Python for Deep Learning

Course Objective:

Python is essential for deep learning due to its readability and extensive libraries like NumPy, pandas, and matplotlib. Its simplicity and large community make it ideal for rapid prototyping. PyTorch, a popular deep learning framework, builds upon Python's strengths, providing dynamic computation graphs and automatic differentiation. This allows for easy experimentation and efficient training of complex neural networks. PyTorch also offers GPU acceleration, enhancing performance for deep learning tasks. Together, Python and PyTorch create a user-friendly and powerful environment for researchers and practitioners, making them indispensable tools for learning and applying deep learning techniques in various domains.

This comprehensive course is designed to equip learners with the essential knowledge and practical skills needed to excel in deep learning using PyTorch. Starting with an introduction to tensors and PyTorch basics, participants will learn to build input pipelines, design neural networks, and grasp computation graphs for efficient training. The course will cover automatic differentiation, enabling the computation of gradients for optimization. Furthermore, students will delve into training deep neural networks, fine-tuning hyperparameters, and applying these techniques to computer vision tasks using CNNs. Additionally, the course will explore processing sequences with RNNs and CNNs and delve into natural language processing using RNNs and attention mechanisms and transformer architecture. Throughout the course, hands-on projects and real-world applications will solidify learners' understanding and ensure they are well-prepared to tackle complex deep learning challenges.

The course will be taught using complete hands-on approach.

The objective of the course is to enable the students to

- Understand the fundamentals of tensors and their role in PyTorch for efficient data representation.
- Master the construction of input pipelines to handle diverse datasets and streamline data pre-processing.
- Learn to design, build, and train neural networks, acquiring skills to tackle various real-world problems.
- Grasp the concept of computation graphs and their importance in optimizing deep learning models.
- Explore automatic differentiation to compute gradients, enabling efficient optimization during training.
- Gain expertise in training deep neural networks and fine-tuning hyperparameters for improved performance.
- Apply convolutional neural networks (CNNs) to solve computer vision tasks, enhancing image recognition and analysis.
- Develop proficiency in processing sequential data using recurrent neural networks (RNNs) and CNNs.
- Apply RNNs with attention mechanisms and transformers to tackle natural language processing tasks, such as language translation, sentiment analysis, Named Entity Recognition etc.
- Complete hands-on projects and real-world applications to reinforce understanding and practical application of deep learning with PyTorch.

Pre-requisites (if any):

Programming knowledge of Python and in-depth knowledge of basic machine learning concepts like supervised vs unsupervised learning, basic classifiers, and machine learning process.

Hardware, Software Requirement and Installation:

- A laptop running 64-bit OS (Linux/OSX/Windows)
- We shall be using Google Colab service and hence we would not require any installation
- Decent Internet Connection

Course Outline and Topics Covered

Module 1: PyTorch and its programming Model (1.5 Hours)

- Concept of tensors and creating it
- Manipulating the data type and shape of a tensor
- Applying mathematical operations to tensors
- Split, stack, and concatenate tensors

Module 2: Building input pipelines in PyTorch (1.5 Hours)

- PyTorch DataLoader
- Shuffle, batch, and repeat
- Creating and fetching datasets

Module 3: PyTorch's computation graphs (2 Hours)

- Understanding computation graphs and creating one.
- Storing and updating model parameters
- Computing gradients via automatic differentiation
- Different modules and custom layers in PyTorch
- Implementations of common architectures

Module 4: Concepts of Deep Learning and Artificial Intelligence (2 Hours)

- Introduction to Deep Learning
- Representation Learning
- Feature Engineering Vs. Automated Learning
- Deep Learning Applications and Frameworks
- Deep Learning Challenges
- Hands-On

Module 5: Training Deep Learning & Hyper-parameter Tuning (8 Hours)

- Multi-Layer Perceptron
- Gradient Descent and Back-Propagation
- Activation Functions at Hidden and Output Layer
- Hyper-parameters Tuning: Number of Hidden Layers, Number of Neurons per Hidden Layer, Learning Rate, Batch Size, and Other Hyperparameters
- The Vanishing/Exploding Gradients Problems
- Reusing Pretrained Layers
- Faster Optimizers
- Learning Rate Scheduling
- Avoiding Overfitting Through Regularization – L1, L2, Dropout, Batch Normalization
- Unsupervised Learning using AutoEncoders and Transfer Learning
- Applications and Hands-On

Module 6: Deep Convolutional Neural Network (8 Hours)

- Convolutions - Filters, Feature Maps, Padding, Striding
- Convolution over multiple channels
- Convolution Neural Network - The building blocks of CNNs
- Subsampling layers (Pooling Layers)
- Activation Functions
- Flattening and Fully Connected Layer
- Receptive Field
- Significance of multiple layers
- MLP Vs. CNN -- Issue with FC layer and CNN with only convolution layers
- 1 * 1 Convolution
- Fully Convolution Layers and Global Average Pooling
- CNN Architectures – VGG, Inception, ResNet, XCeption
- Applications and Hands-On

Module 7: Processing and Modelling Sequential Data using Convolutional Neural Network, Recurrent Neural Networks and Transformers (8 Hours)

- Natural Language Processing
- Pipelines for Text Data – Tokenization (Word level, character level and sub-word level -- Byte Pair Encoding, WordPiece algorithms).
- Types of encodings and its shortcomings
- Word Embeddings (Static): Word2Vec/Glove
- Recurrent Neurons, Layers and Recurrent Neural Networks (RNNs)
- Training and Types of RNNs
- RNN Architectures – Long Short Term Memory (LSTM)/ Gated Recurrent Units (GRU)
- Stacked RNNs, Bidirectional RNNs, Stateful RNNs
- Encoder-Decoder Network
- Teacher Forcing
- Beam Search – Avoiding Greedy Approach
- Model Evaluation – BLEU score
- Attention Mechanism – Bahdanau and Luong Attention
- Transformer Models

- Autoencoding and Auto-regressive Family Models
- Applications and Hands-On using HuggingFace library.