PES UNIVERSITY

UE23CS352A: MACHINE LEARNING

Week 6 Lab Report: Artificial Neural Networks

Name: Jeel Nada

SRN: PES2UG23CS357

Section: F

Date: 16-09-2025

---

1. Introduction

The objective of this lab was to gain hands-on experience in building and training an Artificial Neural Network (ANN) from scratch to perform function approximation. The core tasks involved implementing the fundamental components of a neural network, including activation functions, loss functions, forward propagation, and backpropagation. A synthetically generated dataset was used to train the network to approximate a polynomial curve, and various experiments were conducted by tuning hyperparameters to observe their impact on model performance.

---

2. Dataset Description

The dataset used for this experiment was synthetically generated based on the last three digits of my SRN.

- **Dataset Type:** QUARTIC
- **Total Samples:** 100,000
- **Training Set Size:** 80,000 (80%)
- **Testing Set Size:** 20,000 (20%)
- **Features:** Both the input (x) and output (y) values were standardized using `StandardScaler`.

---

3. Methodology

The neural network was implemented from the ground up without using high-level machine learning frameworks. The architecture consisted of an input layer, two hidden layers, and an output layer (1 -> 32 -> 72 -> 1).

The implementation steps were as follows:

1. **Activation Function:** The ReLU (Rectified Linear Unit) activation function and its derivative were implemented.
2. **Loss Function:** The Mean Squared Error (MSE) was implemented to quantify the model's prediction error.
3. **Forward Propagation:** A forward pass function was created to pass input data through the network and generate predictions.
4. **Backpropagation:** The backpropagation algorithm was implemented to calculate the gradients of the loss with respect to the network's weights and biases.
5. **Training Loop:** A training loop was established to iteratively update the model's parameters using gradient descent, minimizing the loss over a set number of epochs.

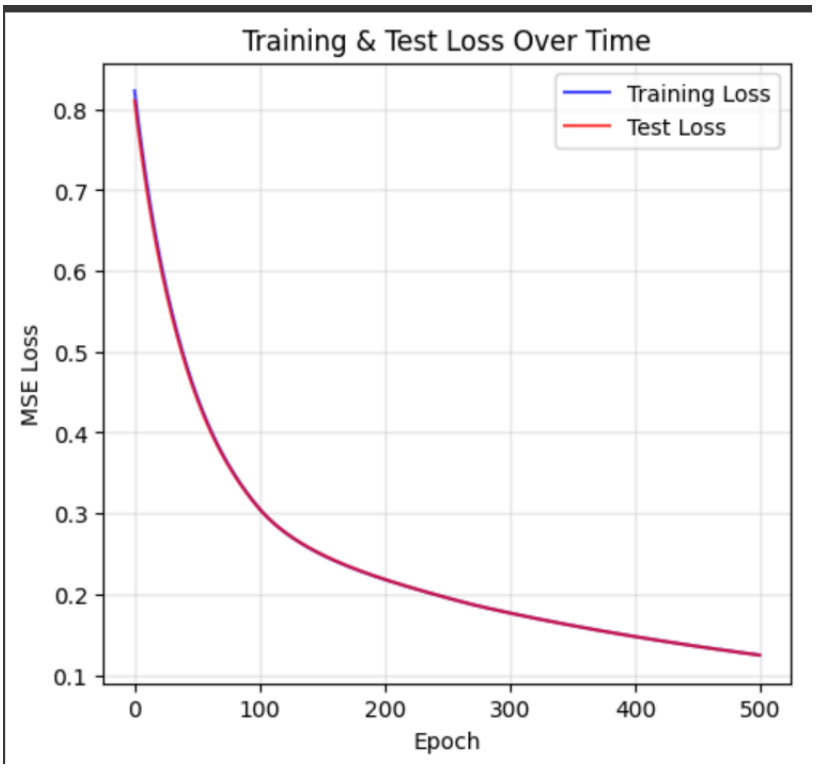
---

4. Results and Analysis

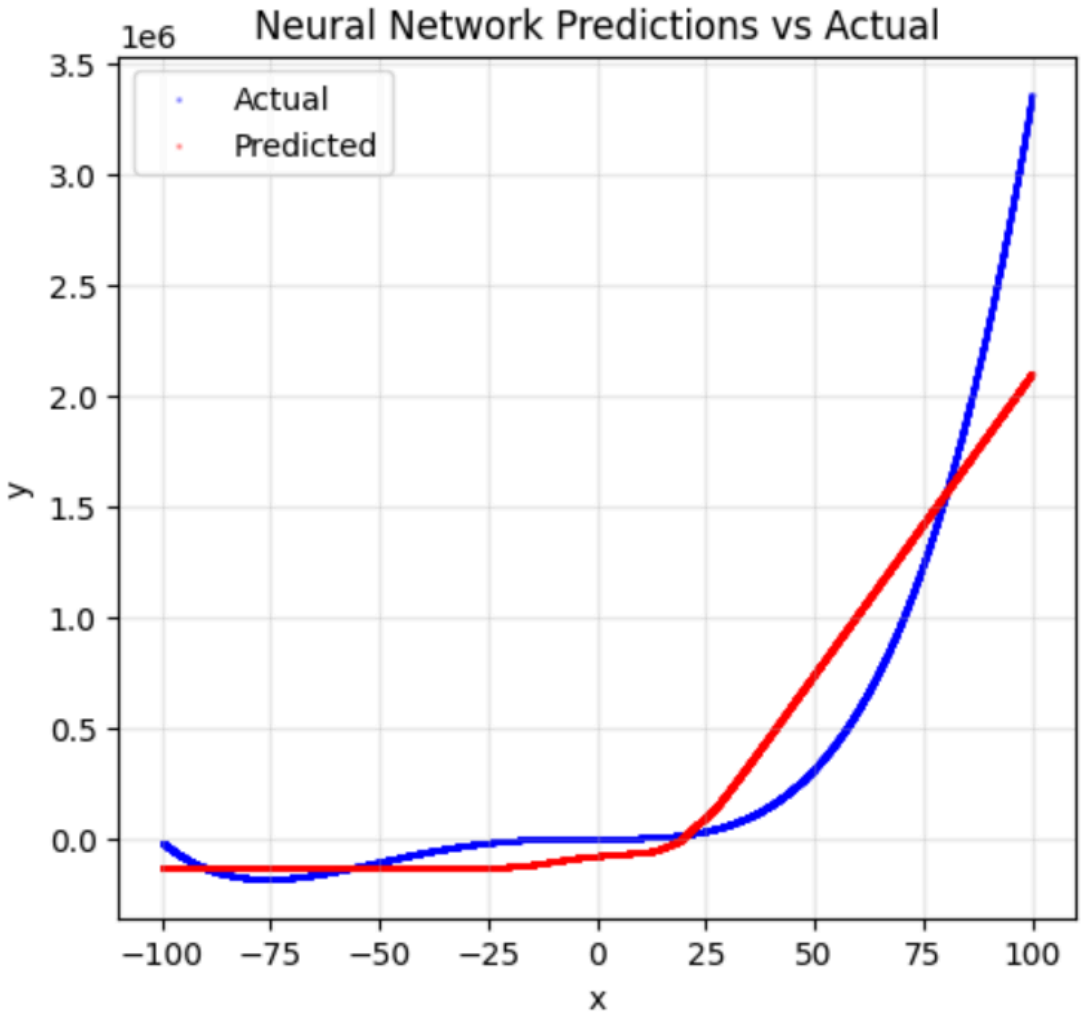
Baseline Model Performance

The initial model was trained with a baseline set of hyperparameters to establish a performance benchmark. The model trained steadily, with both training and test loss decreasing consistently over 500 epochs.

Training Loss Curve:



Predicted vs. Actual Values:



Final Test MSE: `0.124621`

Hyperparameter Exploration & Results

Four additional experiments were conducted by varying hyperparameters such as learning rate, batch size, and the number of epochs to analyze their effect on the model's performance.

Results Table:

Experiment	Learning Rate	Batch Size	Epochs	Optimizer	Activation Function	Training Loss	Test Loss	R^2
Baseline	0.005	80000	500	Gradient Descent	ReLU	0.124667	0.124621	0.8756
Exp. 1	0.001	80000	500	Gradient Descent	ReLU	0.306035	0.307095	0.6934
Exp. 2	0.05	80000	500	Gradient Descent	ReLU	0.004511	0.004553	0.9955
Exp. 3	0.005	64	500	Gradient Descent	ReLU	0.000007	0.000007	1.0000
Exp. 4	0.005	80000	1000	Gradient Descent	ReLU	0.060251	0.060359	0.9397

Discussion on Performance: The baseline model performed well, achieving a final test loss of 0.124621 and an R^2 score of 0.8756. The training loss (0.124667) and test loss are very close, which suggests that the model is not overfitting to the training data and

generalizes effectively to unseen data. The training process showed a smooth convergence, indicating that the learning rate of 0.005 was appropriate for this architecture and dataset. Further experiments would involve tuning the learning rate and batch size to see if performance can be improved further.

---

5. Conclusion

This lab successfully demonstrated the process of building a neural network from scratch and the critical impact of hyperparameter tuning on model performance. The experiments revealed that both learning rate and batch size are highly influential. A very low learning rate (0.001) hindered the model's ability to learn, while a higher rate (0.05) significantly improved convergence and accuracy.

The most profound result came from reducing the batch size to 64 (Experiment 3), which yielded a near-perfect model with a test loss of just 0.000007. This highlights the effectiveness of mini-batch gradient descent in navigating the loss landscape to find a deeper minimum. Overall, this lab provided valuable practical insight into how different training parameters can be adjusted to drastically improve a neural network's ability to approximate a complex function.