TensorFlow

SPEAKER: PYRY TAKALA

True AI

DIALOGUES +

DEEP LEARNING

WAITING TIMES

"…it may take up to 20 working days to provide a full reply where investigation is required. During times of high demand, we may sometimes miss our targets…"

- London Midland trains

LONDON-BASED STARTUP

DEEP LEARNING RESEARCHERS

EXPERT TEAM

UCL

MIT Technology Review

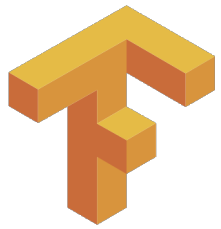
amazon

Université de Montréal

Goldman Sachs

iit

McKinsey&Company

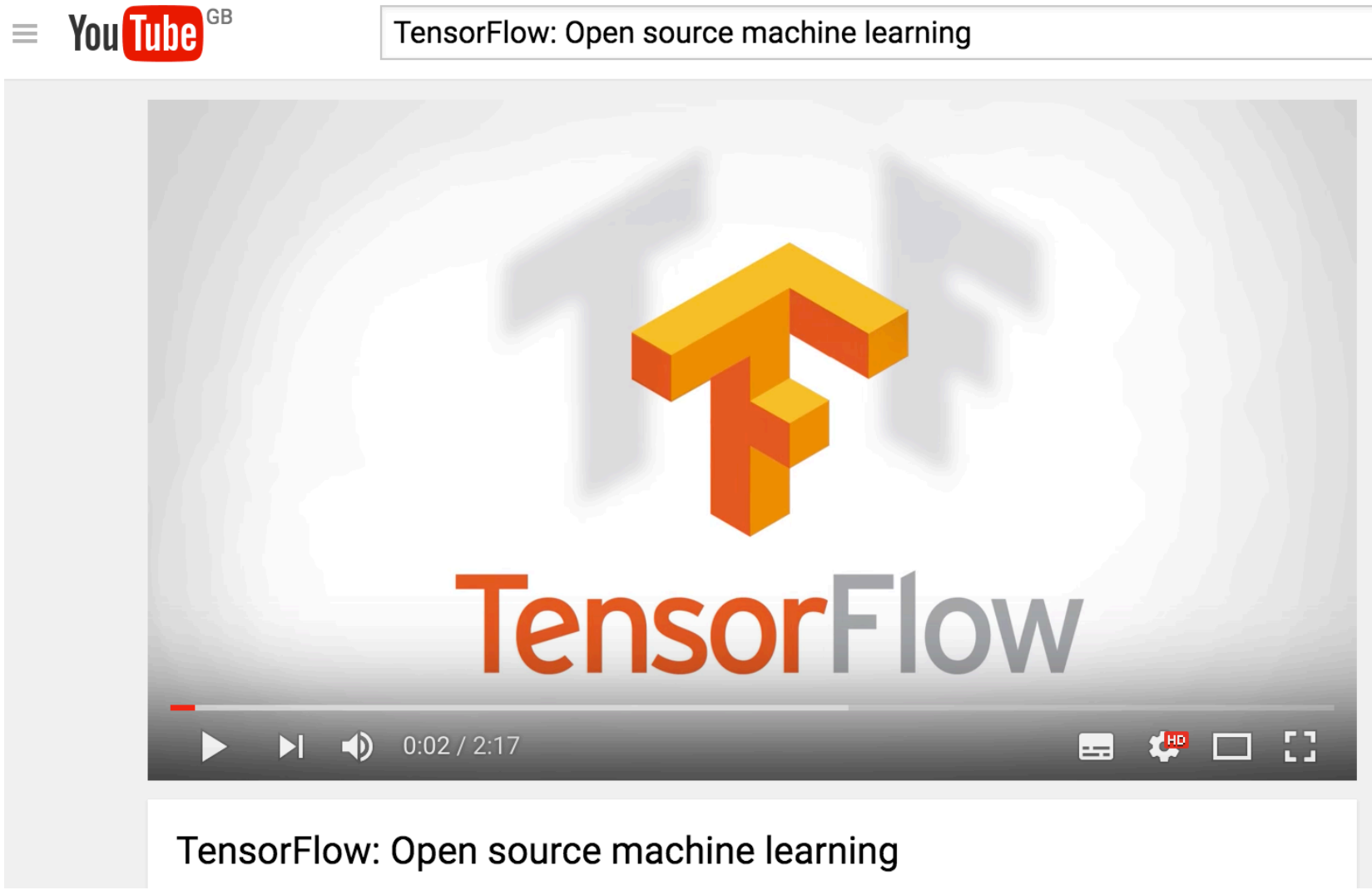


Background

Usage

Under-the-hood

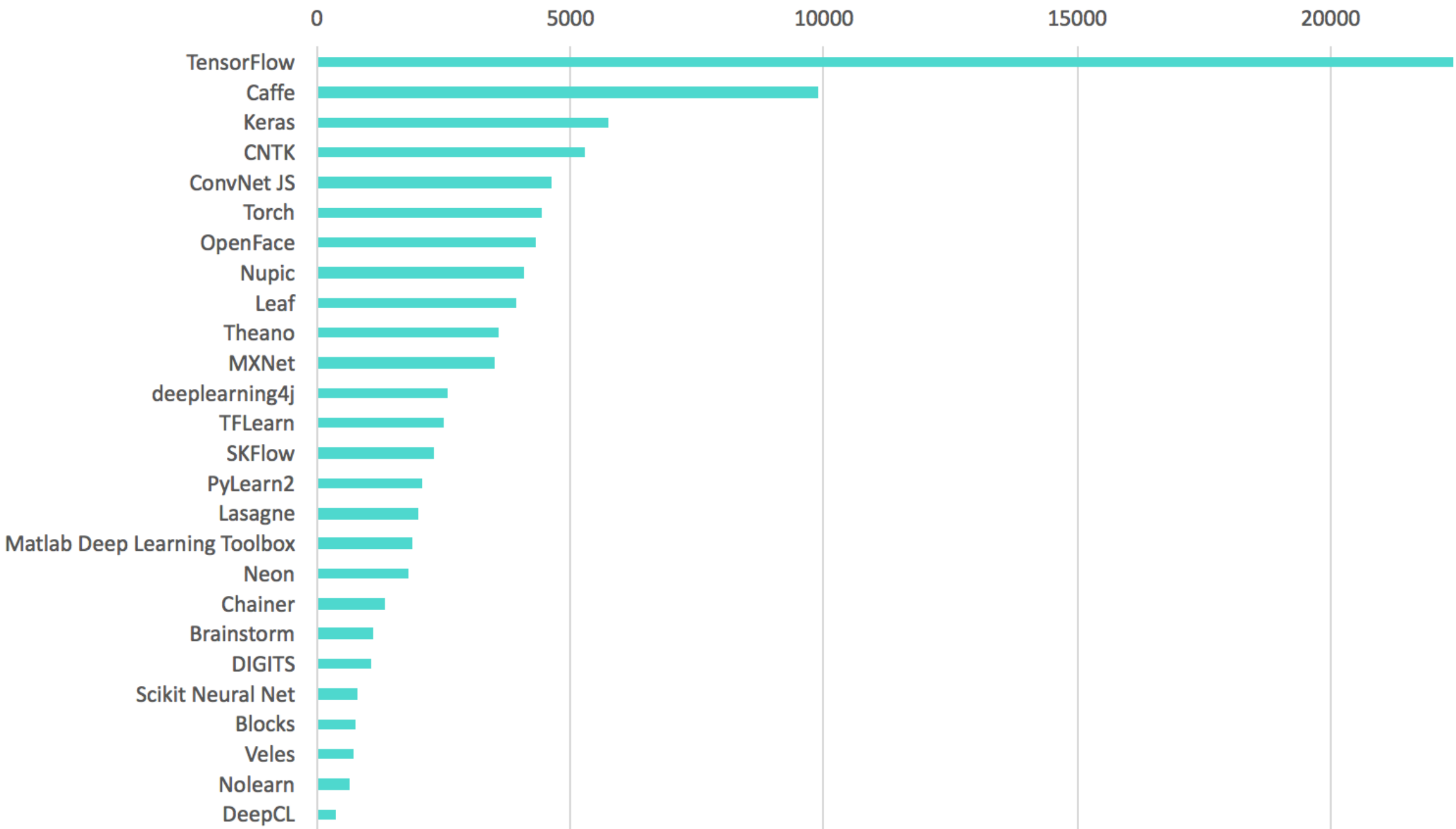
Resources



https://www.youtube.com/watch?v=oZikw5k_2FM

- ✔ Research and production ML library, designed especially for deep learning
- ✔ Started by Google Brain after DistBelief; now open source
- ✔ Strong focus on parallelizing computations
- ✔ Python & C++

GITHUB STARS PER FRAMEWORK



Source: 2-May-2016, https://github.com/aymericdamien/TopDeepLearning

Pros and Cons

- (+) Python + numpy
- (+) Computational graph abstraction, like Theano
- (+) Much faster compile times than Theano
- (+) TensorBoard for visualization
- (+) Data AND model parallelism
- (-) Slower than other frameworks
- (-) Much “fatter” than Torch; more magic
- (-) Not many pretrained models
- (-) Computational graph is pure Python, so slow

Pros and Cons

- (+) Python + numpy
- (+) Computational graph is nice abstraction
- (+) RNNs fit nicely in computational graph
- (-) Raw Theano is somewhat low-level
- (+) High level wrappers (Keras, Lasagne) ease the pain
- (-) Error messages can be unhelpful
- (-) Large models can have long compile times
- (-) Much “fatter” than Torch; more magic
- (-) Patchy support for pretrained mo

Pros and Cons:

- (-) Lua
- You usually write your own training code (Less plug and play)
- (+) Lots of modular pieces that are easy to combine
- (+) Easy to write your own layer types and run on GPU
- (+) Lua. ;) (Most of the library code is in Lua, easy to read)
- (+) Lots of pretrained models!
- (-) Not good for recurrent neural networks

DEEPLEARNING4J

	Caffe	CNTK	TensorFlow	theano	torch
Modeling capability	***	**	****’	****’	*****
Interfaces	***	**’	****’	****	****
Model Deployment	*****	****’	****’	***	***
Performance			*****	***	*****
Architecture	***		*****	***	*****
Ecosystem	C++	C++	Py & C++	Py	Lua
Cross-platform					

https://github.com/zer0n/deepframeworks

WHAT IS A “TENSOR”:

“Tensors are geometric objects that describe linear relations between geometric vectors, scalars, and other tensors.“

W

A scalar is a tensor

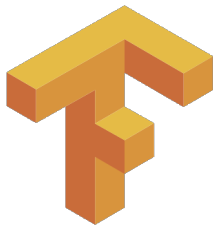
24

A vector is a tensor

[24 98 16]

A matrix is a tensor

$$\begin{bmatrix} 24 & 98 & 16 \\ 5 & 21 & 2 \end{bmatrix}$$



Background

Usage

Under-the-hood

Resources

```
In [1]: import tensorflow as tf

In [2]: with tf.Session() as sess:
   ...:     print tf.constant("Hello world!").eval()
   ...:
Hello world!
```

NUMPY VS TENSORFLOW

```
In [3]: import numpy as np

In [4]: a = np.zeros((2,2))

In [5]: b = np.ones((2,2))

In [6]: print np.sum(b, axis=1)
[ 2.  2.]
```

```
In [1]: import tensorflow as tf

In [2]: with tf.Session() as sess:
   ...:             a = tf.zeros((2,2))
   ...:             b = tf.ones((2,2))
   ...:             print tf.reduce_sum(b, reduction_indices=1).eval()
   ...:
[ 2.  2.]
```

NUMPY VS TENSORFLOW

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,:]`

Source: https://cs224d.stanford.edu/lectures/CS224d-Lecture7.pdf

DATA TYPES

Data type	Python type	Description
`DT_FLOAT`	`tf.float32`	32 bits floating point
`DT_DOUBLE`	`tf.float64`	64 bits floating point
`DT_INT8`	`tf.int8`	8 bits signed integer
`DT_INT16`	`tf.int16`	16 bits signed integer
`DT_INT32`	`tf.int32`	32 bits signed integer
`DT_INT64`	`tf.int64`	64 bits signed integer
`DT_UINT8`	`tf.uint8`	8 bits unsigned integer
`DT_STRING`	`tf.string`	Variable length byte arrays. Each element of a Tensor is a byte array
`DT_BOOL`	`tf.bool`	Boolean
`DT_COMPLEX64`	`tf.complex64`	Complex number made of two 32 bits floating points: real and imaginary parts
`DT_QINT8`	`tf.qint8`	8 bits signed integer used in quantized Ops
`DT_QINT32`	`tf.qint32`	32 bits signed integer used in quantized Ops
`DT_QUINT8`	`tf.quint8`	8 bits unsigned integer used in quantized Ops

Source: www.tensorflow.org

CONSTANTS

Constants, Sequences, and Random Values

Constant Value Tensors

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

tf.zeros_like(tensor, dtype=None,
name=None)

tf.ones(shape, dtype=tf.float32,
name=None)

tf.ones_like(tensor, dtype=None,
name=None)

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

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

Sequences

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

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

Random Tensors

Examples:

```
tf.random_normal(shape, mean=0.0,
stddev=1.0, dtype=tf.float32,
seed=None, name=None)

tf.truncated_normal(shape,
mean=0.0, stddev=1.0,
dtype=tf.float32, seed=None,
name=None)

tf.random_uniform(shape, minval=0,
maxval=None, dtype=tf.float32,
seed=None, name=None)

tf.random_shuffle(value,
seed=None, name=None)

tf.random_crop(value, size,
seed=None, name=None)

tf.multinomial(logits,
num_samples, seed=None, name=None)

tf.set_random_seed(seed)
```

Etc.

Source: www.tensorflow.org

TENSOR OPERATIONS

Reduction

```
tf.reduce_sum(input_tensor,
reduction_indices=None,
keep_dims=False, name=None)

tf.reduce_prod(input_tensor,
reduction_indices=None,
keep_dims=False, name=None)

tf.reduce_min(input_tensor,
reduction_indices=None,
keep_dims=False, name=None)

tf.reduce_max(input_tensor,
reduction_indices=None,
keep_dims=False, name=None)

tf.reduce_mean(input_tensor,
reduction_indices=None,
keep_dims=False, name=None)

tf.reduce_all(input_tensor,
reduction_indices=None,
keep_dims=False, name=None)

tf.reduce_any(input_tensor,
reduction_indices=None,
keep_dims=False, name=None)
```

Basic Math Functions

```
tf.add_n(inputs, name=None)

tf.abs(x, name=None)

tf.neg(x, name=None)

tf.sign(x, name=None)

tf.inv(x, name=None)

tf.square(x, name=None)

tf.round(x, name=None)

tf.sqrt(x, name=None)

tf.rsqrt(x, name=None)

tf.pow(x, y, name=None)
```

Matrix Math Functions

```
tf.trace(x, name=None)

tf.transpose(a, perm=None,
name='transpose')

tf.matmul(a, b, transpose_a=False,
transpose_b=False,
a_is_sparse=False,
b_is_sparse=False, name=None)
```

Etc.

Source: www.tensorflow.org

TENSORFLOW SESSION

- Session encapsulates environment where TF-objects are evaluated

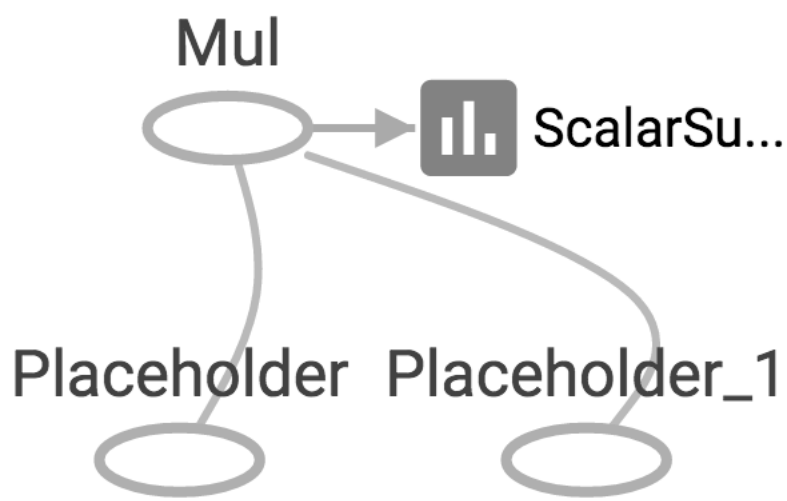
```
In [8]: import tensorflow as tf

In [9]: with tf.Session() as sess:
   ...:         a = tf.zeros((2,2))
   ...:         b = tf.ones((2,2))
   ...:         print sess.run(tf.reduce_sum(b, reduction_indices=1))
   ...:
[ 2.  2.]
```

- sess.run() method runs complete subset of graph corresponding to ops passed as arguments

COMPUTATION GRAPH

- Similarly to Theano, we compile a computation graph

```
2  import tensorflow as tf
3
4  with tf.Session() as sess:
5
6
7      a = tf.placeholder(tf.int32)
8      b = tf.placeholder(tf.int32)
9      c = tf.mul(a, b)
10
11     c_summary = tf.scalar_summary(["c_summary"], c)
12
13     writer = tf.train.SummaryWriter("/tmp/example_logs", sess.graph)
14
15     for i in range(10):
16         feed_dict={a:[2*i], b:[3*i]}
17         summary, c_val = sess.run([c_summary, c], feed_dict=feed_dict)
18         print c_val[0]
19         writer.add_summary(summary, i)
```



TENSORFLOW SESSION - SHORTCUTS

.eval() vs run

```
In [13]: with tf.Session() as sess:
    ....:         print tf.ones((1,)).eval()
    ....:         print sess.run(tf.ones((1,)))
    ....:         print [x for x in sess.run([tf.ones((1,)), tf.ones((1,))])]
    ....:
[ 1.]
[ 1.]
[array([ 1.], dtype=float32), array([ 1.], dtype=float32)]
```

!

.eval() can't be used to evaluate multiple values at once

tf.InteractiveSession()

```
In [6]: tf.ones((1,1)).eval()
---------------------------------------------------------------------------
ValueError: Cannot evaluate tensor using eval(): No default session is registered. Use `with sess.as_default()` or pass an explicit session to eval(session=sess)
```

```
In [7]: sess = tf.InteractiveSession()

In [8]: tf.ones((1,1)).eval()
Out[8]: array([[ 1.]], dtype=float32)
```

VARIABLES

Constants

```
In [1]: import tensorflow as tf

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

In [3]: with tf.Session() as sess:
   ...:         print W1.eval()
   ...:
[[ 1.  1.]
 [ 1.  1.]]
```

- ”In-memory buffers" containing tensors
- Need to initialize
- Good for weights: they ”survive" the execution of the graph

Variables

```
In [4]: import tensorflow as tf

In [5]: W2 = tf.Variable(tf.ones((2,2)), name="w")

In [6]: with tf.Session() as sess:
   ...:         print W2.eval()
   ...:
---------------------------------------------------------------------------
FailedPreconditionError                          Traceback
```

```
In [7]: import tensorflow as tf

In [8]: W2 = tf.Variable(tf.ones((2,2)), name="w")

In [9]: with tf.Session() as sess:
   ...:         sess.run(tf.initialize_all_variables())
   ...:         print W2.eval()
   ...:
[[ 1.  1.]
 [ 1.  1.]]
```

UPDATING VARIABLE STATE

```
In [1]: import tensorflow as tf

In [2]: counter = tf.Variable(0, name = 'counter')

In [3]: next_value = tf.add(counter, 1)

In [4]: update = tf.assign(counter, next_value)

In [5]:

In [5]: with tf.Session() as sess:
   ...:         sess.run(tf.initialize_all_variables())
   ...:         print counter.eval()
   ...:         update.eval()
   ...:         print counter.eval()
   ...:
0
1
```

EXTERNAL DATA TO TENSORFLOW

From Numpy

```
In [20]: import numpy as np

In [21]: a = np.ones((2,2))

In [22]: tf_a = tf.convert_to_tensor(a)

In [23]: with tf.Session() as sess:
   ....:         print tf_a.eval()
   ....:
[[ 1.  1.]
 [ 1.  1.]]
```

From a feed-dictionary

```
In [36]: a = tf.placeholder(tf.int32)

In [37]: b = tf.placeholder(tf.int32)

In [38]: c = tf.mul(a, b)

In [39]: with tf.Session() as sess:
   ....:         print sess.run([c], feed_dict={a:[2, 7], b:[3, 7]})
   ....:
[array([ 6, 49], dtype=int32)]
```

- Better option for feeding large datasets
- Placeholders as keys, data as values

VARIABLE SCOPING AND REUSE

```
In [1]: import tensorflow as tf

In [2]: sess = tf.Session()

In [3]: w = tf.Variable(0.0, name="w")

In [4]:

In [4]: with tf.variable_scope('a') as scope:
   ...:         w1 = tf.get_variable("w", [1])
   ...:

In [5]: with tf.variable_scope('a') as scope:
   ...:         tf.get_variable_scope().reuse_variables()
   ...:         w2 = tf.get_variable("w", [1])
   ...:

In [6]: w1==w2
Out[6]: True

In [7]: w1.name
Out[7]: u'a/w:0'
```

- Scoping helps with naming

```
In [6]: with tf.variable_scope('a') as scope:
   ...:         w3 = tf.get_variable("w", [1])

ValueError: Variable a/w already exists, disal
lowed. Did you mean to set reuse=True in VarSc
ope? Originally defined at:
```

- Reuse needs to be explicit

```
In [7]: with tf.variable_scope('b') as scope:
   ...:         tf.get_variable_scope().reuse_variables()
   ...:         w4 = tf.get_variable("w1", [1])

ValueError: Variable b/w1 does not exist, disa
llowed. Did you mean to set reuse=None in VarS
cope?
```

- Scoping adds prefix to variables names

FORWARD COMPUTATION

LINEAR REGRESSION EXAMPLE

```
'''
A linear regression learning algorithm example using TensorFlow library.
Modified from Aymeric Damien's github.com/aymericdamien/TensorFlow-Examples/
'''

import tensorflow as tf
import numpy as np

# Model
X = tf.placeholder("float")
Y = tf.placeholder("float")

W = tf.Variable(np.random.randn(), name="weight")
b = tf.Variable(np.random.randn(), name="bias")

pred = tf.add(tf.mul(X, W), b)
```

OPTIMIZERS

LINEAR REGRESSION EXAMPLE

```
# Training Data
train_X = np.asarray([3.3,4.4,5.5,6.71,6.93,4.168,9.779,6.182,7.59,2.167,
                      7.042,10.791,5.313,7.997,5.654,9.27,3.1])
train_Y = np.asarray([1.7,2.76,2.09,3.19,1.694,1.573,3.366,2.596,2.53,1.221,
                      2.827,3.465,1.65,2.904,2.42,2.94,1.3])
n_samples = train_X.shape[0]

# Parameters
learning_rate = 0.01
training_epochs = 1000
display_step = 50

# Optimizer
cost = tf.reduce_sum(tf.pow(pred-Y, 2))/(2*n_samples)
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)


# Launch the graph
with tf.Session() as sess:

    sess.run(tf.initialize_all_variables())

    # Fit all training data
    for epoch in range(training_epochs):
        for (x, y) in zip(train_X, train_Y):
            sess.run(optimizer, feed_dict={X: x, Y: y})

        #Display logs per epoch step
        if (epoch+1) % display_step == 0:
            c = sess.run(cost, feed_dict={X: train_X, Y:train_Y})
            print "Epoch:", '%04d' % (epoch+1), "cost=", "{:.9f}".format(c), \
                "W=", sess.run(W), "b=", sess.run(b)

    print "Optimization Finished!"
    training_cost = sess.run(cost, feed_dict={X: train_X, Y: train_Y})
    print "Training cost=", training_cost, "W=", sess.run(W), "b=", sess.run(b), '\n'
```

```
Epoch: 0050 cost= 0.100161634 W= 0.334835 b= 0.188247
Epoch: 0100 cost= 0.097480297 W= 0.329765 b= 0.224718
Epoch: 0150 cost= 0.095108874 W= 0.324997 b= 0.25902
Epoch: 0200 cost= 0.093011498 W= 0.320512 b= 0.291282
Epoch: 0250 cost= 0.091156594 W= 0.316294 b= 0.321625
Epoch: 0300 cost= 0.089516155 W= 0.312327 b= 0.350164
Epoch: 0350 cost= 0.088065267 W= 0.308596 b= 0.377005
Epoch: 0400 cost= 0.086782187 W= 0.305087 b= 0.402251
Epoch: 0450 cost= 0.085647523 W= 0.301787 b= 0.425993
Epoch: 0500 cost= 0.084644087 W= 0.298682 b= 0.448324
Epoch: 0550 cost= 0.083756670 W= 0.295763 b= 0.469328
Epoch: 0600 cost= 0.082971931 W= 0.293017 b= 0.489082
Epoch: 0650 cost= 0.082278028 W= 0.290434 b= 0.507661
Epoch: 0700 cost= 0.081664391 W= 0.288005 b= 0.525135
Epoch: 0750 cost= 0.081121773 W= 0.285721 b= 0.54157
Epoch: 0800 cost= 0.080641970 W= 0.283572 b= 0.557028
Epoch: 0850 cost= 0.080217734 W= 0.281551 b= 0.571567
Epoch: 0900 cost= 0.079842605 W= 0.27965 b= 0.585241
Epoch: 0950 cost= 0.079510957 W= 0.277863 b= 0.598101
Epoch: 1000 cost= 0.079217732 W= 0.276181 b= 0.610197
Optimization Finished!
Training cost= 0.0792177 W= 0.276181 b= 0.610197
```

OPTIMIZERS

- Intialize optimizer
- Setup derivatives graph

```
30 # Optimizer
31 cost = tf.reduce_sum(tf.pow(pred-Y, 2))/(2*n_samples)
32 optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
33
```

```
42         for (x, y) in zip(train_X, train_Y):
43             sess.run(optimizer, feed_dict={X: x, Y: y})
```

- We run the optimizer for all training data
- Behind the scenes: automatic differentiation (similar to Theano)

EXAMPLES: LOGISTIC REGRESSION

```python
import tensorflow as tf

# Import MINST data
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("/tmp/data/", one_hot=True)

# Parameters
learning_rate = 0.01
training_epochs = 25
batch_size = 100
display_step = 1

# tf Graph Input
x = tf.placeholder(tf.float32, [None, 784]) # mnist data image of shape 28*28=784
y = tf.placeholder(tf.float32, [None, 10]) # 0-9 digits recognition => 10 classes

# Set model weights
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))

# Construct model
pred = tf.nn.softmax(tf.matmul(x, W) + b) # Softmax

# Minimize error using cross entropy
cost = tf.reduce_mean(-tf.reduce_sum(y*tf.log(pred), reduction_indices=1))
# Gradient Descent
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)

# Initializing the variables
init = tf.initialize_all_variables()
```

```python
# Launch the graph
with tf.Session() as sess:
    sess.run(init)

    # Training cycle
    for epoch in range(training_epochs):
        avg_cost = 0.
        total_batch = int(mnist.train.num_examples/batch_size)
        # Loop over all batches
        for i in range(total_batch):
            batch_xs, batch_ys = mnist.train.next_batch(batch_size)
            # Run optimization op (backprop) and cost op (to get loss value)
            _, c = sess.run([optimizer, cost], feed_dict={x: batch_xs,
                                                          y: batch_ys})
            # Compute average loss
            avg_cost += c / total_batch
        # Display logs per epoch step
        if (epoch+1) % display_step == 0:
            print "Epoch:", '%04d' % (epoch+1), "cost=", "{:.9f}".format(avg_cost)

    print "Optimization Finished!"

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

Source: https://github.com/aymericdamien/TensorFlow-Examples/tree/master/examples

EXAMPLES: MLP

```python
# Create model
def multilayer_perceptron(x, weights, biases):
    # Hidden layer with RELU activation
    layer_1 = tf.add(tf.matmul(x, weights['h1']), biases['b1'])
    layer_1 = tf.nn.relu(layer_1)
    # Hidden layer with RELU activation
    layer_2 = tf.add(tf.matmul(layer_1, weights['h2']), biases['b2'])
    layer_2 = tf.nn.relu(layer_2)
    # Output layer with linear activation
    out_layer = tf.matmul(layer_2, weights['out']) + biases['out']
    return out_layer

# Store layers weight & bias
weights = {
    'h1': tf.Variable(tf.random_normal([n_input, n_hidden_1])),
    'h2': tf.Variable(tf.random_normal([n_hidden_1, n_hidden_2])),
    'out': tf.Variable(tf.random_normal([n_hidden_2, n_classes]))
}
biases = {
    'b1': tf.Variable(tf.random_normal([n_hidden_1])),
    'b2': tf.Variable(tf.random_normal([n_hidden_2])),
    'out': tf.Variable(tf.random_normal([n_classes]))
}
```

```python
# Construct model
pred = multilayer_perceptron(x, weights, biases)

# Define loss and optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(pred, y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)

# Initializing the variables
init = tf.initialize_all_variables()

# Launch the graph
with tf.Session() as sess:
    sess.run(init)

    # Training cycle
    for epoch in range(training_epochs):
        avg_cost = 0.
        total_batch = int(mnist.train.num_examples/batch_size)
        # Loop over all batches
        for i in range(total_batch):
            batch_x, batch_y = mnist.train.next_batch(batch_size)
            # Run optimization op (backprop) and cost op (to get loss value)
            _, c = sess.run([optimizer, cost], feed_dict={x: batch_x,
                                                          y: batch_y})
            # Compute average loss
            avg_cost += c / total_batch
```

Source: https://github.com/aymericdamien/TensorFlow-Examples/tree/master/examples

EXAMPLES: CONVNET

```python
def conv2d(x, W, b, strides=1):
    # Conv2D wrapper, with bias and relu activation
    x = tf.nn.conv2d(x, W, strides=[1, strides, strides, 1], padding='SAME')
    x = tf.nn.bias_add(x, b)
    return tf.nn.relu(x)


def maxpool2d(x, k=2):
    # MaxPool2D wrapper
    return tf.nn.max_pool(x, ksize=[1, k, k, 1], strides=[1, k, k, 1],
                          padding='SAME')
```

```python
# Create model
def conv_net(x, weights, biases, dropout):
    # Reshape input picture
    x = tf.reshape(x, shape=[-1, 28, 28, 1])

    # Convolution Layer
    conv1 = conv2d(x, weights['wc1'], biases['bc1'])
    # Max Pooling (down-sampling)
    conv1 = maxpool2d(conv1, k=2)

    # Convolution Layer
    conv2 = conv2d(conv1, weights['wc2'], biases['bc2'])
    # Max Pooling (down-sampling)
    conv2 = maxpool2d(conv2, k=2)

    # Fully connected layer
    # Reshape conv2 output to fit fully connected layer input
    fc1 = tf.reshape(conv2, [-1, weights['wd1'].get_shape().as_list()[0]])
    fc1 = tf.add(tf.matmul(fc1, weights['wd1']), biases['bd1'])
    fc1 = tf.nn.relu(fc1)
    # Apply Dropout
    fc1 = tf.nn.dropout(fc1, dropout)

    # Output, class prediction
    out = tf.add(tf.matmul(fc1, weights['out']), biases['out'])
    return out
```

Source: https://github.com/aymericdamien/TensorFlow-Examples/tree/master/examples

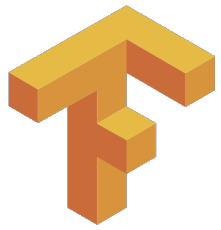
EXAMPLES: RNNS

```python
def RNN(x, weights, biases):

    # Prepare data shape to match `rnn` function requirements
    # Current data input shape: (batch_size, n_steps, n_input)
    # Required shape: 'n_steps' tensors list of shape (batch_size, n_input)

    # Permuting batch_size and n_steps
    x = tf.transpose(x, [1, 0, 2])
    # Reshaping to (n_steps*batch_size, n_input)
    x = tf.reshape(x, [-1, n_input])
    # Split to get a list of 'n_steps' tensors of shape (batch_size, n_input)
    x = tf.split(0, n_steps, x)

    # Define a lstm cell with tensorflow
    lstm_cell = rnn_cell.BasicLSTMCell(n_hidden, forget_bias=1.0)

    # Get lstm cell output
    outputs, states = rnn.rnn(lstm_cell, x, dtype=tf.float32)

    # Linear activation, using rnn inner loop last output
    return tf.matmul(outputs[-1], weights['out']) + biases['out']

pred = RNN(x, weights, biases)
```

Source: https://github.com/aymericdamien/TensorFlow-Examples/tree/master/examples

OTHER FEATURES (NOT COVERED TODAY)

- Checkpoints: can save your model & load if for later use
- Serving: use TF in production
- Details of inner workings of TensorFlow (see academic paper for details)



Background

Usage

Under-the-hood

Resources

OPERATIONS AND KERNELS

OPERATION	Operations have names and represent an abstract computation, such as ‘add’ and ‘matrix multiply’
KERNEL	Implementation of an operation, designed for specific types of devices, such as CPU or GPU

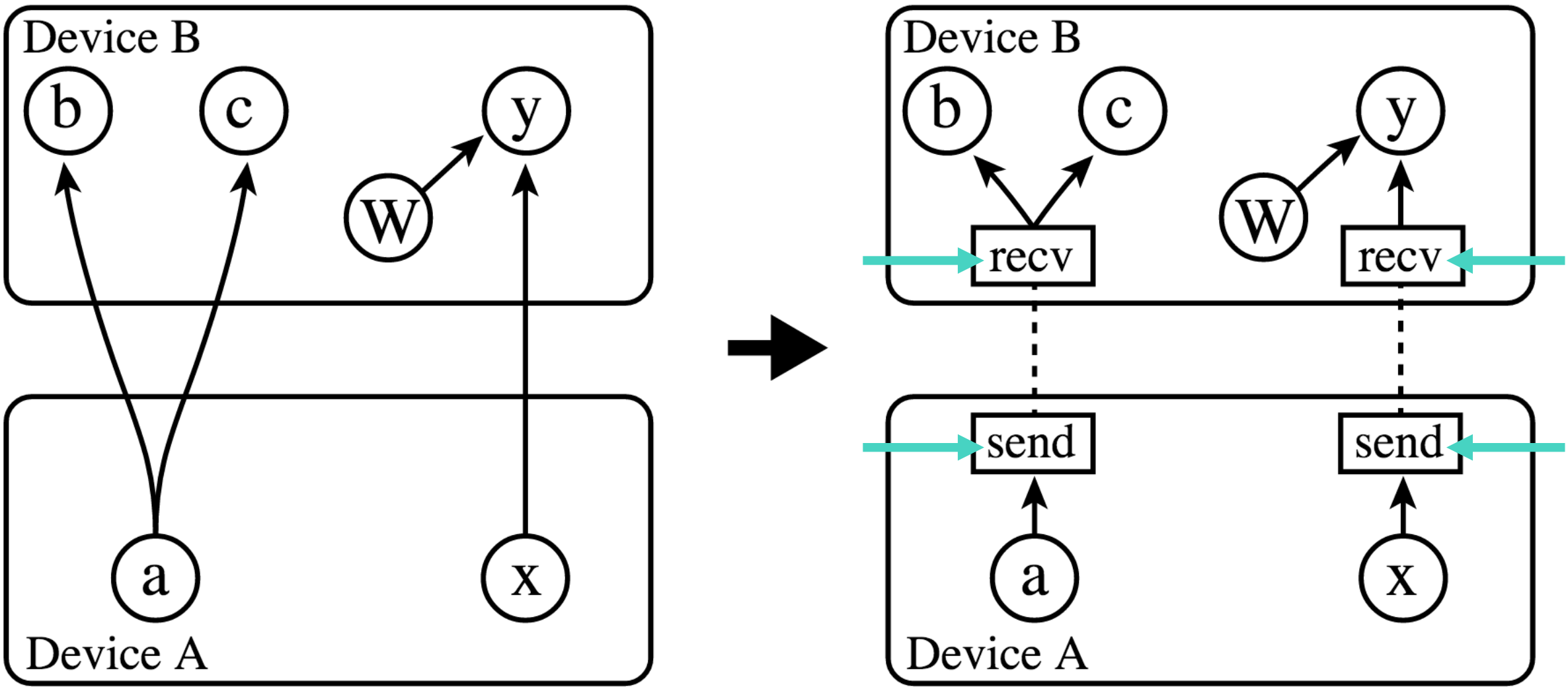
PARALLELIZING

```
In [6]: with tf.device('/cpu:0'):
   ...:         a = tf.constant(2)
   ...:         b = tf.constant(3)
   ...:         c = a*b
   ...:

In [7]: with tf.Session() as sess:
   ...:         print c.eval()
   ...:
6
```

PARALLELIZING: MAPPING ALGORITHM

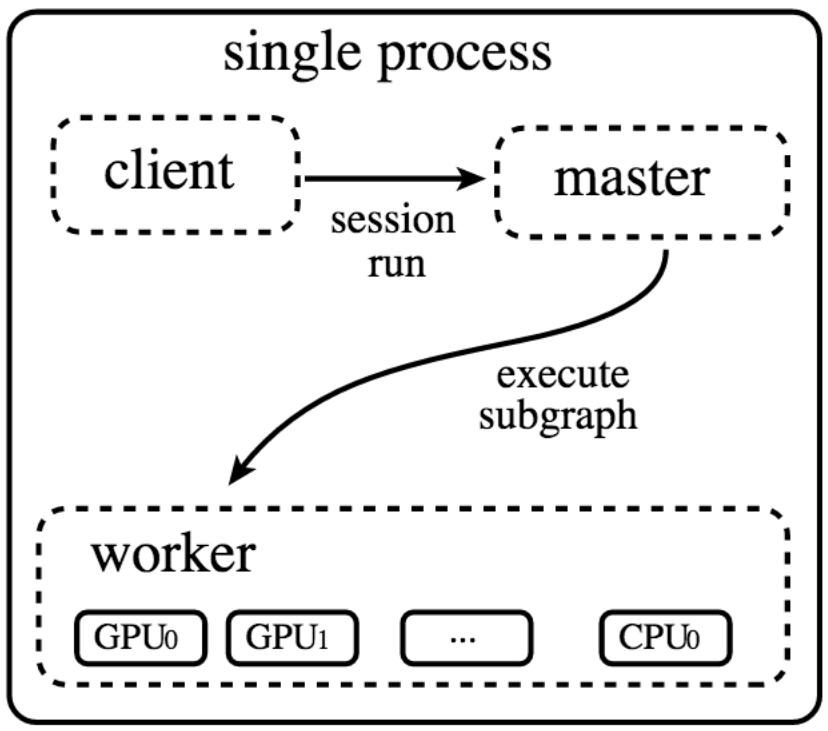
- TF maps computation to available devices
- In summary, during mapping:
  - (1) Algorithm receives cost model
  - (2) Algorithm simulates execution of graph and makes node-placement decisions
  - (3) Real execution is run correspondingly
- Improvements to placement algorithm are in progress

PARALLELIZING: CROSS-DEVICE COMMUNICATION

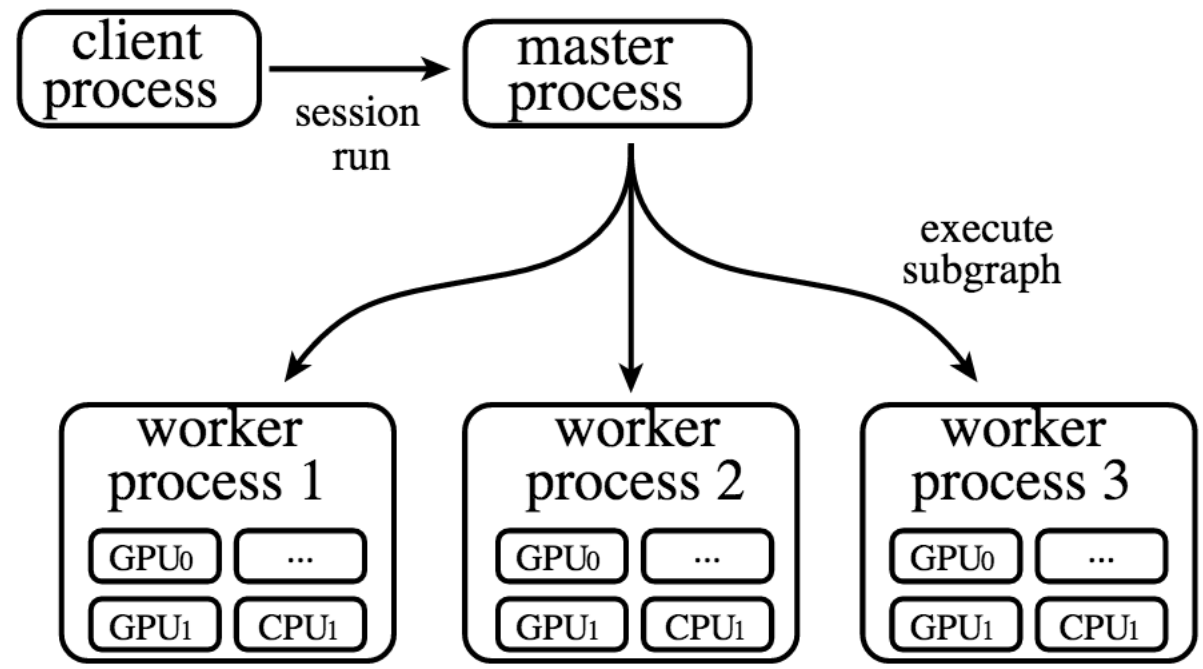


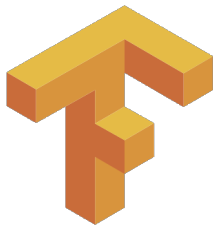
PARALLELIZING:
CLIENT, MASTER AND WORKER

1 machine



Distributed system





Background

Usage

Under-the-hood

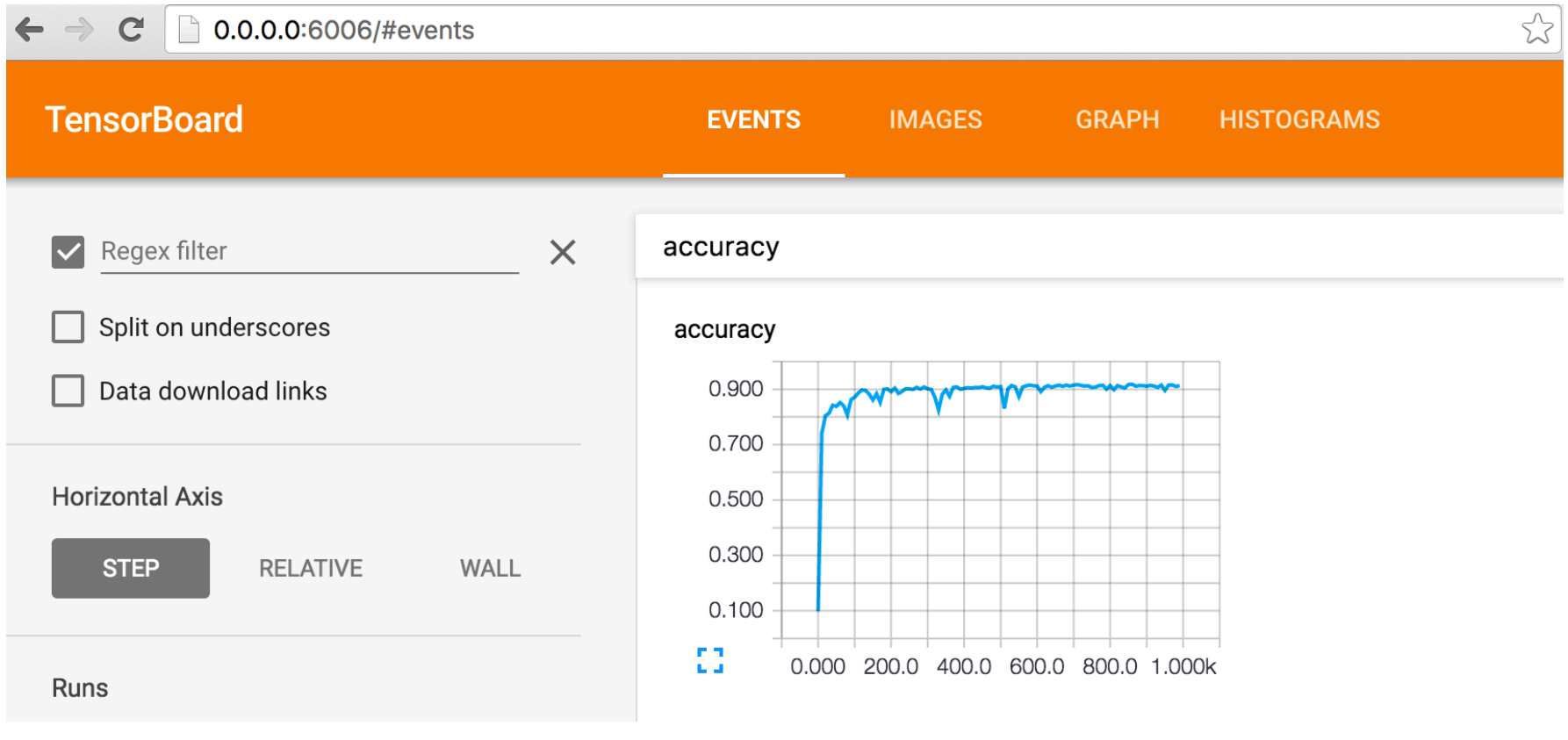
Resources

TENSORBOARD:
(1) WRITE A SUMMARY

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets('MNIST_data', one_hot=True)

with tf.Session() as sess:

    # Create the model
    x = tf.placeholder(tf.float32, [None, 784], name="x-input")
    W = tf.Variable(tf.zeros([784,10]), name="weights")
    b = tf.Variable(tf.zeros([10], name="bias"))

    # use a name scope to organize nodes in the graph visualizer
    with tf.name_scope("Wx_b") as scope:
      y = tf.nn.softmax(tf.matmul(x,W) + b)

    # Add summary ops to collect data
    w_hist = tf.histogram_summary("weights", W)
    b_hist = tf.histogram_summary("biases", b)
    y_hist = tf.histogram_summary("y", y)

    # Define loss and optimizer
    y_ = tf.placeholder(tf.float32, [None,10], name="y-input")
    # More name scopes will clean up the graph representation
    with tf.name_scope("xent") as scope:
      cross_entropy = -tf.reduce_sum(y_*tf.log(y))
      ce_summ = tf.scalar_summary("cross entropy", cross_entropy)
    with tf.name_scope("train") as scope:
      train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)

    with tf.name_scope("test") as scope:
      correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
      accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
      accuracy_summary = tf.scalar_summary("accuracy", accuracy)

    # Merge all the summaries and write them out to /tmp/mnist_logs
    merged = tf.merge_all_summaries()
    writer = tf.train.SummaryWriter("/tmp/mnist_logs", sess.graph_def)
    tf.initialize_all_variables().run()

    # Train the model, and feed in test data and record summaries every 10 steps

    for i in range(1000):
      if i % 10 == 0:  # Record summary data, and the accuracy
        feed = {x: mnist.test.images, y_: mnist.test.labels}
        result = sess.run([merged, accuracy], feed_dict=feed)
        summary_str = result[0]
        acc = result[1]
        writer.add_summary(summary_str, i)
        print("Accuracy at step %s: %s" % (i, acc))
      else:
        batch_xs, batch_ys = mnist.train.next_batch(100)
        feed = {x: batch_xs, y_: batch_ys}
        sess.run(train_step, feed_dict=feed)

    print(accuracy.eval({x: mnist.test.images, y_: mnist.test.labels}))
```

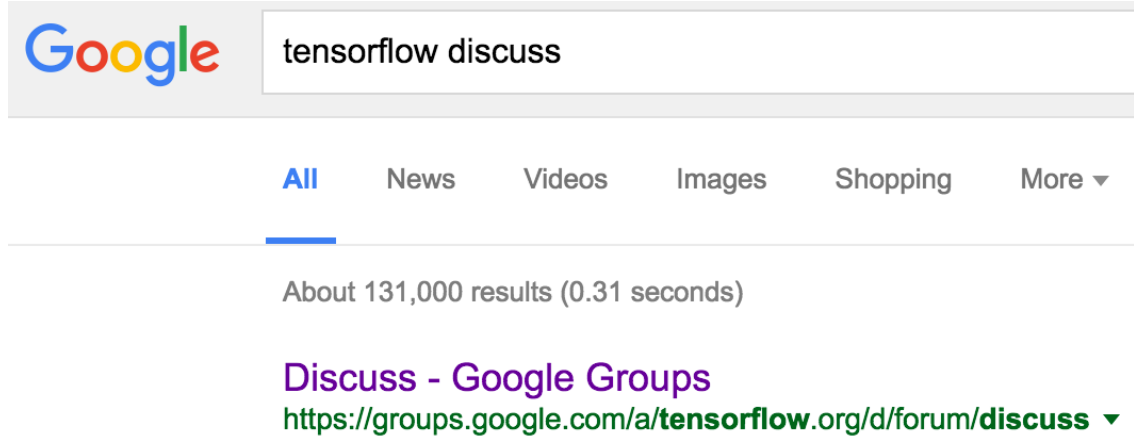
TENSORBOARD:
(2) VISUALIZE SUMMARY

```
MacBook-Pro-60:~ pyry$ tensorboard --logdir=/tmp/mnist_logs
WARNING:tensorflow:Found more than one graph event per run. Overwriting the grap
h with the newest event.
WARNING:tensorflow:Found more than one graph event per run. Overwriting the grap
h with the newest event.
Starting TensorBoard 16 on port 6006
(You can navigate to http://0.0.0.0:6006)
```



DEBUGGING TIPS

- Use Numpy: try implementing forward-graph
- Use iPython and tf.interactiveSession()
- Active community:



- Visualize with TensorBoard
- Review resources (next page)

RESOURCES & REFERENCES

- TF project
  - www.tensorflow.org
  - www.github.com/tensorflow
- TF paper + notes
  - ADD PAPER
  - https://github.com/samjabrahams/tensorflow-white-paper-notes
- Examples
  - https://github.com/aymericdamien/TensorFlow-Examples/tree/master/examples
- Comprehensive list of resources
  - https://github.com/jtoy/awesome-tensorflow

HANDS-ON LAB

- nvlabs.qwiklab.com : **Exploring TensorFlow on GPUs**
  - Linear Regression
  - Multilayer Convolutional Network
  - Sequence Autoencoder
  - Multi-GPU Computation
  - Parallelism on GPUs
  - Distributed Training

True AI IS HIRING

SENIOR SOFTWARE ENGINEER

As a Senior Software Engineer, you will be working closely with our researchers and leading the development of technical infrastructure at True AI from ground up, building highly scalable and real time system architecture that blends well with underlying deep learning algorithms

DEEP LEARNING RESEARCH ENGINEER

We need a Deep Learning Research Engineer who is passionate about taking AI to the next level, and who is interested in building the company alongside the founders.

+ INTERNSHIPS

FULL STACK DEVELOPER

We need a Full Stack Developer who is passionate about taking AI to the next level, and who is interested in building the company alongside the founders. You will be playing a key role in the development of our main web application and browser based plugins, as well as integration on backend.