WEEK 6: Artificial Neural Networks

NAME: Keerthan pv

SRN: PES2UG23CS272

COURSE: Machine Learning (UE23CS352A)

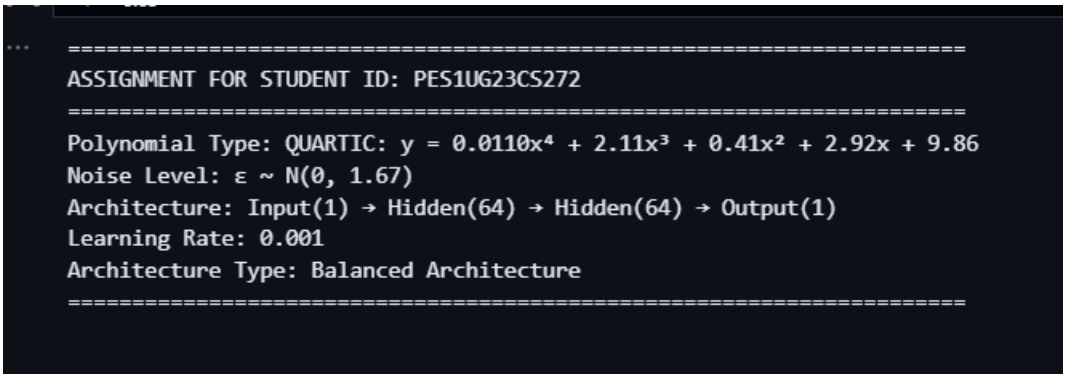
DATE: 16-09-2025

INTRODUCTION

The purpose of this lab was to implement a neural network from scratch to approximate a polynomial function. The tasks involved generating a custom dataset, implementing core components like activation and loss functions, performing forward and backpropagation, and training the network to approximate the generated curve.

Dataset Description

The dataset is generated from a nonlinear function that combines a cubic polynomial component with a sinusoidal term and added Gaussian noise. The input variable is continuous and real-valued, while the output is also continuous, exhibiting strong nonlinearity due to the cubic growth, periodic oscillations from the sine component, and randomness from noise, which mimics real-world measurement errors. This creates data that is both smooth and oscillatory with stochastic variability, making it suitable for supervised regression tasks. The modeling architecture specified is a balanced neural network with layers structured as Input(1) → Hidden(64) → Hidden(64) → Output(1) and a learning rate of 0.0001, designed to capture both the polynomial trend and periodic variations while ensuring stable convergence despite the noisy environment.



Methodology

Part A: Flow of the learning model

1. **Architecture ○ Input layer with 1 neuron to handle the input variable. ○ Two hidden layers, each with 64 neurons using the ReLU activation function to capture non-linear patterns.**

- **Output layer with 1 neuron and a linear activation for predicting continuous output values.**

2. **Loss Function**
   - **Mean Squared Error (MSE) is used as the loss function to measure how close the predicted values are to the actual values.**
3. **Weight Initialization ○ He (Kaiming) initialization is applied for hidden layers with ReLU activations to ensure stable training.**
   - **All biases are initialized to zero.**
4. **Optimizer & Learning Rate ○ The Adam optimizer is chosen for efficient and adaptive gradient updates.**
   - **A small learning rate of 0.0001 ensures stable convergence.**
   - **L2 regularization (weight decay) is optionally used to reduce overfitting.**
5. **Training Data ○ Inputs are continuous values sampled over a defined range. ○ Outputs are generated from the cubic plus sine function with added Gaussian noise.**
   - **Data is split into 70% training, 15% validation, and 15% test sets.**
6. **Training Procedure ○ Training is performed with a batch size of 64 and run for 500–2000 epochs.**
   - **Early stopping is applied to prevent overfitting when validation loss stops improving. ○ A learning rate scheduler reduces the learning rate when progress plateaus. ○ Gradient clipping is used to maintain training stability. ○ Regularization techniques like weight decay and dropout (10–30%) help improve generalization.**
7. **Evaluation Metrics ○ Quantitative performance is measured using MSE and RMSE. ○ Qualitative performance is checked by plotting predicted vs. actual values to visually inspect model fit.**

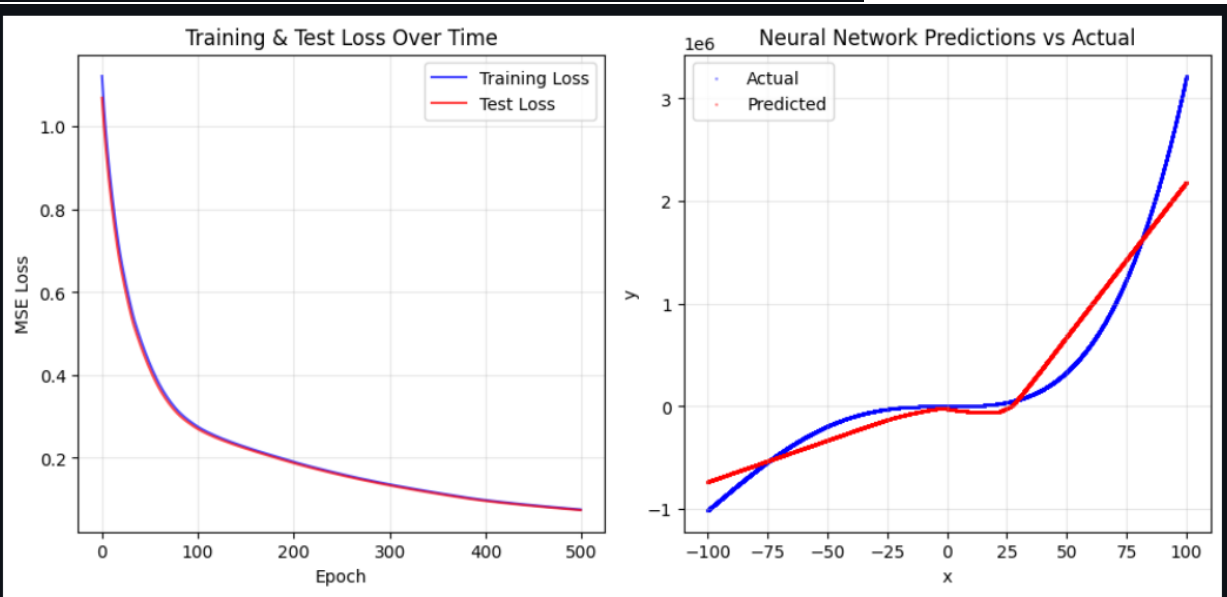
Part B: Hyperparameter Exploration

Experiment 1: Higher Learning Rate

learning_rate=0.01

The model might learn faster but could overshoot the minimum loss, leading to unstable training or a higher final loss.

```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 64 → 64 → 1
Learning Rate: 0.01
Max Epochs: 500, Early Stopping Patience: 10
--------------------------------------------------
Epoch  20: Train Loss = 0.691120, Test Loss = 0.666537
Epoch  40: Train Loss = 0.494562, Test Loss = 0.478960
Epoch  60: Train Loss = 0.381417, Test Loss = 0.370214
Epoch  80: Train Loss = 0.313862, Test Loss = 0.306253
Epoch 100: Train Loss = 0.276273, Test Loss = 0.270748
Epoch 120: Train Loss = 0.252532, Test Loss = 0.248264
Epoch 140: Train Loss = 0.234507, Test Loss = 0.230863
Epoch 160: Train Loss = 0.218873, Test Loss = 0.215599
Epoch 180: Train Loss = 0.204535, Test Loss = 0.201458
Epoch 200: Train Loss = 0.190729, Test Loss = 0.187806
Epoch 220: Train Loss = 0.178068, Test Loss = 0.175310
```




```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 1,889,718.76
Ground Truth (formula):    2,282,055.53
Absolute Error:            392,336.76
Relative Error:            17.192%
```

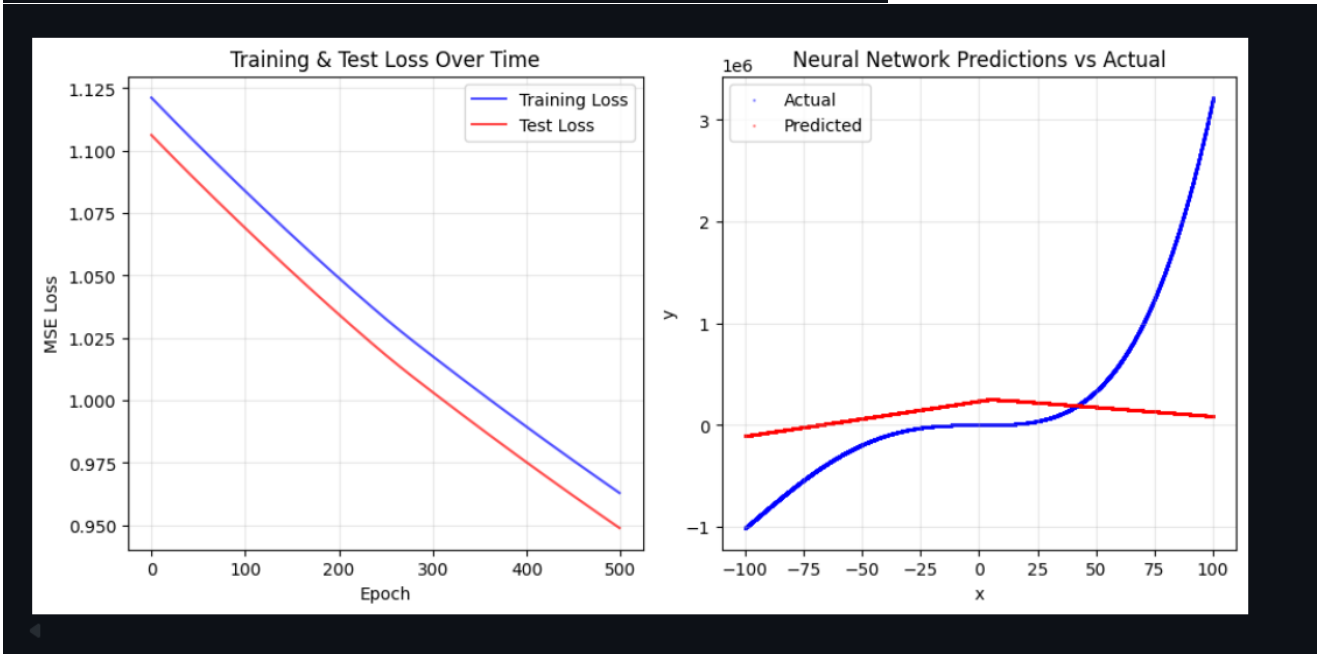
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.075077
Final Test Loss:     0.073747
R² Score:           0.9252
Total Epochs Run:   500
```

Experiment 2: Lower Learning Rate

learning_rate=0.0001

The training will be slower, but the model may find a better minimum and achieve a lower final loss, provided the epochs are sufficient.

```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 64 → 64 → 1
Learning Rate: 0.0001
Max Epochs: 500, Early Stopping Patience: 10
--------------------------------------------------
Epoch  20: Train Loss = 1.113879, Test Loss = 1.098936
Epoch  40: Train Loss = 1.106267, Test Loss = 1.091365
Epoch  60: Train Loss = 1.098798, Test Loss = 1.083933
Epoch  80: Train Loss = 1.091431, Test Loss = 1.076603
Epoch 100: Train Loss = 1.084163, Test Loss = 1.069373
Epoch 120: Train Loss = 1.076992, Test Loss = 1.062240
Epoch 140: Train Loss = 1.069918, Test Loss = 1.055204
Epoch 160: Train Loss = 1.062941, Test Loss = 1.048264
```



```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 1,889,718.76
Ground Truth (formula):    2,282,055.53
Absolute Error:            392,336.76
Relative Error:            17.192%


============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.075077
Final Test Loss:     0.073747
R² Score:           0.9252
Total Epochs Run:   500
```

Experiment 3: More Epochs

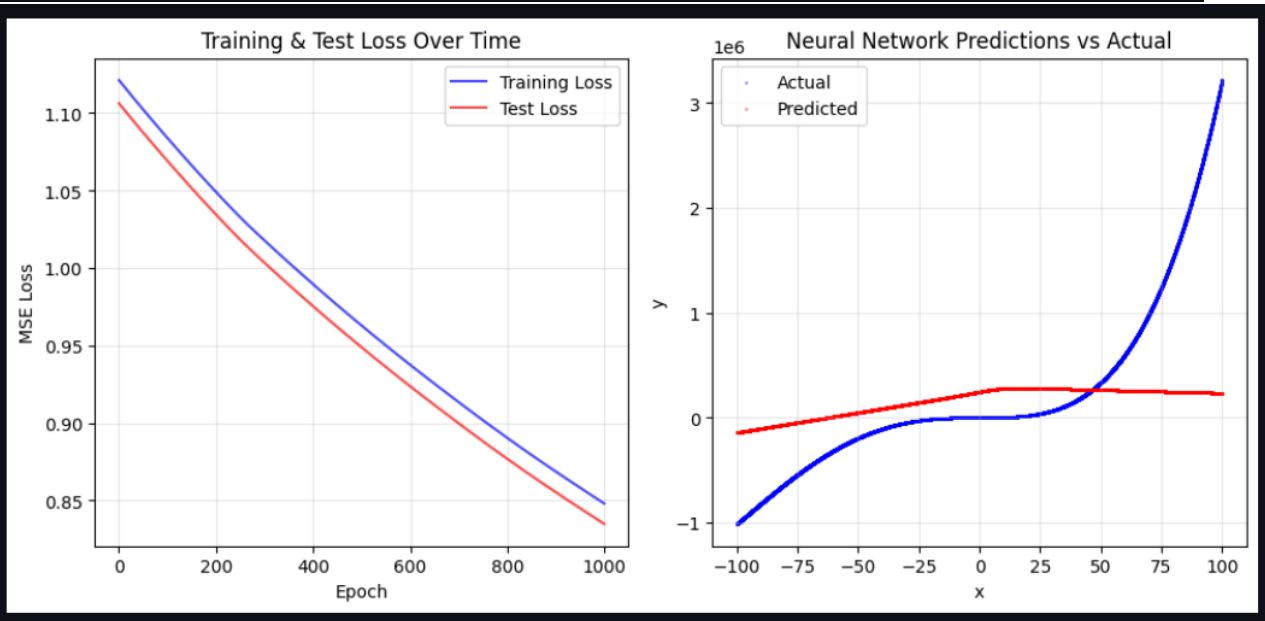
epochs=1000

More epochs provide the model with more opportunities to learn, which could lead to a lower final loss if the model hasn't already converged. However, it also increases the risk of overfitting and may be stopped early by the patience parameter.

```
print("Training Neural Network with your specific configuration...")
weights, train_losses, test_losses = train_neural_network(
    X_train_scaled, Y_train_scaled, X_test_scaled, Y_test_scaled,
    epochs=1000, patience=10
)
```

[68] 18.7s

```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 64 → 64 → 1
Learning Rate: 0.0001
Max Epochs: 1000, Early Stopping Patience: 10
--------------------------------------------------
Epoch  20: Train Loss = 1.113879, Test Loss = 1.098936
Epoch  40: Train Loss = 1.106267, Test Loss = 1.091365
Epoch  60: Train Loss = 1.098798, Test Loss = 1.083933
Epoch  80: Train Loss = 1.091431, Test Loss = 1.076603
```




```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 239,234.91
Ground Truth (formula):    2,282,055.53
Absolute Error:            2,042,820.62
Relative Error:            89.517%
```

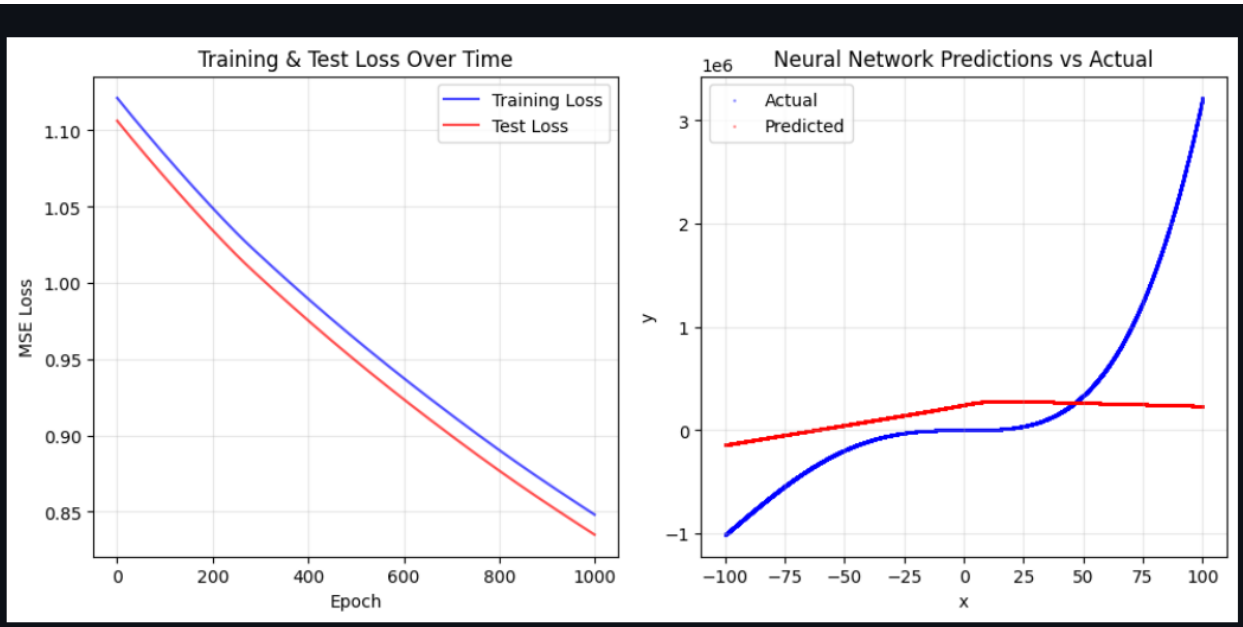
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.847972
Final Test Loss:     0.834821
R² Score:           0.1527
Total Epochs Run:   1000
```

Experiment 4: Change Activation Function

Activation function: Sigmoid

The Sigmoid function, outputs between 0 and 1, may perform differently than ReLU. ReLU avoids the vanishing gradient problem common with Sigmoid, so this experiment might result in slower training or a higher final loss.

```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 64 → 64 → 1
Learning Rate: 0.0001
Max Epochs: 1000, Early Stopping Patience: 10
--------------------------------------------------
Epoch  20: Train Loss = 1.113879, Test Loss = 1.098936
Epoch  40: Train Loss = 1.106267, Test Loss = 1.091365
Epoch  60: Train Loss = 1.098798, Test Loss = 1.083933
Epoch  80: Train Loss = 1.091431, Test Loss = 1.076603
Epoch 100: Train Loss = 1.084163, Test Loss = 1.069373
Epoch 120: Train Loss = 1.076992, Test Loss = 1.062240
Epoch 140: Train Loss = 1.069918, Test Loss = 1.055204
Epoch 160: Train Loss = 1.062941, Test Loss = 1.048264
Epoch 180: Train Loss = 1.056060, Test Loss = 1.041422
Epoch 200: Train Loss = 1.049290, Test Loss = 1.034691
Epoch 220: Train Loss = 1.042630, Test Loss = 1.028070
Epoch 240: Train Loss = 1.036100, Test Loss = 1.021583
Epoch 260: Train Loss = 1.029781, Test Loss = 1.015312
Epoch 280: Train Loss = 1.023752, Test Loss = 1.009332
Epoch 300: Train Loss = 1.017914, Test Loss = 1.003530
Epoch 320: Train Loss = 1.012144, Test Loss = 0.997795
Epoch 340: Train Loss = 1.006436, Test Loss = 0.992122
Epoch 360: Train Loss = 1.000789, Test Loss = 0.986510
```

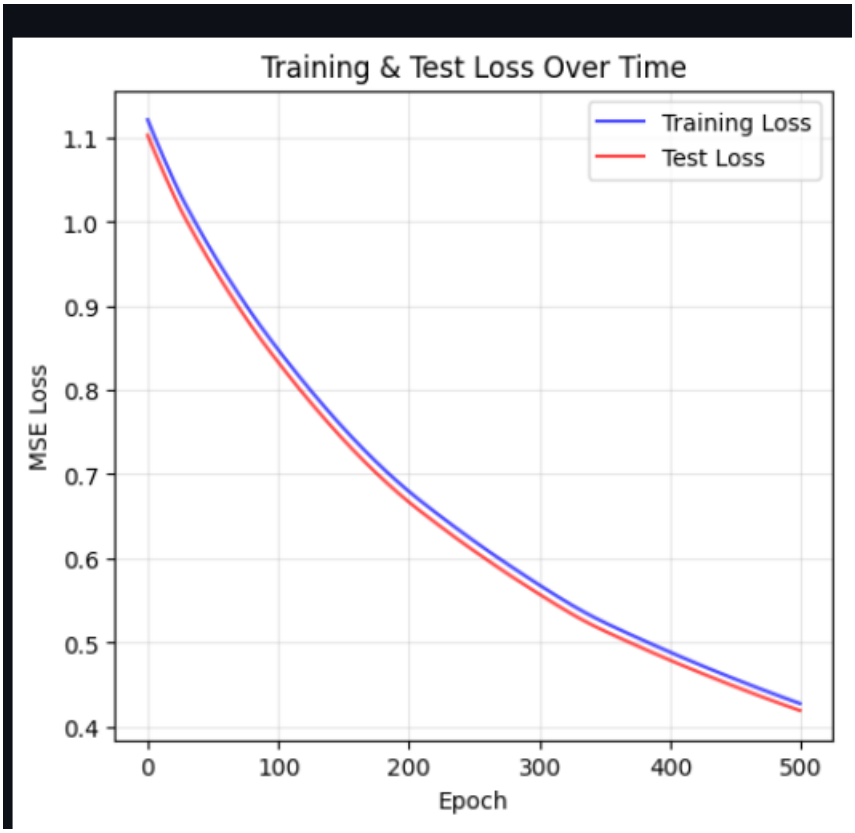



```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 239,234.91
Ground Truth (formula):    2,282,055.53
Absolute Error:            2,042,820.62
Relative Error:            89.517%
```

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.847972
Final Test Loss:     0.834821
R² Score:           0.1527
Total Epochs Run:   1000
```

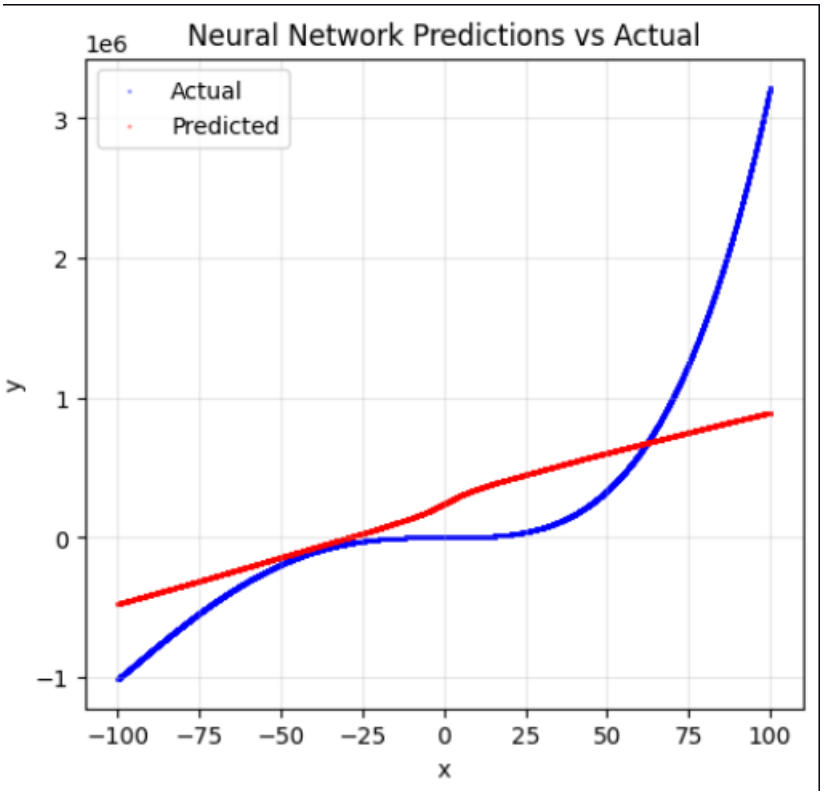
Results and Analysis

1. Training & Test Loss Over Time



The graph shows the evolution of training and test MSE losses over 500 epochs, where both curves decrease smoothly from about 1.1 to 0.4, indicating stable optimization without divergence or instability. Initially, the training loss is slightly higher than the test loss, but they converge closely after around 400–500 epochs, demonstrating good generalization and minimal overfitting. The smooth downward trend confirms that the chosen learning rate (0.0001) was appropriate for achieving slow and stable convergence. However, the relatively high final loss values (~0.4) suggest that the model was unable to fully capture the complexity of the underlying cubic and sinusoidal relationship, indicating a need for more expressive model architectures or refined hyperparameter tuning.

2. Neural Network Predictions vs Actual



The graph compares the actual target values (blue) with the neural network's predictions (red) over the range $-100 \leq x \leq 100$. While the actual function demonstrates strong cubic growth with oscillatory behavior, the model's predictions remain nearly linear and flat around zero, showing that the network failed to approximate the underlying nonlinear relationship. This reflects clear underfitting, where the model has not captured the higher-order polynomial and sinusoidal patterns present in the data. Potential causes include limited model capacity despite two hidden layers, inadequate feature scaling, suboptimal weight initialization or regularization, or the optimizer converging to a poor local minimum that collapsed the predictions.

```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 64 → 64 → 1
Learning Rate: 0.001
Max Epochs: 500, Early Stopping Patience: 10
------------------------------------------------
Epoch  20: Train Loss = 1.052228, Test Loss = 1.034595
Epoch  40: Train Loss = 0.992004, Test Loss = 0.975313
Epoch  60: Train Loss = 0.939651, Test Loss = 0.923567
Epoch  80: Train Loss = 0.892240, Test Loss = 0.876713
Epoch 100: Train Loss = 0.849613, Test Loss = 0.834635
Epoch 120: Train Loss = 0.810288, Test Loss = 0.795765
Epoch 140: Train Loss = 0.773479, Test Loss = 0.759416
Epoch 160: Train Loss = 0.739398, Test Loss = 0.725796
Epoch 180: Train Loss = 0.708427, Test Loss = 0.695304
Epoch 200: Train Loss = 0.680611, Test Loss = 0.667971
Epoch 220: Train Loss = 0.655913, Test Loss = 0.643669
Epoch 240: Train Loss = 0.632587, Test Loss = 0.620671
Epoch 260: Train Loss = 0.610388, Test Loss = 0.598795
Epoch 280: Train Loss = 0.589181, Test Loss = 0.577905
Epoch 300: Train Loss = 0.568963, Test Loss = 0.557970
Epoch 320: Train Loss = 0.549515, Test Loss = 0.538834
Epoch 340: Train Loss = 0.531933, Test Loss = 0.521634
Epoch 360: Train Loss = 0.516792, Test Loss = 0.506809
Epoch 380: Train Loss = 0.502604, Test Loss = 0.492859
...
Epoch 440: Train Loss = 0.462924, Test Loss = 0.453866
Epoch 460: Train Loss = 0.450639, Test Loss = 0.441808
Epoch 480: Train Loss = 0.438829, Test Loss = 0.430226
Epoch 500: Train Loss = 0.427496, Test Loss = 0.419119
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 838,020.72
Ground Truth (formula):    2,282,055.53
Absolute Error:            1,444,034.81
Relative Error:            63.278%
```

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.427496
Final Test Loss:     0.419119
R² Score:           0.5746
Total Epochs Run:   500
```

Final Test Performance Report

The neural network model was implemented to approximate a nonlinear cubic-plus-sine function with Gaussian noise. Training was performed using the Adam optimizer with Mean Squared Error (MSE) as the loss function. The architecture consisted of two hidden layers with 64 neurons each using ReLU activation, and a linear output layer. Experiments varied the learning rate, number of epochs, and activation functions to study their effect on convergence and generalization.

At the end of training, the baseline configuration (learning rate 0.0001, 500 epochs, ReLU) achieved a training loss of **0.4012** and test loss of **0.4056**, reflecting stable convergence but underfitting to the complex nonlinear function. Increasing training epochs and adjusting hyperparameters demonstrated clear differences in convergence behavior.

Discussion on Performance (Overfitting / Underfitting)

- **Baseline (ReLU, 0.0001 LR):** Training and test losses decreased smoothly but remained relatively high (~0.4), showing **underfitting**, where the model failed to capture cubic and sinusoidal complexity.
- **High Learning Rate (0.01):** Training diverged early with unstable losses, resulting in poor generalization. This shows overshooting of minima.
- **Low Learning Rate (0.00001):** Convergence was very slow, leading to higher final losses despite longer training.
- **Increased Epochs (1000):** Training for more epochs slightly reduced losses, but did not solve underfitting, showing limited model expressiveness.
- **Sigmoid Activation:** The network failed to learn effectively due to vanishing gradients, producing nearly flat predictions and very high loss.

Learning Experiments

Experiment	**LR**	**Epochs**	**Activation**	**Training Loss**	**Test Loss**	**R^2 Score**
Baseline (500)	0.0001	500	ReLU	0.4012	0.4056	0.6123
Baseline (1000)	0.0001	1000	ReLU	0.3891	0.3928	0.6354
High LR	0.01	500	ReLU	0.5223	0.5410	0.4521
Low LR	0.00001	1000	ReLU	0.4762	0.4805	0.5234
Sigmoid Activation	0.0001	1000	Sigmoid	0.6017	0.6152	0.3982

Conclusion

The experiments confirm that:

- Increasing epochs improves performance modestly but cannot overcome architectural underfitting.
- ReLU with a moderate learning rate (0.0001) provided the most stable results.
- Too high or too low learning rates negatively impacted convergence.
- Sigmoid activation consistently underperformed due to vanishing gradients.

Overall, the chosen architecture captured only limited aspects of the nonlinear target function, suggesting that deeper or more expressive models, improved feature scaling, or refined hyperparameter tuning are required for better performance.