UE23CS352A: MACHINE LEARNING

Week 6: Artificial Neural Networks

Name: Devraj Naik

SRN: PES2UG23CS167

Course: Machine Learning

Date: 16th September 2025

1. Introduction

Purpose of the Lab

The primary purpose of this lab is to gain hands-on experience implementing Artificial Neural Networks (ANNs) from scratch, without using high-level libraries like TensorFlow or PyTorch. By building each component manually, we develop a deeper understanding of how neural networks function internally and how gradient-based optimization enables learning.

Tasks Performed

- **Dataset Generation**: A synthetic dataset was created based on the last three digits of the SRN.
- **Neural Network Implementation**: Implemented activation functions (ReLU and derivative), Mean Squared Error (MSE) loss, forward propagation, backpropagation, and gradient descent.
- **Training & Evaluation**: Trained the network to approximate the target polynomial curve, tracked training loss over epochs, and evaluated model performance on the test set.
- **Visualization**: Generated plots.
- **Hyperparameter Experiments**: tuned by varying learning rate, activation function and number of epochs to analyse their impact on performance.

2. Dataset Description:

Polynomial: QUARTIC: $y = 0.0188x^4 + 1.62x^3 + -0.48x^2 + 2.82x + 11.87$.

Features:

- Input: x (continuous).
- Output: y, generated from the polynomial equation.
- Preprocessing using Standard Scaler.

Number of Samples: 100,000 total samples.

- 80,000 for training (80%).
- 20,000 for testing (20%).

3. Methodology:

Network Architecture: Input : hidden layer 1 : hidden layer 2 : Output layer.

Forward Propagation: Each layer computes a weighted sum z = XW + b. various Activation function is experimented.

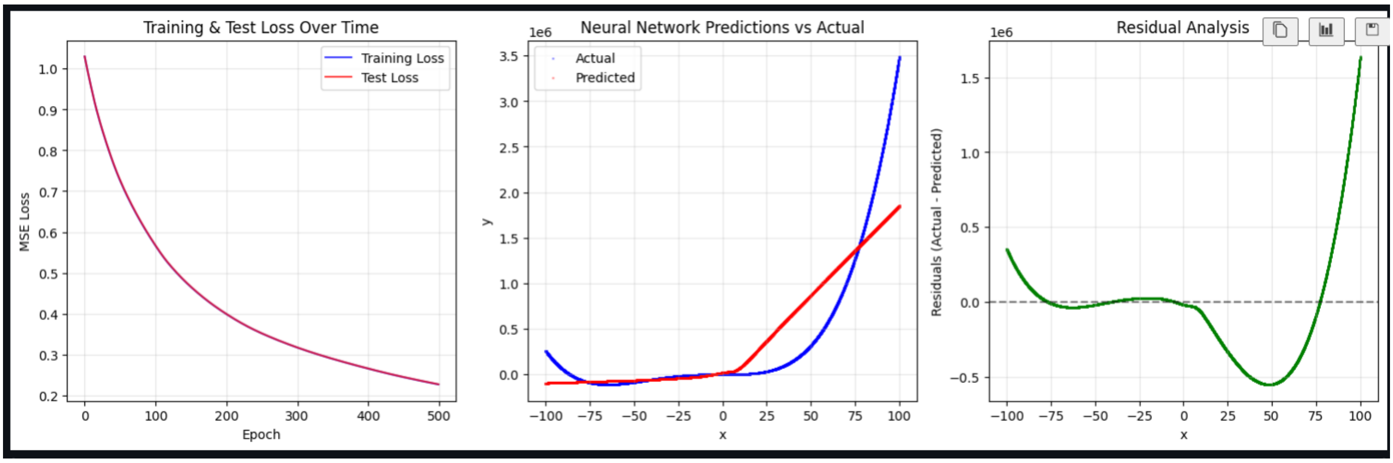
Loss Function: Mean Squared Error (MSE) is used.

Backpropagation: loss computed and parameters are updated.

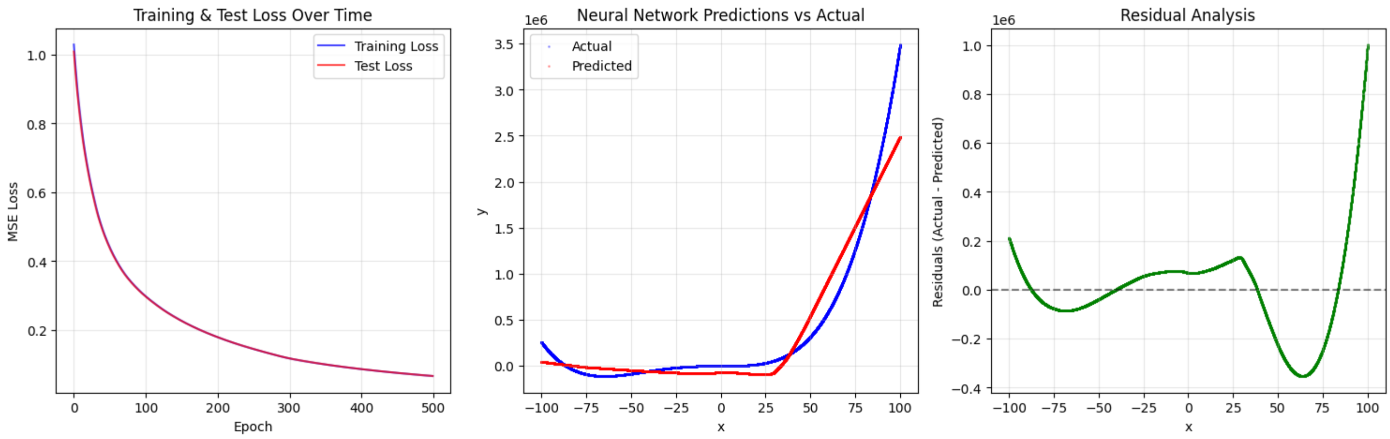
Optimization: Batch Gradient Descent is used.

Training: Multiple epochs with full training dataset and Hyperparameter variations tested to analyse their effect.

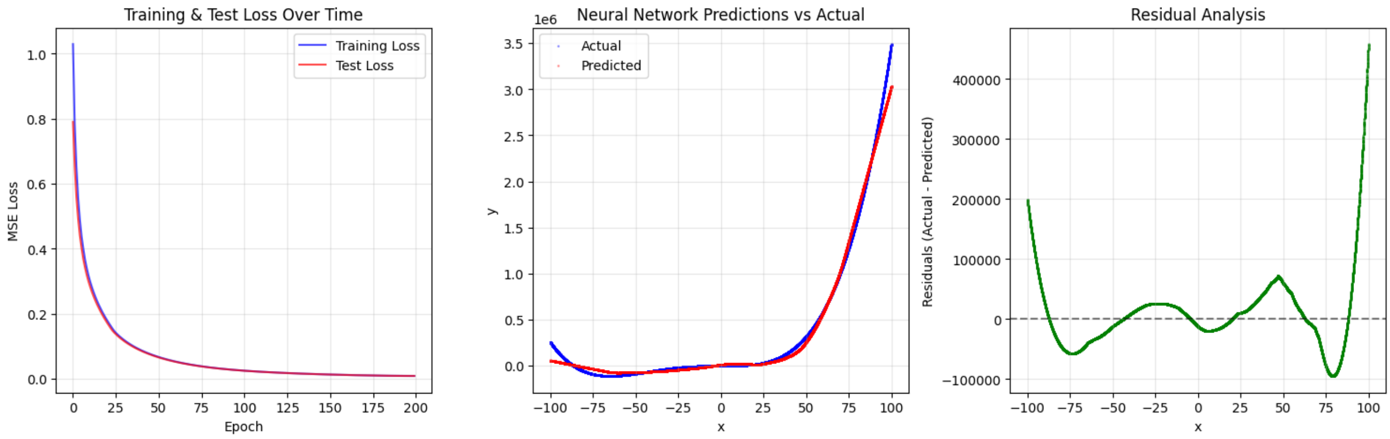
Experiment 1:
Learning rate: 0.003
No of epochs: 500
Activation function: ReLU
Final training Loss: 0.22701
Final test loss: 0.22745
R^2 Score: 0.7745
These are default value modelled: model shows good generalization.



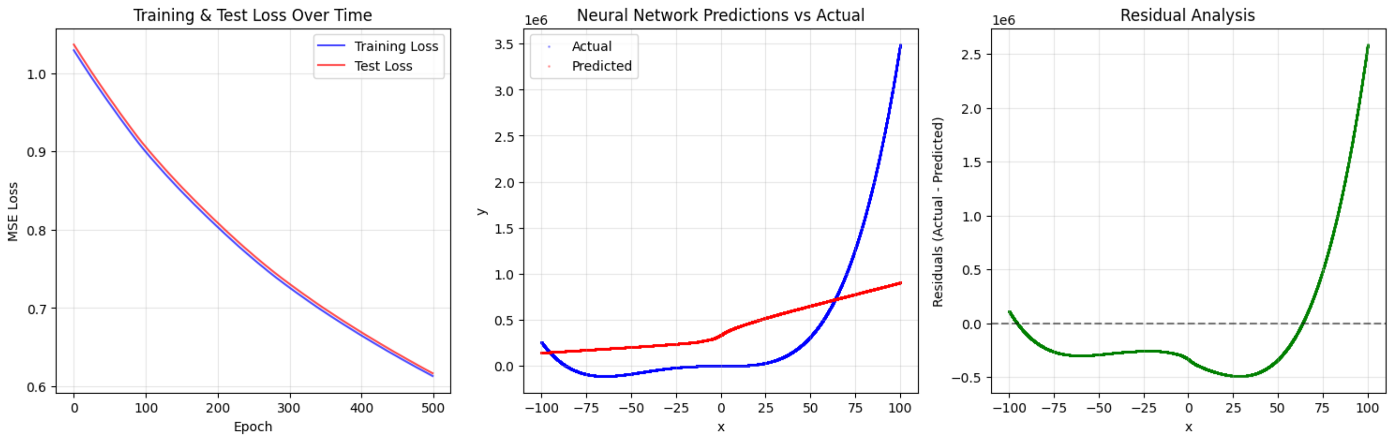
Experiment 2:
Learning rate: 0.01
Activation function: ReLU
Final Training Loss: 0.066635
Final Test Loss: 0.066766
R^2 Score: 0.9338
Total Epochs Run: 500
When learning rate is decreased to 0.01, we got better learning: model gives very solid result.



Experiment 3:
When lr=0.1, but with lower epochs, we got better results:
Final Training Loss: 0.008341
Final Test Loss: 0.008380
R² Score: 0.9917
Total Epochs Run: 200
Learning Rate: 0.1
Activation Function: ReLU



Experiment 4:
Here we kept learning rate low: hence underfit and error;
Final Training Loss: 0.612803
Final Test Loss: 0.616317
R² Score: 0.3890
Total Epochs Run: 500
Learning Rate: 0.0005
Activation Function: ReLU



Experiment 5:

Using Leaky ReLU; we got a slight better result than the normal ReLU;

Final Training Loss: 0.066442

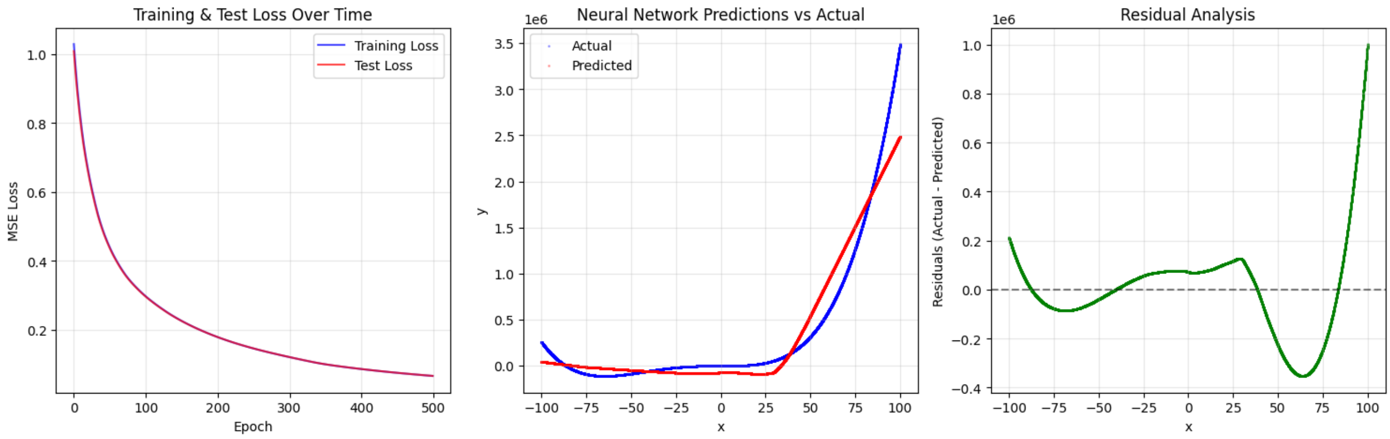
Final Test Loss: 0.066574

R^2 Score: 0.9340

Total Epochs Run: 500

Learning Rate: 0.01

Activation Function: Leaky ReLU



Result Table:

Experiment	Learning Rate	No. of epochs	Activation function	Final training loss	Final Test loss	R^2 Score
1	0.003	500	ReLU	0.22701	0.22745	0.7745
2	0.01	500	ReLU	0.066635	0.066766	0.9338
3	0.1	200	ReLU	0.008341	0.00838	0.9917
4	0.0005	500	ReLU	0.612803	0.616317	0.389
5	0.01	500	Leaky ReLU	0.066442	0.066574	0.934

Conclusion:

Learning rate–epoch trade-off was critical: **High learning rate → weights change faster → fewer epochs are usually needed** before convergence and **vice-versa**. Leaky ReLU improved learning. Early stopping (patience) prevented overfitting. The final model achieved good alignment between predictions and true targets, demonstrating that a simple ANN can approximate complex non-linear functions effectively