Tamkang University
淡江大學

Tamkang University

Social Computing and Big Data Analytics
社群運算與大數據分析
Deep Learning with Google TensorFlow
(Google TensorFlow 深度學習)

1042SCBDA10
MIS MBA (M2226) (8628)
Wed, 8,9, (15:10-17:00) (B309)

Min-Yuh Day
戴敏育
Assistant Professor
專任助理教授
Dept. of Information Management, Tamkang University
淡江大學 資訊管理學系
http://mail. tku.edu.tw/myday/

2016-05-04

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2016/02/17	Course Orientation for Social Computing and Big Data Analytics (社群運算與大數據分析課程介紹)
2	2016/02/24	Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data (資料科學與大數據分析：探索、分析、視覺化與呈現資料)
3	2016/03/02	Fundamental Big Data: MapReduce Paradigm, Hadoop and Spark Ecosystem (大數據基礎：MapReduce典範、Hadoop與Spark生態系統)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
4	2016/03/09	Big Data Processing Platforms with SMACK: Spark, Mesos, Akka, Cassandra and Kafka (大數據處理平台SMACK：Spark, Mesos, Akka, Cassandra, Kafka)
5	2016/03/16	Big Data Analytics with Numpy in Python (Python Numpy 大數據分析)
6	2016/03/23	Finance Big Data Analytics with Pandas in Python (Python Pandas 財務大數據分析)
7	2016/03/30	Text Mining Techniques and Natural Language Processing (文字探勘分析技術與自然語言處理)
8	2016/04/06	Off-campus study (教學行政觀摩日)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
9	2016/04/13	Social Media Marketing Analytics (社群媒體行銷分析)
10	2016/04/20	期中報告 (Midterm Project Report)
11	2016/04/27	Deep Learning with Theano and Keras in Python (Python Theano 和 Keras 深度學習)
12	2016/05/04	Deep Learning with Google TensorFlow (Google TensorFlow 深度學習)
13	2016/05/11	Sentiment Analysis on Social Media with Deep Learning (深度學習社群媒體情感分析)

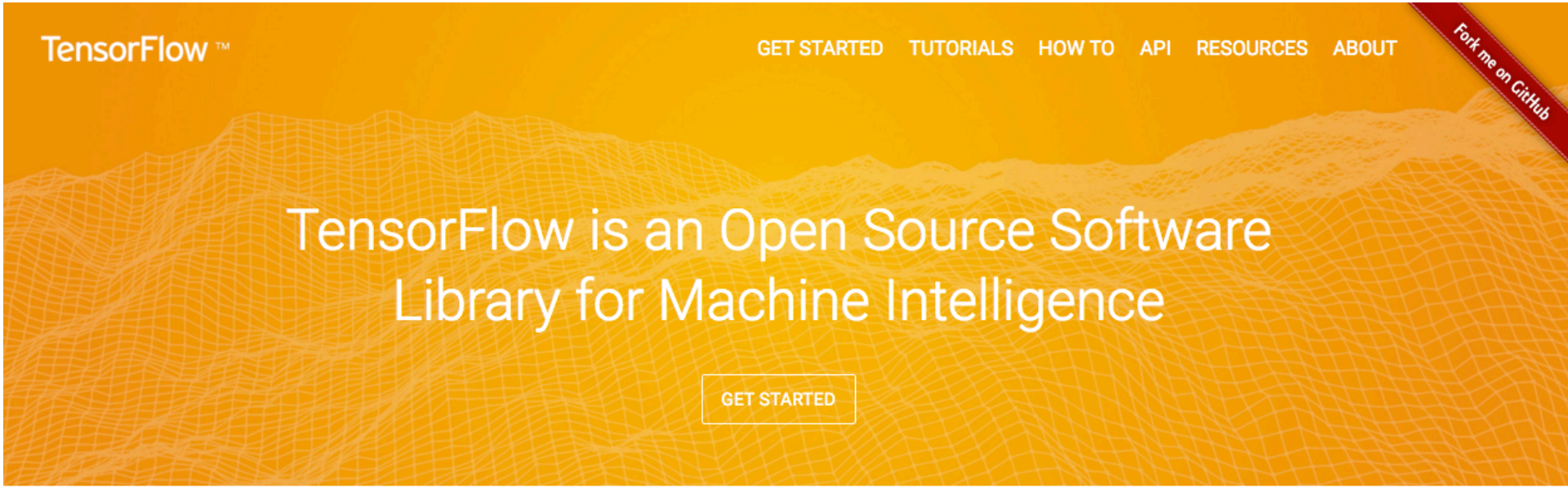
課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
14	2016/05/18	Social Network Analysis (社會網絡分析)
15	2016/05/25	Measurements of Social Network (社會網絡量測)
16	2016/06/01	Tools of Social Network Analysis (社會網絡分析工具)
17	2016/06/08	Final Project Presentation I (期末報告 I)
18	2016/06/15	Final Project Presentation II (期末報告 II)

TensorFlow

Source: https://github.com/tensorflow/tensorflow

Google TensorFlow



About TensorFlow

TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.

https://www.tensorflow.org/

TensorFlow

TensorFlow

is an

Open Source Software Library

for

Machine Intelligence

TensorFlow

numerical computation using data flow graphs

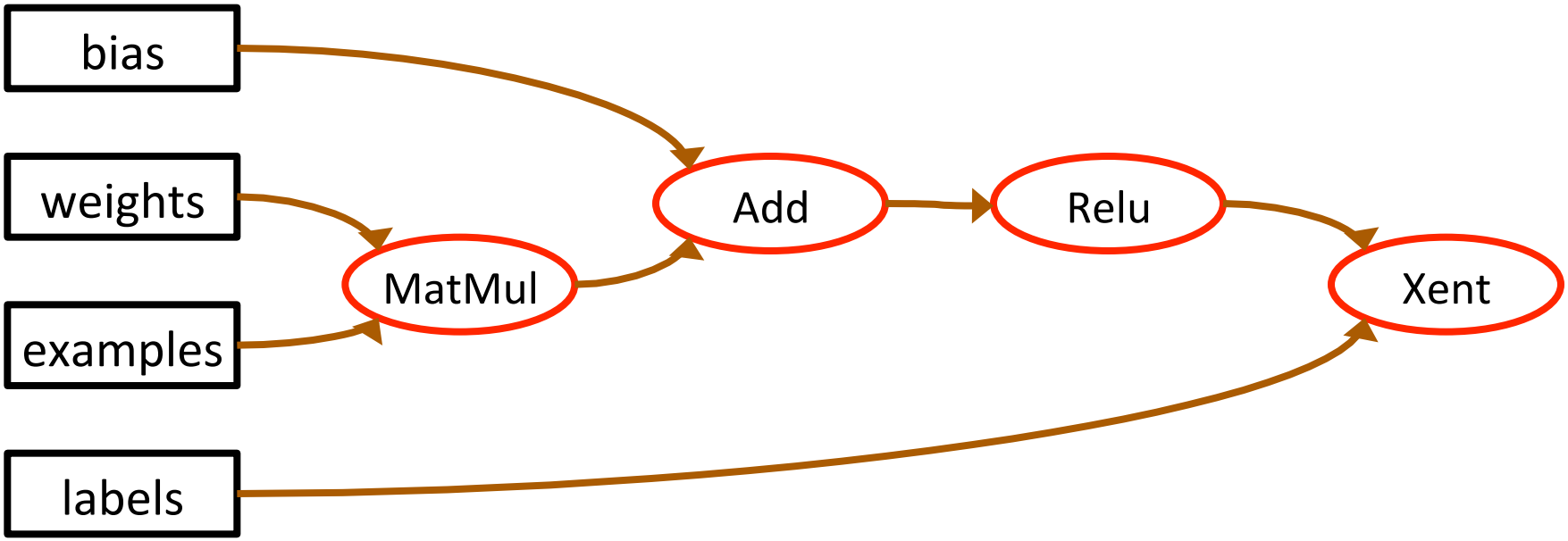
TensorFlow

Nodes:
mathematical operations

edges:
multidimensional data arrays (tensors)
communicated between nodes

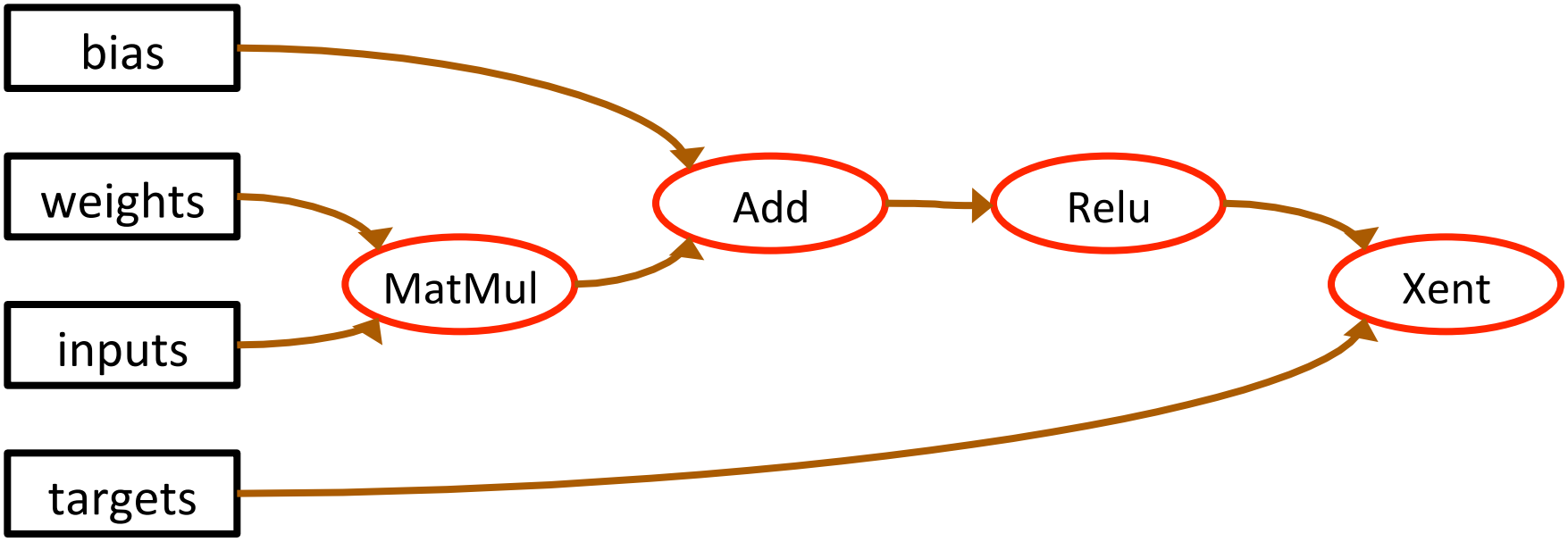
Computation is a Dataflow Graph

Graph of Nodes,
also called Operations or ops.



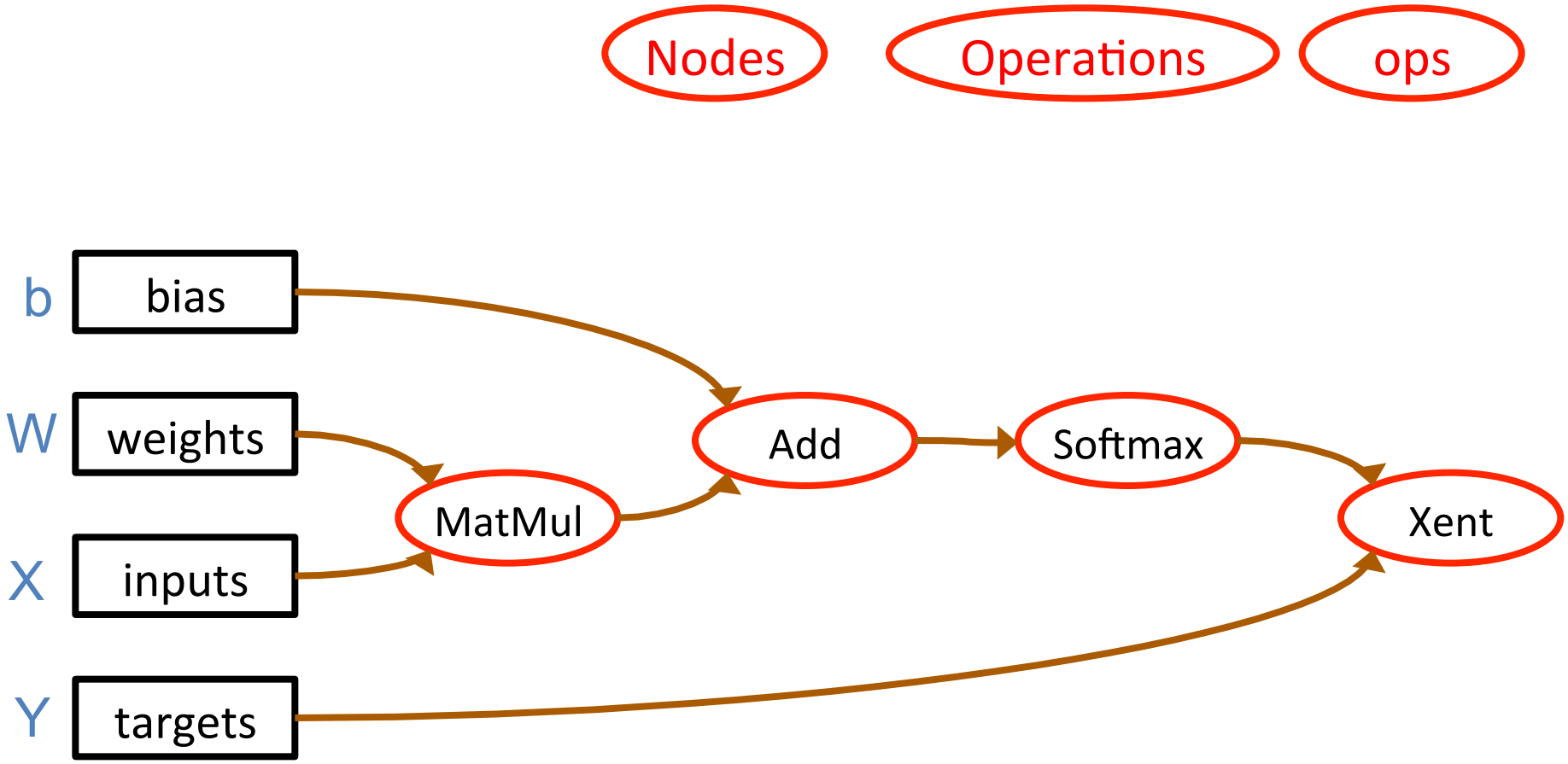
Computation is a Dataflow Graph

Edges are N-dimensional arrays: **Tensors**



Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

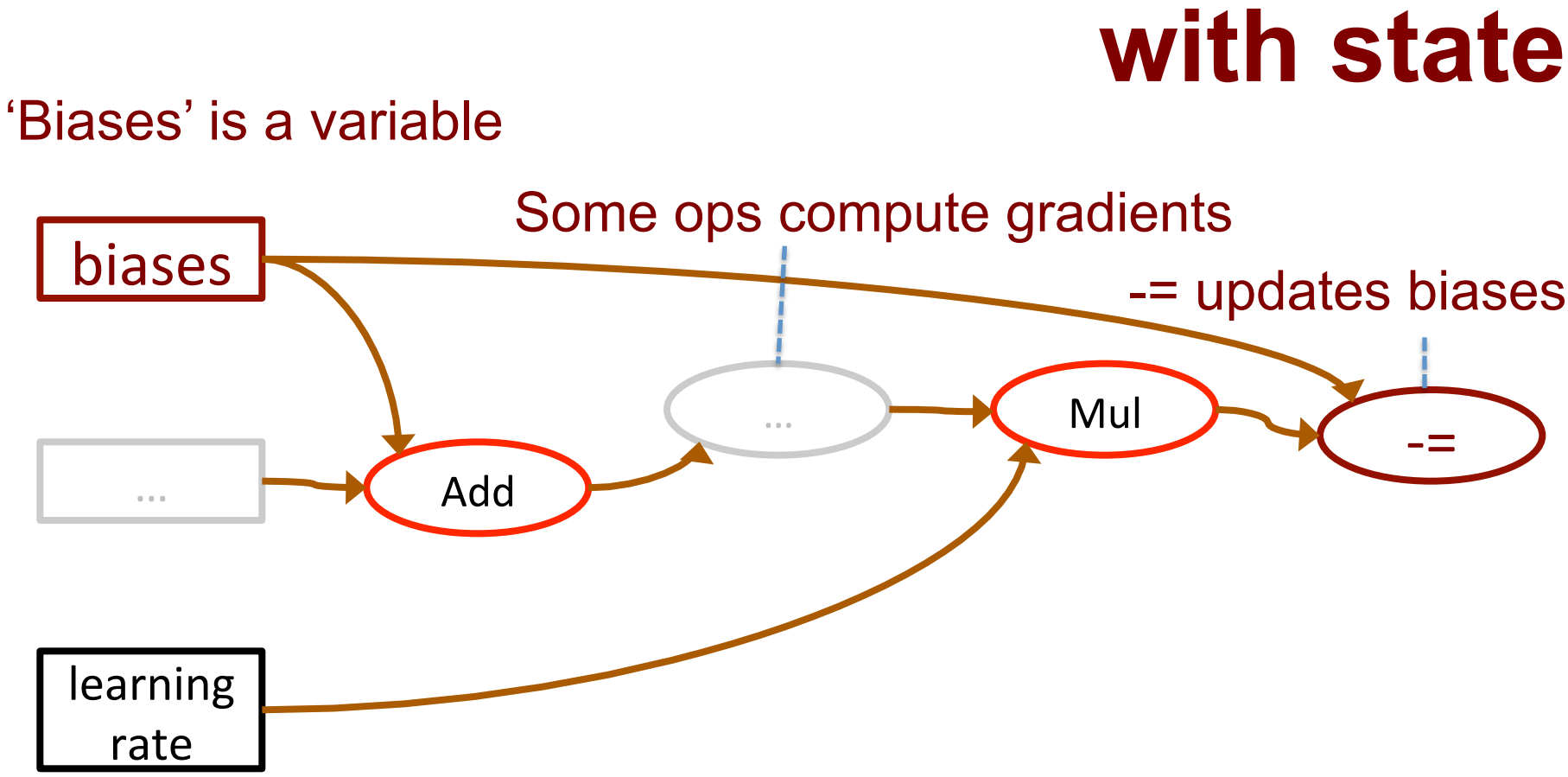
Logistic Regression as Dataflow Graph



Edges are N-dimensional arrays: **Tensors**

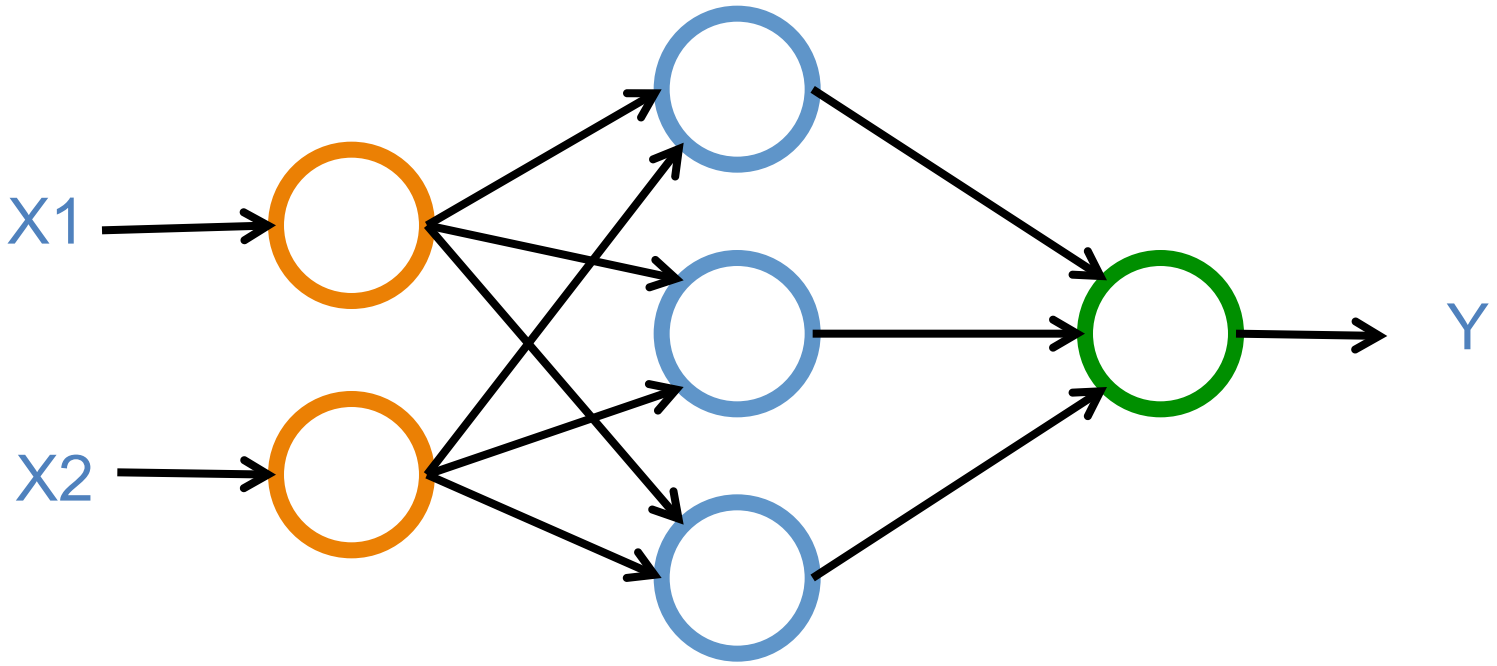
Computation is a Dataflow Graph

with state



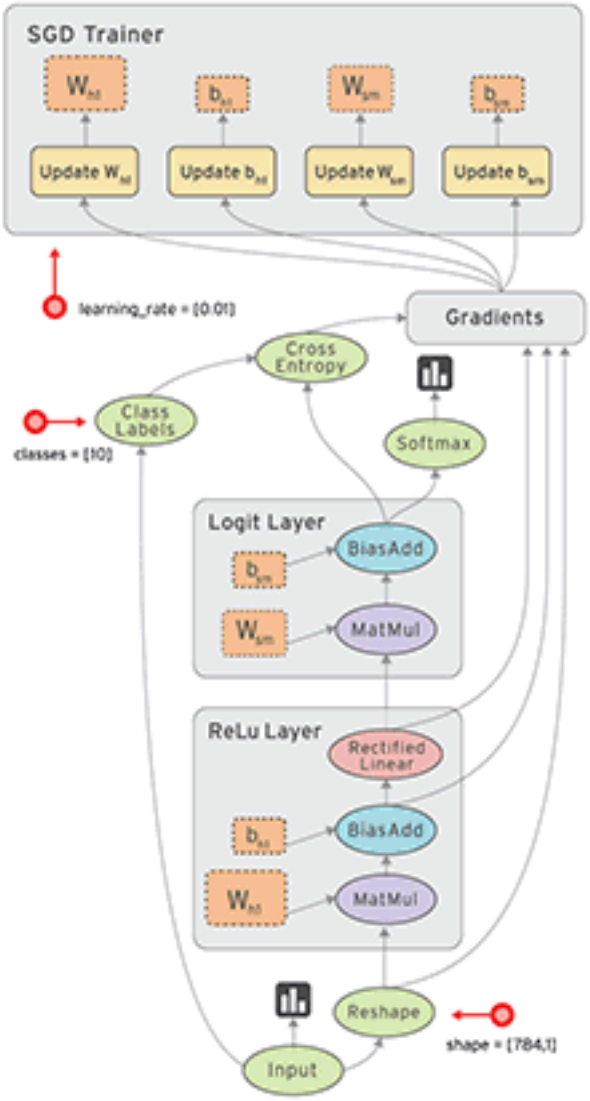
Neural Networks

Input Layer (X) Hidden Layer (H) **Output Layer (Y)**



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

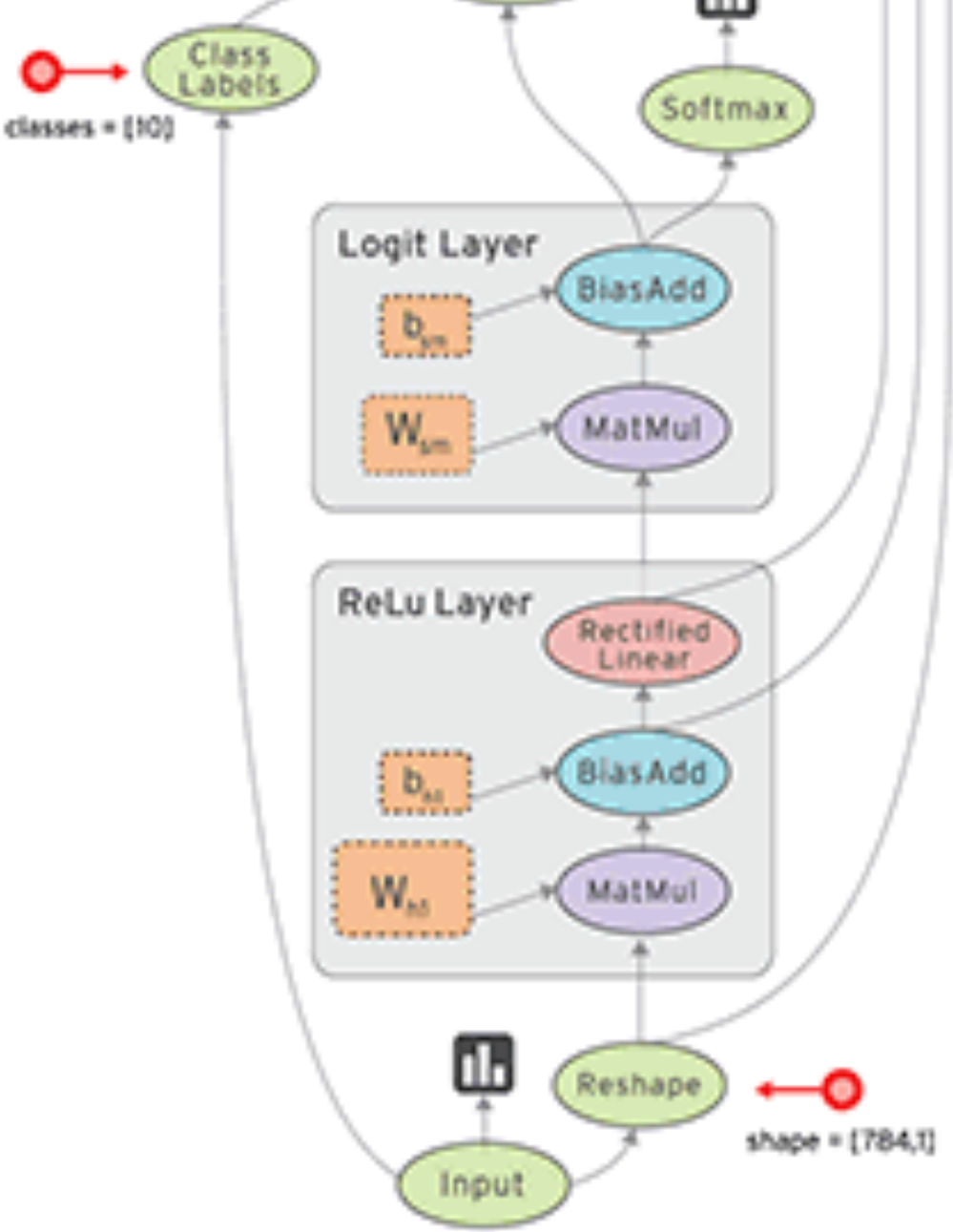
Data Flow Graph



Source: https://www.tensorflow.org/

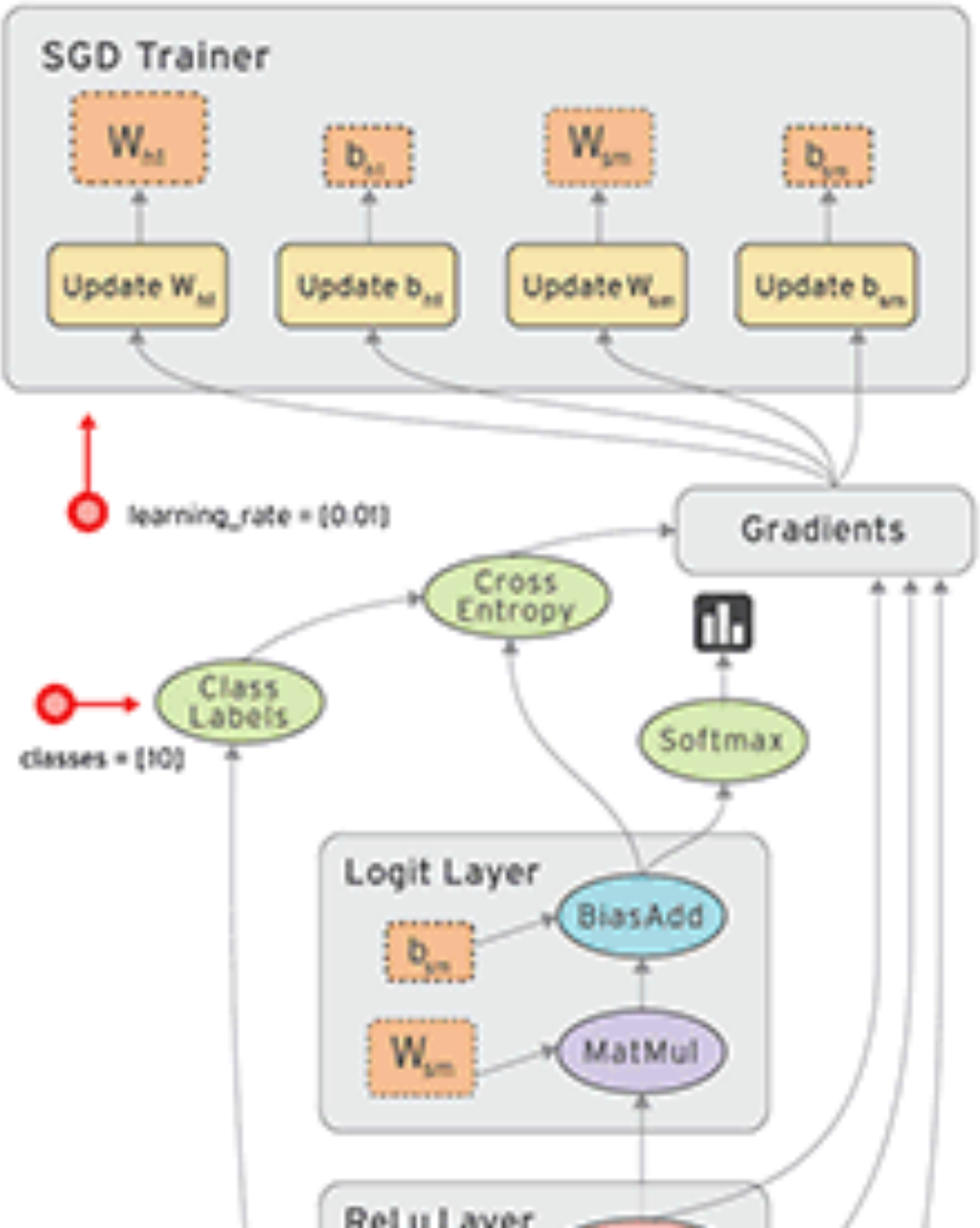
TensorFlow

Data Flow Graph

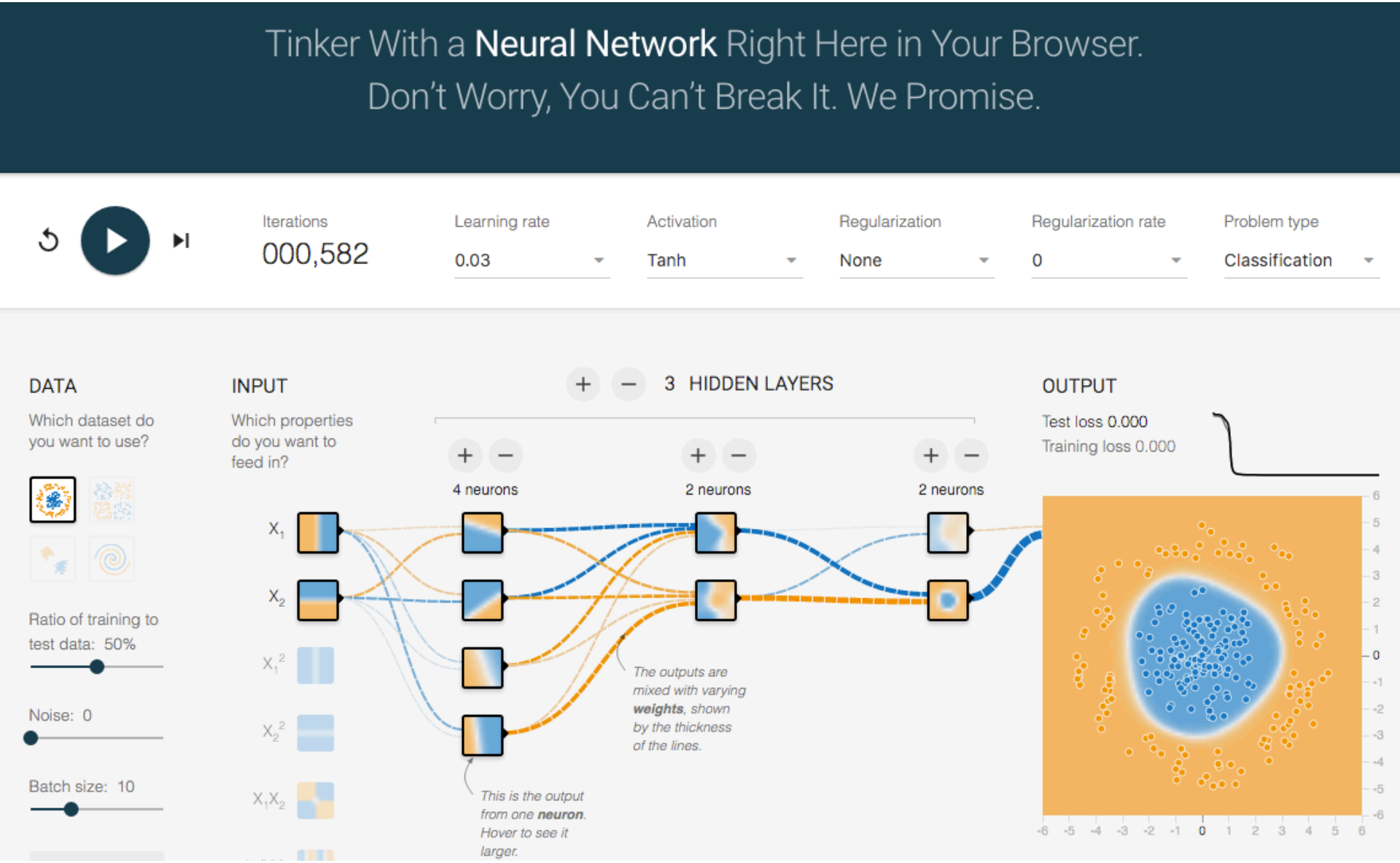


Source: https://www.tensorflow.org/

Data Flow Graph

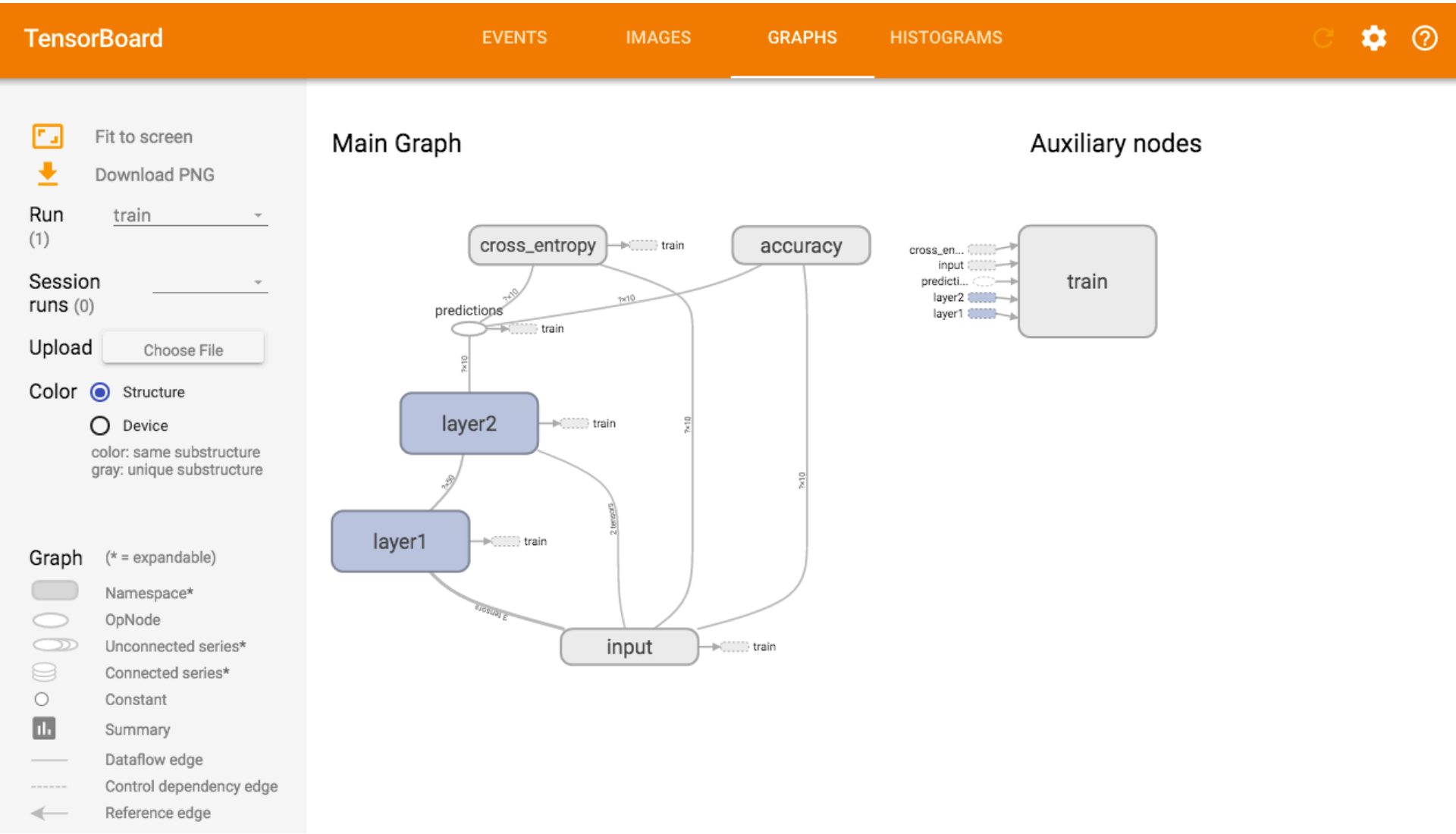


TensorFlow Playground



http://playground.tensorflow.org/

TensorBoard



https://www.tensorflow.org/tensorboard/index.html#graphs

X		Y
Hours Sleep	Hours Study	Score
3	5	75
5	1	82
10	2	93
8	3	?

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

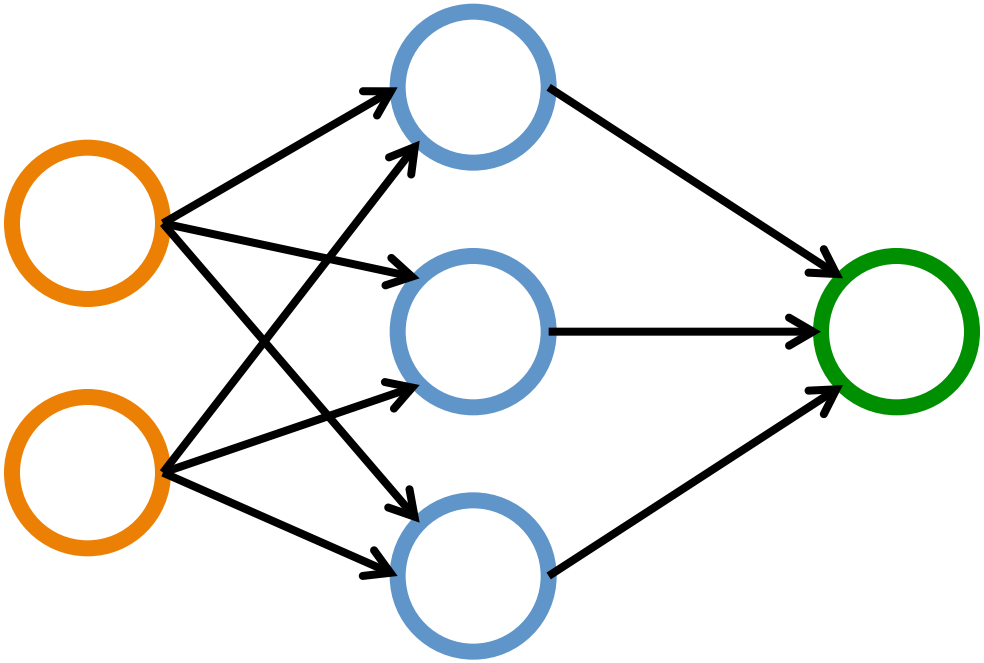
	X		Y
	Hours Sleep	Hours Study	Score
Training	3	5	75
	5	1	82
	10	2	93
Testing	8	3	?

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Training a Network
=
Minimize the Cost Function

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

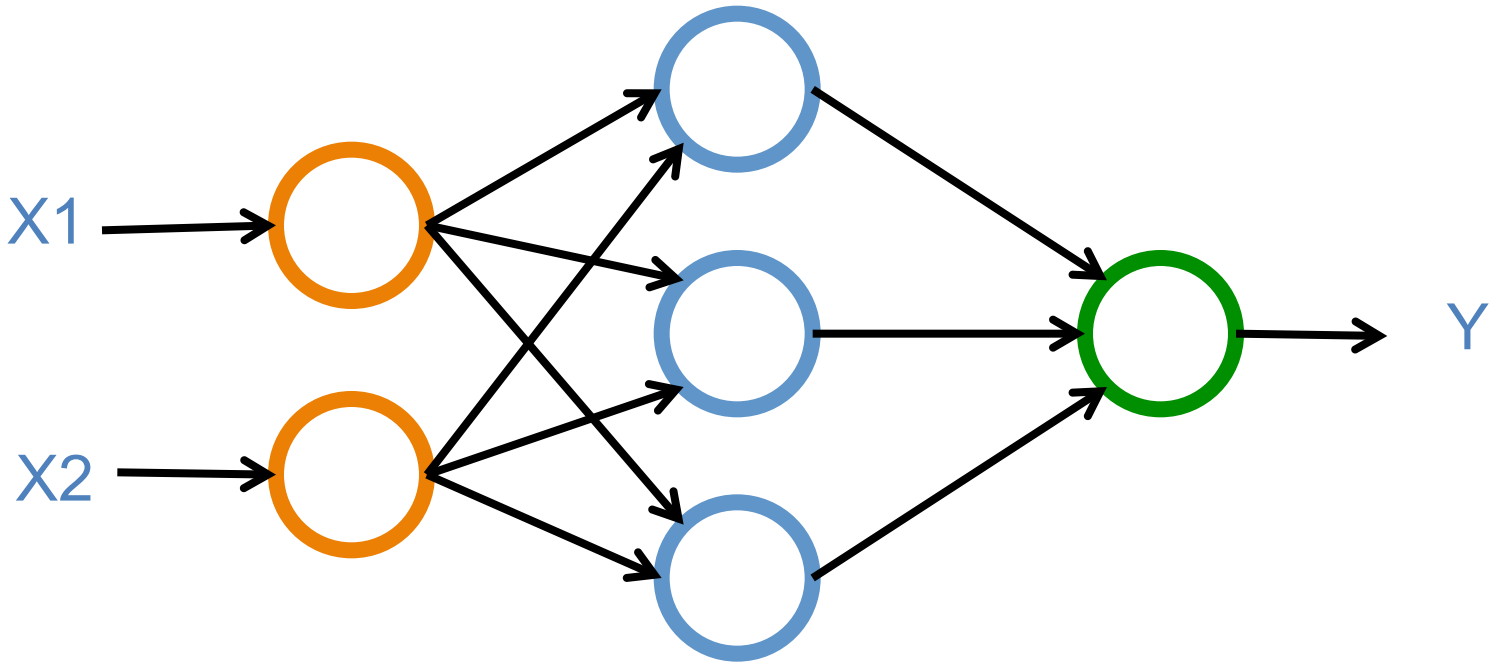
Neural Networks



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks

Input Layer (X) Hidden Layer (H) **Output Layer (Y)**



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

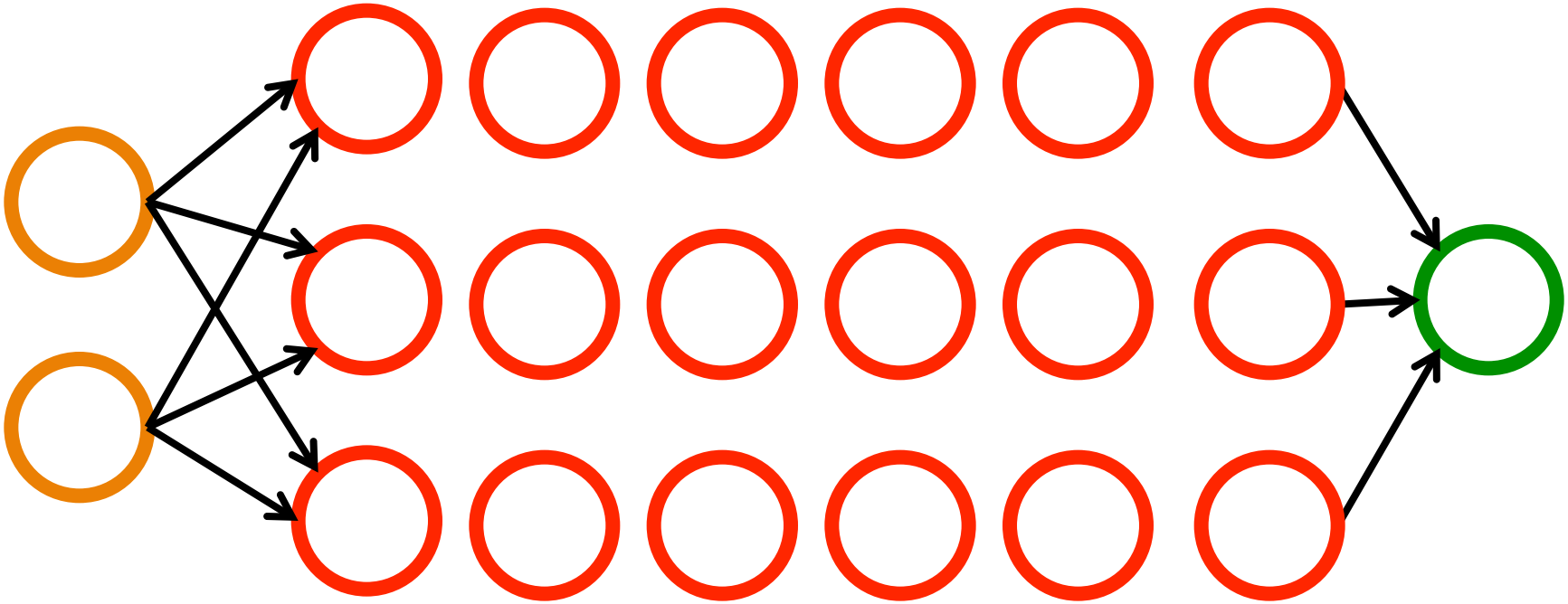
Neural Networks

Input Layer (X)

Hidden Layers (H)

Output Layer (Y)

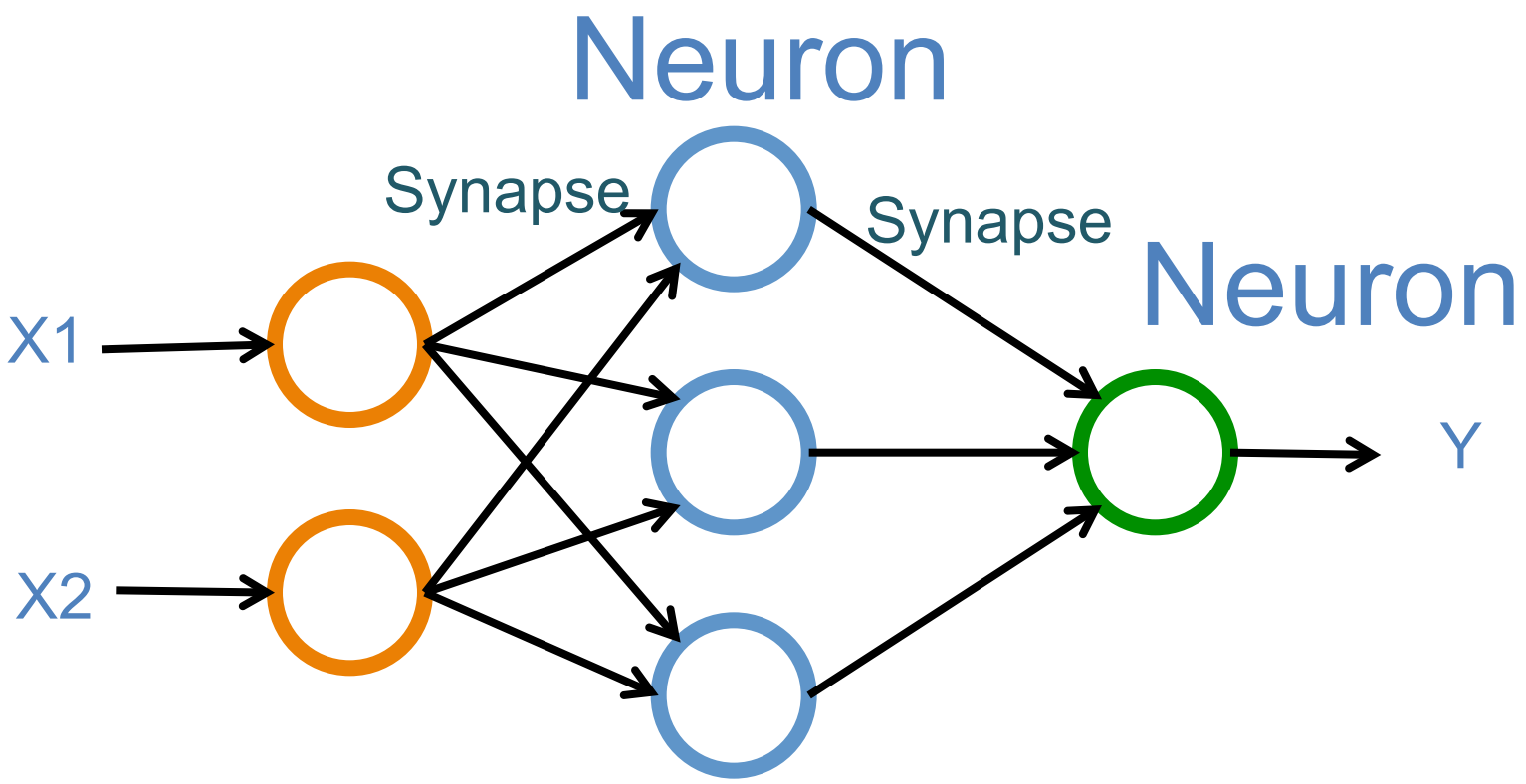
Deep Neural Networks
Deep Learning



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

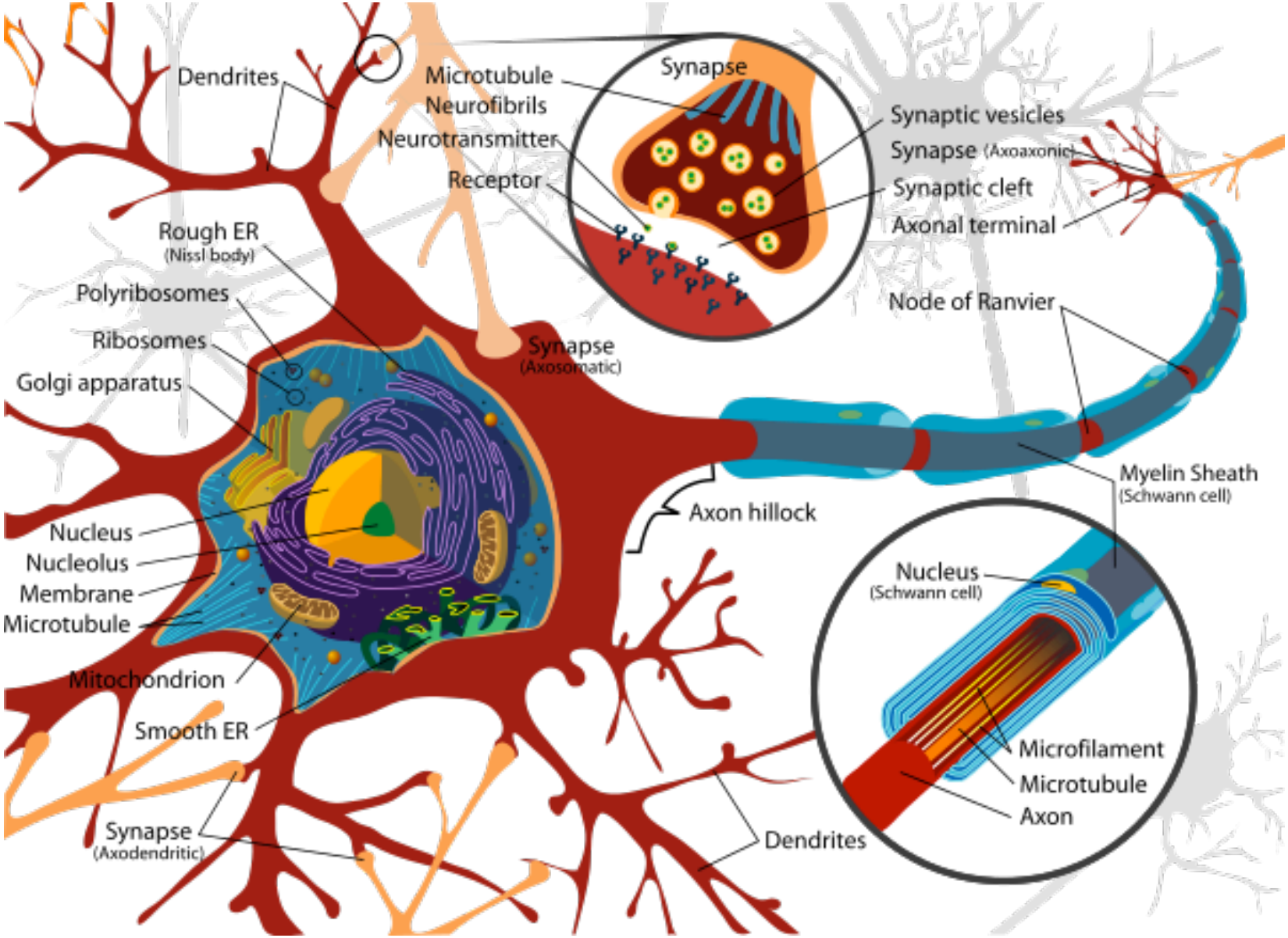
Neural Networks

Input Layer (X) Hidden Layer (H) **Output Layer (Y)**



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neuron and Synapse



Source: https://en.wikipedia.org/wiki/Neuron

Neurons

1 Unipolar neuron

2 Bipolar neuron

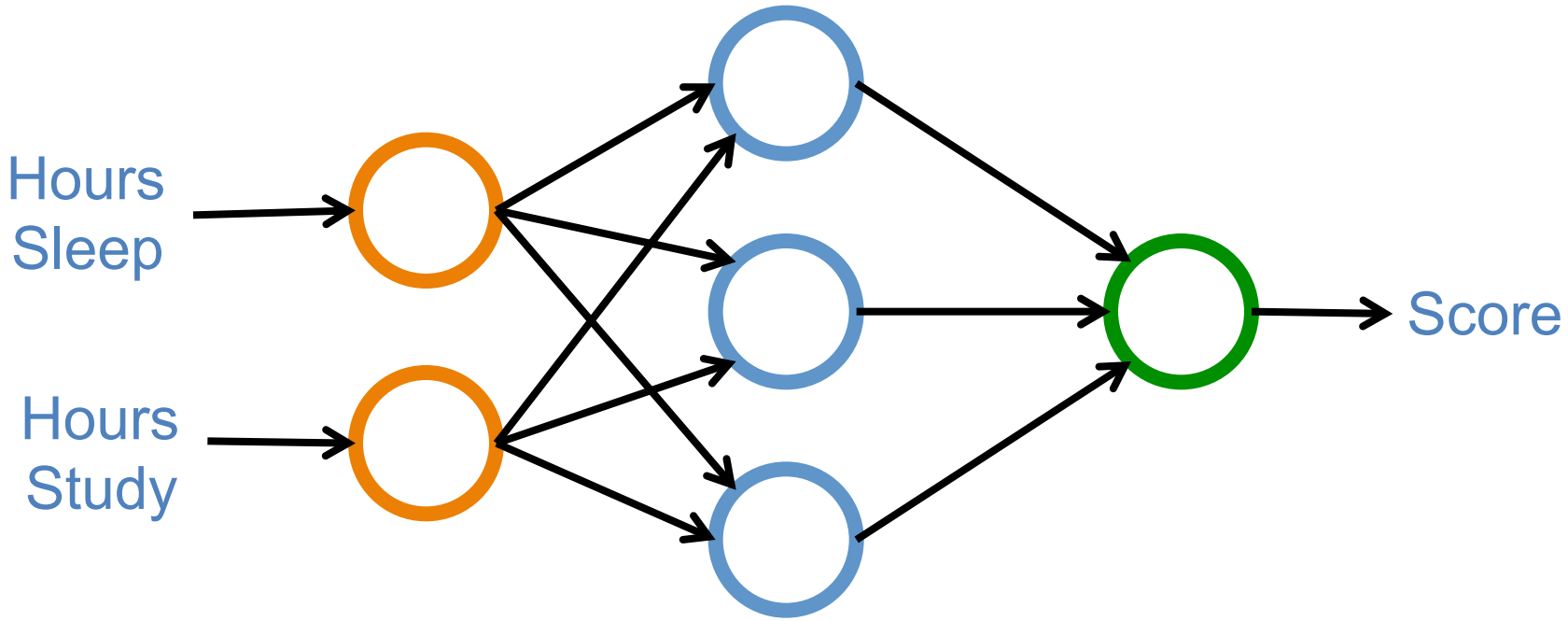
3 Multipolar neuron

4 Pseudounipolar neuron

Source: https://en.wikipedia.org/wiki/Neuron

Neural Networks

Input Layer (X) Hidden Layer (H) **Output Layer (Y)**



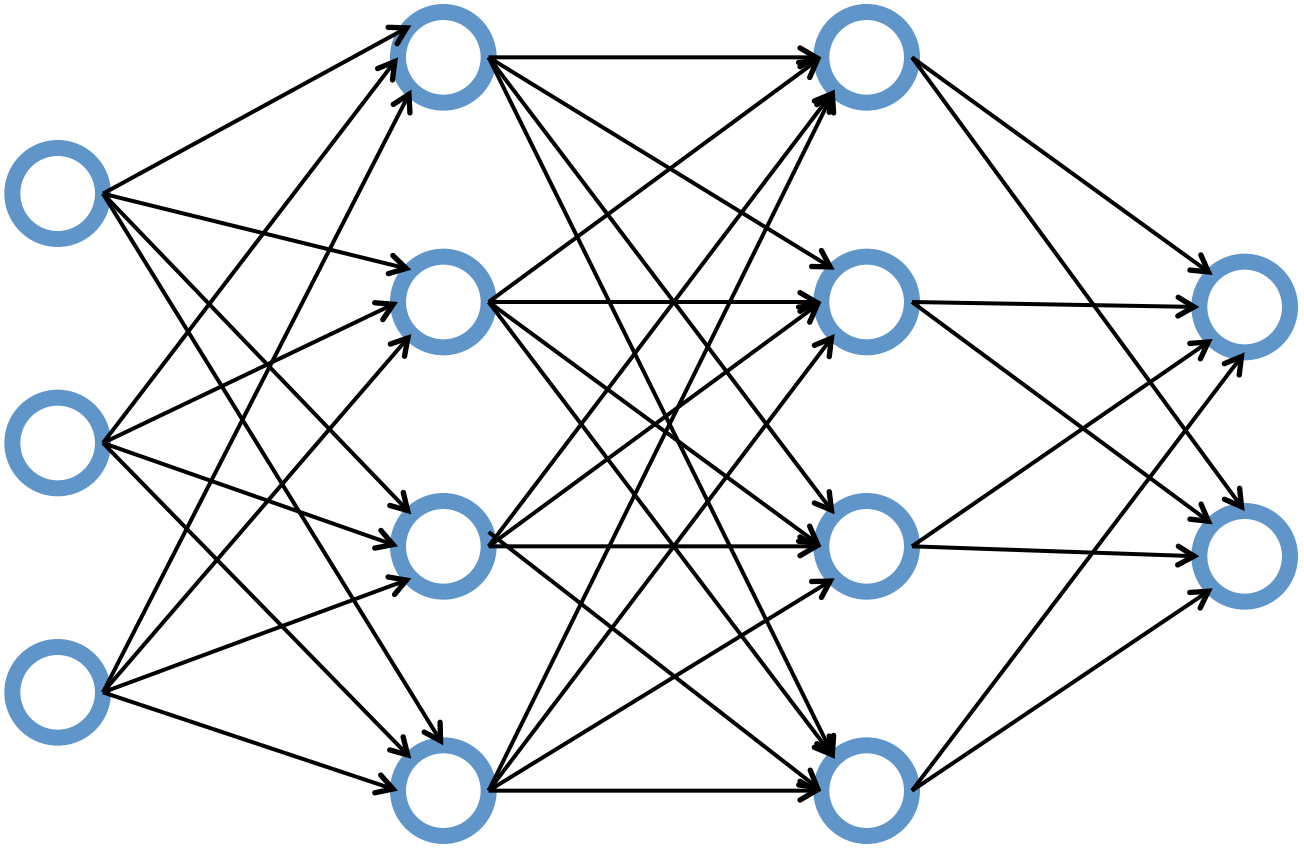
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks

Input Layer (X)

Hidden Layer (H)

Output Layer (Y)



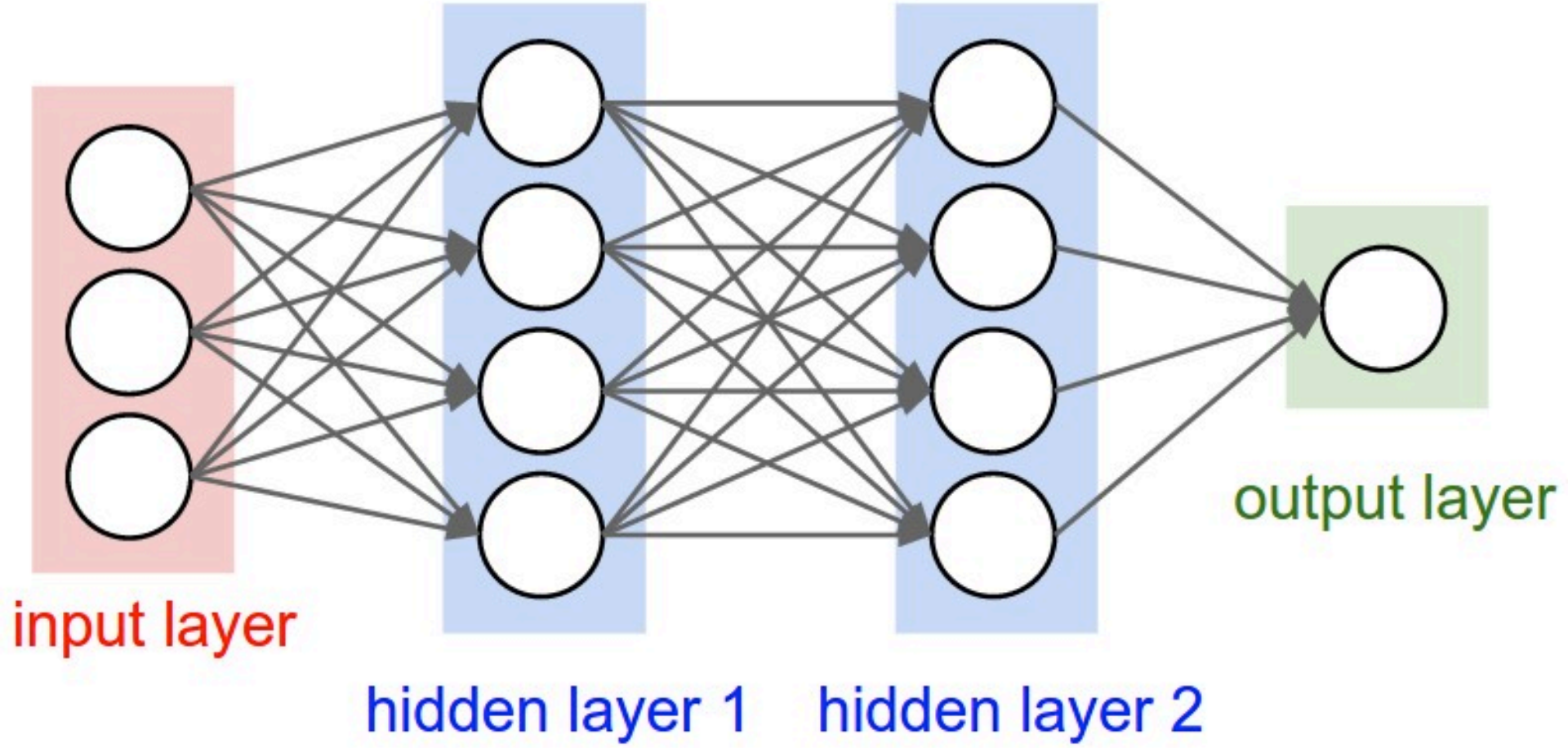
Source: https://www.youtube.com/watch?v=P2HPcj8lRJE&list=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu&index=2

Convolutional Neural Networks

(CNNs / ConvNets)

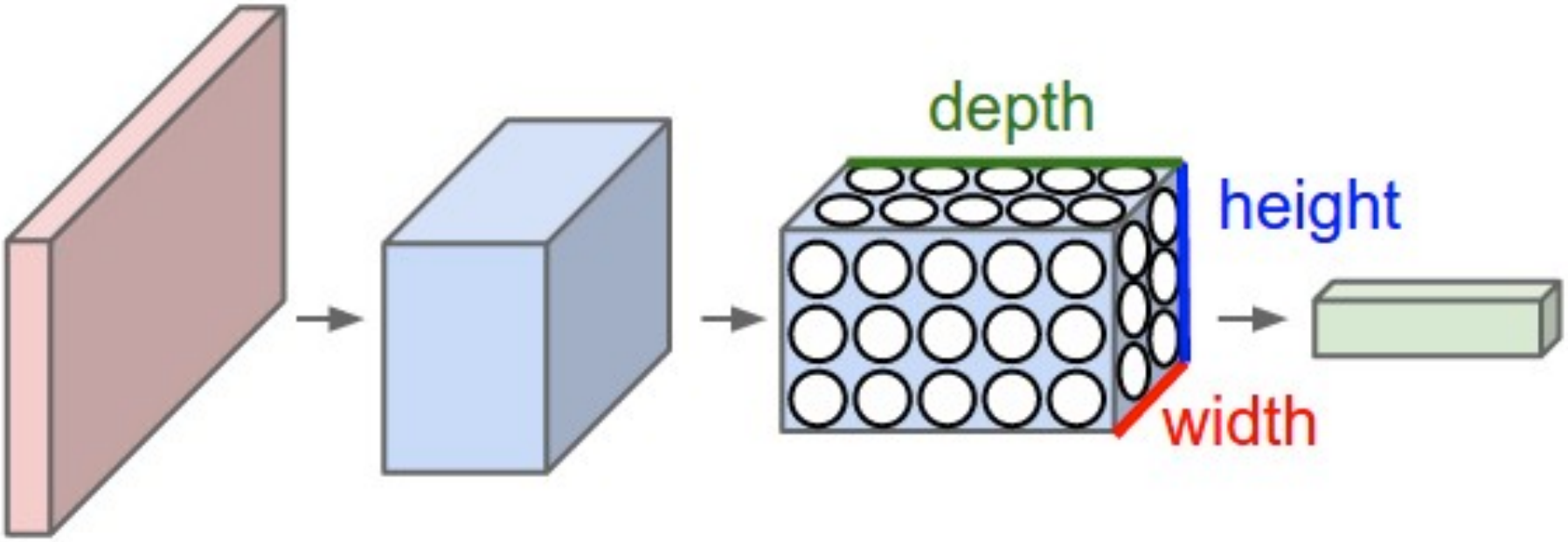
http://cs231n.github.io/convolutional-networks/

A regular 3-layer Neural Network



http://cs231n.github.io/convolutional-networks/

A ConvNet arranges its neurons in three dimensions (width, height, depth)

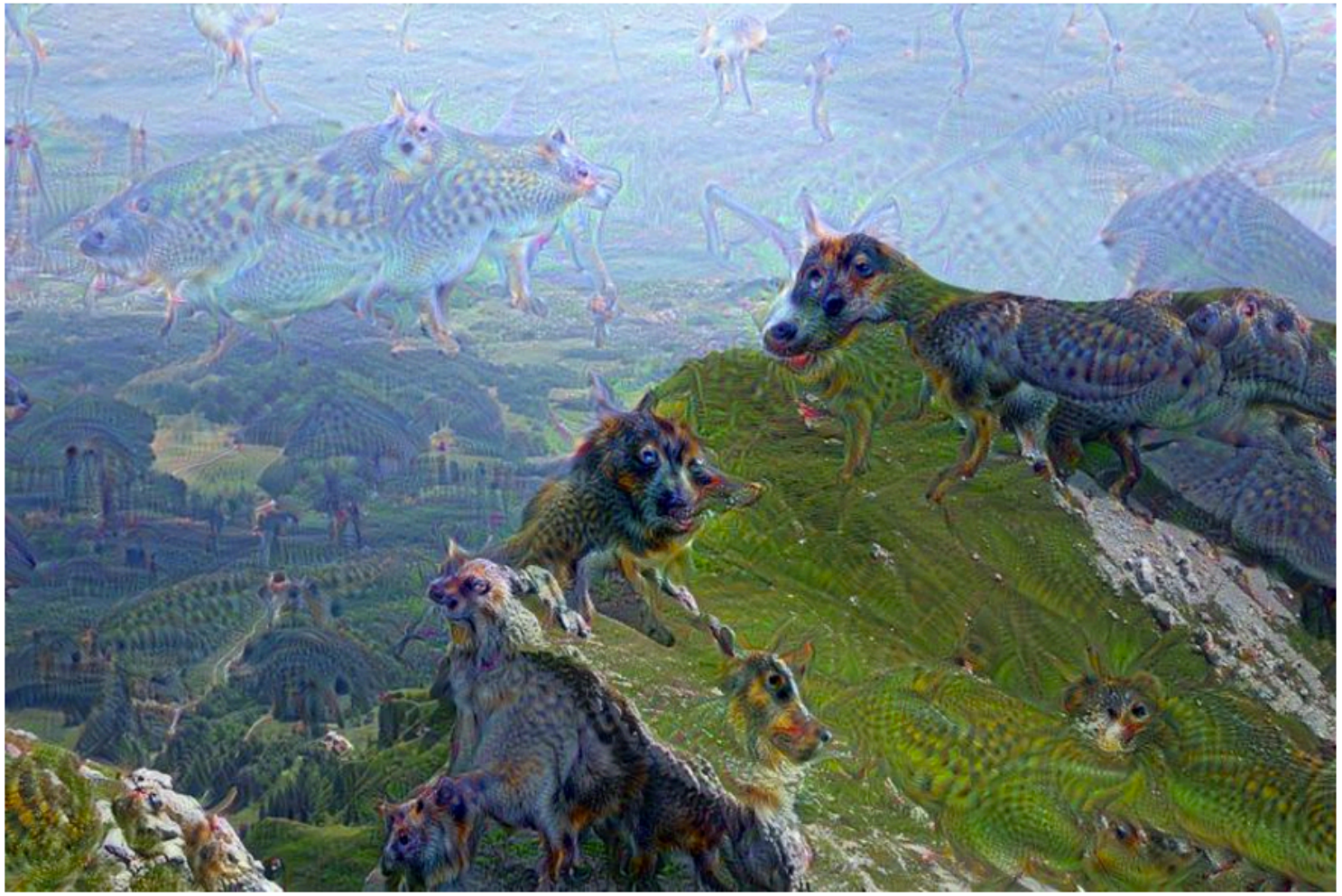


http://cs231n.github.io/convolutional-networks/

DeepDream

GitHub, Inc. [US] https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/deepdream/deepdream.ipynb

In [15]:
```
render_deepdream(tf.square(T('mixed4c')), img0)
```



Note that results can differ from the Caffe's implementation, as we are using an independently trained network. Still, the network seems to like dogs and animal-like features due to the nature of the ImageNet dataset.

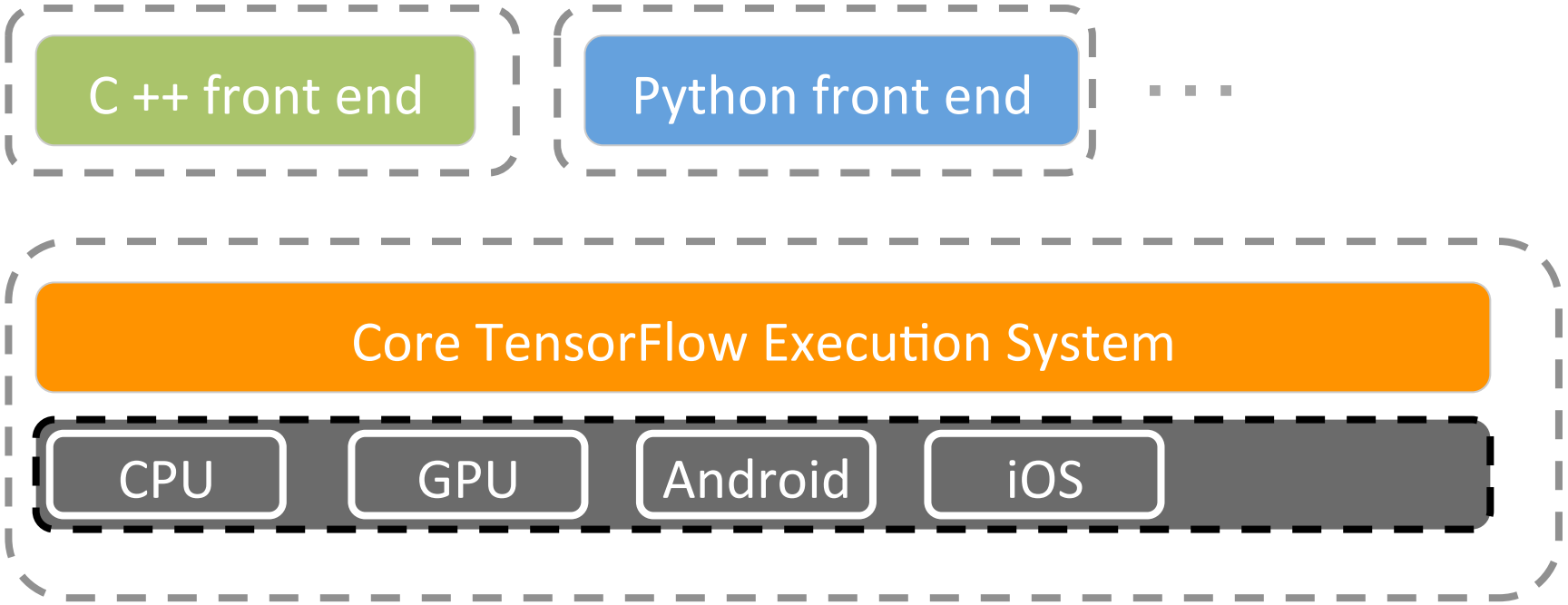
Source: https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/deepdream/deepdream.ipynb

Try your first TensorFlow

```
$ python

>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> sess.run(hello)
Hello, TensorFlow!
>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> sess.run(a+b)
42
>>>
```

https://github.com/tensorflow/tensorflow

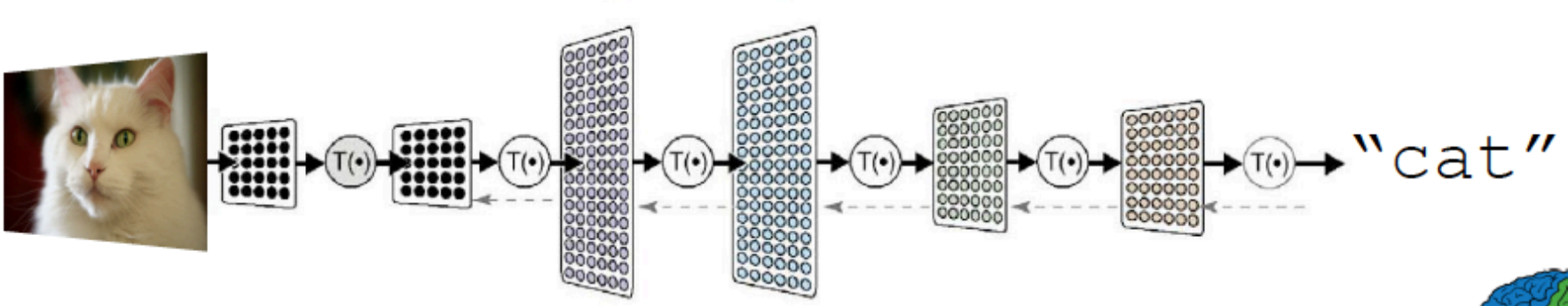
Architecture of TensorFlow



Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

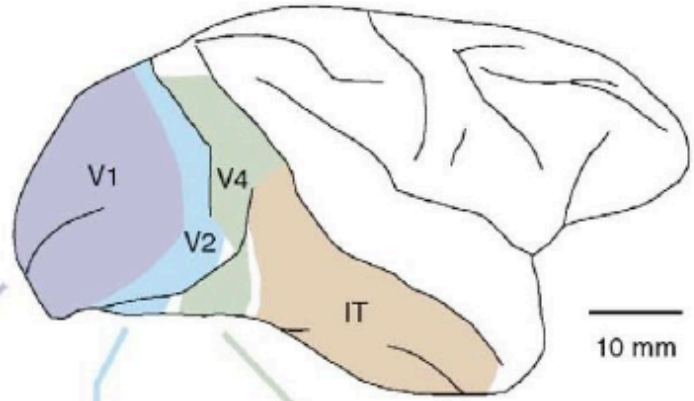
Deep Learning

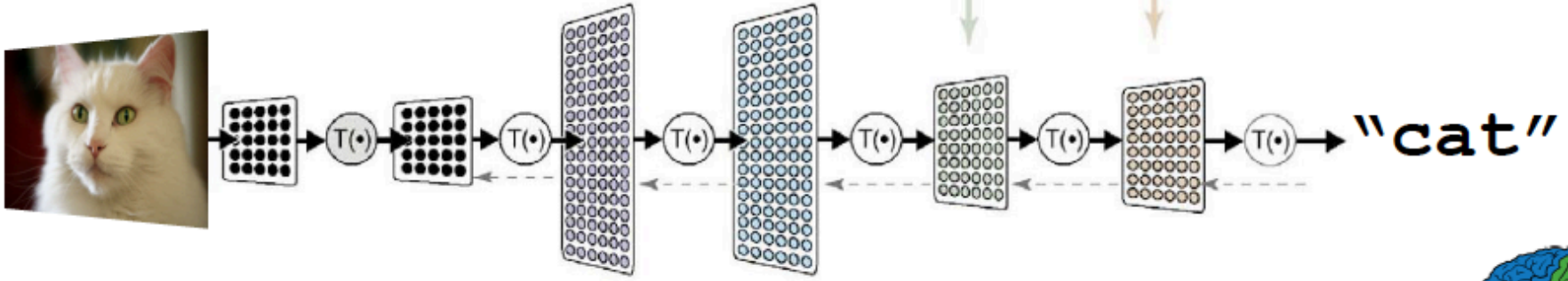
- A powerful class of machine learning model
- Modern reincarnation of artificial neural networks
- Collection of simple,
  trainable mathematical functions
- Compatible with many variants of machine learning



What is Deep Learning?

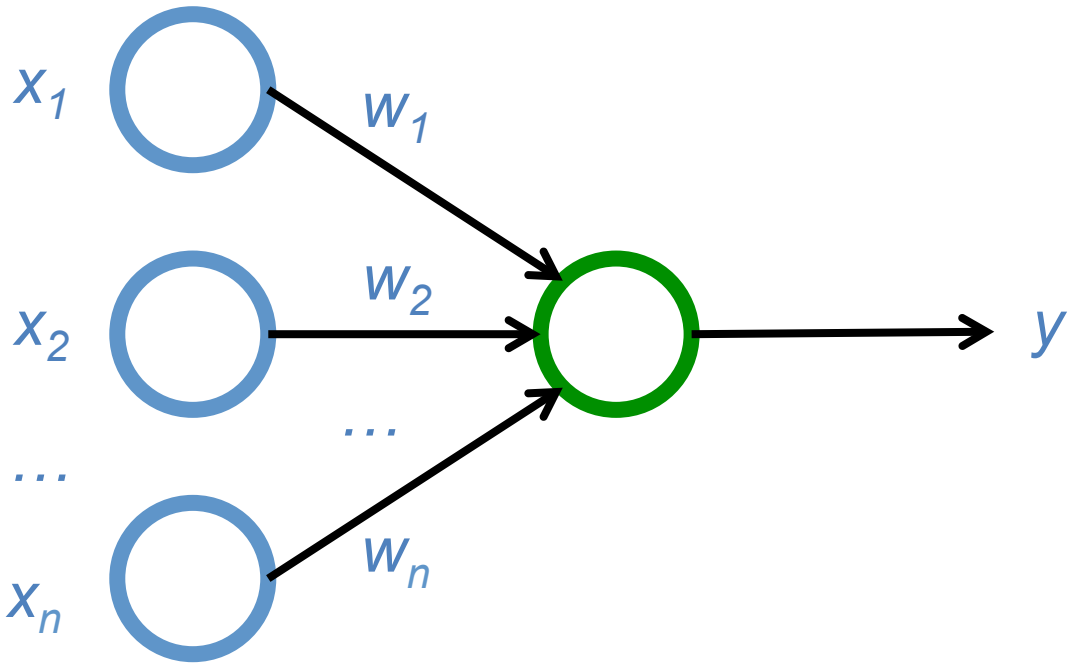
- Loosely based on (what little) we know about the brain





Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

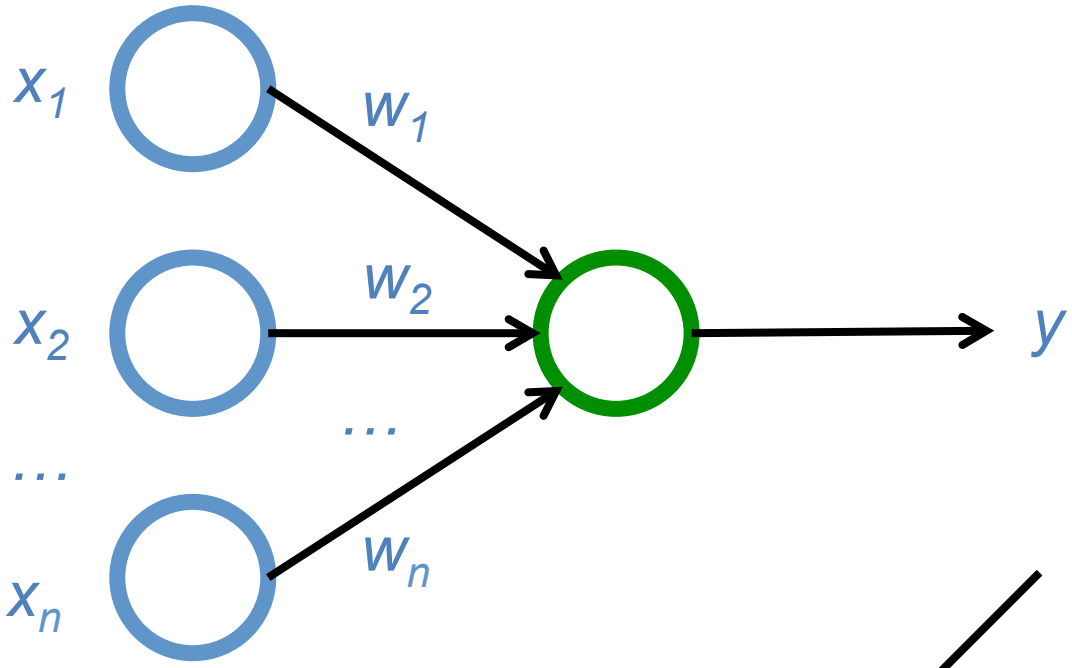
The Neuron

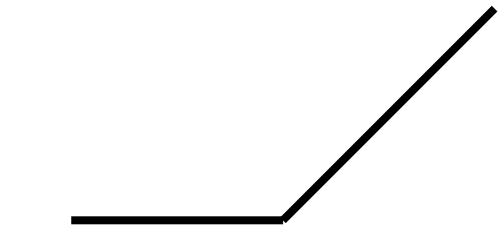


Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

The Neuron

$$y = F\left(\sum_i w_i x_i\right)$$

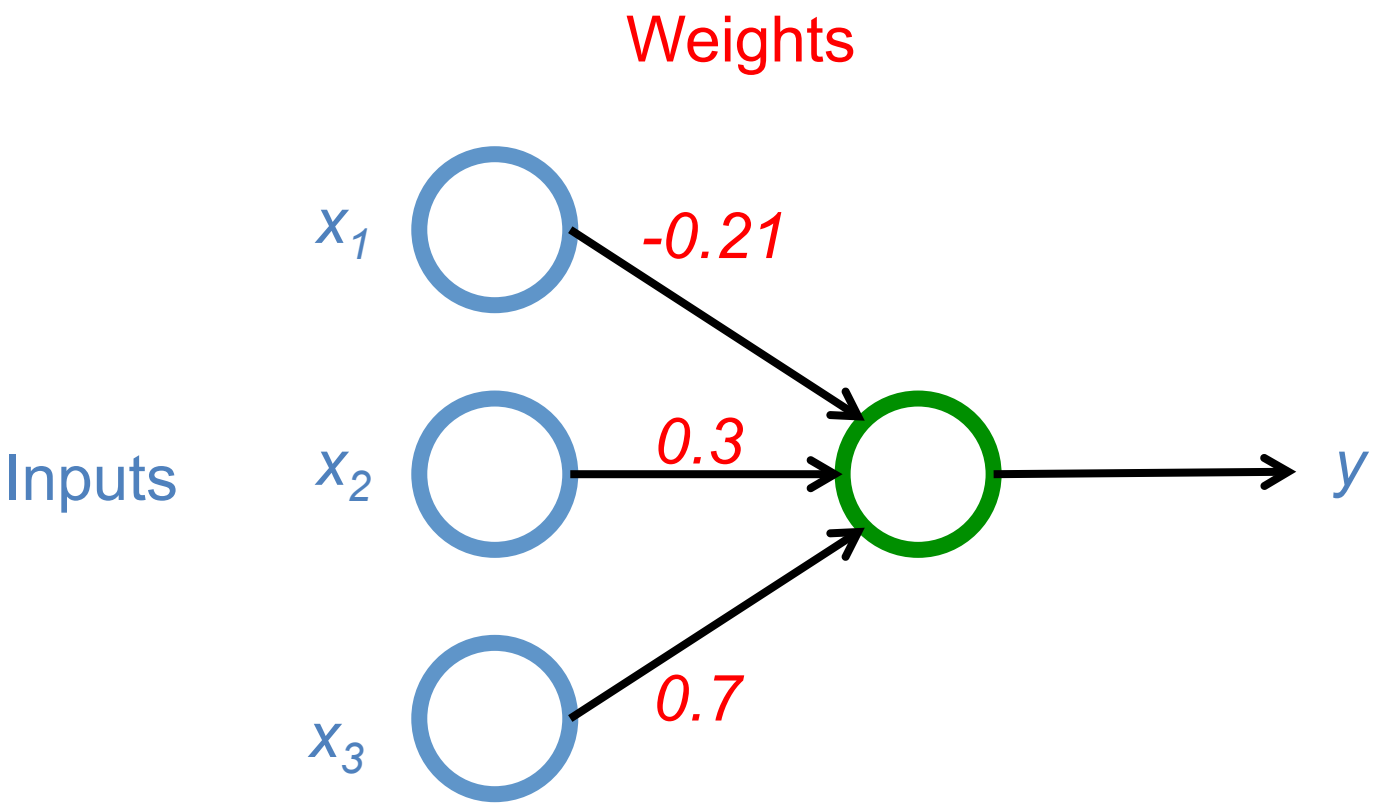




$$F(x) = \max(0, x)$$

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

$$y = max\,(\,0,\ -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3\,)$$



IS THIS A
CAT or DOG?

CAT DOG

OUTPUT LAYER

ACTIVATED NEURONS

INPUT LAYER

DEEP *neural* NETWORK

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Learning Algorithm

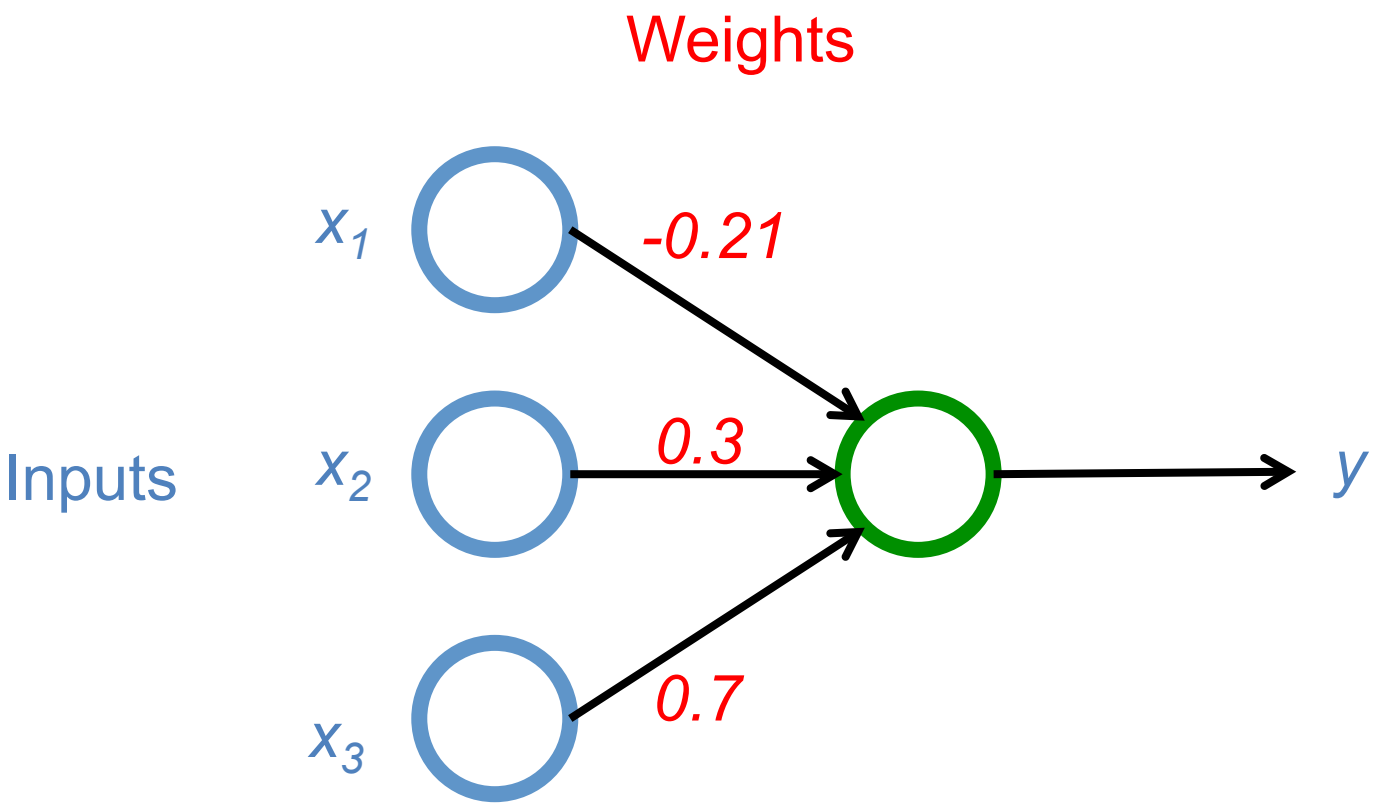
While not done:

Pick a random training example “(input, label)”

Run neural network on “input”

Adjust weights on edges to make output closer to “label”

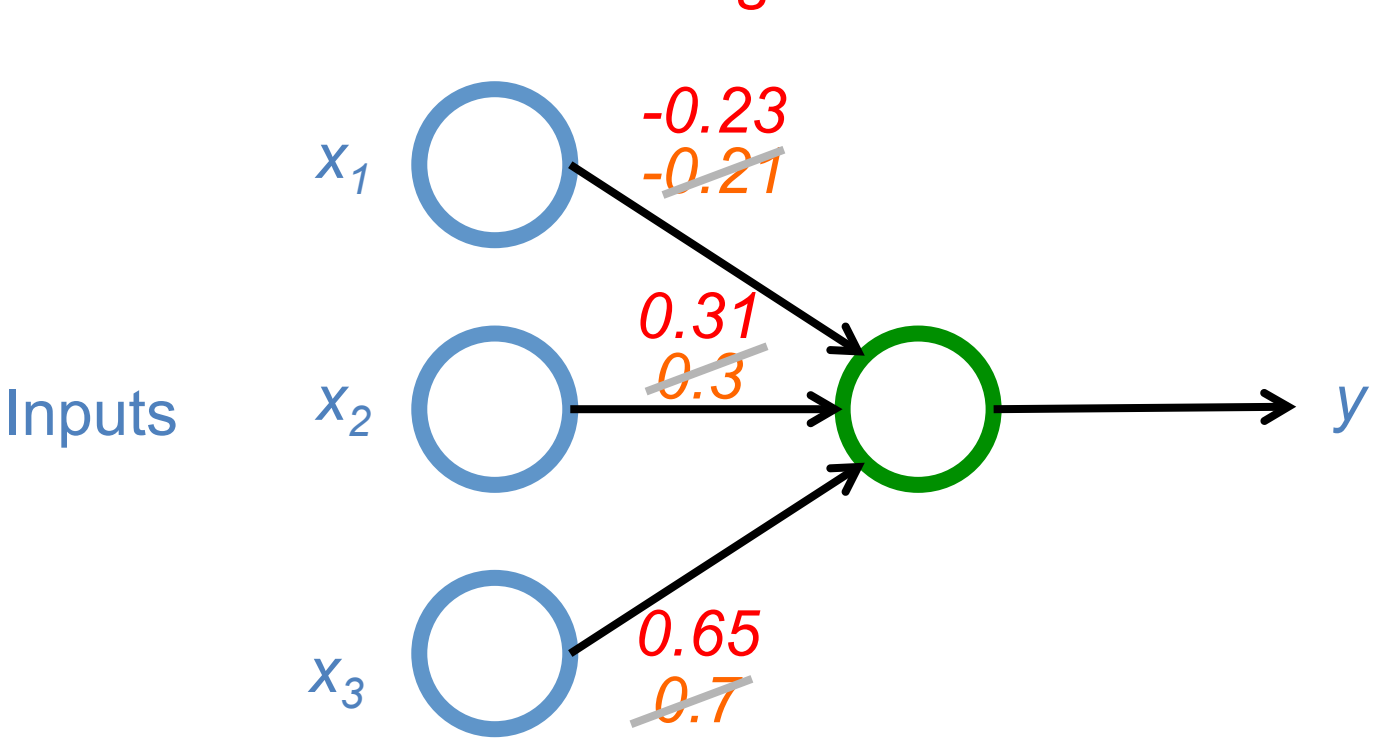
$$y = max\,(\,0,\ -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3\,)$$

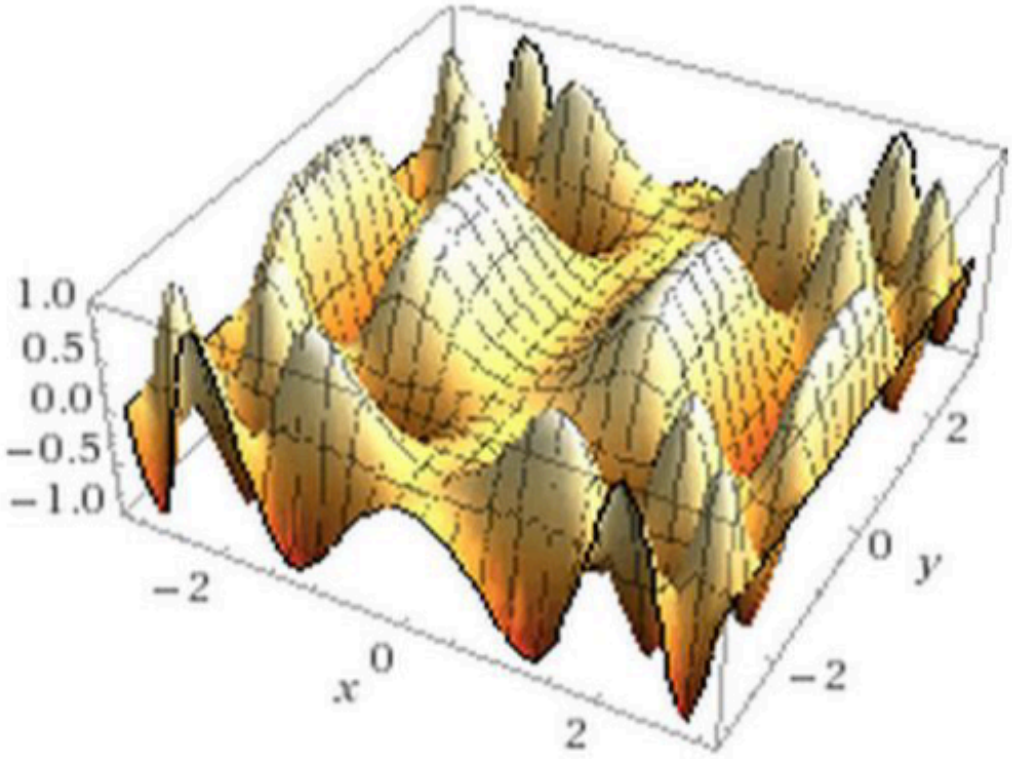


Next time:

$y = max(0, -0.23 * x_1 + 0.31 * x_2 + 0.65 * x_3)$

~~$y = max(0, -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3)$~~





This shows a function of 2 variables: real neural nets are functions of hundreds of millions of variables!

Important Property of Neural Networks

Results get better with

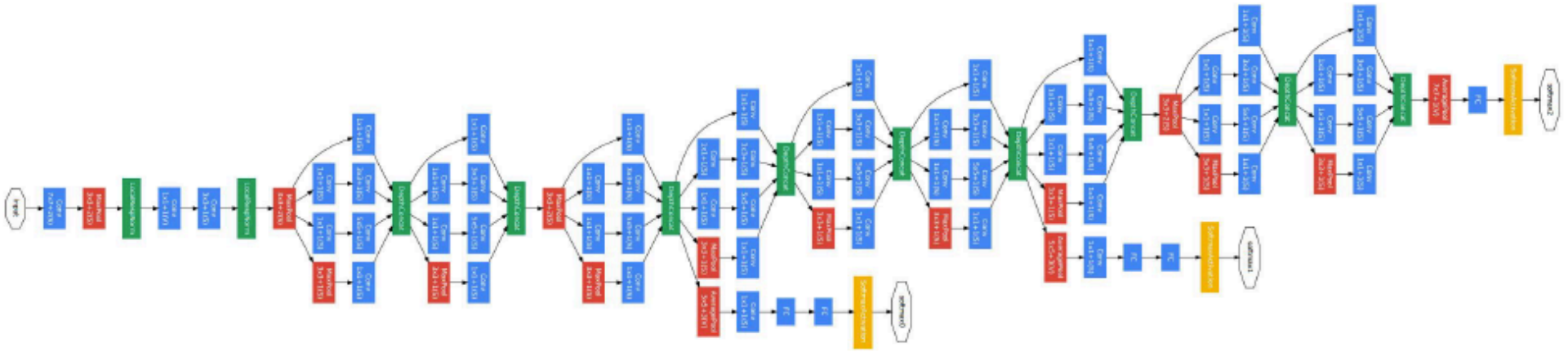
More data +

Bigger models +

More computation

(Better algorithms, new insights
and improved techniques always help, too!)

The Inception Architecture (GoogLeNet, 2014)



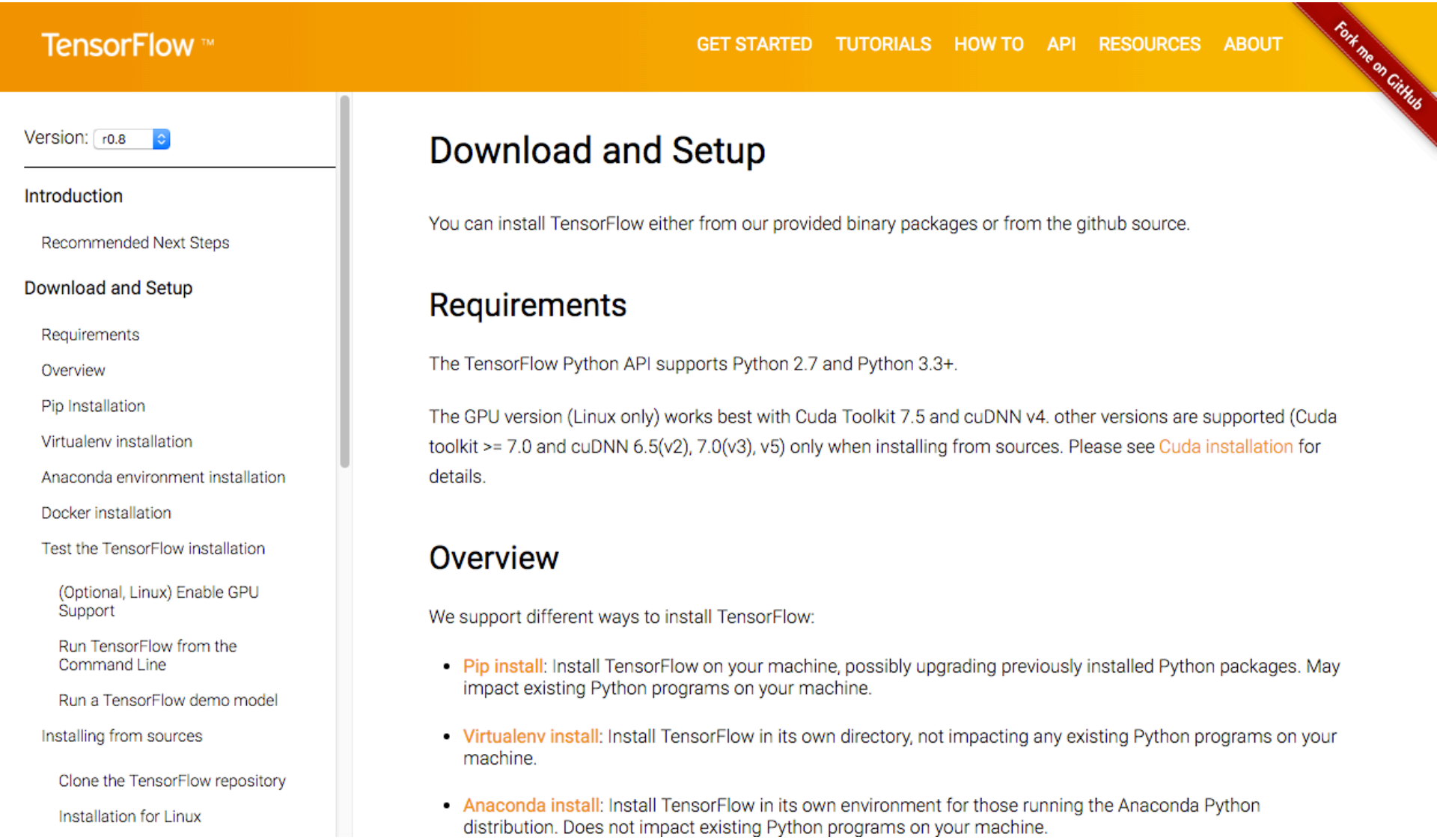
Going Deeper with Convolutions

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich

ArXiv 2014, CVPR 2015

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Install TensorFlow



https://www.tensorflow.org/versions/r0.8/get_started/os_setup.html

Install TensorFlow

TensorFlow ™ GET STARTED TUTORIALS HOW TO API RESOURCES ABOUT Fork me on GitHub

Version: r0.8

- Introduction
  - Recommended Next Steps
- Download and Setup
  - Requirements
  - Overview
  - Pip Installation
  - Virtualenv installation
  - Anaconda environment installation
  - Docker installation
  - Test the TensorFlow installation
    - (Optional, Linux) Enable GPU Support
    - Run TensorFlow from the Command Line
    - Run a TensorFlow demo model
  - Installing from sources
    - Clone the TensorFlow repository
    - Installation for Linux

Follow the instructions on the Anaconda download site

Create a conda environment called `tensorflow`:

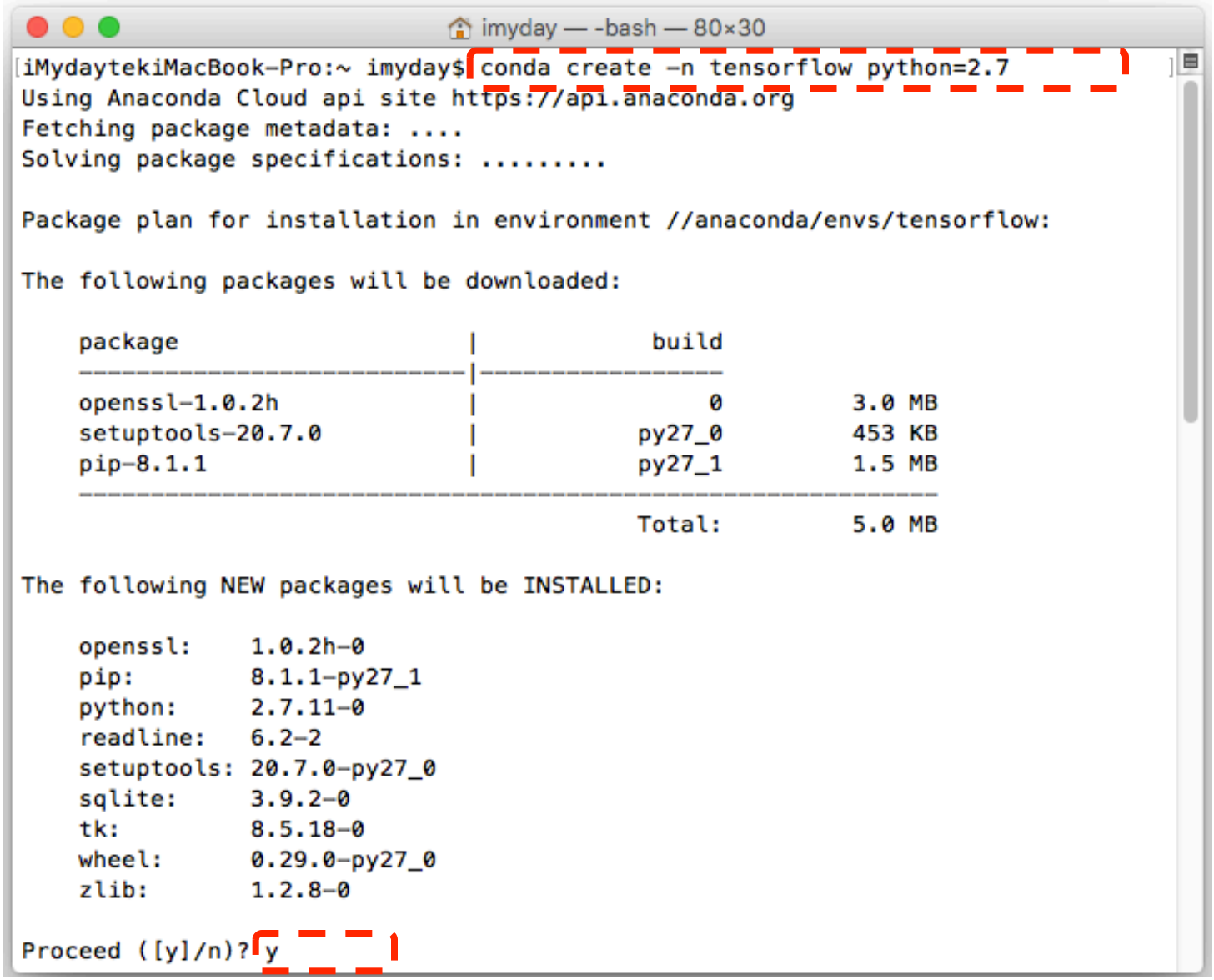
```
# Python 2.7
$ conda create -n tensorflow python=2.7

# Python 3.5
$ conda create -n tensorflow python=3.5
```

Activate the environment and use pip to install TensorFlow inside it. Use the `--ignore-installed` flag to prevent errors about `easy_install`.

```
$ source activate tensorflow
(tensorflow)$  # Your prompt should change

# Ubuntu/Linux 64-bit, CPU only:
(tensorflow)$ pip install --ignore-installed --upgrade https://storage.googleapis.com/tensor

# Ubuntu/Linux 64-bit, GPU enabled. Requires CUDA toolkit 7.5 and CuDNN v4.  For
# other versions, see "Install from sources" below.
(tensorflow)$ pip install --ignore-installed --upgrade https://storage.googleapis.com/tensor

# Mac OS X, CPU only:
(tensorflow)$ pip install --ignore-installed --upgrade https://storage.googleapis.com/tensor
```

https://www.tensorflow.org/versions/r0.8/get_started/os_setup.html#anaconda-environment-installation

conda create -n tensorflow python=2.7



Source: https://www.tensorflow.org/versions/r0.8/get_started/os_setup.html

source activate tensorflow

imyday — -bash — 80×20

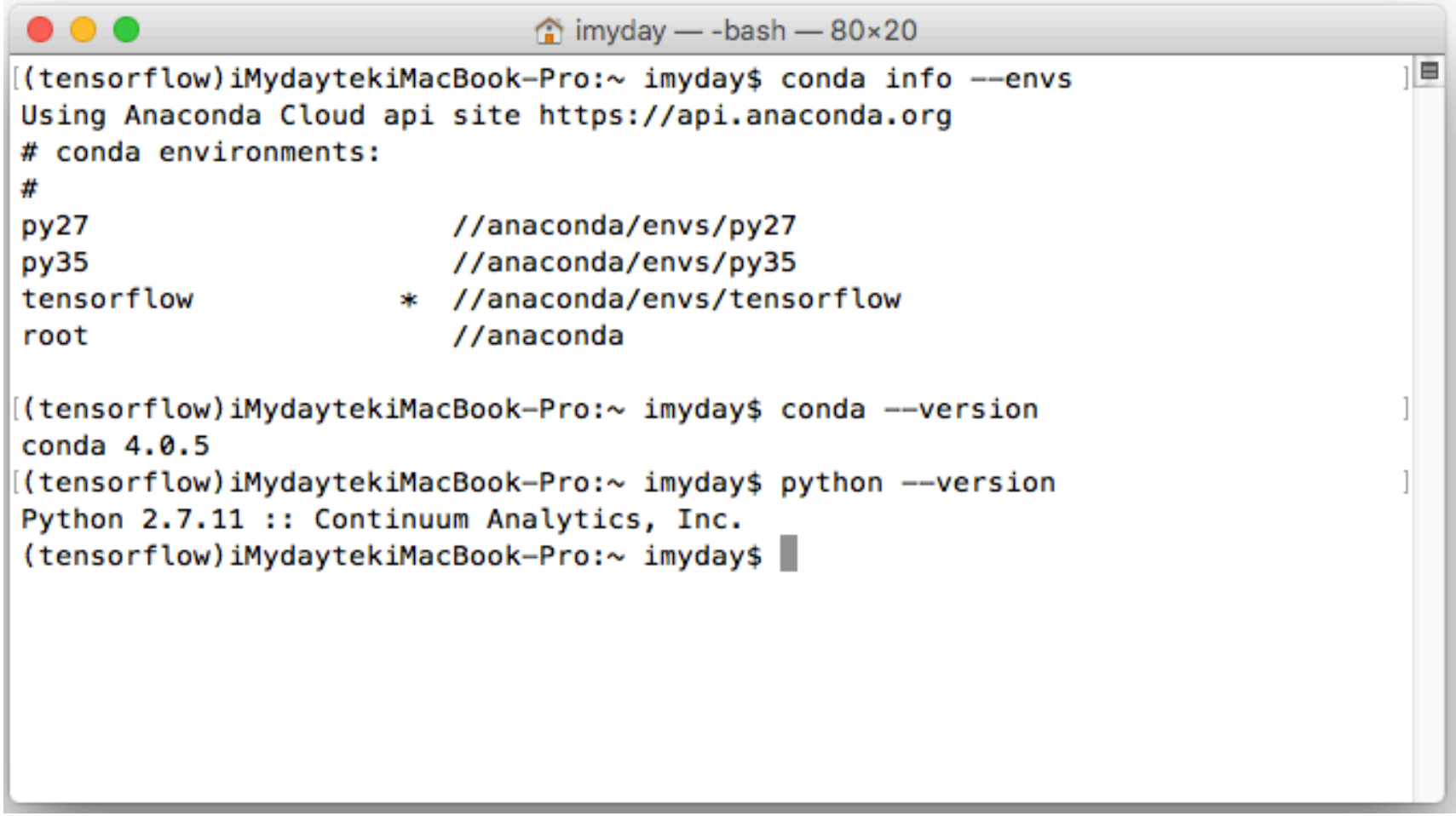
```
Proceed ([y]/n)? y

Fetching packages ...
openssl-1.0.2h 100% |################################| Time: 0:00:04 778.75 kB/s
setuptools-20. 100% |################################| Time: 0:00:05  79.42 kB/s
pip-8.1.1-py27 100% |################################| Time: 0:00:07 216.54 kB/s
Extracting packages ...
[      COMPLETE      ]|##################################################| 100%
Linking packages ...
[      COMPLETE      ]|##################################################| 100%
#
# To activate this environment, use:
# $ source activate tensorflow
#
# To deactivate this environment, use:
# $ source deactivate
#
iMydaytekiMacBook-Pro:~ imyday$ source activate tensorflow
discarding //anaconda/bin from PATH
prepending //anaconda/envs/tensorflow/bin to PATH
```

Source: https://www.tensorflow.org/versions/r0.8/get_started/os_setup.html

```
pip install --ignore-installed --upgrade
https://storage.googleapis.com/tensorflow/
mac/tensorflow-0.8.0-py2-none-any.whl
```

Source: https://www.tensorflow.org/versions/r0.8/get_started/os_setup.html

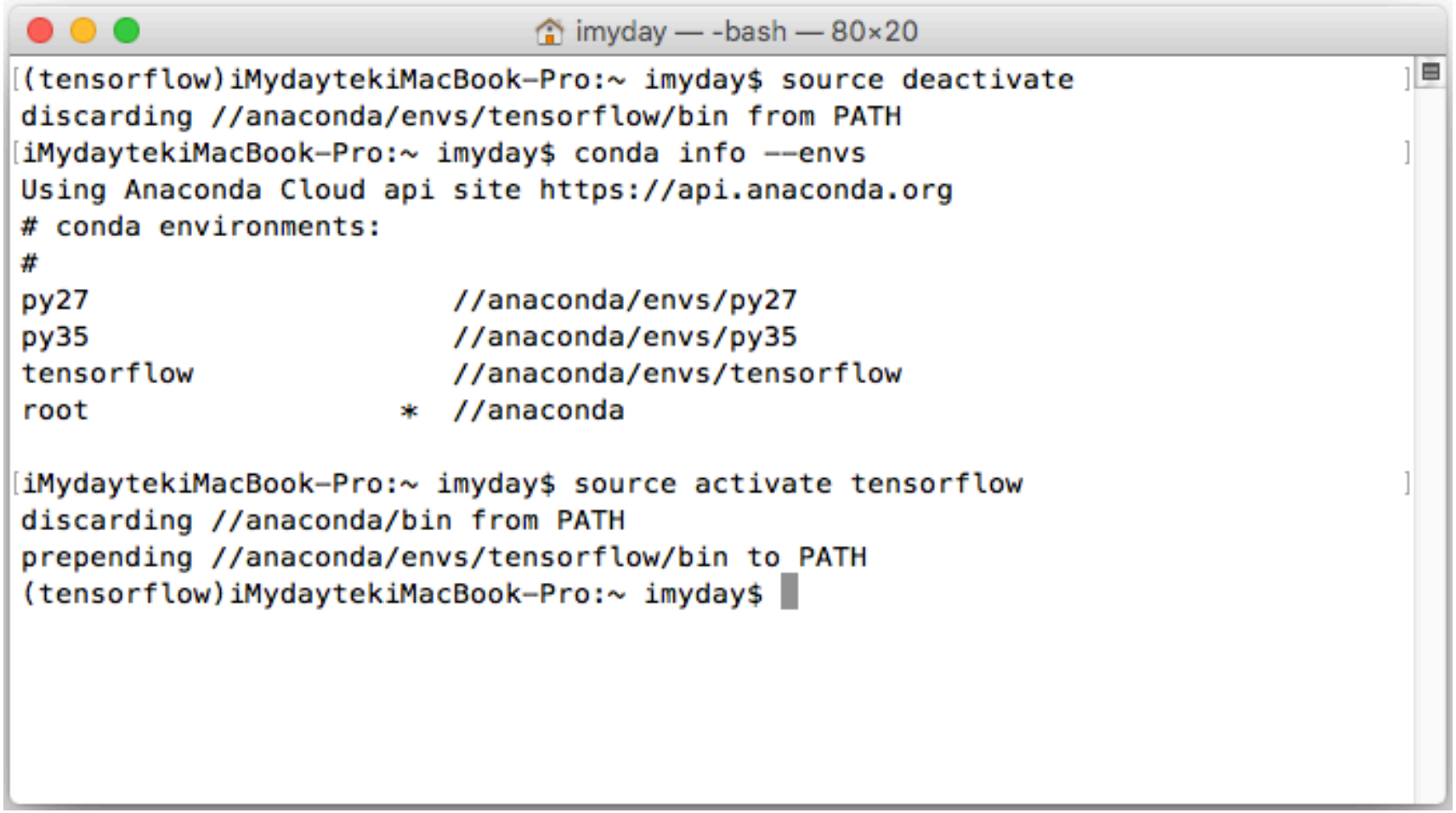
```
conda info --envs
conda --version
python --version
```



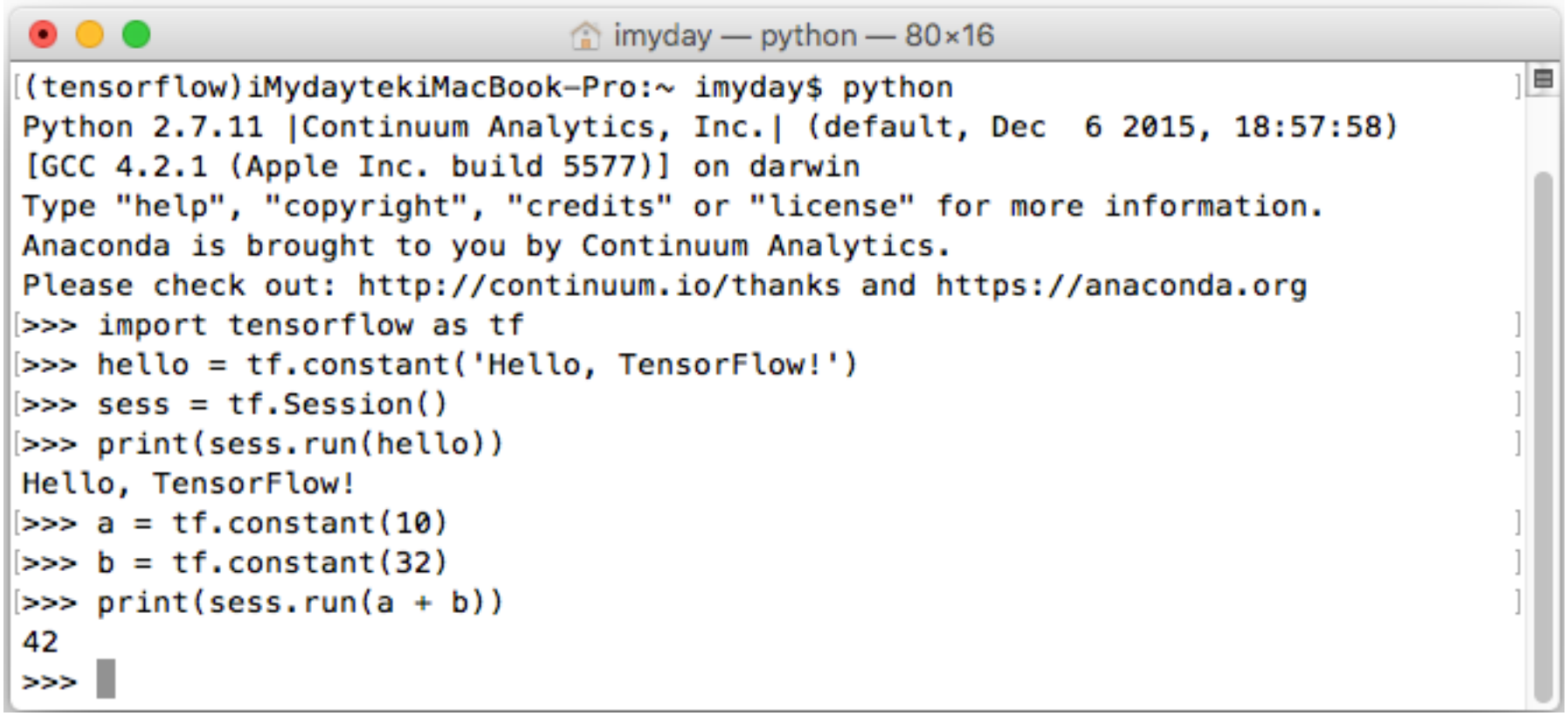
source deactivate

conda info --envs

source activate tensorflow

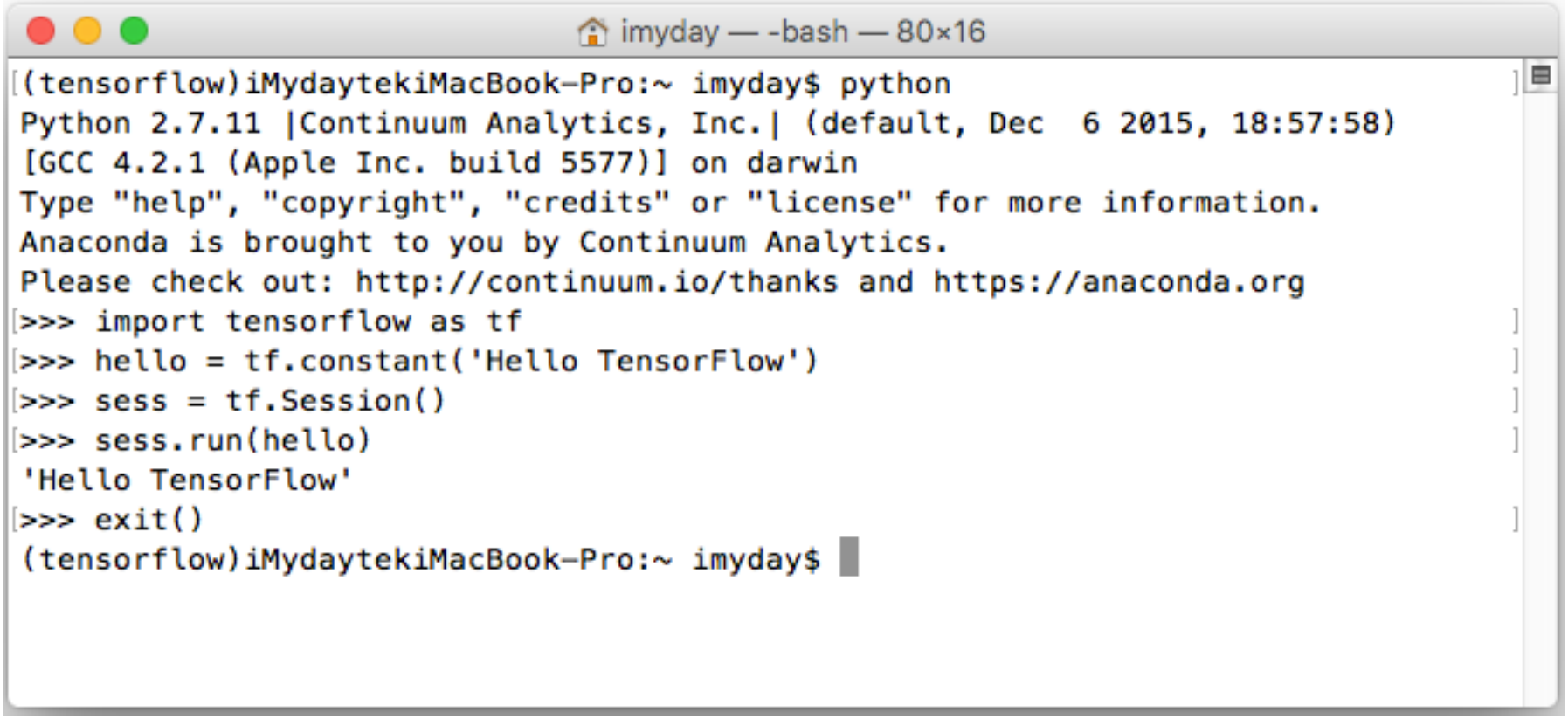


```
$ python
...
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> print(sess.run(hello))
Hello, TensorFlow!
```



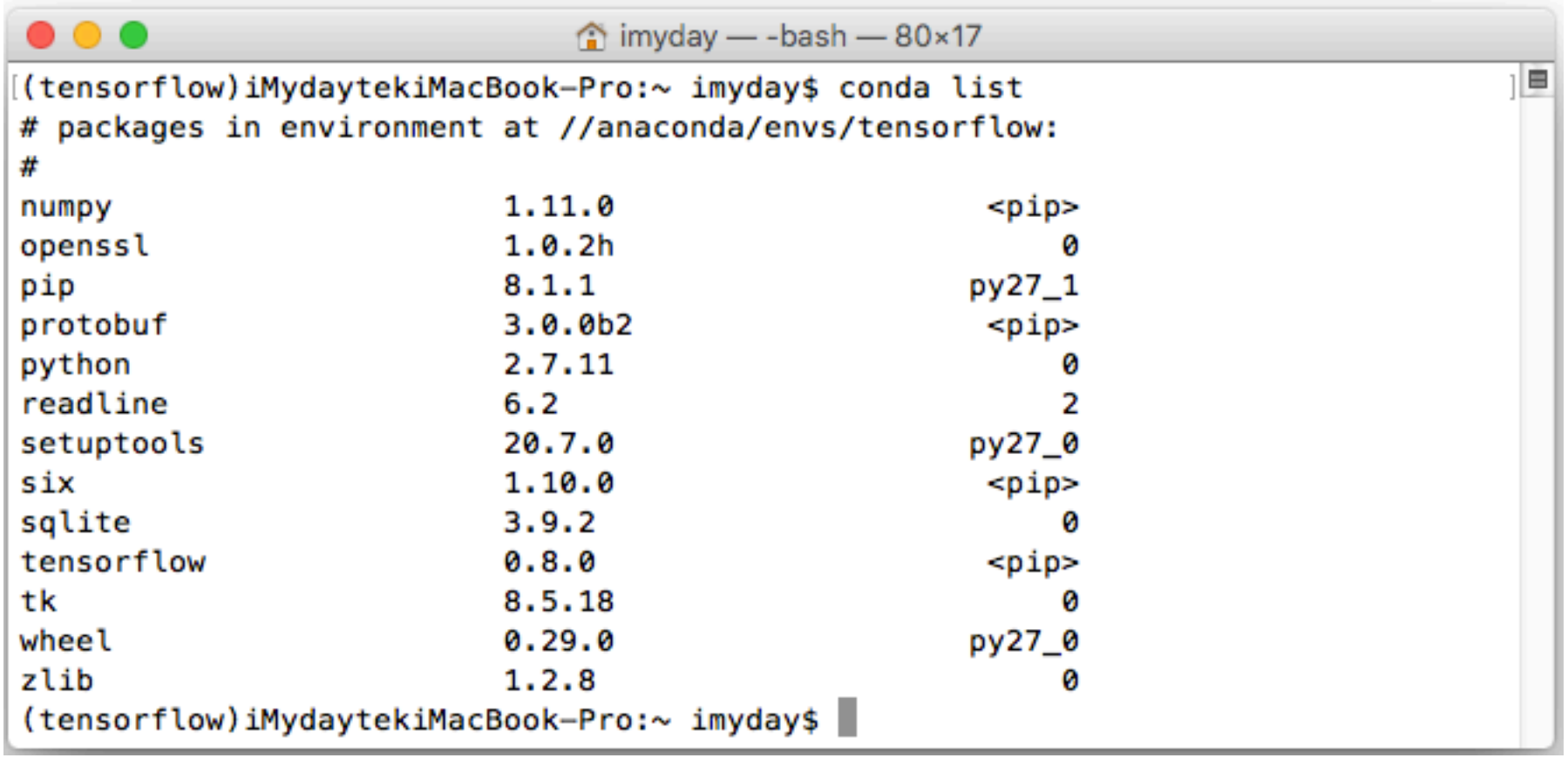
Source: https://www.tensorflow.org/versions/r0.8/get_started/os_setup.html#test-the-tensorflow-installation

```
$ python
>>> import tensorflow as tf
>>> hello = tf.constant('Hello TensorFlow')
>>> sess = tf.Session()
>>> sess.run(hello)
'Hello TensorFlow'
>>> exit()
$
```

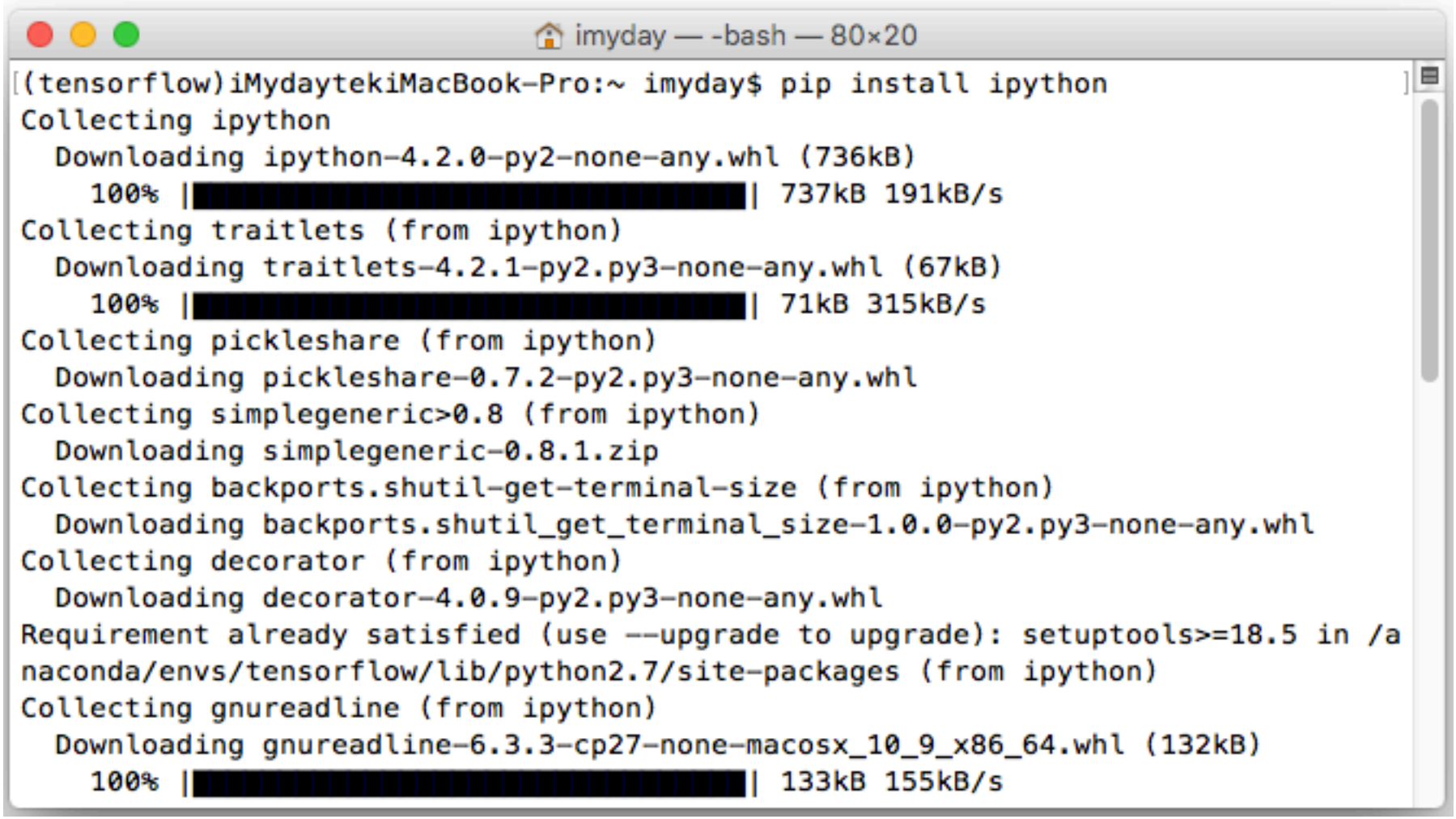


Source: https://www.tensorflow.org/versions/r0.8/get_started/os_setup.html#test-the-tensorflow-installation

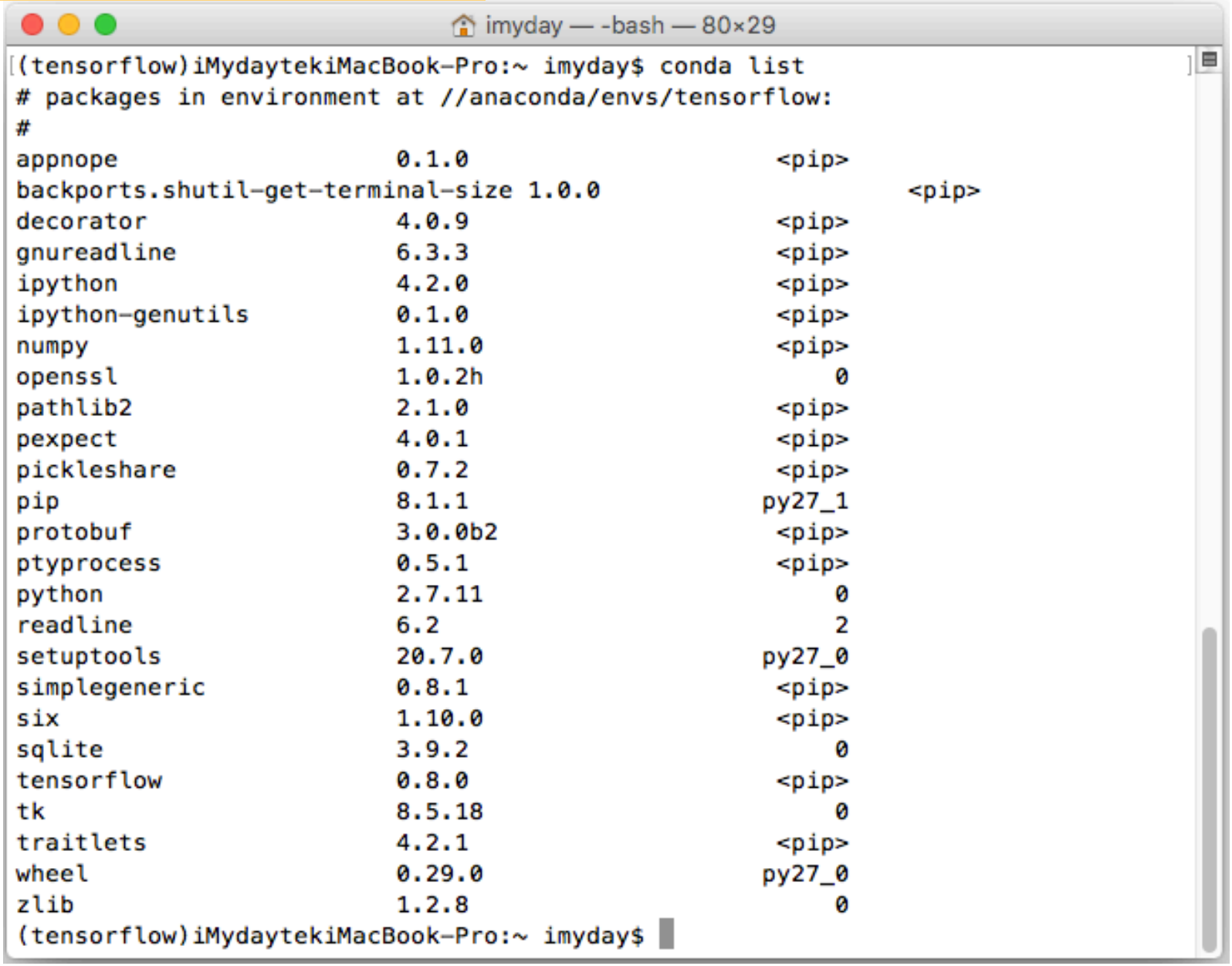
conda list


imyday — -bash — 80×17
(tensorflow)iMydaytekiMacBook-Pro:~ imyday$ conda list
packages in environment at //anaconda/envs/tensorflow:
#
numpy 1.11.0 <pip>
openssl 1.0.2h 0
pip 8.1.1 py27_1
protobuf 3.0.0b2 <pip>
python 2.7.11 0
readline 6.2 2
setuptools 20.7.0 py27_0
six 1.10.0 <pip>
sqlite 3.9.2 0
tensorflow 0.8.0 <pip>
tk 8.5.18 0
wheel 0.29.0 py27_0
zlib 1.2.8 0
(tensorflow)iMydaytekiMacBook-Pro:~ imyday$

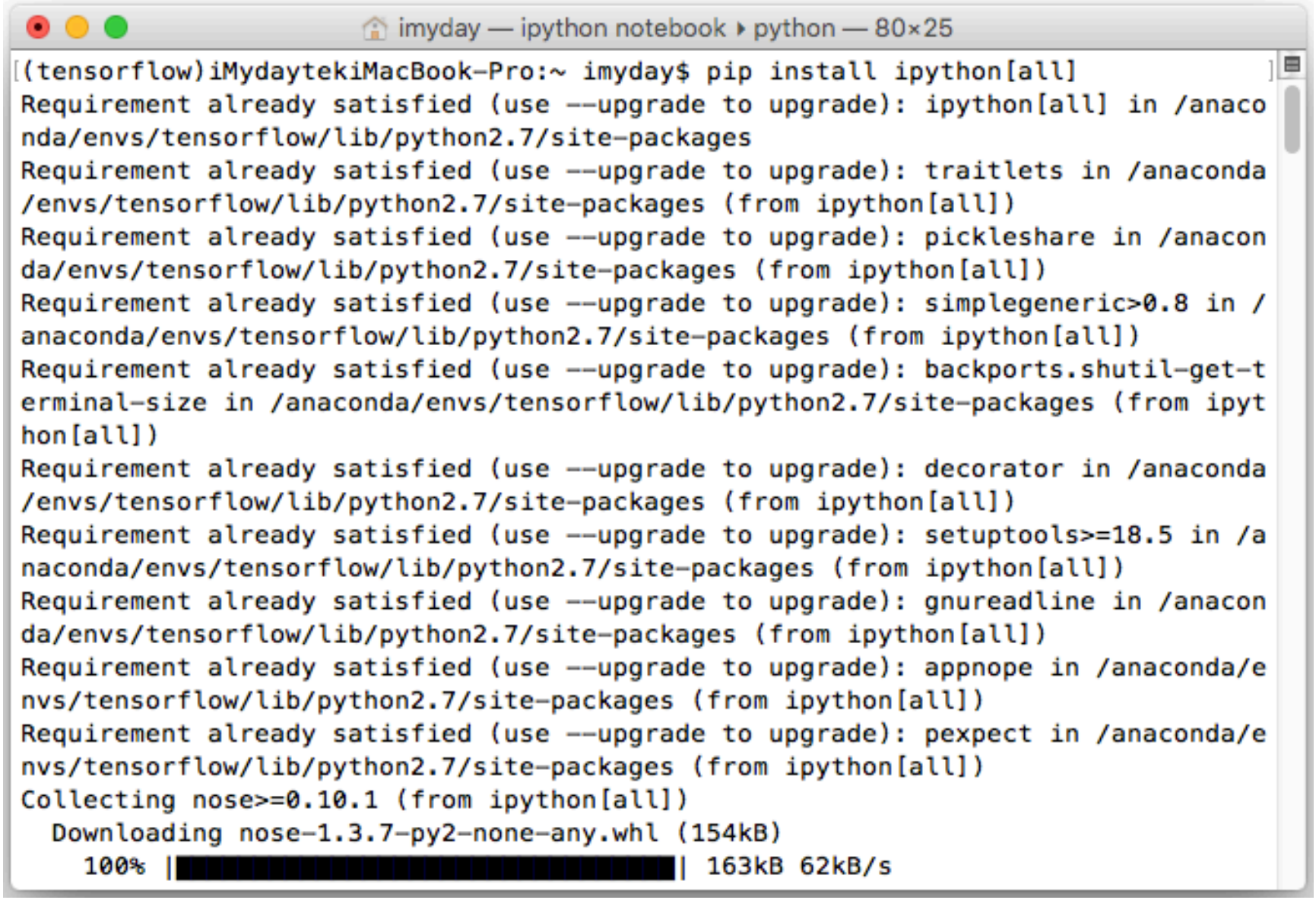
pip install ipython



conda list


imyday — -bash — 80×29
(tensorflow)iMydaytekiMacBook-Pro:~ imyday$ conda list
packages in environment at //anaconda/envs/tensorflow:
#
appnope 0.1.0 <pip>
backports.shutil-get-terminal-size 1.0.0 <pip>
decorator 4.0.9 <pip>
gnureadline 6.3.3 <pip>
ipython 4.2.0 <pip>
ipython-genutils 0.1.0 <pip>
numpy 1.11.0 <pip>
openssl 1.0.2h 0
pathlib2 2.1.0 <pip>
pexpect 4.0.1 <pip>
pickleshare 0.7.2 <pip>
pip 8.1.1 py27_1
protobuf 3.0.0b2 <pip>
ptyprocess 0.5.1 <pip>
python 2.7.11 0
readline 6.2 2
setuptools 20.7.0 py27_0
simplegeneric 0.8.1 <pip>
six 1.10.0 <pip>
sqlite 3.9.2 0
tensorflow 0.8.0 <pip>
tk 8.5.18 0
traitlets 4.2.1 <pip>
wheel 0.29.0 py27_0
zlib 1.2.8 0
(tensorflow)iMydaytekiMacBook-Pro:~ imyday$

pip install ipython[all]

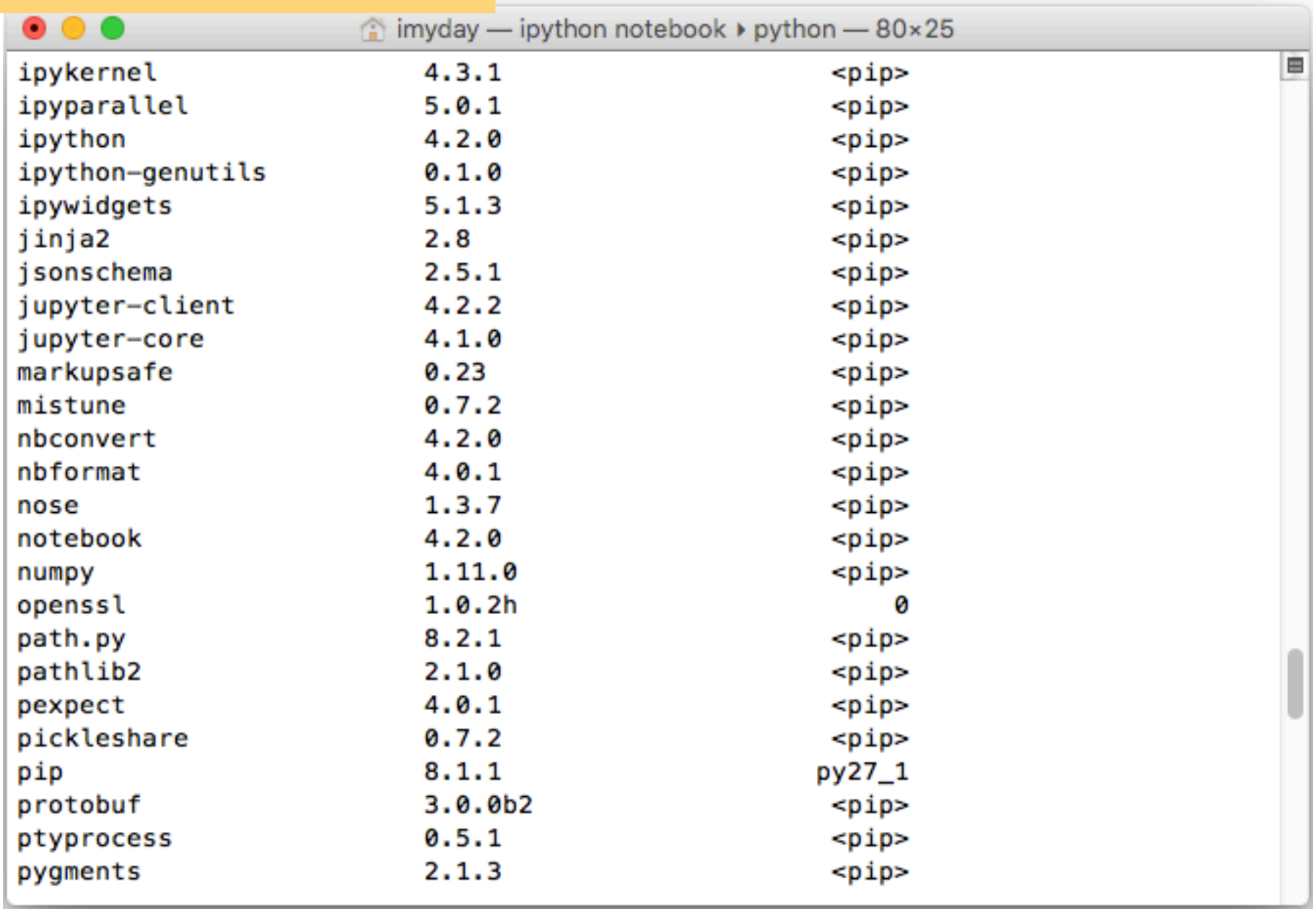


conda list

imyday — ipython notebook ▸ python — 80×25

```
(tensorflow)iMydaytekiMacBook-Pro:~ imyday$ conda list
# packages in environment at //anaconda/envs/tensorflow:
#
alabaster                 0.7.7                    <pip>
appnope                   0.1.0                    <pip>
babel                     2.3.4                    <pip>
backports-abc             0.4                      <pip>
backports.shutil-get-terminal-size 1.0.0                     <pip>
backports.ssl-match-hostname 3.5.0.1                  <pip>
certifi                   2016.2.28                <pip>
configparser              3.3.0.post2              <pip>
decorator                 4.0.9                    <pip>
docutils                  0.12                     <pip>
entrypoints               0.2.1                    <pip>
functools32               3.2.3.post2              <pip>
futures                   3.0.5                    <pip>
gnureadline               6.3.3                    <pip>
imagesize                 0.7.1                    <pip>
ipykernel                 4.3.1                    <pip>
ipyparallel               5.0.1                    <pip>
ipython                   4.2.0                    <pip>
ipython-genutils          0.1.0                    <pip>
ipywidgets                5.1.3                    <pip>
jinja2                    2.8                      <pip>
jsonschema                2.5.1                    <pip>
```

conda list

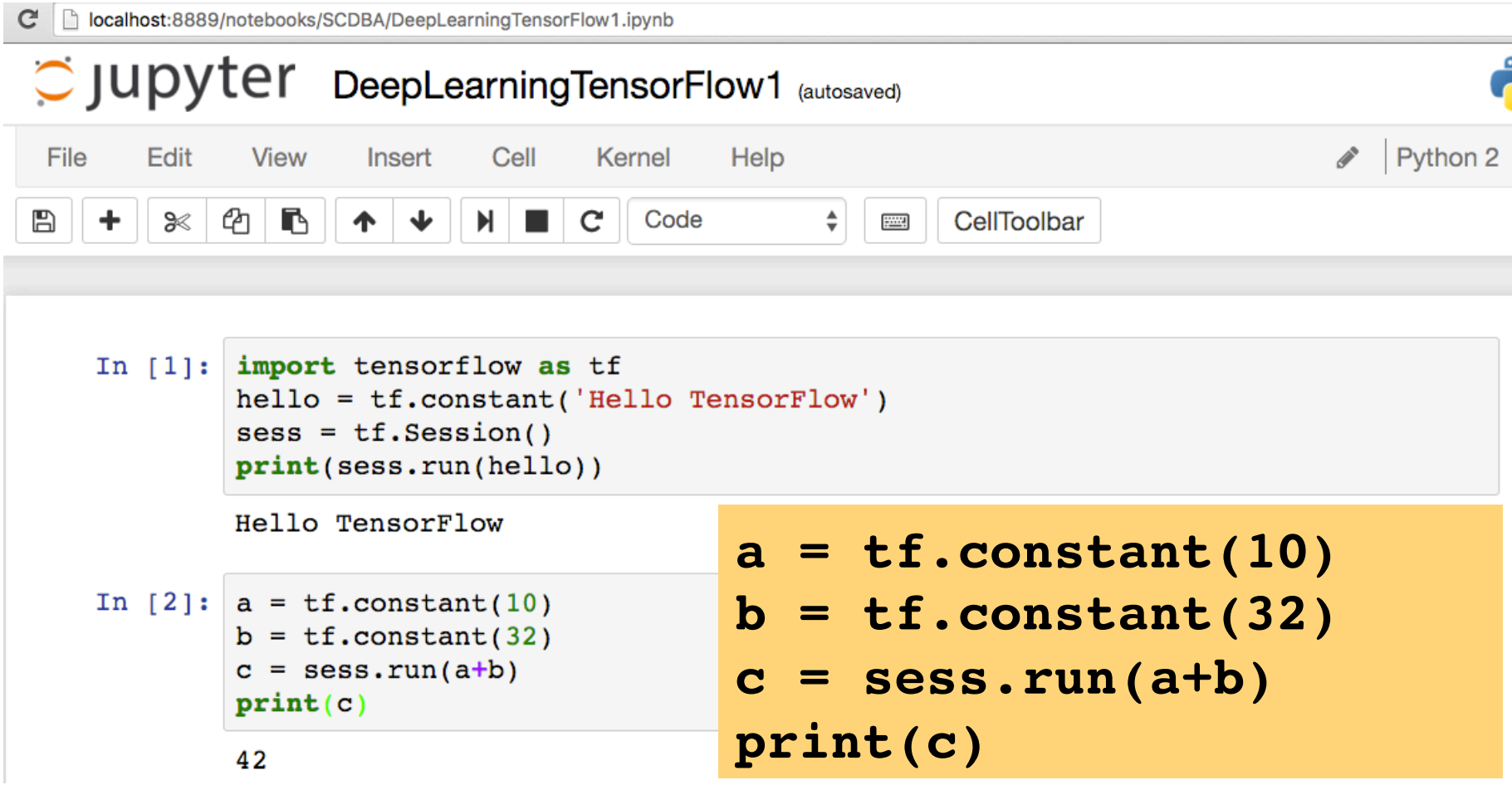

imyday — ipython notebook ▸ python — 80×25
ipykernel 4.3.1 <pip>
ipyparallel 5.0.1 <pip>
ipython 4.2.0 <pip>
ipython-genutils 0.1.0 <pip>
ipywidgets 5.1.3 <pip>
jinja2 2.8 <pip>
jsonschema 2.5.1 <pip>
jupyter-client 4.2.2 <pip>
jupyter-core 4.1.0 <pip>
markupsafe 0.23 <pip>
mistune 0.7.2 <pip>
nbconvert 4.2.0 <pip>
nbformat 4.0.1 <pip>
nose 1.3.7 <pip>
notebook 4.2.0 <pip>
numpy 1.11.0 <pip>
openssl 1.0.2h 0
path.py 8.2.1 <pip>
pathlib2 2.1.0 <pip>
pexpect 4.0.1 <pip>
pickleshare 0.7.2 <pip>
pip 8.1.1 py27_1
protobuf 3.0.0b2 <pip>
ptyprocess 0.5.1 <pip>
pygments 2.1.3 <pip>

ipython notebook

imyday — ipython notebook ▸ python — 80×25

```
simplegeneric                0.8.1                    <pip>
singledispatch               3.4.0.3                  <pip>
six                          1.10.0                   <pip>
snowballstemmer              1.2.1                    <pip>
sphinx                       1.4.1                    <pip>
sqlite                       3.9.2                       0
tensorflow                   0.8.0                    <pip>
terminado                    0.6                      <pip>
testpath                     0.3                      <pip>
tk                           8.5.18                      0
tornado                      4.3                      <pip>
traitlets                    4.2.1                    <pip>
wheel                        0.29.0                  py27_0
widgetsnbextension           1.2.2                    <pip>
zlib                         1.2.8                       0
(tensorflow)iMydaytekiMacBook-Pro:~ imyday$ ipython notebook
[TerminalIPythonApp] WARNING | Subcommand `ipython notebook` is deprecated and w
ill be removed in future versions.
[TerminalIPythonApp] WARNING | You likely want to use `jupyter notebook` in the
future
//anaconda/envs/tensorflow/lib/python2.7/site-packages/widgetsnbextension/__init
__.py:30: UserWarning: To use the jupyter-js-widgets nbextension, you'll need to
 update
    the Jupyter notebook to version 4.2 or later.
  the Jupyter notebook to version 4.2 or later.""")
```

```
import tensorflow as tf
hello = tf.constant('Hello TensorFlow')
sess = tf.Session()
print(sess.run(hello))
```




```
a = tf.constant(10)
b = tf.constant(32)
c = sess.run(a+b)
print(c)
```

Source: https://www.tensorflow.org/versions/r0.8/get_started/os_setup.html#test-the-tensorflow-installation

TensorFlow Example

```
import tensorflow as tf
import numpy as np

# Create 100 phony x, y data points in NumPy, y = x * 0.1 + 0.3
x_data = np.random.rand(100).astype(np.float32)
y_data = x_data * 0.1 + 0.3

# Try to find values for W and b that compute y_data = W * x_data + b
# (We know that W should be 0.1 and b 0.3, but Tensorflow will
# figure that out for us.)
W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))
y = W * x_data + b

# Minimize the mean squared errors.
loss = tf.reduce_mean(tf.square(y - y_data))
optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)

# Before starting, initialize the variables.  We will 'run' this first.
init = tf.initialize_all_variables()

# Launch the graph.
sess = tf.Session()
sess.run(init)

# Fit the line.
for step in xrange(201):
    sess.run(train)
    if step % 20 == 0:
        print(step, sess.run(W), sess.run(b))

# Learns best fit is W: [0.1], b: [0.3]
```

Source: https://www.tensorflow.org/versions/r0.8/get_started/index.html

TensorFlow Example

```
import tensorflow as tf
import numpy as np

# Create 100 phony x, y data points in NumPy, y = x * 0.1 + 0.3
x_data = np.random.rand(100).astype(np.float32)
y_data = x_data * 0.1 + 0.3

# Try to find values for W and b that compute y_data = W * x_data + b
# (We know that W should be 0.1 and b 0.3, but Tensorflow will
# figure that out for us.)
W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))
y = W * x_data + b

# Minimize the mean squared errors.
loss = tf.reduce_mean(tf.square(y - y_data))
optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)

# Before starting, initialize the variables.  We will 'run' this first.
init = tf.initialize_all_variables()

# Launch the graph.
sess = tf.Session()
sess.run(init)

# Fit the line.
for step in xrange(201):
    sess.run(train)
    if step % 20 == 0:
        print(step, sess.run(W), sess.run(b))

# Learns best fit is W: [0.1], b: [0.3]
```

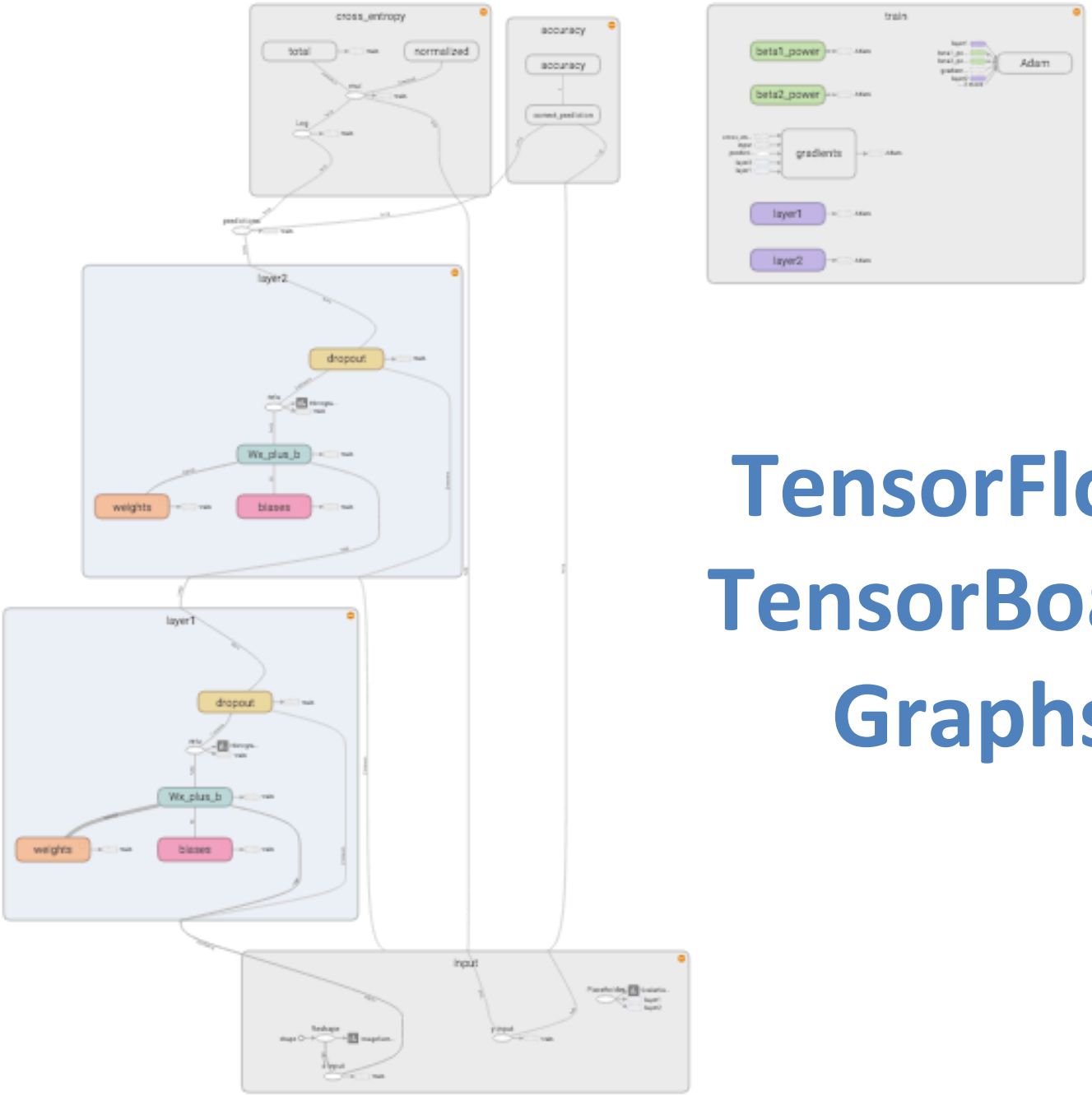
Source: https://www.tensorflow.org/versions/r0.8/get_started/index.html

TensorFlow Example

```
# Launch the graph.
sess = tf.Session()
sess.run(init)

# Fit the line.
for step in xrange(201):
    sess.run(train)
    if step % 20 == 0:
        print(step, sess.run(W), sess.run(b))

# Learns best fit is W: [0.1], b: [0.3]
```

```
(0, array([-0.07500112], dtype=float32), array([ 0.54339123], dtype=float32))
(20, array([ 0.05152683], dtype=float32), array([ 0.32598534], dtype=float32))
(40, array([ 0.09009784], dtype=float32), array([ 0.30530834], dtype=float32))
(60, array([ 0.09797716], dtype=float32), array([ 0.3010844], dtype=float32))
(80, array([ 0.09958676], dtype=float32), array([ 0.30022153], dtype=float32))
(100, array([ 0.09991558], dtype=float32), array([ 0.30004525], dtype=float32))
(120, array([ 0.09998275], dtype=float32), array([ 0.30000925], dtype=float32))
(140, array([ 0.09999647], dtype=float32), array([ 0.30000189], dtype=float32))
(160, array([ 0.09999929], dtype=float32), array([ 0.3000004], dtype=float32))
(180, array([ 0.09999986], dtype=float32), array([ 0.3000001], dtype=float32))
(200, array([ 0.0999999], dtype=float32), array([ 0.30000007], dtype=float32))
```

Source: https://www.tensorflow.org/versions/r0.8/get_started/index.html



TensorFlow TensorBoard Graphs

https://www.tensorflow.org/tensorboard/index.html#graphs

TensorFlow Example MNIST Softmax

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

# Import data
from tensorflow.examples.tutorials.mnist import input_data

import tensorflow as tf

flags = tf.app.flags
FLAGS = flags.FLAGS
flags.DEFINE_string('data_dir', '/tmp/data/', 'Directory for storing data')

mnist = input_data.read_data_sets(FLAGS.data_dir, one_hot=True)

sess = tf.InteractiveSession()

# Create the model
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)

# Define loss and optimizer
y_ = tf.placeholder(tf.float32, [None, 10])
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), reduction_indices=[1]))
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)

# Train
tf.initialize_all_variables().run()
for i in range(1000):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  train_step.run({x: batch_xs, y_: batch_ys})

# Test trained model
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print(accuracy.eval({x: mnist.test.images, y_: mnist.test.labels}))
```

https://github.com/tensorflow/tensorflow/blob/r0.8/tensorflow/examples/tutorials/mnist/mnist_softmax.py

TensorFlow Example MNIST Softmax

```python
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

# Import data
from tensorflow.examples.tutorials.mnist import input_data

import tensorflow as tf

flags = tf.app.flags
FLAGS = flags.FLAGS
flags.DEFINE_string('data_dir', '/tmp/data/', 'Directory for storing data')

mnist = input_data.read_data_sets(FLAGS.data_dir, one_hot=True)

sess = tf.InteractiveSession()

# Create the model
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)

# Define loss and optimizer
y_ = tf.placeholder(tf.float32, [None, 10])
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), reduction_indices=[1]))
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)

# Train
tf.initialize_all_variables().run()
for i in range(1000):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  train_step.run({x: batch_xs, y_: batch_ys})

# Test trained model
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print(accuracy.eval({x: mnist.test.images, y_: mnist.test.labels}))
```

https://github.com/tensorflow/tensorflow/blob/r0.8/tensorflow/examples/tutorials/mnist/mnist_softmax.py

TensorFlow Example MNIST Softmax

```
flags.DEFINE_string('data_dir', '/tmp/data/', 'Directory for storing data')

mnist = input_data.read_data_sets(FLAGS.data_dir, one_hot=True)

sess = tf.InteractiveSession()

# Create the model
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)

# Define loss and optimizer
y_ = tf.placeholder(tf.float32, [None, 10])
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), reduction_indices=[1]))
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)

# Train
tf.initialize_all_variables().run()
for i in range(1000):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  train_step.run({x: batch_xs, y_: batch_ys})

# Test trained model
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print(accuracy.eval({x: mnist.test.images, y_: mnist.test.labels}))
```

```
Extracting /tmp/data/train-images-idx3-ubyte.gz
Extracting /tmp/data/train-labels-idx1-ubyte.gz
Extracting /tmp/data/t10k-images-idx3-ubyte.gz
Extracting /tmp/data/t10k-labels-idx1-ubyte.gz
0.9179
```

https://github.com/tensorflow/tensorflow/blob/r0.8/tensorflow/examples/tutorials/mnist/mnist_softmax.py

TensorFlow
Deep MNIST for Experts

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import gzip
import os
import tempfile

import numpy
from six.moves import urllib
from six.moves import xrange  # pylint: disable=redefined-builtin
import tensorflow as tf
from tensorflow.contrib.learn.python.learn.datasets.mnist import read_data_sets

def weight_variable(shape):
  initial = tf.truncated_normal(shape, stddev=0.1)
  return tf.Variable(initial)

def bias_variable(shape):
  initial = tf.constant(0.1, shape=shape)
  return tf.Variable(initial)

def conv2d(x, W):
  return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2x2(x):
  return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
                        strides=[1, 2, 2, 1], padding='SAME')

W_conv1 = weight_variable([5, 5, 1, 32])
b_conv1 = bias_variable([32])

x_image = tf.reshape(x, [-1,28,28,1])

h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)

W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])

h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)

W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])

h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

keep_prob = tf.placeholder(tf.float32)
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])

y_conv=tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)

cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y_conv), reduction_indices=[1]))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
sess.run(tf.initialize_all_variables())
for i in range(20000):
  batch = mnist.train.next_batch(50)
  if i%100 == 0:
    train_accuracy = accuracy.eval(feed_dict={
        x:batch[0], y_: batch[1], keep_prob: 1.0})
    print("step %d, training accuracy %g"%(i, train_accuracy))
  train_step.run(feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})

print("test accuracy %g"%accuracy.eval(feed_dict={
    x: mnist.test.images, y_: mnist.test.labels, keep_prob: 1.0}))
```

https://www.tensorflow.org/versions/r0.8/tutorials/mnist/pros/index.html

TensorFlow Deep MNIST for Experts

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import gzip
import os
import tempfile

import numpy
from six.moves import urllib
from six.moves import xrange  # pylint: disable=redefined-builtin
import tensorflow as tf
from tensorflow.contrib.learn.python.learn.datasets.mnist import read_data_sets

def weight_variable(shape):
  initial = tf.truncated_normal(shape, stddev=0.1)
  return tf.Variable(initial)

def bias_variable(shape):
  initial = tf.constant(0.1, shape=shape)
  return tf.Variable(initial)

def conv2d(x, W):
  return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2x2(x):
  return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
                        strides=[1, 2, 2, 1], padding='SAME')
```

https://www.tensorflow.org/versions/r0.8/tutorials/mnist/pros/index.html

TensorFlow Deep MNIST for Experts

```
W_conv1 = weight_variable([5, 5, 1, 32])
b_conv1 = bias_variable([32])

x_image = tf.reshape(x, [-1,28,28,1])

h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)

W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])

h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)

W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])

h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

keep_prob = tf.placeholder(tf.float32)
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])

y_conv=tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
```

https://www.tensorflow.org/versions/r0.8/tutorials/mnist/pros/index.html

TensorFlow Deep MNIST for Experts

```
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y_conv), reduction_indices=[1]))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
sess.run(tf.initialize_all_variables())
for i in range(20000):
  batch = mnist.train.next_batch(50)
  if i%100 == 0:
    train_accuracy = accuracy.eval(feed_dict={
        x:batch[0], y_: batch[1], keep_prob: 1.0})
    print("step %d, training accuracy %g"%(i, train_accuracy))
  train_step.run(feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})

print("test accuracy %g"%accuracy.eval(feed_dict={
    x: mnist.test.images, y_: mnist.test.labels, keep_prob: 1.0}))
```

https://www.tensorflow.org/versions/r0.8/tutorials/mnist/pros/index.html

TensorFlow Deep MNIST for Experts

```
step 0, training accuracy 0.06
step 100, training accuracy 0.7
step 200, training accuracy 0.94
step 300, training accuracy 0.94
step 400, training accuracy 0.94
step 500, training accuracy 0.92
step 600, training accuracy 0.88
step 700, training accuracy 0.94
step 800, training accuracy 0.96
step 900, training accuracy 0.94
step 1000, training accuracy 0.92
step 1100, training accuracy 0.98
step 1200, training accuracy 0.92
step 1300, training accuracy 0.96
step 1400, training accuracy 0.98
step 1500, training accuracy 0.98
step 1600, training accuracy 0.9
step 1700, training accuracy 0.94
step 1800, training accuracy 0.98
step 1900, training accuracy 0.92
step 2000, training accuracy 1
step 2100, training accuracy 0.96
step 2200, training accuracy 1
step 2300, training accuracy 1
step 2400, training accuracy 0.98
step 2500, training accuracy 0.94
step 2600, training accuracy 0.98
step 2700, training accuracy 1
step 2800, training accuracy 0.94
step 2900, training accuracy 0.98
step 3000, training accuracy 0.98
```

https://www.tensorflow.org/versions/r0.8/tutorials/mnist/pros/index.html

Deep Learning Software

- Theano
  - CPU/GPU symbolic expression compiler in python (from MILA lab at University of Montreal)
- Keras
  - A theano based deep learning library.
- Tensorflow
  - TensorFlow™ is an open source software library for numerical computation using data flow graphs.

Source: http://deeplearning.net/software_links/

References

- Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
- Sebastian Raschka (2015), Python Machine Learning, Packt Publishing
- TensorFlow: https://www.tensorflow.org/
- Rajat Monga (2016), TensorFlow: Machine Learning for Everyone, https://www.youtube.com/watch?v=wmw8Bbb_eIE
- Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, The 9th ACM International Conference on Web Search and Data Mining (WSDM 2016), San Francisco, California, USA., February 22-25, 2016. http://www.wsdm-conference.org/2016/slides/WSDM2016-Jeff-Dean.pdf
- Deep Learning Basics: Neural Networks Demystified, https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
- Deep Learning SIMPLIFIED, https://www.youtube.com/playlist?list=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu
- Theano: http://deeplearning.net/software/theano/
- Keras: http://keras.io/