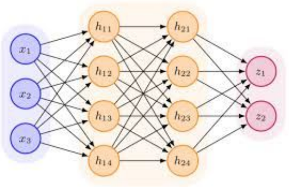


Tensorflow

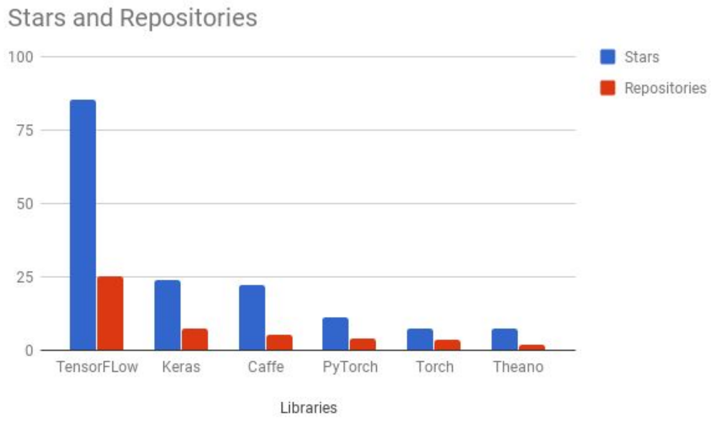
Girish Varma

TF Introduction

Deep Learning Framework by Google (open source release in 2016)

Inspired by Theano (built by University of Toronto)

https://arxiv.org/abs/1603.04467

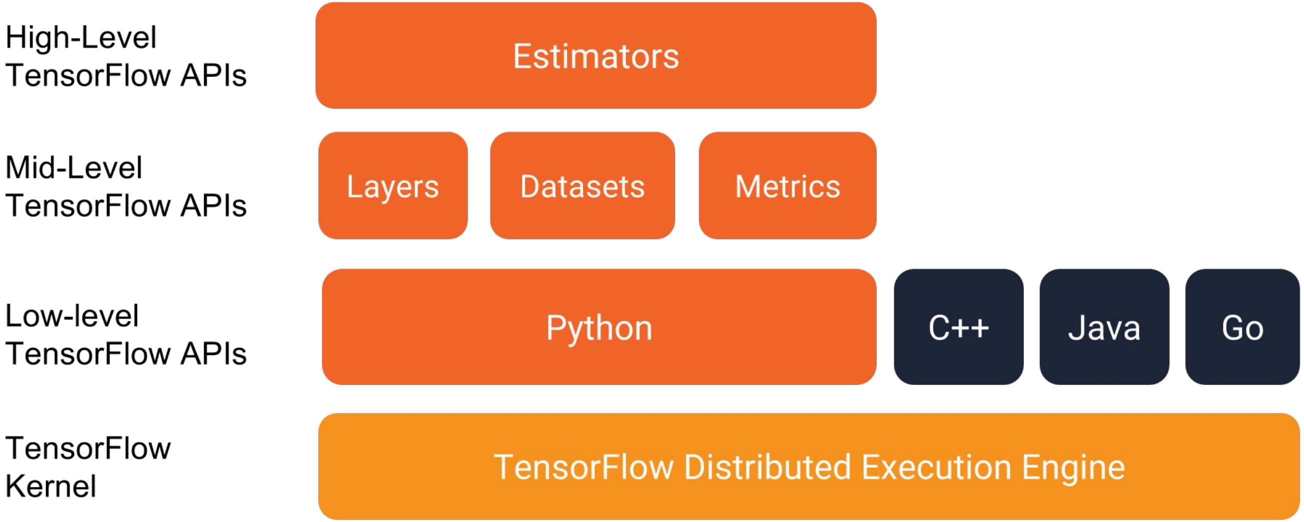


Google OpenAI AIRBUS

Google DeepMind intel

airbnb NVIDIA

Tensorflow



High-Level TensorFlow APIs — Estimators

Mid-Level TensorFlow APIs — Layers, Datasets, Metrics

Low-level TensorFlow APIs — Python, C++, Java, Go

TensorFlow Kernel — TensorFlow Distributed Execution Engine

Multilayered Framework. Multiple styles of coding.

Getting Started

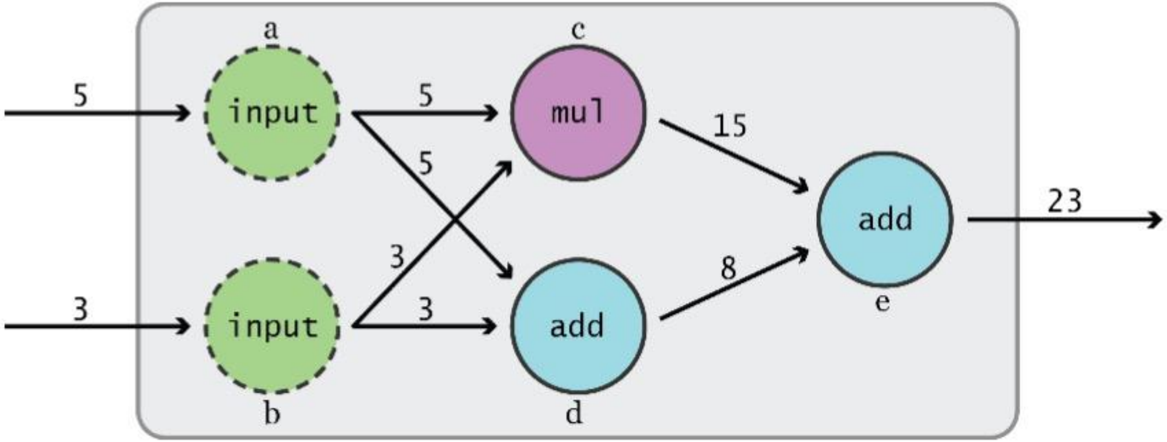
https://www.tensorflow.org/install/

```
import tensorflow as tf
```

Graphs and Sessions

Data Flow Graphs

TensorFlow separates definition of computations from their execution

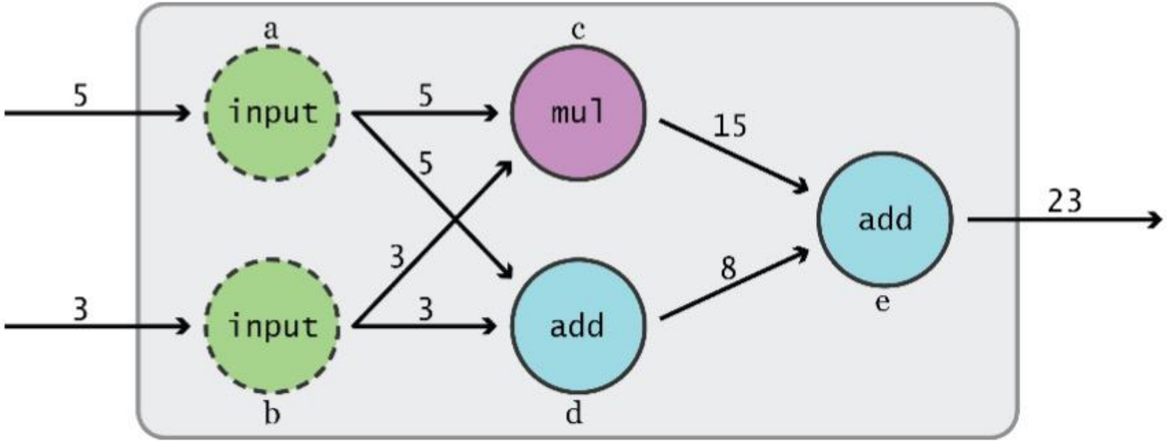


Graph from *TensorFlow for Machine Intelligence*

Data Flow Graphs

Phase 1: assemble a graph

Phase 2: use a session to execute operations in the graph.

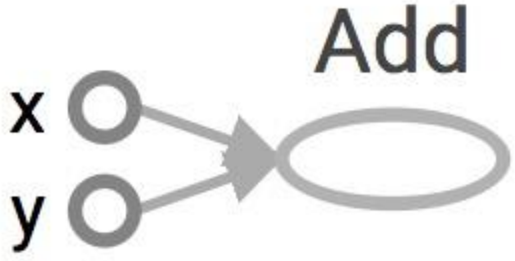


Graph from *TensorFlow for Machine Intelligence*

Data Flow Graphs

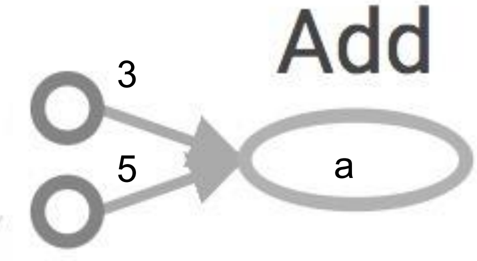
```
import tensorflow as tf
a = tf.add(3, 5)
```

Nodes: operators, variables, and constants
Edges: tensors



Data Flow Graphs

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



```
>> Tensor("Add:0", shape=(), dtype=int32)
```

(Not 8)

How to get the value of a?

Create a **session**, assign it to variable sess so we can call it later

Within the session, evaluate the graph to fetch the value of a

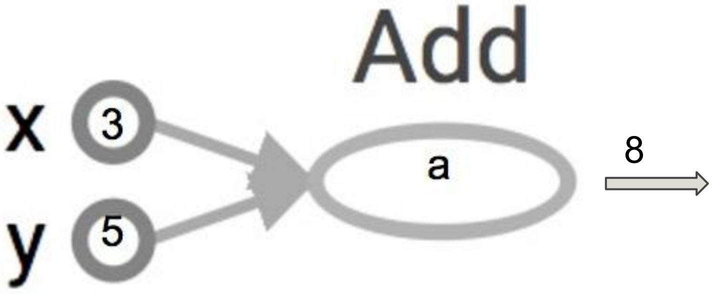
How to get the value of a?

Create a **session**, assign it to variable sess so we can call it later

Within the session, evaluate the graph to fetch the value of a

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

```
>> 8
```



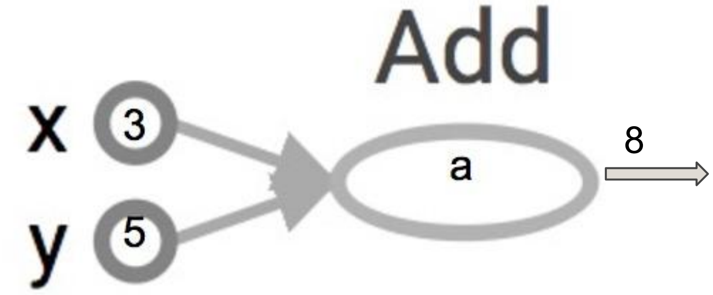
The session will look at the graph, trying to think: hmm, how can I get the value of a, then it computes all the nodes that leads to a.

How to get the value of a?

Create a **session**, assign it to variable sess so we can call it later

Within the session, evaluate the graph to fetch the value of a

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



tf.Session()

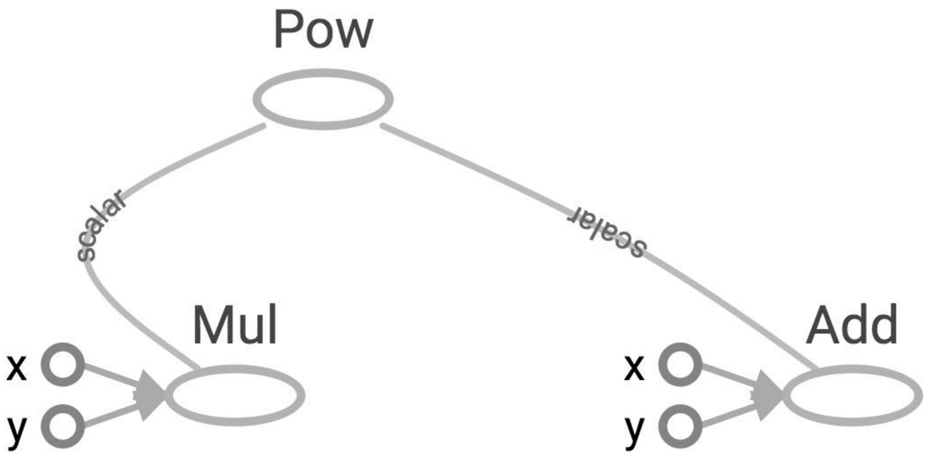
A Session object encapsulates the environment in which Operation objects are executed, and Tensor objects are evaluated.

Session will also allocate memory to store the current values of variables.

More graph

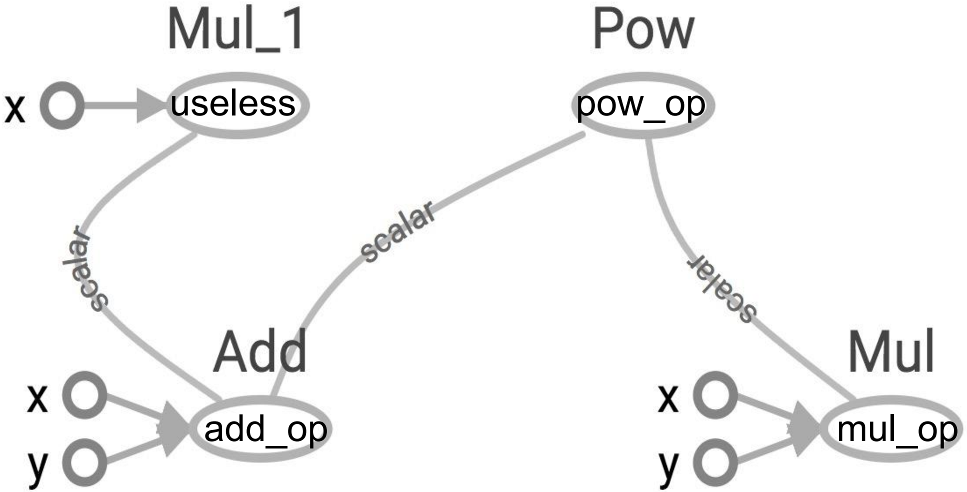
Visualized by TensorBoard

```
x = 2
y = 3
op1 = tf.add(x, y)
op2 = tf.multiply(x, y)
op3 = tf.pow(op2, op1)
with tf.Session() as sess:
      op3 = sess.run(op3)
```



Subgraphs

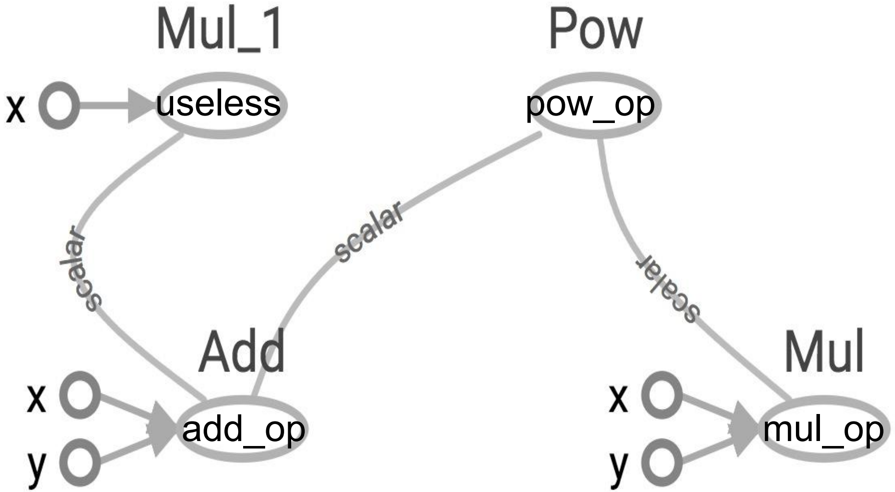
```
x = 2
y = 3
add_op = tf.add(x, y)
mul_op = tf.multiply(x, y)
useless = tf.multiply(x, add_op)
pow_op = tf.pow(add_op, mul_op)
with tf.Session() as sess:
      z = sess.run(pow_op)
```



pow_op

Subgraphs

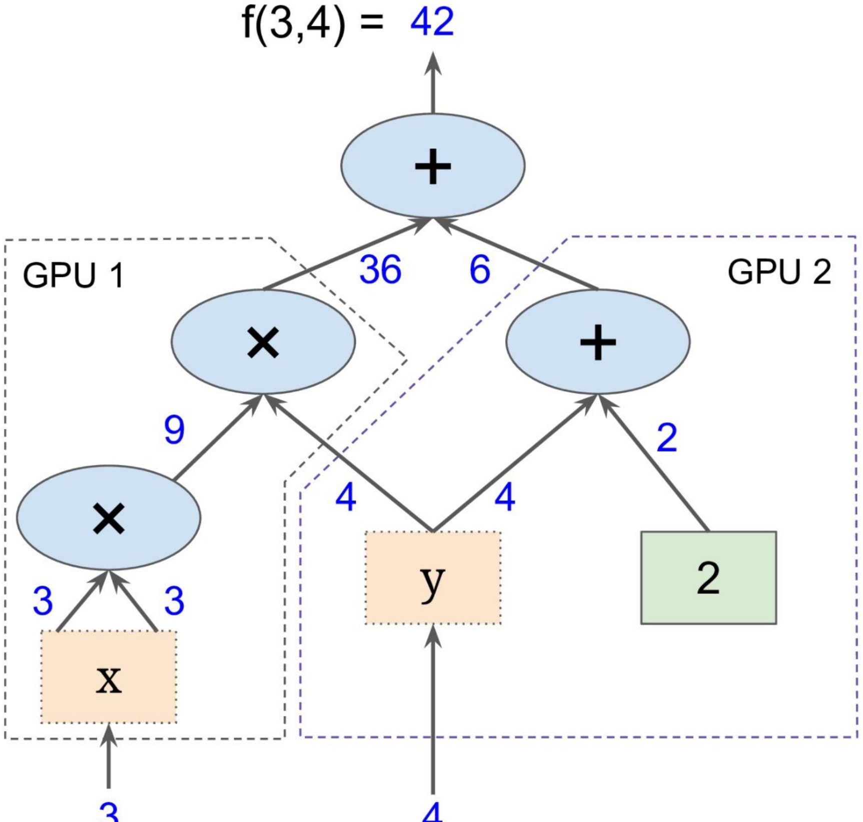
```
x = 2
y = 3
add_op = tf.add(x, y)
mul_op = tf.multiply(x, y)
useless = tf.multiply(x, add_op)
pow_op = tf.pow(add_op, mul_op)
with tf.Session() as sess:
      z, not_useless = sess.run([pow_op, useless])
```



Subgraphs

Possible to break graphs into several chunks and run them parallelly across multiple CPUs, GPUs, TPUs, or other devices

Example: AlexNet



Graph from *Hands-On Machine Learning with Scikit-Learn and TensorFlow*

Distributed Computation

To put part of a graph on a specific CPU or GPU:

```
# Creates a graph.
with tf.device('/gpu:2'):
  a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], name='a')
  b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], name='b')
  c = tf.multiply(a, b)

# Creates a session with log_device_placement set to True.
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))

# Runs the op.
print(sess.run(c))
```

Visualize it with TensorBoard

```
import tensorflow as tf

a = tf.constant(2)
b = tf.constant(3)
x = tf.add(a, b)

writer = tf.summary.FileWriter('./graphs', tf.get_default_graph())
with tf.Session() as sess:
     # writer = tf.summary.FileWriter('./graphs', sess.graph)
     print(sess.run(x))
writer.close() # close the writer when you’re done using it
```

Create the summary writer after graph definition and before running your session

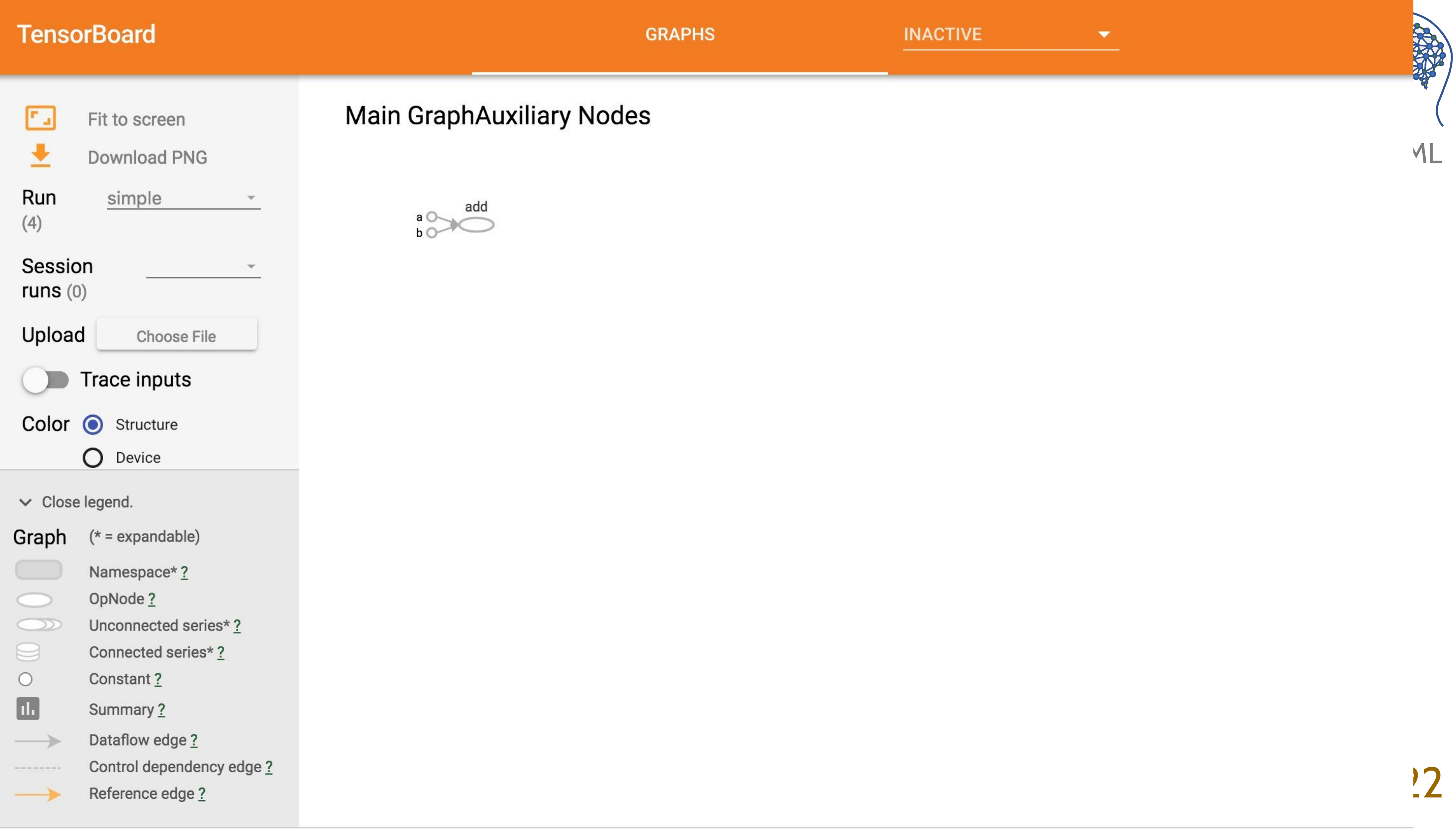
‘graphs’ or any location where you want to keep your event files

Run it

Go to terminal, run:

```
$ python3 [yourprogram].py
$ tensorboard --logdir="./graphs" --port 6006
```

Then open your browser and go to: `http://localhost:6006/`



Main GraphAuxiliary Nodes

Run simple (4)

Session runs (0)

Upload

Trace inputs

Color ● Structure

○ Device

Close legend.

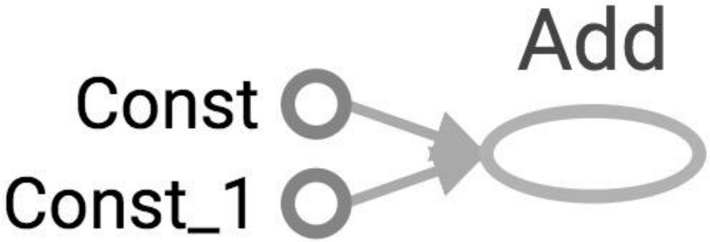
Graph (* = expandable)

▭	Namespace* ?
⬭	OpNode ?
⬭	Unconnected series* ?
☰	Connected series* ?
○	Constant ?
📊	Summary ?
→	Dataflow edge ?
- - -	Control dependency edge ?
→	Reference edge ?

ML

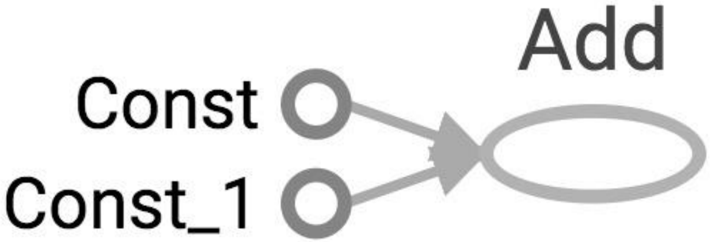
Visualize it with TensorBoard

```
import tensorflow as tf

a = tf.constant(2)
b = tf.constant(3)
x = tf.add(a, b)
writer = tf.summary.FileWriter('./graphs', tf.get_default_graph())
writer.close()
```



Visualize it with TensorBoard

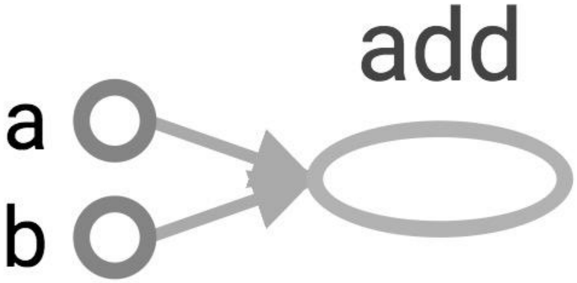
```
import tensorflow as tf

a = tf.constant(2)
b = tf.constant(3)
x = tf.add(a, b)
writer = tf.summary.FileWriter('./graphs', tf.get_default_graph())
writer.close()
```



Question:
How to change Const, Const_1 to the names we give the variables?

Explicitly name them

```
import tensorflow as tf

a = tf.constant(2, name='a')
b = tf.constant(3, name='b')
x = tf.add(a, b, name='add')

writer = tf.summary.FileWriter('./graphs', tf.get_default_graph())
with tf.Session() as sess:
     print(sess.run(x)) # >> 5
```



TensorBoard can do much more than just visualizing your graphs.

Learn to use TensorBoard well and often!

Constants, Sequences, Variables, Ops

Constants

```
import tensorflow as tf

a = tf.constant([2, 2], name='a')
b = tf.constant([[0, 1], [2, 3]], name='b')
```

```
tf.constant(
    value,
    dtype=None,
    shape=None,
    name='Const',
    verify_shape=False
)
```

Tensors filled with a specific value

```
tf.zeros(shape, dtype=tf.float32, name=None)
```

creates a tensor of shape and all elements will be zeros

Similar to numpy.zeros

```
tf.zeros([2, 3], tf.int32) ==> [[0, 0, 0], [0, 0, 0]]
```

Tensors filled with a specific value

```
tf.zeros_like(input_tensor, dtype=None, name=None, optimize=True)
```

creates a tensor of shape and type (unless type is specified) as the input_tensor but all elements are zeros.

```
# input_tensor is [[0, 1], [2, 3], [4, 5]]

tf.zeros_like(input_tensor) ==> [[0, 0], [0, 0], [0, 0]]
```

Tensors filled with a specific value

```
tf.fill(dims, value, name=None)
```

creates a tensor filled with a scalar value.

```
tf.fill([2, 3], 8) ==> [[8, 8, 8], [8, 8, 8]]
```

Constants as sequences

```
tf.lin_space(start, stop, num, name=None)
```

```
tf.lin_space(10.0, 13.0, 4) ==> [10. 11. 12. 13.]
```

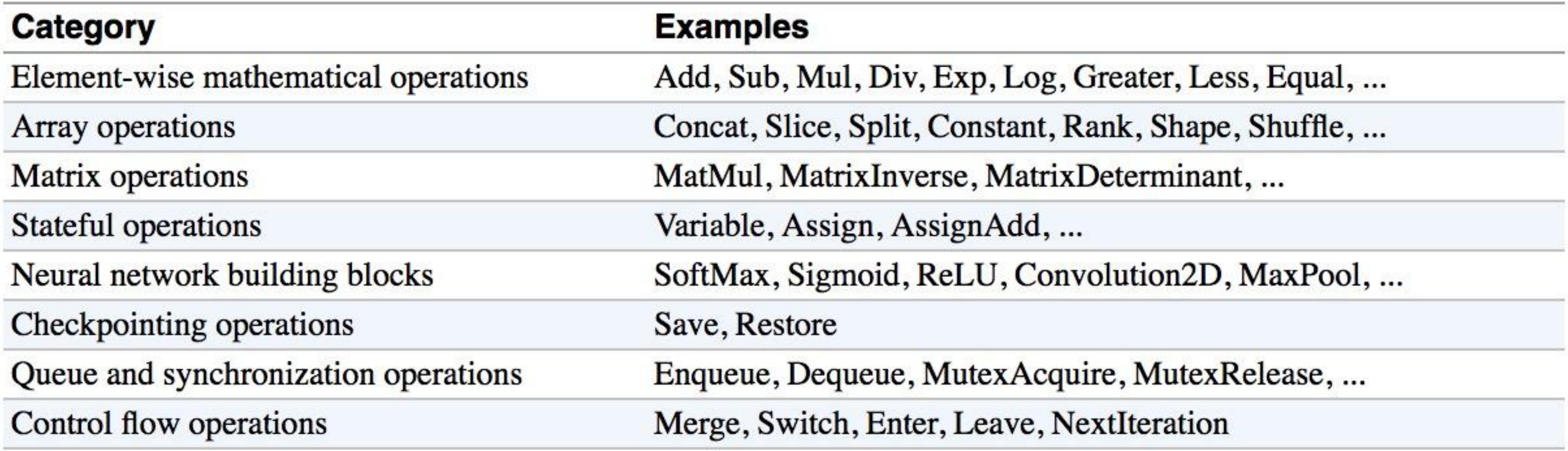
```
tf.range(start, limit=None, delta=1, dtype=None,
name='range')
```

```
tf.range(3, 18, 3) ==> [3 6 9 12 15]
tf.range(5) ==> [0 1 2 3 4]
```

Randomly Generated Constants

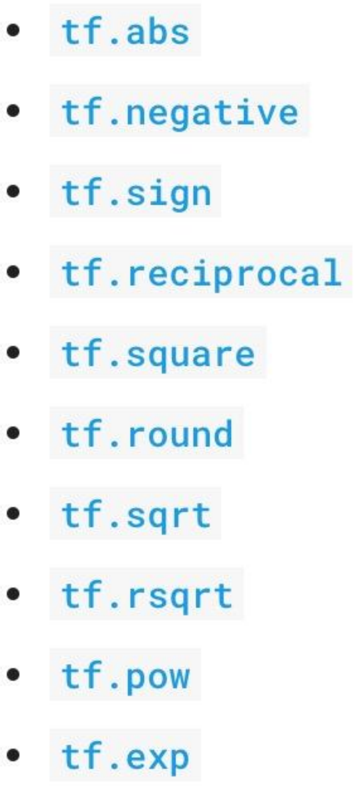
```
tf.random_normal

tf.truncated_normal

tf.random_uniform

tf.random_shuffle

tf.random_crop

tf.multinomial

tf.random_gamma
```

Operations



Category	Examples
Element-wise mathematical operations	Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal, ...
Array operations	Concat, Slice, Split, Constant, Rank, Shape, Shuffle, ...
Matrix operations	MatMul, MatrixInverse, MatrixDeterminant, ...
Stateful operations	Variable, Assign, AssignAdd, ...
Neural network building blocks	SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, ...
Checkpointing operations	Save, Restore
Queue and synchronization operations	Enqueue, Dequeue, MutexAcquire, MutexRelease, ...
Control flow operations	Merge, Switch, Enter, Leave, NextIteration

Arithmetic Ops



- `tf.abs`
- `tf.negative`
- `tf.sign`
- `tf.reciprocal`
- `tf.square`
- `tf.round`
- `tf.sqrt`
- `tf.rsqrt`
- `tf.pow`
- `tf.exp`

Wizard of Div

```
a = tf.constant([2, 2], name='a')

b = tf.constant([[0, 1], [2, 3]], name='b')

with tf.Session() as sess:

    print(sess.run(tf.div(b, a)))            ⇒ [[0 0] [1 1]]

    print(sess.run(tf.divide(b, a)))         ⇒ [[0. 0.5] [1. 1.5]]

    print(sess.run(tf.truediv(b, a)))        ⇒ [[0. 0.5] [1. 1.5]]

    print(sess.run(tf.floordiv(b, a)))       ⇒ [[0 0] [1 1]]

    print(sess.run(tf.realdiv(b, a)))        ⇒ # Error: only works for real values

    print(sess.run(tf.truncatediv(b, a)))    ⇒ [[0 0] [1 1]]

    print(sess.run(tf.floor_div(b, a)))      ⇒ [[0 0] [1 1]]
```

Variables

```
# create variables with tf.Variable

s = tf.Variable(2, name="scalar")

m = tf.Variable([[0, 1], [2, 3]], name="matrix")

W = tf.Variable(tf.zeros([784,10]))
```

Variables

```
# create variables with tf.Variable

s = tf.Variable(2, name="scalar")

m = tf.Variable([[0, 1], [2, 3]], name="matrix")

W = tf.Variable(tf.zeros([784,10]))


# create variables with tf.get_variable

s = tf.get_variable("scalar", initializer=tf.constant(2))

m = tf.get_variable("matrix", initializer=tf.constant([[0, 1], [2, 3]]))

W = tf.get_variable("big_matrix", shape=(784, 10), initializer=tf.zeros_initializer())
```

tf.Variable class

```
# create variables with tf.get_variable

s = tf.get_variable("scalar", initializer=tf.constant(2))

m = tf.get_variable("matrix", initializer=tf.constant([[0, 1], [2, 3]]))

W = tf.get_variable("big_matrix", shape=(784, 10), initializer=tf.zeros_initializer())


with tf.Session() as sess:
     print(sess.run(W))   >> FailedPreconditionError: Attempting to use uninitialized value Variable
```

You have to initialize your variables

The easiest way is initializing all variables at once:

```
with tf.Session() as sess:

      sess.run(tf.global_variables_initializer())
```

You have to initialize your variables

The easiest way is initializing all variables at once:

```
with tf.Session() as sess:

      sess.run(tf.global_variables_initializer())
```

Initialize only a subset of variables:

```
with tf.Session() as sess:

      sess.run(tf.variables_initializer([a, b]))
```

You have to initialize your variables

The easiest way is initializing all variables at once:

```
tf.global_variables_initializer()
```

Initialize only a subset of variables:

```
tf.variables_initializer([a, b])
```

Initialize a single variable

```
W = tf.Variable(tf.zeros(784,10]))

with tf.Session() as sess:

     sess.run(W.initializer)
```

Placeholder

A quick reminder

A TF program often has 2 phases:

1. Assemble a graph
2. Use a session to execute operations in the graph.

Placeholders

A TF program often has 2 phases:

1. Assemble a graph
2. Use a session to execute operations in the graph.

⇒ Assemble the graph first without knowing the values needed for computation

Analogy:

Define the function f(x, y) = 2 * x + y without knowing value of x or y.

x, y are placeholders for the actual values.

Why placeholders?

We, or our clients, can later supply their own data when they need to execute the computation.

Placeholders

```
tf.placeholder(dtype, shape=None, name=None)
```

```
# create a placeholder for a vector of 3 elements, type tf.float32

a = tf.placeholder(tf.float32, shape=[3])

b = tf.constant([5, 5, 5], tf.float32)

# use the placeholder as you would a constant or a variable

c = a + b  # short for tf.add(a, b)

with tf.Session() as sess:

      print(sess.run(c))                       # >> ???
```

Placeholders

tf.placeholder(dtype, shape=None, name=None)

```
# create a placeholder for a vector of 3 elements, type tf.float32

a = tf.placeholder(tf.float32, shape=[3])

b = tf.constant([5, 5, 5], tf.float32)

# use the placeholder as you would a constant or a variable

c = a + b  # short for tf.add(a, b)

with tf.Session() as sess:

      print(sess.run(c))                    # >> InvalidArgumentError: a doesn’t an actual value
```

Supplement the values to placeholders using a dictionary

Placeholders

```
tf.placeholder(dtype, shape=None, name=None)
```

```
# create a placeholder for a vector of 3 elements, type tf.float32

a = tf.placeholder(tf.float32, shape=[3])

b = tf.constant([5, 5, 5], tf.float32)

# use the placeholder as you would a constant or a variable

c = a + b  # short for tf.add(a, b)

with tf.Session() as sess:

      print(sess.run(c, feed_dict={a: [1, 2, 3]}))    # the tensor a is the key, not the string ‘a’

# >> [6, 7, 8]
```

Phase 1: Assemble our graph

Step 1: Read in data

TFRecord

What’s TFRecord

1. The recommended format for TensorFlow
2. Binary file format

   a serialized tf.train.Example protobuf object

Why binary

- make better use of disk cache
- faster to move around
- can handle data of different types

  e.g. you can put both images and labels in one place

Convert to TFRecord format

```
# Step 1: create a writer to write tfrecord to that file
writer = tf.python_io.TFRecordWriter(out_file)

# Step 2: get serialized shape and values of the image
shape, binary_image = get_image_binary(image_file)

# Step 3: create a tf.train.Features object
features = tf.train.Features(feature={'label': _int64_feature(label),
                                    'shape': _bytes_feature(shape),
                                    'image': _bytes_feature(binary_image)})

# Step 4: create a sample containing of features defined above
sample = tf.train.Example(features=features)

# Step 5: write the sample to the tfrecord file
writer.write(sample.SerializeToString())
writer.close()
```

Read TFRecord

Using TFRecordDataset

Read TFRecord

```
dataset = tf.data.TFRecordDataset(tfrecord_files)
dataset = dataset.map(_parse_function)

def _parse_function(tfrecord_serialized):
    features={'label': tf.FixedLenFeature([], tf.int64),
              'shape': tf.FixedLenFeature([], tf.string),
              'image': tf.FixedLenFeature([], tf.string)}

    parsed_features = tf.parse_single_example(tfrecord_serialized, features)

    return parsed_features['label'], parsed_features['shape'], parsed_features['image']
```

Step 2: Create placeholders for inputs and labels

```
tf.placeholder(dtype, shape=None, name=None)
```

Step 3: Create weight and bias

```
tf.get_variable(
    name,
    shape=None,
    dtype=None,
    initializer=None,
   …
)
```

Step 4: Inference

```
Y_predicted = w * X + b
```

Step 5: Specify loss function

```
loss = tf.square(Y - Y_predicted, name='loss')
```

Step 6: Create optimizer

```
opt =
tf.train.GradientDescentOptimizer(learning_rate=0
.001)

optimizer = opt.minimize(loss)
```

Phase 2: Train our model

Step 1: Initialize variables

Step 2: Run optimizer

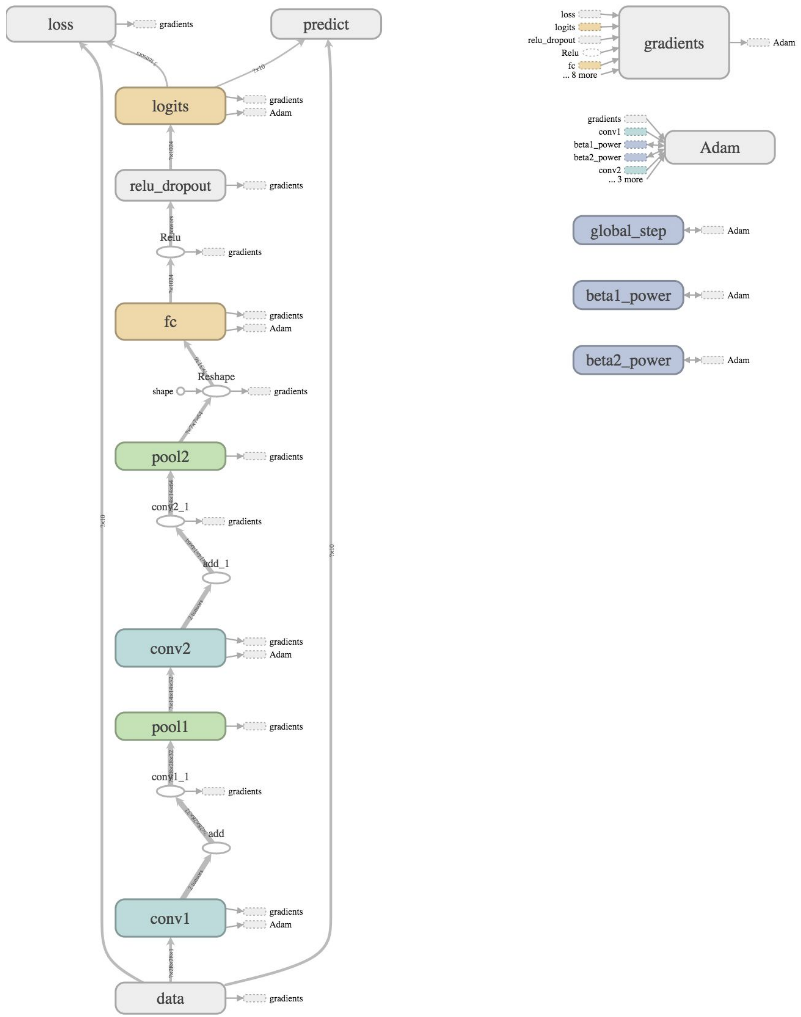
(use a feed_dict to feed data into X and Y placeholders)

Write log files using a FileWriter

```
writer = tf.summary.FileWriter('./graphs/linear_reg', sess.graph)
```

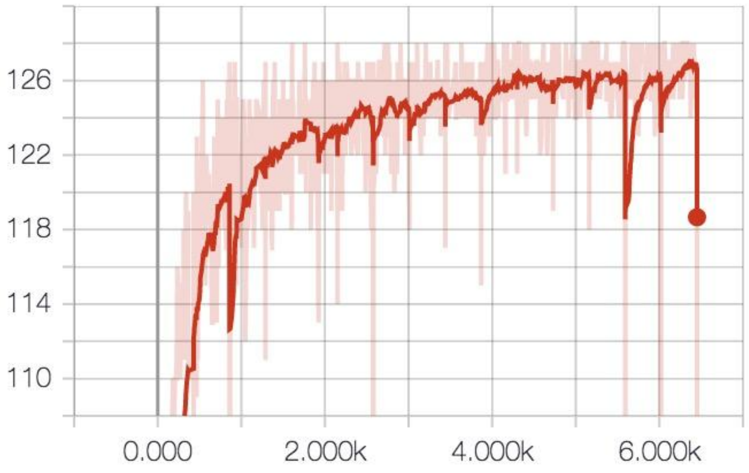
See it on TensorBoard

```
Step 1: $ python3 03_linreg_starter.py

Step 2: $ tensorboard --logdir='./graphs'
```

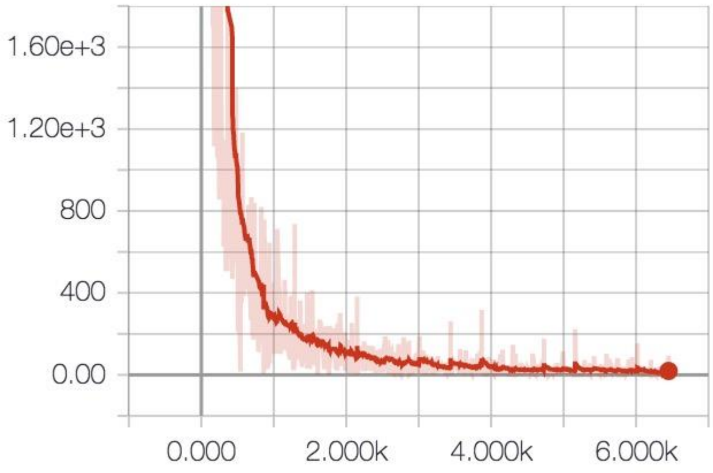

loss
gradients
predict
logits
gradients
Adam
relu_dropout
gradients
Relu
gradients
fc
gradients
Adam
Reshape
shape
gradients
pool2
gradients
conv2_1
gradients
add_1
conv2
gradients
Adam
pool1
gradients
conv1_1
gradients
add
conv1
gradients
Adam
data
gradients
loss
logits
relu_dropout
Relu
fc
... 8 more
gradients
Adam
gradients
conv1
beta1_power
beta2_power
conv2
... 3 more
Adam
global_step
Adam
beta1_power
Adam
beta2_power
Adam

Training progress

summaries/accuracy



summaries/loss



tf.layers

tf.layers.conv2d

```
conv1 = tf.layers.conv2d(inputs=self.img,
                         filters=32,
                         kernel_size=[5, 5],
                         padding='SAME',
                         activation=tf.nn.relu,
                         name='conv1')
```

can choose
non-linearity to use

tf.layers.max_pooling2d

```
pool1 = tf.layers.max_pooling2d(inputs=conv1,
                     pool_size=[2, 2],
                     strides=2,
                     name='pool1')
```

tf.layers.dense

```
fc = tf.layers.dense(pool2, 1024, activation=tf.nn.relu, name='fc')
```

tf.layers.dense

```
dropout = tf.layers.dropout(fc,
                    self.keep_prob,
                    training=self.training,
                    name='dropout')
```

Misc

Tensorflow debugger

Feature Visualization

Deployment

References

TF Documentation

https://www.tensorflow.org/get_started/

Stanford Course (CS 20: Tensorflow for Deep Learning Research)

http://web.stanford.edu/class/cs20si/

Code Examples

https://github.com/aymericdamien/TensorFlow-Examples

TF Youtube Channel

https://www.youtube.com/channel/UC0rqucBdTuFTjJiefW5t-IQ/videos

Feature Visualization

https://github.com/normanheckscher/mnist-tensorboard-embeddings

https://www.tensorflow.org/versions/r1.1/get_started/embedding_viz

http://colah.github.io/posts/2014-10-Visualizing-MNIST/

Thanks!