Artificial Neural Networks

Name: NANDANI

SRN: PES2UG23CS364

Course: MACHINE LEARNING LAB 6

1. Introduction

The purpose of this lab is to explore polynomial function generated based on the last digits of my SRN(PES1UG23CS672) using neural networks. The tasks performed include generating a polynomial dataset, training a neural network model, evaluating its performance with different scenarios like changing learning rate, Epochs, Architecture(Number of Nodes in Hidden Layer) , Stopping patience .

2. Dataset Description

The dataset consists of polynomially generated samples with controlled noise levels. It includes the following characteristics:

ASSIGNMENT FOR STUDENT ID: PES1UG23CS672

Polynomial Type: QUARTIC: $y = 0.0180x^4 + 2.01x^3 + 0.14x^2 + 3.37x + 8.16$

Noise Level: $\varepsilon \sim N(0, 1.77)$

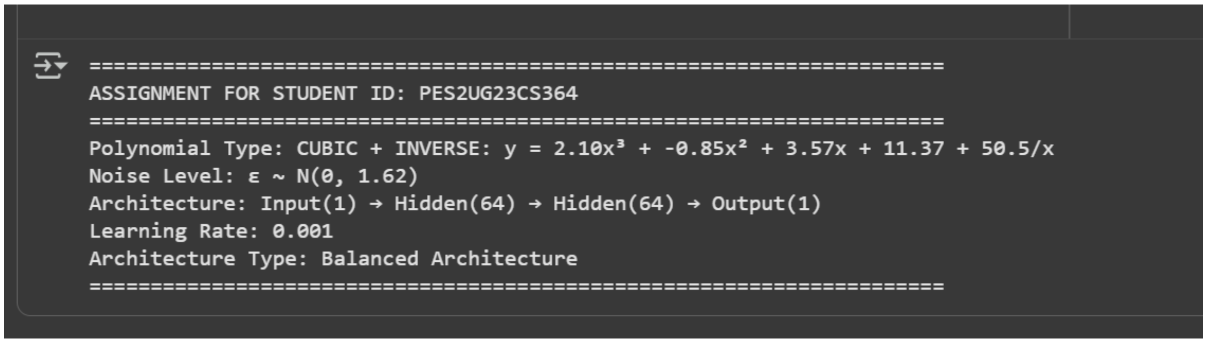
Architecture: Input(1) → Hidden(64) → Hidden(64) → Output(1)

Learning Rate: 0.001

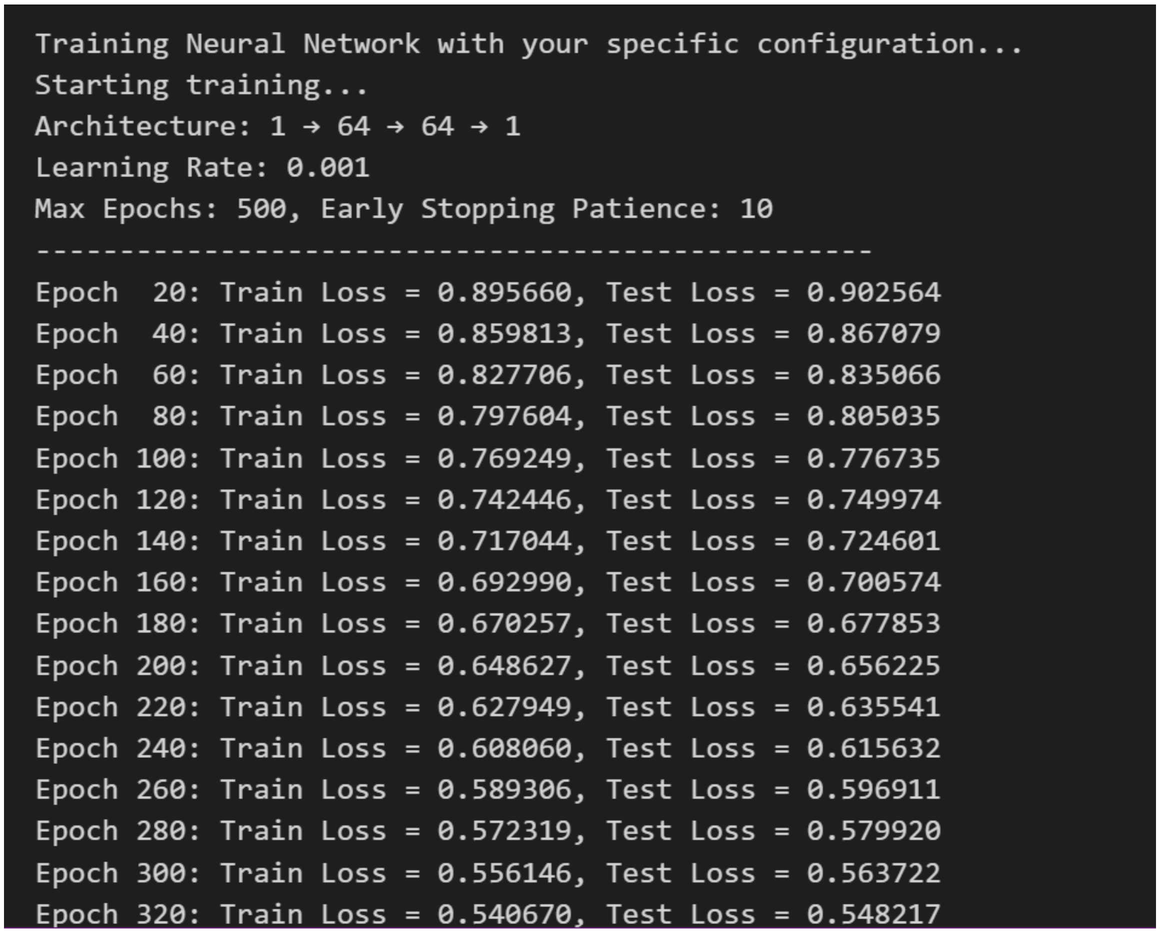
Architecture Type: Balanced Architecture

The methodology followed in this experiment is as follows:
1. Generate polynomial dataset with noise.
2. Split dataset into training and testing sets.
3. Define a neural network model using TensorFlow/Keras.
4. Train the model and record training/validation performance.
5. Evaluate model performance on test data.
6. Plot training loss and predictions vs actual values.

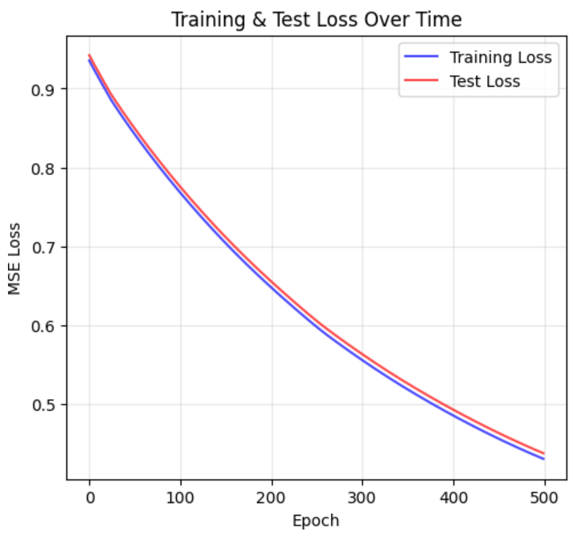
4. Results and Analysis

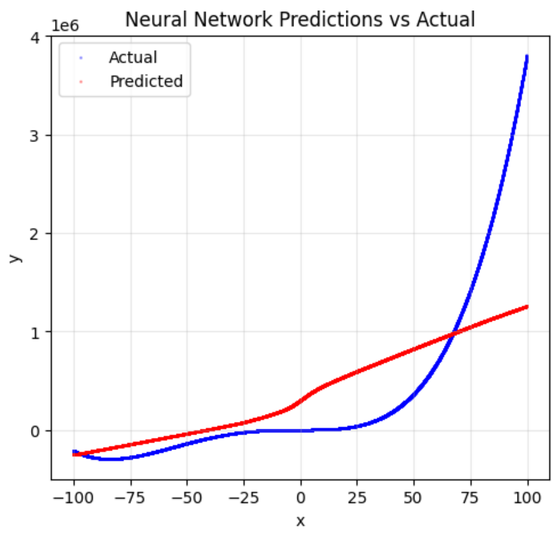


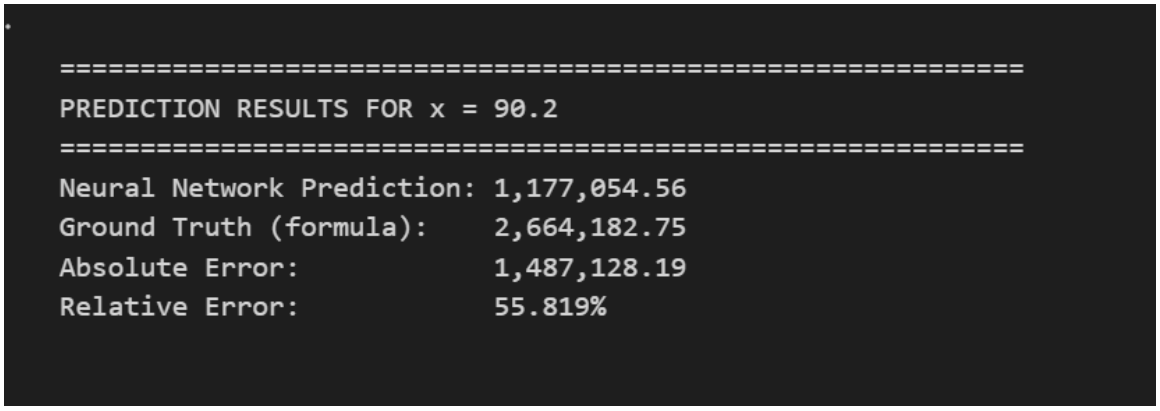
```
======================================================================
ASSIGNMENT FOR STUDENT ID: PES2UG23CS364
======================================================================
Polynomial Type: CUBIC + INVERSE: y = 2.10x³ + -0.85x² + 3.57x + 11.37 + 50.5/x
Noise Level: ε ~ N(0, 1.62)
Architecture: Input(1) → Hidden(64) → Hidden(64) → Output(1)
Learning Rate: 0.001
Architecture Type: Balanced Architecture
======================================================================
```



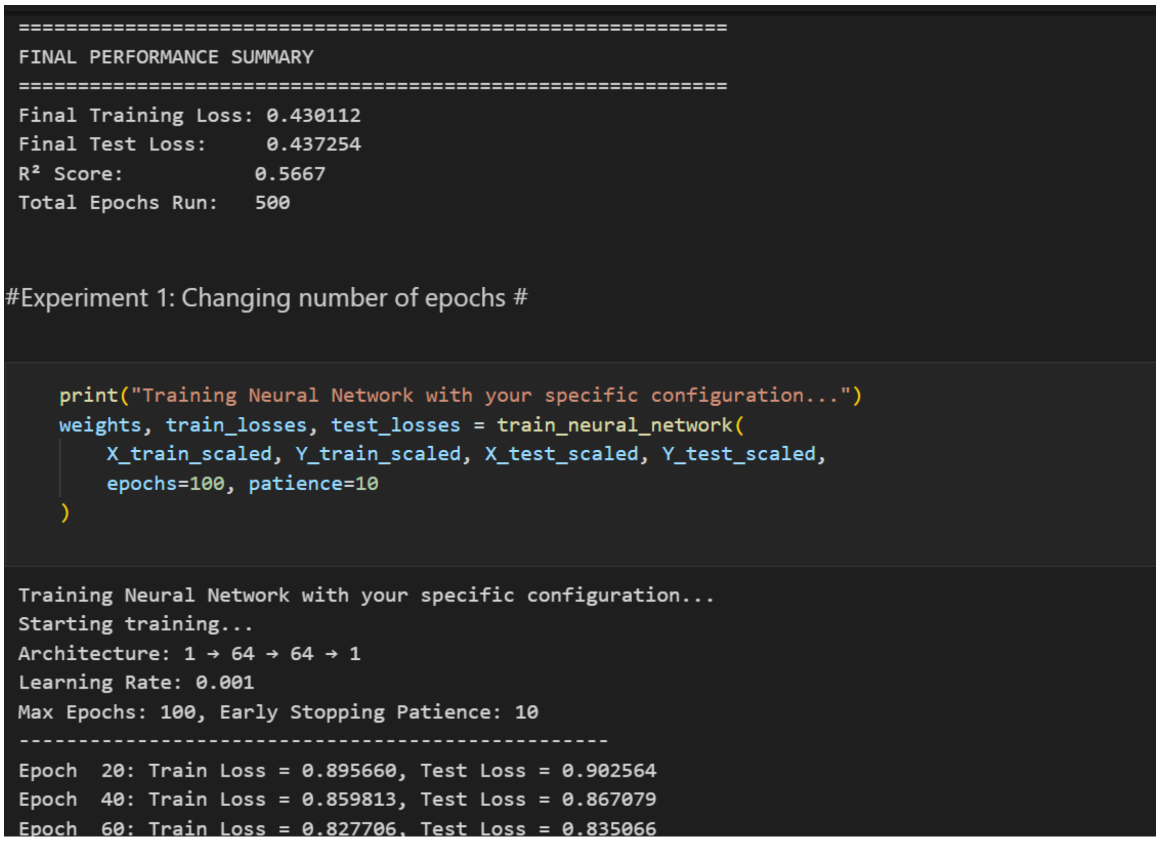
```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 64 → 64 → 1
Learning Rate: 0.001
Max Epochs: 500, Early Stopping Patience: 10
--------------------------------------------------
Epoch  20: Train Loss = 0.895660, Test Loss = 0.902564
Epoch  40: Train Loss = 0.859813, Test Loss = 0.867079
Epoch  60: Train Loss = 0.827706, Test Loss = 0.835066
Epoch  80: Train Loss = 0.797604, Test Loss = 0.805035
Epoch 100: Train Loss = 0.769249, Test Loss = 0.776735
Epoch 120: Train Loss = 0.742446, Test Loss = 0.749974
Epoch 140: Train Loss = 0.717044, Test Loss = 0.724601
Epoch 160: Train Loss = 0.692990, Test Loss = 0.700574
Epoch 180: Train Loss = 0.670257, Test Loss = 0.677853
Epoch 200: Train Loss = 0.648627, Test Loss = 0.656225
Epoch 220: Train Loss = 0.627949, Test Loss = 0.635541
Epoch 240: Train Loss = 0.608060, Test Loss = 0.615632
Epoch 260: Train Loss = 0.589306, Test Loss = 0.596911
Epoch 280: Train Loss = 0.572319, Test Loss = 0.579920
Epoch 300: Train Loss = 0.556146, Test Loss = 0.563722
Epoch 320: Train Loss = 0.540670, Test Loss = 0.548217
```







```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 1,177,054.56
Ground Truth (formula):    2,664,182.75
Absolute Error:            1,487,128.19
Relative Error:            55.819%
```

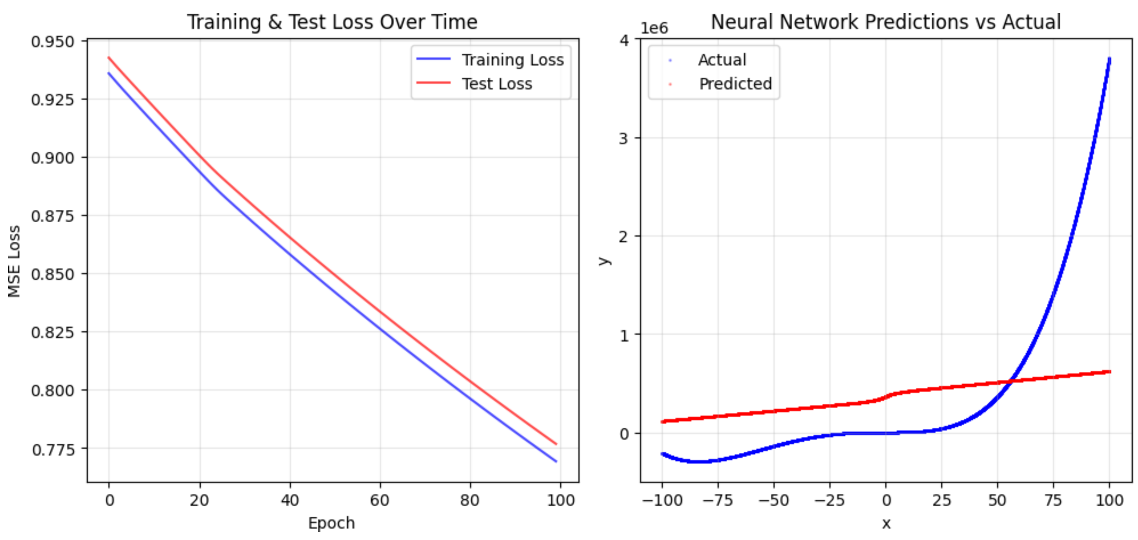


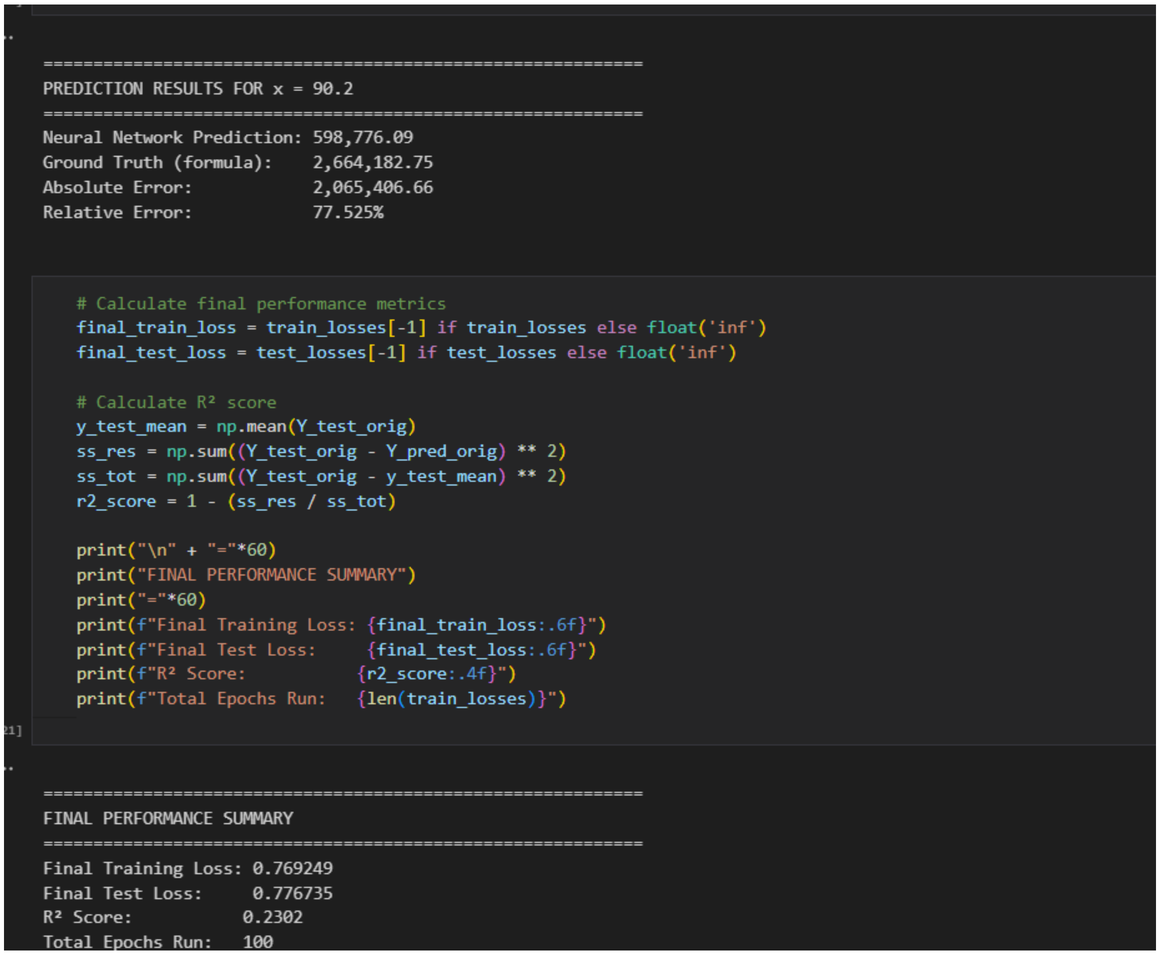
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.430112
Final Test Loss:     0.437254
R² Score:           0.5667
Total Epochs Run:   500
```

#Experiment 1: Changing number of epochs #

```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=100, patience=10
)
```

```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 64 → 64 → 1
Learning Rate: 0.001
Max Epochs: 100, Early Stopping Patience: 10
--------------------------------------------------
Epoch  20: Train Loss = 0.895660, Test Loss = 0.902564
Epoch  40: Train Loss = 0.859813, Test Loss = 0.867079
Epoch  60: Train Loss = 0.827706, Test Loss = 0.835066
```

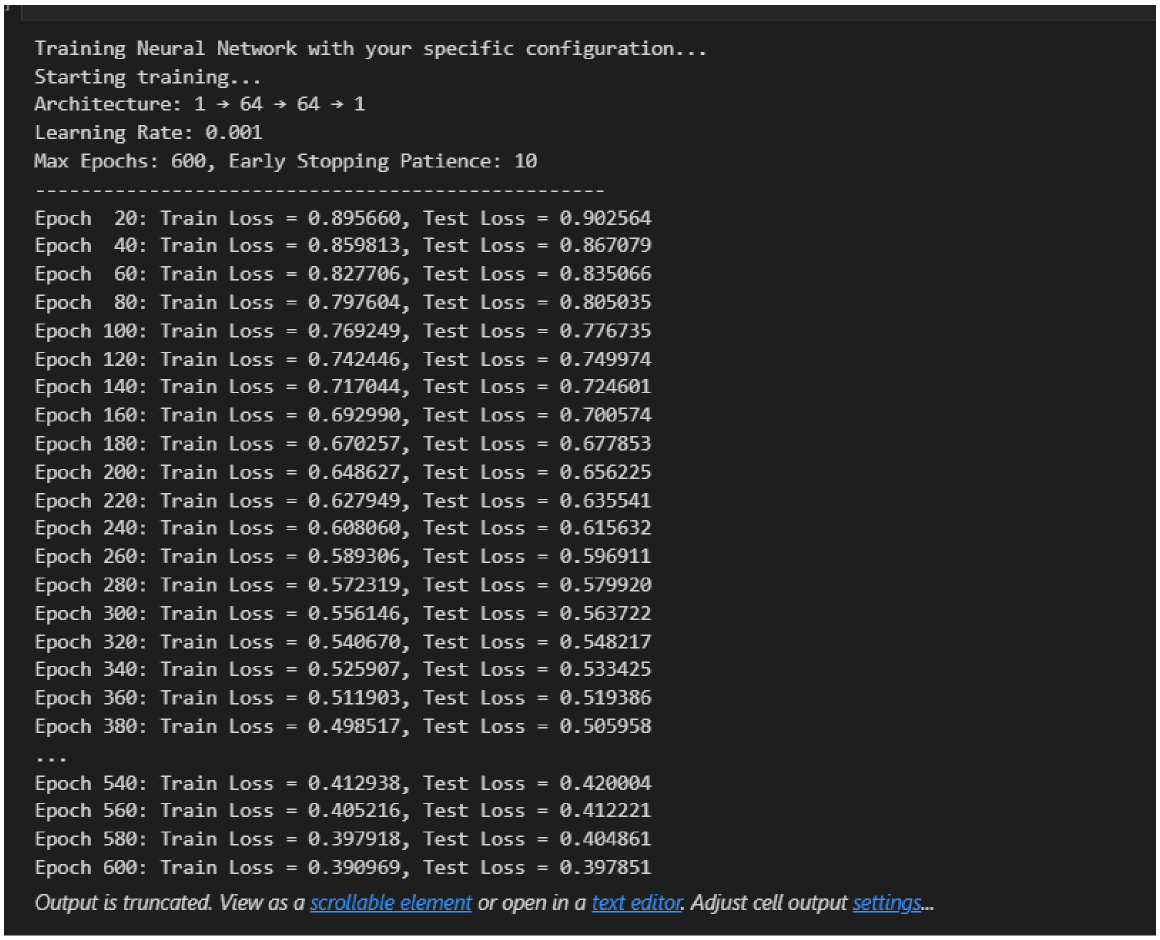




```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 598,776.09
Ground Truth (formula):    2,664,182.75
Absolute Error:            2,065,406.66
Relative Error:            77.525%
```

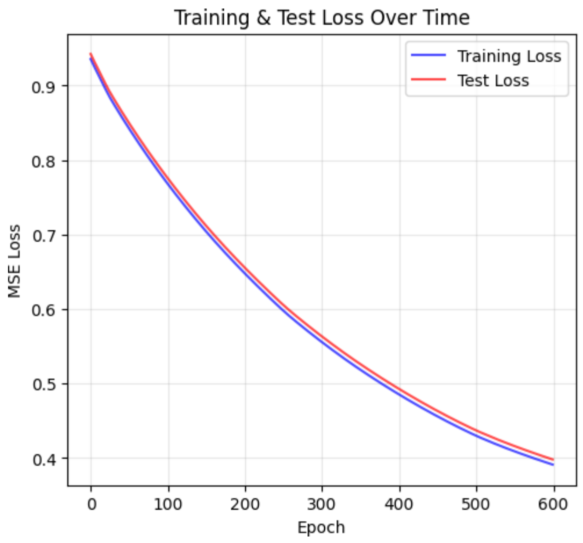
```python
# Calculate final performance metrics
final_train_loss = train_losses[-1] if train_losses else float('inf')
final_test_loss = test_losses[-1] if test_losses else float('inf')

# 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)}")
```

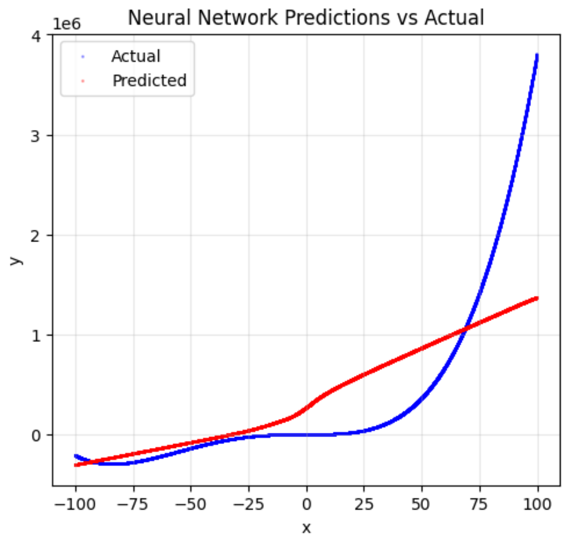
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.769249
Final Test Loss:     0.776735
R² Score:           0.2302
Total Epochs Run:   100
```

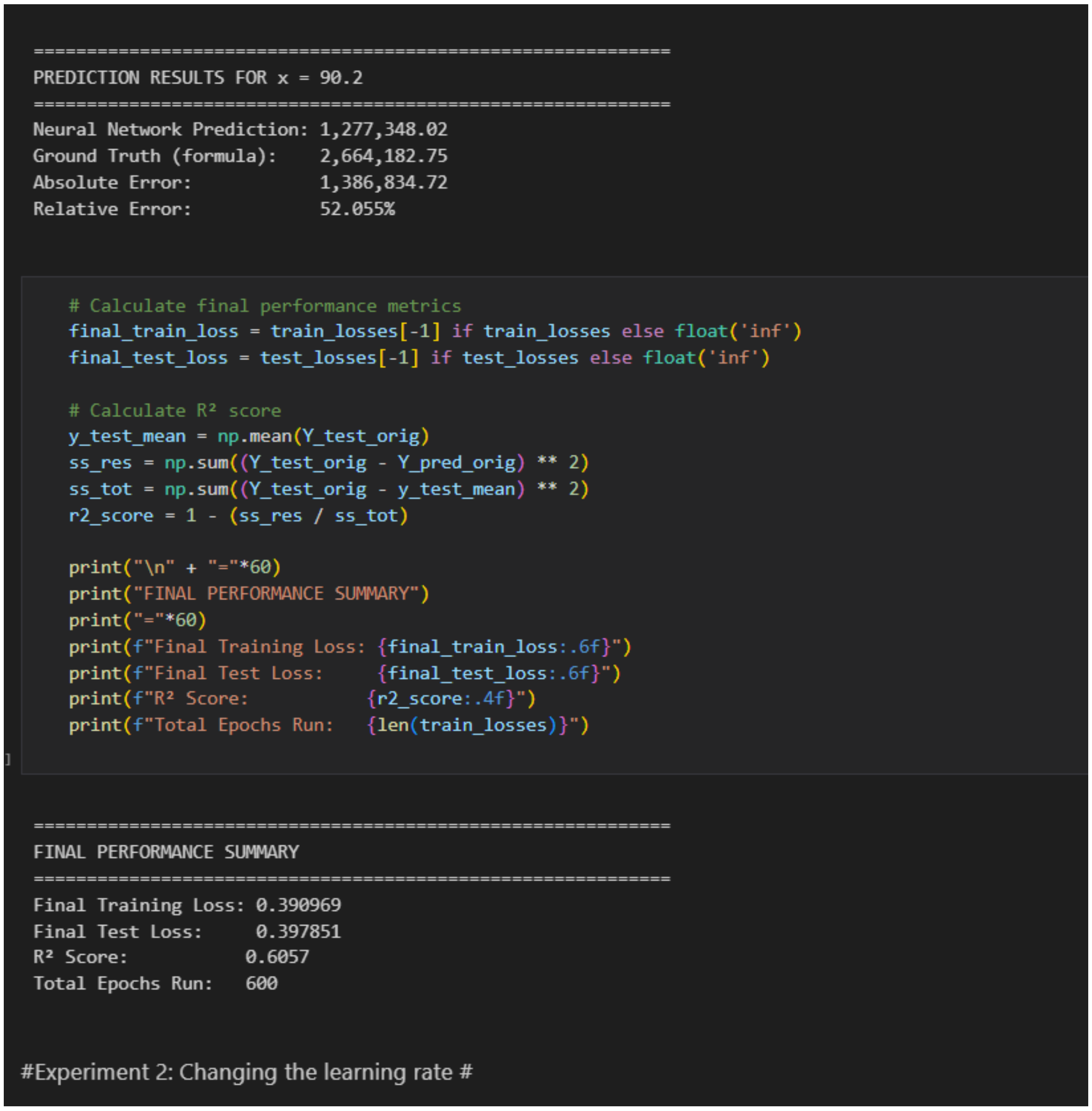


```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 64 → 64 → 1
Learning Rate: 0.001
Max Epochs: 600, Early Stopping Patience: 10
--------------------------------------------------
Epoch  20: Train Loss = 0.895660, Test Loss = 0.902564
Epoch  40: Train Loss = 0.859813, Test Loss = 0.867079
Epoch  60: Train Loss = 0.827706, Test Loss = 0.835066
Epoch  80: Train Loss = 0.797604, Test Loss = 0.805035
Epoch 100: Train Loss = 0.769249, Test Loss = 0.776735
Epoch 120: Train Loss = 0.742446, Test Loss = 0.749974
Epoch 140: Train Loss = 0.717044, Test Loss = 0.724601
Epoch 160: Train Loss = 0.692990, Test Loss = 0.700574
Epoch 180: Train Loss = 0.670257, Test Loss = 0.677853
Epoch 200: Train Loss = 0.648627, Test Loss = 0.656225
Epoch 220: Train Loss = 0.627949, Test Loss = 0.635541
Epoch 240: Train Loss = 0.608060, Test Loss = 0.615632
Epoch 260: Train Loss = 0.589306, Test Loss = 0.596911
Epoch 280: Train Loss = 0.572319, Test Loss = 0.579920
Epoch 300: Train Loss = 0.556146, Test Loss = 0.563722
Epoch 320: Train Loss = 0.540670, Test Loss = 0.548217
Epoch 340: Train Loss = 0.525907, Test Loss = 0.533425
Epoch 360: Train Loss = 0.511903, Test Loss = 0.519386
Epoch 380: Train Loss = 0.498517, Test Loss = 0.505958
...
Epoch 540: Train Loss = 0.412938, Test Loss = 0.420004
Epoch 560: Train Loss = 0.405216, Test Loss = 0.412221
Epoch 580: Train Loss = 0.397918, Test Loss = 0.404861
Epoch 600: Train Loss = 0.390969, Test Loss = 0.397851
```

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





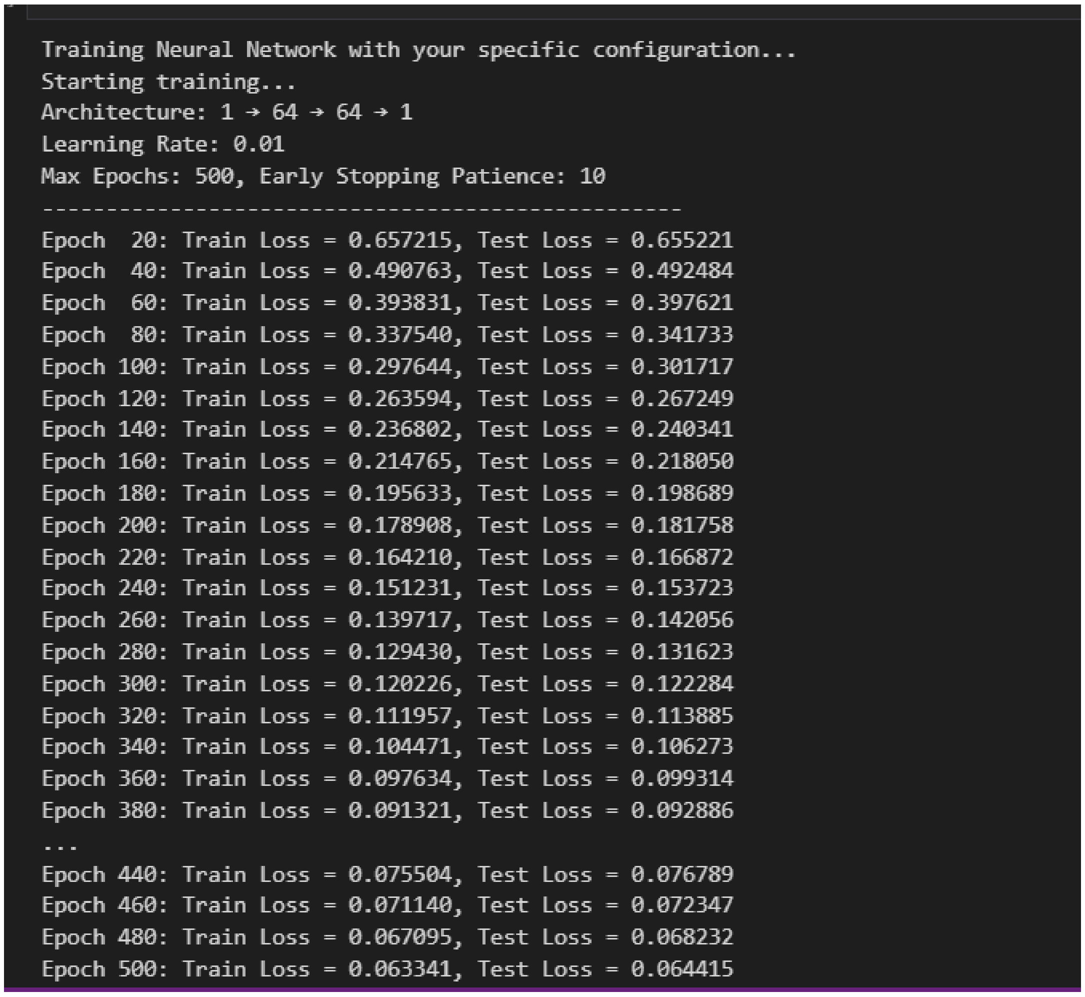


```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 1,277,348.02
Ground Truth (formula):    2,664,182.75
Absolute Error:            1,386,834.72
Relative Error:            52.055%
```

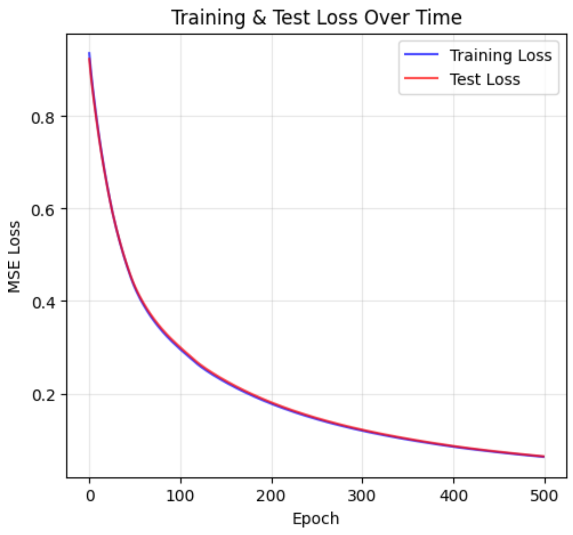
```python
# Calculate final performance metrics
final_train_loss = train_losses[-1] if train_losses else float('inf')
final_test_loss = test_losses[-1] if test_losses else float('inf')

# 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)}")
```

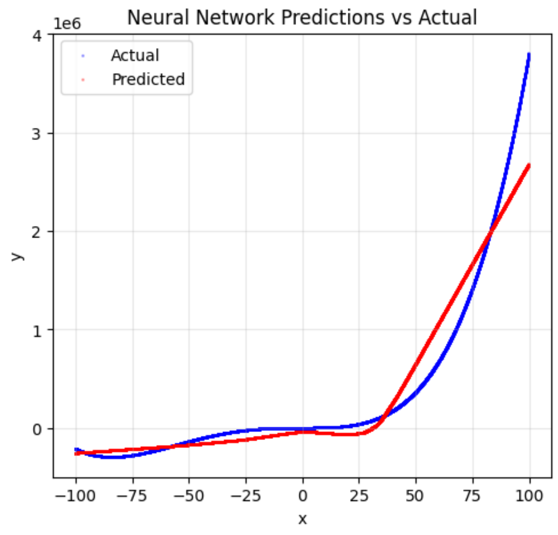
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.390969
Final Test Loss:     0.397851
R² Score:           0.6057
Total Epochs Run:   600
```

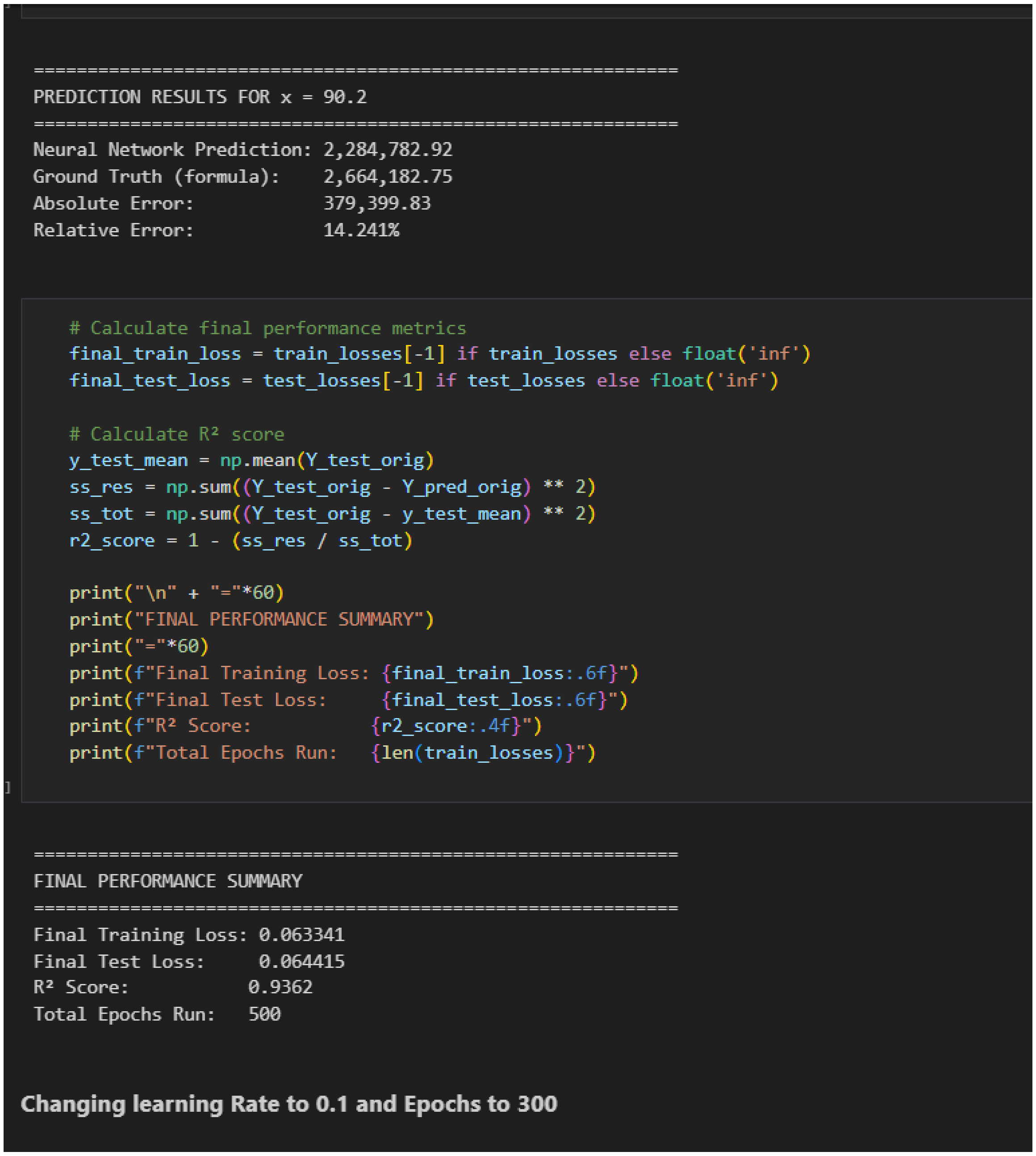
#Experiment 2: Changing the learning rate #



```
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 = 0.657215, Test Loss = 0.655221
Epoch  40: Train Loss = 0.490763, Test Loss = 0.492484
Epoch  60: Train Loss = 0.393831, Test Loss = 0.397621
Epoch  80: Train Loss = 0.337540, Test Loss = 0.341733
Epoch 100: Train Loss = 0.297644, Test Loss = 0.301717
Epoch 120: Train Loss = 0.263594, Test Loss = 0.267249
Epoch 140: Train Loss = 0.236802, Test Loss = 0.240341
Epoch 160: Train Loss = 0.214765, Test Loss = 0.218050
Epoch 180: Train Loss = 0.195633, Test Loss = 0.198689
Epoch 200: Train Loss = 0.178908, Test Loss = 0.181758
Epoch 220: Train Loss = 0.164210, Test Loss = 0.166872
Epoch 240: Train Loss = 0.151231, Test Loss = 0.153723
Epoch 260: Train Loss = 0.139717, Test Loss = 0.142056
Epoch 280: Train Loss = 0.129430, Test Loss = 0.131623
Epoch 300: Train Loss = 0.120226, Test Loss = 0.122284
Epoch 320: Train Loss = 0.111957, Test Loss = 0.113885
Epoch 340: Train Loss = 0.104471, Test Loss = 0.106273
Epoch 360: Train Loss = 0.097634, Test Loss = 0.099314
Epoch 380: Train Loss = 0.091321, Test Loss = 0.092886
...
Epoch 440: Train Loss = 0.075504, Test Loss = 0.076789
Epoch 460: Train Loss = 0.071140, Test Loss = 0.072347
Epoch 480: Train Loss = 0.067095, Test Loss = 0.068232
Epoch 500: Train Loss = 0.063341, Test Loss = 0.064415
```





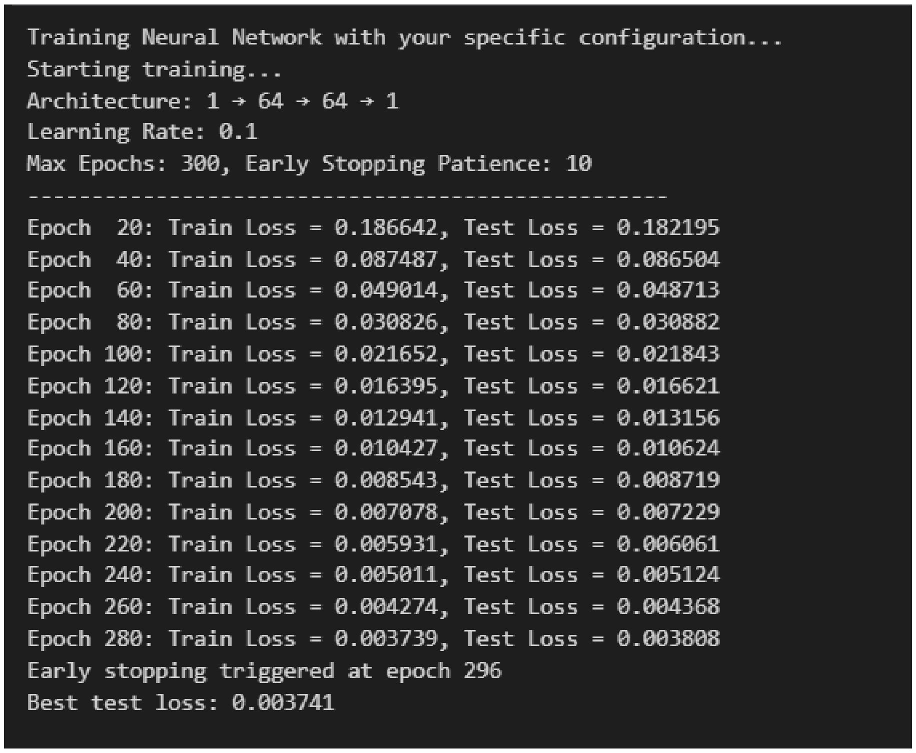


```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 2,284,782.92
Ground Truth (formula):    2,664,182.75
Absolute Error:            379,399.83
Relative Error:            14.241%
```

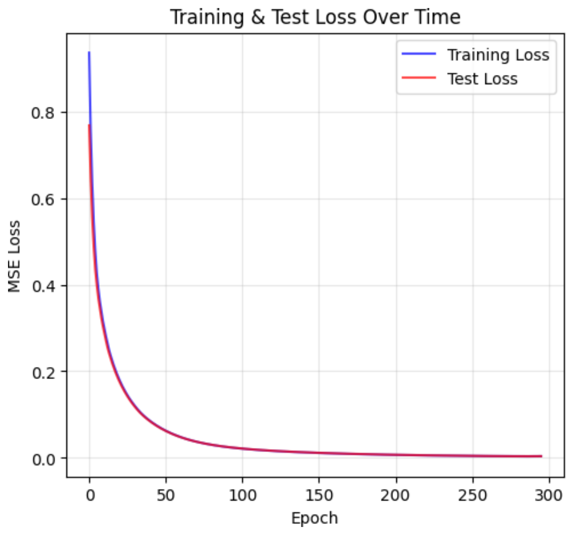
```python
# Calculate final performance metrics
final_train_loss = train_losses[-1] if train_losses else float('inf')
final_test_loss = test_losses[-1] if test_losses else float('inf')

# 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)}")
```

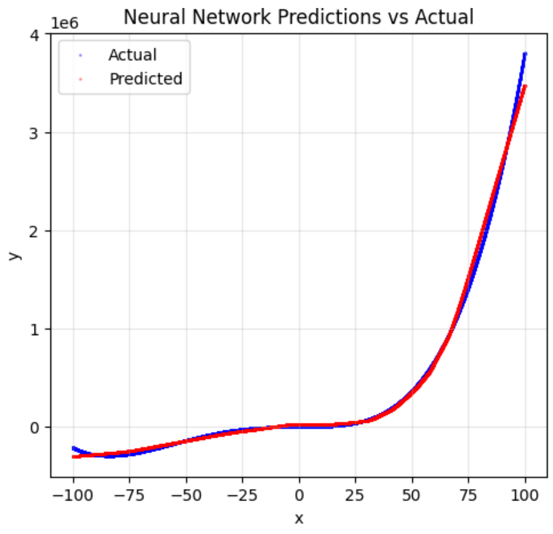
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.063341
Final Test Loss:     0.064415
R² Score:           0.9362
Total Epochs Run:   500
```

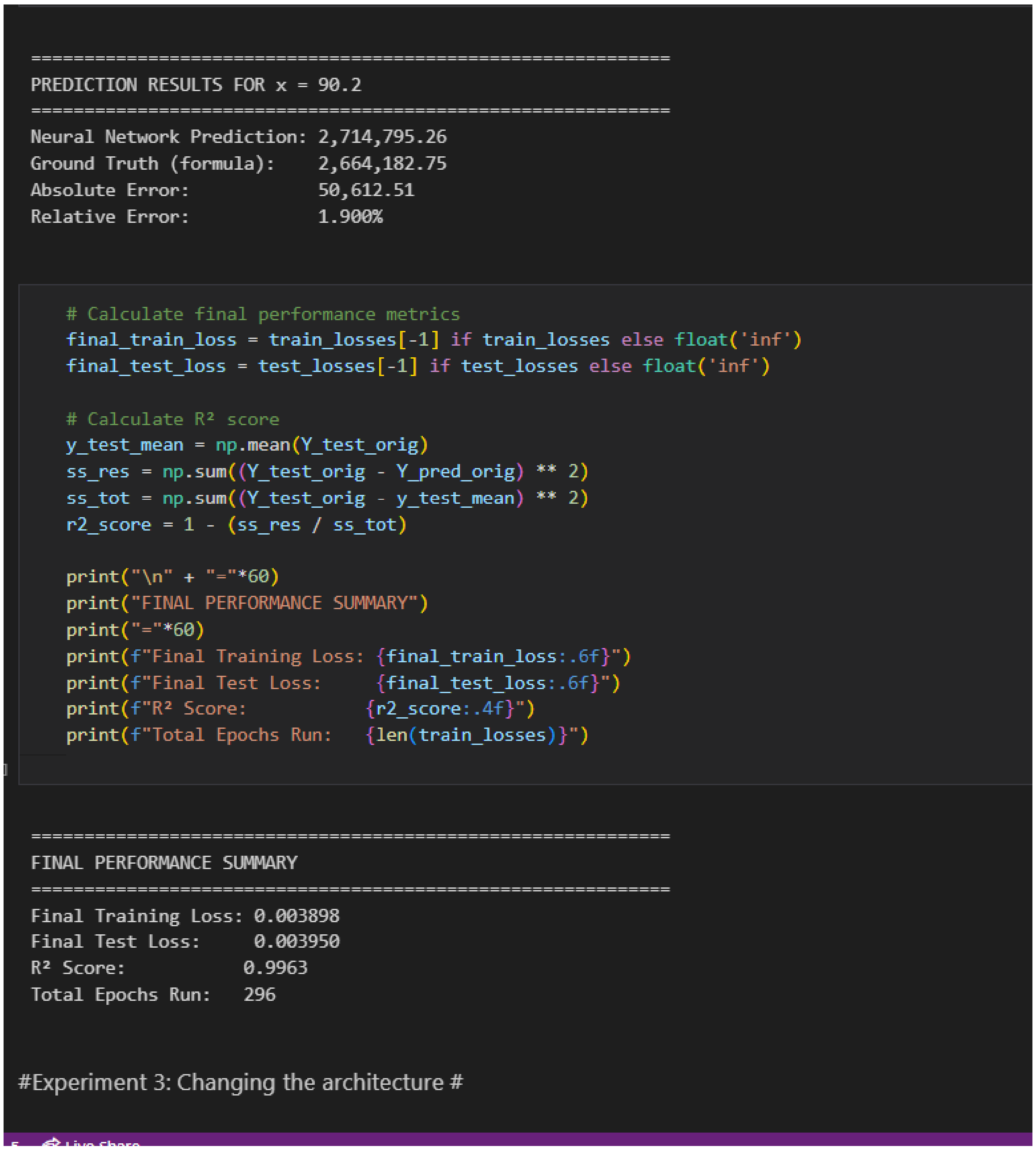
Changing learning Rate to 0.1 and Epochs to 300



```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 64 → 64 → 1
Learning Rate: 0.1
Max Epochs: 300, Early Stopping Patience: 10
--------------------------------------------------
Epoch  20: Train Loss = 0.186642, Test Loss = 0.182195
Epoch  40: Train Loss = 0.087487, Test Loss = 0.086504
Epoch  60: Train Loss = 0.049014, Test Loss = 0.048713
Epoch  80: Train Loss = 0.030826, Test Loss = 0.030882
Epoch 100: Train Loss = 0.021652, Test Loss = 0.021843
Epoch 120: Train Loss = 0.016395, Test Loss = 0.016621
Epoch 140: Train Loss = 0.012941, Test Loss = 0.013156
Epoch 160: Train Loss = 0.010427, Test Loss = 0.010624
Epoch 180: Train Loss = 0.008543, Test Loss = 0.008719
Epoch 200: Train Loss = 0.007078, Test Loss = 0.007229
Epoch 220: Train Loss = 0.005931, Test Loss = 0.006061
Epoch 240: Train Loss = 0.005011, Test Loss = 0.005124
Epoch 260: Train Loss = 0.004274, Test Loss = 0.004368
Epoch 280: Train Loss = 0.003739, Test Loss = 0.003808
Early stopping triggered at epoch 296
Best test loss: 0.003741
```





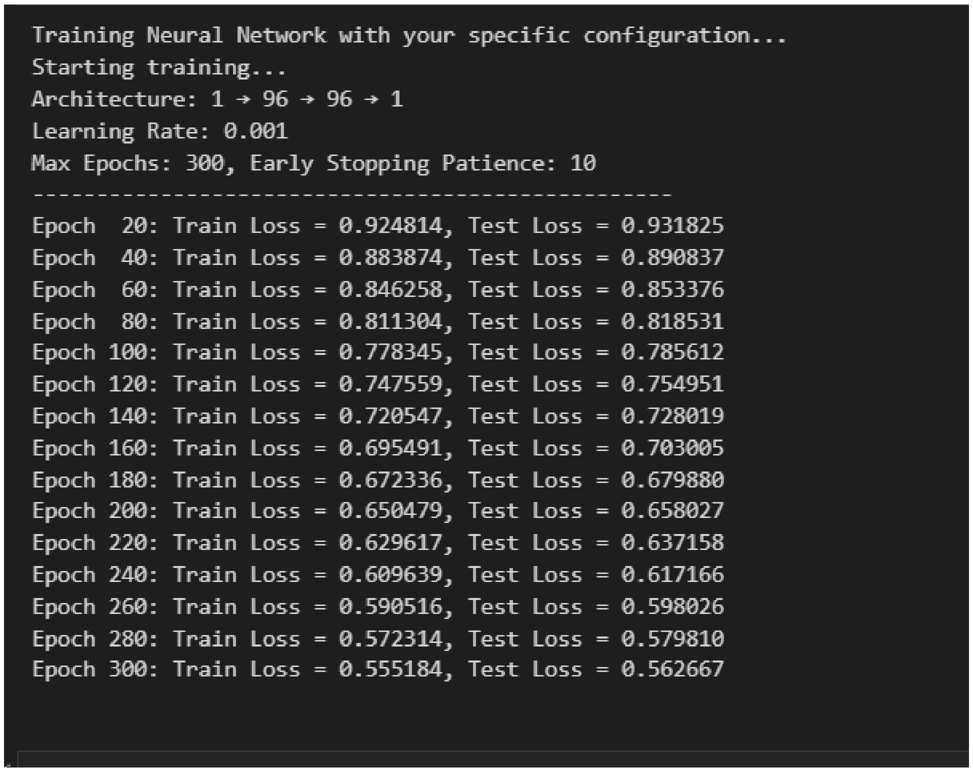


```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 2,714,795.26
Ground Truth (formula):    2,664,182.75
Absolute Error:            50,612.51
Relative Error:            1.900%
```

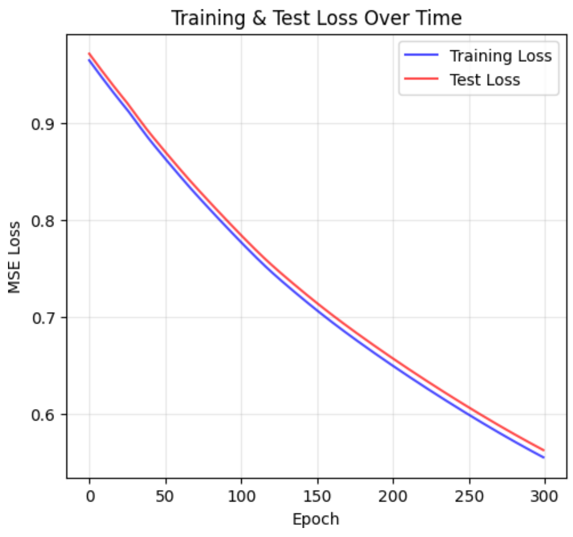
```python
# Calculate final performance metrics
final_train_loss = train_losses[-1] if train_losses else float('inf')
final_test_loss = test_losses[-1] if test_losses else float('inf')

# 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)}")
```

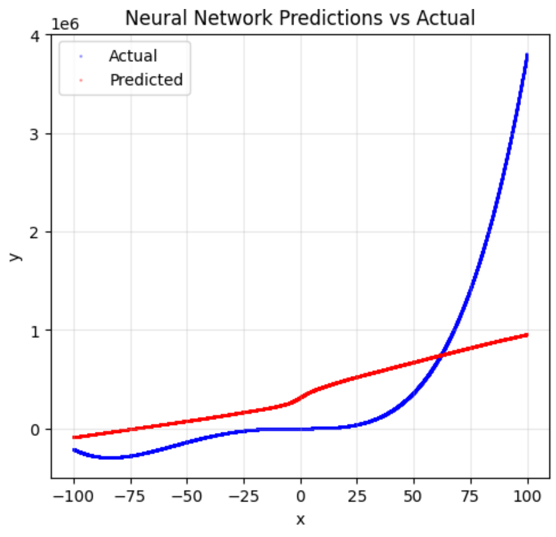
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.003898
Final Test Loss:     0.003950
R² Score:           0.9963
Total Epochs Run:   296
```

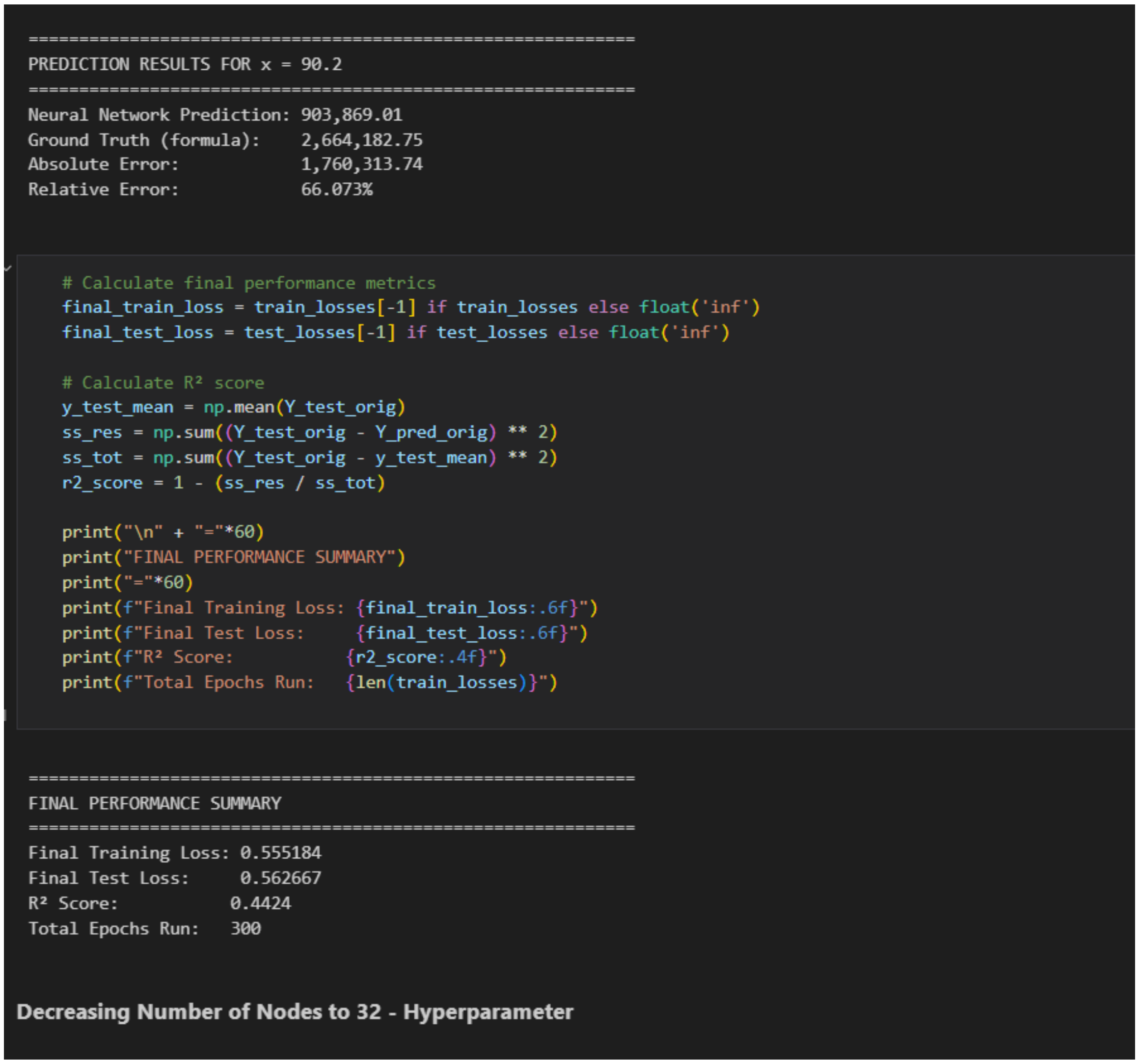
#Experiment 3: Changing the architecture #



```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 96 → 96 → 1
Learning Rate: 0.001
Max Epochs: 300, Early Stopping Patience: 10
--------------------------------------------------
Epoch  20: Train Loss = 0.924814, Test Loss = 0.931825
Epoch  40: Train Loss = 0.883874, Test Loss = 0.890837
Epoch  60: Train Loss = 0.846258, Test Loss = 0.853376
Epoch  80: Train Loss = 0.811304, Test Loss = 0.818531
Epoch 100: Train Loss = 0.778345, Test Loss = 0.785612
Epoch 120: Train Loss = 0.747559, Test Loss = 0.754951
Epoch 140: Train Loss = 0.720547, Test Loss = 0.728019
Epoch 160: Train Loss = 0.695491, Test Loss = 0.703005
Epoch 180: Train Loss = 0.672336, Test Loss = 0.679880
Epoch 200: Train Loss = 0.650479, Test Loss = 0.658027
Epoch 220: Train Loss = 0.629617, Test Loss = 0.637158
Epoch 240: Train Loss = 0.609639, Test Loss = 0.617166
Epoch 260: Train Loss = 0.590516, Test Loss = 0.598026
Epoch 280: Train Loss = 0.572314, Test Loss = 0.579810
Epoch 300: Train Loss = 0.555184, Test Loss = 0.562667
```



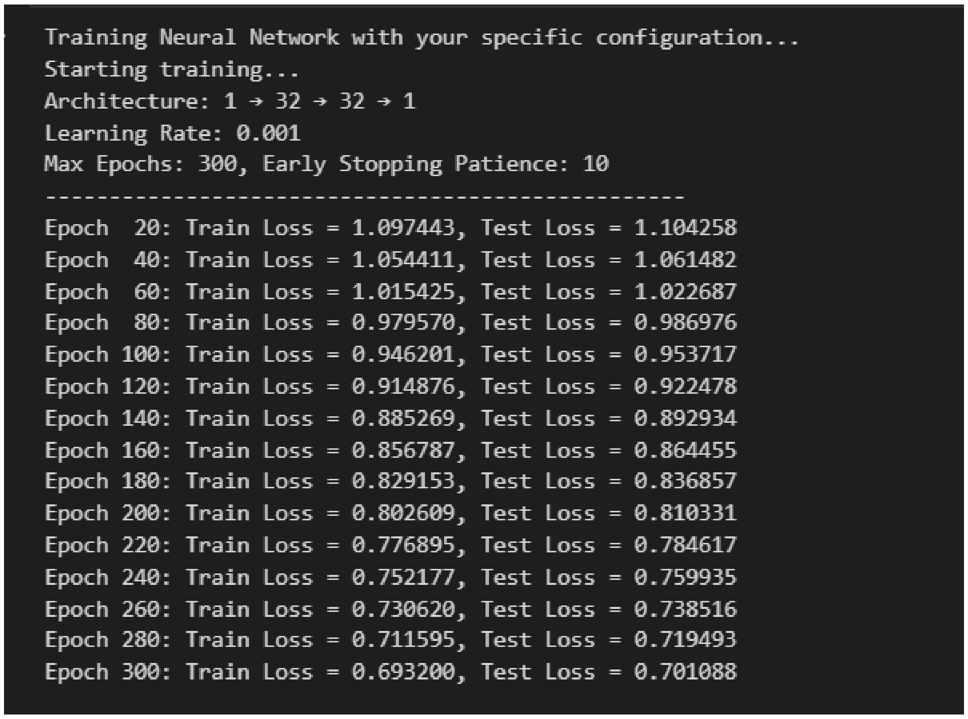



```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 903,869.01
Ground Truth (formula):    2,664,182.75
Absolute Error:            1,760,313.74
Relative Error:            66.073%
```

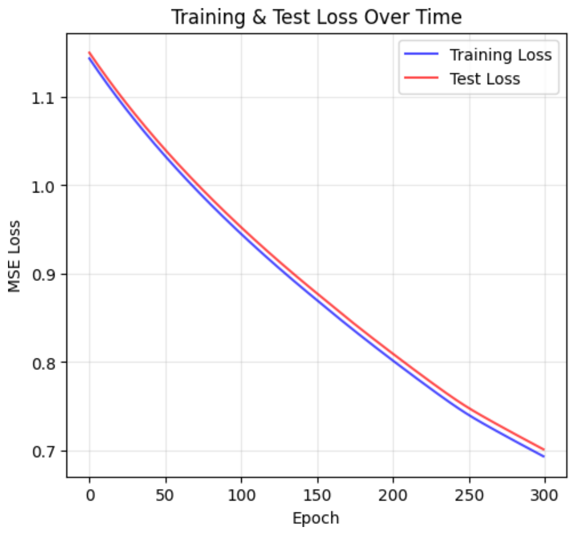
```
# Calculate final performance metrics
final_train_loss = train_losses[-1] if train_losses else float('inf')
final_test_loss = test_losses[-1] if test_losses else float('inf')

# 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)}")
```

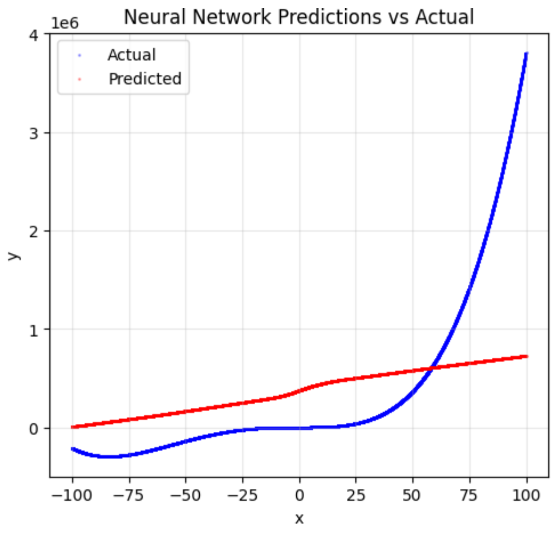
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.555184
Final Test Loss:     0.562667
R² Score:           0.4424
Total Epochs Run:   300
```

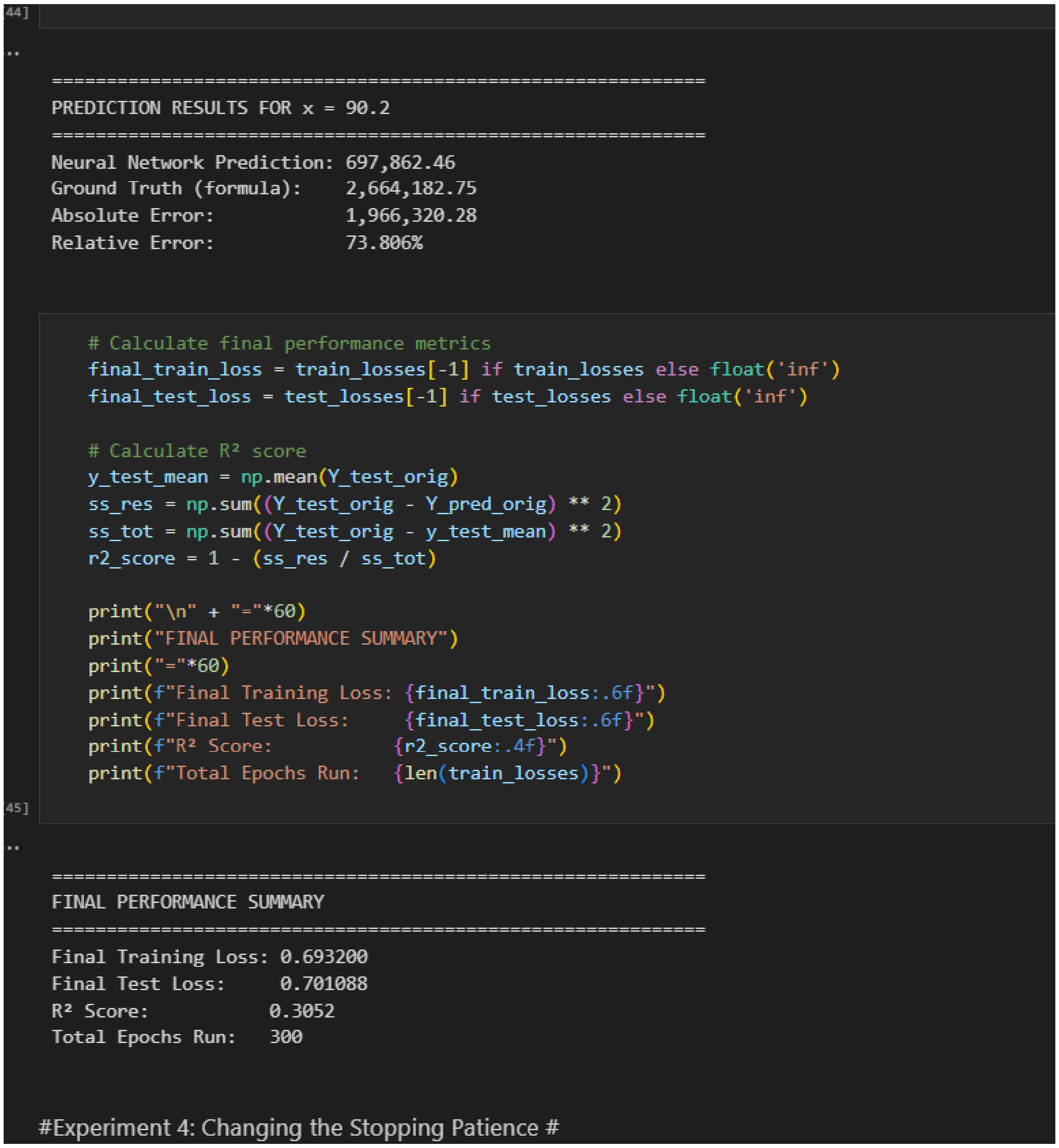
Decreasing Number of Nodes to 32 - Hyperparameter



```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 32 → 32 → 1
Learning Rate: 0.001
Max Epochs: 300, Early Stopping Patience: 10
--------------------------------------------------
Epoch  20: Train Loss = 1.097443, Test Loss = 1.104258
Epoch  40: Train Loss = 1.054411, Test Loss = 1.061482
Epoch  60: Train Loss = 1.015425, Test Loss = 1.022687
Epoch  80: Train Loss = 0.979570, Test Loss = 0.986976
Epoch 100: Train Loss = 0.946201, Test Loss = 0.953717
Epoch 120: Train Loss = 0.914876, Test Loss = 0.922478
Epoch 140: Train Loss = 0.885269, Test Loss = 0.892934
Epoch 160: Train Loss = 0.856787, Test Loss = 0.864455
Epoch 180: Train Loss = 0.829153, Test Loss = 0.836857
Epoch 200: Train Loss = 0.802609, Test Loss = 0.810331
Epoch 220: Train Loss = 0.776895, Test Loss = 0.784617
Epoch 240: Train Loss = 0.752177, Test Loss = 0.759935
Epoch 260: Train Loss = 0.730620, Test Loss = 0.738516
Epoch 280: Train Loss = 0.711595, Test Loss = 0.719493
Epoch 300: Train Loss = 0.693200, Test Loss = 0.701088
```



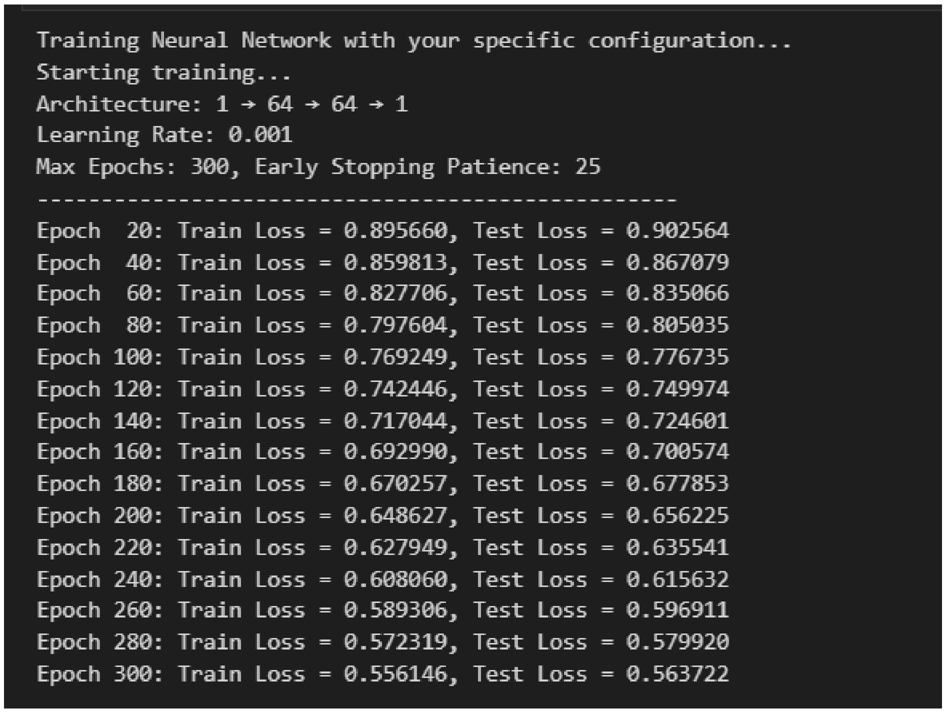



```
[44]
...

    ============================================================
    PREDICTION RESULTS FOR x = 90.2
    ============================================================
    Neural Network Prediction: 697,862.46
    Ground Truth (formula):    2,664,182.75
    Absolute Error:            1,966,320.28
    Relative Error:            73.806%
```

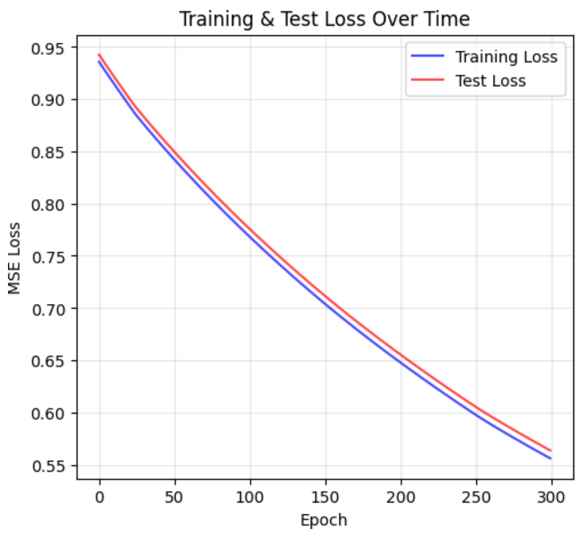
```python
# Calculate final performance metrics
final_train_loss = train_losses[-1] if train_losses else float('inf')
final_test_loss = test_losses[-1] if test_losses else float('inf')

# 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)}")
```

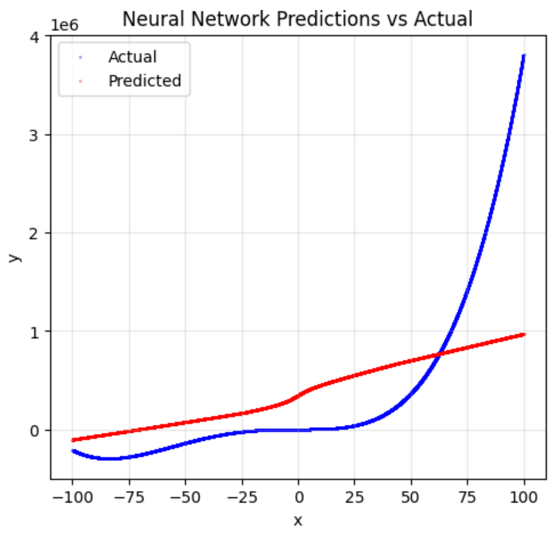
```
[45]
...

    ============================================================
    FINAL PERFORMANCE SUMMARY
    ============================================================
    Final Training Loss: 0.693200
    Final Test Loss:     0.701088
    R² Score:           0.3052
    Total Epochs Run:   300
```

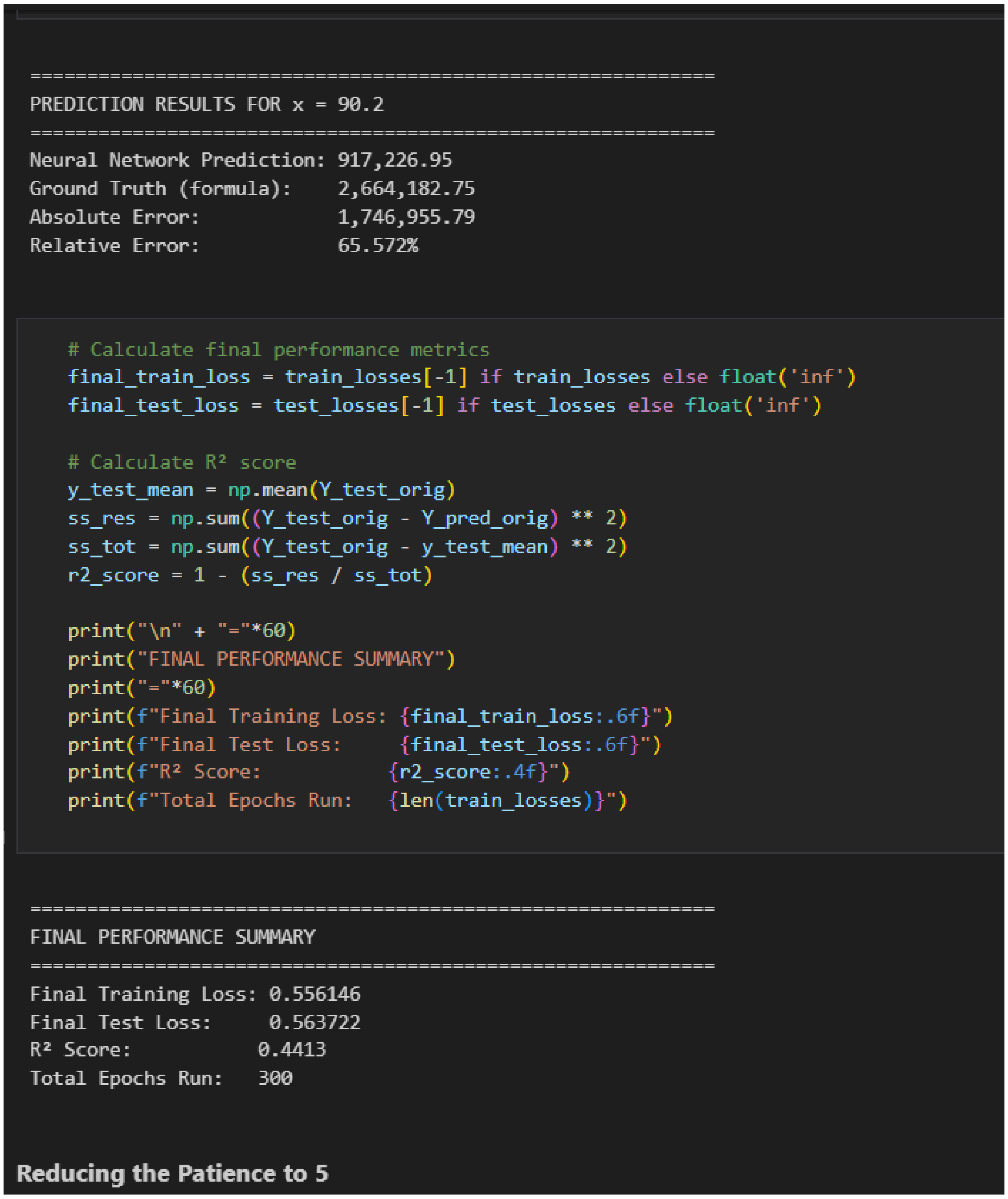
#Experiment 4: Changing the Stopping Patience #



```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 64 → 64 → 1
Learning Rate: 0.001
Max Epochs: 300, Early Stopping Patience: 25
--------------------------------------------------
Epoch  20: Train Loss = 0.895660, Test Loss = 0.902564
Epoch  40: Train Loss = 0.859813, Test Loss = 0.867079
Epoch  60: Train Loss = 0.827706, Test Loss = 0.835066
Epoch  80: Train Loss = 0.797604, Test Loss = 0.805035
Epoch 100: Train Loss = 0.769249, Test Loss = 0.776735
Epoch 120: Train Loss = 0.742446, Test Loss = 0.749974
Epoch 140: Train Loss = 0.717044, Test Loss = 0.724601
Epoch 160: Train Loss = 0.692990, Test Loss = 0.700574
Epoch 180: Train Loss = 0.670257, Test Loss = 0.677853
Epoch 200: Train Loss = 0.648627, Test Loss = 0.656225
Epoch 220: Train Loss = 0.627949, Test Loss = 0.635541
Epoch 240: Train Loss = 0.608060, Test Loss = 0.615632
Epoch 260: Train Loss = 0.589306, Test Loss = 0.596911
Epoch 280: Train Loss = 0.572319, Test Loss = 0.579920
Epoch 300: Train Loss = 0.556146, Test Loss = 0.563722
```





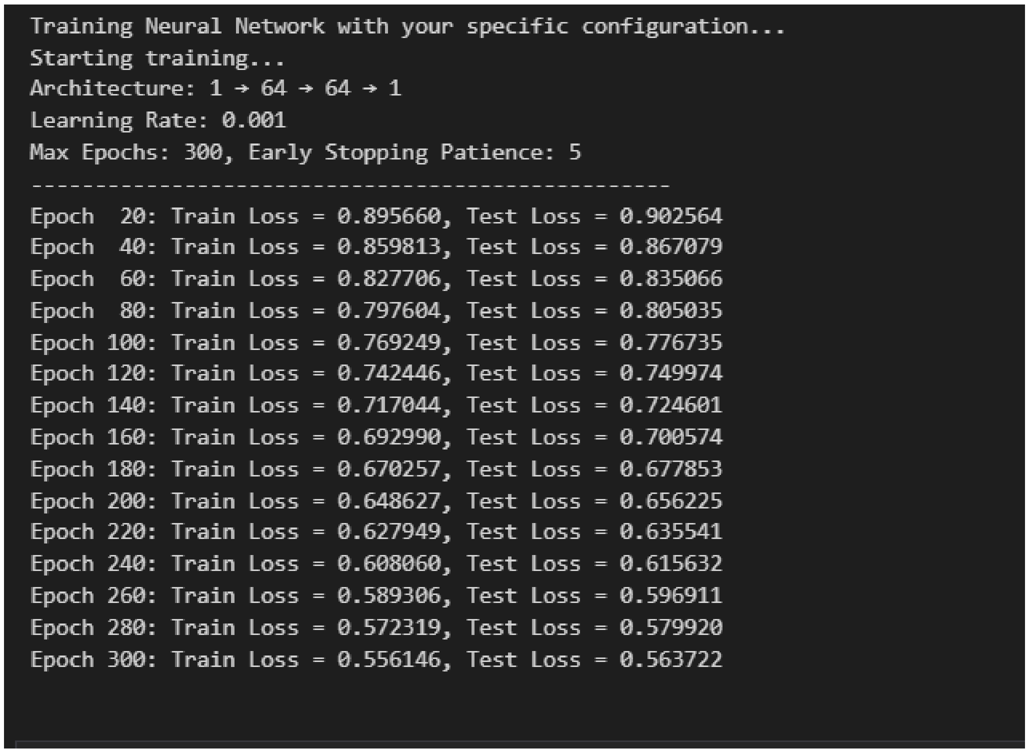


```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 917,226.95
Ground Truth (formula):    2,664,182.75
Absolute Error:            1,746,955.79
Relative Error:            65.572%
```

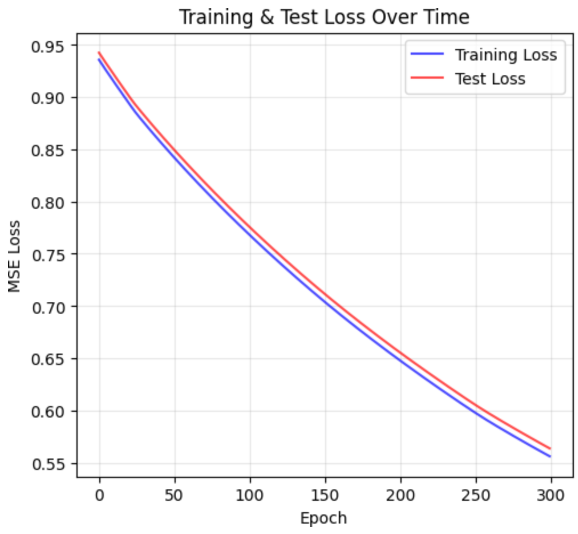
```python
    # Calculate final performance metrics
    final_train_loss = train_losses[-1] if train_losses else float('inf')
    final_test_loss = test_losses[-1] if test_losses else float('inf')

    # 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)}")
```

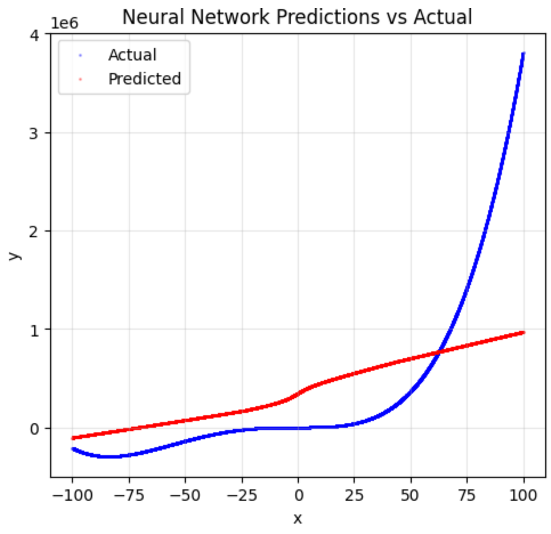
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.556146
Final Test Loss:     0.563722
R² Score:           0.4413
Total Epochs Run:   300
```

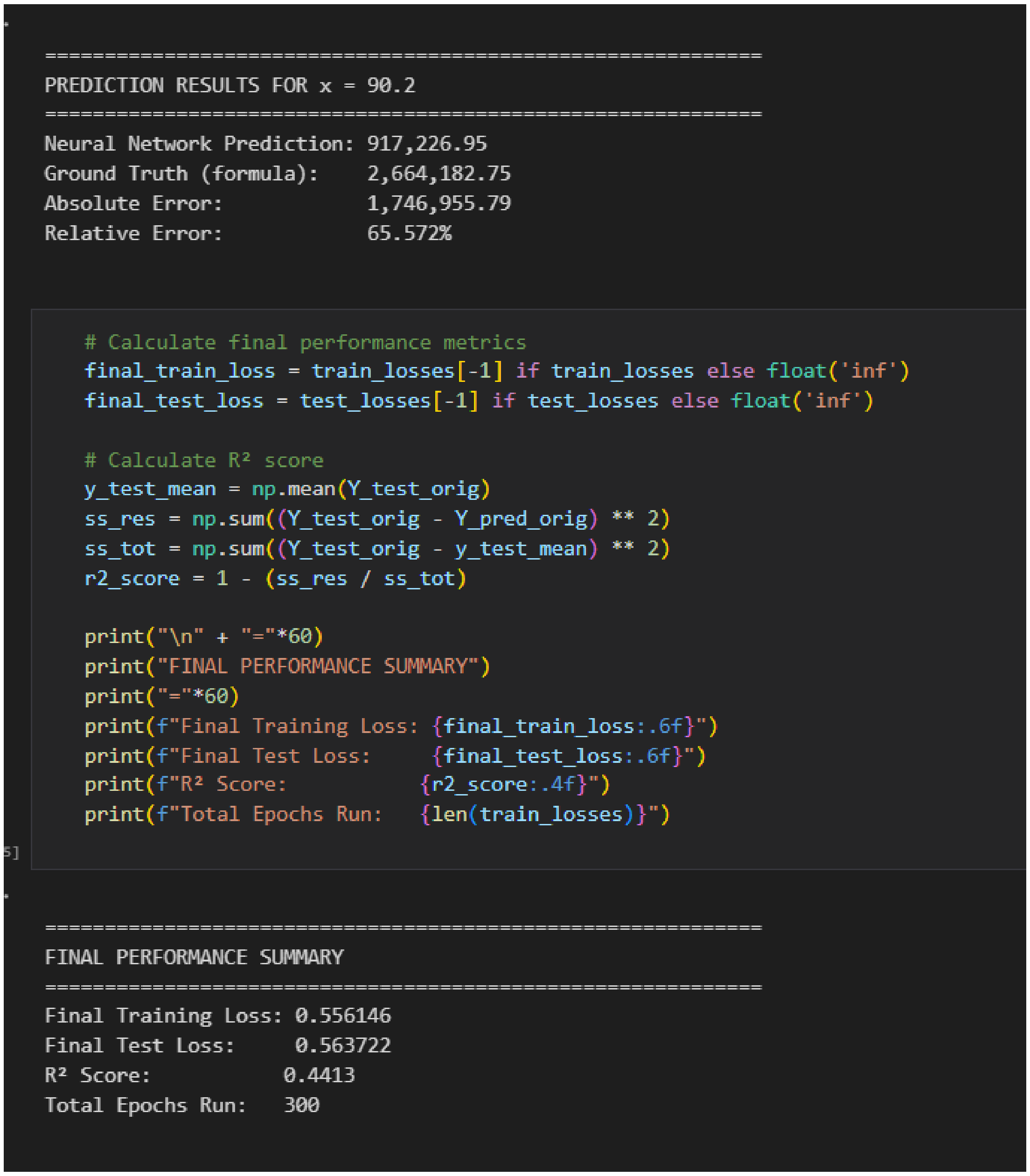
Reducing the Patience to 5



```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 64 → 64 → 1
Learning Rate: 0.001
Max Epochs: 300, Early Stopping Patience: 5
--------------------------------------------------
Epoch  20: Train Loss = 0.895660, Test Loss = 0.902564
Epoch  40: Train Loss = 0.859813, Test Loss = 0.867079
Epoch  60: Train Loss = 0.827706, Test Loss = 0.835066
Epoch  80: Train Loss = 0.797604, Test Loss = 0.805035
Epoch 100: Train Loss = 0.769249, Test Loss = 0.776735
Epoch 120: Train Loss = 0.742446, Test Loss = 0.749974
Epoch 140: Train Loss = 0.717044, Test Loss = 0.724601
Epoch 160: Train Loss = 0.692990, Test Loss = 0.700574
Epoch 180: Train Loss = 0.670257, Test Loss = 0.677853
Epoch 200: Train Loss = 0.648627, Test Loss = 0.656225
Epoch 220: Train Loss = 0.627949, Test Loss = 0.635541
Epoch 240: Train Loss = 0.608060, Test Loss = 0.615632
Epoch 260: Train Loss = 0.589306, Test Loss = 0.596911
Epoch 280: Train Loss = 0.572319, Test Loss = 0.579920
Epoch 300: Train Loss = 0.556146, Test Loss = 0.563722
```



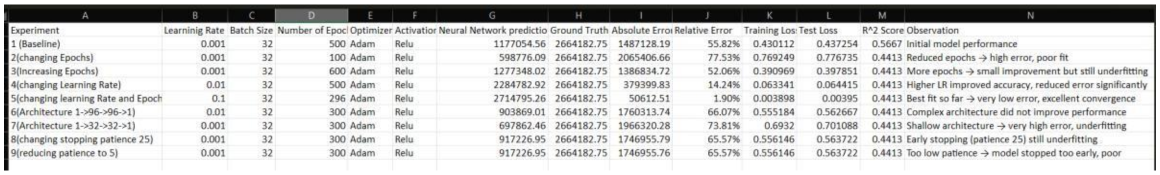




```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 917,226.95
Ground Truth (formula):    2,664,182.75
Absolute Error:            1,746,955.79
Relative Error:            65.572%
```

```python
# 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: 0.556146
Final Test Loss:     0.563722
R² Score:           0.4413
Total Epochs Run:   300
```



A	B	C	D	E	F	G	H	I	J	K	L	M	N
Experiment	Learninig Rate	Batch Size	Number of Epoc	Optimizer	Activatior	Neural Network predictio	Ground Truth	Absolute Erroi	Relative Error	Training Los:	Test Loss	R^2 Score	Observation
1 (Baseline)	0.001	32	500	Adam	Relu	1177054.56	2664182.75	1487128.19	55.82%	0.430112	0.437254	0.5667	Initial model performance
2(changing Epochs)	0.001	32	100	Adam	Relu	598776.09	2664182.75	2065406.66	77.53%	0.769249	0.776735	0.4413	Reduced epochs → high error, poor fit
3(Increasing Epochs)	0.001	32	600	Adam	Relu	1277348.02	2664182.75	1386834.72	52.06%	0.390969	0.397851	0.4413	More epochs → small improvement but still underfitting
4(changing Learning Rate)	0.01	32	500	Adam	Relu	2284782.92	2664182.75	379399.83	14.24%	0.063341	0.064415	0.4413	Higher LR improved accuracy, reduced error significantly
5(changing learning Rate and Epoch	0.1	32	296	Adam	Relu	2714795.26	2664182.75	50612.51	1.90%	0.003898	0.00395	0.4413	Best fit so far → very low error, excellent convergence
6(Architecture 1->96->96->1)	0.01	32	300	Adam	Relu	903869.01	2664182.75	1760313.74	66.07%	0.555184	0.562667	0.4413	Complex architecture did not improve performance
7(Architecture 1->32->32->1)	0.001	32	300	Adam	Relu	697862.46	2664182.75	1966320.28	73.81%	0.6932	0.701088	0.4413	Shallow architecture → very high error, underfitting
8(changing stopping patience 25)	0.001	32	300	Adam	Relu	917226.95	2664182.75	1746955.79	65.57%	0.556146	0.563722	0.4413	Early stopping (patience 25) still underfitting
9(reducing patience to 5)	0.001	32	300	Adam	Relu	917226.95	2664182.75	1746955.76	65.57%	0.556146	0.563722	0.4413	Too low patience → model stopped too early, poor

5. Conclusion

. By tuning hyperparameters such as learning rate, Epochs model performance can be optimized. The results indicate the balance between bias and variance, highlighting issues of overfitting and underfitting depending on the configuration.

The **baseline model** had moderate performance but a high error (~55%).

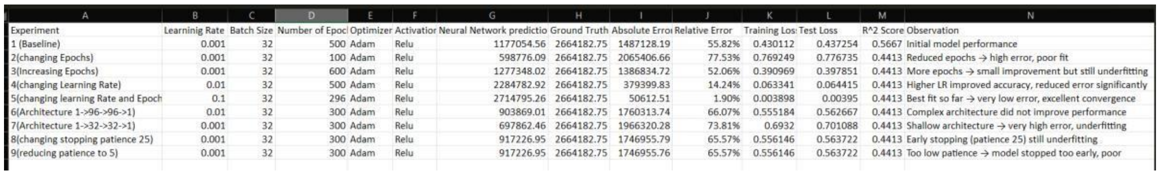
Reducing epochs led to severe underfitting (error > 75%), showing that too few epochs don't allow the model to learn enough.

Increasing epochs helped slightly but did not solve underfitting, meaning just training longer was not sufficient.

Increasing learning rate (0.01 and 0.1) drastically improved performance, with **Experiment 5 (LR = 0.1, Epochs = 296)** achieving the **best results**: only **1.9% relative error**, very low losses, and fast convergence.

Changing architectures (Experiment 6 and 7) did not improve results, and in fact worsened the error, suggesting that the chosen dataset may not need deeper or wider networks.

Early stopping experiments (patience 25 and 5) still showed underfitting, meaning stopping too early harms learning.



A	B	C	D	E	F	G	H	I	J	K	L	M	N
Experiment	Learninig Rate	Batch Size	Number of Epoc	Optimizer	Activatior	Neural Network predictio	Ground Truth	Absolute Erroi	Relative Error	Training Los	Test Loss	R^2 Score	Observation
1 (Baseline)	0.001	32	500	Adam	Relu	1177054.56	2664182.75	1487128.19	55.82%	0.430112	0.437254	0.5667	Initial model performance
2(changing Epochs)	0.001	32	100	Adam	Relu	598776.09	2664182.75	2065406.66	77.53%	0.769249	0.776735	0.4413	Reduced epochs → high error, poor fit
3(Increasing Epochs)	0.001	32	600	Adam	Relu	1277348.02	2664182.75	1386834.72	52.06%	0.390969	0.397851	0.4413	More epochs → small improvement but still underfitting
4(changing Learning Rate)	0.01	32	500	Adam	Relu	2284782.92	2664182.75	379399.83	14.24%	0.063341	0.064415	0.4413	Higher LR improved accuracy, reduced error significantly
5(changing learning Rate and Epoch	0.1	32	296	Adam	Relu	2714795.26	2664182.75	50612.51	1.90%	0.003898	0.00395	0.4413	Best fit so far → very low error, excellent convergence
6(Architecture 1->96->96->1)	0.01	32	300	Adam	Relu	903869.01	2664182.75	1760313.74	66.07%	0.555184	0.562667	0.4413	Complex architecture did not improve performance
7(Architecture 1->32->32->1)	0.001	32	300	Adam	Relu	697862.46	2664182.75	1966320.28	73.81%	0.6932	0.701088	0.4413	Shallow architecture → very high error, underfitting
8(changing stopping patience 25)	0.001	32	300	Adam	Relu	917226.95	2664182.75	1746955.79	65.57%	0.556146	0.563722	0.4413	Early stopping (patience 25) still underfitting
9(reducing patience to 5)	0.001	32	300	Adam	Relu	917226.95	2664182.75	1746955.76	65.57%	0.556146	0.563722	0.4413	Too low patience → model stopped too early, poor