HW1 Report of Asif Ahmed Khan

Github repository link: https://github.com/AsifKhan-27/CPSC8430_Deep-Learning/tree/main

Question 1-1: Deep vs Shallow

Part 1 – Simulate a function

Function used: sin(x)

Shallow model:

Number of parameters – 346

Hidden layers – 2

Activation layer – ReLU

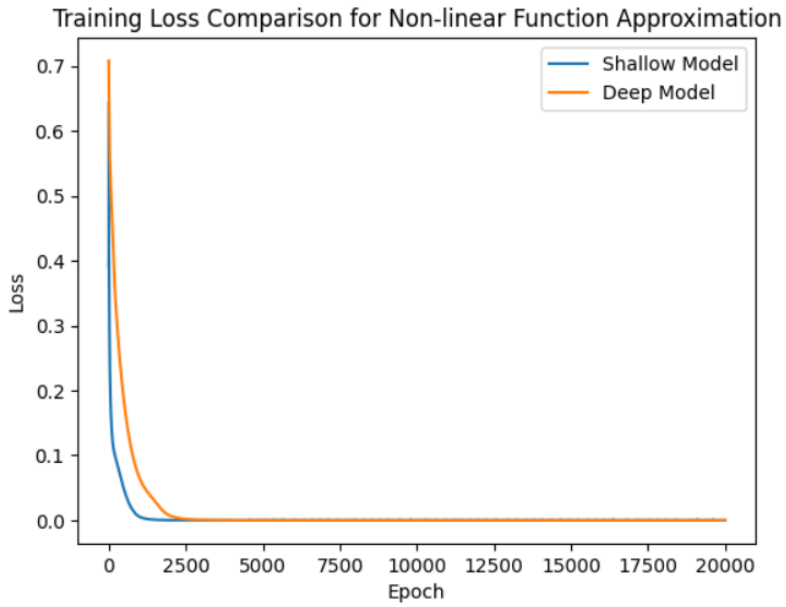
Deep model:

Number of parameters – 351

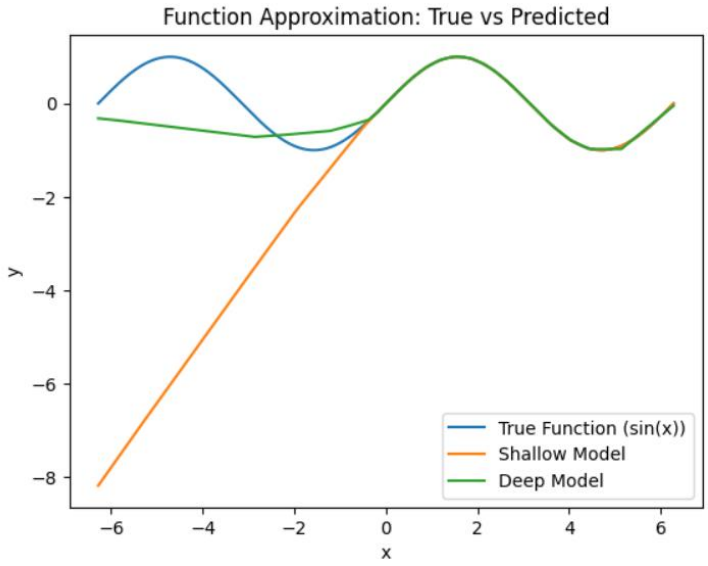
Hidden layers – 4

Activation layer – ReLU

Training loss comparison plots of models:



Prediction vs ground truth plot:



The two models have different number of layers, but they have very similar training loss with the shallow model having slightly lower loss in the initial epochs. However, the prediction form the deep model is much more accurate than the shallow model.

Part 2 – Train on actual task

Task chosen: MNIST classification

Shallow model:

Number of parameters – 39760

Hidden layers – 2

Activation layer – ReLU

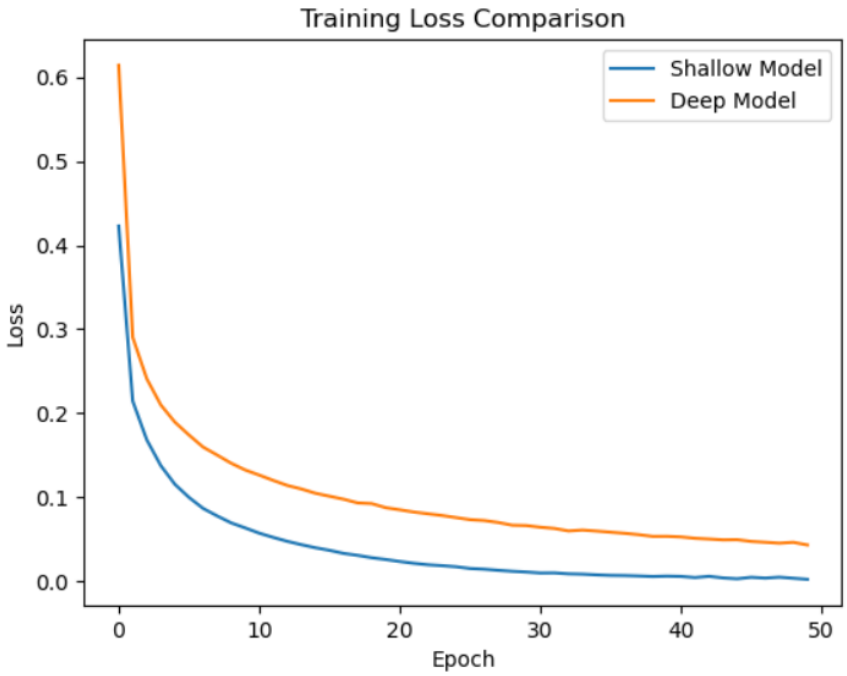
Deep model:

Number of parameters – 16750

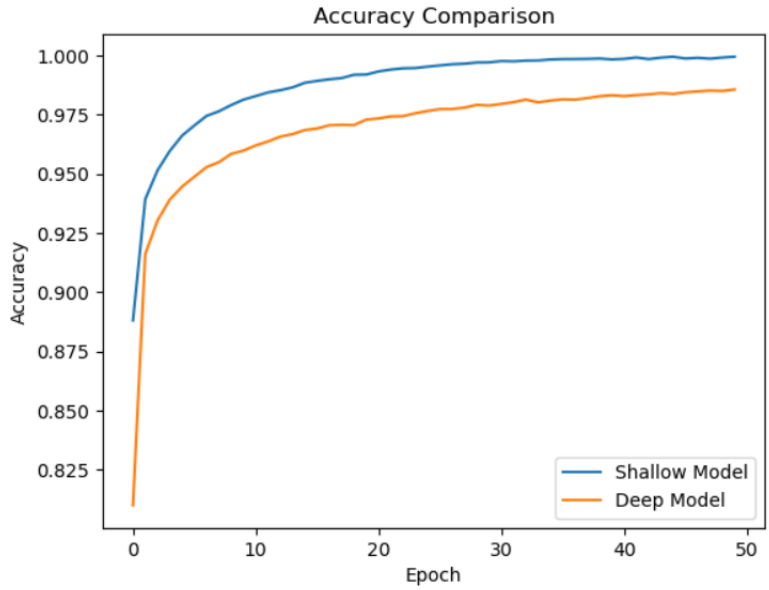
Hidden layers – 4

Activation layer – ReLU

Training loss comparison plots of models:



Training accuracy comparison plots of models:



The shallow model has lower training loss and higher accuracy in comparison to the deep model. This is because the shallow model has a much higher number of model parameters.

Question 1-2: Optimization

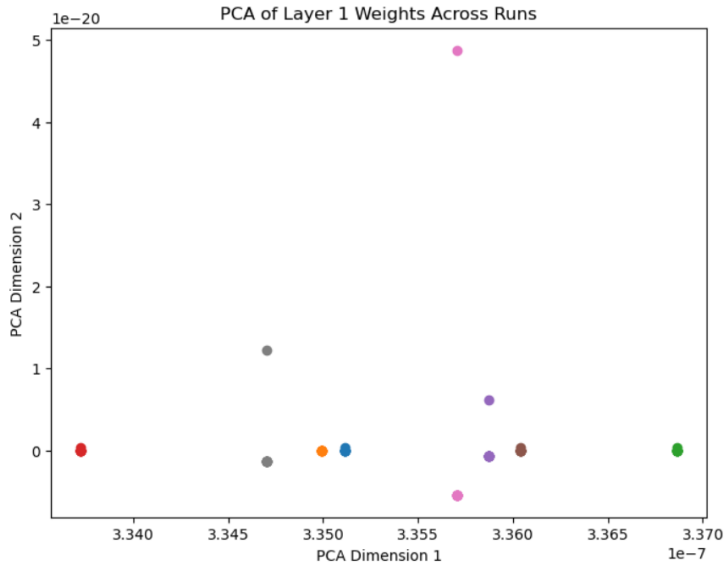
Part 1 – Visualize the optimization process

The experiment trains a DNN on the MNIST dataset for 30 epochs using Adam optimizer and CrossEntropyLoss loss function, recording model parameters every 3 epochs. Gradient norms and

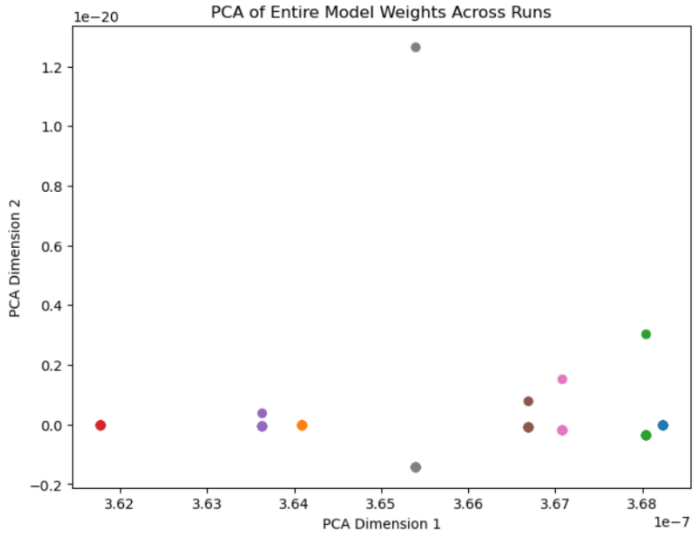
loss are tracked throughout training, with weights collected and flattened into a matrix for the first hidden layer and the entire model separately with the model being run 8 times.

PCA has been used to reduce the dimensions of the collected weights to two dimensions for better visualization of the process. The PCA plot is shown below:

PCA plot for 1st hidden layer parameters:



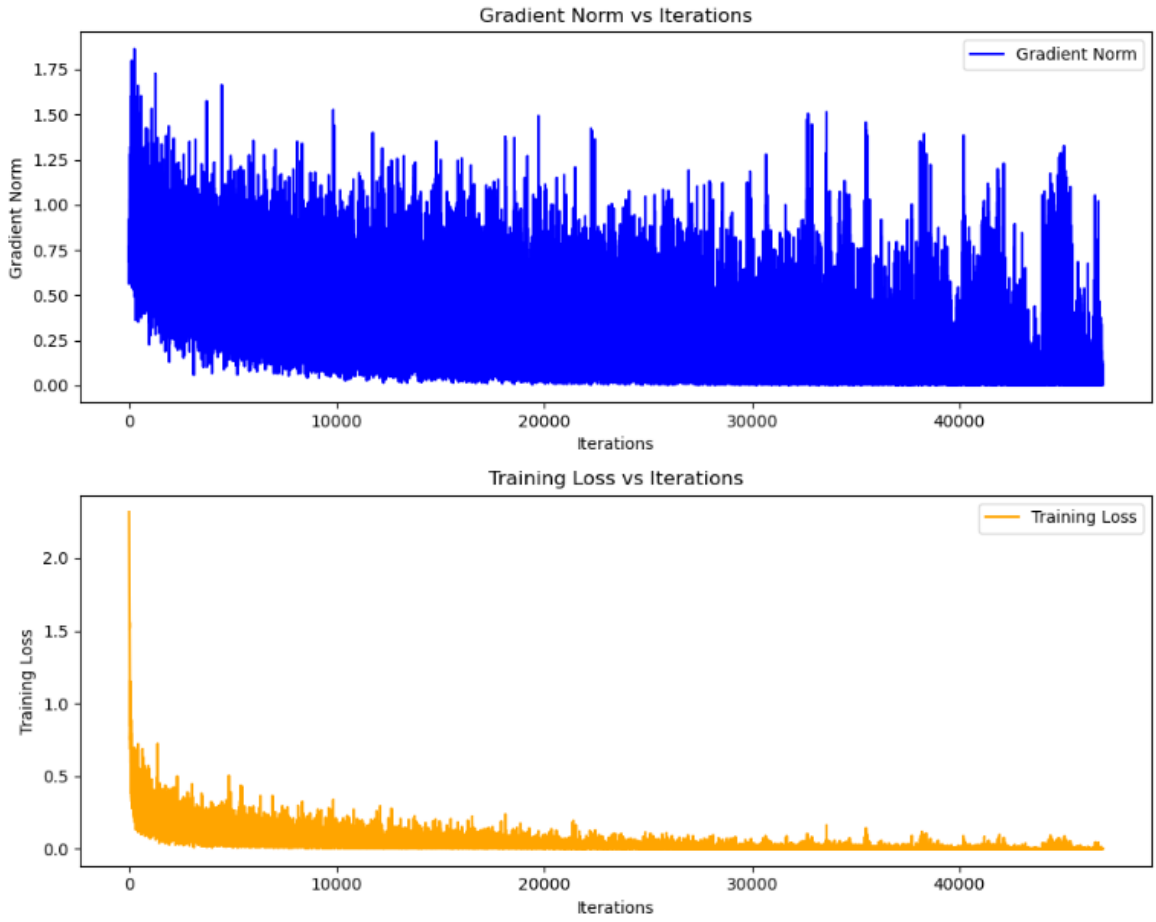
PCA plot for entire model parameters:



The 2D scatter plot from PCA shows the evolution of the model's parameters across epochs and runs, revealing distinct trajectories that reflect the variability in training dynamics but convergence toward a similar region in weight space. This suggests consistent learning patterns across different initializations.

Part 2 – Observe gradient norm during training

Plot of gradient norm and training vs iterations:



The training loss decreases steadily, showing effective learning across all runs, but may suggest room for further optimization if the loss plateaued early.

Question 1-3: Generalization

Part 1 – Can a network fit random labels

Task: Classification of MNIST dataset

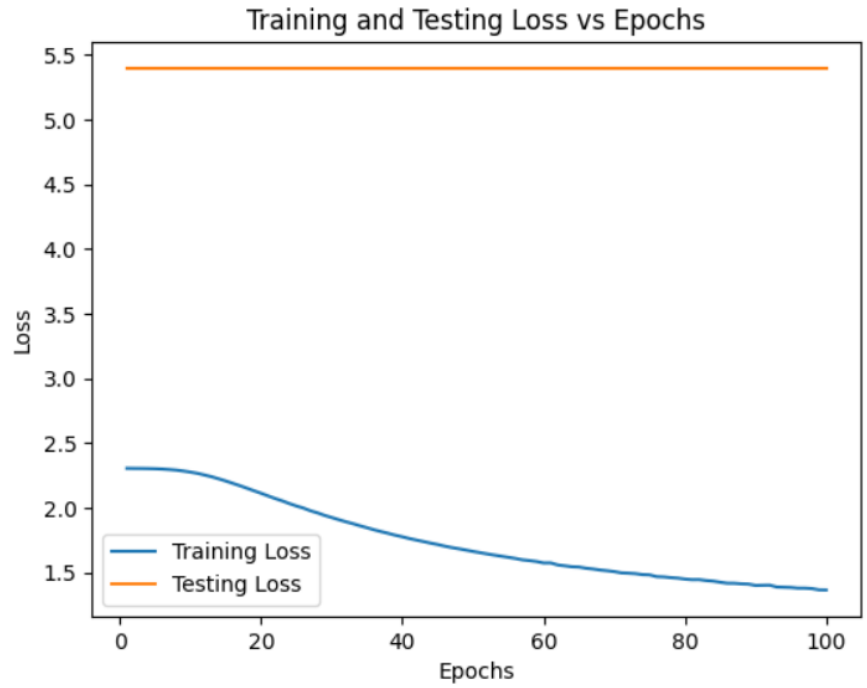
Model: DNN with 3 hidden layers

Optimizer: Adam Optimizer

Learning rate: 0.001

Loss function: Cross-entropy loss

Plot of training and testing loss against epochs:



Part 2 – Number of parameters vs Generalization

Daa

Task: Classification of MNIST dataset

Optimizer: Adam Optimizer

Learning rate: 0.001

Loss function: Cross-entropy loss

Model structures of the 10 models are as follows:

Model 1: 2 hidden layers with 10 and 5 neurons respectively

Model 2: 2 hidden layers with 20 and 10 neurons respectively

Model 3: 2 hidden layers with 30 and 15 neurons respectively

Model 4: 2 hidden layers with 40 and 20 neurons respectively

Model 5: 2 hidden layers with 50 and 25 neurons respectively

Model 6: 2 hidden layers with 60 and 30 neurons respectively

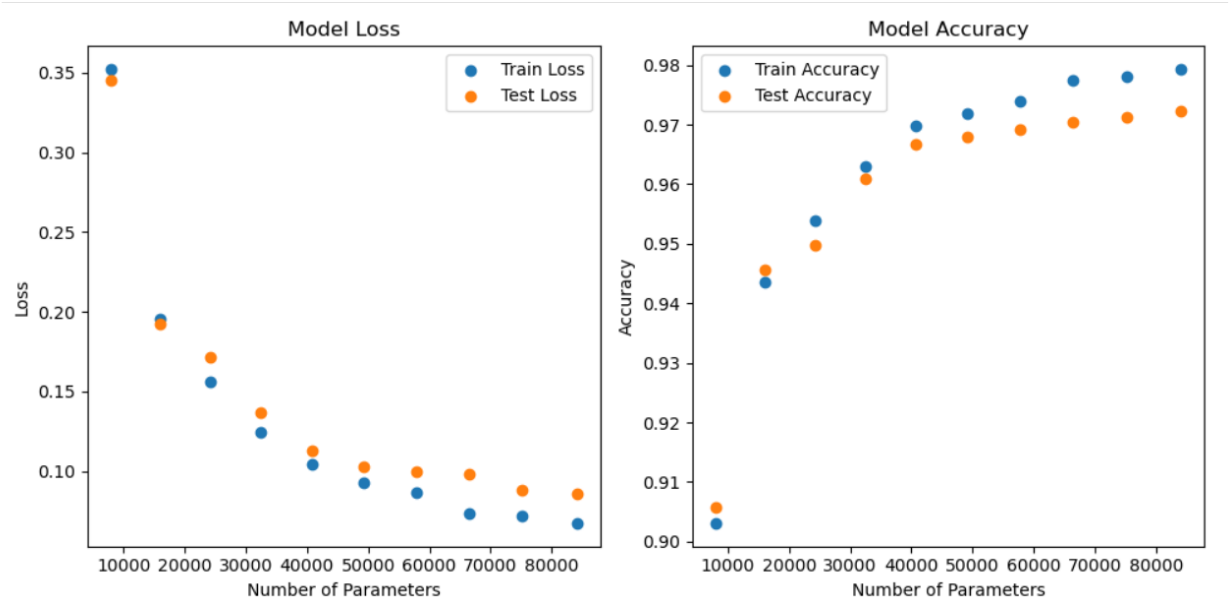
Model 7: 2 hidden layers with 70 and 35 neurons respectively

Model 8: 2 hidden layers with 80 and 40 neurons respectively

Model 9: 2 hidden layers with 90 and 45 neurons respectively

Model 10: 2 hidden layers with 100 and 50 neurons respectively

Plot of training and testing loss and accuracy against number of parameters:



The training and testing losses decrease as the number of parameters in the model increases. The training and testing accuracies increase as the number of parameters in the model increases.