



Tensorflow: Introduction

Alex Ozerin

Yandex

ozerin@phystech.edu

What is TF?



“TensorFlow is an open source software library for numerical computation using **data flow graphs**”

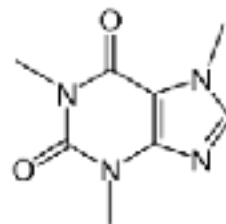
Why TF?



CNTK



theano



deephack.me

Why TF?



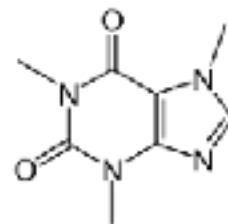
mxnet



CNTK



theano



deephack.me

Why TF?



- Stable
- Well-documented sources
- Flexibility
- Portability
- Scalability
- Popularity

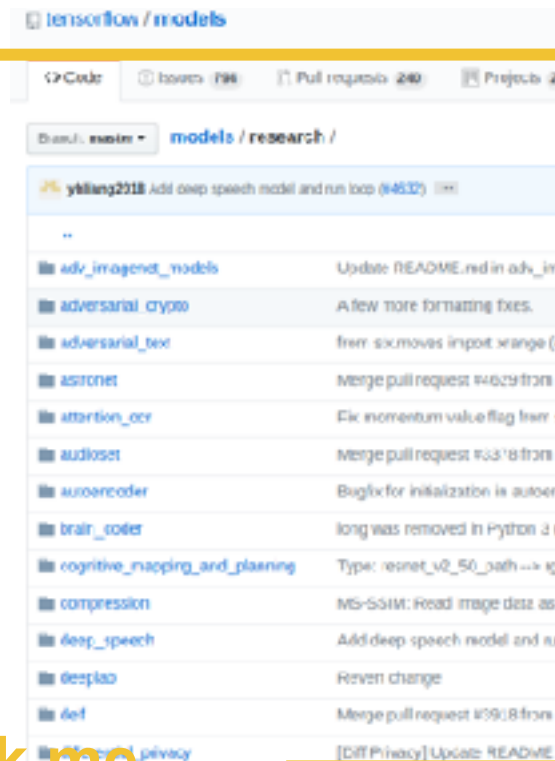


deephack.me

Why TF? Popularity



- Tensorboard
- Tensorflow Hub/Models
- TF Serving
- TF Lite
- Lucid
- Cleverhans
- ...



deephack.me

Getting Started



```
import tensorflow as tf
```



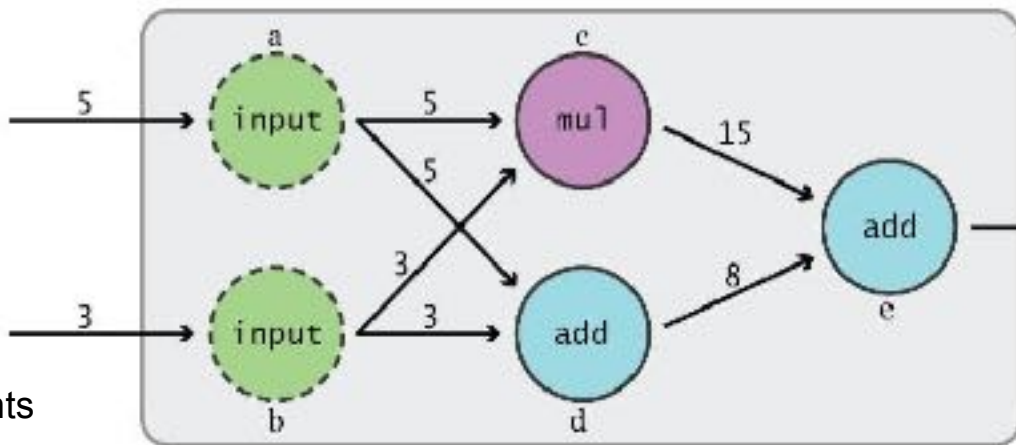
deephack.me

Graphs and sessions



Two phases:

- Define a computation (graph)
- Execute it (session)



Nodes: operators, variables, and constants
Edges: tensors

Tensors



In physics and mathematics:

$$\partial_{\alpha} F^{\alpha\beta} = \mu_0 J^{\beta}$$

In ML/DL just an n-dimensional array

0-d tensor: **scalar**

1-d tensor: **vector**

2-d tensor: **matrix**

...

images:

[batch_size, height, width, **channels**]
NHWC

time series, text:

[batch_size, time_steps, **channels**]

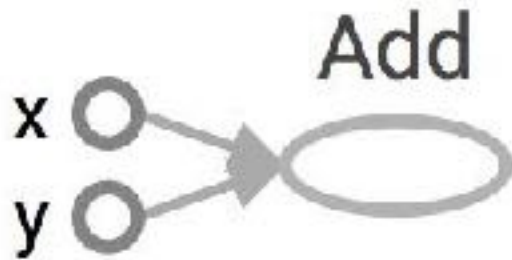
Data Flow Graph



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

```
>> Tensor("Add:0", shape=(),  
dtype=int32)  
(Not 8)
```

Nodes: operators, variables, and constants
Edges: tensors



How to run computation

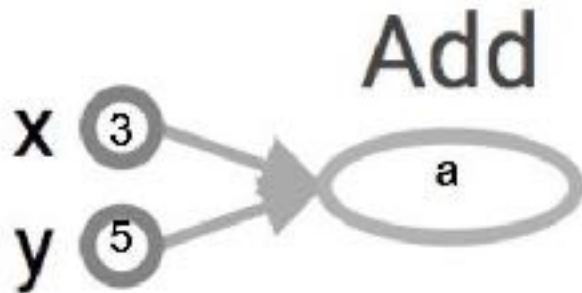


Create a session

Evaluate the graph, feed and fetch

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

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



tf.Session() and session.run(...)



```
import tensorflow as tf

# default graph created, look at tf.get_default_graph()

x, y = tf.placeholder(tf.float32), tf.placeholder(tf.float32) # just input
nodes
a = tf.add(x, y) # some nodes appeared in graph

with tf.Session() as sess: # memory allocated

    sess.run(a, feed_dict={x: 1.0, y: -1.0}) # execution run

    sess.run(a) # error, value is not persistent between sess.run
```

More graphs

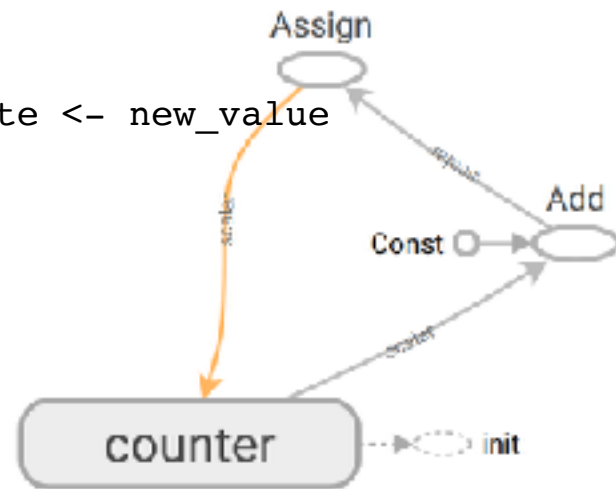


Simple counter with persistent value between executions

```
state = tf.Variable(0, name="counter")
new_value = tf.add(state, tf.constant(1))
update_op = tf.assign(state, new_value) # state <- new_value
```

```
with tf.Session() as sess:
    sess.run(tf.initialize_all_variables())
    sess.run(state) # 0
```

```
for i in range(10):
    sess.run(update_op)
    sess.run(state) # 1,2,3,4, ...
```



Neural Networks

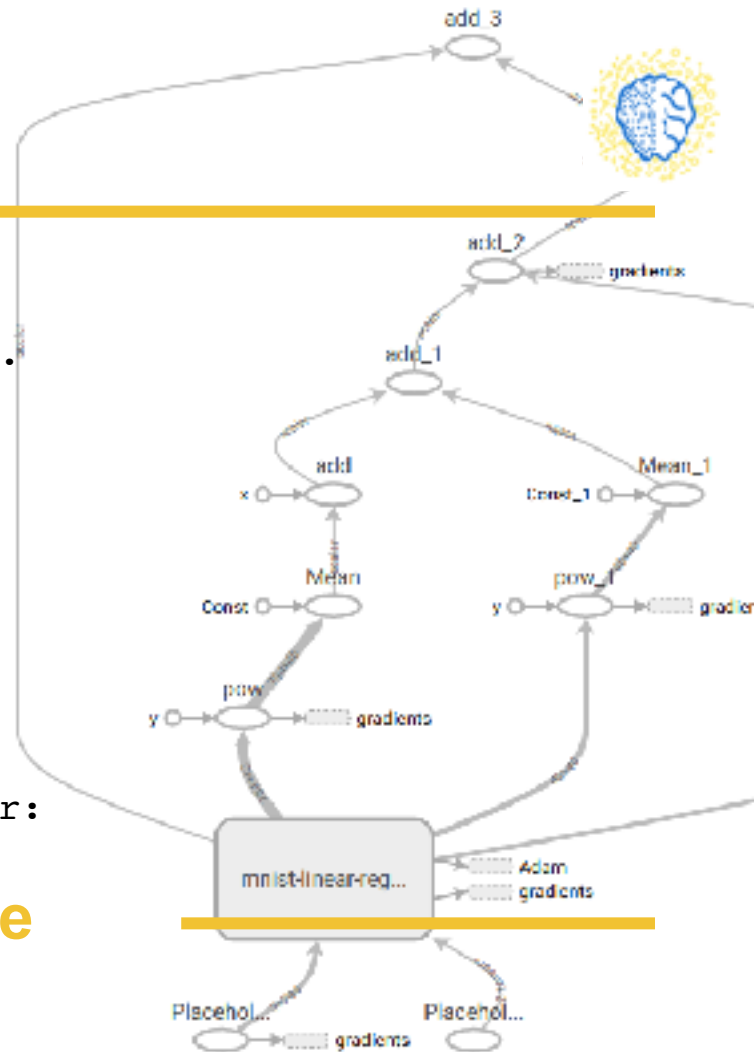
```
x, y = tf.placeholder(tf.float32, ...],  
tf.placeholder(tf.int32, ...)  
W, b = tf.get_variable("very_weight", ....), ...
```

```
y_pred = tf.matmul(x, W) + b  
loss = tf.reduce_mean((y - y_pred) ** 2)
```

```
opt = tf.train.AdamOptimizer()  
train_op = opt.minimize(loss)
```

```
with tf.Session() as sess:  
    sess.run(tf.initialize_all_variables())  
    for (x_batch, y_batch) in very_data_iterator:  
        loss, _ = sess.run([loss, train_op],  
                            feed_dict={x: x_batch, y: y_batch})
```

deephack.me



Why graphs



Pros:

1. Only route from feed to fetch will executes
2. Auto-differentiation
3. Suitable for distributed computation (CPU/GPU/TPU/...)

Cons:

1. Tricky to adapt some algorithms to graph execution
2. Difficult to debug

How to work with TF



- Build iterator for your data
- Make the model with placeholder and optimizer
- Run epochs with training and validation subgraphs
- Monitor the process
- Measure performance

Attributes (1)

T	("type":"DT_FLOAT")
Device	/job:localhost/replica:0/task:0/cpu:0

Inputs (1)

layer1/W/x_plus_b/add	9x500
-----------------------	-------

Outputs (6)

dropout/dropout/Shape	VALUE
dropout/dropout/mul	VALUE
train/gradients/dropout/dropout/mul	VALUE
train/gradients/dropout/dropout/mul	VALUE
train/gradients/layer1/activation_gr	VALUE
layer1/HistogramSummary	VALUE

Node Stats

Memory	195 KB
Compute Time	93 μ s
Tensor Output Sizes	[100, 500]

Remove from main graph

deephack.me

Simplified TF



1. High-level API: Keras, TFLearn, Pretty Tensor
2. Some common tools: TFSlim
3. Standardize boiler-plate code: **tf.Estimators**
4. Input pipeline: **tf.data.Dataset**

Tensorboard



```
tf.summary.scalar('cross_entropy', cross_entropy)

merged = tf.summary.merge_all()

writer = tf.summary.FileWriter('./train', sess.graph)

...

summary, _, step = sess.run([merged, train_step, global_step],
                             feed_dict=...)

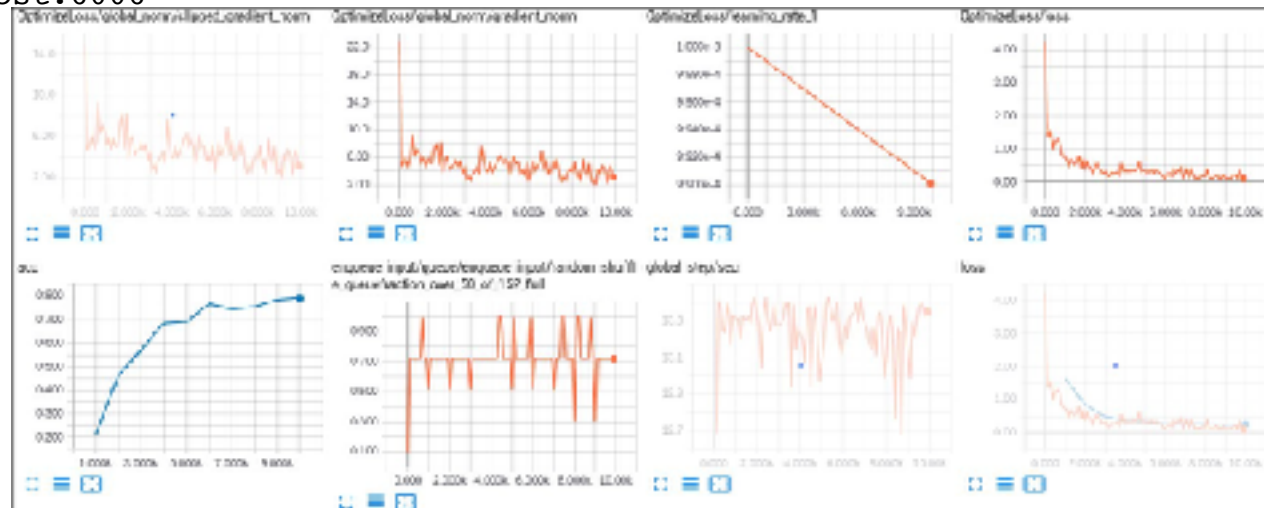
writer.add_summary(summary, step)
```

Tensorboard



```
# >> tensorboard --logdir=./train/
```

```
# goto: http://localhost:6006
```



deephack.me

Hub: images/texts/



```
import tensorflow_hub as hub

with tf.Graph().as_default():
    embed = hub.Module("https://tfhub.dev/sonit = embed(["Hello world!",
                    "Benoît B. Mandelbrot",
                    "deephack.me"])
    with tf.Session() as sess:
        sess.run(tf.global_variables_initialize
        sess.run(tf.tables_initializer())

        sess.run(it) # vectorization of texts
```

deephack.me

Modules trained on ImageNet (ILSVRC-2012-CLS)

Inception and Inception-ResNet

- Inception V1: classification, feature_extractor
- Inception V2: classification, feature_extractor
- Inception V3: classification, feature_extractor
- Inception-ResNet V2: classification, feature_extractor

MobileNet

MobileNet is a family of neural networks controlled by a multiplier for the depth (number of features), and input images. See the module documentation for details.

- MobileNet V2

	224x224	160x160	128x128	96x96
100%	classification feature_extractor	classification feature_extractor	classification feature_extractor	classification feature_extractor
75%	classification feature_extractor	classification feature_extractor	classification feature_extractor	classification feature_extractor
50%	classification feature_extractor	classification feature_extractor	classification feature_extractor	classification feature_extractor
25%	classification feature_extractor	classification feature_extractor	classification feature_extractor	classification feature_extractor

- MobileNet V2 Instrumented for quantization with TF Lite ("Quantize")

	224x224	160x160	128x128	96x96
100%	classification feature_extractor	classification feature_extractor	classification feature_extractor	classification feature_extractor
75%	classification feature_extractor	classification feature_extractor	classification feature_extractor	classification feature_extractor
50%	classification feature_extractor	classification feature_extractor	classification feature_extractor	classification feature_extractor
25%	classification feature_extractor	classification feature_extractor	classification feature_extractor	classification feature_extractor

Conclusion



1. TF is a two-phase framework (define computations, then execute)
2. Nice tools for visualization and monitoring (tensorboard)

What's next?

- `tf.data.Dataset` API
- `MonitoredSession`
- Estimators API
- Distributed execution (horovod)