# Datasets

* MNIST (<http://yann.lecun.com/exdb/mnist/)>
* CIFAR-10 (https://[www.cs.toronto.edu/~kriz/cifar.html](http://www.cs.toronto.edu/~kriz/cifar.html) )

You are allowed to use any python library for the purpose of loading data.

# The Tasks

In all tasks, you can use the built-in libraries of python (math, random, numpy, and matplotlib, etc). If you think that you might benefit from using another library, you can ask about it to the Lecturer on Moodle. You will use PyTorch in Task 2.

Note that you can use any library for the purpose of loading the datasets .

## Task 1:

The task is about classification on the **MNIST dataset**.

You can use other API’s/libraries for loading the dataset, but not for the neural network implementation. The point of this task is to develop a multi-layer neural network for classification using numpy. The task requires following sub-tasks:

##### Implement sigmoid and ReLU layers

For this sub-task, you should implement forward and backward pass for sigmoid and ReLU. You should consider presenting these activation functions in the report with any pros cons if they have.

##### Implement softmax layer

Implement softmax with both forward and backward pass. Present the softmax in the report along with any numerical issues when calculating the softmax function.

##### Implement dropout

Present dropout in the report. Implement inverted dropout. Forward and backward pass should be implemented.

**Note**: Since the test performance is critical, it is also preferable to leaving the forward pass unchanged at test time. Therefore, in most implementations *inverted dropout* is employed to overcome the undesirable property of the original dropout.

1. **Implement a fully parametrizable neural network class**

You should implement a fully-connected NN class where with number of hidden layers, units, activation functions can be changed. In addition, you can add dropout or regularizer (L1 or L2). Report the parameters used (update rule, learning rate, decay, epochs, batch size) and include the plots in your report.

##### Implement optimizer

Implement any two optimizers of your choice. Briefly present the optimizers in the report. The optimizers can be flavours of gradient descent. For instance: Stochastic gradient descent (SGD) and SGD with momentum. SGD and mini-batch gradient descent, etc.

##### Evaluate different neural network architectures/parameters, present and discuss your results.

Be creative in the analysis and discussion. Evaluate different hyperparameters. For instance: different network architectures, activation functions, comparison of optimizers, L1/L2 performance comparison with dropout, etc. Support your results with plots/graph and discussion.

## Task 2:

The second task is about implementing deep learning networks using PyTorch. You can use any imaging or time-series dataset of your choice for this task (the dataset should be discussed and approved by Lecturer)

* 1. Present the dataset and the topic of your study.
  2. Describe and implement a base model. For instance, neural network or base convolutional neural network (in the case of images).
  3. Implement atleast one improvement. Describe your motivation in the report. For instance, architecture changes etc.
  4. Optimize hyperparameters. This might include dropout, regularization, etc.
  5. Present, compare and discuss your results. Please include plots/graphs where necessary.

.