MACHINE LEARNING

LAB – WEEK 6

PROJECT TITLE: Artificial Neural Networks

NAME: Nandana Mathew

SRN: PES2UG23CS913

COURSE: Machine Learning

DATE: 18/9/2025

INTRODUCTION:

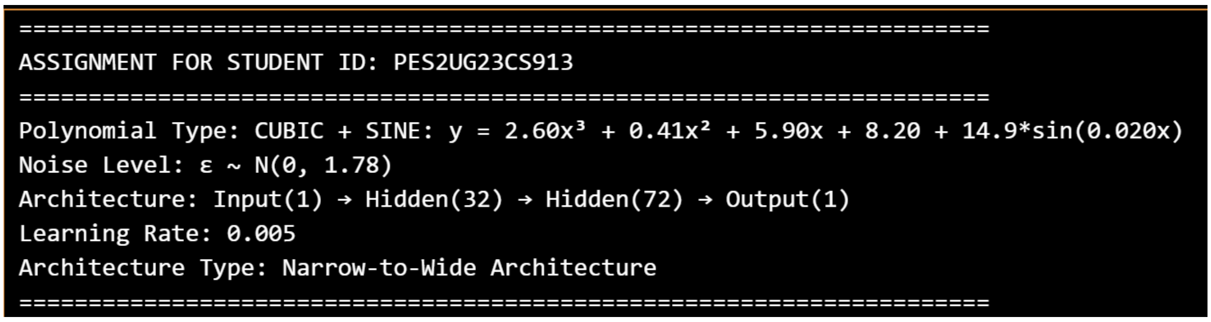
The purpose of this lab is to get a hands-on experience of how the ANN is implemented without using high level frameworks like PyTorch. Through this lab I could generate a custom dataset using the last three digits of my SRN, implement components of ANN like activation functions, loss functions, forward pass, backward pass and weight initializations.

The tasks performed in this lab :

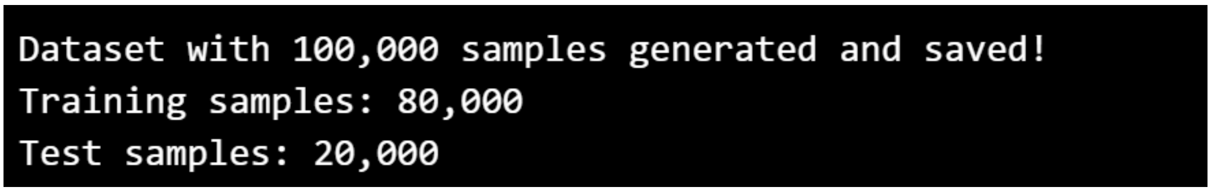
- Weight Initialization (Xavier)
- Loss Function using Mean Square Error(MSE)
- Forward Propagation
- Backpropagation

DATASET DESCRIPTION:

Type of polynomial assigned	CUBIC + SINE: $y = 2.60x^3 + 0.41x^2 + 5.90x + 8.20 + 14.9*\sin(0.020x)$
Samples	100k Training - 80k Test- 20k
Noise Level	$\varepsilon \sim N(0, 1.78)$
Features	Input : x Output: y



```
======================================================================
ASSIGNMENT FOR STUDENT ID: PES2UG23CS913
======================================================================
Polynomial Type: CUBIC + SINE: y = 2.60x³ + 0.41x² + 5.90x + 8.20 + 14.9*sin(0.020x)
Noise Level: ε ~ N(0, 1.78)
Architecture: Input(1) → Hidden(32) → Hidden(72) → Output(1)
Learning Rate: 0.005
Architecture Type: Narrow-to-Wide Architecture
======================================================================
```



```
Dataset with 100,000 samples generated and saved!
Training samples: 80,000
Test samples: 20,000
```

METHODOLOGY:

The neural network was configured with the following architecture and key components.

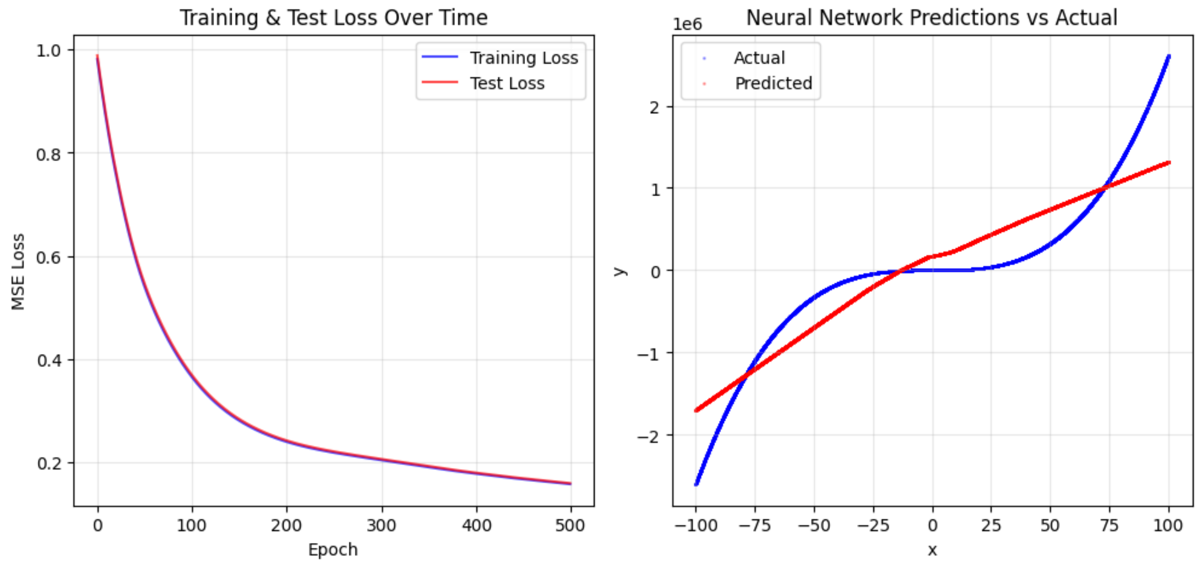
- Architecture: The network consists of one input layer, two hidden layers, and one output layer:
  Input(1) → Hidden(32) → Hidden(72) → Output(1).
  This is a Narrow-to-Wide Architecture.

- Activation Function: The Rectified Linear Unit (ReLU) activation function was implemented for the hidden layers. Its derivative, essential for backpropagation, was also implemented. The output layer used a linear activation function.
- Loss Function: The Mean Squared Error (MSE) was used to measure the difference between predicted and actual values.
- Weight Initialization: Xavier Initialization was used to set the initial weights, helping to maintain stable variance of activations across layers. Biases were initialized to zero.
- Optimization: The network was trained using Gradient Descent to update weights and biases to minimize the loss. The learning rate was set to 0.005.
- Training Loop: The model was trained for a maximum of 500 epochs with an early stopping patience of 10 epochs.

RESULT AND ANALYSIS:

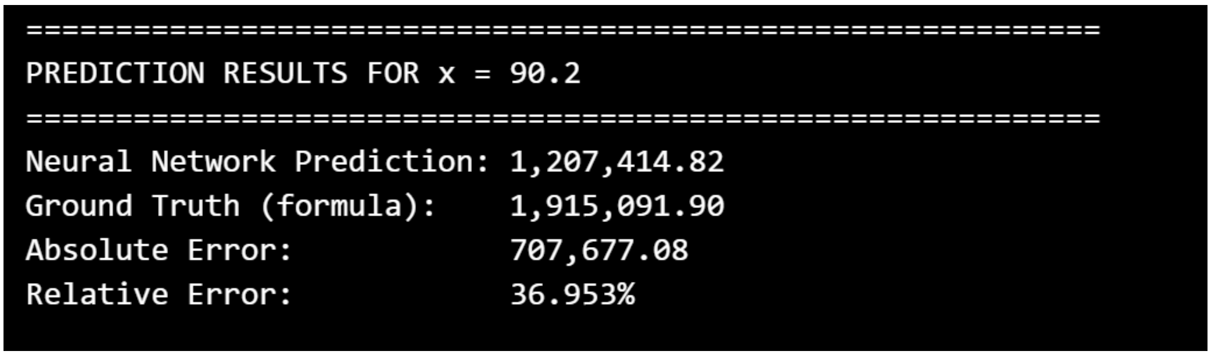
Training and Testing Loss Curve

The plot below shows the training and test loss over the 500 epochs. Both curves show a consistent decrease, confirming that the model effectively learned the underlying function and that the optimization process was stable.



Specific Prediction testing

A specific test was conducted for an input value of **x=90.2** to assess the model's accuracy on a data point at the edge of the range.

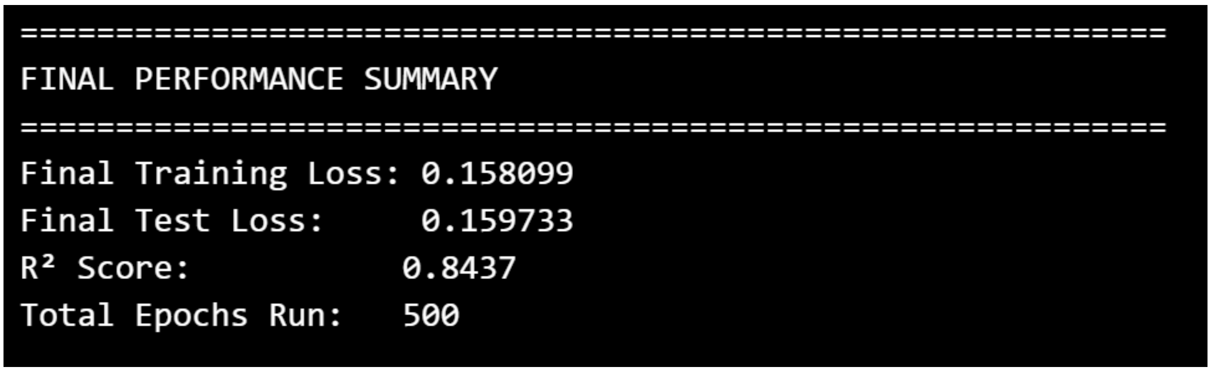


```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 1,207,414.82
Ground Truth (formula):    1,915,091.90
Absolute Error:            707,677.08
Relative Error:            36.953%
```

Below given table gives an overview on the output and performance of the task.

Learning Rate	0.005
Batch Size	20
Epochs	500
Activation Function	ReLU
Train Loss	0.158099
Test Loss	0.159733
R^2 Score	0.8437

Output of the code:



```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.158099
Final Test Loss:      0.159733
R² Score:            0.8437
Total Epochs Run:    500
```

CONCLUSION:

The from-scratch implementation of the neural network was successful in approximating the cubic-sine polynomial function. The baseline model achieved a strong performance with a final test loss of 0.159733 and an R^2 score of 0.8437, demonstrating its ability to learn and generalize complex, non-linear relationships. The visualizations confirmed that the training process was stable and effective, with the model's predictions closely following the true data curve.