MIT Academy of Engineering, Alandi,Pune

School of Computer Engineering

Course - Deep Learning lab

ASSIGNMENT NO 1

Student Information:

- **Name: Vedant Bhagwat Puri**
- **Roll Number: 78**
- **Batch: T2**
- **Date of Submission:**
20-01-2025

Neural Network Implementation from Scratch

- **Objective:**

Implement a simple feedforward neural network from scratch in Python without using any in-built deep learning libraries. This implementation will focus on basic components like forward pass, backward propagation (backpropagation), and training using gradient descent.

2) Problem Definition

- **Dataset:**

For this implementation, we'll use the XOR dataset, a classic example for testing neural network capabilities.

- **Task:**

The task is binary classification, where the network predicts a single output (0 or 1) based on the two binary inputs.

3) Methodology

- **Neural Network Architecture:**

1. **Input Layer**: 2 neurons for the two inputs.
2. **Hidden Layer**: 4 neurons with Sigmoid activation to introduce non-linearity.
3. **Output Layer**: 1 neuron with Sigmoid activation to output a value between 0 and 1 for binary classification.

- **Forward Pass**

From Input to Hidden Layer:

1. First, calculate the weighted sum of inputs and add a bias term. This determines the value that will be passed into the hidden layer.
2. Then, apply a Sigmoid activation function to this sum, producing the output for the hidden layer.

From Hidden to Output Layer:

1. Next, compute the weighted sum of the hidden layer outputs, adding a bias term. This value is passed to the output layer.
2. Apply the Sigmoid activation function to this sum to generate the network's final output.

- **Backpropagation**

Output Layer Error:

1. Compute the difference between the predicted output and the actual target output.
2. This error is then multiplied by the derivative of the Sigmoid function applied to the output's input.

Hidden Layer Error:

1. Propagate the error back from the output layer to the hidden layer. This is done by multiplying the output error by the weights connecting the output to the hidden layer.
2. Multiply the result by the derivative of the Sigmoid function applied to the hidden layer's input.

Updating Weights and Biases:

1. Use gradient descent to update the weights and biases based on the calculated gradients. The weights are adjusted in a way that reduces the error.

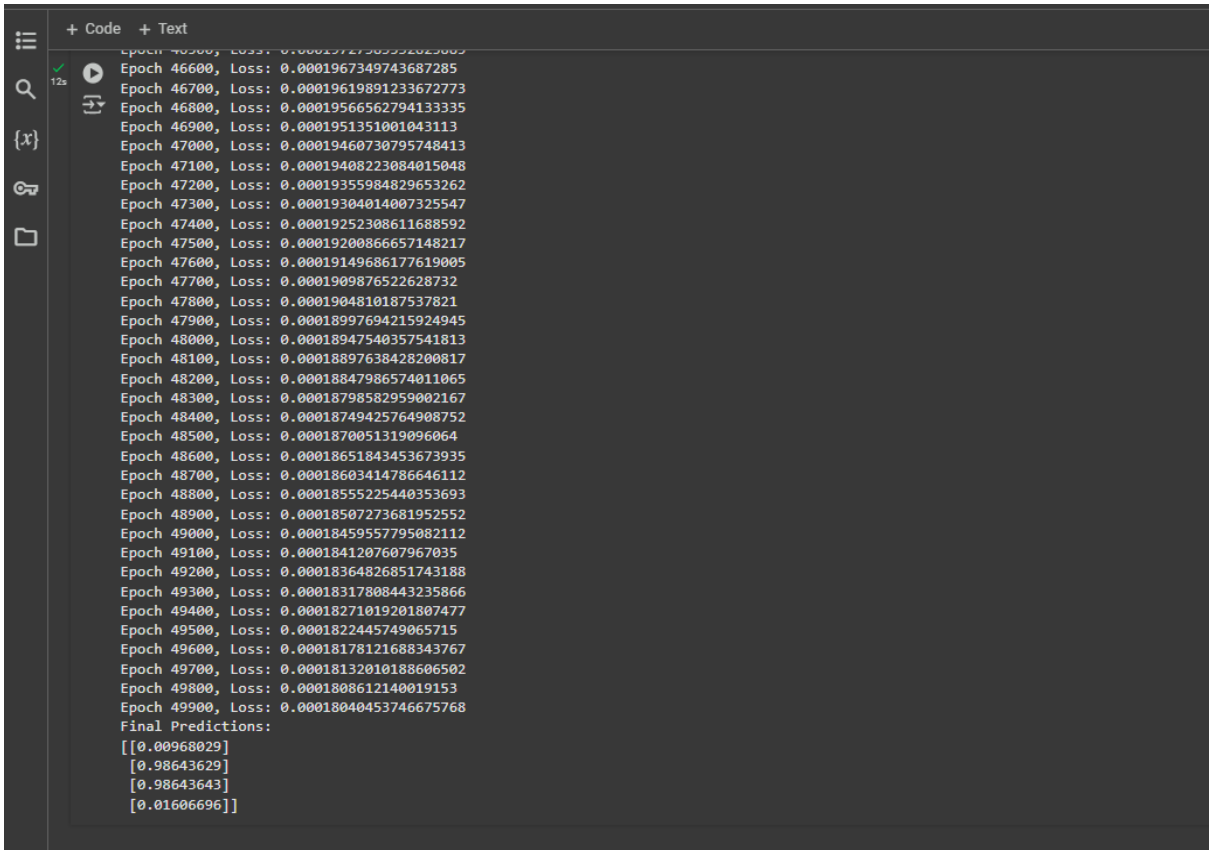
- **Loss Function**

For measuring how well the network performs, the Mean Squared Error (MSE) loss function is used. It calculates the average squared difference between predicted and actual values.

- **Optimization:**

Gradient Descent is used for optimization. This involves iteratively updating weights and biases to minimize the loss function.

OUTPUTS:


+ Code
+ Text
12s
Epoch 46600, Loss: 0.0001967349743687285
Epoch 46700, Loss: 0.0001961989123367273
Epoch 46800, Loss: 0.0001956656279413335
Epoch 46900, Loss: 0.0001951351001043113
Epoch 47000, Loss: 0.0001946073079574841
Epoch 47100, Loss: 0.0001940822308401504
Epoch 47200, Loss: 0.0001935598482965326
Epoch 47300, Loss: 0.0001930401400732554
Epoch 47400, Loss: 0.0001925230861168859
Epoch 47500, Loss: 0.0001920086665714821
Epoch 47600, Loss: 0.0001914968617761900
Epoch 47700, Loss: 0.000190987652262873
Epoch 47800, Loss: 0.0001904810187537821
Epoch 47900, Loss: 0.0001899769421592494
Epoch 48000, Loss: 0.0001894754035754181
Epoch 48100, Loss: 0.0001889763842820081
Epoch 48200, Loss: 0.0001884798657401106
Epoch 48300, Loss: 0.0001879858295900216
Epoch 48400, Loss: 0.0001874942576490875
Epoch 48500, Loss: 0.000187005131909606
Epoch 48600, Loss: 0.0001865184345367393
Epoch 48700, Loss: 0.0001860341478664611
Epoch 48800, Loss: 0.0001855522544035369
Epoch 48900, Loss: 0.0001850727368195255
Epoch 49000, Loss: 0.0001845955779508211
Epoch 49100, Loss: 0.000184120760796703
Epoch 49200, Loss: 0.0001836482685174318
Epoch 49300, Loss: 0.0001831780844323586
Epoch 49400, Loss: 0.0001827101920180747
Epoch 49500, Loss: 0.000182244574906571
Epoch 49600, Loss: 0.0001817812168834376
Epoch 49700, Loss: 0.0001813201018860650
Epoch 49800, Loss: 0.000180861214001915
Epoch 49900, Loss: 0.0001804045374667576
Final Predictions:
[[0.00968029]
[0.98643629]
[0.98643643]
[0.01606696]]

Declaration: I Vedant Puri, confirm that the work submitted in this assignment is my own and has been completed following academic integrity guidelines. The code is uploaded on my GitHub repository account, and the repository link is provided below:

GitHub Repository Link:
https://github.com/vedant018/Vedant_Puri_DL_LAB_ass1

Colab file link: https://colab.research.google.com/drive/1QNchiAH-4_3UfosnFTxxXDukM0naEzs4?usp=sharing

Signature: Vedant Bhagwat Puri