TENSORFLOW FOR DEEP LEARNING – PART 1

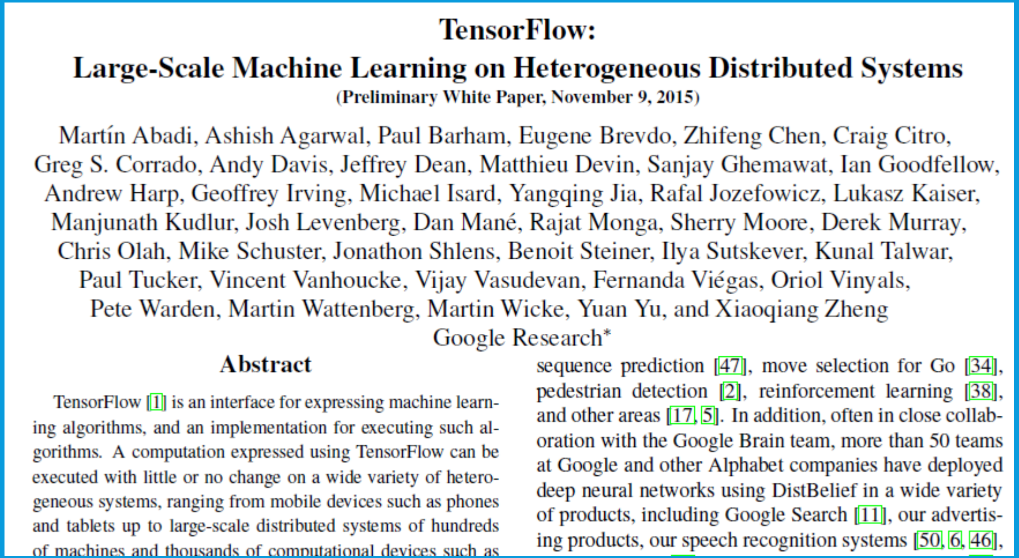
Anantharaman Palacode Narayana Iyer

JNResearch

ananth@jnresearch.com

15 April 2016

REFERENCES



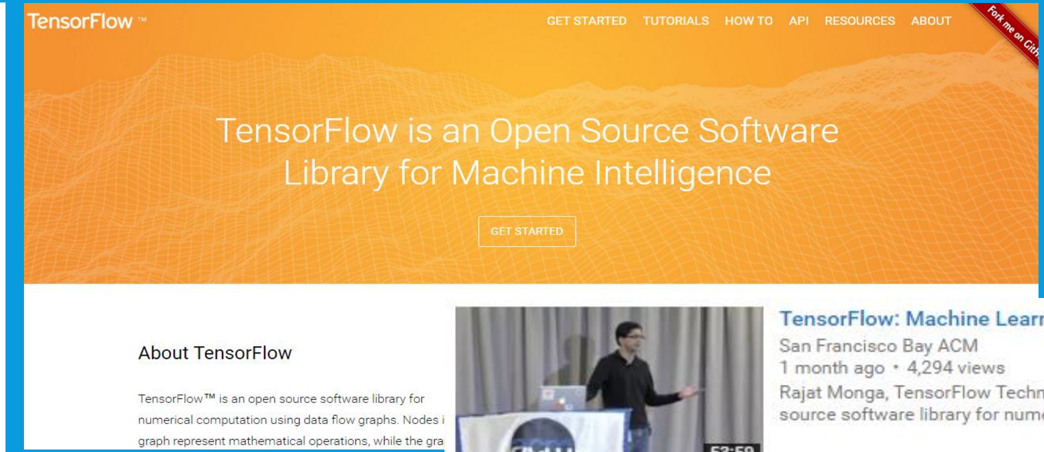
TensorFlow:
Large-Scale Machine Learning on Heterogeneous Distributed Systems
(Preliminary White Paper, November 9, 2015)

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng
Google Research*

Abstract

TensorFlow [1] is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms. A computation expressed using TensorFlow can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones and tablets up to large-scale distributed systems of hundreds of machines and thousands of computational devices such as

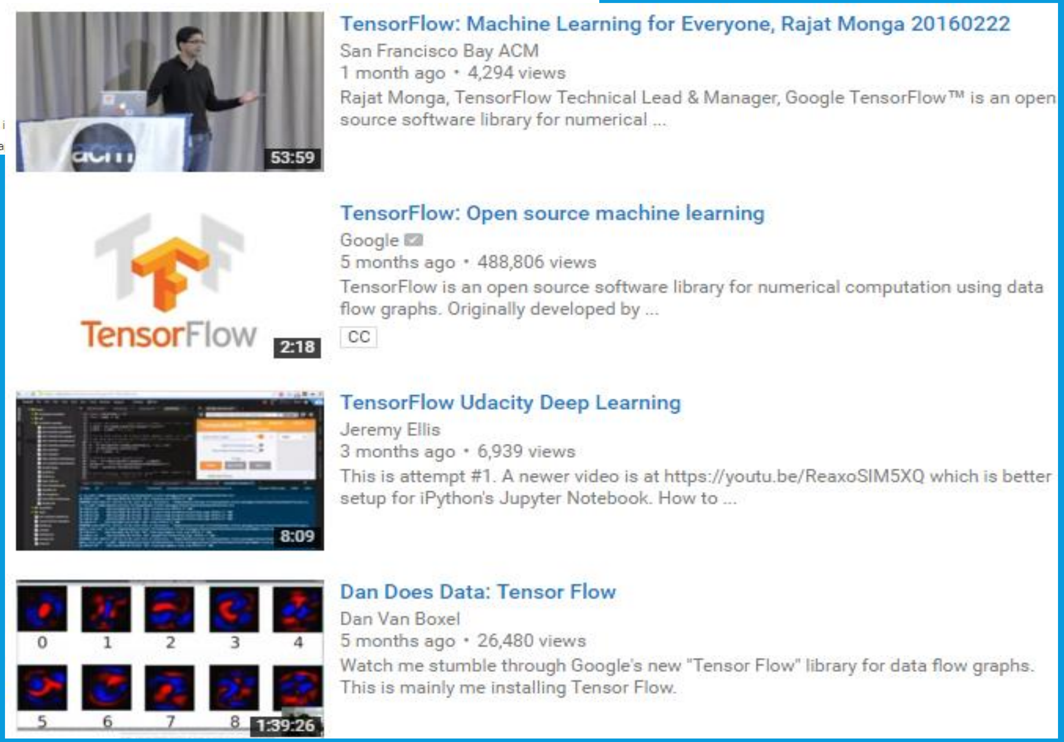
sequence prediction [47], move selection for Go [34], pedestrian detection [2], reinforcement learning [38], and other areas [17, 5]. In addition, often in close collaboration with the Google Brain team, more than 50 teams at Google and other Alphabet companies have deployed deep neural networks using DistBelief in a wide variety of products, including Google Search [11], our advertising products, our speech recognition systems [50, 6, 46],



TensorFlow is an Open Source Software Library for Machine Intelligence

About TensorFlow

TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes i graph represent mathematical operations, while the gra

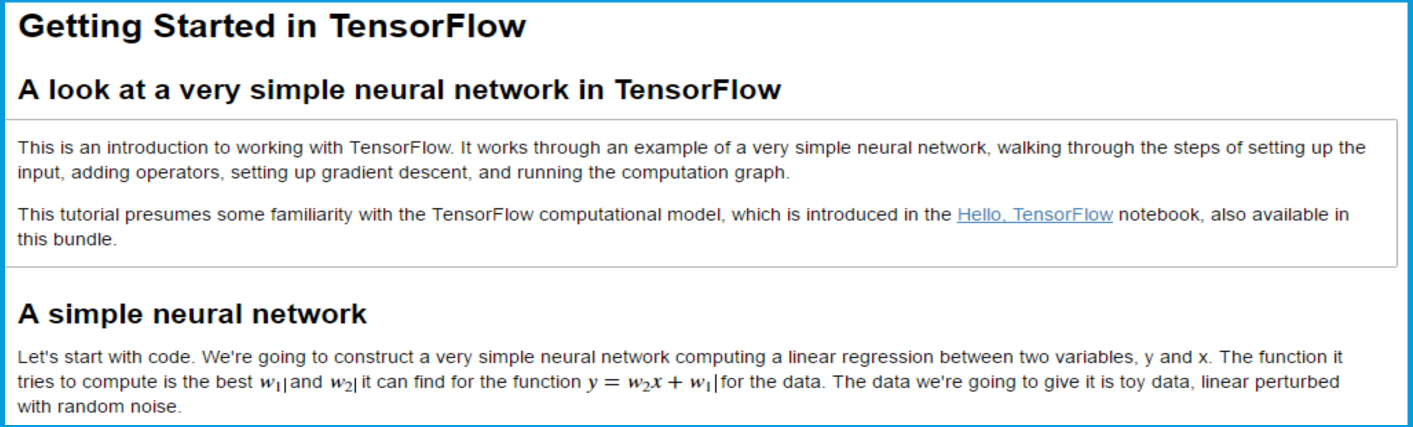


TensorFlow: Machine Learning for Everyone, Rajat Monga 20160222
San Francisco Bay ACM
1 month ago • 4,294 views
Rajat Monga, TensorFlow Technical Lead & Manager, Google TensorFlow™ is an open source software library for numerical ...

TensorFlow: Open source machine learning
Google
5 months ago • 488,806 views
TensorFlow is an open source software library for numerical computation using data flow graphs. Originally developed by ...
CC

TensorFlow Udacity Deep Learning
Jeremy Ellis
3 months ago • 6,939 views
This is attempt #1. A newer video is at https://youtu.be/ReaxoSIM5XQ which is better setup for iPython's Jupyter Notebook. How to ...

Dan Does Data: Tensor Flow
Dan Van Boxel
5 months ago • 26,480 views
Watch me stumble through Google's new "Tensor Flow" library for data flow graphs. This is mainly me installing Tensor Flow.



Getting Started in TensorFlow

A look at a very simple neural network in TensorFlow

This is an introduction to working with TensorFlow. It works through an example of a very simple neural network, walking through the steps of setting up the input, adding operators, setting up gradient descent, and running the computation graph.

This tutorial presumes some familiarity with the TensorFlow computational model, which is introduced in the Hello, TensorFlow notebook, also available in this bundle.

A simple neural network

Let's start with code. We're going to construct a very simple neural network computing a linear regression between two variables, y and x. The function it tries to compute is the best w_1 and w_2 it can find for the function $y = w_2 x + w_1$ for the data. The data we're going to give it is toy data, linear perturbed with random noise.

AGENDA

- Tensorflow overview
  - When should we use it?
  - High level steps
- Basic operations to get started

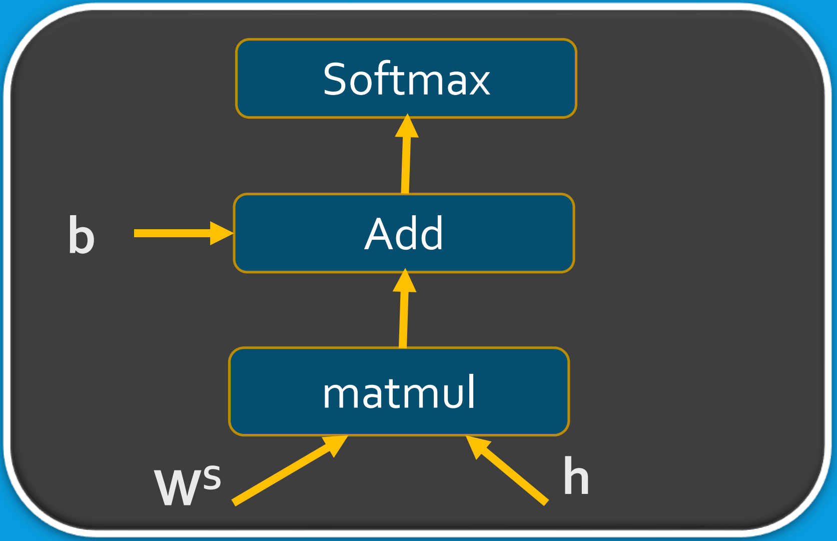
WHAT IS TENSORFLOW?

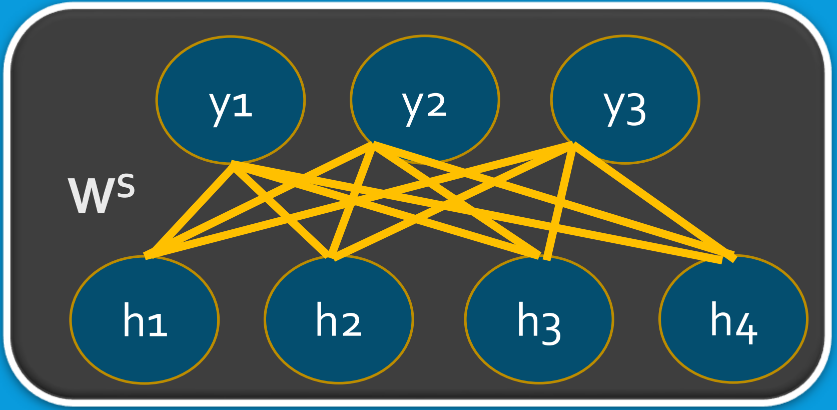
- **From the whitepaper:** "TensorFlow is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms."
- My perspective:
  - Tensorflow is a framework with the following characteristics:
    - A Framework that allows a developer to express his machine learning algorithm symbolically, performing compilation of these statements and executing them.
    - A programming metaphor that requires the developer to model the machine learning algorithm as a computation graph.
    - A set of Python classes and methods that provide an API interface
    - A re-targetable system that can run on different hardware

WHY USE TENSORFLOW?

- Suppose we are addressing a problem using a machine learning approach and our proposed solution can be implemented by one of the standard classifier architectures. (say, an SVM)
- We often use one of the standard ML libraries (say, scikit) for the above situation instead of implementing our own classifier
- If the existing ML models aren't optimally suited for our problem, we might need to come up with a more custom architecture that might give a higher accuracy. In such a case we might end up building the classifier from scratch, starting with mathematical formulations and implementing ground up.
- Problems that call for deep learning oriented solutions often require custom architectures.
- A lego-piece based approach where a library offers basic building blocks would be more suited. With in-built support to run on GPUs, such libraries are well suited for custom designs
- With Tensorflow's building blocks, you can quickly build arbitrarily complex architectures

TENSORFLOW COMPUTE GRAPH



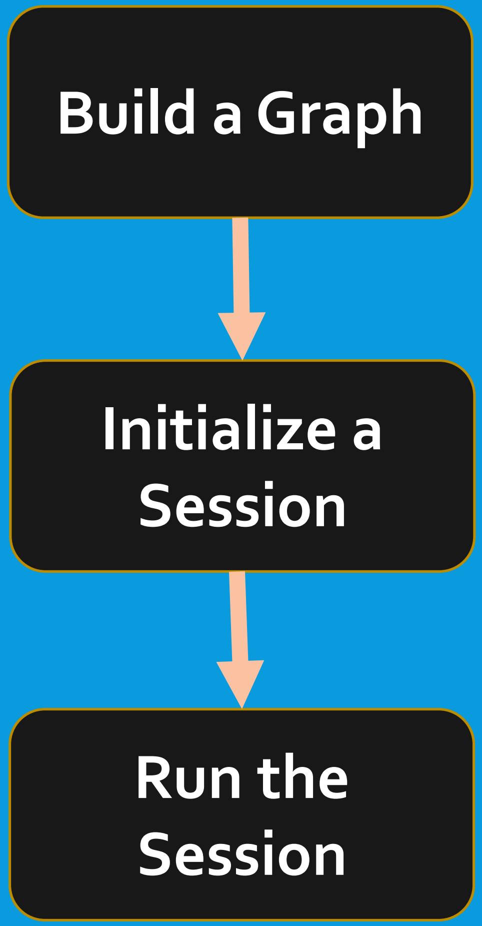


- Operations are the nodes of the graph
- Tensors constitute the edges

$$P(y|h) = Softmax(dot\_prod)$$

$$dot\_prod = (W^S h + b)$$

DEVELOPING WITH TENSORFLOW: HIGH LEVEL STEPS



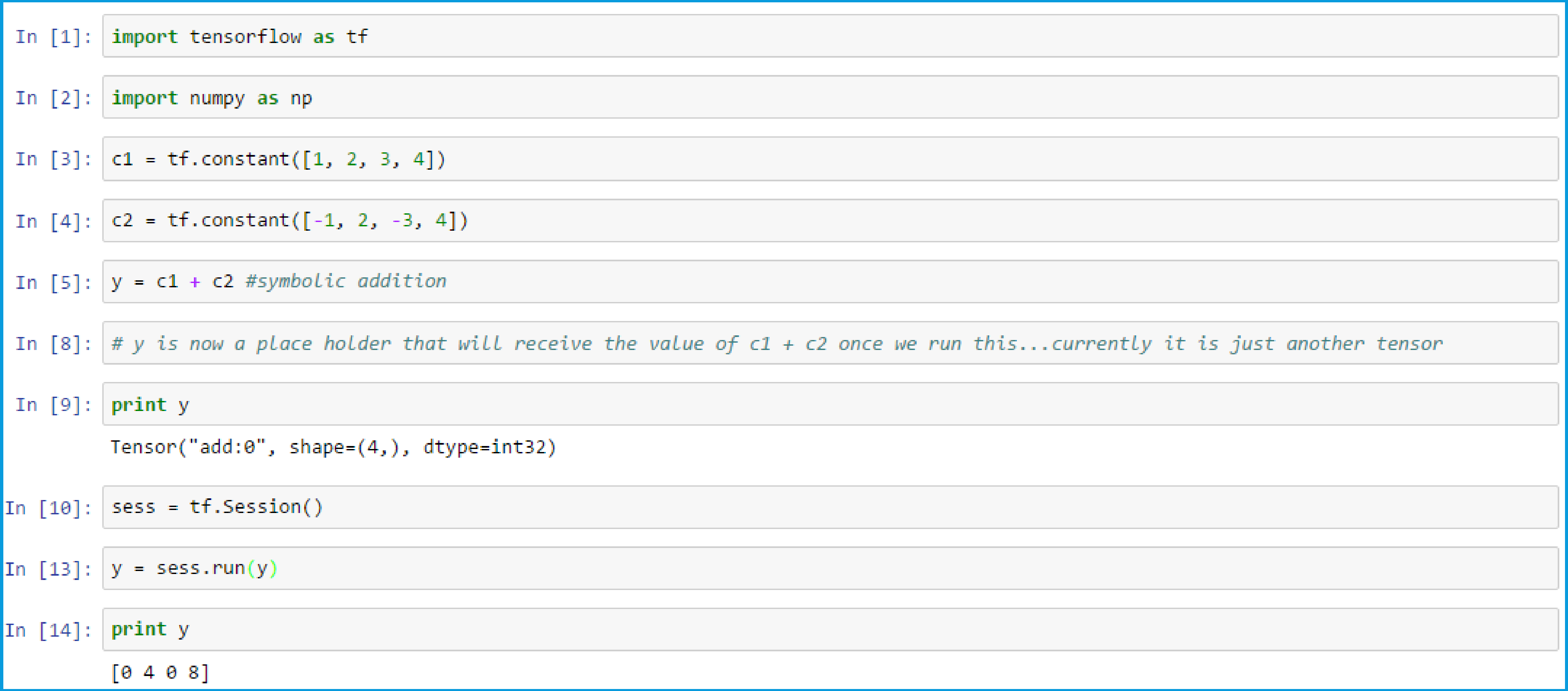
- Perform any pre-processing steps to set up the dataset for the classifier.
- Set up the variables (such as the Weights and biases) and placeholders (such as the input vector) that are fed at runtime
- Define the required computations. E.g. you may define the dot_product computation using the matmul method of tensorflow feeding it the weight matrix variable and the input vector (that may be defined as a placeholder).

  Note: Till this point nothing is executed by Tensorflow
- Start a session instance and initialize all variables
- Run the session

SOME METHODS TO GET STARTED

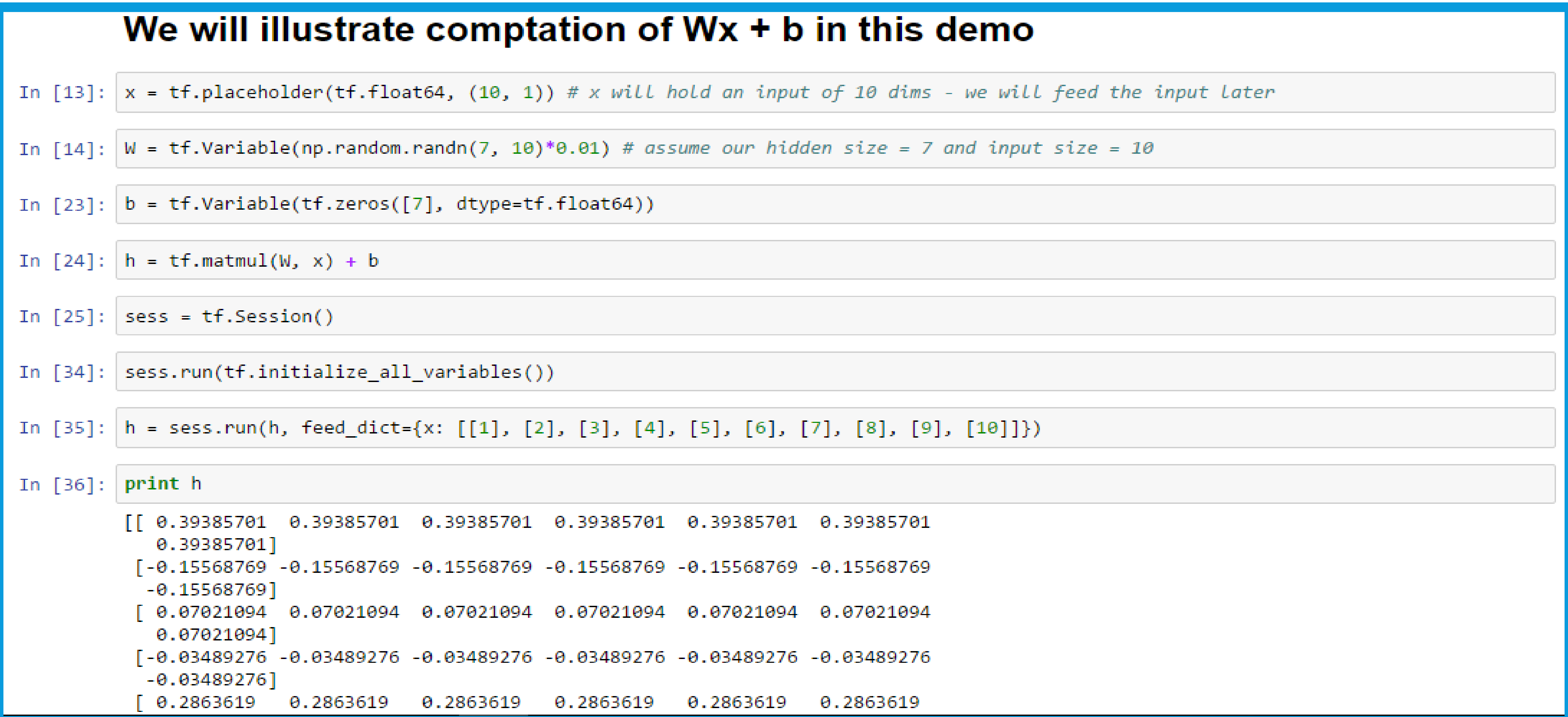
- tf.constant
- tf.placeholder
- tf.variable
- tf.matmul( )
- tf.softmax( )
- tf.reduce_sum( )
- tf.train.GradientDescentOptimizer( )
- tf.initialize_all_variables( )
- tf.Session( )
- tf.run( )

EXAMPLE 1



```
In [1]: import tensorflow as tf

In [2]: import numpy as np

In [3]: c1 = tf.constant([1, 2, 3, 4])

In [4]: c2 = tf.constant([-1, 2, -3, 4])

In [5]: y = c1 + c2 #symbolic addition

In [8]: # y is now a place holder that will receive the value of c1 + c2 once we run this...currently it is just another tensor

In [9]: print y

        Tensor("add:0", shape=(4,), dtype=int32)

In [10]: sess = tf.Session()

In [13]: y = sess.run(y)

In [14]: print y

        [0 4 0 8]
```

EXAMPLE 2



We will illustrate comptation of Wx + b in this demo

```
In [13]: x = tf.placeholder(tf.float64, (10, 1)) # x will hold an input of 10 dims - we will feed the input later

In [14]: W = tf.Variable(np.random.randn(7, 10)*0.01) # assume our hidden size = 7 and input size = 10

In [23]: b = tf.Variable(tf.zeros([7], dtype=tf.float64))

In [24]: h = tf.matmul(W, x) + b

In [25]: sess = tf.Session()

In [34]: sess.run(tf.initialize_all_variables())

In [35]: h = sess.run(h, feed_dict={x: [[1], [2], [3], [4], [5], [6], [7], [8], [9], [10]]})

In [36]: print h

         [[ 0.39385701  0.39385701  0.39385701  0.39385701  0.39385701  0.39385701
            0.39385701]
          [-0.15568769 -0.15568769 -0.15568769 -0.15568769 -0.15568769 -0.15568769
           -0.15568769]
          [ 0.07021094  0.07021094  0.07021094  0.07021094  0.07021094  0.07021094
            0.07021094]
          [-0.03489276 -0.03489276 -0.03489276 -0.03489276 -0.03489276 -0.03489276
           -0.03489276]
          [ 0.2863619   0.2863619   0.2863619   0.2863619   0.2863619   0.2863619
```

SOME OBSERVATIONS

- The major benefits of using tensorflow to develop deep learning algorithms include:
  - A good set of lego pieces that includes methods that support different types of output layers, activation functions, convolution, pooling and so on. This greatly helps when you are researching with custom architectures.
  - Support for GPU: This helps in realizing better performance
  - Strong type checking: tensorflow implements tighter type checks compares to numpy. For instance when you try to mix float32 and float64 types, it throws an error. As tensorflow performs its own compilation, given the computation graph, it's possibly imposing tighter compile time checks.
  - Support from third party – as the product is well invested by Google, we might see a large number of third party libraries, code that aid our development process
- Tensorflow needs some initial effort to get over the learning curve. Initially, debugging might also be a challenge.