1.

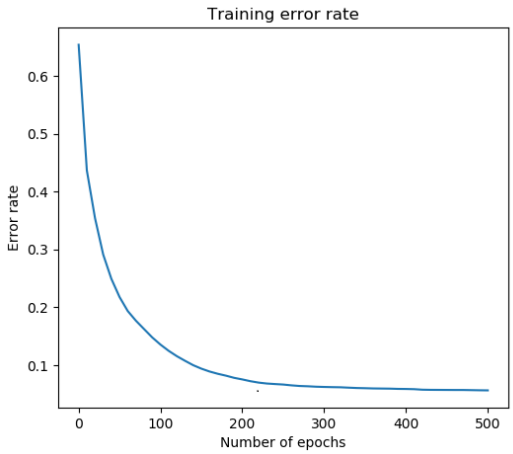
i.

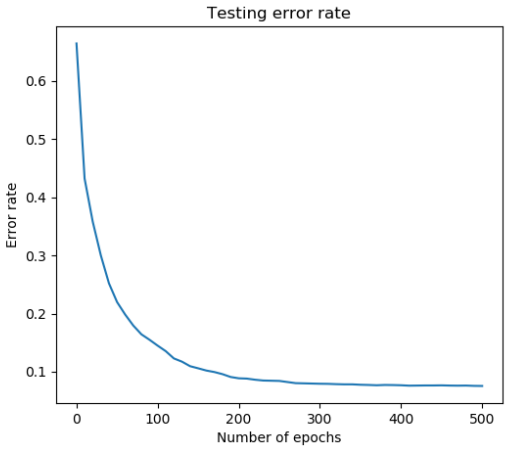
我設計的 Neural network 共為三層，input layer、hidden layer、output layer，input layer 與 output layer 固定為 784 個 neurons 與 10 個 neurons，調整中間的 hidden layer 的 neurons 數，分別記錄下 20、40、80、160、320 的 neurons 數結果。

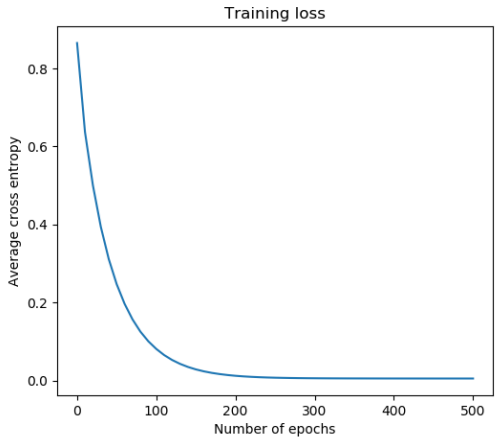
Weight 初始值為 Random initialization 讓每一層的 weight 都是用常態分佈
Bias 初始值為 0

超參數設定 batch size = 10、number of epochs = 500、learning rate = 0.001

當 neural network layer 架構為[784 20 10]:
====== Epoch: 500/500 ======
train error rate: 0.0567
test error rate: 0.0756





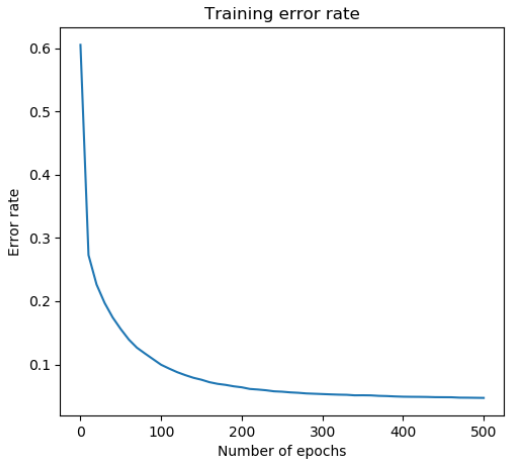


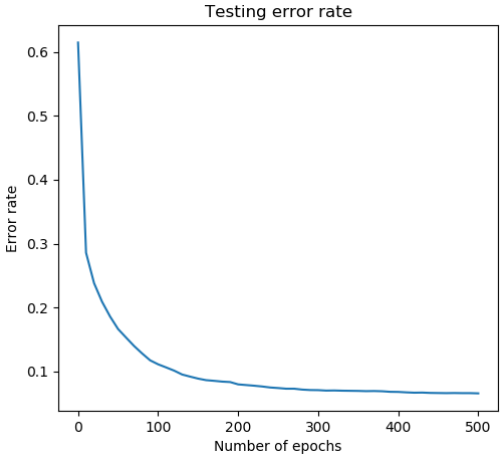
當 neural network layer 架構為[784 40 10]

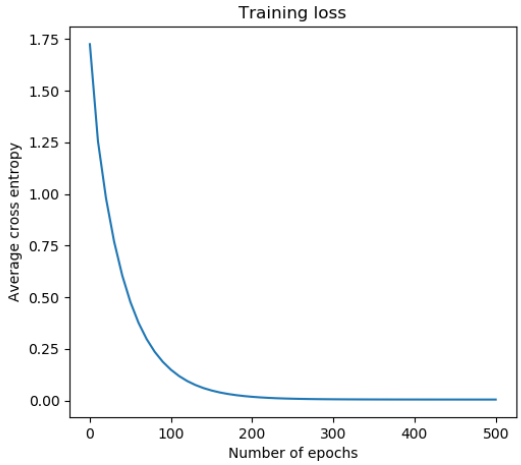
====== Epoch: 500/500 ======

train error rate: 0.0472

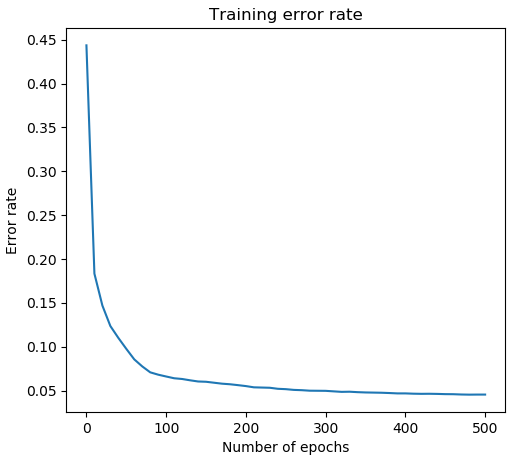
test error rate: 0.0659

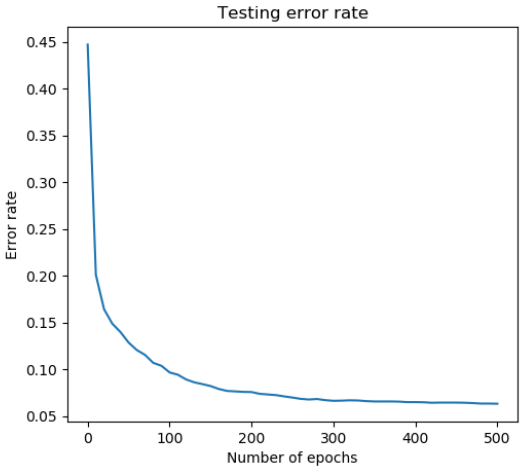


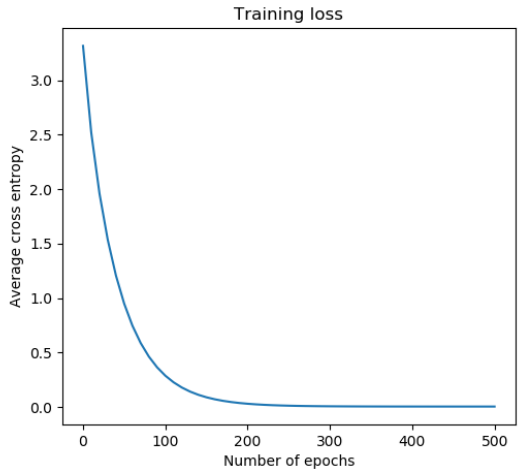




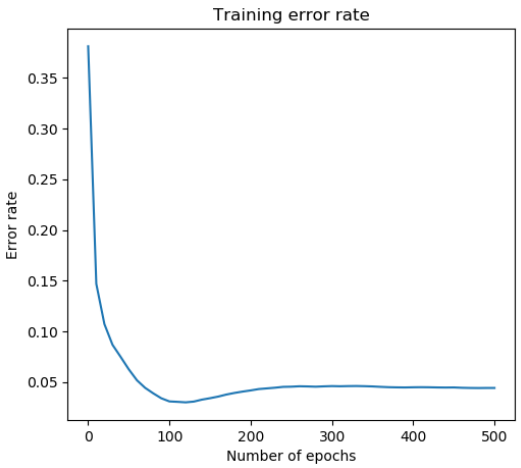
當 neural network layer 架構為[784 80 10]
====== Epoch: 500/500 ======
train error rate: 0.0457
test error rate: 0.0633

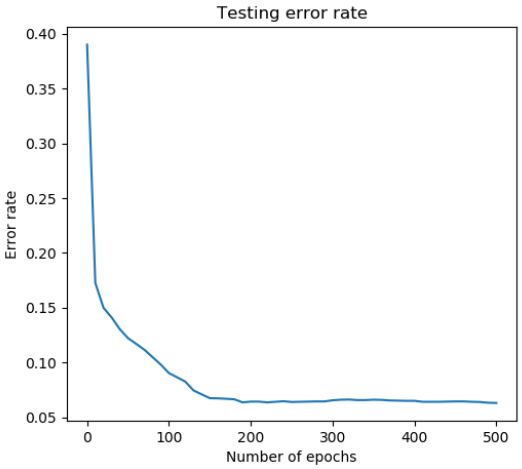


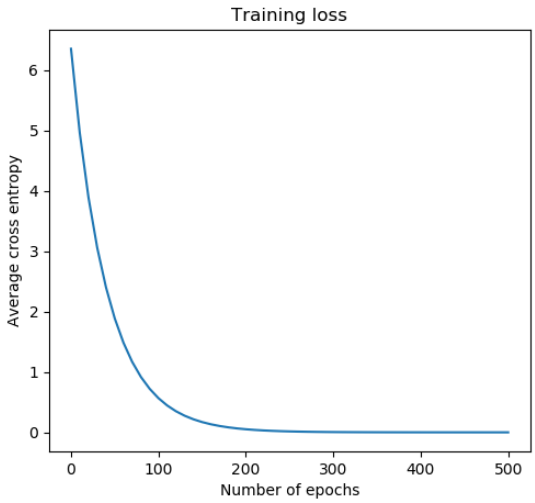




當 neural network layer 架構為[784 160 10]
====== Epoch: 500/500 ======
train error rate: 0.0444
test error rate: 0.0631

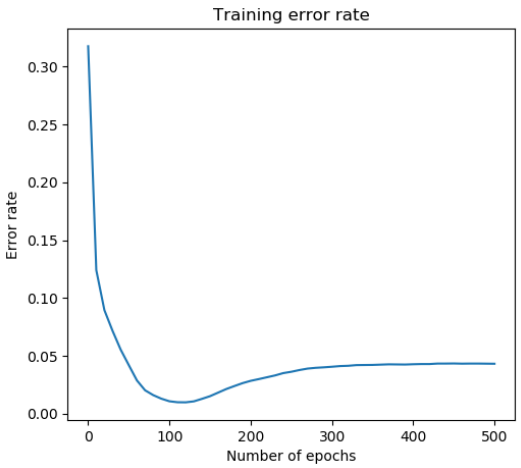


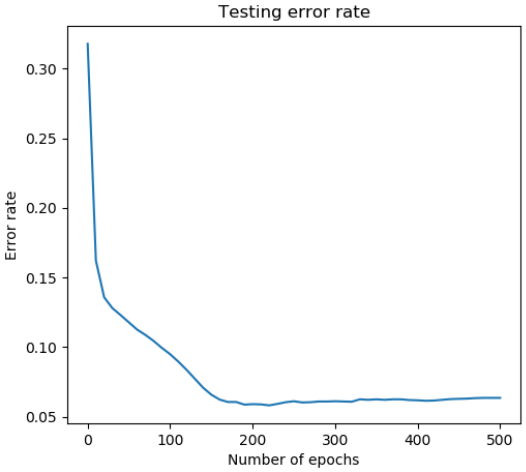


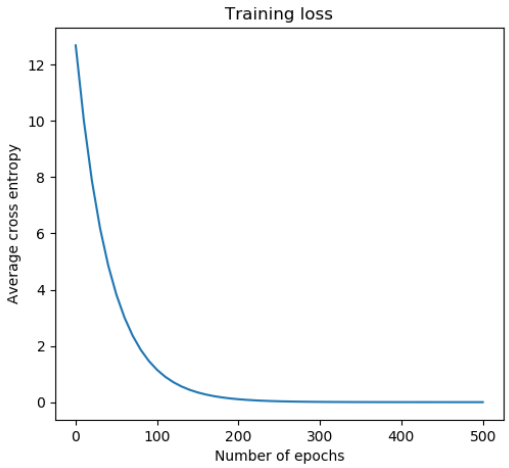


當 neural network layer 架構為[784 320 10]
====== Epoch: 220/500 ======
train error rate: 0.0317
test error rate: 0.0581

====== Epoch: 500/500 ======
train error rate: 0.0433
test error rate: 0.0635







討論

模型越是複雜中間 hidden layer neurons 數量越多，模型練訓的 training error rate 與 testing error rate

下降得就越快，但各模型最後都只會快速下降到某些值後，就呈現極度緩慢下降的 error rate，從實驗結果也發現最複雜的模型[784 320 10]中，最後訓練出的結果反而不是最佳解答，而是在 Epoch: 220/500 時有最低的 training error rate 與 testing error rate，之後更新參數的模型反而越是 overfitting，由此可知調整超參數對於深度學習是非常重要的一環，不同模型因複雜程度的不同、餵入的資料的不同，應調整適合的參數，才可得到最佳的模型。

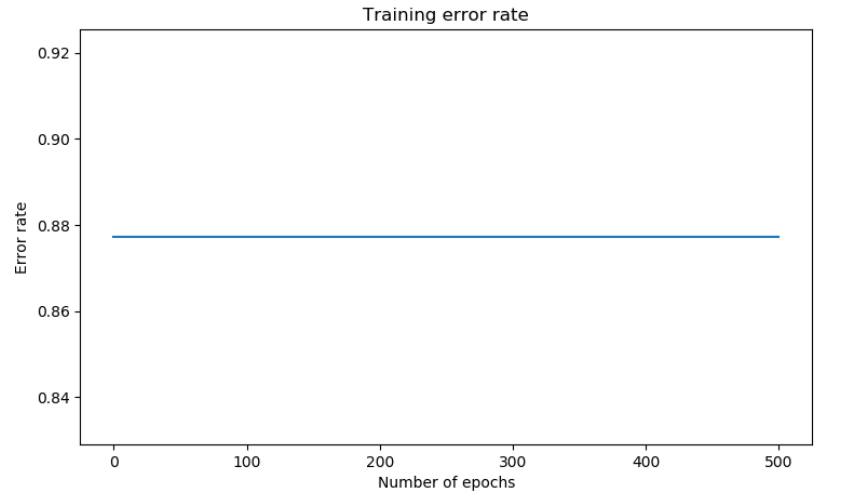
ii.

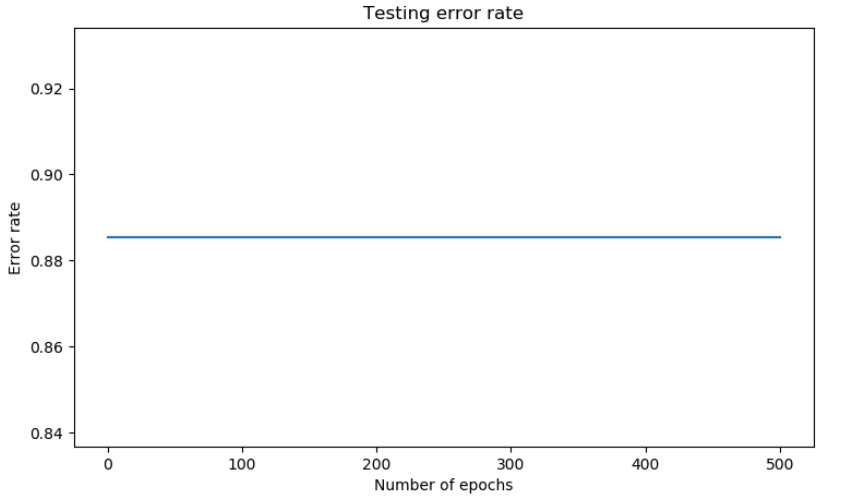
上題實驗結果為 weights 設置為 random initializations，本題測試調整 weights 為 0。

Weight 初始值為 0
Bias 初始值為 0

超參數設定 batch size = 10、number of epochs = 500、learning rate = 0.001
neural network layer 架構為[784 20 10]

實驗結果:





討論

以基礎公式為主的討論:

當 weight 初始值是 0 時，在做 forward 的結果，不管你訓練資料怎麼丟進模型中都會是 0 (W1 = W2 = 0)。

$$\hat{y} = w_1 x_1 + w_2 x_2 = 0$$

神經網路的 weight 是利用 Stochastic Gradient Descent 更新。

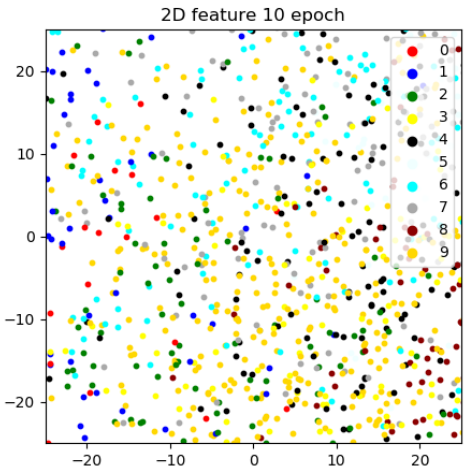
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$

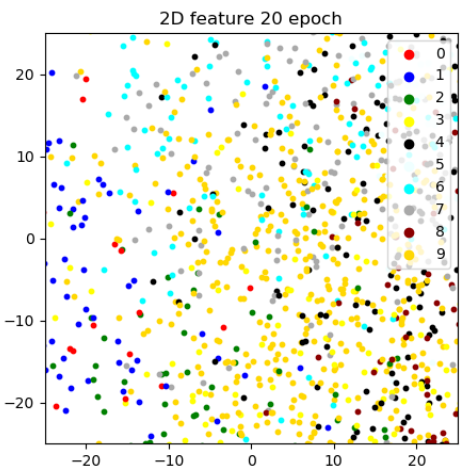
由上可知 Gradient Descent 不管怎麼做都是 0，weight 的更新不管更新幾次結果都一樣，如實驗結果 training error rate 與 testing error rate 都沒有變化。

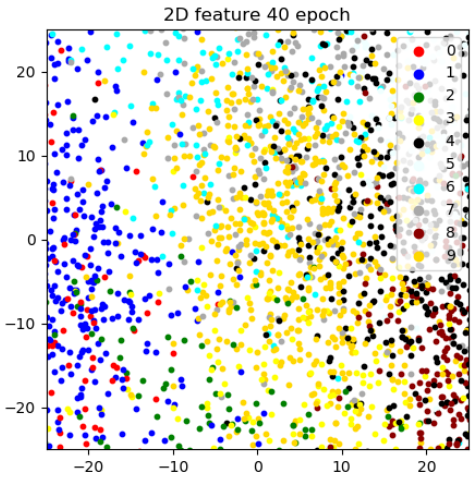
iii.

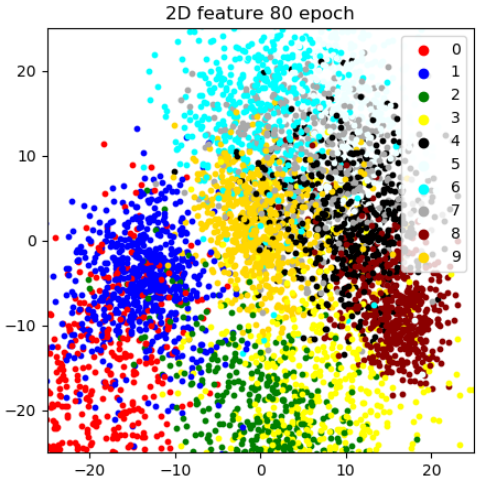
超參數設定 batch size = 10、number of epochs = 500、learning rate = 0.001
Weight 初始值為 Random initialization 讓每一層的 weight 都是用常態分佈
Bias 初始值為 0

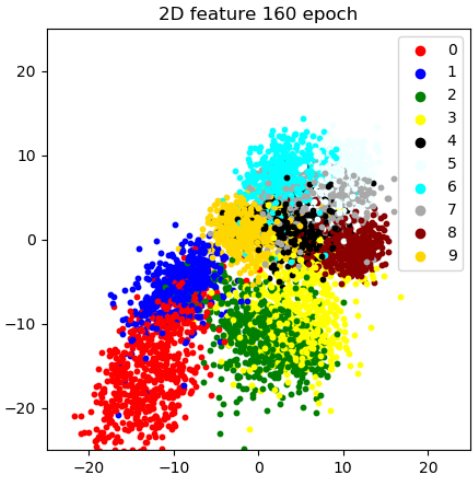
這題我使用的 neural network layer 架構為[784 40 2 10]並取出第二層 hidden layer(兩個 neurons)的值，當作平面 x,y 兩座標繪圖，並記錄 10、20、40、80、160、320、500 epochs 時的狀態。

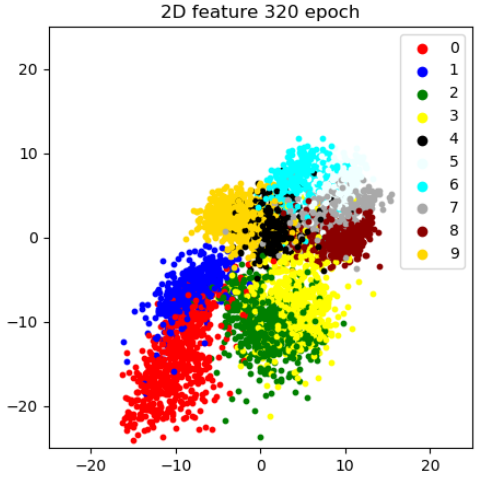


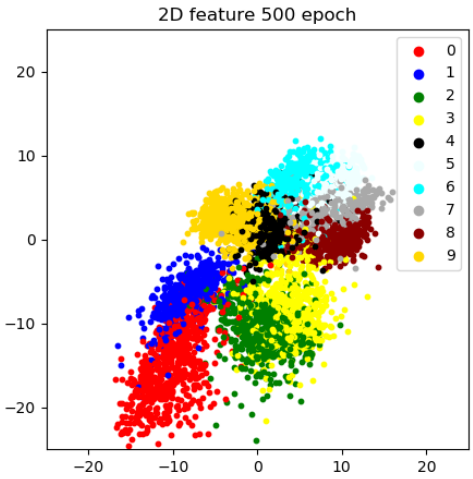



2D feature 40 epoch
0
1
2
3
4
5
6
7
8
9
20
10
0
−10
−20
−20
−10
0
10
20


2D feature 80 epoch
0
1
2
3
4
5
6
7
8
9
20
10
0
−10
−20
−20
−10
0
10
20


2D feature 160 epoch
0
1
2
3
4
5
6
7
8
9
20
10
0
−10
−20
−20
−10
0
10
20

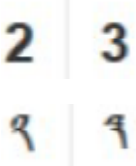

2D feature 320 epoch
0
1
2
3
4
5
6
7
8
9
20
10
0
−10
−20
−20
−10
0
10
20



討論

前面提到模型在訓練初期時，training error rate 與 testing error rate 會快速下降，我們也可以從圖上發現 epoch 在 10~160 的圖中，圖上的 10 種點快速變化，各值間明顯的分類出來，而 160~500 後的圖形，變化較不明顯，training error rate 與 testing error rate 下降的幅度變小。

另外也可以發現圖上各個群聚的點代表如 2、3 有部分可視的點重疊率很高，可以從講義上提供的藏文數值上



發現兩值手寫確實蠻相似的，所以特徵值上有些許接近，

越接近的兩類群，在肉眼視手寫值上，也確實有部分筆畫特徵相似

iv.

分別記錄下不同 neural network 架構的模型，對 train data 與 test data 的 crosstab

當 neural network layer 架構為[784 20 10]:
train error rate: 0.0567
test error rate: 0.0756

```
====== test data crosstab ======
predict    0    1    2    3    4    5    6    7    8    9
labels
0.0      643   16    4    0    0    0    1    0    0    0
1.0        4  652    2    0    0    0    0    0    0    3
2.0        7   18  516   43    0    0    0    0    0    0
3.0        1    5   42  526    2    3    7    3   11    0
4.0        1    0    0    1  576    1   15    1   35   21
5.0        0    0    0    5    2  382    3   10    3    2
6.0        0    0    0    1    4   10  474    9    3    1
7.0        2    0    0    2    8    9   13  409    3    3
8.0        0    0    0    4   17    3    1    6  541    0
9.0        0    3    1    5   43    3    4    3    3  613
====== train data crosstab ======
predict     0     1     2     3     4    5     6     7     8     9
labels
0.0      1291    35     6     0     0    0     0     0     0     0
1.0         6  1460     3     1     0    0     0     0     0     3
2.0        10    28  1185    63     0    0     0     1     0     1
3.0         0     4    47  1123     5    2     1    16    17     1
4.0         1     0     0     0  1213    0    11     1    48    42
5.0         0     0     0     3     7  880    21    16     9     7
6.0         0     0     0     2    11   20  1061    11     7     4
7.0         0     0     2     5    12    7    23   794     3     7
8.0         0     1     0     9    28    4     3    10  1015     0
9.0         0     2     1     7    67    4     3     6     5  1298
```

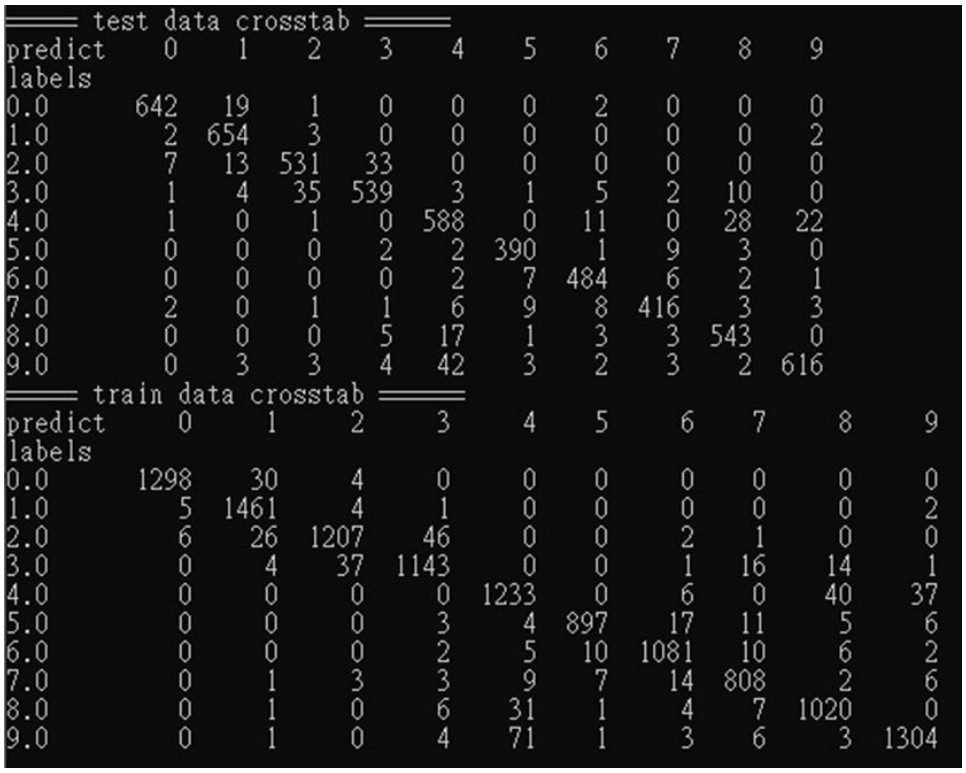
當 neural network layer 架構為[784 40 10]:

train error rate: 0.0472

test error rate: 0.0659

```
====== test data crosstab ======
predict    0    1    2    3    4    5    6    7    8    9
labels
0.0      644   16    2    0    0    0    2    0    0    0
1.0        2  655    2    0    0    0    0    0    0    2
2.0        7   13  529   35    0    0    0    0    0    0
3.0        1    4   37  539    2    1    2    5    9    0
4.0        1    0    1    1  585    0   12    0   28   23
5.0        0    0    0    3    2  386    1   11    3    1
6.0        0    0    0    2    2    8  481    7    1    1
7.0        2    0    1    0    8    9    8  415    4    2
8.0        0    0    0    6   17    3    2    4  540    0
9.0        0    4    4    4   43    2    3    2    2  614
====== train data crosstab ======
predict     0     1     2     3     4    5     6     7     8     9
labels
0.0      1297    30     3     0     0    0     1     0     0     1
1.0         5  1461     3     1     0    0     0     0     0     3
2.0         4    25  1207    50     0    0     0     2     0     0
3.0         0     3    45  1134     0    0     1    16    15     2
4.0         1     0     0     0  1229    0     4     1    42    39
5.0         0     0     0     4     4  894    13    14     8     6
6.0         0     0     0     2     4    9  1083    11     5     2
7.0         0     1     1     4     8    8    12   812     3     4
8.0         0     1     0     7    29    3     4     9  1017     0
9.0         0     2     0     6    73    1     5     3     3  1300
```

當 neural network layer 架構為[784 80 10]:
train error rate: 0.0457
test error rate: 0.0633



```
==== test data crosstab ====
predict    0    1    2    3    4    5    6    7    8    9
labels
0.0      642   19    1    0    0    0    2    0    0    0
1.0        2  654    3    0    0    0    0    0    0    2
2.0        7   13  531   33    0    0    0    0    0    0
3.0        1    4   35  539    3    1    5    2   10    0
4.0        1    0    1    0  588    0   11    0   28   22
5.0        0    0    0    2    2  390    1    9    3    0
6.0        0    0    0    0    2    7  484    6    2    1
7.0        2    0    1    1    6    9    8  416    3    3
8.0        0    0    0    5   17    1    3    3  543    0
9.0        0    3    3    4   42    3    2    3    2  616
==== train data crosstab ====
predict     0     1     2     3     4    5     6    7     8     9
labels
0.0      1298    30     4     0     0    0     0    0     0     0
1.0         5  1461     4     1     0    0     0    0     0     2
2.0         6    26  1207    46     0    0     2    1     0     0
3.0         0     4    37  1143     0    0     1   16    14     1
4.0         0     0     0     0  1233    0     6    0    40    37
5.0         0     0     0     3     4  897    17   11     5     6
6.0         0     0     0     2     5   10  1081   10     6     2
7.0         0     1     3     3     9    7    14  808     2     6
8.0         0     1     0     6    31    1     4    7  1020     0
9.0         0     1     0     4    71    1     3    6     3  1304
```

當 neural network layer 架構為[784 160 10]:
train error rate: 0.0444
test error rate: 0.0631

```
====== test data crosstab ======
predict    0    1    2    3    4    5    6    7    8    9
labels
0.0      643   17    2    0    0    0    2    0    0    0
1.0        2  655    2    0    0    0    0    0    0    2
2.0        7   13  530   34    0    0    0    0    0    0
3.0        1    3   36  542    1    1    5    3    8    0
4.0        1    0    1    0  588    0    9    0   29   23
5.0        0    0    0    3    2  388    1    9    3    1
6.0        0    0    0    1    2    8  480    7    3    1
7.0        2    0    2    2    6    9    9  415    3    1
8.0        0    0    0    4   18    2    4    4  540    0
9.0        0    3    4    3   36    2    2    3    2  623
====== train data crosstab ======
predict     0     1     2     3     4    5     6     7     8     9
labels
0.0      1301    26     4     0     0    0     0     0     0     1
1.0         5  1463     2     1     0    0     0     0     0     2
2.0         6    23  1210    47     0    0     1     1     0     0
3.0         0     3    39  1148     0    0     1    13    11     1
4.0         0     0     1     0  1231    0     5     0    41    38
5.0         0     0     0     4     4  894    14    15     6     6
6.0         0     0     0     2     5   12  1078    11     6     2
7.0         0     0     2     4     9    5    16   810     2     5
8.0         0     1     0     7    28    2     5     7  1020     0
9.0         0     1     0     4    64    1     3     5     3  1312
```

當 neural network layer 架構為[784 320 10]:

train error rate: 0.0433

test error rate: 0.0635

```
====== test data crosstab ======
predict    0    1    2    3    4    5    6    7    8    9
labels
0.0      643   18    1    0    0    0    2    0    0    0
1.0        2  654    3    0    0    0    0    0    0    2
2.0        6   12  529   37    0    0    0    0    0    0
3.0        1    2   36  541    3    1    4    3    8    1
4.0        1    0    1    0  587    0   10    0   29   23
5.0        0    0    0    2    2  389    1    9    4    0
6.0        0    0    0    1    2    8  482    6    2    1
7.0        2    0    1    1    8    9    9  416    2    1
8.0        0    0    0    5   17    2    4    4  540    0
9.0        0    3    4    4   37    2    2    3    2  621
====== train data crosstab ======
predict     0     1     2     3     4    5     6     7     8     9
labels
0.0      1300    26     4     0     0    0     1     0     0     1
1.0         4  1464     2     1     0    0     0     0     0     2
2.0         6    22  1210    48     0    0     1     1     0     0
3.0         0     3    40  1146     0    0     1    14    11     1
4.0         1     0     0     0  1235    0     4     0    40    36
5.0         0     0     0     3     3  900    13    12     6     6
6.0         0     0     0     2     5    9  1082    10     6     2
7.0         0     0     2     4     9    6    15   811     2     4
8.0         0     1     0     7    27    1     5     7  1022     0
9.0         0     2     1     5    63    1     3     5     3  1310
```

討論

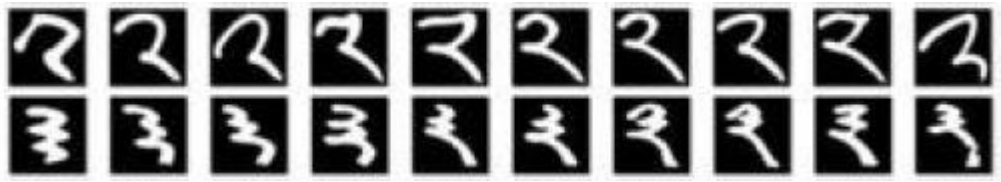
實驗數據發現，模型在辨識 2、3 時的出錯率較高，由上一題的繪圖座標也能發現 2、3 有較相近的特徵值，部分的點也有些重疊。

	0	1	2	3	4	5	6	7	8	9
0	659	2	1	0	0	0	2	0	0	0
1	2	657	2	0	0	0	0	0	0	0
2	2	8	548	26	0	0	0	0	0	0
3	2	1	33	550	2	1	4	4	3	0
4	1	0	1	2	611	0	6	0	9	21
5	0	0	0	2	1	398	2	4	0	0
6	1	0	0	1	1	3	489	6	1	0
7	2	0	0	1	4	5	2	431	1	3
8	0	0	0	1	7	1	3	0	560	0
9	0	2	1	0	18	3	1	3	1	649

跟講義上的 crosstab 表比對後，我模型生成的 crosstab 在辨識結果出錯較多的也和講義上差不多。

2 與 3，4 與 9 的預測上，出錯最高

2 與 3:



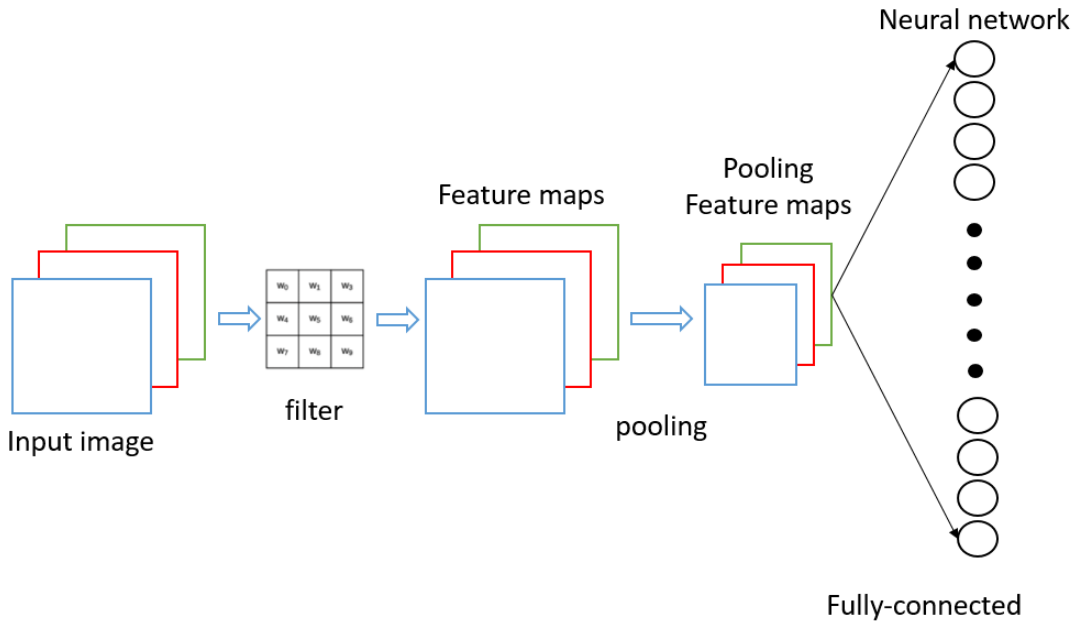
4 與 9

兩組值，也確實在部分特徵筆畫上有些相似。

2.

i. 首先根據 train.csv 與 test.csv 檔案中各張圖片的臉部座標去裁減 image 中的各張臉，由於每張裁減的臉部圖片大小不一樣，所以需要做 resize 把圖片大小統一，我統一的大小為 64*64，我使用的工具為 python cv2。

根據 csv 檔的標籤(good, bad, none)，給各張圖一個 label，接著再做 Convolutional。



Input image 64*64 經過 3x3 filter 後，形成 62*62 的 feature maps，再做 pooling (每 2*2 中，選出最大值)，成 31*31 的 Pooling feature maps 再 reshape 成 1*961 對神經網路做全連結，因為圖片有 rgb 性質，所以第一層的 Neural network 有 961*3 = 2883 個 nodes。

ii.

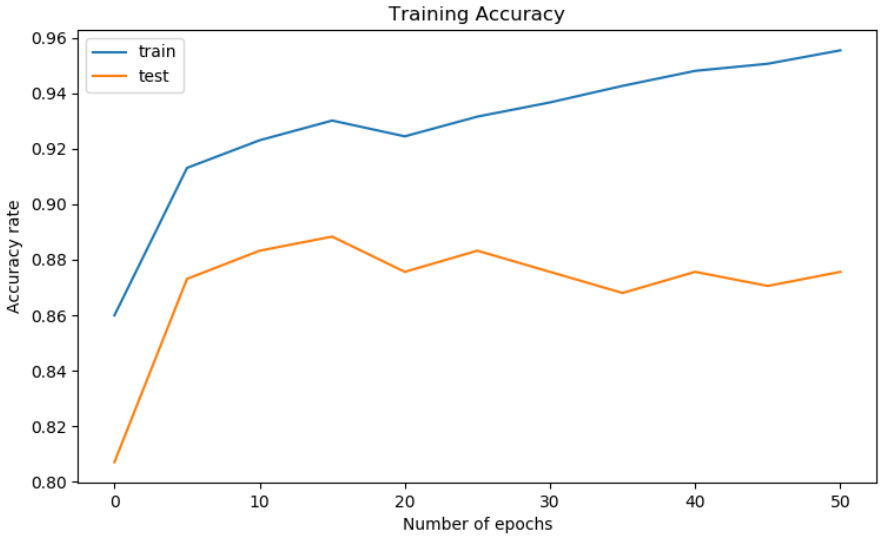
Weight 初始值為 Random initialization 讓每一層的 weight 都是用常態分佈
Bias 初始值為 0

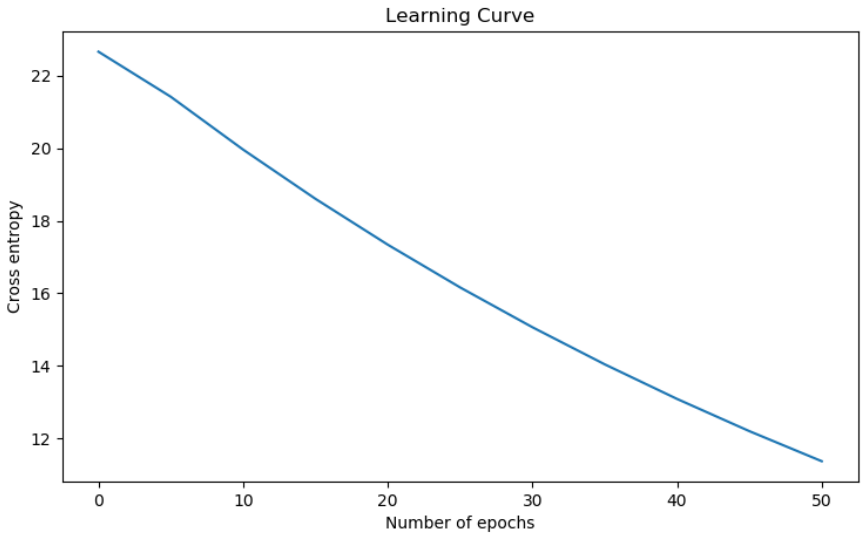
超參數設定 batch size = 5、number of epochs = 50、learning rate = 0.001

Filter:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

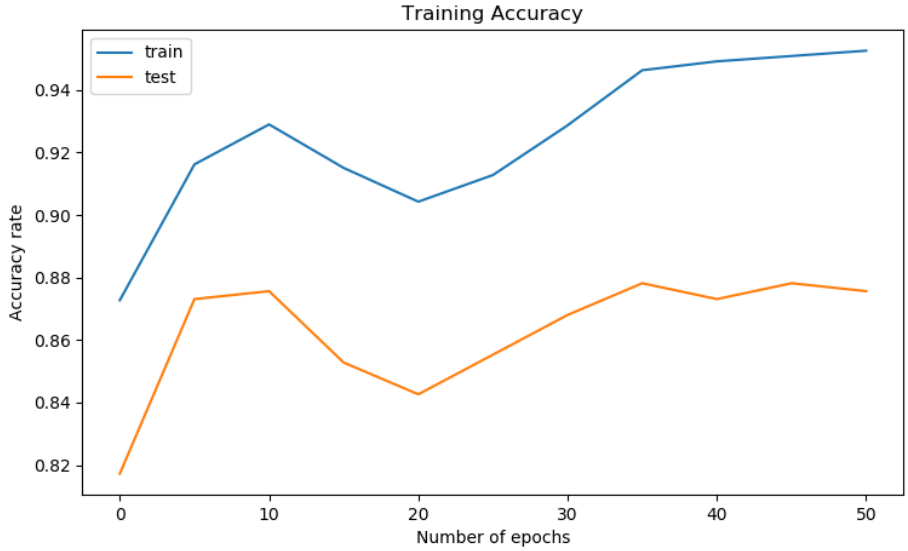
neural network layer 架構為[2883 160 3]:

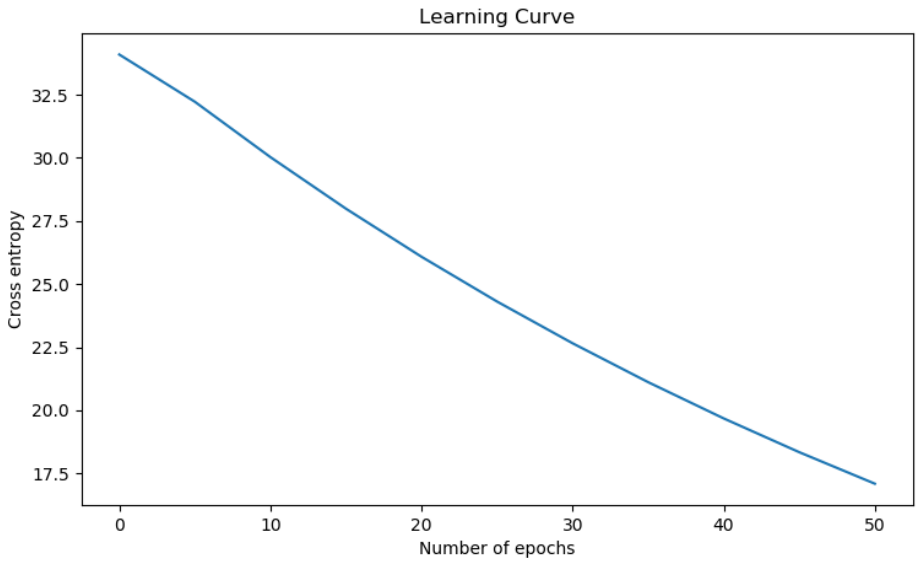




====== Epoch: 1/50 ======
Loss: 22.668
train accuracy rate: 0.8599
test accuracy rate: 0.8071
====== Epoch: 10/50 ======
Loss: 19.966
train accuracy rate: 0.9230
test accuracy rate: 0.8832
====== Epoch: 20/50 ======
Loss: 17.342
train accuracy rate: 0.9244
test accuracy rate: 0.8756
====== Epoch: 30/50 ======
Loss: 15.064
train accuracy rate: 0.9366
test accuracy rate: 0.8756
====== Epoch: 40/50 ======
Loss: 13.085
train accuracy rate: 0.9480
test accuracy rate: 0.8756
====== Epoch: 50/50 ======
Loss: 11.366
train accuracy rate: 0.9554
test accuracy rate: 0.8756

neural network layer 架構為[2883 240 3]:





====== Epoch: 1/50 ======
Loss: 34.093
train accuracy rate: 0.8727
test accuracy rate: 0.8173
====== Epoch: 10/50 ======
Loss: 30.030
train accuracy rate: 0.9290
test accuracy rate: 0.8756
====== Epoch: 20/50 ======
Loss: 26.084
train accuracy rate: 0.9043
test accuracy rate: 0.8426
====== Epoch: 30/50 ======
Loss: 22.657
train accuracy rate: 0.9287
test accuracy rate: 0.8680
====== Epoch: 40/50 ======
Loss: 19.681
train accuracy rate: 0.9491
test accuracy rate: 0.8731
====== Epoch: 50/50 ======
Loss: 17.096
train accuracy rate: 0.9526
test accuracy rate: 0.8756

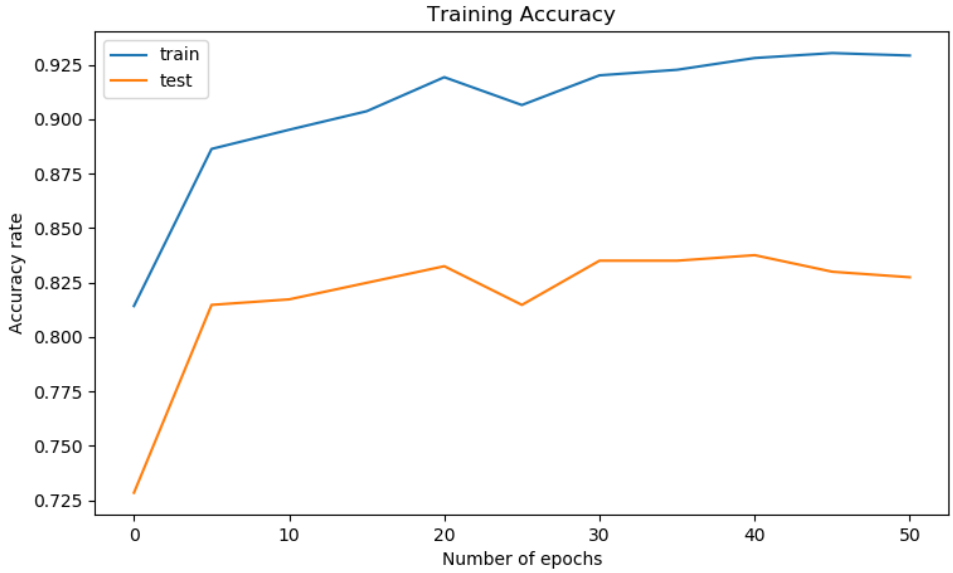
Weight 初始值為 Random initialization 讓每一層的 weight 都是用常態分佈
Bias 初始值為 0

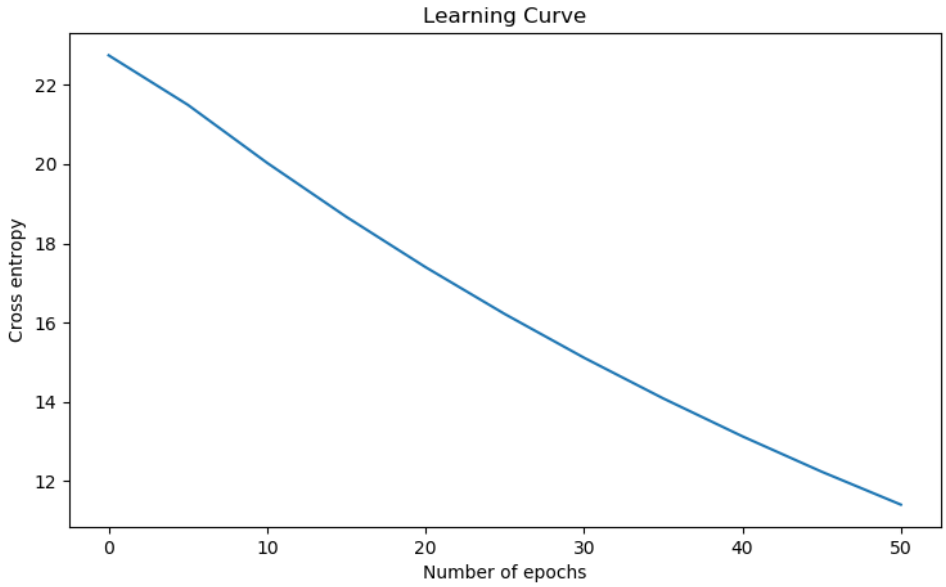
超參數設定 batch size = 5、number of epochs = 50、learning rate = 0.001

Filter:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

neural network layer 架構為[2883 160 3]:





====== Epoch: 1/50 ======
Loss: 22.748
train accuracy rate: 0.8142
test accuracy rate: 0.7284
====== Epoch: 10/50 ======
Loss: 20.034
train accuracy rate: 0.8952
test accuracy rate: 0.8173
====== Epoch: 20/50 ======
Loss: 17.401
train accuracy rate: 0.9193
test accuracy rate: 0.8325
====== Epoch: 30/50 ======
Loss: 15.116
train accuracy rate: 0.9202
test accuracy rate: 0.8350
====== Epoch: 40/50 ======
Loss: 13.131
train accuracy rate: 0.9281
test accuracy rate: 0.8376
====== Epoch: 50/50 ======
Loss: 11.407
train accuracy rate: 0.9293
test accuracy rate: 0.8274

討論

本題我嘗試使用兩種不同 filter 與調整 hidden layer 的 nodes 數量觀察結果，使用 sharpen 的 filter

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

訓練結果較使用

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

來的差，我想是因為銳化後的圖片，特徵的保留較原圖來得少，使得訓練結果較差。

這次的模型，沒有因為 epochs 數量的增加使 accuracy rate 增加，應該是我的圖片前處理沒有找到最佳的方式，讓特徵更好的表現出來。

iii.

Weight 初始值為 Random initialization 讓每一層的 weight 都是用常態分佈
Bias 初始值為 0

超參數設定 batch size = 5、number of epochs = 50、learning rate = 0.001

Filter:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

neural network layer 架構為[2883 160 3]:
====== Epoch: 1/50 ======

train accuracy of good: 93.08%
train accuracy of bad: 64.71%
train accuracy of none: 3.85%
test accuracy of good: 91.87%
test accuracy of bad: 65.17%
test accuracy of none: 0.00%
====== Epoch: 10/50 ======
train accuracy of good: 94.73%
train accuracy of bad: 92.04%
train accuracy of none: 20.19%
test accuracy of good: 94.70%
test accuracy of bad: 87.64%
test accuracy of none: 9.09%
====== Epoch: 30/50 ======
train accuracy of good: 94.98%
train accuracy of bad: 94.81%
train accuracy of none: 44.23%
test accuracy of good: 93.29%
test accuracy of bad: 85.39%
test accuracy of none: 22.73%
====== Epoch: 50/50 ======
train accuracy of good: 97.12%
train accuracy of bad: 92.04%
train accuracy of none: 64.42%
test accuracy of good: 93.29%
test accuracy of bad: 82.02%
test accuracy of none: 36.36%
neural network layer 架構為[2883 240 3]:
====== Epoch: 1/50 ======
train accuracy of good: 95.22%
train accuracy of bad: 61.76%
train accuracy of none: 4.81%
test accuracy of good: 93.29%
test accuracy of bad: 61.80%
test accuracy of none: 13.64%
====== Epoch: 10/50 ======
train accuracy of good: 95.92%
train accuracy of bad: 89.27%

train accuracy of none: 23.08%
test accuracy of good: 93.99%
test accuracy of bad: 83.15%
test accuracy of none: 22.73%
====== Epoch: 30/50 ======
train accuracy of good: 95.01%
train accuracy of bad: 85.29%
train accuracy of none: 69.23%
test accuracy of good: 91.17%
test accuracy of bad: 83.15%
test accuracy of none: 45.45%
====== Epoch: 50/50 ======
train accuracy of good: 96.13%
train accuracy of bad: 96.02%
train accuracy of none: 59.62%
test accuracy of good: 91.52%
test accuracy of bad: 91.01%
test accuracy of none: 22.73%

Weight 初始值為 Random initialization 讓每一層的 weight 都是用常態分佈
Bias 初始值為 0

超參數設定 batch size = 5、number of epochs = 50、learning rate = 0.001

Filter:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

neural network layer 架構為[2883 160 3]:
====== Epoch: 1/50 ======
train accuracy of good: 95.75%
train accuracy of bad: 23.70%
train accuracy of none: 3.85%
test accuracy of good: 96.11%

test accuracy of bad: 14.61%
test accuracy of none: 9.09%
====== Epoch: 10/50 ======
train accuracy of good: 92.94%
train accuracy of bad: 81.14%
train accuracy of none: 35.58%
test accuracy of good: 88.69%
test accuracy of bad: 74.16%
test accuracy of none: 22.73%
====== Epoch: 30/50 ======
train accuracy of good: 94.45%
train accuracy of bad: 86.51%
train accuracy of none: 49.04%
test accuracy of good: 89.75%
test accuracy of bad: 77.53%
test accuracy of none: 27.27%
====== Epoch: 50/50 ======
train accuracy of good: 94.38%
train accuracy of bad: 93.94%
train accuracy of none: 40.38%
test accuracy of good: 88.69%
test accuracy of bad: 83.15%
test accuracy of none: 4.55%

(1)

根據數據，train accuracy of none 的辨識正確率最差，我想可能原因有 none 的 train data 本身比較少，無法訓練出好的模型。

根據圖片分析，none 是沒將口罩戴好的樣本，這樣的樣本特徵部位較小，使再做前處理時，更容易把特徵縮小，最後可能辨識成 bad(沒口罩)。

(2)

1.增加 train data 數量

2.選擇出更好的 filter 與 pooling 方式

3.可嘗試更深層的 Neural network

(3)

這次模型訓練結果 overfitting 比第一題(藏文辨識)的結果更加嚴重，test accuracy rate 都比 train accuracy rate 來得低，我想是因為圖片轉換後為 3 維矩陣，高維的數據會增加模型訓練的難度，須更顧及前處裡與 neural network 的調整，