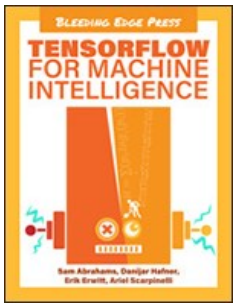


TensorFlow for Machine Intelligence: A Hands-On Introduction to Learning Algorithms

by Sam Abrahams, Danijar Hafner, Erik Erwitt and Ariel Scarpinelli

Chapter 1: Introduction

Data is everywhere

We are truly in "The Information Age." These days, data flows in from everywhere: smartphones, watches, vehicles, parking meters, household appliances- almost any piece of technology you can name is now being built to communicate back to a database somewhere in the cloud. With access to seemingly unlimited storage capacity, developers have opted for a "more-is-better" approach to data warehousing, housing petabytes of data gleaned from their products and customers.

At the same time, computational capabilities continue to climb. While the growth of CPU speeds has slowed, there has been an explosion of parallel processing architectures. Graphics processing units (GPUs), once used primarily for computer games are now being used for general purpose computing, and they have opened the floodgates for the rise of machine learning.

Machine learning, sometimes abbreviated to "ML," uses general-purpose mathematical models to answer specific questions using data. Machine learning has been used to detect spam email, recommend products to customers, and predict the value of commodities for many years. In recent years, a particular kind of machine learning has seen an incredible amount of success across all fields: deep learning.

Deep learning

"Deep learning" has become the term used to describe the process of using multi-layer neural networks- incredibly flexible models that can use a huge variety and combination of different mathematical techniques. They are incredibly powerful, but our ability to utilize neural networks to great effect has been a relatively new phenomena, as we have only recently hit the critical mass of data availability and computational power necessary to boost their capabilities beyond those of other ML techniques.

The power of deep learning is that it gives the model more flexibility in deciding how to use data to best effect. Instead of a person having to make wild guesses as to which inputs are worth including, a properly tuned deep learning model can take all parameters and automatically determine useful, higher-order combinations of its input values. This enables a much more sophisticated decision-making process, making computers more intelligent than ever. With deep learning, we are capable of creating cars that drive themselves and phones that understand our speech. Machine translation, facial recognition, predictive analytics, machine music composition, and countless artificial intelligence tasks have become possible or significantly improved due to deep learning.

While the mathematical concepts behind deep learning have been around for decades, programming libraries dedicated to creating and training these deep models have only been available in recent years. Unfortunately, most of these libraries have a large trade-off between flexibility and production-worthiness. Flexible libraries are invaluable for researching novel model architectures, but are often either too slow or incapable of being used in production. On the other hand, fast, efficient, libraries which can be hosted on distributed hardware are available, but they often specialize in specific types of neural networks and aren't suited to researching new and better models. This leaves decision makers with a dilemma: should we attempt to do research with inflexible libraries so that we don't have to reimplement code, or should we use one library for research and a completely different library for production? If we choose the former, we may be unable to test out different types of neural network models; if we choose the latter, we have to maintain code that may have completely different APIs. Do we even have the resources for this?

TensorFlow aims to solve this dilemma.

TensorFlow: a modern machine learning library

TensorFlow, open sourced to the public by Google in November 2015, is the result of years of lessons learned from creating and using its predecessor, DistBelief. It was made to be flexible, efficient, extensible, and portable. Computers of any shape and size can run it, from smartphones all the way up to huge computing clusters. It comes with lightweight software that can instantly productionize your trained model, effectively eliminating the need to reimplement models. TensorFlow embraces the innovation and community-engagement of open source, but has the support, guidance, and stability of a large corporation. Because of its multitude of strengths, TensorFlow is appropriate for individuals and businesses ranging from startups to companies as large as, well, Google.

If you and your colleagues have data, a question to answer, and a computer that turns on, you're in luck- as TensorFlow could be the missing piece you've been looking for.

TensorFlow: a technical overview

This section aims to provide high level information about the TensorFlow library, such as what it is, its development history, use cases, and how it stacks up against competitors. Decision makers, stakeholders, and anyone who wants to understand the background of TensorFlow will benefit from reading this section.

A brief history of deep learning at Google

Google's original large-scale deep learning tool was DistBelief, a product of the Google Brain team. Since its creation, it has been used by dozens of teams for countless projects involving deep neural networks. However, as with many first-of-its-kind engineering projects, there were design mistakes that have limited the usability and flexibility of DistBelief. Sometime after the creation of DistBelief, Google began working on its successor, whose design would apply lessons learned from the usage and limitations of the original DistBelief. This project became TensorFlow, which was released to the public in November 2015. It quickly turned into a popular library for machine learning, and it is currently

being used for natural language processing, artificial intelligence, computer vision, and predictive analytics.

What is TensorFlow?

Let's take a high-level view of TensorFlow to get an understanding what problems it is trying to solve.

Breaking down the one-sentence description

Looking at the **TensorFlow website**, the very first words greeting visitors is the following (rather vague) proclamation:

> TensorFlow is an open source software library for machine intelligence.

Just below, in the first paragraph under "About TensorFlow," we are given an alternative description:

> TensorFlow™ is an open source software library for numerical computation using data flow graphs.

This second definition is a bit more specific, but may not be the most comprehensible explanation for those with less mathematical or technical backgrounds. Let's break it down into chunks and figure out what each piece means.

OPEN SOURCE:

TensorFlow was originally created by Google as an internal machine learning tool, but an implementation of it was open sourced under the **Apache 2.0 License** in November 2015. As open source software, anyone is allowed to download, modify, and use the code. Open source engineers can make additions/improvements to the code and propose their changes to be included in a future release. Due to the popularity TensorFlow has gained, there are improvements being made to the library on a daily basis- created by both Google and third-party developers.

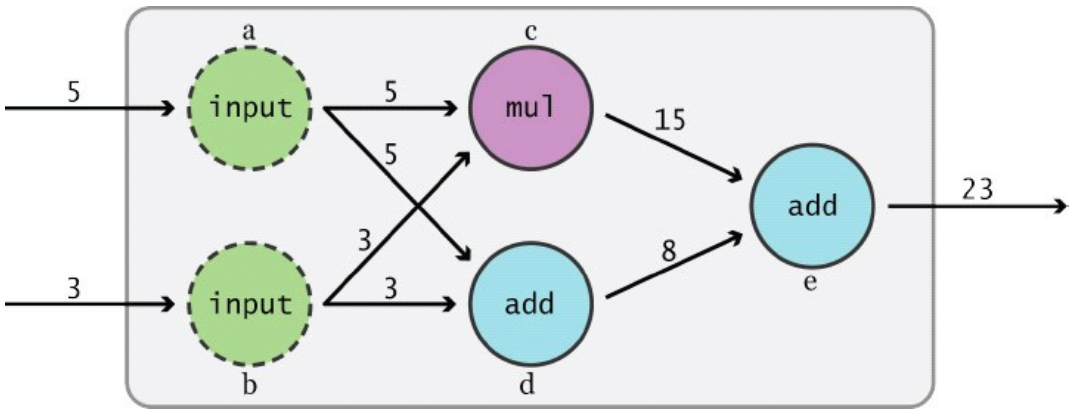
> Notice that we say "an implementation" and not "TensorFlow" was open sourced. Technically speaking, TensorFlow is an interface for numerical computation as described in the **TensorFlow white paper**, and Google still maintains its own internal implementation of it. However, the differences between the open source implementation and Google's internal implementation are due to connections to other internal software, and **not** Google "hoarding the good stuff". Google is constantly pushing internal improvements to the public repository, and for all intents and purposes the open source release contains the same capabilities as Google's internal version.
>
> For the rest of this book, when we say "TensorFlow", we are referring to the open source implementation.

LIBRARY FOR NUMERICAL COMPUTATION

Instead of calling itself a "library for machine learning", it uses the broader term "numerical computation." While TensorFlow does contain a package, **"learn" (AKA "Scikit Flow")**, that emulates the one-line modeling functionality of **Scikit-Learn**, it's important to note that TensorFlow's primary purpose is *not* to provide out-of-the-box machine learning solutions. Instead, TensorFlow provides an extensive suite of functions and classes that allow users to define models from scratch mathematically. This allows users with the appropriate technical background to create customized, flexible models quickly and intuitively. Additionally, while TensorFlow does have extensive support for ML-specific functionality, it is just as well suited to performing complex mathematical computations. However, since this book is focused on machine learning (and deep learning in particular), we will usually talk about TensorFlow being used to create machine learning models.

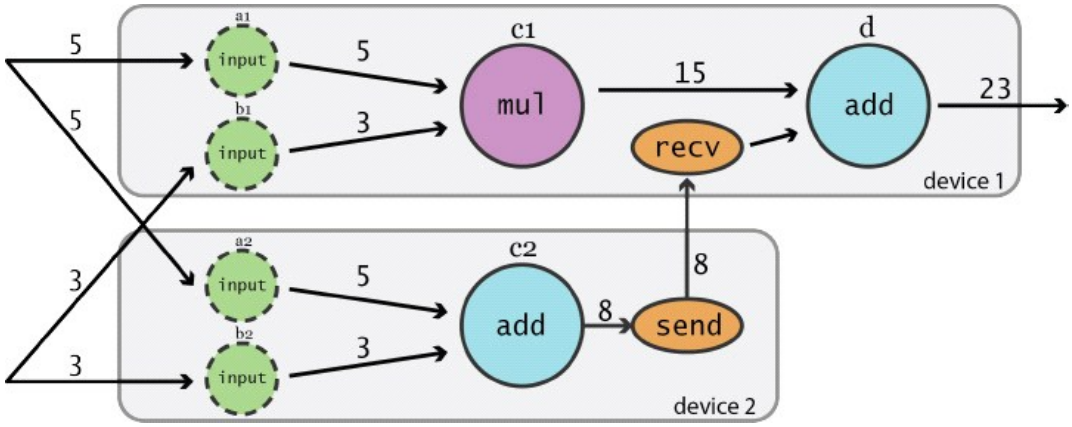
DATA FLOW GRAPHS

The computational model for TensorFlow is a **directed graph**, where the *nodes* (typically represented by circles or boxes) are functions/computations, and the *edges* (typically represented by arrows or lines) are numbers, matrices, or tensors.



There are a number of reasons this is useful. First, many common machine learning models, such as **neural networks**, are commonly taught

and visualized as directed graphs already, which makes their implementation more natural for machine learning practitioners. Second, by splitting up computation into small, easily differentiable pieces, TensorFlow is able to automatically compute the derivative of any node (or "Operation", as they're called in TensorFlow) with respect to any other node that can affect the first node's output. Being able to compute the derivative/gradient of nodes, especially output nodes, is crucial for setting up machine learning models. Finally, by having the computation separated, it makes it much easier to distribute work across multiple CPUs, GPUs, and other computational devices. Simply split the whole, larger graph into several smaller graphs and give each device a separate part of the graph to work on (with a touch of logic to coordinate sharing information across devices)



Quick Aside: What IS a Tensor?

A tensor, put simply, is an n-dimensional matrix. So a 2-dimensional tensor is the same as a standard matrix. Visually, if we view an ***m* × *m*** matrix as a square array of numbers (`m` numbers tall, and `m` numbers wide), we can view an ***m* × *m* × *m*** tensor as a cube array of numbers (`m` numbers tall, `m` numbers wide, and `m` numbers deep). In general, you can think about tensors the same way you would matrices, if you are more comfortable with matrix math!

Beyond the one-sentence description

The phrase "open source software library for numerical computation using data flow graphs" is an impressive feat of information density, but it misses several important aspects of TensorFlow that make it stand out as a machine learning library. Here are a few more components that also help make TensorFlow what it is:

DISTRIBUTED

As alluded to when described data flow graphs above, TensorFlow is designed to be scalable across multiple computers, as well as multiple CPUs and GPUs within single machines. Although the original open source implementation did *not* have distributed capabilities upon release, as of version 0.8.0 the distributed runtime is available as part of the TensorFlow built-in library. While this initial distributed API is a bit cumbersome, it is incredibly powerful. Most other machine learning libraries do not have such capabilities, and it's important to note that native compatibility with certain cluster managers (such as **Kubernetes**) are being worked on.

A SUITE OF SOFTWARE

While "TensorFlow" is primarily used to refer to the API used to build and train machine learning models, TensorFlow is really a bundle of software designed to be used in tandem with:

- **TensorFlow** is the API for defining machine learning models, training them with data, and exporting them for further use. The primary API is accessed through Python, while the actual computation is written in C++. This enables data scientists and engineers to utilize a more user-friendly environment in Python, while the actual computation is done with fast, compiled C++ code. There is a C++ API for executing TensorFlow models, but it is limited at this time and not recommended for most users.
- **TensorBoard** is graph visualization software that is included with any standard TensorFlow installation. When a user includes certain TensorBoard-specific operations in TensorFlow, TensorBoard is able to read the files exported by a TensorFlow graph and can give insight into a model's behavior. It's useful for summary statistics, analyzing training, and debugging your TensorFlow code. Learning to use TensorBoard early and often will make working with TensorFlow that much more enjoyable and productive.
- **TensorFlow Serving** is software that facilitates easy deployment of pre-trained TensorFlow models. Using built-in TensorFlow functions, a user can export their model to a file which can then be read natively by TensorFlow Serving. It is then able to start a simple, high-performance server that can take input data, pass it to the trained model, and return the output from the model. Additionally, TensorFlow Serving is capable of seamlessly switching out old models with new ones, without any downtime for end-users. While Serving is possibly the least recognized portion of the TensorFlow ecosystem, it may be what sets TensorFlow apart from its competition. Incorporating Serving into a production environment enables users to avoid reimplementing their model, who can instead just pass along their TensorFlow export. TensorFlow Serving is written entirely in C++, and its API is only accessible through C++.

We believe that using TensorFlow to its fullest means knowing how to use all of the above in conjunction with one another. Hence, we will be covering all three pieces of software in this book.

When to use TensorFlow

Let's take a look at some use cases for TensorFlow. In general, TensorFlow is generally a good choice for most machine learning purposes. Below is a short list of uses that TensorFlow is specifically targeted at.

Researching, developing, and iterating through new machine learning architectures. Because TensorFlow is incredibly flexible, it's useful when creating novel, less-tested models. With some libraries, you are given rigid, pre-built models that are good for prototyping, but are incapable of being modified.

Taking models directly from training to deployment. As described earlier, TensorFlow Serving enables you to quickly move from training to deployment. As such, it allows for a much faster iteration when creating a product that depends on having a model up-and-running. If your team needs to move fast, or if you simply don't have the resources to reimplement a model in C++, Java, etc., TensorFlow can give your team the ability to get your product off the ground.

Implementing existing complex architectures. Once you learn how to look at a visualization of a graph and build it in TensorFlow, you will be able to take models from recent research literature and implement them in TensorFlow. Doing so can provide insight when building future models or even give a strict improvement over the user's current model.

Large-scale distributed models. TensorFlow is incredibly good at scaling up over many devices, and it's already begun to replace DistBelief for various projects within Google. With the recent release of the distributed runtime, we'll see more and more cases of TensorFlow being run on multiple hardware servers as well as many virtual machines in the cloud.

Create and train models for mobile/embedded systems. While much of the attention on TensorFlow has been about its ability to scale up, it is also more than capable of scaling *down*. The flexibility of TensorFlow extends to systems with less computation power, and can be run on Android devices as well as mini-computers such as the **Raspberry Pi**. The TensorFlow repository includes **an example on running a pre-trained model on Android**.

TensorFlow's strengths

Usability

- The TensorFlow workflow is relatively easy to wrap your head around, and it's consistent API means that you don't need to learn an entire new way to work when you try out different models.
- TensorFlow's API is stable, and the maintainers fight to ensure that every incorporated change is backwards-compatible.
- TensorFlow integrates seamlessly with Numpy, which will make most Python-savvy data scientists feel right at home.
- Unlike some other libraries, TensorFlow does not have any compile time. This allows you to iterate more quickly over ideas without sitting around.
- There are multiple higher-level interfaces built on top of TensorFlow already, such as **Keras** and **SkFlow**. This makes it possible to use the benefits of TensorFlow even if a user doesn't want to implement the entire model by hand.

Flexibility

- TensorFlow is capable of running on machines of all shapes and sizes. This allows it to be useful from supercomputers all the way down to embedded systems- and everything in between.
- It's distributed architecture allows it to train models with massive datasets in a reasonable amount of time.
- TensorFlow can utilize CPUs, GPUs, or both at the same time.

Efficiency

- When TensorFlow was first released, it was surprisingly slow on a number of popular machine learning benchmarks. Since that time, the development team has devoted a ton of time and effort into improving the implementation of much of TensorFlow's code. The result is that TensorFlow now boasts impressive times for much of it's library, vying for the top spot amongst the open-source machine learning frameworks.
- TensorFlow's efficiency is still improving as more and more developers work towards better implementations.

Support

- TensorFlow is backed by Google. Google is throwing a *ton* of resources into TensorFlow, since it wants TensorFlow to be the *lingua franca* of machine learning researchers and developers. Additionally, Google uses TensorFlow in its own work daily, and is invested in the continued support of TensorFlow.
- An incredible community has developed around TensorFlow, and it's relatively easy to get responses from informed members of the the

community or developers on GitHub.

- Google has released several **pre-trained machine learning models in TensorFlow**. They are free to use and can allow prototypes to get off the ground without needing massive data pipelines.

Extra features

- TensorBoard is invaluable when debugging and visualizing your model, and there is nothing quite like it available in other machine learning libraries.
- TensorFlow Serving may be the piece of software that allows more startup companies to devote services and resources to machine learning, as the cost of reimplementing code in order to deploy a model is no joke.

Challenges when using TensorFlow

Distributed support is still maturing

Although it is officially released, using the distributed features in TensorFlow is not as smooth as you might expect. As of this writing, it requires manually defining the role of each device, which is tedious and error-prone. Because it is brand new, there are less examples to learn from, although this should improve in the future. As mentioned earlier, support for Kubernetes is currently in the development pipeline, but for now it's still a work in progress.

Implementing custom code is tricky

There is an **official how-to on creating your own operations in TensorFlow**, but there is a fair amount of overhead involved when implementing customized code into TensorFlow. If, however, you are hoping to contribute it to the master repository, the Google development team is quick to help answer questions and look over your code to bring your work into the fold.

Certain features are still missing

If you are an experience machine learning professional with a ton of knowledge about a different framework, it's likely that there is going to be a small but useful feature you like that hasn't been implemented in TensorFlow yet. Often, there is a way to get around it, but that won't stop you from saying "Why isn't this natively supported yet?"

Onwards and upwards!

Needless to say, we are incredibly excited about the future of TensorFlow, and we are thrilled to give you a running start with such a powerful tool. In the next chapter, you'll install TensorFlow and learn all about the core TensorFlow library, basic use patterns, and environment.