Week 6: Artificial Neural Networks

Mrunal Manjunath Kudtarkar
PES2UG23CS354
Machine Learning
16-09-2025

1. Introduction

Purpose of the Lab

The objective of this lab is to gain hands-on experience in building a neural network from scratch, without using high-level libraries like TensorFlow or PyTorch. By manually implementing each component, the lab deepens understanding of how neural networks learn complex functions and approximate non-linear relationships.

Tasks Performed

- Generated a synthetic dataset based on a polynomial function tied to the last three digits of the SRN
- Standardized input and output data using StandardScaler
- Implemented core neural network components:
  - Activation functions (e.g., ReLU, Sigmoid)
  - Loss function (Mean Squared Error)
  - Forward propagation
  - Backpropagation using the chain rule
  - Weight updates via gradient descent
- Trained the network to approximate the polynomial curve
- Evaluated model performance using training/test loss and R^2 score
- Visualized learning behavior and discussed overfitting/underfitting

2. Dataset Description

- **Type of Polynomial Assigned**: *(e.g., Cubic + Sine term)*
- **Number of Samples**: 100,000
- **Train-Test Split**: 80% training (80,000 samples), 20% testing (20,000 samples)
- **Number of Features**: 1 (univariate input)
- **Noise Level**: Moderate Gaussian noise added to simulate real-world variability
- **Standardization**: Both input x and output y were scaled to zero mean and unit variance using Standard Scaler to improve training stability and convergence

3. Methodology

- Data Preparation

- Generated synthetic data using a predefined polynomial function
- Applied Standard Scaler to normalize both input and output
- Split data into training and testing sets using an 80:20 ratio

- Neural Network Architecture

- **Input Layer**: 1 node
- **Hidden Layers**: Two layers with ReLU activation
- **Output Layer**: 1 node with linear activation (for regression)
- **Loss Function**: Mean Squared Error (MSE)
- **Optimizer**: Gradient descent with manually tuned learning rate
- **Epochs**: 500
- **Batch Size**: 128

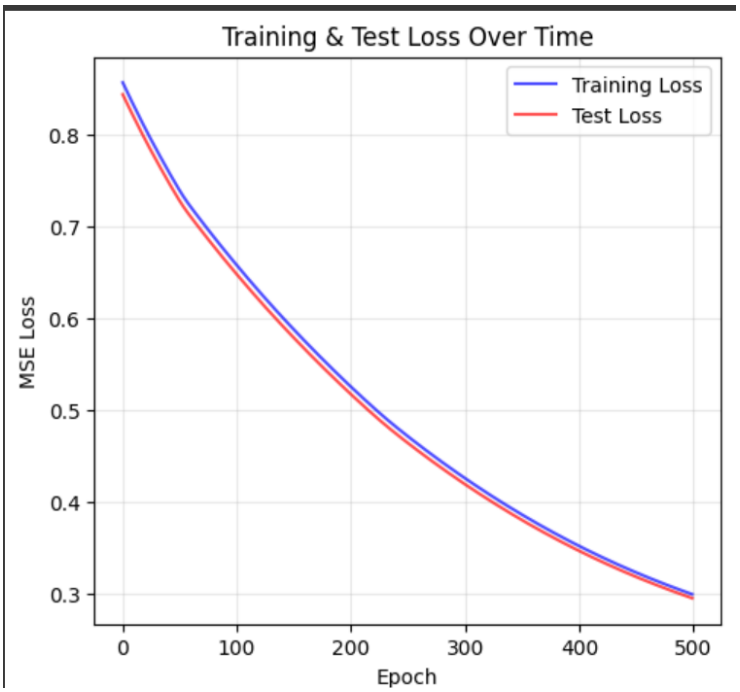
- Training & Evaluation

- Performed forward propagation to compute predictions
- Calculated loss and applied backpropagation to update weights

- Monitored training and test loss across epochs
- Evaluated final performance using:
- **Training Loss**
- **Test Loss**
- **R^2 Score**

4. Results and Analysis

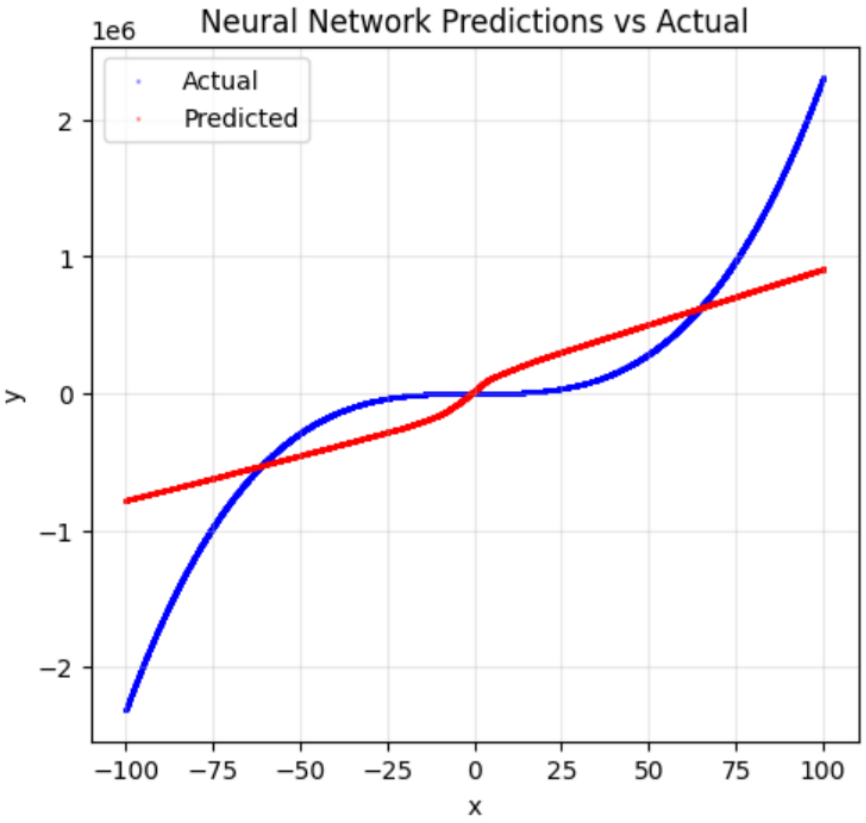
1.



2.

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.299396
Final Test Loss:     0.295144
R² Score:           0.7013
Total Epochs Run:   500
```

3.



4.

- The training and test losses are very close, which suggests that the model generalizes well and is not overfitting. Overfitting would typically show a much lower training loss compared to test loss.
- The R^2 score of 0.7013 indicates that the model explains about 70% of the variance in the target variable. This is a reasonably good fit, though not perfect — suggesting that the model has captured the underlying patterns but may still miss some complexity.
- Since the model trained for the full 500 epochs without early stopping, it's likely that the learning rate and architecture were well-balanced — allowing steady convergence without stagnation or overfitting

5.

Experiment	Learning Rate	No. of Epochs	Optimizer (if used)	Activation Function	Final Training Loss	Final Test Loss	R^2 Score
Exp-1	0.001	500	—	ReLU	0.299396	0.295144	0.7013

5. Conclusion

In this lab, we successfully implemented a neural network from scratch to approximate a synthetic polynomial function. By manually coding each component — from activation functions and loss computation to forward and backward propagation — we gained a deeper understanding of the mechanics behind neural learning.

The model demonstrated solid performance, with closely matched training and test losses and an R^2

score of 0.7013, indicating effective generalization and minimal overfitting. The use of standardized data and a well-tuned architecture contributed to stable convergence over 500 epochs.
This exercise not only reinforced core concepts in regression and optimization but also highlighted the importance of thoughtful design choices — such as learning rate, activation functions, and network depth — in building reliable models. The hands-on approach provided valuable insight into how neural networks learn complex patterns, laying a strong foundation for future work in deep learning and model diagnostics.