Homework 1 – PRAMOD KARKHANI – CPSC 8430 – DEEP LEARNING

HW 1-1 Simulate a Function

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

https://github.com/pramod-karkhani/CPSC-8430---Deep-Learning---HW1/blob/main/HW1_SIMULATE_FUNCTION.ipynb

Models: 3 models were defined. A weight decay parameter was added to regularize the models adding a penalty to the cost function of the neural network. This caused shrinking of weights during back-propagation

- **MODEL #1:**
  7 Dense Layers, number of parameters : 571 , LeakyRelu
  Loss Function: MSELoss
  Optimizer: RMSProp
  Learning Rate: 1e-3

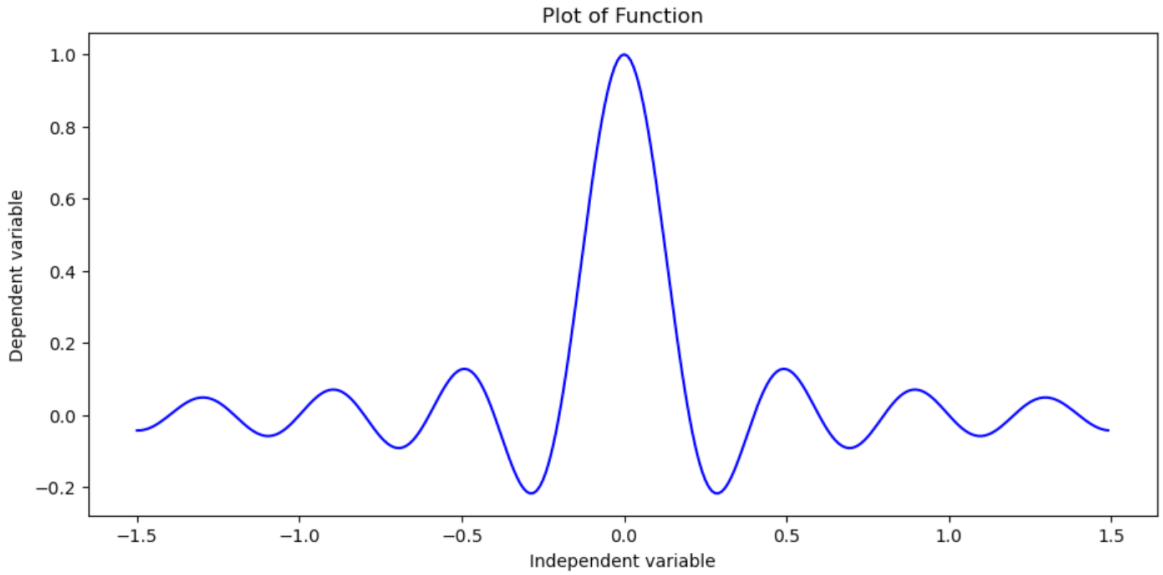
- **MODEL #2:**
  - 4 Dense Layers, number of parameters :546 , LeakyRelu
  - Loss Function: MSELoss
  - Optimizer: RMSProp
  - Learning Rate: 1e-3

- **MODEL #3**:
  - 1 Dense Layer, number of parameters :546
  - Loss Function: MSELoss
  - Optimizer: RMSProp
  - Learning Rate: 1e-3

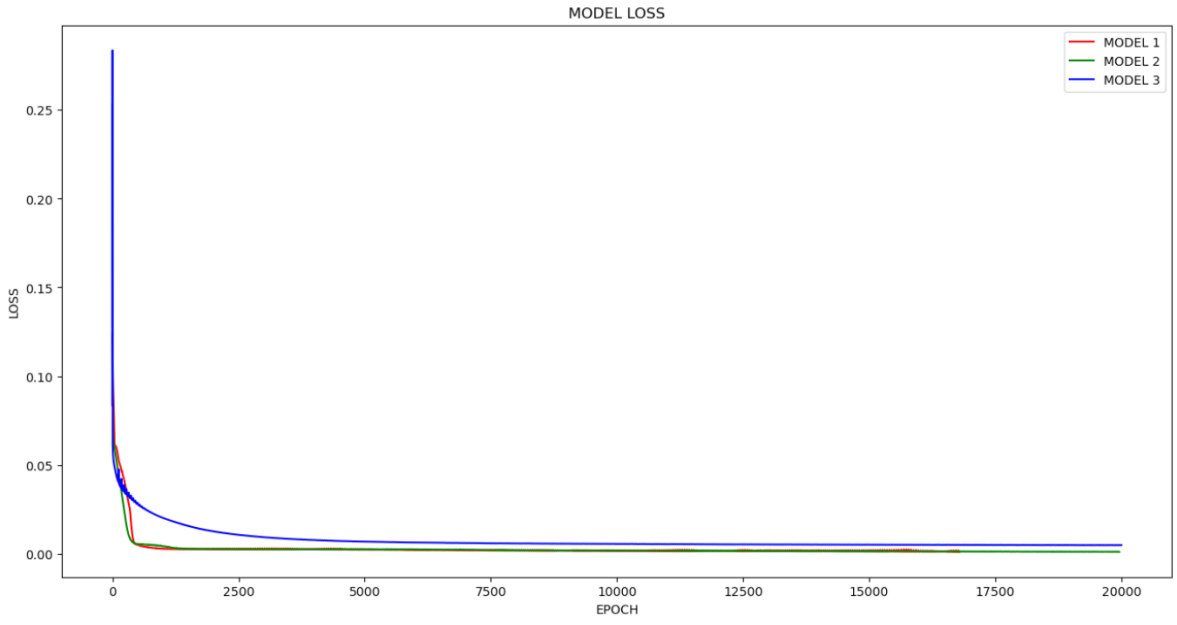
Function 1

Function: sin(5*pi*x) / 5*pi*x

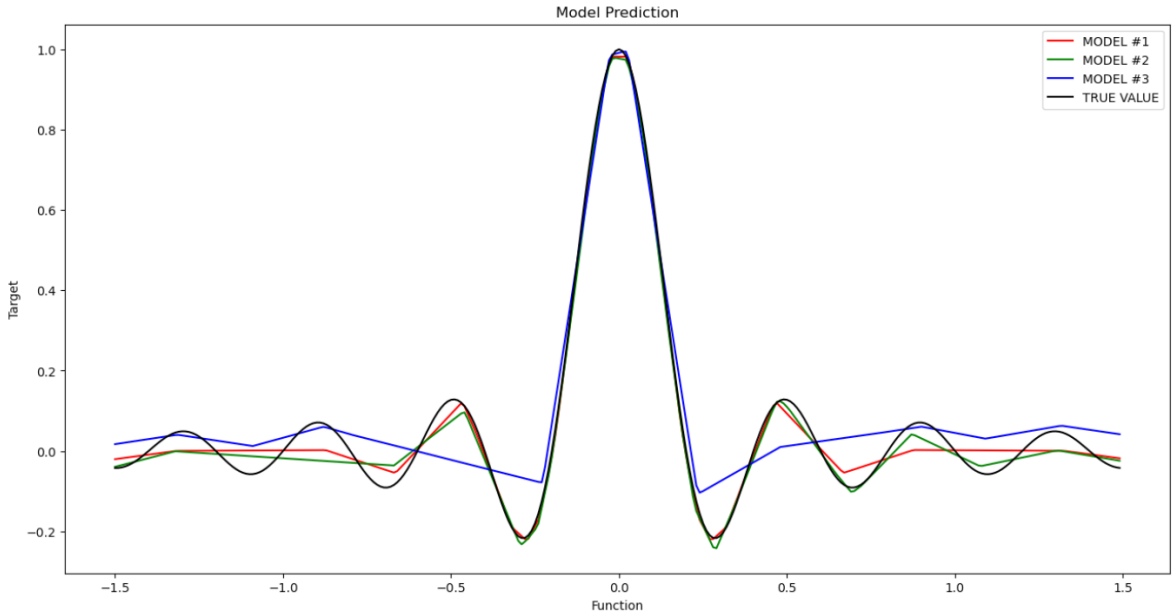
Below is the function plot for the same function:



Simulation of the function:
The models converged after reaching the epoch limit or when the model learns at a slow rate. Below is a graph showing the loss each of these models:



The following graph shows the actual vs prediction for the 3 models.



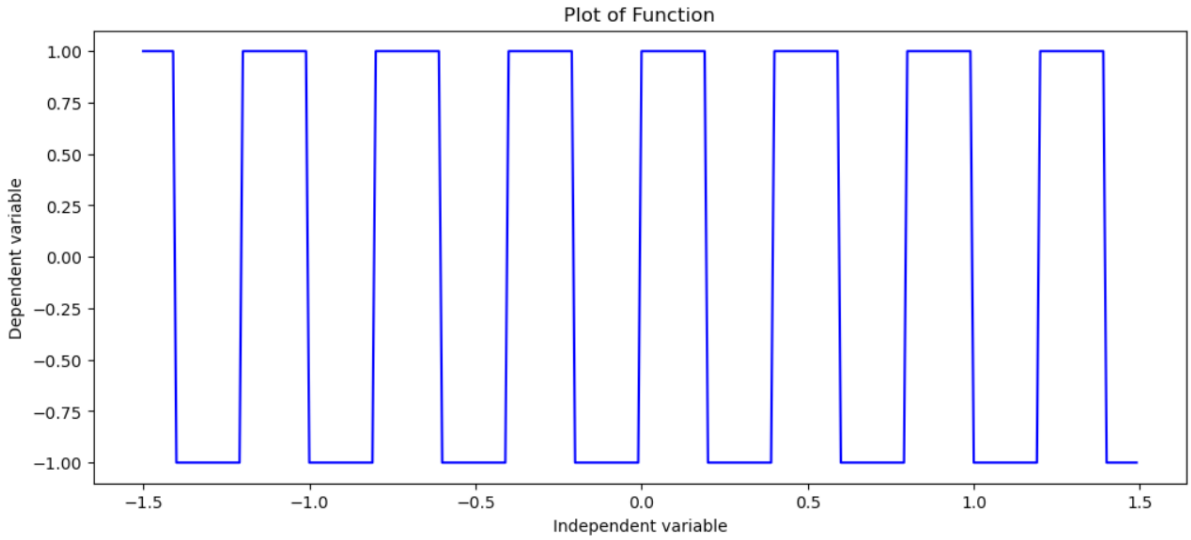
OBSERVATION

Model #1 and Model #2 converged relatively quickly. Model #3 reached epoch limit before it could converge. As observed in the graph, Models 1 & 2 had a lower loss value. Basically, the models with more layers learned faster and better.

FUNCTION 2

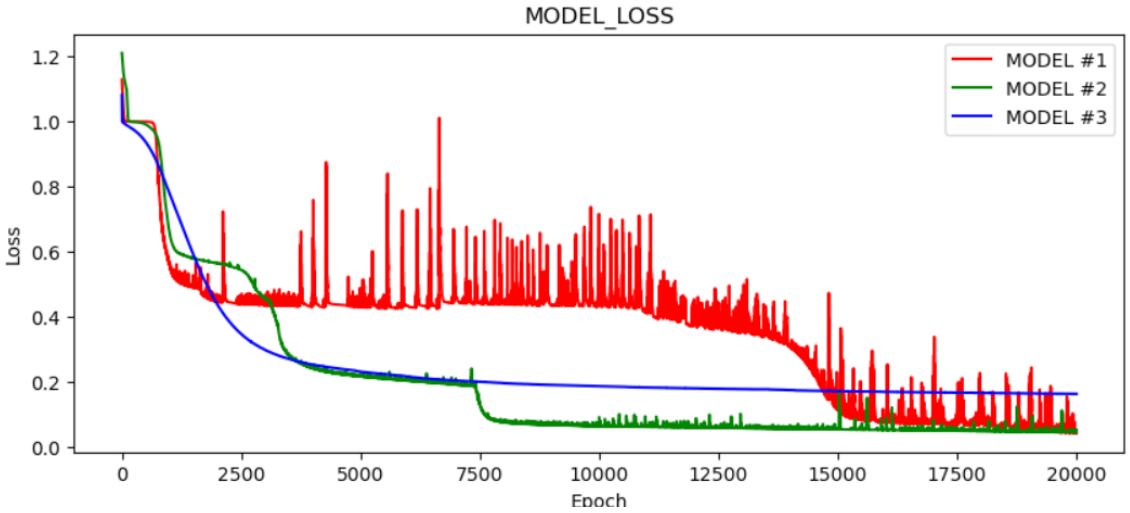
Function: sgn(sin(5*pi*x) / 5*pi*x)

function plot for the same:

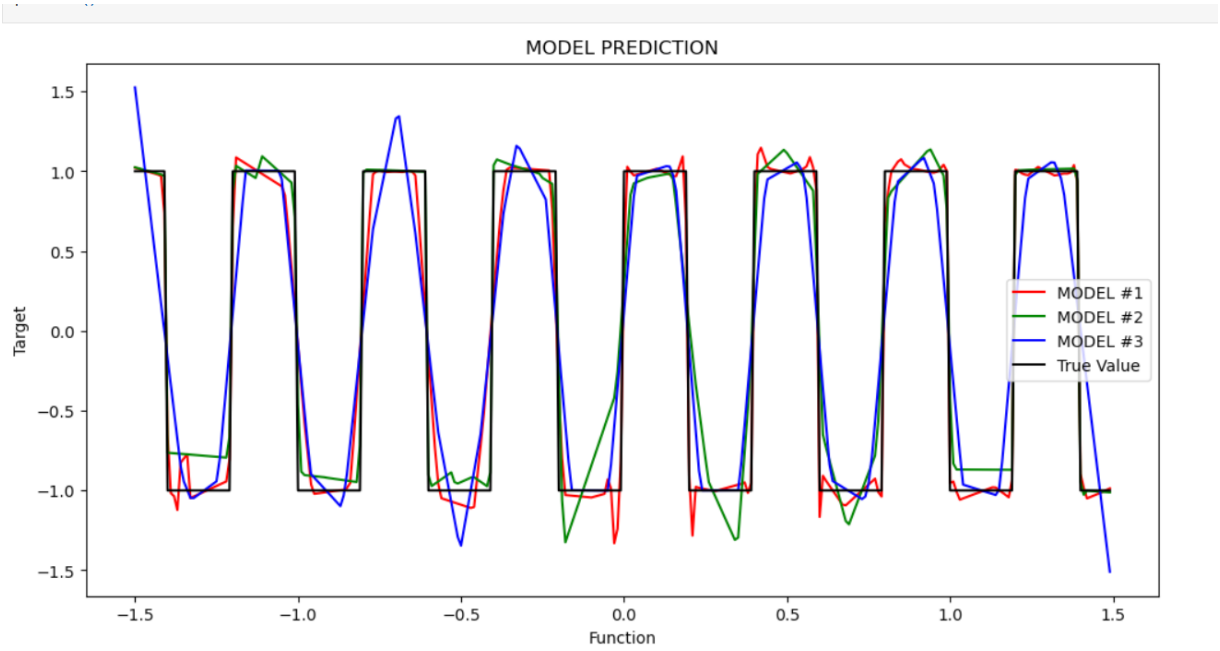


SIMULATION OF FUNCTION

The models converge after reaching maximum number of epochs. Below is a graph showing the loss each of these models:



The graph below shows actual vs predicted values for the models discussed:



OBSERVATION

All the models reached the maximum number of epochs before convergence. Model 1 had the lowest loss, slightly outperforming model 2. Model 3 failed to reach a low loss and failed to converge.

HW 1-1 Train on Actual Tasks

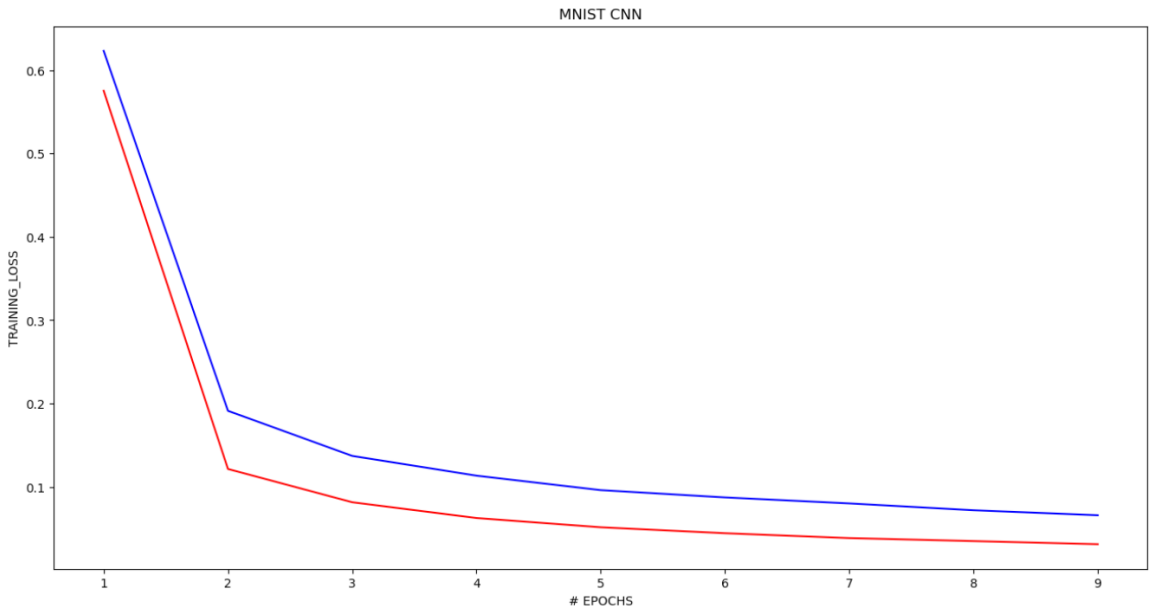
https://github.com/pramod-karkhani/CPSC-8430---Deep-Learning---HW1/blob/main/HW1_Train_On_Actual_Task_MNIST.ipynb

Model 1: CNN (LeNet)
• 2D convolution layer: apply a 2D max pooling
• 2D convolution layer: apply a 2D max pooling
• 2D Dense Layer: ReLu
• 2D Dense Layer: ReLu

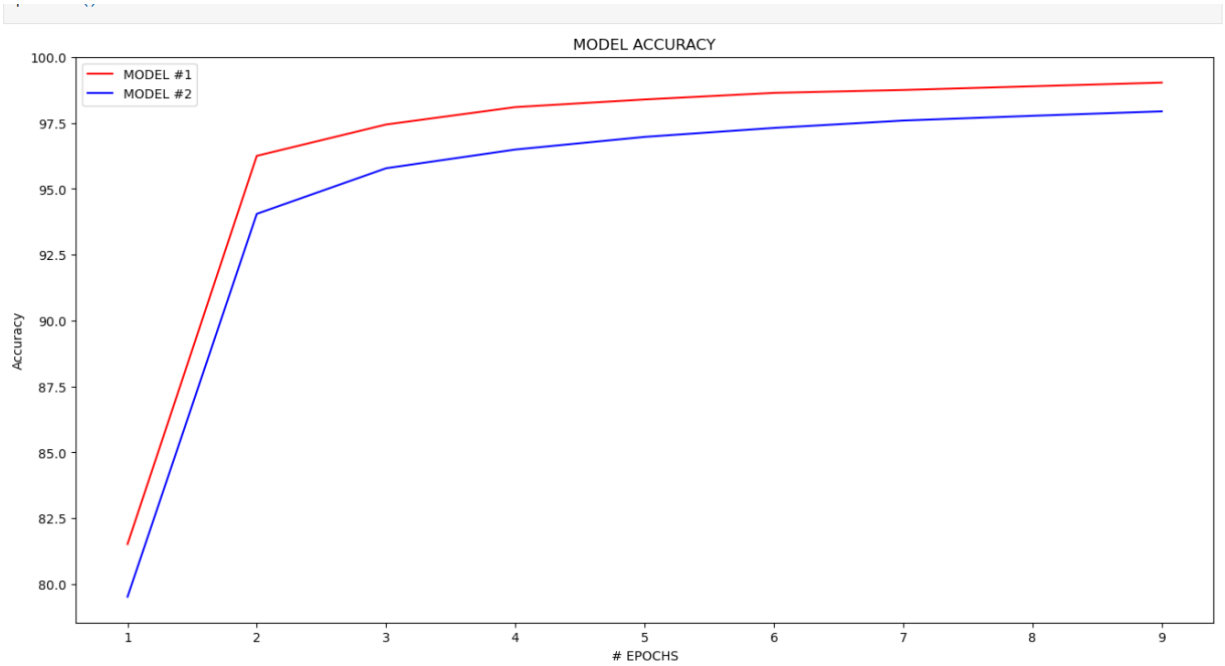
Model 2: CNN (custom)

• 2D convolution layer: applied 2-D max pooling
• 2D convolution layer: applied a 2-D max pooling
• 2D Dense Layer: ReLu - Dropout
• 2D Dense Layer: Log_softmax (Output)

Hyperparameters
• Learning Rate: 0.01
• Momentum: 0.5
• Optimizer: Stochastic Gradient Descent
• batch_size = 64
• epochs = 10
• Loss: Cross Entropy



training loss for Model 1 and Model 2



training accuracy for Model 1 and Model 2

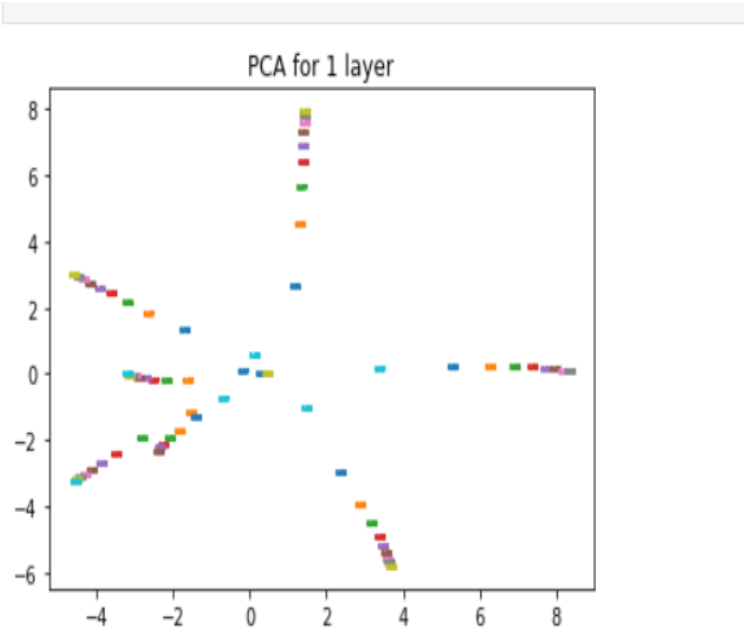
OBSERVATION

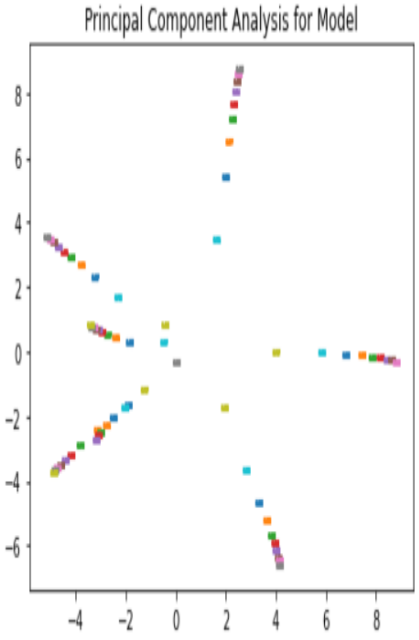
Model 1 had a lower loss and performed better than Model 2. The structure of Model 1 was an optimized CNN model based on an existing LeNet structure. It outperformed the custom CNN model #2.

HW 1-2 Visualize the optimization process

https://github.com/pramod-karkhani/CPSC-8430---Deep-Learning---HW1/blob/main/HW1_Principal_Component.ipynb

The below model is trained on the MNIST dataset.

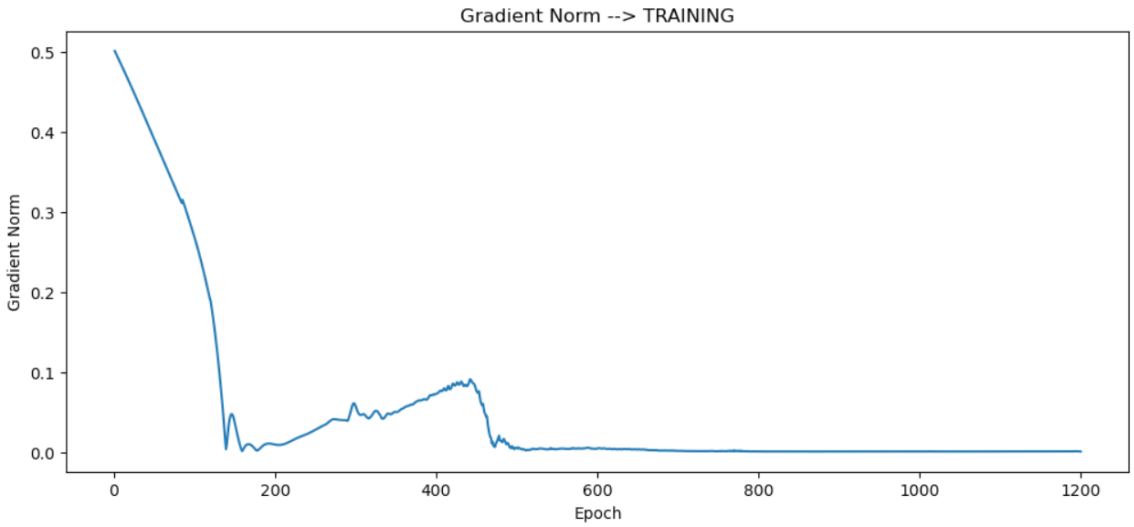




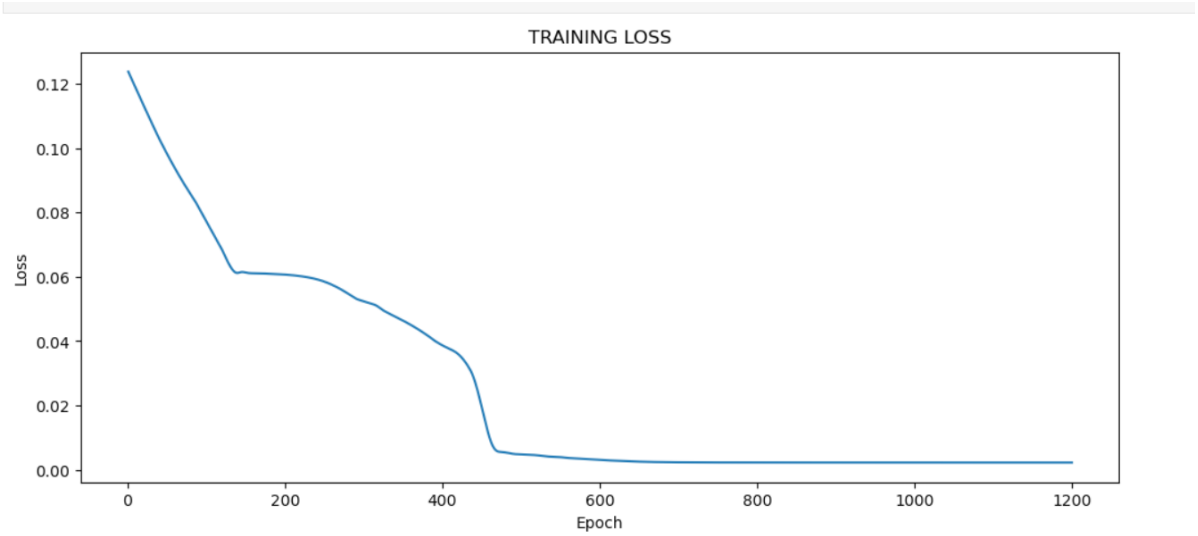
HW 1-2 Observe gradient norm during training

https://github.com/pramod-karkhani/CPSC-8430---Deep-Learning---HW1/blob/main/HW1_gradnorm.ipynb

The function sin(5*pi*x) / 5*pi*x has been reused to calculate the gradient norm and the loss.



Below is a graph for loss across the epochs

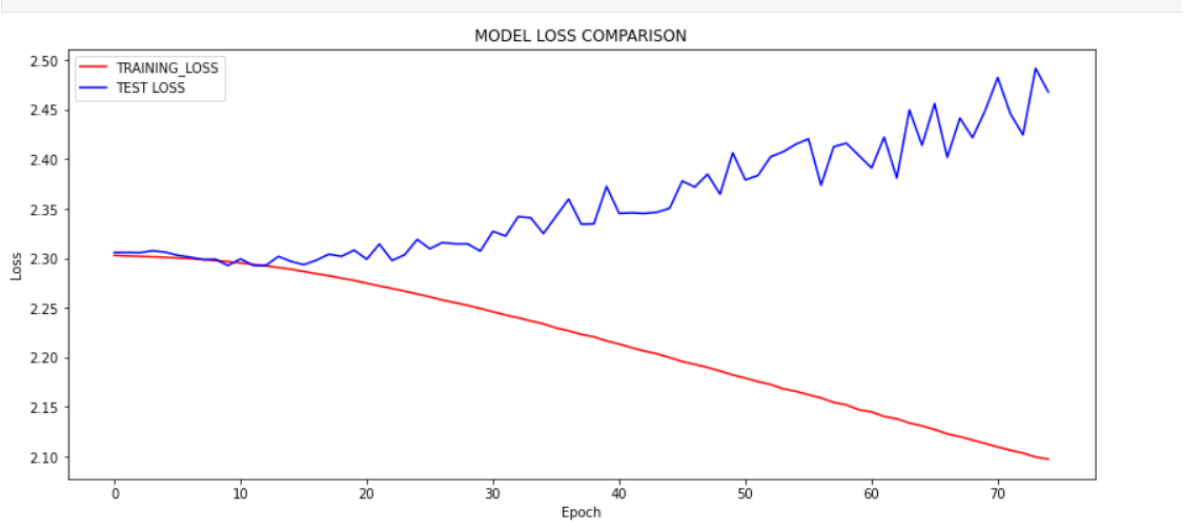


HW 1-3 Can network fit random labels?

https://github.com/pramod-karkhani/CPSC-8430---Deep-Learning---HW1/blob/main/HW1_Labelfit.ipynb

The below model is trained on the MNIST dataset.

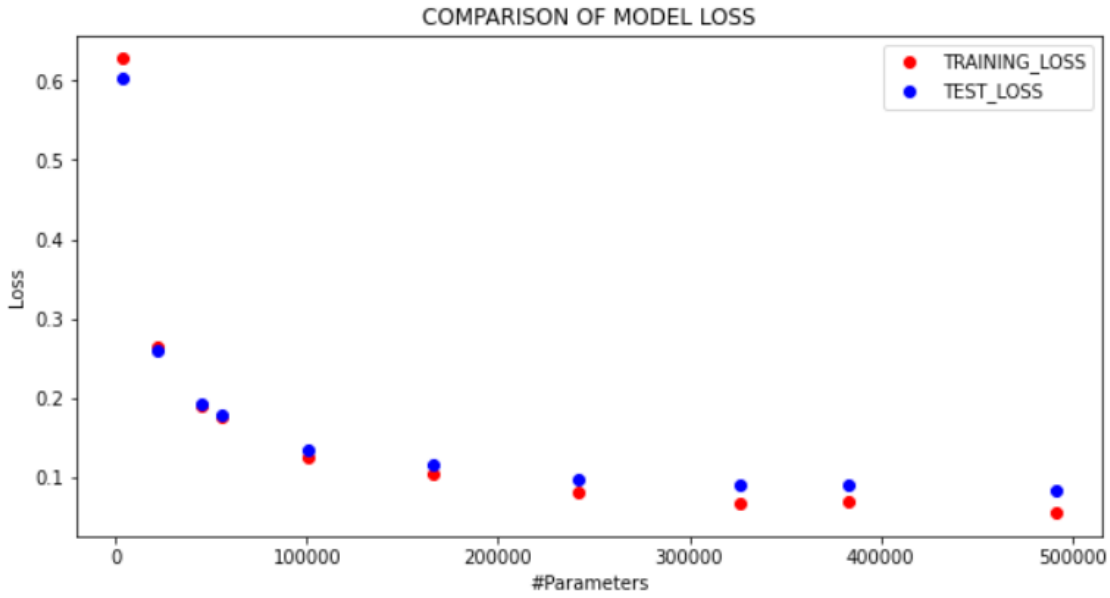
Model 1: CNN (LeNet)
• 2D convolution layer: apply a 2D max pooling
• 2D convolution layer: apply a 2D max pooling



The above model is trained on random models. The training process is slow as the model must learn
on random labels, it tries to memorize the labels as we move through epochs and reduce the loss.
The test loss continues to increase as the epochs increase with a gradual decrease in the train loss.

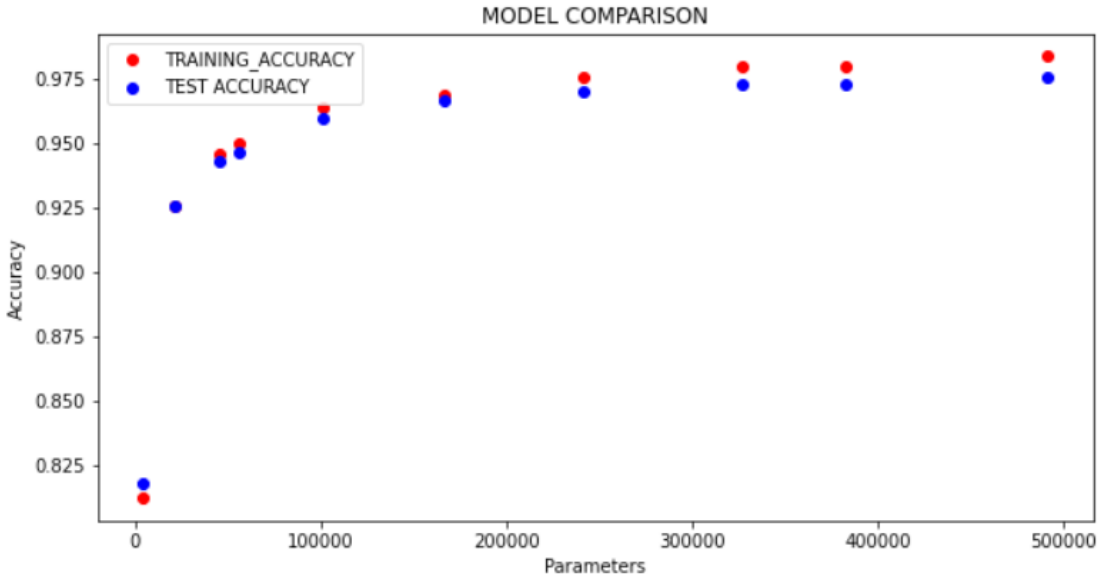
HW 1-3 Number of parameters vs Generalization

https://github.com/pramod-karkhani/CPSC-8430---Deep-Learning---HW1/blob/main/HW1_PARAMETER_COMPARE.ipynb



Graph for Accuracy comparison for the various models

```
plt.show()
```

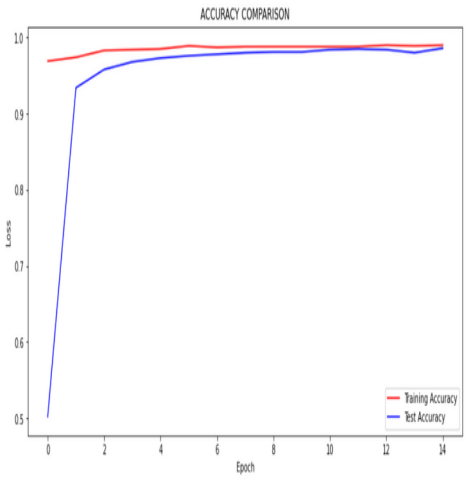


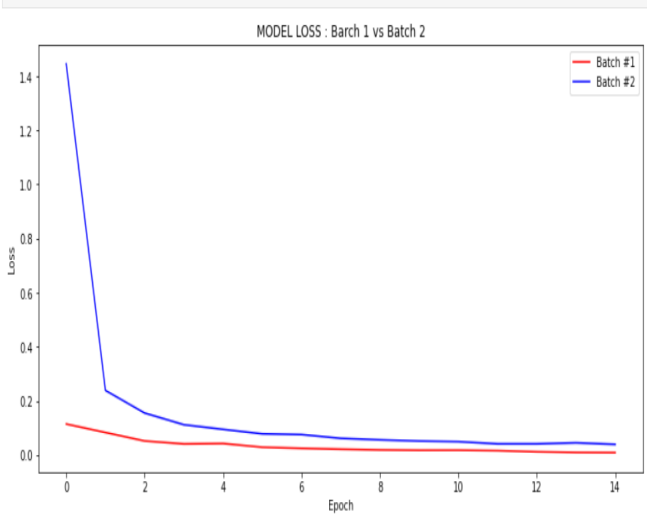
With increase in number of parameters the difference between the train and test loss/accuracy increases. The test loss starts plateauing much before the training loss or accuracy. This is due to overfitting (because there are a greater number of parameters for the model to train.)

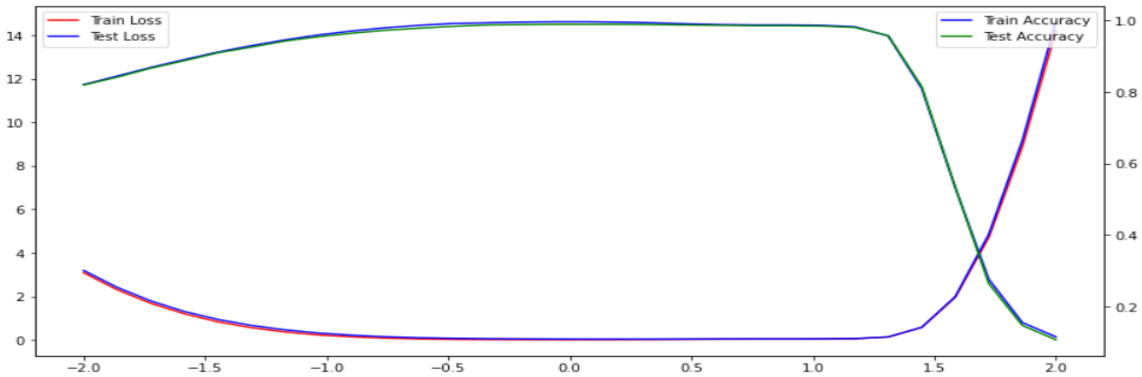
HW 1-3 Flatness vs Generalization

https://github.com/pramod-karkhani/CPSC-8430---Deep-Learning---HW1/blob/main/HW1_Interpolation.ipynb

The below model is trained on the MNIST dataset.

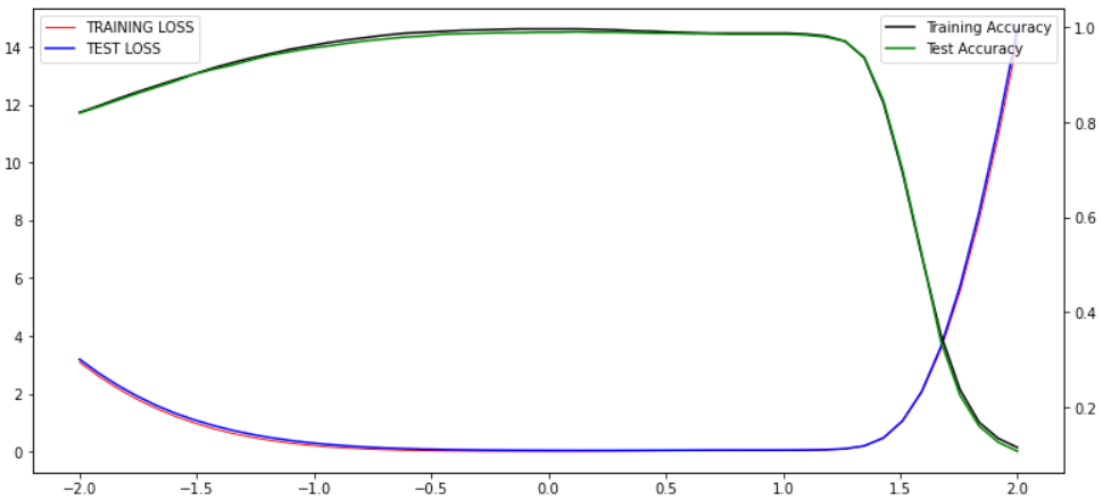






S

```
Out[30]: <matplotlib.legend.Legend at 0x7f87e6f10880>
```



Learning Rate 1e-3
Observed Result
We can see the accuracy of test and train drop to around 1.5 for the models with different learning rates. The accuracy begins to increase for the graphs at alpha value of 1.5.