UE23CS352A: MACHINE LEARNING

Week 6: Artificial Neural Networks

Implementing Neural Networks from scratch

Name : CHARAN M REDDY

SRN : PES2UG23CS146

Course : MACHINE LEARNING

Date : 19/09/2025

---

1. Introduction

Purpose of the Lab :

The purpose of the lab is to implement a neural network from scratch without using the high level frameworks .

Tasks performed :

Using the Custom Dataset Generated based on SRN , we Implemented :

-Activation functions

-Loss functions

-Weight initialization using Xavier Technique.
-Forward propagation

-Backward propagation

-Training Function
We executed the training and visualised the results by plotting Training and Test Loss graphs and Prediction vs Actual graph .
Then we did the same 4 times , changing the hyperparameters or the activation function each time .

A comparison table was created comparing all the important metrics and the graphs for the 4 experiments .

---

2. Dataset Description

Type of polynomial assigned :

CUBIC POLYNOMIAL

$y = 2.15x^3 + -0.33x^2 + 3.90x + 9.04$

Number of samples, features, noise level :

Samples : 100,000 (Training - 80,000 , Test – 20,000)

Input feature : 1(x)

Target variable : 1(y)

Noise Level : $\varepsilon \sim N(0, 1.83)$

---

3. Methodology

A dataset was generated based on the last three digits of my SRN.

The chosen architecture is **1 -> 72 -> 32 -> 1** (wide-to-narrow).

Input layer: 1 neuron (for x)

Hidden Layer 1: 72 neurons (ReLU activation)

Hidden Layer 2: 32 neurons (ReLU activation)

Output layer: 1 neuron (Linear activation) (for y)

Xavier initialization was used to avoid vanishing gradients and the biases were set to zero.

Computed activations in Forward Propagation

MSE(Mean Squared Error) was used in loss function

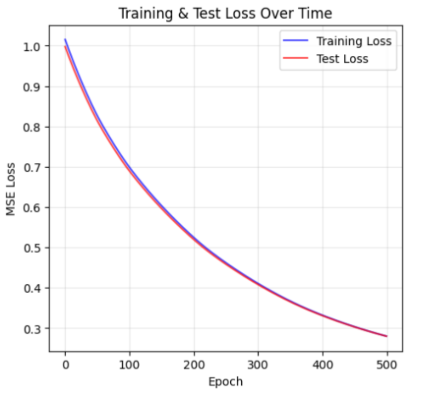
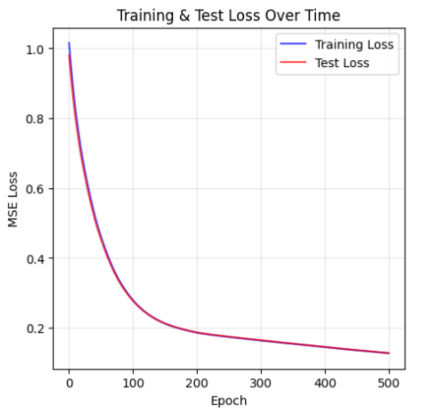
In backpropagation gradients were found using the chain rule, ReLU derivative was applied and ,weights and biases were updated using gradient descent.

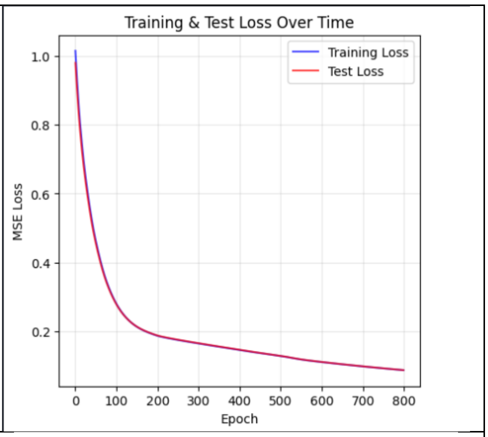
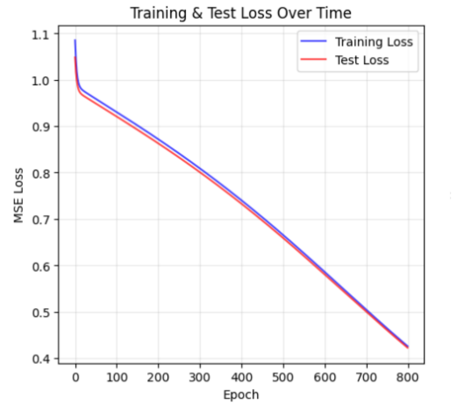
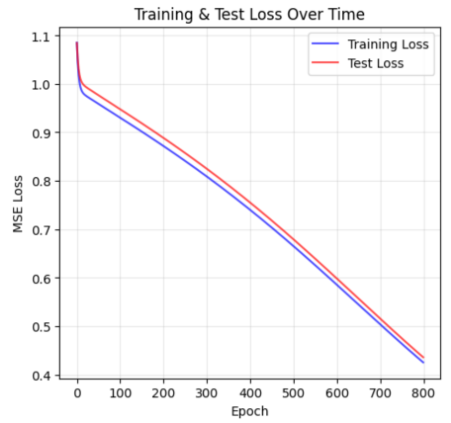
Training loop executed for a maximum of 500 epochs.

4. Results and Analysis

Experiment	Changes
0	Baseline
1	LR = 0.005
2	Epochs = 800
3	AF = sigmoid
4	Batch size = 50000

<u>Training and Test Loss Curves :</u>

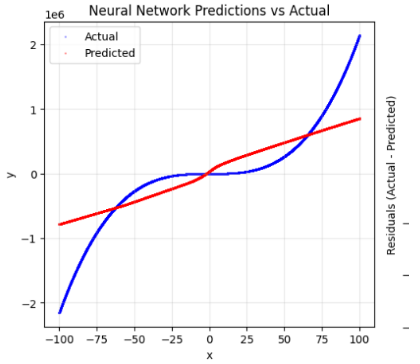
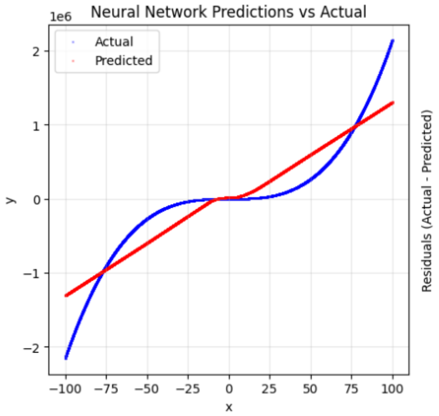
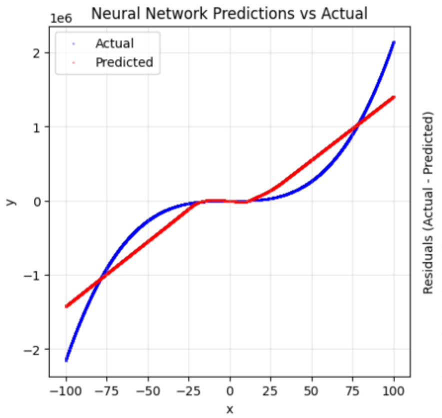
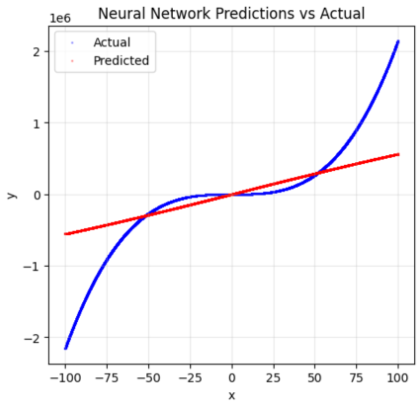
Experiment	Graph
0	
1	

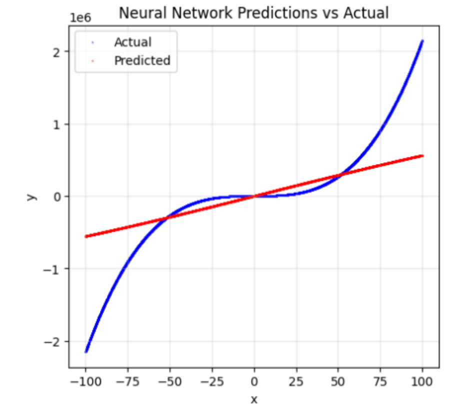
2	
3	
4	

Final test MSE :

Experiment	MSE
0	0.279700
1	0.279704
2	0.086562
3	0.422575
4	0.435997

Plot of predicted vs. actual values :

Experiment	Graph
0	
1	
2	
3	

4	

Discussion on performance (overfitting / underfitting):

Experiment 0 :

The training and test losses are close, showing that the model generalized well.

Relatively high loss and moderate R^2 indicate slight underfitting.

Experiment 1 :

The performance is almost identical to the baseline. This shows that increasing the learning rate alone does not significantly impact the model, since convergence is already limited by the number of epochs.

The model shows mild underfitting.

Experiment 2 :

Lower losses and a much higher R^2 demonstrate improved learning.

The training and test losses are close, which means the model does not overfit.

This is the best-performing model.

Experiment 3 :

The model performs significantly worse compared to ReLU.

The high losses and low R^2 show clear underfitting.

Sigmoid is not well-suited here due to vanishing gradient issues with large input values.

Experiment 4 :

Performance is identical to the Sigmoid experiment , showing that the activation function had a stronger influence on model quality than batch size.

The model continues to underfit.

Results Table :

Experiment	Learning Rate	No. of epochs	Activation function	Final Training Loss	Final Test Loss	R^2 value
0	0.001	500	ReLU	0.279888	0.279700	0.7175
1	0.005	500	ReLU	0.279888	0.279700	0.7175
2	0.005	800	ReLU	0.279888	0.279700	0.9126
3	0.005	800	Sigmoid	0.425919	0.422575	0.5731
4	0.005	800	Sigmoid	0.425487	0.435997	0.5721

---

5. Conclusion

In this lab, a neural network was implemented from scratch to approximate a cubic polynomial with added noise , which was assigned based on out SRN . The baseline model showed moderate performance, while experiments with hyperparameter variations provided change in the same . Increasing the number of training epochs significantly improved accuracy and generalization. ReLU activation consistently outperformed Sigmoid, which suffered from vanishing gradient issues and led to underfitting. Variations in learning rate and batch size produced minimal effect compared to activation function and training duration.