AI Deep Learning: Introduction to TensorFlow

Thuan L Nguyen, PhD

AI Deep Learning: Introduction to TensorFlow

1. TensorFlow: What is It? & Why Now?

2. TensorFlow: What are Tensors?

3. TensorFlow Programs: The Basics

4. TensorFlow Programs: tf.constant and tf.Variable

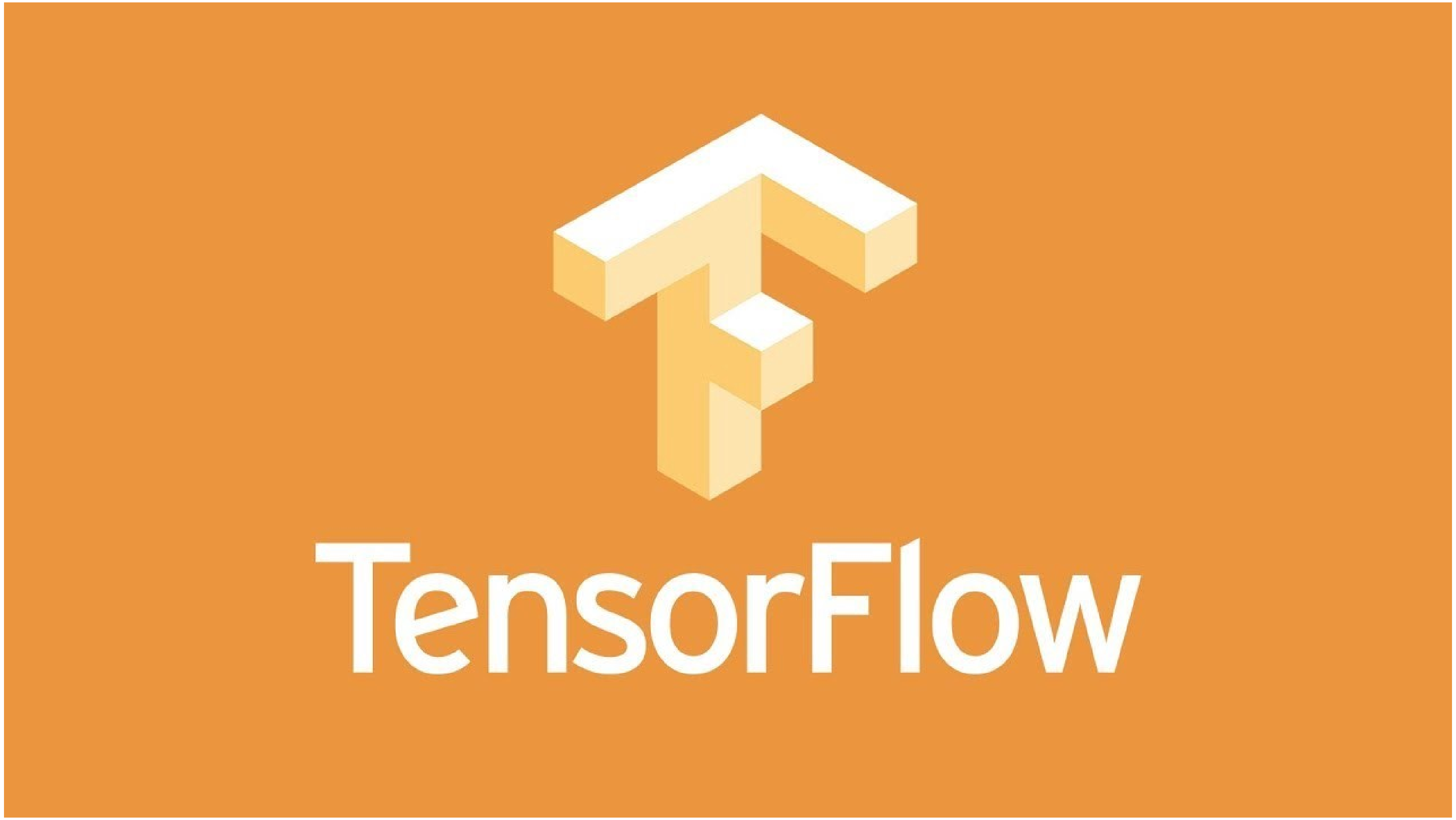
5. TensorFlow Programs: tf.placehoder & Feed Dictionaries

6. TensorFlow Programs: Graph and Session

7. TensorFlow & Python Numpy Library

8. TensorFlow: Artificial Neural Networks with TensorFlow

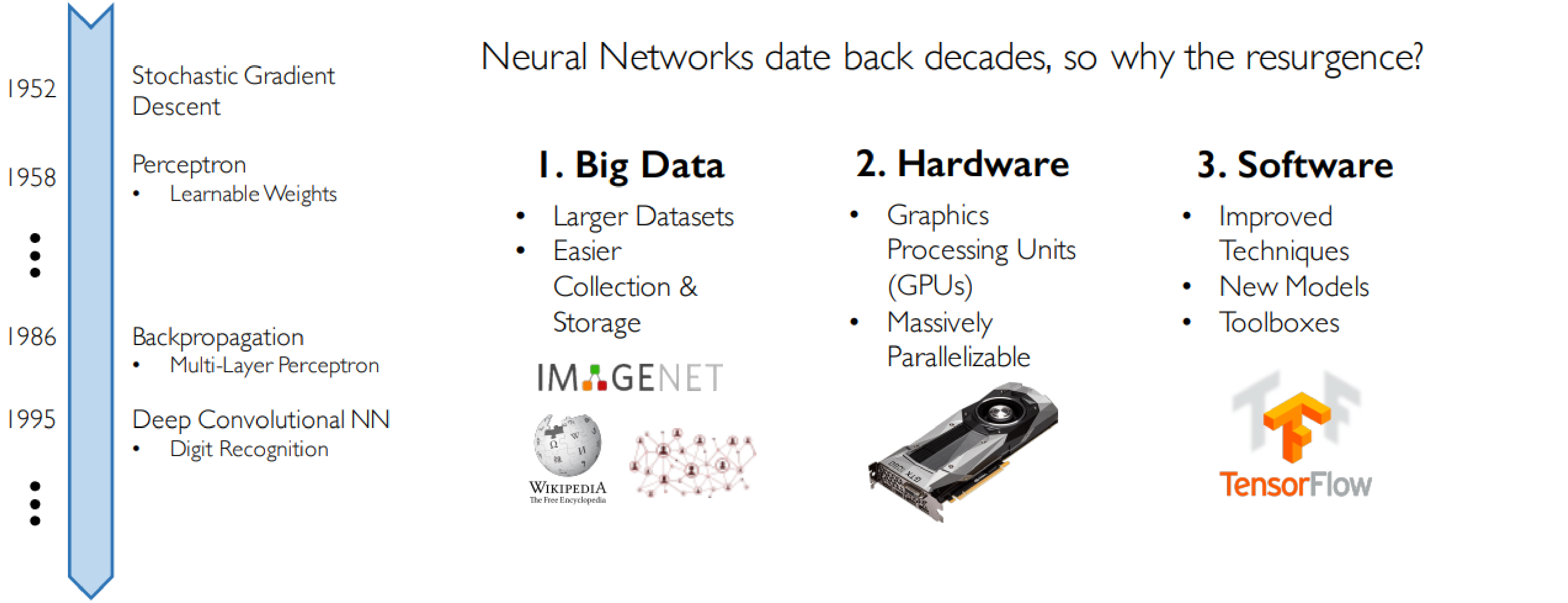
AI Deep Learning: Introduction to TensorFlow



AI Deep Learning: Introduction to TensorFlow

TensorFlow: What is It?

- TensorFlow is a very popular library for deep learning computation.
- It is open-sourced by Google.



1952 Stochastic Gradient Descent

1958 Perceptron
- Learnable Weights

⋮

1986 Backpropagation
- Multi-Layer Perceptron

1995 Deep Convolutional NN
- Digit Recognition

⋮

Neural Networks date back decades, so why the resurgence?

1. Big Data
- Larger Datasets
- Easier Collection & Storage

2. Hardware
- Graphics Processing Units (GPUs)
- Massively Parallelizable

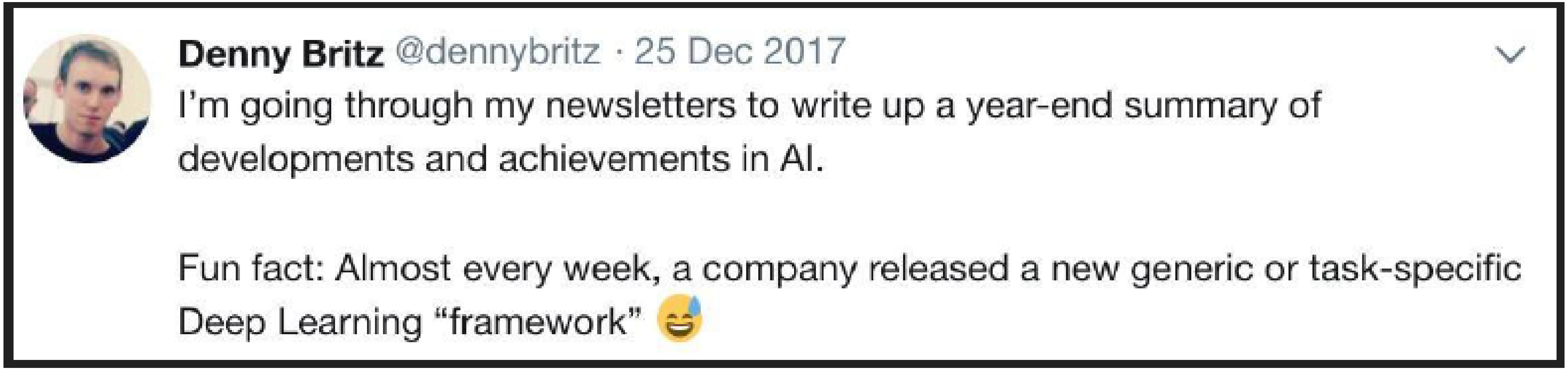
3. Software
- Improved Techniques
- New Models
- Toolboxes

Sources: MIT

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Why?

- Are there any other frameworks that can be used for deep learning computation?
- Yes. Actually, too many!



Denny Britz @dennybritz · 25 Dec 2017

I'm going through my newsletters to write up a year-end summary of developments and achievements in AI.

Fun fact: Almost every week, a company released a new generic or task-specific Deep Learning "framework" 😅

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Why?

- Are there any other frameworks that can be used for deep learning computation?
- Yes. Actually, too many!

- Torch
- Caffe
- Theano (Keras, Lasagne)
- CuDNN
- Tensorflow
- Mxnet
- Etc.

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Why?

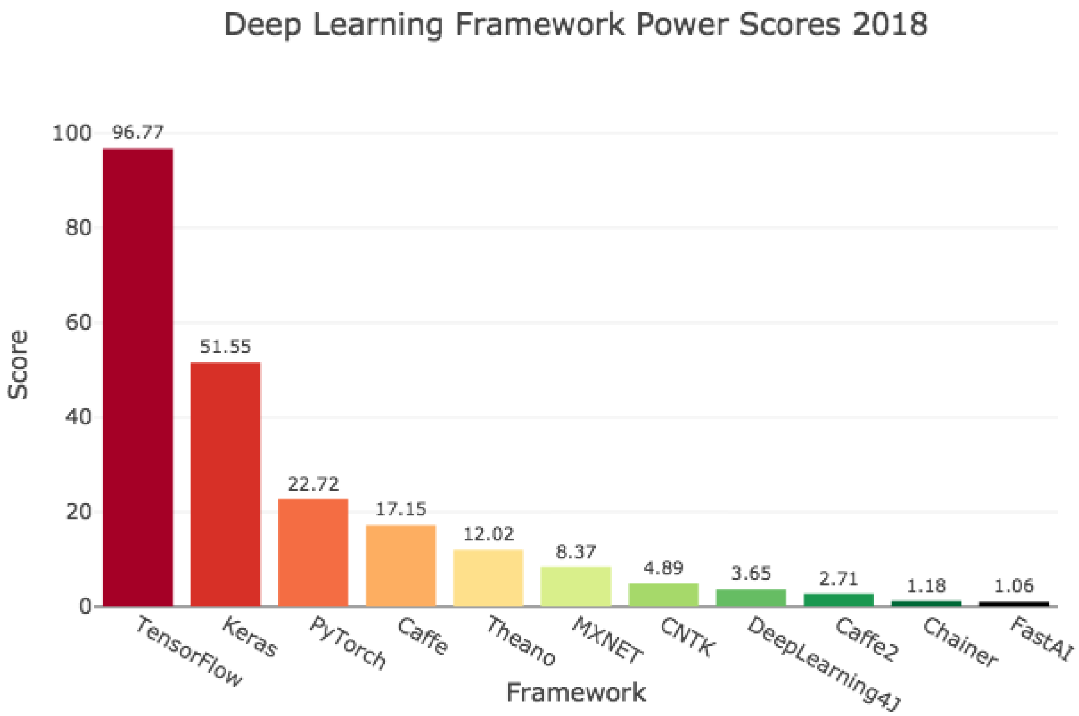
- TensorFlow is a suite of software that has powerful features for deep learning.
  - The features can be used in both academic research and industry production.

- Some major benefits of using TensorFlow:
  - It works with all the cool languages. TensorFlow works with Python, C++, Java, R, and Go.
  - TensorFlow works on multiple platforms, even mobile and distributed.
  - It is supported by all cloud providers – AWS, Google, and Azure.
  - Keras, a high-level neural network API, has been integrated with TensorFlow.
  - It has better computational graph visualizations because it is native.
  - TensorFlow allows model deployment and ease of use in production.
  - TensorFlow has very good community support.
  - TensorFlow is more than a software library; it is a suite of software that includes TensorFlow, TensorBoard, and TensorServing.

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Why?

- TensorFlow is widely used for deep learning projects.
  - It is currently the top deep learning framework in research and production.

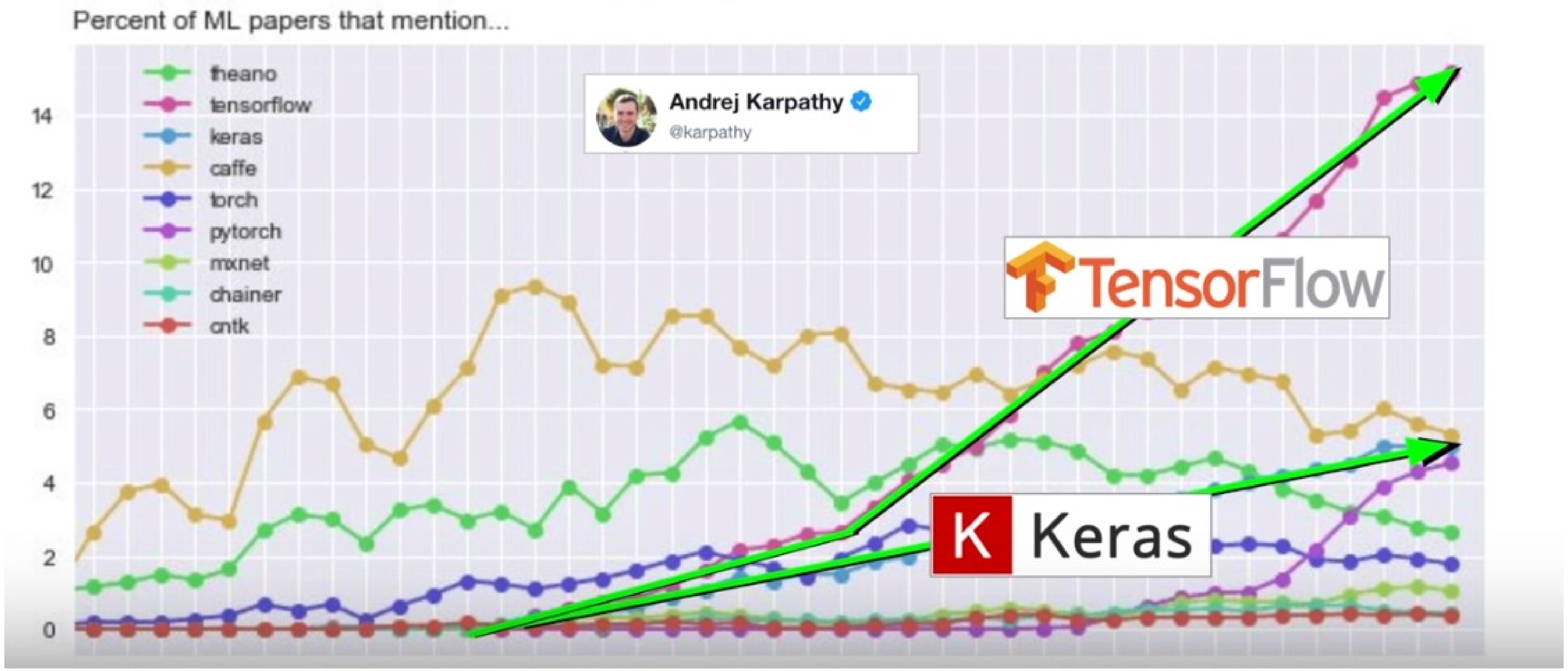


Sources: Jeff Hale – Towards Data Science

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Why?

- TensorFlow is widely used for deep learning projects.
  - It is currently the top deep learning framework in research and production.

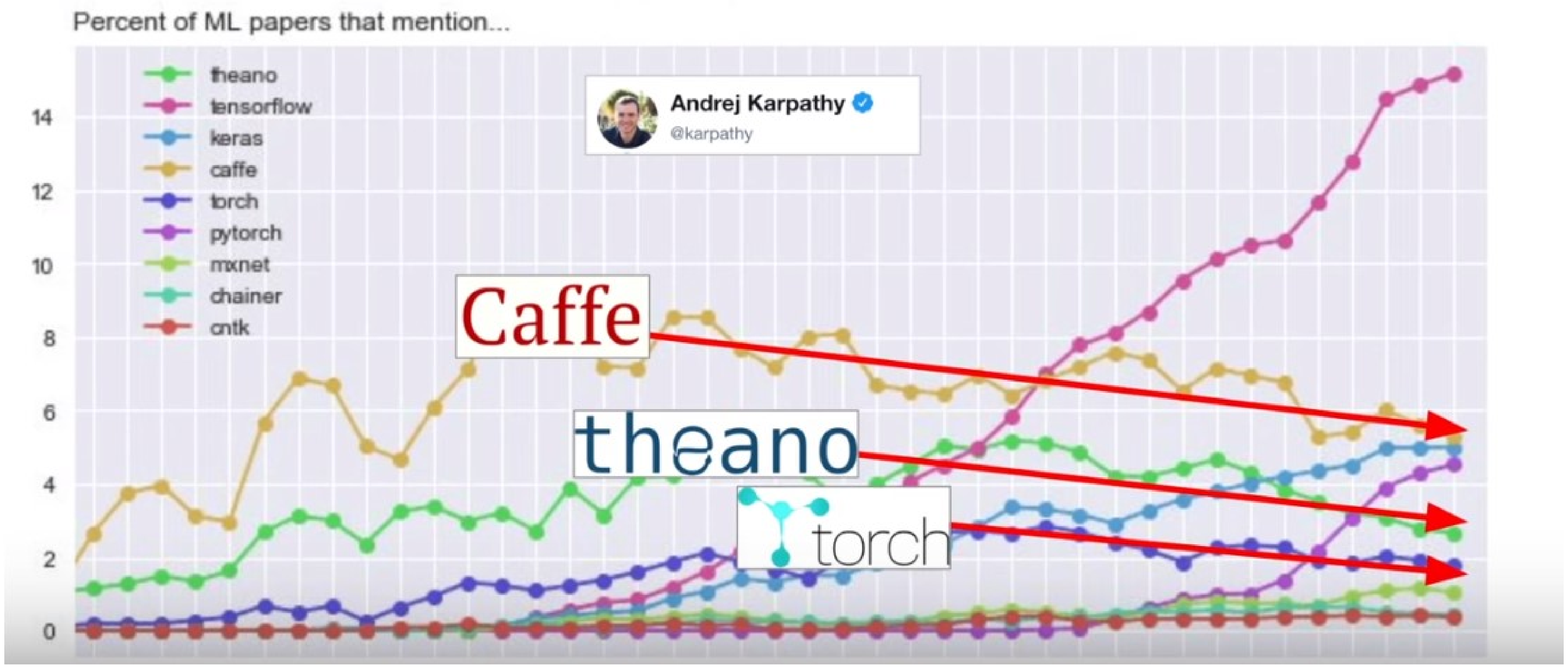


Sources: Andrej Karpathy and Kiwisoft

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Why?

- TensorFlow is widely used for deep learning projects.
  - It is currently the top deep learning framework in research and production.

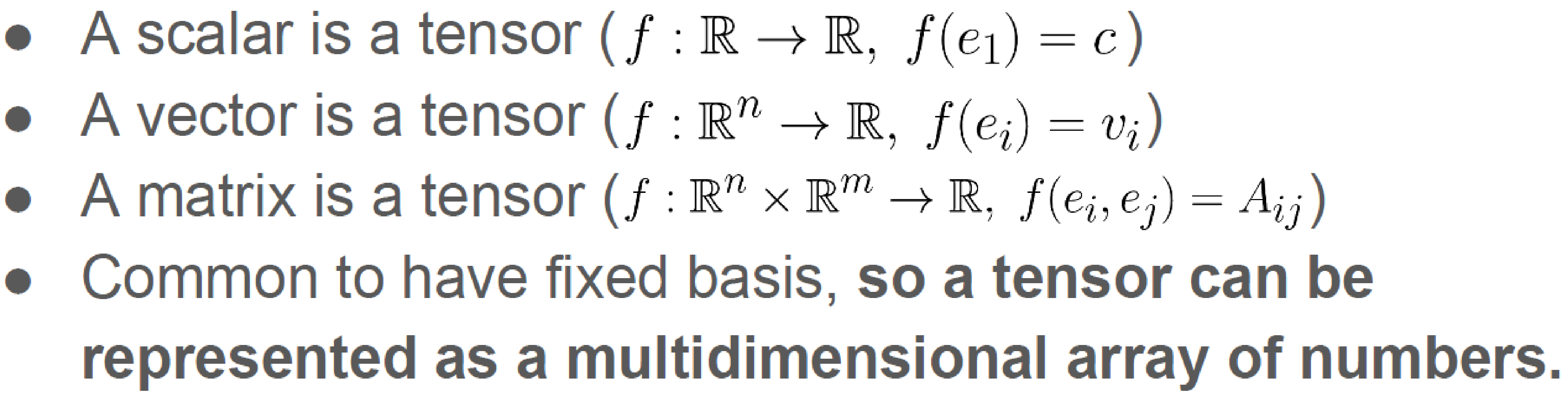


Sources: Andrej Karpathy and Kiwisoft

AI Deep Learning: Introduction to TensorFlow

TensorFlow: What Is a Tensor?

- TensorFlow provides primitives for defining functions on **tensors** and automatically computing their derivatives.



- A scalar is a tensor ($f : \mathbb{R} \rightarrow \mathbb{R},\ f(e_1) = c$)
- A vector is a tensor ($f : \mathbb{R}^n \rightarrow \mathbb{R},\ f(e_i) = v_i$)
- A matrix is a tensor ($f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R},\ f(e_i, e_j) = A_{ij}$)
- Common to have fixed basis, **so a tensor can be represented as a multidimensional array of numbers.**

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Constants and Variables

- In TensorFlow programming: Three basic elements are constants, variables, and placeholder.

- Constants (tf.constant):
  - a constant has a constant value and once you set it, it cannot be changed.
    - a = tf.constant(2, tf.int16)
    - b = tf.constant(4, tf.float32)
    - c = tf.constant(8, tf.float32)

- Variables (tf.Variable):
  - The value of a variable can be changed after it has been set, but its type and shape cannot be changed.
    - d = tf.Variable(2, tf.int16)
    - e = tf.Variable(4, tf.float32)
    - f = tf.Variable(8, tf.float32)

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Placeholders and Feed_Dicts

- Tensorflow also has placeholders:
  - Placeholders do not require an initial value.
  - Placeholders only serve to allocate the necessary amount of memory.
    - During a TensorFlow session, these placeholder can be filled in with (external) data using a feed_dict.
      - *Feed-dict: a python dictionary used to feed data into the computation process*

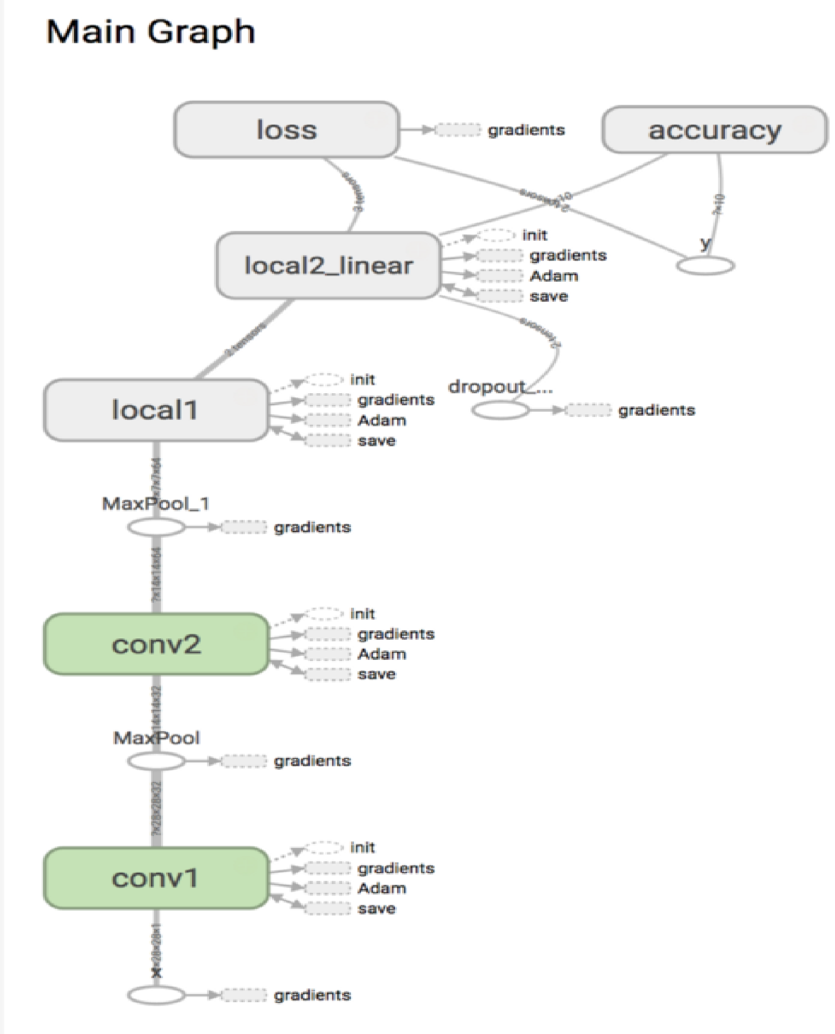
- Placeholders:
  - Placeholders only serve to allocate the necessary amount of memory.
    - point1 = tf.placeholder(tf.float32, shape=(1, 2))
    - point2 = tf.placeholder(tf.float32, shape=(1, 2))

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Graphs and Session

- In Tensorflow, all the variables and the operations done on these variables are saved in a **graph**:
  - After a graph has been built, it contains all of the computational steps necessary for the model.
  - The graph can be run (executed) within a Tensorflow **session**.
  - The session distributes all of the computations across the available CPU and GPU resources.

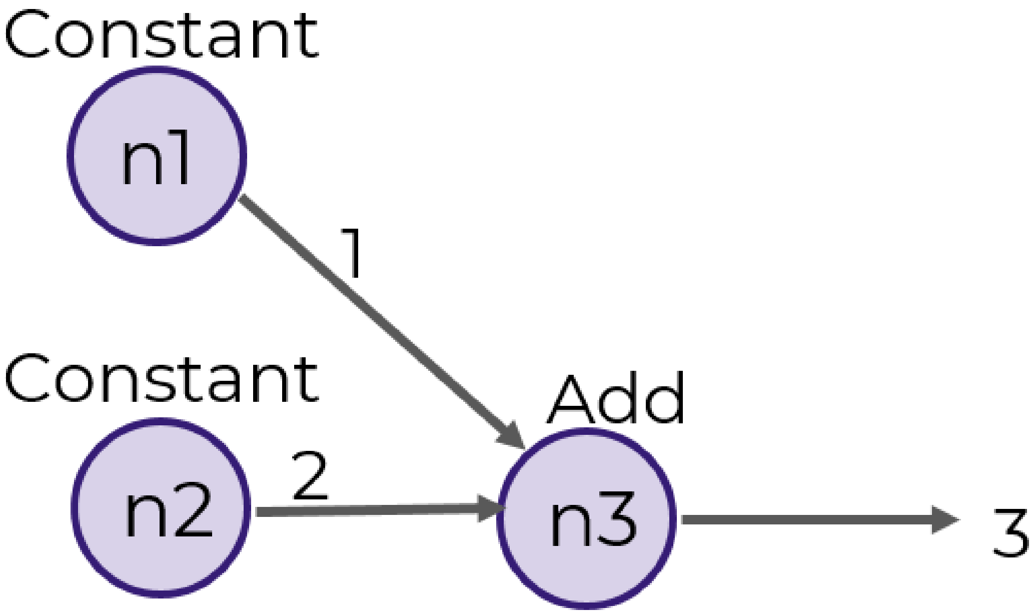
- TensorFlow programs are usually structured into two phases:
  - A construction phase, that assembles a graph.
  - And an execution phase that uses a session to execute operations in the graph.
  - All computations add nodes to global default graph.



AI Deep Learning: Introduction to TensorFlow

TensorFlow: Graph

- Graphs are sets of connected nodes (vertices).
  - The connections are referred to as edges.
  - In TensorFlow: Each node is an operation with possible inputs that can supply some output.



AI Deep Learning: Introduction to TensorFlow

TensorFlow: Graphs and Session

- Graphs are sets of connected nodes (vertices).
    - The connections are referred to as edges.
    - In TensorFlow: Each node is an operation with possible inputs that can supply some output.

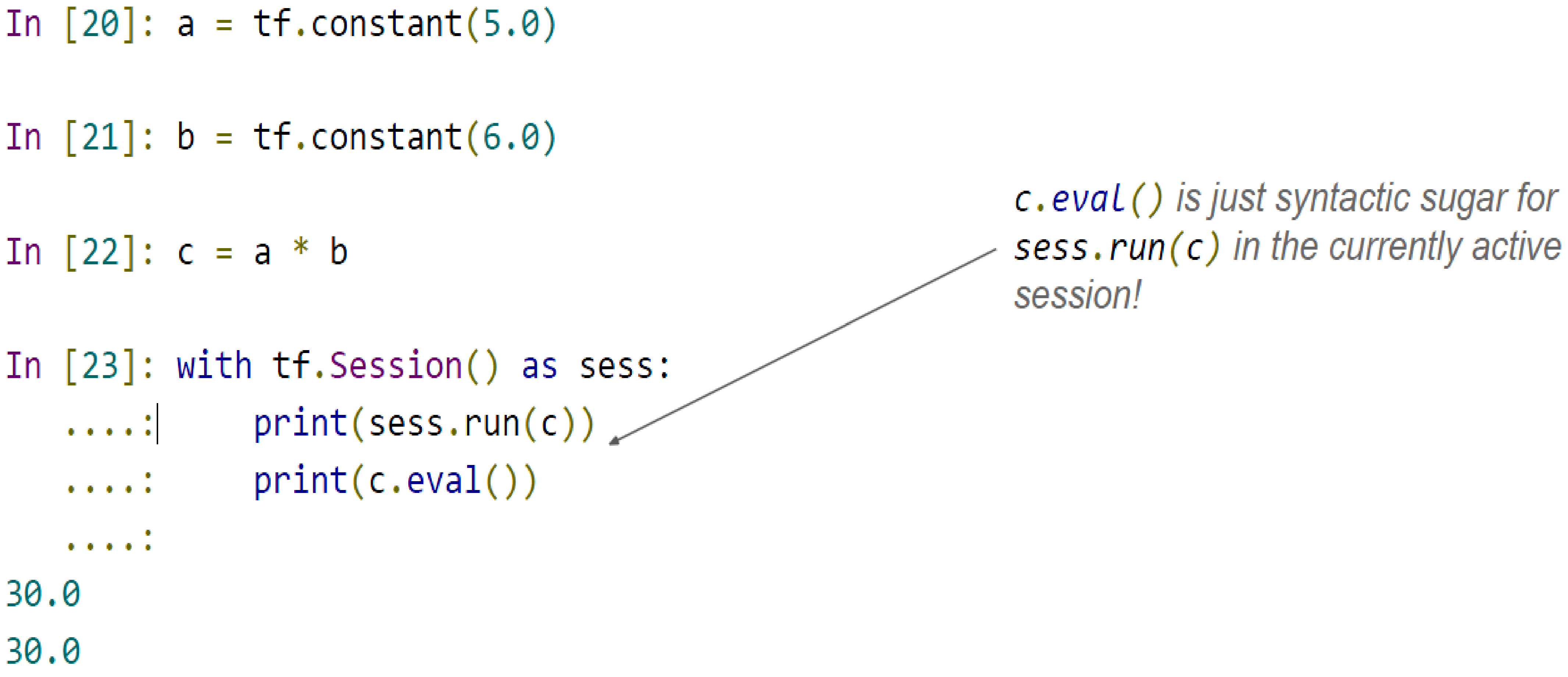
- Examples of graphs and the session:

```
graph = tf.Graph()
with graph.as_default():
    a = tf.Variable(8, tf.float32)
    b = tf.Variable(tf.zeros([2,2], tf.float32))

with tf.Session(graph=graph) as sess:
    anInitializer = tf.global_variables_initializer()
    sess.run(anInitializer)
    print(sess.run(a))
    print(sess.run(b))
```

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Session

- A **Session** object encapsulates the environment in which Tensor objects are evaluated



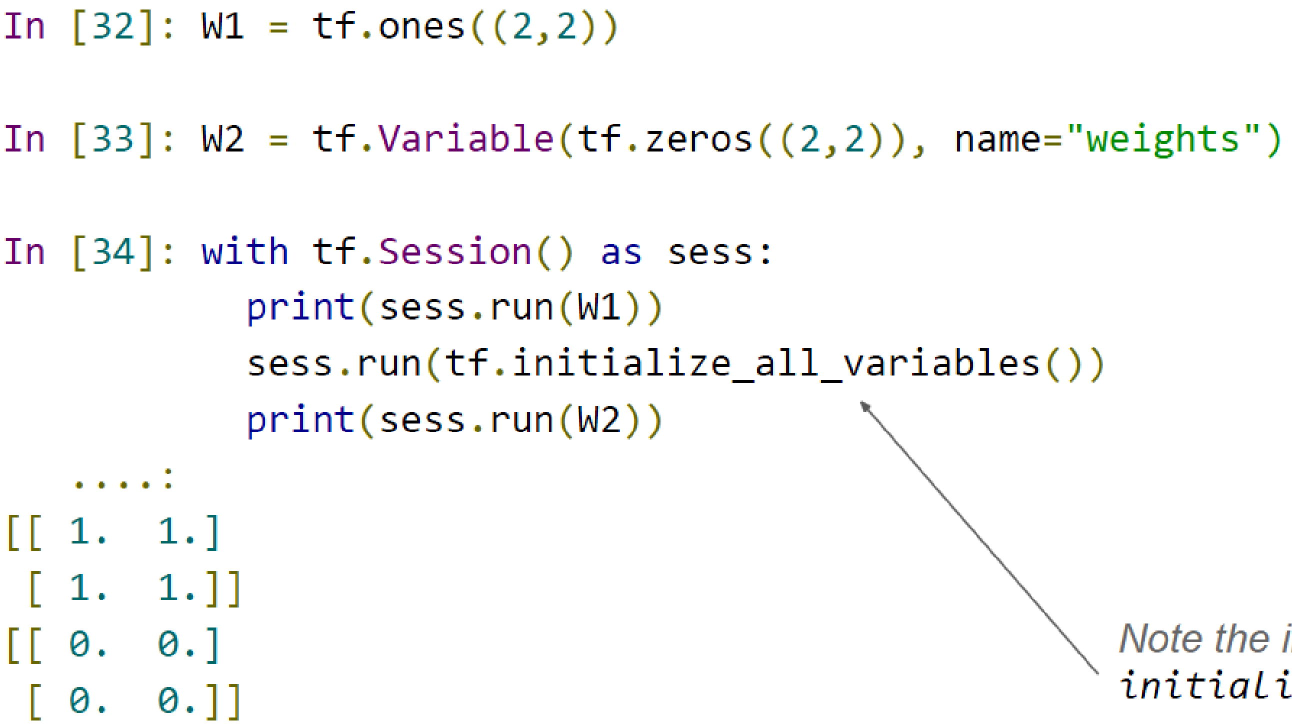
```
In [20]: a = tf.constant(5.0)

In [21]: b = tf.constant(6.0)

In [22]: c = a * b

In [23]: with tf.Session() as sess:
    ....:     print(sess.run(c))
    ....:     print(c.eval())
    ....:
30.0
30.0
```

`c.eval()` is just syntactic sugar for `sess.run(c)` in the currently active session!

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Variables

- When you train a model you use variables to hold and update parameters.
  - Variables are in-memory buffers containing tensors



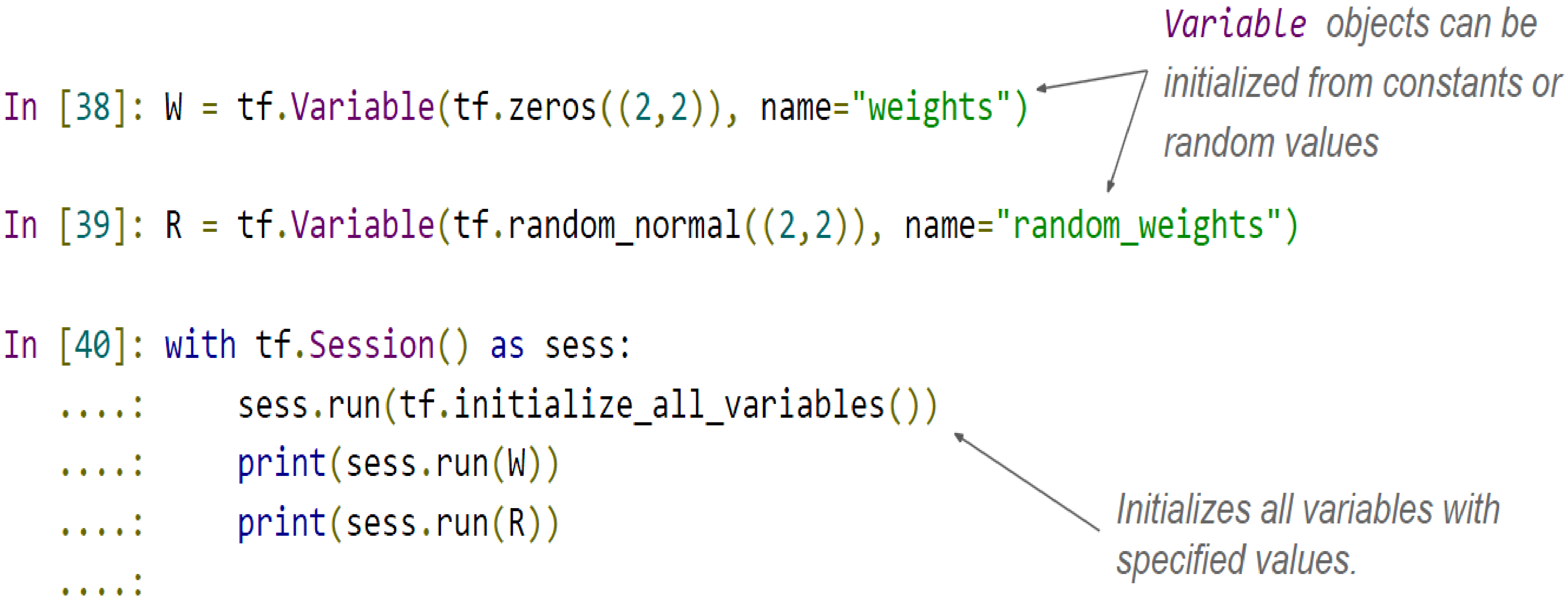
```
In [32]: W1 = tf.ones((2,2))

In [33]: W2 = tf.Variable(tf.zeros((2,2)), name="weights")

In [34]: with tf.Session() as sess:
           print(sess.run(W1))
           sess.run(tf.initialize_all_variables())
           print(sess.run(W2))
   ....:
[[ 1.  1.]
 [ 1.  1.]]
[[ 0.  0.]
 [ 0.  0.]]
```

Note the initialization step `tf.initialize_all_variables()`

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Variables

- TensorFlow variables must be initialized before they can be used.
    - Contrast with constant tensors.



```
In [38]: W = tf.Variable(tf.zeros((2,2)), name="weights")

In [39]: R = tf.Variable(tf.random_normal((2,2)), name="random_weights")

In [40]: with tf.Session() as sess:
   ....:     sess.run(tf.initialize_all_variables())
   ....:     print(sess.run(W))
   ....:     print(sess.run(R))
   ....:
```

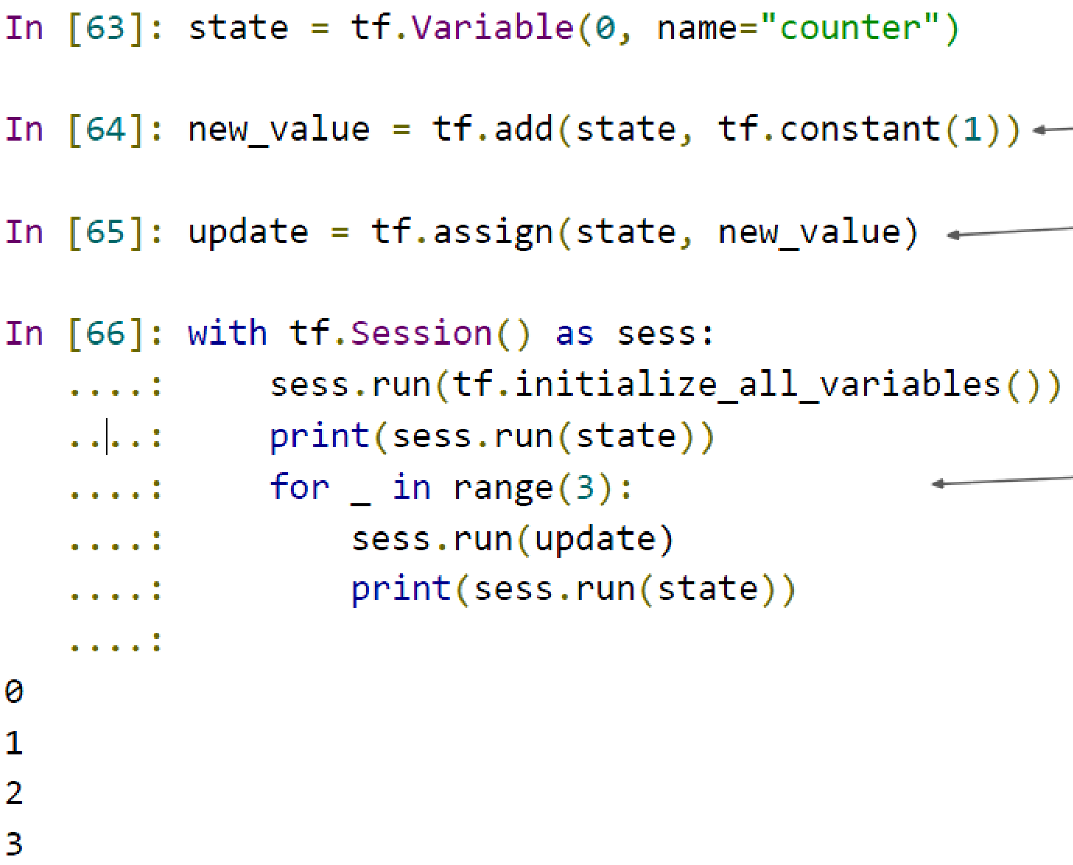
`Variable` objects can be initialized from constants or random values

Initializes all variables with specified values.

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Variables

- TensorFlow variables can be updated via its state



```
In [63]: state = tf.Variable(0, name="counter")

In [64]: new_value = tf.add(state, tf.constant(1))

In [65]: update = tf.assign(state, new_value)

In [66]: with tf.Session() as sess:
    ....:     sess.run(tf.initialize_all_variables())
    ....:     print(sess.run(state))
    ....:     for _ in range(3):
    ....:         sess.run(update)
    ....:         print(sess.run(state))
    ....:
0
1
2
3
```

Roughly `new_value = state + 1`

Roughly `state = new_value`

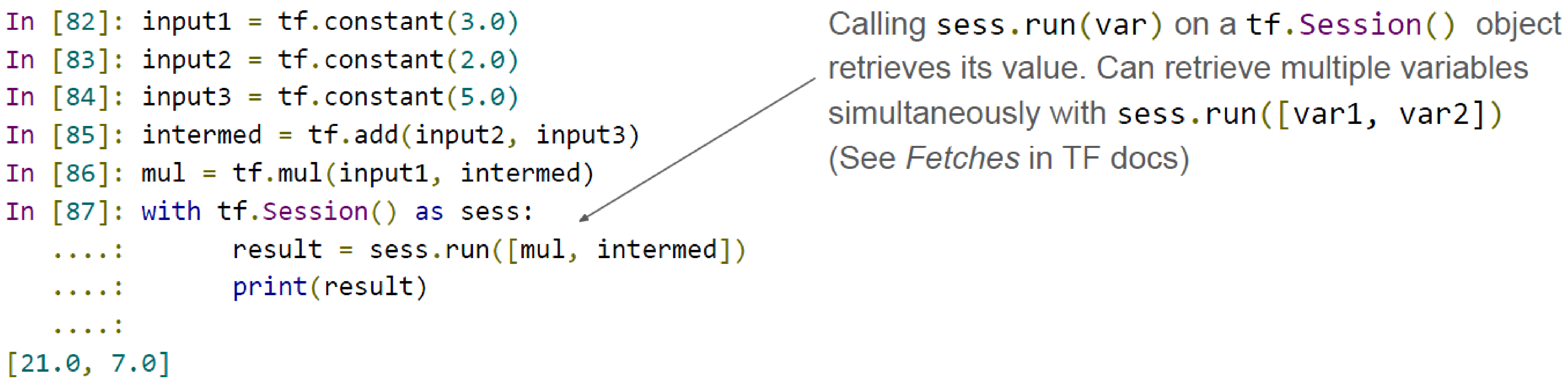
Roughly

```
state = 0
print(state)
for _ in range(3):
   state = state + 1
   print(state)
```

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Variables

- The states of TensorFlow variables can be retrieved by running a session.



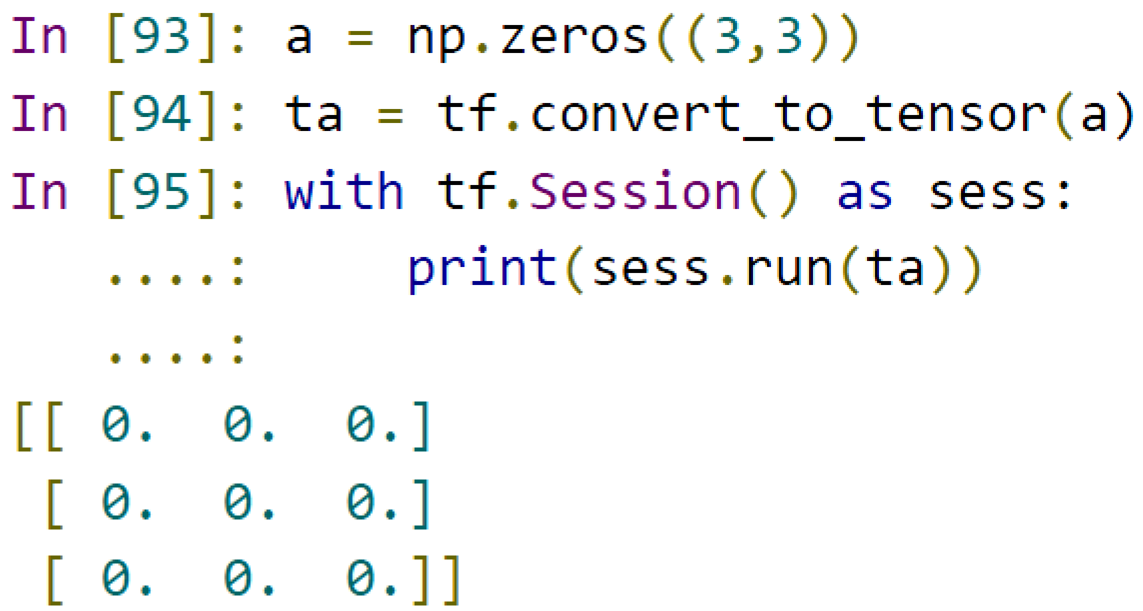
```
In [82]: input1 = tf.constant(3.0)
In [83]: input2 = tf.constant(2.0)
In [84]: input3 = tf.constant(5.0)
In [85]: intermed = tf.add(input2, input3)
In [86]: mul = tf.mul(input1, intermed)
In [87]: with tf.Session() as sess:
   ....:       result = sess.run([mul, intermed])
   ....:       print(result)
   ....:
[21.0, 7.0]
```

Calling `sess.run(var)` on a `tf.Session()` object retrieves its value. Can retrieve multiple variables simultaneously with `sess.run([var1, var2])` (See *Fetches* in TF docs)

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Inputting Data Directly from Numpy

- It is possible to feed data into a TensorFlow program by converting data from Numpy.

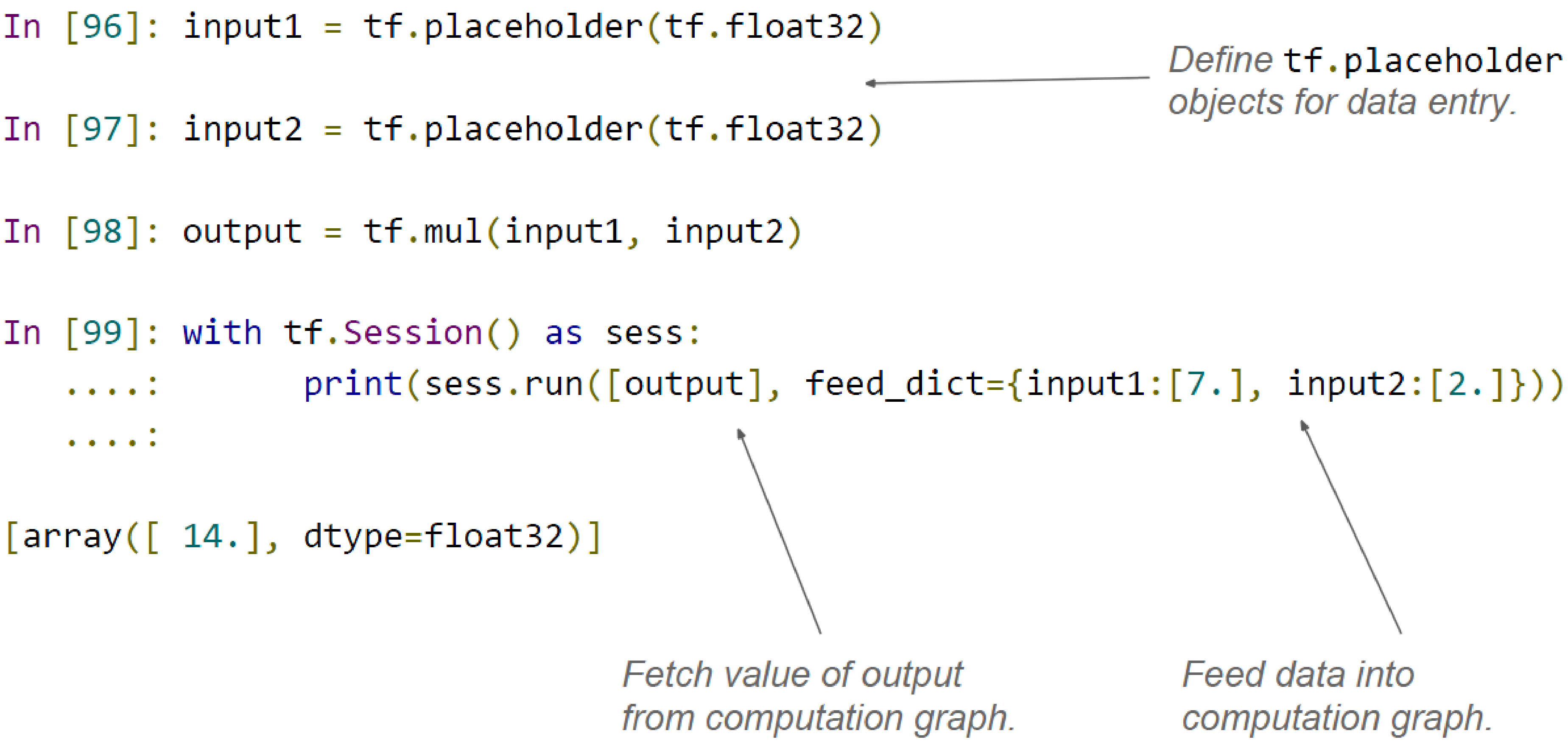


```
In [93]: a = np.zeros((3,3))
In [94]: ta = tf.convert_to_tensor(a)
In [95]: with tf.Session() as sess:
    ....:     print(sess.run(ta))
    ....:
[[ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]]
```

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Inputting Data Using Placeholders & Feed Dictionaries

- It is possible to feed data into a TensorFlow program using tf.placeholder and feed dictionaries.



```
In [96]: input1 = tf.placeholder(tf.float32)

In [97]: input2 = tf.placeholder(tf.float32)

In [98]: output = tf.mul(input1, input2)

In [99]: with tf.Session() as sess:
   ....:     print(sess.run([output], feed_dict={input1:[7.], input2:[2.]}))
   ....:

[array([ 14.], dtype=float32)]
```

Define `tf.placeholder` *objects for data entry.*

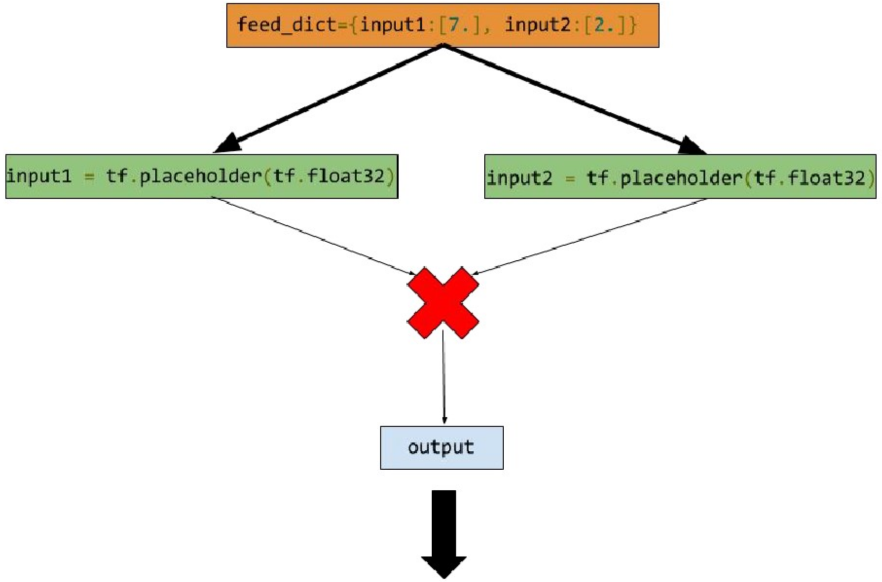
Fetch value of output from computation graph.

Feed data into computation graph.

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Inputting Data Using Placeholders & Feed Dictionaries

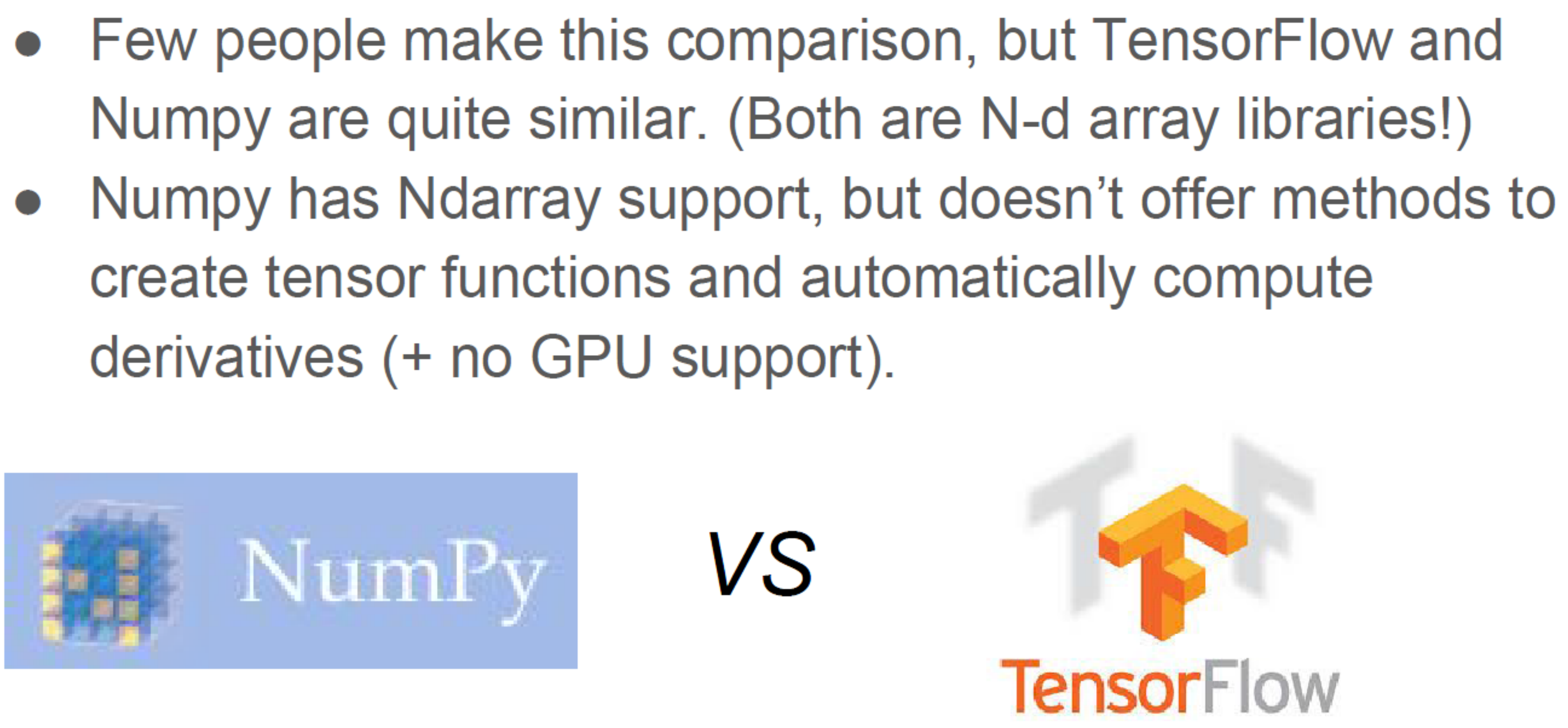
- It is possible to feed data into a TensorFlow program using tf.placeholder and feed dictionaries.
    - Placeholders are initially empty.
    - They are used to feed in the actual training examples.
    - However they do need a declared expected data type (tf.float32) with an optional shape argument.


feed_dict={input1:[7.], input2:[2.]}
input1 = tf.placeholder(tf.float32)
input2 = tf.placeholder(tf.float32)
output

AI Deep Learning: Introduction to TensorFlow

TensorFlow: vs Numpy

- TensorFlow seems to be similar to Python Numpy library.
- Are they the same?



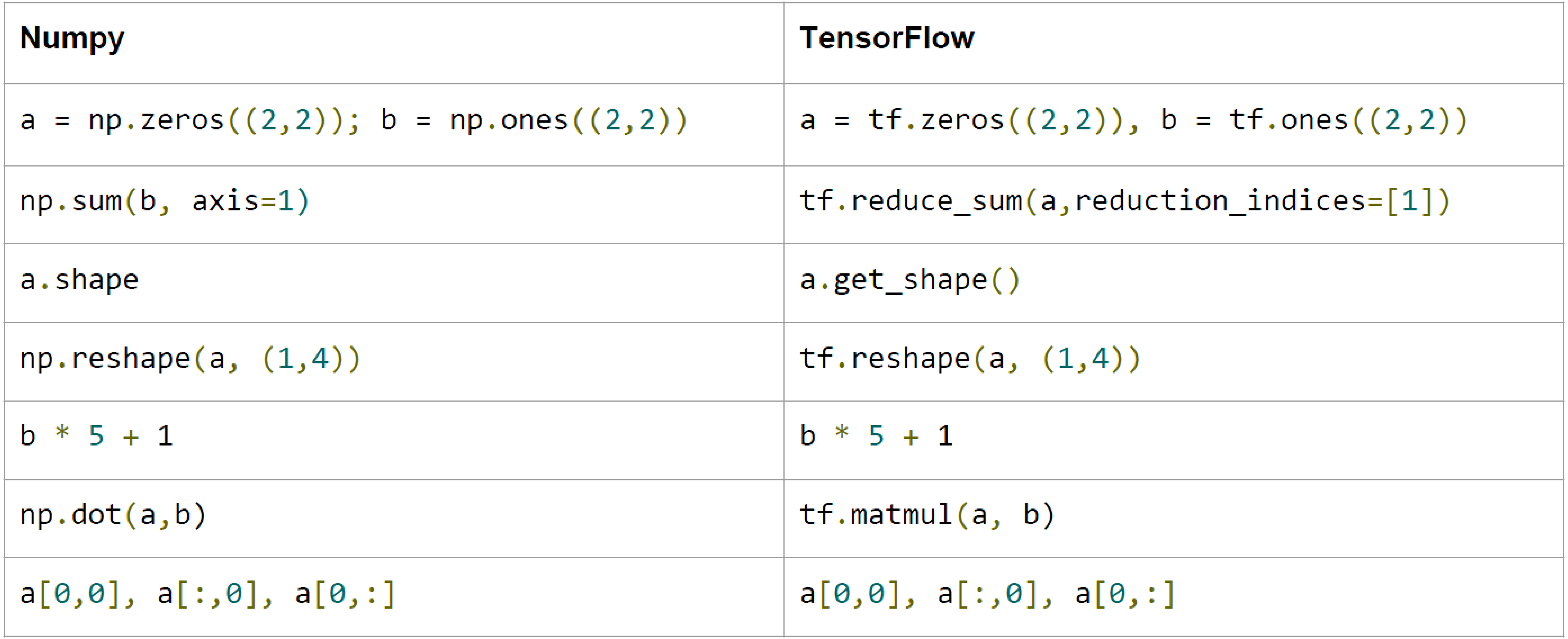
- Few people make this comparison, but TensorFlow and Numpy are quite similar. (Both are N-d array libraries!)
- Numpy has Ndarray support, but doesn't offer methods to create tensor functions and automatically compute derivatives (+ no GPU support).

NumPy
VS
TensorFlow

AI Deep Learning: Introduction to TensorFlow

TensorFlow: vs Numpy

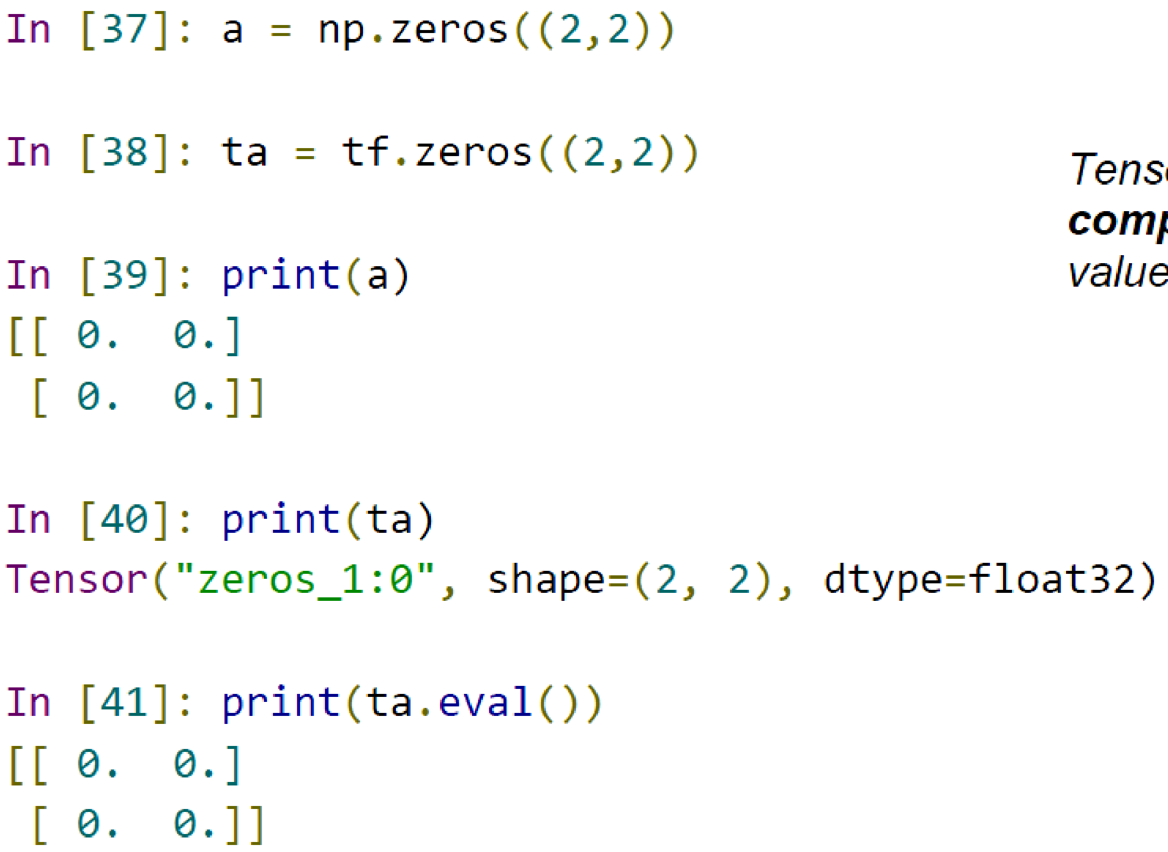
- TensorFlow seems to be similar to Python Numpy library.
- Are they the same?



Numpy	TensorFlow
`a = np.zeros((2,2)); b = np.ones((2,2))`	`a = tf.zeros((2,2)), b = tf.ones((2,2))`
`np.sum(b, axis=1)`	`tf.reduce_sum(a,reduction_indices=[1])`
`a.shape`	`a.get_shape()`
`np.reshape(a, (1,4))`	`tf.reshape(a, (1,4))`
`b * 5 + 1`	`b * 5 + 1`
`np.dot(a,b)`	`tf.matmul(a, b)`
`a[0,0], a[:,0], a[0,:]`	`a[0,0], a[:,0], a[0,:]`

TensorFlow: vs Numpy

- TensorFlow seems to be similar to Python Numpy library.
- Are they the same?



```
In [37]: a = np.zeros((2,2))

In [38]: ta = tf.zeros((2,2))

In [39]: print(a)
[[ 0.  0.]
 [ 0.  0.]]

In [40]: print(ta)
Tensor("zeros_1:0", shape=(2, 2), dtype=float32)

In [41]: print(ta.eval())
[[ 0.  0.]
 [ 0.  0.]]
```

TensorFlow computations define a ***computation graph*** *that has no numerical value until evaluated!*

AI Deep Learning: Introduction to TensorFlow

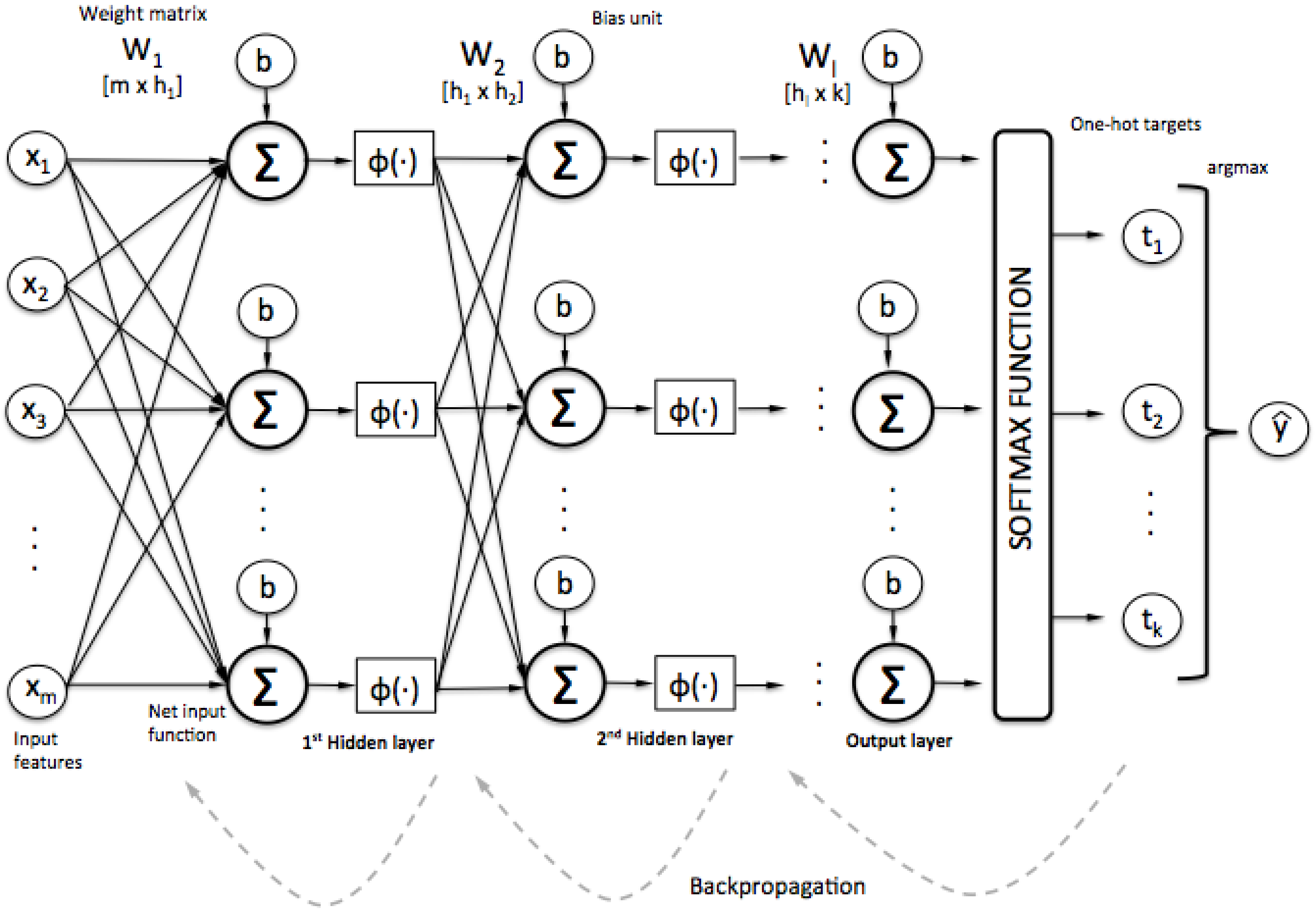
TensorFlow: Tensors in Flows

- The **tensor** and how does it have 'flow'?
  - **Tensors are everywhere** in AI machine learning mathematic expressions:
    - A vector is a list of values.
    - A matrix is a table (or list of lists).
    - And more:
      - a list of tables (or list of lists of lists)
      - a table of tables (or list of lists of tables…).
      - And so on.

  - Let's take a multi-layer neural network as an example:
    - Input data features ('x1', 'x2', …) going through 2 hidden layers:
      - Each with nodes ('neurons')
      - Each with weights ('W') and bias ('b')
    - Output is y.

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Tensors in Flows



Sources: gk_ at https://chatbotslife.com/

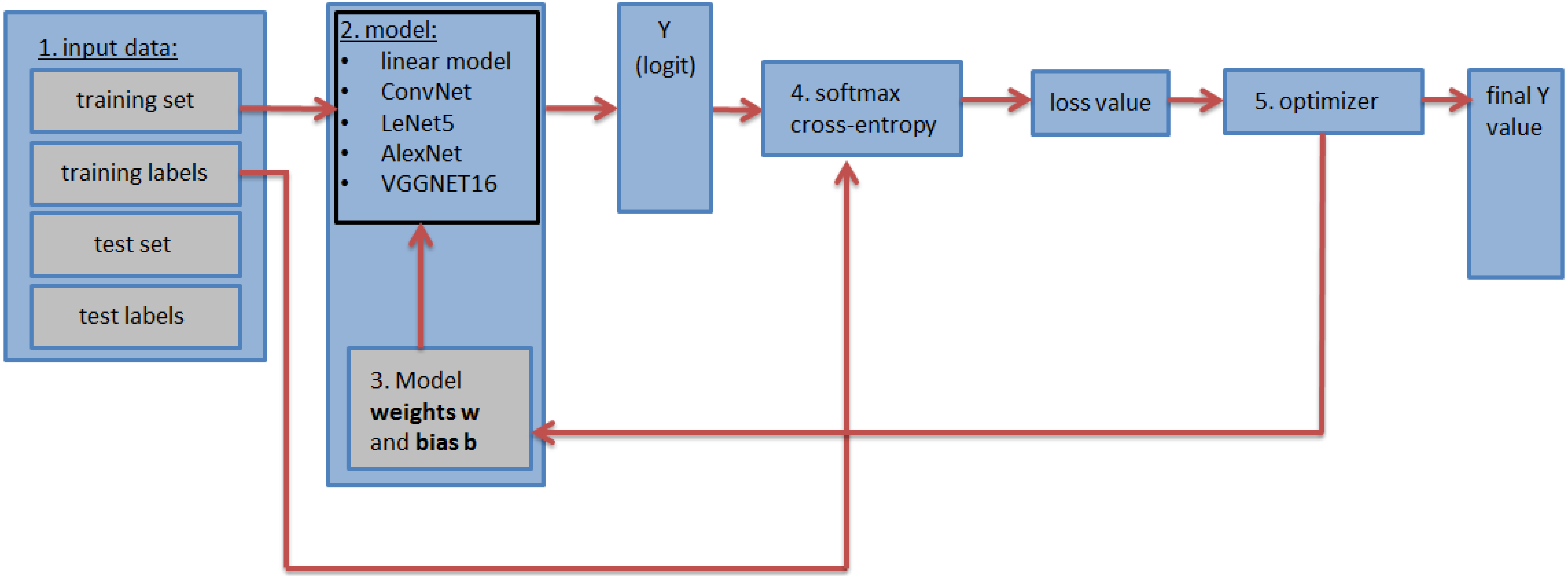
AI Deep Learning: Introduction to TensorFlow

TensorFlow: Artificial Neural Networks with Tensorflow

- The graph containing the Neural Network should contain the following components
  - The **input datasets**: the training and the test dataset and labels (and the validation dataset and labels).
    - The test and validation datasets can be placed inside a tf.constant().
    - The training dataset is placed in a tf.placeholder() so that it can be feeded in batches during the training (stochastic gradient descent).
  - The **artificial neural network model** with all of its layers.
  - The **weight matrices and bias vectors** defined in the proper shape and initialized to their initial values. (One weight matrix and bias vector per layer.)
  - The **softmax cross-entropy** and **loss value**:
    - The model has as output the logit vector (estimated training labels).
    - By comparing the logit with the actual labels, it is possible to calculate the loss value (with the softmax with cross-entropy function).
    - The loss value is an indication of how close the estimated training labels are to the actual training labels and will be used to update the weight values.
  - An **optimizer**, which will use the calculated loss value to update the weights and biases with backpropagation.

AI Deep Learning: Introduction to TensorFlow

TensorFlow: Artificial Neural Networks in Tensorflow



Sources: Ahmet Taspinar