Experiment No: 01

Aim: Study of Deep learning Packages: Tensorflow, Keras, Theano and PyTorch. Document the distinct features and functionality of the packages.

Problem Statement: Study of following Deep learning Packages:

i. Keras

ii. PyTorch

iii. TensorFlow

iv. Theano

Theory:

A) Keras

Keras is an effective high-level neural network Application Programming Interface (API) written in Python. This open-source neural network library is designed to provide fast experimentation with deep neural networks, and it can run on top of CNTK, TensorFlow, and Theano. Keras focuses on being modular, user friendly, and extensible. Following are the distinct features of Keras:

1. Keras is an API that was made to be easy to learn. It offers consistent & simple APIs, reduces the actions required to implement common code, and explains user error clearly.
2. Prototyping time in Keras is less. This means that your ideas can be implemented and deployed in a shorter time. Keras also provides a variety of deployment options depending on user needs.
3. Languages with a high level of abstraction and inbuilt features are slow and building custom features can be hard. But Keras runs on top of TensorFlow and is relatively fast. Keras is also deeply integrated with TensorFlow, so you can create customized workflows with ease.

4. Keras is used commercially by many companies like Netflix, Uber, Square, Yelp, etc which have deployed products in the public domain which are built using Keras.

Apart from this, Keras has features such as:

1. It runs smoothly on both CPU and GPU.
2. It supports almost all neural network models.
3. It is modular in nature, which makes it expressive, flexible, and apt for innovative research.

Steps to build model in Keras:

1. Define a network: In this step, you define the different layers in our model and the connections between them. Keras has two main types of models: Sequential and Functional models. You choose which type of model you want and then define the dataflow between them.
2. Fit the network: Using this, we fit our model to our data after compiling. This is used to train the model on our data.
3. Evaluate the network: After fitting our model, we need to evaluate the error in our model.
4. Make Predictions: We use model.predict() to make predictions using our model on new data.

B) PyTorch

PyTorch is an optimized Deep Learning tensor library based on Python and Torch and is mainly used for applications using GPUs and CPUs. PyTorch is favored over other Deep Learning Frameworks like TensorFlow and Keras since it uses dynamic computation graphs and is completely Pythonic. It allows scientists, developers, and neural network debuggers to run and test portions of the code in real-time. Thus users don't have to wait for the entire code to be implemented to check if a part of the code works or not.

The two main features of PyTorch are:

1. Tensor Computation (similar to NumPy) with strong GPU support
2. Automatic Differentiation for creating and training deep neural networks

In machine learning, when we represent data, we need to do that numerically. A tensor is simply a container that can hold data in multiple dimensions. It can be a number, vector, matrix, or multi-dimensional array like Numpy arrays. Tensors can also be handled by the CPU or GPU to make operations faster. There are various types of tensors like Float Tensor, Double Tensor, Half Tensor, Int Tensor, and Long Tensor, but PyTorch uses the 32-bit Float Tensor as the default type.

 Mathematical Operations

The codes to perform mathematical operations are the same in PyTorch as in Numpy. Users need to initialize two tensors and then perform operations like addition, subtraction, multiplication, and division on them.

 Autograd

The autograd module is PyTorch's automatic differentiation engine that helps to compute the gradients in the forward pass in quick time. Autograd generates a directed acyclic graph where the leaves are the input tensors while the roots are the output tensors.

 Optim

The Optim module is a package with pre-written algorithms for optimizers that can be used to build neural networks.

C) TensorFlow

TensorFlow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning. TensorFlow was originally developed for large numerical computations without keeping deep learning in mind. However, it proved to be very useful for deep learning development as well, and therefore Google open-sourced it. TensorFlow works on the basis of data flow graphs that have nodes and edges. As the execution mechanism is in the form of graphs, it is much easier to execute TensorFlow code in a distributed manner across a cluster of computers while using GPUs.

Deep learning applications are very complicated, with the training process requiring a lot of computation. It takes a long time because of the large data size, and it involves several iterative processes, mathematical calculations, matrix multiplications, and so on. If you perform these activities on a normal Central Processing Unit (CPU), typically it would take much longer. Graphical Processing Units (GPUs) are popular in the context of games, where you need the screen and image to be of high resolution. GPUs were originally designed for this purpose. However, they are being used for developing deep learning applications as well. One of the major advantages of TensorFlow is that it supports GPUs, as well as CPUs.

The graph consists of nodes that represent a mathematical operation. A connection or edge between nodes is a multidimensional data array. It takes inputs as a multi-dimensional array where you can construct a flowchart of operations that can be performed on these inputs.

Tensorflow architecture works in three significant steps:

1. Data pre-processing - structure the data and brings it under one limiting value
2. Building the model - build the model for the data
3. Training and estimating the model - use the data to train the model and test it with unknown data Introduction to Components of TensorFlow

Tensor: Tensor forms the core framework of TensorFlow. All the computations in TensorFlow involve tensors. It is a matrix of n-dimensions that represents multiple types of data. A tensor can be the result of a computation or it can originate from the input data.

Graphs: Graphs describe all the operations that take place during the training. Each operation is called an op node and is connected to the other. The graph shows the op nodes and the connections between the nodes, but it does not display values.

D) Theano

Theano is a Python library for fast numerical computation that can be run on the CPU or GPU. It is a key foundational library for Deep Learning in Python that you can use directly to create Deep Learning models or wrapper libraries that greatly simplify the process. Theano is a Python library that allows us to evaluate mathematical operations including multi-dimensional arrays so efficiently. It is mostly used in building Deep Learning Projects. It works a way faster

on Graphics Processing Unit (GPU) rather than on CPU. It can take advantage of GPUs which makes it perform better than C on a CPU by considerable orders of magnitude under some certain circumstances. It knows how to take structures and convert them into very efficient code that uses numpy and some native libraries. It is mainly designed to handle the types of computation required for large neural network algorithms used in Deep Learning. That is why, it is a very popular library in the field of Deep Learning.

How to install Theano:

pip install Theano

Several of the symbols we will need to use are in the tensor subpackage of Theano. We often import such packages with a handy name, let's say, T. Theano is a sort of hybrid between numpy and sympy, an attempt is made to combine the two into one powerful library. Some advantages of theano are as follows:

1. Stability Optimization: Theano can find out some unstable expressions and can use more stable means to evaluate them
2. Execution Speed Optimization: As mentioned earlier, theano can make use of recent GPUs and execute parts of expressions in your CPU or GPU, making it much faster than Python
3. Symbolic Differentiation: Theano is smart enough to automatically create symbolic graphs for computing gradients. It supports convolutional networks and recurrent networks, as well as
4. combinations of the two

Conclusion:

Thus we have studied the distinct features of Keras, TensorFlow, PyTorch and Theano python libraries used for Deep Learning Implementation.