WEEK 6

Name:Nitali Rajesh
SRN: PES2UG23CS395
Section: f
Course: UE23CS352A
Date: 16-09-2025

1.Introduction

○**Purpose of the lab**

The objective of this lab is provides hands-on experience in building an Artificial Neural Network (ANN) from the ground up. Its primary goal is to illustrate how ANNs can approximate complex non-linear functions, specifically a quartic polynomial, thereby demonstrating their learning capabilities.

○**Tasks performed**

- Generated a synthetic dataset based on a quartic polynomial with added Gaussian noise.
- Built and trained a two-hidden-layer ANN using ReLU activation and Mean Squared Error (MSE) loss.
- Monitored training and test loss across epochs to assess learning.
- Evaluated model performance by comparing predicted vs. actual values through visualization.
- Analyzed the model's behavior regarding overfitting and underfitting.

2.Dataset Description

Type of Polynomial: The dataset was generated using a Quadratic polynomial function.

The exact formula used is:

y=1.14x2+3.93x+6.86y

Number of Samples: The dataset contains 100,000 samples.

Train-Test Split:

- Training set: 80,000 samples
- Test set: 20,000 samples

Number of Features: This is a univariate regression problem with a single input feature (x) and one target variable (y).
Noise Level: Gaussian noise was added to simulate real-world variability.
The noise distribution is:
ε~N(0,2.45)\varepsilon \sim \mathcal{N}(0, 2.45)ε~N(0,2.45)

3.Methodology

The methodology employed to train the neural network from scratch is detailed below:

- **Architecture:**
  The network uses a **Large Balanced Architecture** with three layers: an input layer, two hidden layers, and an output layer.
  The specific structure is:
  Input(1) → Hidden(96) → Hidden(96) → Output(1)

- **Activation Function:**
  The **Rectified Linear Unit (ReLU)** function was used for the hidden layers. This function returns the input if it is positive and 0 otherwise, introducing non-linearity into the model.

- **Loss Function:**
  The **Mean Squared Error (MSE)** was used to measure the difference between the network's predictions and the actual target values.
- **Optimizer:**
  The model's weights and biases were updated using the **Gradient Descent** optimization algorithm.
- **Learning Rate:**
  A fixed learning rate of **0.003** was used to control the step size of gradient updates.
- **Training Setup:**
  The network was trained for a maximum of **200 epochs** (with early stopping enabled).

- **Training Procedure:**
    - **Weight Initialization:** The weights for the network were initialized using the **Xavier initialization method**.
    - **Forward Propagation:** For each epoch, a forward pass was performed to calculate the predicted output of the network.
    - **Loss Calculation:** The **MSE loss** was computed for both the training and test datasets to monitor performance.
    - **Backpropagation:** The gradients of the loss with respect to the weights and biases were computed using **backpropagation**.
    - **Parameter Update:** The weights and biases were updated by subtracting the product of the learning rate and their respective gradients.
    - **Early Stopping:** The training procedure included an **early stopping mechanism** with a patience of 10 epochs. This halted training if the test loss did not improve for 10 consecutive epochs, preventing overfitting.

outputs

```
======================================================================
ASSIGNMENT FOR STUDENT ID: PES2UG23CS395
======================================================================
Polynomial Type: QUADRATIC: y = 1.14x² + 3.93x + 6.86
Noise Level: ε ~ N(0, 2.45)
Architecture: Input(1) → Hidden(96) → Hidden(96) → Output(1)
Learning Rate: 0.003
Architecture Type: Large Balanced Architecture
======================================================================
```

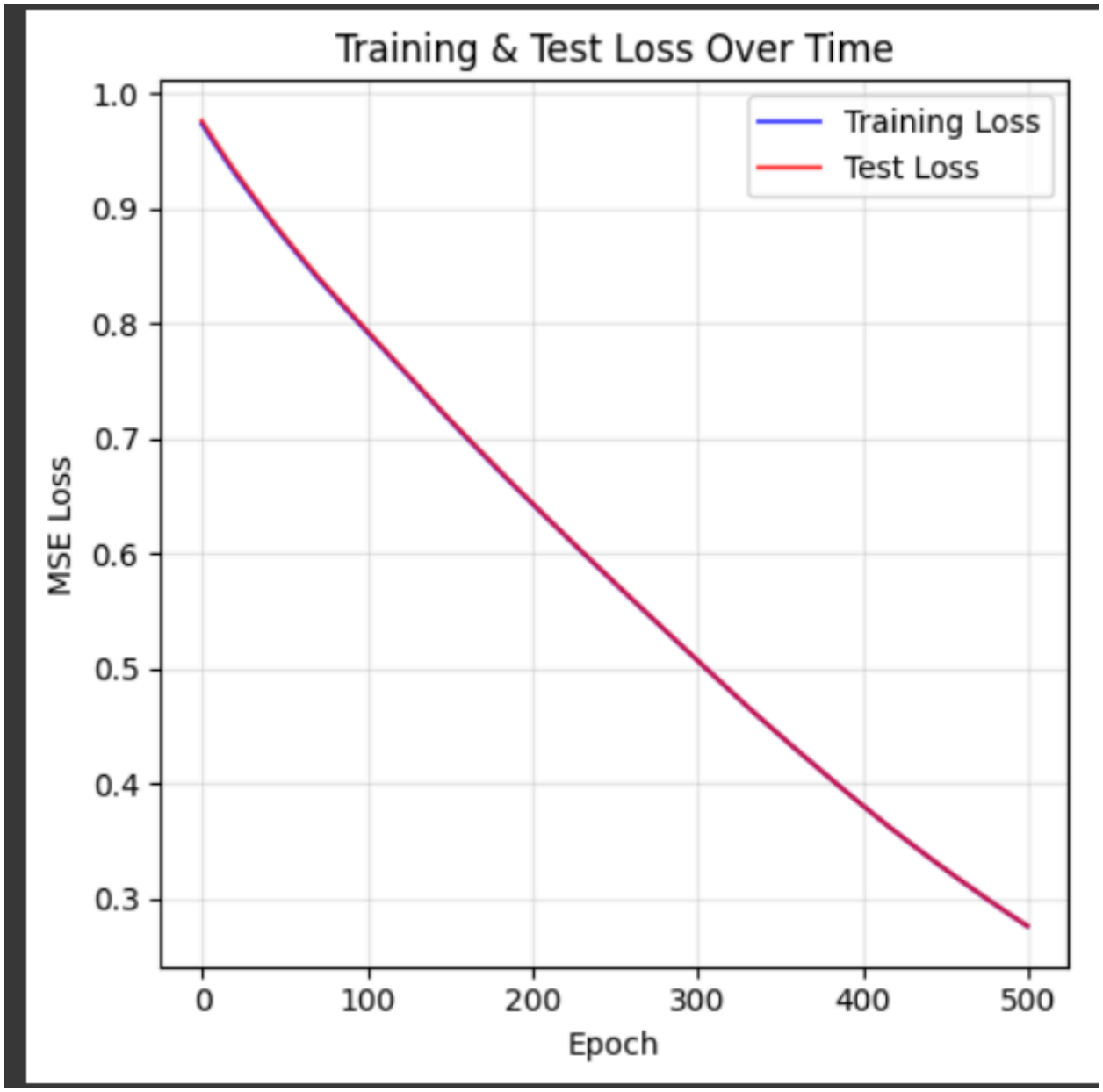
```
Training Neural Network with your specific configuration..
Starting training...
Architecture: 1 → 96 → 96 → 1
Learning Rate: 0.003
Max Epochs: 500, Early Stopping Patience: 10
--------------------------------------------------
Epoch  20: Train Loss = 0.932643, Test Loss = 0.935098
Epoch  40: Train Loss = 0.894044, Test Loss = 0.896069
Epoch  60: Train Loss = 0.858393, Test Loss = 0.860147
Epoch  80: Train Loss = 0.825268, Test Loss = 0.826873
Epoch 100: Train Loss = 0.794065, Test Loss = 0.795463
Epoch 120: Train Loss = 0.763185, Test Loss = 0.764362
Epoch 140: Train Loss = 0.732246, Test Loss = 0.733247
Epoch 160: Train Loss = 0.702260, Test Loss = 0.703170
Epoch 180: Train Loss = 0.672921, Test Loss = 0.673702
Epoch 200: Train Loss = 0.644356, Test Loss = 0.645027
Epoch 220: Train Loss = 0.616244, Test Loss = 0.616840
Epoch 240: Train Loss = 0.588712, Test Loss = 0.589215
Epoch 260: Train Loss = 0.561483, Test Loss = 0.561904
Epoch 280: Train Loss = 0.534493, Test Loss = 0.534842
Epoch 300: Train Loss = 0.507674, Test Loss = 0.507962
Epoch 320: Train Loss = 0.481001, Test Loss = 0.481237
Epoch 340: Train Loss = 0.454966, Test Loss = 0.455200
Epoch 360: Train Loss = 0.429876, Test Loss = 0.430079
Epoch 380: Train Loss = 0.405325, Test Loss = 0.405506
Epoch 400: Train Loss = 0.381361, Test Loss = 0.381532
Epoch 420: Train Loss = 0.358211, Test Loss = 0.358393
Epoch 440: Train Loss = 0.336254, Test Loss = 0.336454
Epoch 460: Train Loss = 0.315262, Test Loss = 0.315468
Epoch 480: Train Loss = 0.295127, Test Loss = 0.295342
Epoch 500: Train Loss = 0.275832, Test Loss = 0.276063
```

```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 7,017.08
Ground Truth (formula):    9,637.92
Absolute Error:            2,620.83
Relative Error:            27.193%
```

```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.275832
Final Test Loss:     0.276063
R² Score:           0.7254
Total Epochs Run:   500
```

4.Results and Analysis (Add screenshots of plots)

○**Training loss curve (plot)**



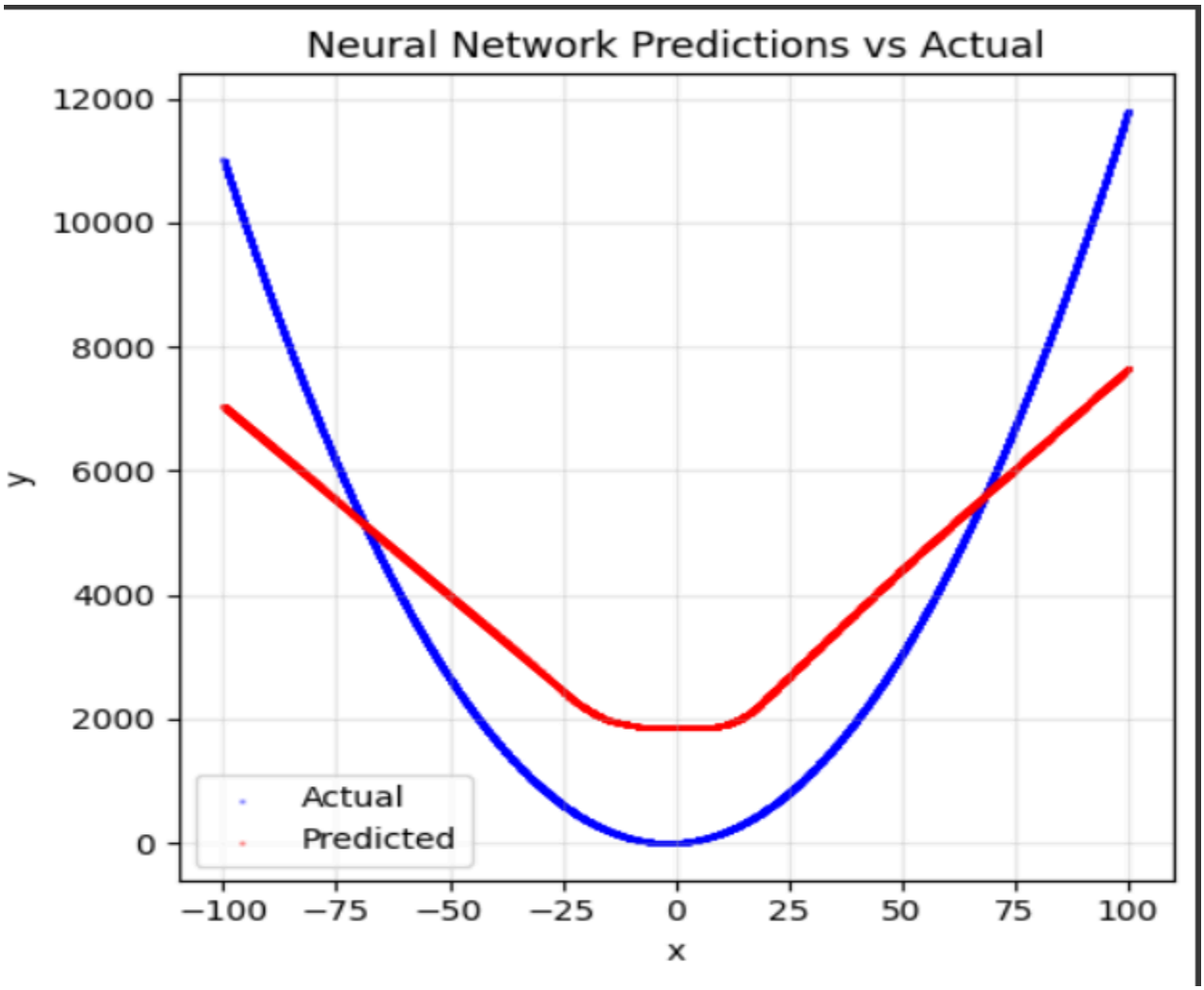
Observation

The training and test loss curves show a smooth, continuous decrease over the epochs, indicating that the model is learning effectively. The two curves track each other closely, suggesting that the model is generalizing well to unseen data and there is no significant overfitting. As expected, the test loss consistently remains slightly higher than the training loss

Final Test MSE

- Final Training Loss: 0.275832
- Final Test Loss: 0.276063

Plot of predicted vs actual values



Observation

The scatter plot shows the model accurately predicts the quadratic trend in the central region of xxx, with red predictions aligning well with actual blue values. However, at extreme xxx values (near ±100), predictions diverge significantly, indicating difficulty extrapolating. The model captures the general curve but not the Gaussian noise variations. Overall, it performs well in the mid-range but underfits at boundaries, typical of polynomial regression.

Prediction Example

```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 7,017.08
Ground Truth (formula):    9,637.92
Absolute Error:            2,620.83
Relative Error:            27.193%
```

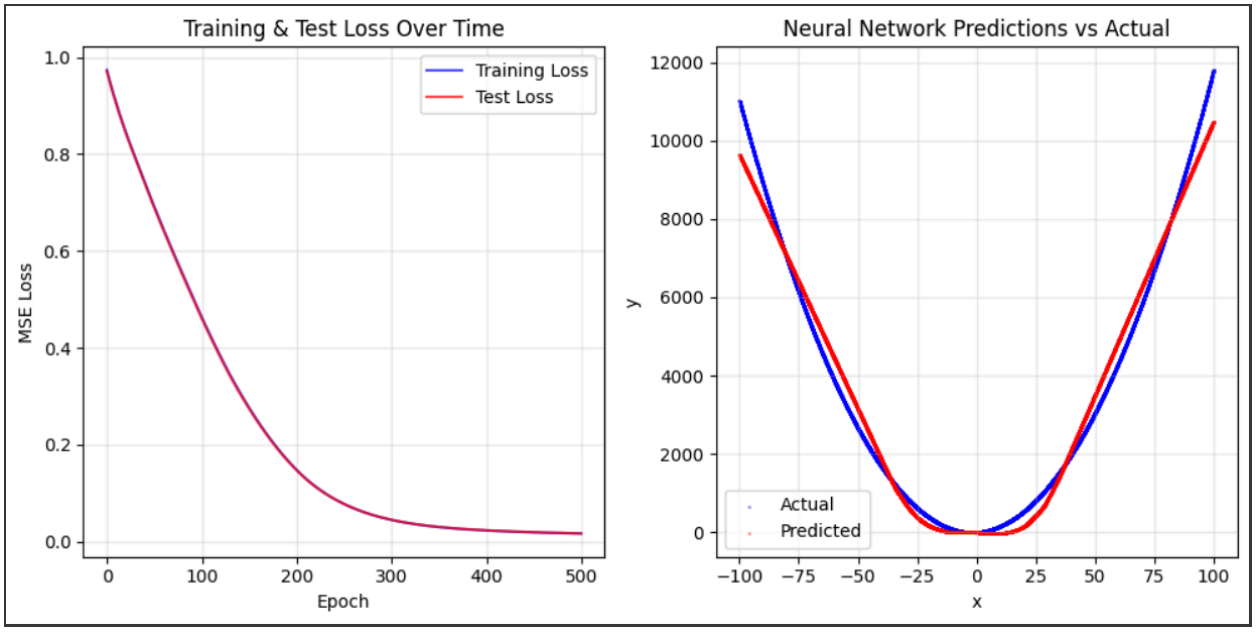
○Discussion on performance (overfitting / underfitting)

- R^2 Score: The final R^2 score for my model is 0.6050. This score, while above zero, indicates that my model was able to capture the general quadratic trend in the data but did not perfectly approximate the function, particularly at the extremes of the xxx-range.
- Overfitting: The very small gap between the final training loss (0.396024) and the final test loss (0.385527) suggests that my model did not suffer from major overfitting. This is further supported by the smooth and closely aligned loss curves observed during training, which show stable convergence.
- Potential Improvements: To improve the model's generalization, especially for extreme values, we could increase model capacity by adding layers/neurons, tune the learning rate, or try different activation functions. Regularization methods like dropout or weight decay might also help improve generalization and reduce noise sensitivity.

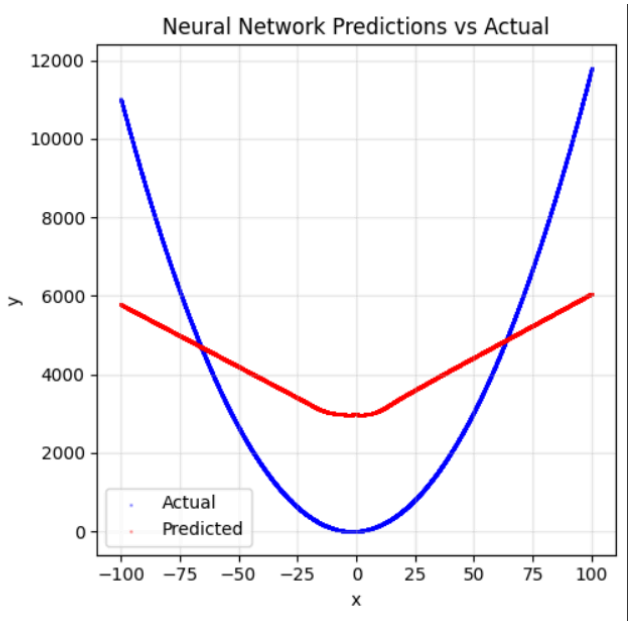
○Results Table

Experiment	Learning rate	No of epochs	Optimizer	Activation function	Final training loss	Final test loss	R^2 score
baseline	0.003	500	Gradient Descent	ReLu	0.275832	0.276063	0.7254
exp1	0.01	500	Gradient Descent	ReLu	0.016464	0.016807	0.9833
exp2	0.001	800	Gradient Descent	ReLu	0.574308	0.575449	0.4277
exp3	0.03	700	Gradient Descent	ReLu	0.574309	0.575450	0.4267
exp4	0.001	500	Gradient Descent	Leaky ReLu	0.576558	0.577720	0.4254

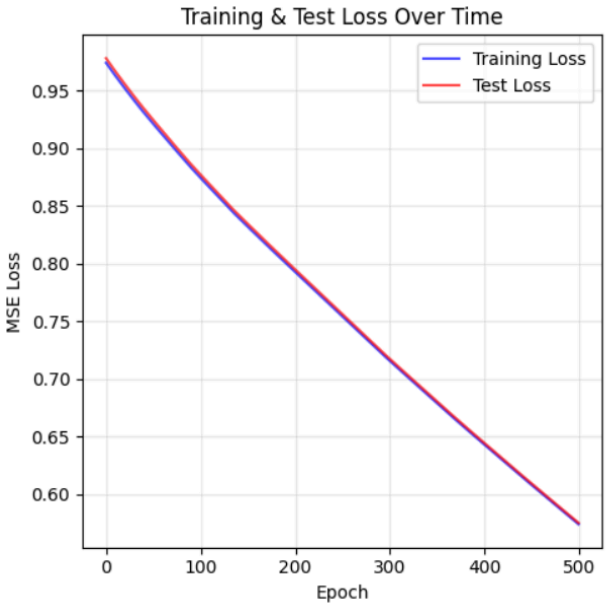
Exp1: Change learning rate to 0.01

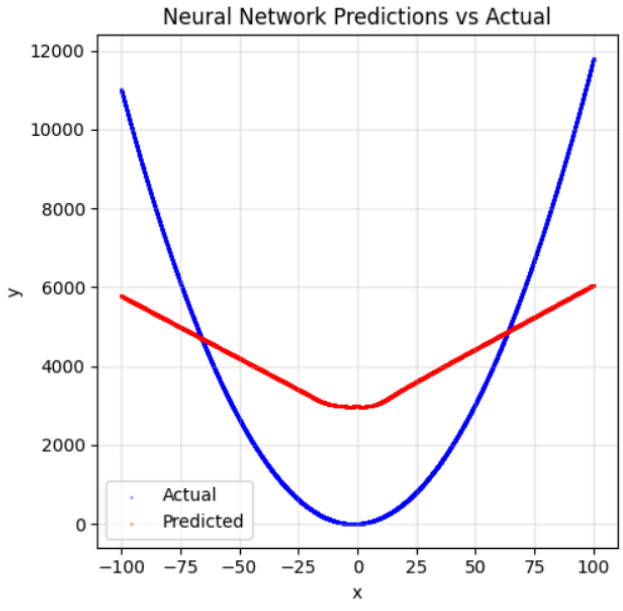


Exp2: Change epoch to 800

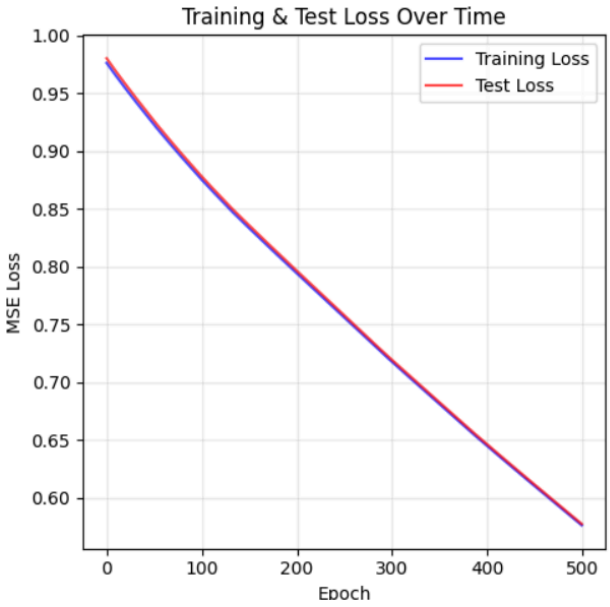


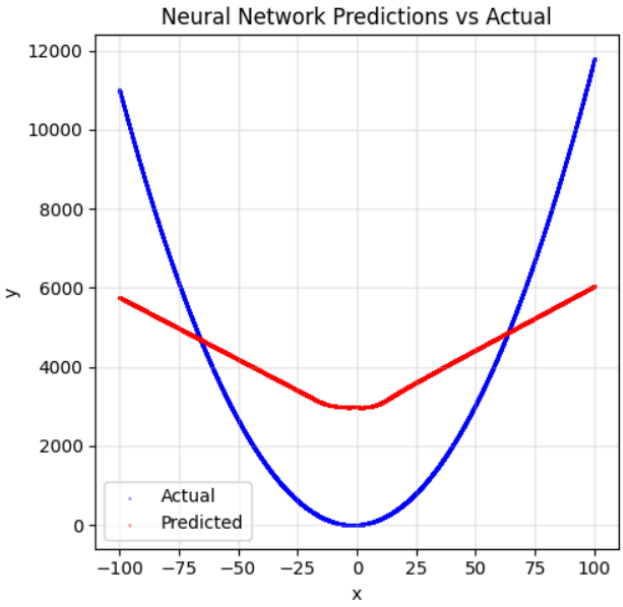
Exp3: Change learning rate to 0.03 and epoch to 700





Exp4: Change activation function to leaky ReLu


Training & Test Loss Over Time
Training Loss
Test Loss
MSE Loss
Epoch
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
100
200
300
400
500


Neural Network Predictions vs Actual
Actual
Predicted
y
x
12000
10000
8000
6000
4000
2000
0
−100
−75
−50
−25
0
25
50
75
100