TensorBoard

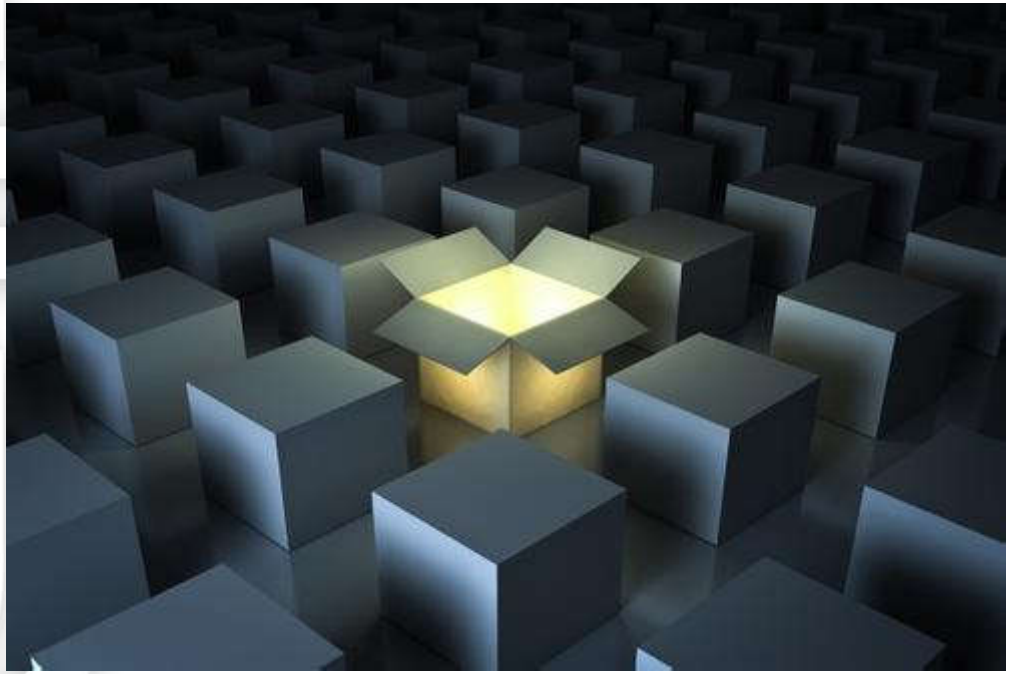
计算机学院并行与分布处理国家重点实验室

主要内容

- 1. TensorBoard简单介绍
- 2. 通过TensorBoard查找错误
- 3. 通过TensorBoard进行超参数搜索
- 4. 参数、特征、图像可视化
- 5. 数据分布可视化

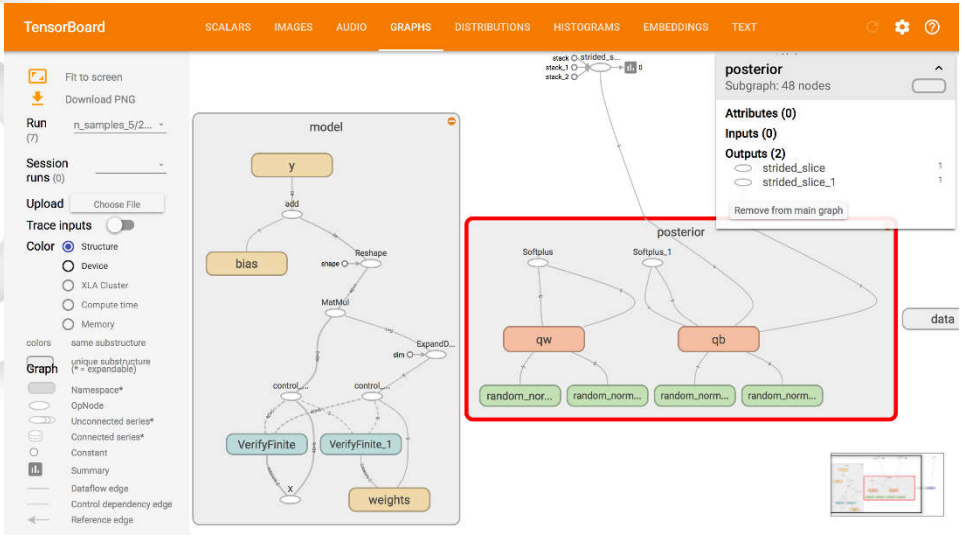
可视化的目的

- 神经网络黑盒子
- 原理、解释？探测、调试？



1. TensorBoard简单介绍

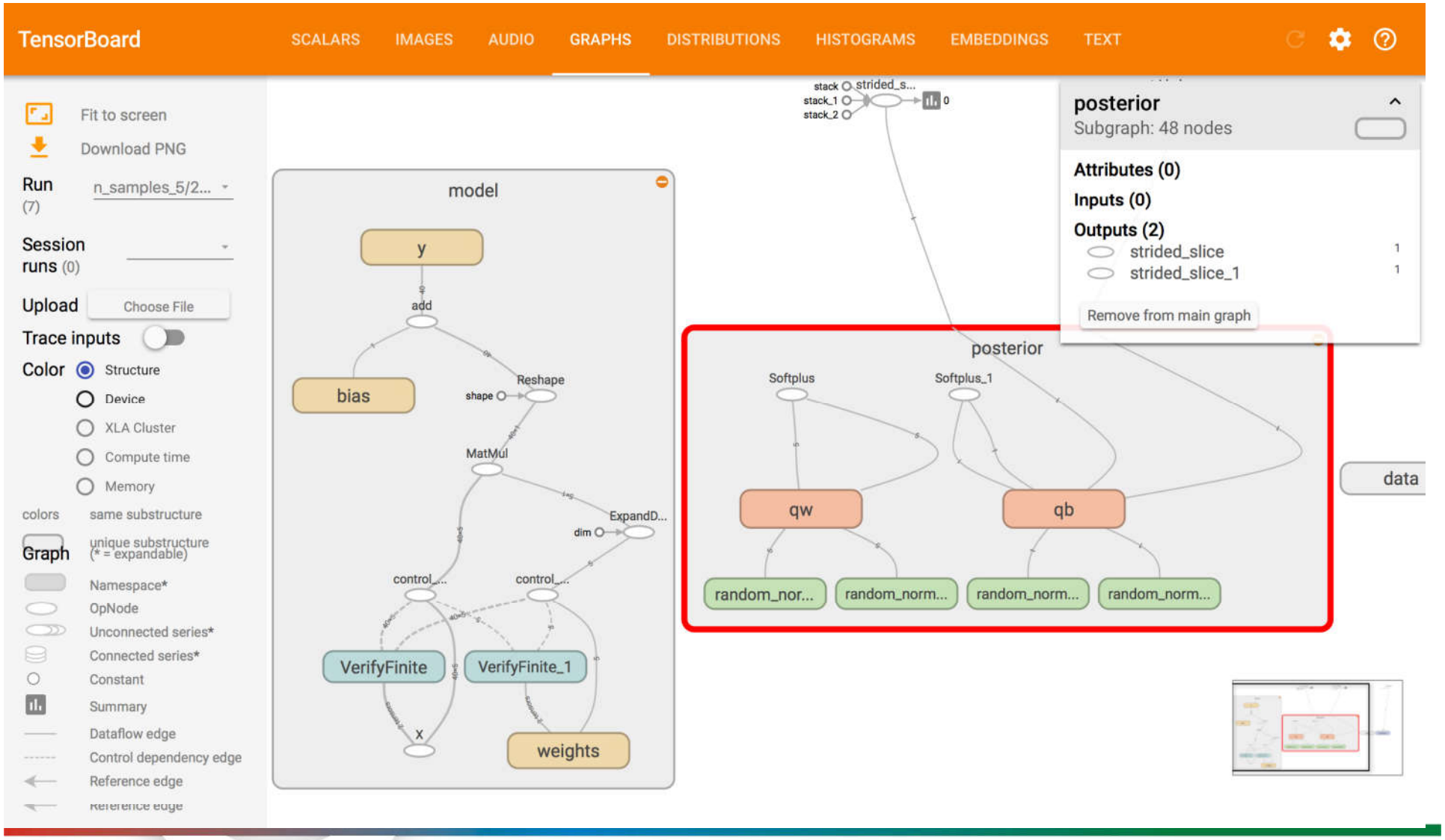
- Tensorflow是个啥？
- TF提供的可视化工具
- 通过网页浏览的方式可视化展示与我们TF构建的模型相关的信息



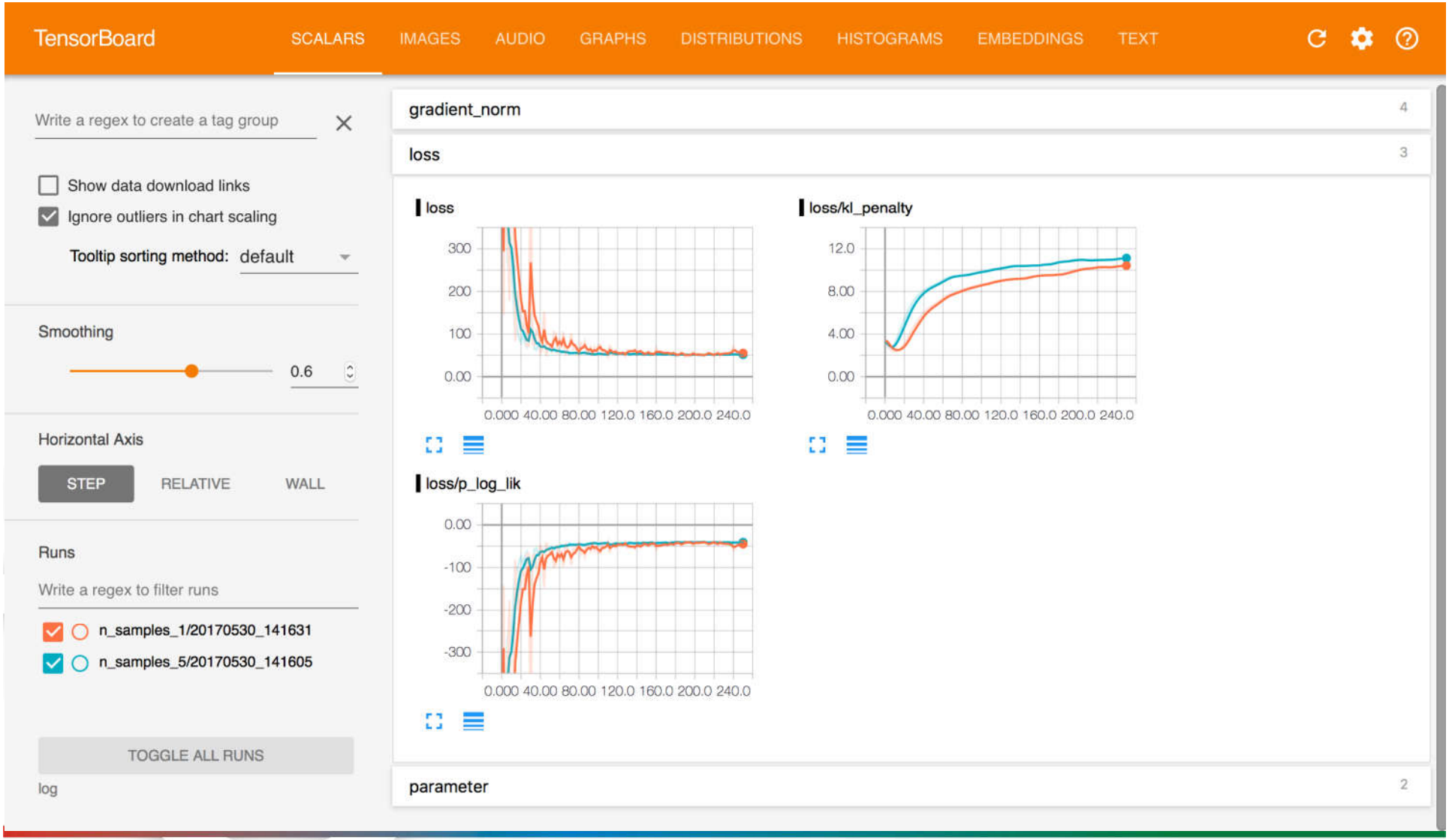
1. TensorBoard介绍

- Tensorboard能看啥？
  - 看网络结构：graph
  - 看训练过程中的指标：loss、acc…
  - 看参数变化：参数均值、方差、分布
  - 看中间结果：中间生成图像、语音等
  - 看数据关系：样本、分类结果分布

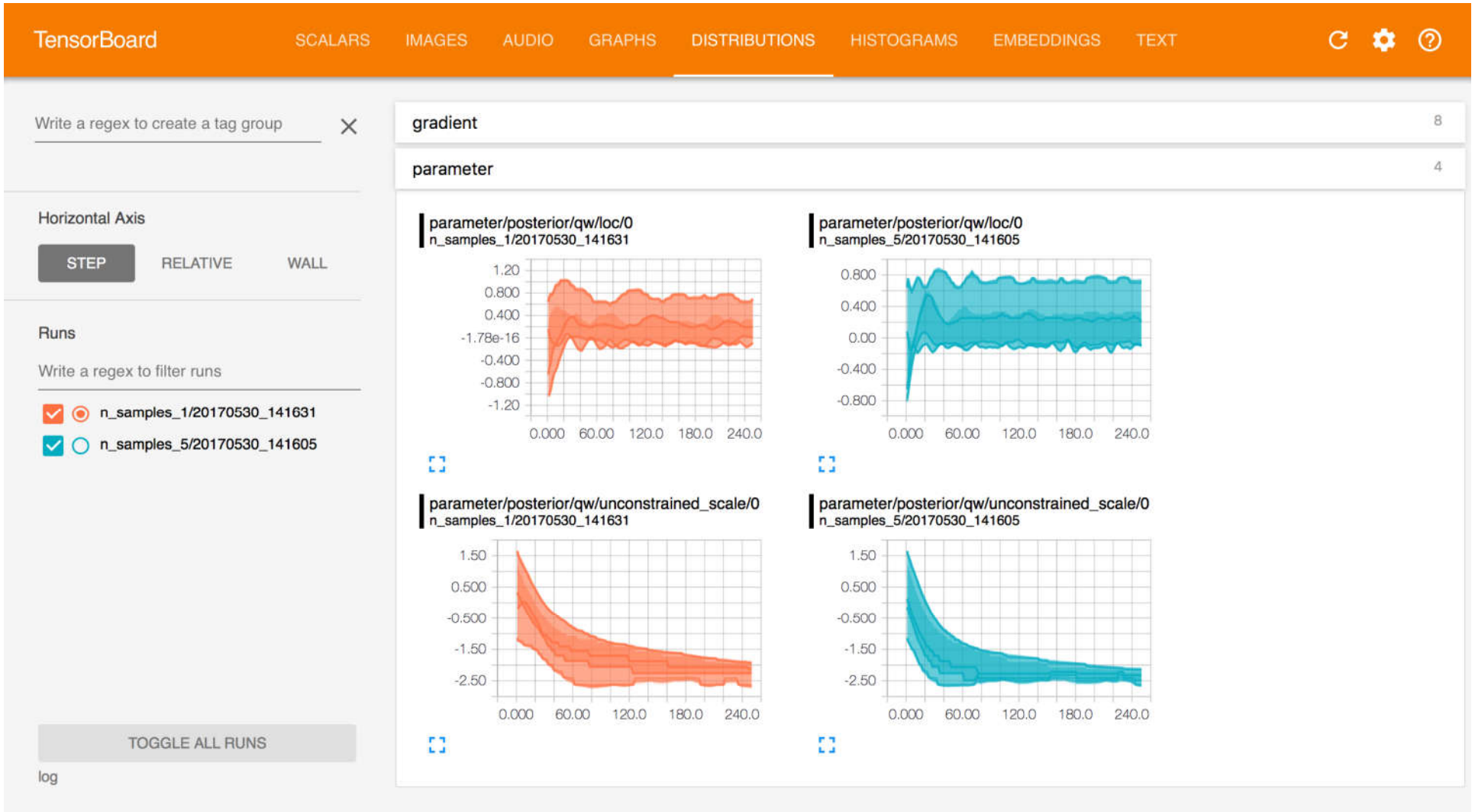
TensorFlow流图



训练过程的loss值


TensorBoard
SCALARS
IMAGES
AUDIO
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
EMBEDDINGS
TEXT
Write a regex to create a tag group
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.6
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
n_samples_1/20170530_141631
n_samples_5/20170530_141605
TOGGLE ALL RUNS
log
gradient_norm 4
loss 3
loss
300
200
100
0.00
0.000 40.00 80.00 120.0 160.0 200.0 240.0
loss/kl_penalty
12.0
8.00
4.00
0.00
0.000 40.00 80.00 120.0 160.0 200.0 240.0
loss/p_log_lik
0.00
-100
-200
-300
0.000 40.00 80.00 120.0 160.0 200.0 240.0
parameter 2

训练参数值的分布


TensorBoard
SCALARS
IMAGES
AUDIO
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
EMBEDDINGS
TEXT
Write a regex to create a tag group
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
n_samples_1/20170530_141631
n_samples_5/20170530_141605
TOGGLE ALL RUNS
log
gradient
8
parameter
4
parameter/posterior/qw/loc/0
n_samples_1/20170530_141631
1.20
0.800
0.400
-1.78e-16
-0.400
-0.800
-1.20
0.000
60.00
120.0
180.0
240.0
parameter/posterior/qw/loc/0
n_samples_5/20170530_141605
0.800
0.400
0.00
-0.400
-0.800
0.000
60.00
120.0
180.0
240.0
parameter/posterior/qw/unconstrained_scale/0
n_samples_1/20170530_141631
1.50
0.500
-0.500
-1.50
-2.50
0.000
60.00
120.0
180.0
240.0
parameter/posterior/qw/unconstrained_scale/0
n_samples_5/20170530_141605
1.50
0.500
-0.500
-1.50
-2.50
0.000
60.00
120.0
180.0
240.0

1. TensorBoard介绍

- Tensorboard咋看？
  - 1. 在TF程序中添加记录并存储日志
  - events..xxxx
  - 2. 命令行启动读取日志文件
  - tensorboard --logdir=logs
  - 3. 打开浏览器（用chrome），按命令行提示输入本机地址和tensorboard通讯端口，刷新浏览
  - 例：http://DESKTOP-xxx:6006

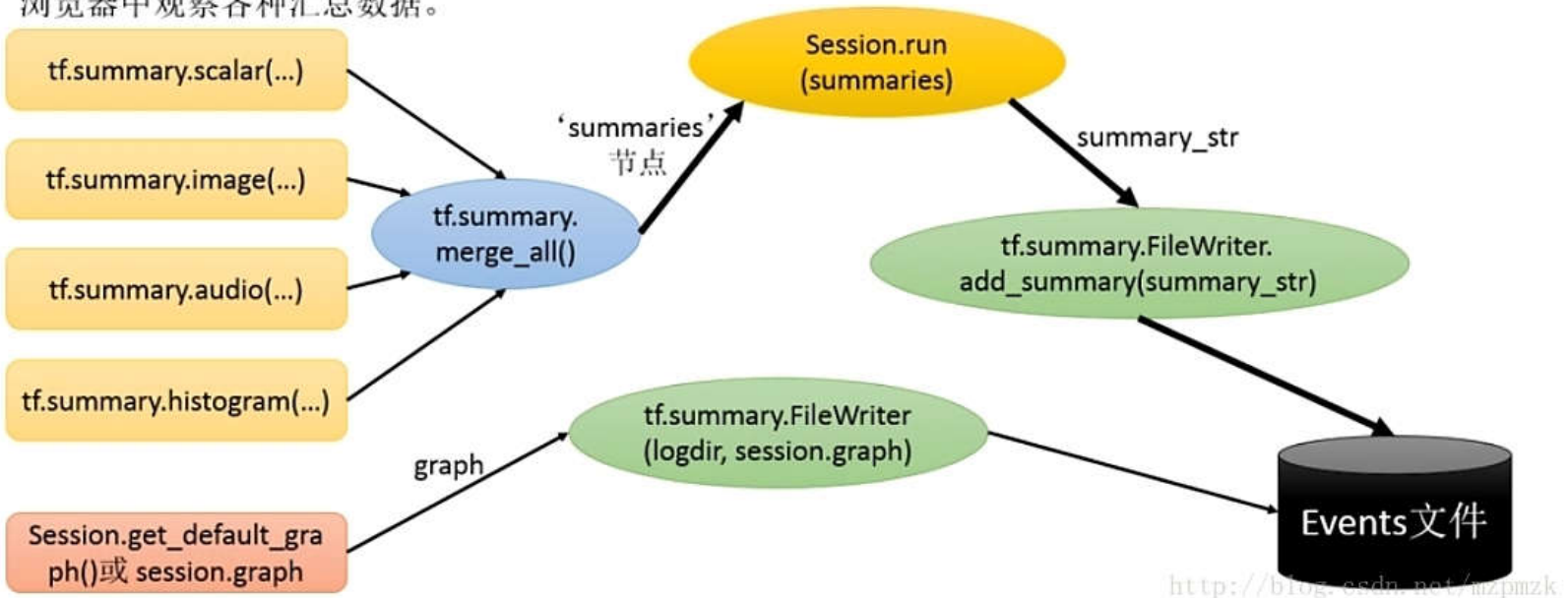
events.out.tfevents.1465366286.ubuntu

1. TensorBoard介绍

- Tensorboard 工作原理

Summary类：负责汇总数据并写入事件文件

使用TensorBoard展示数据，需要在执行Tensorflow就算图的过程中，将各种类型的数据汇总并记录到日志文件中。然后使用TensorBoard读取这些日志文件，解析数据并生产数据可视化的Web页面，让我们可以在浏览器中观察各种汇总数据。



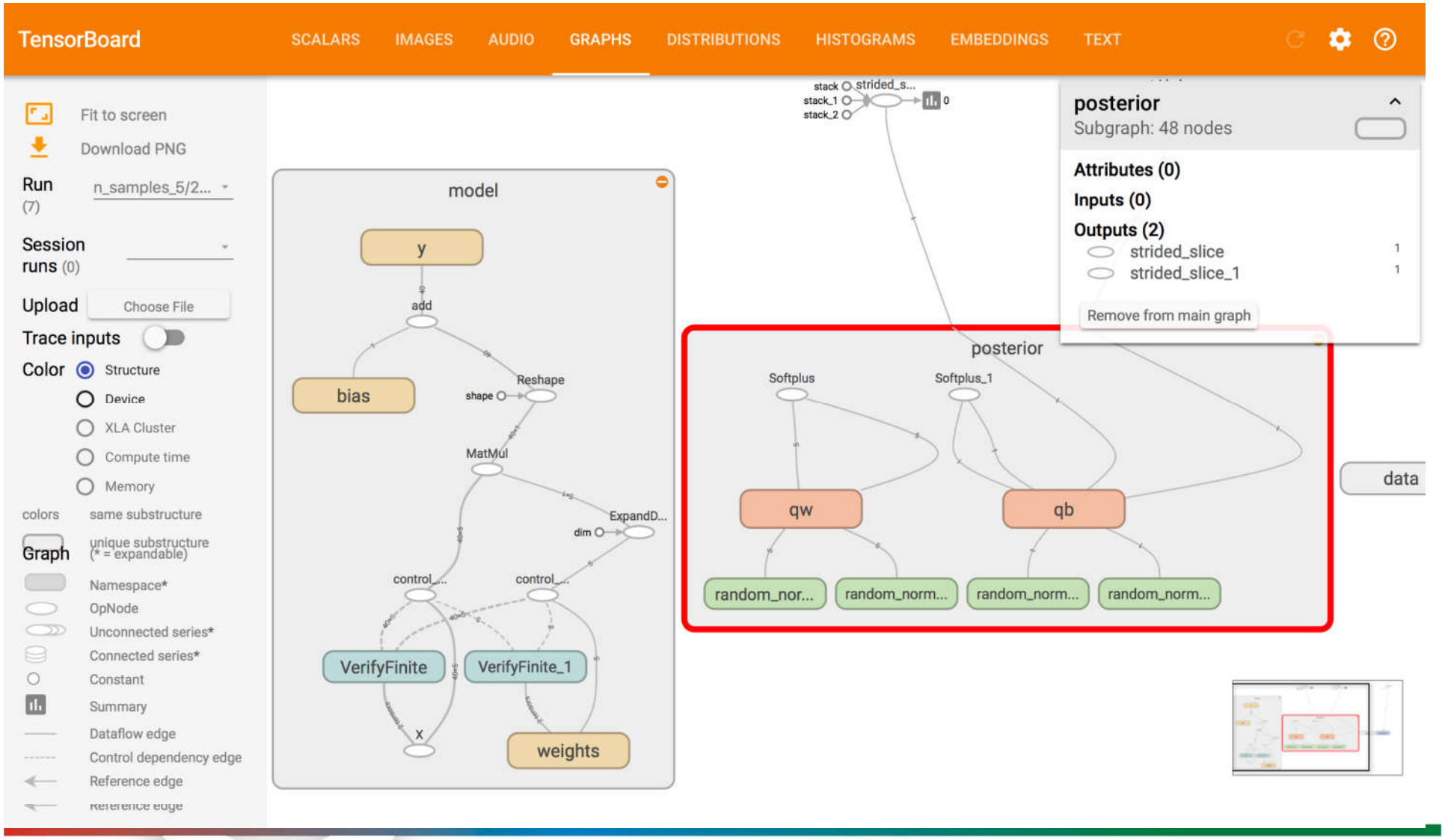
1. TensorBoard介绍

- TF程序中添加Tensorboard 日志记录方法：
  - 1.对感兴趣tensor添加记录操作：summary operation:
    - 例如：tf.scalar_summary('标签',想要记录的变量)
  - 2.汇总需要写入日志的记录：
    - merged = tf.merge_all_summaries()
  - 3.实例化一个日志书写器：
    - summary_writer = tf.summary.FileWriter(logdir, graph=None, flush_secs=120, max_queue=10)，可选同时传入模型graph。或之后用add_graph(graph, global_step=None)添加

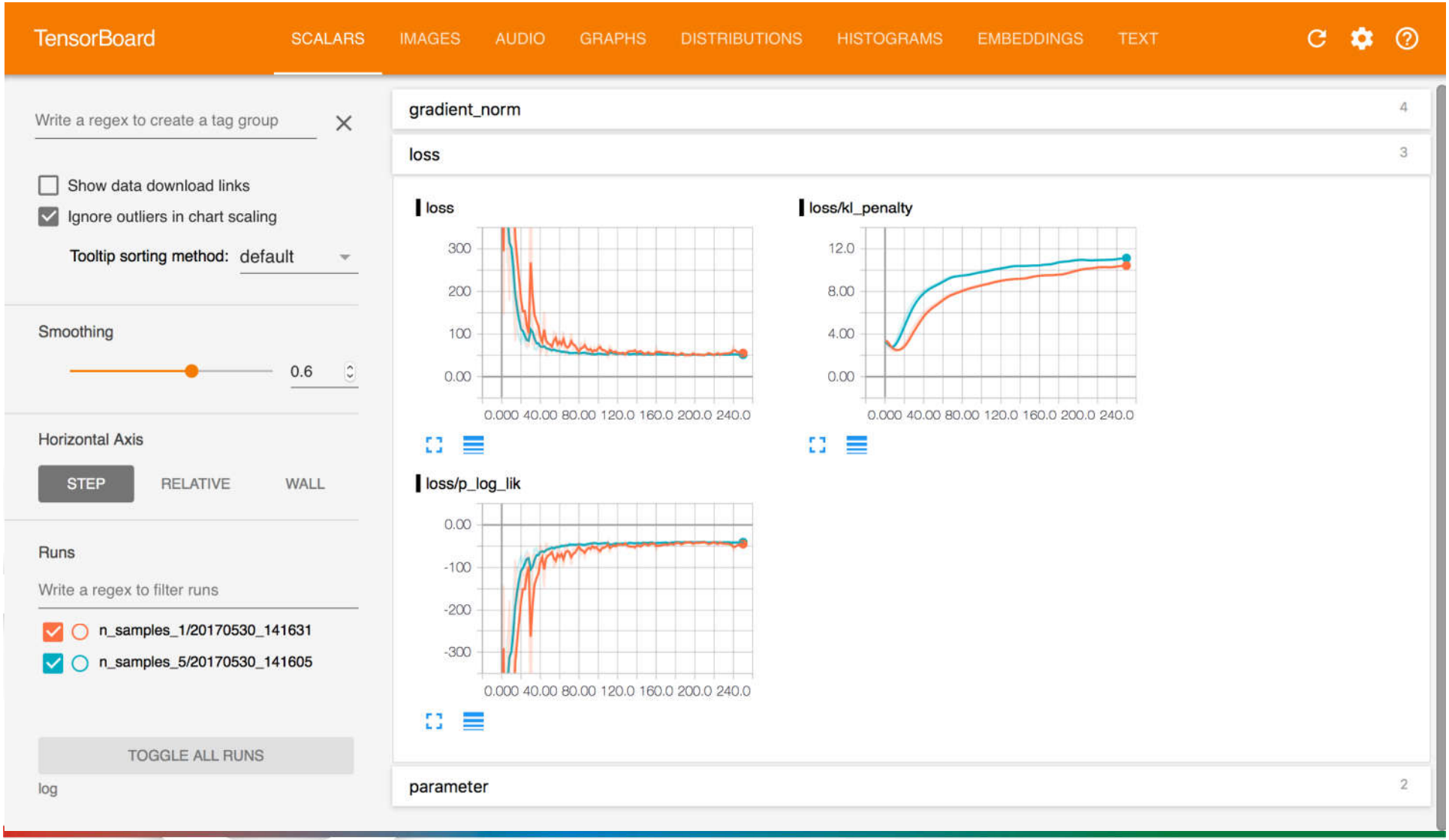
1. TensorBoard介绍

- TF程序中添加Tensorboard 日志记录方法：
  - 4. 运行汇总节点，得到汇总结果：
    - summary = sess.run(merged)，
  - 5.调用书写器实例将汇总日志写入文件
    - summary_writer.add_summary(summary, global_step=i)
    - global_step, 记录的编号要写入！
  - 6. 缓存写入磁盘文件，关闭文件：
    - summary_writer.flush(), 写入，否则按flush_secs间隔写入
    - summary_writer.close(), 写入加关闭文件

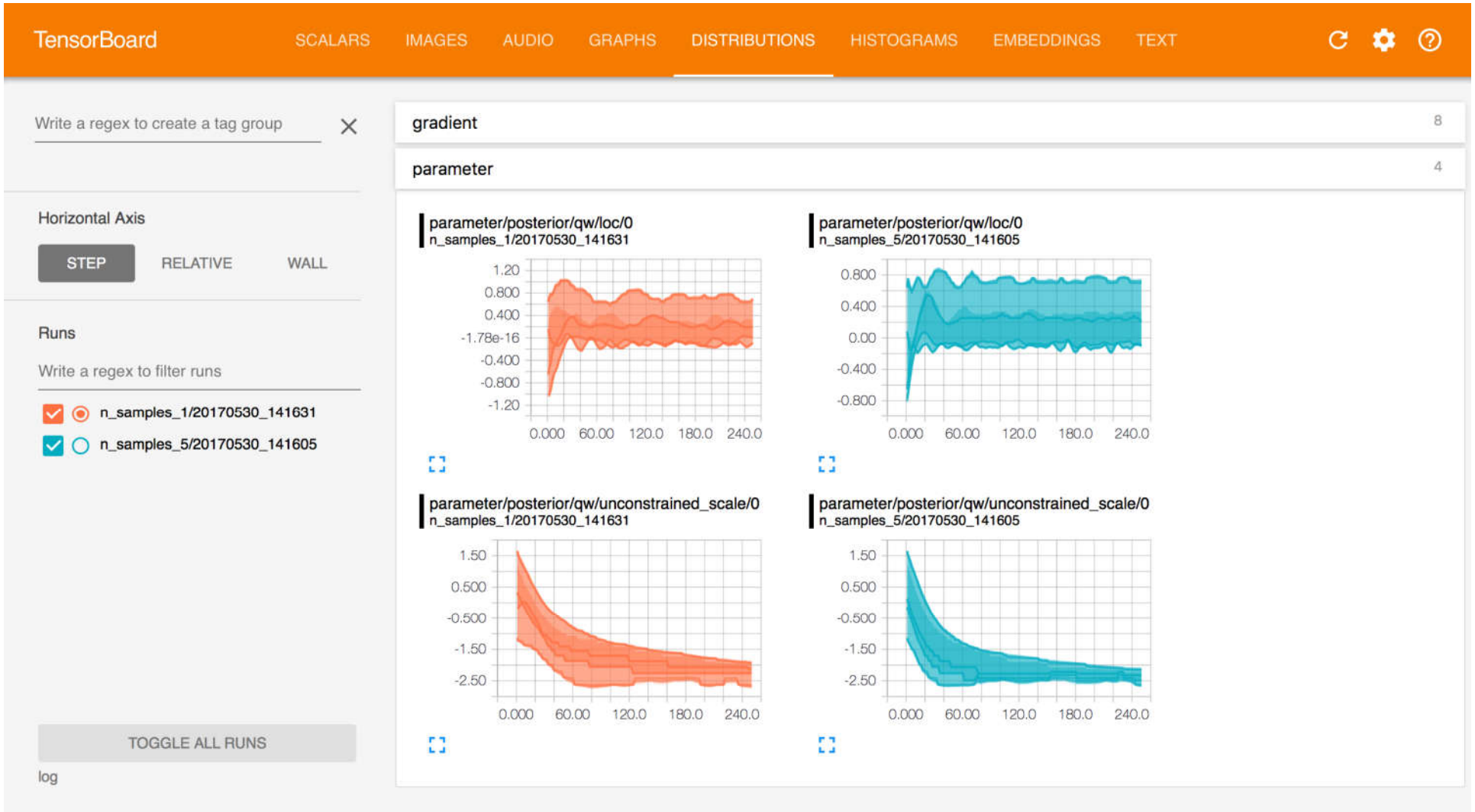
TensorFlow流图



训练过程的loss值

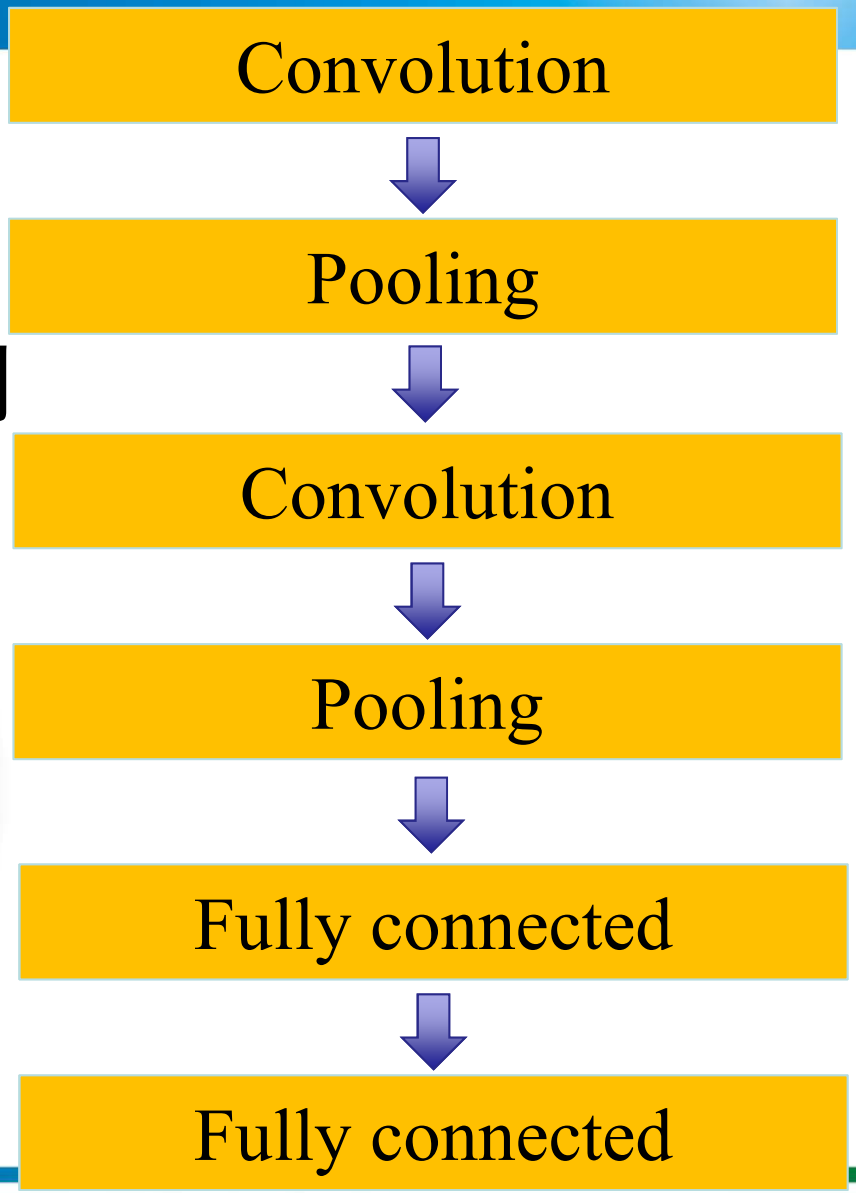

TensorBoard
SCALARS
IMAGES
AUDIO
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
EMBEDDINGS
TEXT
Write a regex to create a tag group
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.6
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
n_samples_1/20170530_141631
n_samples_5/20170530_141605
TOGGLE ALL RUNS
log
gradient_norm 4
loss 3
loss
300
200
100
0.00
0.000 40.00 80.00 120.0 160.0 200.0 240.0
loss/kl_penalty
12.0
8.00
4.00
0.00
0.000 40.00 80.00 120.0 160.0 200.0 240.0
loss/p_log_lik
0.00
-100
-200
-300
0.000 40.00 80.00 120.0 160.0 200.0 240.0
parameter 2

训练参数值的分布


TensorBoard
SCALARS
IMAGES
AUDIO
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
EMBEDDINGS
TEXT
Write a regex to create a tag group
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
n_samples_1/20170530_141631
n_samples_5/20170530_141605
TOGGLE ALL RUNS
log
gradient
8
parameter
4
parameter/posterior/qw/loc/0
n_samples_1/20170530_141631
1.20
0.800
0.400
-1.78e-16
-0.400
-0.800
-1.20
0.000
60.00
120.0
180.0
240.0
parameter/posterior/qw/loc/0
n_samples_5/20170530_141605
0.800
0.400
0.00
-0.400
-0.800
0.000
60.00
120.0
180.0
240.0
parameter/posterior/qw/unconstrained_scale/0
n_samples_1/20170530_141631
1.50
0.500
-0.500
-1.50
-2.50
0.000
60.00
120.0
180.0
240.0
parameter/posterior/qw/unconstrained_scale/0
n_samples_5/20170530_141605
1.50
0.500
-0.500
-1.50
-2.50
0.000
60.00
120.0
180.0
240.0

通过TensorBoard查找错误

- MNIST手写数字识别



网络定义

```
# 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
```

前向传播(feed-forward)

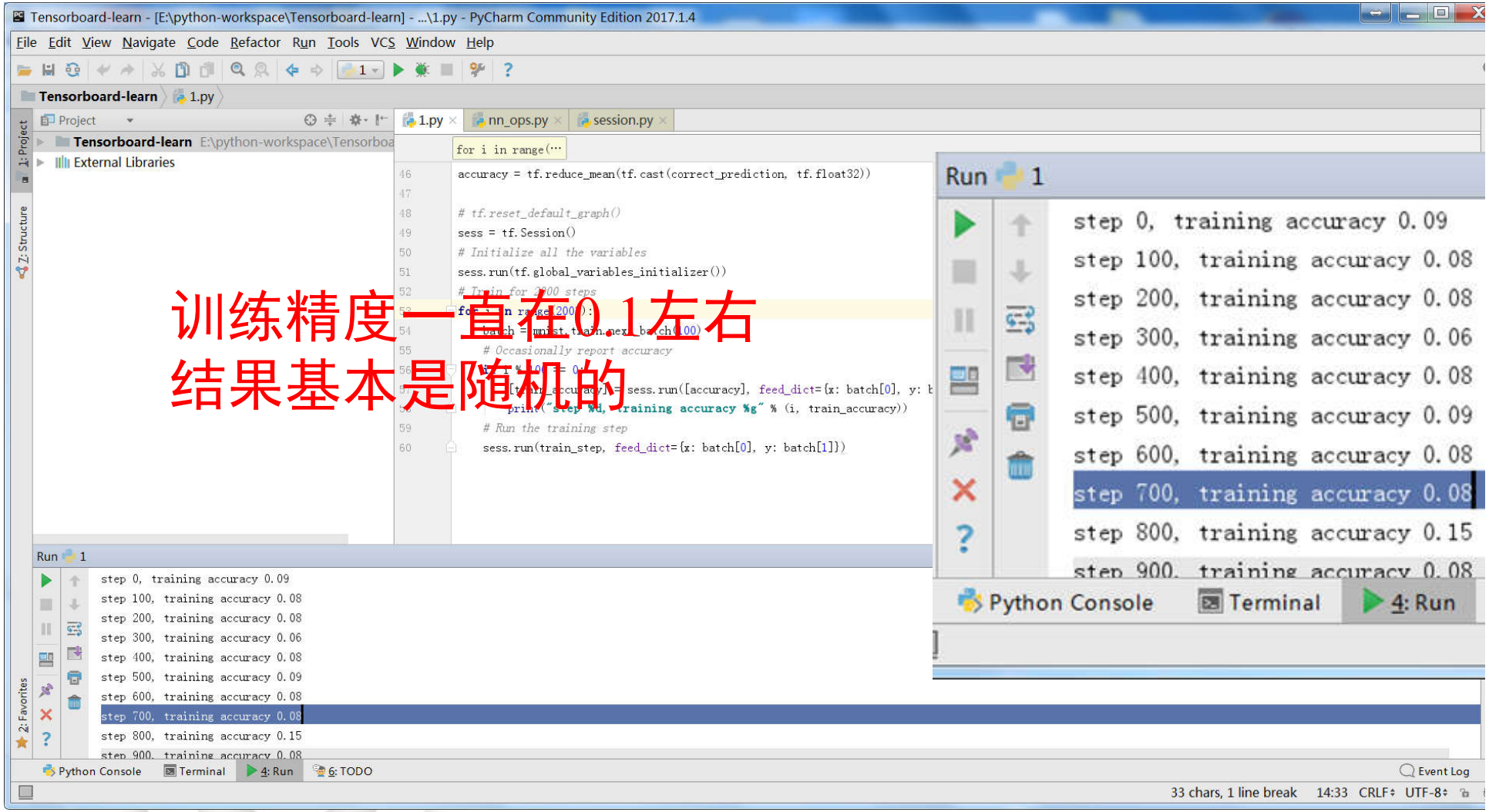
```
# Setup placeholders, and reshape the data
x = tf.placeholder(tf.float32, shape=[None, 784])
y = tf.placeholder(tf.float32, shape=[None, 10])
x_image = tf.reshape(x, [-1, 28, 28, 1])
# Create the network
conv1 = conv_layer(x_image, 1, 32)
pool1 = tf.nn.max_pool(conv1, ksize=[1, 2, 2, 1], strides=[1, 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)
y_predicted = fc_layer(fc1, 1024, 10)
```

损失函数

```
# Compute cross entropy as our loss function
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y*tf.log(y_predicted)))
# 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))
```

训练

```
sess = tf.Session()
# Initialize all the variables
sess.run(tf.global_variables_initializer())
# Train for 2000 steps
for i in range(2001):
    batch = mnist.train.next_batch(100)
    # Occasionally report accuracy
    if i % 100 == 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: batch[1]})
```

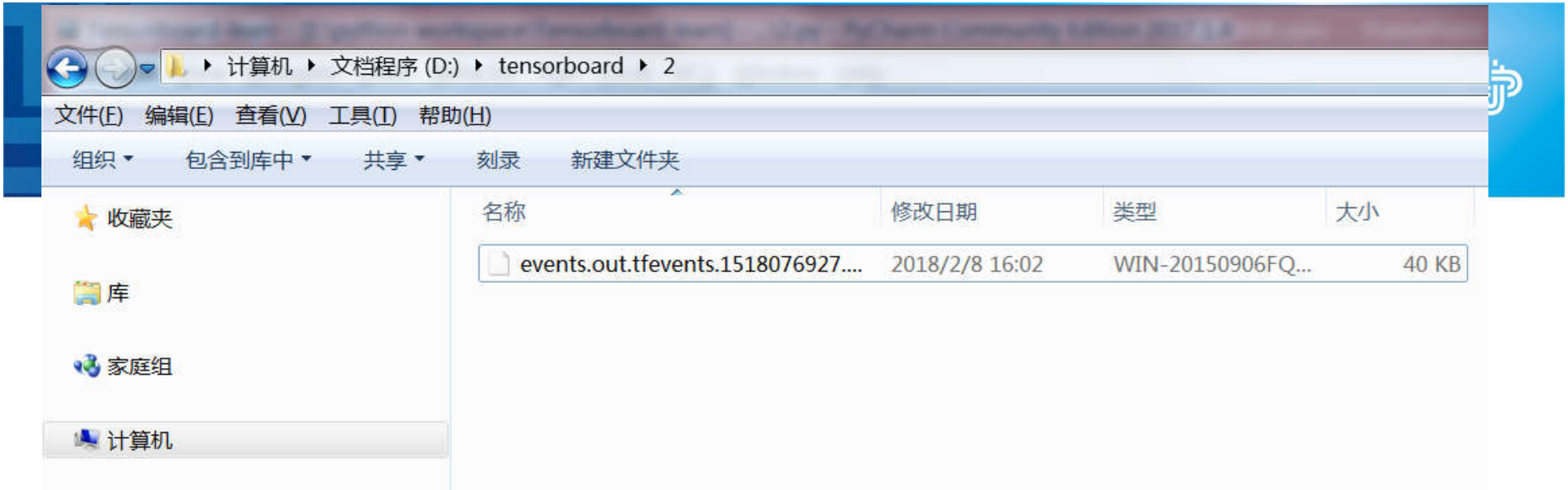

Tensorboard-learn - [E:\python-workspace\Tensorboard-learn] - ...\1.py - PyCharm Community Edition 2017.1.4
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Tensorboard-learn
1.py
Project
Tensorboard-learn E:\python-workspace\Tensorboa
External Libraries
1.py
nn_ops.py
session.py
for i in range(...
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
# tf.reset_default_graph()
sess = tf.Session()
# 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 % 100 == 0:
train_accuracy = sess.run([accuracy], feed_dict={x: batch[0], y: b
print("step %d, training accuracy %g" % (i, train_accuracy))
# Run the training step
sess.run(train_step, feed_dict={x: batch[0], y: batch[1]})
Run 1
step 0, training accuracy 0.09
step 100, training accuracy 0.08
step 200, training accuracy 0.08
step 300, training accuracy 0.06
step 400, training accuracy 0.08
step 500, training accuracy 0.09
step 600, training accuracy 0.08
step 700, training accuracy 0.08
step 800, training accuracy 0.15
step 900, training accuracy 0.08
Python Console
Terminal
4: Run
6: TODO
Event Log
33 chars, 1 line break
14:33
CRLF
UTF-8
训练精度一直在0.1左右
结果基本是随机的

用TensorBoard来查错

- tf.summary.FileWriter (n):
  - 一个用于用于输出Tensorboard数据的 Python 类

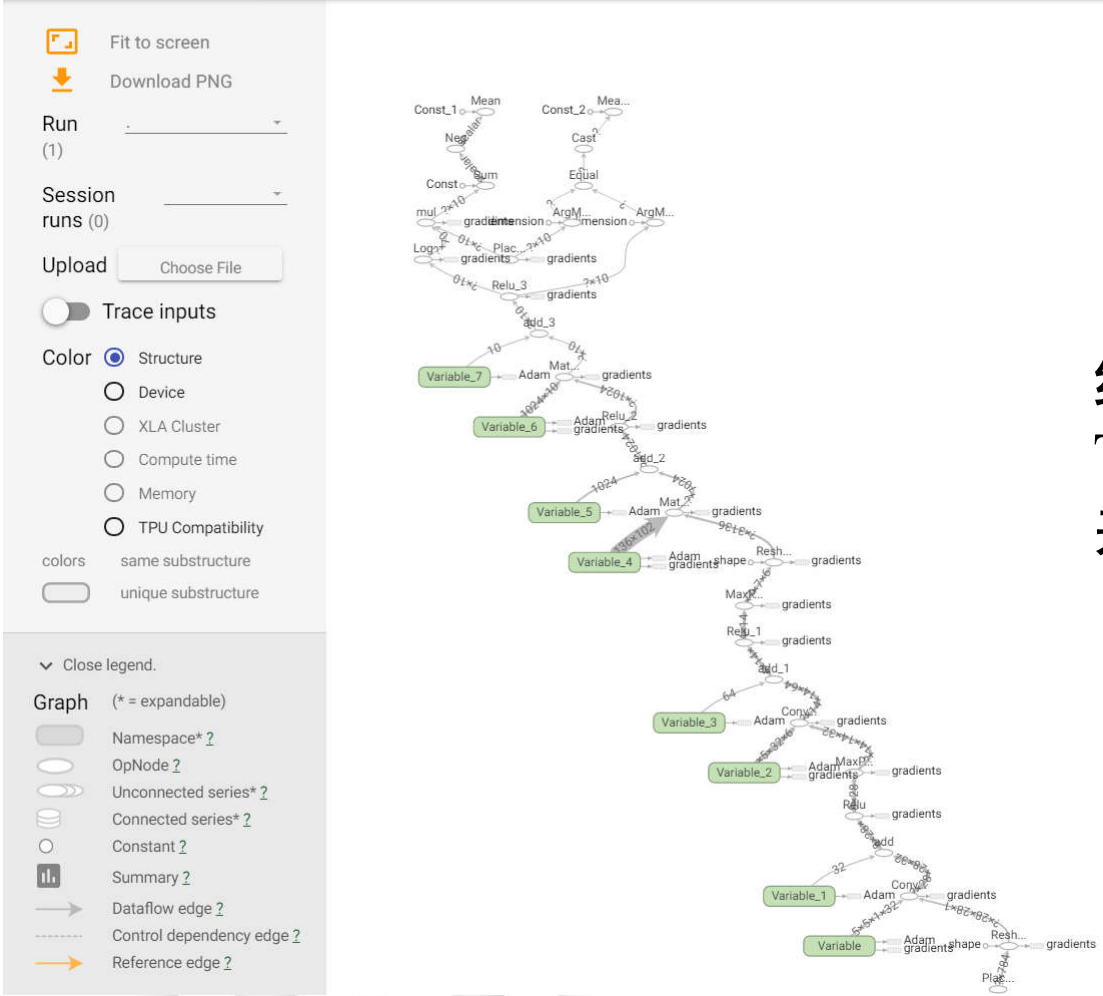
```
sess = tf.Session()
writer = tf.summary.FileWriter(LOGDIR + "2")
writer.add_graph(sess.graph)
```

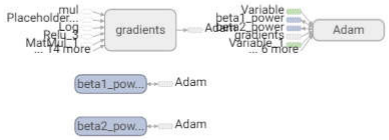
- 将结果输出到LOGDIR的子文件夹中



- 在LOGDIR的子目录可以见到一个新的文件
- 启动cmd，在命令行中输入Tensorboard –logdir D:/tensorboard/2

流图





结果有点乱，在TensorBoard里指定变量名，来提高流图的可读性

命名在Tensorboard中显示的变量

```
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 act

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")
        act = tf.nn.relu(tf.matmul(input, w) + b)

        return act
```

命名变量

```
conv1 = conv_layer(x_image, 1, 32, "conv1")
with tf.name_scope("Pool1"):
    pool1 = tf.nn.max_pool(conv1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1],
                           padding="SAME", name = "Pool1")
conv2 = conv_layer(pool1, 32, 64, "conv2")
with tf.name_scope("Pool2"):
    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, "fc1")

y_predicted = fc_layer(fc1, 1024, 10, "fc2")
```

```
writer = tf.summary.FileWriter(LOGDIR + "3")
```

TensorBoard GRAPHS INACTIVE

Fit to screen
Download PNG

Run (2) .

Session runs (0)

Upload Choose File

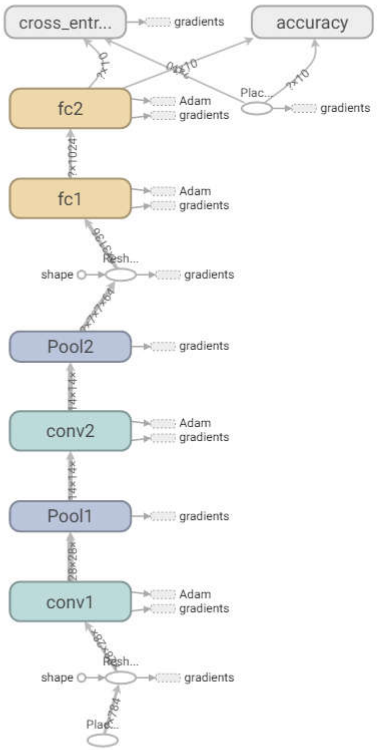
Trace inputs

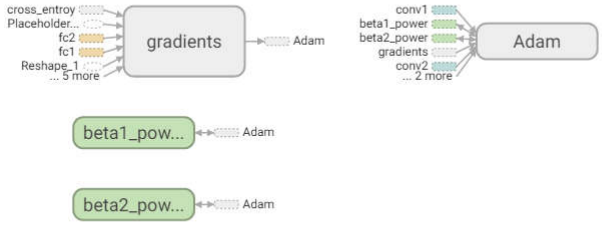
Color
- Structure
- Device
- XLA Cluster
- Compute time
- Memory
- TPU Compatibility

colors: same substructure, unique substructure

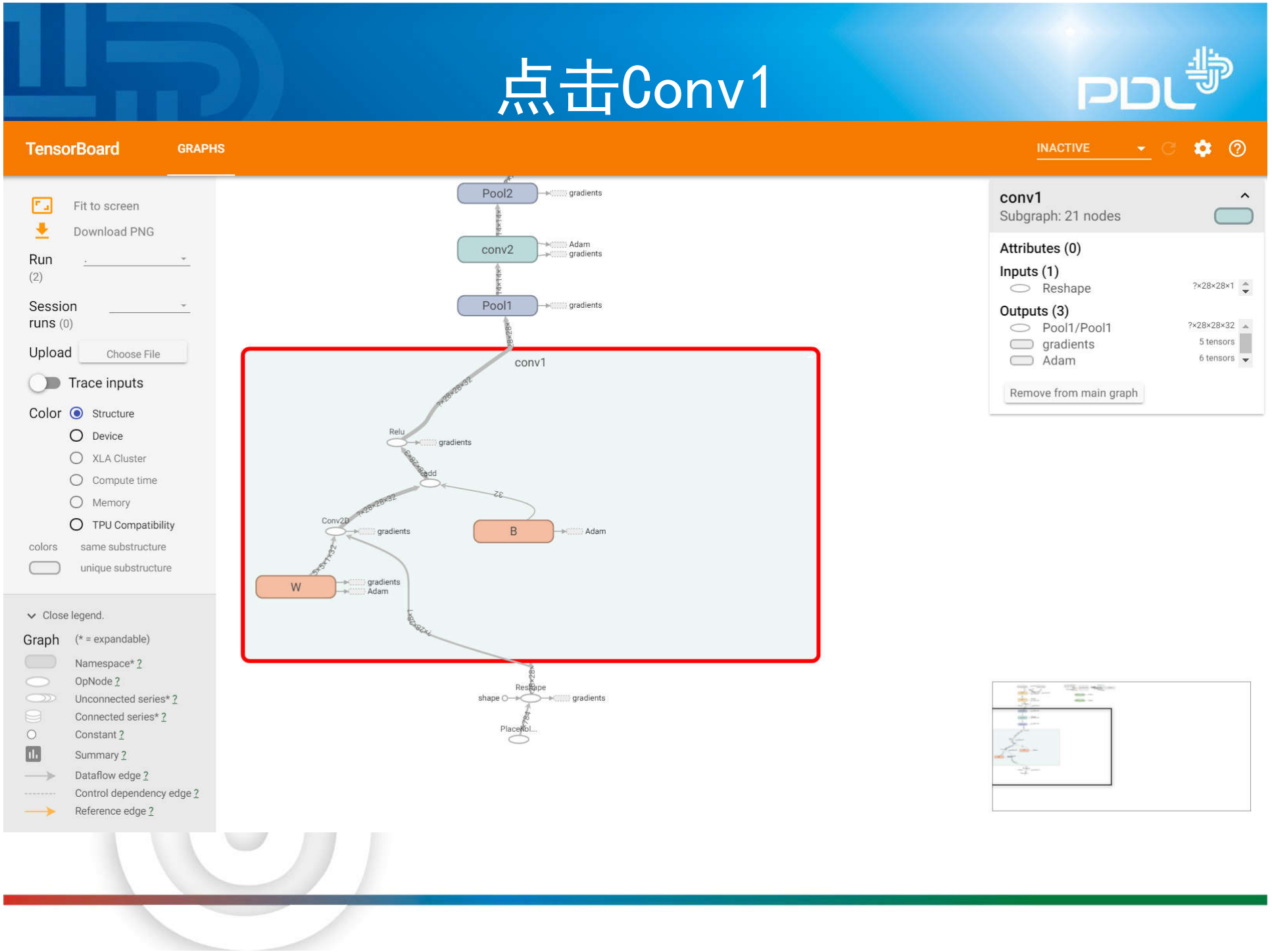
Close legend.

Graph (* = expandable)
- Namespace* ?
- OpNode ?
- Unconnected series* ?
- Connected series* ?
- Constant ?
- Summary ?
- Dataflow edge ?
- Control dependency edge ?
- Reference edge ?

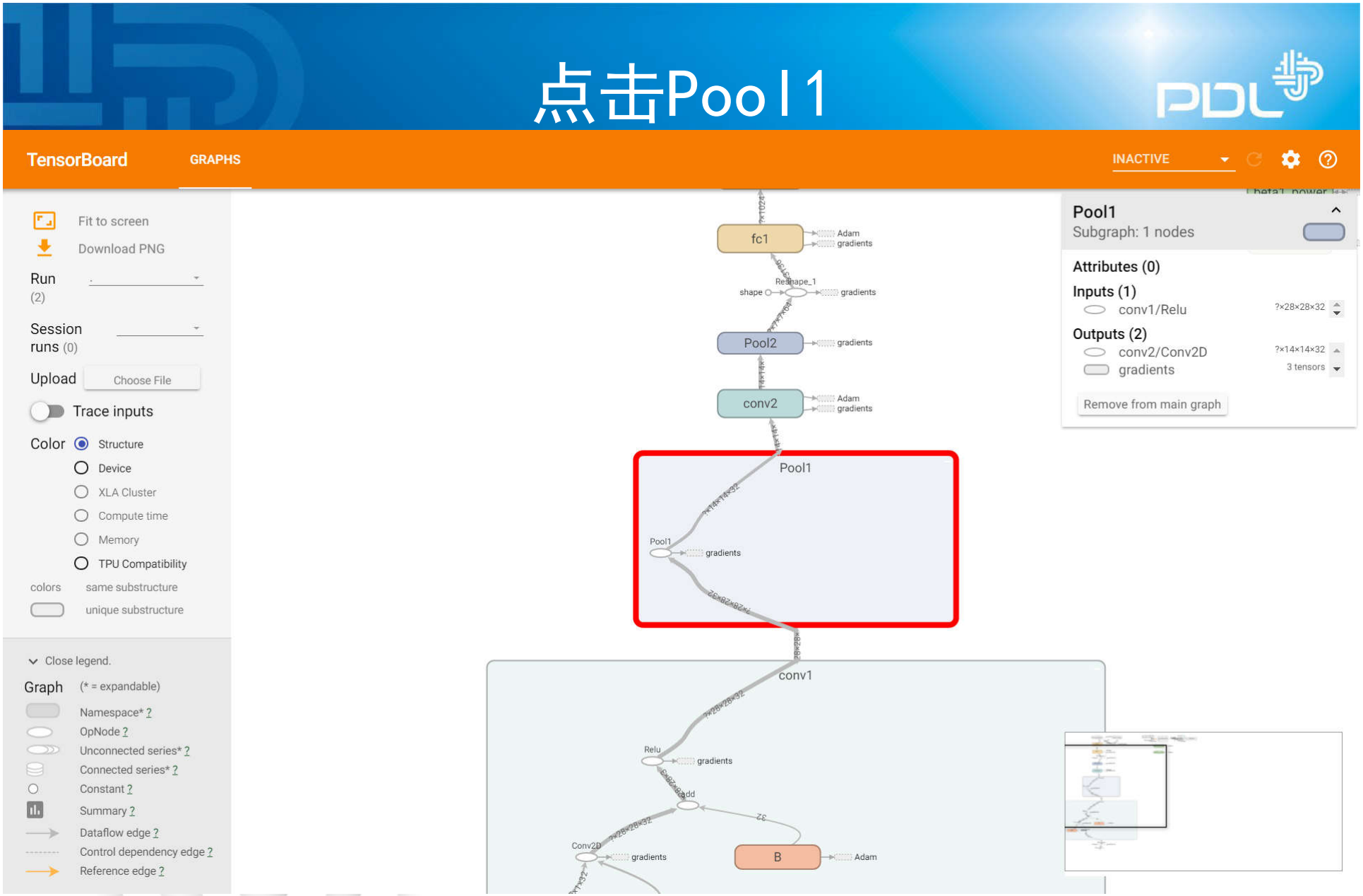




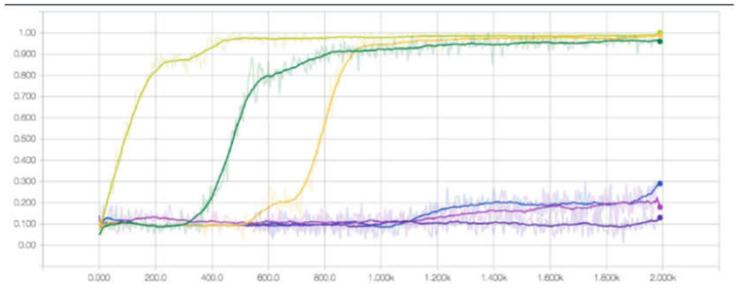
点击Conv1


TensorBoard
GRAPHS
INACTIVE
Fit to screen
Download PNG
Run (2)
Session runs (0)
Upload
Choose File
Trace inputs
Color
Structure
Device
XLA Cluster
Compute time
Memory
TPU Compatibility
colors
same substructure
unique substructure
Close legend.
Graph (* = expandable)
Namespace* ?
OpNode ?
Unconnected series* ?
Connected series* ?
Constant ?
Summary ?
Dataflow edge ?
Control dependency edge ?
Reference edge ?
Pool2
gradients
conv2
Adam
gradients
Pool1
gradients
conv1
Relu
gradients
add
Conv2D
gradients
B
Adam
W
gradients
Adam
Reshape
shape
gradients
conv1
Subgraph: 21 nodes
Attributes (0)
Inputs (1)
Reshape
?×28×28×1
Outputs (3)
Pool1/Pool1
?×28×28×32
gradients
5 tensors
Adam
6 tensors
Remove from main graph

点击Pool1


TensorBoard
GRAPHS
INACTIVE
Fit to screen
Download PNG
Run (2)
Session runs (0)
Upload
Choose File
Trace inputs
Color
Structure
Device
XLA Cluster
Compute time
Memory
TPU Compatibility
colors
same substructure
unique substructure
Close legend.
Graph (* = expandable)
Namespace* ?
OpNode ?
Unconnected series* ?
Connected series* ?
Constant ?
Summary ?
Dataflow edge ?
Control dependency edge ?
Reference edge ?
fc1
Adam
gradients
Reshape_1
shape
Pool2
conv2
Pool1
conv1
Relu
add
Conv2D
B
Pool1
Subgraph: 1 nodes
Attributes (0)
Inputs (1)
conv1/Relu
?×28×28×32
Outputs (2)
conv2/Conv2D
?×14×14×32
gradients
3 tensors
Remove from main graph

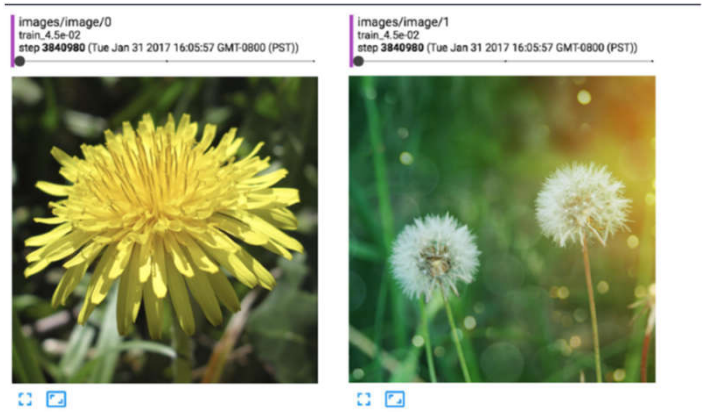
输出中间数据

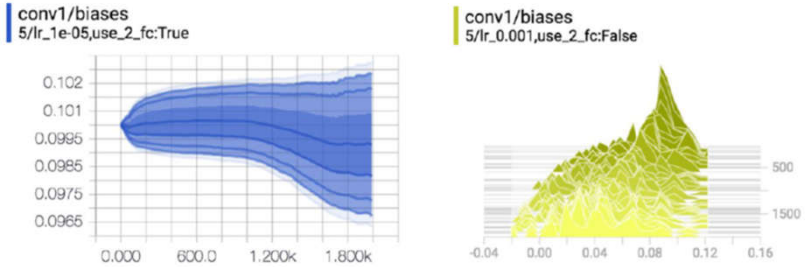


- Summary(n)
  - 输出中间数据的函数

举例:

- tf.summary.scalar （标量）
- **tf.summary.image （图片）**
- **tf.summary.audio （声音)**
- **tf.summary.histogram (统计数据)**





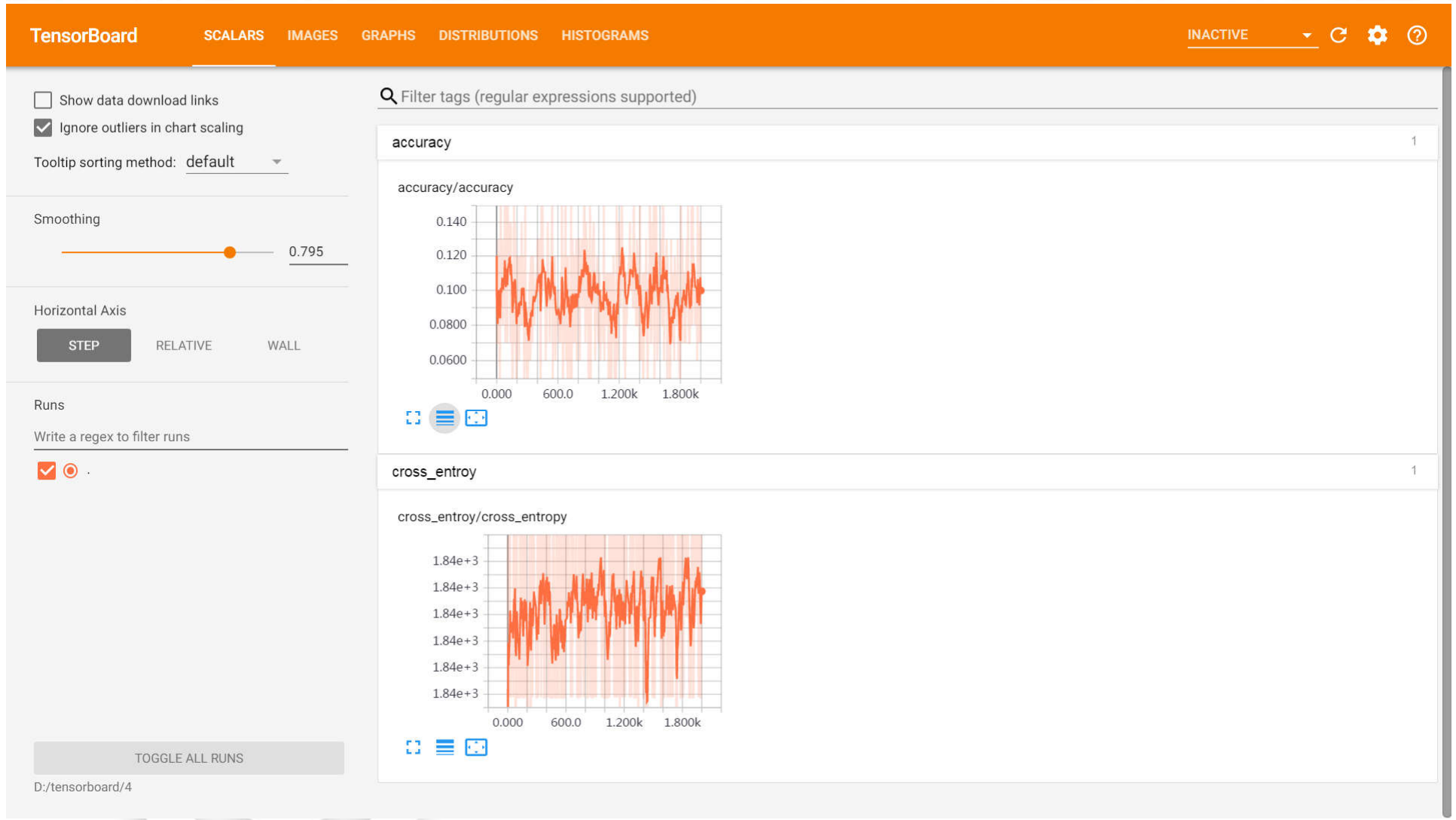
添加要查看的变量

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

```
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 act
```

```
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")
        act = tf.nn.relu(tf.matmul(input, w) + b)
        tf.summary.histogram("weights", w)
        tf.summary.histogram("biases", b)
        tf.summary.histogram("activations", act)

        return act

merged_summary = tf.summary.merge_all()
sess = tf.Session()
writer = tf.summary.FileWriter(LOGDIR + "4")
writer.add_graph(sess.graph)
sess.run(tf.global_variables_initializer()) # Initialize all the variables
# Train for 2000 steps
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]})
```

查看Accuracy以及cross entropy



查看输入


TensorBoard
SCALARS
IMAGES
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
INACTIVE
Show actual image size
Brightness adjustment
RESET
Contrast adjustment
RESET
Runs
Write a regex to filter runs
TOGGLE ALL RUNS
D:/tensorboard/4
Filter tags (regular expressions supported)
input
3
input/image/0
step 2,000 Thu Feb 08 2018 19:31:51 GMT+0800 (中国标准时间)
input/image/1
step 2,000 Thu Feb 08 2018 19:31:51 GMT+0800 (中国标准时间)
input/image/2
step 2,000 Thu Feb 08 2018 19:31:51 GMT+0800 (中国标准时间)

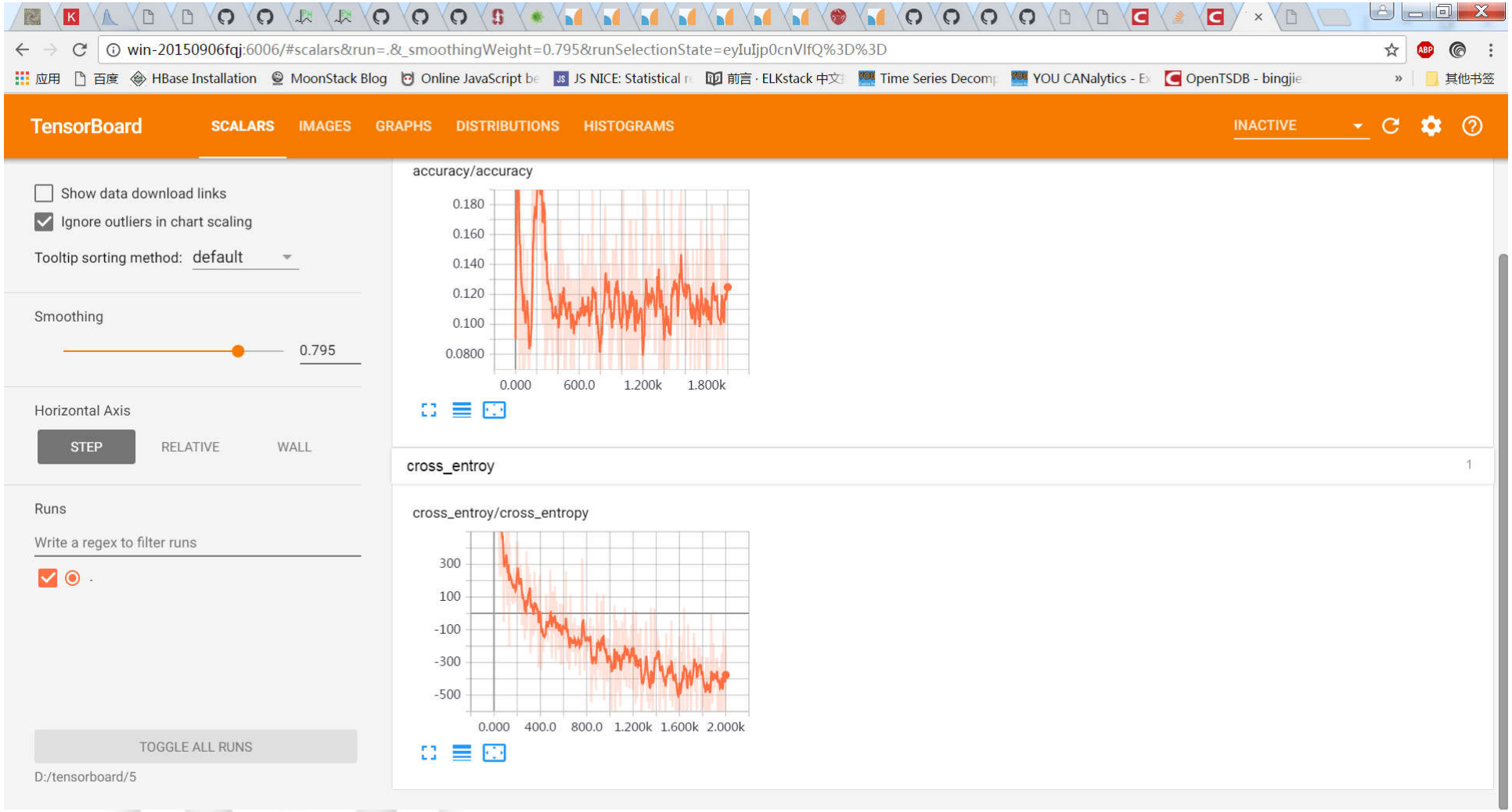
查看参数分布

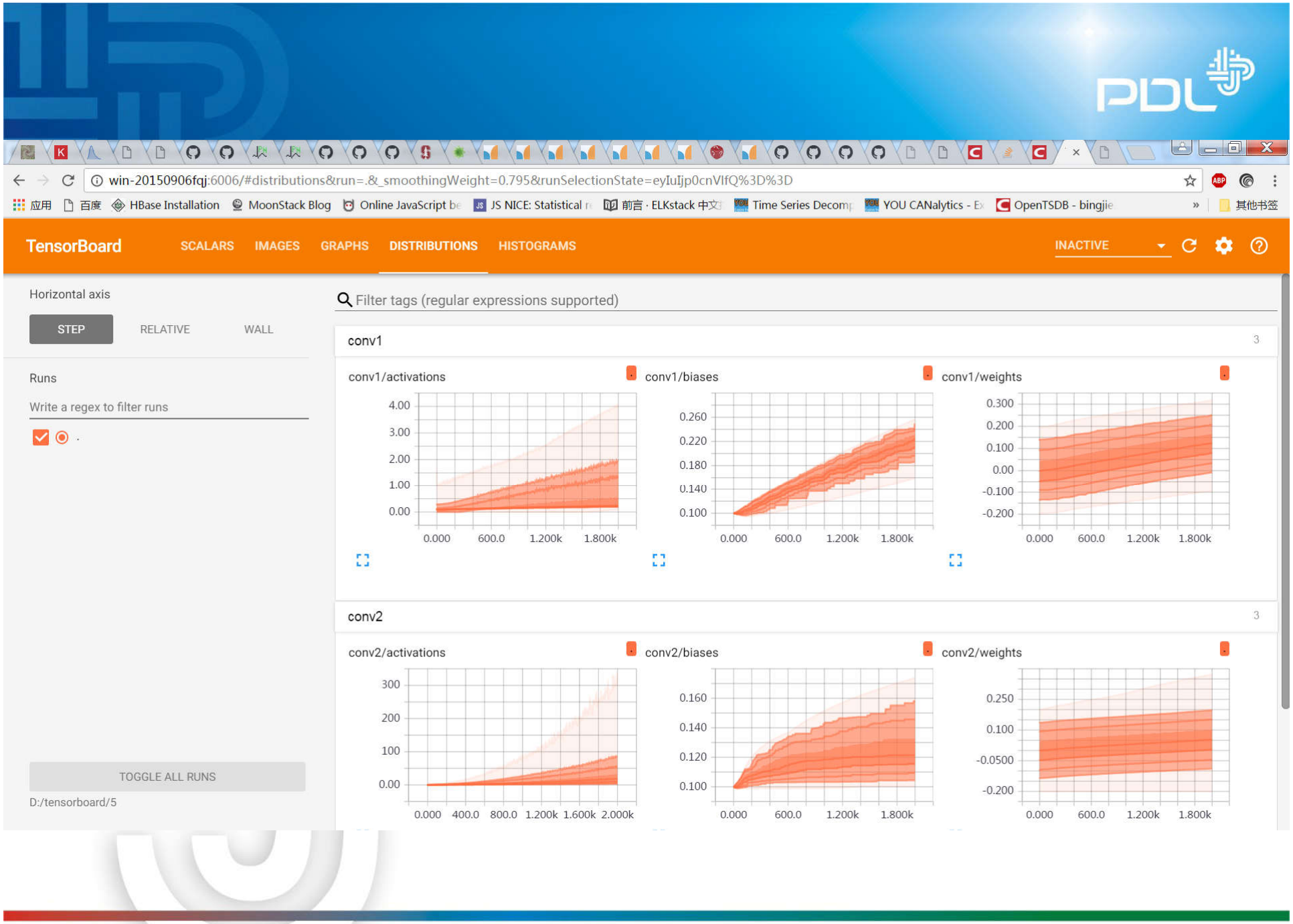

TensorBoard
SCALARS
IMAGES
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
INACTIVE
Horizontal axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
TOGGLE ALL RUNS
D:/tensorboard/4
Filter tags (regular expressions supported)
conv1
3
conv1/activations
conv1/biases
conv1/weights
0.800
0.400
0.00
-0.400
-0.800
0.000
600.0
1.200k
1.800k
conv2
3
conv2/activations
conv2/biases
conv2/weights
fc1
3
fc2
3

变量不能都初始化为0

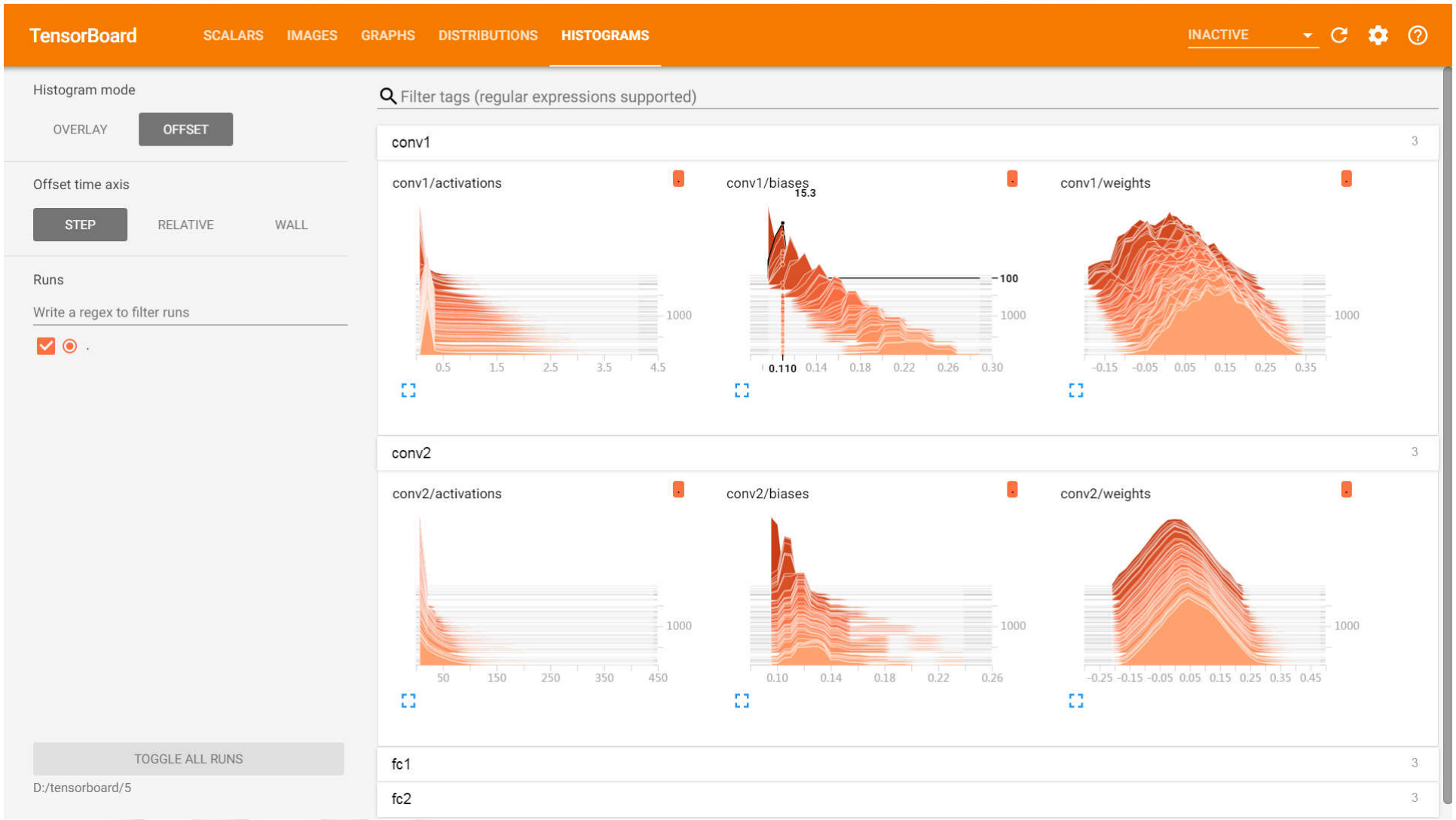
```
def conv_layer(input, channels_in, channels_out, name="conv"):
    w = tf.Variable(tf.zeros([5, 5, channels_in, channels_out]), name="W")
    b = tf.Variable(tf.zeros([channels_out]), name = "B")

    w = tf.Variable(tf.truncated_normal([5, 5, channels_in, channels_out],
stddev=0.1), name = "W")
    b = tf.Variable(tf.constant(0.1, shape=[channels_out]), name="B")

def fc_layer(input, channels_in, channels_out, name = "fc"):
    w = tf.Variable(tf.zeros([channels_in, channels_out]), name = "W")
    b = tf.Variable(tf.zeros([channels_out]), name = "B")

    w = tf.Variable(tf.truncated_normal([channels_in, channels_out],
stddev=0.1), name="W")
    b = tf.Variable(tf.constant(0.1, shape=[channels_out]), name="B")
```

Entropy 下降了，但是Accuracy不行

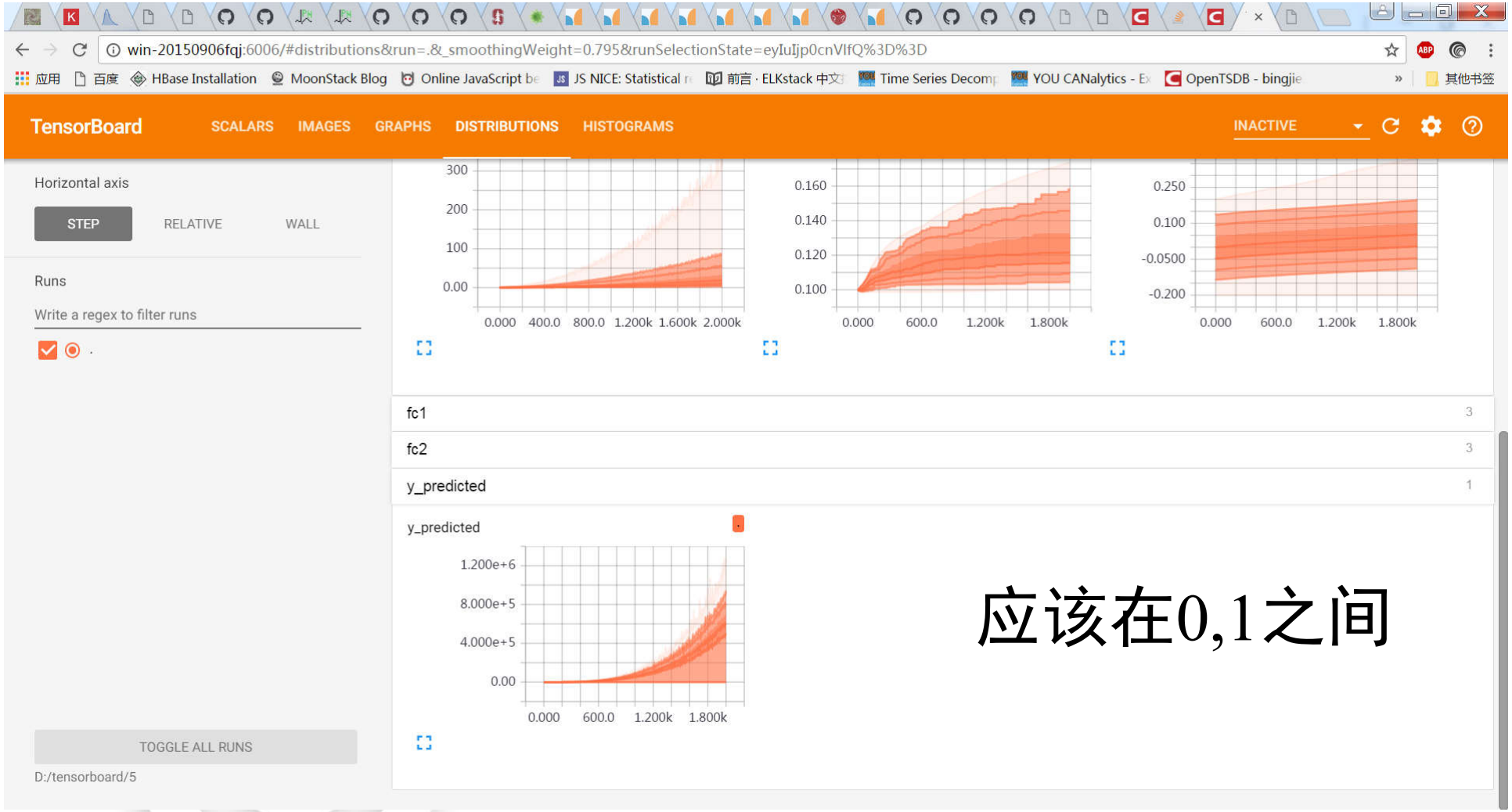



win-20150906fqj:6006/#distributions&run=.&_smoothingWeight=0.795&runSelectionState=eyIuIjp0cnVlfQ%3D%3D
TensorBoard
SCALARS
IMAGES
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
INACTIVE
Horizontal axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
TOGGLE ALL RUNS
D:/tensorboard/5
Filter tags (regular expressions supported)
conv1
3
conv1/activations
conv1/biases
conv1/weights
conv2
3
conv2/activations
conv2/biases
conv2/weights

训练结果还是有问题


TensorBoard
SCALARS
IMAGES
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
INACTIVE
Histogram mode
OVERLAY
OFFSET
Offset time axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
TOGGLE ALL RUNS
D:/tensorboard/5
Filter tags (regular expressions supported)
conv1
3
conv1/activations
conv1/biases
15.3
100
conv1/weights
1000
0.5 1.5 2.5 3.5 4.5
0.110 0.14 0.18 0.22 0.26 0.30
-0.15 -0.05 0.05 0.15 0.25 0.35
conv2
3
conv2/activations
conv2/biases
conv2/weights
50 150 250 350 450
0.10 0.14 0.18 0.22 0.26
-0.25 -0.15 -0.05 0.05 0.15 0.25 0.35 0.45
fc1
3
fc2
3

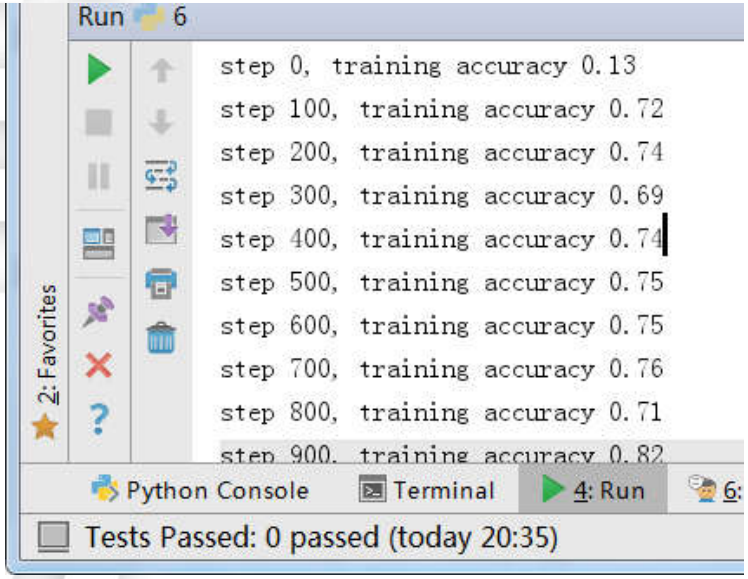
y_predicted 不对



应该在0,1之间

忘记加Softmax了

```
fc1 = fc_layer(flattened, 7 * 7 * 64, 1024, "fc1")
y_predicted = fc_layer(fc1, 1024, 10, "fc2")
fc2 = fc_layer(fc1, 1024, 10, "fc2")
y_predicted = tf.nn.softmax(fc2)
```



Accuracy 提高

TensorBoard SCALARS IMAGES GRAPHS DISTRIBUTIONS HISTOGRAMS INACTIVE

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0.795

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

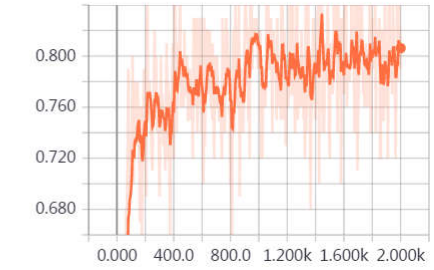
TOGGLE ALL RUNS

D:/tensorboard/6

Filter tags (regular expressions supported)

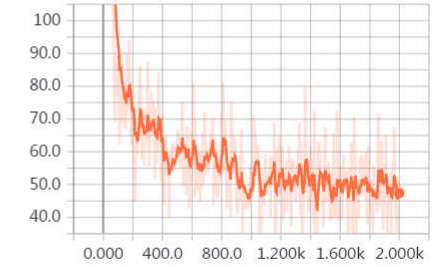
accuracy 1

accuracy/accuracy



cross_entroy 1

cross_entroy/cross_entropy



主要内容

- 1. TensorBoard简单介绍
- 2. 通过TensorBoard查找错误
- 3. 通过TensorBoard进行超参数搜索
- 4. 导入预训练模型

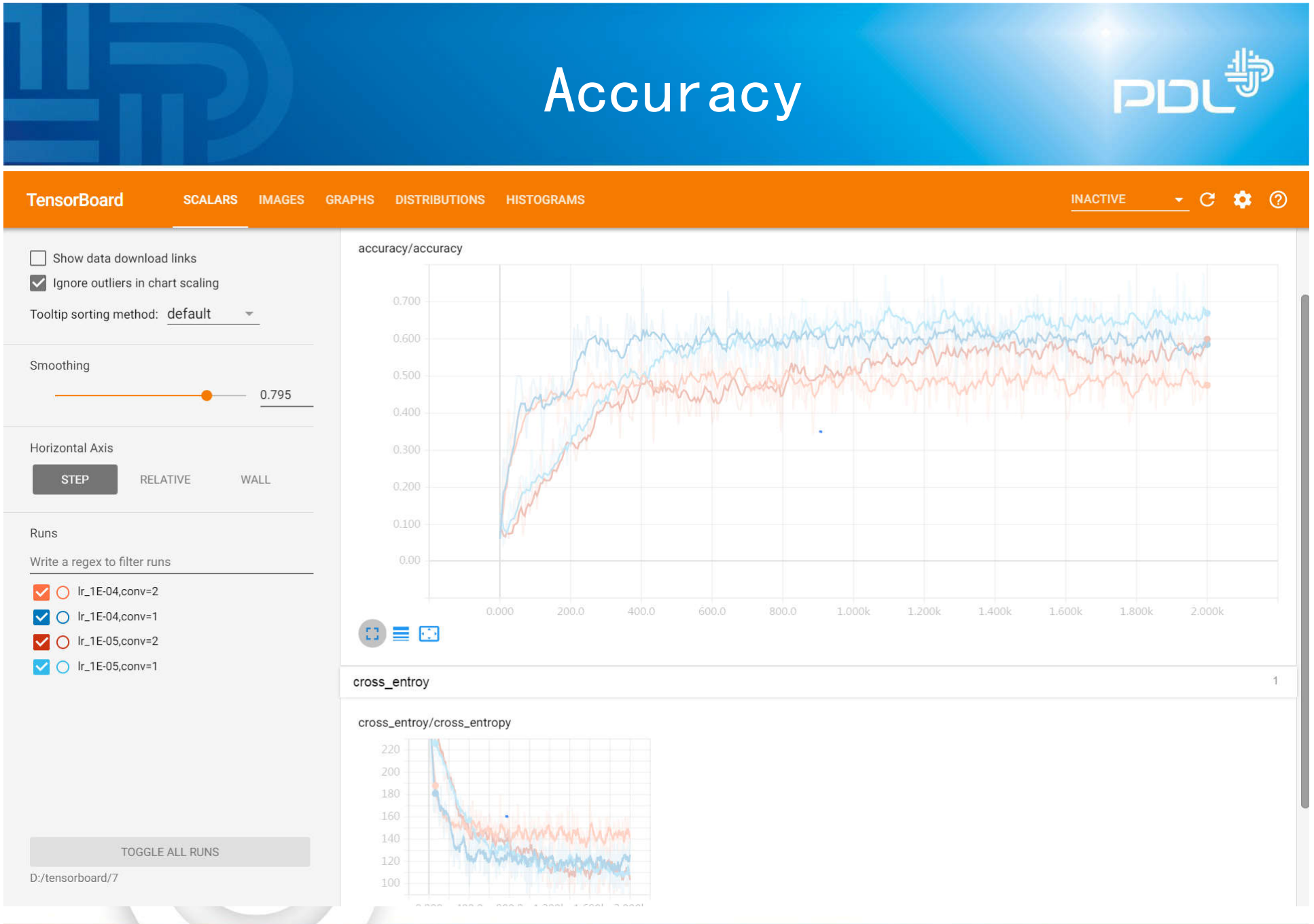
超参数搜索

- 不同学习率以及卷积层数

```
for learning_rate in [1E-4, 1E-5]:
    for use_two_conv in [True, False]:
            hparam = make_hparam_string(learning_rate, use_two_conv)
            print('Starting run for %s' % hparam)
            mnist_training(learning_rate, use_two_conv, hparam)
```

```
def make_hparam_string(learning_rate, use_two_conv):
  conv_param = "conv=2" if use_two_conv else "conv=1"
  return "lr_%.0E,%s" % (learning_rate, conv_param)
```

Accuracy


TensorBoard
SCALARS
IMAGES
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
INACTIVE
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.795
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
lr_1E-04,conv=2
lr_1E-04,conv=1
lr_1E-05,conv=2
lr_1E-05,conv=1
TOGGLE ALL RUNS
D:/tensorboard/7
accuracy/accuracy
0.700
0.600
0.500
0.400
0.300
0.200
0.100
0.00
0.000
200.0
400.0
600.0
800.0
1.000k
1.200k
1.400k
1.600k
1.800k
2.000k
cross_entroy
1
cross_entroy/cross_entropy
220
200
180
160
140
120
100

TensorBoard SCALARS IMAGES GRAPHS DISTRIBUTIONS HISTOGRAMS INACTIVE

Horizontal axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

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

TOGGLE ALL RUNS

D:/tensorboard/7

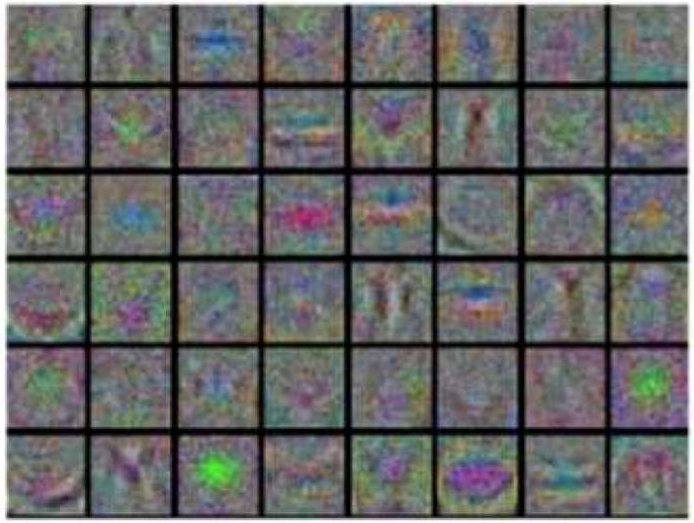
Filter tags (regular expressions supported)

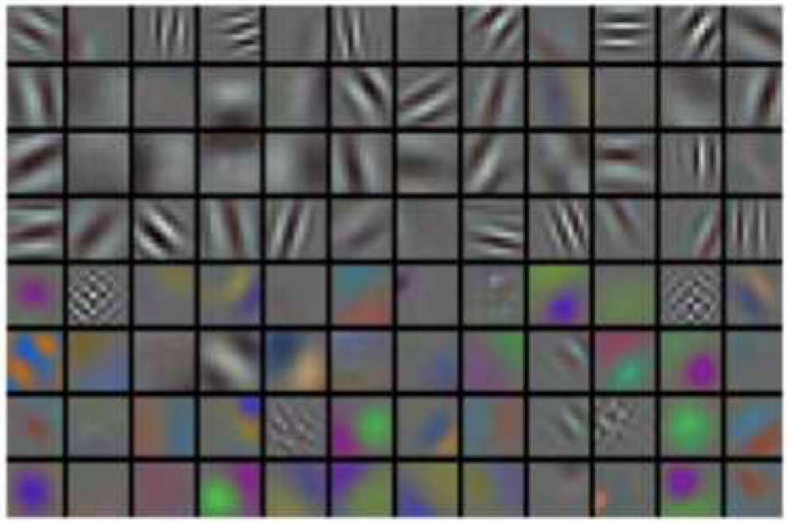
conv1 12

卷机核参数、梯度、特征图可视化方法

- 卷积核：滤波器、模式相关性
  - 有意义的卷积核：有一定规律的pattern，不是特别随机，特别是底层
  - 不同层的模式规律不同
  - 参数的泛化性能？冗余？稀疏？

- 举例：下图两张图都是将一个神经网络的第一个卷机层的filter weight可视化出来的效果图，左图存在很多噪点，右图则比较平滑，出现左图这个情形，往往意味着我们模型训练过程出现了问题。



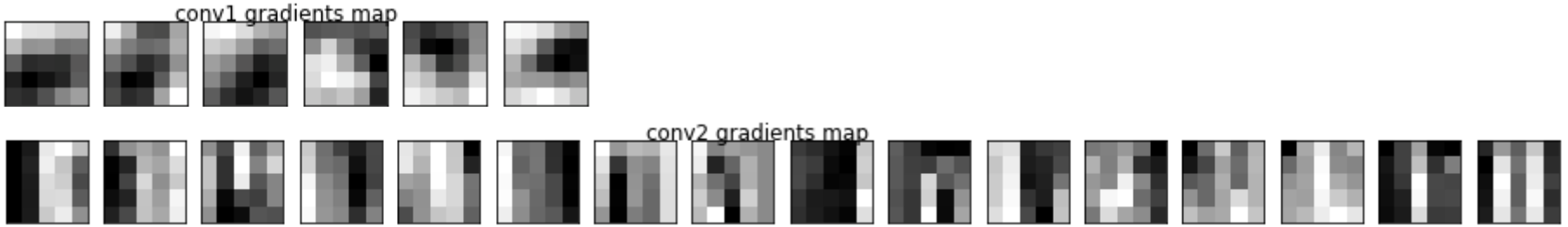


- 梯度可视化对网络调参的好处

在训练过程中，由于设置了较高的学习率，学习跨度太大，中间层的梯度可能会随着训练过程的推进逐渐变为0。

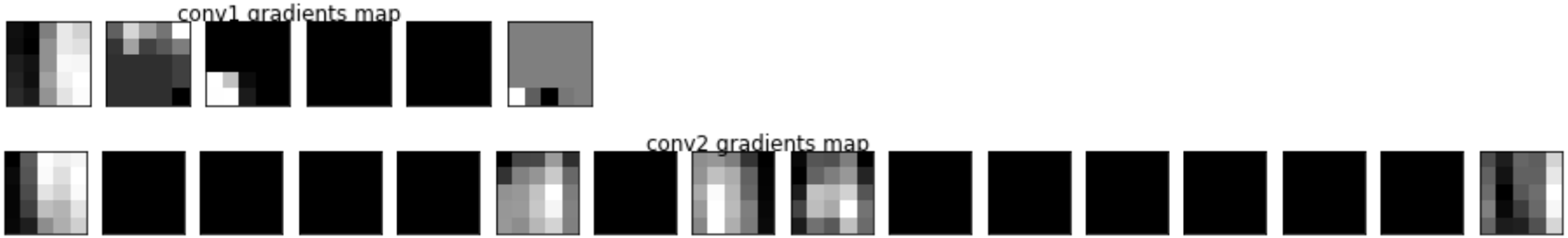
具体事例参照：example1.ipynb

Iter0: (lr=0.1)



Iter1: (lr=0.1)

Iter2: (lr=0.1)



Iter4: (lr=0.1)

- 梯度消失

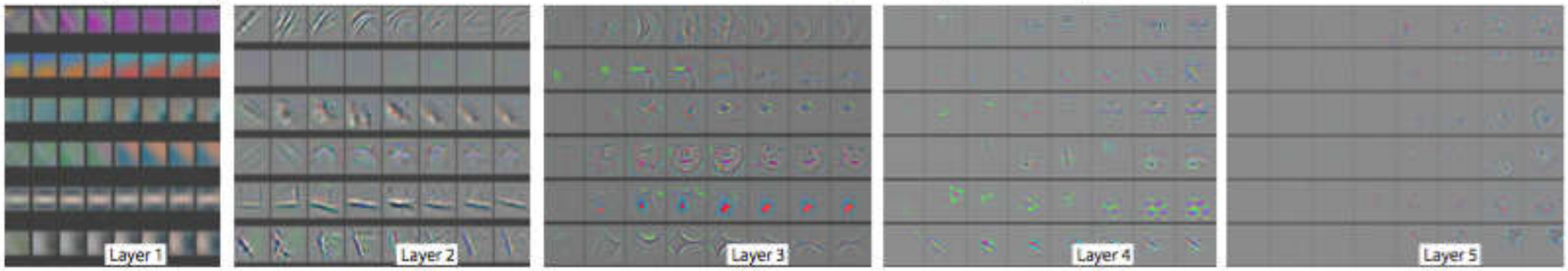
当出现梯度消失或梯度弥散时，梯度图能很好的表现出来。

具体事例参照：example2.ipynb

- 提取特征是否有良好的特点？
- 从图像反映：
  - 1. 任务相关性：例如分类，是否提取的是目标的主要特点
  - 2. 特征冗余？网络没训好？；稀疏？抗噪强？
  - 3. 特征的稳定性？其他样本也能提取出相似的结果？
  - 4. 不同层特征的目标是否实现？
  - 。。。

➢ Featuremap可视化对网络调参的好处

在网络训练过程中，每一层学习到的特征是怎么变化的，上面每一整张图片是网络的某一层特征图，然后每一行有8个小图片，分别表示网络epochs次数为：1、2、5、10、20、30、40、64的特征图：



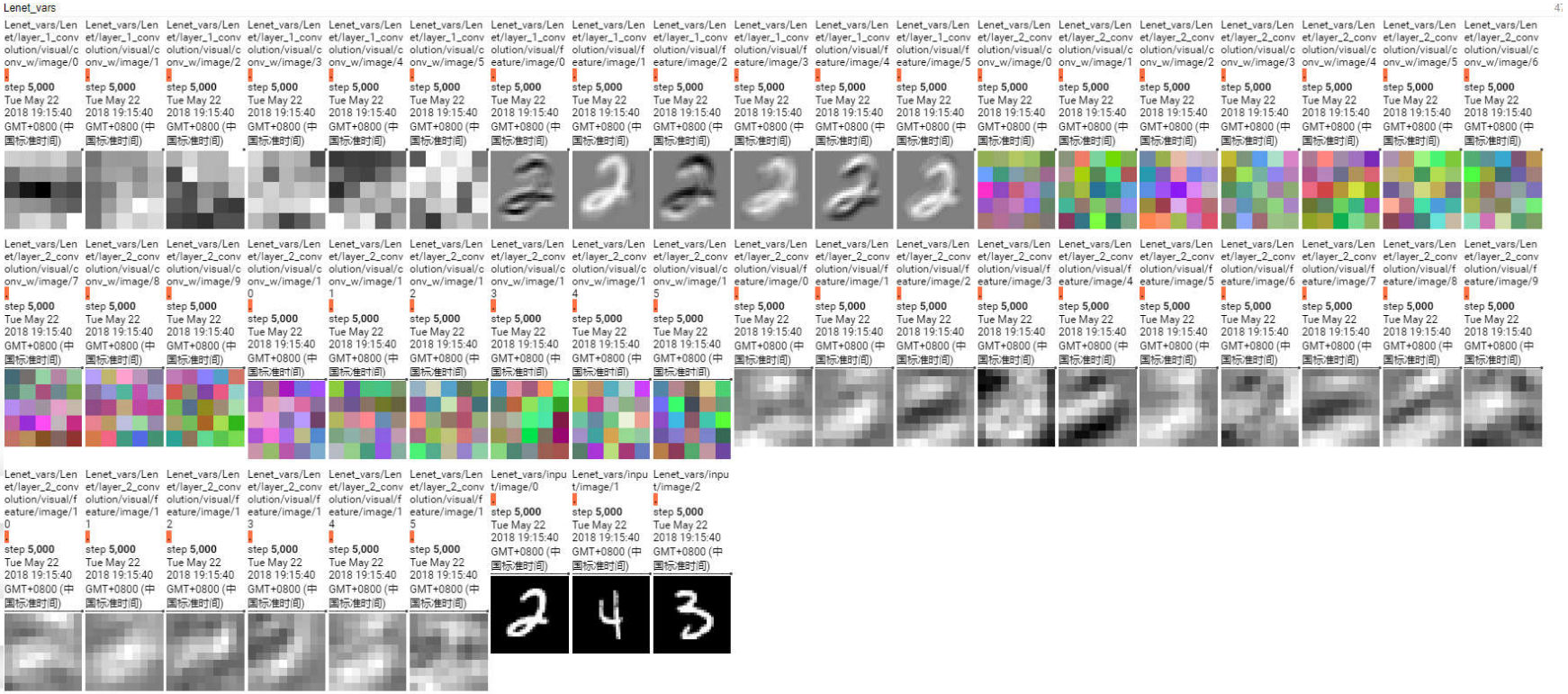
- 卷积核、特征图等可视化调参
  - 1. 哪层的特征出些突然变化？
  - 2. 不同层的变化速率？哪些层一直没动？
  - 3. 哪些是开始不动，后面有显著变化？

- Featuremap变化

正常的学习率：0.001，可以看出学习到了一些明显的边缘特征

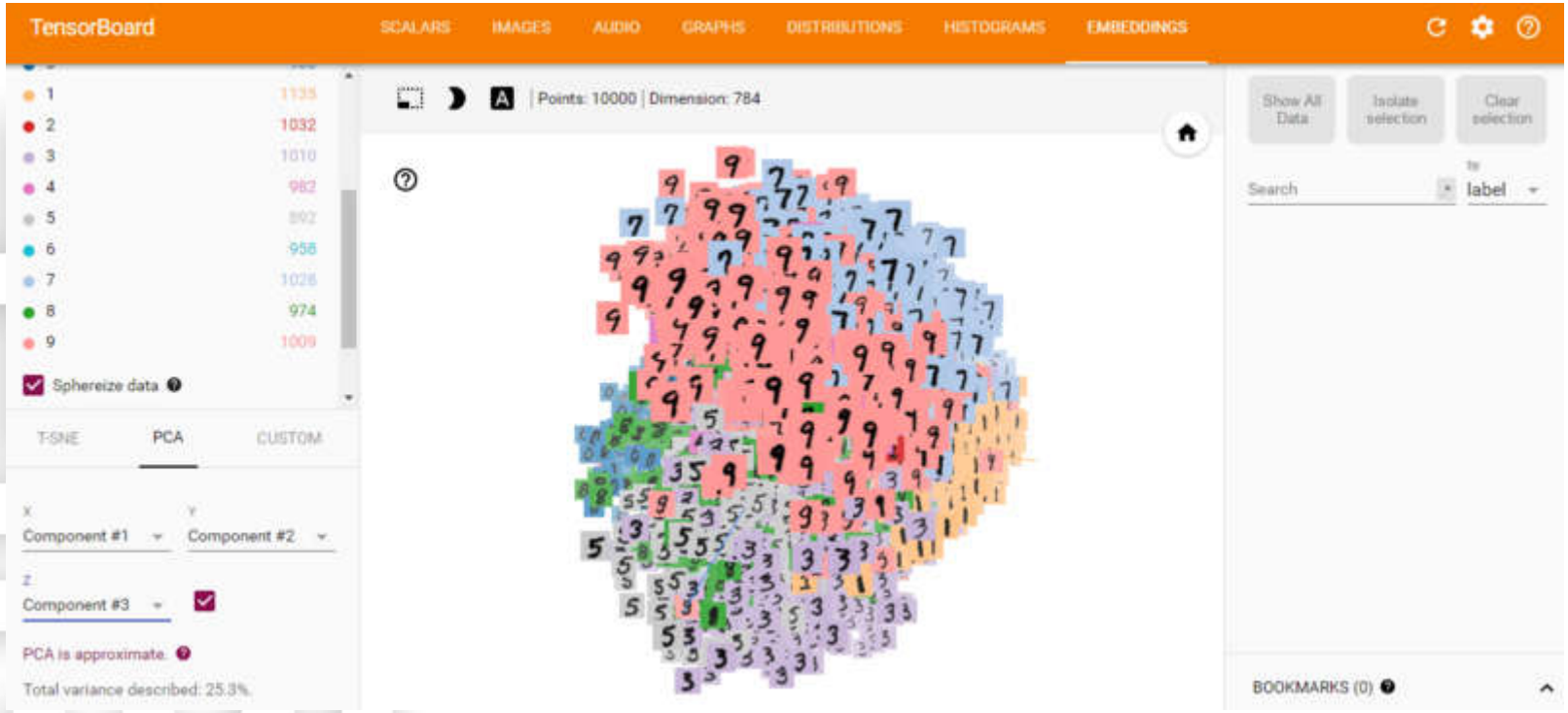
过高的学习率：0.1， 导致了零激活图像

TB参数、特征、图像可视化

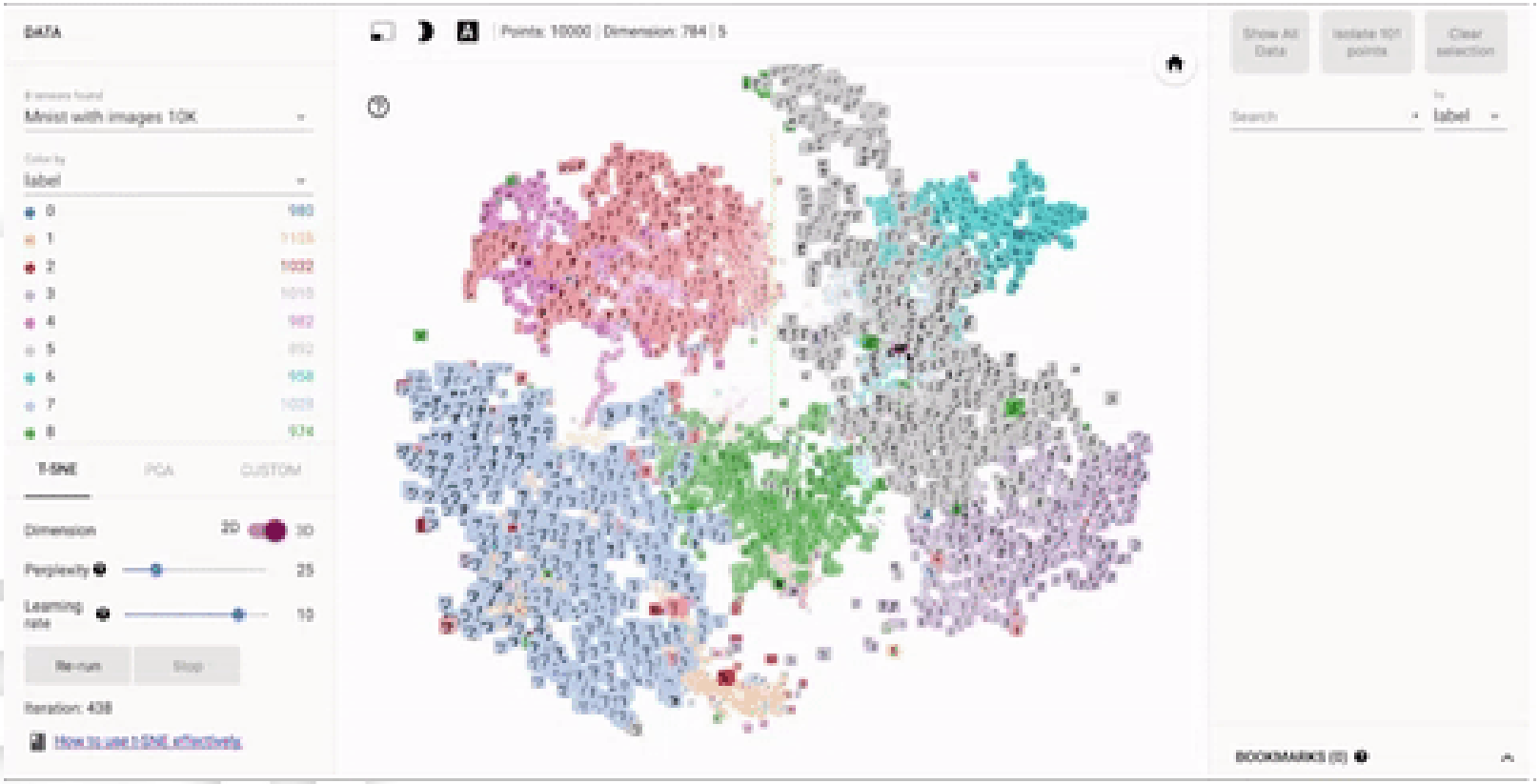


数据分布可视化

- TensorBoard Embedding



数据分布可视化


DATA
Points: 10000 | Dimension: 784 | 5
Mnist with images 10K
label
0
1
2
3
4
5
6
7
8
T-SNE
PCA
CUSTOM
Dimension
2D
3D
Perplexity
25
Learning rate
10
Re-run
Stop
Iteration: 438
How to use t-SNE effectively
Show All Data
Isolate 101 points
Clear selection
Search
label
BOOKMARKS (0)

数据分布可视化

- 动机：查看学习出的数据特征，是否满足我们期望的分布要求
  - 好的特征：同类的数据分布，有一定的相似性，关系紧密
  - 流型特征：数据中暗含的流型关系是否被很好的学习？例如是否被很好的映射在一个环或者一个曲线上

数据分布可视化

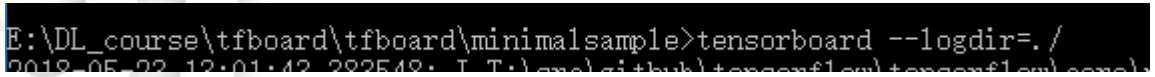
- 高维特征数据如何可视化？->2D，3D
- Embeding： 保持结构的单射,不改变原始数据的结构信息
  - PCA
  - t-SNE

数据分布可视化

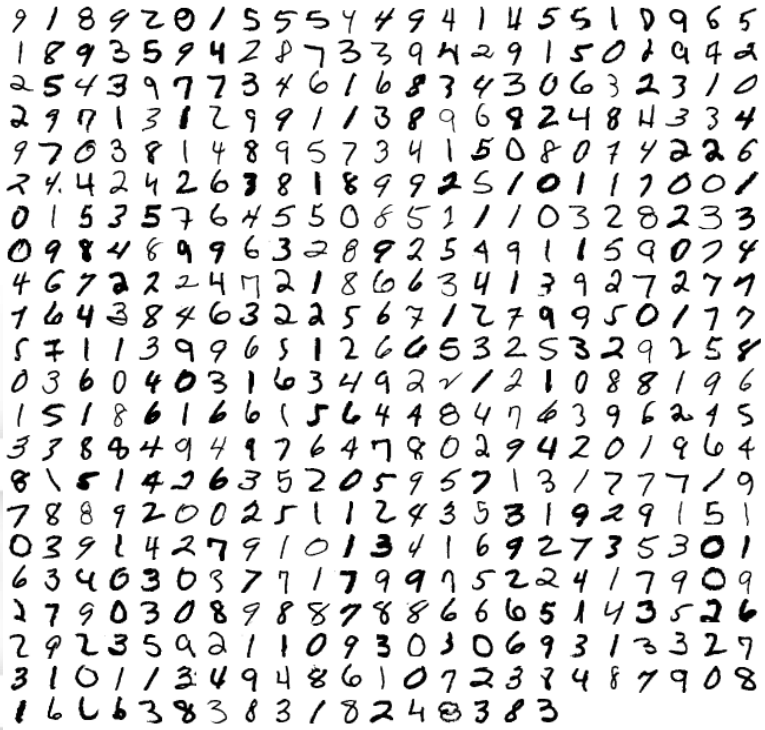
- Tensorboard: embedding projector
- 关键产生两个文件：

```
path_for_mnist_sprites =  os.path.join(LOG_DIR,'mnistdigits.png')
path_for_mnist_metadata =  os.path.join(LOG_DIR,'metadata.tsv')
```

- Sprites:要显示的样本图标文件
- Meta:样本序号和类别号对应文件
- 启动：注意windows下相对路径问题，命令行需进入log所在文件夹“tensorboard –logdir=./”



数据分布可视化



500个样本的sprites图标文件：
mnistdigits.png

```
Index   Label
0       9
1       1
2       8
3       9
4       2
5       0
6       1
7       5
8       5
9       5
10      4
11      4
12      9
13      4
14      1
15      4
16      5
17      5
18      1
19      0
20      9
21      6
22      5
23      1
24      8
25      9
26      3
```

500个样本的meta文件：
metadata.tsv

数据分布可视化

- TF中构建方法：
- 1. 提取可视化数据：例如：

```
mnist = input_data.read_data_sets("MNIST_data/", one_hot=False)
batch_xs, batch_ys = mnist.train.next_batch(TO_EMBED_COUNT)
```

- 2. 建立embeddings var，对应步骤1需要可视化的数据，重点是提供embedding的变量名

```
embedding_var = tf.Variable(batch_xs, name=NAME_TO_VISUALISE_VARIABLE)
summary_writer = tf.summary.FileWriter(LOG_DIR)
```

数据分布可视化

- 3.建立 embedding projector：指定想要可视化的 variable，metadata 文件的位置

```
config = projector.ProjectorConfig()
embedding = config.embeddings.add()
embedding.tensor_name = embedding_var.name

# Specify where you find the metadata
embedding.metadata_path = path_for_mnist_metadata #'metadata.tsv'

# Specify where you find the sprite (we will create this later)
embedding.sprite.image_path = path_for_mnist_sprites #'mnistdigits.png'
embedding.sprite.single_image_dim.extend([28,28])

# Say that you want to visualise the embeddings
projector.visualize_embeddings(summary_writer, config)
```

数据分布可视化

- 4. TB从模型存储文件中读取embedding 变量值，故需要一段时间存储一次模型：

```
all_vars = tf.global_variables()
saver = tf.train.Saver(all_vars)
```

- 5. 创建sprites 图标文件、meta文件

```
def create_sprite_image(images):
    """Returns a sprite image consis
```

```
with open(path_for_mnist_metadata,'w') as f:
    f.write("Index\tLabel\n")
    for index,label in enumerate(batch_ys):
        f.write("%d\t%d\n" % (index,label))
```