Deep learning is defined to be a type of AI that uses neural networks to learn from data and perform complex tasks. It is a subset of Machine Learning and was heavily relied on the concept of feature engineering in the past decade.

The term features refer to performing transformations on any given data to form a relationship between the input and the output. Extracting features on any given data helps us determine the output when new input data is fed to the system.

We will be using PyTorch for our deep learning algorithms as it has been proven to have a clear syntax coupled with APIs and debugging to make it easier to use. The library provides us with a data structure called tensor which is a multi-dimensional array, with properties similar to a NumPy array. The library also allows us to perform accelerating mathematical operations by leveraging the hardware on the system making it easy for us to train the neural networks. The library along with its support massive support also comes with a C++ runtime for model deployment, to help us avoid the python interpreter when not required.

When the library was released, we had Theano and TensorFlow as their low-level competitors/alternatives. Later in time, a lot of frameworks were released, each with its own pros and cons. Currently, PyTorch has an impact on the research and teaching communities, and TensorFlow mostly focuses on industrial productions and deployments.

For performance reasons and the leverage of the GPU capabilities of CUDA-enabled Nvidia GPUs, PyTorch has been mainly developed in C++. Another core feature of the library is that the tensors can keep track of the operations performed on them and can analytically derivate the output with respect to any of the inputs. This “autograd” feature enables people to leverage the library for physics, rendering, optimization and simulation tasks as well.

The Module for building neural networks is called torch.nn, which provides common neural network layers and other architectural components. As we need a data source for any model to train, we have this Dataset class in the torch.utils.data file, which helps in loading the dataset to the memory. We then have an optimizer at torch.optim that would do the updates about the results and our set criteria for the model.

The training process can be sped up using the multi-GPU support provided by the torch. The large networks can take days or months in the training process. We also have a few cloud-based service providers such as dawn.cs.stanford.edu and colab.research.google.com.

In order to keep the codes modular, maintained, and clean, we use the Jupyter Notebooks which also allow us to run the code interactively.