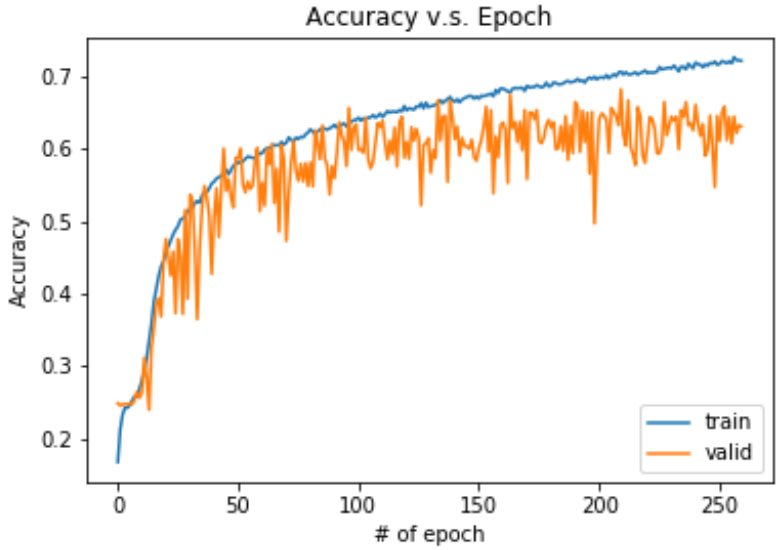
Homework3 Report

系級:資工碩二 學號:R06922134 姓名:葉沛陽

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？

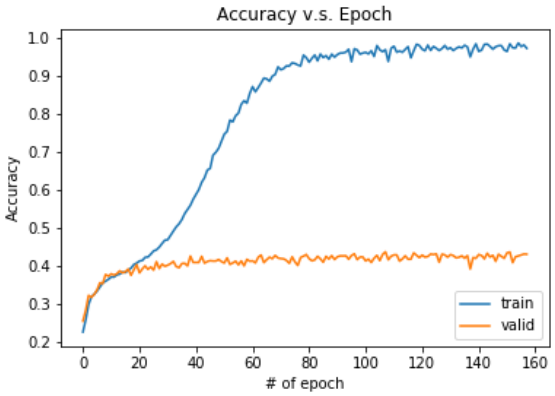
```
_________________________________________________________________
Layer (type)
=================================================================
conv2d_1 (Conv2D)
_________________________________________________________________
batch_normalization_1 (Batch
_________________________________________________________________
p_re_lu_1 (PReLU)
_________________________________________________________________
conv2d_2 (Conv2D)
_________________________________________________________________
batch_normalization_2 (Batch
_________________________________________________________________
p_re_lu_2 (PReLU)
_________________________________________________________________
dropout_1 (Dropout)
_________________________________________________________________
conv2d_3 (Conv2D)
_________________________________________________________________
batch_normalization_3 (Batch
_________________________________________________________________
p_re_lu_3 (PReLU)
_________________________________________________________________
max_pooling2d_1 (MaxPooling2
_________________________________________________________________
dropout_2 (Dropout)
_________________________________________________________________
conv2d_4 (Conv2D)
_________________________________________________________________
batch_normalization_4 (Batch
_________________________________________________________________
p_re_lu_4 (PReLU)
_________________________________________________________________
conv2d_5 (Conv2D)
_________________________________________________________________
batch_normalization_5 (Batch
_________________________________________________________________
p_re_lu_5 (PReLU)
_________________________________________________________________
max_pooling2d_2 (MaxPooling2
_________________________________________________________________
dropout_3 (Dropout)
_________________________________________________________________
conv2d_6 (Conv2D)
_________________________________________________________________
batch_normalization_6 (Batch
_________________________________________________________________
p_re_lu_6 (PReLU)
_________________________________________________________________
dropout_4 (Dropout)
_________________________________________________________________
conv2d_7 (Conv2D)
_________________________________________________________________
batch_normalization_7 (Batch
_________________________________________________________________
p_re_lu_7 (PReLU)
_________________________________________________________________
max_pooling2d_3 (MaxPooling2
_________________________________________________________________
dropout_5 (Dropout)
_________________________________________________________________
flatten_1 (Flatten)
_________________________________________________________________
dense_1 (Dense)
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)
_________________________________________________________________
batch_normalization_8 (Batch
_________________________________________________________________
dropout_6 (Dropout)
_________________________________________________________________
dense_2 (Dense)
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)
_________________________________________________________________
batch_normalization_9 (Batch
_________________________________________________________________
dropout_7 (Dropout)
_________________________________________________________________
dense_3 (Dense)
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)
_________________________________________________________________
batch_normalization_10 (Batc
_________________________________________________________________
dropout_8 (Dropout)
_________________________________________________________________
dense_4 (Dense)
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)
_________________________________________________________________
batch_normalization_11 (Batc
_________________________________________________________________
dropout_9 (Dropout)
_________________________________________________________________
dense_5 (Dense)
=================================================================
Total params: 7,547,390
Trainable params: 7,541,568
Non-trainable params: 5,822
_________________________________________________________________
```

	Training set		Testing set(Kaggle)	
	Train	**Validation**	**Public**	**Private**
Accuracy	**0.72760**	**0.68289**	**0.65728**	**0.66759**
Epoch	**Best at 257**	**Best at 210**	**at 210**	**at 210**



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)
=================================
dense_1 (Dense)
_________________________________
dense_2 (Dense)
_________________________________
dense_3 (Dense)
_________________________________
dense_4 (Dense)
_________________________________
dense_5 (Dense)
_________________________________
dropout_1 (Dropout)
_________________________________
dense_6 (Dense)
=================================
Total params: 7,417,991
Trainable params: 7,417,991
Non-trainable params: 0
```



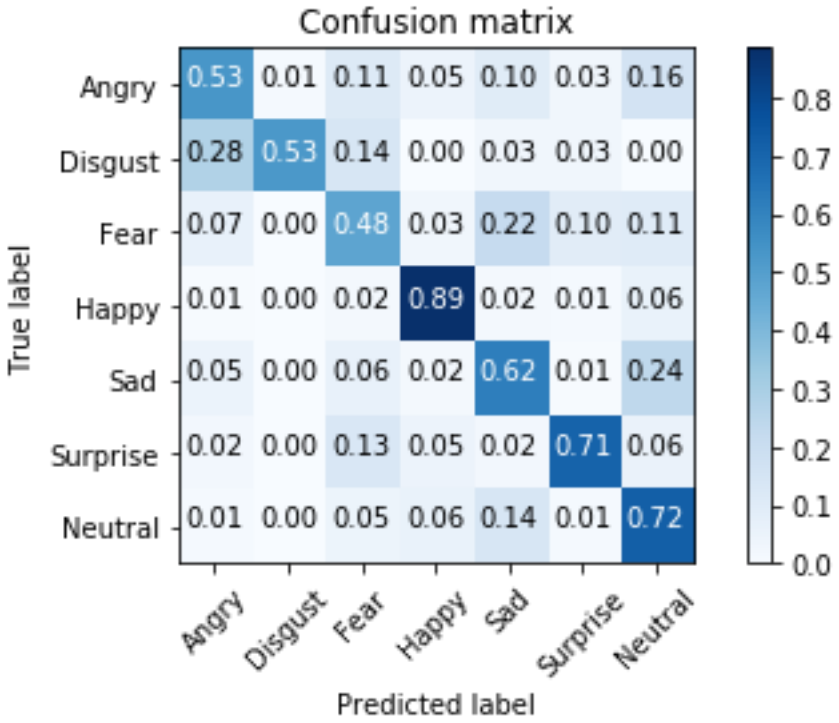
	Training set		Testing set(Kaggle)	
	Train	**Validation**	**Public**	**Private**
Accuracy	**0.98620**	**0.43626**	**0.43048**	**0.44413**
Epoch	**Best at 155**	**Best at 97**	**at 97**	**at 97**

上面 CNN DNN 結果皆有使用 earlystop , epochs_to_wait_for_improve 設為 50。

可以看到雖然 CNN 跟 DNN 都有大概 7,500,000 左右的參數，但 CNN 可以達到 65%~70%的準確率，而 DNN 只能達到 40%~45%的準確率。

CNN 的 train curve 一直持續較慢的往上升，到了 250epoch 雖然只有 0.72 的準確率，但依然有繼續往上的趨勢，反而 DNN 的 train curve 很快的往上升，未到 100epoch 就達到了大約 0.90 以上的準確率。
CNN 的 valid curve 一直持續波動的往上升，有看出一直緊跟在 train curve 後面 最高有達到 68%的準確率，反而 DNN 的 valid curve 未到 50epoch 就達到了大約 0.40 的準確率，然後就脫離了 train curve，在 0.4 左右起伏。
這次比較可以看出，CNN 使用了 convolution 跟 maxpooling 的確能大大改善機器對於圖形識別的準確度，即使是在參數量接近的情況下，準確率卻與只使用 DNN 有這麼大的差距。

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]



由 confusion matrix 可以看出 Happy 是分得最好的，有 0.89 的準確率。
Fear 是分的最差的，只有 0.48 的準確率。
Angry 比較容易被誤認成 Fear 跟 Neutral 跟 Sad。
Disgust 容易被誤認成 Angry 以及 Fear，但有趣的是 Angry 被誤認成 Disgust 只有 0.1。
Sad 則是容易被分去 Neutral，而也能看到 Neutral 也是最容易被誤認成 Sad。
Surprise 最容易被誤認成 Fear，Fear 也有 0.1 的機會被誤認為 Surprise。
Neutral 則容易被誤認成 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 = (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: filter(2*2*5*6)+bias(6) = 126
Layer B: filter(2*2*6*4)+bias(4) = 100

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

Layer A:
Multiplications: 2*2*5*9*6 = 1080
Additions: ((2*2*5-1)*9)*6 = 1026
Layer B:
Multiplications: 2*2*6*1*4 = 96
Additions: (((3+1)*6)-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 layer, you can use Cl,Cl-1 as lth and l-1th layer)

ANS:

$$O(\sum_{i=1}^{l} c_i * k_i^2 * c_{i-1} * \left\lfloor \frac{n_i + 2p_i - 1}{s} \right\rfloor^2)$$

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)

求出 Cov(x)(即 10 個 samples 的 covariance 加總)的 Eigenvalues 以及 Eigenvectors

(1) What are the principal axes?

1 principal axes (即第一大 eigenvalue 對應的 eigenvector)
= (-0.85214385, -0.02728563, -0.52259579)

2 principal axes (即第二大 eigenvalue 對應的 eigenvector)
= (0.33758926, 0.73439013, -0.58881629)

3 principal axes (即第三大 eigenvalue 對應的 eigenvector)
= (0.39985541, -0.67817891, -0.6165947)

(2) Compute the principal components for each sample.

3 個 eigenvectors 跟(原始 sample-原始 sample 的平均) 的內積 得出的向量即是 principal component

```
(1,2,3)   -> (3.41958007, -4.83186423, 4.85381913)
(4,8,5)   -> (-0.68311652, -0.59038823, 1.08848223)
(3,12,9)  -> (-6.26206634, -0.34568212, -0.25889959)
(1,8,5)   -> (-1.88268276, -1.60315602, 3.64491377)
(5,14,2)  -> (-2.50255045, 5.9199907, 1.64041196)
(7,4,1)   -> (5.69554413, -0.1599158, 0.73157636)
(9,8,9)   -> (-1.15021824, -1.25770707, -5.26262016)
(3,8,1)   -> (1.38340686, 1.42728767, 4.03100923)
(11,5,6)  -> (3.53381339, -1.01925007, -5.3172636)
(10,11,7) -> (-1.55171015, 2.46068517, -5.15142933)
```

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

因為要降到 2D，所以取前兩大 eigenvalues 對應的 eigenvectors
因此，原始 samples 經過 PCA 降到 2D 變成如下，

```
(1,2,3)   -> (3.41958007, -4.83186423)
(4,8,5)   -> (-0.68311652, -0.59038823)
(3,12,9)  -> (-6.26206634, -0.34568212)
(1,8,5)   -> (-1.88268276, -1.60315602)
(5,14,2)  -> (-2.50255045, 5.9199907)
(7,4,1)   -> (5.69554413, -0.1599158)
(9,8,9)   -> (-1.15021824, -1.25770707)
(3,8,1)   -> (1.38340686, 1.42728767)
(11,5,6)  -> (3.53381339, -1.01925007)
(10,11,7) -> (-1.55171015, 2.46068517)
```

再來，Reconstruction 步驟:

令

$$X' = \begin{bmatrix} 3.41958007 & -4.83186423 \\ -0.68311652 & -0.59038823 \\ -6.26206634 & -0.34568212 \\ -1.88268276 & -1.60315602 \\ -2.50255045 & 5.9199907 \\ 5.69554413 & -0.1599158 \\ -1.15021824 & -1.25770707 \\ 1.38340686 & 1.42728767 \\ 3.53381339 & -1.01925007 \\ -1.55171015 & 2.46068517 \end{bmatrix}$$

$$V' = \begin{bmatrix} -0.85214385 & -0.02728563 & -0.52259579 \\ 0.33758926 & 0.73439013 & -0.58881629 \end{bmatrix}$$

$$X'V' = \begin{bmatrix} -0.26384788 & -5.86756048 & 0.73658543 \\ -0.47245656 & 0.02969992 & 0.76883623 \\ -2.62061969 & 3.99293576 & 4.06470016 \\ -1.29400915 & 0.09945377 & 2.10481658 \\ 0.99786696 & 6.04475968 & -1.94272762 \\ 2.22340829 & -3.98003848 & -3.41768128 \\ -0.88450939 & -0.14359391 & 1.44977688 \\ 1.03499971 & 0.10998863 & -1.69341156 \\ 1.06892653 & -3.14508489 & -1.57877955 \\ 0.21024119 & 2.85944 & -0.49211526 \end{bmatrix}$$

$$X_{reconstruction} = X'V' + \bar{X} = \begin{bmatrix} 5.13615212 & 2.13243952 & 5.53658543 \\ 4.92754344 & 8.02969992 & 5.56883623 \\ 2.77938031 & 11.99293576 & 8.86470016 \\ 4.10599085 & 8.09945377 & 6.90481658 \\ 6.39786696 & 14.04475968 & 2.85727238 \\ 7.62340829 & 4.01996152 & 1.38231872 \\ 4.51549061 & 7.85640609 & 6.24977688 \\ 6.43499971 & 8.10998863 & 3.10658844 \\ 6.46892653 & 4.85491511 & 3.22122045 \\ 5.61024119 & 10.85944 & 4.30788474 \end{bmatrix}$$

$$ReconstructionError_1 = ||x_{reconstruction_1} - x_1|| = 4.853819134273743$$

$$ReconstructionError_2 = ||x_{reconstruction_2} - x_2|| = 1.088482230217353$$

$$ReconstructionError_3 = ||x_{reconstruction_3} - x_3|| = 0.258899592159177$$

$$ReconstructionError_4 = ||x_{reconstruction_4} - x_4|| = 3.644913774964859$$

$$ReconstructionError_5 = ||x_{reconstruction_5} - x_5|| = 1.640411955946382$$

$$ReconstructionError_6 = ||x_{reconstruction_6} - x_6|| = 0.7315763560955457$$

$$ReconstructionError_7 = ||x_{reconstruction_7} - x_7|| = 5.262620157658155$$

$$ReconstructionError_8 = ||x_{reconstruction_8} - x_8|| = 4.031009225096187$$

$$ReconstructionError_9 = ||x_{reconstruction_9} - x_9|| = 5.317263601187219$$

$$ReconstructionError_{10} = ||x_{reconstruction_{10}} - x_{10}|| = 5.151429325589517$$

$$ReconstructionError = \frac{1}{10}\sum_{n=1}^{10} ||x_{reconstruction_n} - x_n|| = 3.1980425353188138$$