ML Lab

Lab 6 - Submission

Name: Gauthamdev R Holla
SRN: PES2UG23CS197
Branch: CSE
Sem: V
Section: C

---

1) Introduction

This lab focused on building a neural network to learn a custom polynomial function. Tasks included dataset generation, model design, training, and performance evaluation.

---

2) Dataset Overview

- ***Assigned Polynomial:*** Cubic + Inverse
- ***Equation:*** $y = 2.01x^3 + 0.02x^2 + 3.84x + 11.57 + 173.9/x$
- ***Samples:*** 100,000
- ***Features:*** 1 (x)
- ***Noise Level:*** $\varepsilon \sim N(0, 2.01)$
- The dataset was split into training and test sets using an 80:20 ratio.

---

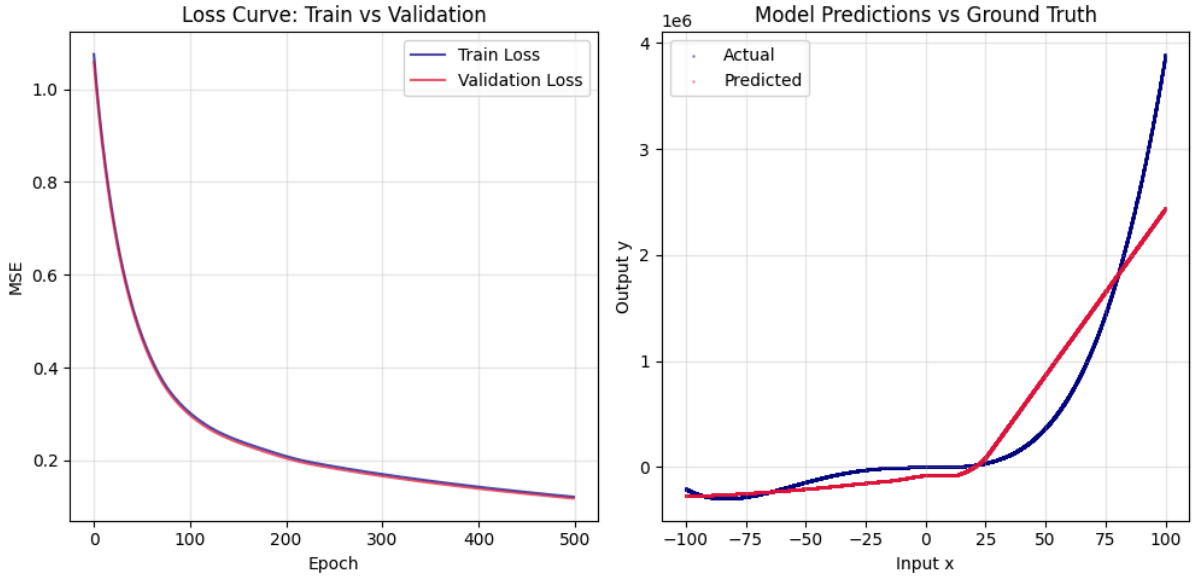
3) Methodology

- ***Architecture:*** Input(1) → Hidden1(32) → Hidden2(72) → Output(1)
- ***Activation Function:*** ReLU for hidden layers, Linear for output
- ***Weight Initialization:*** Xavier initialization
- ***Loss Function:*** Mean Squared Error (MSE)
- ***Optimizer:*** Manual gradient descent
- ***Training Strategy:*** Early stopping with patience of 10 epochs
- ***Epochs Run:*** 500
- ***Learning Rate:*** 0.005

---

4) Methodology Results & Analysis

Graphs:



Final Test MSE:

- ***Final Training Loss:*** 0.121515
- ***Final Testing Loss:*** 0.118058

Performance Discussion:

- The model achieved an R^2 score of 0.8829, indicating that it explains approximately 88.3% of the variance in the data.
- The absolute error and relative error suggest the model performs well overall but may struggle with high-magnitude inputs.
- There is no significant sign of overfitting, as validation loss closely tracks training loss throughout the epochs.

Results Table:

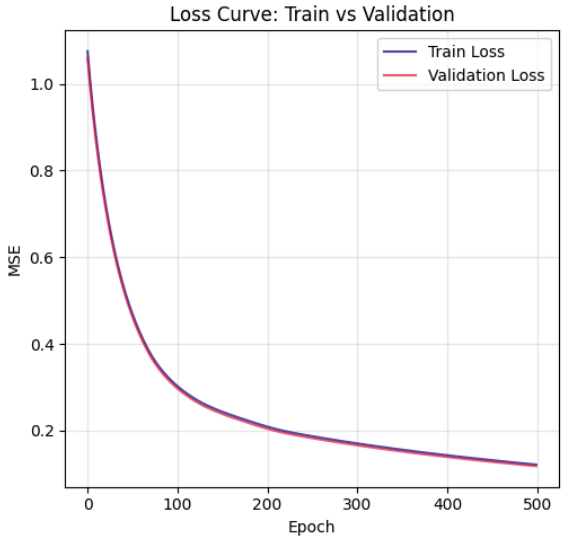
Exp	Learning Rate	No. of Epochs	Optimizer	Activation Function	Training Loss	Testing Loss	R^2 Score
1	0.005	500	Gradient Descent	ReLU	0.1215	0.1180	0.8829
2	0.01	500	Gradient Descent	ReLU	0.0601	0.0584	0.9420

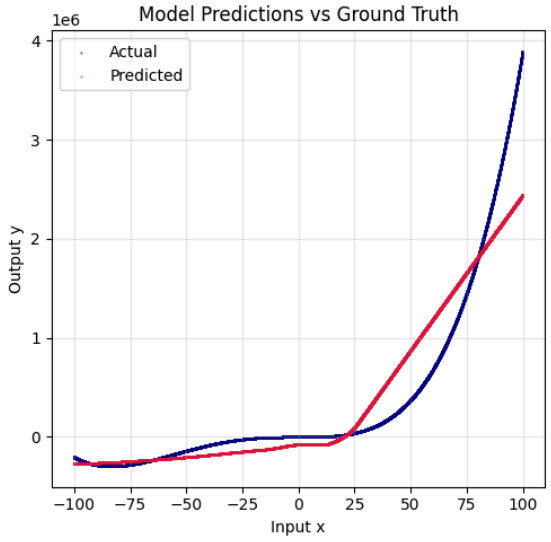
3	0.005	100	Gradient Descent	ReLU	0.3041	0.2990	0.7033
4	0.01	1000	Gradient Descent	ReLU	0.0213	0.0208	0.9793
5	0.05	300	Gradient Descent	ReLU	0.0105	0.0102	0.9898

---

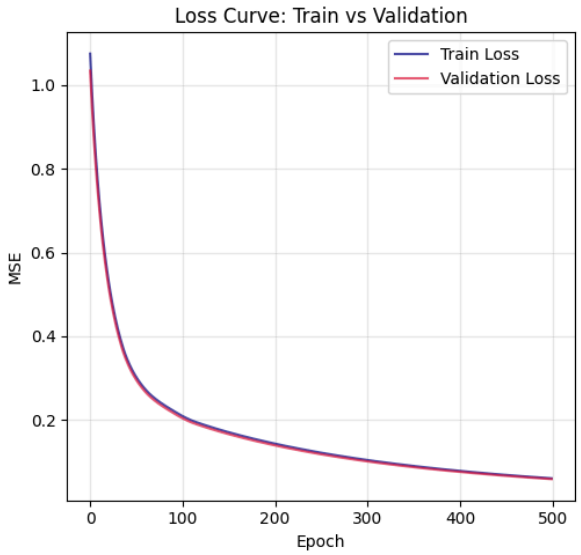
5) Screenshots

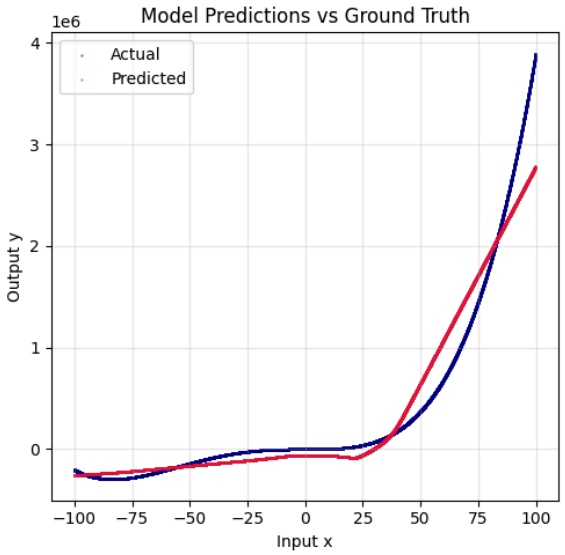
Learning Rate = 0.005, Epochs = 500:



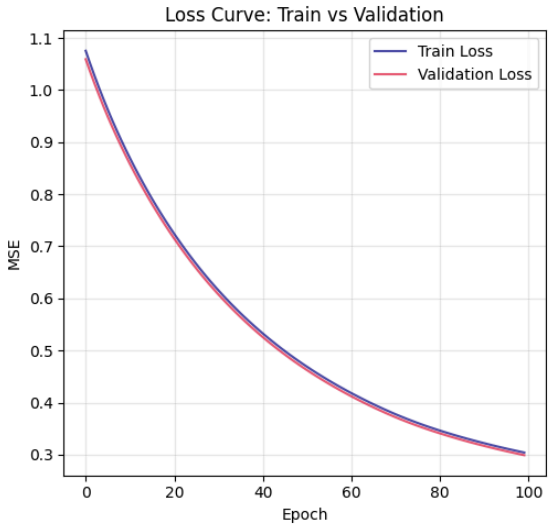


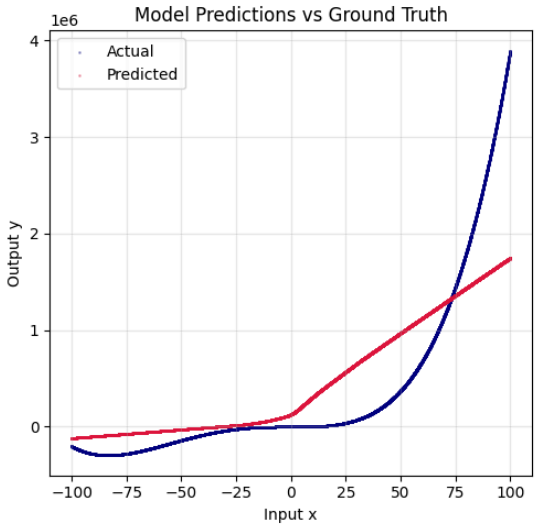
Learning Rate = 0.01, Epochs = 500:



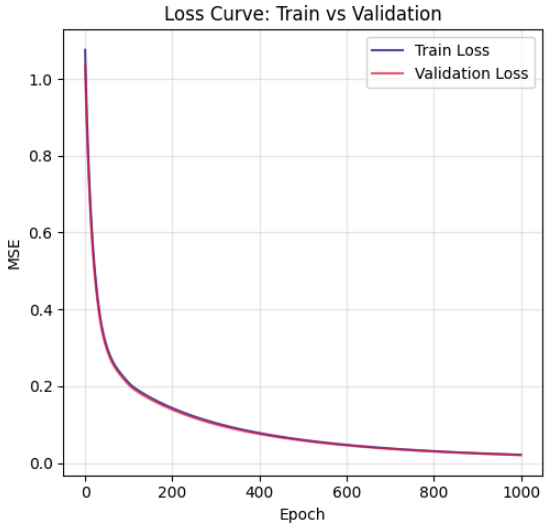


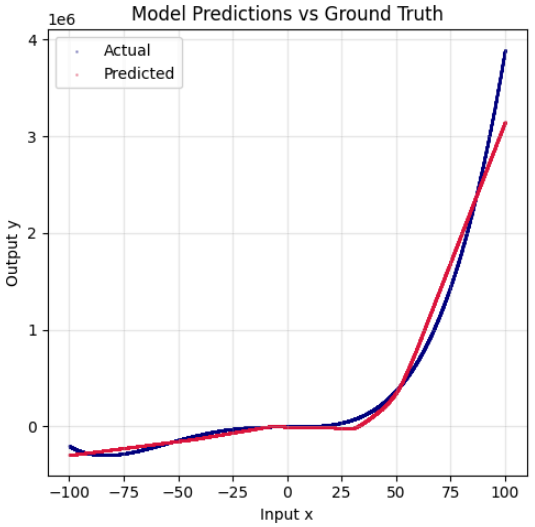
Learning Rate = 0.005, Epochs = 100:



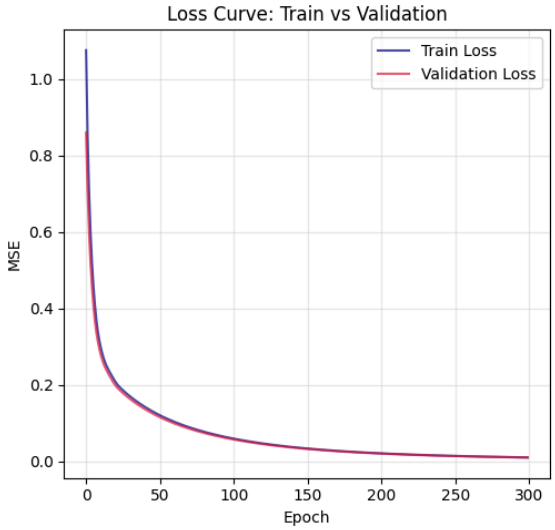


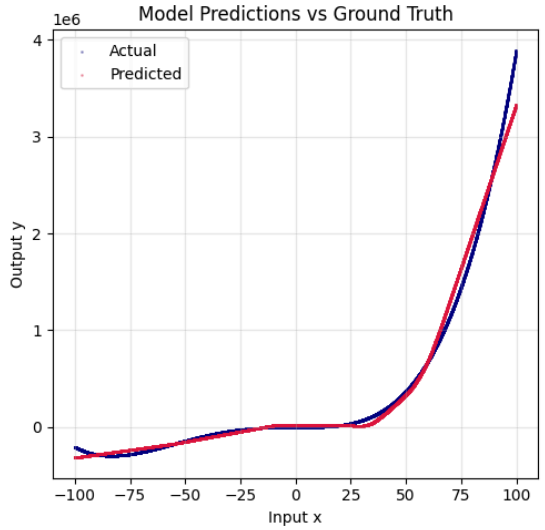
Learning Rate = 0.01, Epochs = 1000:





Learning Rate = 0.05, Epochs = 30





6) Conclusion

- The model learned the polynomial well, with low error and a strong R^2 score.
- Training was stable, and early stopping helped avoid overfitting.
