Reading Assignment 1

Authors:
Daniele Montesi, Francesco Staccone

Paper title:
TensorFlow: A System for Large-Scale Machine Learning

1 Motivation

Google employees have worked with Machine Learning since many years, when they had chance to discover which are the most important requirements for a software platform to run their algorithms. Google's first-generation system, DistBelief, has proved to be well performing on simple tasks, but very unmanageable on some difficult ones.
With this paper, the authors explain how to solve the problems encountered by Data Scientists with the older Google's distributed system, DistBelief. The main concerns the authors are aiming to solve are related to **scalability** issues, which are crucial especially when talking about deep neural networks.

2 Contributions

The older Google's distributed system for training neural networks was limited in several respects:

- the Python-based scripting interface for composing pre-defined layers didn't provide adequate **flexibility**, especially for the most complex tasks;
- its **fixed execution pattern** (read batch data, compute loss function, calculate gradient, write on parameter server) worked well only with simple feed-forward neural networks;
- its **single platform**, a large distributed cluster of multi-core servers, made impossible to *scale down*. This is crucial for users willing to hone their algorithm on smaller data, before starting the train on the whole dataset.

3 Solution

Tensorflow was designed to be way more flexible. It introduces many improvements aiming to solve the aforementioned issues:

- Tensorflow has no parameter server. Instead, their functions are deployed as a set of *tasks* some of which are called Parameter Server (PS) tasks, and they can run arbitrary dataflow graphs directly programmed by the users. This enables way more **flexibility** than a conventional parameter server;
- Instead of defining complex layers, Tensorflow defines **single mathematical operations** that can be easily combined in order to create novel layers;
- Tensorflow's simple **dataflow-based abstraction** allows users to scale down, deploying the application on local workstations/devices.

The reason for **flexibility** in Tensorflow resides in the adoption of a different model than the batch dataflow model. In this way, the users can mutate individual vertices of the graph while being used spreading the effects on the other nodes. In detail, TensorFlow allows *vertices* to represent computations that own or update mutable state, while *edges* carry **tensors** between nodes, that are nothing but multi-dimensional arrays of primitive values used as a common interchange format that all devices understand.
Flexibility, then, is a crucial functionality when training large models because allows to perform in-place updates that applies very quickly to the whole training process.

4 Strong Points

It is possible to select several points of strength among the several insights offered by the authors of this paper. We have chosen some of them:

- S1. The first strong point of the paper regards the contributions that the provided solution offers: in summary, a **flexible** system capable of good performances in fulfilling machine learning computational tasks, in **handling a wide array of architectures** and in **abstracting** many low-level programming details;
- S2. The distinctive advantage of Tensorflow is that it supports **large scale dataset training and inference** which can hold hundreds of GPUs training together. It also supports **multiple platforms** such as distributed clusters and mobile devices. Moreover, it provides an abstraction that **works well for novice users** to train their first neural network, while also providing the knobs needed for machine learning **researchers** to try new things, supporting experiments such as studying new machine learning models or system level optimizations;
- S3. In the evaluation section, it is properly and clearly shown that TensorFlow achieves **similar or even superior performances** compared to widely used deep learning frameworks (Torch, Caffe) in tasks such as training neural network image classification architectures such as AlexNet and GoogleNet.

5 Weak Points

The downsides of this paper seem to be minor. Among the possible drawbacks present, we selected:

- W1. The **static dataflow graph** used by TensorFlow offers great performance but the paper mentions also its limitations, especially for deep reinforcement learning algorithms; this has recently become a big problem since more and more users have begun using PyTorch to tackle this issue, also thanks to its user-friendlier **interface**. TensorFlow's increasing flexibility and increasing options involve making the learning curve for using a system steeper, thus it is hard for users to master it;
- W2. Another weakness of this paper is that while TensorFlow does very well in terms of flexibility and abstraction, it is not necessarily the best choice when it comes to **raw performance**. In this respect, it seems that Tensorflow is overkill for personal, smaller machine learning projects that run on a single computer with limited computational resources (GPUs). It can perform well under a huge cluster, instead;
- W3. In the benchmarks related to the **convolutional models training times**, authors show that Neon library manages to outperform TensorFlow in nearly all architectures compared (3 out of 4). The authors think this is mainly due to Neon's use of hand-optimized convolutional kernels implemented in assembly language; they also say that in the future a same approach could be followed for TensorFlow. We believe that they could have gone further into this problem in the paper.