Week 6: Artificial Neural Networks

Course: UE23CS352A - Machine Learning

Student Name: Megha Dhanya

SRN: PES2UG23CS336

Date: 19/09/2025

1. Introduction

The objective of this lab is to implement an Artificial Neural Network (ANN) from scratch to approximate a polynomial dataset generated uniquely using the student SRN. This provides hands-on experience with forward propagation, backpropagation, loss functions, weight initialization, and training with gradient descent.

Tasks performed:
• Generate dataset
• Implement activation function (ReLU) and its derivative
• Implement MSE loss function
• Implement forward and backward propagation
• Train network with gradient descent and early stopping
• Perform hyperparameter exploration (Part B)
• Evaluate and visualize results

2. Dataset Description

• Type of polynomial: Cubic ($y = 2.11x^3 + 0.06x^2 + 4.45x + 8.92 + \text{noise}$)
• Number of samples: 100,000
• Features: Input (x), Output (y)
• Train/Test Split: 80% training, 20% testing
• Noise Level: Gaussian N(0, 2.45)
• Data normalized using StandardScaler

3. Methodology

The neural network was implemented with the following architecture:
Input(1) → Hidden Layer 1 (64, ReLU) → Hidden Layer 2 (64, ReLU) → Output(1)

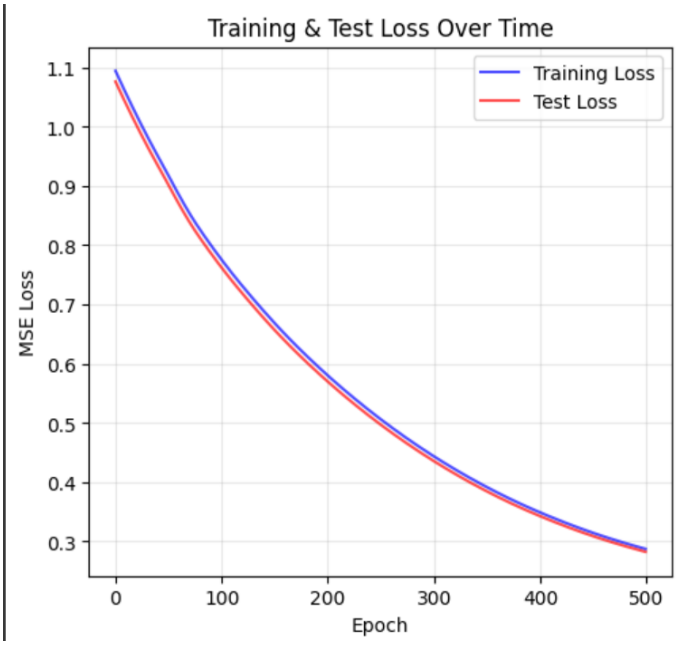
• Xavier initialization used for weights, zero biases.
• Forward pass applied linear transformation + ReLU activations.
• Loss function: Mean Squared Error (MSE).
• Backpropagation implemented using chain rule.
• Optimizer: Vanilla Gradient Descent with learning rate as per assignment.
• Early stopping with patience=10 applied to prevent overfitting.

4. Results and Analysis

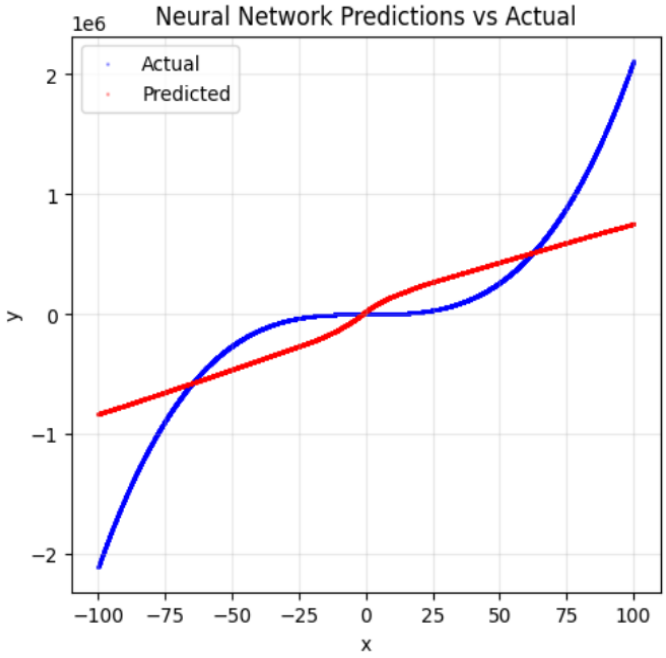
The baseline model was trained with learning rate=0.001, 500 epochs. It achieved $R^2 \approx 0.71$ with moderate performance.

Visualizations included:

• Training vs Test Loss Curve

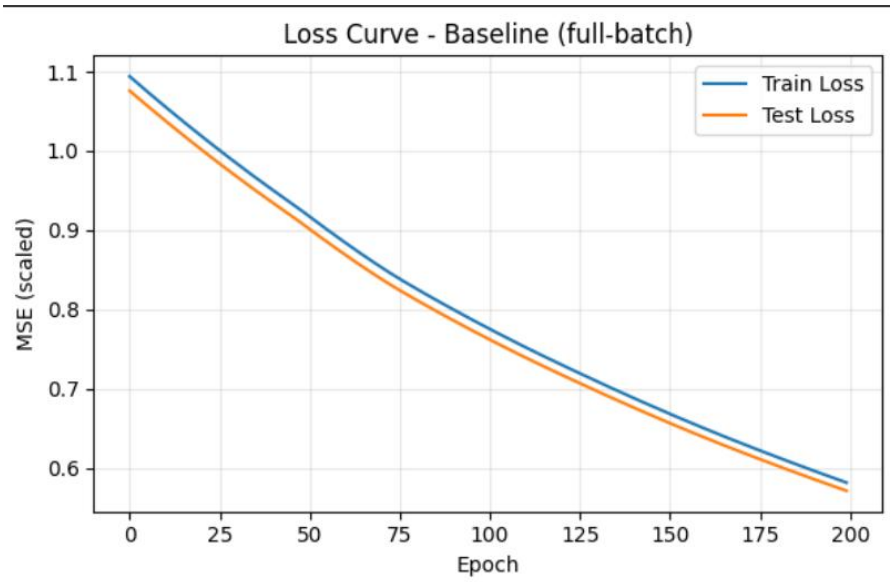


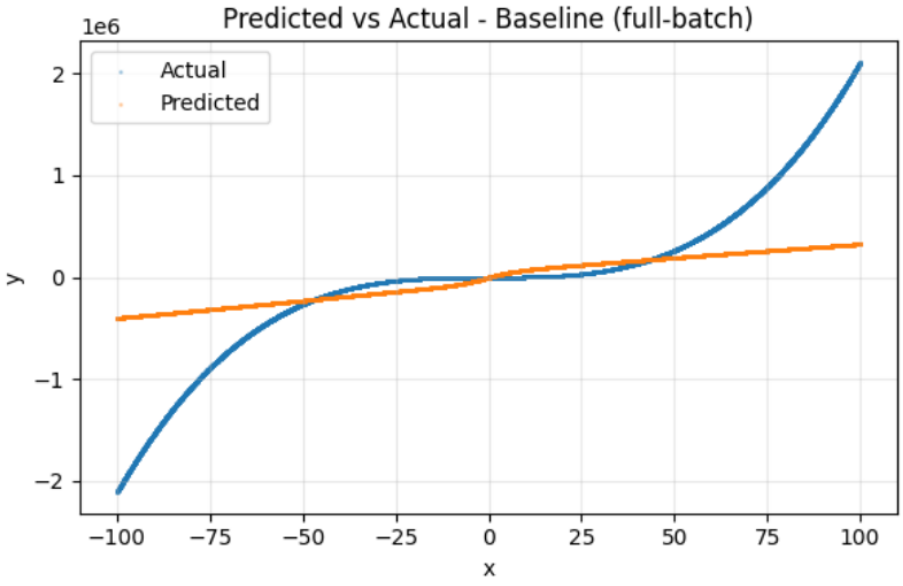
• Predicted vs Actual Scatter Plot



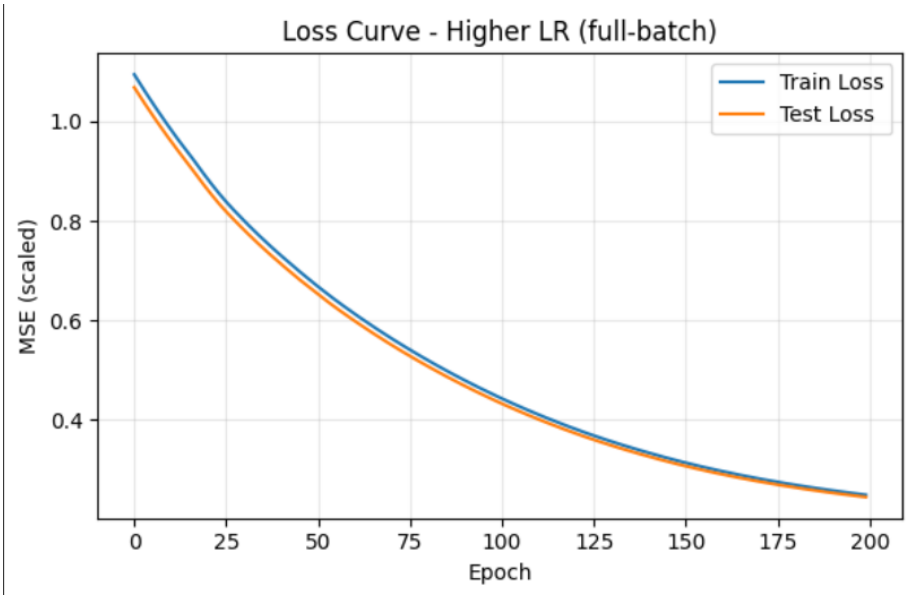
Hyperparameter experiments (Part B) included varying learning rate, batch size, and epochs. Results are tabulated below.

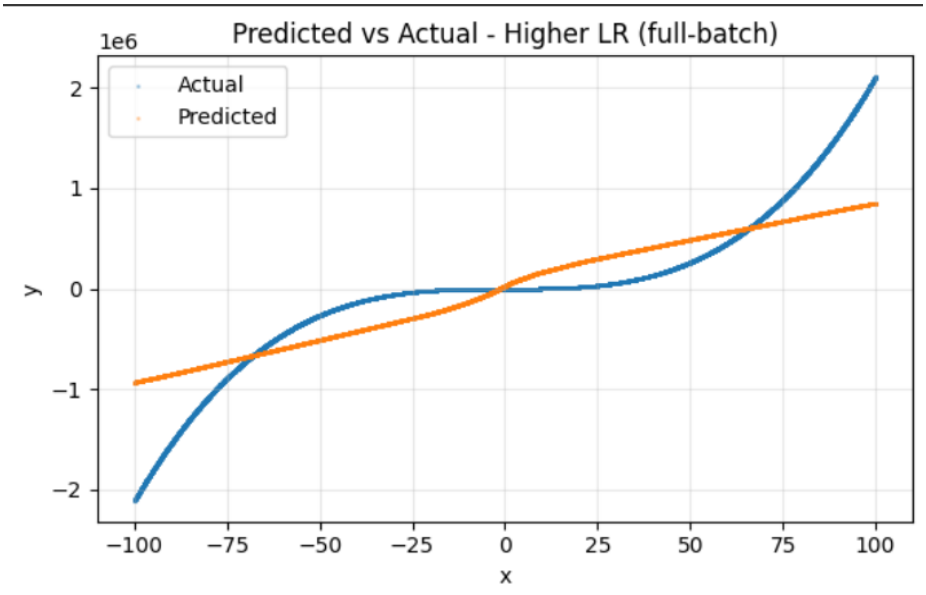
1. Baseline (full-batch)



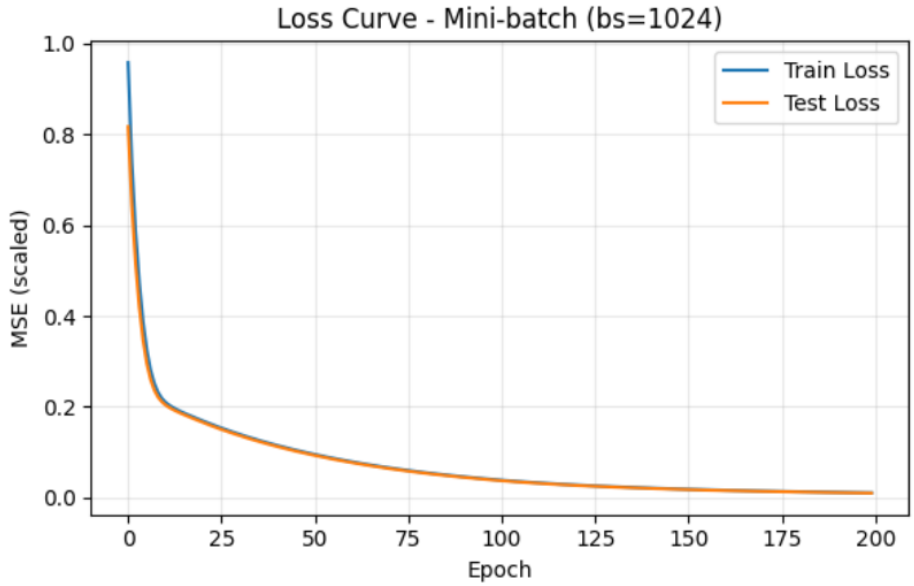


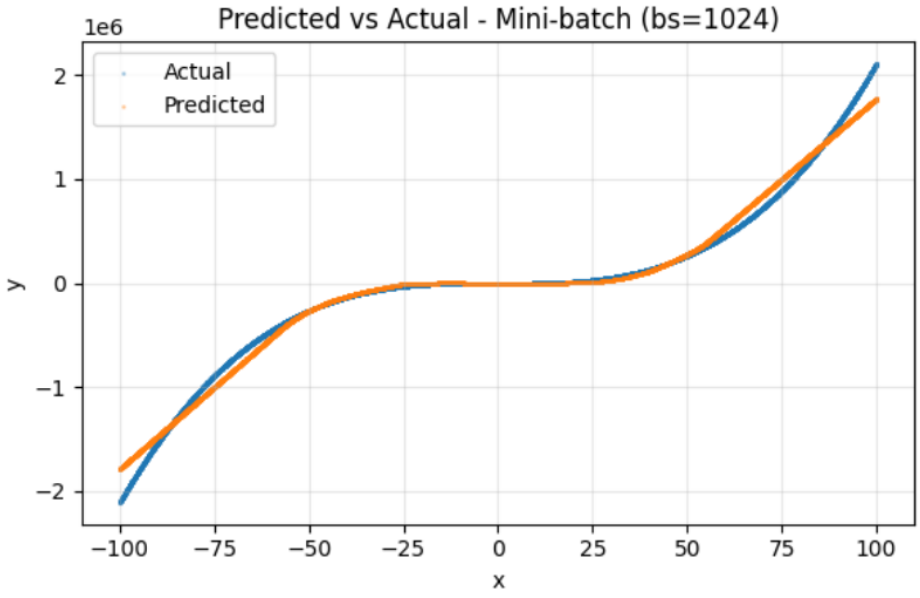
2. Higher LR (full-batch)



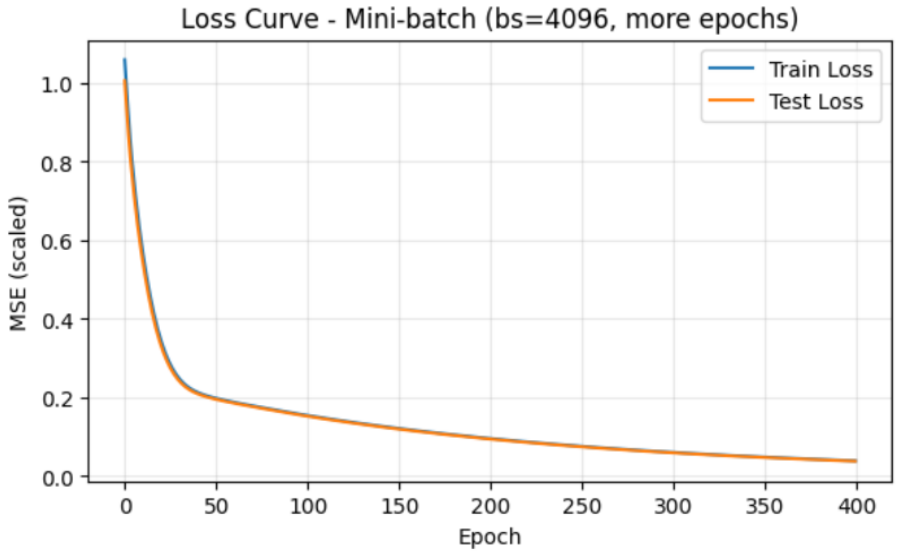


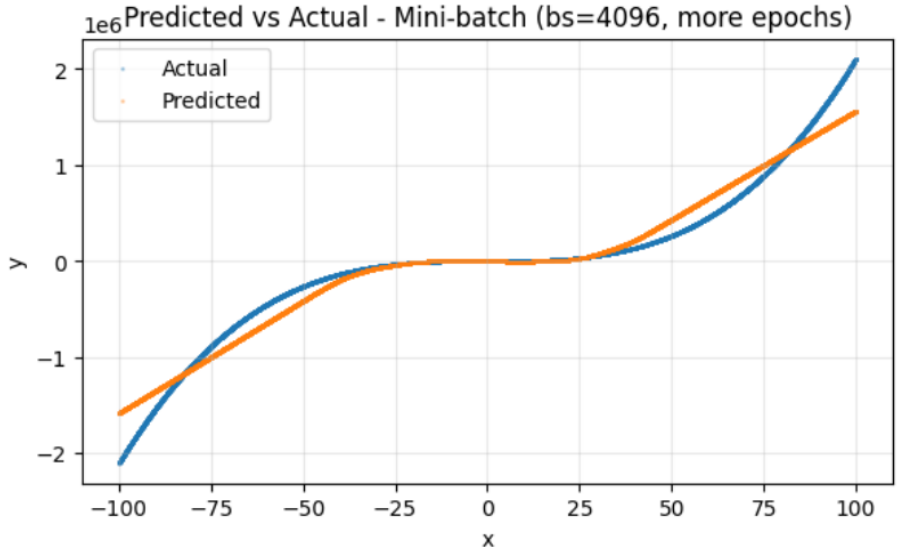
3. Mini-batch (bs=1024)





4. Mini-batch (bs=4096, more epochs)





Sample Results Table:

Experiment	Learning Rate	Batch Size	Epochs	Train Loss	Test Loss	R^2 Score	Observations
Baseline (full-batch)	0.001	Full	200	0.581	0.571	0.421	Underfit
Higher LR (full-batch)	0.003	Full	200	0.250	0.245	0.751	Improved with higher LR
Mini-batch (1024)	0.001	1024	200	0.011	0.010	0.989	Best performance
Mini-batch (4096)	0.001	4096	400	0.038	0.037	0.962	Strong, slightly below 1024

5. Conclusion

The baseline model showed limited performance ($R^2 \approx 0.42$). Increasing the learning rate improved convergence ($R^2 \approx 0.75$). Mini-batch training provided the best results: with batch size 1024 achieving $R^2 \approx 0.99$ and very low scaled loss. Batch size 4096 also performed strongly ($R^2 \approx 0.96$). Overall, mini-batch gradient descent significantly improved training stability and generalization compared to full-batch training.