=================================================================== **Exp No 3 Neural Network With One Hidden Layer  
 ====================================================================  
  
Aim…**

To build a neural network with one hidden layer for 2-class classification.

**Code…**

**import numpy as np**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.preprocessing import StandardScaler**

**from sklearn.datasets import load\_breast\_cancer**

**import matplotlib.pyplot as plt**

**# Load dataset**

**data = load\_breast\_cancer()**

**X = data.data**

**y = data.target.reshape(-1, 1) # Ensure y is a column vector**

**# Normalize**

**scaler = StandardScaler()**

**X = scaler.fit\_transform(X)**

**# Train-test split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=1)**

**# Initialize weights and biases**

**input\_dim = X.shape[1]**

**hidden\_dim = 10**

**output\_dim = 1**

**W1 = np.random.randn(input\_dim, hidden\_dim) \* 0.01**

**b1 = np.zeros((1, hidden\_dim))**

**W2 = np.random.randn(hidden\_dim, output\_dim) \* 0.01**

**b2 = np.zeros((1, output\_dim))**

**def sigmoid(z):**

**return 1 / (1 + np.exp(-z))**

**def tanh\_derivative(a):**

**return 1 - np.power(a, 2)**

**# Training parameters**

**epochs = 1000**

**lr = 0.01**

**losses = []**

**# Training loop**

**for epoch in range(epochs):**

**# Forward pass**

**Z1 = np.dot(X\_train, W1) + b1**

**A1 = np.tanh(Z1)**

**Z2 = np.dot(A1, W2) + b2**

**A2 = sigmoid(Z2)**

**# Loss calculation (binary cross-entropy)**

**loss = -np.mean(y\_train \* np.log(A2 + 1e-9) + (1 - y\_train) \* np.log(1 - A2 + 1e-9))**

**losses.append(loss)**

**# Backward pass**

**dZ2 = A2 - y\_train**

**dW2 = np.dot(A1.T, dZ2)**

**db2 = np.sum(dZ2, axis=0, keepdims=True)**

**dA1 = np.dot(dZ2, W2.T)**

**dZ1 = dA1 \* tanh\_derivative(A1)**

**dW1 = np.dot(X\_train.T, dZ1)**

**db1 = np.sum(dZ1, axis=0, keepdims=True)**

**# Gradient update**

**W1 -= lr \* dW1**

**b1 -= lr \* db1**

**W2 -= lr \* dW2**

**b2 -= lr \* db2**

**# Print loss every 100 epochs**

**if epoch % 100 == 0:**

**print(f"Epoch {epoch}, Loss: {loss:.4f}")**

**# Prediction on test set**

**Z1\_test = np.dot(X\_test, W1) + b1**

**A1\_test = np.tanh(Z1\_test)**

**Z2\_test = np.dot(A1\_test, W2) + b2**

**A2\_test = sigmoid(Z2\_test)**

**y\_pred = (A2\_test > 0.5).astype(int)**

**accuracy = np.mean(y\_pred == y\_test) \* 100**

**print(f"Test Accuracy: {accuracy:.2f}%")**

**# Plot the loss curve**

**plt.plot(losses)**

**plt.title("Loss Curve")**

**plt.xlabel("Epochs")**

**plt.ylabel("Loss")**

**plt.grid(True)**

**plt.show()**

Output…

====== RESTART: C:/Users/tgane/OneDrive/Desktop/5th sem/dlt/Exp 3/Exp 3.py =====

Epoch 0, Loss: 0.6935

Epoch 100, Loss: 0.0138

Epoch 200, Loss: 0.0052

Epoch 300, Loss: 0.0026

Epoch 400, Loss: 0.0017

Epoch 500, Loss: 0.0012

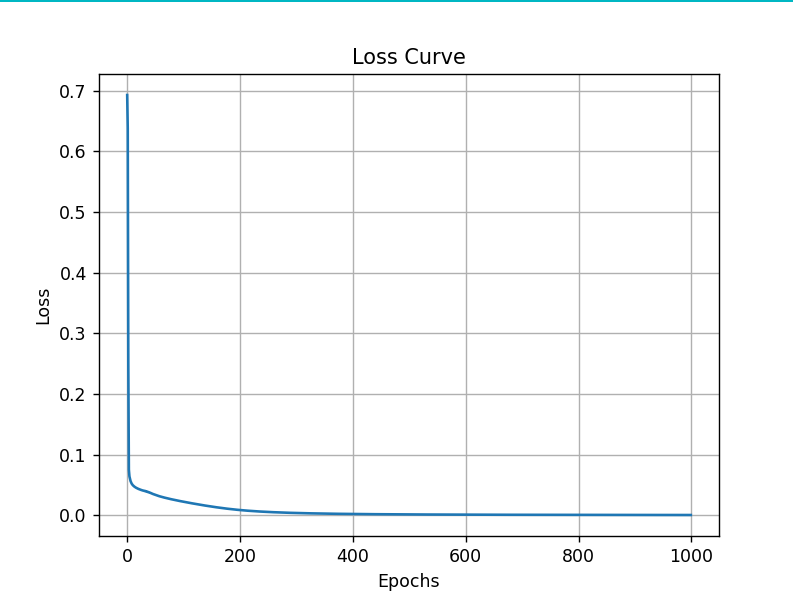
Epoch 600, Loss: 0.0009

Epoch 700, Loss: 0.0007

Epoch 800, Loss: 0.0006

Epoch 900, Loss: 0.0005

Test Accuracy: 97.37%

****