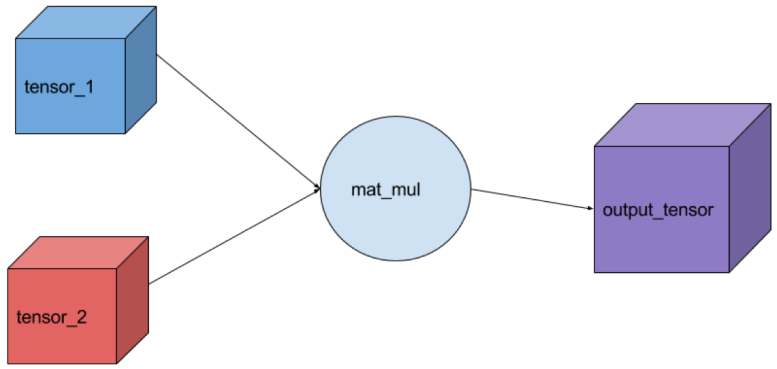
Tensorboard + Tensorflow Debugger Tutorial

Nicole Peinelt

9 August 2018

Tensorflow in a nutshell

- Computational graph: a network of nodes, with each node known as an operation



Code for tutorial

https://github.com/wuningxi/tf-tools-tutorial

```
starter_code.py
tensorboard_1.py
tensorboard_2.py
tensorboard_3.py
debugger.py
```

Overview

1. TF in a nutshell
2. Starter code
3. Tensorboard
    1. Visualising TF graph
    2. Logging summaries
4. Tensorflow debugger

Starter Code

- MNIST dataset
- Simple 2 layer network

```
[nicole:~/code/tf-tools]$ python starter_code.py
Extracting /tmp/tensorflow/mnist/input_data/train-images-idx3-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data/train-labels-idx1-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data/t10k-images-idx3-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data/t10k-labels-idx1-ubyte.gz
2018-08-07 17:23:51.714366: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU
 supports instructions that this TensorFlow binary was not compiled to use: SSE4.2 AVX AV
X2 FMA
Accuracy at step 0: [0.0788]
Accuracy at step 10: [0.6783]
Accuracy at step 20: [0.8064]
Accuracy at step 30: [0.8568]
Accuracy at step 40: [0.8794]
```

Using Tensorboard

- Create writers, save TF graph (next slide)

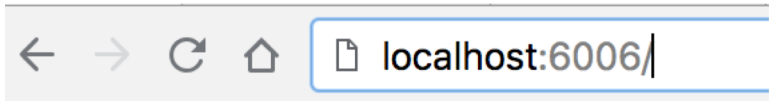
- Run script

```
python tensorboard_1.py
```

- Start Tensorboard

```
[nicole:~/code/tf-tools]$ cd tb_1
[nicole:~/code/tf-tools/tb_1]$ tensorboard --logdir .
TensorBoard 1.5.1 at http://NPMacBook:6006 (Press CTRL+C to quit)
```

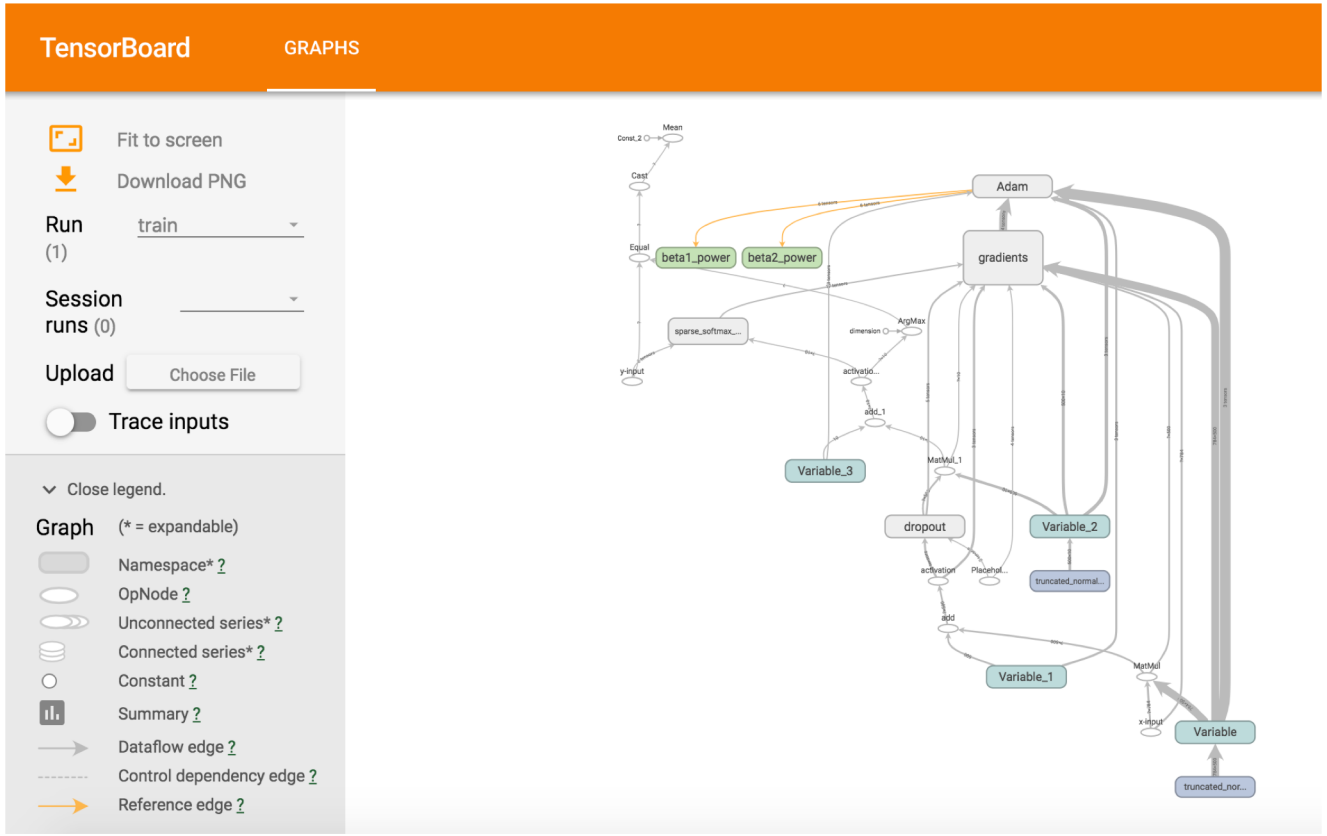
- Open browser



Visualising TF graph

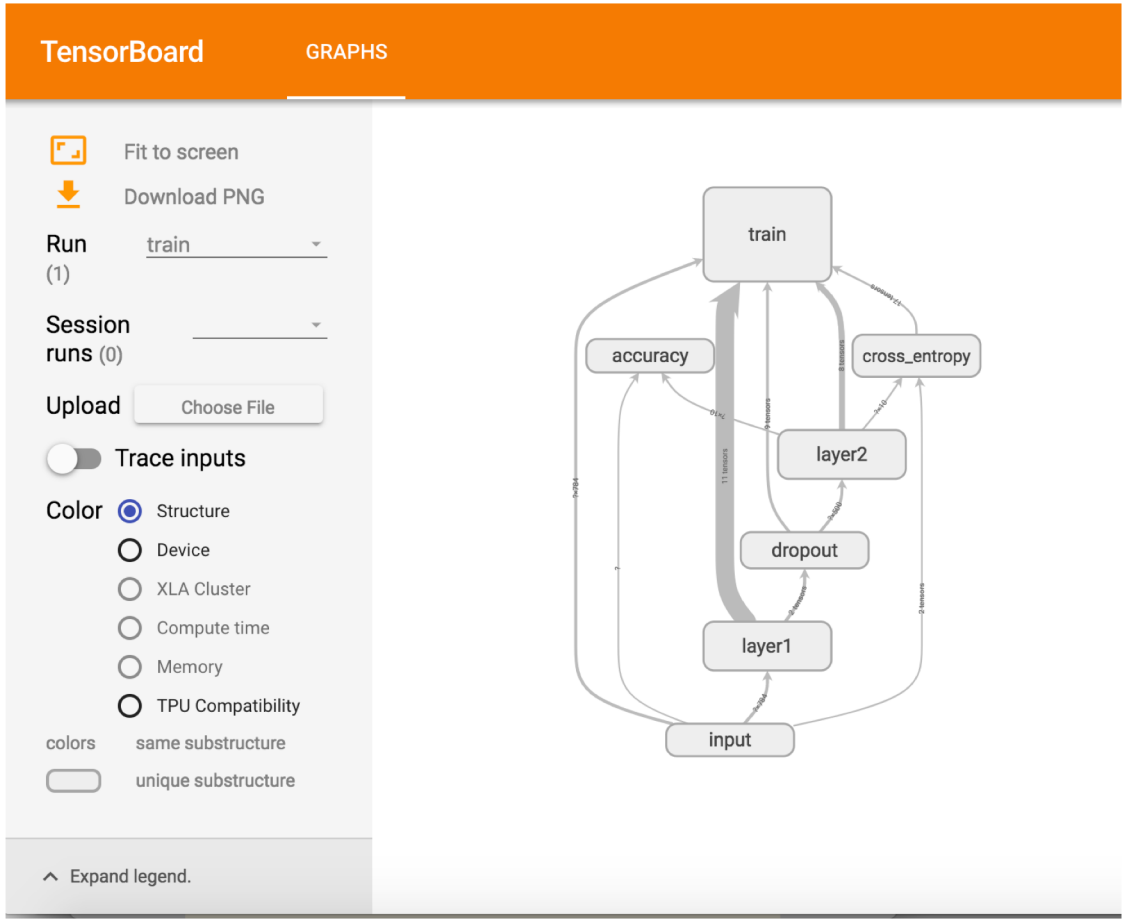
```
with tf.Session() as sess:

    # Merge all the summaries and write them out to
    # /tmp/tensorflow/mnist/logs/mnist_with_summaries (by default)
    train_writer = tf.summary.FileWriter(FLAGS.log_dir + '/train', sess.graph)
    test_writer = tf.summary.FileWriter(FLAGS.log_dir + '/test')
    tf.global_variables_initializer().run()

    # Train the model, and also write summaries.
    # Every 10th step, measure test-set accuracy, and write test summaries
    # All other steps, run train_step on training data, & add training summaries

    def feed_dict(train):
        """Make a TensorFlow feed_dict: maps data onto Tensor placeholders."""
        if train or FLAGS.fake_data:
            xs, ys = mnist.train.next_batch(100, fake_data=FLAGS.fake_data)
            k = FLAGS.dropout
        else:
            xs, ys = mnist.test.images, mnist.test.labels
            k = 1.0
        return {x: xs, y_: ys, keep_prob: k}

    for i in range(FLAGS.max_steps):
        if i % 10 == 0:  # Record summaries and test-set accuracy
            acc = sess.run([accuracy], feed_dict=feed_dict(False))
            print('Accuracy at step %s: %s' % (i, acc))
        else:  # Record train set summaries, and train
            _ = sess.run([train_step], feed_dict=feed_dict(True))
    train_writer.close()
    test_writer.close()

def main(_):
    if tf.gfile.Exists(FLAGS.log_dir):
        tf.gfile.DeleteRecursively(FLAGS.log_dir)
    tf.gfile.MakeDirs(FLAGS.log_dir)
    train()
```

tensorboard_1.py



Using name scopes

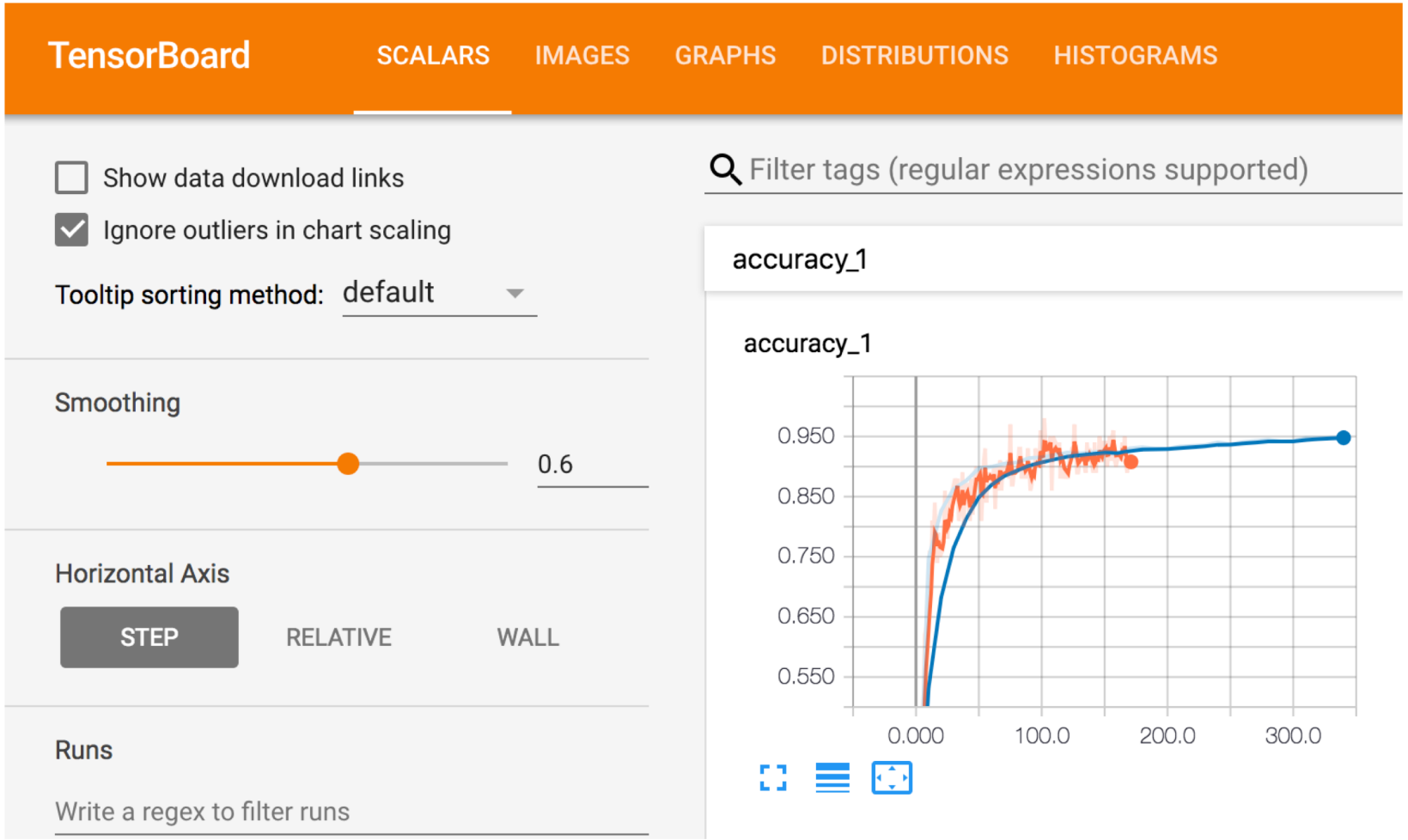
```python
# Input placeholders
with tf.name_scope('input'):
    x = tf.placeholder(tf.float32, [None, 784], name='x-input')
    y_ = tf.placeholder(tf.int64, [None], name='y-input')

# We can't initialize these variables to 0 - the network will get stuck.
def weight_variable(shape):
    """Create a weight variable with appropriate initialization."""
    initial = tf.truncated_normal(shape, stddev=0.1)
    return tf.Variable(initial)

def bias_variable(shape):
    """Create a bias variable with appropriate initialization."""
    initial = tf.constant(0.1, shape=shape)
    return tf.Variable(initial)

def nn_layer(input_tensor, input_dim, output_dim, layer_name, act=tf.nn.relu):
    """Reusable code for making a simple neural net layer.
It does a matrix multiply, bias add, and then uses ReLU to nonlinearize.
It also sets up name scoping so that the resultant graph is easy to read
"""
    # Adding a name scope ensures logical grouping of the layers in the graph.
    with tf.name_scope(layer_name):
        # This Variable will hold the state of the weights for the layer
        with tf.name_scope('weights'):
            weights = weight_variable([input_dim, output_dim])
        with tf.name_scope('biases'):
            biases = bias_variable([output_dim])
        with tf.name_scope('Wx_plus_b'):
            preactivate = tf.matmul(input_tensor, weights) + biases
        activations = act(preactivate, name='activation')
        return activations

hidden1 = nn_layer(x, 784, 500, 'layer1')

with tf.name_scope('dropout'):
    keep_prob = tf.placeholder(tf.float32)
    dropped = tf.nn.dropout(hidden1, keep_prob)
```



tensorboard_2.py

Using summaries

```python
with tf.name_scope('input_reshape'):
    image_shaped_input = tf.reshape(x, [-1, 28, 28, 1])
    tf.summary.image('input', image_shaped_input, 10)

# We can't initialize these variables to 0 - the network will get stuck.
def weight_variable(shape):
    """Create a weight variable with appropriate initialization."""
    initial = tf.truncated_normal(shape, stddev=0.1)
    return tf.Variable(initial)

def bias_variable(shape):
    """Create a bias variable with appropriate initialization."""
    initial = tf.constant(0.1, shape=shape)
    return tf.Variable(initial)

def variable_summaries(var):
    """Attach a lot of summaries to a Tensor (for TensorBoard visualization)."""
    with tf.name_scope('summaries'):
        mean = tf.reduce_mean(var)
        tf.summary.scalar('mean', mean)
        with tf.name_scope('stddev'):
            stddev = tf.sqrt(tf.reduce_mean(tf.square(var - mean)))
        tf.summary.scalar('stddev', stddev)
        tf.summary.scalar('max', tf.reduce_max(var))
        tf.summary.scalar('min', tf.reduce_min(var))
        tf.summary.histogram('histogram', var)

def nn_layer(input_tensor, input_dim, output_dim, layer_name, act=tf.nn.relu):
    """Reusable code for making a simple neural net layer.
It does a matrix multiply, bias add, and then uses ReLU to nonlinearize.
It also sets up name scoping so that the resultant graph is easy to read,
and adds a number of summary ops.
"""
    # Adding a name scope ensures logical grouping of the layers in the graph.
    with tf.name_scope(layer_name):
        # This Variable will hold the state of the weights for the layer
        with tf.name_scope('weights'):
            weights = weight_variable([input_dim, output_dim])
            variable_summaries(weights)
        with tf.name_scope('biases'):
            biases = bias_variable([output_dim])
            variable_summaries(biases)
        with tf.name_scope('Wx_plus_b'):
            preactivate = tf.matmul(input_tensor, weights) + biases
            tf.summary.histogram('pre_activations', preactivate)
        activations = act(preactivate, name='activation')
        tf.summary.histogram('activations', activations)
        return activations
```



tensorboard_3.py

Tensorflow Debugger

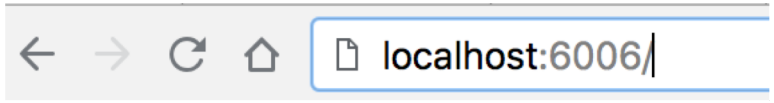
- Run script

```
[nicole:~/code/tf-tools]$ python debugger.py
```

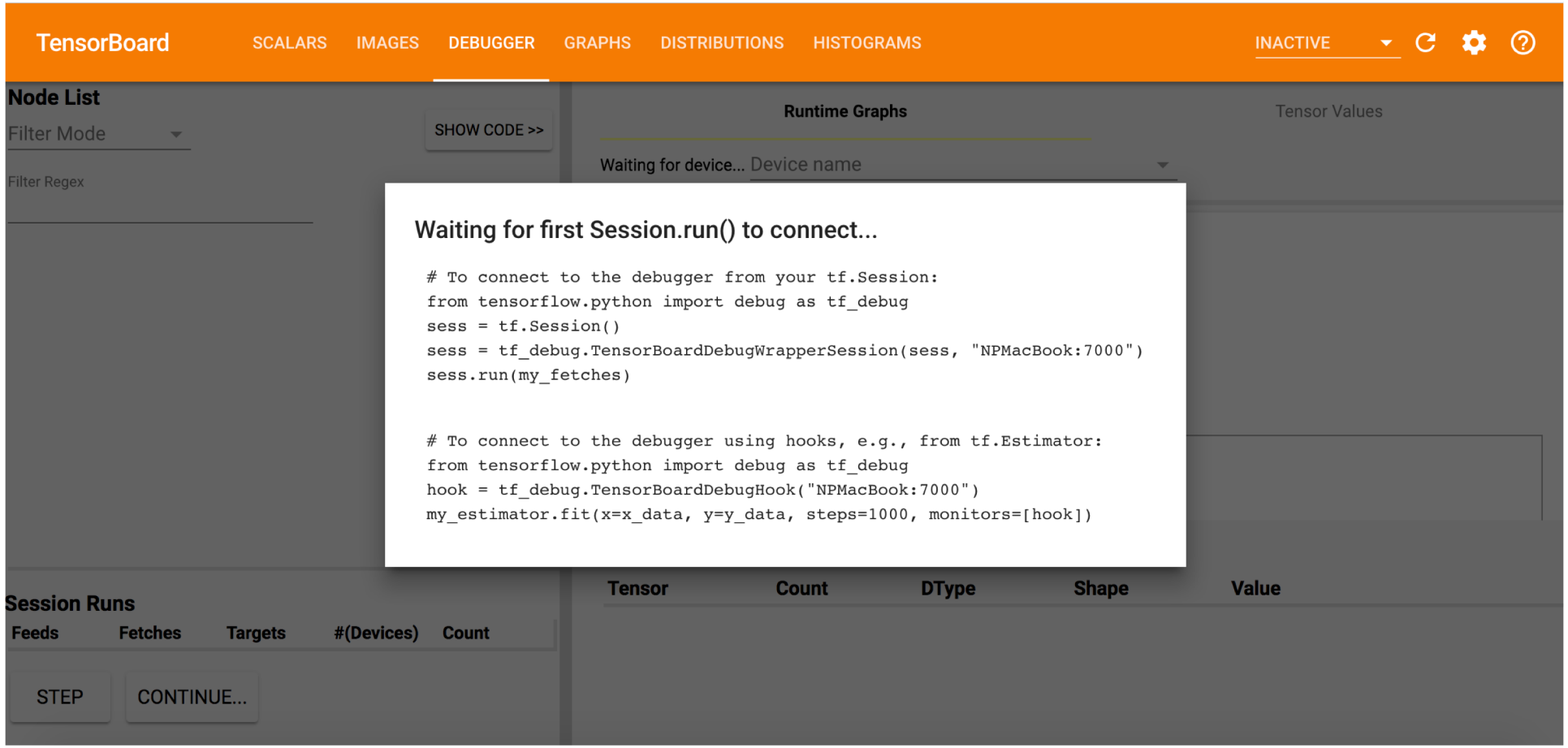
- Start Tensorboard with debugger_port

```
[nicole:~/code/tf-tools/tf_dbg]$ tensorboard --logdir . --debugger_port 7000
Creating InteractiveDebuggerPlugin at port 7000
TensorBoard 1.5.1 at http://NPMacBook:6006 (Press CTRL+C to quit)
```

- Open browser



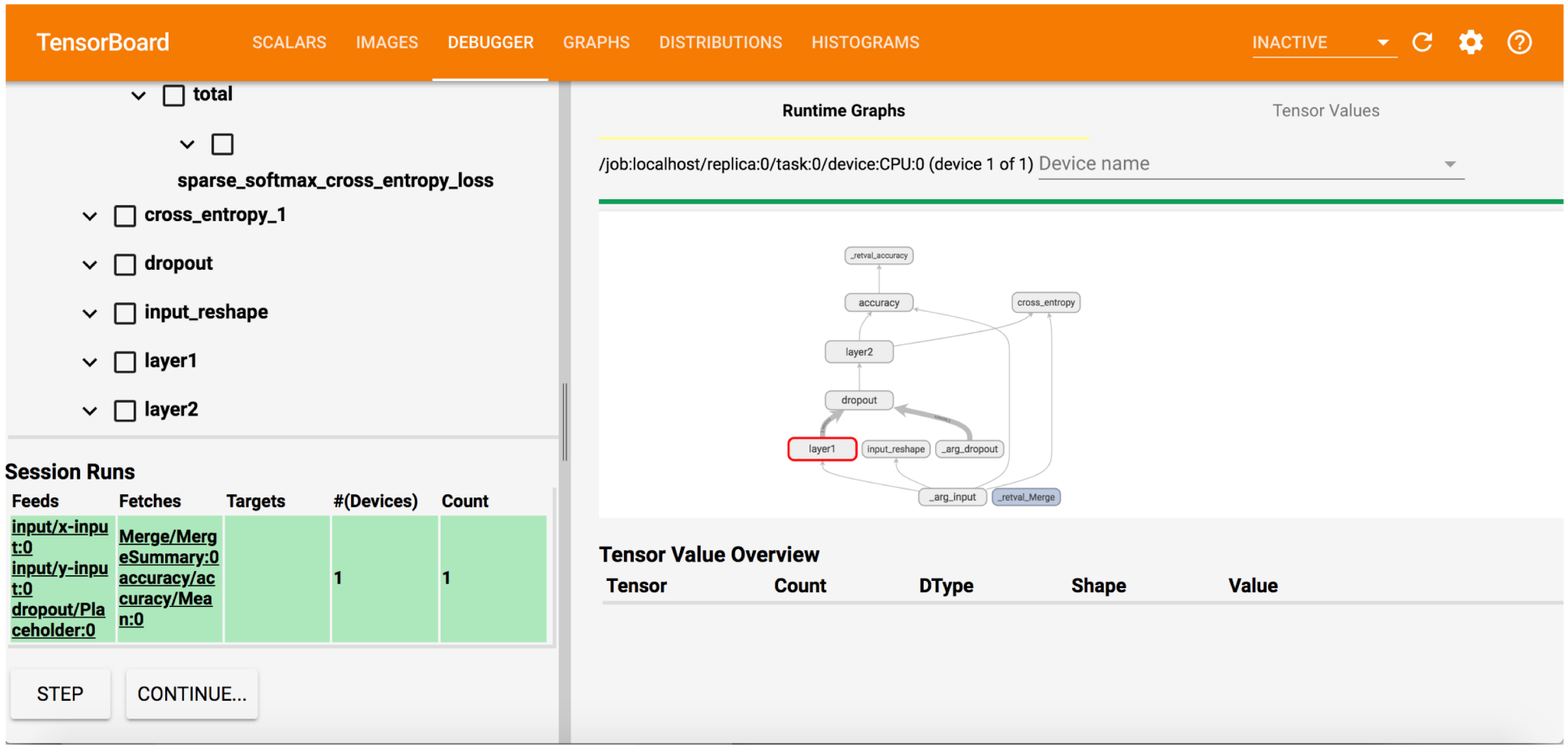
Tensorflow Debugger



Tensorflow Debugger

```
from tensorflow.python import debug as tf_debug

...

  # Execute graph

  with tf.Session() as sess:

      sess = tf_debug.TensorBoardDebugWrapperSession(sess, "NPMacBook:7000")

      # Merge all the summaries and write them out to
      # /tmp/tensorflow/mnist/logs/mnist_with_summaries (by default)
      merged = tf.summary.merge_all()
      train_writer = tf.summary.FileWriter(FLAGS.log_dir + '/train', sess.graph)
      test_writer = tf.summary.FileWriter(FLAGS.log_dir + '/test')
      tf.global_variables_initializer().run()

      # Train the model, and also write summaries.
      # Every 10th step, measure test-set accuracy, and write test summaries
      # All other steps, run train_step on training data, & add training summaries
```

Tensorflow Debugger



Recommended Resources

- https://www.tensorflow.org/guide/summaries_and_tensorboard
- https://www.youtube.com/watch?v=XcHWLsVmHvk&vl=en