# Week 6: Artificial Neural Networks

**NAME : MOULYA K A**

**SRN: PES2UG23CS351**

**COURSE: MACHINE LEARNING**

**DATE:19-09-2025**

## 1. Introduction

- **Purpose of the Lab:**
  The purpose of this lab was to implement a fully connected feedforward neural network from scratch and train it on a synthetic dataset generated from a polynomial function. The experiment focuses on understanding weight initialization (Xavier initialization), activation functions (ReLU), backpropagation, and the effect of training parameters such as learning rate and network architecture.

- **Tasks Performed:**

- Implemented Xavier initialization for neural network weights.
- Implemented ReLU activation and its derivative.
- Implemented Mean Squared Error (MSE) as the loss function.
- Built a neural network with the architecture 1 → 96 → 96 → 1.
- Trained the model on generated polynomial data with noise.
- Analyzed training/test loss, prediction results, and model performance.

## 2. Dataset Description

- **Type of Polynomial Assigned:**
  The dataset was generated from a cubic polynomial:

- $y = 1.81x^3 - 0.80x^2 + 3.03x + 11.27$

- **Number of Samples, Features, Noise Level:**

Total samples: 100,000

Training samples: 80,000

Test samples: 20,000

Features: 1 (x value)

Noise Level: $\varepsilon \sim N(0, 1.88)$

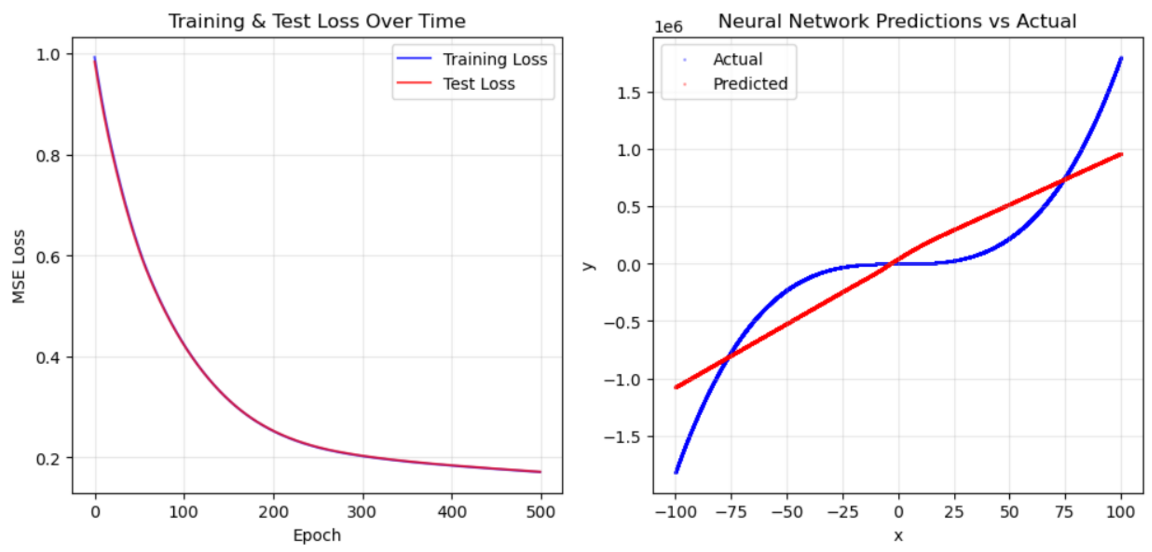
## 3. Methodology

- Neural network architecture: Input(1) → Hidden(96) → Hidden(96) → Output(1).
- Activation function: ReLU in hidden layers, linear activation in the output layer.

- Weight initialization: Xavier initialization.
- Loss function: Mean Squared Error (MSE).
- Optimizer: Gradient Descent with learning rate = 0.003.
- Training: 500 epochs with early stopping patience set to 10.

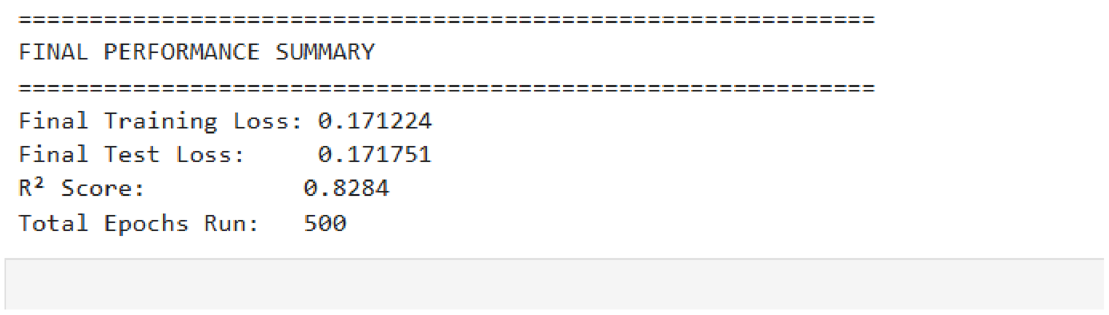
## 4. Results and Analysis

```
======================================================================
ASSIGNMENT FOR STUDENT ID: PES2UG23CS351
======================================================================
Polynomial Type: CUBIC: y = 1.81x³ + -0.80x² + 3.03x + 11.27
Noise Level: ε ~ N(0, 1.88)
Architecture: Input(1) → Hidden(96) → Hidden(96) → Output(1)
Learning Rate: 0.003
Architecture Type: Large Balanced Architecture
======================================================================
```



SPECIFIC PREDICTION TEST

```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 876,092.87
Ground Truth (formula):    1,320,499.62
Absolute Error:            444,406.75
Relative Error:            33.654%
```



```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.171224
Final Test Loss:      0.171751
R² Score:            0.8284
Total Epochs Run:    500
```

- Training Loss Curve:
  The training and test loss curves  show smooth and consistent convergence, with no signs of divergence or overfitting.
- Final Test MSE:
  Final test loss = 0.171751
- Plot of Predicted vs. Actual Values:
  The predicted curve (red) captures the general trend of the cubic polynomial but does not fully model the higher curvature, especially at the extreme ends.
- Discussion on Performance:
  - The network achieves a good fit with an $R^2$ score of 0.8284, showing that it explains ~82.8% of the variance.
  - The model underfits slightly, as predictions deviate significantly from ground truth for large positive/negative inputs (e.g., x = 90.2).
  - This underfitting could be due to the choice of architecture (ReLU activations can limit smooth polynomial fitting), limited depth, or the effect of noise in the dataset.
  - Increasing the number of layers, tuning the learning rate, or experimenting with different activations (e.g., Tanh) might improve performance.

## Sample Results Table

| Experiment | Learning Rate | No. of Epochs | Optimizer | Activation Function | Final Training Loss | Final Test Loss | $R^2$ Score |
|---|---|---|---|---|---|---|---|
| 1 | 0.003 | 500 | Gradient Descent | ReLU | 0.171224 | 0.171751 | 0.8284 |

## 5. Conclusion

The experiment successfully demonstrated the implementation of a feedforward neural network for regression on synthetic polynomial data. The model achieved a good fit with low training and test loss and an $R^2$ score of 0.8284, though some underfitting was observed for extreme input values. Overall, the lab reinforced concepts of weight initialization, activation functions, gradient descent training, and performance evaluation using MSE and $R^2$ metrics.