tensorboard教程

网上教程，一大堆人瞎BB。东拉西扯，净是些没用的。我自己总结一下真正的数据图教程。

# 1.数据流图(graph)

数据流，就是tensor以及tensor之间的连接关系等。实际上，“会话”运行之前，“会话”本身就需要，**建立河流**，将数据之河，流过的每一个源头，干流，支流连接在一起。它就存储在”会话”的graph属性中(sess.graph)。我们将这个graph 导出来，再用tensorboard显示出来，使用即可。它需要三个步骤：

1. **在tensorflow中用tf.summary.FileWriter()将graph保存；**
2. **用tensorboard将graph解析成网页；**
3. **用网页浏览器，观看graph。**

## 加法器数据流图：

1. **保存graph代码：**

import tensorflow as tf

a = tf.constant(2, name="a")

b = tf.constant(3, name="b")

x = tf.add(a, b, name="add")

with tf.Session() as sess:

writer = tf.summary.FileWriter('./graphs', sess.graph)# 运行之前，将会话（sess）的graph写入本地

sess.run(x)

writer.close()

\*:实际上，writer在sess.run()之前还同之后启动，都是可以的。但是考虑到，有可能run()中是一个循环，出不来，所以writer放sess.run()之前更好。

1. **解析graph：**

在终端运行：

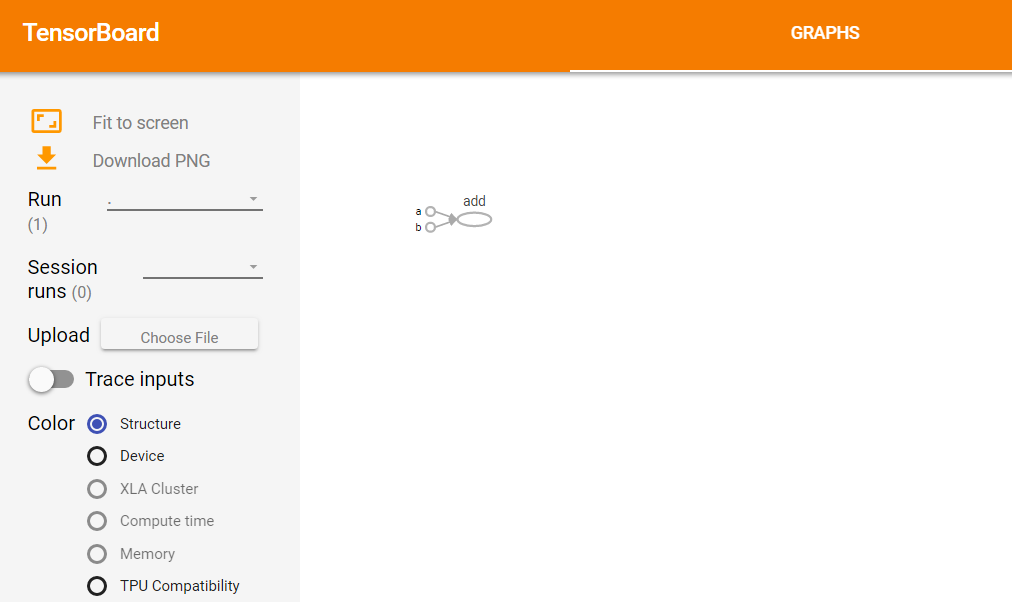
tensorboard --logdir="./graphs"

1. **网页显示**

在浏览器打开上一步提示的网址：

比如：http://127.0.0.1:6006

然后，我们就能看到，如下数据图：



## CNN网络数据流图：

1. **源码：**

网络结构代码：

self.input\_states = tf.placeholder(# 看到placeholder就应该想到，这个变量肯定是输入了。

tf.float32, shape=[None, 4, board\_height, board\_width]) # None表示不定，即，第一维不定。

self.input\_states\_reshaped = tf.reshape(

self.input\_states, [-1, board\_height, board\_width, 4]) # -1表示自动计算。

self.conv1 = tf.layers.conv2d(inputs=self.input\_states\_reshaped,

filters=32, kernel\_size=[3, 3],

padding="same", activation=tf.nn.relu)

self.conv2 = tf.layers.conv2d(inputs=self.conv1, filters=64,

kernel\_size=[3, 3], padding="same",

activation=tf.nn.relu)#卷积核，尺寸: 3 x 3，64个，补0，激励函数relu。

self.conv3 = tf.layers.conv2d(inputs=self.conv2, filters=128,

kernel\_size=[3, 3], padding="same",

activation=tf.nn.relu)#卷积核，尺寸: 3 x 3，128个，补0，激励函数relu。

self.action\_conv = tf.layers.conv2d(inputs=self.conv3, filters=4,

kernel\_size=[1, 1], padding="same",

activation=tf.nn.relu)#卷积核，尺寸: 1 x 1，4个，补0，激励函数relu。

self.action\_conv\_flat = tf.reshape(

self.action\_conv, [-1, 4 \* board\_height \* board\_width])

self.action\_fc = tf.layers.dense(inputs=self.action\_conv\_flat,

units=board\_height \* board\_width,

activation=tf.nn.log\_softmax)

self.evaluation\_conv = tf.layers.conv2d(inputs=self.conv3, filters=2,# 它的输入是卷积层conv3，不是落子网络

kernel\_size=[1, 1],

padding="same",

activation=tf.nn.relu)# 核：1 X 1，2个，补0，激励函数relu。

self.evaluation\_conv\_flat = tf.reshape(

self.evaluation\_conv, [-1, 2 \* board\_height \* board\_width])

self.evaluation\_fc1 = tf.layers.dense(inputs=self.evaluation\_conv\_flat,

units=64, activation=tf.nn.relu)

self.evaluation\_fc2 = tf.layers.dense(inputs=self.evaluation\_fc1,

units=1, activation=tf.nn.tanh)

self.labels = tf.placeholder(tf.float32, shape=[None, 1])

self.value\_loss = tf.losses.mean\_squared\_error(self.labels,

self.evaluation\_fc2)# 损失函数就简单了，调用mean\_squared\_error把两个值加入就行

self.mcts\_probs = tf.placeholder(

tf.float32, shape=[None, board\_height \* board\_width])

self.policy\_loss = tf.negative(tf.reduce\_mean(#reduce\_mean()：计算均值；negative(): 取负值。

tf.reduce\_sum(tf.multiply(self.mcts\_probs, self.action\_fc), 1)))

保存graph代码：

writer = tf.summary.FileWriter('./graphs', self.session.graph)# 运行之前，将会话（sess）的graph写入本地

log\_act\_probs, value = self.session.run(

[self.action\_fc, self.evaluation\_fc2],# 想只前向传播，那么，数据出口选落子与策略网络口就行了。

feed\_dict={self.input\_states: state\_batch}

)

act\_probs = np.exp(log\_act\_probs)

writer.close()

1. **解析graph：**

在终端运行：

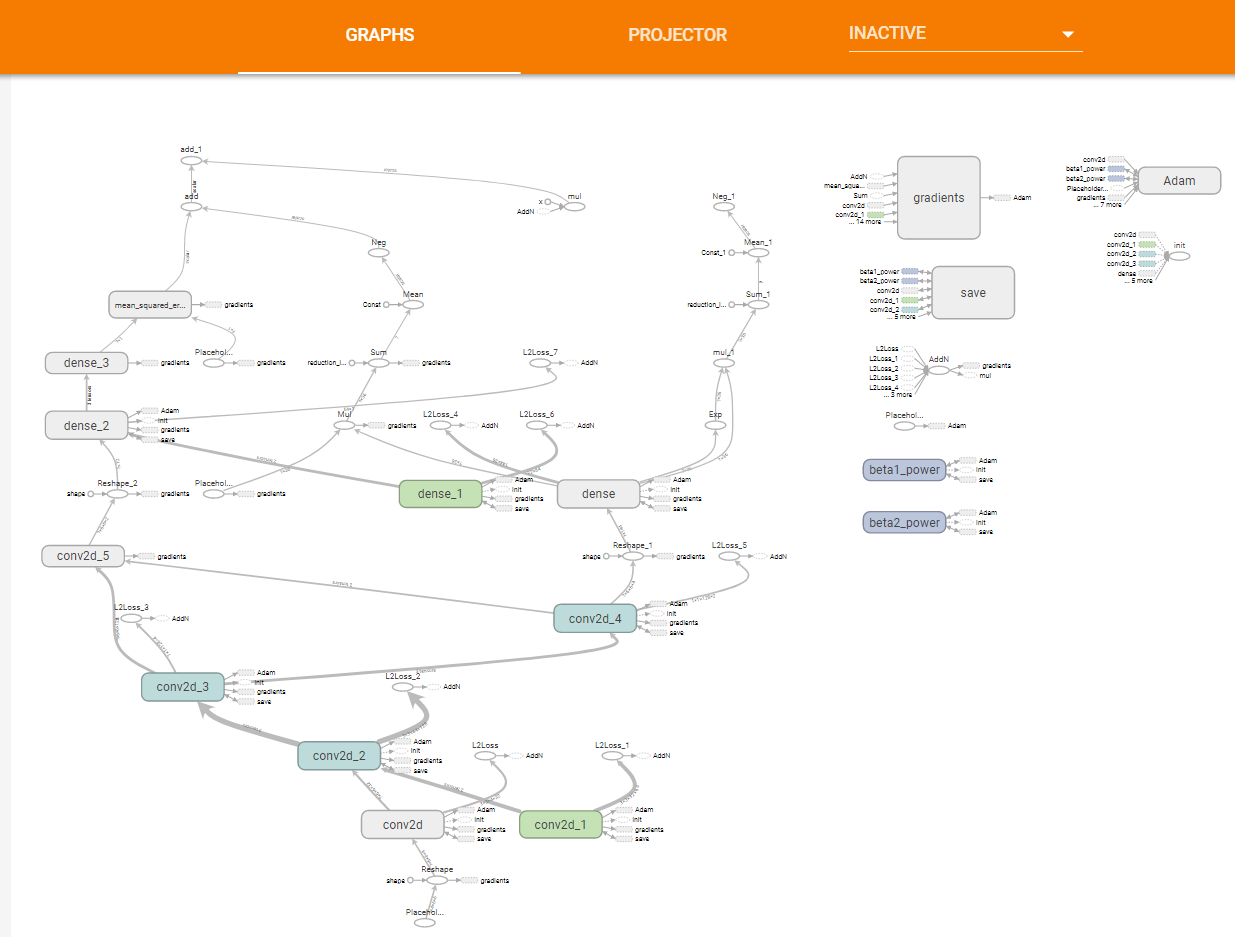
tensorboard --logdir="./graphs"

1. **网页显示**

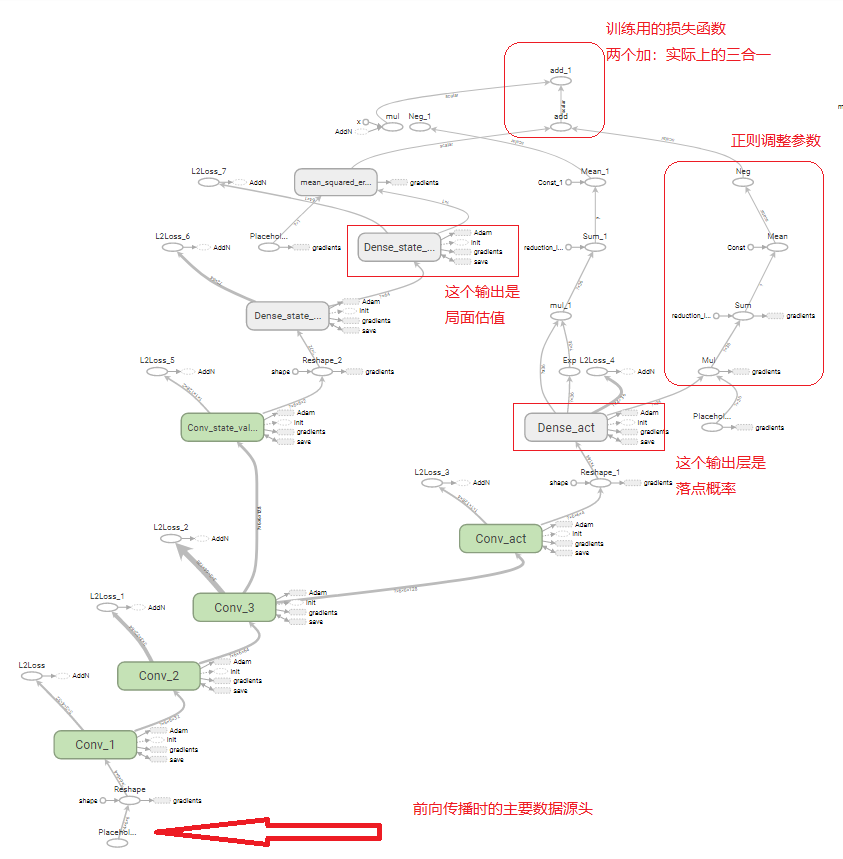
在浏览器打开上一步提示的网址：

比如：http://127.0.0.1:6006

然后，我们就能看到，如下数据图：



我在源码中，给他们改个名字，如下：



**这样的图，用来分析，不错吧！！！**

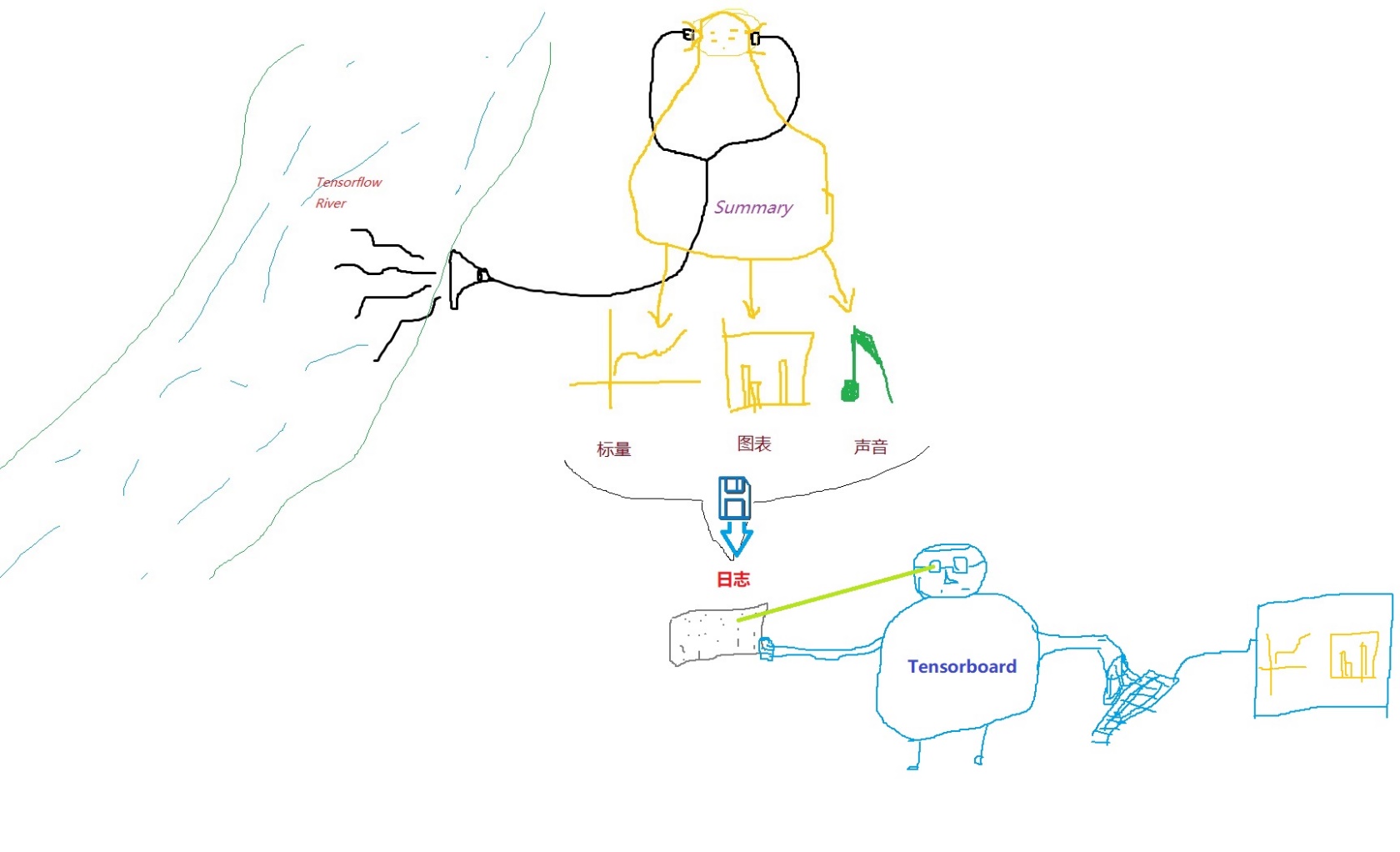
【补充：】

1. 实线表示数据依赖(tensor在运算符之间的流动关系)，而虚线表示控制依赖关系，
2. TensorBoard Graph的上下方位代表了数据依赖的方向：数据总是从下方的节点流向上方的节点，上方节点依赖于下方节点;

# 2.画’总结’图(summary)

tensorflow提供了非常好的tensorflow数据流图的分析，总结，显示工具给我们，让我们能直观地追踪，数据这条河流的每一个小浪花。这真真是极好，用过的都知道（当然，前提是你会用，否则，你只能回到从前，存log文件去了）。

summary包就是这是这套分析，总结工具。基本架构是：由summary将“监听”的数据写入日志文件（当然，写日志，也要靠summary），然后由tensorboard将日志解析出来，通过网页，呈现给观众。而且，summary的监听，写文件过程，是异步的，基本不会影响tensorflow本身的运行。tensorflow，summary，tensorboard三者的关系，基本如下图：



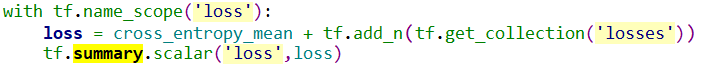
（领会精神，领会精神）

## 1. 标量(scalar)

数值，像**学习率**，**损失函数**用scalar函数：

tf.summary.scalar(name,tensor)

例如：

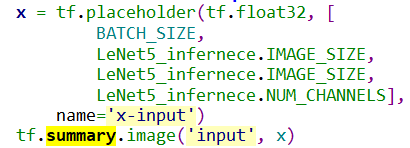


## 2. 图片(image)

如果河流里面有图片的话，也是可以显示的哦，函数：

tf.summary.image(name,tensor)

例如：

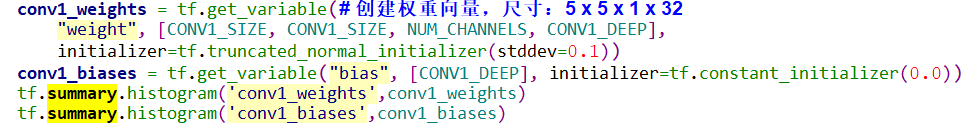


## 3. 直方图(histogram)

各层网络权重，偏置的分布，用histogram\_summary函数：

tf.summary.histogram(name,tensor)

例如：

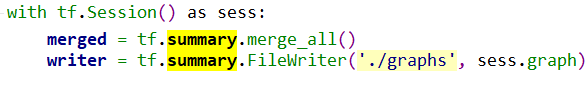


## 收集保存这些数据

虽然我们到处加监听器，但是却不用一一去保存它们。summary提供了一个接口，合并保存所有summary日志。

函数：tf.summary.merge\_all()

例如：



\*：上图第二句话是保存数据流图，非必要。

## 启动运行写日志【重】

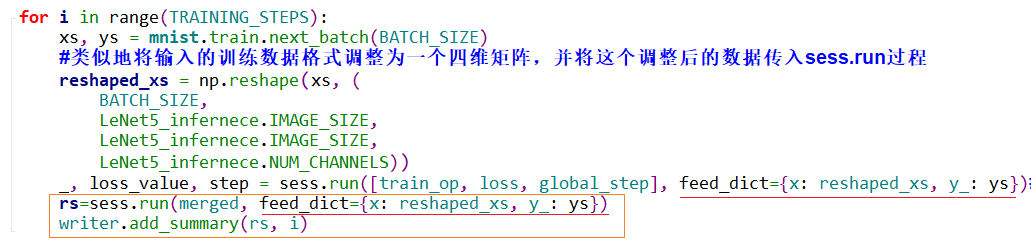
这一步千万不要忘了。我们使用的是tensorflow，它的河流是要启动后，才会流滴，我们的summary也要单独启动它，然后再写入日志文件。这样有一个好处是，我们可以记录那种，对训练没有用处，但却是我们想要看到的数据用summary来计算。（比如正确率）

函数：

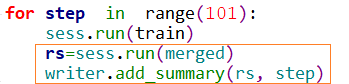
result=sess.run(merged);

writer.add\_summary(result, step)

例如：



\*：上面的例子与示例函数相比，有点不同。再看下面这个例子：



两者结合一下，你明白了吧？summary的sess.run()后面必须携带与**数据流**网络一致的填充字典（河有它就有，河没它就没），少了它，你会一直收到如下错误：

tensorflow.python.framework.errors\_impl.InvalidArgumentError: <exception str() failed>

**【日他仙人板板，没有一个地方有说过这件事，劳资搞清楚这个事情浪费了半天】**

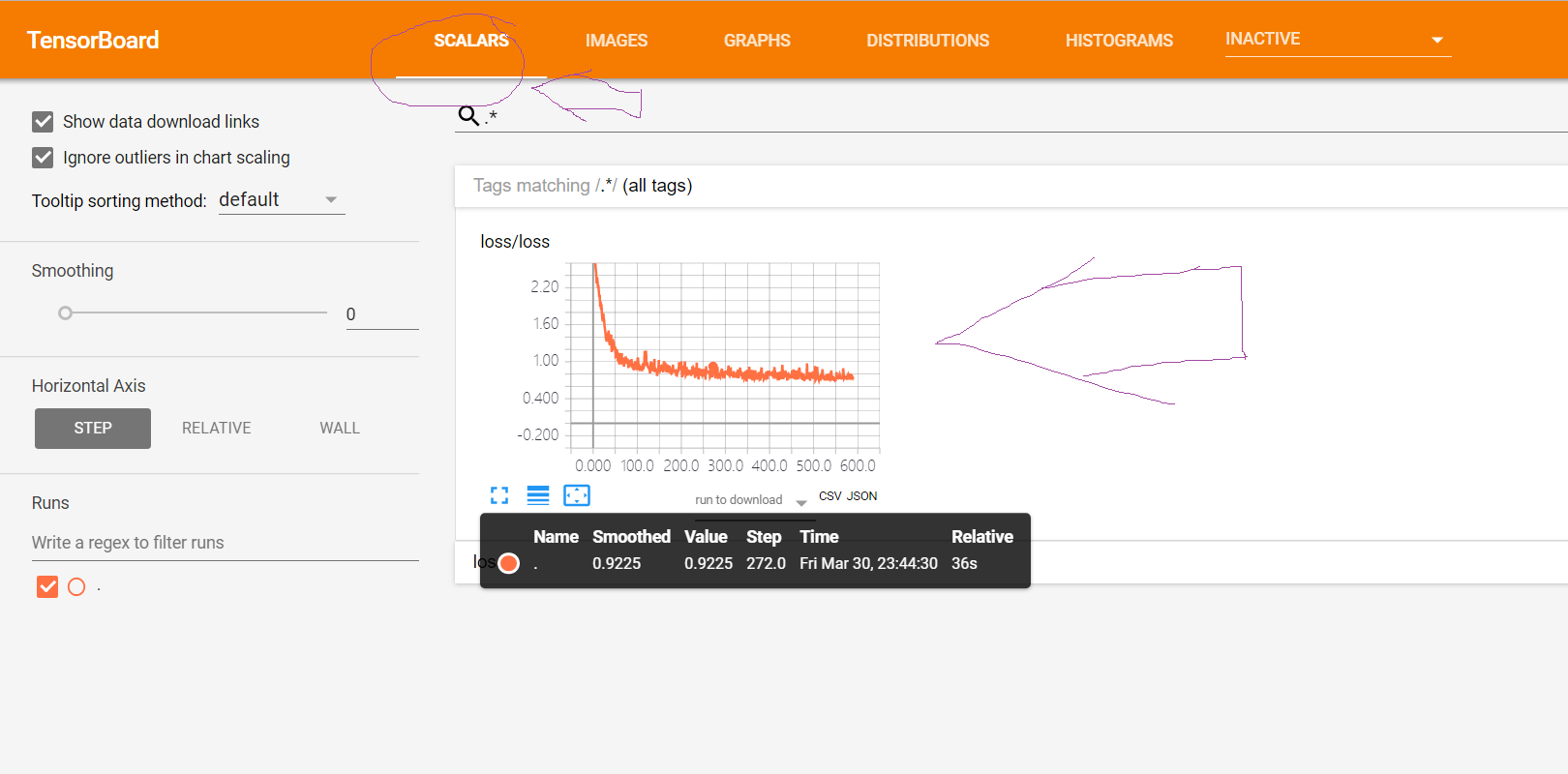
## 网页上查看成果

说了这么久，终于可以看到结果了，激动吧。别忙，先打开tensorboard：

tensorboard --logdir="./graphs"

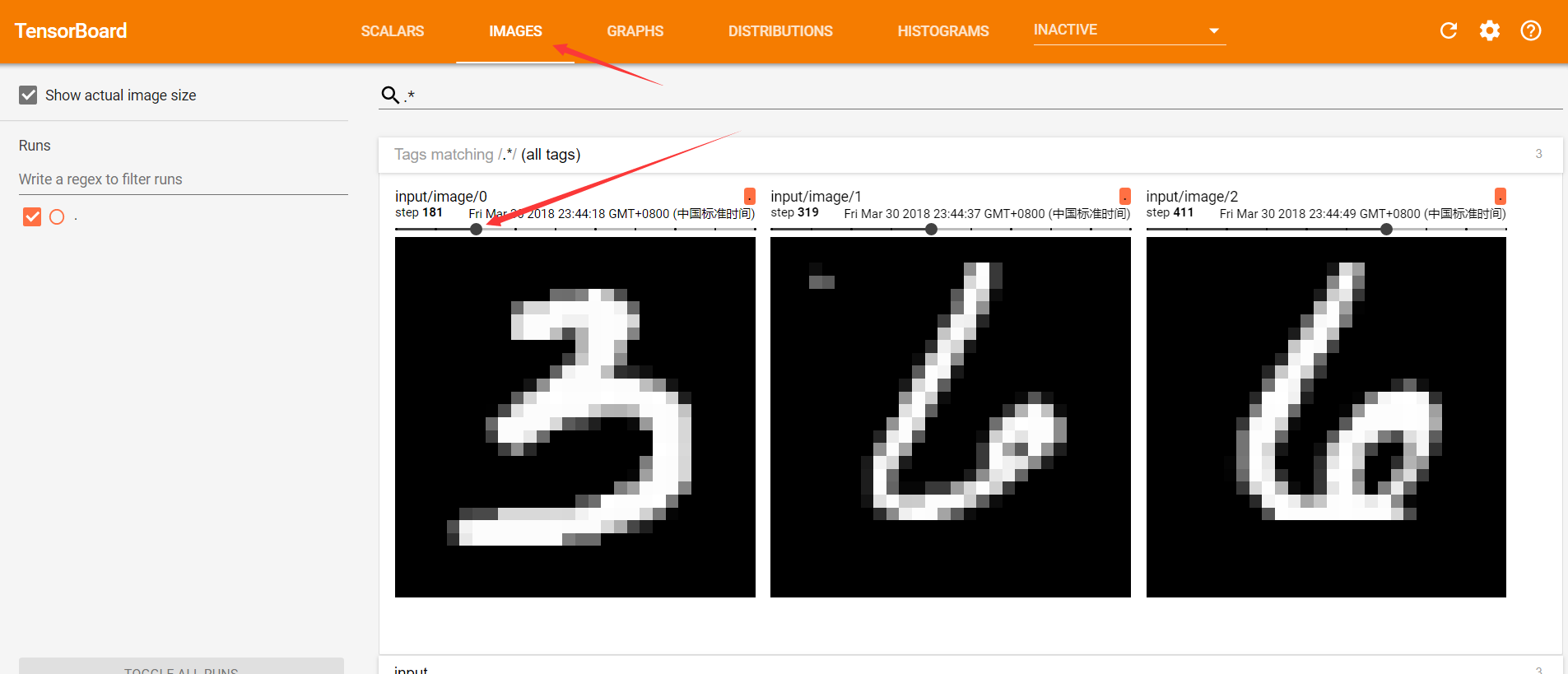
然后，打开网页，一个一个地看：

### **scalar**：



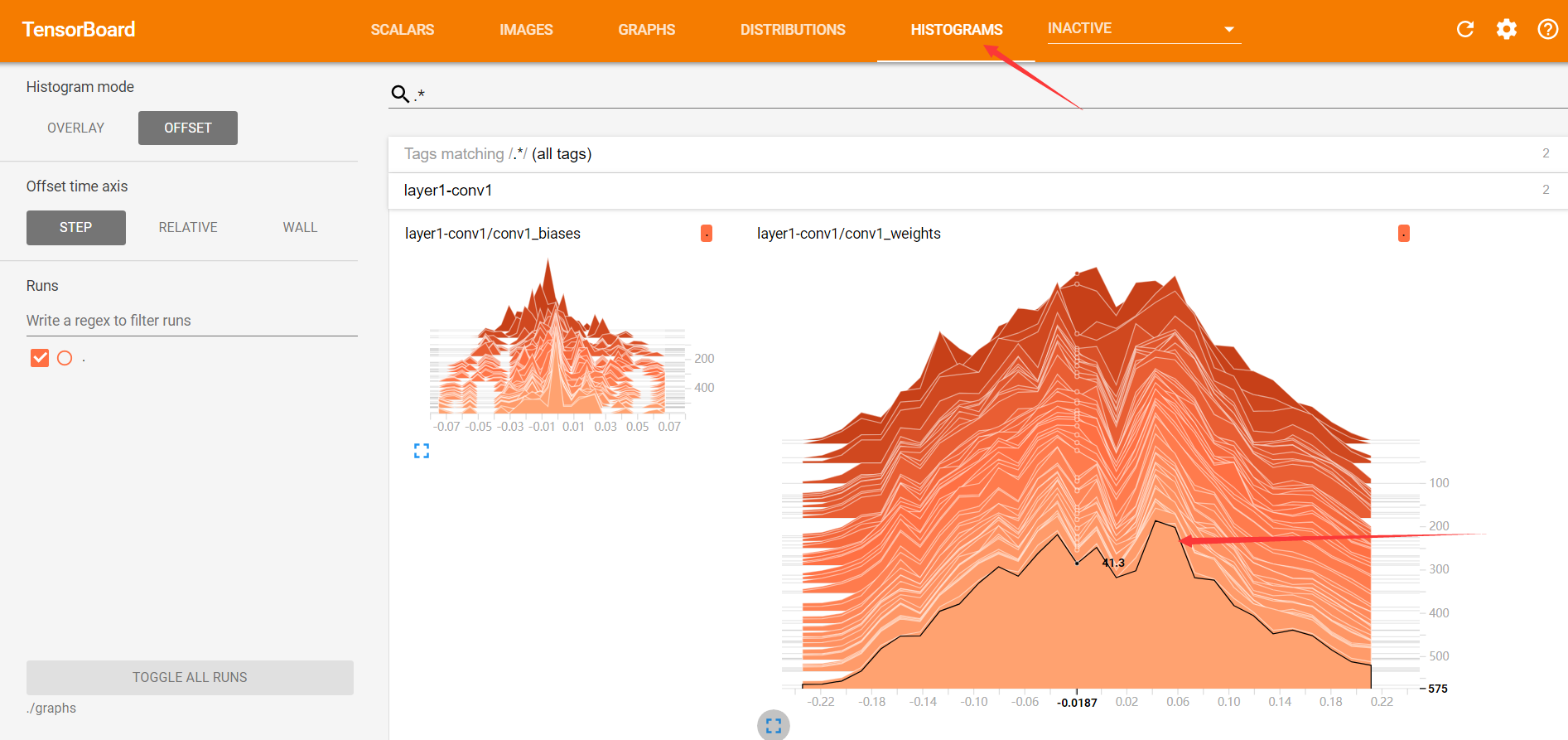
上面就是我们的loss，可以看出，大概训练50轮后，loss就低于1了。鼠标还可以在图上滑动，而且这个图是实时的，如果此时，一直在训练的话，还会自动更新哦。

### image:



如上图，第二个箭头还可以滑动，看不同的step时的图片。当然，我现在还是不太明白，为什么有三个通道，而且它的step图片是不连续的（这个可能是写文件来不及，有丢失吧）。

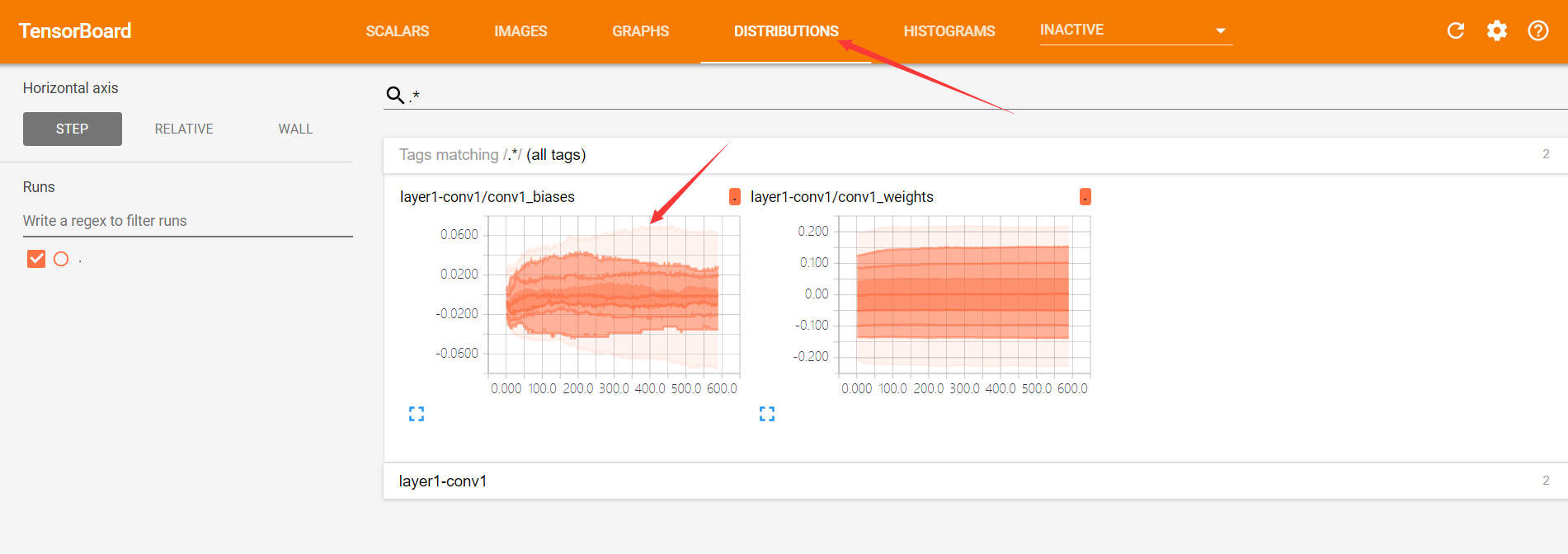
### histogram:



每一个step的weights, biases的正态分步图。

### distributions:

我们明明只设置了三种图，但是却得到四种图（双十一，买一送一），这多出来的图，叫**分布图**。其实和histogram差不多，不过是它的二维表示法。



# 3. 一个图两根线

一句话总结：两个目录，两个FileWriter。

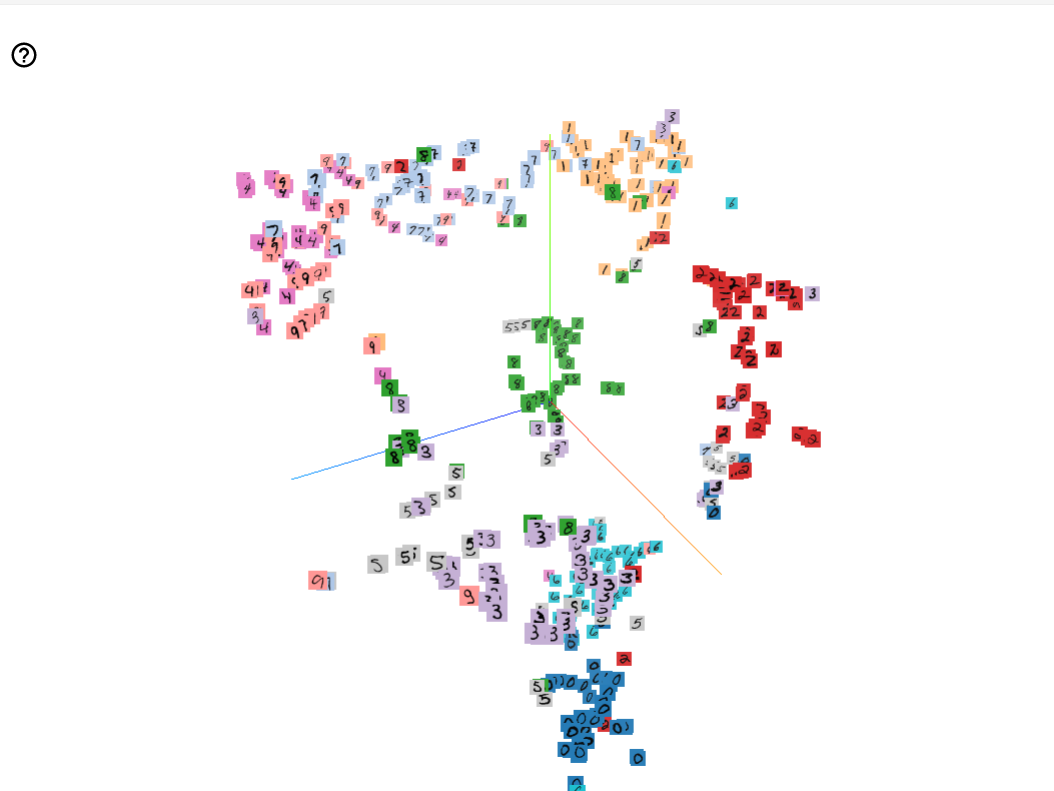
代码例子：



\*：两个目录，两个writer一样不能少，否则还是一根线。

# 4. Embedding Projector（嵌入式投影仪）

就是能实时通过2D/3D观察训练过程中的数据的图，如下图是MNIST分类实时图：



看起来好看，实际上操作起来很麻烦，而且最终结果，也只是看起来好看而已。

步骤：

* 1. 保存projector:

projector.visualize\_embeddings(summary\_writer, config)

* 1. 制作保存spirit image:
  2. 制作保存meta data:

绣花枕头。