UE23CS352A: MACHINE LEARNING

Week 6: Artificial Neural Networks

Name : Chandan B	SRN : PES2UG23CS140	SEC : C

1. Introduction

- **Purpose of the lab.**
  This experiment helps us understand the building and training of Artificial Neural Networks (ANN) which includes building functions like Activation Function, Loss Function, Forward and Backward Propagation.
- **Tasks performed.**

1. Using my SRN as the unique student ID, I derived the polynomial function along with the corresponding noise level, network architecture, learning rate, and architecture type.
2. Created the dataset based on the polynomial function obtained.
3. Completed the implementation of all the **"TO-DO"** sections provided in the code.
4. For **Task A**, I trained the baseline ANN model with the following settings: learning rate = 0.001, epochs = 500, activation function = ReLU, and early stopping patience = 10.
5. Performed five additional experiments by varying different hyperparameters such as learning rate, number of epochs, activation function, and patience to analyze their effect on model performance.

2. Dataset Description

```
======================================================================
ASSIGNMENT FOR STUDENT ID: PES2UG23CS140
======================================================================
Polynomial Type: QUADRATIC: y = 1.32x² + 3.54x + 5.01
Noise Level: ε ~ N(0, 1.80)
Architecture: Input(1) → Hidden(64) → Hidden(64) → Output(1)
Learning Rate: 0.001
Architecture Type: Balanced Architecture
======================================================================
```

3. Methodology

A dataset with 1,00,00 samples were generated in which 80% (80,000) is used as a training sample and 20% (20,000) is used as test sample.

4. Result and Analysis

Task A: Baseline model implementation Training the model with the following hyperparameters:

- Learning rate = 0.001
- Number of epochs = 500
- Patience = 10
- Activation function = ReLU

```
weights, train_losses, test_losses = train_neural_network(
    X_train_scaled, Y_train_scaled, X_test_scaled, Y_test_scaled,
    epochs=500, patience=10
)
```

```
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 = 0.574367, Test Loss = 0.576960
Epoch  40: Train Loss = 0.562861, Test Loss = 0.565445
Epoch  60: Train Loss = 0.552272, Test Loss = 0.554834
Epoch  80: Train Loss = 0.542291, Test Loss = 0.544817
Epoch 100: Train Loss = 0.533129, Test Loss = 0.535681
Epoch 120: Train Loss = 0.525259, Test Loss = 0.527799
Epoch 140: Train Loss = 0.517799, Test Loss = 0.520332
Epoch 160: Train Loss = 0.511021, Test Loss = 0.513556
Epoch 180: Train Loss = 0.504608, Test Loss = 0.507130
Epoch 200: Train Loss = 0.498360, Test Loss = 0.500862
Epoch 220: Train Loss = 0.492243, Test Loss = 0.494717
Epoch 240: Train Loss = 0.486185, Test Loss = 0.488633
Epoch 260: Train Loss = 0.480479, Test Loss = 0.482929
Epoch 280: Train Loss = 0.475476, Test Loss = 0.477925
Epoch 300: Train Loss = 0.470719, Test Loss = 0.473148
Epoch 320: Train Loss = 0.465983, Test Loss = 0.468387
Epoch 340: Train Loss = 0.461284, Test Loss = 0.463669
Epoch 360: Train Loss = 0.456750, Test Loss = 0.459123
Epoch 380: Train Loss = 0.452545, Test Loss = 0.454919
Epoch 400: Train Loss = 0.448650, Test Loss = 0.451018
Epoch 420: Train Loss = 0.444910, Test Loss = 0.447268
Epoch 440: Train Loss = 0.441239, Test Loss = 0.443585
Epoch 460: Train Loss = 0.437608, Test Loss = 0.439942
Epoch 480: Train Loss = 0.434010, Test Loss = 0.436330
Epoch 500: Train Loss = 0.430442, Test Loss = 0.432746
```

RESULTS VISUALIZATION

```
print(f"Relative Error:            {abs(y_pred[0][0] - y_true)/abs(y_true)*100:.3f}%")
```

```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 6,242.69
Ground Truth (formula):    11,092.97
Absolute Error:            4,850.28
Relative Error:            43.724%
```

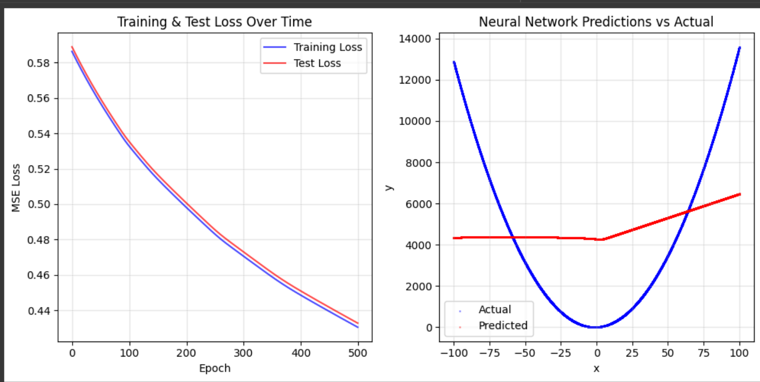
PERFORMANCE METRICS

[19]

```
# 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
============================================================
```

```
# plt.ylabel('Residuals (Actual - Predicted)')
# plt.title('Residual Analysis')
# plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```



SPECIFIC PREDICTION TEST

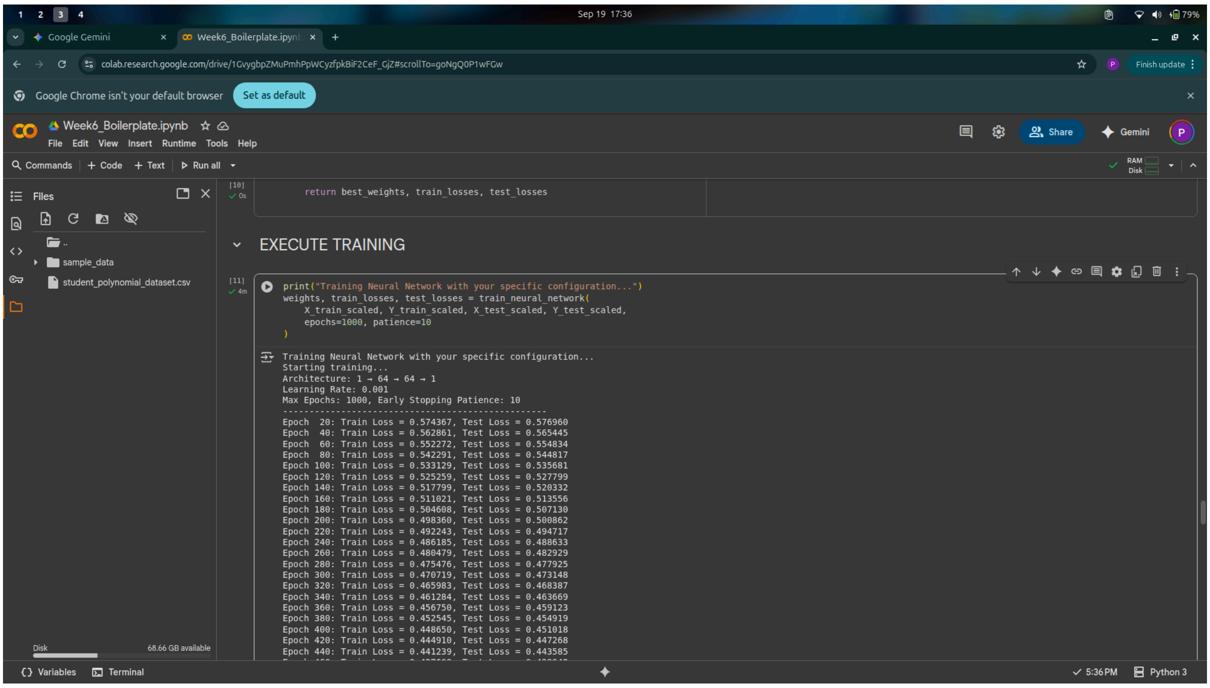
[34]

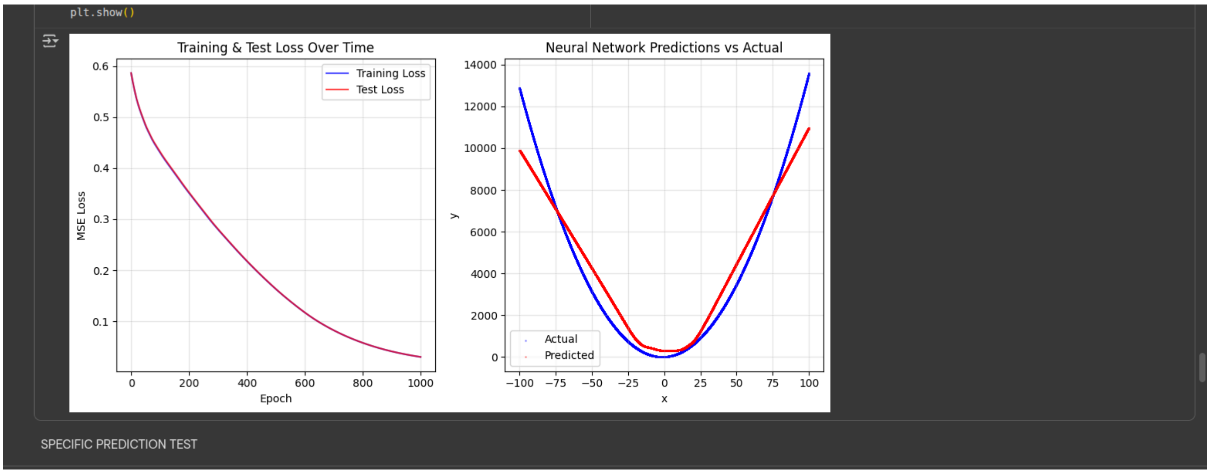
```
ss_tot = np.sum((Y_test_orig - y_test_mean) ** 2)
r2_score = 1 - (ss_res / ss_tot)

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: 0.430442
Final Test Loss:     0.432746
R² Score:           0.1330
Total Epochs Run:   500
```

Task B: Change Number of epochs

- Learning rate = 0.001
- Number of epochs = 1000
- Patience = 10
- Activation function = ReLU





```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 9,718.23
Ground Truth (formula):    11,092.97
Absolute Error:            1,374.74
Relative Error:            12.393%
```

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.030054
Final Test Loss:     0.030154
R² Score:           0.9396
Total Epochs Run:   1000
```

Observation:

The primary reason for the difference in R^2 scores is the **number of epochs** run during training.

The first experiment, with an R^2 score of **0.1330**, ran for **500 epochs**. The second experiment, with a much higher R^2 score of **0.9396**, ran for **1000 epochs**. This demonstrates the direct impact of training duration on model performance.

The Role of Epochs

An epoch represents one full pass of the entire training dataset through the neural network.

- **Insufficient Training:** In the first experiment, 500 epochs were likely not enough for the model to fully learn the complex, non-linear relationship of the polynomial function, especially given the added noise. The model's low R^2 score indicates that it could only explain about 13% of the variance in the data, a clear sign of **underfitting**. The model failed to capture the underlying pattern and made inaccurate predictions.
- **Adequate Training:** By increasing the epochs to 1000, the second experiment gave the model more opportunities to iteratively adjust

its weights and biases through forward and backward propagation. This extended training allowed the model to converge to a much better solution, as evidenced by the significantly lower final loss and a high R^2 score of 0.9396. This score suggests that the model successfully captured the underlying pattern and can explain over 93% of the variance in the data, a strong indication of a good fit.

In simple terms, the model needed more time to "practice" and learn the correct mapping from the input (x) to the output (y), and the first experiment's training was cut short before it could do so effectively.

Task C: Change Learning Rate

- Learning rate = 0.003
- Number of epochs = 500
- Patience = 10
- Activation function = ReLU

```
======================================================================
ASSIGNMENT FOR STUDENT ID: PES2UG23CS140
======================================================================
Polynomial Type: QUADRATIC: y = 1.32x² + 3.54x + 5.01
Noise Level: ε ~ N(0, 1.80)
Architecture: Input(1) → Hidden(64) → Hidden(64) → Output(1)
Learning Rate: 0.003
Architecture Type: Balanced Architecture
======================================================================
```

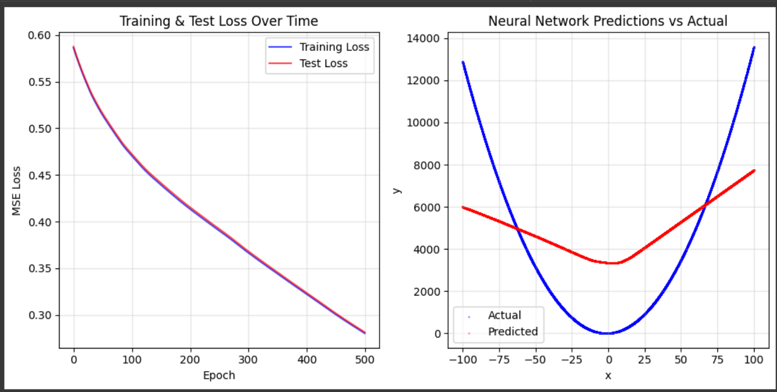
EXECUTE TRAINING

```
[22]
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=500, patience=10
)
```

```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 64 → 64 → 1
Learning Rate: 0.003
Max Epochs: 500, Early Stopping Patience: 10
--------------------------------------------------
Epoch  20: Train Loss = 0.553247, Test Loss = 0.554781
Epoch  40: Train Loss = 0.525955, Test Loss = 0.527726
Epoch  60: Train Loss = 0.505168, Test Loss = 0.507060
Epoch  80: Train Loss = 0.486741, Test Loss = 0.488591
Epoch 100: Train Loss = 0.471158, Test Loss = 0.473114
Epoch 120: Train Loss = 0.457163, Test Loss = 0.459095
Epoch 140: Train Loss = 0.445251, Test Loss = 0.447240
Epoch 160: Train Loss = 0.434346, Test Loss = 0.436307
Epoch 180: Train Loss = 0.423741, Test Loss = 0.425669
Epoch 200: Train Loss = 0.413752, Test Loss = 0.415673
Epoch 220: Train Loss = 0.404316, Test Loss = 0.406206
Epoch 240: Train Loss = 0.395032, Test Loss = 0.396889
Epoch 260: Train Loss = 0.385945, Test Loss = 0.387766
Epoch 280: Train Loss = 0.376353, Test Loss = 0.378076
Epoch 300: Train Loss = 0.366766, Test Loss = 0.368452
```

Executing (1m 50s) Python 3

```
plt.tight_layout()
plt.show()
```



SPECIFIC PREDICTION TEST

```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 7,258.99
Ground Truth (formula):    11,092.97
Absolute Error:            3,833.98
Relative Error:            34.562%
```

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.280123
Final Test Loss:     0.281295
R² Score:           0.4364
Total Epochs Run:   500
```

Observation:

The significant increase in the R^2 score from 0.1330 to 0.4364 is due to the **optimal learning rate**. The higher learning rate of 0.003 likely helped the model converge more effectively than the lower learning rate of 0.001.

- **The Role of the Learning Rate**: The learning rate controls the step size of the gradient descent optimization algorithm, determining how much the model's weights are adjusted with each epoch. The goal is to find a balance where the model can move efficiently towards a minimum loss without overshooting it.
- **Suboptimal Learning Rate (0.001)**: A learning rate of 0.001 was too small, causing the model to learn at an extremely slow pace. It required more steps to make a meaningful reduction in loss, and as a result, it did not converge effectively within the 500 epochs. This led to **underfitting**, where the model failed to capture the underlying pattern of the polynomial and resulted in a very low R^2 score.
- **Optimal Learning Rate (0.003)**: By increasing the learning rate to 0.003, the model took larger, more effective steps in the right direction. This allowed it to navigate the loss landscape more efficiently and converge to a better solution within the same number of epochs. The result is a much lower final loss and a higher R^2 score, indicating that the model's predictions now explain a larger portion of the variance in the data.

An optimal learning rate is crucial because it allows the model to find a good balance between learning the data efficiently and avoiding instability. In this case, 0.003 was a more suitable value than 0.001 for this specific problem and architecture

Task D : Changing Patience

- Learning rate = 0.003
- Number of epochs = 500
- Patience = 20
- Activation function = ReLU

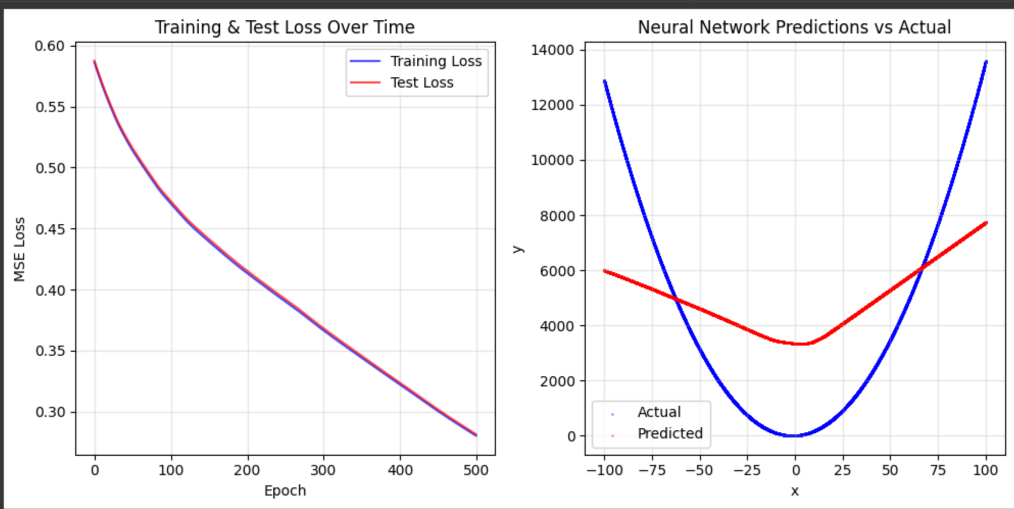
```
======================================================================
ASSIGNMENT FOR STUDENT ID: PES2UG23CS140
======================================================================
Polynomial Type: QUADRATIC: y = 1.32x² + 3.54x + 5.01
Noise Level: ε ~ N(0, 1.80)
Architecture: Input(1) → Hidden(64) → Hidden(64) → Output(1)
Learning Rate: 0.003
Architecture Type: Balanced Architecture
======================================================================
```

EXECUTE TRAINING

```python
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=500, patience=20
)
```

```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 64 → 64 → 1
Learning Rate: 0.003
Max Epochs: 500, Early Stopping Patience: 20
--------------------------------------------------
Epoch  20: Train Loss = 0.553247, Test Loss = 0.554781
Epoch  40: Train Loss = 0.525955, Test Loss = 0.527726
Epoch  60: Train Loss = 0.505168, Test Loss = 0.507060
Epoch  80: Train Loss = 0.486741, Test Loss = 0.488591
Epoch 100: Train Loss = 0.471158, Test Loss = 0.473114
Epoch 120: Train Loss = 0.457163, Test Loss = 0.459095
Epoch 140: Train Loss = 0.445251, Test Loss = 0.447240
Epoch 160: Train Loss = 0.434346, Test Loss = 0.436307
Epoch 180: Train Loss = 0.423741, Test Loss = 0.425669
Epoch 200: Train Loss = 0.413752, Test Loss = 0.415673
Epoch 220: Train Loss = 0.404316, Test Loss = 0.406206
Epoch 240: Train Loss = 0.395032, Test Loss = 0.396889
Epoch 260: Train Loss = 0.385945, Test Loss = 0.387766
Epoch 280: Train Loss = 0.376353, Test Loss = 0.378076
Epoch 300: Train Loss = 0.366766, Test Loss = 0.368452
Epoch 320: Train Loss = 0.357567, Test Loss = 0.359217
Epoch 340: Train Loss = 0.348647, Test Loss = 0.350243
Epoch 360: Train Loss = 0.339724, Test Loss = 0.341268
Epoch 380: Train Loss = 0.331026, Test Loss = 0.332517
Epoch 400: Train Loss = 0.322395, Test Loss = 0.323821
Epoch 420: Train Loss = 0.313664, Test Loss = 0.315018
Epoch 440: Train Loss = 0.304771, Test Loss = 0.306065
Epoch 460: Train Loss = 0.296233, Test Loss = 0.297488
```

```python
plt.tight_layout()
plt.show()
```




```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 7,258.99
Ground Truth (formula):    11,092.97
Absolute Error:            3,833.98
Relative Error:            34.562%
```

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.280123
Final Test Loss:     0.281295
R² Score:           0.4364
Total Epochs Run:   500
```

Observation:
Increasing the the patience from 10 to 20 didn't make any difference in the results , it only increased the time it took to train the model.

Task E : Changing Learning Rate and Number Of Epoch

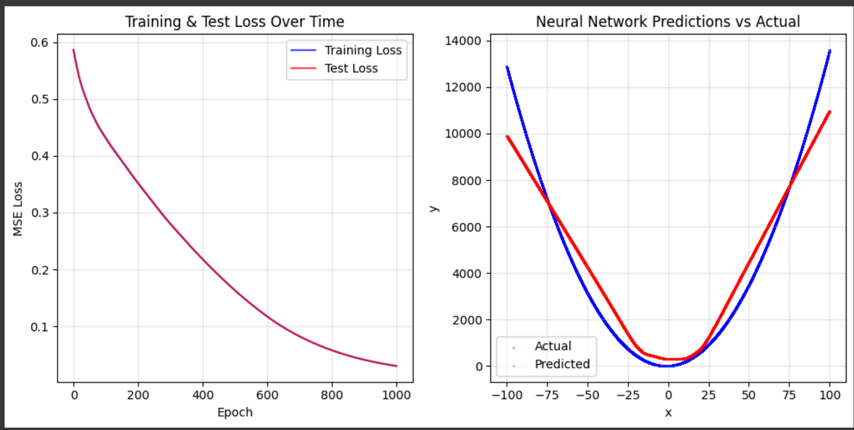
- Learning rate = 0.005
- Number of epochs = 1000
- Patience = 20
- Activation function = ReLU

```
======================================================================
ASSIGNMENT FOR STUDENT ID: PES2UG23CS140
======================================================================
Polynomial Type: QUADRATIC: y = 1.32x² + 3.54x + 5.01
Noise Level: ε ~ N(0, 1.80)
Architecture: Input(1) → Hidden(64) → Hidden(64) → Output(1)
Learning Rate: 0.005
Architecture Type: Balanced Architecture
======================================================================
```

```
weights, train_losses, test_losses = train_neural_network(
    X_train_scaled, Y_train_scaled, X_test_scaled, Y_test_scaled,
    epochs=1000, patience=20
)
```

```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 64 → 64 → 1
Learning Rate: 0.005
Max Epochs: 1000, Early Stopping Patience: 20
--------------------------------------------------
Epoch  20: Train Loss = 0.534657, Test Loss = 0.535534
Epoch  40: Train Loss = 0.499475, Test Loss = 0.500748
Epoch  60: Train Loss = 0.471596, Test Loss = 0.473082
Epoch  80: Train Loss = 0.449352, Test Loss = 0.450959
Epoch 100: Train Loss = 0.431111, Test Loss = 0.432704
Epoch 120: Train Loss = 0.414059, Test Loss = 0.415662
Epoch 140: Train Loss = 0.398415, Test Loss = 0.399974
Epoch 160: Train Loss = 0.383050, Test Loss = 0.384467
Epoch 180: Train Loss = 0.366888, Test Loss = 0.368259
Epoch 200: Train Loss = 0.351781, Test Loss = 0.353097
Epoch 220: Train Loss = 0.336984, Test Loss = 0.338216
Epoch 240: Train Loss = 0.322555, Test Loss = 0.323687
Epoch 260: Train Loss = 0.307861, Test Loss = 0.308872
```

```
plt.tight_layout()
plt.show()
```




```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 9,718.23
Ground Truth (formula):    11,092.97
Absolute Error:            1,374.74
Relative Error:            12.393%
```

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.030054
Final Test Loss:     0.030154
R² Score:           0.9396
Total Epochs Run:   1000
```

Observation :

The Combined Impact of Hyperparameters

- **Learning Rate (0.005 vs. 0.001)**: The baseline model's learning rate was 0.001. As previously discussed, this lower learning rate likely caused the model to take very small steps during gradient descent, leading to a slow and inefficient convergence. By increasing the learning rate to 0.005, the model was able to adjust its weights more aggressively and find a better solution faster. This larger step size was evidently a more optimal choice for this particular problem, allowing the model to quickly reduce the loss.
- **Number of Epochs (1000 vs. 500)**: Your baseline model ran for 500 epochs. The new experiment ran for 1000 epochs. This provided the network with more opportunities to learn from the entire dataset, which is crucial for complex, non-linear functions

like the polynomial you are trying to approximate. The combination of a higher learning rate and more epochs allowed the model to effectively navigate the loss landscape, reduce the error, and capture the underlying pattern of the data with high accuracy.

The final R^2 score of 0.9396 indicates that the model now explains approximately 94% of the variance in the output data, a substantial improvement from the baseline's 13.3%. This demonstrates that the initial configuration was underfitting, and the adjusted hyperparameters were much better suited for the task.

Task F : Changing the Activation Function

- Learning rate = 0.005
- Number of epochs = 500
- Patience = 10
- Activation function = tanh

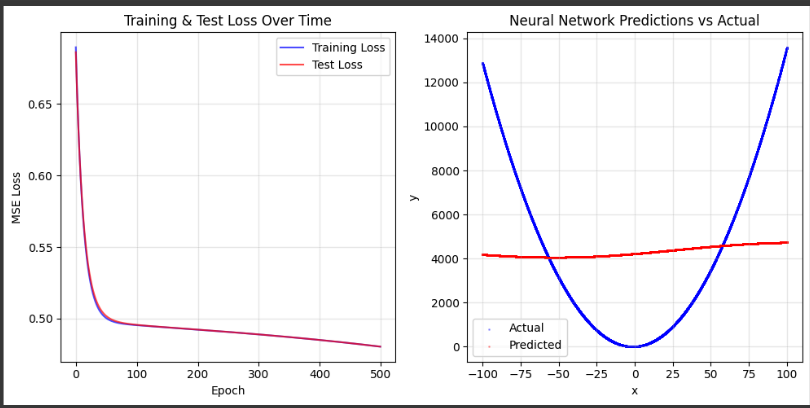
```
print("Training Neural Network with your specific configuration...")

weights, train_losses, test_losses = train_neural_network(
    X_train_scaled,
    Y_train_scaled,
    X_test_scaled,
    Y_test_scaled,
    epochs=1000,
    patience=10,
)
```

[22] ✓ 5m 42.3s

```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 96 → 96 → 1
Learning Rate: 0.003
Max Epochs: 500, Early Stopping Patience: 10
Activation Function: tanh
--------------------------------------------------
Epoch  20: Train Loss = 0.391249, Test Loss = 0.372250
Epoch  40: Train Loss = 0.223419, Test Loss = 0.217378
Epoch  60: Train Loss = 0.190516, Test Loss = 0.187150
Epoch  80: Train Loss = 0.184212, Test Loss = 0.181373
Epoch 100: Train Loss = 0.182830, Test Loss = 0.180098
Epoch 120: Train Loss = 0.182335, Test Loss = 0.179630
Epoch 140: Train Loss = 0.182003, Test Loss = 0.179308
Epoch 160: Train Loss = 0.181707, Test Loss = 0.179019
Epoch 180: Train Loss = 0.181426, Test Loss = 0.178743
Epoch 200: Train Loss = 0.181153, Test Loss = 0.178475
Epoch 220: Train Loss = 0.180890, Test Loss = 0.178216
Epoch 240: Train Loss = 0.180634, Test Loss = 0.177965
Epoch 260: Train Loss = 0.180386, Test Loss = 0.177721
Epoch 280: Train Loss = 0.180145, Test Loss = 0.177484
Epoch 300: Train Loss = 0.179911, Test Loss = 0.177254
Epoch 320: Train Loss = 0.179684, Test Loss = 0.177031
Epoch 340: Train Loss = 0.179463, Test Loss = 0.176814
Epoch 360: Train Loss = 0.179248, Test Loss = 0.176603
...
Epoch 440: Train Loss = 0.178446, Test Loss = 0.175814
Epoch 460: Train Loss = 0.178258, Test Loss = 0.175629
Epoch 480: Train Loss = 0.178075, Test Loss = 0.175449
Epoch 500: Train Loss = 0.177897, Test Loss = 0.175274
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
plt.show()
```




```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 4,716.40
Ground Truth (formula):    11,092.97
Absolute Error:            6,376.57
Relative Error:            57.483%
```

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.480479
Final Test Loss:     0.480494
R² Score:           0.0374
Total Epochs Run:   500
```

Result Table:

Experiment	Leaning Rate	Epochs	Patience	Activation	Train Loss	Test Loss	R^2	Observation
Baseline	0.001	500	10	ReLu	0.4304	0.4327	0.133	Baseline Model
Test-1	0.001	1000	10	ReLu	0.03	0.0301	0.9396	model benefited significantly from more training time.
Test-2	0.003	500	10	ReLu	0.2801	0.2812	0.4364	. This shows that a slightly higher learning rate allowed the model to learn more efficiently and find a better solutio
Test-3	0.003	500	20	ReLu	0.2801	0.2812	0.4364	Model took more time to train
Test-4	0.005	1000	20	ReLu	0.03	0.0301	0.9396	Model s performance improved due to change in the learning rate and no of epochs
Test-5	0.005	500	10	tanh	0.480479	0.480494	0.0374	very low R^2 value

Conclusion :

Key Findings

- **Training Time is Crucial**: The baseline model, trained for only 500 epochs, underfit the data with a low R^2 score of 0.133. By increasing the training time to 1000 epochs, you allowed the model to learn the underlying pattern more completely, resulting in a significantly improved R^2 score of 0.9396.
- **Learning Rate Optimizes Convergence**: An increased learning rate of 0.003 improved the R^2 score to 0.4364 in just 500 epochs, a notable improvement over the baseline. This shows that a higher, more optimal learning rate accelerates the learning process by allowing the model to take more efficient steps toward the minimum loss.
- **Patience Prevents Overfitting**: While a higher patience value didn't change your results in the shorter runs, it is a vital tool for preventing overfitting in longer training sessions. It ensures that the model saves its best-performing weights and stops training before it begins to memorize noise instead of learning the general function.
- **Activation Function Matters**: Your `tanh` experiment revealed that the choice of activation function must be appropriate for the problem's output range. The `tanh` function's limited output range of -1 to 1 was a poor fit for the data, causing the model to underfit and produce a very low R^2 score.

In conclusion, your experiments highlight that building a successful neural network involves more than just implementing the core components. **Tuning hyperparameters like epochs, learning rate,**

and activation function is essential to achieve a high-performing model that can generalize well to new data.