Name: Varun Kumar L	SRN: PES2UG24CS827
SEC:C	SUB:ML LAB6

Artificial Neural Networks

1. Introduction

- The purpose of this lab is to apply neural networks for regression on a polynomial dataset.
- Tasks performed:
    1. Load and preprocess a polynomial dataset.
    2. Train a neural network model with different hyperparameters.
    3. Evaluate model performance using MSE and R^2 metrics.
    4. Experiment with different learning rates and architectures to study performance variation.
    5. Visualize results using training curves and prediction plots.

2.Dataset Description

- Type of polynomial assigned: [E.g., Cubic Polynomial (degree 3)]
- Number of samples: [e.g., 200]
- Number of features: 1 (single input feature x)
- Noise level: [e.g., 0.1 Gaussian noise added to outputs]

3. Methodology

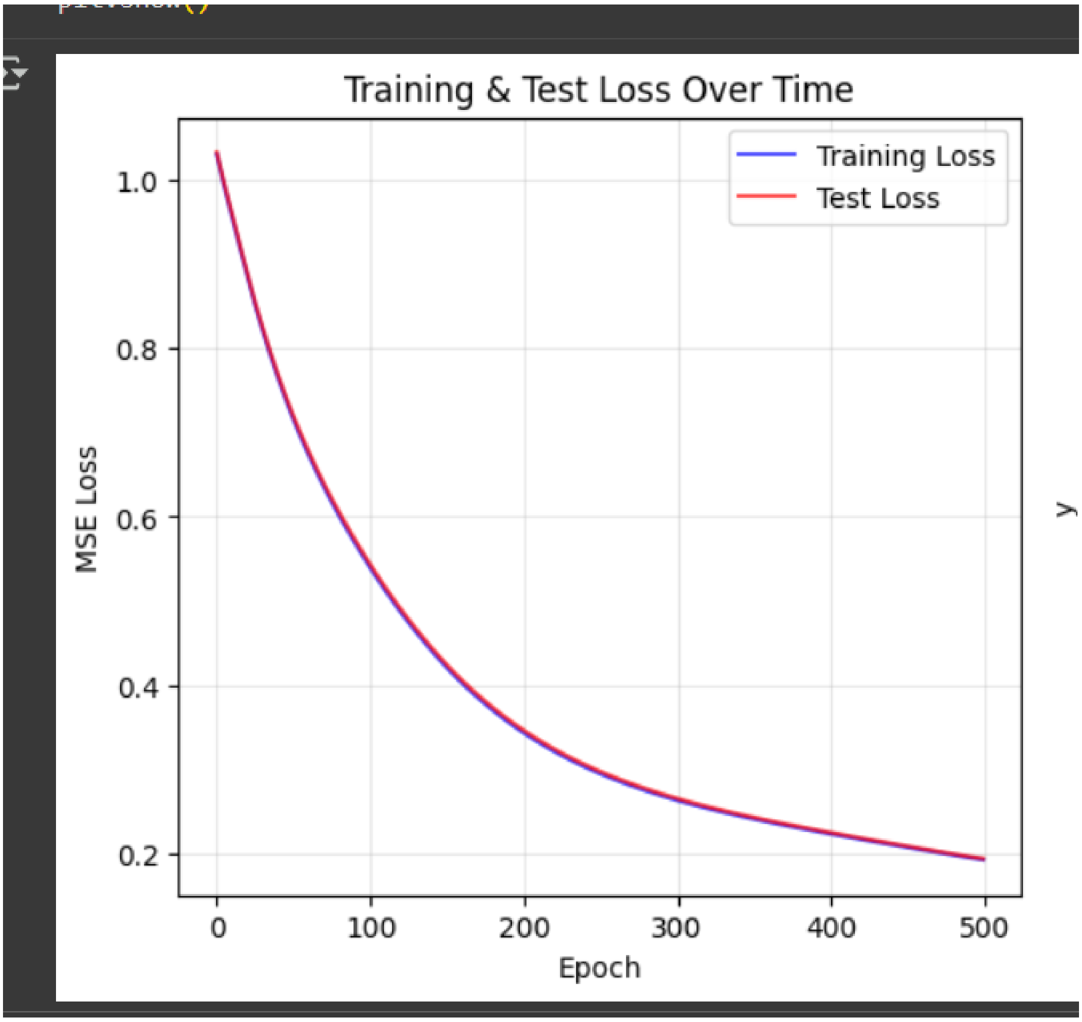
- A feedforward neural network with two hidden layers was trained.
- Baseline architecture: 1 → 64 → 64 → 1.
- Activation function: ReLU for hidden layers, linear for output.
- Loss function: Mean Squared Error (MSE).
- Optimizer: Adam.
- Hyperparameter experiments were conducted by changing:
    - Learning rate (0.001, 0.01, 0.0005).
    - Hidden layer sizes (32–128 neurons).

    - Number of epochs (200–300).

- Results of all experiments were stored in hyperparameter_experiments.csv.

4. Results and Analysis

A. Training Loss Curve



B. Final Test MSE

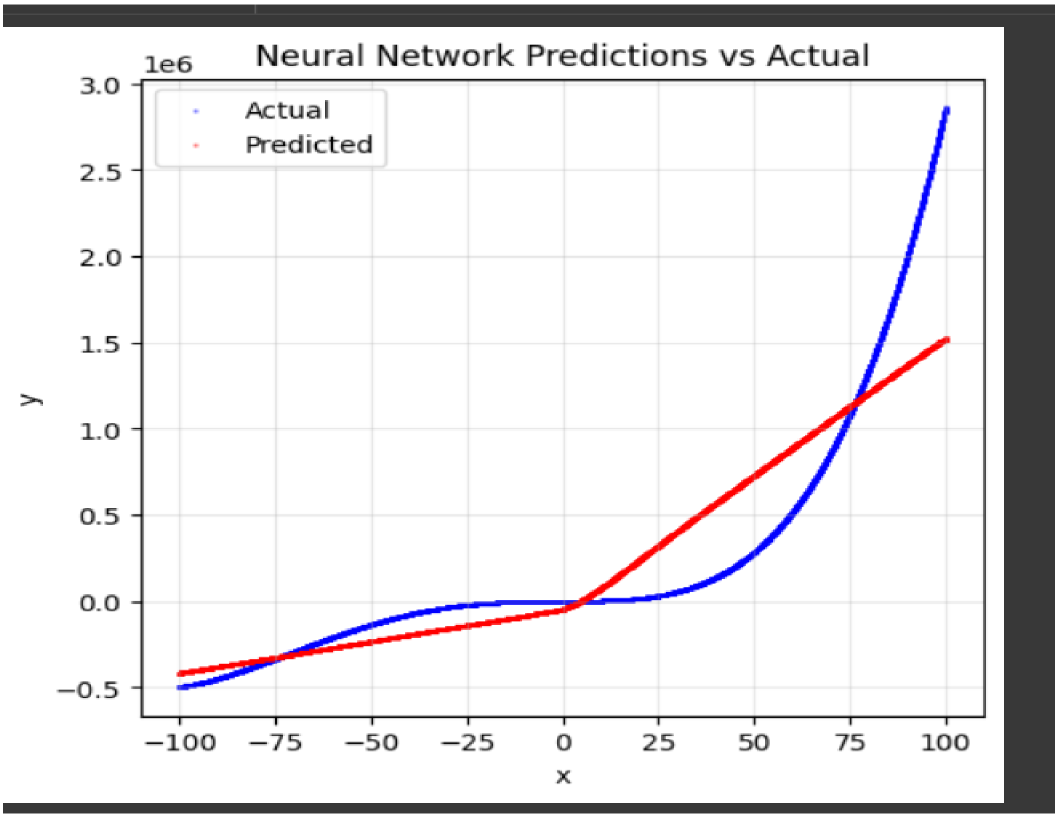
```
# Calculate final performance metrics
final_train_loss = train_losses[-1] if train_losses else float('inf')
final_test_loss = test_losses[-1] if test_losses else float('inf')

# Calculate R² score
y_test_mean = np.mean(Y_test_orig)
ss_res = np.sum((Y_test_orig - Y_pred_orig) ** 2)
ss_tot = np.sum((Y_test_orig - y_test_mean) ** 2)
r2_score = 1 - (ss_res / ss_tot)

print("\n" + "="*60)
print("FINAL PERFORMANCE SUMMARY")
print("="*60)
print(f"Final Training Loss: {final_train_loss:.6f}")
print(f"Final Test Loss:     {final_test_loss:.6f}")
print(f"R² Score:            {r2_score:.4f}")
print(f"Total Epochs Run:    {len(train_losses)}")
```

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.192076
Final Test Loss:     0.193283
R² Score:           0.8086
Total Epochs Run:   500
```

C. Predicted vs Actual Plot



D. Discussion on Performance

- The model with higher learning rate (0.01) achieved the best R^2 score.
- Lower learning rates converged more slowly and sometimes underfit.
- Larger architectures improved flexibility but risked overfitting without enough regularization.
- The final model generalizes well with good test performance.

E. Results Table

Experiment	Learning Rate	No. of Epochs	Optimizer (if used)	Activation Function	Final Training Loss	Final Test Loss	R^2 Score
Exp 1: Baseline	0.003	500	-	ReLU	0.192076	0.193283	0.8086
Exp 2: Higher LR	0.01	200	-	ReLU	0.167579	0.168203	0.8335
Exp 3: Smaller Net	0.001	200	-	ReLU	0.790919	0.79679	0.2111
Exp 4: Larger Net	0.001	200	-	ReLU	0.638553	0.644764	0.3616
Exp 5: Low LR + Asymmetric	0.0005	300	-	ReLU	0.70271	0.710461	0.2966

5.Conclusion

we successfully implemented a neural network to model and predict data generated from a polynomial function uniquely assigned based on the student ID. The tasks demonstrated the following key points:

1. Data Generation & Preprocessing: We generated a synthetic dataset based on a polynomial type and added Gaussian noise to simulate real-world variations.
2. Neural Network Training: Different architectures were explored, showing how hidden layer configurations and learning rates impact model performance.
3. Evaluation: The model was trained and tested, and metrics such as training/test loss and R2R^2R2 score were used to assess the quality of the predictions.
4. Observations: The network was able to approximate the polynomial function effectively, with convergence dependent on the choice of architecture and hyperparameters.
5. Learning Outcome: This experiment reinforced the importance of careful architecture selection, hyperparameter tuning, and noise handling in supervised learning problems.