Homework 3 Report - Image Sentiment Classification

B05901022 電機三 許睿洋

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？

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？

<table>
<tr><td>模型</td><td>CNN</td><td>DNN</td></tr>
<tr><td>架構</td><td>

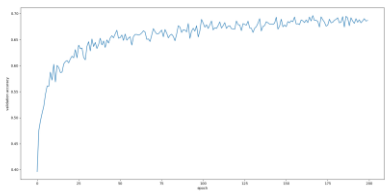
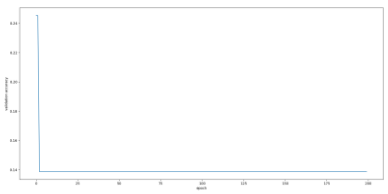
Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 48, 48, 64)	640
batch_normalization_1 (Batch	(None, 48, 48, 64)	256
leaky_re_lu_1 (LeakyReLU)	(None, 48, 48, 64)	0
conv2d_2 (Conv2D)	(None, 48, 48, 64)	36928
batch_normalization_2 (Batch	(None, 48, 48, 64)	256
leaky_re_lu_2 (LeakyReLU)	(None, 48, 48, 64)	0
conv2d_3 (Conv2D)	(None, 48, 48, 64)	36928
batch_normalization_3 (Batch	(None, 48, 48, 64)	256
leaky_re_lu_3 (LeakyReLU)	(None, 48, 48, 64)	0
max_pooling2d_1 (MaxPooling2	(None, 24, 24, 64)	0
dropout_1 (Dropout)	(None, 24, 24, 64)	0
conv2d_4 (Conv2D)	(None, 24, 24, 128)	73856
batch_normalization_4 (Batch	(None, 24, 24, 128)	512
leaky_re_lu_4 (LeakyReLU)	(None, 24, 24, 128)	0
conv2d_5 (Conv2D)	(None, 24, 24, 128)	147584
batch_normalization_5 (Batch	(None, 24, 24, 128)	512
leaky_re_lu_5 (LeakyReLU)	(None, 24, 24, 128)	0
max_pooling2d_2 (MaxPooling2	(None, 12, 12, 128)	0
dropout_2 (Dropout)	(None, 12, 12, 128)	0
conv2d_6 (Conv2D)	(None, 12, 12, 256)	295168
batch_normalization_6 (Batch	(None, 12, 12, 256)	1024
leaky_re_lu_6 (LeakyReLU)	(None, 12, 12, 256)	0
conv2d_7 (Conv2D)	(None, 12, 12, 256)	590080
batch_normalization_7 (Batch	(None, 12, 12, 256)	1024
leaky_re_lu_7 (LeakyReLU)	(None, 12, 12, 256)	0
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 256)	0
dropout_3 (Dropout)	(None, 6, 6, 256)	0
flatten_1 (Flatten)	(None, 9216)	0
dense_1 (Dense)	(None, 512)	4719104
batch_normalization_8 (Batch	(None, 512)	2048
leaky_re_lu_8 (LeakyReLU)	(None, 512)	0
dropout_4 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 256)	131328
batch_normalization_9 (Batch	(None, 256)	1024
leaky_re_lu_9 (LeakyReLU)	(None, 256)	0
dropout_5 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 7)	1799

Total params: 6,040,327
Trainable params: 6,036,871
Non-trainable params: 3,456

</td><td>

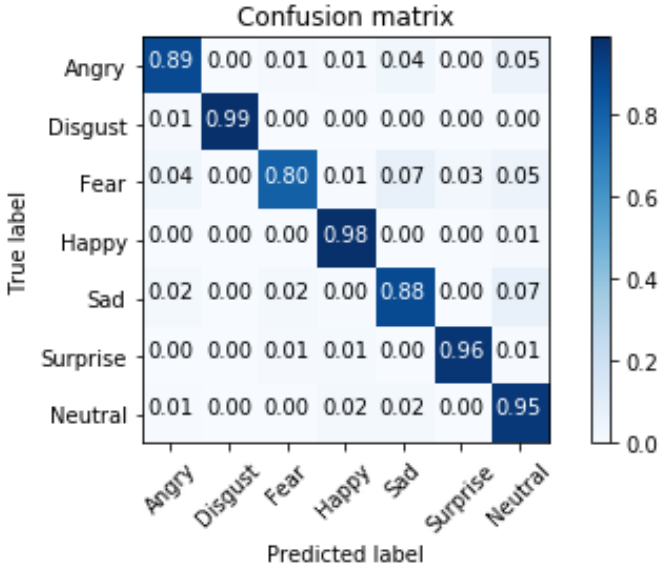
Layer (type)	Output Shape	Param #
flatten_1 (Flatten)	(None, 2304)	0
dense_1 (Dense)	(None, 1024)	2360320
leaky_re_lu_1 (LeakyReLU)	(None, 1024)	0
dense_2 (Dense)	(None, 1024)	1049600
leaky_re_lu_2 (LeakyReLU)	(None, 1024)	0
dense_3 (Dense)	(None, 1024)	1049600
leaky_re_lu_3 (LeakyReLU)	(None, 1024)	0
dense_4 (Dense)	(None, 512)	524800
leaky_re_lu_4 (LeakyReLU)	(None, 512)	0
dense_5 (Dense)	(None, 512)	262656
leaky_re_lu_5 (LeakyReLU)	(None, 512)	0
dense_6 (Dense)	(None, 512)	262656
leaky_re_lu_6 (LeakyReLU)	(None, 512)	0
dense_7 (Dense)	(None, 512)	262656
leaky_re_lu_7 (LeakyReLU)	(None, 512)	0
dense_8 (Dense)	(None, 256)	131328
leaky_re_lu_8 (LeakyReLU)	(None, 256)	0
dense_9 (Dense)	(None, 256)	65792
leaky_re_lu_9 (LeakyReLU)	(None, 256)	0
dense_10 (Dense)	(None, 128)	32896
leaky_re_lu_10 (LeakyReLU)	(None, 128)	0
dense_11 (Dense)	(None, 128)	16512
leaky_re_lu_11 (LeakyReLU)	(None, 128)	0
dense_12 (Dense)	(None, 64)	8256
leaky_re_lu_12 (LeakyReLU)	(None, 64)	0
dense_13 (Dense)	(None, 64)	4160
leaky_re_lu_13 (LeakyReLU)	(None, 64)	0
dense_14 (Dense)	(None, 32)	2080
leaky_re_lu_14 (LeakyReLU)	(None, 32)	0
dense_15 (Dense)	(None, 32)	1056
leaky_re_lu_15 (LeakyReLU)	(None, 32)	0
dense_16 (Dense)	(None, 32)	1056
leaky_re_lu_16 (LeakyReLU)	(None, 32)	0
dense_17 (Dense)	(None, 16)	528
leaky_re_lu_17 (LeakyReLU)	(None, 16)	0
dense_18 (Dense)	(None, 16)	272
leaky_re_lu_18 (LeakyReLU)	(None, 16)	0
dense_19 (Dense)	(None, 16)	272
leaky_re_lu_19 (LeakyReLU)	(None, 16)	0
dense_20 (Dense)	(None, 8)	136
leaky_re_lu_20 (LeakyReLU)	(None, 8)	0
dense_21 (Dense)	(None, 8)	72
leaky_re_lu_21 (LeakyReLU)	(None, 8)	0
dense_22 (Dense)	(None, 8)	72
leaky_re_lu_22 (LeakyReLU)	(None, 8)	0
dense_23 (Dense)	(None, 7)	63

Total params: 6,036,839
Trainable params: 6,036,839
Non-trainable params: 0

</td></tr>
<tr><td>訓練過程
(橫:epoch;
縱:val acc)</td><td>  </td><td>  </td></tr>
<tr><td>準確率</td><td>69%</td><td>14%</td></tr>
</table>

比較結果	使用相同參數量的 DNN 模型很顯然會在前幾個 epoch 便嚴重 overfit 了。雖然本來使用 fully connected 就會很容易 overfit，但是相同數量的參數下 CNN 還有相當漂亮的結果，因此可以得知使用圖片的特性來訓練一個模型在處理影響辨識上相當重要。

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]



容易搞混的 class:

(1) Fear⇔Sad, Neutral, Angry

(2) Sad⇔Neutral

(3) Angry⇔Sad, Neutral

Neutral(中性)的 label 針對自己的預測準確率很高，但其他的 label(Fear, Sad, Angry)卻很容易預測到 Neutral 上。在所有的 label 中，Fear 的準確率最低，Disgust 跟 Happy 的準確率最高。

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:

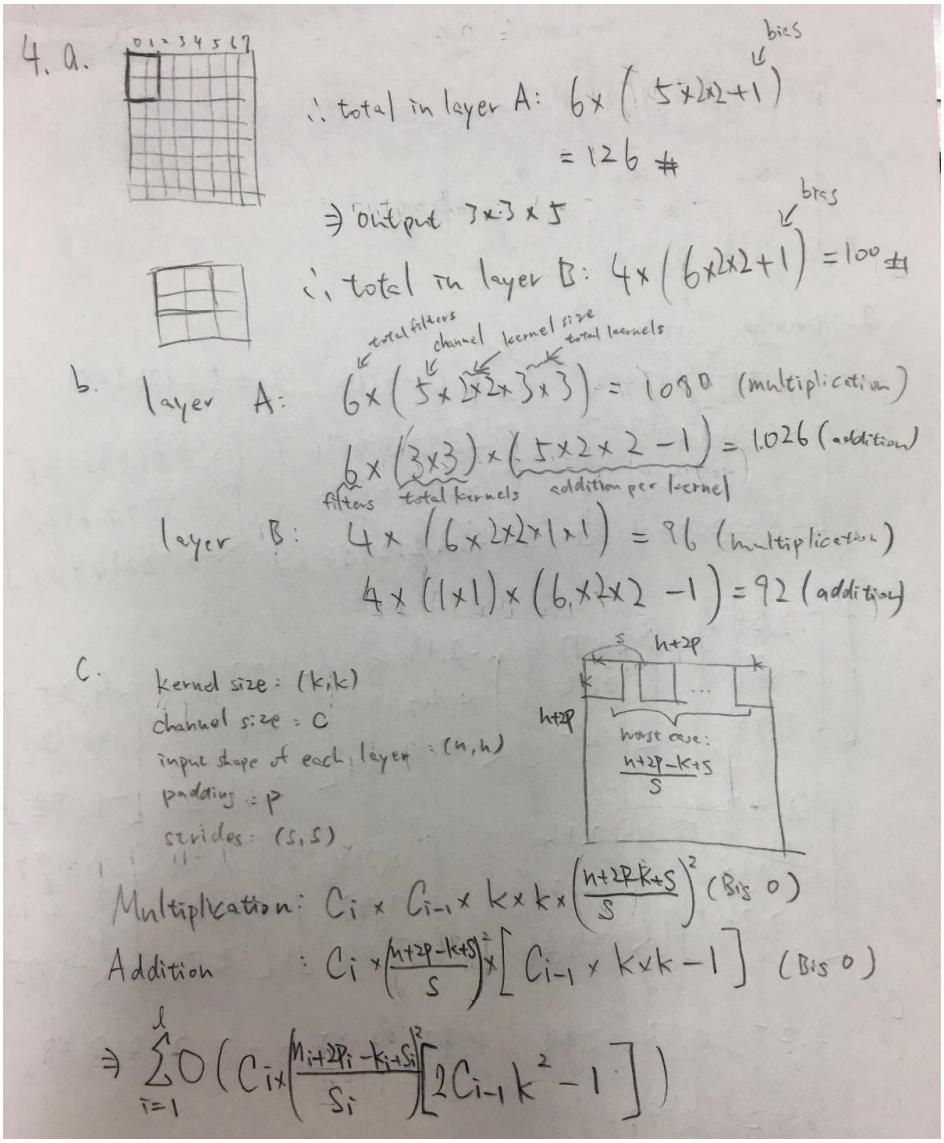
kernel size = (k, k); channel size = c; input shape of each layer = (n, n);

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)

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

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 Cl, Cl-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)

(1) What are the principal axes?

(2) Compute the principal components for each sample.

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

Covariance matrix

$$\begin{bmatrix} 12.04 & 0.5 & 3.28 \\ 0.5 & 12.2 & 2.9 \\ 3.29 & 2.9 & 8.16 \end{bmatrix}$$

a. principle axes:

$\lambda_1 = 15.2974434$ $\lambda_2 = 11.6305\,2369$ $\lambda_3 = 5.47203291$

$$\begin{bmatrix} -0.6165.747 \\ -0.5\,888\,1629 \\ -0.522595\,79 \end{bmatrix} \quad \begin{bmatrix} -0.67\,817\,891 \\ 0.73\,432\,013 \\ -0.027\,285\,63 \end{bmatrix} \quad \begin{bmatrix} 0.39\,855\,41 \\ 0.3375\,896 \\ -0.852\,143\,85 \end{bmatrix}$$

b.

$$\left[\begin{bmatrix} -2.25 \\ -1.37 \\ 7.19 \end{bmatrix}, \begin{bmatrix} -0.73 \\ 0.94 \\ 0.75 \end{bmatrix}, \begin{bmatrix} -2.19 \\ 4.45 \\ -3.07 \end{bmatrix}, \begin{bmatrix} -1.93 \\ 2.98 \\ 2.61 \end{bmatrix}, \begin{bmatrix} 4.25 \\ 4.75 \\ -1.82 \end{bmatrix}, \right.$$

$$\left. \begin{bmatrix} 2.53 \\ -3.92 \\ 3.35 \end{bmatrix}, \begin{bmatrix} -2.14 \\ -2.56 \\ -4.41 \end{bmatrix}, \begin{bmatrix} 2.28 \\ 1.73 \\ 7.47 \end{bmatrix}, \begin{bmatrix} 0.20 \\ -6.03 \\ -2.31 \end{bmatrix}, \begin{bmatrix} 0.98 \\ -0.98 \\ -5.75 \end{bmatrix} \right]$$

c. 2.25, 0.73, 3.19, 1.93, 4.25, 2.53, 2.14, 2.28, 2.04, 0.98