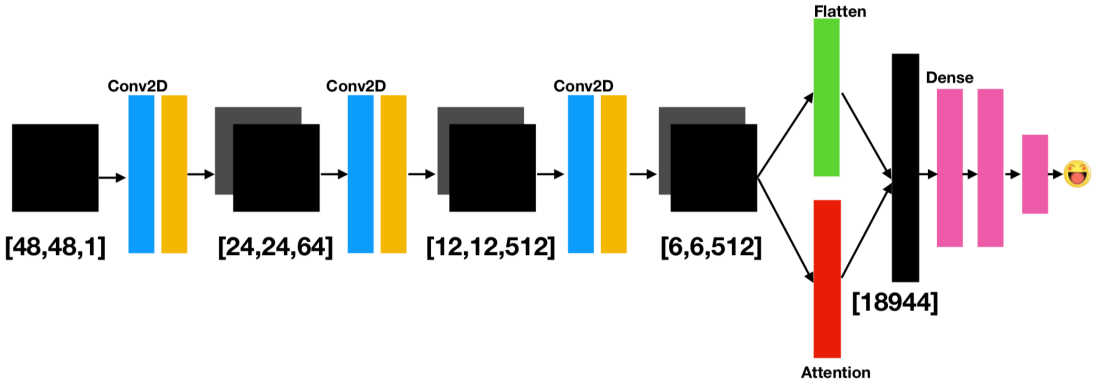
Homework3 Report

姓名：劉兆鵬 學號：r07921052

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？

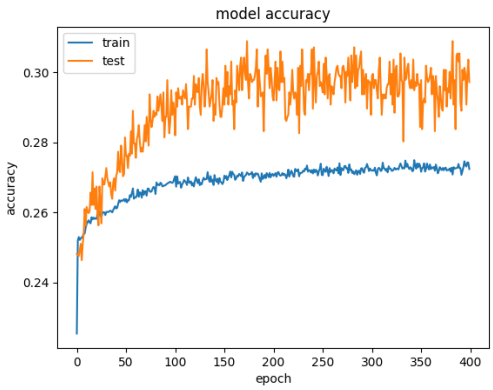


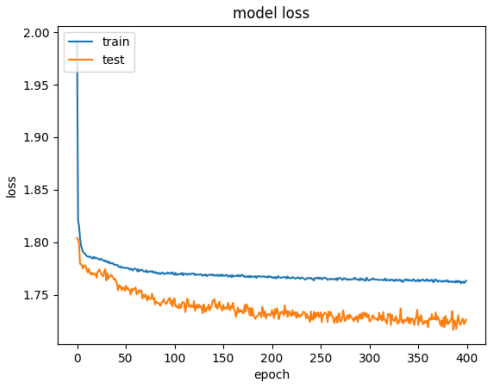
	Single Model	Ensemble Model
Public Accuracy	0.70325	0.71412
Private Accuracy		

我使用的 CNN 模型為仿照 VGG 以及 Alexnet 的整合方式如上圖一所示，在特徵層得部分使用三層 Conv layer，且在每一層 Conv layer 都包含著 Batchnormalization、Activation 以及 Dropout 層，並使得 filter 的個數隨著層數逐漸增長，漸增的做法能夠使得 back propagation 時降低 gradient vanish 的可能，並在最後一層 Conv layer 加入 Attention model，以 feature map 的位置作為注意力增強，使得模型能夠了解要注意圖片哪個位置來學習判斷，最後再加入三層 Fully connected layer，而最後一層為使用 7 個 neuron 代表為 7 種情緒並使用 softmax 使輸出為各個類別的機率．

在訓練方面為了能夠使得資料的能夠增加多樣性及數量，我採用 data augmentation 來增加我的訓練資料．而其中我採用了選轉，裁切以及平移的方式，使得同一張圖片能夠生成更多相同類別的圖片．此外，我也採用了 ensemble 的方式來預測結果，為訓練多個不同架構的模型且使用不同的訓練資料，使得多種模型能夠填補每個模型所切出來的 validation data．最後將各個模型所預測的結果相加並取平均，就能夠得出大家對於同一張圖片所給予的分數，使得預測結果能夠更加穩定且正確如上表所示．而單一模型的訓練過程如上圖二所示．

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？



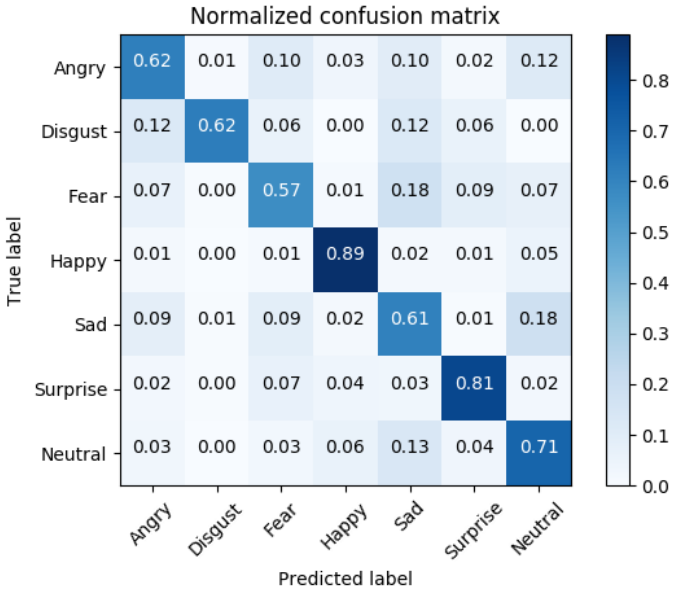


我使用了14層 Dense layer 來使得模型參數與 CNN 參數量接近，DNN 的參數量為 12,392,391 而 CNN 的參數量為 12,664,328，在每一個 Dense 後面都接上了 Activation 以及 Dropout layer，而在資料處理以及 augmentation 的部分也都與上述 CNN 相同，而模型輸入的方式不同的地方在於DNN須將圖片維度從 (48,48,1) flatten 為 (48*48*1,) 才能夠使用 Dense layer 來計算。

訓練過程中可以發現，DNN 在訓練時的速度比 CNN 還要快上許多，但是 DNN 的 loss 都一直維持很高且 Accuracy 也很快的就已經卡住上不去了，準確度也才在 0.29 左右，少了 CNN 的準確度一半以上，訓練過程圖上圖所示。

與 CNN 的比較可以得知，CNN 的時間複雜度是比 DNN 要來得高的，並且 CNN 也有更好的效果，我想是因為使用 Convolution 的方式才能夠了解圖片在各的區塊所代表的特徵為何，而使用 DNN 就是將整張圖所有的位置都一同計算，導致模型無法完善的了解圖片每個位置所代表的特徵為何。

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]



從上圖 confusion matrix 中可以發現，Fear 在整體中是較差的，在這 Fear 出錯的情況下，可以看出被認定為 Sad 的情況最多，而在 Sad 出錯的情況下又為 Neutral 的最多，可以看出 Fear、Sad 以及 Neutral 彼此之間都常會有辨識出錯的可能，其實人有 Fear 及 Sad 時也都是較無表情的情緒，故單從圖片中要辨識此三種情緒倒是比較難完整的辨別的．

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

手寫題從下頁開始

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);

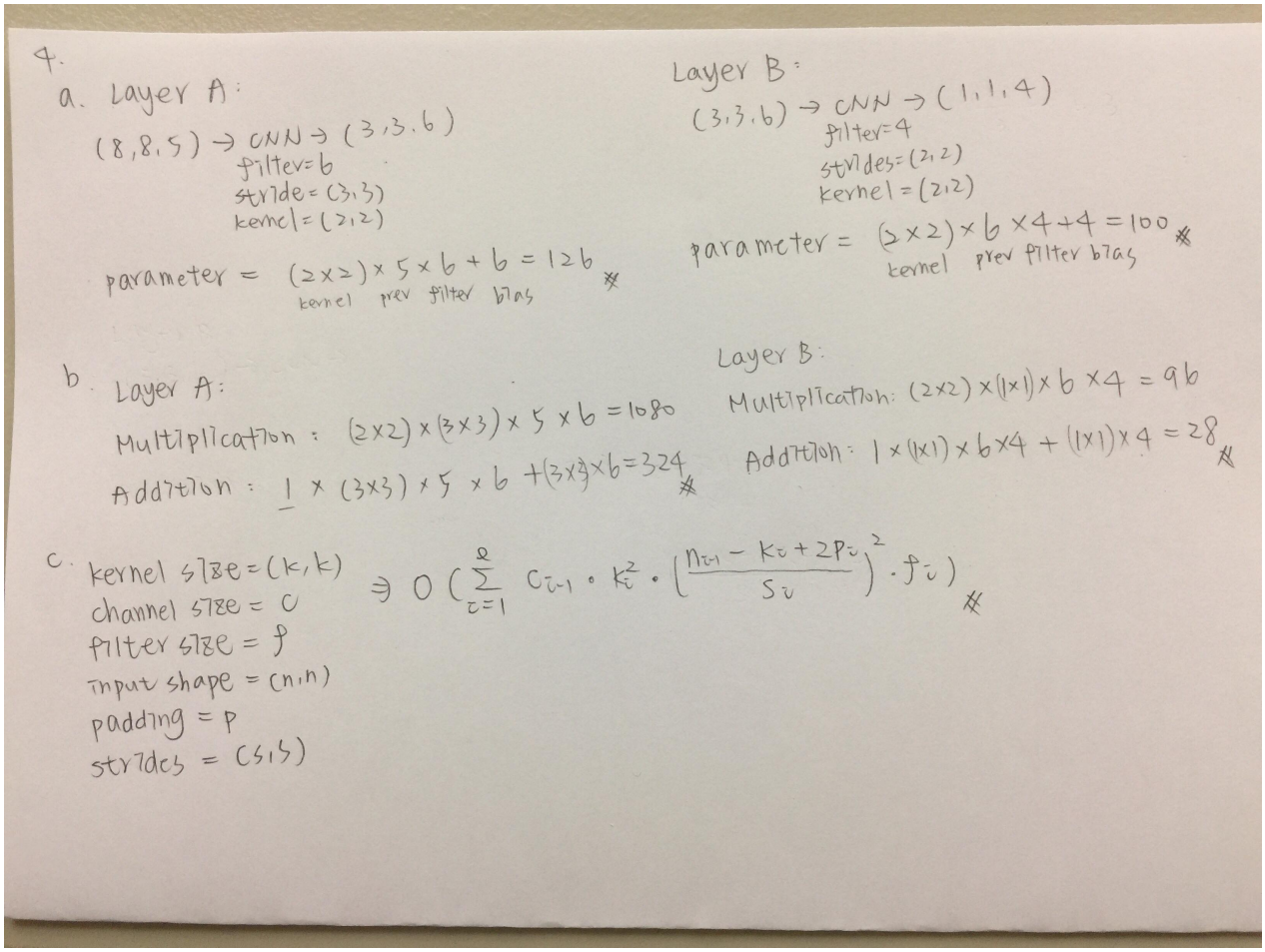
channel size = c;

filter size = f;

input shape = (n,n);

padding = 1;

strides = (s,s);



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?
   b. (2) Compute the principal components for each sample.
   c. (3) Reconstruction error if reduced to 2D.(Calculate the L2-norm)

5, Given 10 samples: (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. 令 10 sample 為 X, $X \in R^{M \times N}$, $M=3$, $N=10$

令 μ 為 X 的 mean, $\mu \in R^M$

$\Sigma = \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)(x_i-\mu)^T = U \Lambda U^T$, $U=[u_1,u_2,u_3]$, 為 Σ 的 eigenvector

$u_1 = [-0.61659447, -0.58881629, -0.52259579]$

$u_2 = [0.67817891, -0.73439013, 0.02728563]$

$u_3 = [-0.39985541, -0.33758926, 0.85214385]$ ✗

b. 求各 sample 對 U 的 principle components ⇒ $C = XU$

① (1,2,3) ⇒ $C_1 = -3.3620$, $C_2 = -0.7087$, $C_3 = 1.4813$
② (4,8,5) ⇒ $C_1 = -9.7898$, $C_2 = -3.0259$, $C_3 = -0.0394$
③ (3,12,9) ⇒ $C_1 = -13.6189$, $C_2 = -6.5325$, $C_3 = -0.0394$
④ (1,8,5) ⇒ $C_1 = -7.9401$, $C_2 = -5.0605$, $C_3 = 1.1601$
⑤ (5,14,2) ⇒ $C_1 = -12.3715$, $C_2 = -6.8359$, $C_3 = -5.0212$
⑥ (7,4,1) ⇒ $C_1 = -7.1940$, $C_2 = 1.8369$, $C_3 = -3.2972$
⑦ (9,8,9) ⇒ $C_1 = -14.9632$, $C_2 = 0.4740$, $C_3 = 1.3698$
⑧ (3,8,1) ⇒ $C_1 = -7.0829$, $C_2 = -3.8132$, $C_3 = -3.0481$
⑨ (11,5,6) ⇒ $C_1 = -12.8619$, $C_2 = 3.9517$, $C_3 = -0.9734$
⑩ (10,11,7) ⇒ $C_1 = -16.3010$, $C_2 = -1.1055$, $C_3 = -1.7470$ ✗

c. Reconstruction error for 2D (L2-norm)

⇒ 令 $\hat{X}$ 為 reconstruction data $\hat{X} = CU^T$

error $= \sqrt{\sum_{i=1}^{M}(x_i-\hat{x}_i)}$, for each sample.

① error for (1,2,3) = 1.4813
② error for (4,8,5) = 0.0394
③ error for (3,12,9) = 2.4186
④ error for (1,8,5) = 1.1601
⑤ error for (5,14,2) = 5.0212
⑥ error for (7,4,1) = 3.2972
⑦ error for (9,8,9) = 1.3698
⑧ error for (3,8,1) = 3.0481
⑨ error for (11,5,6) = 0.9734
⑩ error for (10,11,7) = 1.7470 ✗