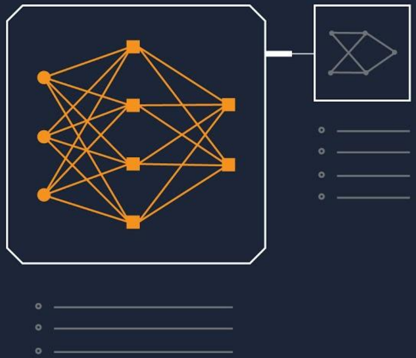
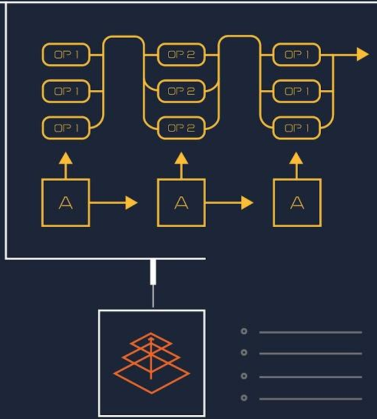
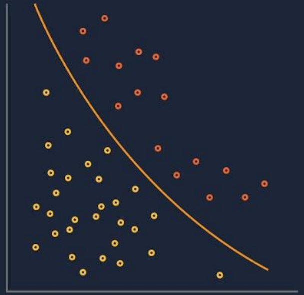






TensorFlow

DEV SUMMIT 2017

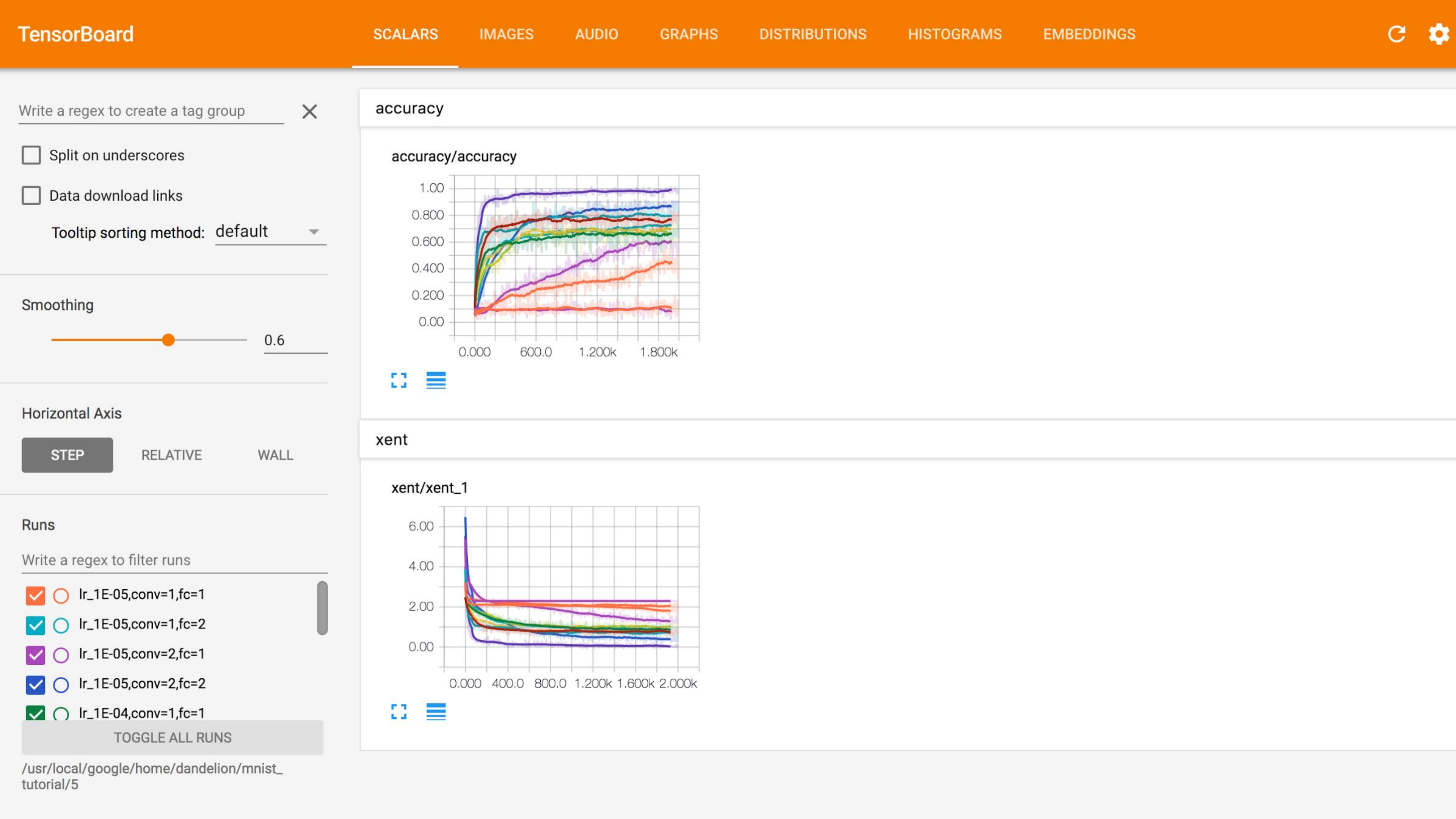
TensorBoard

Dandelion Mané
dandelion@google.com

Neural Nets can be a black box

TensorBoard is a flashlight

(but what **is** it)



☐ Split on underscores

☐ Data download links

Tooltip sorting method: default

Smoothing: 0.6

Horizontal Axis: [✓ STEP] RELATIVE WALL

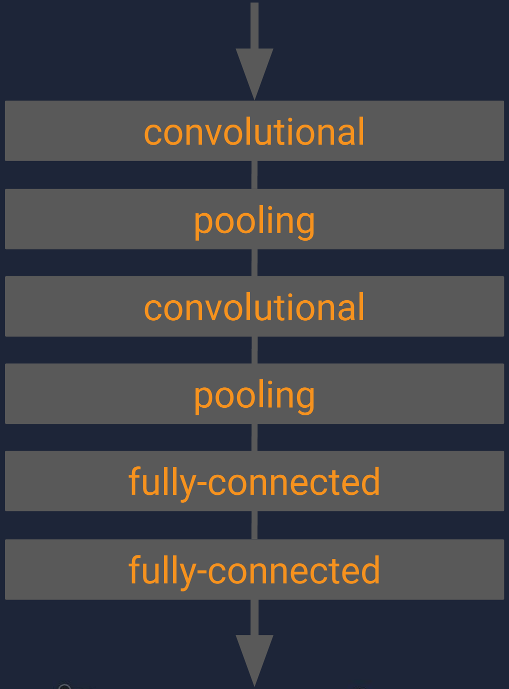
Runs

- ✓ lr_1E-05,conv=1,fc=1
- ✓ lr_1E-05,conv=1,fc=2
- ✓ lr_1E-05,conv=2,fc=1
- ✓ lr_1E-05,conv=2,fc=2
- ✓ lr_1E-04,conv=1,fc=1

/usr/local/google/home/dandelion/mnist_tutorial/5

Let's see an example

MNIST Architecture



Layer Definition

```
# Define a simple convolutional layer
def conv_layer(input, channels_in, channels_out):
    w = tf.Variable(tf.zeros([5, 5, channels_in, channels_out]))
    b = tf.Variable(tf.zeros([channels_out]))
    conv = tf.nn.conv2d(input, w, strides=[1, 1, 1, 1], padding="SAME")
    act = tf.nn.relu(conv + b)
    return act

# And a fully connected layer
def fc_layer(input, channels_in, channels_out):
    w = tf.Variable(tf.zeros([channels_in, channels_out]))
    b = tf.Variable(tf.zeros([channels_out]))
    act = tf.nn.relu(tf.matmul(input, w) + b)
    return act
```

Feed-forward Setup

```
# Setup placeholders, and reshape the data
x = tf.placeholder(tf.float32, shape=[None, 784])
y = tf.placeholder(tf.float32, shape=[None, 10])
x_image = tf.reshape(x, [-1, 28, 28, 1])

# Create the network
conv1 = conv_layer(x_image, 1, 32)
pool1 = tf.nn.max_pool(conv1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding="SAME")

conv2 = conv_layer(pooled, 32, 64)
pool2 = tf.nn.max_pool(conv2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding="SAME")
flattened = tf.reshape(pool2, [-1, 7 * 7 * 64])

fc1 = fc_layer(flattened, 7 * 7 * 64, 1024)
logits = fc_layer(fc1, 1024, 10)
```

Loss & Training

```
# Compute cross entropy as our loss function
cross_entropy = tf.reduce_mean(
      tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=y))

# Use an AdamOptimizer to train the network
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)

# compute the accuracy
correct_prediction = tf.equal(tf.argmax(logits, 1), tf.argmax(y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

Train the Model

```
# Initialize all the variables
sess.run(tf.global_variables_initializer())

# Train for 2000 steps
for i in range(2000):
  batch = mnist.train.next_batch(100)

    # Occasionally report accuracy
    if i % 500 == 0:
      [train_accuracy] = sess.run([accuracy], feed_dict={x: batch[0], y: batch[1]})
      print("step %d, training accuracy %g" % (i, train_accuracy))

    # Run the training step
    sess.run(train_step, feed_dict={x: batch[0], y_true: batch[1]})
```

We run our model and...

```
step 0,     training accuracy 10%
step 500,   training accuracy 12%
step 1500,  training accuracy  9%
step 2000,  training accuracy 13%
```

it didn't learn anything!

Let's turn on our flashlight!

Let's visualize the TensorFlow graph.

Visualizing the Graph

```
tf.summary.FileWriter (n):
    a Python class that writes data for TensorBoard
```

```
writer = tf.summary.FileWriter("/tmp/mnist_demo/1")
writer.add_graph(sess.graph)
```

```
>> tensorboard --logdir /tmp/mnist_demo/1
```

(DEMO)

Cleaning the Graph

- Node Names
- Name Scopes

Cleaning The Graph

```
def conv_layer(input, channels_in, channels_out, name="conv"):
  with tf.name_scope(name):
    w = tf.Variable(tf.zeros([5, 5, channels_in, channels_out]), name="W")
    b = tf.Variable(tf.zeros([channels_out]), name="B")
    conv = tf.nn.conv2d(input, w, strides=[1, 1, 1, 1], padding="SAME")
    act = tf.nn.relu(conv + b)
    return tf.nn.max_pool(act, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding="SAME")

def fc_layer(input, channels_in, channels_out, name="fc"):
  with tf.name_scope(name):
    w = tf.Variable(tf.zeros([channels_in, channels_out]), name="W")
    b = tf.Variable(tf.zeros([channels_out]), name="B")
    return tf.nn.relu(tf.matmul(input, w) + b)
```

Cleaning The Graph

```
# Setup placeholders, and reshape the data
x = tf.placeholder(tf.float32, shape=[None, 784], name="x")
x_image = tf.reshape(x, [-1, 28, 28, 1])
y = tf.placeholder(tf.float32, shape=[None, 10], name="labels")

conv1 = conv_layer(x_image, 1, 32, "conv1")
conv2 = conv_layer(conv1, 32, 64, "conv2")

flattened = tf.reshape(conv2, [-1, 7 * 7 * 64])
fc1 = fc_layer(flattened, 7 * 7 * 64, 1024, "fc1")
logits = fc_layer(fc1, 1024, 10, "fc2")
```

Cleaning The Graph

```
with tf.name_scope("xent"):
  xent = tf.reduce_mean(
      tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=y))

with tf.name_scope("train"):
  train_step = tf.train.AdamOptimizer(1e-4).minimize(xent)

with tf.name_scope("accuracy"):
  correct_prediction = tf.equal(tf.argmax(logits, 1), tf.argmax(y, 1))
  accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

Cleaning the Graph

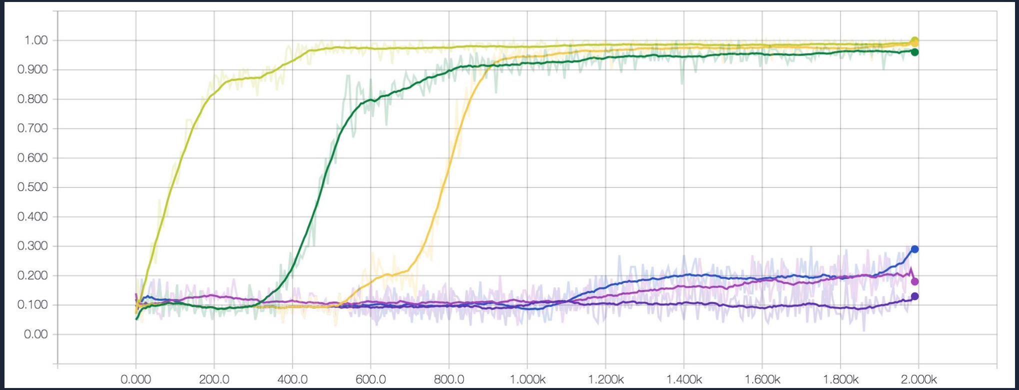
```
writer = tf.summary.FileWriter("/tmp/mnist_demo/2")
writer.add_graph(sess.graph)
```

```
>> tensorboard --logdir /tmp/mnist_demo/2
```

(DEMO)

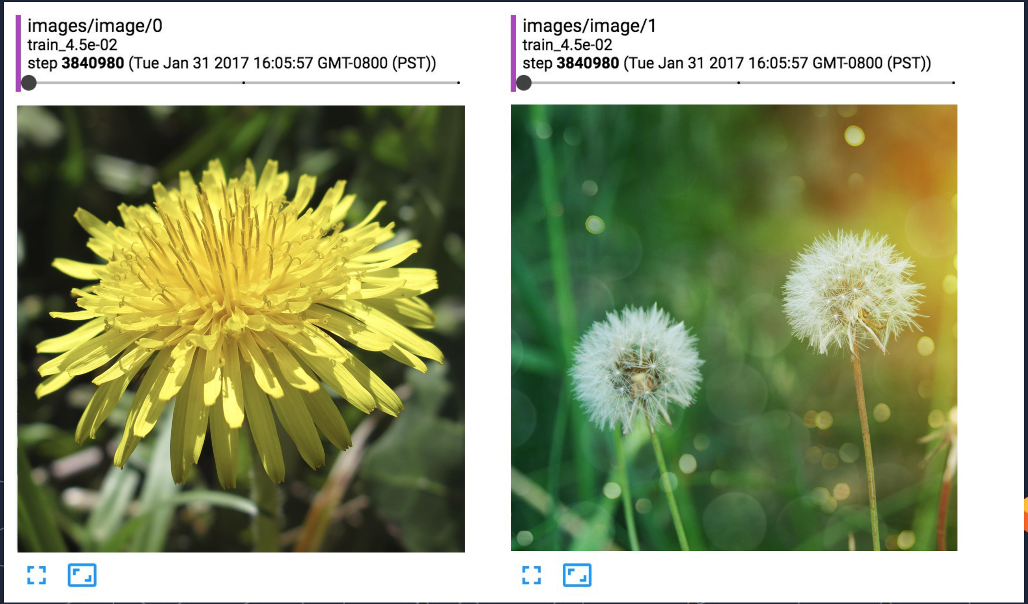
Collect some summaries

```
summary (n):
    a TensorFlow op that output protocol buffers containing
"summarized" data

Examples:
 -  tf.summary.scalar
```



Collect some summaries

```
summary (n):
    a TensorFlow op that output protocol buffers containing
"summarized" data

Examples:
 -  tf.summary.scalar
 -  tf.summary.image
```


images/image/0
train_4.5e-02
step **3840980** (Tue Jan 31 2017 16:05:57 GMT-0800 (PST))

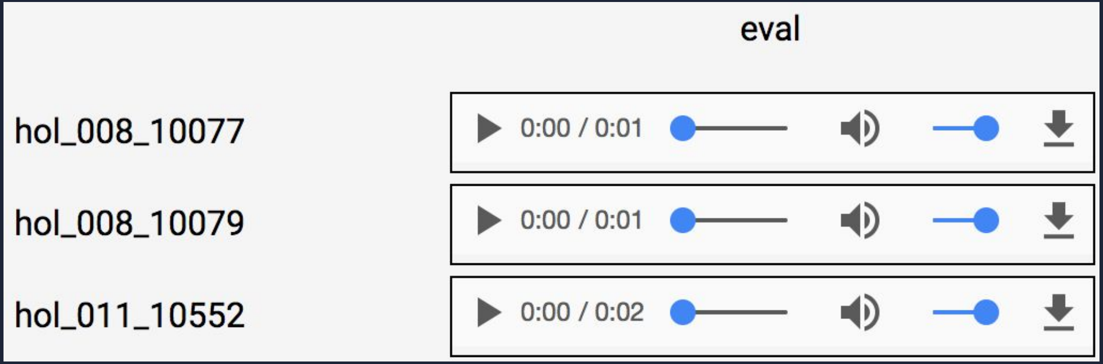
images/image/1
train_4.5e-02
step **3840980** (Tue Jan 31 2017 16:05:57 GMT-0800 (PST))

Collect some summaries

```
summary (n):
    a TensorFlow op that output protocol buffers containing
"summarized" data
```

Examples:

- `tf.summary.scalar`
- `tf.summary.image`
- **`tf.summary.audio`**

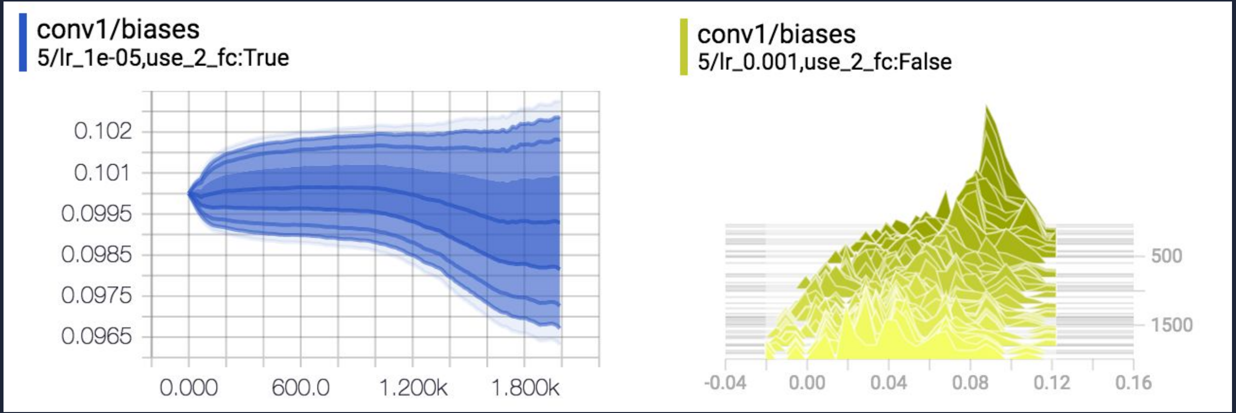


eval

hol_008_10077	0:00 / 0:01
hol_008_10079	0:00 / 0:01
hol_011_10552	0:00 / 0:02

Collect some summaries

```
summary (n):
    a TensorFlow op that output protocol buffers containing
"summarized" data

Examples:
 -   tf.summary.scalar
 -   tf.summary.image
 -   tf.summary.audio
```
 -   tf.summary.histogram



Collect some summaries

```
summary (n):
    a TensorFlow op that output protocol buffers containing
"summarized" data

Examples:
 -  tf.summary.scalar
 -  tf.summary.image
 -  tf.summary.audio
 -  tf.summary.histogram
 -  tf.summary.tensor
```

Collect some summaries

```
summary (n):
    a TensorFlow op that output protocol buffers containing
"summarized" data

Examples:
 -  tf.summary.scalar
 -  tf.summary.image
 -  tf.summary.audio
 -  tf.summary.histogram
 -  tf.summary.tensor (under development)
```

Collect some summaries

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

Collect some summaries

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

tf.summary.image('input', x_image, 3)
```

Collect some summaries

```
def conv_layer(input, channels_in, channels_out, name="conv"):
   with tf.name_scope(name):
     w = tf.Variable(tf.zeros([5, 5, channels_in, channels_out]), name="W")
     b = tf.Variable(tf.zeros([channels_out]), name="B")
     conv = tf.nn.conv2d(input, w, strides=[1, 1, 1, 1], padding="SAME")
     act = tf.nn.relu(conv + b)
     tf.summary.histogram("weights", w)
     tf.summary.histogram("biases", b)
     tf.summary.histogram("activations", act)
     return tf.nn.max_pool(act, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding="SAME")
```

How Summaries Work

- Summary Op returns protocol buffers
- tf.summary.FileWriter writes them to disk

Collect some summaries

```
merged_summary = tf.summary.merge_all()
```

Collect some summaries

```
merged_summary = tf.summary.merge_all()
writer = tf.summary.FileWriter("/tmp/mnist_demo/3")
writer.add_graph(sess.graph)
```

Collect some summaries

```
merged_summary = tf.summary.merge_all()
writer = tf.summary.FileWriter("/tmp/mnist_demo/3")
writer.add_graph(sess.graph)

for i in range(2001):
  batch = mnist.train.next_batch(100)
  if i % 5 == 0:
    s = sess.run(merged_summary, feed_dict={x: batch[0], y: batch[1]})
    writer.add_summary(s, i)
  sess.run(train_step, feed_dict={x: batch[0], y: batch[1]})
```

Collect some summaries

```
merged_summary = tf.summary.merge_all()
writer = tf.summary.FileWriter("/tmp/mnist_demo/3")
writer.add_graph(sess.graph)

for i in range(2001):
  batch = mnist.train.next_batch(100)
  if i % 5 == 0:
    s = sess.run(merged_summary, feed_dict={x: batch[0], y: batch[1]})
    writer.add_summary(s, i)
  sess.run(train_step, feed_dict={x: batch[0], y: batch[1]})
```

```
>> tensorboard --logdir /tmp/mnist_demo/3
```

(DEMO)

Fixing Our Model

```
def conv_layer(input, size_in, size_out, name="conv"):
  with tf.name_scope(name):
    w = tf.Variable(tf.zeros([5, 5, size_in, size_out]), name="W")
    b = tf.Variable(tf.zeros([size_out]), name="B")
    conv = tf.nn.conv2d(input, w, strides=[1, 1, 1, 1], padding="SAME")
    act = tf.nn.relu(conv + b)
    tf.summary.histogram("weights", w)
    tf.summary.histogram("biases", b)
    tf.summary.histogram("activations", act)
    return tf.nn.max_pool(act, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding="SAME")
```

Fixing Our Model

```
def conv_layer(input, size_in, size_out, name="conv"):
  with tf.name_scope(name):
    w = tf.Variable(tf.truncated_normal([5, 5, size_in, size_out], stddev=0.1), name="W")
    b = tf.Variable(tf.constant(0.1, shape=[size_out]), name="B")
    conv = tf.nn.conv2d(input, w, strides=[1, 1, 1, 1], padding="SAME")
    act = tf.nn.relu(conv + b)
    tf.summary.histogram("weights", w)
    tf.summary.histogram("biases", b)
    tf.summary.histogram("activations", act)
    return tf.nn.max_pool(act, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding="SAME")
```

(DEMO)

Hyperparameter Search

- What about different learning rates?
- What about different model architectures?

Hyperparameter Search

```
# Try a few learning rates
for learning_rate in [1E-3, 1E-4, 1E-5]:

  # Try a model with fewer layers
  for use_two_fc in [True, False]:
    for use_two_conv in [True, False]:

       # Construct a hyperparameter string for each one (example: "lr_1E-3,fc=2,conv=2)
       hparam_str = make_hparam_string(learning_rate, use_two_fc, use_two_conv)

       writer = tf.summary.FileWriter("/tmp/mnist_tutorial/" + hparam_str)

       # Actually run with the new settings
       mnist(learning_rate, use_two_fully_connected_layers, use_two_conv_layers, writer)
```

```
>> tensorboard --logdir /tmp/mnist_tutorial
```

(DEMO)

Embedding Visualization

```
embedding = tf.Variable(tf.zeros([10000, embedding_size]), name="test_embedding")
assignment = embedding.assign(embedding_input)

config = tf.contrib.tensorboard.plugins.projector.ProjectorConfig()
embedding_config = config.embeddings.add()
embedding_config.tensor_name = embedding.name
embedding_config.sprite.image_path = os.path.join(LOG_DIR, 'sprite.png')
# Specify the width and height of a single thumbnail.
embedding_config.sprite.single_image_dim.extend([28, 28])
tf.contrib.tensorboard.plugins.projector.visualize_embeddings(writer, config)
```

Embedding Visualization

```
for i in range(2001):
  batch = mnist.train.next_batch(100)
  if i % 5 == 0:
    [train_accuracy, s] = sess.run([accuracy, summ], feed_dict={x: batch[0], y: batch[1]})
    writer.add_summary(s, i)
  if i % 500 == 0:
    sess.run(assignment, feed_dict={x: mnist.test.images, y_true: mnist.test.labels})
    saver.save(sess, os.path.join(LOG_DIR, "model.ckpt"), i)
  sess.run(train_step, feed_dict={x: batch[0], y_true: batch[1]})
```

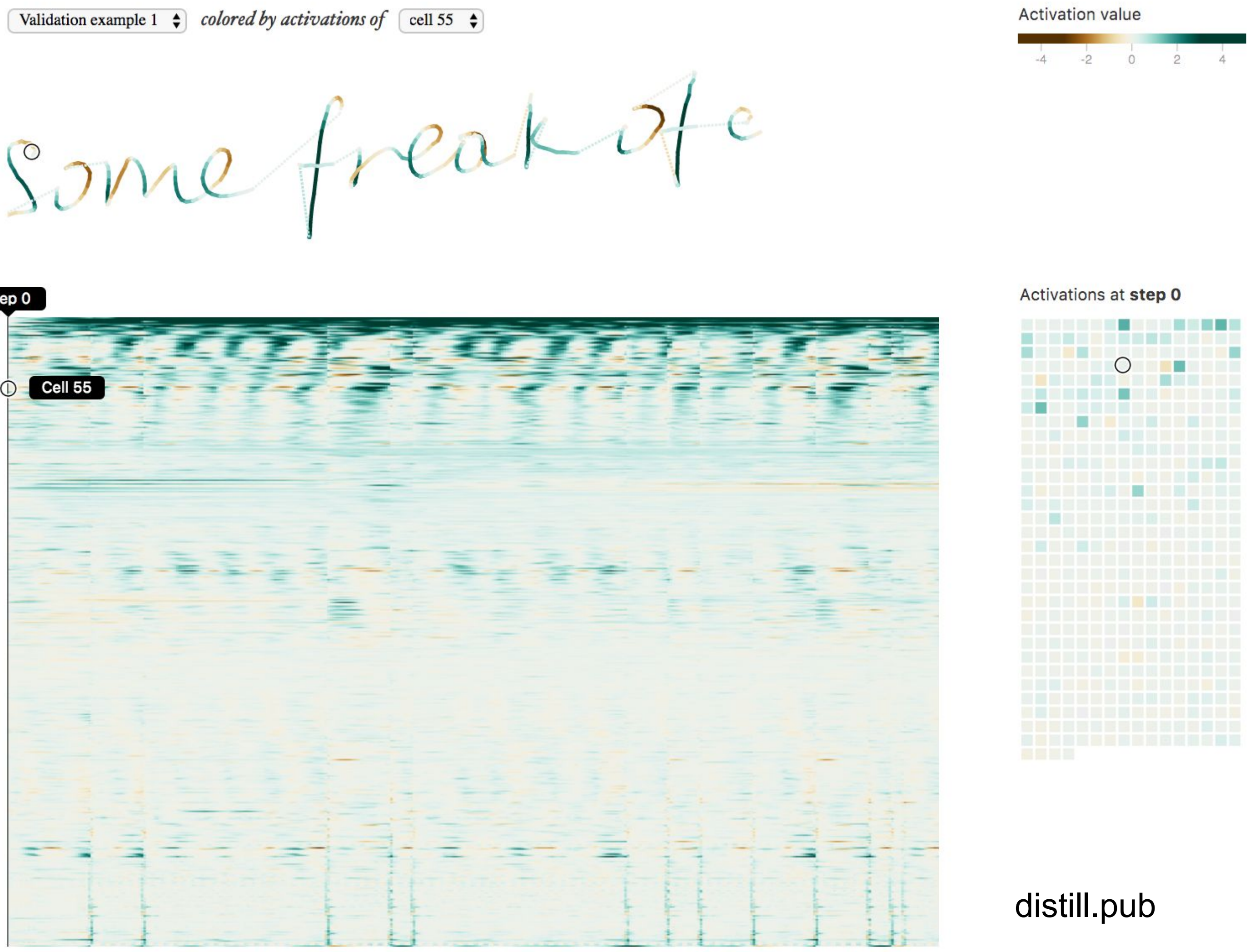
```
>> tensorboard --logdir /tmp/mnist_tutorial
```

(DEMO)

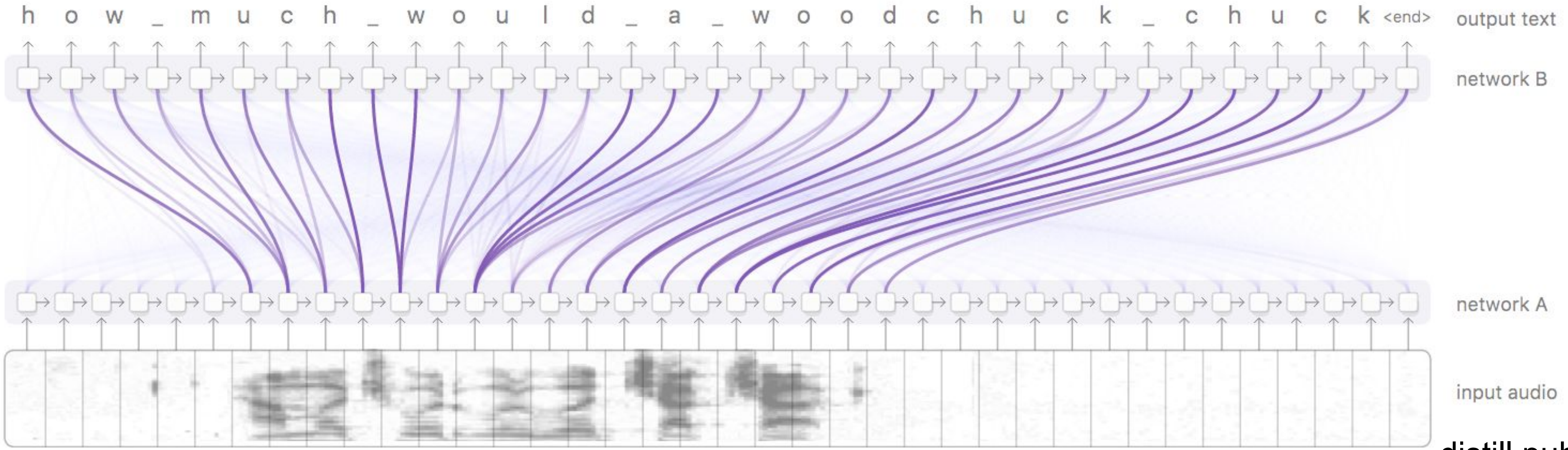
Future for TensorBoard

Future for TensorBoard

- TensorFlow Debugger Integration
- Plugins


Validation example 1
colored by activations of
cell 55
Activation value
-4
-2
0
2
4
Step 0
Cell 55
Activations at step 0

distill.pub

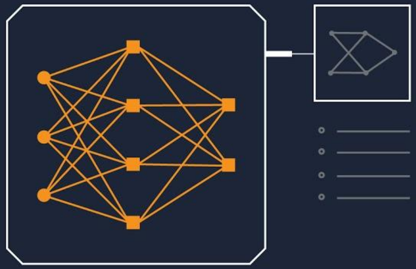


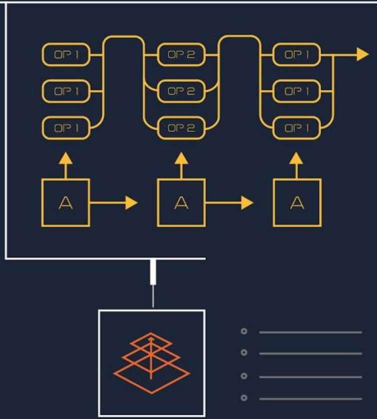
h o w _ m u c h _ w o u l d _ a _ w o o d c h u c k _ c h u c k <end>
output text
network B
network A
input audio

distill.pub

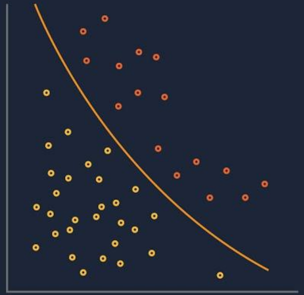
Future for TensorBoard

- TensorFlow Debugger Integration
- Plugins
- Org-scale TensorBoard





Code: https://goo.gl/San2uR



Slides: https://goo.gl/4IfwZy