深度學習 Homework 01

統計所 0852617 曾鈺評

Due date : 4/19/2020

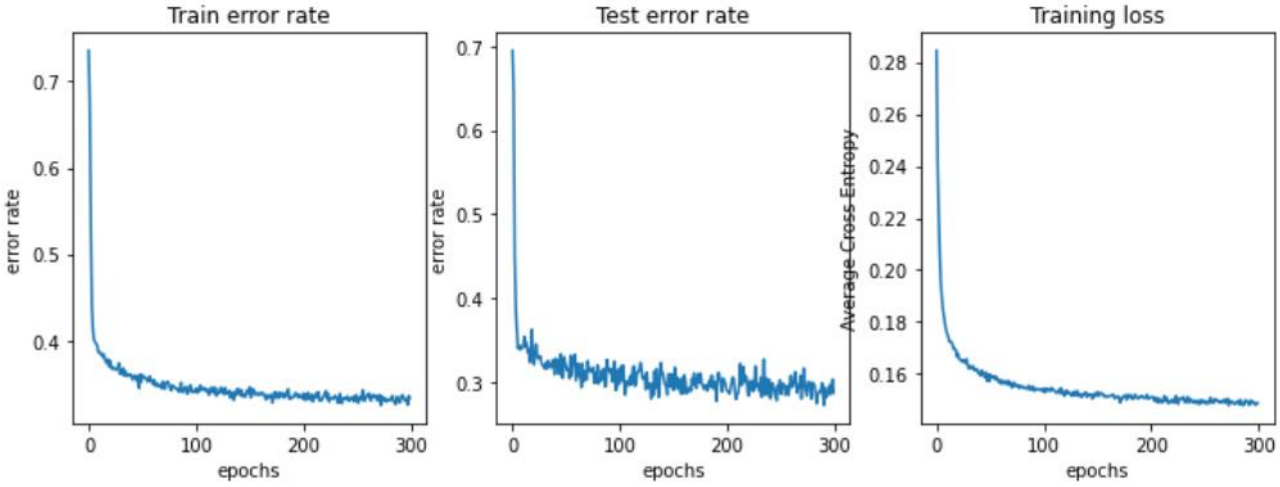
1. Deep Neural Network for Classification

I. Please construct a DNN for classification. For N samples and K categories. Please minimize the objective function E(w) by running the **error backpropagation** algorithm using the **Stochastic Gradient Descent** (SGD). You should decide the following hyperparameters: number of hidden layers, number of hidden units, learning rate, number of iterations and mini-batch size. You have to show your **(a) learning curve, (b) training error rate** and **(c) test error rate** in the report. You should design the network architecture by yourself.

(1) 在最一開始的 model 中，本來先設一層只有 2 個 units 的 hidden layer，不過發現到效果很差，故我們以下皆選擇使用 2 層 hidden layer，在以下的 model 中 hidden units 先用較為簡單的只有兩個節點的情況下去做

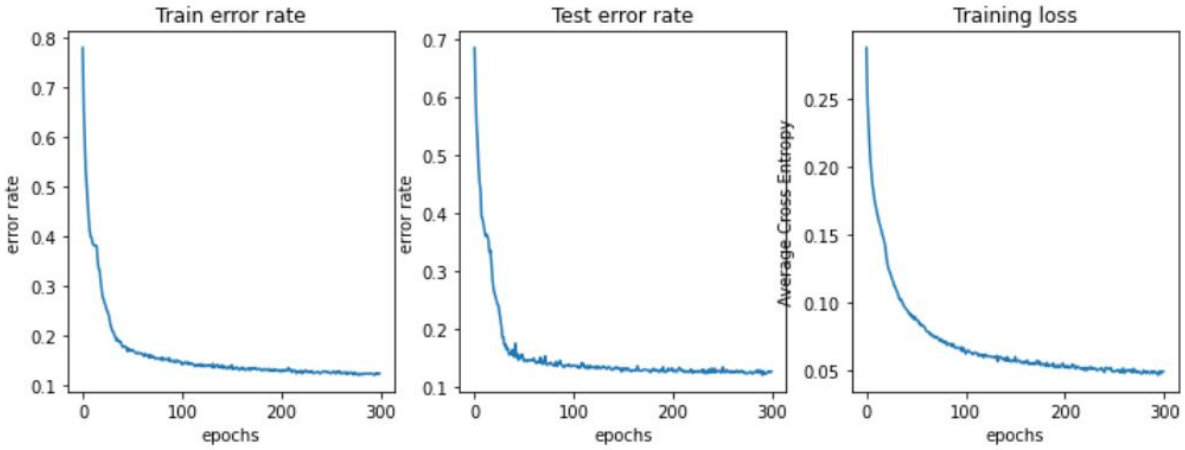
number of hidden layers	2
number of 1st hidden units	2
number of 2nd hidden units	2
learning rate	0.05
number of epochs	300
Number of mini-batch size	100

從下圖結果可發現，training loss 其實還有下降的趨勢，training error rate 和 testing error rate 最高大約落在 30%附近，最可能要再多跑幾個 epoch 才會達到比較好的成效。



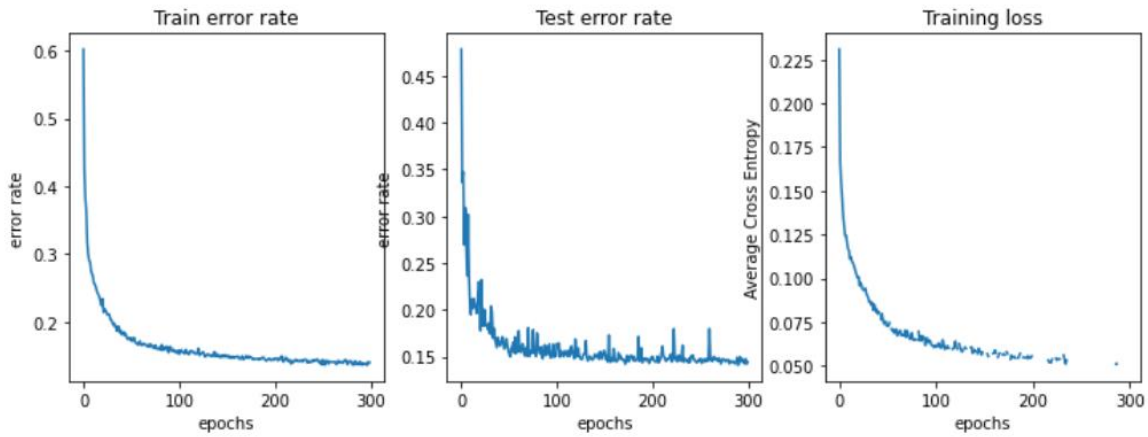
(2) 嘗試增加 hidden units 的個數，去看效果如何。將第一層的 hidden units 增加到 500，可發現其 error rate 降低很多，大約在 13%左右。

number of hidden layers	2
number of 1st hidden units	500
number of 2nd hidden units	2
learning rate	0.05
number of epochs	300
Number of mini-batch size	100



(3) 將 mini batch size 由原來的 100 個改變為 10 個，我們可以發現 error rate 比前一步驟略為上升，大約在 15%，且改變的趨勢反而變得比較不穩定。此外，可注意到的是，因為 bini match size 太小，造成 model 在 train 的過程中不穩定，有時候 loss 會因為 overflow 的緣故而無法計算出，所以我們可以嘗試在 log 公式裡加上一個極小的數值，例如 10^{-9}，以幫助避免程式出現 error 訊息。

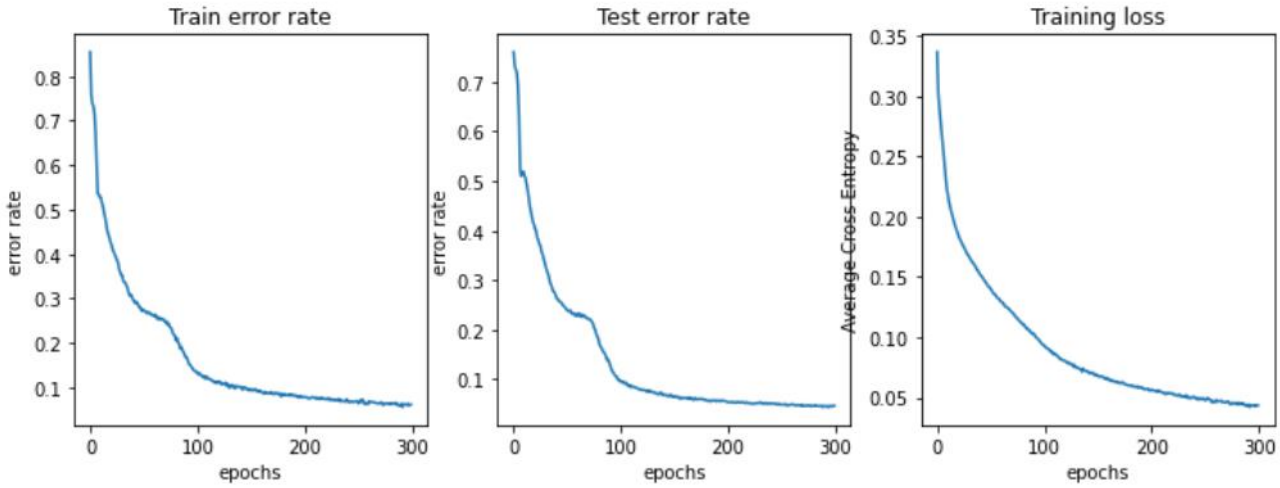
number of hidden layers	2
number of 1st hidden units	500
number of 2nd hidden units	2
learning rate	0.05
number of epochs	300
Number of mini-batch size	10



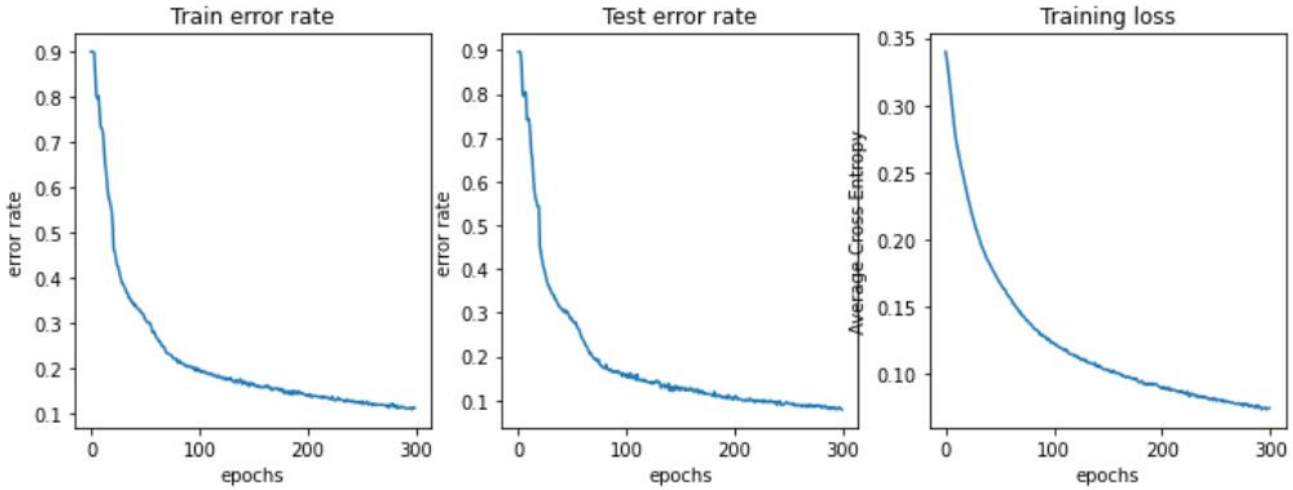
(4) Learning rate 初始值設為 0.05，以下我們改變 learning rate 調降到 0.01、0.005 與 0.001：

number of hidden layers	2
number of 1st hidden units	500
number of 2nd hidden units	2
learning rate	0.01, 0.005, 0.001
number of epochs	300
Number of mini-batch size	100

A. 當 learning rate 為 0.01 時，error rate 僅有 7%，比原來 learning rate 設定在 0.05 的時候來的更好，我們推測是因為 learning rate 變小以後，使 model 不會走的太快而跳過最 local minimum，而且從 loss 看起來是還有進步空間的。

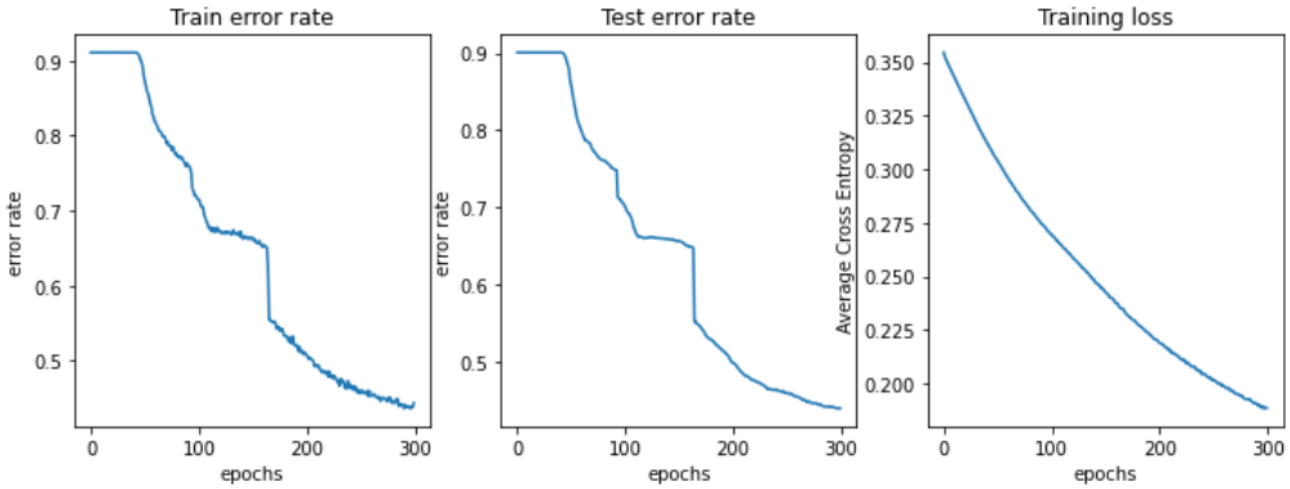


B. 當 learning 為 0.005 時，accuracy 有 12%，與 learning rate 為 0.05 的情況差不多，從 loss 看起來是還有進步空間的。



C. 當 learning 為 0.001 時，會使整個 training 的過程變得極為緩慢，因此可以發現最後 error rate 仍有 45%左右，且整體的 loss 其實還有下降的空間，如果能跑更多

epoch，應該還會有進步的空間，但過小的 learning rate 也會造成學習速度緩慢。

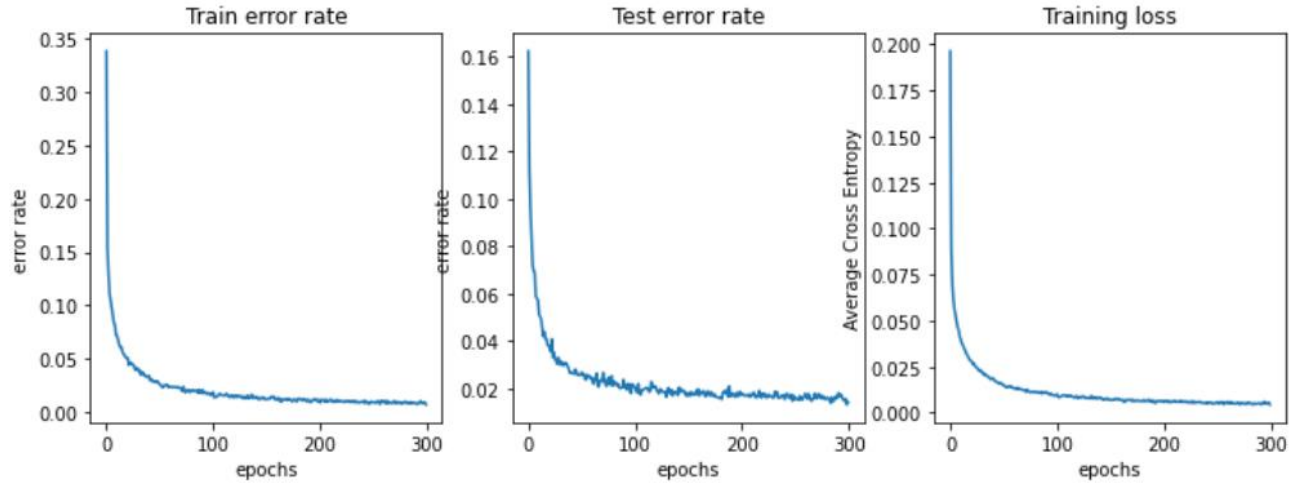


針對以上結果我們可推論，當 learning rate 過大會造成 model 不穩定，而過小則會學習過程過於緩慢，當記憶體或時間不足便無法完成訓練。

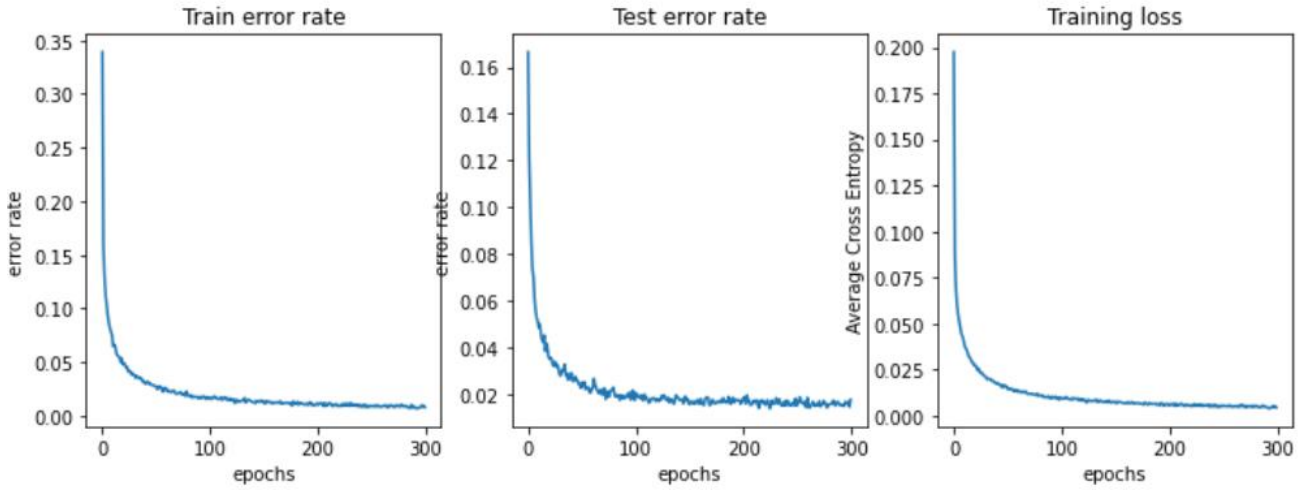
(5) 在不限定最後一層 hidden layer 的 units 為 2 的情況下，將第二層的 hidden units 也跟著調高，分別調至 60 與 120

number of hidden layers	2
number of 1st hidden units	500
number of 2nd hidden units	60, 120
learning rate	0.05
number of epochs	300
Number of mini-batch size	100

A. 第二層 hidden units 調到 60 個，error rate 可達到 1.4%



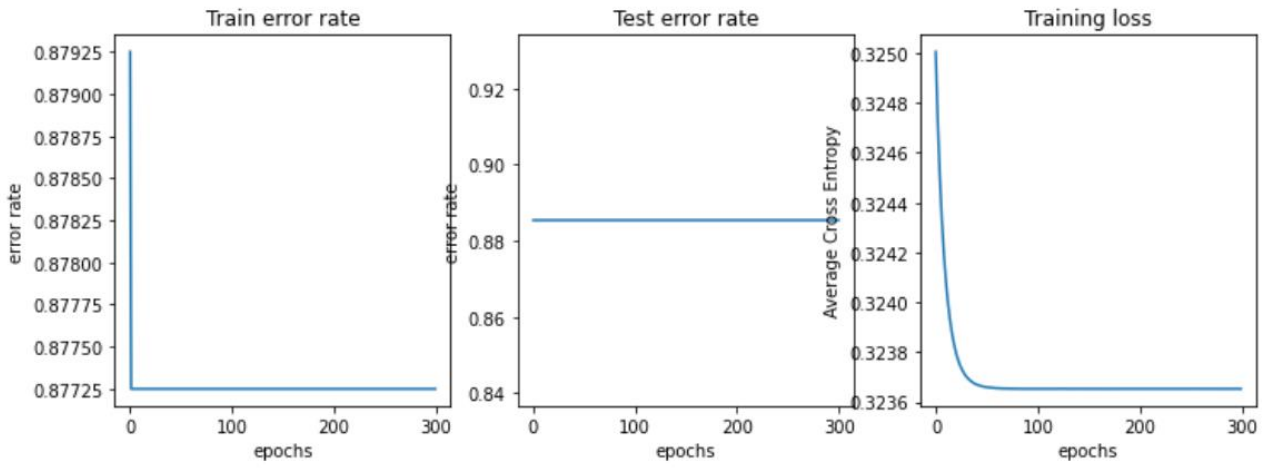
B. 當第二層 hidden units 設定成 120 個 error rate 可達到 0.74%，神經元變多時，可以讓訓練的 acuracy 提高，loss 似乎還有繼續下降的空間



II. Please perform zero and random initializations for the model weights and compare the corresponding error rates. Are there any difference between two initializations? Please discuss in the report.

zero initialization 的 error rate 與 loss 只有在最一開始時有變化而已，這是因為模型的權重是依賴最小化 cross entropy loss function 在更新的，但當初始權重設為 0 時，梯度下降的方向都會是一致的， 所有神經元會做同樣的事情，造成模型的權重無法順利更新，所以可以看到我們的 error rate 最後仍然停在 87.725%；而 random initialization 如果在一開始給定權重適當範圍的隨機值，可以讓每次梯度下降遵循不同的方向與得到不同的權重大小，才能夠順利一直不斷更新，直到 loss 無法再變的更小

以下是 zero initialization 的 error rate 與 loss



印出 softmax 前一層的權重 w3 均為 0：

```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

僅有 softmax 前一層的截距項權重 b3 不為 0：

```
array([[ 0.11775984,  0.21848714,  0.08426134,  0.02687743,  0.10568028,
        -0.22765853, -0.05916448, -0.32783879, -0.10116877,  0.16276453]])
```

以下是 random initialization softmax 前一層的權重 w3：

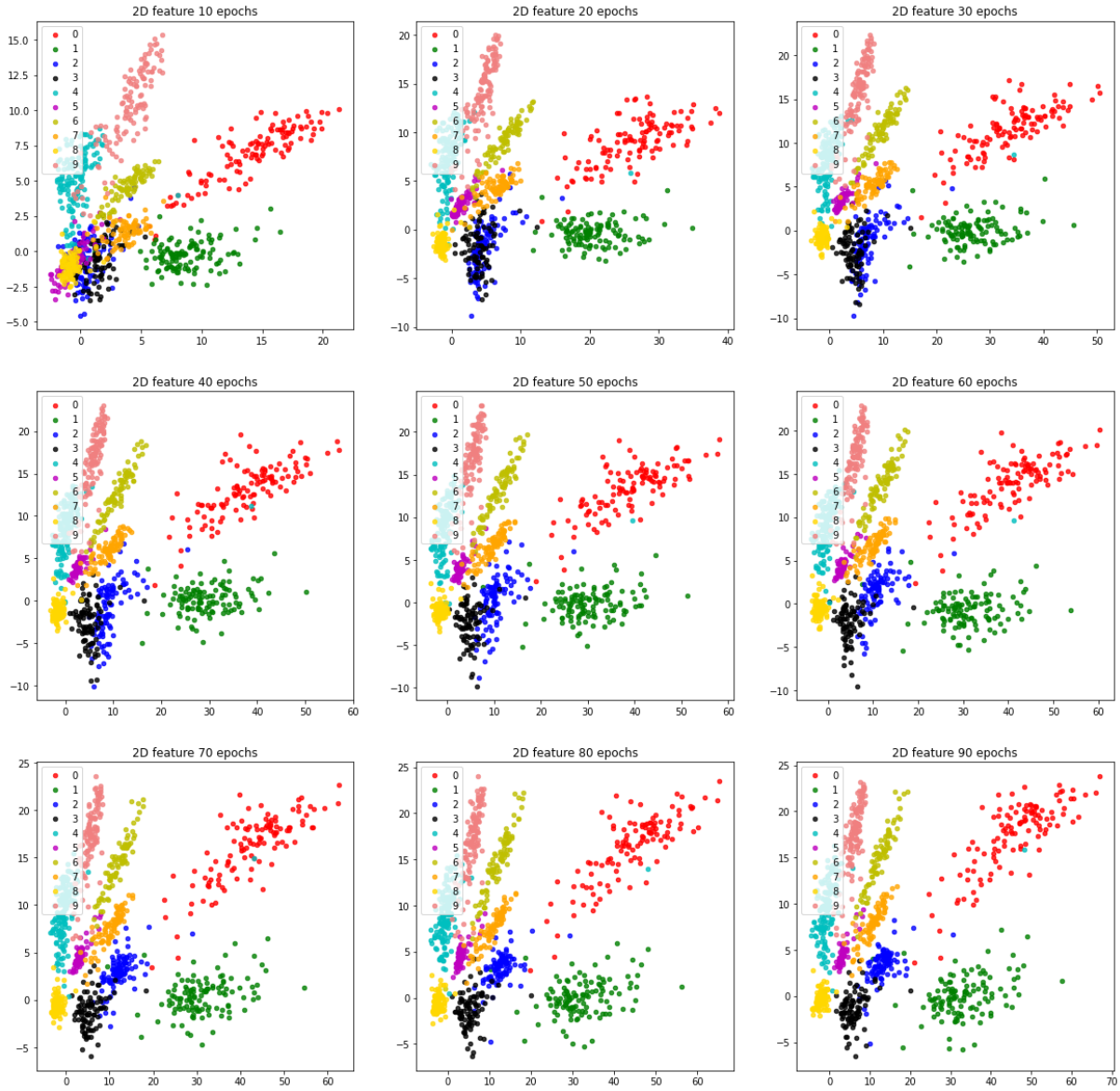
```
array([[ 2.39620397,  2.90059966,  2.37549361,  2.34835852, -
5.11805562,
        -0.83665217, -0.69461637,  1.22524766, -2.3236821 , -
2.73534937],
       [-1.19347132, -3.48053372, -1.96481438, -3.98924786,
3.29441692,
         1.41594627,  2.24175077,  0.1726708 ,  0.29600705,
3.26520974]])
```

印出 softmax 前一層的截距項權重 b3：

```
array([[-7.18627981, -4.85393238,  0.85252395,  4.91873567,
0.53949417,
         4.25338806, -4.6007553 , -0.7005027 ,  9.20641088, -
3.83435456]])
```

III. Design your network architecture with the layer of 2 nodes before the output layer.

(1) Plot the distributions of latent features at different training stages. For example, you may show the results when running at 20th and 80th learning epochs.



(2) Please discuss the evolution of latent features at different training stage.

以上我們畫出從 10, 20,…到第 90 個 epoch 的 latent feature 的分配，可以看到在第 10 個 epoch 時，兩個 latent feature 還沒有分的很好，左下角仍可看到有一大重疊的部分，隨著訓練次數變多，2 個 latent feature 的分配會逐漸分離，一直到最後第 90 個 epoch 時可以見到已經 10 個類別的分配已經幾乎分開了，少有重疊的部分。

IV. Please list your **confusion matrix** and discuss about your results.

列代表 true label，行代表 predicted label，斜對角的數字極大至少都有 300~600，代表大多數的數字都有被正確預測到，不過我們仍可發現少部分數字被錯誤分類，可能是因為兩個

數字長的比較接近，例如數字 4 vs. 9, 2 vs. 3, 5 vs. 6。

	0	1	2	3	4	5	6	7	8	9
0	647	3	0	0	1	1	0	0	12	0
1	0	649	1	0	0	0	0	0	6	5
2	0	6	543	27	1	0	0	1	0	6
3	0	0	37	541	6	0	0	14	0	2
4	0	1	0	0	614	1	8	0	6	21
5	0	0	0	0	2	394	11	0	0	0
6	0	0	0	0	2	8	488	4	0	0
7	0	0	0	11	5	1	11	421	0	0
8	1	0	0	0	16	2	3	0	550	0
9	0	2	7	0	27	0	0	2	0	640

2. Convolutional Neural Network for Image Recognition

I. Please **describe** in details how to **preprocess images** because of the different resolution images and various bounding boxes region in Medical Masks dataset and **explain** why. You have to **submit your preprocessing code**.

首先我們要針對資料做裁切，一開始檢查資料可發現當圖片下載後的名稱跟csv檔案名稱有些不同，所以要先針對csv裡面的名稱做處理，把不必要的字元刪除，才能順利讀到檔案。

```
/kaggle/input/cnn-hw1-2/B9722571616Z.1_20200213123830_000+GH5FG2SJ5.2-0.jpg
OpenCV(4.2.0) /io/opencv/modules/imgproc/src/resize.cpp:4045: error: (-215:Assertion failed) !ssize.em
pty() in function 'resize'
```

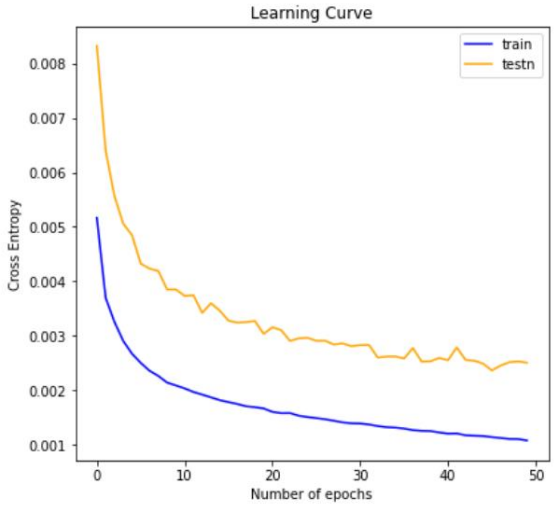
接著我們用cv2.imread()將圖片讀進來，並依照xmin, xmax, ymin, ymax進行裁切，再將裁切後的array儲存於另外的檔案以供接下來使用。接著我們在另一個.py檔將這些array讀進來後，再進行resize為3X64X64的大小，64為圖片長寬，3表示RGB三個顏色，resize成相同大小才能方便我們輸入model進行訓練時有同樣大小的輸入。

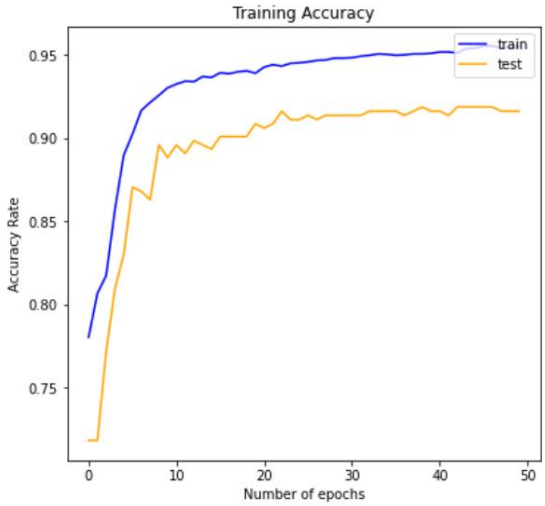
II. Please implement a CNN for image recognition by using Medical Masks dataset. You need to design the network architecture, describe your network architecture and analyze the effect of different settings including stride size and filter size. Plot the learning curve and the accuracy rate of training and test data.

我們在這個架構中先使用一層為64X64的CNN，filter matrix為3X3，其中經過ReLU function與MaxPool(2)以後，再flatten經過神經元分別為512與128個神經元，最後連接到3個輸出神

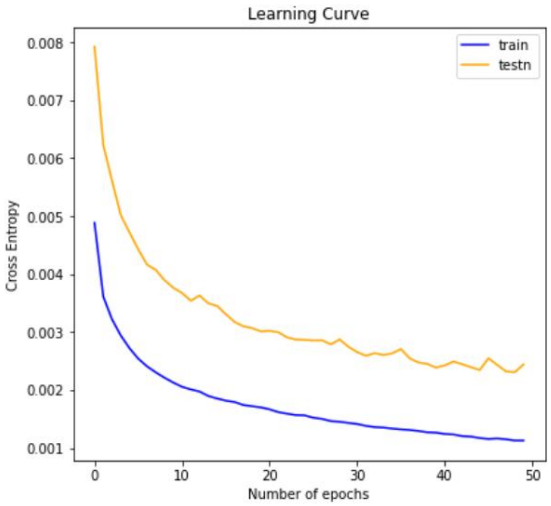
經元，learning rate為0.001。

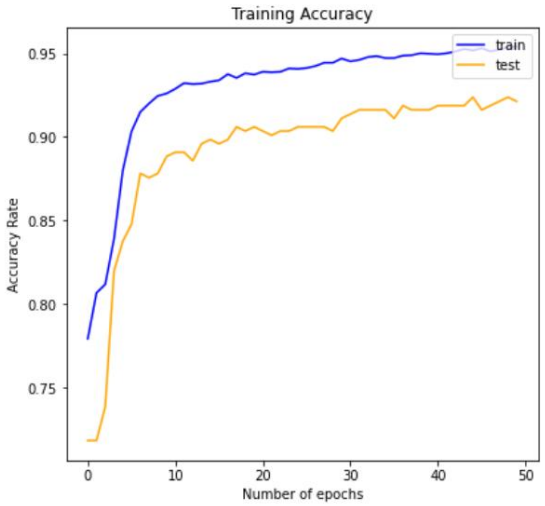
可發現僅一層CNN架構便能得到還不錯的結果，accuracy rate大約在92%左右。



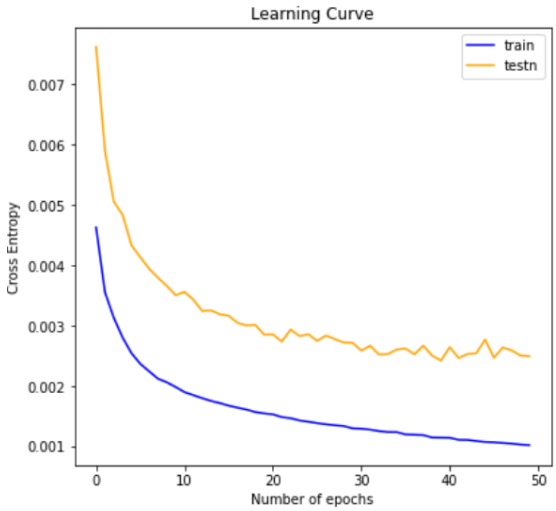


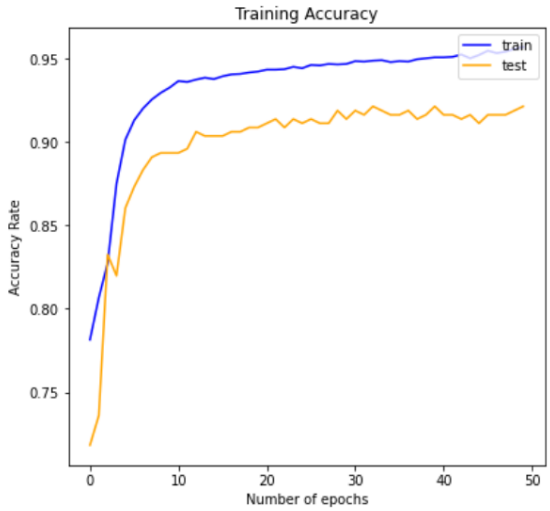
將filter matrix改為5X5，與3X3的結果差不多



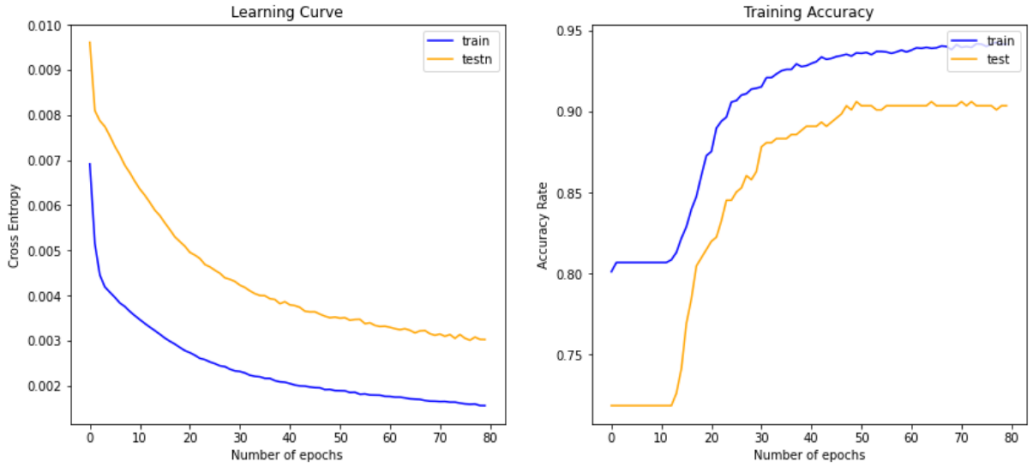


將filter matrix改為11X11，與前兩者的結果差不多，顯示filter matrix的大小在這裡並沒有太大的影響

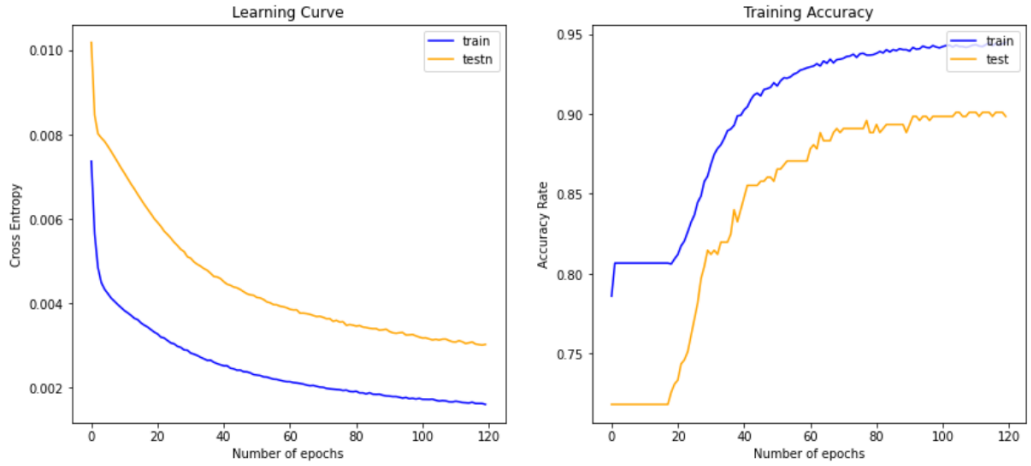




在filter matrix為3X3的情況下，stride設為3，使filter matrix一次移動3步，可發現flatten之後的神經元變少，運算時間提高，accuracy rate降低一些，只有約91%，適合用在不那麼要求正確率但想提高運算速度的時候；另外我們可以發現初期accuracy幾乎沒什麼改變，不過到後期testing set的loss變得比先前平穩很多推測是因為flatten後的那層神經元變少，比較不易overfitting



在filter matrix為3X3的情況下，stride設為6，padding設為8，單個epoch運算速度更快了，但相對來說整體的accuracy rate約接近90%左右，為了確定是否真的只到90%，我們又多跑到120個epoch，也發現結果真的沒有很好，推測是因為padding設的比較大填補了過多不必要的pixel



III. Show some examples of classification result, list your accuracy of each classes for both training and test data, and answer the following questions:

以下三張由左到右分類結果分別為good, none, bad

(1) Which class has the worst classification result and why?

以下是3個類別分別對於training set與test set的accuracy，我們可以看到good和bad兩類別的資料分類的正確性相對於none高出許多，none的分類效果最不好，推測這是因為none的資料類別只有104筆，相對於good有2846筆，bad有578筆算是少了很多，因此沒有辦法得到有效的訓練

	Train_acc	Test_acc
good	0.983148	0.981201
none	0.030000	0.016364
bad	0.855571	0.841348

(2) How to solve this problem? (explain and do some experiment to compare the result)

原始資料裁切後共有3528筆資料，我認為model看到none的比例過低，因此試想能否透過調整把資料調成比例盡量一致，故接下來只選取了500張圖片，none選取原來全部的圖片，good與none則是大概挑了約200張，一樣進行相同模型的訓練，可得到以下結果，我們可以發現none的準確性仍然很低，但是相對於good和bad類別，它的accuracy可說是上升最多的

	Train_acc	Test_acc
good	0.983914	0.983180
none	0.037500	0.024545
bad	0.857439	0.844944

(3) Do some discussion about your results.

A. 在訓練模型時有發現到神經元多一點的模型訓練出來可能正確率可以比神經元較少的情況好上1%~2%，但是相對來說花費時間也比較久，learning curve也變得比較不平穩，要注意是否有overfitting的問題

B. 模型的準確率會受到資料特別少的那個類別所影響，所以如果能夠提升該類別的正確率，能提升整個模型的正確率很多，在上面方法中我使用的是減少資料量使得三個類別的比例能夠平均一點，不過後來發現還有其他方法，像是增加cross entropy中資料少的那個類別的權重，這樣做的好處是不會減少資料量；或是在data augmentation做旋轉翻轉等，增加資料多樣性