**Experiment No : 01**

**TensorFlow**

**Introduction**

TensorFlow is an open-source software library. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google’s Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well! Let us first try to understand what the word TensorFlow actually mean! TensorFlow is basically a software library for numerical computation using data flow graphs

where:

* **nodes** in the graph represent mathematical operations.
* **edges** in the graph represent the multidimensional data arrays (called tensors) communicated between them. (Please note that tensor is the central unit of data in TensorFlow).

TensorFlow provides multiple APIs (Application Programming Interfaces). These can be classified into 2 major categories:

1.Low level API:

* complete programming control
* recommended for machine learning researchers
* provides fine levels of control over the models
* TensorFlow Core is the low level API of TensorFlow.

2.High level API:

* built on top of TensorFlow Core
* easier to learn and use than TensorFlow Core
* make repetitive tasks easier and more consistent between different users
* tf.contrib.learn is an example of a high level API.

**Installing TensorFlow**

An easy to follow guide for TensorFlow installation is available here: Installing TensorFlow. Once installed, you can ensure a successful installation by running this command in python interpreter:

import tensorflow as tf

**Tensors**

TensorFlow operates on multidimensional arrays or tensors represented as **tf.Tensor** objects. Here is a two-dimensional tensor:

*import tensorflow as tf*

*x = tf.constant([[1., 2., 3.],*

*[4., 5., 6.]])*

*print(x)*

*print(x.shape)*

*print(x.dtype)*

The most important attributes of a **tf.Tensor** are its **shape** and **dtype**:

* **Tensor.shape**: tells you the size of the tensor along each of its axes.
* **Tensor.dtype**: tells you the type of all the elements in the tensor.

**Methods**

**Tensorflow log() method**

The module tensorflow.math provides support for many basic mathematical operations. Function tf.log() [alias tf.math.log] provides support for the natural logarithmic function in Tensorflow. It expects the input in form of complex numbers as a+bi or floating point numbers. The input type is tensor and if the input contains more than one element, an element-wise logarithm is computed, y=loge x .

**Syntax**: tf.log(x, name=None) or tf.math.log(x, name=None)

**Parameters**:

**x**: A Tensor of type bfloat16, half, float32, float64, complex64 or complex128.

**name** (optional): The name for the operation.

**Return type**: A Tensor with the same size and type as that of x.

**Tensorflow abs() method**

The module tensorflow.math provides support for many basic mathematical operations. Function tf.abs() [alias tf.math.abs] provides support for the absolute function in Tensorflow. It expects the input in form of complex numbers as a+bi or floating point numbers. The input type is tensor and if the input contains more than one element, an element-wise absolute value is computed.

For a complex number a+bi , the absolute value is computed as \sqrt{a2+b2} .

For floating point numbers a , the absolute value is computed as a if a>=0, else -a.

Syntax: tf.abs(x, name=None) or tf.math.abs(x, name=None)

Parameters:

x: A Tensor or SparseTensor of type float16, float32, float64, int32, int64, complex64 or complex128.

name (optional): The name for the operation.

Return type: A Tensor or SparseTensor with the same size and type as that of x with absolute values. For complex64 or complex128 input, the returned Tensor will be of type float32 or float64, respectively.

**Tensorflow exp() method**

The module tensorflow.math provides support for many basic mathematical operations. Function tf.exp() [alias tf.math.exp] provides support for the exponential function in Tensorflow. It expects the input in form of complex numbers as a+bi or floating point numbers. The input type is tensor and if the input contains more than one element, an element-wise exponential value is computed, y=ex .

**Syntax**: tf.exp(x, name=None) or tf.math.exp(x, name=None)

**Parameters**:

**x**: A Tensor of type bfloat16, half, float32, float64, complex64 or complex128.

**name** (optional): The name for the operation.

**Return type**: A Tensor with the same size and type as that of x.

**Keras**

**Introduction**

Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the Google engineers, Francois Chollet. It is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination.

It cannot handle low-level computations, so it makes use of the **Backend** library to resolve it. The backend library act as a high-level API wrapper for the low-level API, which lets it run on TensorFlow, CNTK, or Theano.

Initially, it had over 4800 contributors during its launch, which now has gone up to 250,000 developers. It has a 2X growth ever since every year it has grown. Big companies like Microsoft, Google, NVIDIA, and Amazon have actively contributed to the development of Keras. It has an amazing industry interaction, and it is used in the development of popular firms likes Netflix, Uber, Google, Expedia, etc.

**What makes Keras special?**

* Focus on user experience has always been a major part of Keras.
* Large adoption in the industry.
* It is a multi backend and supports multi-platform, which helps all the encoders come together for coding.
* Research community present for Keras works amazingly with the production community.
* Easy to grasp all concepts.
* It supports fast prototyping.
* It seamlessly runs on CPU as well as GPU.
* It provides the freedom to design any architecture, which then later is utilized as an API for the project.
* It is really very simple to get started with.
* Easy production of models actually makes Keras special.

**Keras Backend**

Keras being a model-level library helps in developing deep learning models by offering high-level building blocks. All the low-level computations such as products of Tensor, convolutions, etc. are not handled by Keras itself, rather they depend on a specialized tensor manipulation library that is well optimized to serve as a backend engine. Keras has managed it so perfectly that instead of incorporating one single library of tensor and performing operations related to that particular library, it offers plugging of different backend engines into Keras.

Keras consist of three backend engines, which are as follows:

* **TensorFlow**

TensorFlow is a Google product, which is one of the most famous deep learning tools widely used in the research area of machine learning and deep neural network. It came into the market on 9th November 2015 under the Apache License 2.0. It is built in such a way that it can easily run on multiple CPUs and GPUs as well as on mobile operating systems. It consists of various wrappers in distinct languages such as Java, C++, or Python.

* **Theano**

Theano was developed at the University of Montreal, Quebec, Canada, by the MILA group. It is an open-source python library that is widely used for performing mathematical operations on multi-dimensional arrays by incorporating scipy and numpy. It utilizes GPUs for faster computation and efficiently computes the gradients by building symbolic graphs automatically. It has come out to be very suitable for unstable expressions, as it first observes them numerically and then computes them with more stable algorithms.

**CNTK**

Microsoft Cognitive Toolkit is deep learning's open-source framework. It consists of all the basic building blocks, which are required to form a neural network. The models are trained using C++ or Python, but it incorporates C# or Java to load the model for making predictions.

**Advantages of Keras**

* It is very easy to understand and incorporate the faster deployment of network models.
* It has huge community support in the market as most of the AI companies are keen on using it.
* It supports multi backend, which means you can use any one of them among TensorFlow, CNTK, and Theano with Keras as a backend according to your requirement.

**Disadvantages of Keras**

* The only disadvantage is that Keras has its own pre-configured layers, and if you want to create an abstract layer, it won't let you because it cannot handle low-level APIs. It only supports high-level API running on the top of the backend engine (TensorFlow, Theano, and CNTK).