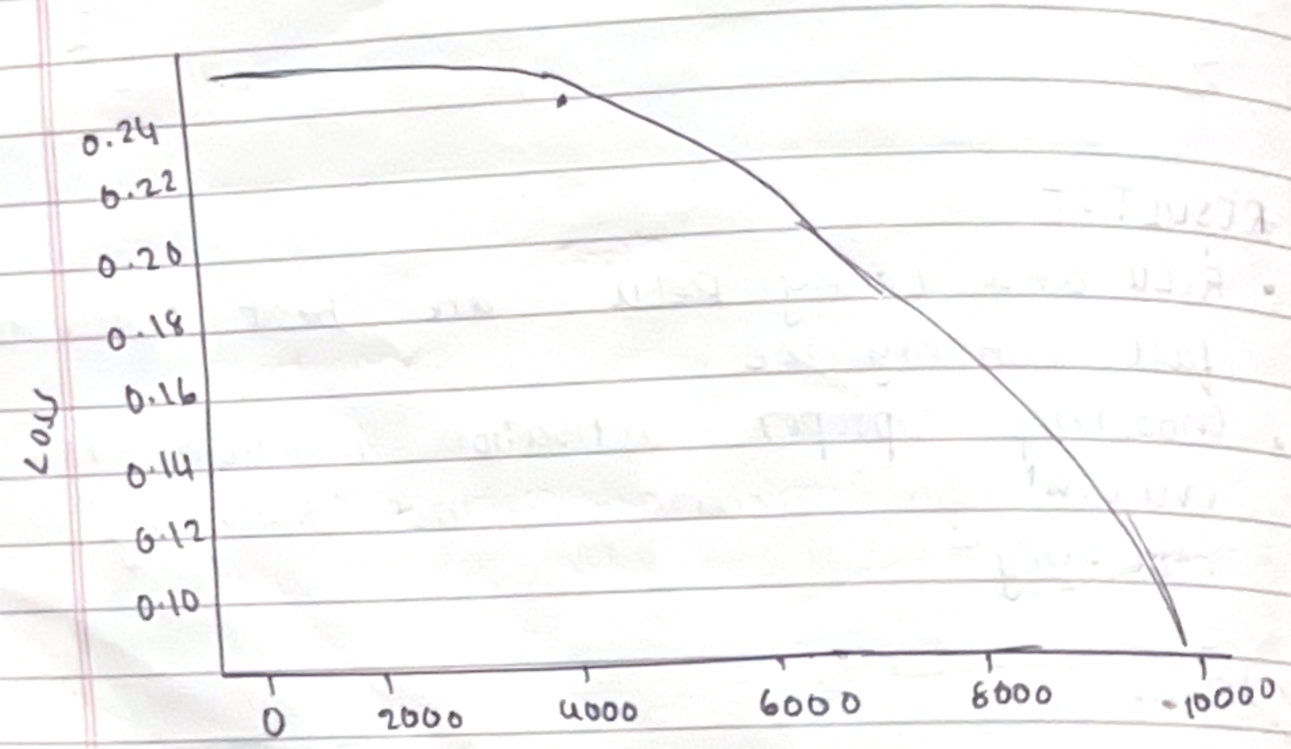
OUTPUT

Epoch 0 loss :02501 , Accuracy : 0.25
Epoch 2000 loss: 0.2495 , Accuracy 0.350
Epoch 4000 loss: 0.2490 , Accuracy 0.350
Epoch 6000 loss: 0.2308 , Accuracy 0.75
Epoch 8000 loss: 0.1779 , Accuracy 0.75
Epoch 10000 loss: 0.1673 , Accuracy 1.00

Training loss Curve.



09/09/25

LAB-6:

Implement Gradient Descent and Backpropogation in Deep Neural Network.

AIM:-

To implement gradient descent and back propogation in a deep neural network and study their role in optimizing model parameters.

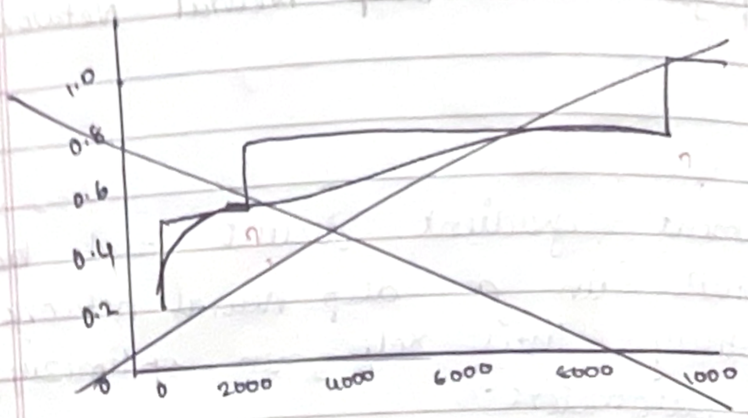
OBJECTIVES:-

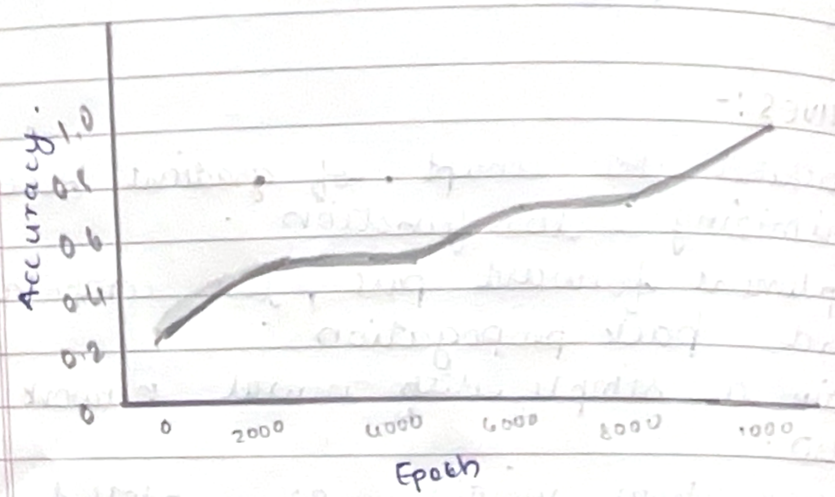
> Understand the concept of gradient descent for minimizing loss function.
> Implement foward pass, loss compatation, and back propogation.
> Train a simple deep neural network using SGD.
> Observe how weights are updated through iterations.
> Compare convergence behaviour with different learning rates.

PSEUDOCODE:-

1. Import required libraries
2. Load dataset
3. Define a deep neural network with multiple layer.
4. Forward pass:
   Compute weighted sum & apply

Training Accuracy Curve.





OBSERVATIONS:

> loss curve decreases steadily as epoch increase
> Accuracy curve rises to 100% as the network learn dataset.
> Decision Boundary separates XOR output correc
> Final predictions match XOR truth table.

[0, 1, 1, 0]

activation functions.
5. Compute Loss MSE Loss
6. Backward pass:
   - Compute gradients of loss w.r.t weights using back propogations.
   - Update weights using Gradient Descent
   $W = W - \eta * dL/dW$
7. Repeat for multiple epochs.
8. Evaluate training & testing accuracy.
9. Visualize loss vs epochs.

- Gradient Descent: Optimization method to minimize loss by updating weights step by step.
- Back propogation: Uses chain rule to calculate gradients layer by layer.
- Learning Rate: Controls update size;
  too high = unstable,
  too low = slow

Result:

The neural network successfully learned the XOR problem using gradient descent and back propogation, achieving 100% training accuracy.