CPSC8430_Deep_Learning_Homework_1
Name: Atharva R Jadhav

1-1 Simulate a Function

In this section we trained three deep neural network (DNN) models with same amount of parameters on three different non-linear single functions.

DNN models used were as follows

Model 1:

- 8 layers
- Activation function: ReLU
- Total number of parameters: 861
- Loss function : “MSELoss”
- Optimizer: “RMSprop”
- Hyperparameters:
    - Learning rate = 0.0012
    - Weight_decay = 1e-4

Model 2:

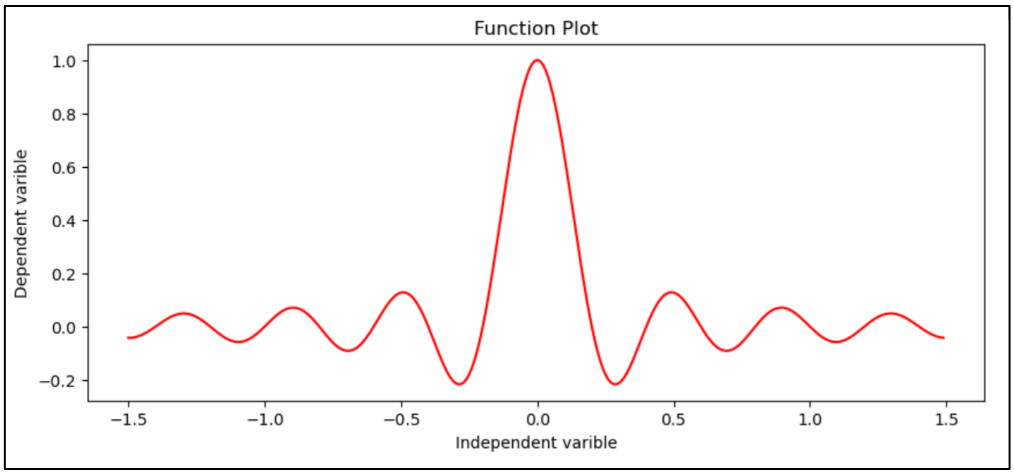
- 4 layers
- Activation function: ReLU
- Total number of parameters:752
- Loss function : “MSELoss”
- Optimizer: “RMSprop”
- Hyperparameters:
    - Learning rate = 0.0012
    - Weight_decay = 1e-4

Model 3:

- 2 layers
- Activation function: ReLU
- Total number of parameters: 751
- Loss function : “MSELoss”
- Optimizer: “RMSprop”
- Hyperparameters:
    - Learning rate = 0.0012
    - Weight_decay = 1e-4

Functions :

Function 1 : (sin(5Πx) /5Πx) :



Convergence reached at:
Model 1: Epoch 15500
Model 2: Epoch 9000
Model 3: Max Epoch Reached at 20000

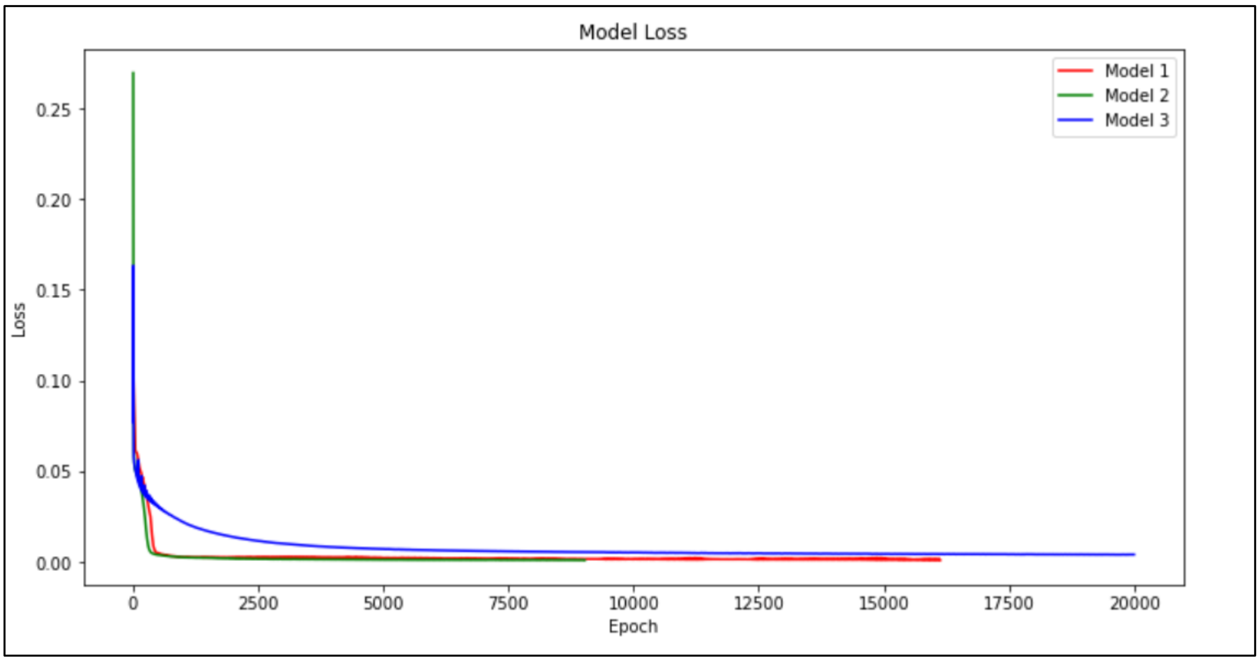


Fig. Model Loss

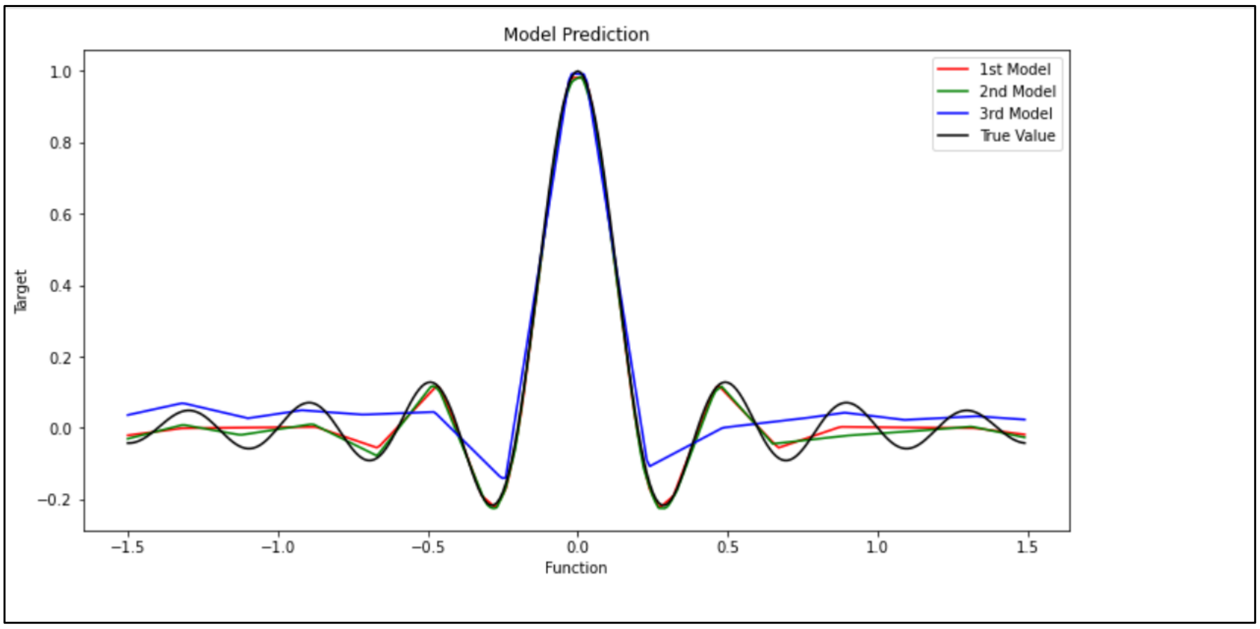


Fig. Model Prediction

Function 2: sgn(sin(5Πx))

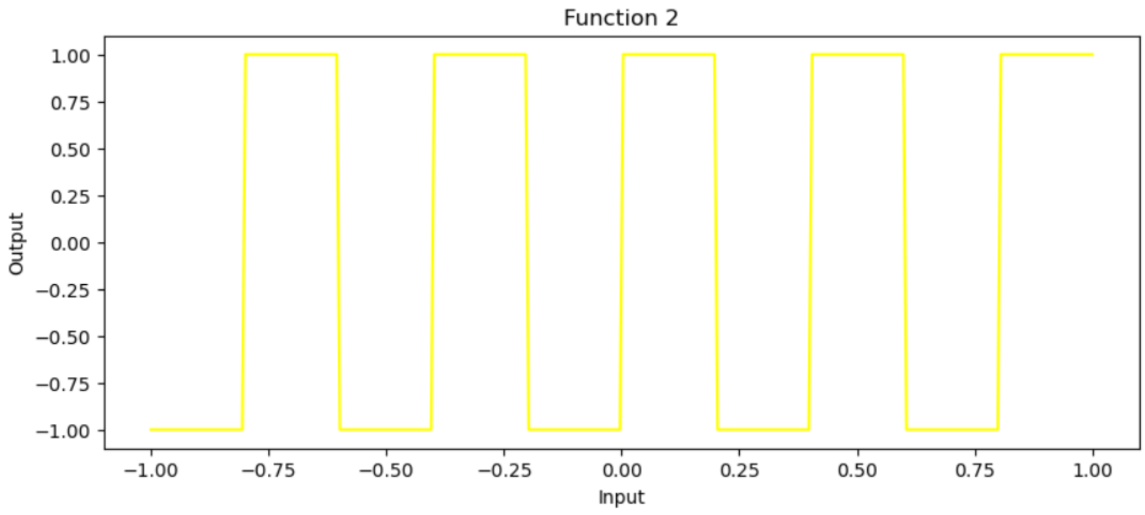


Fig 2.1: Original Model

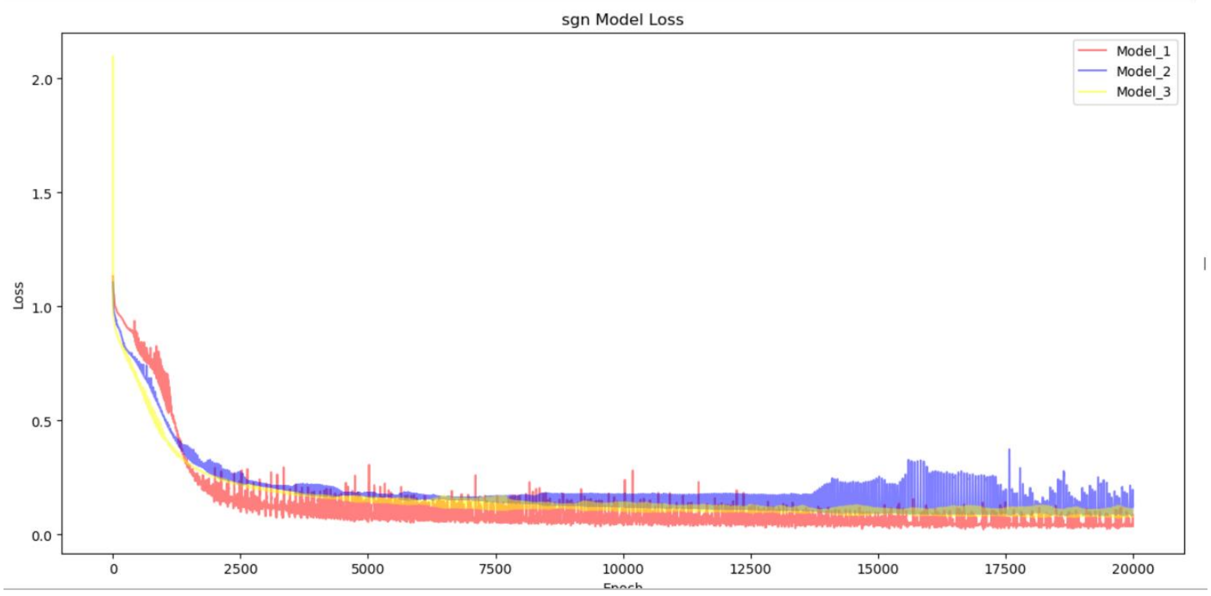


Fig2.2 Model Loss

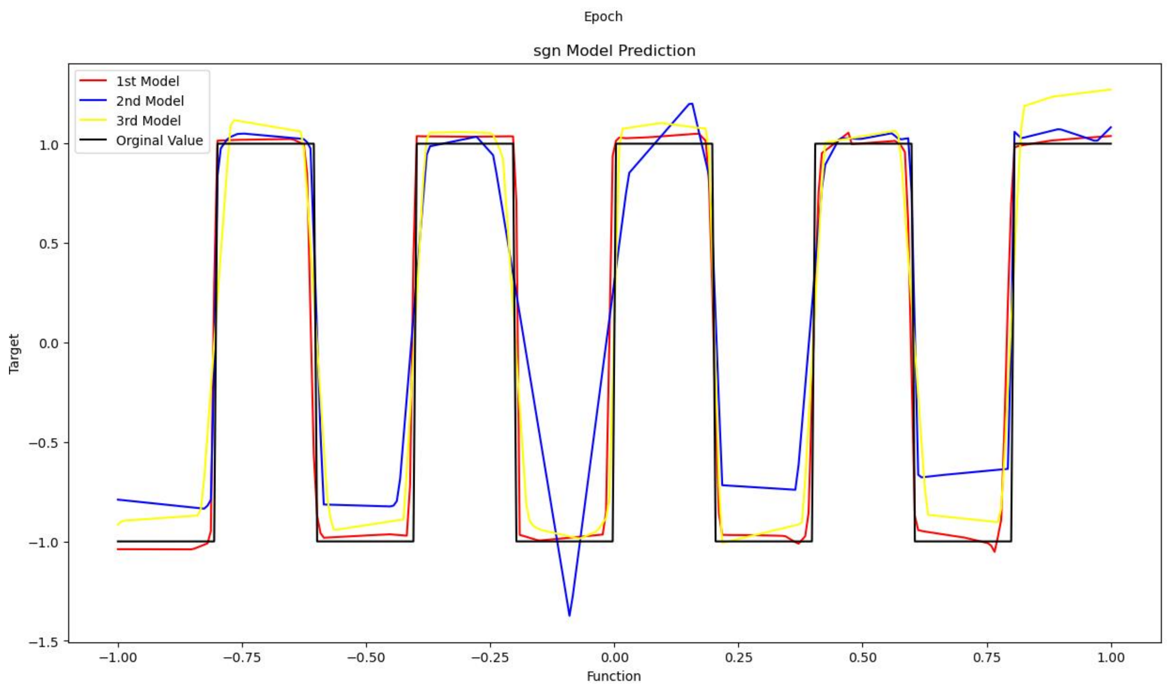


Fig 2.3 Model Predictions

Function 3 : S(x)= \frac {1}{1+e^{-x}} (Sigmoid function):

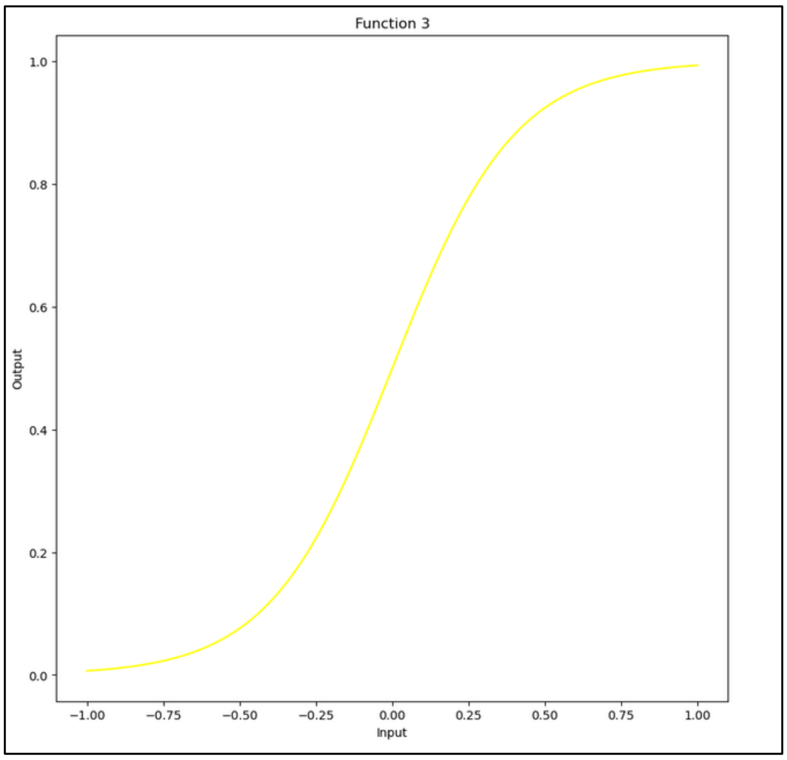


Fig 3.1 Original Function

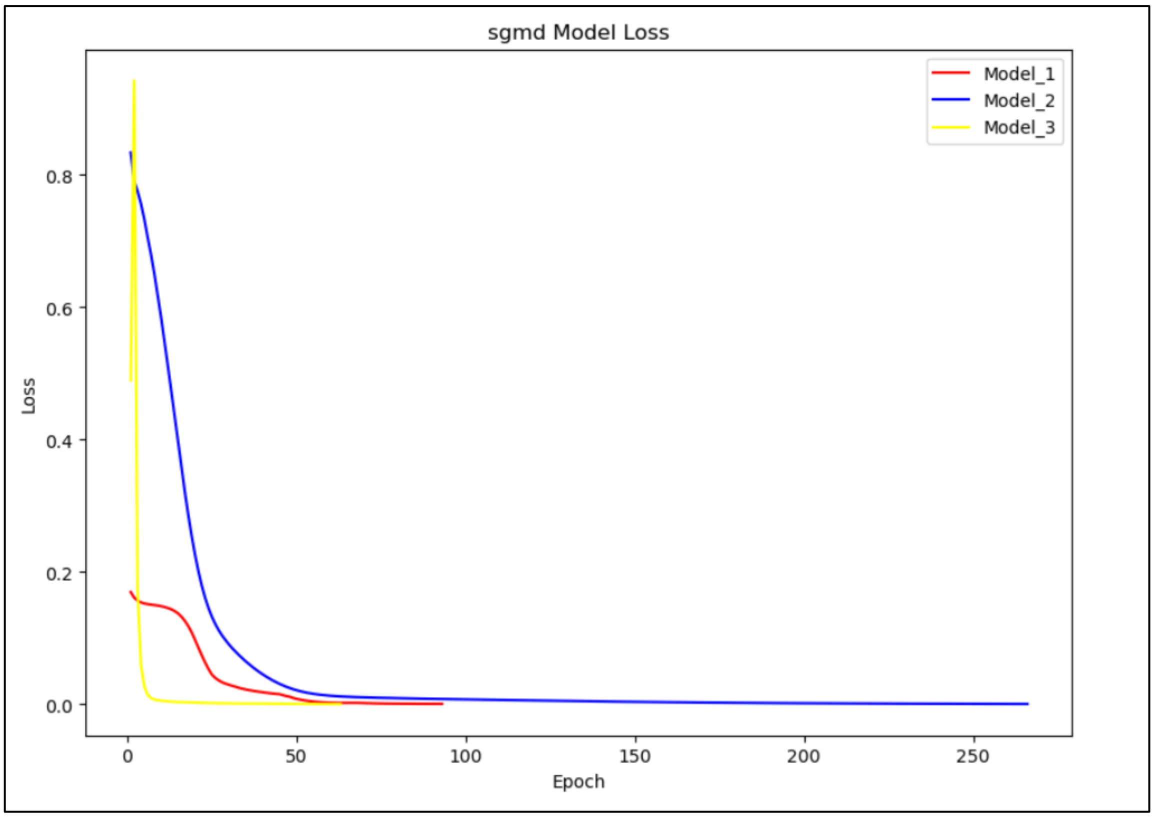


Fig. 3.2 Model loss

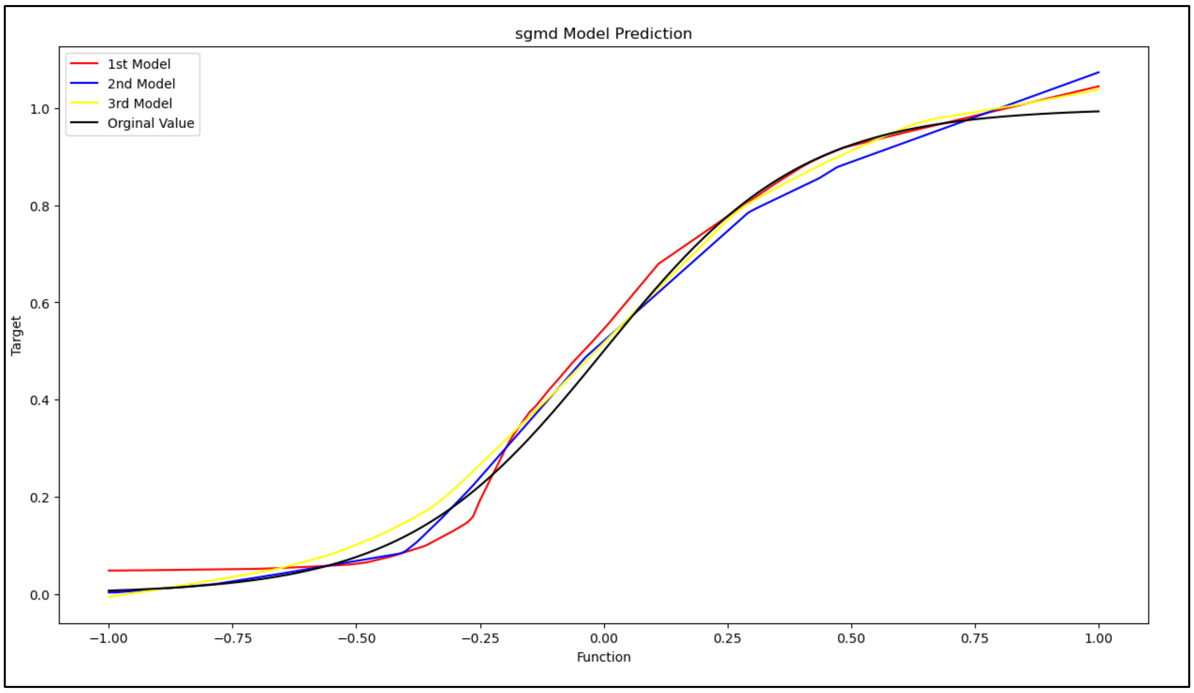


Fig 3.3 Model Predictions

Result: As seen by the models above we can infer that models with more no of hidden layers converge better in case of complicated non – linear function

1.2 Train on actual task:

We created 3 CNN models and trained them on MNIST dataset. The details of these CNN used are as follows

CNN_1:
- Convolution layers : 2
- Kernel Size : 4
- Max pool size = 2
- Activation Function : ReLU

Hyperparameters
-Learning rate: 0.001
-max_epoch:15
- Weight Decay: 1e-4

CNN_2:
- Convolution layers : 4
- Kernel Size : 4
- Max pool size = 2
- Activation Function : ReLU

Hyperparameters
-Learning rate: 0.001

-max_epoch:15
- Weight Decay: 1e-4

CNN_3:
- Convolution layers : 2
- Kernel Size : 4
- Max pool size = 2
- Activation Function : ReLU

Hyper parameters
-Learning rate: 0.001
-max_epoch:15
- Weight Decay: 1e-4

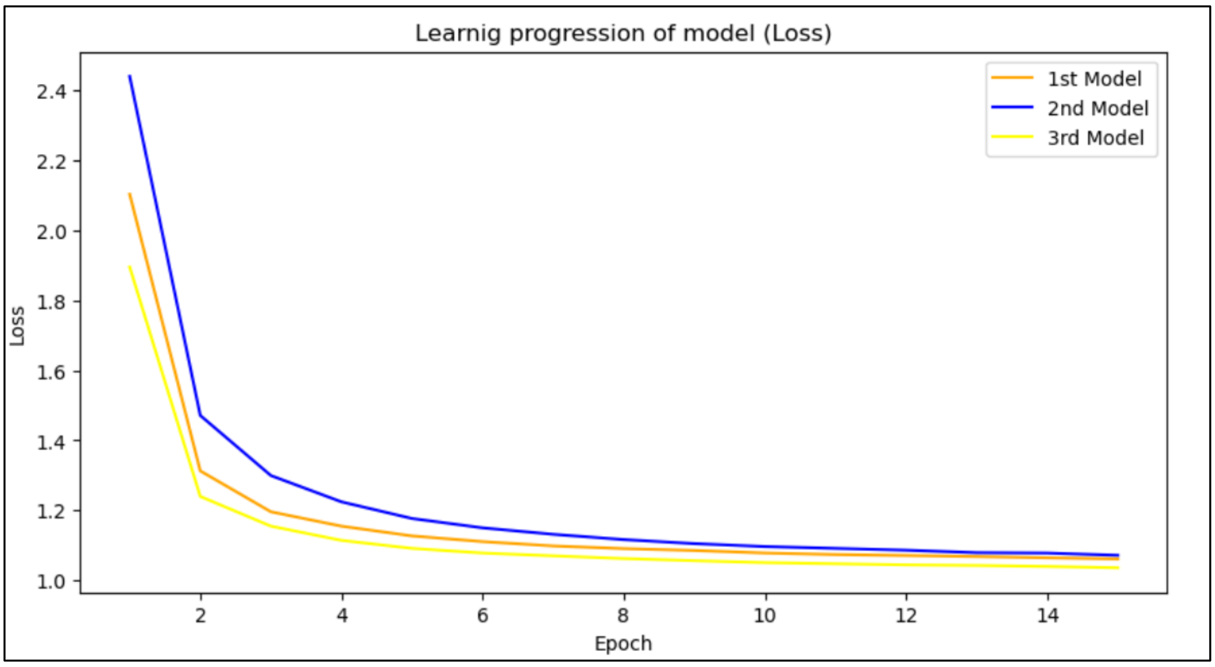


Fig.1.2.1 Loss during Learning process

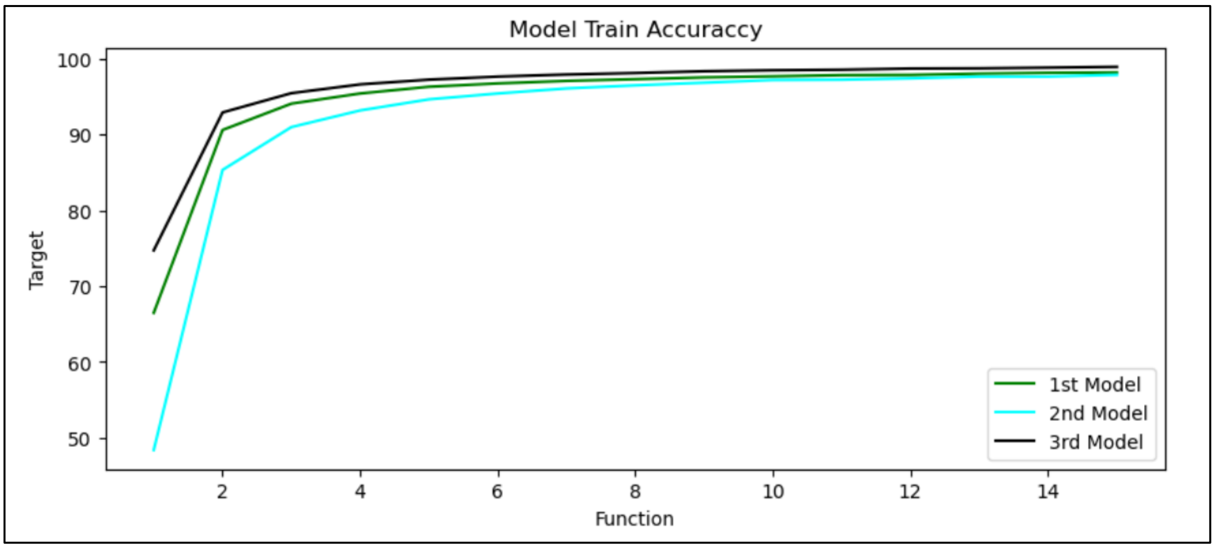


Fig. 1.2.2 Model Accuracy

Result: For this task 3rd CNN model gives the highest accuracy. Model 1 has accuracy of 98.71, whereas Model 2 has accuracy of 98.03 and finally Model 3 with the highest accuracy of 98.81

2. Optimization:

2.1 Visualize the optimization process

We used the DNN to visualize the optimization process with epoch :16, training batch size of 64 learning rate of 0.001 and interval value of 100.

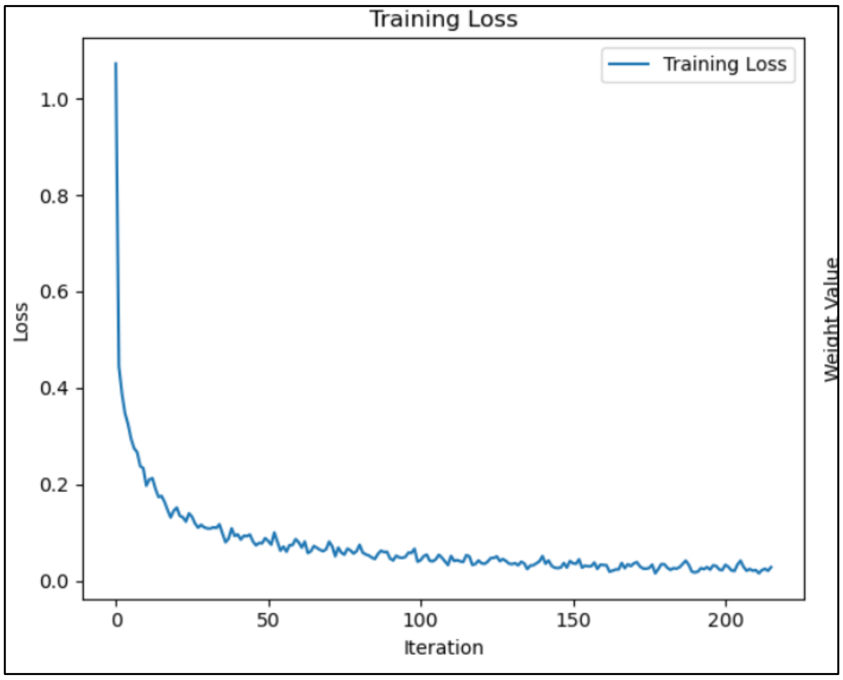


Fig. 2.1.1 Training loss

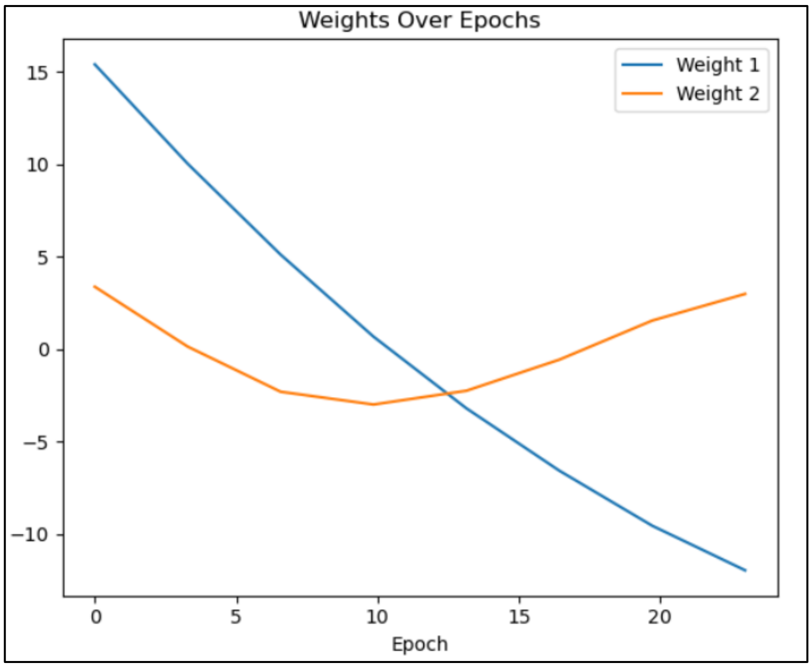


Fig. 2.1.2 Weights over epoch

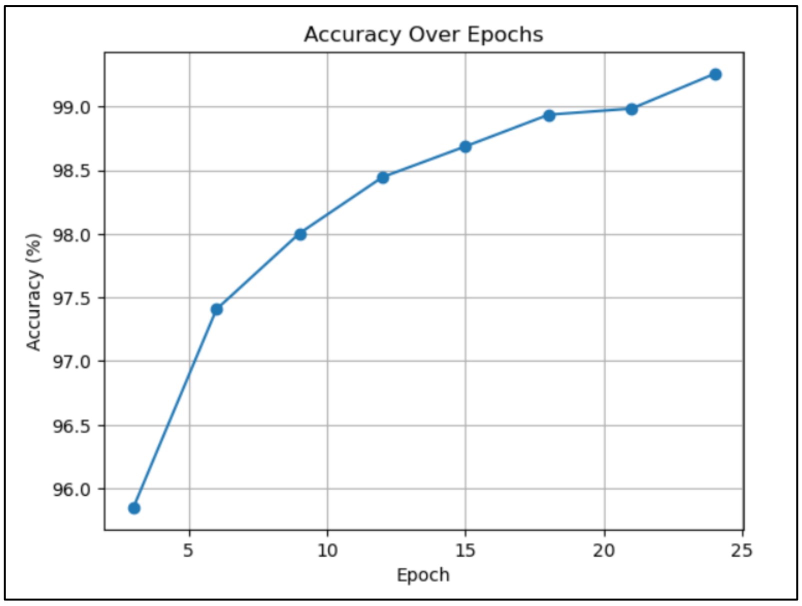


Fig. 2.1.3 Accuracy over epoch

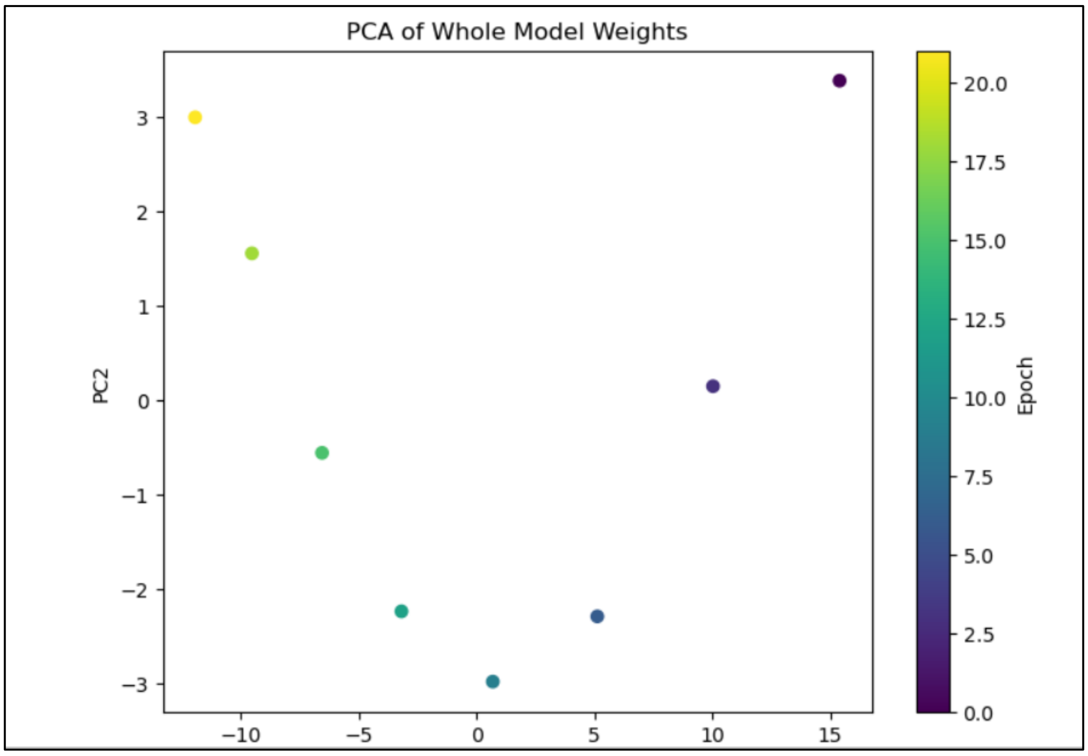


Fig.2.1.4 PCA of whole model

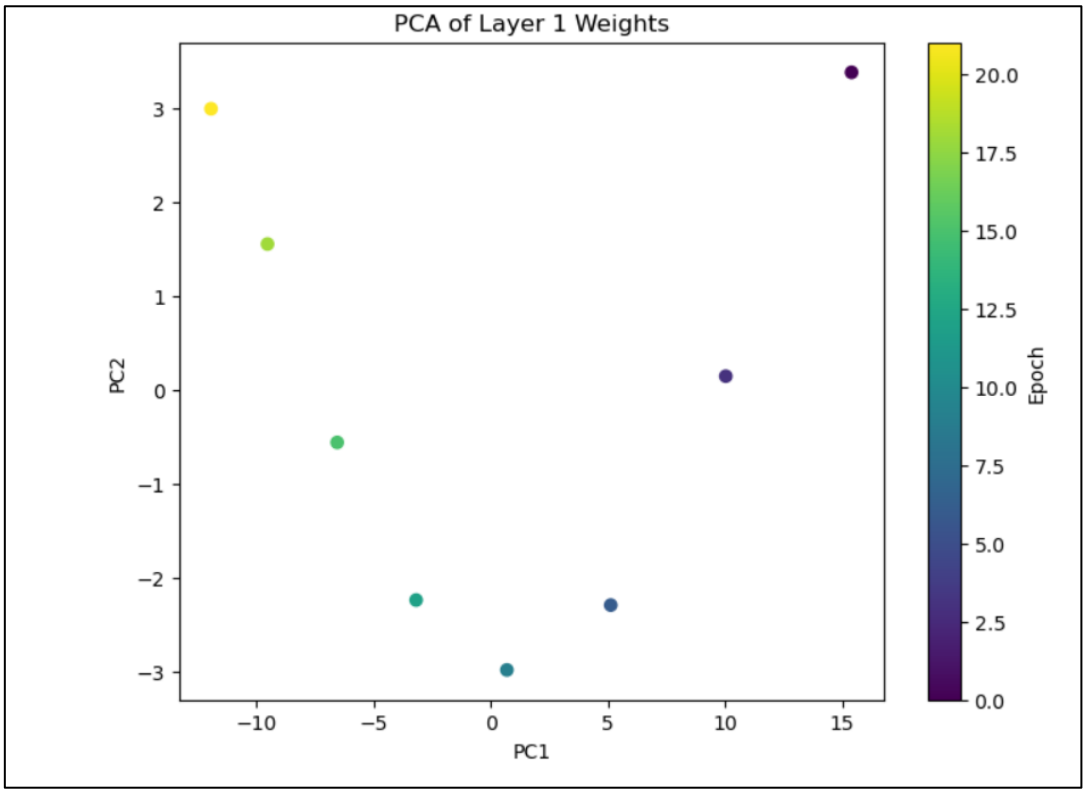


Fig 2.1.5 PCA of Layer 1

2.2 Observing Gradient norm during training

Model used: Deep Neural Network (DNN)

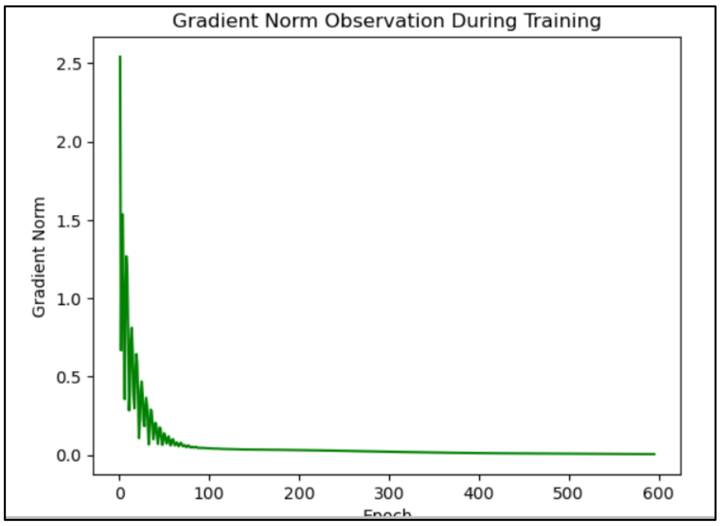


Fig. 2.2.1: Gradient norm observed during Training

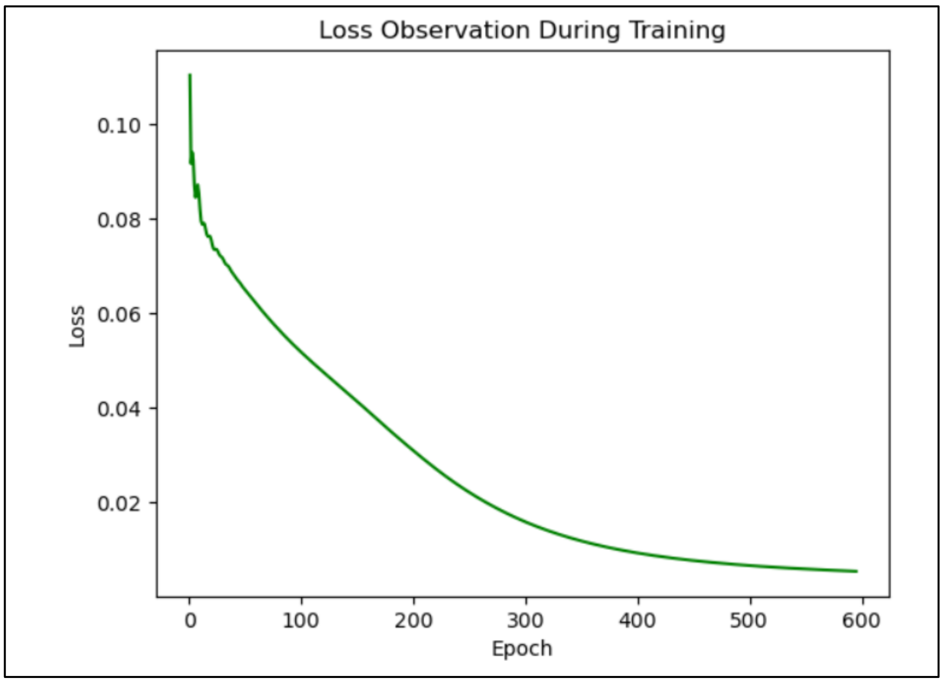


Fig 2.2.2 Loss during Training

Result: In training deep learning model iterates over updating its parameters. In our case model stopped at 600 epoch. And during the process we could see rapid decrease in gradient till epoch 100. After the gradient decrease rate slowed down indicating that model might have reached certain level of optimization.

2.3 When Gradient is almost zero (Using Functions)

In this task we observe when the gradient is zero or almost close to zero and computer minimum ratio.
Model Used: DNN
Layers : 7

Activation Function : ReLU
Loss Function: MSE Loss

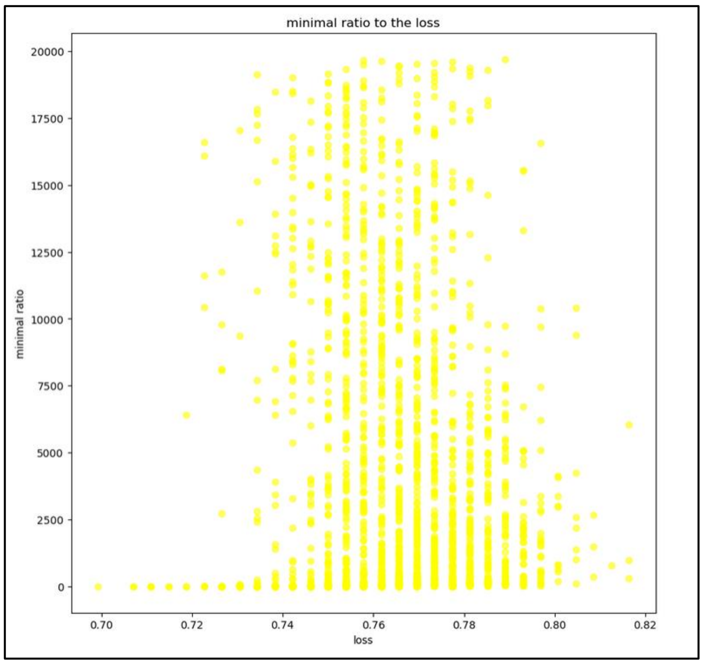


Fig. 2.3.1 Graph of Loss vs. Minimal Ratio

3 Generalization
3.1 Can our network fit random labels?

In this task we train CNN on MNIST dataset with randomized labels.

CNN used:
Layers: 2
activation function: ReLU
Optimizer: Adam
Loss Function: CrossEntropyLos( )

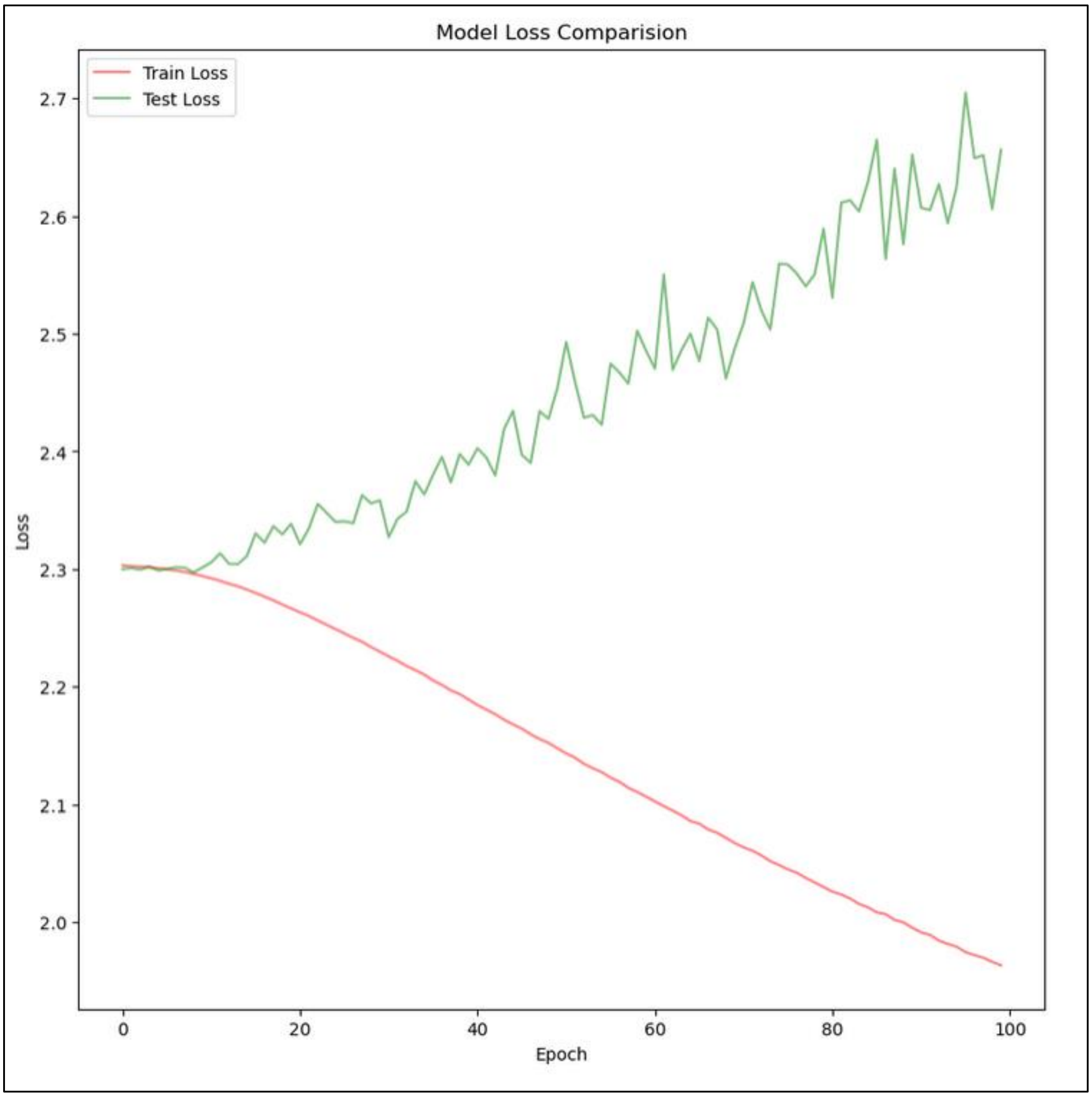


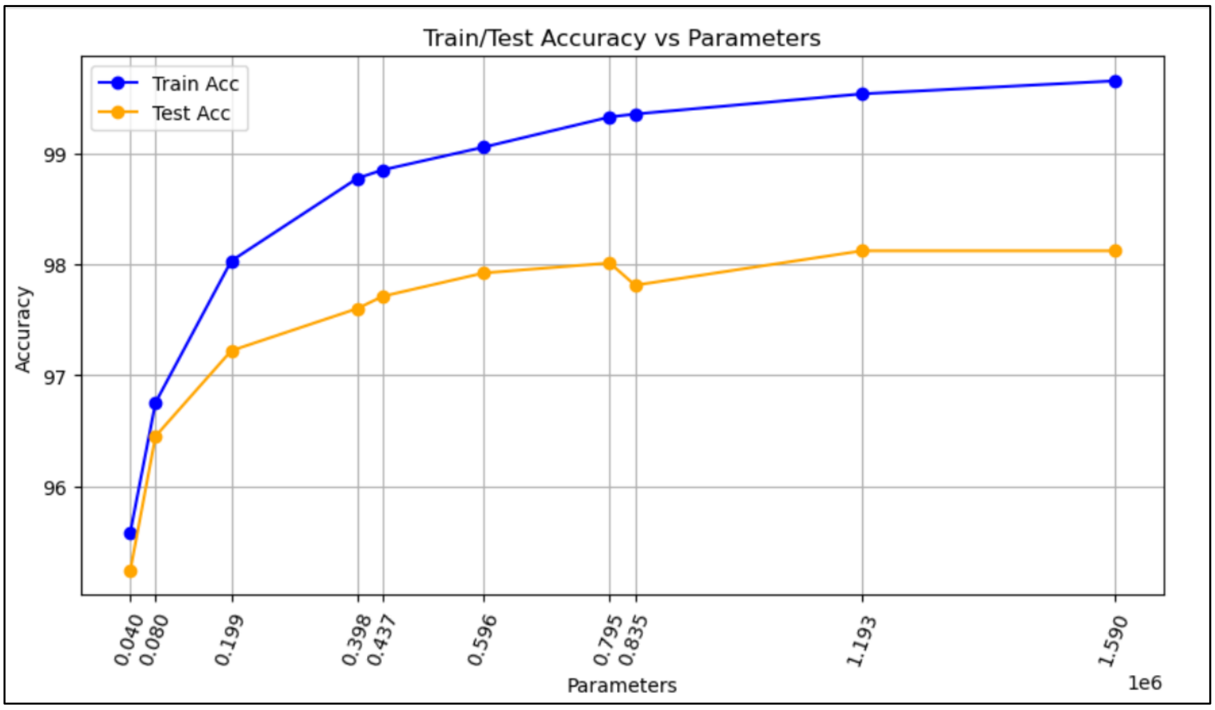
Fig. 3.1.1 Random Training vs Test Loss

Result:

3.2 Number of parameters V.S. Generalization

In this section we will train ten CNN with same architecture

Models Used:
layers : 2
Kernel : 4
Activation Function: ReLU
Learning Rate: 0.001



Result:
As evident from the graph the models with high no of parameters generally perform better until a certain point after that they reach a saturation point. It may lead to over fitting.