Homework3 Report

Professor Pei-Yuan Wu
EE5184 - Machine Learning

姓名：陳譽仁

學號：R06521504

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？

   模型架構：

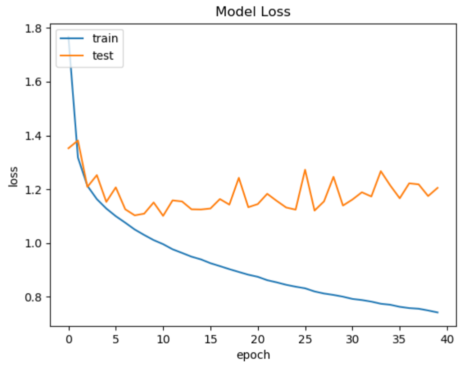
   如右圖，依序為 4 個卷積層，中間穿插 max pooling2D、batch normalization、Dropout，activation function 使用 LeakyReLu，接著經過 flatten 之後銜接兩層 500 個 node 的 dence 層，分別有設定 regularizers.l2(1e-4)與加上 0.5 的 dropout。資料方面，將 training dataset 的前 25% (7177 筆)切出做為 validation set，用剩下的資料進行訓練時，使用 ImageDataGenerator，data augmentation 設定為 10 倍。最後執行 40 個 epochs、batch size 為 128 的 training。

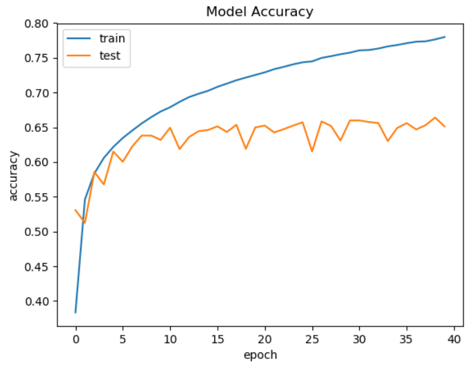
   訓練過程與準確率：

   結果於 Kaggle 上的分數約為 0.655，訓練過程與準確率隨 epoch 的變化如次頁，可以發現 validation set 的 test accuracy 在大概 10 個 epochs 之後就開始變化不大，但 training accuracy、training loss 還是持續改善，代表有些 overfitting 的情形發生。

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 46, 46, 64)	640
leaky_re_lu_1 (LeakyReLU)	(None, 46, 46, 64)	0
batch_normalization_1 (Batch	(None, 46, 46, 64)	256
max_pooling2d_1 (MaxPooling2	(None, 23, 23, 64)	0
dropout_1 (Dropout)	(None, 23, 23, 64)	0
conv2d_2 (Conv2D)	(None, 23, 23, 128)	73856
leaky_re_lu_2 (LeakyReLU)	(None, 23, 23, 128)	0
batch_normalization_2 (Batch	(None, 23, 23, 128)	512
max_pooling2d_2 (MaxPooling2	(None, 11, 11, 128)	0
dropout_2 (Dropout)	(None, 11, 11, 128)	0
conv2d_3 (Conv2D)	(None, 11, 11, 256)	295168
leaky_re_lu_3 (LeakyReLU)	(None, 11, 11, 256)	0
batch_normalization_3 (Batch	(None, 11, 11, 256)	1024
max_pooling2d_3 (MaxPooling2	(None, 5, 5, 256)	0
dropout_3 (Dropout)	(None, 5, 5, 256)	0
conv2d_4 (Conv2D)	(None, 5, 5, 512)	1180160
leaky_re_lu_4 (LeakyReLU)	(None, 5, 5, 512)	0
batch_normalization_4 (Batch	(None, 5, 5, 512)	2048
max_pooling2d_4 (MaxPooling2	(None, 2, 2, 512)	0
dropout_4 (Dropout)	(None, 2, 2, 512)	0
flatten_1 (Flatten)	(None, 2048)	0
dense_1 (Dense)	(None, 500)	1024500
activation_1 (Activation)	(None, 500)	0
batch_normalization_5 (Batch	(None, 500)	2000
dropout_5 (Dropout)	(None, 500)	0
dense_2 (Dense)	(None, 500)	250500
activation_2 (Activation)	(None, 500)	0
batch_normalization_6 (Batch	(None, 500)	2000
dropout_6 (Dropout)	(None, 500)	0
dense_3 (Dense)	(None, 7)	3507
activation_3 (Activation)	(None, 7)	0

Total params: 2,836,171
Trainable params: 2,832,251
Non-trainable params: 3,920

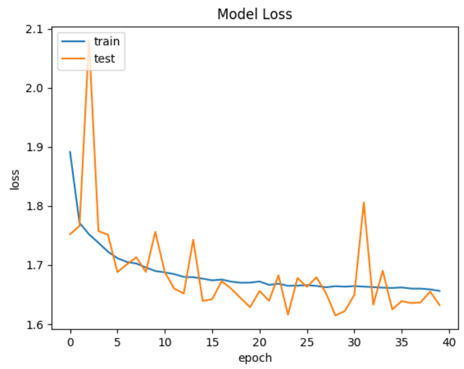


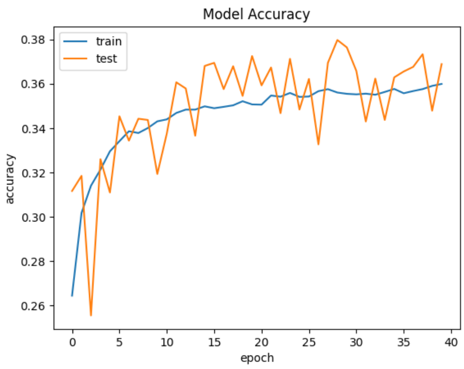


2. (1%) 承上題，請用與上述 CNN 接近的參數量，實做簡單的 DNN model，其模型架構、訓練過程和準確率為何？試與上題結果做比較，並說明你觀察到了什麼？

前述的 CNN model 中，總共有 2836171 個參數，因此設定如右圖的 DNN 模式進行比較，其中共有 3321007 個參數，先將輸入的影像經過 flatten，再分別由兩層各 1000 個 neuron 所組成，其餘迭代次數與資料生成的設定和前述的 CNN 模型相同。結果如下圖，由於直接將圖像扁平之後的資料較為雜亂，而且調整參數的時間較為不足，於 Kaggle 上的分數約為 0.37。從結果可以發現 DNN 模型的改善比 CNN 模型慢很多，而表現也比 CNN 模型差。

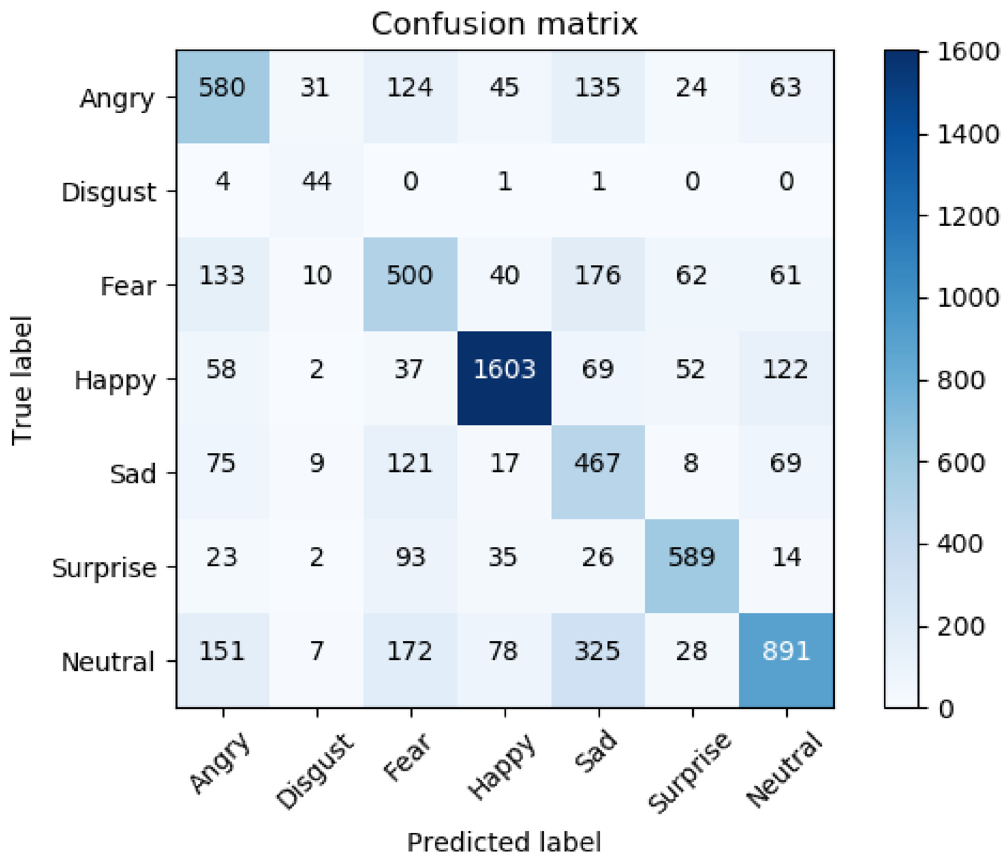
```
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              2305000
_________________________________________________________________
activation_1 (Activation)    (None, 1000)              0
_________________________________________________________________
batch_normalization_1 (Batch (None, 1000)              4000
_________________________________________________________________
dropout_1 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              1001000
_________________________________________________________________
activation_2 (Activation)    (None, 1000)              0
_________________________________________________________________
batch_normalization_2 (Batch (None, 1000)              4000
_________________________________________________________________
dropout_2 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 7007
_________________________________________________________________
activation_3 (Activation)    (None, 7)                 0
=================================================================
Total params: 3,321,007
Trainable params: 3,317,007
Non-trainable params: 4,000
_________________________________________________________________
```





3. (1%) 觀察答錯的圖片中，哪些 class 彼此間容易用混？ 並說明你觀察到了什麼？ [繪出 confusion matrix 分析]

Confusion matrix 位於下方，是利用第一題所述，切出的 7177 筆資料所繪成。可以發現切出的資料當中，disgust 的表情很少，happy 則是資料數量很多，而且大部分都能夠正確辨識，而對角線之外，最常判斷錯的組合是將 neutral 判斷成 sad。



---------------Handwritten question---------------

4. (1.5%,each 0.5%)CNN time/space complexity:

For a. b. Given a CNN model as

```
model = Sequential()
model.add(Conv2D(filters=6,
                 strides=(3, 3),
"""Layer A"""    padding ="valid",
                 kernel_size=(2,2),
                 input_shape=(8,8,5),
                 activation='relu'))
model.add(Conv2D(filters=4,
                 strides=(2, 2),
"""Layer B"""    padding ="valid",
                 kernel_size=(2,2),
                 activation='relu'))
```

And for the c. given the parameter as:

kernel size = (k,k);

channel size = c;

input shape of each layer = (n,n);

padding = p;

strides = (s,s);

a. How many parameters are there in each layer (Hint: you may consider whether the number of parameter is related with)

   Layer A: [(2 * 2 (kernel_size)) * 5 (# of input channels) + 1 (constant)] * 6 (filters) = 126

   Layer B: [(2 * 2 (kernel_size)) * 6 (# of input channels from Layer A) + 1 (constant)] * 4 (filters) = 100

b. How many multiplications/additions are needed for a forward pass (each layer)?

   Layer A:

   multiplications : [2 * 2 * 5 (per filters)] * [3 * 3 * 6 (# of kernels)] = 1080

   additions: [2 * 2 * 5 - 1 (per filters)] * [3 * 3 * 6 (# of kernels)] = 1026

   Layer B:

   multiplications : [2 * 2 * 6 (per filters)] * [1 * 1 * 4 (# of kernels)] = 96

   additions: [2 * 2 * 6 - 1 (per filters)] * [1 * 1 * 4 (# of kernels)] = 92

c. What is the time complexity of convolutional neural networks?(note: you must use big-O upper bound, and there are l (lower case of L) layer, you can use C_l, C_{l-1} as lth and l-1th layer)

   本題參考以下文獻中對 CNN 時間複雜度的描述：

   K. He and J. Sun, "Convolutional neural networks at constrained time cost," *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 5353-5360, 2015.

   對於第l個 layer，需要對每一個 filter (C_l個)之中進行$\left\lfloor\frac{n+2p-k}{s}\right\rfloor^2$次 kernel 的計算，對每一個 kernel，需要進行$2C_{l-1}k_i^2-1$次的加法與乘法，綜合以上則將時間複雜度表示下，並假設 n 遠大於 p 與 k：

$$O\left(\sum_{i=2}^{l}\left(C_l\left\lfloor\frac{n+2p-k}{s}\right\rfloor^2\left(2C_{l-1}k_i^2-1\right)\right)\right)=O\left(\sum_{i=2}^{l}\left(C_l\left(\frac{n}{s}\right)^2\left(2C_{l-1}k_i^2\right)\right)\right)$$

5. (1.5%,each 0.5%)PCA practice:Problem statement: Given 10 samples in 3D space. (1,2,3),(4,8,5),(3,12,9),(1,8,5),(5,14,2),(7,4,1),(9,8,9),(3,8,1),(11,5,6),(10,11,7)

本題計算 PCA 的方法參考 stackoverflow 上面的其中一個問答 (https://stackoverflow.com/questions/13224362/principal-component-analysis-pca-in-python)，先將資料平均平移至 0 後，再利用 python、numpy、scipy 根據講義的流程計算特徵值、內積。

a. (1) What are the principal axes?

   依序為：

   [[-0.6165947 0.67817891 0.39985541]
    [-0.58881629 -0.73439013 0.33758926]
    [-0.52259579 0.02728563 -0.85214385]]

b. (2) Compute the principal components for each sample.

   直接將資料轉換的結果如下：

   [[7.18658682 1.37323947 -2.25104047]
    [0.75871342 -0.94399334 -0.73022635]
    [-3.07034019 -4.45059025 -3.1883001]
    [2.60849751 -2.97853006 -1.92979259]
    [-1.82299166 -4.75401212 4.25159619]
    [3.35457763 3.91896138 2.52755823]
    [-4.41464321 2.55604371 -2.13952468]
    [3.46569126 -1.73131477 2.27849363]
    [-2.31359638 6.03371503 0.2038499]
    [-5.75249521 0.97648096 0.97738622]]

c. (3) Reconstruction error if reduced to 2D.(Calculate the L2-norm)

   降維度至 2D 後結果如下：

   [[7.18658682 1.37323947]
    [0.75871342 -0.94399334]
    [-3.07034019 -4.45059025]
    [2.60849751 -2.97853006]
    [-1.82299166 -4.75401212]
    [3.35457763 3.91896138]
    [-4.41464321 2.55604371]
    [3.46569126 -1.73131477]
    [-2.31359638 6.03371503]
    [-5.75249521 0.97648096]]

   回推回 3D 資料：

   [[-3.49990928 -5.24007291 -3.71821029]

[-1.10801504 0.24651657 -0.42225789]
[-1.12514095 5.07633588 1.48310968]
[-3.62836199 0.65147726 -1.44446088]
[-2.10002375 4.56470677 0.82297154]
[0.58934216 -4.85327652 -1.6461568]
[4.45550052 0.72228056 2.37681721]
[-3.31106801 -0.76919499 -1.85839567]
[5.51848951 -3.06881754 1.37370944]
[4.20918683 2.6700449 3.03287366]]

其 L2-norm 為：7.397319049934147