# DA312 Advanced Machine Learning Lab
# Assignment 2

05 February, 2024

- You can write the code in Google Colab platform.
- Submit .ipynb or .py file to the Teams assignment. The code should be well commented.
- You can use in-built commands of PyTorch unless stated otherwise.

## Task A

Implement a Multi-Layer Perceptron (MLP) for digit classification on the MNIST dataset, featuring two hidden layers. Incorporate Xavier initialization for initializing weights and develop a custom RMSprop optimizer for the training process.

### MLP Model Implementation

1. Construct an MLP with two hidden layers.
2. Utilize ReLU activation for hidden layer neurons.
3. Employ Xavier initialization for weight setup.
4. Develop an RMSprop optimizer from scratch for weight updates during training.

### Model Training

Train the model with the following specifications:

1. Preprocess MNIST images by flattening and normalizing.
2. Train the MLP (for 50 epochs) on the MNIST training dataset, recording and plotting the training loss.
3. Calculate cross-entropy loss for the classification task.

### Model Evaluation

1. Assess your model on the MNIST test set.
2. Generate and analyze a confusion matrix to evaluate classification accuracy across digits.
3. Visualize the abstract representation based on the first layer activations for 10 images from the MNIST dataset (to visualize "image patterns").

4. Visualize the error surface based on two weights in the MLP's first layer (use a small subset ([:100]) of the training data for loss computation to reduce computational load). Iterate over a range of values for two selected weights, computing the loss for a subset of the data, and plotting these losses to visualize the error surface.

# Task B

Implement an Autoencoder for feature extraction on the MNIST dataset and use these features to classify digits using a Support Vector Machine (SVM) classifier. Train an Autoencoder to compress MNIST digit images into a lower-dimensional representation and then reconstruct them. The learned compressed features will then be utilized to train an SVM classifier to predict digit labels.

## Autoencoder Implementation

1. Model should contain atleast two hidden layers for the encoder and two for the decoder.

## Feature Extraction and SVM Classification

1. Extract the compressed features from the trained encoder for both the training and test sets.
2. Train an SVM classifier on the extracted features from the training set. Use the linear kernel for the SVM. (Write custom SVM code)
3. Evaluate the classifier on the test set features and report the classification accuracy.

## Model Evaluation

1. Plot a confusion matrix for the SVM classifier predictions on the test set.