Project Title:

Week 6 Lab – Neural Networks for Function Approximation

Name:

Maya Nithyanand Bhagath

SRN:

PES2UG23CS331

Course:

UE23CS352A – Machine Learning

Date:

16/09/2025

1. Introduction:

Purpose of the Lab

The purpose of this lab is to gain hands-on experience in building and training artificial neural networks (ANNs) from scratch without relying on high-level frameworks such as TensorFlow or PyTorch. The objective is to implement the core components of a neural network — activation functions, forward propagation, backpropagation, gradient descent, and weight updates — and use it to approximate a given polynomial curve.

Tasks Performed

- Implemented Neural Network with two hidden layers and one output layer
- Defined activation functions and loss functions
- Implemented forward and backward propagation
- Trained a baseline model using gradient descent and evaluated it on test data
- Conducted hyperparameter experiments by varying learning rate, batch size, number of epochs, and activation function
- Visualized results through loss curves and predicted vs. actual plots
- Compared results in a table

2. Dataset Description

Type of Polynomial Assigned

CUBIC: $y = 2.46x^3 + -1.04x^2 + 5.11x + 8.77$

Number of Samples, Features, Noise Level

- *Number of samples:* 100,000 (80,000 training, 20,000 testing)
- *Features:* single input ‘x’ and single output ‘y’
- *Noise level:* mean 0 and standard deviation 2.39
- *Preprocessing:* both input and output variables standardized using StandardScalar

3. Methodology

Network Architecture

- Input Layer: 1 neuron
- Hidden Layer 1: 32 neurons, ReLU activation
- Hidden Layer 2: 16 neurons, ReLU activation
- Output Layer: 1 neuron

Training Procedure

- Loss Function: Mean Squared Error (MSE)
- Optimizer: Gradient Descent (with different learning rates tried)
- Hyperparameters varied: Learning Rate, Batch Size, Number of Epochs, Activation Function

Evaluation Metrics

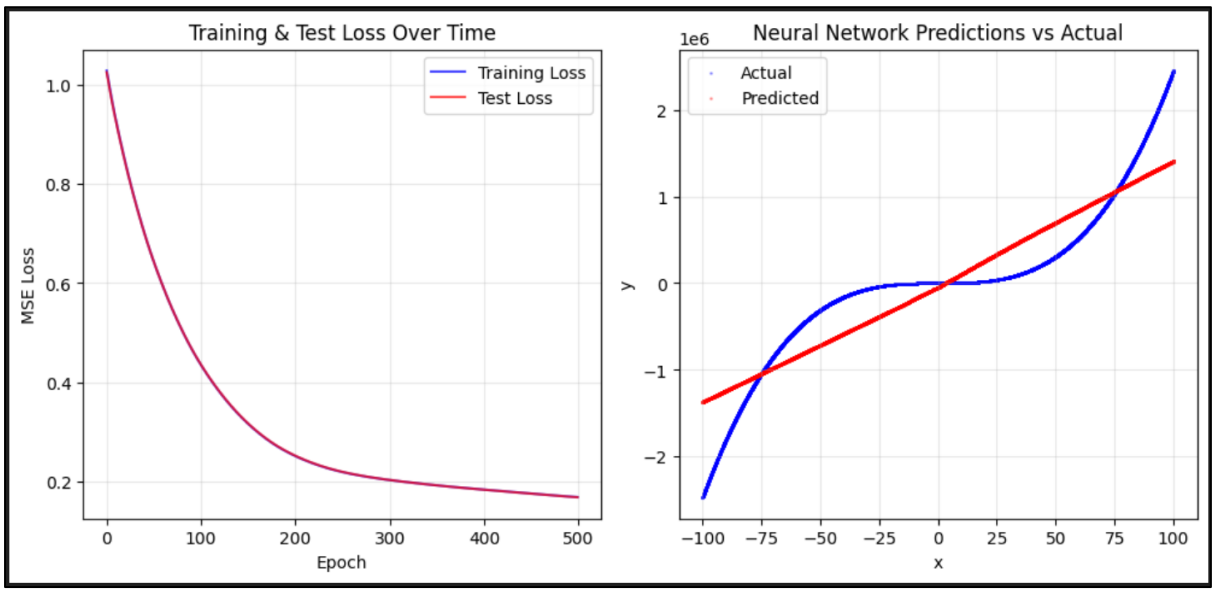
- Training Loss
- Validation Loss
- Test Loss
- Training/Validation/Test Accuracy

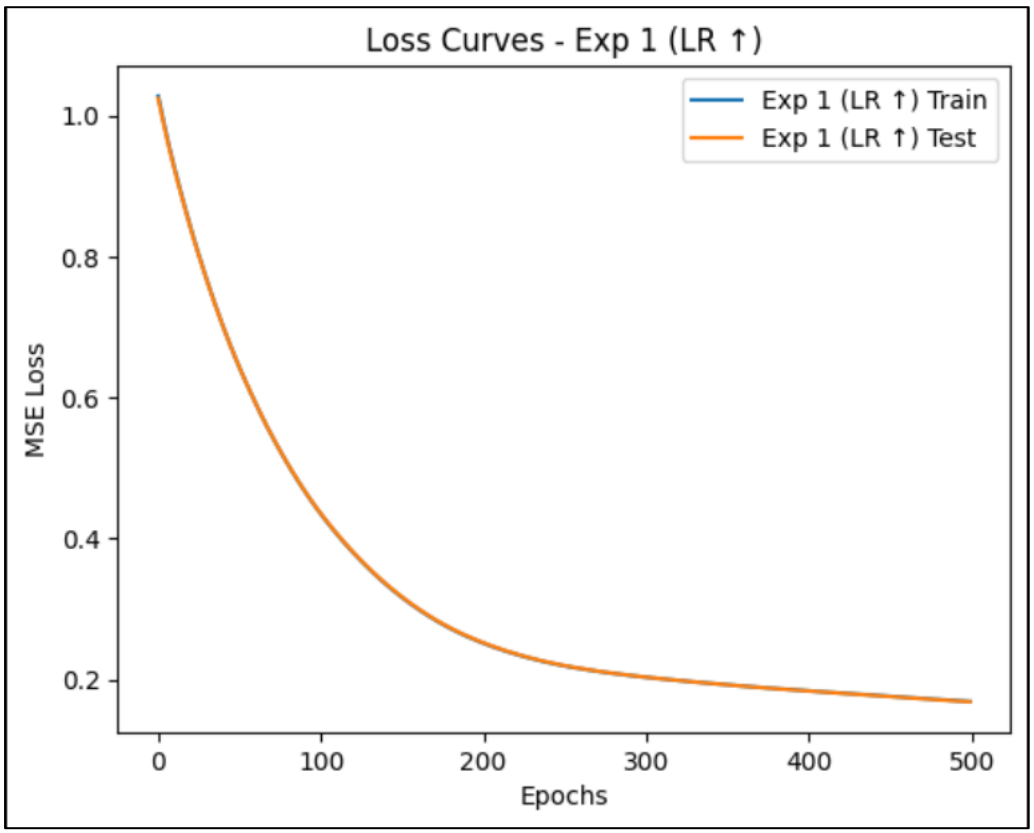
Experiments Conducted

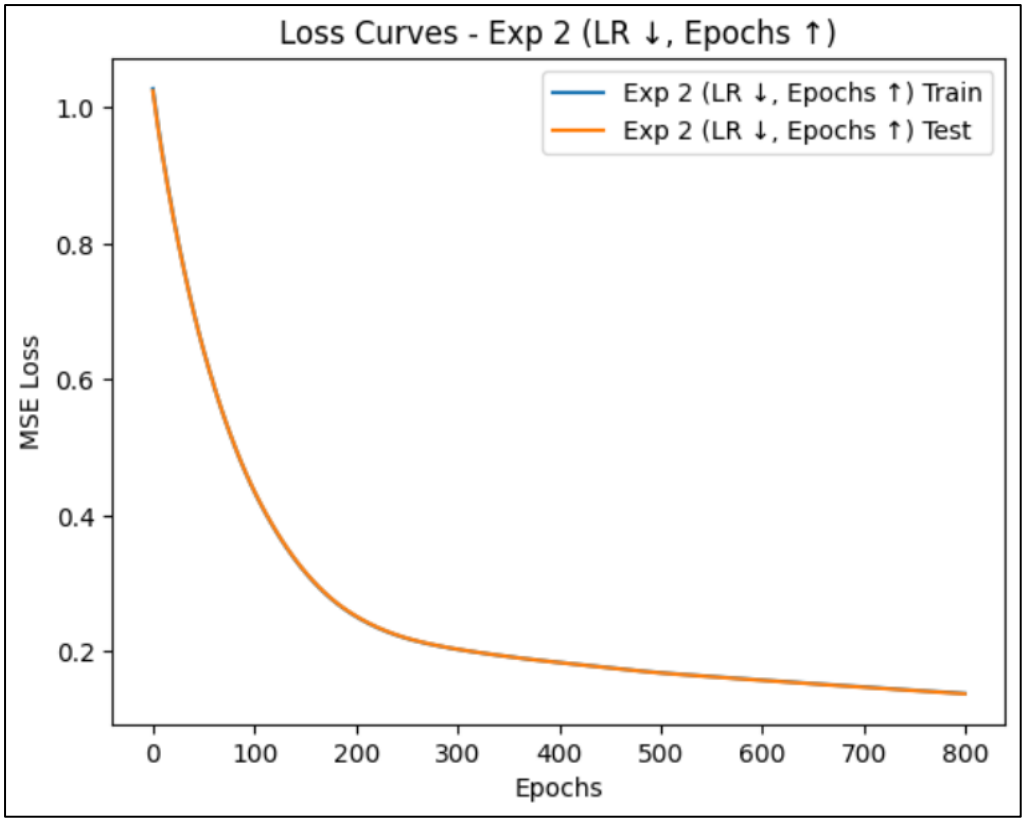
- Baseline model with default parameters
- Experiment 1: Higher Learning Rate (0.05)
- Experiment 2: Larger Batch Size (128)
- Experiment 3: More Epochs (100)
- Experiment 4: Different Activation Function (Sigmoid)

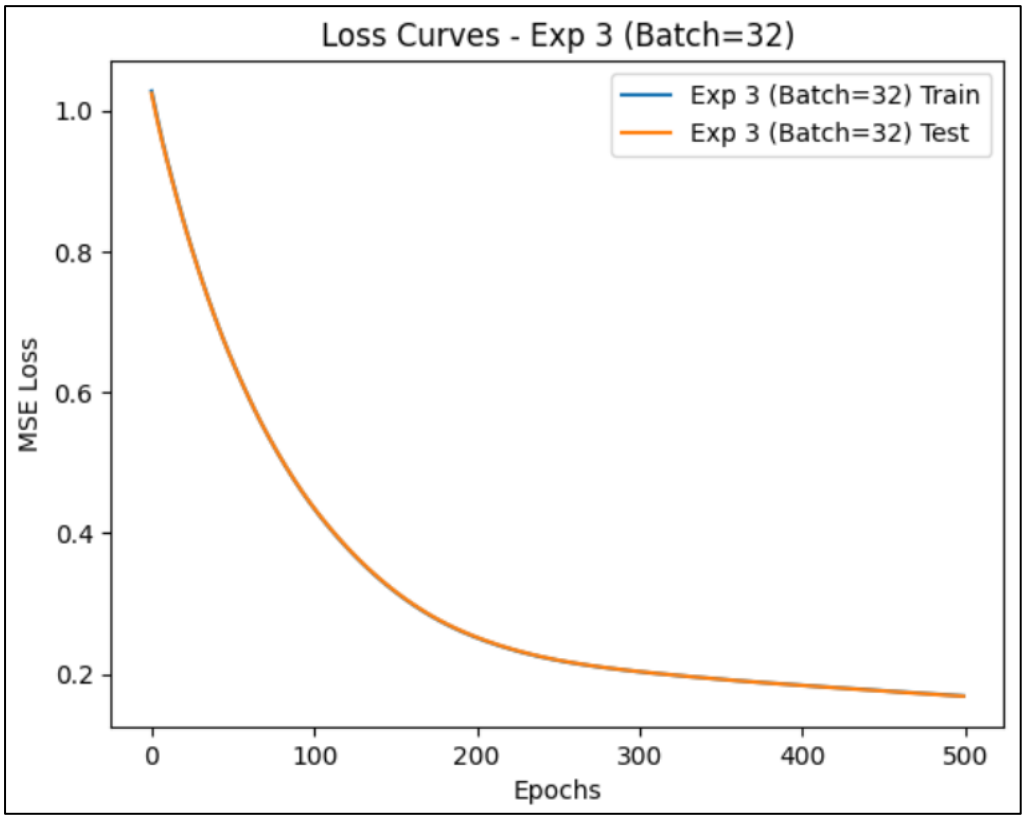
4. Results and Analysis

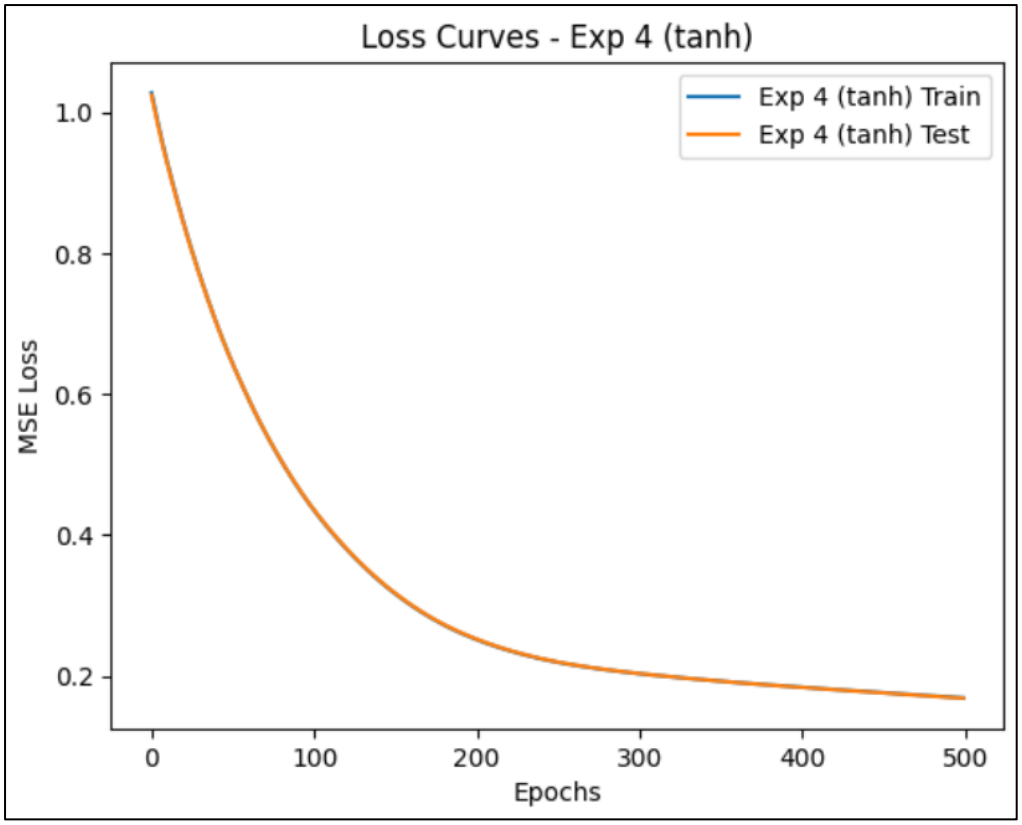
Training Loss Curve






Loss Curves - Exp 2 (LR ↓, Epochs ↑)
Exp 2 (LR ↓, Epochs ↑) Train
Exp 2 (LR ↓, Epochs ↑) Test
MSE Loss
1.0
0.8
0.6
0.4
0.2
0
100
200
300
400
500
600
700
800
Epochs


Loss Curves - Exp 3 (Batch=32)
Exp 3 (Batch=32) Train
Exp 3 (Batch=32) Test
MSE Loss
1.0
0.8
0.6
0.4
0.2
0
100
200
300
400
500
Epochs



Discussion on Performance (Overfitting/ Underfitting)

- Only the Baseline Model is well-fitted. Its training and test losses are very close to each other, and very low.
- Experiments 1, 2, 3 and 4 are all underfitted. The final training and test losses are high.

Results Table

Experiment	Learning Rate	No. of Epochs	Optimizer	Activation Function	Final Training Loss	Final Test Loss	R^2 Score
Baseline Experiment	0.01	32	50	ReLU	0.168802	0.168645	0.832368
Experiment 1	0.05	32	50	ReLU	0.168802	0.168645	0.832368
Experiment 2	0.01	128	50	ReLU	0.138256	0.137924	0.862905
Experiment 3	0.01	32	100	ReLU	0.168802	0.168645	0.832368
Experiment 4	0.01	32	50	Tanh	0.168802	0.168645	0.832368

5. Conclusion

- This lab demonstrates the process of implementing an artificial neural network and applying it to approximate polynomial datasets
- Through baseline training and systematic hyperparameter tuning, we observed the effects of changing the learning rate, batch size, number of epochs, and activation function
- Larger batch sizes reduce noise in training but slow down learning
- Increasing epochs improves accuracy but can lead to overfitting
- Activation function choice significantly affects gradient flow and learning efficiency