

With TF 1.0!



Lab I

TensorFlow Basics

Sung Kim <hunkim+ml@gmail.com>

Code: <https://github.com/hunkim/DeepLearningZeroToAll/>



Call for comments

Please feel free to add comments directly on these slides

Other slides: <https://goo.gl/jPtVNT>



With TF 1.0!



Lab I

TensorFlow Basics

Sung Kim <hunkim+ml@gmail.com>

Code: <https://github.com/hunkim/DeepLearningZeroToAll/>



An open-source software library for Machine Intelligence

[GET STARTED](#)

TensorFlow 1.0 has arrived!

We're excited to announce the release of TensorFlow 1.0! Check out the migration guide to upgrade your code with ease.

[UPGRADE NOW](#)

Dynamic graphs in TensorFlow

We've open-sourced TensorFlow Fold to make it easier than ever to work with input data with varying shapes and sizes.

[LEARN MORE](#)

The 2017 TensorFlow Dev Summit

Thousands of people from the TensorFlow community participated in the first flagship event. Watch the keynote and talks.

[WATCH VIDEOS](#)

<https://www.tensorflow.org>

Call for comments

Please feel free to add comments directly on these slides

Other slides: <https://goo.gl/jPtVNT>



With TF 1.0!

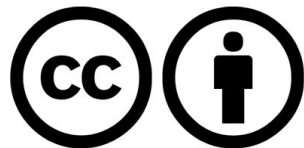


Lab I

TensorFlow Basics

Sung Kim <hunkim+ml@gmail.com>

Code: <https://github.com/hunkim/DeepLearningZeroToAll/>



An open-source software library for Machine Intelligence

[GET STARTED](#)

TensorFlow 1.0 has arrived!

We're excited to announce the release of TensorFlow 1.0! Check out the migration guide to upgrade your code with ease.

[UPGRADE NOW](#)

Dynamic graphs in TensorFlow

We've open-sourced TensorFlow Fold to make it easier than ever to work with input data with varying shapes and sizes.

[LEARN MORE](#)

The 2017 TensorFlow Dev Summit

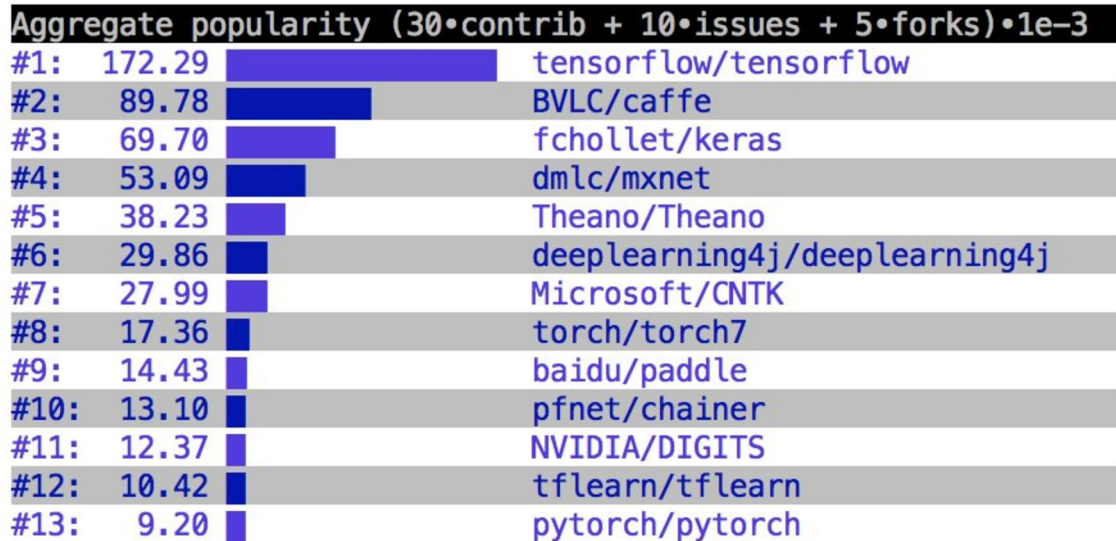
Thousands of people from the TensorFlow community participated in the first flagship event. Watch the keynote and talks.

[WATCH VIDEOS](#)

<https://www.tensorflow.org>

TensorFlow

Deep learning libraries:
Accumulated GitHub metrics



Deep learning libraries: growth over past three months

new contributors from 2016-10-09 to 2017-02-10

#1:	192	tensorflow/tensorflow
#2:	89	dmlc/mxnet
#3:	78	fchollet/keras
#4:	42	baidu/paddle
#5:	29	Microsoft/CNTK
#6:	23	pfnet/chainer
#7:	21	Theano/Theano
#8:	20	deeplearning4j/deeplearning4j
#9:	20	tflearn/tflearn
#10:	19	BVLC/caffe
#11:	9	torch/torch7
#12:	3	NVIDIA/DIGITS

new forks from 2016-10-09 to 2017-02-10

#1:	6525	tensorflow/tensorflow
#2:	1822	BVLC/caffe
#3:	1316	fchollet/keras
#4:	999	dmlc/mxnet
#5:	909	deeplearning4j/deeplearning4j
#6:	887	Microsoft/CNTK
#7:	324	tflearn/tflearn
#8:	321	baidu/paddle
#9:	287	Theano/Theano
#10:	257	torch/torch7
#11:	175	NVIDIA/DIGITS
#12:	142	pfnet/chainer

new issues from 2016-10-09 to 2017-02-10

#1:	1563	tensorflow/tensorflow
#2:	979	fchollet/keras
#3:	871	dmlc/mxnet
#4:	646	baidu/paddle
#5:	486	Microsoft/CNTK
#6:	361	deeplearning4j/deeplearning4j
#7:	318	BVLC/caffe
#8:	217	NVIDIA/DIGITS
#9:	214	Theano/Theano
#10:	167	tflearn/tflearn
#11:	150	pfnet/chainer
#12:	90	torch/torch7

aggregate metrics growth from 2016-10-09 to 2017-02-10

#1:	54.01	tensorflow/tensorflow
#2:	18.71	fchollet/keras
#3:	16.38	dmlc/mxnet
#4:	12.86	BVLC/caffe
#5:	10.17	Microsoft/CNTK
#6:	9.32	baidu/paddle
#7:	8.75	deeplearning4j/deeplearning4j
#8:	4.21	Theano/Theano
#9:	3.89	tflearn/tflearn
#10:	3.14	NVIDIA/DIGITS
#11:	2.90	pfnet/chainer
#12:	2.46	torch/torch7



François Chollet @fchollet · Feb 11

Time for an update: what does the deep learning library landscape look like, seen from GitHub? [pic.twitter.com/QDZyVrYBd](https://twitter.com/fchollet/status/830499993450450944)

[https://twitter.com/fchollet/status/830499993450450944/](https://twitter.com/fchollet/status/830499993450450944)

TensorFlow

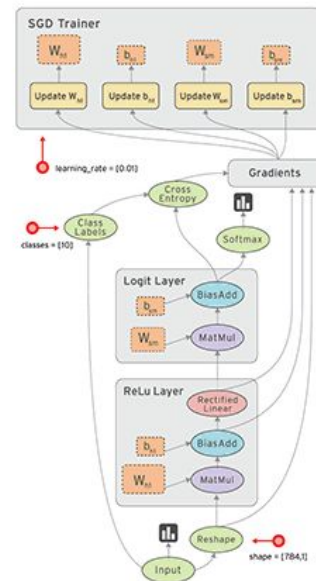
- TensorFlow™ is an open source software library for numerical computation using data flow graphs.
- Python!



<https://www.tensorflow.org/>

What is a Data Flow Graph?

- Nodes in the graph represent mathematical operations
- Edges represent the multidimensional data arrays (tensors) communicated between them.



Installing TensorFlow

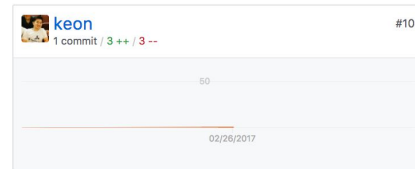
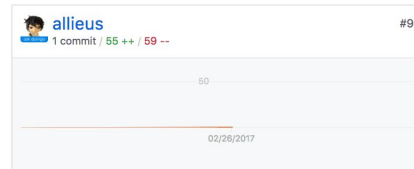
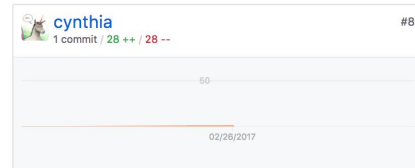
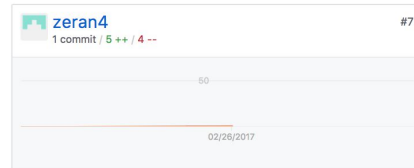
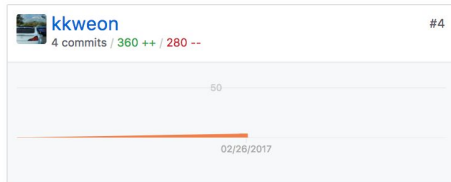
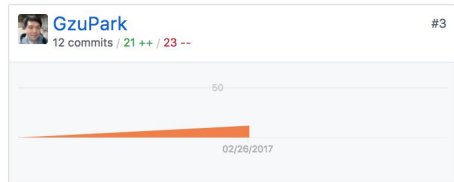
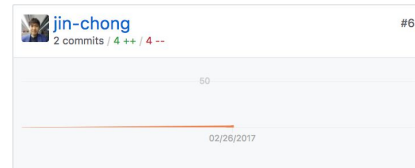
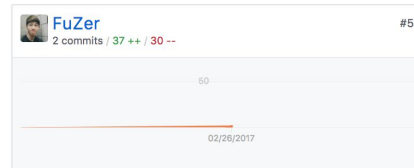
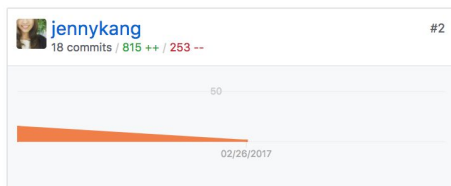
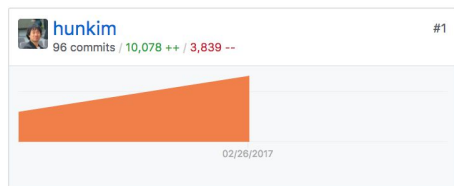
- Linux, Mac OSX, Windows
 - (sudo -H) pip install --upgrade tensorflow
 - (sudo -H) pip install --upgrade tensorflow-gpu
- From source
 - bazel ...
 - https://www.tensorflow.org/install/install_sources
- Google search/Community help
 - <https://www.facebook.com/groups/TensorFlowKR/>

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

Check installation and version

```
Sungs-MacBook-Pro:hunkim$ python3
Python 3.6.0 (v3.6.0:41df79263a11, Dec 22 2016, 17:23:13)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> import tensorflow as tf
>>> tf.__version__
'1.0.0'
>>>
```

<https://github.com/hunkim/DeepLearningZeroToAll/>



TensorFlow Hello World!

Hello TensorFlow!

```
In [2]: # Create a constant op
# This op is added as a node to the default graph
hello = tf.constant("Hello, TensorFlow!")

# start a TF session
sess = tf.Session()

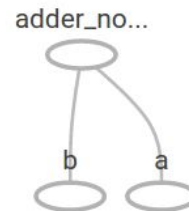
# run the op and get result
print(sess.run(hello))

b'Hello, TensorFlow!'
```

b'String' 'b' indicates *Bytes literals*. <http://stackoverflow.com/questions/6269765/>

<https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-01-basics.ipynb>

Computational Graph



```
In [4]: node1 = tf.constant(3.0, tf.float32)
node2 = tf.constant(4.0) # also tf.float32 implicitly
node3 = tf.add(node1, node2)
```

```
In [5]: print("node1:", node1, "node2:", node2)
print("node3: ", node3)
```

```
node1: Tensor("Const_1:0", shape=(), dtype=float32) node2: Tensor("Const_2:0", shape=(), dtype=float32)
node3: Tensor("Add:0", shape=(), dtype=float32)
```

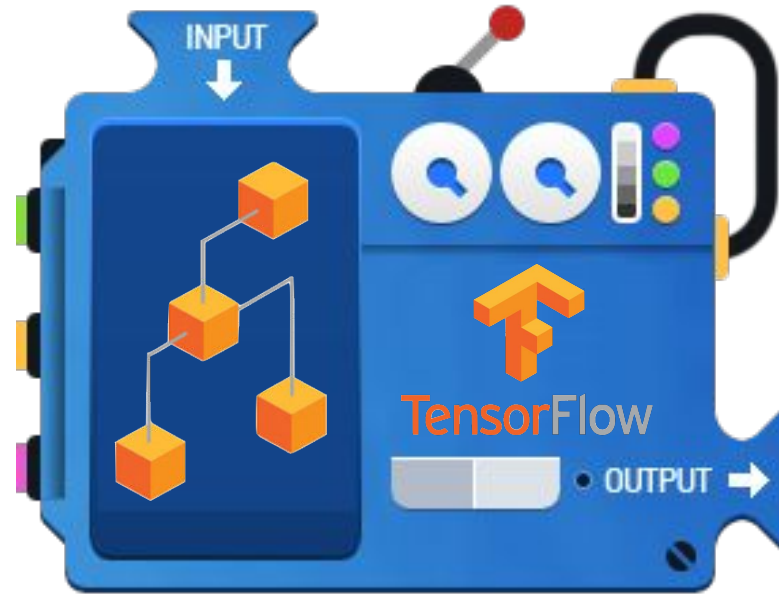
```
In [6]: sess = tf.Session()
print("sess.run(node1, node2): ", sess.run([node1, node2]))
print("sess.run(node3): ", sess.run(node3))
```

```
sess.run(node1, node2): [3.0, 4.0]
sess.run(node3): 7.0
```


TensorFlow Mechanics

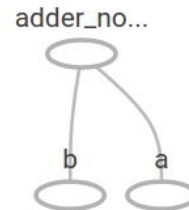
2 feed data and run graph (operation)
sess.run (op)

1 Build graph using
TensorFlow operations



3 update variables
in the graph
(and return values)

Computational Graph



(1) Build graph (tensors) using TensorFlow operations

```
In [4]: node1 = tf.constant(3.0, tf.float32)
node2 = tf.constant(4.0) # also tf.float32 implicitly
node3 = tf.add(node1, node2)
```

(2) feed data and run graph (operation)
sess.run (op)

(3) update variables in the graph
(and return values)

```
In [6]: sess = tf.Session()
print("sess.run(node1, node2): ", sess.run([node1, node2]))
print("sess.run(node3): ", sess.run(node3))

sess.run(node1, node2): [3.0, 4.0]
sess.run(node3): 7.0
```

Placeholder

```
In [7]: a = tf.placeholder(tf.float32)
b = tf.placeholder(tf.float32)
adder_node = a + b # + provides a shortcut for tf.add(a, b)

print(sess.run(adder_node, feed_dict={a: 3, b: 4.5}))
print(sess.run(adder_node, feed_dict={a: [1,3], b: [2, 4]}))
```

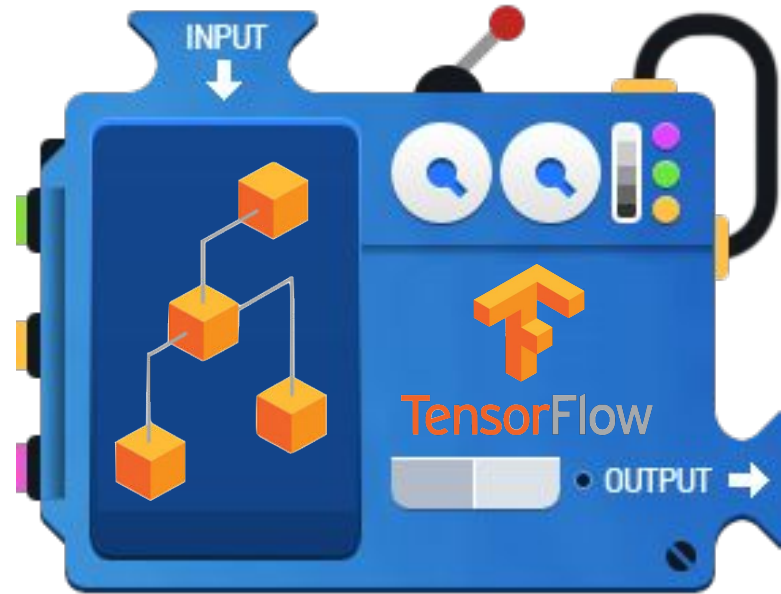
7.5

[3. 7.]

TensorFlow Mechanics

2 feed data and run graph (operation)
`sess.run (op, feed_dict={x: x_data})`

1 Build graph using
TensorFlow operations



3 update variables
in the graph
(and return values)

Everything is **Tensor**

Tensors

```
In [3]: 3 # a rank 0 tensor; this is a scalar with shape []  
        [1., 2., 3.] # a rank 1 tensor; this is a vector with shape [3]  
        [[1., 2., 3.], [4., 5., 6.]] # a rank 2 tensor; a matrix with shape [2, 3]  
        [[[1., 2., 3.]], [[7., 8., 9.]]] # a rank 3 tensor with shape [2, 1, 3]
```

```
Out[3]: [[[1.0, 2.0, 3.0]], [[7.0, 8.0, 9.0]]]
```

```
t = tf.Constant([1., 2., 3.])
```

Tensor Ranks, Shapes, and Types

```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Rank	Math entity	Python example
0	Scalar (magnitude only)	<code>s = 483</code>
1	Vector (magnitude and direction)	<code>v = [1.1, 2.2, 3.3]</code>
2	Matrix (table of numbers)	<code>m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]</code>
3	3-Tensor (cube of numbers)	<code>t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]]</code>
n	n-Tensor (you get the idea)	<code>....</code>

Tensor Ranks, Shapes, and Types

```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Rank	Shape	Dimension number	Example
0	[]	0-D	A 0-D tensor. A scalar.
1	[D0]	1-D	A 1-D tensor with shape [5].
2	[D0, D1]	2-D	A 2-D tensor with shape [3, 4].
3	[D0, D1, D2]	3-D	A 3-D tensor with shape [1, 4, 3].
n	[D0, D1, ... Dn-1]	n-D	A tensor with shape [D0, D1, ... Dn-1].

Tensor Ranks, Shapes, and Types

```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

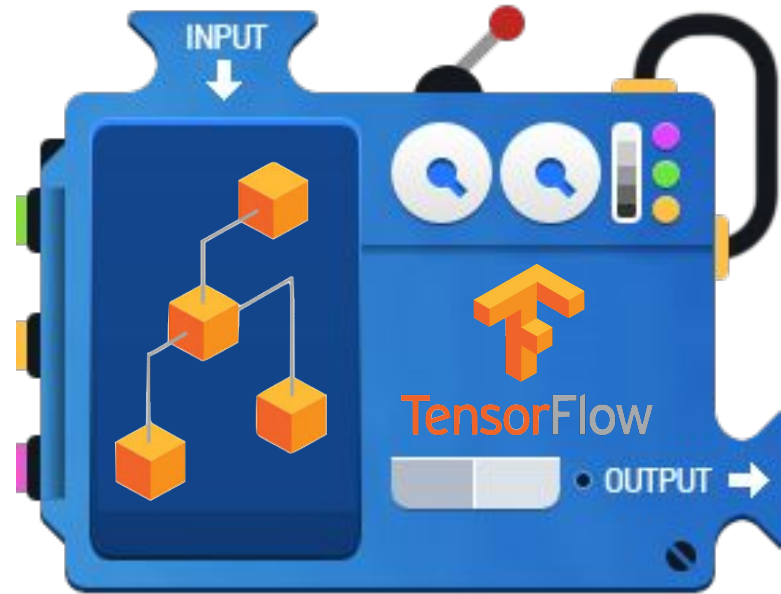
Data type	Python type	Description
DT_FLOAT	<code>tf.float32</code>	32 bits floating point.
DT_DOUBLE	<code>tf.float64</code>	64 bits floating point.
DT_INT8	<code>tf.int8</code>	8 bits signed integer.
DT_INT16	<code>tf.int16</code>	16 bits signed integer.
DT_INT32	<code>tf.int32</code>	32 bits signed integer.
DT_INT64	<code>tf.int64</code>	64 bits signed integer.

...

TensorFlow Mechanics

2 feed data and run graph (operation)
`sess.run (op, feed_dict={x: x_data})`

1 Build graph using
TensorFlow operations

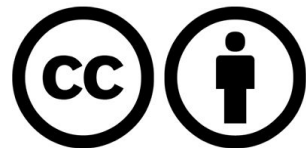


3 update variables
in the graph
(and return values)

Lab 2

Linear Regression

Sung Kim <hunkim+ml@gmail.com>





Variables

```
# Create two variables.
weights = tf.Variable(tf.random_normal([784, 200], stddev=0.35),
                      name="weights")
biases = tf.Variable(tf.zeros([200]), name="biases")
...
# Add an op to initialize the variables.
init_op = tf.global_variables_initializer()

# Later, when launching the model
with tf.Session() as sess:
    # Run the init operation.
    sess.run(init_op)
    ...
    # Use the model
    ...
```