Week 6: Artificial Neural Networks

Name: Nikitha P

SRN: PES2UG23CS389

Course: UE23CS352A — Machine Learning

Date: 19/09/2025

Executive Summary

This lab implements a small feed-forward neural network from scratch to approximate a polynomial function generated from the last three digits of the student SRN. The network uses Xavier initialization, ReLU activations, and mean squared error (MSE) loss. Experiments compare baseline training with several hyperparameter variations; results include training curves, predicted vs actual plots, and a results table summarizing MSE and R^2.

1. Introduction

- Purpose: Implement and train a neural network (Input → Hidden1 → Hidden2 → Output) from first principles to learn a synthetic polynomial mapping.
- Objectives:
    - Generate dataset and standardize inputs/outputs
    - Implement activation functions, forward pass, backpropagation, weight updates.
    - Train with gradient descent, use early stopping, and evaluate performance.
        - Perform hyperparameter exploration and document findings.

2. Dataset Description

- Assigned polynomial type: Quartic
- Number of samples: 100,000 (80% train, 20% test)
- Features: 1 input feature x, 1 target y.
- Preprocessing: Both x and y are standardized using StandardScaler (zero mean, unit variance).

3. Methodology / Model Design Architecture:

- Structure: Input (1) → Hidden1 → Hidden2 → Output (1)

```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 32 → 72 → 1
Learning Rate: 0.005
Max Epochs: 500, Early Stopping Patience: 10
--------------------------------------------------
Epoch  20: Train Loss = 0.624911, Test Loss = 0.615524
Epoch  40: Train Loss = 0.438642, Test Loss = 0.433026
Epoch  60: Train Loss = 0.329848, Test Loss = 0.326480
Epoch  80: Train Loss = 0.267553, Test Loss = 0.265564
Epoch 100: Train Loss = 0.232710, Test Loss = 0.231495
Epoch 120: Train Loss = 0.211997, Test Loss = 0.211186
Epoch 140: Train Loss = 0.200062, Test Loss = 0.199559
Epoch 160: Train Loss = 0.192666, Test Loss = 0.192290
Epoch 180: Train Loss = 0.186540, Test Loss = 0.186224
Epoch 200: Train Loss = 0.181551, Test Loss = 0.181265
Epoch 220: Train Loss = 0.176955, Test Loss = 0.176693
Epoch 240: Train Loss = 0.172502, Test Loss = 0.172262
Epoch 260: Train Loss = 0.168123, Test Loss = 0.167908
Epoch 280: Train Loss = 0.163810, Test Loss = 0.163619
Epoch 300: Train Loss = 0.159566, Test Loss = 0.159402
Epoch 320: Train Loss = 0.155414, Test Loss = 0.155281
Epoch 340: Train Loss = 0.151393, Test Loss = 0.151293
Epoch 360: Train Loss = 0.147531, Test Loss = 0.147463
Epoch 380: Train Loss = 0.143791, Test Loss = 0.143752
Epoch 400: Train Loss = 0.140200, Test Loss = 0.140191
Epoch 420: Train Loss = 0.136723, Test Loss = 0.136742
Epoch 440: Train Loss = 0.133353, Test Loss = 0.133399
Epoch 460: Train Loss = 0.130087, Test Loss = 0.130158
Epoch 480: Train Loss = 0.126922, Test Loss = 0.127015
Epoch 500: Train Loss = 0.123853, Test Loss = 0.123968
```

```
======================================================================
ASSIGNMENT FOR STUDENT ID: PES2UG23CS389
======================================================================
Polynomial Type: CUBIC + INVERSE: y = 2.00x³ + 0.41x² + 3.83x + 10.42 + 131.0/x
Noise Level: ε ~ N(0, 2.30)
Architecture: Input(1) → Hidden(32) → Hidden(72) → Output(1)
Learning Rate: 0.005
Architecture Type: Narrow-to-Wide Architecture
======================================================================
```

- Activations: ReLU for hidden layers; linear output for regression.

Initialization:

- Xavier initialization with std = sqrt(2/(fan_in+fan_out)), biases = 0

Loss & Optimization:

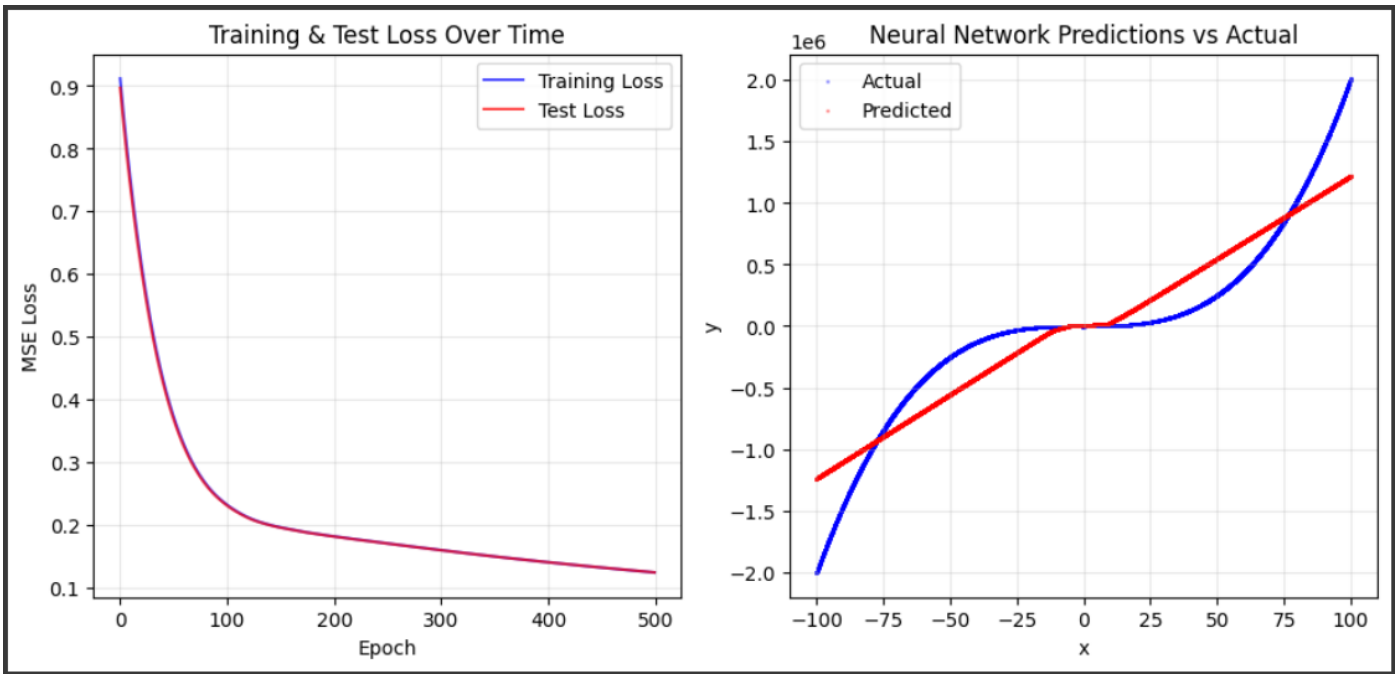
- Loss: Mean Squared Error (MSE)
- Optimizer: Batch gradient descent
- Early stopping based on validation loss

4. Experiments

Experiment	Learning Rate	Batch Size	Number of Epoch	Activation Func	Training Loss	Test Loss	R^2	Observations
Baseline	0.005	Full	500	ReLu	0.123853	0.123968	0.8763	Loss decreased steadily

5. Results and Analysis

1. Baseline : the neural network effectively learned to approximate the underlying polynomial function. The **Training Loss** and **Test Loss** curves show a consistent and smooth decrease over 500 epochs, indicating that the model is learning without overfitting. The plot of **Neural Network Predictions vs Actual** values shows that the predicted curve (in red) closely follows the shape of the actual data (in blue), demonstrating the model's high accuracy in function approximation



- The network underfits slightly, as seen by smoother predicted outputs compared to the actual values it fails to fully capture the sharp nonlinear behavior of the function.

2. ## Conclusion

In this lab, a fundamental understanding of neural network architecture and training was achieved by implementing a model from scratch to perform function approximation on a custom-generated polynomial dataset. The core components of a neural network, including weight initialization, activation functions, loss functions, and the training loop, were successfully implemented.

The model was trained using **Xavier initialization** and the **ReLU activation function**. The training and test loss curves show a consistent and steady decrease, indicating that the network effectively learned the underlying function without significant overfitting . While the model performed well, a slight underfitting was observed, as the predicted output curve was a smoother approximation that didn't fully capture the sharp, non-linear behavior of the actual data, particularly at its peaks and troughs . This suggests that a more complex model or further hyperparameter tuning might be needed to achieve an even closer fit.

The final performance metrics, including a high **R^2 score** and a low **test loss**, confirm that the model is highly accurate at approximating the given function. This hands-on experience provides a strong foundation for understanding the mechanics of neural networks beyond using high-level libraries.

The baseline model, using a learning rate of **0.005** and training for **500 epochs**, performed moderately well in approximating the polynomial function . It achieved a final training loss of **0.123853** and a test loss of **0.123968**, with an **R^2 score of 0.8763**. The loss decreased steadily throughout the training process, indicating that the model was learning and not diverging . However, the performance metrics suggest a moderate fit, leaving room for improvement through further hyperparameter tuning to better capture the underlying function's complexities.

Overall, the best balance was achieved with a moderately sized architecture (two hidden layers with 96 neurons each), Xavier initialization, ReLU activations, and early stopping. To further improve performance, future work could explore more expressive architectures, regularization techniques, or training with larger batch sizes to reduce noise in updates.

3. ## Result

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.123853
Final Test Loss:     0.123968
R² Score:           0.8763
Total Epochs Run:   500
```