Homework3 Report

Professor Pei-Yuan Wu
EE5184 - Machine Learning

姓名：歐政鷹
學號：R07922142

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 **?**

```
Using TensorFlow backend.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 48, 48, 64)        640
_________________________________________________________________
batch_normalization_1 (Batch (None, 48, 48, 64)        256
_________________________________________________________________
dropout_1 (Dropout)          (None, 48, 48, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 48, 48, 64)        36928
_________________________________________________________________
batch_normalization_2 (Batch (None, 48, 48, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 24, 24, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 128)       73856
_________________________________________________________________
batch_normalization_3 (Batch (None, 24, 24, 128)       512
_________________________________________________________________
dropout_3 (Dropout)          (None, 24, 24, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 128)       147584
_________________________________________________________________
batch_normalization_4 (Batch (None, 24, 24, 128)       512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
dropout_4 (Dropout)          (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 12, 12, 256)       295168
_________________________________________________________________
batch_normalization_5 (Batch (None, 12, 12, 256)       1024
_________________________________________________________________
dropout_5 (Dropout)          (None, 12, 12, 256)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 12, 12, 256)       590080
_________________________________________________________________
batch_normalization_6 (Batch (None, 12, 12, 256)       1024
_________________________________________________________________
dropout_6 (Dropout)          (None, 12, 12, 256)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 12, 12, 256)       590080
_________________________________________________________________
batch_normalization_7 (Batch (None, 12, 12, 256)       1024
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 256)         0
_________________________________________________________________
dropout_7 (Dropout)          (None, 6, 6, 256)         0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 6, 6, 512)         1180160
_________________________________________________________________
batch_normalization_8 (Batch (None, 6, 6, 512)         2048
_________________________________________________________________
dropout_8 (Dropout)          (None, 6, 6, 512)         0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 6, 6, 512)         2359808
_________________________________________________________________
batch_normalization_9 (Batch (None, 6, 6, 512)         2048
_________________________________________________________________
dropout_9 (Dropout)          (None, 6, 6, 512)         0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 6, 6, 512)         2359808
_________________________________________________________________
batch_normalization_10 (Batc (None, 6, 6, 512)         2048
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 3, 3, 512)         0
_________________________________________________________________
dropout_10 (Dropout)         (None, 3, 3, 512)         0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 3, 3, 512)         2359808
_________________________________________________________________
batch_normalization_11 (Batc (None, 3, 3, 512)         2048
_________________________________________________________________
dropout_11 (Dropout)         (None, 3, 3, 512)         0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 3, 3, 512)         2359808
_________________________________________________________________
batch_normalization_12 (Batc (None, 3, 3, 512)         2048
_________________________________________________________________
dropout_12 (Dropout)         (None, 3, 3, 512)         0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 3, 3, 512)         2359808
_________________________________________________________________
batch_normalization_13 (Batc (None, 3, 3, 512)         2048
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 1, 512)         0
_________________________________________________________________
dropout_13 (Dropout)         (None, 1, 1, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
batch_normalization_14 (Batc (None, 256)               1024
_________________________________________________________________
dropout_14 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               65792
_________________________________________________________________
batch_normalization_15 (Batc (None, 256)               1024
_________________________________________________________________
dropout_15 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                16448
_________________________________________________________________
batch_normalization_16 (Batc (None, 64)                256
_________________________________________________________________
dropout_16 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 7)                 455
=================================================================
Total params: 14,946,759
Trainable params: 14,937,159
Non-trainable params: 9,600
_________________________________________________________________
None
```

以上是我的 **CNN** 模型架構，主要是參考了 **VGG-16** 的架構，並加入漸進式的 **dropout layer**、**batch normalization layer** 及對某幾層的 **filters** 數進行了調整而得出。

在訓練開始前，我取了 10%的資料作為 validation data，其餘的資料則利用了 image data augmentation 的技巧對圖片進行了旋轉、平移、縮放等不同方式的前處理，增加訓練資料的數量。接著再利用我的模型架構在 batch_size=128,epochs=300 的情況下進行訓練。在每層 layer，我都是以 padding='same',kernel_initializer='glorot_normal',activation='relu'作為基本的設定。

在訓練過程中，我利用 keras 的 ModelCheckpoint 來幫助我找出所有 epochs 中 validation test 中結果最好的 model。最後分別取得了 Public Score = 0.65422 及 Private Score = 0.65923 的準確度。

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？

Layer (type)	Output Shape	Param #
flatten_1 (Flatten)	(None, 2304)	0
dense_1 (Dense)	(None, 256)	590080
batch_normalization_1 (Batch	(None, 256)	1024
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 256)	65792
batch_normalization_2 (Batch	(None, 256)	1024
dropout_2 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 512)	131584
batch_normalization_3 (Batch	(None, 512)	2048
dropout_3 (Dropout)	(None, 512)	0
dense_4 (Dense)	(None, 512)	262656
batch_normalization_4 (Batch	(None, 512)	2048
dropout_4 (Dropout)	(None, 512)	0
dense_5 (Dense)	(None, 1024)	525312
batch_normalization_5 (Batch	(None, 1024)	4096
dropout_5 (Dropout)	(None, 1024)	0
dense_6 (Dense)	(None, 1024)	1049600
batch_normalization_6 (Batch	(None, 1024)	4096
dropout_6 (Dropout)	(None, 1024)	0
dense_7 (Dense)	(None, 1024)	1049600
batch_normalization_7 (Batch	(None, 1024)	4096
dropout_7 (Dropout)	(None, 1024)	0
dense_8 (Dense)	(None, 2048)	2099200
batch_normalization_8 (Batch	(None, 2048)	8192
dropout_8 (Dropout)	(None, 2048)	0
dense_9 (Dense)	(None, 2048)	4196352
batch_normalization_9 (Batch	(None, 2048)	8192
dropout_9 (Dropout)	(None, 2048)	0
dense_10 (Dense)	(None, 2048)	4196352
batch_normalization_10 (Batch	(None, 2048)	8192
dropout_10 (Dropout)	(None, 2048)	0
dense_11 (Dense)	(None, 7)	14343

Total params: 14,223,879
Trainable params: 14,202,375
Non-trainable params: 21,504

以上是我的 **DNN** 模型架構，跟 **CNN** 架構一樣擁有漸進式的 **dropout layer** 及 **batch normalization layer**。

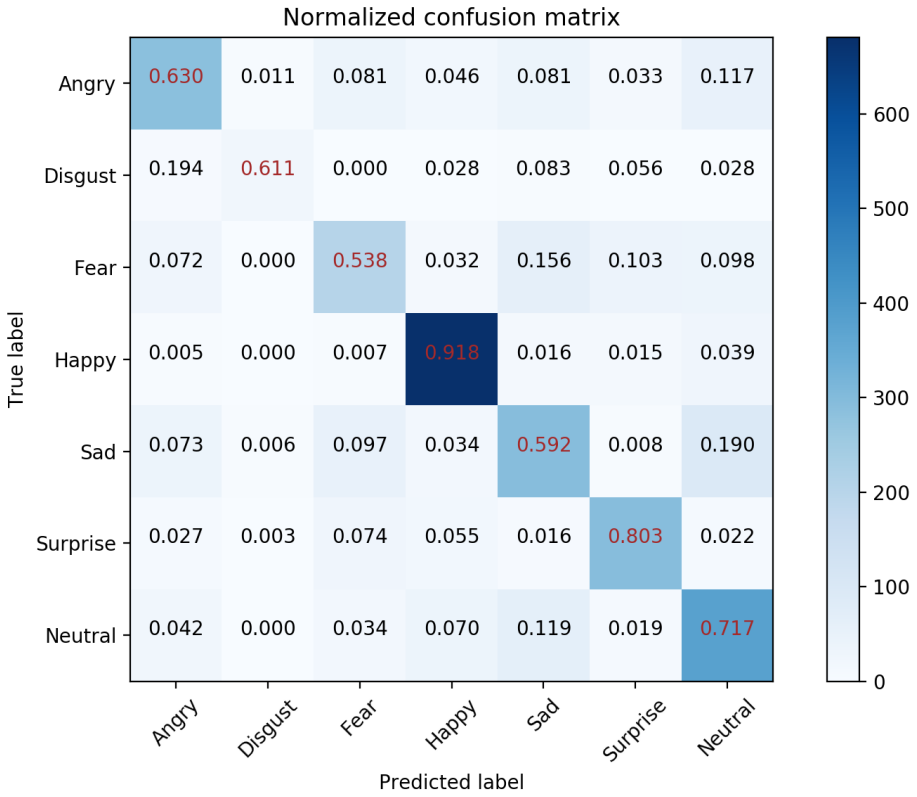
訓練過程與 **CNN** 大同小異，10%的資料作為 validation data，其餘的資料則利用了 image data augmentation 的技巧對圖片進行了旋轉、平移、縮放等不同方式的前處理，增加訓練資料的數量。

模型架構在 batch_size=128,epochs=300 的情況下進行訓練。在每層 layer，我都是以 padding='same',activation='relu'作為基本的設定。

在訓練過程中，也利用 keras 的 ModelCheckpoint 來幫助我找出所有 epochs 中 validation test 中結果最好的 model。最後分別取得了 Public Score = 0.30844 及 Private Score = 0.32153 的準確度。

與 CNN 比較，根據我的 model 檔案紀錄，我發現 DNN 會在比較短時間內出現收斂的情況。其次 DNN 的運算速度比 CNN 快很多，這是因為在卷積過程中會有很多的運算。最後很明顯的 DNN 的準確度遠不及 CNN。

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]



很明顯的看的出來，disgust 比較容易誤判成 angry、sad 容易誤判成 neutral、fear 跟 neutral 容易誤判成 sad。

我有去 data set 中找出容易被誤判的幾類照片來觀察，發現有些照片比較起來看下去第一瞬間真的不好判斷說應該要分到哪一類，所以程式分不出來我覺得也是正常的。

----------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*6*5 + 6 = 126
   Layer B:
   2*2*4*6 + 4 = 100
b. How many multiplications/additions are needed for a forward pass(each layer).
   Layer A:
   multiplications : 2*2*5*3*3*6 = 1080
   additions : (2*2*5 - 1)*3*3*6 = 1026
   Layer B:
   multiplications : 2*2*6*1*1*4 = 96
   additions : (2*2*6 - 1)*1*1*4 = 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 □ □,□ □−1as lth and l-1th layer)

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)
   a. (1) What are the principal axes?

$$X = \begin{bmatrix} 1 & 4 & 3 & 1 & 5 & 7 & 9 & 3 & 11 & 10 \\ 2 & 8 & 12 & 8 & 14 & 4 & 8 & 8 & 5 & 11 \\ 3 & 5 & 9 & 5 & 2 & 1 & 9 & 1 & 6 & 7 \end{bmatrix}$$

$$\mu = \text{mean}(X) = \begin{bmatrix} 54/10 & 8 & 48/10 \end{bmatrix}^T = \begin{bmatrix} 5.4 & 8 & 4.8 \end{bmatrix}^T$$

計算 covariance martix

$$\Sigma = \frac{1}{10} \sum_{i=1}^{10} (x_i - \mu)(x_i - \mu)^T$$

$$= \begin{bmatrix} 301/25 & 1/2 & 82/25 \\ 1/2 & 61/5 & 29/10 \\ 82/25 & 29/10 & 204/25 \end{bmatrix}$$

$$= \begin{bmatrix} 12.04 & 0.5 & 3.28 \\ 0.5 & 12.2 & 2.9 \\ 3.28 & 2.9 & 8.16 \end{bmatrix}$$

計算 eigenvalues 及 eigenvectors：

求 eigenvalues：

$$\det(\Sigma - \lambda I) = 0$$

$$\frac{-2500\lambda^3 + 81000\lambda^2 - 813170\lambda + 2433923}{2500} = 0$$

$$\lambda \approx 5.47203,\ 11.63052,\ 15.29744$$

$$\therefore \lambda_1 \approx 5.47203,\ \lambda_2 \approx 11.63052,\ \lambda_3 \approx 15.29744$$

Eigenvectors of $\lambda_1 \approx \begin{pmatrix} -0.46923 \\ -0.39616 \\ 1 \end{pmatrix}$

of $\lambda_2 \approx \begin{pmatrix} 24.85480 \\ -26.91490 \\ 1 \end{pmatrix}$

$\lambda_3 \approx \begin{pmatrix} 1.17987 \\ 1.2671 \\ 1 \end{pmatrix}$

} Principle Axes

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

(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)

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