

分工表 (0.5%)

組員	分工
B04901040 電機三 林哲賢	HW1-2 Observe gradient norm during training, What happens when gradient is almost zero HW1-3 Can the network fit random variables,
B04901117 電機三 毛弘仁	整理報告和 Github repo、協調分工 HW1-1 Train on Actual Tasks HW1-2 Visualize error surface HW1-3 Flatness v.s. Generalization
B04901118 電機三 王克安	HW1-1 Simulate a Function HW1-2 Visualize the optimization process HW1-3 Number of parameters v.s. Generalization

HW1-1: Deep v.s. Shallow

Simulate a Function (王)

Describe the models you use, including the number of parameters (at least two models) and the function you use. (0.5%)

這題我們使用兩個函數來做實驗：

函數一：

$$\frac{\sin \pi x}{\pi x} \times \tanh x$$

函數二：

$$\frac{\sin \pi x}{\pi x} \times |x| \times (11 \cos x - 6 \cos\left(\frac{11}{6}\right) x)$$

使用的模型有以下四種fully connected的架構：

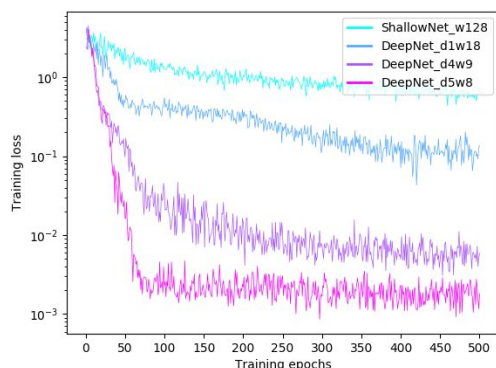
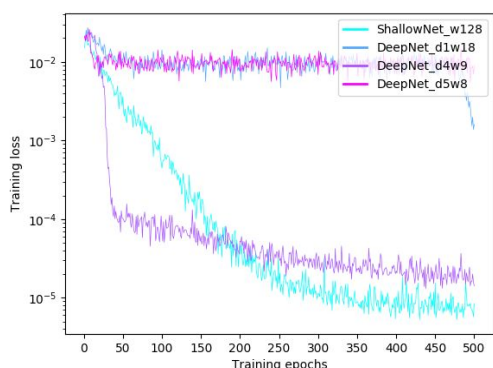
```
<bound method Module.parameters of ShallowNet(
  (input): Linear(in_features=1, out_features=128)
  (output): Linear(in_features=128, out_features=1)
)>
Trainable parameters: 385
```

```
(input): Linear(in_features=1, out_features=18)
(linear): ModuleList(
  (0): Linear(in_features=18, out_features=18)
)
(output): Linear(in_features=18, out_features=1)
)>
Trainable parameters: 397
```

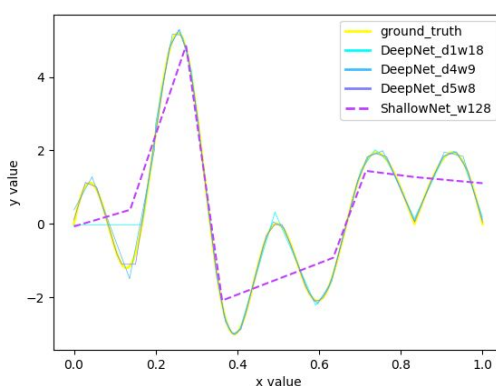
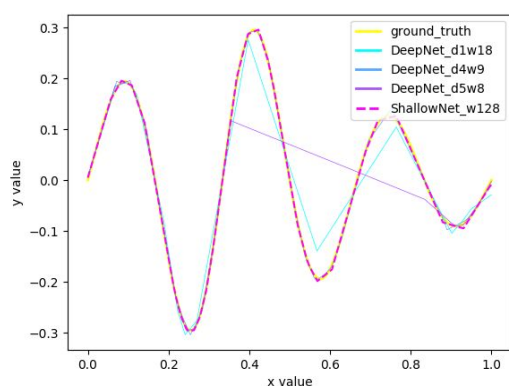
```
(input): Linear(in_features=1, out_features=9)
(linear): ModuleList(
  (0): Linear(in_features=9, out_features=9)
  (1): Linear(in_features=9, out_features=9)
  (2): Linear(in_features=9, out_features=9)
  (3): Linear(in_features=9, out_features=9)
)
(output): Linear(in_features=9, out_features=1)
)>
Trainable parameters: 388
```

```
(input): Linear(in_features=1, out_features=8)
(linear): ModuleList(
  (0): Linear(in_features=8, out_features=8)
  (1): Linear(in_features=8, out_features=8)
  (2): Linear(in_features=8, out_features=8)
  (3): Linear(in_features=8, out_features=8)
  (4): Linear(in_features=8, out_features=8)
)
(output): Linear(in_features=8, out_features=1)
)>
Trainable parameters: 385
```

In one chart, plot the training loss of all models. (0.5%)



In one graph, plot the predicted function curve of all models and the ground-truth function curve. (0.5%)



Comment on your results. (1%)

在第一個函數中shallownet竟然是準確率最高的一個model，而其他model的準確率似乎不受到depth的影響，我們推測是因為這個函數太過簡單，如同老師上課所提到的，函數要超過一定的複雜度才看得出來deep跟shallow的差別。在第一個函數這個case當中可能是因為函數不夠複雜，所以model的準確度受到training epochs跟random initialize的影響較大，而跟本身架構比較沒有關係。於是我們嘗試使用第二個比較複雜的函數。

從第二個函數的實驗結果可以明顯觀察到deepnet比起shallownet不論是在loss或是reconstruction上都有明顯的優勢，而且隨著model越深其loss跟inference的結果都是隨之變好的。shallownet在這個函數上完全沒辦法預測原本的函數，而deepnet則是隨著深度增加有著更好的表現，驗證了deepnet在一定複雜度以上的函數是會有比較好的結果。

Use more than two models in all previous questions. (bonus 0.25%)

以上兩個function都有用4個model跑過了。

Use more than one function. (bonus 0.25%)

有使用兩個function做驗證。

Train on Actual Tasks(毛)

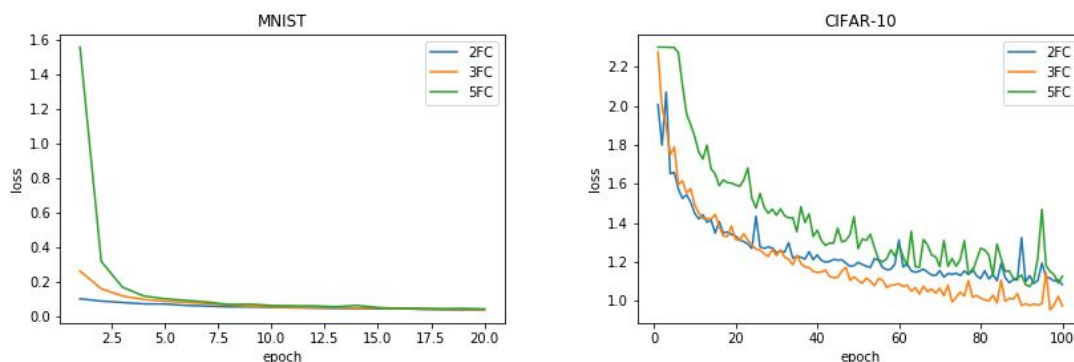
Describe the models you use and the task you chose (0.5%).

這次我們跑了兩個 task，每個 task 各有三個 model：

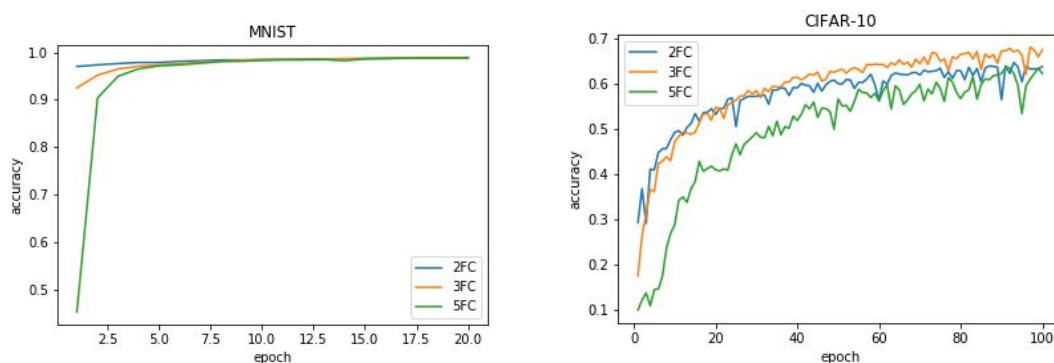
1. 用 CNN 跑 MNIST，三個 model 皆有大約 21,840 個 parameters
2. 用 CNN 跑 CIFAR-10，三個 model 皆有大約 31,340 個 parameters

我們兩個 task 都使用右圖的 model architecture，三個 model 則分別選擇 2, 3, 5 層的 fully-connected layer。

In one chart, plot the training loss of all models. (0.5%)



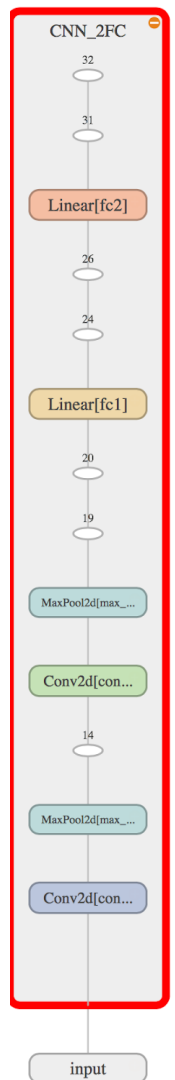
In one chart, plot the training accuracy. (0.5%)



Comment on your results. (1%)

雖然題目只說要看 training loss 和 accuracy，但我們在 train 的時候其實也有看 test loss 和 accuracy，並且沒有觀察到 overfit（training loss 下降但 test loss 增加）的情形發生。

由圖表發現，越深的神經網路，converge 所需要花的 epoch 數比較多。這個現象又在資料較複雜的 dataset（相同 model architecture 之下，CIFAR-10 的 loss 一直比 MNIST 高）更加明顯。



我們也探討**相同 parameter 數量下，更深的 feedforward network 表現是否比較好**。在 MNIST 上，deep 和 shallow 表現差不多，不過這有可能是因為 MNIST 本身比較簡單，shallow network 不需要 train 出太複雜的 function 就可以表現不錯，deep feedforward network 沒辦法「拉開差距」。在 CIFAR-10 上，3 層 fully-connected layer 比 2 層表現好，但是 5 層又比 2 層差，因此我們**不確定這個 model 的 feedforward network 是不是越 deep 越好**。老師上課曾講過，越前面的 layer 遇到 noise，越容易使 performance 降低，或許越深的網路 **vanishing gradient 問題越嚴重，使前面 layer 無法被順利 train 好**。探討各 layer 的 gradient 大小，以及嘗試使用 layer pre-training，是未來可以繼續研究的方向。另外，探討 convolution layer 的深度是否影響 performance，也是可以做的實驗。

其實越 deep 就越好，應該不是一個鐵則，不然大家只要瘋狂把 model 做深就行了。2014 年的 **NIPS**，就有一篇 **paper** 在 CIFAR-10 上面，用 shallow model 去學 deep model 的 output，達到差不多的 performance。期待這學期的 MLDS，可以繼續探索 neural network 的各種神秘面目！

Use more than two models in all previous questions. (bonus 0.25%)

以上兩個function都有用 3 個 model 跑過了。

Train on more than one task. (bonus 0.25%)

使用了 MNIST 及 CIFAR-10 這兩個 task。

HW1-2: Optimization

Visualize the optimization process(王)

Describe your experiment settings. (The cycle you record the model parameters, optimizer, dimension reduction method, etc) (1%)

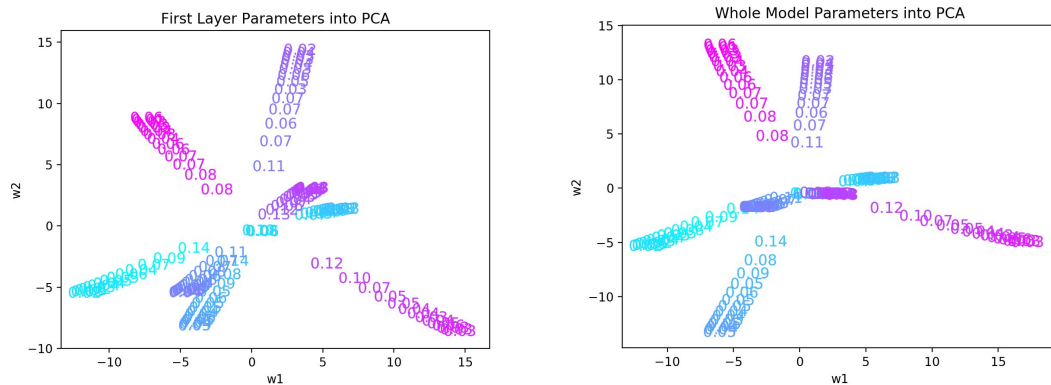
在這題中我們使用的dataset是MNIST，model架構是DNN，詳細資訊如圖所示。optimizer是adam，lr=0.0003，使用的降維方法是 PCA。

Training過程中我們總共train了8次，每次train 42個epoch，其中每三次記錄一次 model weight跟training loss。

```
(input): Linear(in_features=784, out_features=64)
(layers): ModuleList(
  )
(hidden): Linear(in_features=64, out_features=16)
(output): Linear(in_features=16, out_features=10)
```

<-- model architecture

Train the model for 8 times, selecting the parameters of any one layer and whole model and plot them on the figures separately. (1%)



Comment on your result. (1%)

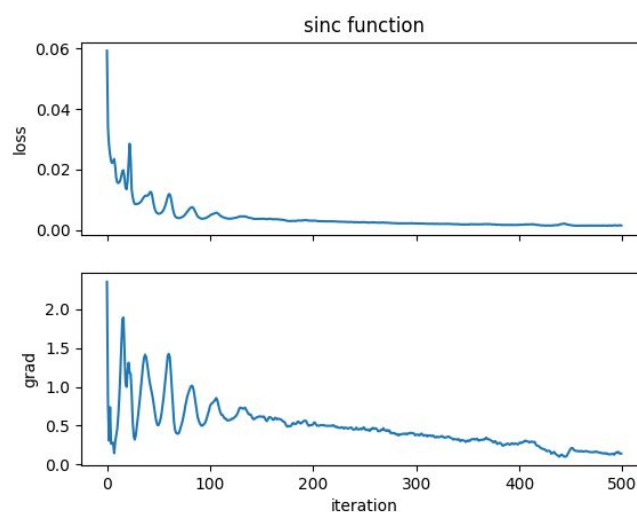
上圖中每一個顏色代表一次 training event，數字代表該 epoch 的 loss。不同的 training event 會 random initialize 在不一樣的初始點，經過 gradient descent 之後會往不同的方向走，所以每個 training event 的參數都不一樣，導致經過 PCA 降維之後在二維平面上的分佈會不一樣。

從圖中可以觀察到在 loss 比較大（前幾個 epoch）的時候數字跟數字之間的距離相對比較遠。這代表在 training 初期的 gradient 是比較大的，導致 model update parameter 的速度比較快，而到了後期 training loss 開始下降以後 model 更新參數的速度也開始下降，代表 gradient 開始變小，可能是 loss 開始走到 saddle point 或是 minima。

另外我們可以觀察到兩張圖的參數分佈十分相近，這意味著整個 model 中重要的參數都集中在第一層當中，這跟老師上課提到的現象吻合。

Observe gradient norm during training. (林)

Plot one figure which contain gradient norm to iterations and the loss to iterations. (1%)



Comment on your result. (1%)

我使用了3層各128個neuron來simulate sinc function($\sin x/x$)，可以看出來 **gradient** 在前期的震盪非常大，很有可能就是老師上課所說的當gradient下降時到了一個**saddle point**，使得gradient norm到了一個低點，而當逃出了這個**saddle point**後gradient norm會迅速上升，所以會造成gradient norm一上一下的震盪，而在逐漸收斂的過程中線也是一直在做微小的震盪而非像loss一樣平緩的下降。

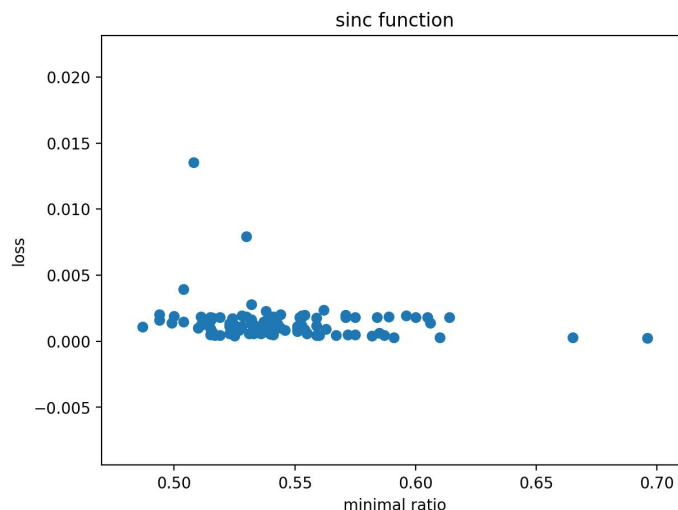
What happens when gradient is almost zero? (林)

State how you get the weight which has zero gradient norm, and how you define the minimal ratio. (2%)

1. 和1-2-2相同，我用了三層MLP來simulate sinc functon，而我使用的方法是先用MSE來算loss，然後利用這個loss來算出gradient，最後把算出來的gradient norm取代原本的loss再做backpropagation，這樣的做法讓我從minimize MSE變成minimize gradient norm，可以迅速的走到gradient norm趨近於0的點並得到model 的weight，這裡有一個小技巧就是optimizer要選擇Adadelta會更快的收斂到gradient norm趨近0的點。

2. 因為pytorch計算hessian matrix會有很多的困難，因此我是在終點random sample 1000個點，sample的方式是把每一個weight加一個-0.0001~0.0001的值，最後我把minimal ratio定成 sample出來的network去算出MSE比原本的大的比率。

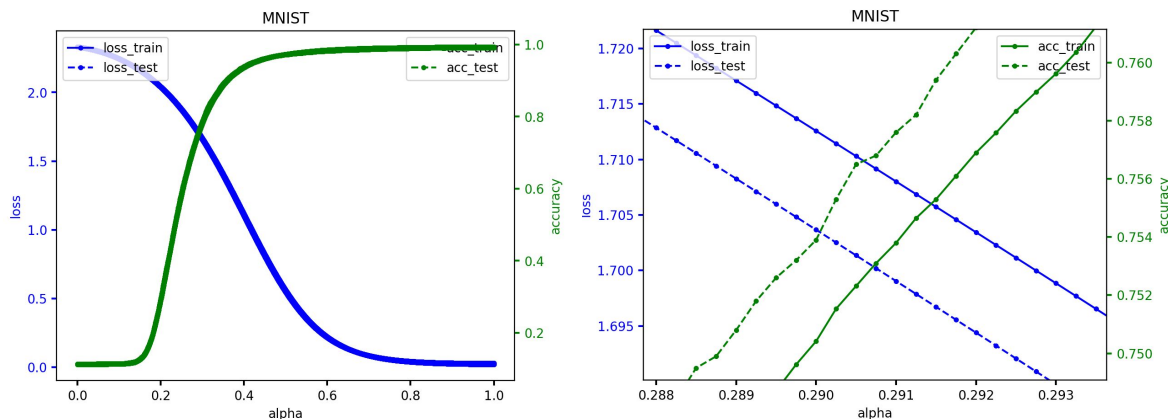
Train the model for 100 times. Plot the figure of minimal ratio to the loss. (2%)



Comment your result. (1%)

我去做了一下迴歸分析，發現迴歸出來是一條斜率小於0的線，雖然這是用sample而非去分析hessian matrix的eigenvalue，但還是可以大概印證老師上課說的minimum ratio越高會越像是一個local minimum，此時 loss 也有可能會比較低。

Bonus (1%) Use any method to visualize the error surface. Concretely describe your method and comment your result. (毛)



我們使用 $m1 * (1-\alpha) + m2 * \alpha$ 的 parameter 內插方式來看 error surface, 其中 $m1$ 是剛 initialize 的 model (random parameters)、 $m2$ 是 train 好的 model。雖然助教的圖只有畫 loss, 但我們決定也來看看 accuracy。

由圖可看出, MNIST 大概在 $\alpha = 0.4$ 的時候就有超過 0.9 的 accuracy, 可見它真的是很好 train, 只要參數不要爛到太誇張, 就有不錯的 performance。另外, 在 $\alpha = 0.2 \sim 0.4$ 的區間中, accuracy 上升的速度很快, 但在 $\alpha = 0.4 \sim 0.5$ 的區間, 雖然 loss 的下降幅度還是差不多, 但 accuracy 上升速度減緩許多, 可見 $0.2 \sim 0.4$ 的區間, model 可能「學會」大部分的「需知」。

我們 α 取了 4,000 個點 (花五個小時才跑完), 雖然有一些不平滑的線段, 但得出來的圖還是相當平滑, 放大後 (如右圖) 還是看不到高低起伏, 或許代表 MNIST 不太容易卡在 local minima。

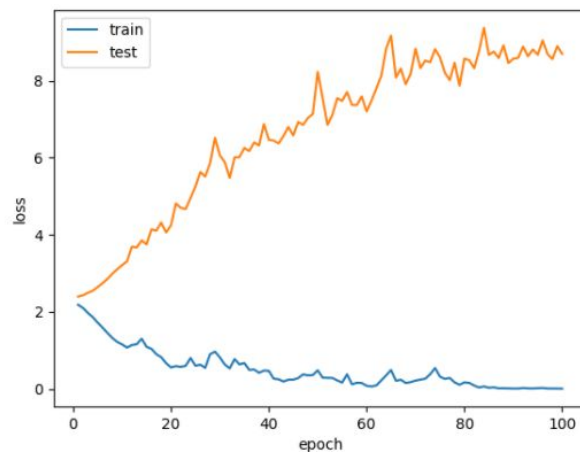
HW1-3: Generalization

Can the network fit random variables? (林)

Describe your settings of the experiments. (e.g. which task, learning rate, optimizer) (1%)

我們使用 MNIST, 並把 training label 設成 random 去 train (test data 的 label 沒有動), 架構是三層的 MLP, 每層各 256 個 neuron, loss function 使用的是 cross entropy, optimizer 是 adam, learning rate 為 0.001。

Plot the figure of the relationship between training and testing, loss and epochs (1%)

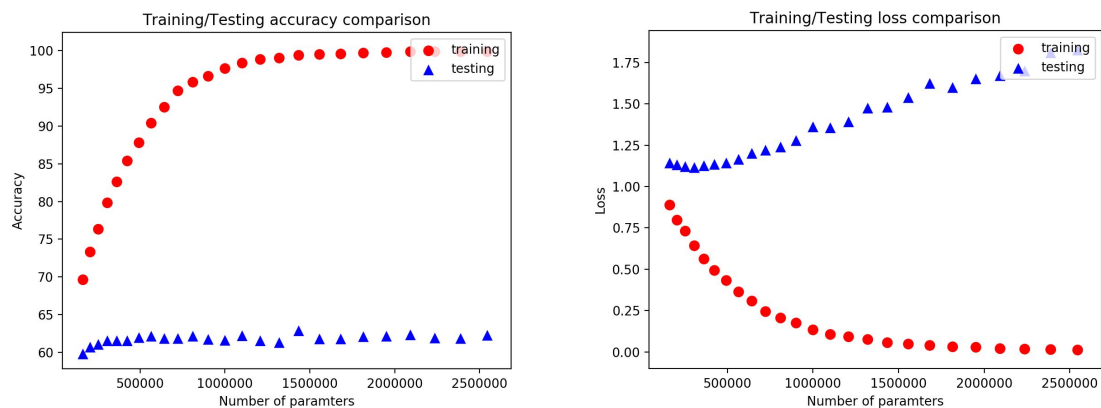


Number of parameters v.s. Generalization (王)

Describe your settings of the experiments. (e.g. which task, the 10 or more structures you choose) (1%)

這題我們使用cifar10來做實驗，使用的model如圖所示。model的變化是藉由調整CNN的output channel來改變參數量。我們的output channel從128增加到256個，每次增加八個。

Plot the figures of both training and testing, loss and accuracy to the number of parameters. (1%)



Comment on your result. (1%)

從實驗結果可以看到training accuracy隨著參數量增加而上升，而testing accuracy則是停在約60左右的地方。這邊可以觀察到的是**testing accuracy並不會因為參數量的增加而變低**，顯示出model並沒有想像中的容易overfit，不會因為參數量的增加而讓performance變更差。

在loss graph中有一樣的觀察結果。**training loss**會因為參數量的增加而下降，而**testing loss**則是大約在1500000個參數的時候開始不再有大變化的變化。

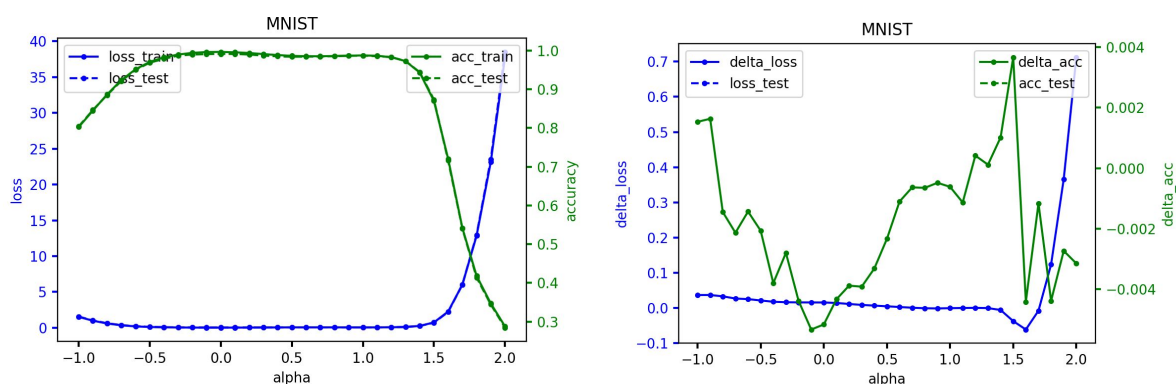
Flatness v.s. Generalization (毛)

Part 1

Describe the settings of the experiments (e.g. which task, what training approaches) (0.5%)

我們選擇的 task 是 MNIST，使用的 model 是第一頁畫出來的 CNN_2FC，m1 使用 batch size=64，m2 使用 batch size=1024，以這兩個 model 的 parameters 去做 interpolation，alpha 的範圍是 -1 到 2，每隔 0.1 取一點量測 loss、accuracy。兩個 model 都 train 了 200 個 epoch，以 MNIST 來講應該綽綽有餘，我們也看到 loss 和 accuracy 都 converge 了。

Plot the figures of both training and testing. loss and accuracy to the number of interpolation ratio. (1%)



左圖的 y 軸是 loss 和 accuracy，右圖的 y 軸是 (train_loss - test_loss) 和 (train_accuracy - test_accuracy)。

Comment on your result. (1%)

一開始看到左圖，還以為是不是 code 寫錯了，因此跑了右圖，發現只是因為左圖比例尺比較大，看不出 train 和 test 的差異（其實把左圖放大，有些地方還是看得出來有差）。

由圖表可看出：

- 只要是在兩個 model 之間做內插 ($0 < \alpha < 1$)，loss 仍低、accuracy 仍高，可能象徵 MNIST 是很容易 train 起來的 task，對於 hyperparameter tuning 沒有很苛刻
- 如果變成外插，就會得到明顯比較爛的 performance
- Train-test mismatch 的情況，無論是用內插還是外插，似乎都不會有大的變化，除了 $\alpha \geq 1.8$ 的地方看到 delta_loss 變很大之外

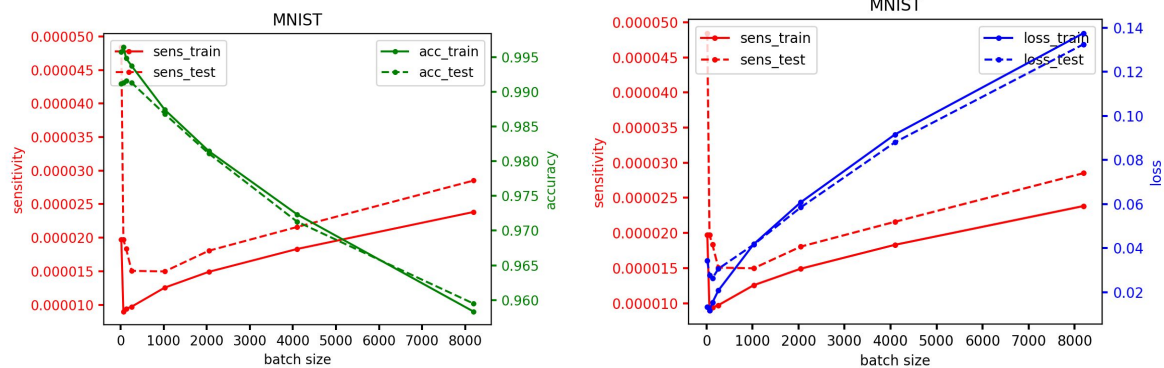
Part 2

Describe the settings of the experiments (e.g. which task, what training approaches) (0.5%)

我們選擇的 task 是 MNIST，使用的 model 是第一頁畫出來的 CNN_2FC，改變的 hyperparameter 是 batch size，分別使用了 16, 64, 128, 256, 1024, 2048, 4096, 8192 這八個

數字去 train。所有 model 都 train 了 200 個 epoch，確保 convergence。

Plot the figures of both training and testing, loss and accuracy, sensitivity to your chosen variable. (1%)



Comment your result. (1%)

由實驗結果可發現：

- Test set 的 sensitivity 比 train set 高
- Sensitivity 下降的時候，通常會 accuracy 上升、loss 下降，若 sensitivity 上升，通常結果相反。但是，最高的 sensitivity（發生在 batch size = 16）並沒有對照到最低的 accuracy、最高的 loss，我們不能說 sensitivity 下降很多，accuracy 一定上升很多（但通常至少會上升一點）。

Bonus : Use other metrics or methods to evaluate a model's ability to generalize and concretely describe it and comment your results.