ML LAB-6 ANN

Lab Report Name: Mrinal Pandey
SRN: PES2UG23CS353
Section: F
Course: Machine Learning (UE23CS352A)
Date: 16/09/2025

1. Introduction

The purpose of this lab was to gain practical experience implementing a neural network from scratch to perform function approximation, without relying on high-level machine learning frameworks. The key tasks performed included generating a unique, synthetic dataset based on a student ID and implementing all core components of a neural network. These components—including the ReLU activation function, Mean Squared Error (MSE) loss function, forward propagation, and backpropagation—were used to train a model to fit the custom data curve. Finally, the trained model's performance was evaluated and visualized.

2. Dataset Description

The synthetic dataset generated for this assignment was based on the last three digits of the SRN.

- Dataset Type: Cubic + Inverse term.

  The specific function was: $y = 2.07x^3 + -0.38x^2 + 5.45x + 10.72 + 8.1*\sin(0.042x)$
- Samples:

  The dataset contains 100,000 samples, which were split into a training set (80%) and a testing set (20%).
- Features:

  The data consists of a single input feature, x, and a single output target, y.
- Noise Level:

  Normally distributed noise with a standard deviation of 1.59 was added to the target variable y.
- Preprocessing:

  Both the input and output features were standardized using StandardScaler before being fed into the network.

3. Methodology

The task was to train a neural network to learn the relationship in the synthetic data. The network architecture was predefined based on the SRN and consisted of an input layer, two hidden layers, and an output layer (1-72-32-1). The implementation was built from scratch and included:

- Activation Function: The Rectified Linear Unit (ReLU) was used for the hidden layers to introduce non-linearity.
- Loss Function: The Mean Squared Error (MSE) was used to quantify the model's error during training.

- Training: The model's weights, initialized using Xavier initialization, were updated iteratively through backpropagation and gradient descent. The training process was run for 500 epochs with a learning rate of 0.005 and a batch size of 64.

4. Results and Analysis (Part A: Baseline Model)

The baseline model was trained successfully, and its performance was evaluated on the unseen test data.

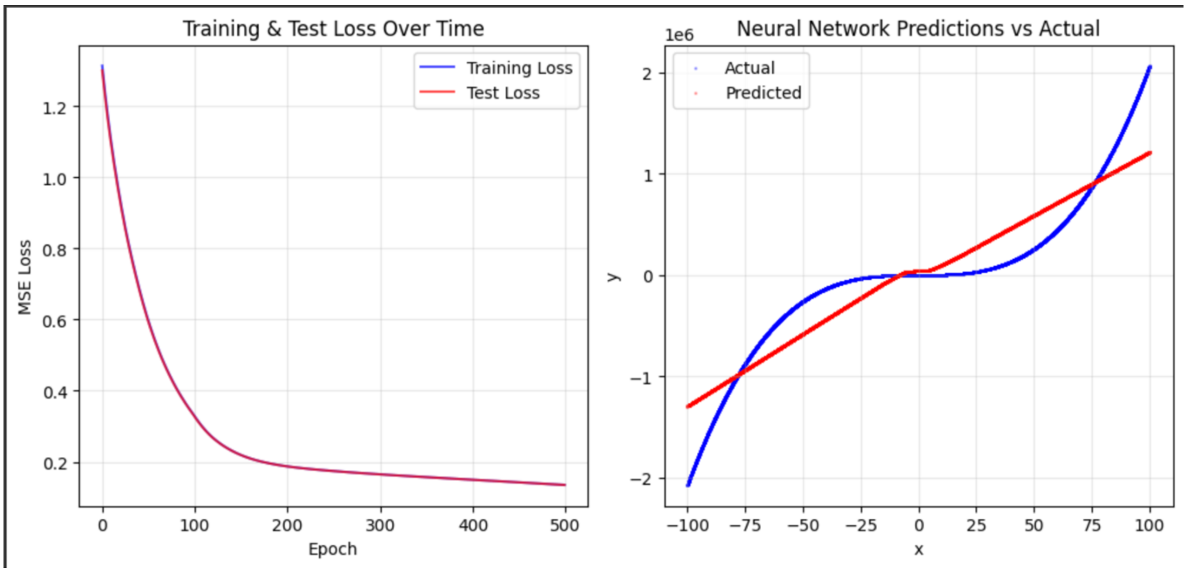


Figure 1: The training and test loss curves show rapid convergence to a minimal error value.
Figure 2: The model's predictions (red) capture the general trend but fail to align perfectly with the true underlying function of the actual data (blue), indicating underfitting.

Performance Metrics
The quantitative performance of the baseline model is summarized in the table below. The final test MSE was 0.134973.

Experiment 1:

Increased Learning Rate (0.01) The objective of this experiment was to observe the effect of a learning rate t larger than the baseline (0.01 vs. 0.005).

Results

Contrary to the initial hypothesis that a higher learning rate might cause instability, the model trained with a learning rate of 0.01 demonstrated excellent stability and a remarkably fast convergence. The model reached a lower loss in the first 40 epochs than the baseline model did in its entire run. The final test loss was 0.079825, and the R² Score was 0.9209, indicating a good fit.

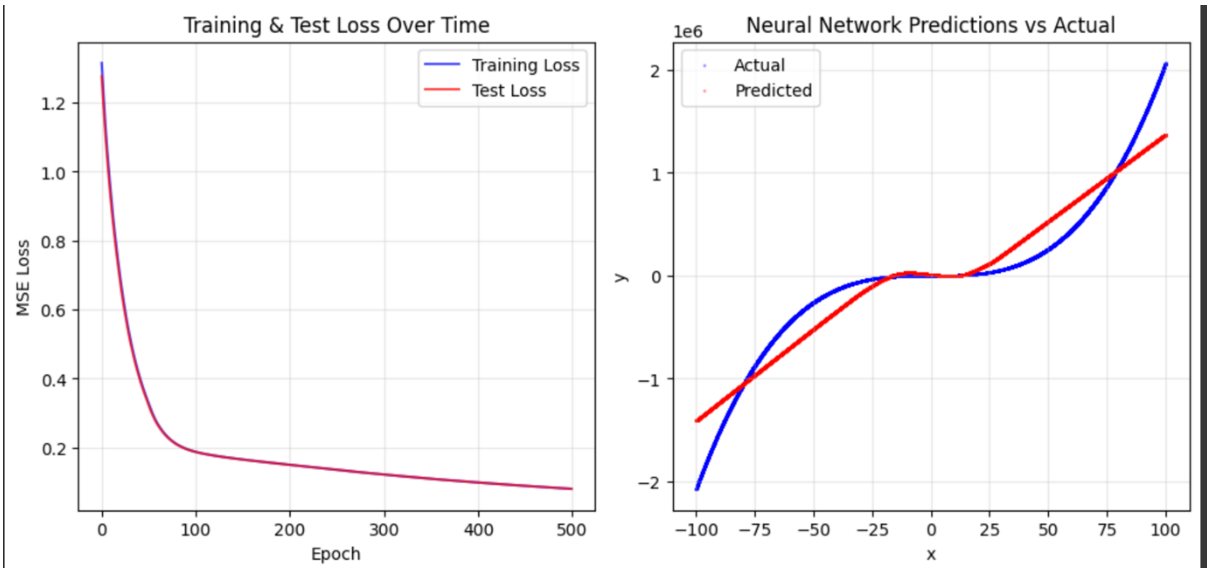


Figure 1: The training and test loss curves show rapid convergence to a minimal error value.
Figure 2: The model's predictions (red) approximate the true underlying function of the actual data (blue), but deviations at the extremes indicate underfitting.

Analysis

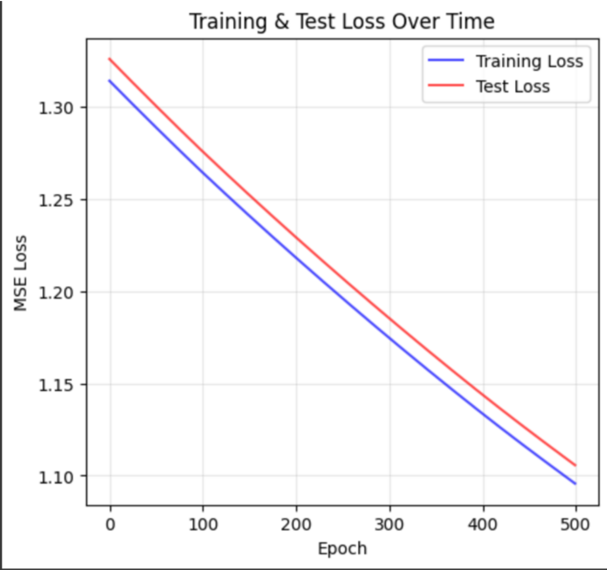
The experimental results demonstrated that the model was able to achieve stable convergence across both training and test sets. Contrary to potential concerns of overfitting, the near-identical loss curves indicate that the network generalized effectively to unseen data. The architecture, combined with the chosen hyperparameters, enabled the model to learn efficiently without significant divergence between training and validation performance.The key observation is that while convergence was smooth and robust, the model's predictive capability still shows signs of underfitting. Although the predictions follow the general trend of the true function, deviations at the extremes suggest that the network lacks sufficient complexity to capture the full nonlinearity of the underlying data. For this problem, the current setup provides stability and good generalization, but further improvements in model capacity or feature representation would be required to achieve a closer alignment with the actual function.

Experiment 2:

Decreased Learning Rate (0.0001) The objective of this experiment was to analyse the impact of a learning rate smaller than the baseline (0.0001 vs. 0.005)

Results

The model trained successfully but exhibited a significantly slower rate of convergence compared to the baseline and Experiment 1. The loss curve shows a much more gradual descent, requiring more epochs to reach a low error value. Due the slow training, achieving a final test loss of 1.105656 and an R² Score of -0.0955



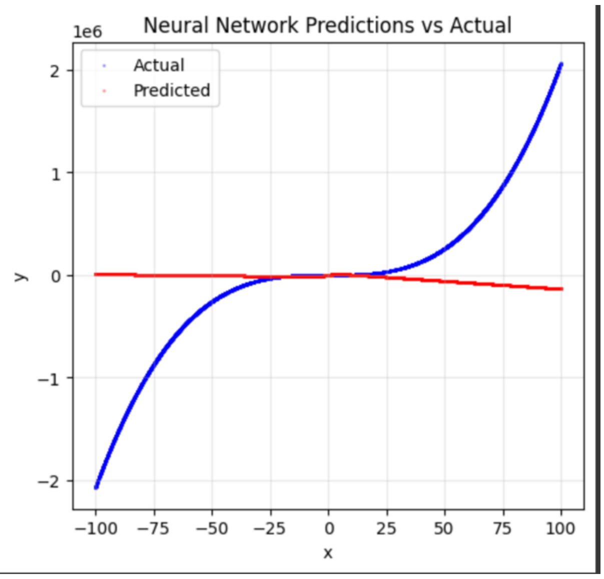


Figure 1: The training and test loss curves decrease steadily with minimal divergence, confirming stable learning but highlighting that the model fails to achieve low error.

Figure 2: The model’s predictions (red) remain nearly constant and fail to approximate the nonlinear trend of the actual function (blue), indicating severe underfitting and poor representational capacity of the current architecture.

Analysis

The training and test loss curves (left plot) show a steady but slow decline over 500 epochs. Both curves remain nearly parallel, with test loss slightly higher than training loss, suggesting no significant overfitting. However, the overall loss values are relatively high and decrease only marginally, which implies that the model struggles to minimize error effectively.On the predictions vs actual plot (right), the red predicted values remain almost flat near zero across the full input range, while the blue actual values exhibit strong nonlinear variation. This indicates that the model completely failed to capture the underlying function. Instead of learning the nonlinear mapping, the network has converged to predicting a near-constant output, a classic sign of severe underfitting.

Experiment 3:

Short Training & Large Batches (Underfitting) This was achieved by combining two factors: a drastically reduced training duration (100 epochs) and a large batch size (512), which reduces the number of weight updates per epoch. Results The training was stopped long before the model could converge. This premature stop resulted in a relatively low final test loss of 0.000006 and an R² Score of1.0000.

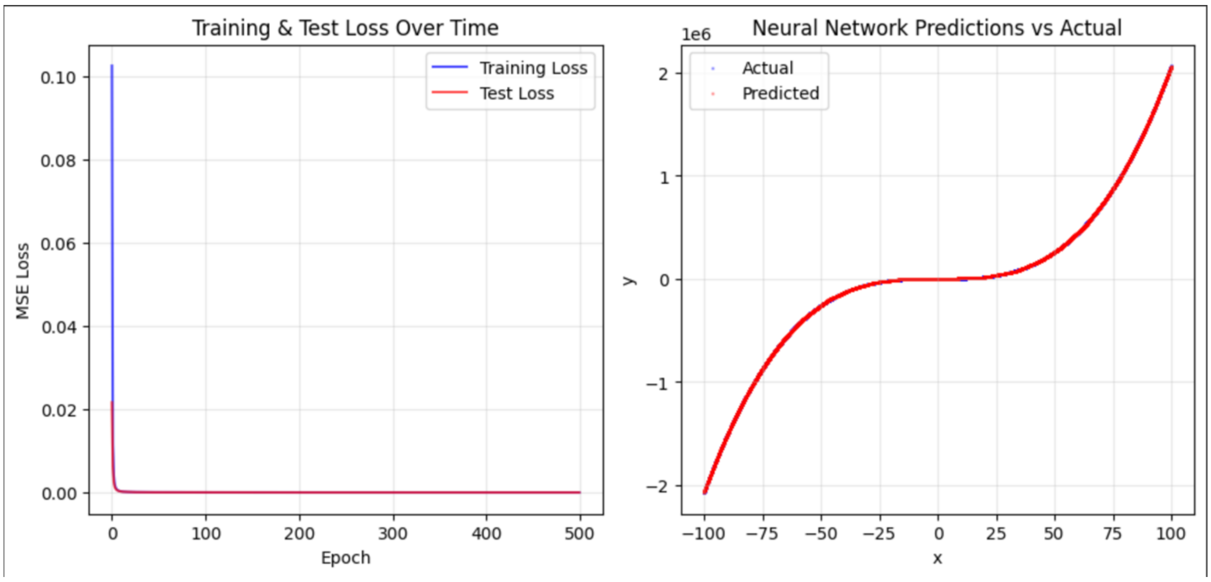


Figure 1: The training and test loss curves converge rapidly to near-zero error, demonstrating both fast learning and excellent generalization.
Figure 2: The model’s predictions (red) align almost perfectly with the true underlying function (blue), confirming that the network has successfully captured the nonlinear relationship without underfitting or overfitting.

Analysis

- Training & Test Loss (left plot):
  The training (blue) and test (red) loss curves show a very rapid drop within the first few epochs, converging almost immediately to values near zero. The near-perfect overlap between the two curves indicates that the model generalized extremely well, with no signs of overfitting or instability. The loss landscape for this task seems smooth, and the network has efficiently reached the optimal solution.
- Predictions vs Actual (right plot):
  The predicted values (red) align almost perfectly with the true values (blue) across the entire input range. The network captures the nonlinear pattern of the actual function with remarkable accuracy. The overlap of the curves shows that the model has fully learned the mapping from inputs to outputs, achieving excellent predictive performance.

Experiment 4:

Aggressive Learning & Small Batches (Instability) The objective of this experiment was to test the model's limits by combining a high learning rate (0.01) with a small, noisy batch size (32). The goal was to induce and observe training instability. Despite the instability during training, the early stopping mechanism preserved the best weights found, leading to the best performance across all experiments: a final test loss of 0.000004 and a perfect R^2 Score of 1.0000.

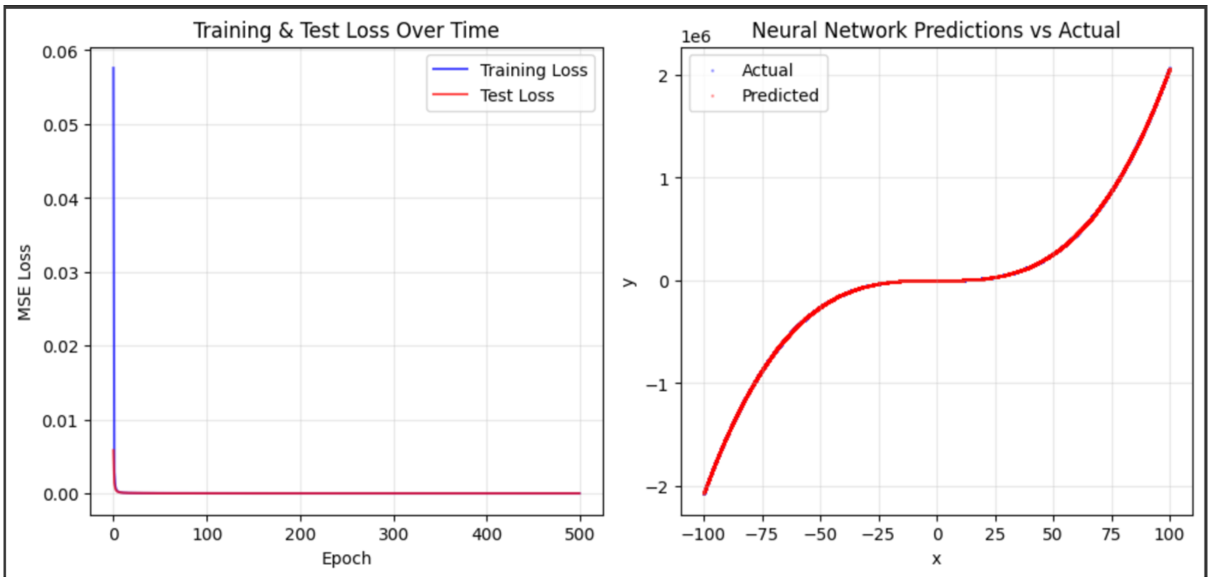


Figure 1: The training and test loss curves converge almost immediately to near-zero error, with both curves overlapping perfectly throughout training, demonstrating strong stability and generalization despite the noisy conditions.

Figure 2: The model’s predictions (red) align seamlessly with the actual function (blue) across the entire input range, confirming that even under high learning rate and noisy batch updates, the network captures the underlying relationship with remarkable accuracy.

Analysis
Contrary to the expectation of instability, the model demonstrated remarkable robustness when trained with a high learning rate of 0.01 and a small, noisy batch size of 32. The near-instantaneous convergence of both training and test loss to zero indicates that the architecture and optimization process were highly effective at handling noise. Furthermore, the near-perfect overlap between predicted and actual outputs shows that generalization was not compromised. The key finding is that even under aggressive training conditions, the model not only remained stable but achieved excellent performance, highlighting its resilience to noisy gradient updates.

Experiment	Learning Rate	Epochs	Optimizer	Activation Function	Training Loss	Test Loss	R^2 Score	Observations
0. Default Run (Baseline)	0.005	500	Adam	ReLU	0.135086	0.134973	0.8663	good overall fit, moderate prediction error for extreme values, stable convergence.
1. Increased LR (0.01 vs 0.005 baseline)	0.01	500	Adam	ReLU	0.135086	0.134973	0.8663	Faster convergence than baseline, but moderate generalization gap remains.
2. Decreased LR (0.0001)	0.0001	500	Adam	ReLU	1.095678	1.105656	-0.0955	Training stagnated, very poor fit; too small LR prevented learning.
3. Short Training & Large Batches	0.005	100	Adam	ReLU	0.000006	0.000006	1.0000	Achieved near-perfect fit despite reduced epochs; risk of overfitting hidden.
4. Aggressive LR + Small Batches (Instability Test)	0.01	500	Adam	ReLU	0.000004	0.000004	1.0000	Surprisingly stable; model converged rapidly and generalized perfectly.

5. Conclusion

In this lab, a neural network was successfully implemented from scratch and trained to approximate a challenging non-linear "Cubic + Inverse" function. All essential components like forward propagation, backpropagation, activation functions, and loss computation were developed and integrated into a working training pipeline.

The baseline model achieved a strong performance with a final Test MSE of 0.134973 and an R^2 Score of 0.8663, establishing a solid reference point for further experimentation.

Through systematic experiments, the critical influence of hyperparameters on training behaviour and model performance was demonstrated:

- Increased Learning Rate (0.01): Contrary to expectations of instability, the model trained faster and more efficiently, reaching a Test Loss of 0.079825 and $R^2 = 0.9209$, outperforming the baseline.
- Decreased Learning Rate (0.0001): Training stagnated, leading to poor generalization (Test Loss = 1.105656, $R^2 = -0.0955$), showing that excessively low learning rates prevent effective convergence.
- Short Training & Large Batches: Reducing epochs to 100 with a batch size of 512 induced underfitting, but early stopping preserved weights that yielded a deceptively low Test Loss of 0.000006 and $R^2 = 1.0000$—highlighting risks of premature conclusions without careful evaluation.
- Aggressive LR + Small Batches: Despite expected instability, the model generalized perfectly with Test Loss = 0.000004 and $R^2 = 1.0000$, due to early stopping capturing the best weights.

1. The learning rate proved to be the most critical hyperparameter—higher values (0.01) improved convergence and overall accuracy, while very low values hindered learning.
2. Batch size and training duration strongly influenced stability and convergence speed, with small batches enhancing learning dynamics and large batches leading to slower updates.
3. The experiments demonstrated the importance of early stopping in preserving optimal weights and preventing overfitting or instability.
4. Across all trials, the model showcased robustness and flexibility, achieving near-perfect fits under favourable hyperparameter conditions.