TensorFlow

HANUMANTHARAYAPPA, JIALI

What is TensorFlow?

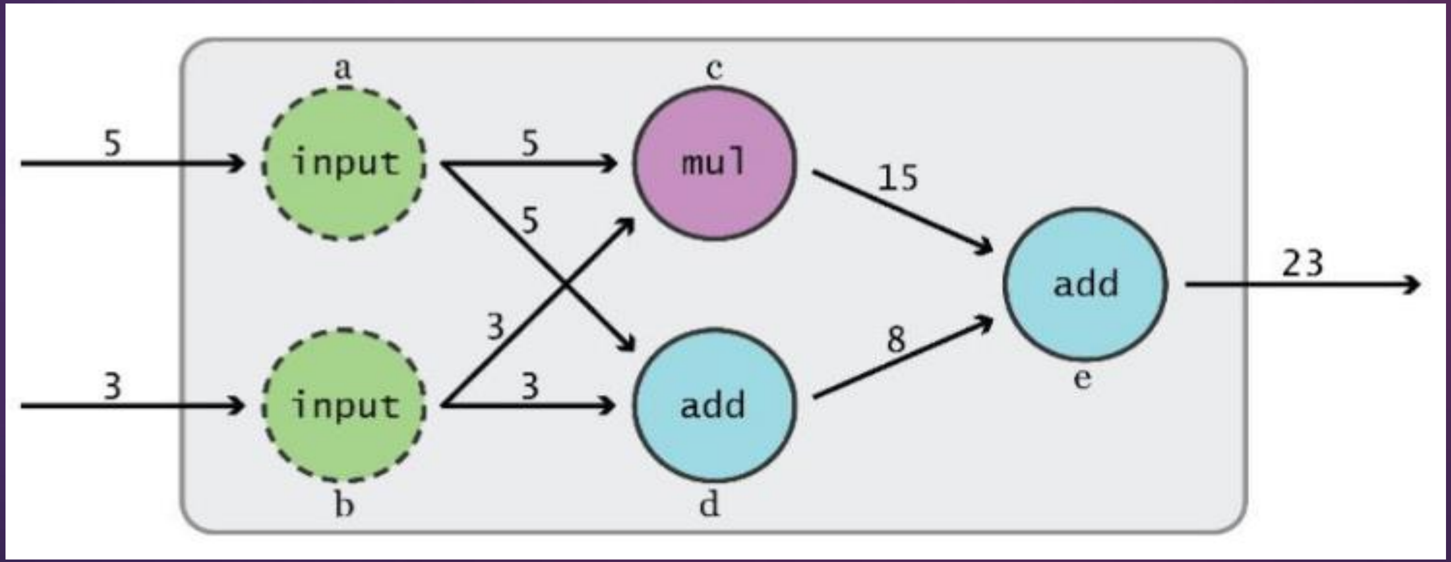
- It’s a open source library for numerical computation.
- Developed and maintained by Google.
- Optimized for Deep Learning.
- TensorFlow provides primitive for defining functions on tensors and automatically computing their gradients
- It’s similar to Numpy package in Python(there are a couple of differences!)

Why TensorFlow?

- Built -in support for distributed computing
  - We can train a Deep Learning network on multiple bank of GPUs.
  - Train portion of the Network on CPU and rest on GPUs
- Auto-differentiation (no more taking gradients by hand ☺)
- TensorFlow support Python API, like Numpy. So, learning curve is small
  - It’s as simple as import tensoflow as tf (similar to import numpy as np)
- Fully compatible with Python and C++
  - Front-end is python and everyone loves coding in Python ☺

Graphs and Session

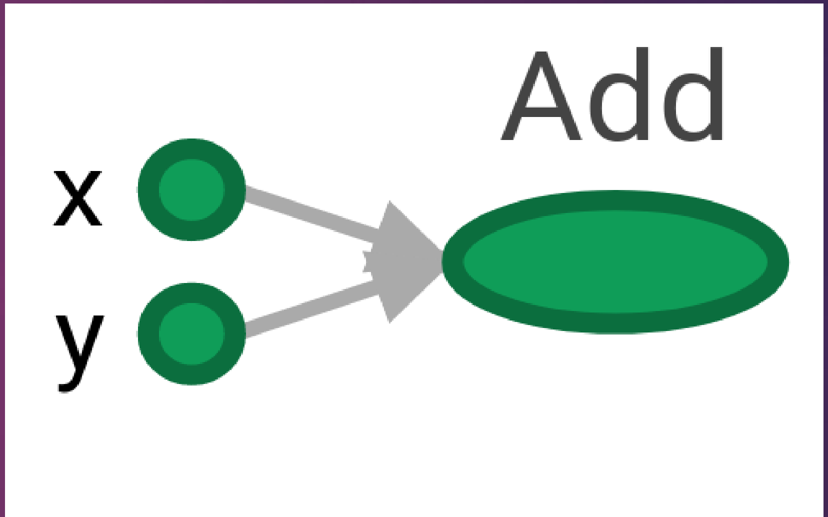
- In TensorFlow all the computations are represented in dataflow graphs
  - First step in computing is to assemble the computational graph
  - To perform computation we need to create a session and evaluate the graph inside the session



Dataflow Graphs

```
import tensorflow as tf
a = tf.add(3, 5)
```

Interpreted Graph from Tensorboard

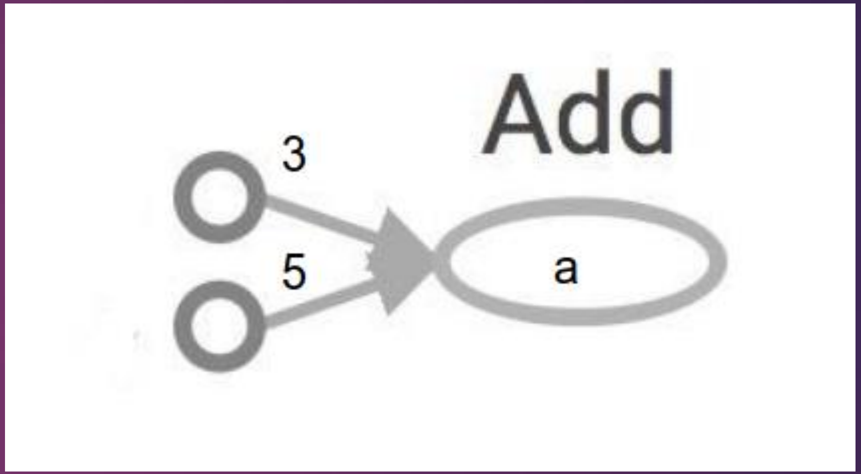


- Why x and y in the graph?
- TensorFlow automatically names the nodes when you don't explicitly name them

Continued..

```
import tensorflow as tf
a = tf.add(3, 5)
```

Interpreted Graph from Tensorboard

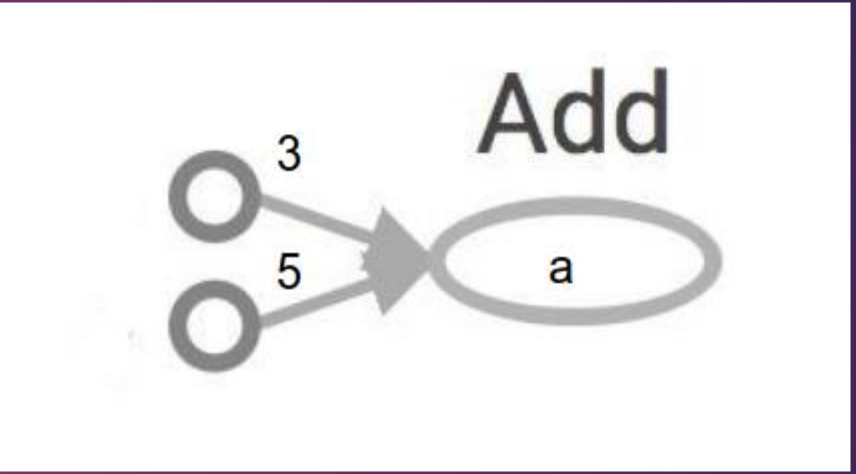


- Nodes: operators, variables, and constants
- Edges: tensors

Create a Session to evaluate graph

```
import tensorflow as tf
a = tf.add(3, 5)
sess =tf. Session()
sess.run(print a)
sess.close()
```

Interpreted Graph from Tensorboard

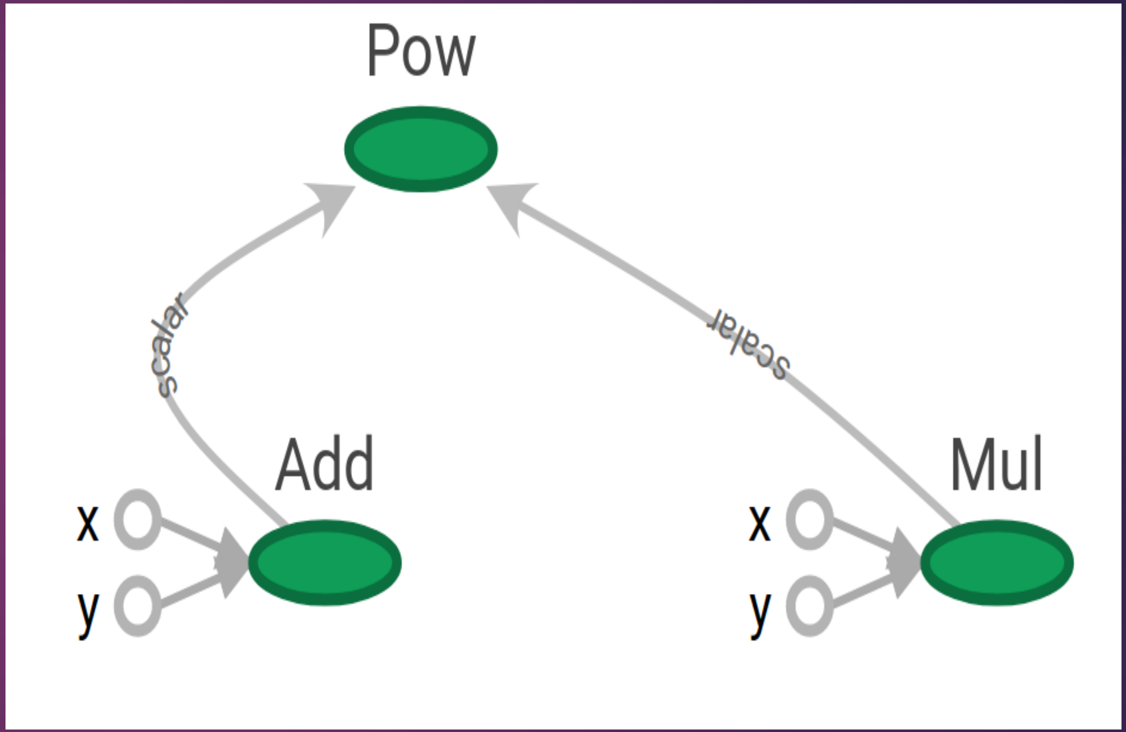


- Create a session and evaluate the graph inside the session
- Session object encapsulates the environment in which Operation objects are executed, and Tensor objects are evaluated.

More Graphs

```
import tensorflow as tf
X = 5
Y = 5
op1 = tf.add(X,Y)
op2 = tf.multiply(X,Y)
op3 = tf.pow(op2, op1)
with tf.Session() as sess:
    op3 = sess.run(op3)
```

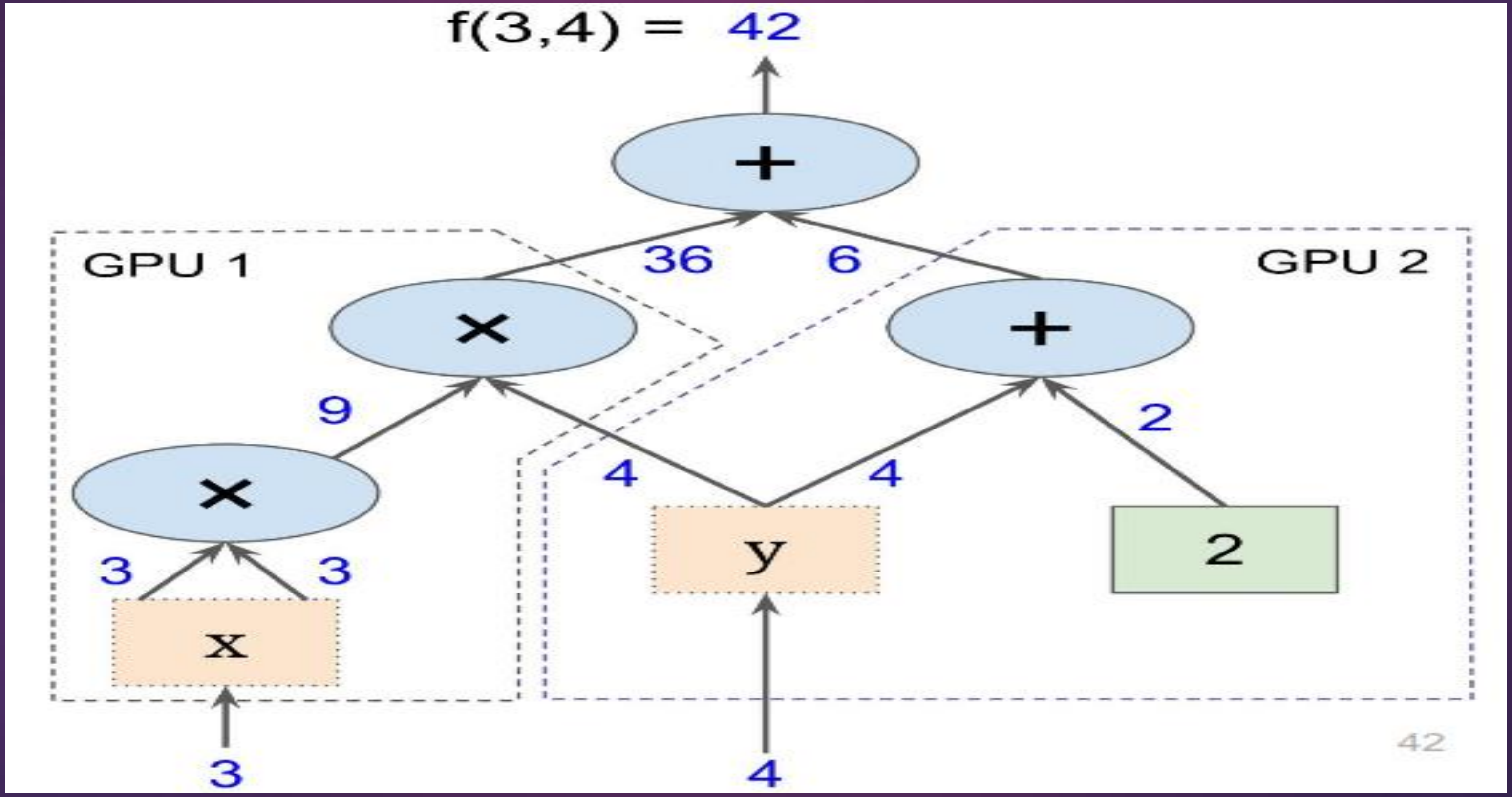
Interpreted Graph from Tensorboard



- Three nodes because we have three computations

Distributed Computing

- We can break graphs into several subgraphs and run them parallelly across multiple CPUs, GPUs, or devices



CPU as the Computing Device

- We can select the compute device as in the below code snippet

```
import tensorflow as tf
with tf.device('/cpu:0'):
    a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3], name='a')
    b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2], name='b')
    c = tf.matmul(a, b)

with tf.Session() as sess:
    print (sess.run(c))
~
```

CPU as the Compute Device!

GPU as the Computing Device

- We can select the compute device as in the below code snippet

```
import tensorflow as tf
with tf.device('/gpu:0'):
    a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3], name='a')
    b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2], name='b')
    c = tf.matmul(a, b)

with tf.Session() as sess:
    print (sess.run(c))
~
```

GPU as the Compute Device!

Heterogenous Computing!

- We can select the compute devices as in the below code snippet

```
import tensorflow as tf

with tf.device('/gpu:0'):
    a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3], name='a')
    b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2], name='b')
    c = tf.matmul(a, b)

with tf.device('/cpu:0'):
    c = tf.multiply(c, 2)

with tf.Session() as sess:
    print (sess.run(c))
~
~
```

GPU Performs portion of the computation!

CPU Performs portion of the computation!

Continued..

```
MatMul: (MatMul): /job:localhost/replica:0/task:0/device:GPU:0
2017-12-03 10:21:39.456439: I tensorflow/core/common_runtime/placer.cc:874] MatMul: (MatMul)/job:localhost/replica:0/task:0/device:GPU:0
Mul: (Mul): /job:localhost/replica:0/task:0/device:CPU:0
2017-12-03 10:21:39.456471: I tensorflow/core/common_runtime/placer.cc:874] Mul: (Mul)/job:localhost/replica:0/task:0/device:CPU:0
Mul/y: (Const): /job:localhost/replica:0/task:0/device:CPU:0
2017-12-03 10:21:39.456482: I tensorflow/core/common_runtime/placer.cc:874] Mul/y: (Const)/job:localhost/replica:0/task:0/device:CPU:0
b: (Const): /job:localhost/replica:0/task:0/device:GPU:0
2017-12-03 10:21:39.456491: I tensorflow/core/common_runtime/placer.cc:874] b: (Const)/job:localhost/replica:0/task:0/device:GPU:0
a: (Const): /job:localhost/replica:0/task:0/device:GPU:0
2017-12-03 10:21:39.456513: I tensorflow/core/common_runtime/placer.cc:874] a: (Const)/job:localhost/replica:0/task:0/device:GPU:0
[[  44.   56.]
 [  98.  128.]]
```

CPU portion of the computation!

GPU portion of the computation!

Why Graphs?

- Save computation time
  - Only run subgraphs whose result is needed
- We can divide computation into small manageable graphs
  - This makes our life easier in computing the gradient
- We can exploit distributed computing
  - Scatter work across CPUs and multi-bank GPUs
- It's more intuitive to think of machine learning problems as graphs
  - This makes performing forward and backward propagation easier

Demo Time ☺

Thank you