機率最大的不是最好的結果，因為每個類別本來出現在每個訓練過程的機率就不一樣。就像是上述圖所示，3(Happy)的類別最多，那神經網絡看到一張新的圖片，在沒有其他信息情況下，自然就會猜這個新圖片是Happy。但是這樣的預測方法不好，會導致其他類別的準確率下降。所以可以assign不同weighting給每一個預測結果，這個weighting可以是根據其類別在training dataset佔的比重而定。或者是使用分層抽樣，在抽樣的時候根據類別把訓練集分成多個子分組，然後再從子分組進行抽樣，這樣就可以解決這個訓練集各類別數目不均衡的問題。

5. (3%) Refer to math problem

https://hackmd.io/@Gf0kB4kgS66YhhM7j6TJew/SJy_akYUK

參數連接：https://drive.google.com/file/d/1Q_94krmQFJiOa2ePcgWqQvQlc9IE2yTN/view?usp=sharing