In [1]:
```
#importing the Libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_iris
from sklearn.metrics import confusion_matrix, f1_score
```

In [2]:
```
#Activation Functions and loss function as Categorical cross entropy and one hot encod

#1.Ada-Act Activation function g(x) = k0+k1*x and its derivative d(g(x))/dx = k1
def ada_act(x,k0,k1):
    return k0+k1*x

def ada_act_der(Z):
    return k1

#2.softmax activation Function
def softmax(x):
    exp_x = np.exp(x - np.max(x, axis=1, keepdims=True))
    return exp_x / np.sum(exp_x, axis=1, keepdims=True)

#3.Loss function
def categorical_crossentropy(y_true,y_pred):
    return -np.sum(y_true*np.log(y_pred+1e-10))/y_true.shape[0]

#4.one hot encoding
def one_hot_encoding(y):
    n_classes = np.max(y)+1
    return np.eye(n_classes)[y]
```

In [3]:
```
# Defining our assumed MLP architecture with two hidden layers
class MLP:
    def __init__(self, input_dim, hidden_dim1, hidden_dim2, output_dim, k0_h1, k1_h1,
        self.input_dim = input_dim
        self.hidden_dim1 = hidden_dim1
        self.hidden_dim2 = hidden_dim2
        self.output_dim = output_dim
        self.k0_h1 = k0_h1
        self.k1_h1 = k1_h1
        self.k0_h2 = k0_h2
        self.k1_h2 = k1_h2

        # Initialize weights and biases for each layer and assuming the stdandard norm
        self.w1 = np.random.randn(input_dim, hidden_dim1)
        self.b1 = np.random.randn(hidden_dim1)
        self.w2 = np.random.randn(hidden_dim1, hidden_dim2)
        self.b2 = np.random.randn(hidden_dim2)
        self.w3 = np.random.randn(hidden_dim2, output_dim)
        self.b3 = np.random.randn(output_dim)

        #Forward Propagation of network
    def forward_propagation(self, X):
        z1 = np.dot(X, self.w1) + self.b1
        a1 = ada_act(z1, self.k0_h1, self.k1_h1)
```

```
        z2 = np.dot(a1, self.w2) + self.b2
        a2 = ada_act(z2, self.k0_h2, self.k1_h2)

        z3 = np.dot(a2, self.w3) + self.b3
        a3 = softmax(z3)

        return a1, a2, a3


    #Backward propagtaion of network
    def backward_propagation(self, X, y, a1, a2, a3, learning_rate):
        m = X.shape[0]

        # Compute gradients for the output layer
        dz3 = a3 - y
        dw3 = np.dot(a2.T, dz3) / m
        db3 = np.mean(dz3, axis=0)

        # Compute gradients for the second hidden layer
        da2 = np.dot(dz3, self.w3.T)
        dz2 = da2 * self.k1_h2
        dw2 = np.dot(a1.T, dz2) / m
        db2 = np.mean(dz2, axis=0)

        # Compute gradients for the first hidden layer
        da1 = np.dot(dz2, self.w2.T)
        dz1 = da1 * self.k1_h1
        dw1 = np.dot(X.T, dz1) / m
        db1 = np.mean(dz1, axis=0)

        # Update weights and biases using gradient descent
        self.w1 -= learning_rate * dw1
        self.b1 -= learning_rate * db1
        self.w2 -= learning_rate * dw2
        self.b2 -= learning_rate * db2
        self.w3 -= learning_rate * dw3
        self.b3 -= learning_rate * db3


    #Training of neural network
    def train(self, X_train, y_train, X_test, y_test, num_epochs, learning_rate):
        train_losses = []
        test_losses = []
        train_accs = []
        test_accs = []
        weight_updates = []

        for epoch in range(num_epochs):
            # Forward propagation
            a1, a2, a3 = self.forward_propagation(X_train)

            # Compute loss and accuracy for training set
            train_loss = categorical_crossentropy(y_train, a3)
            train_losses.append(train_loss)
            train_pred_labels = np.argmax(a3, axis=1)
            train_true_labels = np.argmax(y_train, axis=1)
            train_acc = accuracy_score(train_true_labels, train_pred_labels)
            train_accs.append(train_acc)

            # Backward propagation and weight updates
```

```
            self.backward_propagation(X_train, y_train, a1, a2, a3, learning_rate)

            # Compute loss and accuracy for test set
            a1_test, a2_test, a3_test = self.forward_propagation(X_test)
            test_loss = categorical_crossentropy(y_test, a3_test)
            test_losses.append(test_loss)
            test_pred_labels = np.argmax(a3_test, axis=1)
            test_true_labels = np.argmax(y_test, axis=1)
            test_acc = accuracy_score(test_true_labels, test_pred_labels)
            test_accs.append(test_acc)


            # Save the weight updates for each layer
            weight_updates.append((np.copy(self.w1), np.copy(self.b1), np.copy(self.w2
            print(f"Epoch {epoch + 1}/{num_epochs}, Train Loss: {train_loss:.4f}, Test

        return train_losses, test_losses, train_accs, test_accs,weight_updates


    def predict(self, X):
        _, _, predictions = self.forward_propagation(X)
        return np.argmax(predictions, axis=1)
```

In [4]:
```
# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
```

In [5]:
```
#one hot encoding
y_encoded = one_hot_encoding(y)
```

In [6]:
```
# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, randc
```

In [7]:
```
# Normalize the input features (optional but recommended)
X_train = (X_train - X_train.mean(axis=0)) / X_train.std(axis=0)
X_test = (X_test - X_test.mean(axis=0)) / X_test.std(axis=0)
```

In [8]:
```
# Initialize the MLP with the specified architecture and parameters
input_dim = X_train.shape[1]
hidden_dim1 = 16
hidden_dim2 = 8
output_dim = y_train.shape[1]  # Number of classes
k0_h1, k1_h1 = np.random.randn(), np.random.randn()
k0_h2, k1_h2 = np.random.randn(), np.random.randn()
```

In [14]:
```
mlp = MLP(input_dim, hidden_dim1, hidden_dim2, output_dim, k0_h1, k1_h1, k0_h2, k1_h2)

# Set hyperparameters
num_epochs = 100
learning_rate = 0.01

# Train the model and get the evaluation metrics
train_losses, test_losses, train_accs, test_accs,weight_updates = mlp.train(
    X_train, y_train, X_test, y_test, num_epochs, learning_rate
)
```

```
Epoch 1/100, Train Loss: 8.4827, Test Loss: 8.9482, Train Acc: 0.5083, Test Acc: 0.46
67
Epoch 2/100, Train Loss: 8.1236, Test Loss: 8.5586, Train Acc: 0.5083, Test Acc: 0.43
33
Epoch 3/100, Train Loss: 8.0801, Test Loss: 4.9950, Train Acc: 0.4083, Test Acc: 0.43
33
Epoch 4/100, Train Loss: 5.0570, Test Loss: 1.5439, Train Acc: 0.4250, Test Acc: 0.66
67
Epoch 5/100, Train Loss: 1.5262, Test Loss: 0.2851, Train Acc: 0.6167, Test Acc: 0.83
33
Epoch 6/100, Train Loss: 0.3315, Test Loss: 0.1120, Train Acc: 0.9083, Test Acc: 1.00
00
Epoch 7/100, Train Loss: 0.1790, Test Loss: 0.0760, Train Acc: 0.9333, Test Acc: 1.00
00
Epoch 8/100, Train Loss: 0.1328, Test Loss: 0.0651, Train Acc: 0.9500, Test Acc: 1.00
00
Epoch 9/100, Train Loss: 0.1146, Test Loss: 0.0603, Train Acc: 0.9583, Test Acc: 1.00
00
Epoch 10/100, Train Loss: 0.1073, Test Loss: 0.0579, Train Acc: 0.9583, Test Acc: 1.0
000
Epoch 11/100, Train Loss: 0.1037, Test Loss: 0.0565, Train Acc: 0.9583, Test Acc: 1.0
000
Epoch 12/100, Train Loss: 0.1015, Test Loss: 0.0556, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 13/100, Train Loss: 0.1000, Test Loss: 0.0551, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 14/100, Train Loss: 0.0989, Test Loss: 0.0548, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 15/100, Train Loss: 0.0979, Test Loss: 0.0546, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 16/100, Train Loss: 0.0970, Test Loss: 0.0545, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 17/100, Train Loss: 0.0962, Test Loss: 0.0544, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 18/100, Train Loss: 0.0955, Test Loss: 0.0544, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 19/100, Train Loss: 0.0948, Test Loss: 0.0544, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 20/100, Train Loss: 0.0941, Test Loss: 0.0544, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 21/100, Train Loss: 0.0935, Test Loss: 0.0545, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 22/100, Train Loss: 0.0929, Test Loss: 0.0545, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 23/100, Train Loss: 0.0923, Test Loss: 0.0546, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 24/100, Train Loss: 0.0917, Test Loss: 0.0547, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 25/100, Train Loss: 0.0912, Test Loss: 0.0548, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 26/100, Train Loss: 0.0906, Test Loss: 0.0549, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 27/100, Train Loss: 0.0901, Test Loss: 0.0551, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 28/100, Train Loss: 0.0896, Test Loss: 0.0552, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 29/100, Train Loss: 0.0891, Test Loss: 0.0553, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 30/100, Train Loss: 0.0886, Test Loss: 0.0555, Train Acc: 0.9667, Test Acc: 1.0
000
```

```
Epoch 31/100, Train Loss: 0.0882, Test Loss: 0.0556, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 32/100, Train Loss: 0.0877, Test Loss: 0.0558, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 33/100, Train Loss: 0.0873, Test Loss: 0.0559, Train Acc: 0.9667, Test Acc: 1.0
000
Epoch 34/100, Train Loss: 0.0869, Test Loss: 0.0561, Train Acc: 0.9750, Test Acc: 1.0
000
Epoch 35/100, Train Loss: 0.0865, Test Loss: 0.0563, Train Acc: 0.9750, Test Acc: 1.0
000
Epoch 36/100, Train Loss: 0.0861, Test Loss: 0.0564, Train Acc: 0.9750, Test Acc: 1.0
000
Epoch 37/100, Train Loss: 0.0857, Test Loss: 0.0566, Train Acc: 0.9750, Test Acc: 1.0
000
Epoch 38/100, Train Loss: 0.0853, Test Loss: 0.0568, Train Acc: 0.9750, Test Acc: 1.0
000
Epoch 39/100, Train Loss: 0.0849, Test Loss: 0.0570, Train Acc: 0.9750, Test Acc: 1.0
000
Epoch 40/100, Train Loss: 0.0846, Test Loss: 0.0571, Train Acc: 0.9750, Test Acc: 1.0
000
Epoch 41/100, Train Loss: 0.0842, Test Loss: 0.0573, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 42/100, Train Loss: 0.0839, Test Loss: 0.0575, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 43/100, Train Loss: 0.0836, Test Loss: 0.0577, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 44/100, Train Loss: 0.0833, Test Loss: 0.0579, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 45/100, Train Loss: 0.0829, Test Loss: 0.0581, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 46/100, Train Loss: 0.0826, Test Loss: 0.0582, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 47/100, Train Loss: 0.0824, Test Loss: 0.0584, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 48/100, Train Loss: 0.0821, Test Loss: 0.0586, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 49/100, Train Loss: 0.0818, Test Loss: 0.0588, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 50/100, Train Loss: 0.0815, Test Loss: 0.0590, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 51/100, Train Loss: 0.0813, Test Loss: 0.0592, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 52/100, Train Loss: 0.0810, Test Loss: 0.0593, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 53/100, Train Loss: 0.0807, Test Loss: 0.0595, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 54/100, Train Loss: 0.0805, Test Loss: 0.0597, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 55/100, Train Loss: 0.0803, Test Loss: 0.0599, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 56/100, Train Loss: 0.0800, Test Loss: 0.0601, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 57/100, Train Loss: 0.0798, Test Loss: 0.0603, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 58/100, Train Loss: 0.0796, Test Loss: 0.0604, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 59/100, Train Loss: 0.0794, Test Loss: 0.0606, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 60/100, Train Loss: 0.0791, Test Loss: 0.0608, Train Acc: 0.9750, Test Acc: 0.9
667
```

```
Epoch 61/100, Train Loss: 0.0789, Test Loss: 0.0610, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 62/100, Train Loss: 0.0787, Test Loss: 0.0612, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 63/100, Train Loss: 0.0785, Test Loss: 0.0613, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 64/100, Train Loss: 0.0783, Test Loss: 0.0615, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 65/100, Train Loss: 0.0781, Test Loss: 0.0617, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 66/100, Train Loss: 0.0779, Test Loss: 0.0619, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 67/100, Train Loss: 0.0778, Test Loss: 0.0620, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 68/100, Train Loss: 0.0776, Test Loss: 0.0622, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 69/100, Train Loss: 0.0774, Test Loss: 0.0624, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 70/100, Train Loss: 0.0772, Test Loss: 0.0625, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 71/100, Train Loss: 0.0771, Test Loss: 0.0627, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 72/100, Train Loss: 0.0769, Test Loss: 0.0629, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 73/100, Train Loss: 0.0767, Test Loss: 0.0630, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 74/100, Train Loss: 0.0766, Test Loss: 0.0632, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 75/100, Train Loss: 0.0764, Test Loss: 0.0634, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 76/100, Train Loss: 0.0763, Test Loss: 0.0635, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 77/100, Train Loss: 0.0761, Test Loss: 0.0637, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 78/100, Train Loss: 0.0760, Test Loss: 0.0638, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 79/100, Train Loss: 0.0758, Test Loss: 0.0640, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 80/100, Train Loss: 0.0757, Test Loss: 0.0641, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 81/100, Train Loss: 0.0755, Test Loss: 0.0643, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 82/100, Train Loss: 0.0754, Test Loss: 0.0644, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 83/100, Train Loss: 0.0753, Test Loss: 0.0646, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 84/100, Train Loss: 0.0751, Test Loss: 0.0647, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 85/100, Train Loss: 0.0750, Test Loss: 0.0649, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 86/100, Train Loss: 0.0749, Test Loss: 0.0650, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 87/100, Train Loss: 0.0747, Test Loss: 0.0652, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 88/100, Train Loss: 0.0746, Test Loss: 0.0653, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 89/100, Train Loss: 0.0745, Test Loss: 0.0654, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 90/100, Train Loss: 0.0744, Test Loss: 0.0656, Train Acc: 0.9750, Test Acc: 0.9
667
```

```
Epoch 91/100, Train Loss: 0.0742, Test Loss: 0.0657, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 92/100, Train Loss: 0.0741, Test Loss: 0.0659, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 93/100, Train Loss: 0.0740, Test Loss: 0.0660, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 94/100, Train Loss: 0.0739, Test Loss: 0.0661, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 95/100, Train Loss: 0.0738, Test Loss: 0.0663, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 96/100, Train Loss: 0.0737, Test Loss: 0.0664, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 97/100, Train Loss: 0.0736, Test Loss: 0.0665, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 98/100, Train Loss: 0.0735, Test Loss: 0.0666, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 99/100, Train Loss: 0.0734, Test Loss: 0.0668, Train Acc: 0.9750, Test Acc: 0.9
667
Epoch 100/100, Train Loss: 0.0732, Test Loss: 0.0669, Train Acc: 0.9750, Test Acc: 0.
9667
```

In [15]:

```python
# Predict on the test set
y_pred = mlp.predict(X_test)

# Convert one-hot encoded labels back to original labels
y_test_original = np.argmax(y_test, axis=1)

# Calculate the confusion matrix
conf_matrix = confusion_matrix(y_test_original, y_pred)

# Calculate the F1-score
f1_score_value = f1_score(y_test_original, y_pred, average='weighted')
```

In [16]:

```python
#printing the values
print("Final Parameter Values:")
print("Weight 1:", mlp.w1)
print("Bias 1:", mlp.b1)
print("Weight 2:", mlp.w2)
print("Bias 2:", mlp.b2)
print("Weight 3:", mlp.w3)
print("Bias 3:", mlp.b3)
```

```
Final Parameter Values:
Weight 1: [[-1.12227582e-01 -5.04136053e-01  2.04456023e+00 -1.98367302e+00
  -6.07512108e-01  1.03841982e+00 -1.79393241e+00 -1.61402252e+00
  -8.76402712e-01 -6.82653764e-01 -5.44430365e-01  1.63305570e+00
   8.40899176e-01 -1.32131255e+00 -1.63064162e+00  1.44506603e-03]
 [ 7.80821689e-03 -1.19391000e+00 -4.17480967e-01 -5.32338223e-01
   1.02533759e+00  2.38609974e-02 -5.50969675e-01 -4.72362009e-02
  -1.64120198e+00 -6.68056973e-01 -6.71467333e-03 -2.10687554e+00
   6.14337092e-01 -1.07987317e+00 -4.08365004e-01 -6.46592349e-01]
 [-7.08837898e-01 -7.31106008e-01 -1.93305113e+00 -6.71078162e-01
   9.13256468e-01  2.45231323e-01 -2.60804622e+00  6.22603045e-01
   2.59669651e-01  4.05810213e-01  9.53566344e-01 -1.38128814e-03
   2.02035965e+00  1.53240720e+00 -9.30027956e-01  3.89330991e-01]
 [ 1.79807496e-01 -1.22265716e+00  1.02541367e+00  4.24992460e-01
  -1.33719983e+00  1.01916791e+00  1.93898755e-01 -2.73774580e-01
  -1.05056053e-01 -1.09966437e+00 -6.39616964e-01  1.58640805e+00
  -2.35738921e+00 -5.46209311e-01  5.91330941e-01  5.40560825e-01]]
Bias 1: [-0.3619391  -1.90166435 -1.13071976 -0.81583999  0.90682824 -1.07918614
 -1.49210842  0.22500749  0.44209684 -0.97728235 -0.44357085 -1.00535647
  0.01107369 -0.87532247 -0.7999712   0.36955016]
Weight 2: [[-4.68290417e-02  1.52071917e+00 -7.11330465e-02 -3.28673412e-01
   1.29136881e+00 -1.45013421e+00  1.56837975e+00 -8.05480209e-01]
 [ 1.73419433e-01  8.82901527e-01 -1.24387024e+00 -1.10986677e+00
   2.71971991e-01  1.45932915e+00 -6.19913093e-01 -8.10342696e-01]
 [ 2.25234966e+00 -1.28915870e+00  9.78105926e-01 -5.34133793e-01
  -1.12388113e+00  9.12401289e-02 -7.97972954e-01 -4.61934778e-01]
 [-8.31476637e-01 -2.94785611e-01 -2.28879513e-01  1.50477786e+00
   6.36529776e-01 -7.48678799e-01 -1.92149713e+00 -2.54495461e+00]
 [ 1.14542113e-01  5.40091163e-01  1.52660438e-01 -7.30355637e-01
  -1.66354334e-01 -4.45910209e-01 -2.22502847e+00  2.59597089e-01]
 [-2.75919189e-01 -2.12836700e-01  3.79483330e-01 -1.46352198e-01
   4.19303167e-01 -1.17359364e+00 -8.34459314e-01 -1.30379820e+00]
 [-2.06108619e+00 -1.34161687e+00 -8.15637863e-01 -9.43376485e-01
   1.08262296e+00  6.85462517e-01 -1.74804863e-01 -4.17459290e-01]
 [-3.96667888e-01 -2.23564621e+00 -7.20380366e-01  3.48257160e-03
  -9.58009871e-01  1.17274025e+00 -1.57215380e-01 -1.10494740e+00]
 [ 3.95161633e-01 -9.21074956e-01  1.70538534e+00 -5.04673347e-01
   1.55755908e+00  1.85856530e-03  7.32190104e-01 -1.21640645e+00]
 [ 1.39116773e+00 -1.24877166e+00  1.02205609e+00  8.32134497e-01
   3.38745024e-01 -1.36305279e-01 -9.58277207e-02  4.57273019e-01]
 [-4.04084922e-01  1.36818871e-01  7.86253491e-01 -1.30141743e+00
   2.84534476e-01  4.58310251e-01 -1.28000359e-01  1.65056821e+00]
 [-1.94679253e+00  7.25097337e-01 -7.22442601e-01 -2.83245814e-01
  -2.10611548e+00  4.47784024e-01  2.22454041e-02  1.74512023e+00]
 [ 9.85156973e-01 -1.85810689e-01 -4.36865745e-01  6.09811213e-01
  -2.14935367e-01  1.21736329e+00  2.49257566e-03 -9.39307207e-01]
 [ 8.47223281e-01  1.00241388e+00 -3.49682519e-02  1.28179715e-01
   1.40382076e-01 -5.64311359e-01  1.00303180e+00 -7.01526847e-01]
 [ 2.77483023e-01 -6.65721877e-01  3.00836522e-01 -4.46152332e-01
  -2.13110411e-01 -1.93014362e+00  1.12680052e+00 -4.49763043e-02]
 [ 6.60327462e-02 -1.92276487e+00  6.97672174e-01  6.29334981e-01
  -1.70712361e-01 -8.81594193e-01 -1.01516222e+00  3.89839848e-01]]
Bias 2: [ 0.60644606 -0.06602809  1.33671505 -0.54004893  0.64722409 -1.03025852
 -0.4607758  -2.50322505]
Weight 3: [[-0.75637102  0.65794745 -0.85431375]
 [ 0.14280159 -0.48514916  0.13897866]
 [-1.18708156  0.69926332  0.23299656]
 [ 1.75415862 -1.47623375  0.51854271]
 [ 0.18449729 -0.52420117 -0.24938856]
 [ 1.06002153 -1.28186584 -0.59023671]
```

```
 [-0.81544912 -0.77775859  0.12546583]
 [-1.05160702 -0.71584714 -0.01956184]]
Bias 3: [-0.49252972 -0.36400587 -0.34932442]
```

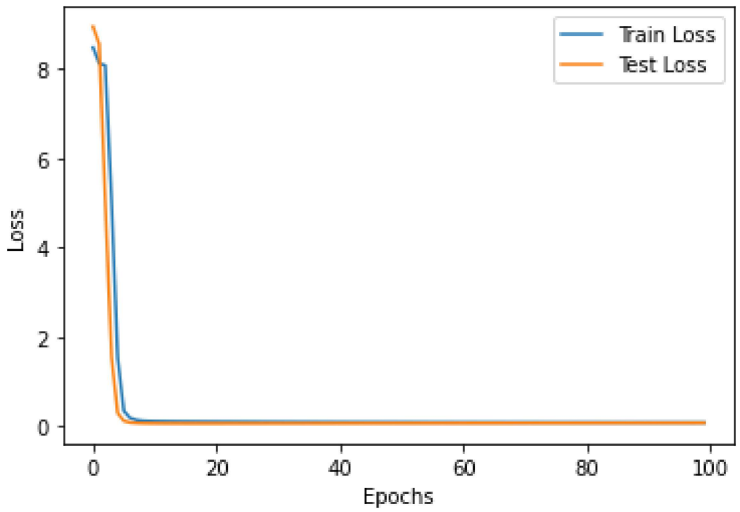
In [17]:
```
#for confusion Matrix

print("Confusion Matrix:")
print(conf_matrix)
print("F1-score:", f1_score_value)
```

```
Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  1 10]]
F1-score: 0.966750208855472
```

In [18]:
```
# Plot the loss function vs. epochs
import matplotlib.pyplot as plt

plt.plot(range(num_epochs), train_losses, label='Train Loss')
plt.plot(range(num_epochs), test_losses, label='Test Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [ ]:
```

```