Reading Assignment 1
Authors: Sri Datta, Nik Vaessen

1 Motivation

We decided to read *TensorFlow: A system for large-scale machine learning*. The authors present a new computational system for machine learning models. This system, called *TensorFlow*, aims to alleviate the drawbacks of *DistBelief*, an (at the time) widely-used alternative system (at Google) for machine learning model development.

It also provides the machine learning community with a system that allows for experimentation, training models on large datasets and using those models in a variety of production environments.

2 Contributions

The main contribution of the paper is to present *TensorFlow*, which was made open-source so it can be used by the whole machine learning community.

Tensorflow improves on *DistBelief* by alleviating three major faults:

1. The Neural network layers are written in C++, which makes it more difficult for researchers to implement and experiment with novel layer architectures.
2. The parameter-server model makes it difficult and error-prone to use an alternative optimization method from stochastic gradient descent.
3. The execution pattern (directed acyclic graph) doesn't work for algorithms such as recurrent neural networks, adversarial networks, and reinforcement learning.

Moreover, *TensorFlow* was designed to run in a wide variety of computational environments, such as mobile devices, GPU workstations, custom made processors such as TPU's, and huge server farms. This was not true for *DistBelief*, which was designed only to run on huge server farms.

3 Solution

The important design principles behind TensorFlow are the following:

1. Dataflow graphs of primitive operators

Replacing complex layer nodes with simpler operators like matrix multiplication and convolutions made the system more modular. A higher level scripting interface in Python is introduced to give users the freedom and power to design custom layers and functionalities easily.

2. Deferred execution

Though TensorFlow provides a higher level Python interface as mentioned earlier, the execution sequence is very different from general programming languages. TensorFlow program first requires all the initialization of parameters to be done which then will be used to create a symbolic graph of the program. This graph is used to optimize the whole program and allocate GPU resources without having to wait for the intermediate computations to be performed.

3. Common abstraction for heterogeneous accelerators

TensorFlow uses tensors of primitive values as common interchange format that would be understandable across different devices. As a result of such common abstractions, the same TensorFlow program can easily be tuned to run across GPUs, TPUs or mobile devices.

4 Strong Points

1. A very strong point is that the *Tensorflow* system, unlike *DistBelief*, has been made open-source. This allows for external reproducibility tests, as well as in general improving the software infrastructure available to the machine learning community.
2. The mutable state in the dataflow graphs allows for very novel architectures to be implemented, which can drastically increase the use-cases and effectiveness of machine learning as new algorithms can be efficiently implemented and put in production.
3. Having an abstraction layer over the computation graph allows for easy implementation in a high-level programming language like Python. This is a lot better than having to write a research code in C++.

5 Weak Points

1. In the single machine benchmark, Tensorflow only performs better on *AlexNet*: In the bigger networks like *GoogLeNet,* another system called *Neon* significantly outperforms Tensorflow.
2. The dataflow graph is static: once initialized in the first phase, it cannot be modified after. Some algorithms, like deep reinforcement learning, rely on dynamic computation graphs.
3. *TensorFlow* is only weakly consistent. If a new deep learning algorithm requires strong consistency during all phases of training, it cannot be implemented using *TensorFlow*.