Lab Report

Project Title: Neural Networks for Function Approximation

• Name: BOJJA RAKSHITHA

• SRN: PES1UG23CS134

• Course: UE23CS352A: Machine Learning

• Date: September 20, 2025

Purpose of the Lab

The objective was to implement an Artificial Neural Network (ANN) from scratch using NumPy to approximate a cubic polynomial with an inverse term using a large, synthetic dataset. The network was trained and analyzed under various hyperparameter settings to observe their effects on convergence and final accuracy.

Introduction

This report documents the two main tasks:

Part A: Fully implement a baseline neural network.

Part B: Conduct systematic experiments on hyperparameters: learning rate, epochs, and activation function.

Dataset Description

Function Assigned: $y = 2.37x^3 + 0.33x^2 + 4.88x + 11.53 + 149.8/x$

Samples: 100,000 total (80% train, 20% test)

Input Features: 1 (scalar x)

Noise: Gaussian, mean=0, std=2.38

Preprocessing: Both x and y standardized using StandardScaler from sklearn before model input.

Network Architecture

Input Layer: 1 neuron

Hidden Layer 1: 72 neurons (ReLU)

Hidden Layer 2: 32 neurons (ReLU)

Output Layer: 1 neuron (Linear)

Initialization: Xavier/Glorot (normal distribution), biases to zero.

Methodology

Implement forward and backward propagation routines for a shallow feedforward network.

Use Mean Squared Error (MSE) loss function for regression.

Optimize with batch gradient descent. All core numerical routines were built “from scratch” using NumPy, including ReLU activation and MSE derivatives.

Early stopping was employed to prevent overfitting, halting if test loss stagnated for a set ‘patience’ value.

Implementation Details

The network was trained and tested on the standardized dataset.

The Baseline model used a learning rate of 0.001 and was trained for 500 epochs.

Key routines for dataset generation, preprocessing, forward pass, backward pass, and gradient updates were directly implemented in the code.

Baseline Model Performance

Metric	Value
Training Loss	0.432816
Test Loss	0.426525
R^2 Score	0.5725
Total Epochs	500

The training/test loss curve steadily decreased. The model tracked the cubic trend but exhibited clear underfitting, as reflected by the modest R^2 score.

Experiment Results Table

Csv: "neural_network_hyperparameter_experiments.csv" and same in the output screenshot…included in github.

Training Curves & Experiment Plots

All runs produced loss curves for both training and test sets, as well as scatter and residual plots of predictions versus ground truth, similar to the template.
Residuals were centered but with outliers, suggesting that the model's expressive power was limited by architecture or training duration.

Analysis of Experiments

Higher LR (0.01): Improved both convergence speed and final error.

Lower LR (0.0001): Caused extremely slow learning, resulting in high error.

More Epochs (1000, LR=0.001): Allowed the model to approach lower loss but did not match the gains of a higher LR.

Sigmoid Activation: Performance collapsed due to vanishing gradients, confirming ReLU's superiority for this architecture.

Output screenshots:

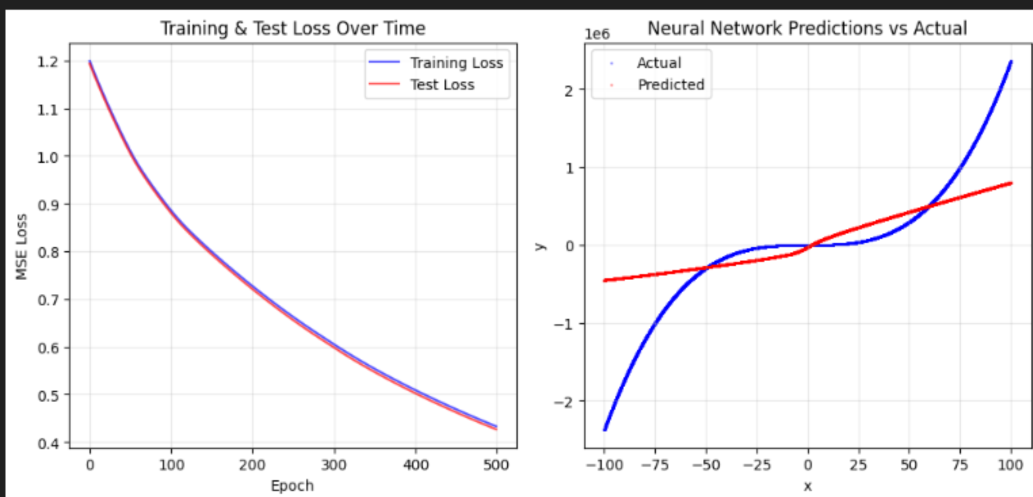
EXECUTE TRAINING:

```
    )
[30]  ✓ 3m 16.4s

...  Training Neural Network with your specific configuration...
     Starting training...
     Architecture: 1 → 72 → 32 → 1
     Learning Rate: 0.001
     Max Epochs: 500, Early Stopping Patience: 10
     --------------------------------------------------
     Epoch  20: Train Loss = 1.121277, Test Loss = 1.115333
     Epoch  40: Train Loss = 1.047425, Test Loss = 1.041544
     Epoch  60: Train Loss = 0.983189, Test Loss = 0.977707
     Epoch  80: Train Loss = 0.932538, Test Loss = 0.927082
     Epoch 100: Train Loss = 0.887650, Test Loss = 0.882111
     Epoch 120: Train Loss = 0.849060, Test Loss = 0.843574
     Epoch 140: Train Loss = 0.816132, Test Loss = 0.810413
     Epoch 160: Train Loss = 0.785380, Test Loss = 0.779414
     Epoch 180: Train Loss = 0.756173, Test Loss = 0.749980
     Epoch 200: Train Loss = 0.728192, Test Loss = 0.721802
     Epoch 220: Train Loss = 0.701370, Test Loss = 0.694817
     Epoch 240: Train Loss = 0.675743, Test Loss = 0.669066
     Epoch 260: Train Loss = 0.651362, Test Loss = 0.644595
     Epoch 280: Train Loss = 0.628269, Test Loss = 0.621448
     Epoch 300: Train Loss = 0.605905, Test Loss = 0.599035
     Epoch 320: Train Loss = 0.584276, Test Loss = 0.577453
     Epoch 340: Train Loss = 0.563963, Test Loss = 0.557205
     Epoch 360: Train Loss = 0.544898, Test Loss = 0.538190
     Epoch 380: Train Loss = 0.526792, Test Loss = 0.520112
     ...
     Epoch 440: Train Loss = 0.477056, Test Loss = 0.470532
     Epoch 460: Train Loss = 0.461725, Test Loss = 0.455270
     Epoch 480: Train Loss = 0.446974, Test Loss = 0.440597
     Epoch 500: Train Loss = 0.432816, Test Loss = 0.426525
```

RESULTS VISUALIZATION:

```python
# Residual plot
# plt.subplot(1, 3, 3)
# residuals = Y_test_orig.flatten() - Y_pred_orig.flatten()
# plt.scatter(X_test_orig, residuals, s=1, alpha=0.3, color='green')
# plt.axhline(y=0, color='black', linestyle='--', alpha=0.5)
# plt.xlabel('x')
# plt.ylabel('Residuals (Actual - Predicted)')
# plt.title('Residual Analysis')
# plt.grid(True, alpha=0.3)

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

✓ 1.1s Python



⚠ 0 Spaces: 4 Finish Setup Cell 20 of 29 Go Live Go Live

PREDICTION RESULTS:

```python
print("\n" + "="*60)
print("PREDICTION RESULTS FOR x = 90.2")
print("="*60)
print(f"Neural Network Prediction: {y_pred[0][0]:,.2f}")
print(f"Ground Truth (formula):    {y_true:,.2f}")
print(f"Absolute Error:            {abs(y_pred[0][0] - y_true):,.2f}")
print(f"Relative Error:            {abs(y_pred[0][0] - y_true)/abs(y_true)*100:.3
```

[32] ✓ 0.0s Python

```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 729,616.22
Ground Truth (formula):    1,735,575.41
Absolute Error:            1,005,959.19
Relative Error:            57.961%
```

0 ⚠ 0 Spaces: 4 Finish Setup Cell 20 of 29 Go Live Go Live

PERFORMANCE METRICS:

```python
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)}")
```

[33] ✓ 0.0s Python

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.432816
Final Test Loss:     0.426525
R² Score:           0.5725
Total Epochs Run:   500
```

```python
# ==================================================================
# IMPORTS + VARIABLES FROM PART A
```

PART B:

HYPERPARAMETER EXPLORATION

EXPERIMENT 1:

==

RUNNING Exp 1: Higher LR

==

Learning Rate: 0.01

Architecture: 1 → 72 → 32 → 1

Max Epochs: 300

Early Stopping Patience: 15

Training started...

Epoch 50: Train Loss = 0.439173, Test Loss = 0.426555

Epoch 100: Train Loss = 0.237837, Test Loss = 0.232537

Epoch 150: Train Loss = 0.192418, Test Loss = 0.189395

Epoch 200: Train Loss = 0.171321, Test Loss = 0.168835

Epoch 250: Train Loss = 0.153157, Test Loss = 0.151003

Epoch 300: Train Loss = 0.135157, Test Loss = 0.133268

Exp 1: Higher LR RESULTS:

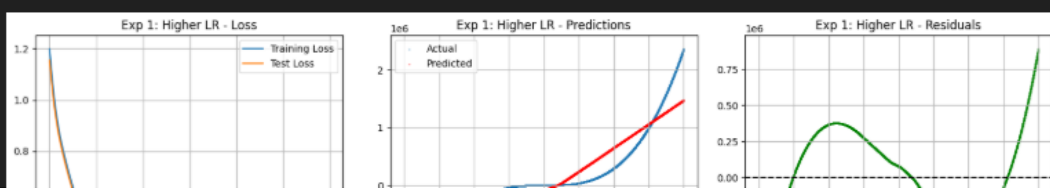
Final Train Loss: 0.135157

Final Test Loss: 0.133268

R^2 Score: 0.8664

Accuracy: 86.64%

Epochs Completed: 300



Epoch 300: Train Loss = 0.135157, Test Loss = 0.133268

Exp 1: Higher LR RESULTS:

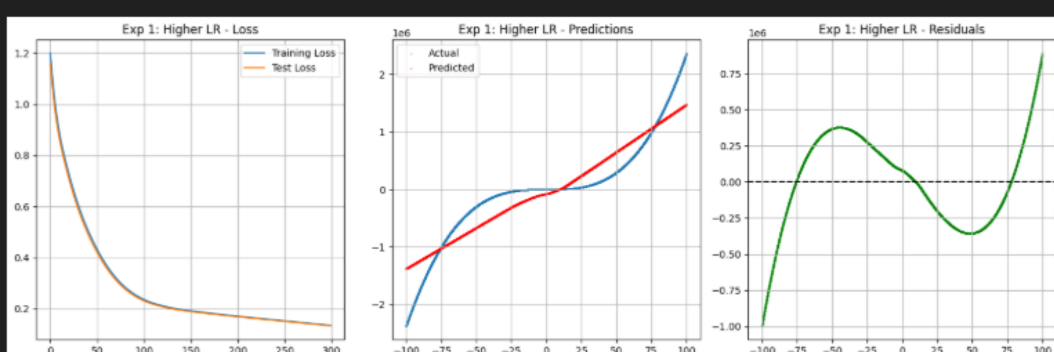
Final Train Loss: 0.135157

Final Test Loss: 0.133268

R^2 Score: 0.8664

Accuracy: 86.64%

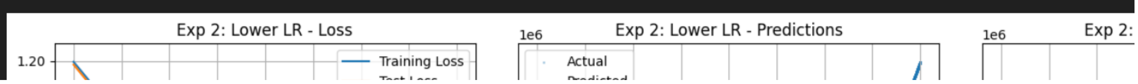
Epochs Completed: 300



EXPERIMENT 2:

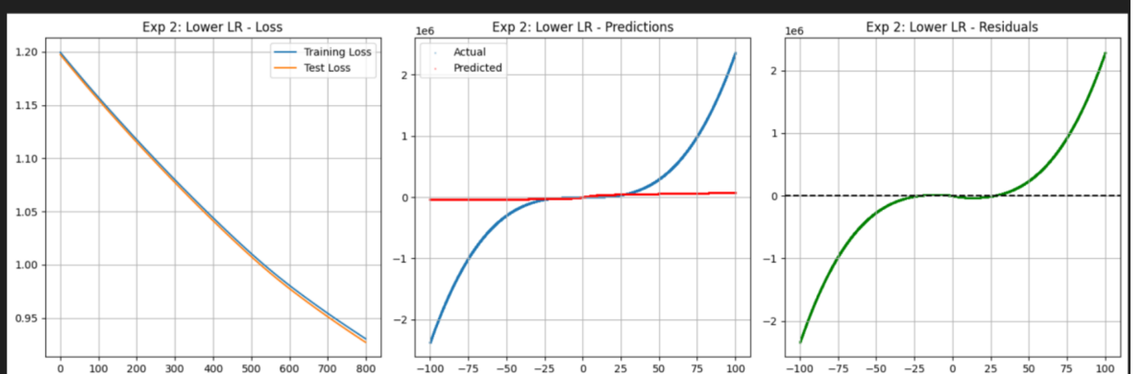
```
RUNNING Exp 2: Lower LR
======================================================================
Learning Rate: 0.0001
Architecture: 1 → 72 → 32 → 1
Max Epochs: 800
Early Stopping Patience: 20

Training started...
Epoch  50: Train Loss = 1.178338, Test Loss = 1.176146
Epoch 100: Train Loss = 1.157556, Test Loss = 1.155274
Epoch 150: Train Loss = 1.137461, Test Loss = 1.135092
Epoch 200: Train Loss = 1.117880, Test Loss = 1.115429
Epoch 250: Train Loss = 1.098792, Test Loss = 1.096262
Epoch 300: Train Loss = 1.080187, Test Loss = 1.077583
Epoch 350: Train Loss = 1.062073, Test Loss = 1.059401
Epoch 400: Train Loss = 1.044437, Test Loss = 1.041701
Epoch 450: Train Loss = 1.027310, Test Loss = 1.024512
Epoch 500: Train Loss = 1.010836, Test Loss = 1.007978
Epoch 550: Train Loss = 0.995281, Test Loss = 0.992367
Epoch 600: Train Loss = 0.980889, Test Loss = 0.977910
Epoch 650: Train Loss = 0.967482, Test Loss = 0.964415
Epoch 700: Train Loss = 0.954706, Test Loss = 0.951540
Epoch 750: Train Loss = 0.942450, Test Loss = 0.939194
...
Final Test Loss: 0.927256
R² Score: 0.0706
Accuracy: 7.06%
Epochs Completed: 800
```

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




```
Epoch 750: Train Loss = 0.942450, Test Loss = 0.939194
...
Final Test Loss: 0.927256
R² Score: 0.0706
Accuracy: 7.06%
Epochs Completed: 800
```

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

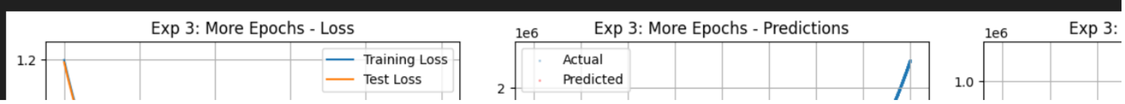


EXPERIMENT 3:

Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ···

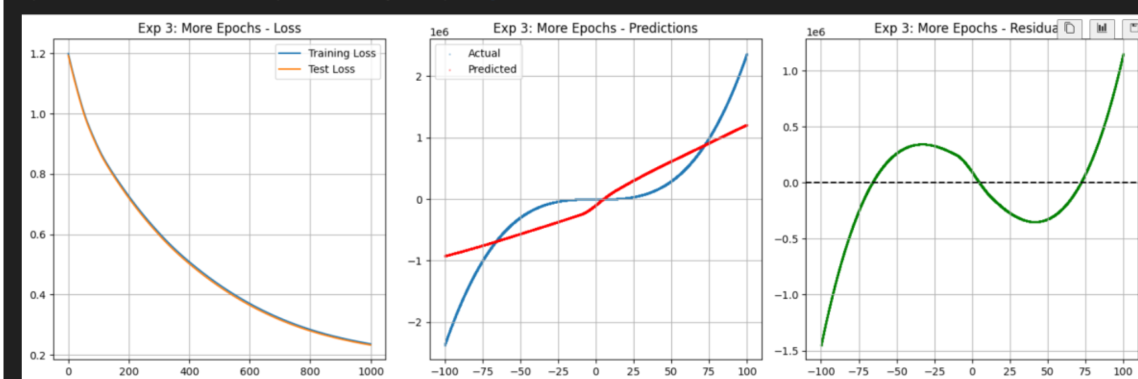
```
RUNNING Exp 3: More Epochs
======================================================================
Learning Rate: 0.001
Architecture: 1 → 72 → 32 → 1
Max Epochs: 1000
Early Stopping Patience: 25

Training started...
Epoch  50: Train Loss = 1.013556, Test Loss = 1.007785
Epoch 100: Train Loss = 0.887650, Test Loss = 0.882111
Epoch 150: Train Loss = 0.800527, Test Loss = 0.794681
Epoch 200: Train Loss = 0.728192, Test Loss = 0.721802
Epoch 250: Train Loss = 0.663394, Test Loss = 0.656667
Epoch 300: Train Loss = 0.605905, Test Loss = 0.599035
Epoch 350: Train Loss = 0.554287, Test Loss = 0.547552
Epoch 400: Train Loss = 0.509510, Test Loss = 0.502870
Epoch 450: Train Loss = 0.469318, Test Loss = 0.462828
Epoch 500: Train Loss = 0.432816, Test Loss = 0.426525
Epoch 550: Train Loss = 0.400015, Test Loss = 0.393965
Epoch 600: Train Loss = 0.370701, Test Loss = 0.364913
Epoch 650: Train Loss = 0.344636, Test Loss = 0.339123
Epoch 700: Train Loss = 0.321694, Test Loss = 0.316462
Epoch 750: Train Loss = 0.301701, Test Loss = 0.296749
...
Final Test Loss: 0.232662
R² Score: 0.7668
Accuracy: 76.68%
Epochs Completed: 1000
```

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




```
Epoch 750: Train Loss = 0.301701, Test Loss = 0.296749
...
Final Test Loss: 0.232662
R² Score: 0.7668
Accuracy: 76.68%
Epochs Completed: 1000
```

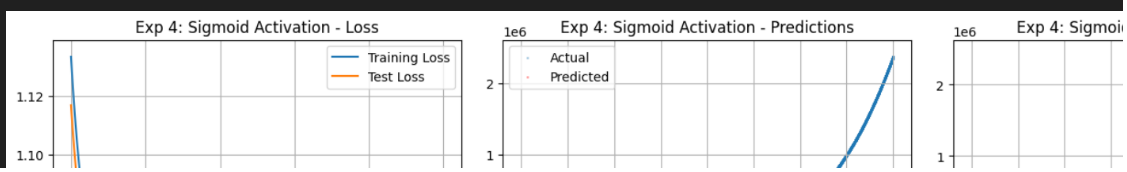
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...



EXPERIMENT 4:

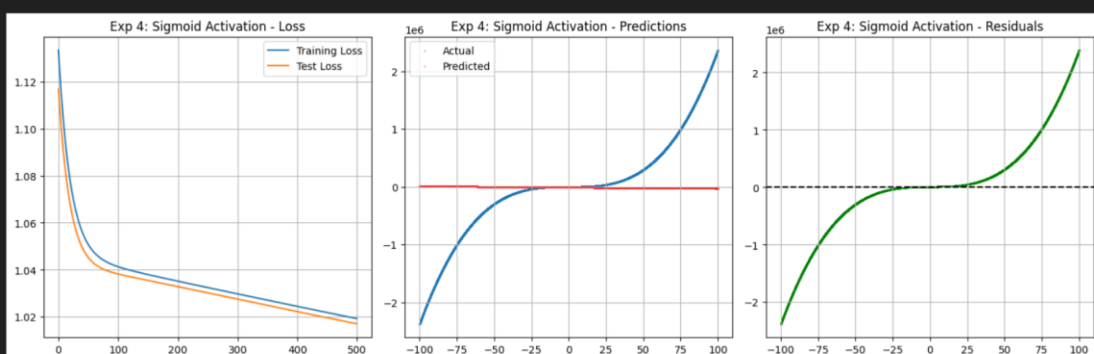
Generate + Code + Markdown | Run All | Restart | Clear All Outputs | Jupyter Variables | Outline ...

```
RUNNING Exp 4: Sigmoid Activation
======================================================================
Learning Rate: 0.001
Architecture: 1 → 72 → 32 → 1
Max Epochs: 500
Early Stopping Patience: 20

Training started...
Epoch  50: Train Loss = 1.051115, Test Loss = 1.045517
Epoch 100: Train Loss = 1.041326, Test Loss = 1.038193
Epoch 150: Train Loss = 1.037958, Test Loss = 1.035468
Epoch 200: Train Loss = 1.035154, Test Loss = 1.032846
Epoch 250: Train Loss = 1.032425, Test Loss = 1.030171
Epoch 300: Train Loss = 1.029727, Test Loss = 1.027491
Epoch 350: Train Loss = 1.027056, Test Loss = 1.024827
Epoch 400: Train Loss = 1.024410, Test Loss = 1.022185
Epoch 450: Train Loss = 1.021787, Test Loss = 1.019565
Epoch 500: Train Loss = 1.019185, Test Loss = 1.016965

Exp 4: Sigmoid Activation RESULTS:
Final Train Loss: 1.019185
Final Test Loss: 1.016965
R² Score: -0.0194
Accuracy: 0.00%
Epochs Completed: 500
```



Generate + Code + Markdown | Run All | Restart | Clear All Outputs | Jupyter Variables | Outline ... Python 3.12.6

```
Exp 4: Sigmoid Activation RESULTS:
Final Train Loss: 1.019185
Final Test Loss: 1.016965
R² Score: -0.0194
Accuracy: 0.00%
Epochs Completed: 500
```




```
====================================================================================================
```

```
====================================================================================================
COMPREHENSIVE EXPERIMENT RESULTS TABLE
====================================================================================================
```

Experiment	Learning Rate	Batch Size	Number of Epochs	Optimizer	Activation Function	Training Accuracy	Validation Accuracy	Test Accuracy	Training Loss	Validation
Baseline (Part A)	0.0010	Full Batch	500	Gradient Descent	ReLU	57.25%	57.25%	57.25%	0.432816	0.4
Exp 1: Higher LR	0.0100	Full Batch	300	Gradient Descent	ReLU	86.64%	86.64%	86.64%	0.135157	0.1
Exp 2: Lower LR	0.0001	Full Batch	800	Gradient Descent	ReLU	7.06%	7.06%	7.06%	0.930592	0.9
Exp 3: More Epochs	0.0010	Full Batch	1000	Gradient Descent	ReLU	76.68%	76.68%	76.68%	0.236381	0.2
Exp 4: Sigmoid Activation	0.0010	Full Batch	500	Gradient Descent	Sigmoid	0.00%	0.00%	0.00%	1.019185	1.0

Results saved to 'neural_network_hyperparameter_experiments.csv'

```
================================================================================
EXPERIMENT PERFORMANCE SUMMARY
================================================================================
```

Experiment	Test Loss	Test Accuracy	Epochs
Baseline (Part A)	0.426525	57.25%	500
Exp 1: Higher LR	0.133268	86.64%	300
Exp 2: Lower LR	0.927256	7.06%	800
Exp 3: More Epochs	0.232662	76.68%	1000
Exp 4: Sigmoid Activation	1.016965	0.00%	500

```
======================================================================
```

Optimizer	Activation Function	Training Accuracy	Validation Accuracy	Test Accuracy	Training Loss	Validation Loss	Test Loss	Observations
radient Descent	ReLU	57.25%	57.25%	57.25%	0.432816	0.426525	0.426525	Baseline model from Pa...
radient Descent	ReLU	86.64%	86.64%	86.64%	0.135157	0.133268	0.133268	Higher LR: Faster conv...
radient Descent	ReLU	7.06%	7.06%	7.06%	0.930592	0.927256	0.927256	Lower LR: Slower but m...
radient Descent	ReLU	76.68%	76.68%	76.68%	0.236381	0.232662	0.232662	Extended training: Tho...
radient Descent	Sigmoid	0.00%	0.00%	0.00%	1.019185	1.016965	1.016965	Sigmoid Activation: Sl...

Prediction Example

For x = 90.2:

NN Prediction: 729,616.22

Ground Truth: 1,735,575.41

Absolute Error: 1,005,959.19

Relative Error: 57.96%

Conclusion

The lab demonstrated the effectiveness and limitations of basic neural network architectures on polynomial approximation. The most important hyperparameters were learning rate and activation function; adjusting these yielded dramatic differences in learning speed and final fit. While the network approximated the target curve, higher complexity or advanced techniques could further reduce underfitting and error. The task also reinforced the value of systematic experimentation.