TensorFlow Basics

The TensorFlow package is an API that currently supports function calls from two prominent programming languages: C++ and Python. The backend for the library has been written using highly optimized algorithms in C/C++ code. The language of choice for me is Python. Thus, the tutorial that follows has been written and explained using Python code.

TensorFlow computation model works as follows:

- It represents computations in the form of data-flow **Graphs**. Here, computations are specified by function calls with data flowing along directed edges in between the nodes.
- The Graph Nodes are called **Operations**. They represent all the methods or active parts of the Graph that are executed once the Graph evaluation occurs.
- A **Tensor**, as defined in the documentation, is a 'typed multi-dimensional array' [1]. It defines an n-dimensional list/vector data structure with associated meta-information pertaining to the data that it represents such as it's shape, size and rank (the dimensionality). Being 'typed' implies that each tensor has a data-type attribute. Tensors can be type-casted to other data-types as well and subjected to a great-many transformations [2]. Tensors can either be Python lists or numpy arrays.
    - ➢ E.g.1 :-
        - tensr1 = [1., 6., 7.]
        - tensr2 = [[1, 3], [4, 5]] # A 2x2 matrix
        - tensr3 = [1, 2, 3, 4]
        - tensr4 = [[1, 2], [3, 4], [5, 6]] # A 3x2 matrix
        - tensr5 = **[[** [1, 2], [5, 3] **] [** [4, 6], [8, 7] **]]** # A 2x2x2 matrix
    - ➢ E.g.2 :- For images, the corresponding Tensor representation has the following structure:- [**batch_size**, **height**, **width**, **channels**] -- The 'channel' parameter can assume the following values according to the Input type: {1: Grayscale, 2: RGB, 3: RGBA (RGB with Alpha channel(s))}
    - ➢ For a more in-depth explanation of Tensor attributes, you can refer to the support document specified at [3].
- Each operation node takes 0 or more Tensors as input and produces 0 or more Tensors as output.

- A Session is an encapsulation of the environment in which the Operations are specified and the Graph is executed. Thus the Session class has to be instantiated for the Graph evaluation to occur. The requisite graph must be specified as an argument to the constructor. If not specified, it assumes the default graph as its argument. A sample instantiation is as follows:

    - **Curr_session = tensorflow.Session()**

- It's a good practise to close a Session once it has exhausted it's operation, in order to release all the resources that it held (such as Reader and Queue objects).

    - **Curr_session.close()**

- Much more like Apache Spark, the trigger or the execution of the operations in the Graph is only brought about by the call to special methods, i.e. the Session.run() and Tensor.eval().

- Every TensorFlow program can be structured into two major phases:

    - **The Construction Phase** - Here, the Graph is assembled, it's operation nodes specified.

    - **The Execution Phase** - A Session is initiated to execute the Operations in the Graph.

References:

[1] TensorFlow - Basic Usage
[https://www.tensorflow.org/versions/r0.7/get_started/basic_usage.html#the-computation-graph]

[2] Tensor Transformations:
[https://www.tensorflow.org/versions/r0.7/api_docs/python/array_ops.html]

[3] Tensor Ranks, Shapes, and Types
[https://www.tensorflow.org/versions/master/resources/dims_types.html#rank]