**Module Assignment | Deep Learning**

**Objective**

Design, train, and tune a **Multilayer Perceptron (MLP)** to classify points in a **challenging 3-class spiral dataset**. You must choose the model architecture and hyperparameters thoughtfully to achieve good performance despite noise and complex decision boundaries.

**Dataset**

You will **generate the dataset yourself** using the following code:

—------------------------------------------------------------------------------------------------------------------------

import numpy as np

import matplotlib.pyplot as plt

def generate\_harder\_spiral\_data(points\_per\_class=200, noise=0.4, num\_classes=3):

N = points\_per\_class # points per class

D = 2 # input dimension

K = num\_classes # number of classes

X = np.zeros((N\*K,D))

y = np.zeros(N\*K, dtype='uint8')

for j in range(K):

ix = range(N\*j, N\*(j+1))

r = np.linspace(0.0, 1, N)

t = np.linspace(j\*4, (j+1)\*4, N) + np.random.randn(N)\*noise

X[ix] = np.c\_[r\*np.sin(t), r\*np.cos(t)]

y[ix] = j

return X, y

# Generate data

X, y = generate\_harder\_spiral\_data()

# Plot

plt.figure(figsize=(6,6))

plt.scatter(X[:, 0], X[:, 1], c=y, cmap="Spectral")

plt.title('Harder Spiral Data')

plt.show()

—------------------------------------------------------------------------------------------------------------------------

## **Points per class**: 200

## **Noise level**: 0.4

## **Classes**: 3

## **Tasks to Perform**

### **1. Data Preparation**

* Split into **training** and **testing** sets (e.g., 80%-20%).

### **2. Model Building**

* Build an MLP that classifies the spiral data.
* You must decide:
  + Number of hidden layers
  + Number of neurons per layer
  + Activation functions
  + Optimizer and learning rate
  + Batch size
  + Epochs (training for **at least 300–500 epochs** may be needed)

### **3. Model Evaluation**

* Plot **training and validation loss/accuracy** curves.
* Plot the **decision boundary**

### **4. Reflection Questions**

Answer these inside the notebook:

* How did you decide the number of hidden layers and neurons?
* How did different learning rates affect the results?
* Did you encounter overfitting or underfitting? How did you deal with it?
* If you had more time, how would you further improve the model?