Artificial Neural Networks

Name: Dinakar Emmanuel SRN: PES2UG32CS178

Course: Machine Learning Date: 19/09/2025

Introduction

The purpose of this lab is to implement a neural network from scratch, without using any high-level frameworks. The neural network is trained to approximate the value of a given polynomial curve.
The tasks are to

1. Implement the components of a neural network: activation and loss functions, forward and backpropagation logic.
2. Train and evaluate this model. This will act as our baseline model.
3. Change and experiment with the hyperparameters and similarly, train and evaluate the model for each experiment.
4. Compare the performance of the model at each step with visualisations to see the impact of the hyperparameters on the model's performance.

Dataset Description:

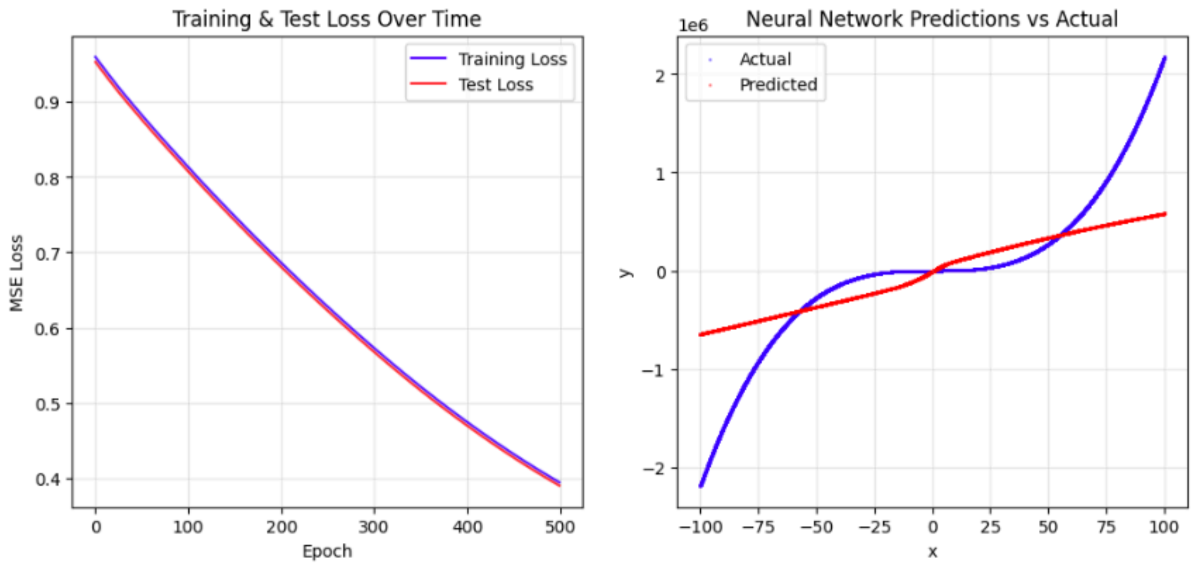
The dataset contains values of the polynomial + sine curve
$y\ =\ 2.18x^3 - 0.74x^2 + 5.24x + 10.93 + 14.6 \times sin(0.046x)$
A synthetic dataset with 100,000 samples was generated containing 1 feature column with x values and a target column with the corresponding y values. Random noise following a normal distribution with a mean of 0 and a standard deviation of 2.06 was added to the true y values when generating the dataset.

Methodology

Different components of a neural network: activation function, loss function, weight initialisation, forward propagation and backpropagation, were implemented. For the baseline model - relu activation, MSE loss calculation, Xavier Weight Initialisation. This model was trained for 500 epochs with a patience parameter of 10.
The hyperparameters are then changed for 4 more iterations and the model performance is evaluated and visualised for each experiment.

Results and Analysis

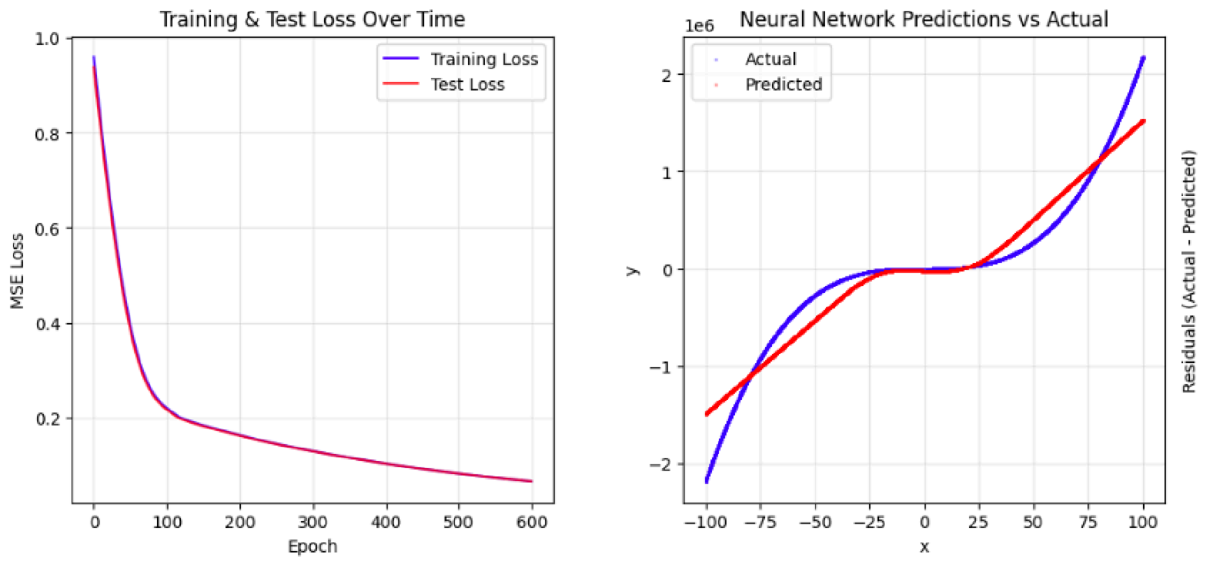
1. Baseline model:



R^2 score: 0.6082
Final Test MSE: 0.390396

- The model does not capture the patterns of the dataset very well since it explains only 60.82% of the variance of the dataset.
- When the model was tested with a test value (90.2), the prediction was off by 66.318%, which is a significant error.
- The model is not capturing the relationship in the data accurately and it underfits.

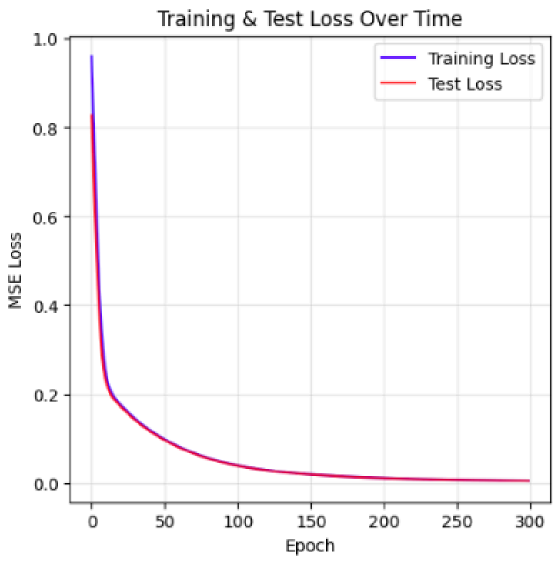
2. Test model 1:

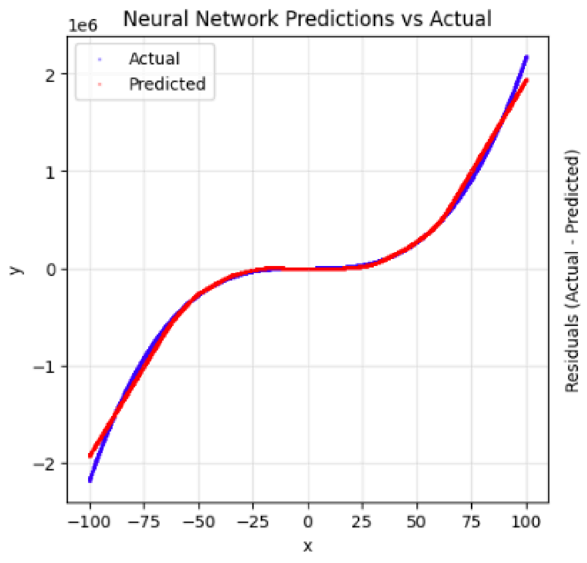


R^2 score: 0.9342
Final Test MSE: 0.065576

- The model explains 93.42% of the variance in the dataset. It captures the patterns quite well.
- The model had a relative error of 16.864% when tested with the test value (90.2).
- Though the model has a high R^2 score, the plots indicate that the model is not fully capturing the nuances of the dataset and slightly underfits.

3. Test model 2:

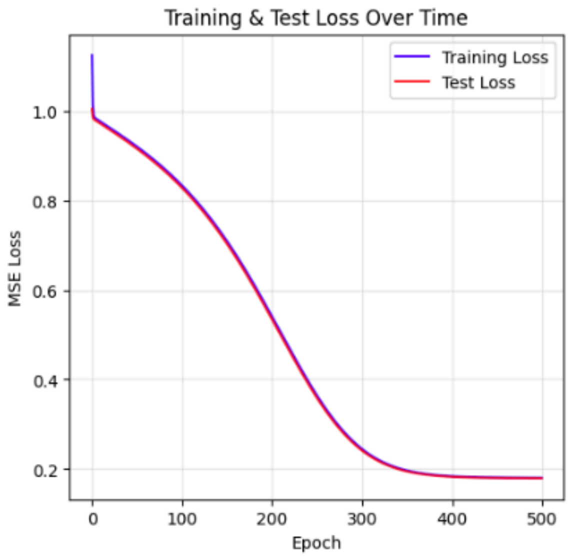


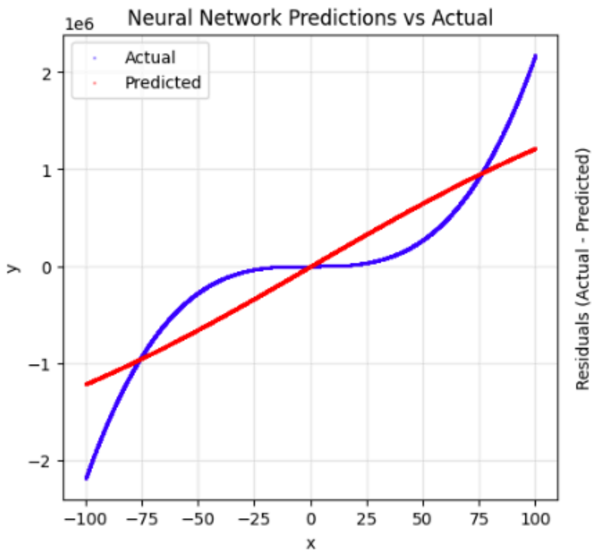


R^2 score: 0.9954
Final Test MSE: 0.004559

- The model captures a significantly large portion of the variance in the dataset.
- The model had a small relative error of 1.504% when tested with the test value (90.2).
- The low training and testing losses, the high R^2 score and the almost correct prediction indicate that this model has a high level of accuracy. It has a good balance and is neither significantly overfitting or underfitting.

4. Test model 3:



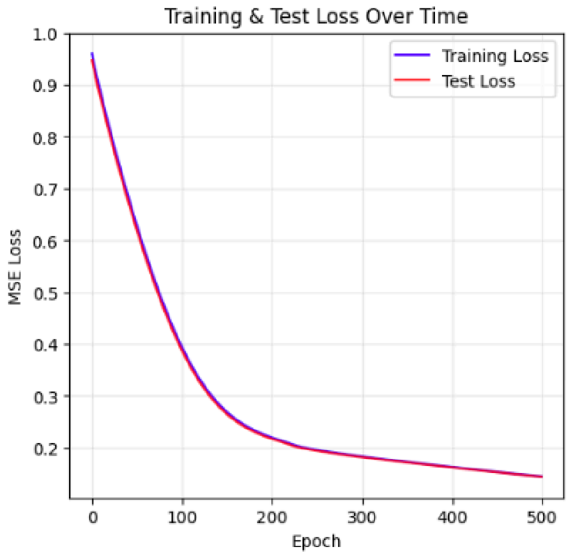


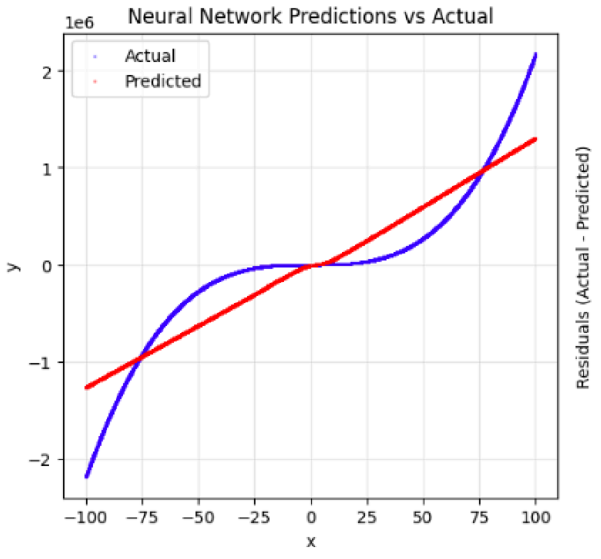
R^2 score: 0.8201

Final Test MSE: 0.179273

- The model captures 82.01% of the variance in the dataset.
- The model's prediction of the test value (90.2) was off by 30.193%.
- Based on the plots (low training and test losses) and the results, this model does not have a very high accuracy but captures the general pattern of the data points and does not significantly underfit or overfit.

5. Test model 4:





R^2 score: 0.8552

Final Test MSE: 0.144300

- The model explains a good amount of the variance.

- The model's prediction of the test value (90.2) was off by 26.746%.
- The model exhibits low training and testing losses. The predictions follow the general trend of the actual polynomial curve. The accuracy may not be very high but the model does not significantly underfit or overfit.

Experiment	Learning rate	No. of epochs	Optimiser (if used)	Activation function	Final training loss	Final test loss	R^2 score
1	0.001	500	Gradient descent	ReLU	0.394796	0.390396	0.6082
2	0.010	600	Gradient descent	ReLU	0.065654	0.065576	0.9342
3	0.085	300	Gradient descent	ReLU	0.004641	0.004559	0.9954
4	0.025	500	Gradient descent	Sigmoid	0.180013	0.179273	0.8201
5	0.005	500	Gradient descent	ReLU	0.144838	0.144300	0.8552

Conclusion:

Different hyperparameters have different impacts on the model.

1. Learning rate
    a. Too small values (0.001) cause the model to underfit due to slow convergence
    b. Large values (0.085) cause the model to perform great.
    c. Learning rate of 0.010 cause the model to have good performance

   The optimal learning rate to predict polynomial values seem to be in the range 0.01 to 0.085

2. Activation function
   ReLU function outperforms the sigmoid mostly because sigmoid restricts the inputs to the range between 0 and 1, due to which, for large positive or negative values, gradients become very small and weight updates slow down. The ReLU function has a range of 0 to ∞, which makes it more flexible for approximating polynomial functions with large magnitudes.
3. Epochs
   a. High learning rate with lower epochs give better performance then low learning rate with high epochs.
   b. Epochs alone don’t improve performance, learning rate is the stronger factor.