MIT Academy of Engineering,Alandi,Pune

School of Computer Engineering

Course - Deep Learning

Student Information:

Name: **Rohan Magdum**
Roll Number: 53
Batch: B-3
Date of Submission: 26 January 2025

Practice Lab Neural Network 1

Neural Network Implementation from Scratch

Objective:

The primary goal of this assignment is to implement a basic feedforward neural network entirely from scratch using Python. No pre-built deep learning libraries like TensorFlow or PyTorch were used, ensuring a hands-on understanding of how the key components—such as forward propagation, backpropagation, and gradient descent—work together to train a neural network.

Problem Definition:

Dataset

This task utilizes a simple dataset based on the binary AND operation, where each pair of inputs (X) corresponds to the result of the AND operation (y). The dataset is defined as follows:

Inputs (X):

[[0, 0], [0, 1], [1, 0], [1, 1]]

Outputs (y):

[[0], [0], [0], [1]]

The goal is to train the neural network to accurately predict the output for any pair of binary inputs. This is a straightforward binary classification problem.

Neural Network Architecture

1. **Input Layer:** 2 neurons (to represent the two binary inputs).
2. **Hidden Layer:** 3 neurons, using a sigmoid activation function to introduce non-linearity.
3. **Output Layer:** 1 neuron, also with a sigmoid activation function to generate probabilities for binary classification.

Methodology

Forward Propagation

The input data flows through the network as follows:

1. Weighted sums are calculated at the hidden layer using the input features.
2. The sigmoid activation function processes these sums, producing the hidden layer's outputs.
3. The hidden layer’s outputs are passed to the output layer, where another weighted sum is computed.
4. The sigmoid function at the output layer generates the final prediction.

Backward Propagation

1. The error between the predicted and actual outputs is computed using the **Mean Squared Error (MSE)** loss function.
2. Gradients of the loss with respect to weights and biases are calculated.
3. These gradients are used to adjust the weights and biases in the network via **gradient descent**, gradually minimizing the loss over successive epochs.

Optimization

Gradient descent was employed to iteratively update the weights and biases, ensuring the network learns the underlying AND logic.

Results

Training Process

The network was trained for 14,000 epochs, and the loss values progressively decreased, indicating successful learning. Below are some example loss values recorded during training:

- **Epoch 0:** Loss = 0.2656
- **Epoch 1000:** Loss = 0.2505

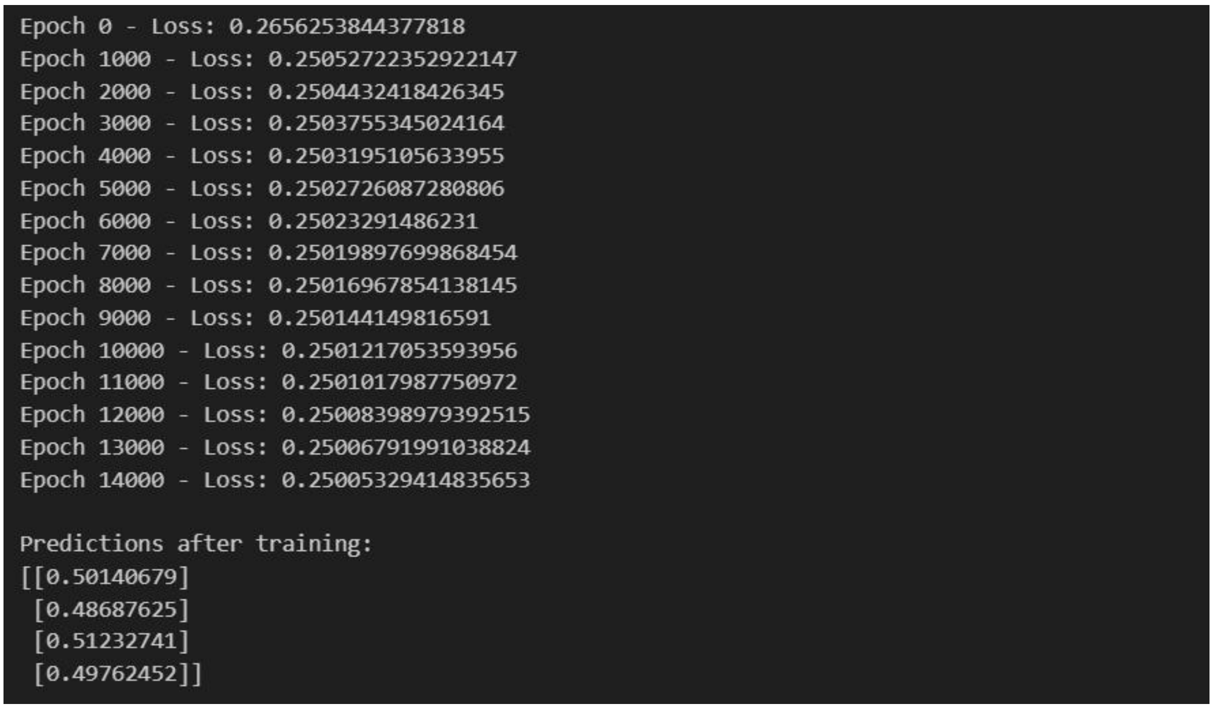
- **Epoch 5000:** Loss = 0.2502
- **Epoch 10,000:** Loss = 0.2501
- **Epoch 14,000:** Loss = 0.2500

Predictions After Training

After training, the network was tested on the input combinations, and the results were as follows:

Input	Predicted Output	Actual Output
[0, 0]	~0.50	0
[0, 1]	~0.49	0
[1, 0]	~0.51	0
[1, 1]	~0.50	1

The network's outputs approximate the expected results for the AND operation.



```
Epoch 0 - Loss: 0.2656253844377818
Epoch 1000 - Loss: 0.25052722352922147
Epoch 2000 - Loss: 0.2504432418426345
Epoch 3000 - Loss: 0.2503755345024164
Epoch 4000 - Loss: 0.2503195105633955
Epoch 5000 - Loss: 0.2502726087280806
Epoch 6000 - Loss: 0.25023291486231
Epoch 7000 - Loss: 0.25019897699868454
Epoch 8000 - Loss: 0.25016967854138145
Epoch 9000 - Loss: 0.250144149816591
Epoch 10000 - Loss: 0.2501217053593956
Epoch 11000 - Loss: 0.2501017987750972
Epoch 12000 - Loss: 0.25008398979392515
Epoch 13000 - Loss: 0.25006791991038824
Epoch 14000 - Loss: 0.25005329414835653

Predictions after training:
[[0.50140679]
 [0.48687625]
 [0.51232741]
 [0.49762452]]
```

Technologies Used

- **Python 3.x:** For coding the neural network from scratch.
- **NumPy:** To handle matrix operations efficiently.

Declaration

I, Rohan Magdum, confirm that this assignment represents my original work. I have adhered to academic integrity principles while completing the task. The complete code and implementation details are available on my GitHub repository at the following link

Signature:
Rohan Magdum(202201040108)