

A quick introduction to Tensorflow

Machine Learning Spring 2019



Many ML libraries:



An open-source library by Google:

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*

Further reading:

- Official website: <https://www.tensorflow.org/>

If want to master every details.

- *Deep Learning with Python* by Francois Chollet

Focus on Keras

- Hands-On Machine Learning with Scikit-Learn and TensorFlow

Tensorflow part is somewhat outdated. (Though this book is published in 2017).

Core Functionalities:

- Augmented tensor operations (nearly identical to numpy)
Seamless interfaces with existing programs.
- Automatic differentiation
The very core of Optimization based algorithms.
- Parallel(CPU/GPU/TPU) and Distributed(multi-machine) Computation
Essential for large(industrial level) applications.
Implemented in C++. Highly Efficient.

Automatic differentiation: Through back-propagation

- Only operations with “sub-gradient” can be applied on Tensor

In general

For convenience, we generalize the notation
 h is the layer number, w_{mn}^h connects m -th node in
layer $h - 1$ (i.e., z_m^{h-1}) to the n -th node in layer h (i.e., z_n^h)

To compute each $\frac{\partial L}{\partial w_{mn}^h}$

1. Initialize $\frac{\partial L}{\partial w_{mn}^h} \leftarrow 0$

2. Find all the paths from z_n^h to the output node y

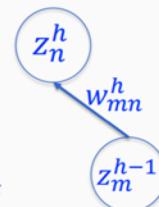
3. For each path s

3.1 For each node z in s

- Compute the partial derivative of z 's parent over z
if the node is the output node y , then compute $\frac{\partial L}{\partial y}$

- Multiply all the partial derivatives along the path s to obtain g_s

- Add to the derivative: $\frac{\partial L}{\partial w_{mn}^h} \leftarrow \frac{\partial L}{\partial w_{mn}^h} + g_s \frac{\partial z_n^h}{\partial w_{mn}^h}$



Automatic differentiation: Through back-propagation

- Only operations with “sub-gradient” can be applied on Tensor

Arithmetic : +, -, *, /

Elementary functions: exp, log, max, sin, tan

In general

To compute each $\frac{\partial L}{\partial w_{mn}^h}$

1. Initialize $\frac{\partial L}{\partial w_{mn}^h} \leftarrow 0$

1. Find all the paths from z_n^h to the output node y

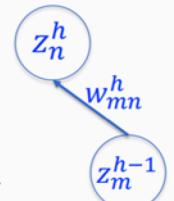
2. For each path s

2.1 For each node z in s

- Compute the partial derivative of z 's parent over z if the node is the output node y , then compute $\frac{\partial L}{\partial y}$

- Multiply all the partial derivatives along the path s to obtain g_s

- Add to the derivative: $\frac{\partial L}{\partial w_{mn}^h} \leftarrow \frac{\partial L}{\partial w_{mn}^h} + g_s \frac{\partial z_n^h}{\partial w_{mn}^h}$



For convenience, we generalize the notation
 h is the layer number, w_{mn}^h connects m -th node in
layer $h - 1$ (i.e., z_m^{h-1}) to the n -th node in layer h (i.e., z_n^h)

Automatic differentiation: Through back-propagation

- Only operations with “sub-gradient” can be applied on Tensor

Arithmetic : +, -, *, /

Elementary functions: exp, log, max, sin, tan

- What operations are not “differentiable”?

In general

To compute each $\frac{\partial L}{\partial w_{mn}^h}$

1. Initialize $\frac{\partial L}{\partial w_{mn}^h} \leftarrow 0$

1. Find all the paths from z_n^h to the output node y

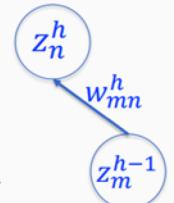
2. For each path s

2.1 For each node z in s

- Compute the partial derivative of z 's parent over z if the node is the output node y , then compute $\frac{\partial L}{\partial y}$

- Multiply all the partial derivatives along the path s to obtain g_s

- Add to the derivative: $\frac{\partial L}{\partial w_{mn}^h} \leftarrow \frac{\partial L}{\partial w_{mn}^h} + g_s \frac{\partial z_n^h}{\partial w_{mn}^h}$



Automatic differentiation: Through back-propagation

- Only operations with “sub-gradient” can be applied on Tensor

Arithmetic : +, -, *, /

Elementary functions: exp, log, max, sin, tan

- What operations are not “differentiable”?

For example: sampling

In general

To compute each $\frac{\partial L}{\partial w_{mn}^h}$

1. Initialize $\frac{\partial L}{\partial w_{mn}^h} \leftarrow 0$

1. Find all the paths from z_n^h to the output node y

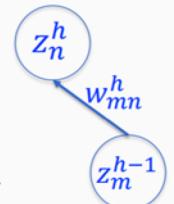
2. For each path s

2.1 For each node z in s

- Compute the partial derivative of z 's parent over z if the node is the output node y , then compute $\frac{\partial L}{\partial y}$

- Multiply all the partial derivatives along the path s to obtain g_s

- Add to the derivative: $\frac{\partial L}{\partial w_{mn}^h} \leftarrow \frac{\partial L}{\partial w_{mn}^h} + g_s \frac{\partial z_n^h}{\partial w_{mn}^h}$



Working process: Tensor, flow

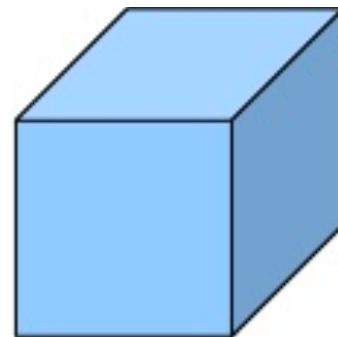
- Tensor: multi-dimension array



1d-tensor



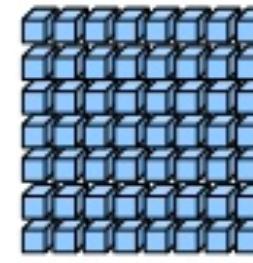
2d-tensor



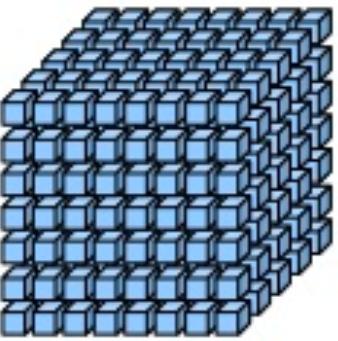
3d-tensor



4d-tensor



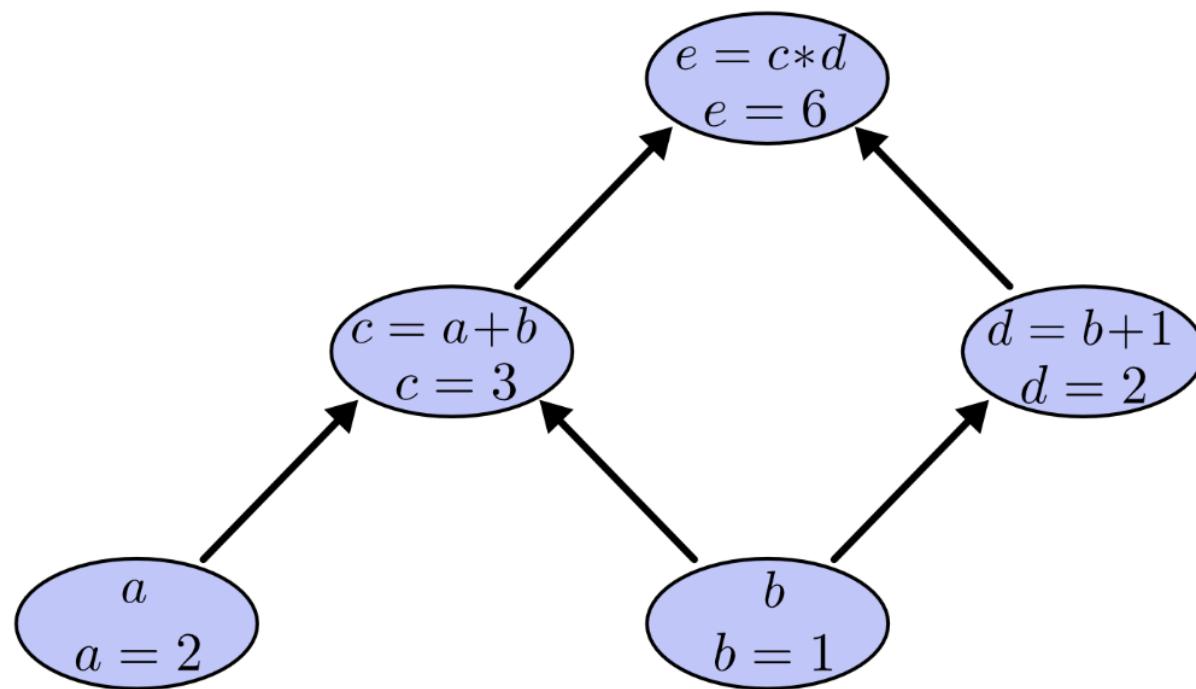
5d-tensor



6d-tensor

Working process: Tensor, flow

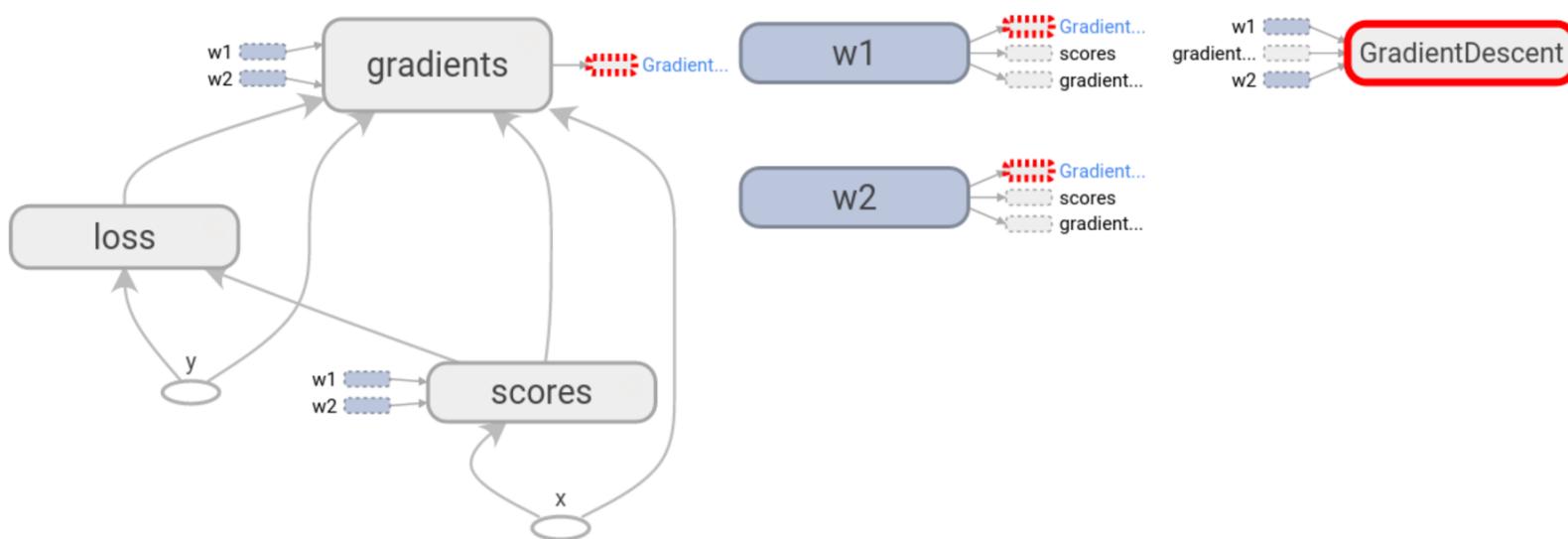
- flow: computation graph



Working process: Tensor, flow

- flow: computation graph

Can be visualize by tensorboard



Static vs Eager Mode

- Eager mode

Just like using numpy

- Static mode

Predefine tensors and computation graphs then let TF engine to execute the graphs. Similar to defining Python functions.

Static vs Eager Mode

- Eager mode

Just like using numpy

- Static mode: We focus solely on this mode in this tutorial

Subtlety appears here.

3 levels of tensorflow:

- Primitive tensorflow: lowest, finest control and most flexible
Suitable for most machine learning and deep learning algorithms.
- Keras(Mostly for deep learning):highest, most convenient to use, lack flexibility
- Tensorflow layers (Mostly for deep learning): somewhere at the middle.

General pipeline:

- Define inputs and variable tensors(weights/parameters).
*Keras will take care of these for you.
- Define computation graphs from inputs tensors to output tensors.
- Define loss function and optimizer
Once the loss is defined, the optimizer will compute the gradient for you!
- Execute the graphs.
*Keras will take care of this for you as well

Getting started today:

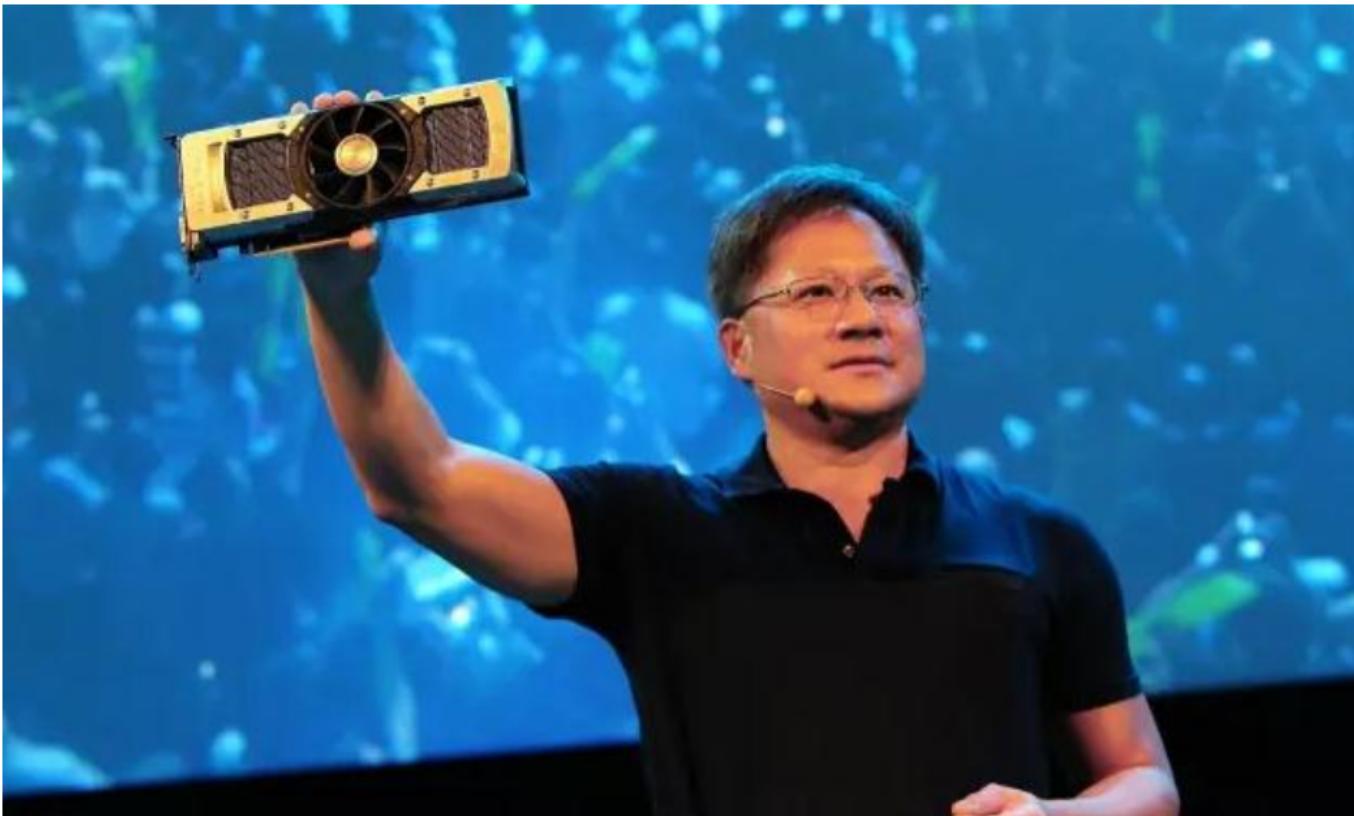
- GPU acceleration
- Installation
- Demos
 - Arithmetic and tensor operations
 - Primal SVM
 - Simple neural network in Keras.
 - Primitive and tensorflow layer if time allowed

GPU acceleration:

- Literally need one if training on non-toy models and datasets.

GPU acceleration:

- Literally need one if training on non-toy models and datasets.
- Nvidia GPUs Only



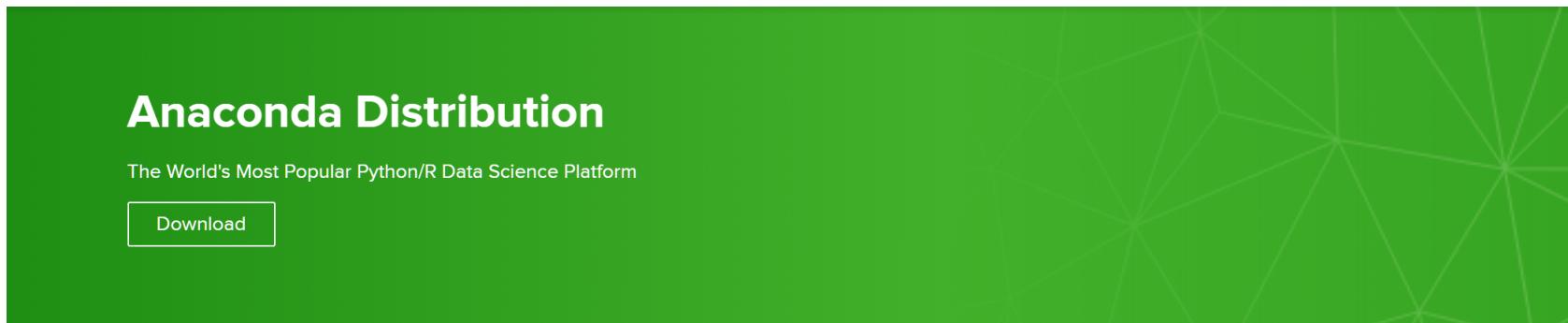
Where to find (free) computing resources:

- Your own Gaming PC
- CHPC(University) , CADE (Collage of Engineering)
- AWS/Google Cloud Platform: First time coupon.
- Google colab: Always free, equipped with GPU and TPU!

Installation: Anaconda

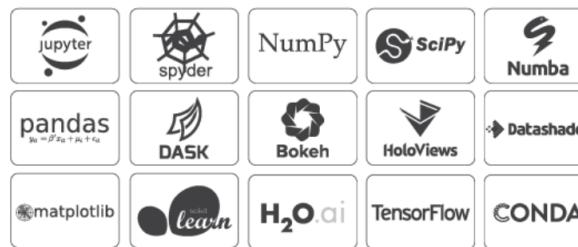
- Installed with Anaconda could save you much work.

<https://www.anaconda.com/>



The open-source Anaconda Distribution is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 11 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

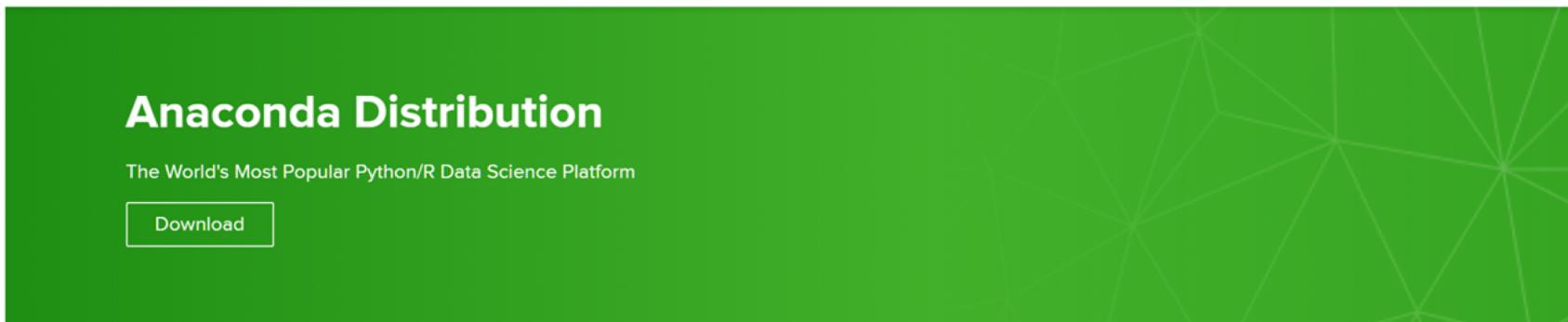
- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with **Conda**
- Develop and train machine learning and deep learning models with **scikit-learn**, **TensorFlow**, and **Theano**
- Analyze data with scalability and performance with **Dask**, **NumPy**, **pandas**, and **Numba**
- Visualize results with **Matplotlib**, **Bokeh**, **Datashader**, and **Holoviews**



Installation: Anaconda

- Installed with Anaconda could save you much work.

<https://www.anaconda.com/>



The open-source [Anaconda Distribution](#) is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 11 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with [Conda](#)
- Develop and train machine learning and deep learning models with [scikit-learn](#), [TensorFlow](#), and [Theano](#)
- Analyze data with scalability and performance with [Dask](#), [NumPy](#), [pandas](#), and [Numba](#)
- Visualize results with [Matplotlib](#), [Bokeh](#), [Databricks](#), and [HoloViews](#)

