Deep Learning Homework 1

0510894 電機 4D 翁紹恩

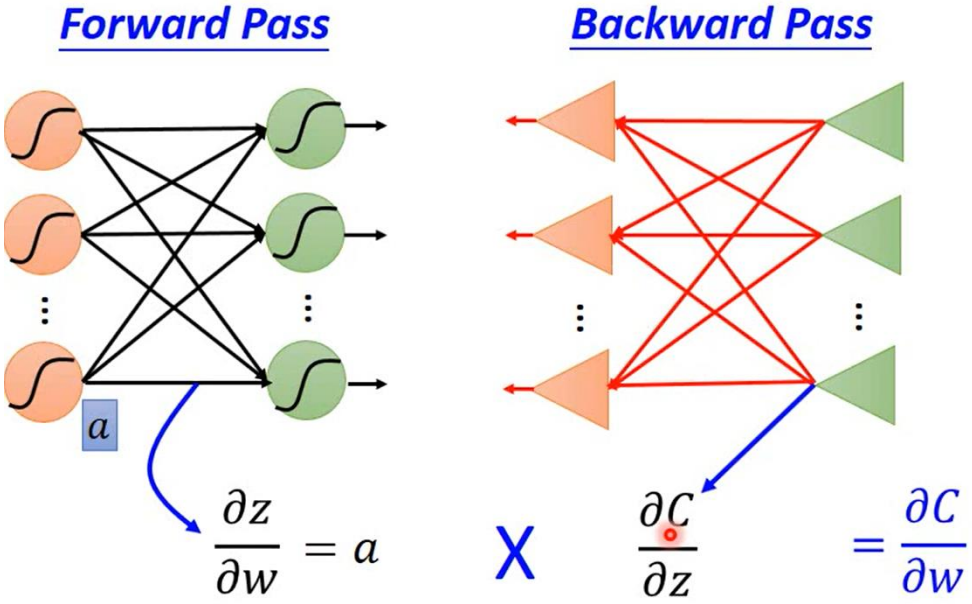
一 Deep Neural Network for Classification

1 系統設計架構：

這次架構總共使用三層，input layer、output layer、hidden layer，中間的 hidden layer 總共有 1024 個 nodes，所有 layer 使用 fully connected layer，在進入 hidden layer 時使用 sigmoid function，因為層數沒有使用非常多層，因此不需要太擔心 sigmoid 之後梯度消失造成資訊丟失的問題，最後輸出會再經過 softmax 以分類，最後在使用分類的 cross entropy loss；input layer 大小為 mini batch size * 784，將輸入圖片從二維 14 * 14 轉成一維 784，output layer 大小為 mini batch * 10，因為最後需要分十類。在 input layer 和 hidden layer 乘上 weight 傳到下一層之前，都會多增加一個數值為 1 的 bias，因此 weight 大小會多增加一個 bias 項，在 backward propagation 時會將計算 bias 的 weight 項省略傳回，以符合 feature map 大小；一個 mini batch 中執行一次 forward propagation 和一次 backward propagation，此次 mini batch 設為 200，epoch 次數為 500，learning rate = 0.001。

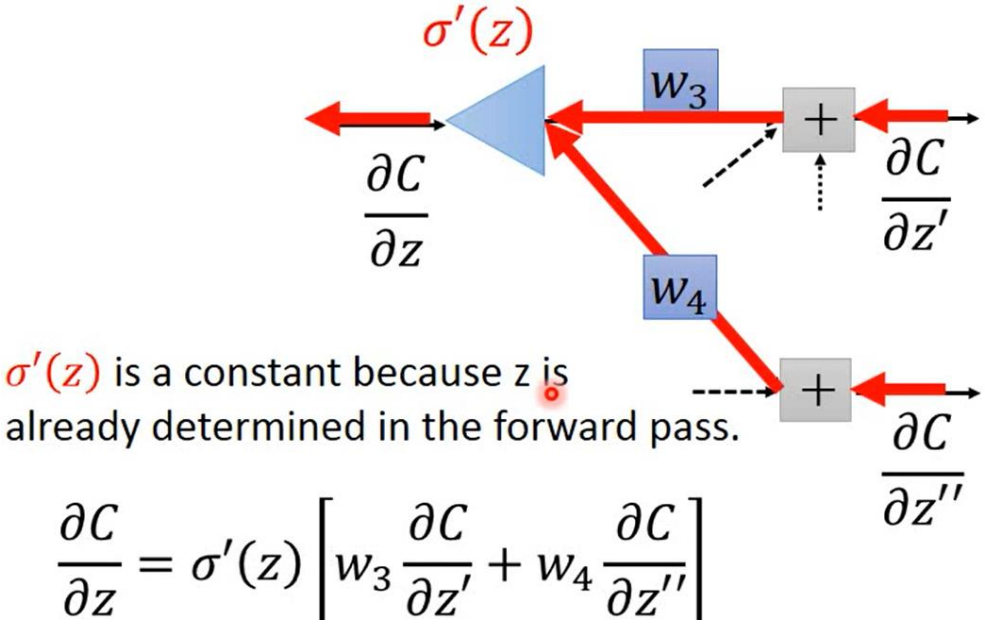
2 系統設計公式：

整段過程都是利用矩陣乘法，較為困難的是需要推斷出 loss 對各層 weight 微分結果，以使用 Stochastic Gradient Descent，這裡使用台大李宏毅教授的上課資料做解釋輔助：



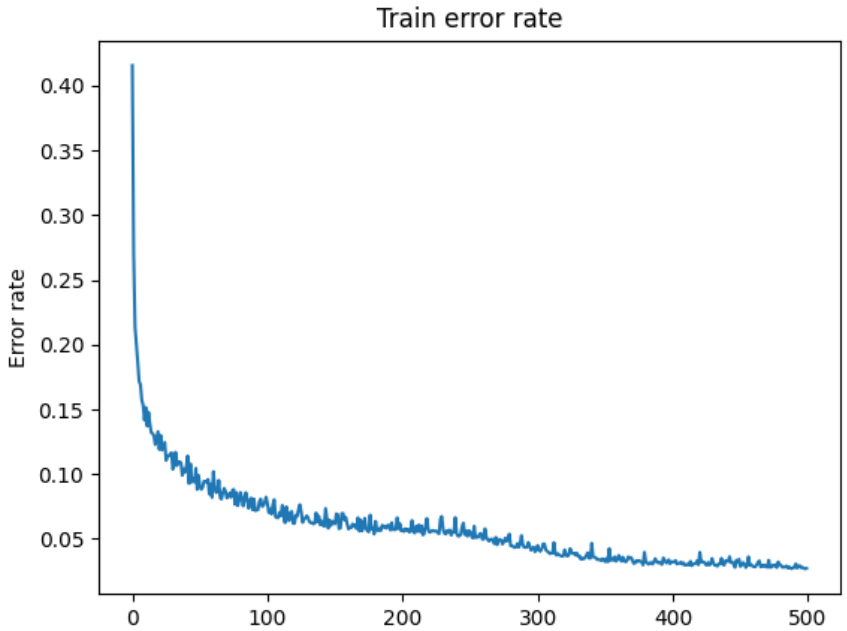
我們只需要知道各層的結果，和額外推導出 loss 對 activation 之前的結果微分，即可計算出各層 loss 對 weight 的微分，而各層 loss 對 activation 前的結果可以由後推回，此處也就是 backward propagation

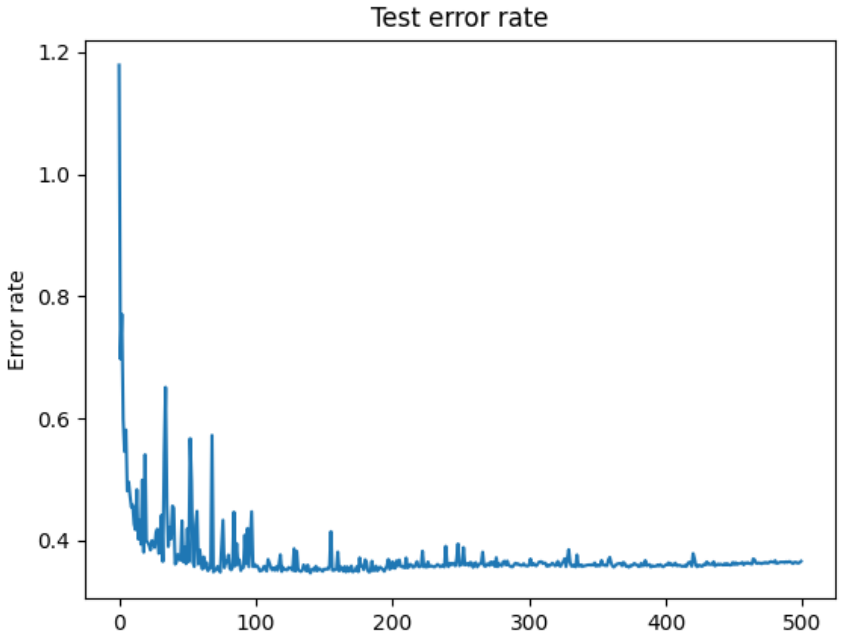
的精神，由於最後一層為 softmax，最先推出結果為 y – t，再乘上倒數第二層 layer 結果可得最後一層 weight 微分，而緊接著往前算回對 sigmoid layer 的微分，同樣此處使用台大李宏毅教授的上課資料做示意：

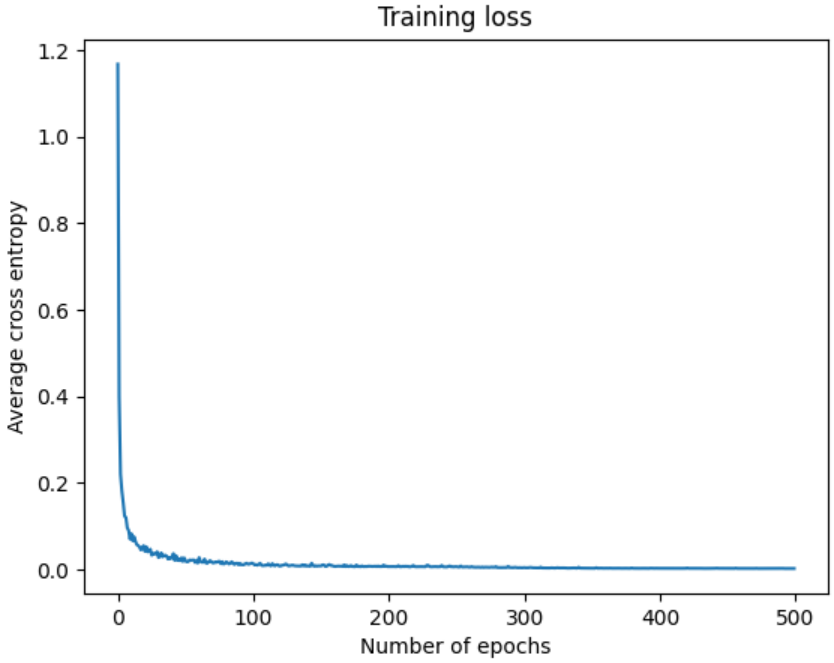


$$\frac{\partial C}{\partial z} = \sigma'(z)\left[w_3\frac{\partial C}{\partial z'} + w_4\frac{\partial C}{\partial z''}\right]$$

然而在此次作業中，由於乘上對 sigmoid function 的微分後結果皆不佳，accuracy 大約到 0.12 之後就無法提升，又針對上述對 sigmoid 的微分為常數項，因此最後直接將該項省略並更改為$\frac{\partial L}{\partial Z_n} = w\frac{\partial L}{\partial Z_{n+1}}$，實驗結果顯示大約在 10 個 epochs 左右，accuracy 就能提升至 0.7、0.8 以上，實驗數據如下：



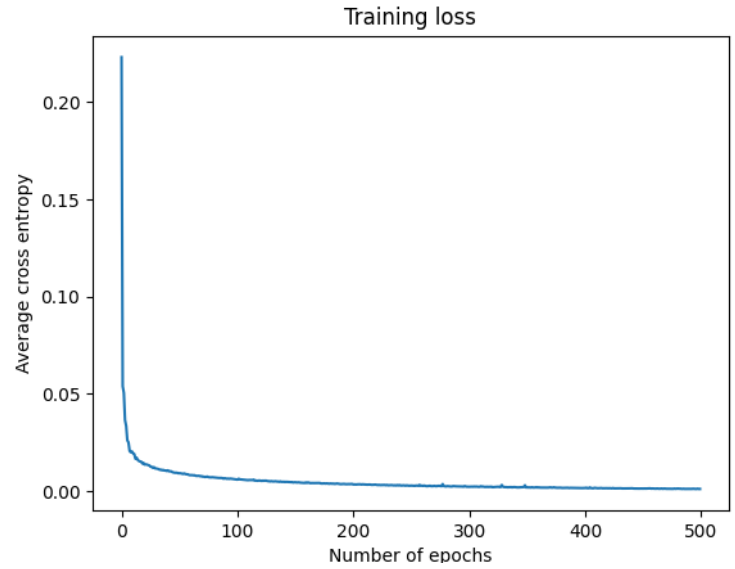


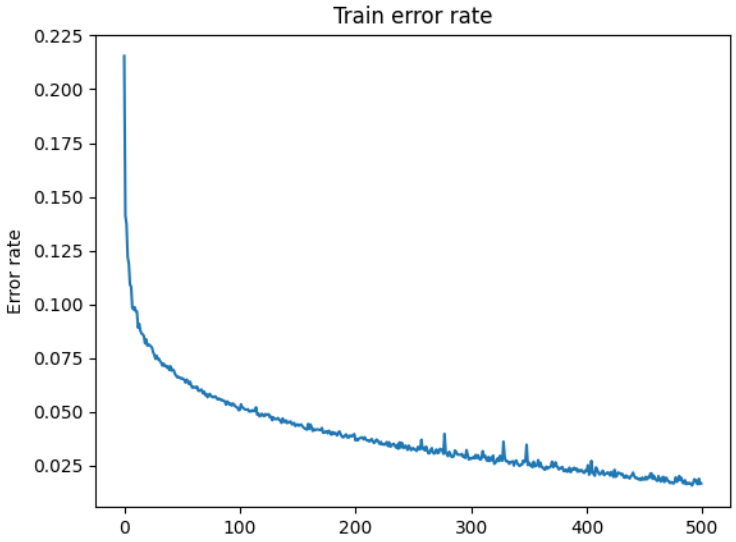


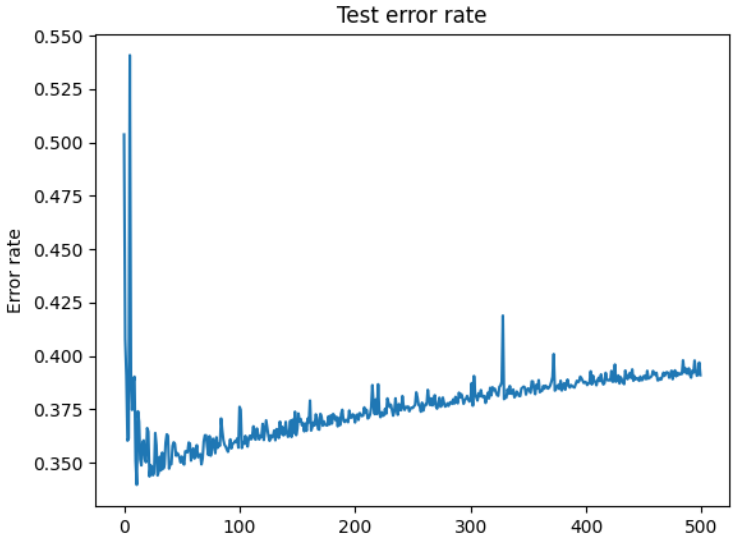

```
$ python3 hw1_q1.py
Epoch 0     :Accuracy 0.122    Loss 1.325    Error_rate  0.416
Epoch 1     :Accuracy 0.601    Loss 0.473    Error_rate  0.270
Epoch 2     :Accuracy 0.737    Loss 0.278    Error_rate  0.218
Epoch 3     :Accuracy 0.791    Loss 0.211    Error_rate  0.194
Epoch 4     :Accuracy 0.815    Loss 0.178    Error_rate  0.182
Epoch 5     :Accuracy 0.829    Loss 0.153    Error_rate  0.173
Epoch 6     :Accuracy 0.840    Loss 0.141    Error_rate  0.167
Epoch 7     :Accuracy 0.856    Loss 0.123    Error_rate  0.159
Epoch 8     :Accuracy 0.865    Loss 0.109    Error_rate  0.153
Epoch 9     :Accuracy 0.867    Loss 0.104    Error_rate  0.151
Epoch 10    :Accuracy 0.870    Loss 0.105    Error_rate  0.150
```

3 Please perform zero and random initializations for the model weights and compare the corresponding error rates.

a Zero initializations for the model weights (此處的 zero initializations 只有針對第一層的 weight，因若兩層 weight 皆 zero initializations，accuracy 極低，像用猜的，因此就不列入討論)

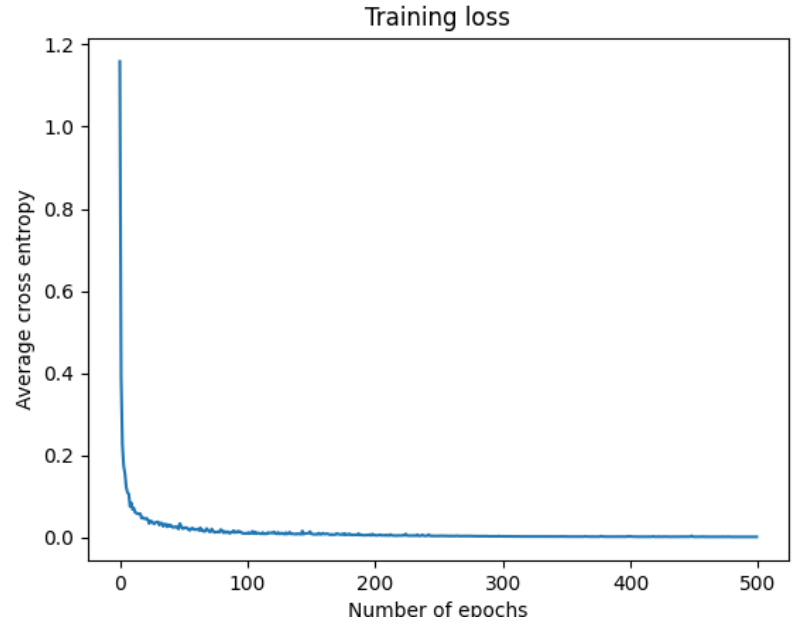


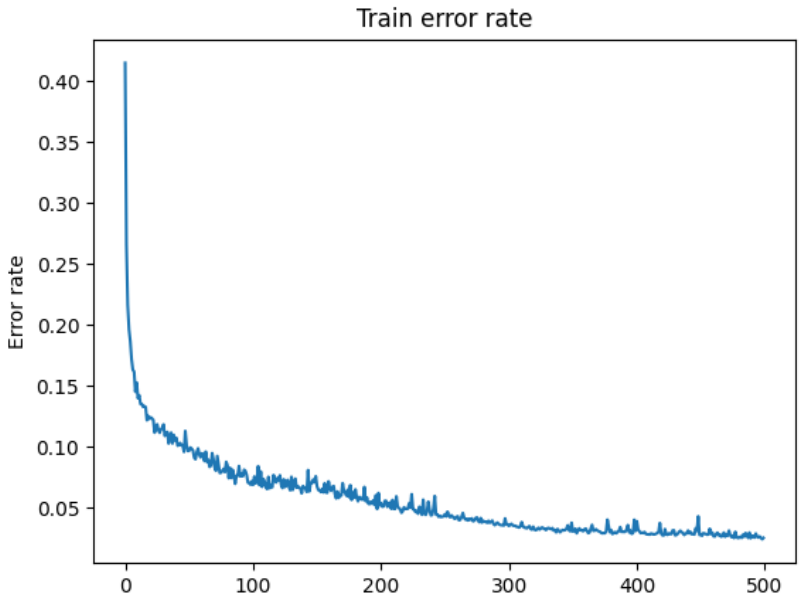


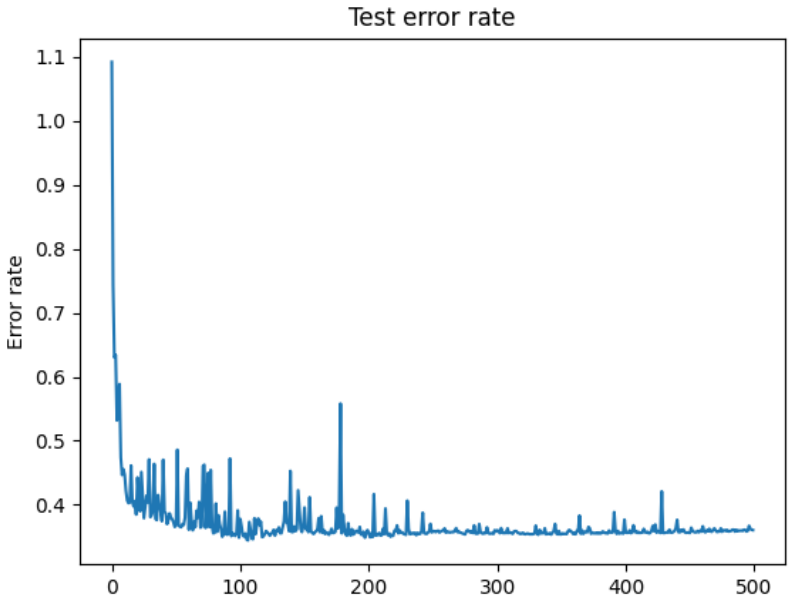


```
Epoch 480   :Accuracy 0.998    Loss 0.001    Error_rate  0.020
Epoch 481   :Accuracy 0.999    Loss 0.001    Error_rate  0.018
Epoch 482   :Accuracy 0.999    Loss 0.001    Error_rate  0.019
Epoch 483   :Accuracy 0.999    Loss 0.001    Error_rate  0.017
Epoch 484   :Accuracy 0.999    Loss 0.001    Error_rate  0.017
Epoch 485   :Accuracy 0.999    Loss 0.001    Error_rate  0.018
Epoch 486   :Accuracy 1.000    Loss 0.001    Error_rate  0.016
Epoch 487   :Accuracy 0.999    Loss 0.001    Error_rate  0.017
Epoch 488   :Accuracy 0.999    Loss 0.001    Error_rate  0.017
Epoch 489   :Accuracy 0.999    Loss 0.001    Error_rate  0.017
Epoch 490   :Accuracy 0.999    Loss 0.001    Error_rate  0.017
Epoch 491   :Accuracy 1.000    Loss 0.001    Error_rate  0.016
Epoch 492   :Accuracy 0.999    Loss 0.001    Error_rate  0.017
Epoch 493   :Accuracy 0.999    Loss 0.001    Error_rate  0.019
Epoch 494   :Accuracy 0.999    Loss 0.001    Error_rate  0.018
Epoch 495   :Accuracy 0.999    Loss 0.001    Error_rate  0.017
Epoch 496   :Accuracy 0.999    Loss 0.001    Error_rate  0.016
Epoch 497   :Accuracy 0.999    Loss 0.001    Error_rate  0.019
Epoch 498   :Accuracy 0.999    Loss 0.001    Error_rate  0.016
Epoch 499   :Accuracy 0.999    Loss 0.001    Error_rate  0.017
```

b Random initializations for the model weights



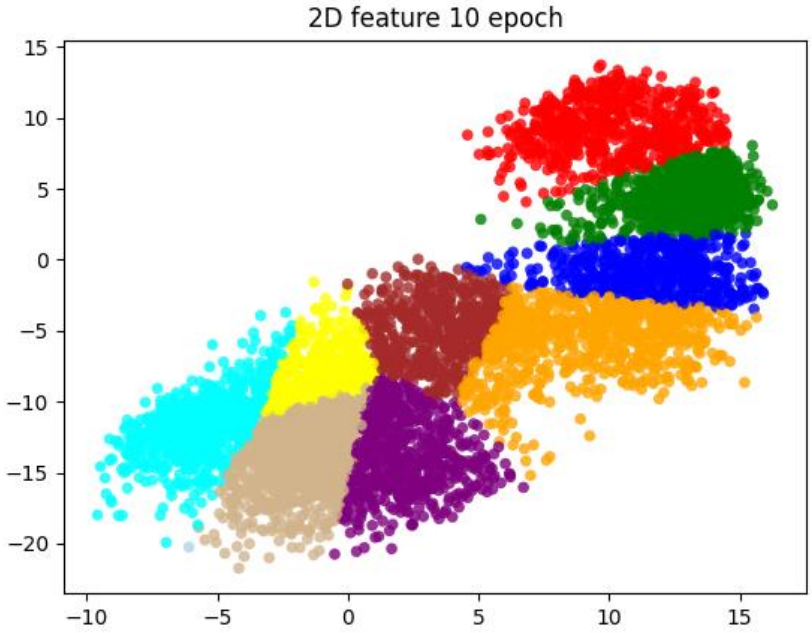


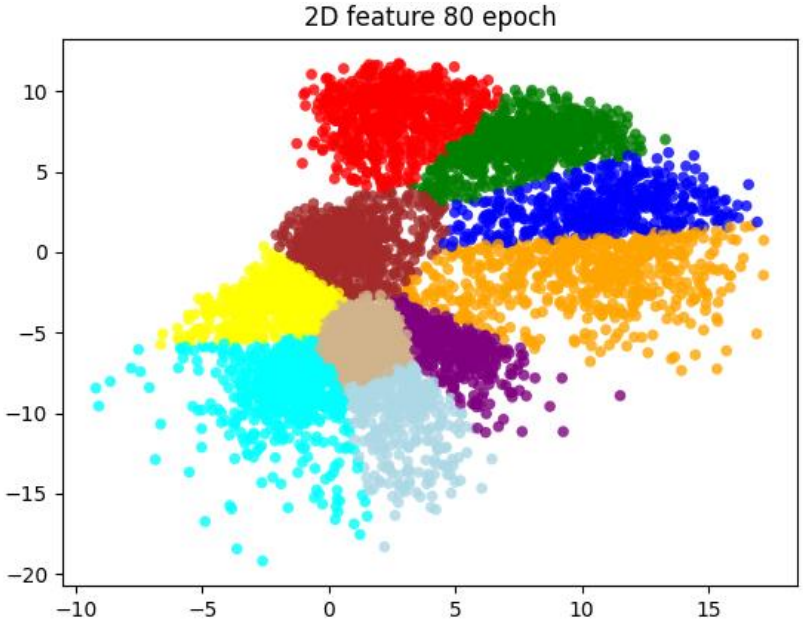



```
Epoch 478   :Accuracy 0.998    Loss 0.002    Error_rate  0.026
Epoch 479   :Accuracy 0.998    Loss 0.002    Error_rate  0.025
Epoch 480   :Accuracy 0.998    Loss 0.002    Error_rate  0.026
Epoch 481   :Accuracy 0.998    Loss 0.002    Error_rate  0.027
Epoch 482   :Accuracy 0.998    Loss 0.002    Error_rate  0.026
Epoch 483   :Accuracy 0.997    Loss 0.002    Error_rate  0.028
Epoch 484   :Accuracy 0.997    Loss 0.002    Error_rate  0.027
Epoch 485   :Accuracy 0.996    Loss 0.002    Error_rate  0.030
Epoch 486   :Accuracy 0.997    Loss 0.002    Error_rate  0.028
Epoch 487   :Accuracy 0.998    Loss 0.002    Error_rate  0.026
Epoch 488   :Accuracy 0.996    Loss 0.002    Error_rate  0.030
Epoch 489   :Accuracy 0.998    Loss 0.002    Error_rate  0.025
Epoch 490   :Accuracy 0.997    Loss 0.002    Error_rate  0.027
Epoch 491   :Accuracy 0.997    Loss 0.002    Error_rate  0.027
Epoch 492   :Accuracy 0.998    Loss 0.002    Error_rate  0.026
Epoch 493   :Accuracy 0.996    Loss 0.002    Error_rate  0.029
Epoch 494   :Accuracy 0.997    Loss 0.002    Error_rate  0.027
Epoch 495   :Accuracy 0.997    Loss 0.002    Error_rate  0.026
Epoch 496   :Accuracy 0.997    Loss 0.002    Error_rate  0.027
Epoch 497   :Accuracy 0.998    Loss 0.002    Error_rate  0.026
Epoch 498   :Accuracy 0.999    Loss 0.002    Error_rate  0.025
Epoch 499   :Accuracy 0.998    Loss 0.002    Error_rate  0.026
```

c 根據以上結果可以發現，zero initialization 的 loss 和 error rate 曲線都較為平緩，相較之下，random initialization 極為震盪，但是 zero initialization 在 test 時很快就 overfitting 了，而在 random initialization 時則是趨於穩定，由於一開始所有的 weight 都使用 zero initialization 時反而有 train 不起來的趨勢，因此在後續 model 訓練上都會傾向使用 random initialization。

4 為了畫出分布圖因此在原架構輸出層前多加上一層 2 nodes layer，此處就不經過 activation function，由於不經過 activation layer 類似 autoencoder 的結果，weight 就相當於可以讓結果正交的 component，因此只做線性相乘，結果如下：





由上圖可以觀察出，兩者分布區域不同外，且只訓練 20 epochs 時，有一類完全不見，相較 20 epochs，80 epochs 的分布也較為集中，也可以推斷此為出分類越來越精準。

5 Confusion matrix

結果 / 實際	0	1	2	3	4	5	6	7	8	9
0	648	7	5	0	1	1	1	2	0	1
1	3	638	9	0	1	0	1	0	0	5
2	8	10	513	64	1	0	2	3	0	9

3	2	3	54	515	6	6	4	2	7	3
4	0	0	0	2	573	0	3	8	12	28
5	0	0	0	0	2	387	10	13	2	2
6	2	0	1	2	15	2	463	23	1	2
7	1	0	0	5	2	8	15	389	5	6
8	0	0	0	9	16	0	2	2	545	2
9	0	3	2	3	34	3	1	7	0	620

二 Convolutional Neural Network for Image Recognition

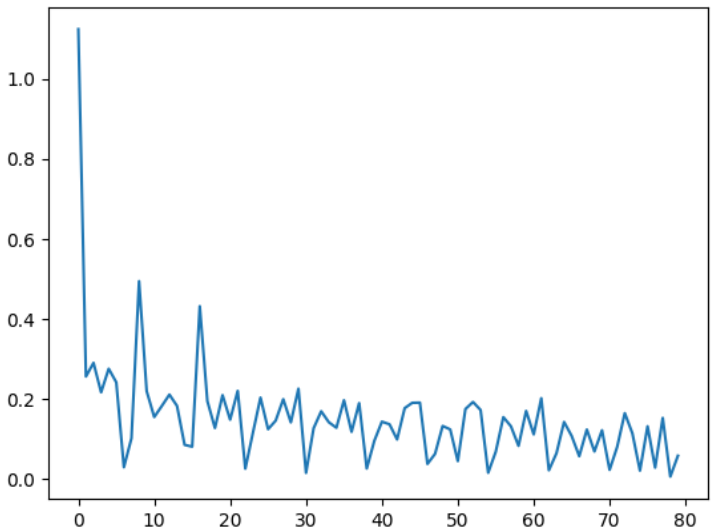
1 Preprocess images

由於對於 pytorch 的應用尚未非常熟悉，所以方便起見，最後將圖片都先依照.csv 檔中的位置裁剪後，存成新的資料庫，但由於各張圖片大小皆不同，輸入 tensor 中卻要符合相同維度，因此在資料前處理的後半使用 PIL 中的 Image 做圖片讀取和縮放，由此才可以真正儲存縮放成統一指定大小，此處統一將資料縮放為 128 * 128。另外再寫一個 dataloader，需要將圖片和類別讀出並寫入__getitem__及需要定義一個__len__。

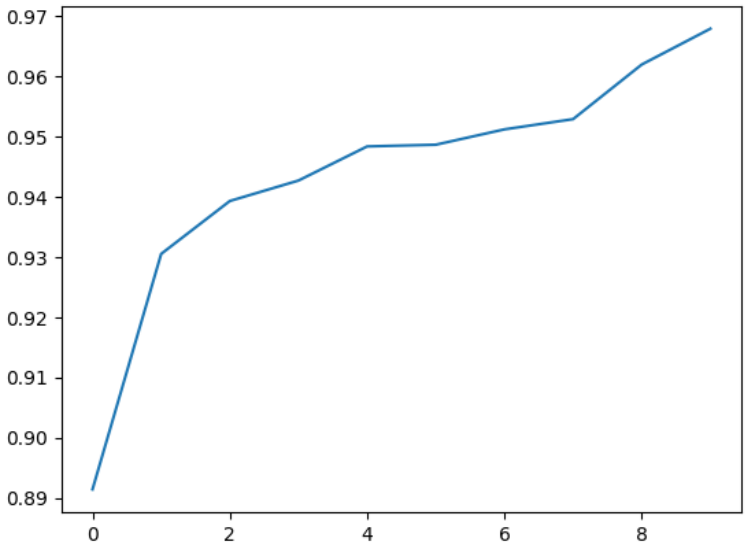
2 Implement a CNN for image recognition by using Medical Masks dataset.

使用 torch.nn 建立 CNN 架構，在這邊使用了五層大架構，前三層中每一層分別為使用 convolution kernel、再經過 ReLU、最後做 2 * 2 的 MaxPooling，convolution 的 output channel 分別為 32、64、128，再經過兩個 fully connected layer 做降維，最後輸出經過 softmax，得出分類結果。

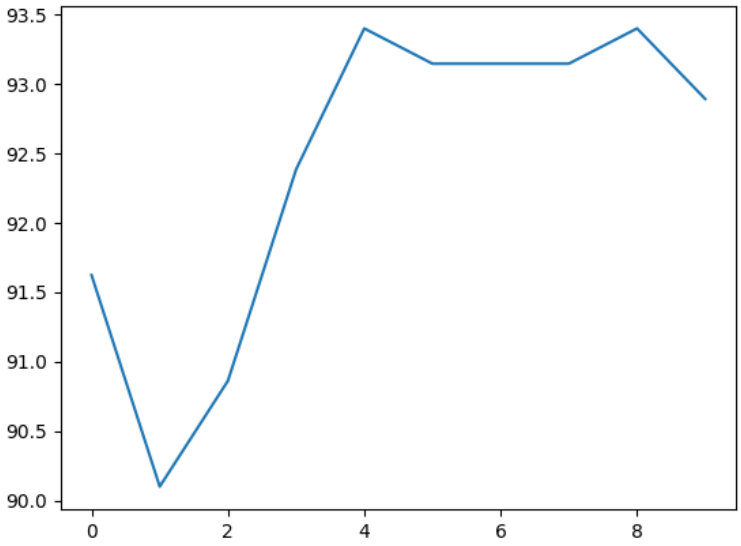
a Learning Curve(kernel size = 5, stride = 1)



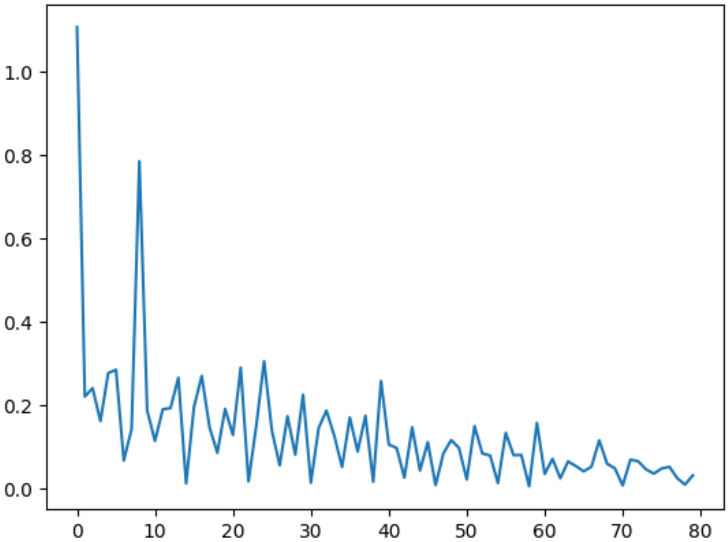
b Training Accuracy(kernel size = 5, stride = 1)



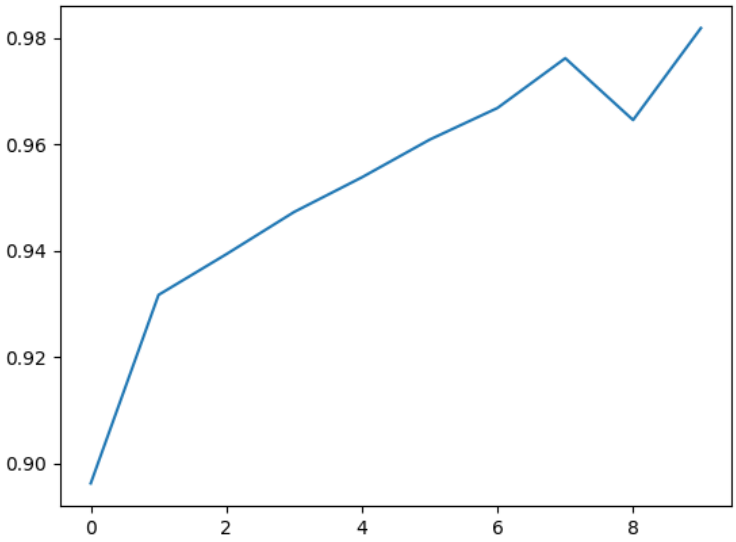
c Testing Accuracy(kernel size = 5, stride = 1)



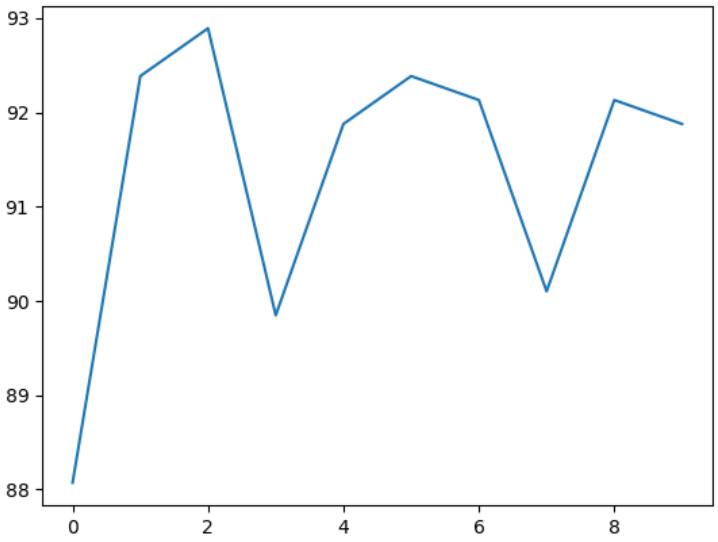
d Learning Curve(kernel size = 3, stride = 1)



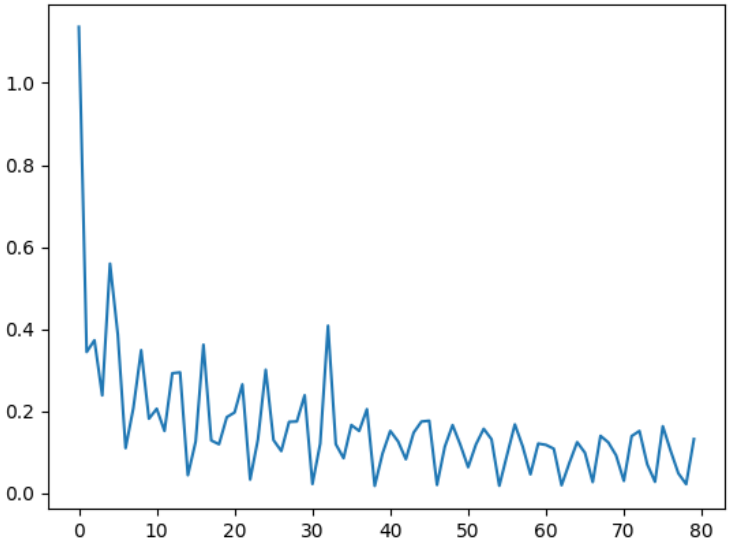
e Training Accuracy (kernel size = 3, stride = 1)



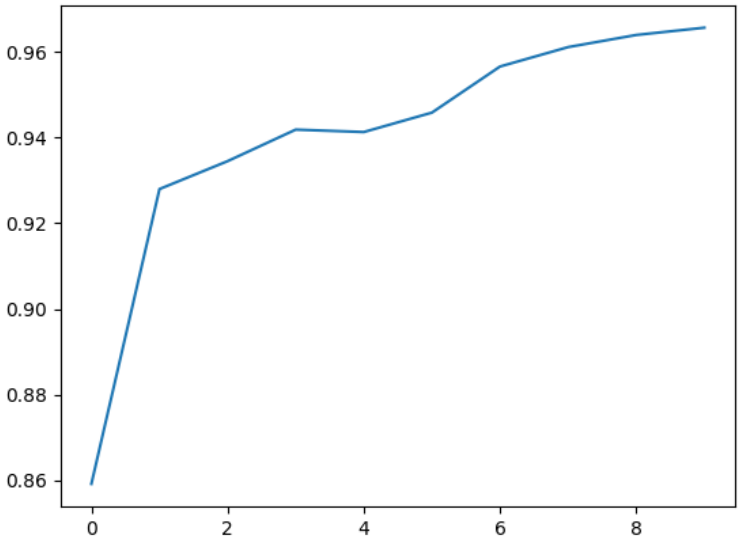
f Testing Accuracy(kernel size = 3, stride = 1)



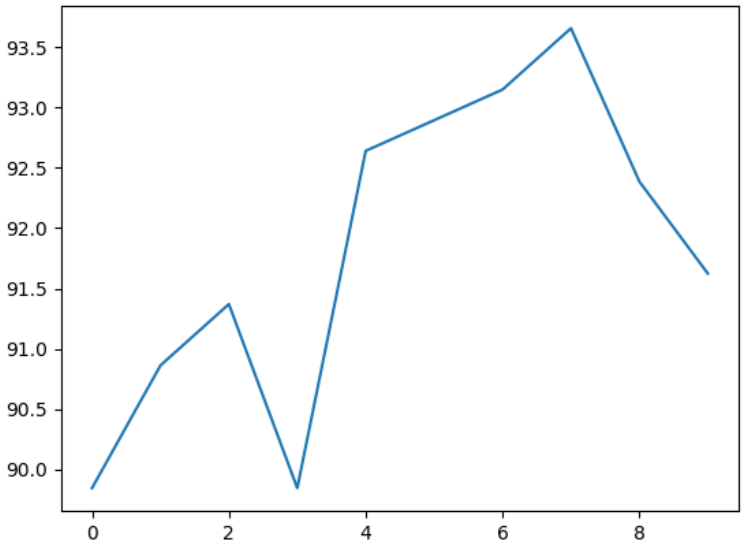
g Learning Curve(kernel size = 5, stride = 2)



h Training Accuracy (kernel size = 5, stride = 2)



i Testing Accuracy(kernel size = 5, stride = 2)



j Disscuss

由以上結果可以發現 stride 數大準確率較低，但是訓練速度較快，因為參數量相對少很多，而 kernel size 5*5 略為優於 3*3，我認為這和資料有關，因為我的前處理訓練資料為一整張人臉戴口罩的照片，因此，若是 kernel 越大可以看到的範圍越完整，結果也就當然會越精準，但若是資料的前處理是原始照片只縮放大小，那麼需要偵測的細節就較小，此時應用較小的 kernel size 不過當我將 kernel size 改為 9 時，準確度卻又下降，因此這或許不太一定。如下表為 kernel size 為 9 的結果，和最後一個表格為 kernel size 5 比較結果較差。

Class	Train Acc	Test Acc
good	98.56%	97.17%

none	50.00%	27.27%
bad	95.12%	89.89%

3 Classification result

a Which class has the worst classification result and why?

在這次實驗中，none 得到了最差的結果，但這是很合理且可預期的，因為 none 類別的張數最少且會被預測為 none 的都是較為不清楚的，否則應該就只有有無戴口罩兩類。

b How to solve this problem?

根據理論，如果有這種 class imbalance 的 long tail 現象，可能需要額外訓練一個超參數做 imbalnace 的校正，或是擴增較少類別的資料，由於我最後只有做 kernel 和 stride 的調整根據實驗，用 5*5 的 kernel、stride = 1 的效果是最好的。

c Do some discussion about your results.

針對於 none 類別特別少的現象，雖然為預測的一項缺陷，但其實針對有沒有戴口罩本就應該只有有無兩類，因此如果這兩類的準確率是十分不錯的，且 accuracy 高，對實際應用應不會有太大影響。而由於在少數 epoch 後就快速達到收斂且一個 epoch 的時間滿久的，因此下表為 10 epochs 的訓練資料，10 epochs 是在 train 幾次之後發現太多 epoch 會有 overtraining 現象且準確度無改善，因此取 10。

Class	Train Acc	Test Acc
good	98.66%	95.76%
none	53.85%	40.90%
bad	95.33%	96.63%