HW 1-1 Report

1. **Simulate a Function:**
   a. Describe the models you use, including the number of parameters (at least two models) and the function you use

Ans : The function I used f(x) = sin(5πx)/ 5πx on x=(0.01,1)

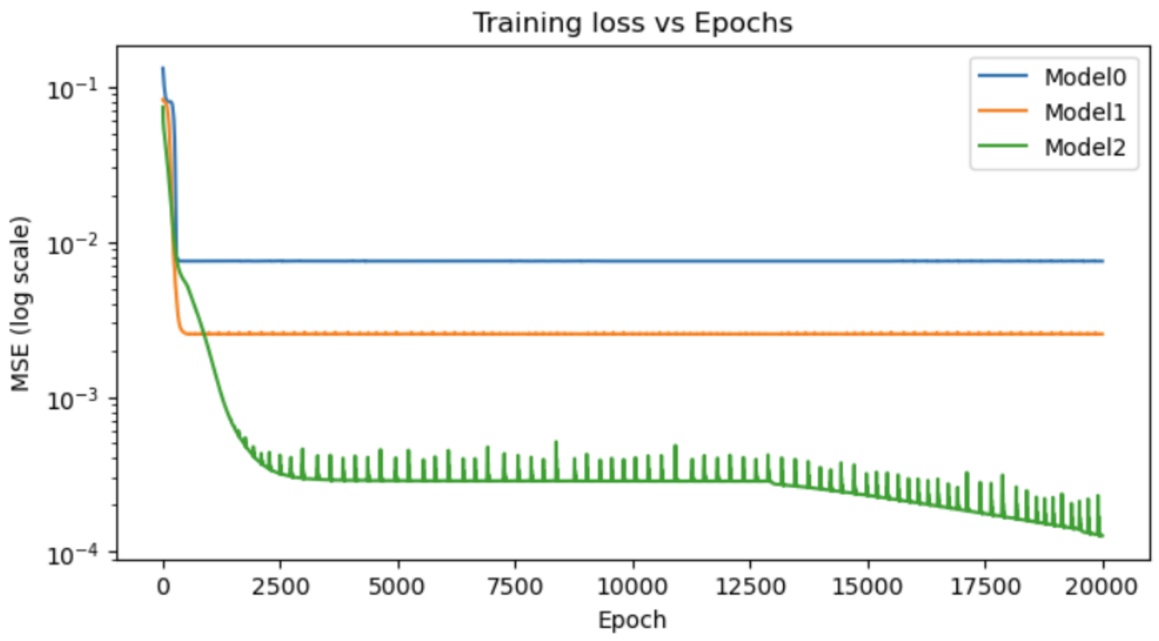
I used 3 models model0, model1, model2

Model0 = 7 small layers(1 → 5 → 10 → 10 → 10 → 10 → 5 → 1) ~ 571 parameters

Model1 = 6 layers (1 → 10 → 18 → 15 → 10 → 4 → 1) 712 parameters

Model2 = 2 layers(1 → 190 → 1) ~ 571 parameters

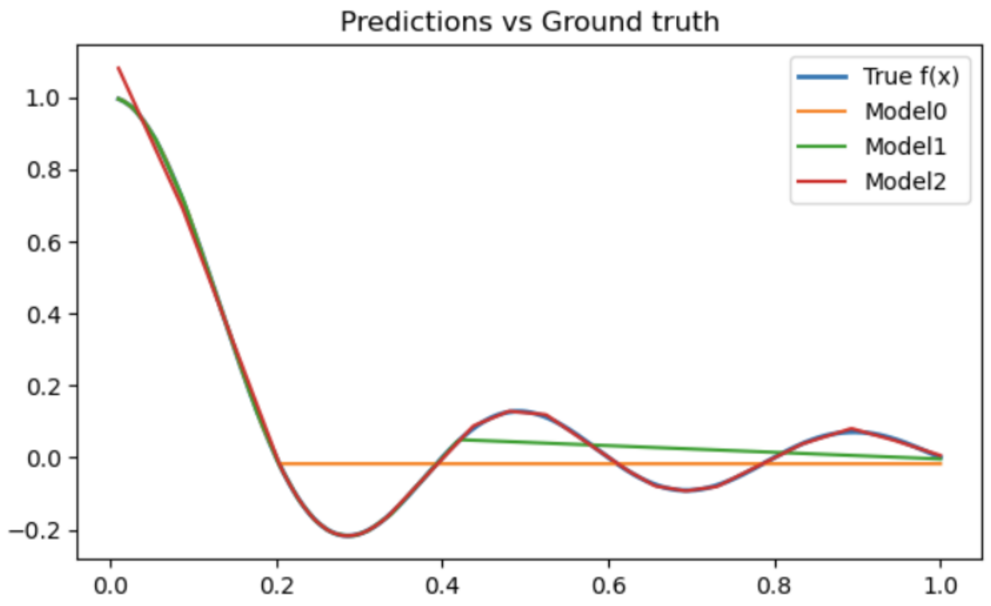
   b. In one chart, plot the training loss of all models.



Ans:

a. Model2 (green) drops fastest and reaches the lowest loss. Model1(orange) is second best. Model 0 (blue) plateaus highest.
b. With similar parameters, model 2 optimizes best and achieves the best fit. The model0 trains slowest and underfits.

c. In one graph, plot the predicted function curve of all models and the ground truth function curve.
Ans:



The true curve and model 2 curve are closely matched. Model1 follows overall trend but smoothing out the oscillations.
Model0 collapses near zero indicates underfitting.

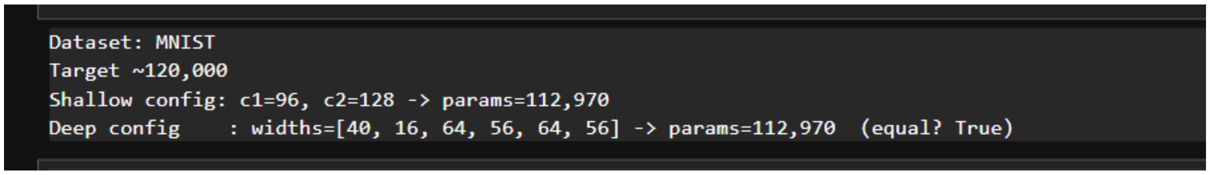
2. Train on Actual Tasks

a. Describe the models you use and the task you chose.
Ans:
Task : MNIST digit classification. Input normalized, batch=128 , 8 epochs

Models : ShallowCNN and DeepCNN

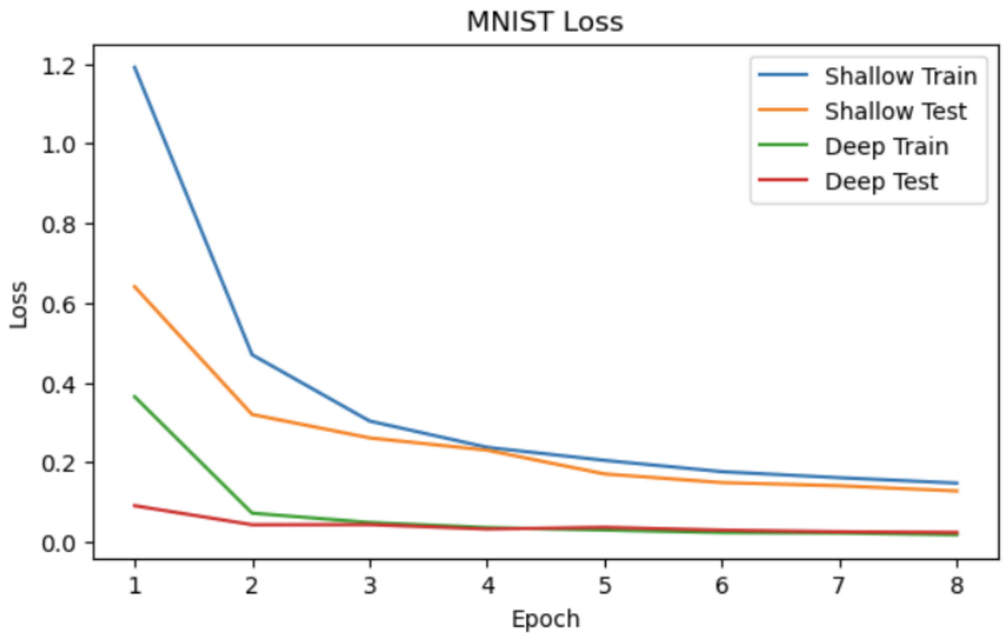
1.ShallowCNN : conv3X3(c1)-> ReLU-> MaxPool-> conv3X3(c2)->ReLU->MaxPool->GAP->FC(10). params = 112,970

2.DeepCNN : 4-6 Conv3X3 layers with widths (40, 16, 64, 56, 64, 56),MaxPool after every 2 Conv -> GAP -> FC(10). Params = 112,970



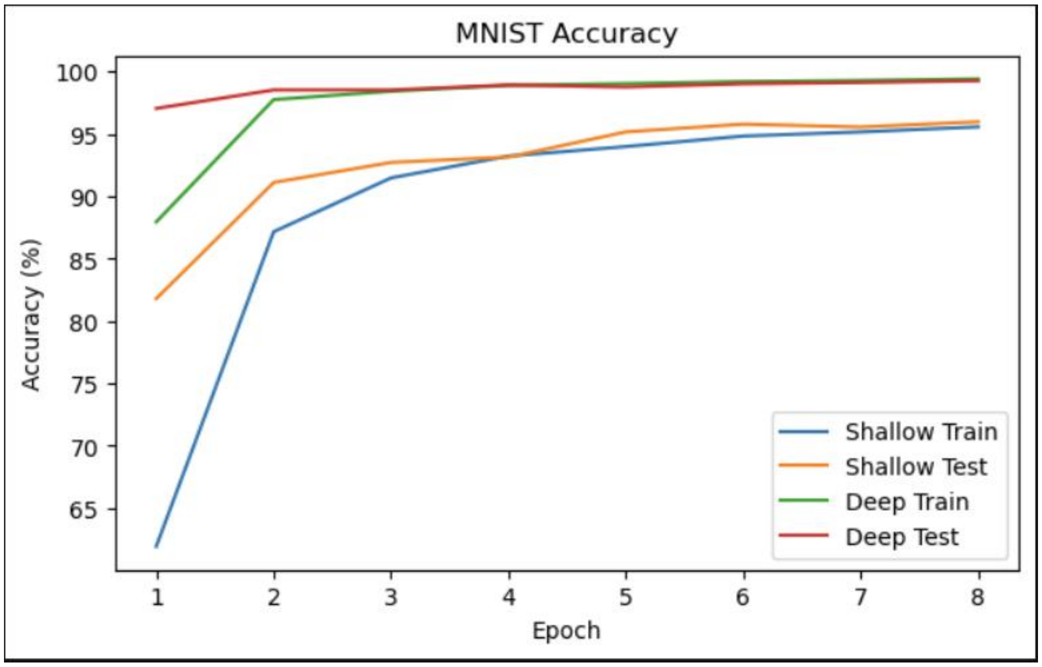
```
Dataset: MNIST
Target ~120,000
Shallow config: c1=96, c2=128 -> params=112,970
Deep config    : widths=[40, 16, 64, 56, 64, 56] -> params=112,970  (equal? True)
```

b. In one chart, plot the training loss of all models.



The DeepCNN converges much faster and to lower loss for train and test both.
The curves of DeepCNN flattens within 2-3 epochs while ShallowCNN keeps declining but stays higher throughout.

c. In one chart, plot the training accuracy.



The DeepCNN achieves 99-100% accuracy both train and test by ~3 epochs and stays as it is.
The shallow CNN climbs steadily from ~62%(train) / ~82%(test) to ~(95-96%) .
This shows faster convergence and better generalization for deep CNN with same parameter budget.

HW 1-2 Report

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

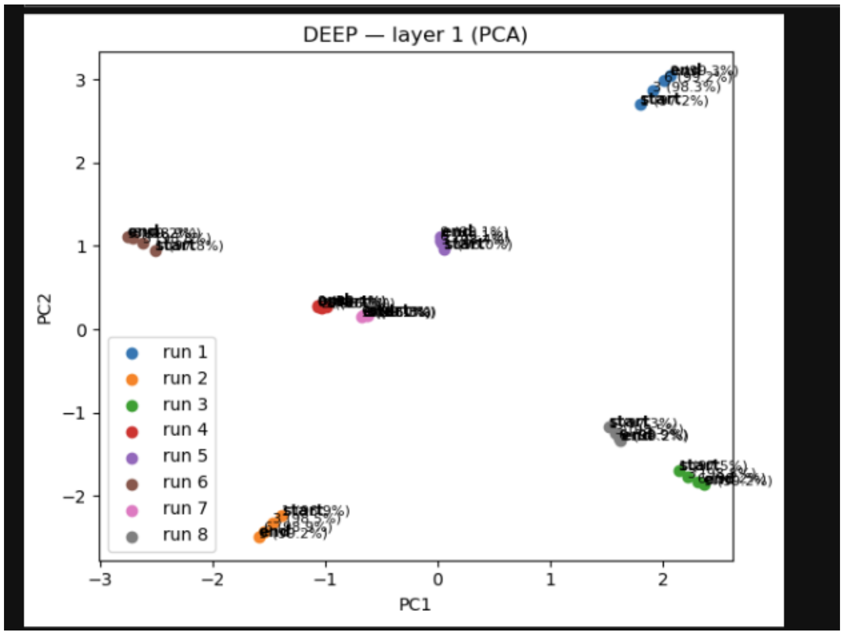
Ans: Experiment Settings = MNIST(28X28,in_ch=1) with DeepCNN (same as before), Adam optimizer(lr=1e-3), batch size = 128, CrossEntropy loss , trained for 9 epochs.

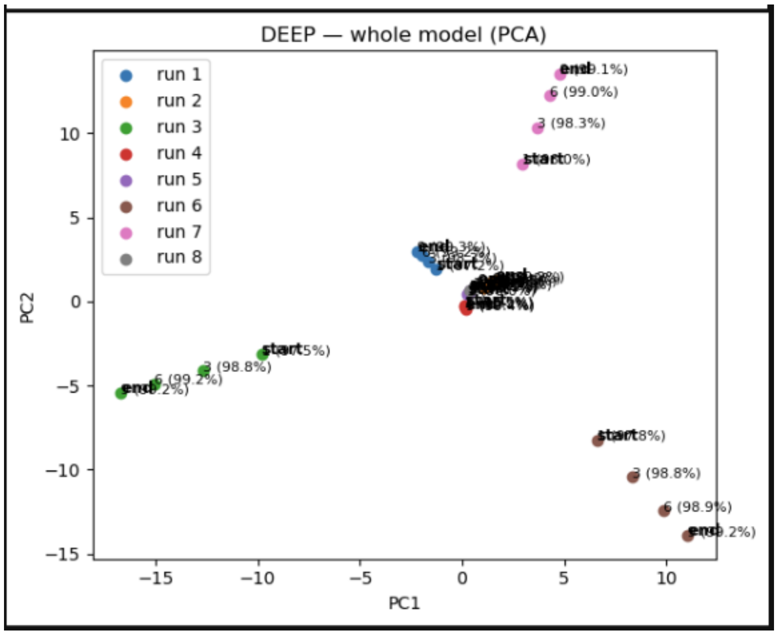
Recorded the parameters at epochs 1,3,6,9(Snapshot_Every=3) across 8 independent runs

For visualization,flatten both layer 1 and whole model wights and apply one global PCA to all snapshots. Finally plotted PC1 vs PC2 with start ,end and accuracy labels.

   b. Train the model for 8 times, selecting the parameters of any one layer and whole model and plot them on the figures separately. Comment on your result.

Ans:

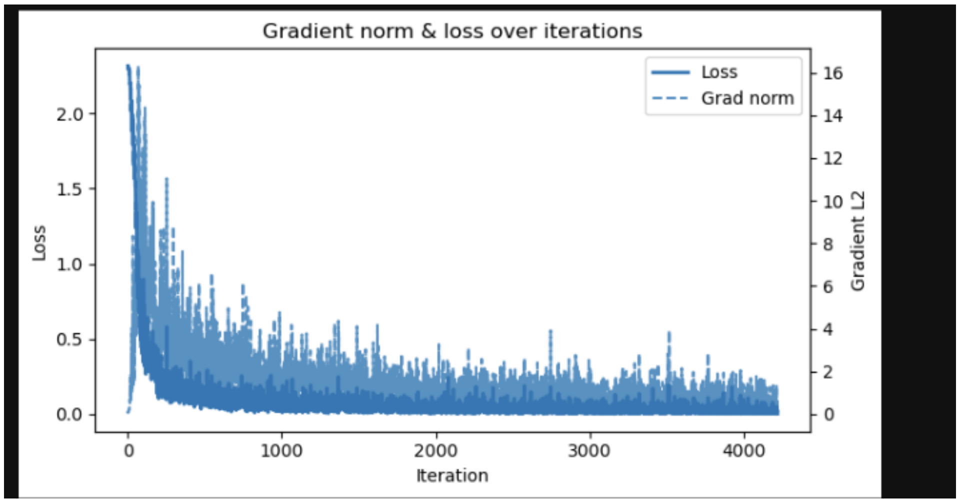




Across all the 8 runs, along the low dimensional path both the plots show consistent star and end drift. It reached 98-99% test accuracy with end points clustering.

The Whole model PCA executes much larger spread than the layer-1 PCA, indicate during training later layers contribute most to parameter variation, while early features stabilize quickly.

2. Observe gradient norm during training.
    a. Plot one figure which contains gradient norm to iterations and the loss to iterations.



Both loss and gradient L2 norm drop sharply in the first few hundred iterations then decay slowly towards a plateau.

The spikes are due to mini-batch stochasticity. Even if noise was there, gradient norms generally coincide with lower loss and once grad norm comes near 0-1 the loss stabilizes around a small value indicating convergence.

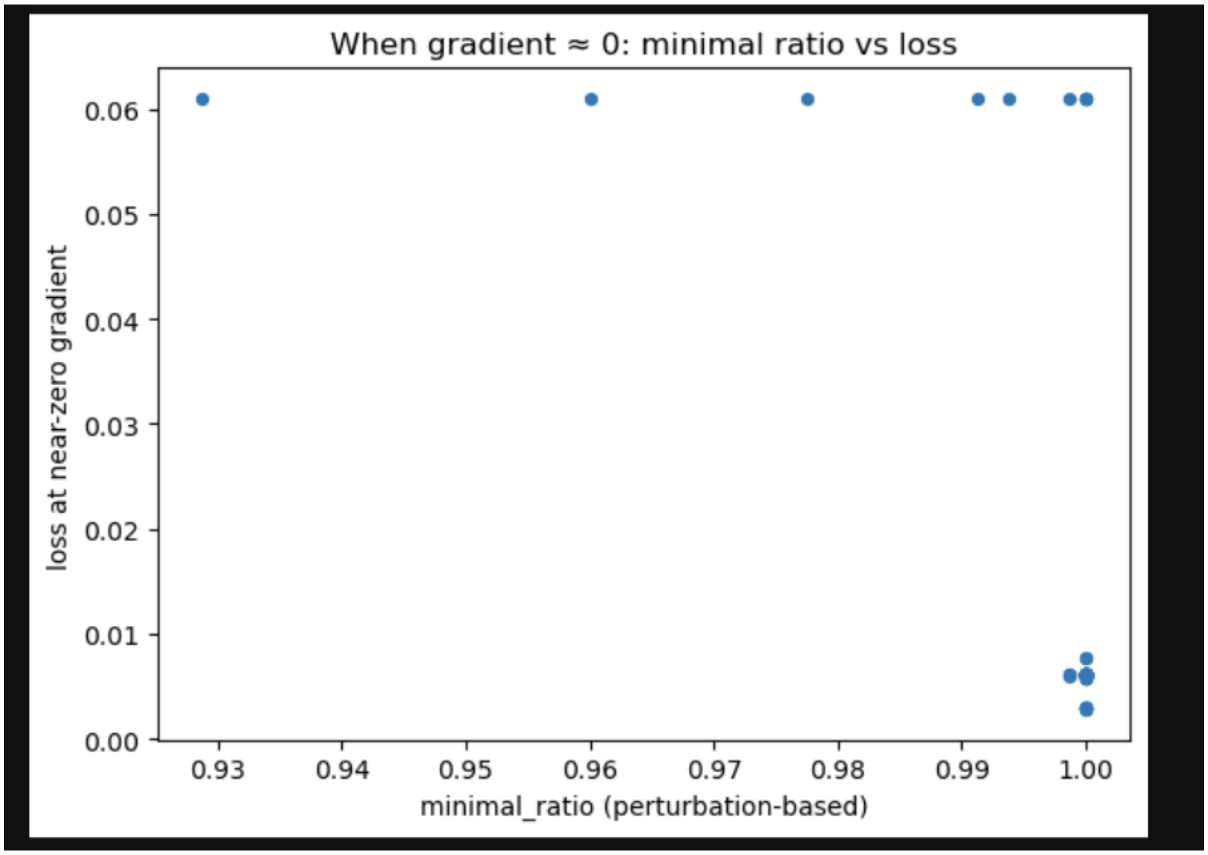
3. What happens when gradient is almost zero?

State how you get the weight which gradient norm is zero and how you define the minimal ratio.

Train the model for 100 times. Plot the figure of minimal ratio to the loss.

Comment your result.

Ans:



Trained a tiny MLP on synthetic function until convergence. I checked the final weights specifically for L2 norm of all parameter gradients.

How minimal ratio is defined = I use a perturbation-based estimate which samples gaussian noise around theta, restored the model at each time and computed

Result:

Most runs land at high minimal ratio with low loss, indicating it is true minima with locally positive curvature. A few runs show higher loss even with high ratios. This tells that not all minima are equally good.

HW 1-3 Report

1. Can network fit random labels?
   a. Describe your settings of the experiments. (e.g. which task, learning rate, optimizer)

   Ans:

   Experiment : MNIST classification with a small CNN(3 conv blocks + GAP -> FC)

   Trained on randomly permuted training labels, test labels unchanged.

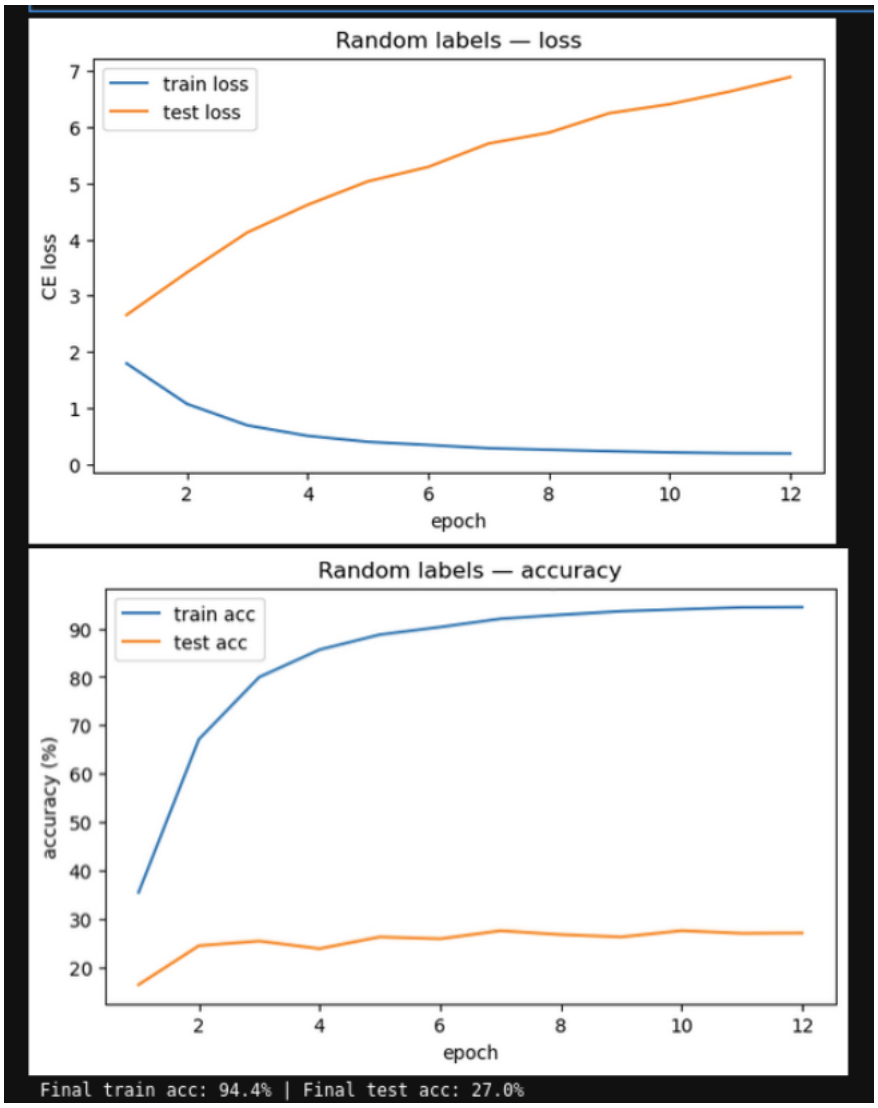
   Optimizer : Adam, lr=1e-2, batch sizes 128(train)/512 (test), 12 epochs on a 12k sample train subset.

   Loss: Cross Entropy .

   Metrics : train/test loss and accuracy each epoch.

b. Plot the figure of the relationship between training and testing, loss and epochs

Ans:



```
Final train acc: 94.4% | Final test acc: 27.0%
```

The curve shows training loss drops to 0 and training accuracy rises to 94% , while test loss steadily increase and test accuracy stays low 27%

This growing generalization gap mans the network is fitting noise in the training set rather than learning meaningful structure.

2. Number of parameters v.s. Generalization
    a. Describe your settings of the experiments. (e.g. which task, the 10 or more structures you choose)

Ans:

Task : MNIST classification with smallCNN backbone.

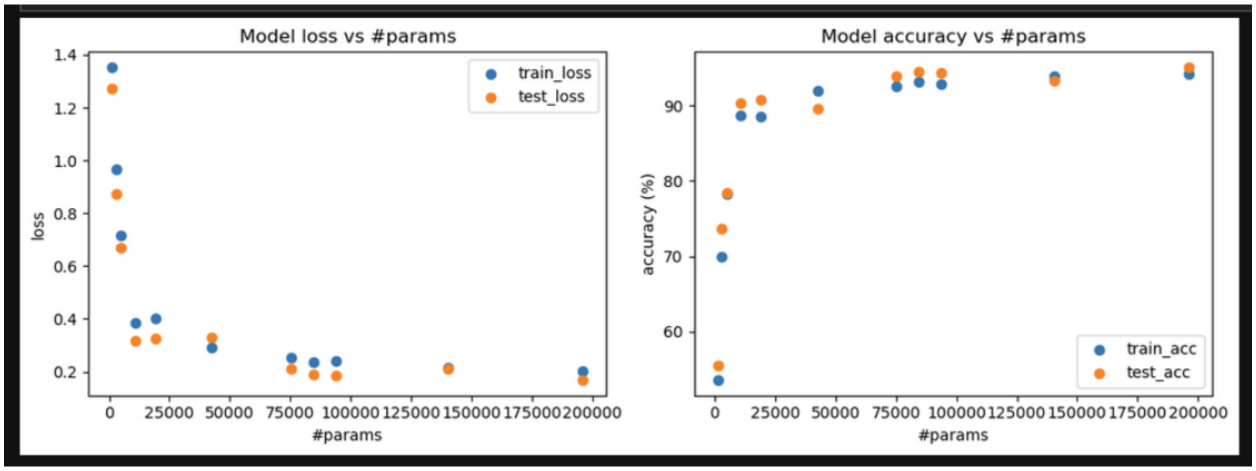
Architecture: 11 variants by changing conv channels (c1,c2) ~(8,8),(12,12),(16,16),(24,24),(32,32),(48,48),(64,64),(80,64),(96,64),(112,80),(128,96)

Data Loader: 20k random MNIST training subset(batch=128),full test set (batch=512)

Training: adam , lr=1e-3,5 epochs, cross-entropy

    b. Plot the figures of both training and testing, loss and accuracy to the number of parameters.

Ans.



As parameter increases , train loss drops and train accuracy increases rapidly.

It shows higher capacity fits data more easily.

Test loss/accuracy improves up to ~10 raise to 5 params and then saturate , indicates diminishing returns than overfitting in this range.

3. Flatness v.s. Generalization
    a. Part 1 Describe the settings of the experiments (e.g. which task, what training approaches)

Ans: Setting:

Task/data : MNIST digit classification.

Model : ShallowCNN(conv channels 32->64,GAP +FC).

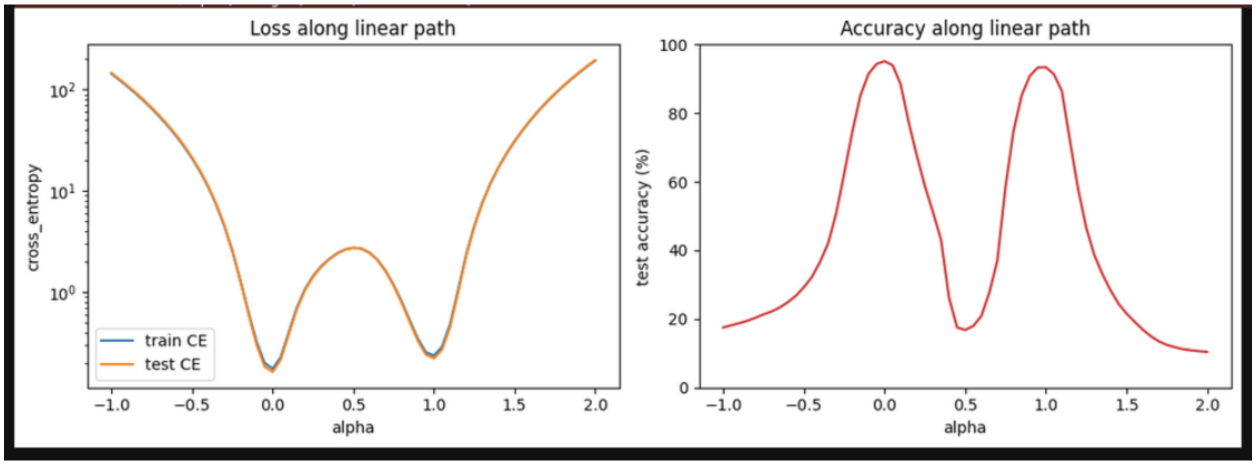
Two training approaches : train two separate solutions with Adam + CE for 6 epochs

A(bs =64 , lr=1e-3), B (bs =1024, lr=1e-2).

Path Exploration : form interpolated weights, computed train/test and test accuracy along the path

    b. Plot the figures of both training and testing, loss and accuracy to the number of interpolation ratio.

Ans:



The double well CE curve has minima near alpha = 0 and alpha =1 .

Accuracy peaks near each endpoint and dips sharply between them, mirroring the loss barrier.

Outside the interval alpha = 0 and alpha =1 both loss increases and accuracy collapses, confirming that simpler linear interpolation moves the model into high loss region except near the original solution.

4. Part 2 Flatness v.s. Generalization

   a. Describe the settings of the experiments (e.g. which task, what training approaches

   Ans.

   Task : MNIST classification (normalized)

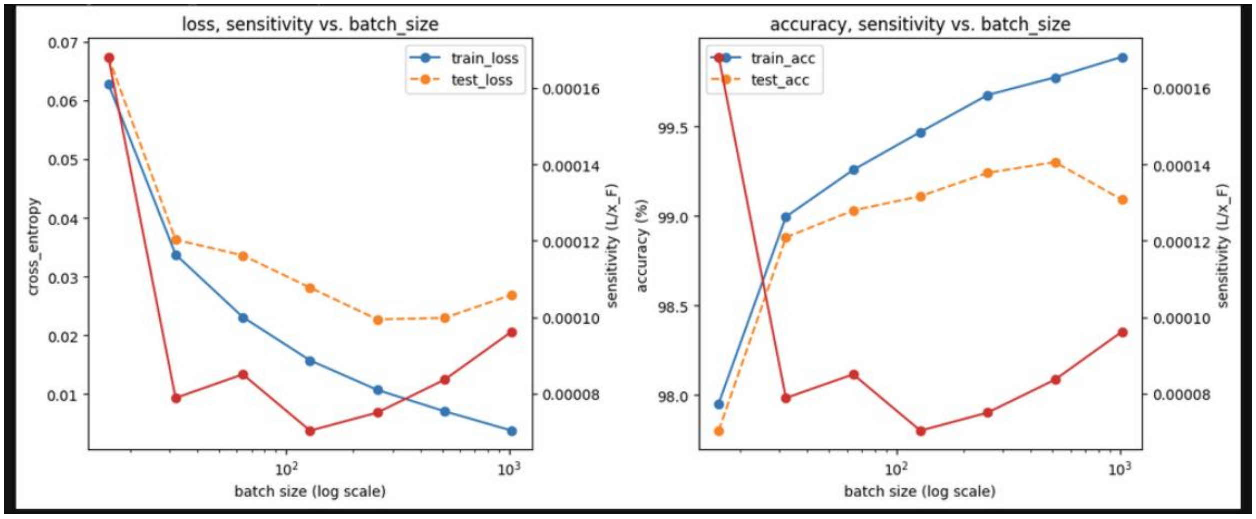
   Model : SimpleCNN (3conv blocks 32-> 64-> 128, Gap,FC)

   Training : Adam, lr=1e-3 , CE loss, fixed 6,000 optimizer steps with batch sizes(16,32,64,128,256,512,1024)

   Metrics: train/test loss accuracy , plus sensitivity = average frobenius norm of input gradients

   b. Plot the figures of both training and testing, loss and accuracy to the number of interpolation ratio.

   Ans.



   As batch size increases , train loss/acc increases steadily.

   Test loss/acc increases upto 128 to 512 batch but then slightly decreases at 1024

Sensitivity drops sharply from 16 to 128 batch size but then stays low around 256 to 512 batch size , then rises at 1024.

The best generalization coincides with lower sensitivity. The mid range batches performs better as compared to the very small and large batches.