UE23CS352A: MACHINE LEARNING

Week 6: Artificial Neural Networks

NAME: Nikita Kolathaya

SRN: PES2UG23CS387

DATE: 16/09/25

SECTION: F

INTRODUCTION:

The purpose of this lab is to implement and understand the working of a neural network from scratch using NumPy.

Tasks:

- Implementing neural network components such as ReLU activation, MSE loss, forward, and backward propagation
- Initializing network weights and biases using the Xavier initialization method
- Setting up and executing a complete training loop with gradient descent
- Visualizing training results: plotting training/test loss curves and prediction vs true values

DATASET DESCRIPTION:

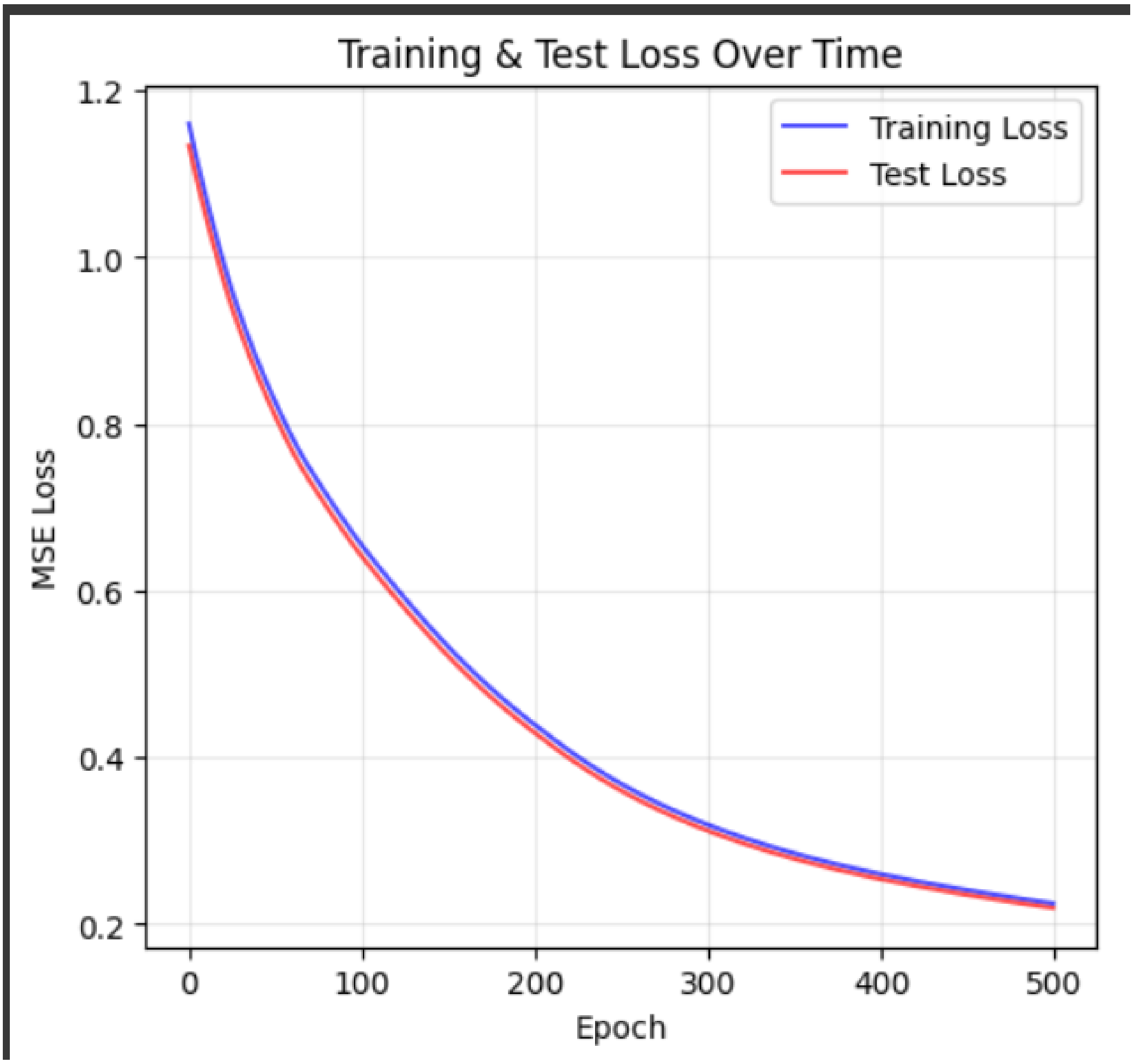
```
======================================================================
ASSIGNMENT FOR STUDENT ID: PES2UG23CS387
======================================================================
Polynomial Type: QUARTIC: y = 0.0151x⁴ + 1.65x³ + -0.91x² + 3.58x + 9.75
Noise Level: ε ~ N(0, 2.38)
Architecture: Input(1) → Hidden(96) → Hidden(96) → Output(1)
Learning Rate: 0.003
Architecture Type: Large Balanced Architecture
======================================================================
```

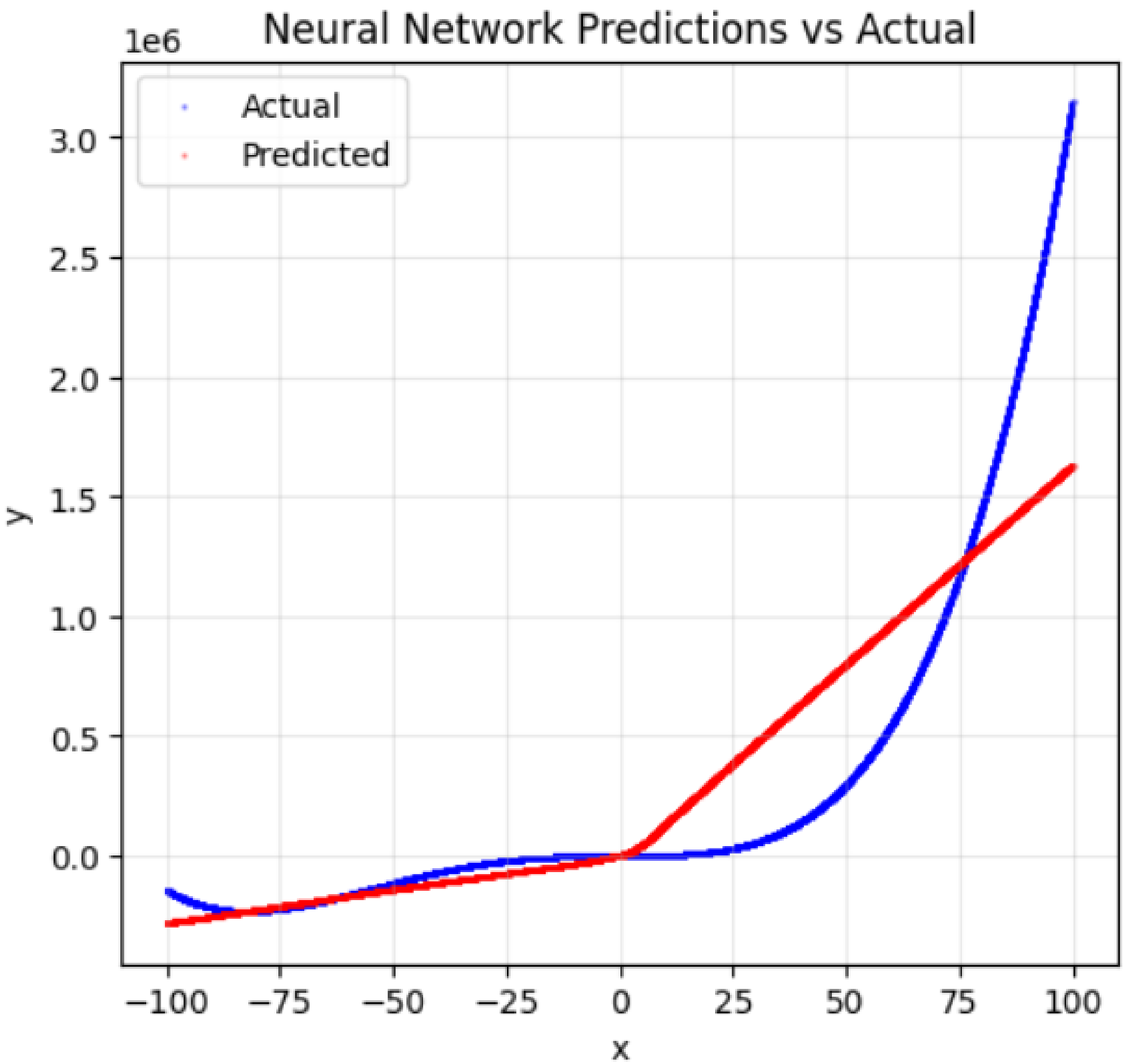
```
Dataset with 100,000 samples generated and saved!
Training samples: 80,000
Test samples: 20,000
```

METHODOLOGY:

- Architecture: Designed a 3-layer feedforward neural network with one input node, two hidden layers, and one output node.
- Functions Implemented: Developed NumPy functions for activation (ReLU), activation derivative, mean squared error loss, forward propagation, backward propagation (for gradient computation), and Xavier weight initialization.
- Forward Propagation: Passed input data through all layers, applying activations and linear transformations to obtain predictions.
- Loss Computation: Calculated mean squared error (MSE) between predicted and true output values.
- Backpropagation: Computed gradients of loss with respect to all weights and biases using the chain rule and updated network parameters using gradient descent.
- Early Stopping: Monitored test loss and stopped training if performance ceased improving for a defined number of epochs (patience).

RESULTS AND ANALYSIS:






```
==================================================
PREDICTION RESULTS FOR x = 90.2
==================================================
Neural Network Prediction: 1,469,493.06
Ground Truth (formula):    2,200,309.97
Absolute Error:            730,816.91
Relative Error:            33.214%
```

Final error:

```
==================================================
FINAL PERFORMANCE SUMMARY
==================================================
Final Training Loss: 0.224590
Final Test Loss:     0.219223
R² Score:           0.7775
Total Epochs Run:   500
```

DISCUSSION ON PERFORMANCE:

- The test loss closely follows the train loss throughout, with no divergence, implying the network is not memorizing the training data. Therefore we can say **no overfitting** occurs.
- The prediction for x = 90.2 does not match the truth - relative error is 33.2%. This error implies the model has not perfectly captured the underlying function signalling **underfitting.**

CONCLUSION:

- The model was able to reduce loss steadily across training and testing datasets, with parallel curves indicating healthy generalization and no sign of overfitting.
- The final performance reached an R^2 of 0.7775, consistent with moderate predictive power.
- The 33% error in sample prediction points to underfitting, suggesting that additional model capacity or hyperparameter tuning may be required for further improvement.