EECS 510: Social Media Mining
Spring 2016

Introduction to Deep Learning: Software Packages

Rosanne Liu
rosanne.liu@northwestern.edu
May 5, 2016

Outline

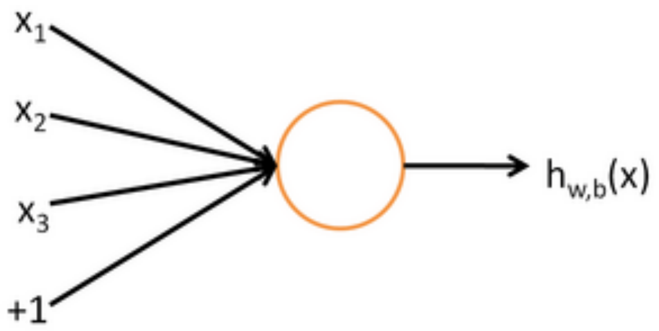
- Review
    - Deep neural networks
    - Backpropagation
- A quick start with Torch
- A quick start with Theano
- A quick start with TensorFlow

Deep learning review

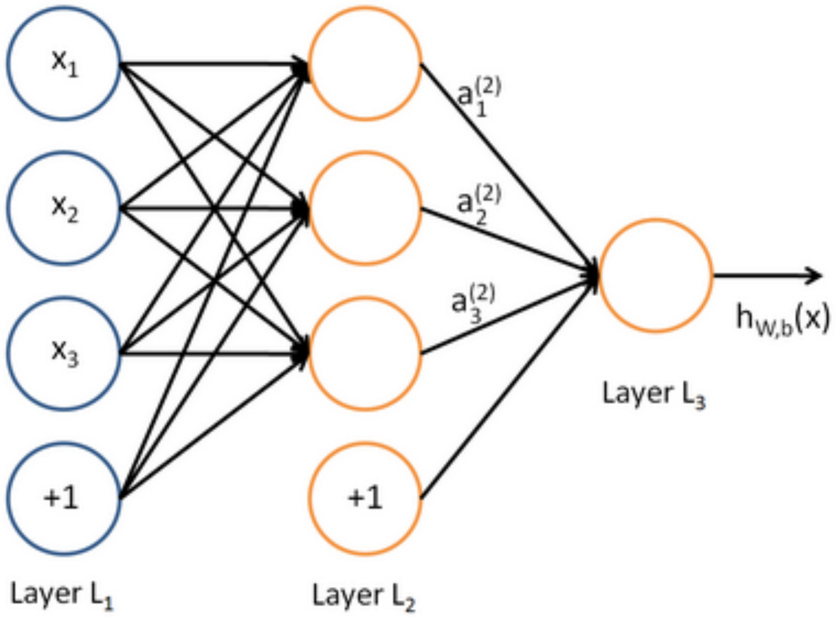
(Artificial) neural networks and training

- Artificial neural networks
  - biology inspired, supervised, weights, activation, backpropagation

A “neuron”



A small neural net



Training of a NN

Loop until tired:

1. **Sample** a batch of data.
2. **Forward** it through the network to get predictions.
3. **Backprop** the errors.
4. **Update** the weights.

Backprop

- Backprop: a way of computing *gradients* of expressions through recursive application of **chain rule**

- Problem: given a function f(x) where x is a vector of inputs, were are interested in computing the gradient of f at x, i.e., $\nabla f(x)$

- Autodifferentiation
  - http://arxiv.org/pdf/1502.05767v2.pdf

Deep learning package zoo

- Torch
- Caffe
- Theano (keras, lasagne)
- CuDNN
- Tensorflow
- Mxnet
- Etc.

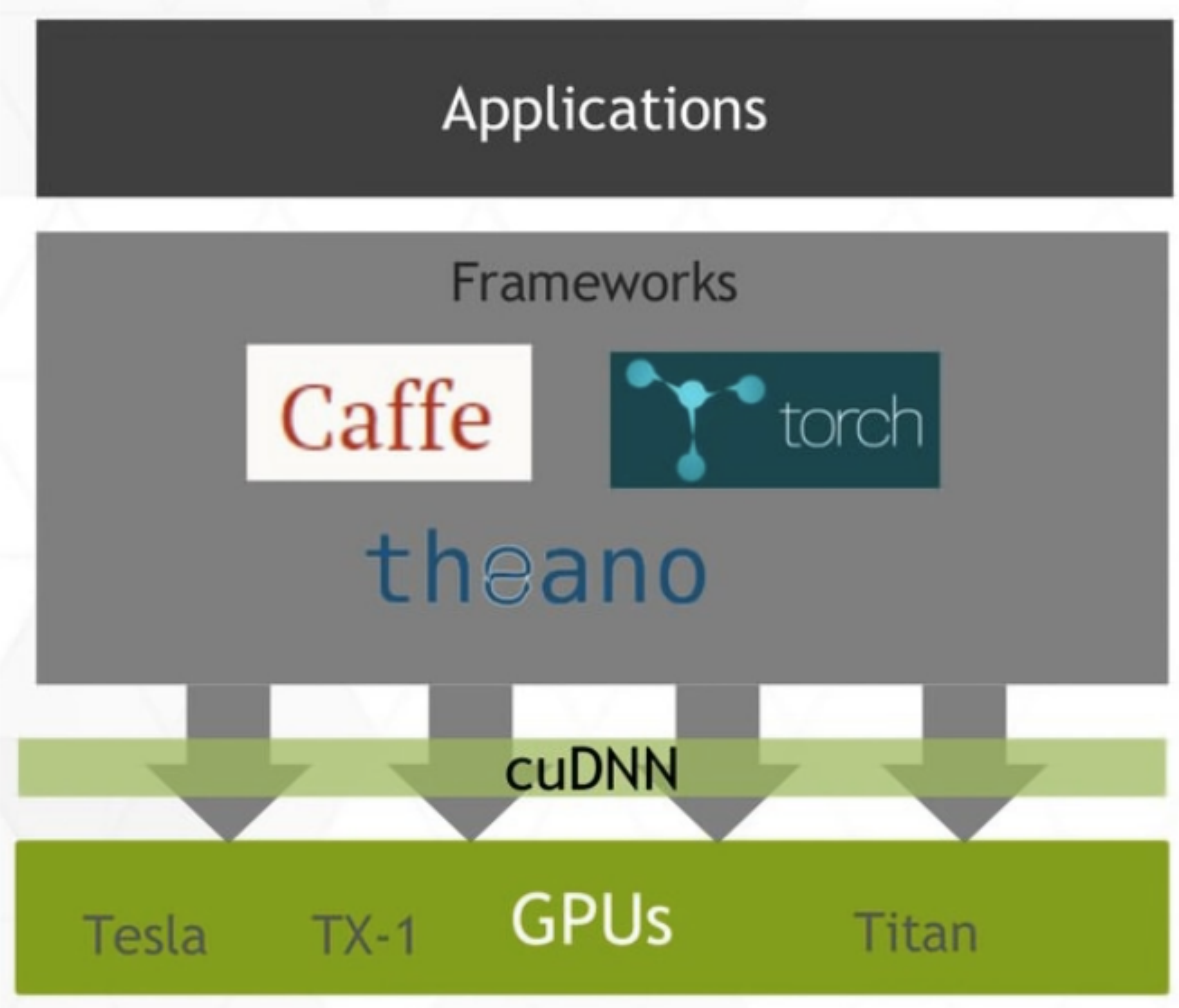
Details of package

- Backend
  - Krizhevsky (AlexNet) first built CNN in CUDA (cunn)
  - Then Nvidia took the charge and built cudnn
- Berkeley's Caffe
  - modular CNN package in C++
  - with both CPU/GPU training
- Bengio's Theano
  - Python project (w/ numpy & scipy)
  - works with GPU through cunn, cudnn
  - compile to C on-the-fly
- Facebook/NYU's Torch7
  - LuaJIT interface to C
- Google's TensorFlow
  - C with python interface

What they all have in common

- Tensor
- Symbolic differentiation
- GPU support (through backend like CuDNN)
- Open source

CuDNN is behind every package



CuDNN

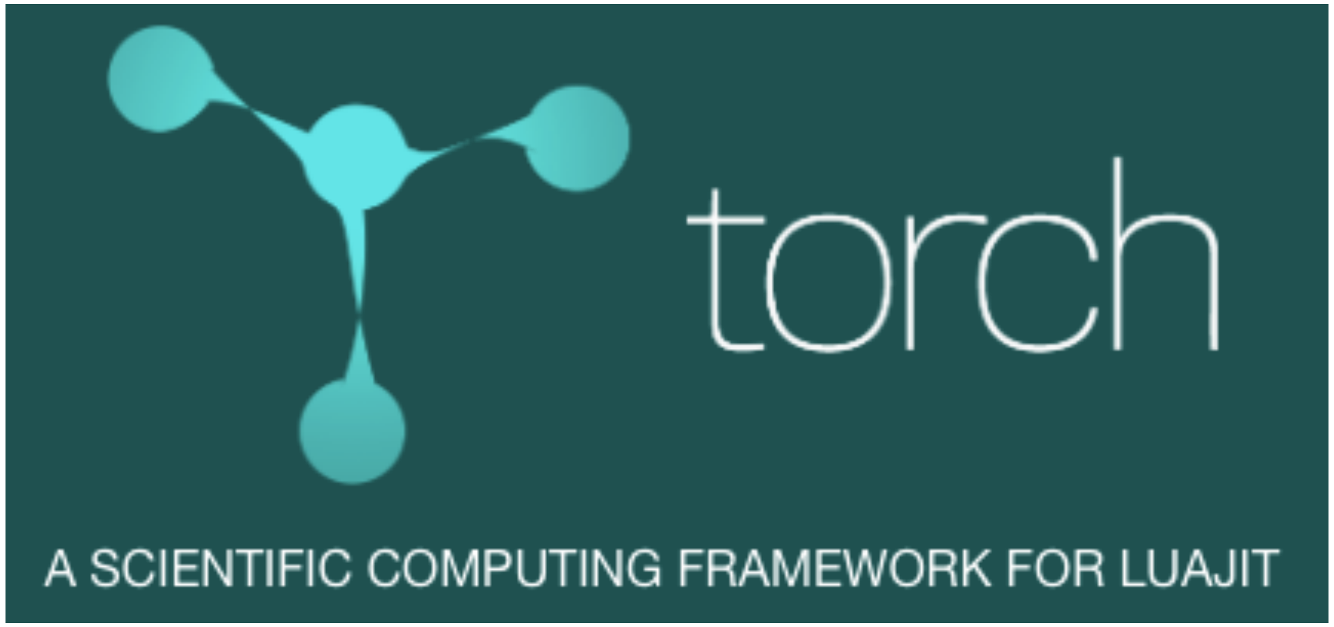
- GPU-accelerated Deep Learning subroutines
- High performance neural network training
- Accelerates major deep learning frameworks: Caffe, Theano, Torch, TensorFlow
- Up to 3.5x faster AlexNet training in Caffe than baseline GPU

Deep learning package design choices

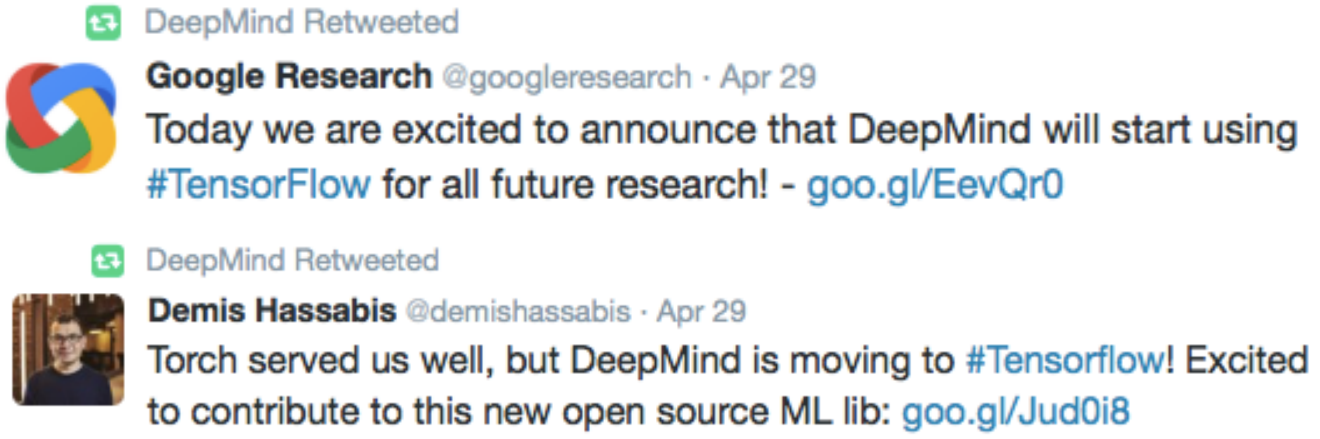
- Model specification
    - configuration file (e.g. Caffe, CNTK)
    - Programmatic generation (e.g. Torch, Theano, TensorFlow)
- Programming language
    - Everything links to C at the bottom
    - Lua (Torch)
    - Python (Theano, TensorFlow)

A quick start with Torch

A quick start with Torch



- Project started ~2000 at Facebook, now Torch7 - 4th generation (using odd numbers only 1,3,5,7)
- Web-scale learning in speech, image and video applications
- Maintained by top researchers (@Facebook, twitter, DeepMind)
- Up until 6 days ago DeepMind has been using it



DeepMind Retweeted

Google Research @googleresearch · Apr 29

Today we are excited to announce that DeepMind will start using #TensorFlow for all future research! - goo.gl/EevQr0

DeepMind Retweeted

Demis Hassabis @demishassabis · Apr 29

Torch served us well, but DeepMind is moving to #Tensorflow! Excited to contribute to this new open source ML lib: goo.gl/Jud0i8

Cheatsheet

- Cheatsheet
  - https://github.com/torch/torch7/wiki/Cheatsheet
- Github
  - https://github.com/torch/torch7
- Google Group for new users and installation questions
  - https://groups.google.com/forum/embed/?place=forum%2Ftorch7#!forum/torch7
- **Advanced only**
  - https://gitter.im/torch/torch7

Tensors

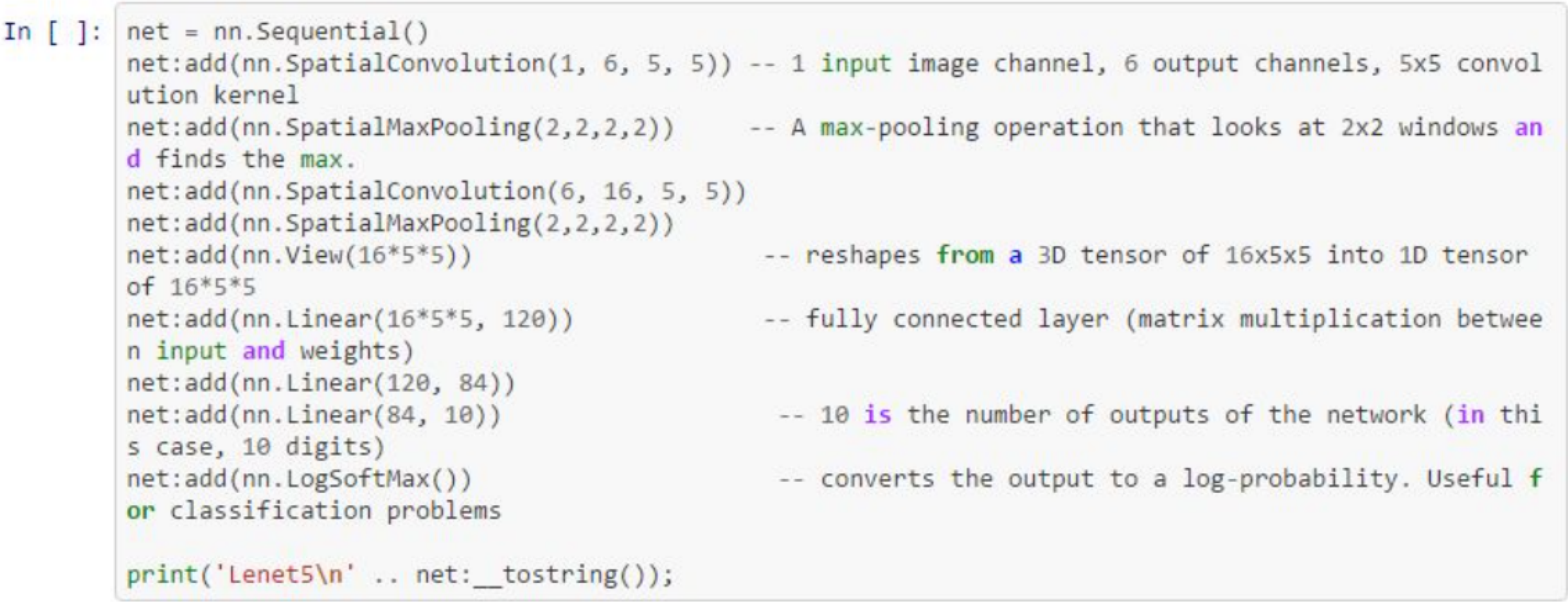
- The Tensor class is the most important class in Torch (and others)
- A Tensor is a serializable, potentially multi-dimensional matrix
- The number of dimensions in Torch is unlimited (created using LongStorage)

- A scalar is a tensor $(f : \mathbb{R} \rightarrow \mathbb{R})$
- A vector is a tensor $(f : \mathbb{R}^n \rightarrow \mathbb{R})$
- A matrix is a tensor $(f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R})$

Torch Core

- The Torch core consists of the following packages:
    - torch: tensors, class factory, serialization, BLAS
    - nn: neural network Modules and Criterions
    - opitm: SGD and other optimization functions
    - gnuplot: ploting and data visualization
    - paths: make directories, concatenate file paths, and other filesystem utilities
    - image: save, load, crop, scale, warp, translate image and such
    - trepl: the torch LuaJIT interpreter
    - cwrap: used for wrapping C/CUDA functions in Lua

A snapshot of Torch code



In [ ]:

```
net = nn.Sequential()
net:add(nn.SpatialConvolution(1, 6, 5, 5)) -- 1 input image channel, 6 output channels, 5x5 convolution kernel
net:add(nn.SpatialMaxPooling(2,2,2,2))     -- A max-pooling operation that looks at 2x2 windows and finds the max.
net:add(nn.SpatialConvolution(6, 16, 5, 5))
net:add(nn.SpatialMaxPooling(2,2,2,2))
net:add(nn.View(16*5*5))                    -- reshapes from a 3D tensor of 16x5x5 into 1D tensor of 16*5*5
net:add(nn.Linear(16*5*5, 120))             -- fully connected layer (matrix multiplication between input and weights)
net:add(nn.Linear(120, 84))
net:add(nn.Linear(84, 10))                   -- 10 is the number of outputs of the network (in this case, 10 digits)
net:add(nn.LogSoftMax())                     -- converts the output to a log-probability. Useful for classification problems

print('Lenet5\n' .. net:__tostring());
```

A quick start with Theano

What is Theano?

- Developed and used since January 2008, created at Universite de Montreal
- Tutorials and examples: http://deeplearning,net/tutorial/

- A mathematical symbolic expression compiler
- A Python library for symbolic maths
    - far broader than just Deep Learning
- Tightly integrated with the Python ecosystem
- Fast C/CUDA back-end and transparent GPU acceleration

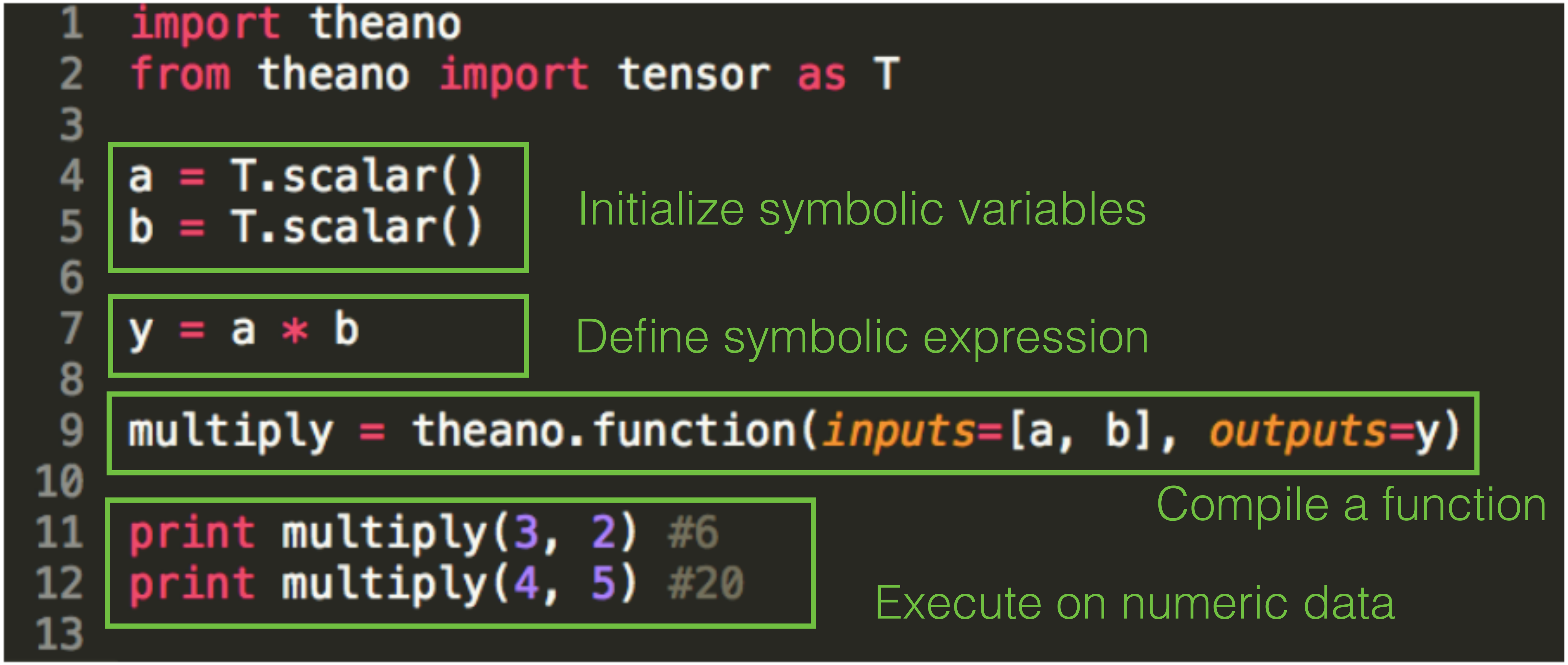
Theano Recipe

- Recipe for a Theano application:
    - Define symbolic expressions
    - Compile a function that can compile numeric values using those expressions
    - Execute that function on data

Theano Example

$$y = a \times b$$

$$a, b \in \mathbb{R}$$



```python
import theano
from theano import tensor as T

a = T.scalar()
b = T.scalar()

y = a * b

multiply = theano.function(inputs=[a, b], outputs=y)

print multiply(3, 2) #6
print multiply(4, 5) #20

```

Initialize symbolic variables

Define symbolic expression

Compile a function

Execute on numeric data

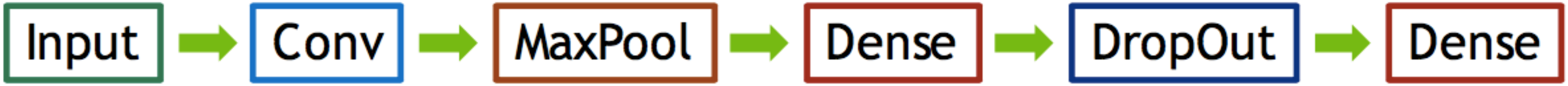
Related Projects (ML)

- Keras
- Lasagne
- Pylearn2
- PyMC 3
- sklearn-theano
- theano-rnn
- more…

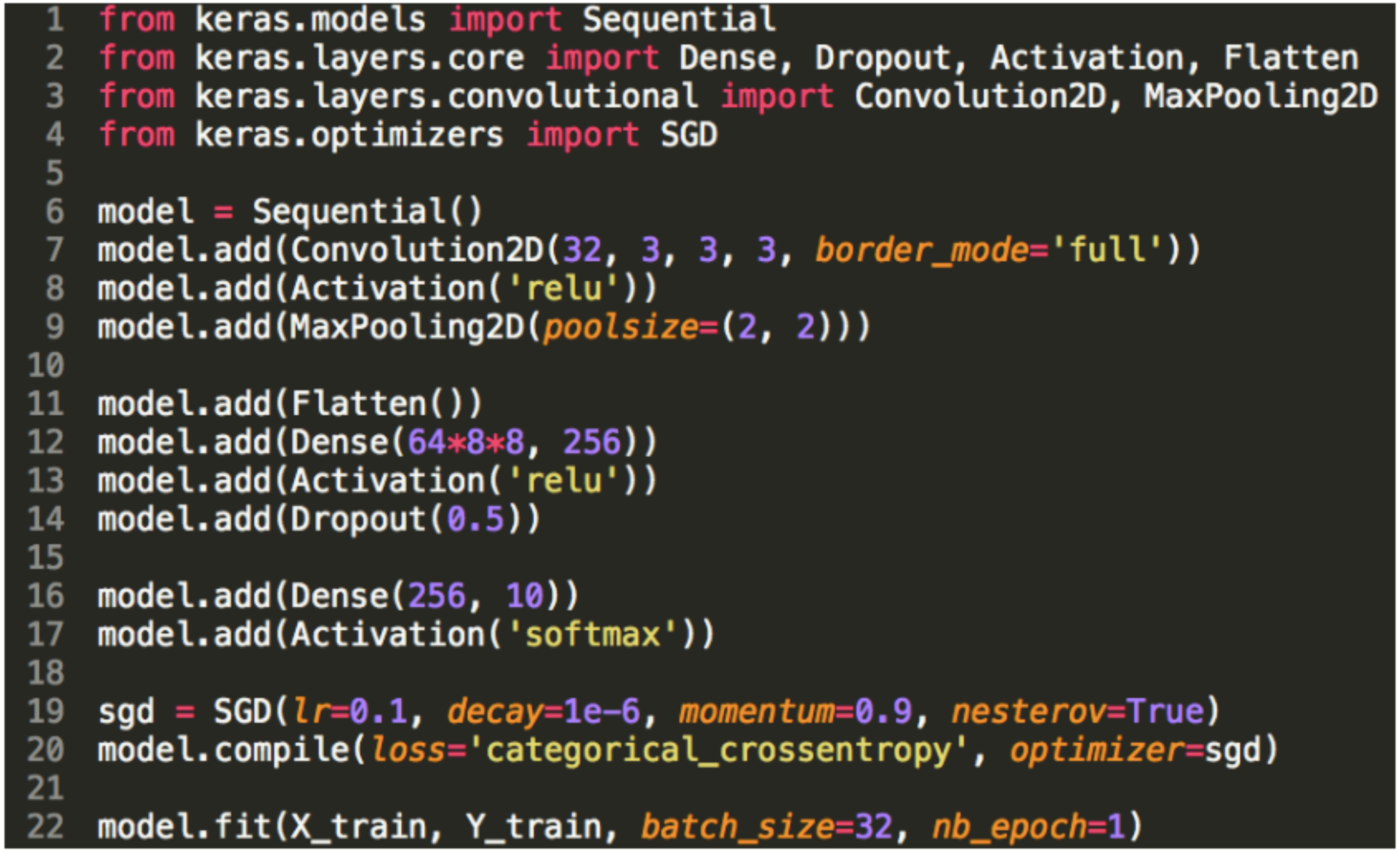
Built on top of Theano

Typically simplify syntax and interface for NN training

CNN example in Keras



Input → Conv → MaxPool → Dense → DropOut → Dense



```python
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.optimizers import SGD

model = Sequential()
model.add(Convolution2D(32, 3, 3, 3, border_mode='full'))
model.add(Activation('relu'))
model.add(MaxPooling2D(poolsize=(2, 2)))

model.add(Flatten())
model.add(Dense(64*8*8, 256))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(256, 10))
model.add(Activation('softmax'))

sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd)

model.fit(X_train, Y_train, batch_size=32, nb_epoch=1)
```

A quick start with TensorFlow

TensorFlow vs. Theano

- Both are deep learning libraries with Python wrapper
- Theano came out first (was inspiration for TensorFlow)
- TensorFlow has better support for distributed systems
- TensorFlow has development funded by Google, while Theano is an academic project
- Everything about TensorFlow is here:
    - https://www.tensorflow.org

TensorFlow Essentials

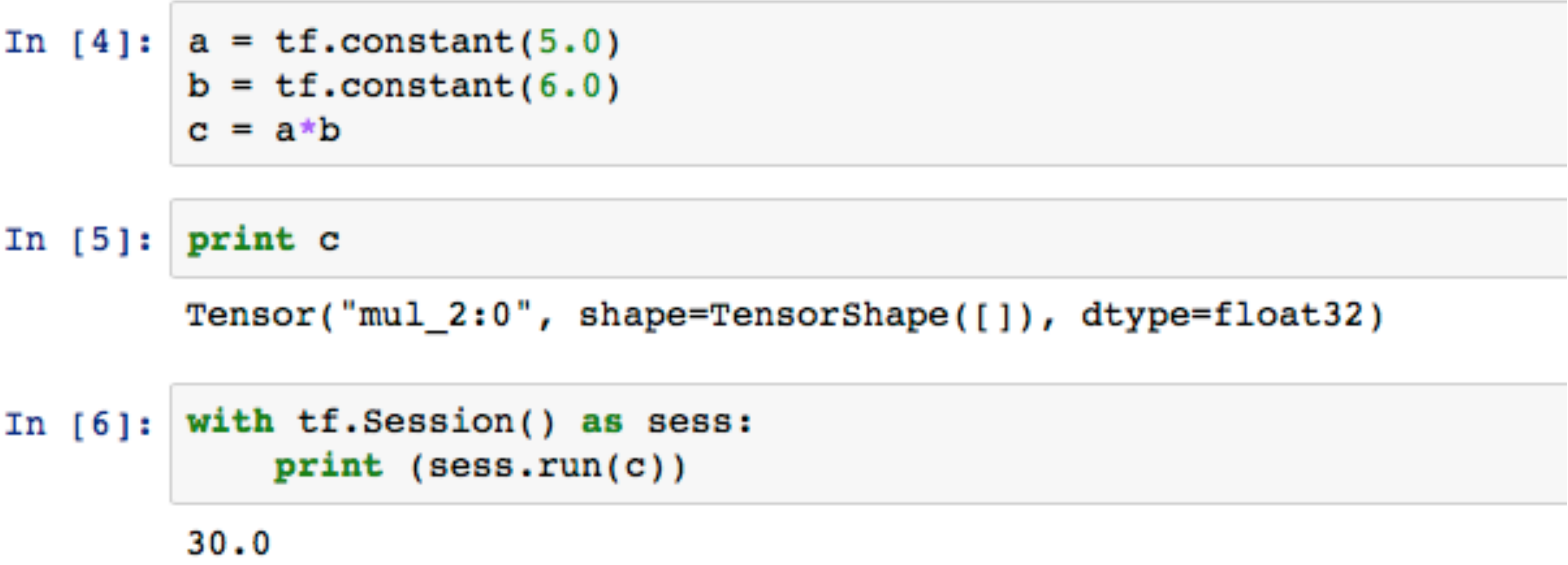
- Four types of objects make TensorFlow unique from other frameworks
    - Session
    - Computational graph
    - Variables
    - Placeholder

TensorFlow Recipe

- Recipe for a TensorFlow application:
  - Define a series of expressions
  - Initialize variables
  - Start a session (launch a graph)
  - Run the graph, feed some data, fetch some values

TensorFlow Session

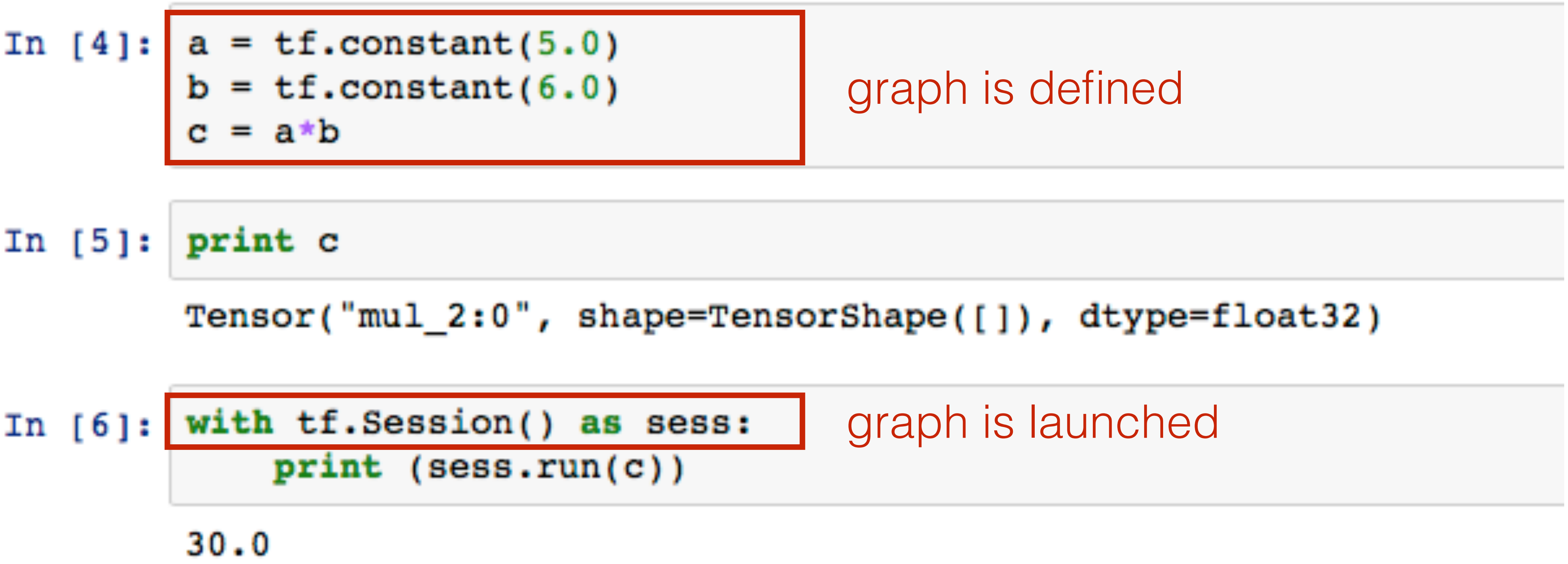
- “A Session object encapsulates the environment in which Tensor objects are evaluated.” — TensorFlow Docs



```
In [4]: a = tf.constant(5.0)
        b = tf.constant(6.0)
        c = a*b

In [5]: print c

        Tensor("mul_2:0", shape=TensorShape([]), dtype=float32)

In [6]: with tf.Session() as sess:
            print (sess.run(c))

        30.0
```

TensorFlow Computational Graph

- “TensorFlow programs are usually structured into a construction phase, that assembles a graph, and an execution phase that uses a session to execute ops in the graph.” — TensorFlow Docs



```
In [4]: a = tf.constant(5.0)
        b = tf.constant(6.0)
        c = a*b
```

graph is defined

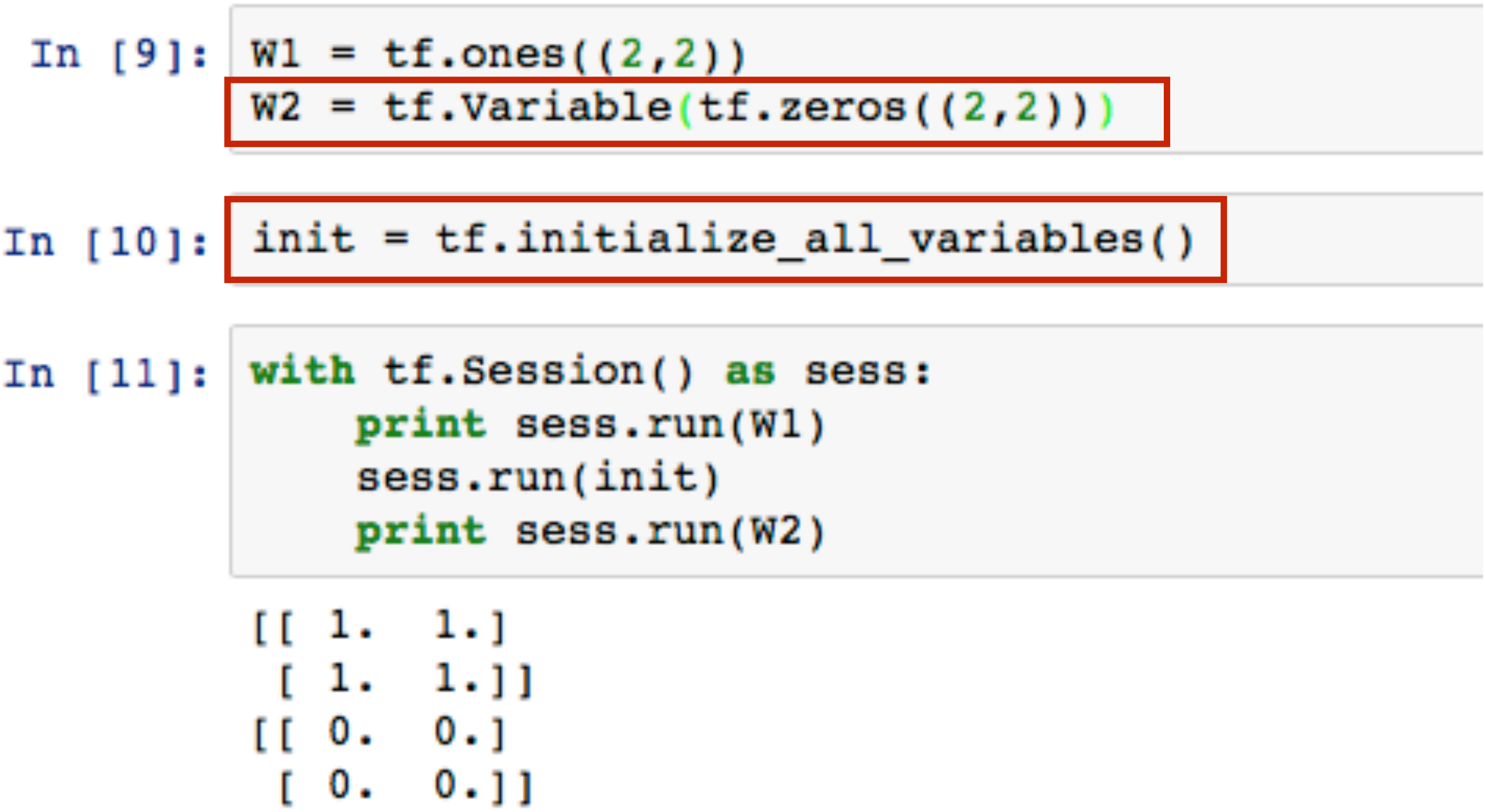
```
In [5]: print c

        Tensor("mul_2:0", shape=TensorShape([]), dtype=float32)
```

```
In [6]: with tf.Session() as sess:
            print (sess.run(c))

        30.0
```

graph is launched

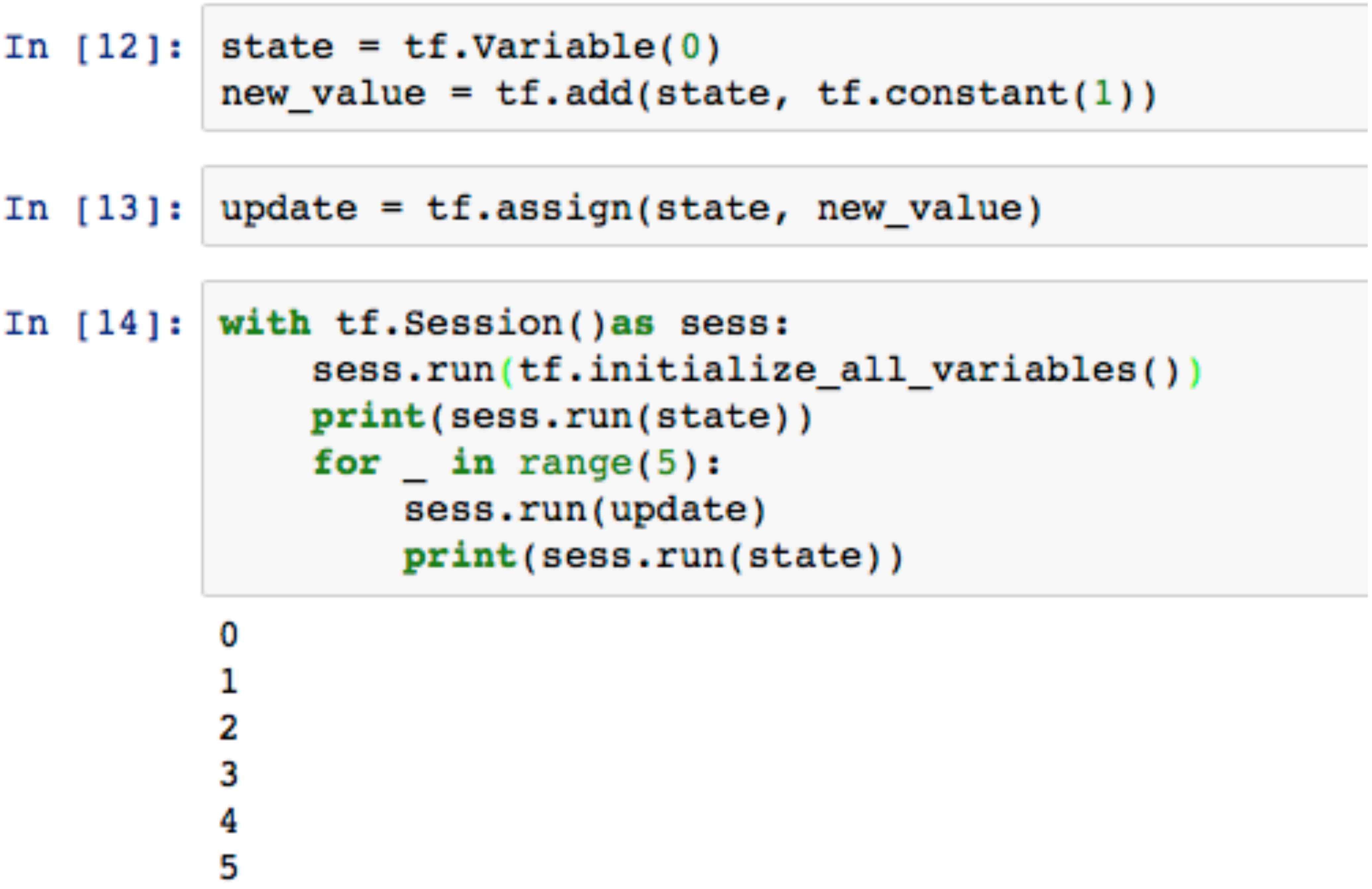
TensorFlow Variables

- TensorFlow Variables: hold and update **parameters**



```
In [9]: W1 = tf.ones((2,2))
        W2 = tf.Variable(tf.zeros((2,2)))

In [10]: init = tf.initialize_all_variables()

In [11]: with tf.Session() as sess:
             print sess.run(W1)
             sess.run(init)
             print sess.run(W2)

[[ 1.  1.]
 [ 1.  1.]]
[[ 0.  0.]
 [ 0.  0.]]
```

Update Variables



```
In [12]: state = tf.Variable(0)
         new_value = tf.add(state, tf.constant(1))
```

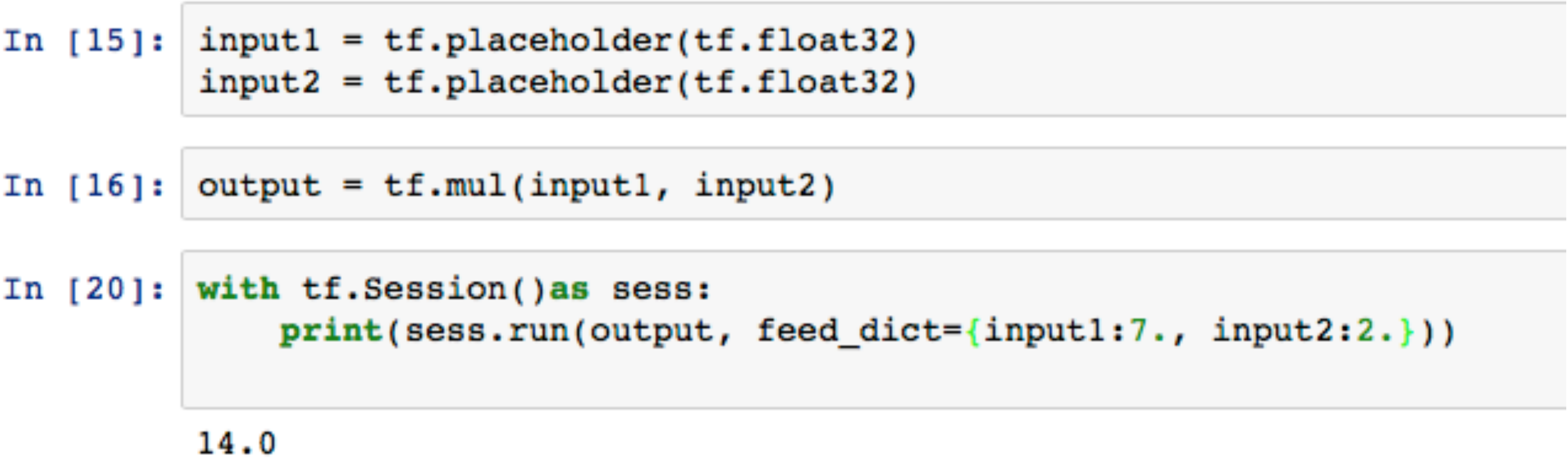
```
In [13]: update = tf.assign(state, new_value)
```

```
In [14]: with tf.Session()as sess:
             sess.run(tf.initialize_all_variables())
             print(sess.run(state))
             for _ in range(5):
                 sess.run(update)
                 print(sess.run(state))
```

```
0
1
2
3
4
5
```

TensorFlow Placeholders

- TensorFlow placeholders: dummy nodes that provide entry points for data to computational graph



```
In [15]: input1 = tf.placeholder(tf.float32)
         input2 = tf.placeholder(tf.float32)

In [16]: output = tf.mul(input1, input2)

In [20]: with tf.Session()as sess:
             print(sess.run(output, feed_dict={input1:7., input2:2.}))

         14.0
```

Thank you!

Feedbacks and Questions: rosanne.liu@northwestern.edu