TensorFlow: a Framework for Scalable Machine Learning

ACM Learning Center, 2016
You probably want to know...

- What is TensorFlow?
- Why did we create TensorFlow?
- How does TensorFlow work?
- Code: Linear Regression
- Code: Convolution Deep Neural Network
- Advanced Topics: Queues and Devices
- Fast, flexible, and scalable open-source machine learning library
- One system for research and production
- Runs on CPU, GPU, TPU, and Mobile
- Apache 2.0 license
Machine learning gets complex quickly

Modeling complexity
Machine learning gets complex quickly

Distributed System

Heterogenous System
TensorFlow Handles Complexity

Modeling complexity

Distributed System

Heterogenous System
Under the Hood
A multidimensional array.

TensorFlow

A graph of operations.
The TensorFlow Graph

Computation is defined as a graph

- Graph is defined in high-level language (Python)
- Graph is compiled and optimized
- Graph is executed (in parts or fully) on available low level devices (CPU, GPU, TPU)
- Nodes represent computations and state
- Data (tensors) flow along edges
Build a graph; then run it.

...  
c = tf.add(a, b)  
...

session = tf.Session()  
value_of_c = session.run(c, {a=1, b=2})
Any Computation is a TensorFlow Graph

biases
weights
examples
labels
MatMul
Add
Relu
Xent
Any Computation is a TensorFlow Graph

variables

biases

weights

examples

labels

with state

MatMul

Add

Relu

Xent
Automatic Differentiation

Automatically add ops which compute gradients for variables
Any Computation is a TensorFlow Graph

Simple gradient descent:

\[
\text{learning rate} \times \text{grad} = \text{Mul} \quad \text{with state}
\]
Any Computation is a TensorFlow Graph

Devices: Processes, Machines, CPUs, GPUs, TPUs, etc
Send and Receive Nodes

Devices: Processes, Machines, CPUs, GPUs, TPUs, etc
Send and Receive Nodes

Devices: Processes, Machines, CPUs, GPUs, TPUs, etc
Linear Regression
Linear Regression

\[ y = Wx + b \]
What are we trying to do?

**Mystery equation:** $y = 0.1 \times x + 0.3 + \text{noise}$

**Model:** $y = W \times x + b$

**Objective:** Given enough $(x, y)$ value samples, figure out the value of $w$ and $b$. 
\[ y = Wx + b \] in TensorFlow

```python
import tensorflow as tf
```
\[ y = Wx + b \] in TensorFlow

```python
import tensorflow as tf

x = tf.placeholder(shape=[None],
                   dtype=tf.float32, name="x")
```
y = Wx + b in TensorFlow

```python
import tensorflow as tf

x = tf.placeholder(shape=[None],
                   dtype=tf.float32, name="x")

W = tf.get_variable(shape=[], name="W")
```
y = Wx + b in TensorFlow

```python
import tensorflow as tf

x = tf.placeholder(shape=[None],
                   dtype=tf.float32, name="x")

W = tf.get_variable(shape=[], name="W")

b = tf.get_variable(shape=[], name="b")
```
import tensorflow as tf

x = tf.placeholder(shape=[None],
                   dtype=tf.float32, name="x")

W = tf.get_variable(shape=[], name="W")

b = tf.get_variable(shape=[], name="b")

y = W * x + b
Variables Must be Initialized

\[
\text{init\_op} = \text{tf.initialize\_all\_variables}()
\]

Collects all variable initializers

\[
\text{sess = tf.Session}()
\]

Makes an execution environment

\[
\text{sess.run(init\_op)}
\]

Actually initialize the variables

Graph:

- \(x\) is input
- \(W\) is weight
- \(b\) is bias
- \(+\) is addition operator
- \(y\) is output
Running the Computation

\[ x_{\text{in}} = 3 \]

\[
\text{sess.run}(y, \text{feed_dict} = \{x: x_{\text{in}}\})
\]

- Only what’s used to compute a fetch will be evaluated
- All Tensors can be fed, but all placeholders must be fed
Putting it all together

```python
import tensorflow as tf
x = tf.placeholder(shape=[None],
                   dtype=tf.float32,
                   name='x')
W = tf.get_variable(shape=[], name='W')
b = tf.get_variable(shape=[], name='b')
y = W * x + b

with tf.Session() as sess:
    sess.run(tf.initialize_all_variables())
    print(sess.run(y, feed_dict={x: x_in}))
```
Define a Loss

Given \( x, y \) compute a loss, for instance:

\[
L = (y - y_{\text{label}})^2
\]

# create an operation that calculates loss.
loss = tf.reduce_mean(tf.square(y - y_data))
Minimize loss: optimizers

- `tf.train.AdadeltaOptimizer`
- `tf.train.AdagradOptimizer`
- `tf.train.AdagradDAOptimizer`
- `tf.train.AdamOptimizer`

...
Feed \((x, y_{\text{label}})\) pairs and adjust \(W\) and \(b\) to decrease the loss.

\[
W \leftarrow W - \eta \left( \frac{dL}{dW} \right)
\]

\[
b \leftarrow b - \eta \left( \frac{dL}{db} \right)
\]

# Create an optimizer

```python
optimizer = tf.train.GradientDescentOptimizer(0.5)
```

# Create an operation that minimizes loss.

```python
train = optimizer.minimize(loss)
```
loss = tf.reduce_mean(tf.square(y - y_label))

optimizer = tf.train.GradientDescentOptimizer(0.5)

train = optimizer.minimize(loss)

with tf.Session() as sess:
    sess.run(tf.initialize_all_variables())
    for i in range(1000):
        sess.run(train, feed_dict={x: x_in[i],
                                    y_label: y_in[i]})
TensorBoard
Deep Neural Network
Is this a cat or dog?

Deep neural network

Input layer

Activated neurons

Output layer
Remember linear regression?

import tensorflow as tf
x = tf.placeholder(shape=[None],
   dtype=tf.float32,
   name='x')

W = tf.get_variable(shape=[], name='W')
b = tf.get_variable(shape=[], name='b')
y = W * x + b

loss = tf.reduce_mean(tf.square(y - y_label))

optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)
...

Build the graph
Convolutional DNN

```
x = tf.contrib.layers.conv2d(x, kernel_size=[5,5], ...)
x = tf.contrib.layers.max_pool2d(x, kernel_size=[2,2], ...)
x = tf.contrib.layers.conv2d(x, kernel_size=[5,5], ...)
x = tf.contrib.layers.max_pool2d(x, kernel_size=[2,2], ...)
x = tf.contrib.layers.fully_connected(x, activation_fn=tf.nn.relu)
x = tf.contrib.layers.dropout(x, 0.5)
logits = tf.config.layers.linear(x)
```

https://github.com/martinwicke/tensorflow-tutorial/blob/master/2_mnist.ipynb
Defining Complex Networks

- Parameters
- network
- loss
- gradients
- Mul
- learning rate
- \( \text{learning rate} \times \text{network gradients} \)
Distributed TensorFlow
Data Parallelism

Parameter Servers

Model Replicas

Data

$\Delta p'$

$p'$
Describe a cluster: ClusterSpec

tf.train.ClusterSpec({
    "worker": [
        "worker0.example.com:2222",
        "worker1.example.com:2222",
        "worker2.example.com:2222",
    ],
    "ps": [
        "ps0.example.com:2222",
        "ps1.example.com:2222",
    ]})
Share the graph across devices

with tf.device("/job:ps/task:0"):
    weights_1 = tf.Variable("")
    biases_1 = tf.Variable("")

with tf.device("/job:ps/task:1"):
    weights_2 = tf.Variable("")
    biases_2 = tf.Variable("")

with tf.device("/job:worker/task:7"):
    input, labels = ...
    layer_1 = tf.nn.relu(tf.matmul(input, weights_1) + biases_1)
    logits = tf.nn.relu(tf.matmul(layer_1, weights_2) + biases_2)
    train_op = ...

with tf.Session("grpc://worker7.example.com:2222") as sess:
    for _ in range(10000):
        sess.run(train_op)
Input Pipelines with Queues
Tutorials & Courses

Tutorials on tensorflow.org:

Image recognition: https://www.tensorflow.org/tutorials/image_recognition

Word embeddings: https://www.tensorflow.org/versions/word2vec

Language Modeling: https://www.tensorflow.org/tutorials/recurrent

Translation: https://www.tensorflow.org/versions/seq2seq

Deep Dream: https://tensorflow.org/code/tensorflow/examples/tutorials/deepdream/deepdream.ipynb
Thank you and have fun!

Martin Wicke
@martin_wicke

Rajat Monga
@rajatmonga
Extras
Inception

An Alaskan Malamute (left) and a Siberian Husky (right). Images from Wikipedia.

Show and Tell

Parsey McParseface

https://research.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html
Alice and Bob took the train to visit the zoo. They saw a baby giraffe, a lion, and a flock of colorful tropical birds.

Alice and Bob visited the zoo and saw animals and birds.
Mobile TensorFlow

TensorFlow was designed with mobile and embedded platforms in mind. We have sample code and build support you can try now for these platforms:

Android

iOS

Raspberry Pi

Many applications can benefit from on-device processing. Google Translate's instant visual translation is a great example. By running its processing locally, users get an incredibly responsive and interactive experience.

Mobile TensorFlow makes sense when there is a poor or missing network connection, or where sending continuous data to a server would be too expensive. We are working to help developers make lean mobile apps using TensorFlow, both by continuing to reduce the code footprint, and supporting quantization and lower precision arithmetic that reduce model size.
TensorFlow Distributed Execution System

C++ front end
Python front end

Bindings + Compound Ops

Architecture

Ops
add
mul
Print
reshape
...

Kernels
CPU
GPU
TPU
Android
iOS
...

TensorFlow Distributed Execution System