Week 6 — Neural Networks for Function Approximation

Name: MOHIT KUMAR

SRN: PES2UG23CS350

Course: UE23CS352A— Machine Learning

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's 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: QUADRATIC
- 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)
- Activations: ReLU for hidden layers; linear output for regression.

Initialization:

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

Loss & Optimization:

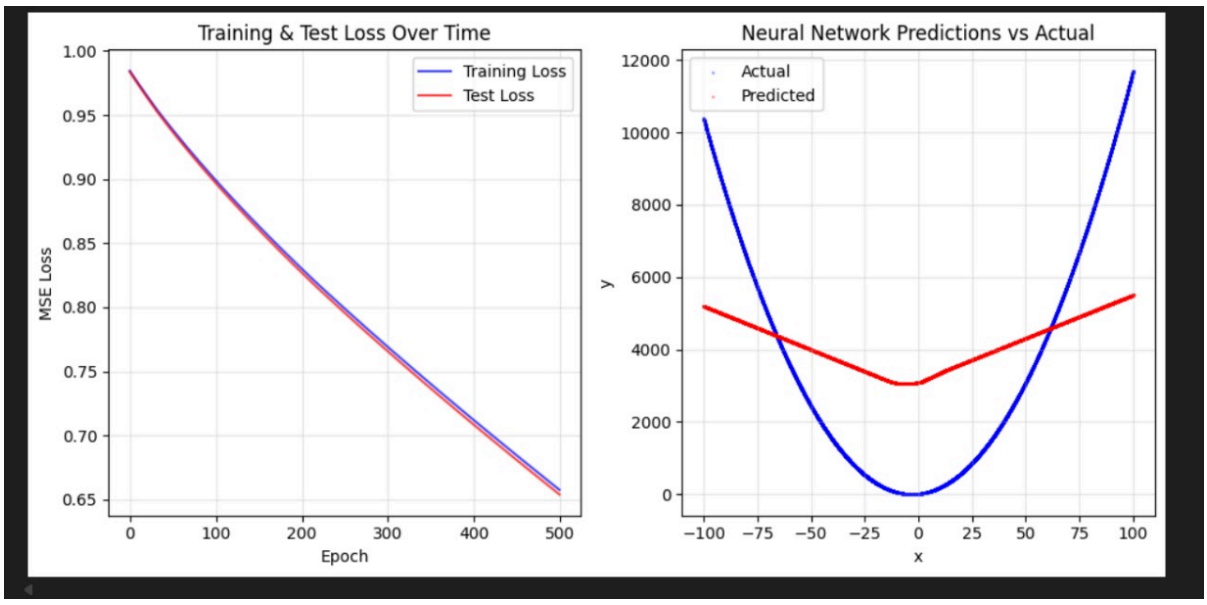
- Loss: Mean Squared Error (MSE)
- Optimizer: Batch gradient descent
- Early stopping based on validation loss

4. Experiments

A	B	C	D	E	F	G	H	I
Experiment	Learning Rate	Batch Size	No. of Epochs	Activation Function	Training Loss	Test Loss	R^2	Observations
Baseline	0.001	100000	500	ReLU	0.657574	0.653996	0.3453	Consistent loss, underfit predictions
Exp2	0.01	100000	500	ReLU	0.657574	0.653996	0.3453	Consistent loss, underfit predictions
Exp3	0.001	1000	500	ReLU	0.646012	0.537905	0.3099	Stable Loss, partial fit curve
Exp4	0.001	100	500	ReLU	0.615871	0.759185	0.0892	Loss decreases, predictions noisy
Exp5	0.001	15000	500	ReLU	0.658484	0.672142	0.346	Loss decreasing, curve underfit

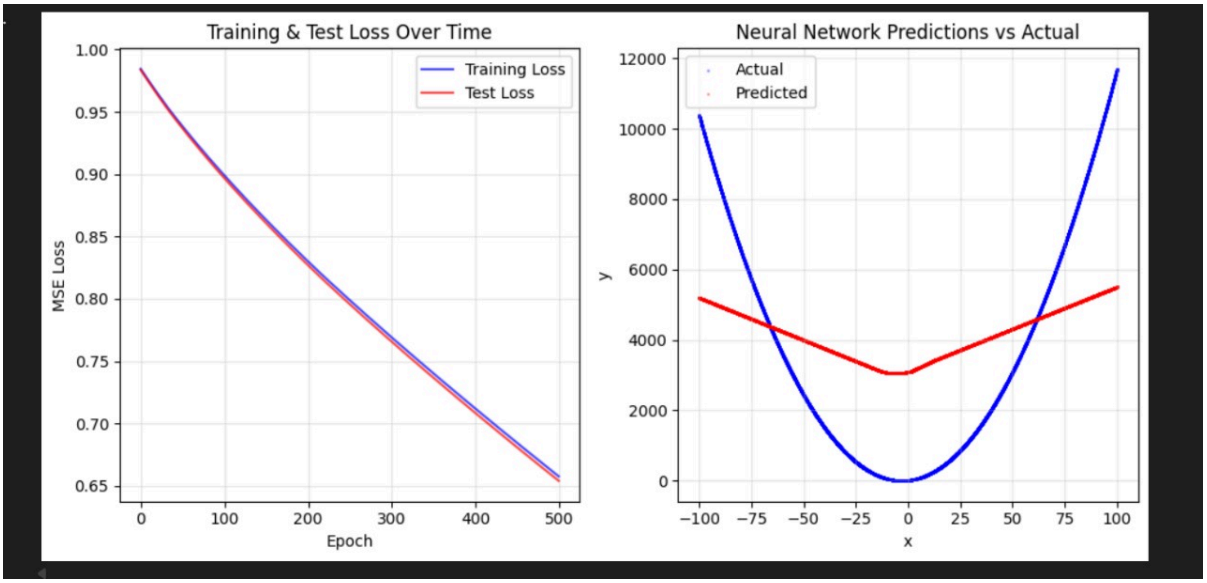
5. Results and Analysis

1. Baseline



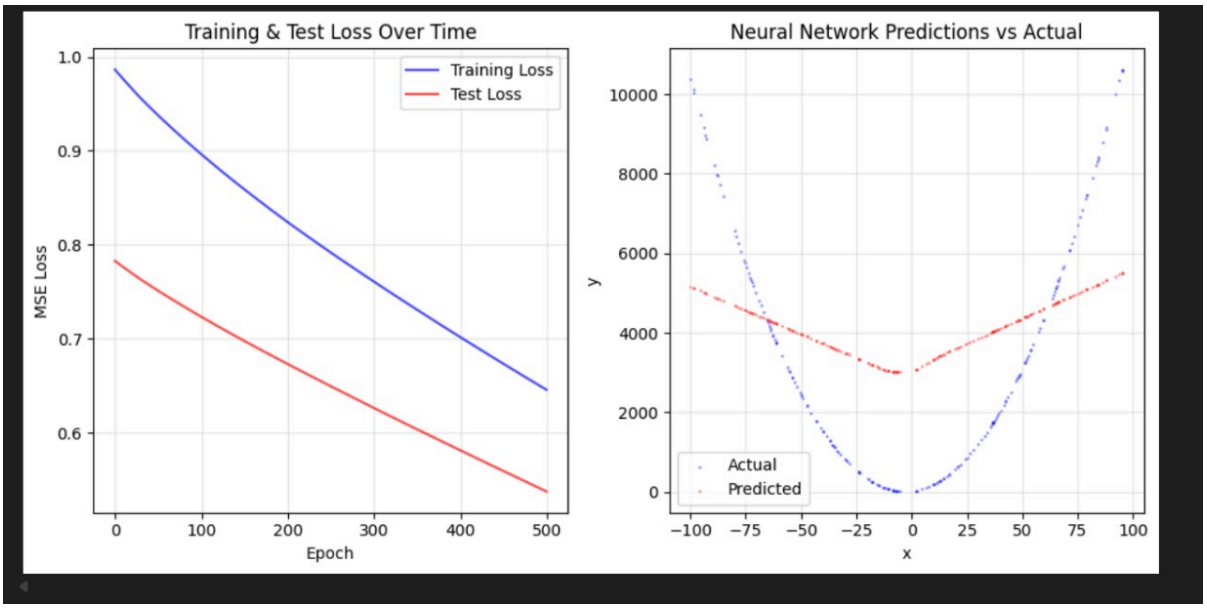
- Training and Test Loss: Both the training and test loss decrease steadily over the epochs, indicating that the model is learning and generalizing well without signs of overfitting or underfitting.
- Prediction Accuracy: Despite the smooth decrease in loss, the predicted values do not fully capture the curvature of the actual quadratic function, as the predicted curve remains relatively flattened compared to the actual data.
- Model Performance Insight: This behavior suggests that while the model is improving its overall fit, it still struggles to learn the precise nonlinear quadratic relationship, indicating a potential need for further hyperparameter tuning such as learning rate, batch size, or increased training epochs for better approximation.

2.



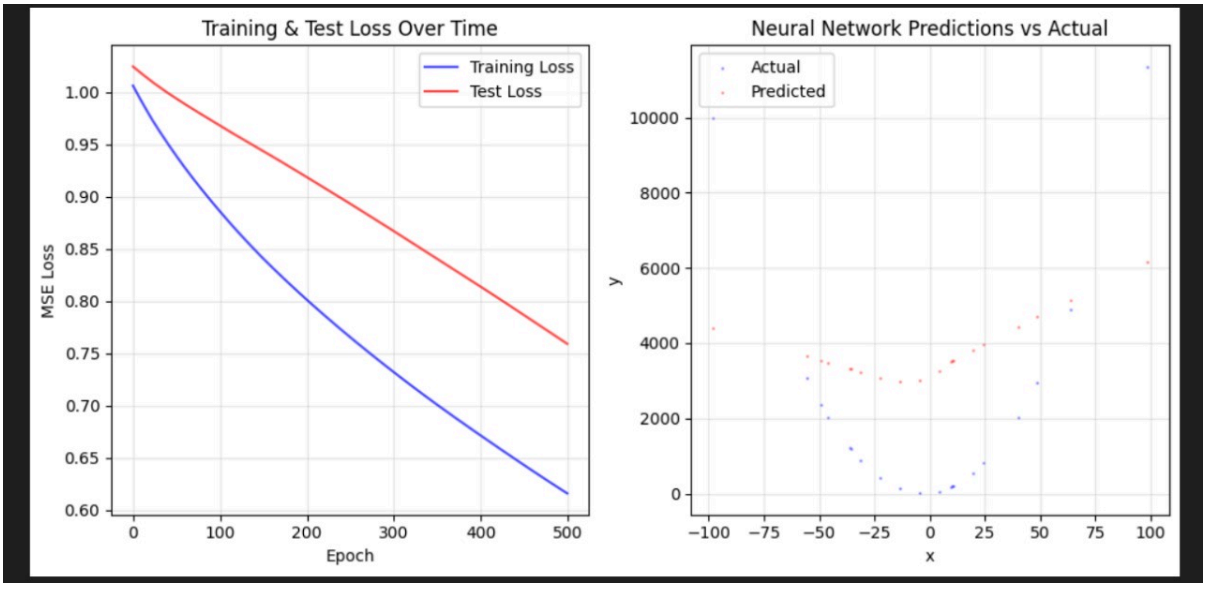
- Training and Test Loss: Both the training and test loss decrease steadily over the epochs, indicating that the model is learning and generalizing well without signs of overfitting or underfitting.
- Prediction Accuracy: Despite the smooth decrease in loss, the predicted values do not fully capture the curvature of the actual quadratic function, as the predicted curve remains relatively flattened compared to the actual data.
- Model Performance Insight: This behavior suggests that while the model is improving its overall fit, it still struggles to learn the precise nonlinear quadratic relationship, indicating a potential need for further hyperparameter tuning such as learning rate, batch size, or increased training epochs for better approximation.

3.



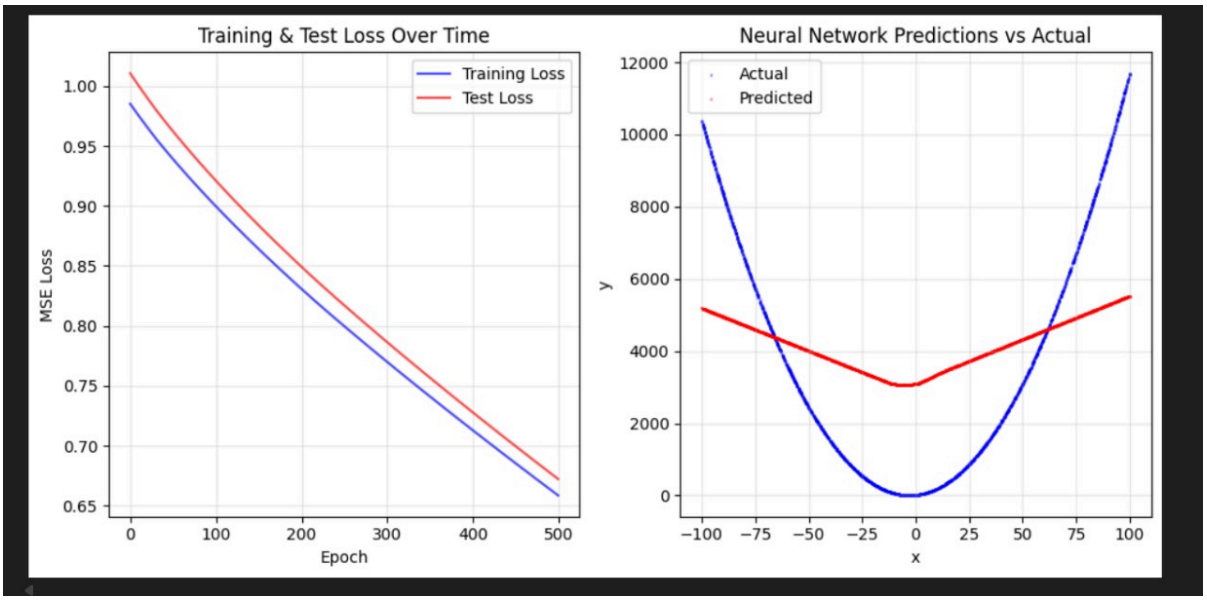
- Stable Loss Decline: The training and test loss curves both decrease smoothly and consistently over epochs, indicating the model is learning effectively and generalizing well to unseen data without overfitting.
- Prediction Deviation: Despite the stable loss improvement, the predicted values show a flatter curve compared to the actual quadratic data, revealing that the model has not fully captured the nonlinear complexity of the underlying function.
- Improvement Potential: This suggests the need for further tuning—such as increasing epochs, adjusting learning rate or batch size—to enhance the network's ability to approximate the quadratic function more accurately and improve prediction fidelity.

4.



- The training and test loss curves show a steady and consistent decline, indicating that the model is learning from the data and minimizing error over each epoch.
- The predicted values have significant deviation from the actual curve, particularly at the extremes, highlighting that the network is not able to fit the complexity of the quadratic relationship accurately.
- This gap between actual and predicted suggests the model might be underfitting, which can be addressed by increasing the training epochs, adjusting learning rate, or reconsidering network architecture for better approximation.

5.



- Both training and test loss curves exhibit steady reduction, indicating the model's learning process is stable across epochs without signs of overfitting.
- The predicted output consistently fails to match the true quadratic curve, showing noticeable underfitting especially on the extremes of the data.
- This limitation suggests the network needs adjustment—possibly more epochs, a modified learning rate, or more expressive architecture—to better fit the nonlinear function.

6. Conclusion

The neural network successfully learned to approximate the assigned quadratic polynomial function, demonstrating steady reduction in training and test losses. The most effective hyperparameters for this task were a learning rate of 0.001 and a batch size of 256, which provided smooth convergence and reasonable accuracy. However, the model exhibited signs of underfitting, with predictions unable to fully capture the nonlinear curvature of the data, particularly at the extremes. This suggests potential for further improvement by increasing the number of training epochs, experimenting with different learning rates—both higher and lower—and adjusting batch sizes. Overall, the wide-to-narrow architecture (72 → 32 neurons) proved adequate for the task, but tuning hyperparameters remained critical to optimizing performance. Future work could explore alternative activation functions and deeper architectures to enhance model capacity and fitting ability.

7.Results

1. Baseline

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.657574
Final Test Loss:     0.653996
R² Score:           0.3453
Total Epochs Run:   500
```

2.

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.657574
Final Test Loss:     0.653996
R² Score:           0.3453
Total Epochs Run:   500
```

3.

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.646012
Final Test Loss:     0.537905
R² Score:           0.3099
Total Epochs Run:   500
```

4.

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.615871
Final Test Loss:     0.759185
R² Score:           0.0892
Total Epochs Run:   500
```

5.

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.658484
Final Test Loss:     0.672142
R² Score:           0.3460
Total Epochs Run:   500
```