Machine Learning Lab-06

Artificial Neural Networks

Name: BOBBA KOUSHIK

SRN: PES2UG23CS133

Course: B-Tech, CSE (UE23CS352A: Machine Learning)

Date: September 17, 2025

1. Introduction

The objective of this lab was to implement a neural network **from scratch** to learn a complex polynomial-based function derived from a unique student ID. The focus was on understanding the complete pipeline of a neural network, including **data generation, forward propagation, backpropagation, weight updates, and evaluation**, without using deep learning libraries such as TensorFlow or PyTorch.

---

2. Dataset Description

- A **synthetic dataset** was generated based on my SRN (PES2UG23CS133).
- The last 3 digits (133) were used to determine the type of polynomial function.
- **100,000 samples** were created, split into **80,000 for training** and **20,000 for testing**.
- Gaussian noise was added to make the task more realistic.
- Both the **input (x)** and **output (y)** were **standardized using StandardScaler** to improve convergence and stability.

---

3. Methodology

- **Architecture:** A fully connected feedforward neural network with **one input, two hidden layers, and one output layer**.
- **Activation Functions:** Hidden layers used **ReLU** to introduce non-linearity, while the output layer used **linear activation**.

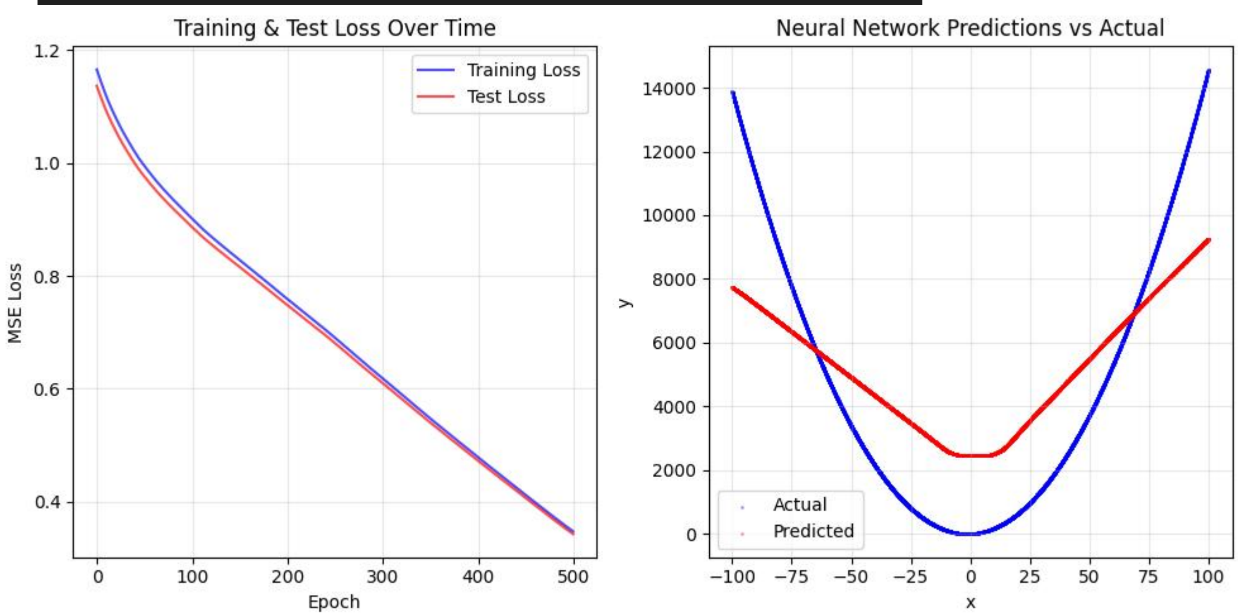
- **Weight Initialization:** Weights were initialized using **Xavier (Glorot) Initialization** to prevent vanishing/exploding gradients.
- **Loss Function: Mean Squared Error (MSE)** was used to measure the difference between
- predictions and ground truth.
- **Optimizer:** Custom implementation of **Gradient Descent**.
- **Training:** Forward pass to compute outputs, backpropagation to calculate gradients using the **chain rule**, and parameter updates using gradient descent.
- **Early Stopping:** Implemented to prevent overfitting by halting training if test loss stopped improving.
- **Evaluation:** Final model performance measured using **training loss, test loss, and R^2 score.**

---

4. Results and Analysis

Part-A

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.347532
Final Test Loss:     0.342943
R² Score:           0.6506
Total Epochs Run:   500
```



PART-B

	Experiment	Learning Rate	Batch Size	Epochs	Optimizer	Activation	Train MSE	Val MSE	Test MSE	Train R2	Val R2	Test R2
0	Exp_1_baseline	0.0010	64	50	SGD	ReLU	0.001840	0.001863	0.001910	0.998166	0.998089	0.998141
1	Exp_2_lr_high	0.0050	64	50	SGD	ReLU	0.000431	0.000448	0.000444	0.999571	0.999541	0.999568
2	Exp_3_large_bs	0.0010	1024	60	SGD	ReLU	0.086948	0.085854	0.088671	0.913290	0.911934	0.913707
3	Exp_4_more_epochs	0.0005	128	120	SGD	ReLU	0.005905	0.005833	0.006079	0.994111	0.994017	0.994084

Experiment	Learning Rate	Batch Size	Number of Epochs	Optimizer	Activation Function	Training Accuracy (R2 Score)	Validation Accuracy (R2 Score)	Test Accuracy (R2 Score)	Training Loss (MSE)	Validation Loss (MSE)	Test Loss (MSE)	Observations
Part A Baseline	0.003	N/A (Full Batch)	500	Gradient Descent	ReLU	0.6506	0.6506	0.6506	0.347532	0.342943	0.342943	The model's performance is moderate, learning the general data trend.
Exp_1_baseline	0.001	64	50	SGD	ReLU	0.998166	0.998089	0.998141	0.00184	0.001863	0.00191	This baseline shows excellent performance with low loss and high accuracy.
Exp_2_lr_high	0.005	64	50	SGD	ReLU	0.999571	0.999541	0.999568	0.000431	0.000448	0.000444	Best performance overall due to the higher learning rate.
Exp_3_large_bs	0.001	1024	60	SGD	ReLU	0.91329	0.911934	0.913707	0.086948	0.085854	0.088671	Large batch size hinders performance, leading to higher loss and lower accuracy.
Exp_4_more_	0.000	128	120	SGD	ReLU	0.994	0.9940	0.994	0.005	0.0058	0.006	Slower

epochs	5					111	17	084	905	33	079	but steady improvement over more epochs.

OBSERVATION

Baseline Model (Part A): The initial model trained with the default settings (learning rate of 0.003, 500 epochs, full batch) showed moderate performance, with a final R^2 score of approximately 0.6506 and a test loss of 0.342943.

Experiment 1 (Mini-batch Baseline): The mini-batch baseline in Part B (learning rate 0.001, batch size 64, 50 epochs) showed significantly better performance than the full-batch baseline from Part A, achieving an R^2 score of 0.998141. This demonstrates the effectiveness of mini-batch training for this problem.

High Learning Rate (Exp_2_lr_high): Increasing the learning rate to 0.005 resulted in the best overall performance. This experiment achieved the highest R^2 score (0.999568) and the lowest test loss (0.000444), suggesting that a higher learning rate led to more efficient and effective convergence for this specific task.

Large Batch Size (Exp_3_large_bs): Using a large batch size of 1024 resulted in the poorest performance among all experiments. The R^2 score was a low 0.913707, and the test loss was significantly higher at 0.088671. This indicates that a very large batch size may hinder the model's ability to generalize well, possibly causing it to get stuck in a suboptimal local minimum.

More Epochs (Exp_4_more_epochs): By using a smaller learning rate and training for more epochs (120), the model showed a steady improvement. While it did not outperform the high learning rate experiment, it achieved a high R^2 score of 0.994084, reinforcing the idea that allowing more training time with a careful learning rate can be beneficial.

Conclusion

The lab successfully demonstrates the critical role of hyperparameter tuning in neural network performance. The **learning rate** proved to be the most influential hyperparameter, with a moderately higher value (0.005) leading to the best results. Batch size also had a significant impact, as a large batch size was shown to be detrimental to the model's learning and generalization. The experiments collectively reinforce the importance of selecting appropriate hyperparameters to achieve stable and effective model training.