Lecture note 1: Introduction to TensorFlow

"CS 20SI: TensorFlow for Deep Learning Research" (cs20si.stanford.edu)
Prepared by Chip Huyen (huyenn@stanford.edu)
Reviewed by Danijar Hafner, Jon Gautier, Minh-Thang Luong, Paul Warren

The guys who wrote the book "TensorFlow for Machine Intelligence" did a wonderful justification for the existence of Tensorflow, and this is what I'm going to quote:

"*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 di erent APIs. Do we even have the resources for this?

TensorFlow aims to solve this dilemma."

So what's Tensorflow?

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. To best understand what it is, let's look at what its creators said about it.

On the TensorFlow's website, we saw:

Tagline:

An open-source software library for Machine Intelligence

Definition:

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

We can see that TensorFlow is open-source. However, note that only the implementation of TensorFlow that we see on GitHub is open-source. Google maintains its own internal version.

It's said that Google did it because of the complicated relationships TensorFlow has with its other internal tools, and not because Google is "hoarding good stuff". Let's hope for the best.

The next key phrase we see is that TensorFlow is a "software library for Machine Intelligence". It's one of more than a dozen of machine intelligence libraries developed by big companies, and is probably one of the newest ones. For the list of current deep learning libraries, please visit this link.

Why TensorFlow

Given the plethora of these libraries, Why did we choose Tensorflow to teach in this class?

Our first criterion was that the library should be somewhat popular. Among those listed, the four most popular libraries, according to our observation among our peers at Stanford, are Theano, Torch, and TensorFlow.

Torch framework is written in Lua, which is a wonderful language but unfortunately the one that we are not terribly familiar with. It is also not very popular outside the game developing community and the deep learning community.

For Theano, the book "Fundamentals of Deep Learning" did a great comparison of Tensorflow and Theano:

"*First, Theano has an additional "graph compilation" step that took significant amounts of time while setting up certain kinds of deep learning architectures. While small in comparison to train time, this compilation phase proved frustrating while writing and debugging new code.*

Second, TensorFlow has a much cleaner interface as compared to Theano. Many classes of models can be expressed in significantly fewer lines without sacrificing the expressiveness of the framework.

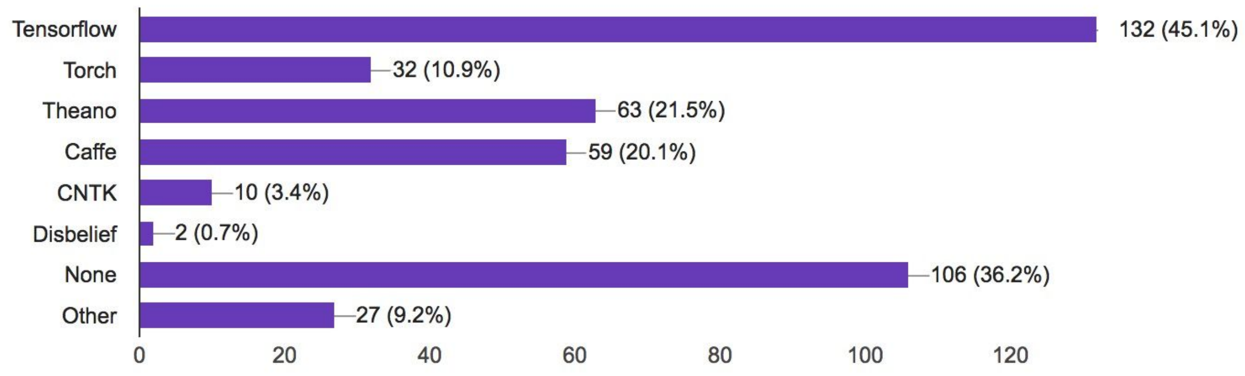
Finally, TensorFlow was built with production use in mind, whereas Theano was designed by researchers almost purely for research purposes.

As a result, TensorFlow has many features out of the box and in the works that make it a better choice for real systems (the ability to run in mobile environments, to easily build models that span multiple GPUs on a single machine, and to train large-scale networks in a distributed fashion)."

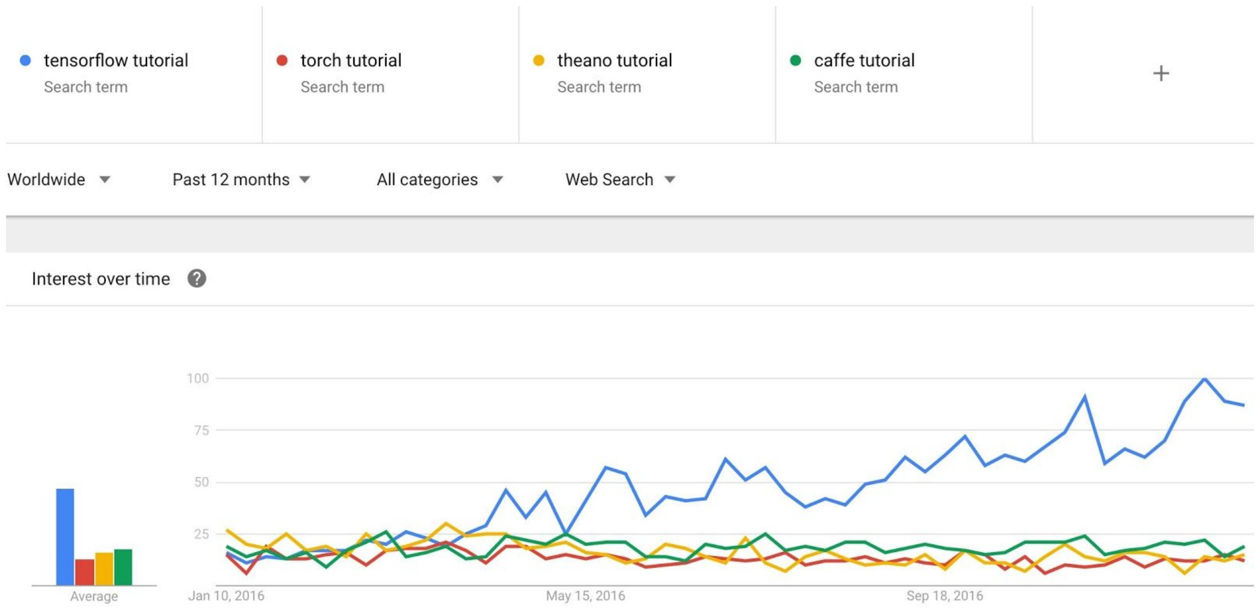
For those reasons,TensorFlow became out deep learning library of choice. In summary, we chose TensorFlow because:

- Python API
- Portability: deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API
- Flexibility: from Raspberry Pi, Android, Windows, iOS, Linux to server farms
- Visualization (TensorBoard is da bomb)
- Checkpoints (for managing experiments)
- Auto-differentiation autodiff (no more taking derivatives by hand. Yay)
- Large community (> 10,000 commits and > 3000 TF-related repos in one year)
- Awesome projects already using TensorFlow

TensorFlow is new but it has already gained traction in both industries and research communities. Breakdown of usage of deep learning libraries among 293 students who signed up for the course CS 20SI: “TensorFlow for Deep Learning Research”. None means students have never used any deep learning library before.



Demand for TensorFlow tutorials compared to tutorials for other deep learning libraries



<u>Companies using TensorFlow</u>
Google
DeepMind
OpenAI
Snapchat
Uber
Airbus
eBay
Dropbox
A bunch of startups

<u>Some cool projects using Tensorflow</u>

DeepMind's WaveNet Text to speech
Google Brain's Magenta project that uses machine learning to create compelling art and music
Neural Style Translation
Major Improvement to Google Translate

Some more examples of real world projects using TensorFlow, taken from Google Research Blog, 2016:

- Australian marine biologists are using TensorFlow to find sea cows in tens of thousands of hi-res photos to better understand their populations, which are under threat of extinction.
- An enterprising Japanese cucumber farmer trained a model with TensorFlow to sort cucumbers by size, shape, and other characteristics.

- Radiologists have adapted TensorFlow to identify signs of Parkinson’s disease in medical scans.
- Data scientists in the Bay Area have rigged up TensorFlow and the Raspberry Pi to keep track of the Caltrain.

Isn’t that cool?

Getting started with one-liner Tensorflow

This part is cool but not so important. You can skip the world of one-liner TensorFlow.

1. TF Learn (tf.contrib.learn)
TensorFlow has a simplified interface, TF Learn (tensorflow.contrib.learn), that provides readily available models that users can simply call. This was purposely created to mimic scikit learn for deep learning “to smooth the transition from the scikit-learn world of one-liner machine learning into the more open world of building different shapes of ML models. In fact, TF Learn was originally an independent project called Scikit Flow (SKFlow).

TF Learn allows you to load in data, construct a model, fit your model using the training data, evaluate the accuracy, each using a single line. Some models that you can call using one line in TF Learn include LinearClassifier, LinearRegressor, DNNClassifier. Google has really good tutorials on how to build custom models using TF Learn.

Document on TF Learn’s DNNClassifier: An example of TF Learn taken from TensorFlow Examples repo on GitHub.

iris.py

```
# Load dataset.
  iris = tf.contrib.learn.datasets.load_dataset('iris')
  x_train, x_test, y_train, y_test = cross_validation.train_test_split(
        iris.data, iris.target, test_size=0.2, random_state=42)

  # Build 3 layer DNN with 10, 20, 10 units respectively.
  feature_columns = tf.contrib.learn.infer_real_valued_columns_from_input(
        x_train)
  classifier = tf.contrib.learn.DNNClassifier(
        feature_columns=feature_columns, hidden_units=[10, 20, 10], n_classes=3)

  # Fit and predict.
  classifier.fit(x_train, y_train, steps=200)
  predictions = list(classifier.predict(x_test, as_iterable=True))
  score = metrics.accuracy_score(y_test, predictions)
```

```
    print('Accuracy: {0:f}'.format(score))
```

You can see that TF Learn lets you load data with one single line, split data in another line, and you can call the built in deep neueral network classifier DNNClassifier with the number of hidden units of your choice. In this case, the first layer has 10 hidden units, the second layer has 20 hidden units, the third layer has 10 hidden units. You can also specify the number of label classes, the type of optimization (for example, gradient descent), or the initial weight.

You can visit TF Learn examples repository for more examples of TF Learn. However, keep in mind that most of the built-in models are implemented using deprecated functions, so you see a lot of warnings if you call them.

2. TF-Slim (tf.contrib.slim)
Another simple API called TF-Slim to simplify building, training and evaluating neural networks.

3. High level APIs on top TensorFlow
There are many high level APIs built on top of TensorFlow. Some of the most popular APIs included Keras (keras@GitHub), TFLearn (tflearn@GitHub), and Pretty Tensor (prettytensor@GitHub)
.
Note: You should not confuse the high level API TFLearn (no space between TF and Learn) with the simplified interface TF Learn. TFLearn supports most of recent deep learning models, such as ConvNets, LSTM, BiRNN, ResNets, Generative networks and features such as BatchNorm, PReLU. TFLearn is developed by Aymeric Damien.

Spoiler alert: Aymeric is giving a guest lecture in a few weeks!

However, the primary purpose of TensorFlow 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. This is more complicated, but offers much more flexibility. You can build almost any architecture you can think of in TensorFlow.

Data Flow Graph

The next key phrase in the definition is “data flow graph”. This key phrase means that TF does all its computation in graphs.

Please refer to the slides of the first lecture (from page 23) for more details!

http://danijar.com/what-is-a-tensorflow-session/

The session will also allocate memory to store the current value of the variable.

As you can see, the value of our variable is only valid within one session. If we try to query the value afterwards in a second session, TensorFlow will raise an error because the variable is not initialized there.

References:

https://www.tensorflow.org/
“Tensorflow for Machine Intelligence”
“Hands-On Machine Learning with Scikit-Learn and TensorFlow”. Chapter 9: Up and running with TensorFlow
“Fundamentals of Deep Learning”. Chapter 3: Implementing Neural Networks in TensorFlow.