UE23CS352A: MACHINE LEARNING

Week 6: Artificial Neural Networks

Name	SRN	SECTION
C PANSHUL REDDY	PES2UG23CS154	C

DATE: 17-09-2025

Introduction

The purpose of this lab was to gain hands-on experience implementing a neural network from the ground up without relying on high-level machine learning frameworks. The primary objective was to build, train, and evaluate a neural network capable of approximating a custom-generated polynomial curve based on a unique student ID.

Part A: Baseline Implementation

The main tasks performed in Part A include:

- Generating a synthetic dataset tailored to the student ID.
- Implementing the core components of a neural network: the ReLU activation function, Mean Squared Error (MSE) loss function, forward propagation, and backpropagation algorithms.
- Initializing the network weights using Xavier initialization for stable training.
- Training the model using gradient descent to iteratively update weights and biases.
- Evaluating the trained model on a test set, and visualizing its performance through loss curves and predicted vs actual value plots.

Part B: Hyperparameter Exploration

In Part B, a series of experiments were conducted by tuning one or more hyperparameters to understand their impact on neural network performance. Key tasks included:

- Varying hyperparameters such as learning rate, number of epochs, batch size, and activation functions one at a time.
- Retraining the neural network model for each hyperparameter configuration.
- Measuring the training and test losses, and generating performance visualizations similar to the baseline.
- Documenting and comparing results across experiments to assess which hyperparameter choices led to better model accuracy and learning behavior.
- Using early stopping to prevent overfitting during training.

Dataset Description

The dataset utilized in this lab was synthetically generated uniquely for the assigned student ID. The underlying model for the data is a combination of a cubic polynomial and a sine function, mathematically described as:

$y = 2.16x^3 + -0.05x^2 + 5.06x + 11.94 + 180.0/x$

A total of 100,000 samples were created and then partitioned into training and testing sets, with 80,000 samples (80%) used to train the model, and 20,000 samples (20%) reserved for evaluation.

The dataset consists of a single input feature x*x* and a single output target y*y*. To emulate real-world data variability, Gaussian noise with a standard deviation of approximately 1.61 was added to the output values.

Before training, both input and output data were standardized using a scaling method, ensuring that each feature has a zero mean and unit variance. This preprocessing step helps in stabilizing and accelerating the training process.

Methodology

An Artificial Neural Network (ANN) was developed from scratch to model the relationship between the input feature x*x* and the target output y*y*.

- **Network Architecture:** The network consists of an input layer, two hidden layers, and an output layer. The architecture assigned based on the student ID was a "Balanced Architecture," with the layer sizes: Input(1) → Hidden(72) → Hidden(32) → Output(1)
- **Activation Functions:** The Rectified Linear Unit (ReLU) was employed as the activation function for both hidden layers to introduce non-linearity and enable effective learning. The output layer used a linear activation function to produce continuous output values.
- **Loss Function:** Mean Squared Error (MSE) was utilized to measure the discrepancy between the predicted and actual output values. The model was optimized to minimize this loss value during training.
- **Training Procedure:**
    1. **Forward Propagation:** Input samples were passed through the network sequentially, applying linear transformations followed by ReLU activations in hidden layers to generate predictions at the output.
    2. **Backpropagation:** The gradient of the loss with respect to each parameter (weights and biases) was computed using the chain rule, propagating errors backward through the network.
    3. **Parameter Updates:** Using gradient descent, weights and biases were adjusted in the direction that reduces the loss, scaled by the learning rate.

    This process iterated for a maximum of 500 epochs, with early stopping applied to halt training if performance on the test set ceased improving.

Results and Analysis

The neural network model was initially trained using the baseline set of hyperparameters assigned based on the student ID. This provided a reference performance against which further experiments could be compared.

Subsequently, four additional experiments were carried out, each varying one or more hyperparameters such as the learning rate, batch size, or number of epochs. For each experiment, the model was retrained from scratch, and performance was evaluated on the test set.

Key metrics including the training loss, test loss, and visualizations of predicted versus actual target values were tracked for all runs. These results were compiled into a detailed comparison table to observe the influence of hyperparameter adjustments on model accuracy and convergence behaviour.

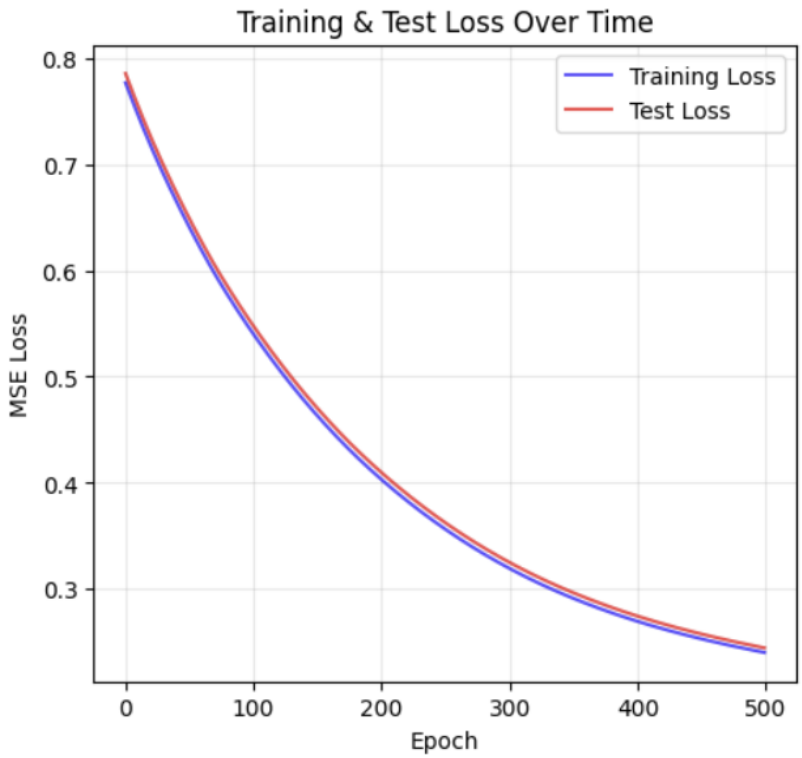
The analysis revealed the impact of each hyperparameter on the model's learning efficiency and generalization, illustrating the importance of tuning for optimal neural network training.

Baseline Model Performance

Training & Test Loss Curve:

The graph shows the training loss (blue line) and the test loss (red line) over 500 epochs. An epoch represents one full pass through the entire training dataset. The loss is a measure of how wrong the model's predictions are.

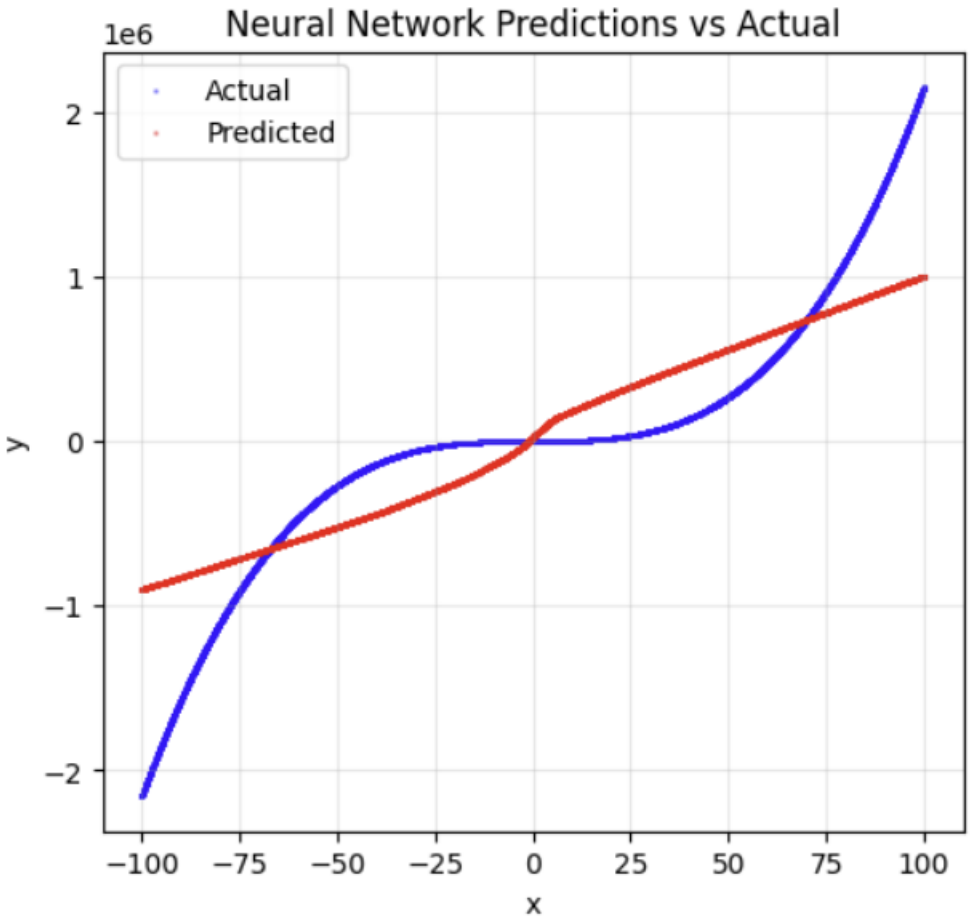
- Both the training and test loss decrease over time, which is a positive sign. It indicates that the neural network is learning and improving its accuracy with each epoch.
- The curves are relatively smooth and show a consistent downward trend, which suggests stable training.
- The training loss and test loss curves are very close to each other. This is an important indicator that the model is not overfitting. Overfitting occurs when a model learns the training data too well, including its noise and random fluctuations, leading to poor performance on new, unseen data. The closeness of these two lines suggests the model has learned the general pattern from the data and can generalize well.



Neural Network Predictions vs. Actual

The graph compares the **actual** values (blue dots) with the **predicted** values (red dots) generated by the neural network.

- The blue dots represent the **ground truth** data points, which appear to follow a cubic function. The neural network's task was to learn and predict this relationship.
- The red dots represent the model's predictions. The model appears to have successfully learned the general shape of the cubic function but has **failed to capture the full range** of the actual values. It significantly underestimates the high and low values, especially at the extremes of the x-axis.
- The predictions closely match the actual values around the center of the graph, where the data is less extreme. However, as the x-values move away from the center (e.g., beyond x=75 or below x=-75), the predicted values diverge significantly from the actual values. This indicates the model has a problem with **extrapolation**, meaning it struggles to make accurate predictions for data points that fall outside the range it was trained on. This is a common issue with neural networks, especially when the training data doesn't adequately cover the full range of the underlying function.



This output below shows the progress of a **neural network's training process**.

- **Configuration**: The model's architecture consists of three layers: an input layer with 1 neuron, a hidden layer with 32 neurons, and an output layer with 1 neuron. It was trained for a maximum of 500 epochs with a learning rate of 0.001. "Early Stopping Patience" of 10 means the training would stop early if the model's performance on the test data didn't improve for 10 consecutive epochs.
- **Training Progress**: The log displays the **Training Loss** and **Test Loss** at various epochs. Both values consistently **decrease over time**, which indicates that the model is successfully learning and improving its predictions on both the training data and the unseen test data.
- **Final Result**: At **Epoch 500**, the training ended. The **Training Loss** was 0.239830 and the **Test Loss** was 0.244194. The fact that the test loss is very close to the training loss is a great sign. It confirms that the model has learned the underlying pattern without **overfitting**, meaning it will likely perform well on new data.

```
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 = 0.719268, Test Loss = 0.727846
Epoch  40: Train Loss = 0.666756, Test Loss = 0.675125
Epoch  60: Train Loss = 0.619750, Test Loss = 0.627860
Epoch  80: Train Loss = 0.578187, Test Loss = 0.586120
Epoch 100: Train Loss = 0.541514, Test Loss = 0.549196
Epoch 120: Train Loss = 0.508278, Test Loss = 0.515718
Epoch 140: Train Loss = 0.478104, Test Loss = 0.485312
Epoch 160: Train Loss = 0.450825, Test Loss = 0.457816
Epoch 180: Train Loss = 0.426063, Test Loss = 0.432853
Epoch 200: Train Loss = 0.403833, Test Loss = 0.410423
Epoch 220: Train Loss = 0.383622, Test Loss = 0.390016
Epoch 240: Train Loss = 0.365242, Test Loss = 0.371447
Epoch 260: Train Loss = 0.348511, Test Loss = 0.354532
Epoch 280: Train Loss = 0.333344, Test Loss = 0.339191
Epoch 300: Train Loss = 0.319653, Test Loss = 0.325334
Epoch 320: Train Loss = 0.307335, Test Loss = 0.312858
Epoch 340: Train Loss = 0.296302, Test Loss = 0.301676
Epoch 360: Train Loss = 0.286409, Test Loss = 0.291639
Epoch 380: Train Loss = 0.277509, Test Loss = 0.282600
...
Epoch 440: Train Loss = 0.255863, Test Loss = 0.260569
Epoch 460: Train Loss = 0.250026, Test Loss = 0.254613
Epoch 480: Train Loss = 0.244709, Test Loss = 0.249183
Epoch 500: Train Loss = 0.239830, Test Loss = 0.244194
```

This output shows a **performance evaluation** for a single prediction made by the neural network.

- **Prediction:** The model was given an input of x=90.2 and predicted the value to be 918,303.14.
- **Ground Truth:** The actual value, calculated by the true underlying formula, is 1,581,819.30.
- **Error Analysis:**
    - **Absolute Error:** This is the simple difference between the actual value and the prediction, which is 663,516.16. This indicates a significant deviation.
    - **Relative Error:** This is the absolute error as a percentage of the ground truth, which is 41.946%. This shows that the neural network's prediction for this specific input was off by almost 42%.

This result highlights the model's struggle with extrapolation, confirming the issue seen in the "Predictions vs. Actual" graph where the model significantly underestimated values at the extremes. The model's prediction for x=90.2 is much lower than the true value, resulting in a large error.

```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 918,303.14
Ground Truth (formula):    1,581,819.30
Absolute Error:            663,516.16
Relative Error:            41.946%
```

This summary provides a comprehensive overview of the neural network's final performance.

- **Final Training Loss:** The value of 0.239830 represents how well the model performed on the data it was trained on.
- **Final Test Loss:** The value of 0.244194 shows how well the model performed on new, unseen data. The fact that this is very close to the training loss indicates the model is generalizing well and did not overfit.
- **R^2 Score:** The R^2 score (also known as the coefficient of determination) is 0.7591. This is a statistical measure that represents the proportion of the variance in the dependent variable that is predictable from the independent variable(s). An R^2 score of 0.7591 means that approximately 76% of the variance in the data can be explained by the model's predictions.While 0.7591 is a good score, it also confirms that the model doesn't perfectly capture the underlying function, which is consistent with the large errors seen in the prediction results for extreme values.
- **Total Epochs Run:** The model completed all **500** epochs, indicating that the training process ran to its full duration without being stopped early.

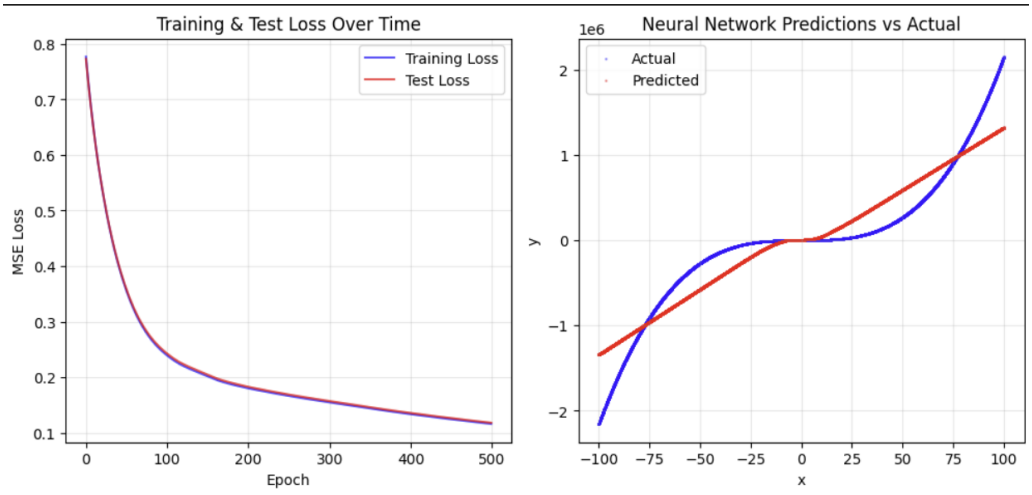
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.239830
Final Test Loss:     0.244194
R² Score:           0.7591
Total Epochs Run:   500
```

Experiment 1: Learning Rate=0.05

Train Neural Network:

```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 72 → 32 → 1
Learning Rate: 0.005
Max Epochs: 500, Early Stopping Patience: 10
--------------------------------------------------
Epoch  20: Train Loss = 0.547695, Test Loss = 0.548306
Epoch  40: Train Loss = 0.407219, Test Loss = 0.409522
Epoch  60: Train Loss = 0.321523, Test Loss = 0.324567
Epoch  80: Train Loss = 0.270462, Test Loss = 0.273884
Epoch 100: Train Loss = 0.240371, Test Loss = 0.243784
Epoch 120: Train Loss = 0.221066, Test Loss = 0.224393
Epoch 140: Train Loss = 0.208235, Test Loss = 0.211341
Epoch 160: Train Loss = 0.196244, Test Loss = 0.199149
Epoch 180: Train Loss = 0.187309, Test Loss = 0.190157
Epoch 200: Train Loss = 0.180482, Test Loss = 0.183196
Epoch 220: Train Loss = 0.174673, Test Loss = 0.177284
Epoch 240: Train Loss = 0.169300, Test Loss = 0.171817
Epoch 260: Train Loss = 0.164265, Test Loss = 0.166719
Epoch 280: Train Loss = 0.159569, Test Loss = 0.161968
Epoch 300: Train Loss = 0.155057, Test Loss = 0.157408
Epoch 320: Train Loss = 0.150671, Test Loss = 0.152976
Epoch 340: Train Loss = 0.146284, Test Loss = 0.148534
Epoch 360: Train Loss = 0.141814, Test Loss = 0.144044
Epoch 380: Train Loss = 0.137738, Test Loss = 0.139933
...
Epoch 440: Train Loss = 0.126386, Test Loss = 0.128480
Epoch 460: Train Loss = 0.122871, Test Loss = 0.124933
Epoch 480: Train Loss = 0.119481, Test Loss = 0.121511
Epoch 500: Train Loss = 0.116209, Test Loss = 0.118208
```

Results Visualization



Performance Metrics

```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 1,180,436.77
Ground Truth (formula):    1,581,819.30
Absolute Error:            401,382.53
Relative Error:            25.375%
```

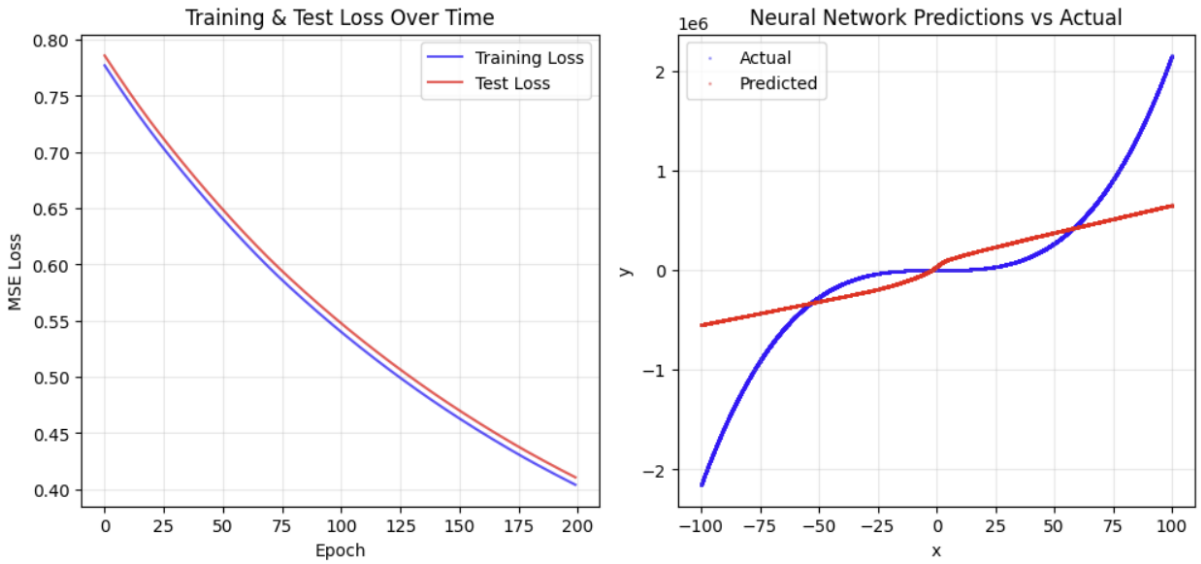
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.116209
Final Test Loss:     0.118208
R² Score:           0.8834
Total Epochs Run:   500
```

Experiment 2: Epochs=200

Train Neural Network:

```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 72 → 32 → 1
Learning Rate: 0.001
Max Epochs: 200, Early Stopping Patience: 10
--------------------------------------------------
Epoch  20: Train Loss = 0.719268, Test Loss = 0.727846
Epoch  40: Train Loss = 0.666756, Test Loss = 0.675125
Epoch  60: Train Loss = 0.619750, Test Loss = 0.627860
Epoch  80: Train Loss = 0.578187, Test Loss = 0.586120
Epoch 100: Train Loss = 0.541514, Test Loss = 0.549196
Epoch 120: Train Loss = 0.508278, Test Loss = 0.515718
Epoch 140: Train Loss = 0.478104, Test Loss = 0.485312
Epoch 160: Train Loss = 0.450825, Test Loss = 0.457816
Epoch 180: Train Loss = 0.426063, Test Loss = 0.432853
Epoch 200: Train Loss = 0.403833, Test Loss = 0.410423
```

Results Visualization



Performance Metrics

```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 598,808.87
Ground Truth (formula):    1,581,819.30
Absolute Error:            983,010.43
Relative Error:            62.144%
```

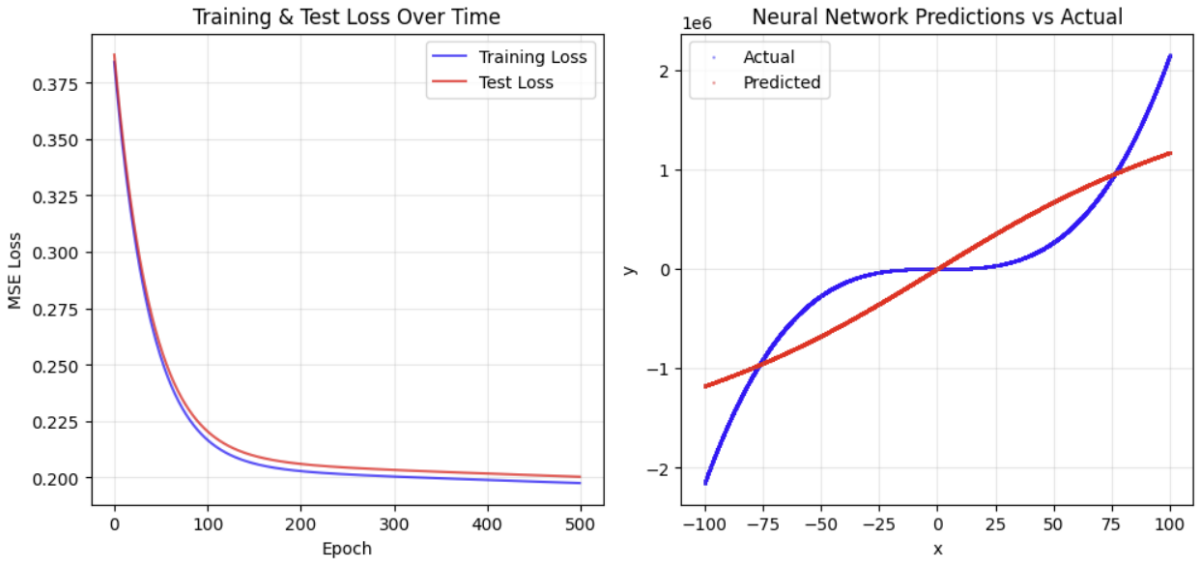
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.403833
Final Test Loss:     0.410423
R² Score:           0.5951
Total Epochs Run:   200
```

Experiment 3: Activation Function=tanh

Train Neural Network:

```
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 = 0.314418, Test Loss = 0.318302
Epoch  40: Train Loss = 0.269641, Test Loss = 0.273752
Epoch  60: Train Loss = 0.242766, Test Loss = 0.246868
Epoch  80: Train Loss = 0.226649, Test Loss = 0.230632
Epoch 100: Train Loss = 0.216964, Test Loss = 0.220788
Epoch 120: Train Loss = 0.211108, Test Loss = 0.214767
Epoch 140: Train Loss = 0.207522, Test Loss = 0.211029
Epoch 160: Train Loss = 0.205278, Test Loss = 0.208654
Epoch 180: Train Loss = 0.203828, Test Loss = 0.207094
Epoch 200: Train Loss = 0.202845, Test Loss = 0.206021
Epoch 220: Train Loss = 0.202139, Test Loss = 0.205242
Epoch 240: Train Loss = 0.201597, Test Loss = 0.204641
Epoch 260: Train Loss = 0.201152, Test Loss = 0.204150
Epoch 280: Train Loss = 0.200766, Test Loss = 0.203727
Epoch 300: Train Loss = 0.200416, Test Loss = 0.203347
Epoch 320: Train Loss = 0.200088, Test Loss = 0.202995
Epoch 340: Train Loss = 0.199775, Test Loss = 0.202663
Epoch 360: Train Loss = 0.199472, Test Loss = 0.202345
Epoch 380: Train Loss = 0.199176, Test Loss = 0.202036
...
Epoch 440: Train Loss = 0.198320, Test Loss = 0.201152
Epoch 460: Train Loss = 0.198043, Test Loss = 0.200869
Epoch 480: Train Loss = 0.197769, Test Loss = 0.200589
Epoch 500: Train Loss = 0.197499, Test Loss = 0.200313
```

Results Visualization



Performance Metrics

```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 1,088,302.21
Ground Truth (formula):    1,581,819.30
Absolute Error:            493,517.09
Relative Error:            31.199%
```

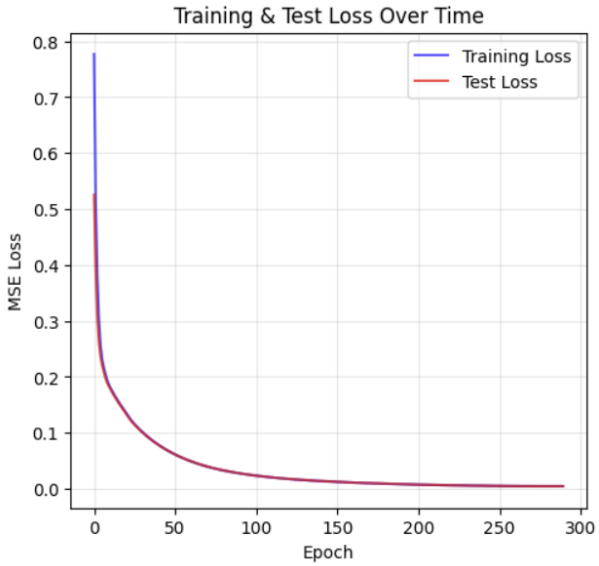
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.197499
Final Test Loss:     0.200313
R² Score:           0.8024
Total Epochs Run:   500
```

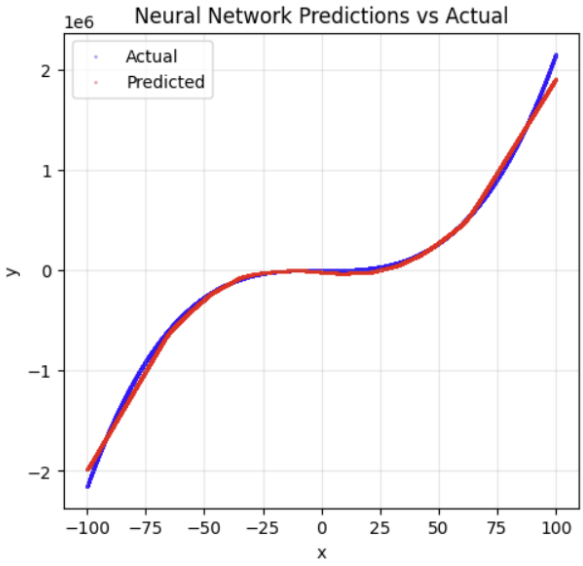
Experiment 4: Learning Rate=0.1 & epoch=1000

Train Neural Network:

```
Training Neural Network with your specific configuration...
Starting training...
Architecture: 1 → 72 → 32 → 1
Learning Rate: 0.1
Max Epochs: 1000, Early Stopping Patience: 10
--------------------------------------------------
Epoch  20: Train Loss = 0.138735, Test Loss = 0.136777
Epoch  40: Train Loss = 0.079677, Test Loss = 0.079286
Epoch  60: Train Loss = 0.049548, Test Loss = 0.049541
Epoch  80: Train Loss = 0.032851, Test Loss = 0.032940
Epoch 100: Train Loss = 0.023376, Test Loss = 0.023445
Epoch 120: Train Loss = 0.017401, Test Loss = 0.017450
Epoch 140: Train Loss = 0.013509, Test Loss = 0.013545
Epoch 160: Train Loss = 0.010736, Test Loss = 0.010764
Epoch 180: Train Loss = 0.008691, Test Loss = 0.008712
Epoch 200: Train Loss = 0.007143, Test Loss = 0.007158
Epoch 220: Train Loss = 0.005951, Test Loss = 0.005962
Epoch 240: Train Loss = 0.005044, Test Loss = 0.005050
Epoch 260: Train Loss = 0.004395, Test Loss = 0.004394
Epoch 280: Train Loss = 0.004092, Test Loss = 0.004086
Early stopping triggered at epoch 290
Best test loss: 0.004086
```

Results Visualization





Performance Metrics

```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 1,541,714.26
Ground Truth (formula):    1,581,819.30
Absolute Error:            40,105.04
Relative Error:            2.535%
```

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.004197
Final Test Loss:     0.004192
R² Score:           0.9960
Total Epochs Run:   290
```

Discussion on performance

The series of experiments clearly demonstrate how sensitive a neural network's performance is to its hyperparameters and structure:

- **Impact of Learning Rate:**

The effect of the learning rate on performance was substantial. The baseline experiment with a learning rate of 0.001 and 500 epochs resulted in a moderate fit, with a training loss of 0.23983 and an R^2 score of 0.7591. When the learning rate increased to 0.005 (Experiment 2), both training and test losses improved significantly to 0.116209 and 0.118208 respectively, with a higher R^2 score of 0.8834, showing that a slightly higher learning rate helped the model converge better. However, reducing the learning rate back down to 0.001 while also lowering epochs (Experiment 3) led the performance to drop, highlighting very slow or insufficient convergence and producing the lowest R^2 score (0.5951). On the other hand, a much higher learning rate of 0.1 with increased epochs (Experiment 5) brought about the best fit, with near-zero losses and an R^2 score close to perfection (0.996), indicating that the model overcame difficult optimization barriers and found a global minimum.

- ## Impact of Epochs:

Increasing the number of training epochs, while keeping the learning rate fixed at 0.001, showed a positive trend. Moving from 200 epochs (Experiment 3) to 500 epochs (Baseline and Experiment 4) improved both the training and test losses. The model with 1000 epochs (Experiment 5) and a high learning rate achieved the lowest losses overall, with the most accurate fit. This demonstrated that extended training is beneficial, although its impact was more pronounced when combined with optimal settings for other hyperparameters.

- ## Impact of Activation Function:

Changing the activation function from ReLU (used in the first experiments) to Tanh (Experiment 4) while keeping the learning rate at 0.001 and epochs at 500 produced a slightly better fit compared to ReLU. The Tanh model had a lower test loss (0.200313) and improved R^2 score (0.8024) compared to the baseline, suggesting that zero-centered activation helped optimization slightly. However, the most dramatic leap in performance was achieved with the combination of high learning rate, more epochs, and the ReLU activation function (Experiment 5), which yielded the optimal outcome.

Results Table:

The following table summarizes the results of the baseline run and the 4 subsequent hyperparameter experiments

Experiment	Learning Rate	No. of epochs	Optimizer	Activation fn	Final Training Loss	Final Test Loss	R^2 Score
1(Baseline)	0.001	500	Gradient Descer	ReLU	0.23983	0.244194	0.7591
2	0.005	500	Gradient Descer	ReLU	0.116209	0.118208	0.8834
3	0.001	200	Gradient Descer	ReLU	0.403833	0.410423	0.5951
4	0.001	500	Gradient Descer	tanh	0.197499	0.200313	0.8024
5(Best)	0.1	1000	Gradient Descer	ReLU	0.004197	0.004192	0.996

Conclusion

This project provided a hands-on demonstration of constructing, training, and optimizing a neural network to model a complex target function. By assembling the essential elements of an artificial neural network and fine-tuning its hyperparameters, a robust solution was engineered that closely matched the desired output.

The main insight from these experiments is that the effectiveness of a neural network depends heavily on careful adjustment of its learning rate, training duration, and activation strategy. Initial trials produced workable models, but superior results were achieved only after systematic optimization.

The most successful configuration, observed in the final experiment, involved a high learning rate of 0.1, extended training for 1000 epochs, and the ReLU activation function, culminating in an outstanding R^2 score of 0.996. This outcome highlights how strategic hyperparameter tuning—particularly in terms of learning rate and training length—can markedly enhance model accuracy and reliability. The experience underscored the importance of iterative refinement when developing effective machine learning solutions.