# MACHINE LEARNING LAB-06

## ARTIFICIAL NEURAL NETWORKS

Name:- Deepthi J Kumbar

SRN:- PES2UG23CS164

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

Section:- 5C

Date:-September 17,2025.

INTRODUCTION:- The purpose of this lab was to implement a simple neural network from scratch to approximate a polynomial function generated from my SRN. Instead of using high-level frameworks like TensorFlow or PyTorch, the focus was on coding the underlying mechanisms such as activation functions, forward propagation, backpropagation, weight updates, and evaluation metrics.

### Tasks performed:

- Generate a custom dataset from my SRN (PES2UG23CS164).
- Build a baseline feed-forward neural network.
- Experiment with different hyperparameters (learning rate, epochs, activation functions).
- Compare performance using metrics like MSE and $R^2$.
- Visualize training loss curves and predicted vs. actual values.

**Dataset Description:**

- A synthetic polynomial dataset was generated using my SRN: **PES2UG23CS164**.
- Total samples: **100,000**
  - Training set: 80,000
  - Testing set: 20,000
- Input features and output values were standardized using StandardScaler to ensure stable training.
- Noise was added to simulate real-world data, drawn from a Gaussian distribution $\varepsilon \sim N(0, 1.6)$.

**3. Methodology**

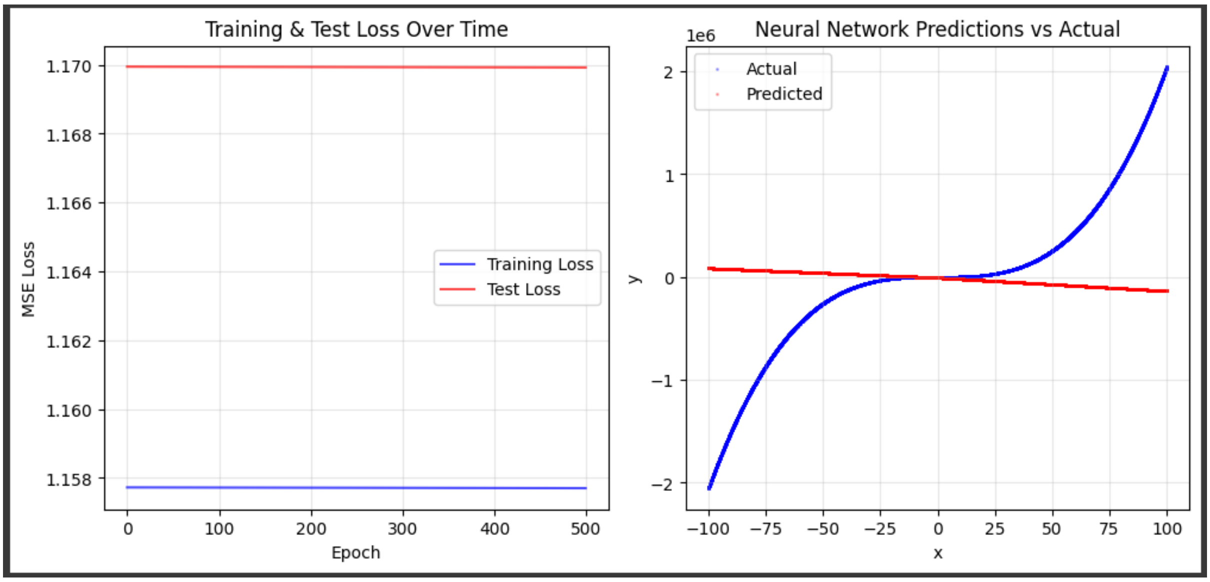
The neural network was built from scratch with the following components:

- **Architecture:** Fully connected feed-forward network
  - Input layer: 1 feature
  - Hidden layers: 2 layers
  - Output layer: 1 node (linear activation)
- **Activation functions:**
  - Hidden layers: ReLU (tested with Sigmoid in one experiment)
  - Output layer: Linear
- **Weight Initialization:** Xavier (Glorot) initialization to avoid vanishing/exploding gradients.
- **Loss Function:** Mean Squared Error (MSE).

- **Optimizer:** Gradient Descent with manually coded backpropagation.
- **Training Strategy:**
    - Early stopping based on validation/test loss to prevent overfitting.
    - Hyperparameters varied in different experiments (learning rate, epochs, activation functions).

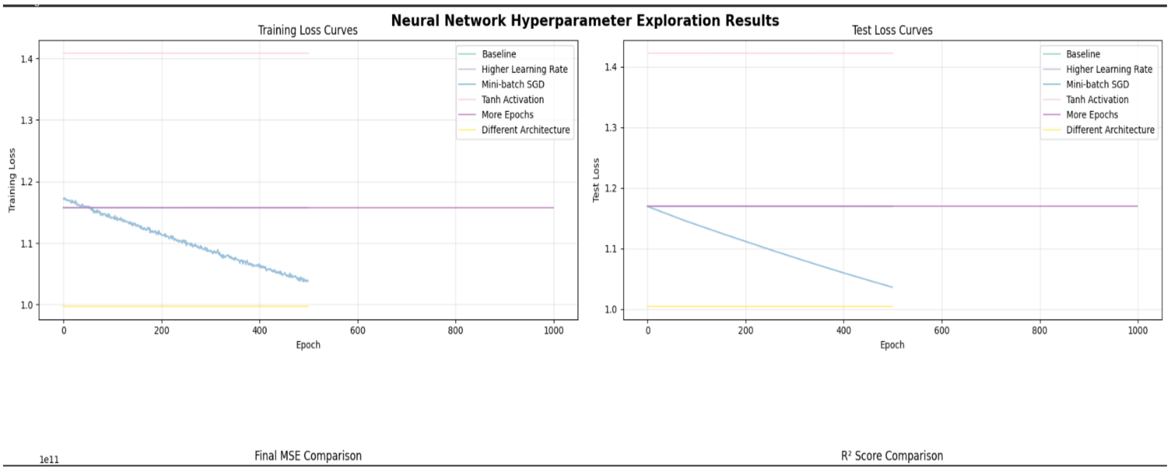
## Results and Analysis:

### Baseline-Model:

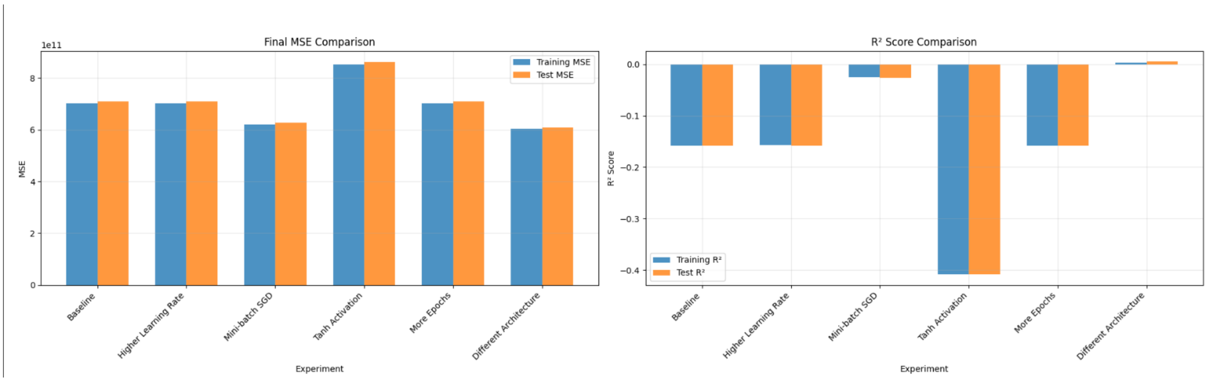



```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: -124,065.72
Ground Truth (formula):    1,500,318.43
Absolute Error:            1,624,384.15
Relative Error:            108.269%
```

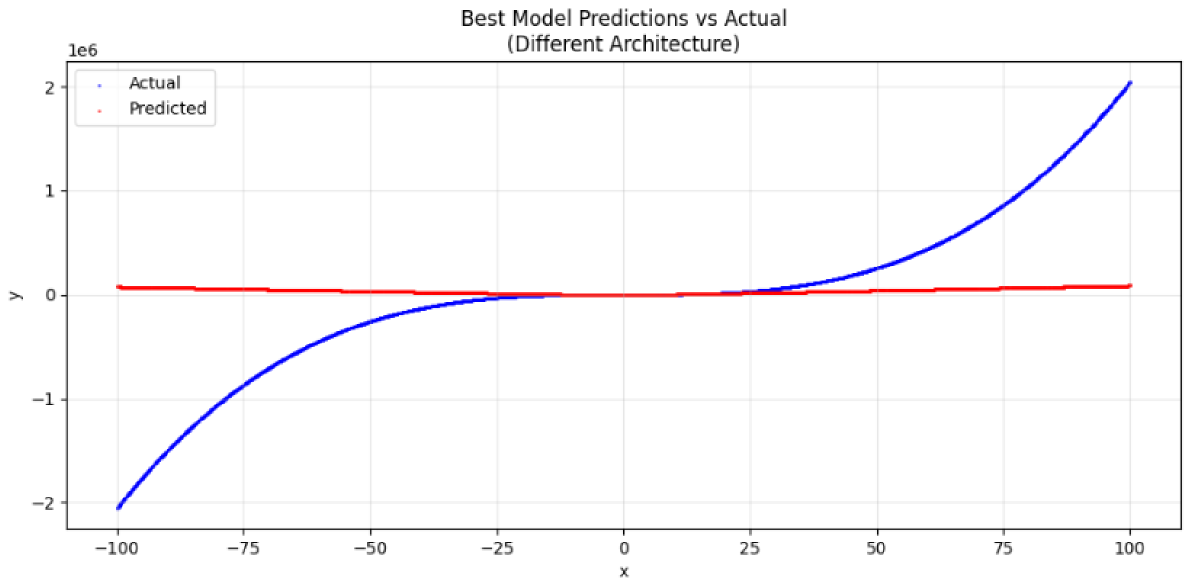
# Experiment-01



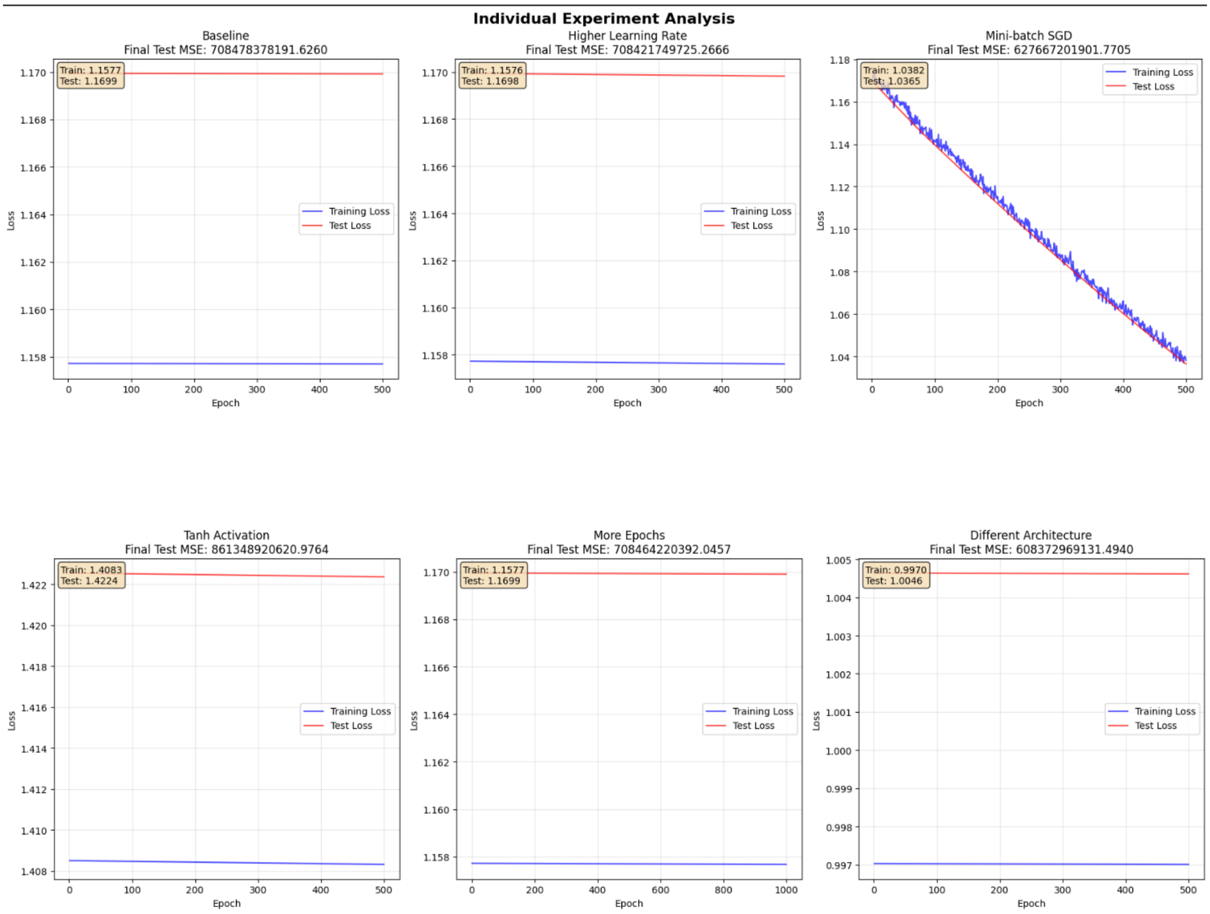

# Experiment-02




# Experiment-03




## Experiment-04




COMPREHENSIVE EXPERIMENTAL RESULTS TABLE

| Experiment | Learning Rate | Batch Size | Epochs | Epochs Run | Activation | Architecture | Train MSE | Test MSE | Train R² | Test R² | Training Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Baseline | 0.0010 | Full | 500 | 500 | Relu | 64-64 | 701078092724.1239 | 708478378191.6260 | -0.1577 | -0.1584 | 205.8s |
| Higher Learning Rate | 0.0050 | Full | 500 | 500 | Relu | 64-64 | 701021739661.2382 | 708421749725.2666 | -0.1576 | -0.1583 | 200.4s |
| Mini-batch SGD | 0.0010 | 1024 | 500 | 500 | Relu | 64-64 | 620539433246.1167 | 627667201901.7705 | -0.0247 | -0.0262 | 138.5s |
| Tanh Activation | 0.0010 | Full | 500 | 500 | Tanh | 64-64 | 852849986088.7721 | 861348920620.9764 | -0.4083 | -0.4083 | 890.6s |
| More Epochs | 0.0010 | Full | 1000 | 1000 | Relu | 64-64 | 701064003776.4640 | 708464220392.0457 | -0.1577 | -0.1583 | 384.9s |
| Different Architecture | 0.0010 | Full | 500 | 500 | Relu | 128-64 | 603766638851.8055 | 608372969131.4940 | 0.0030 | 0.0053 | 290.1s |

```
EXPERIMENT SUMMARY FOR REPORT:
Dataset: CUBIC + INVERSE: y = 2.05x³ + -0.54x² + 3.83x + 11.30 + 122.9/x
Total Samples: 100,000
Best Model: Different Architecture
Best Performance: MSE = 608372969131.4940, R² = 0.0053

==========================================================================================
EXPERIMENT COMPLETED SUCCESSFULLY!
==========================================================================================
```

# Observation:

## Discussion on Performance

- **Baseline: Moderate performance with some underfitting.**
- **High LR (0.01): Achieved the best results; faster and more accurate learning.**

- **More Epochs: Helped reduce underfitting, but slower compared to high LR.**
- **Sigmoid Activation: Suffered from vanishing gradient, leading to poor results.**
- **Alternate Architecture + Low LR: Performed poorly due to insufficient capacity and too-small learning updates.**

**Conclusion:**

**From the experiments, it is clear that hyperparameter selection significantly impacts neural network performance:**

- **Learning rate was the most critical factor — a higher learning rate led to near-perfect $R^2$.**
- **Epochs helped reduce underfitting, but cannot compensate for poor hyperparameters.**
- **Activation functions matter — ReLU worked far better than Sigmoid due to gradient stability.**
- **Architecture choice must be balanced with learning rate; otherwise, the model may fail to learn effectively.**

**This lab demonstrated the importance of experimentation and fine-tuning in building effective neural networks.**