TensorFlow Introduction:

Duke-Tsinghua Machine Learning Summer School 2017

KEVIN LIANG

DUKE UNIVERSITY – ELECTRICAL AND COMPUTER ENGINEERING

26 JULY 2017

TensorFlow: What is it?

- A software library for machine learning
  - Computation using data flow graphs
  - Neural Networks
- Released by Google November 9, 2015
- An open source successor to DistBelief
  - Apache 2.0 License
- APIs:
  - **Python**
  - C++
  - Java
  - Go

TensorFlow: Alternatives

- Caffe
  - UC Berkeley (BVLC: Berkeley Vision and Learning Center)
- Microsoft Cognitive Toolkit (CNTK 2.0)
  - Microsoft
- Theano
  - Université de Montréal (MILA/LISA: Montreal Institute for Learning Algorithms)
- Torch

TensorFlow: Why?

- CPU/GPU/TPU support, easy to scale up
- Large and active user-base
    - Academia, industry, enthusiasts
- Rapid development, research, and support by Google
- TensorBoard visualizations
- Integration with Google Cloud Platform
- Pre-trained models and high-level libraries (Slim, Keras, TFLearn)

TensorFlow: Tensors

- *Mathematics*: Geometric objects defining linear relations
  - Generalization of vectors and matrices:
    - 0th Order (Scalar): 8
    - 1st Order (Vector): [4, 2, 9]
    - 2nd Order (Matrix): [[5, 1, 9], [2, 2, 0]]

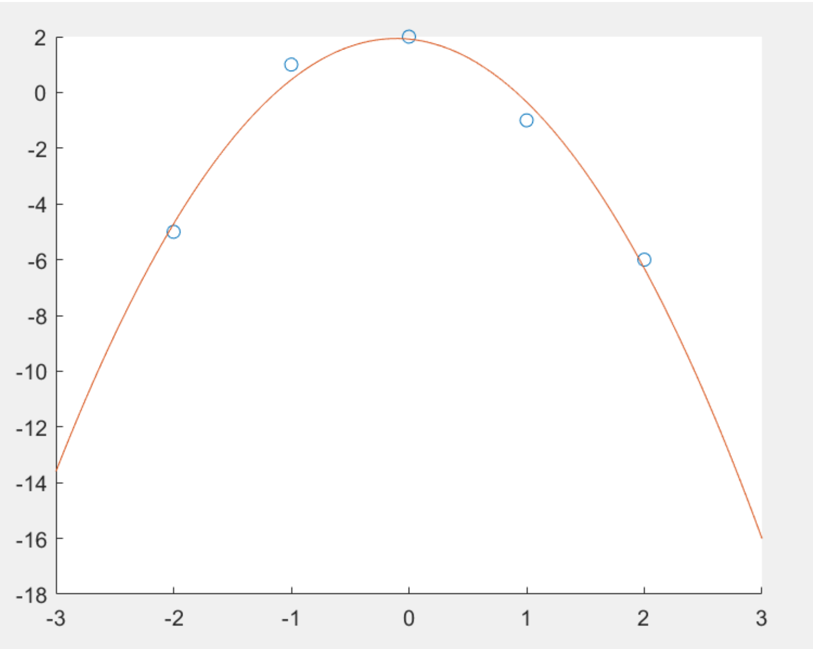
- TensorFlow: unit for data and variables
  - 0th Order: `scalar_node = tf.constant(8.0, dtype=tf.float32)`
  - 4th Order: `weights = tf.Variable(tf.random_normal([3, 3, 256, 512]), name=“conv_weights”)`

TensorFlow: Data Flow Graphs

- Typical computational program operates directly on the data:
  - Python:

```
import numpy
x = [-2, -1, 0, 1, 2]
y = [-5 , 1,  2,  -1,  -6]
p = numpy.polyfit(x, y, deg=2)

# y_hat = p[0] * x**2 + p[1] * x + p[2]
```

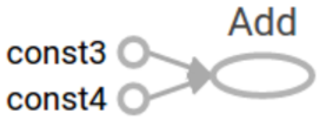


- Note that operations were performed on the variables holding the data itself

TensorFlow: Data Flow Graphs

- TensorFlow: 2 steps
  - Define a graph:

```
a = tf.constant(3.0, dtype=tf.float32)
b = tf.constant(4.0, dtype=tf.float32)
sum_a_b = tf.add(a, b)
```



  - Run the graph and get outputs:

```
sess = tf.Session()
print(sess.run(sum_a_b))      # Prints "7.0" to the screen
sess.close()
```

TensorFlow: Data Flow Graphs

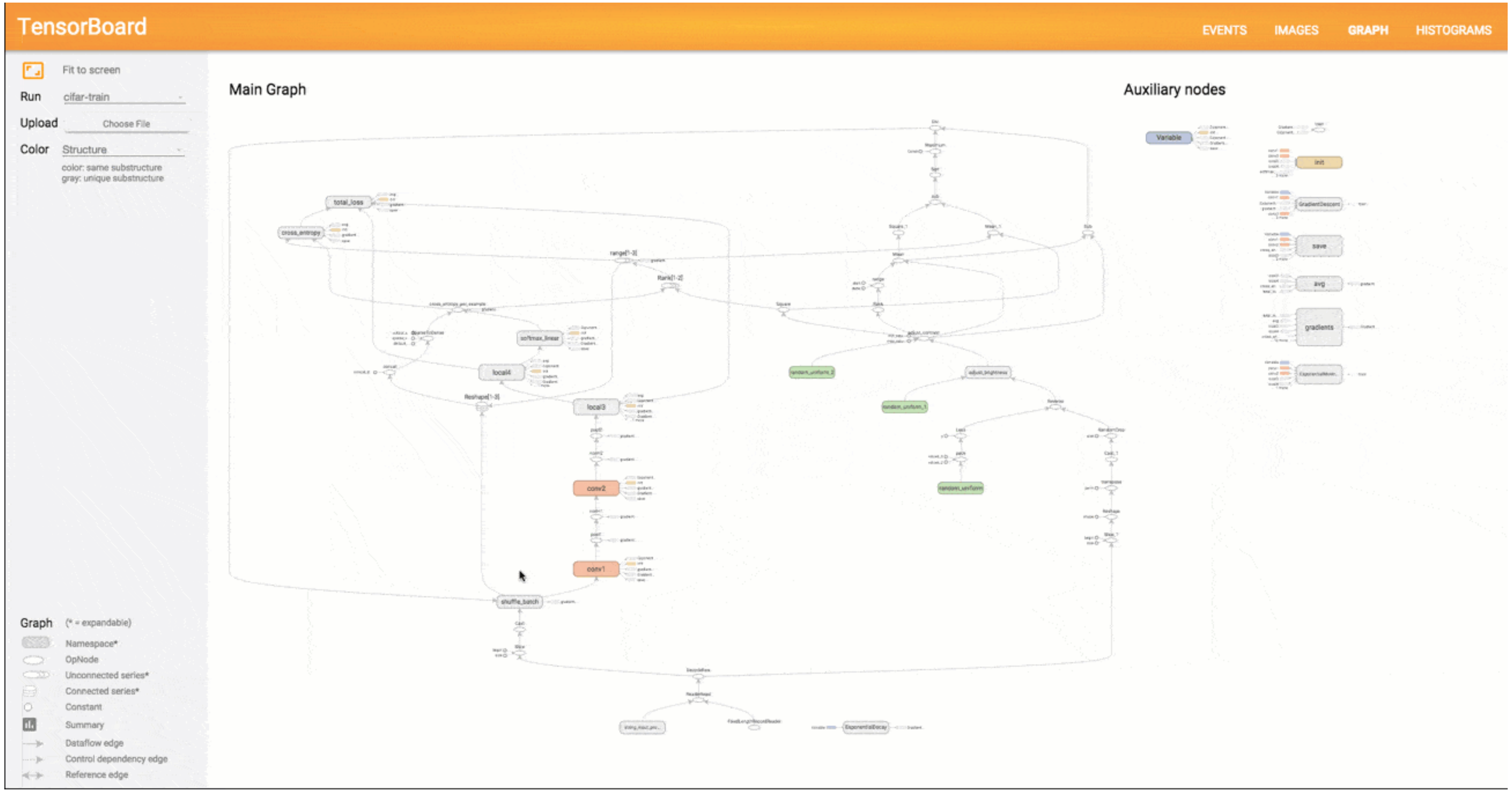
- Convolutional Neural Network

```
# Define graph
import tensorflow.contrib.slim as slim
x = tf.placeholder(tf.float32, [None,28,28,1])
net = slim.conv2d(x, 32, [3,3], scope=‘conv1_1’)
net = slim.conv2d(net, 32, [3,3], scope=‘conv1_2’)
net = slim.maxpool2d(net, [3,3], scope=‘pool1’)
net = slim.conv2d(net, 64, [3,3], scope=‘conv2_1’)
net = slim.conv2d(net, 64, [3,3], scope=‘conv2_2’)
net = slim.maxpool2d(net, [3,3], scope=‘pool2’)
logits = slim.fully_connected(net, 256, activation_fn=None, scope=‘fc’)

# Run graph
with tf.Session() as sess:
    feed_dict = {x = np.zeros([28,28])}                     # Feed in blank image
    print(sess.run(logits, feed_dict=feed_dict))            # Get some (garbage) prediction out
```

- More examples in tutorial: 02_TensorFlow_Basics.ipynb

TensorBoard: Graph Visualization



TensorBoard

Fit to screen

Run cifar-train

Upload Choose File

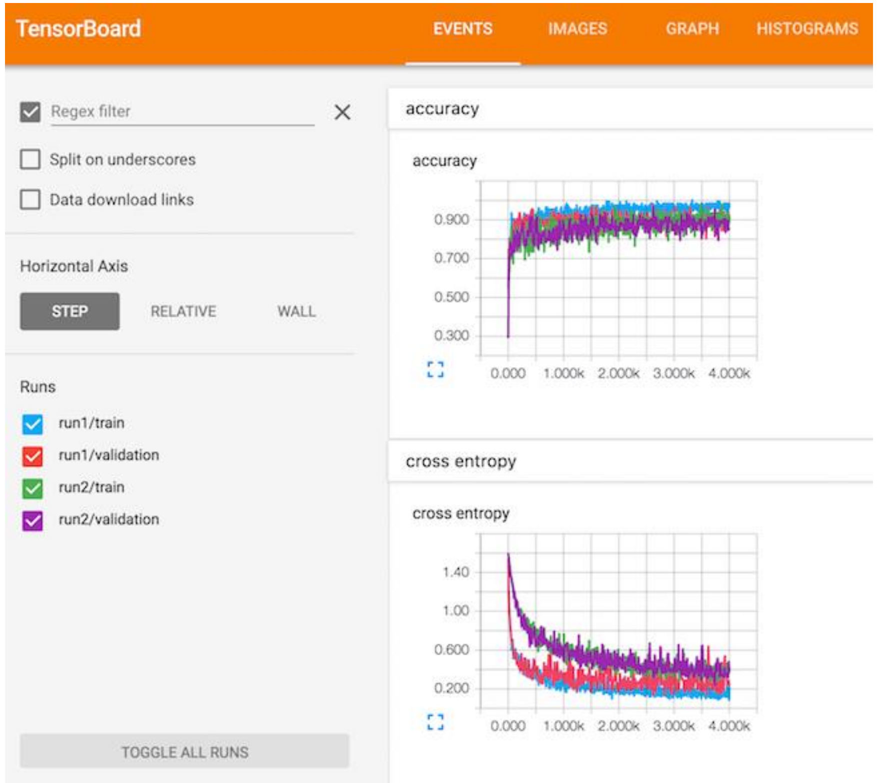
Color Structure

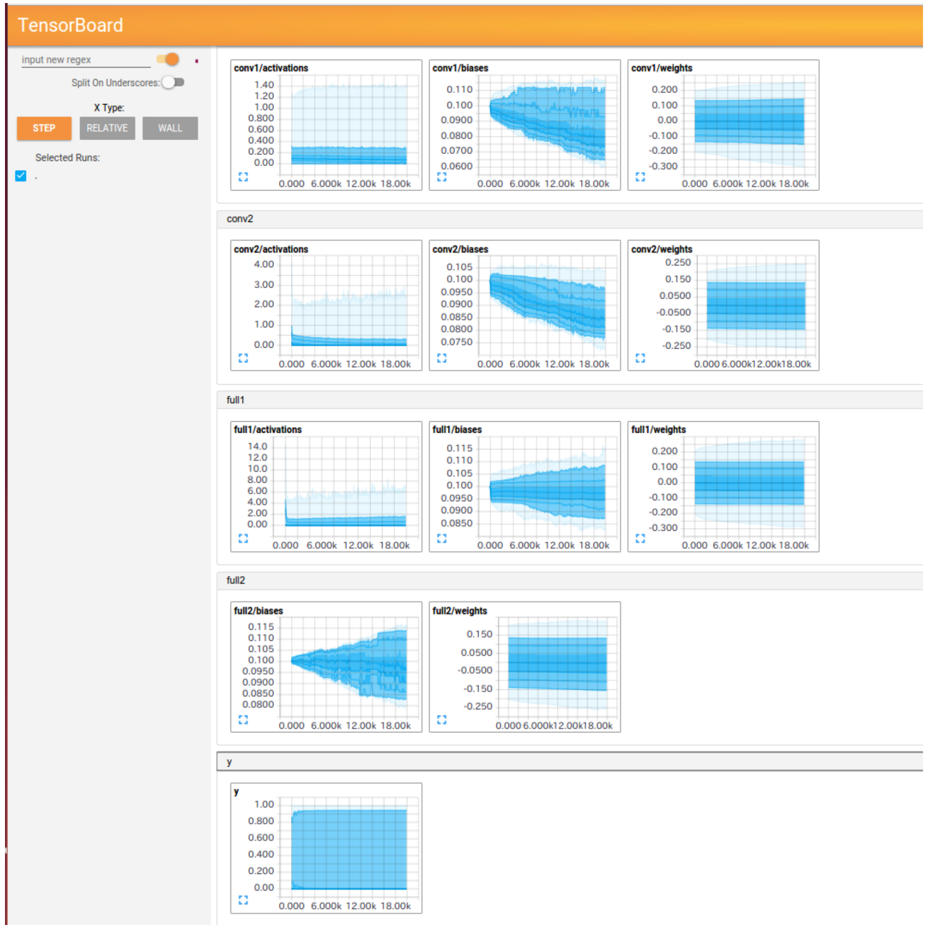
color: same substructure
gray: unique substructure

Graph (* = expandable)

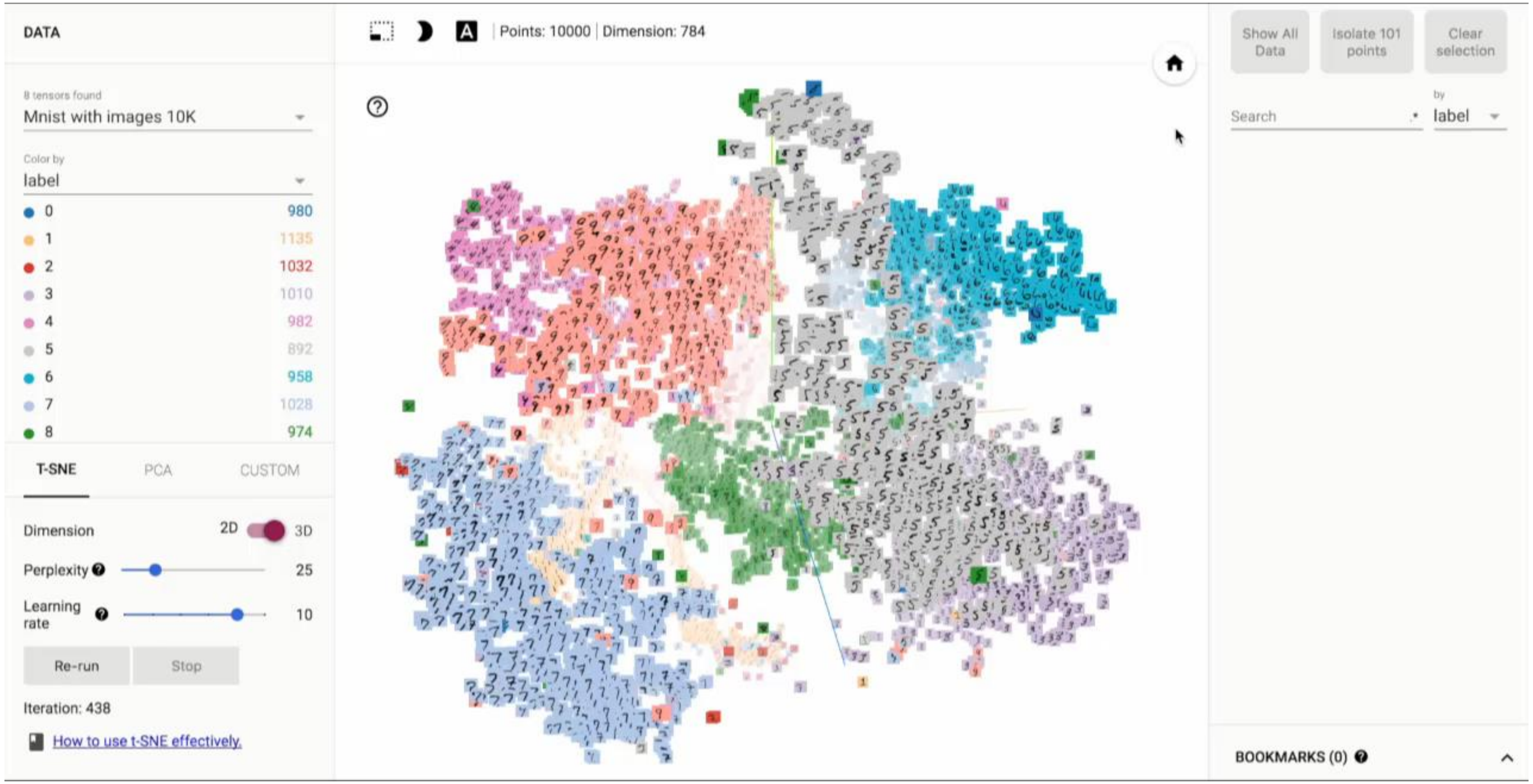
Namespace*
OpNode
Unconnected series*
Connected series*
Constant
Summary
Dataflow edge
Control dependency edge
Reference edge

TensorBoard: Learning Visualization





TensorFlow: Embedding Visualization



DATA

8 tensors found

Mnist with images 10K

Color by

label

●	0	980
●	1	1135
●	2	1032
●	3	1010
●	4	982
●	5	892
●	6	958
●	7	1028
●	8	974

T-SNE PCA CUSTOM

Dimension 2D 3D

Perplexity 25

Learning rate 10

Re-run Stop

Iteration: 438

How to use t-SNE effectively.

Points: 10000 | Dimension: 784

Show All Data

Isolate 101 points

Clear selection

Search

by label

BOOKMARKS (0)

Questions?