

TensorBoard是Tensorflow自带的一个强大的可视化工具，也是一个web应用程序套件。在众多机器学习库中，Tensorflow是目前唯一自带可视化工具的库，这也是Tensorflow的一个优点。学会使用TensorBoard，将可以帮助我们构建复杂模型。

这里需要理解“可视化”的意义。“可视化”也叫做数据可视化。是关于数据之视觉表现形式的研究。这种数据的视觉表现形式被定义为一种以某种概要形式抽提出来的信息，包括相应信息单位的各种属性和变量。例如我们需要可视化算法运行的错误率，那么我们可以取算法每次训练的错误率，绘制成折线图或曲线图，来表达训练过程中错误率的变化。可视化的方法有很多种。但无论哪一种，均是对数据进行摘要(summary)与处理。

通常使用TensorBoard有三个步骤：

- 首先需要在需要可视化的相关部位添加可视化代码，即创建摘要、添加摘要；
- 其次运行代码，可以生成了一个或多个事件文件(event files)；
- 最后启动TensorBoard的Web服务器。

完成以上三个步骤，就可以在浏览器中可视化结果，Web服务器将会分析这个事件文件中的内容，并在浏览器中将结果绘制出来。

如果我们已经拥有了一个事件文件，也可以直接利用TensorBoard查看这个事件文件中的摘要。

1. 创建事件文件

在使用TensorBoard的第一个步骤中是添加可视化代码，即创建一个事件文件。创建事件文件需要使用 `tf.summary.FileWriter()`。具体如下：

```
tf.summary.FileWriter(  
    logdir, # 事件文件目录  
    graph=None, # `Graph`对象  
    max_queue=10, # 待处理事件和摘要的队列大小  
    flush_secs=120, # 多少秒将待处理的事件写入磁盘  
    graph_def=None, # [弃用参数]使用graph代替  
    filename_suffix=None) # 事件文件的后缀名
```

`tf.summary.FileWriter()` 类提供了一个在指定目录创建了一个事件文件并向其中添加摘要和事件的机制。该类异步地更新文件内容，允许训练程序调用方法来直接从训练循环向文件添加数据，而不会减慢训练，即既可以在训练的同时更新事件文件以供TensorBoard实时可视化。

例如：

```
import tensorflow as tf

writer = tf.summary.FileWriter('./graphs', filename_suffix='.file')
writer.close()
```

运行上述代码将会在 `./graphs` 目录下生成一个后缀名是 `.file` 的文件，这个文件正是事件文件。

注意：由于这个 `writer` 没有做任何操作，所以这是一个空的事件文件。如果不写 `filename_suffix='.file'` 这个属性，那么默认不创建空的事件文件。这时候如果使用TensorBoard的web服务器运行这个事件文件，看不到可视化了什么，因为这是一个空的事件文件。

2. TensorBoard Web服务器

TensorBoard通过运行一个本地Web服务器，来监听6006端口。当浏览器发出请求时，分析训练时记录的数据，绘制训练过程中的图像。

TensorBoard的Web服务器启动的命令是：`tensorboard`，主要的参数如下：

```
--logdir # 指定一个或多个目录。tensorboard递归查找其中的事件文件。
--host # 设置可以访问tensorboard服务的主机
--port # 设置tensorboard服务的端口号
--purge_orphaned_data # 是否清除由于TensorBoard重新启动而可能已被修改的数据。禁用purge_orphaned_data可用于调试修改数据。
--nopurge_orphaned_data
--reload_interval # 后端加载新产生的数据的间隔
```

假如现在当前目录下 `graphs` 文件夹中存在事件文件，可以使用如下命令打开tensorboard web服务器：

```
tensorboard --logdir=./graphs
```

然后打开浏览器，进入 `localhost:6006` 即可查看到可视化的结果。

3. 可视化面板与操作详述

在Tensorflow中，summary模块用于可视化相关操作。TensorBoard目前内置8中可视化面板，即SCALARS、IMAGES、AUDIO、GRAPHS、DISTRIBUTIONS、HISTOGRAMS、EMBEDDINGS、TEXT。这8中可视化的主要功能如下。

- SCALARS：展示算法训练过程中的参数、准确率、代价等的变化情况。
- IMAGES：展示训练过程中记录的图像。
- AUDIO：展示训练过程中记录的音频。
- GRAPHS：展示模型的数据流图，以及训练在各个设备上消耗的内存与时间。
- DISTRIBUTIONS：展示训练过程中记录的数据的分布图。

- HISTOGRAMS: 展示训练过程中记录的数据的柱状图。
- EMBEDDINGS: 展示词向量（如Word2Vec）的投影分布。
- TEXT:

除此以外，TensorBoard还支持自定义可视化操作。可以参考[官方博客](#)。

3.1 GRAPHS

TensorBoard可以查看我们构建的数据流图，这对于复杂的模型的构建有很大的帮助。

创建graph摘要

将数据流图可视化，需要将图传入summary的writer对象中。即在初始化 `tf.summary.FileWriter()` 时，给 `graph` 参数传入一个图。也可以在writer对象初始化之后使用 `tf.add_graph()` 给writer添加一个图。

如下，我们构建一个简单的图，并可视化：

```
import tensorflow as tf

a = tf.constant(2)
b = tf.constant(3)
x = tf.add(a, b)

with tf.Session() as sess:
    writer = tf.summary.FileWriter('./graphs', sess.graph)
    print(sess.run(x))

writer.close()
```

在TensorBoard的GRAPHS中显示如下：



图中 `Const` 、 `Const_1` 、 `Add` 表示的是节点的名称。

小圆圈表示为一个常量(constant)。椭圆形表示一个节点(OpNode)。箭头是一个引用边(reference edge)。当我们给代码中的op重命名之后，

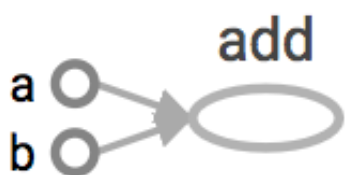
```
import tensorflow as tf

a = tf.constant(2, name='a')
b = tf.constant(3, name='b')
x = tf.add(a, b, name='add')

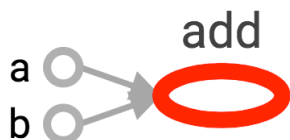
with tf.Session() as sess:
    writer = tf.summary.FileWriter('./graphs', sess.graph)
    print(sess.run(x))

writer.close() # 注意关闭fileWriter
```

在TensorBoard中显示如下：



点击一个节点，可以查看节点的详情，如下查看add节点的详情：



add
^

Operation: Add

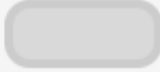





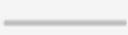
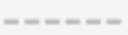

Attributes (1)
T {"type": "DT_INT32"}

Inputs (2)
○ a scalar
○ b scalar

Outputs (0)

Add to main graph

图中不仅仅有节点和箭头，tensorflow有很多的用于图可视化的图形。所有的图像包括命名空间、op节点、没有连接关系的节点组、有连接关系的节点组、常量、摘要、数据流边、控制依赖边、引用边。具体如下：

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

Unconnected Series

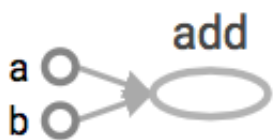
一系列的不相连接的操作相同且name相同（不包括系统给加的后缀）节点可以组成一个没有连接关系的节点组。

```
import tensorflow as tf

with tf.Graph().as_default() as graph:
    a = tf.constant(2, name='a')
    b = tf.constant(3, name='b')
    x = tf.add(a, b, name='add')
    y = tf.add(a, b, name='add')

with tf.Session() as sess:
    writer = tf.summary.FileWriter('./graphs')
    writer.add_graph(graph)
    writer.close()
```

可视化之后如下：



这两个子图可以组成节点组，如下：



Connected Series

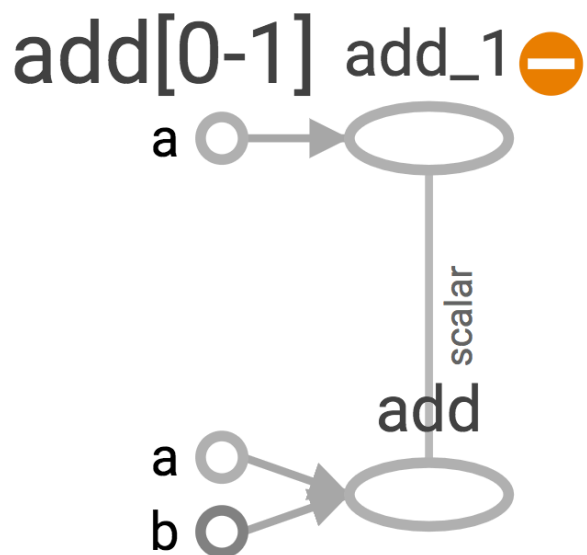
一系列的相连接的操作相同且name相同（不包括系统给加的后缀）节点可以组成一个有连接关系的节点组。

```
import tensorflow as tf

with tf.Graph().as_default() as graph:
    a = tf.constant(2, name='a')
    b = tf.constant(3, name='b')
    x = tf.add(a, b, name='add')
    y = tf.add(a, x, name='add')

with tf.Session() as sess:
    writer = tf.summary.FileWriter('./graphs')
    writer.add_graph(graph)
    writer.close()
```

可视化之后如下：



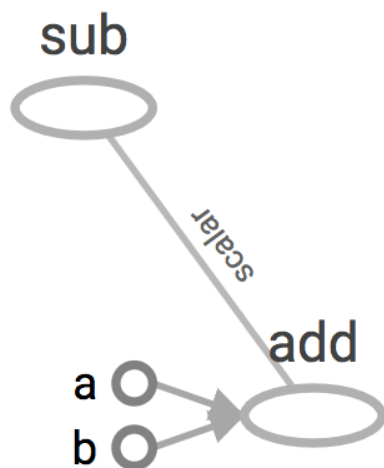
这两个子图可以组成节点组，如下：



Dataflow edge

```
with tf.Graph().as_default() as graph:
    a = tf.constant(2, name='a')
    b = tf.constant(3, name='b')
    x = tf.add(a, b, name='add')
    y = tf.subtract(x, x, name='sub')

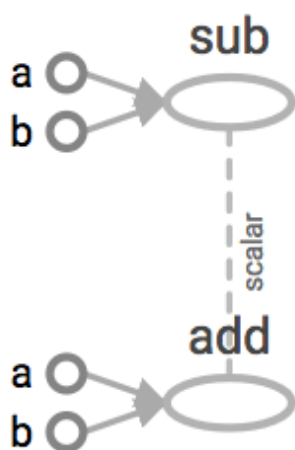
with tf.Session() as sess:
    writer = tf.summary.FileWriter('./graphs')
    writer.add_graph(graph)
writer.close()
```



Control Dependency edge

```
with tf.Graph().as_default() as graph:
    a = tf.constant(2, name='a')
    b = tf.constant(3, name='b')
    x = tf.add(a, b, name='add')
    with graph.control_dependencies([x]):
        y = tf.subtract(a, b, name='sub')

with tf.Session() as sess:
    writer = tf.summary.FileWriter('./graphs')
    writer.add_graph(graph)
writer.close()
```



其它

在图可视化面板中，还有有常用的功能，比如：切换不同的图；查看某次迭代时运行的内存、时间；标注节点使用的计算设备；导入别的图等。

3.2 SCALARS

SCALARS面板可以展示训练过程中某些标量数据的变化情况，例如模型参数的更新情况、模型的正确率等。

使用scalars面板，主要是使用 `tf.summary.scalar()` 函数对张量进行采样，并返回一个包含摘要的protobuf的类型为string的0阶张量，函数的用法如下：

```
# name 节点名称，也将作为在SCALARS面板中显示的名称
# tensor 包含单个值的实数数字Tensor
tf.summary.scalar(name, tensor, collections=None)
```

`tf.summary.scalar()` 可以提取标量摘要，提取到的摘要需要添加到事件文件中，这时候可以使用 `tf.summary.FileWriter.add_summary()` 方法添加到事件文件中。`add_summary()` 的用法如下：

```
add_summary(summary, global_step=None)
```

第一个参数是protobuf的类型为string的0阶张量摘要，第二个参数是记录摘要值的步数。例如训练100次模型，每次取一个摘要，这时候第二个参数就是第n次，这样可以得到标量随训练变化的情况。

注意： `tf.summary.scalar()` 获取到的并不是直接可以添加到事件文件中的数据，而是一个张量对象，必须在会话中执行了张量对象之后才能得到可以添加到事件文件中的数据。

完整创建标量摘要的代码如下：

```
import tensorflow as tf

with tf.Graph().as_default() as graph:
    var = tf.Variable(0., name='var')
    summary_op = tf.summary.scalar('value', var) # 给变量var创建摘要

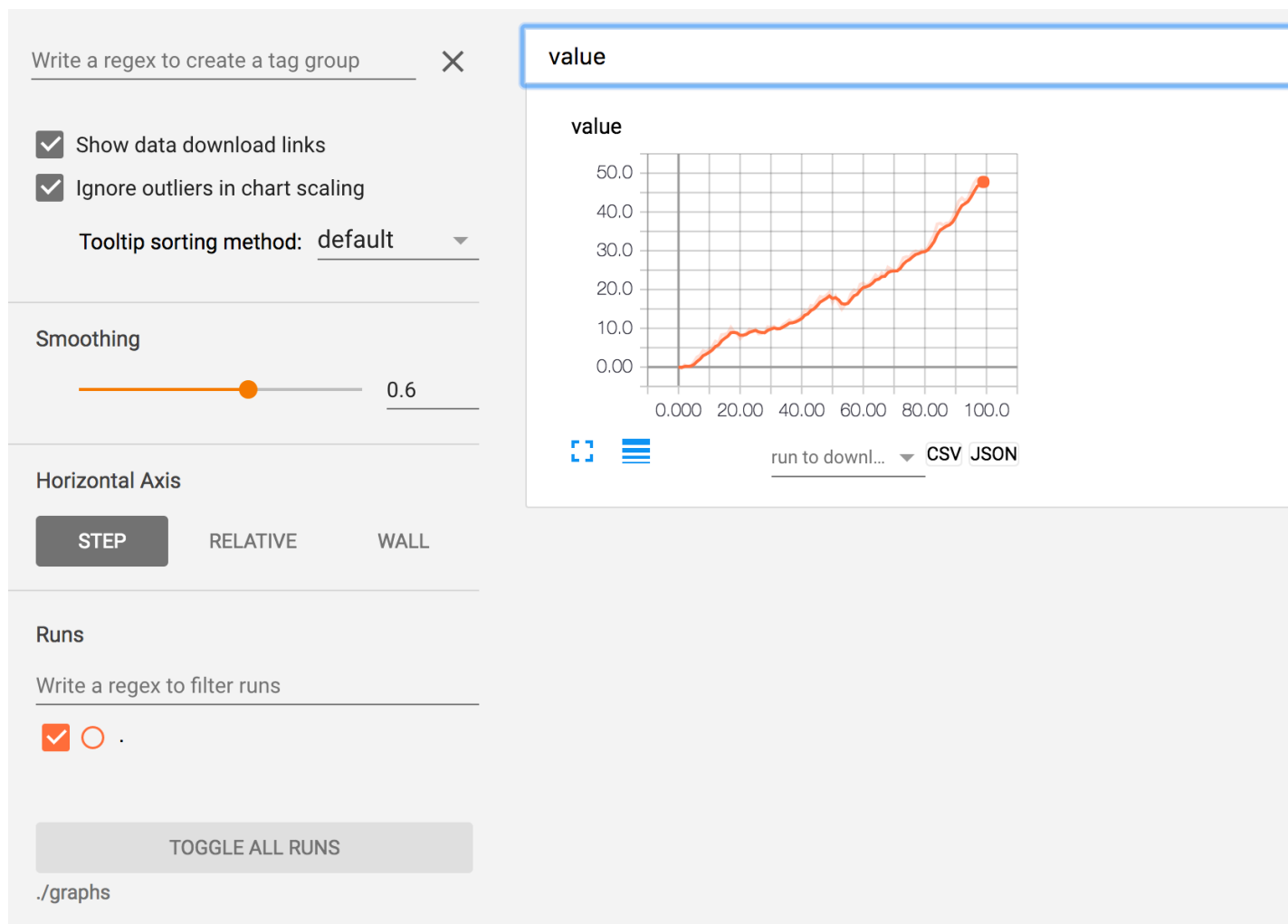
    random_val = tf.random_normal([], mean=0.5, stddev=1.)
    assign_op = var.assign_add(random_val) # var累加一个随机数

with tf.Session(graph=graph) as sess:
    # 创建一个事件文件管理对象
    writer = tf.summary.FileWriter('./graphs')
    writer.add_graph(graph)

    sess.run(tf.global_variables_initializer())
    for i in range(100):
        # 得到summary_op的值
        summary, _ = sess.run([summary_op, assign_op])
        # summary_op的值写入事件文件
        writer.add_summary(summary, i)

    writer.close()
```

上述代码执行之后，在SCALARS面板中显示如下：



面板的右侧，可以看到一个曲线图，这个曲线图是我们创建的变量 `var` 的值的变化的曲线。纵坐标是我们设置的 `summary_op` 的 `name`。曲线图可以放大、缩小，也可以下载。

面板左侧可以调整右侧的显示内容。我们看到的右侧的线是一个曲线，然而我们得到应该是一个折线图，这是因为左侧设置了平滑度(Smoothing)。横坐标默认是步数(STEP)，也可以设置为按照相对值(RELATIVE)或按照时间顺序(WALL)显示。

SCALARS面板也可以同时显示多个标量的摘要。

3.3 HISTOGRAMS

HISTOGRAMS面板显示的是柱状图，对于2维、3维等数据，使用柱状图可以更好的展示出其规律。例如

```
import tensorflow as tf

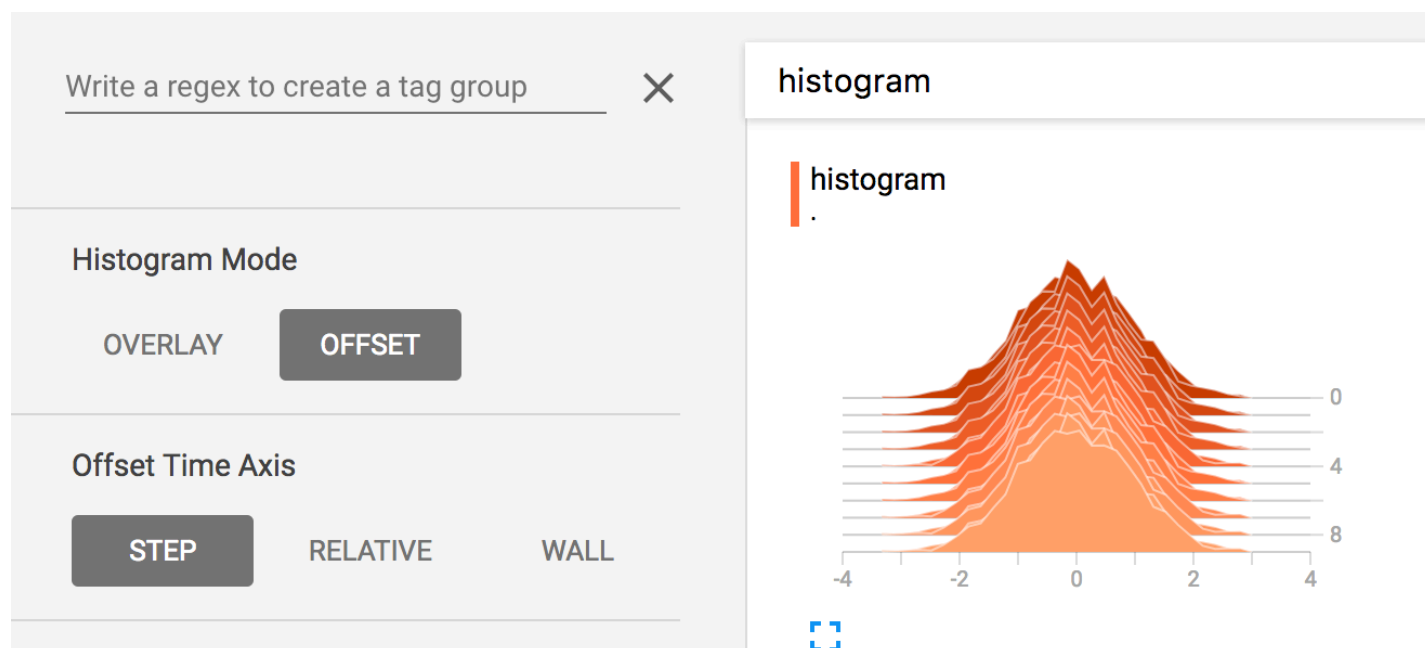
with tf.Graph().as_default() as graph:
    var = tf.Variable(tf.random_normal([200, 10], mean=0., stddev=1.), name='var')
    histogram = tf.summary.histogram('histogram', var)

with tf.Session(graph=graph) as sess:
    writer = tf.summary.FileWriter('./graphs')
    writer.add_graph(graph)

    sess.run(tf.global_variables_initializer())
    for i in range(10):
        summaries, _ = sess.run([histogram, var])
        writer.add_summary(summaries, i)

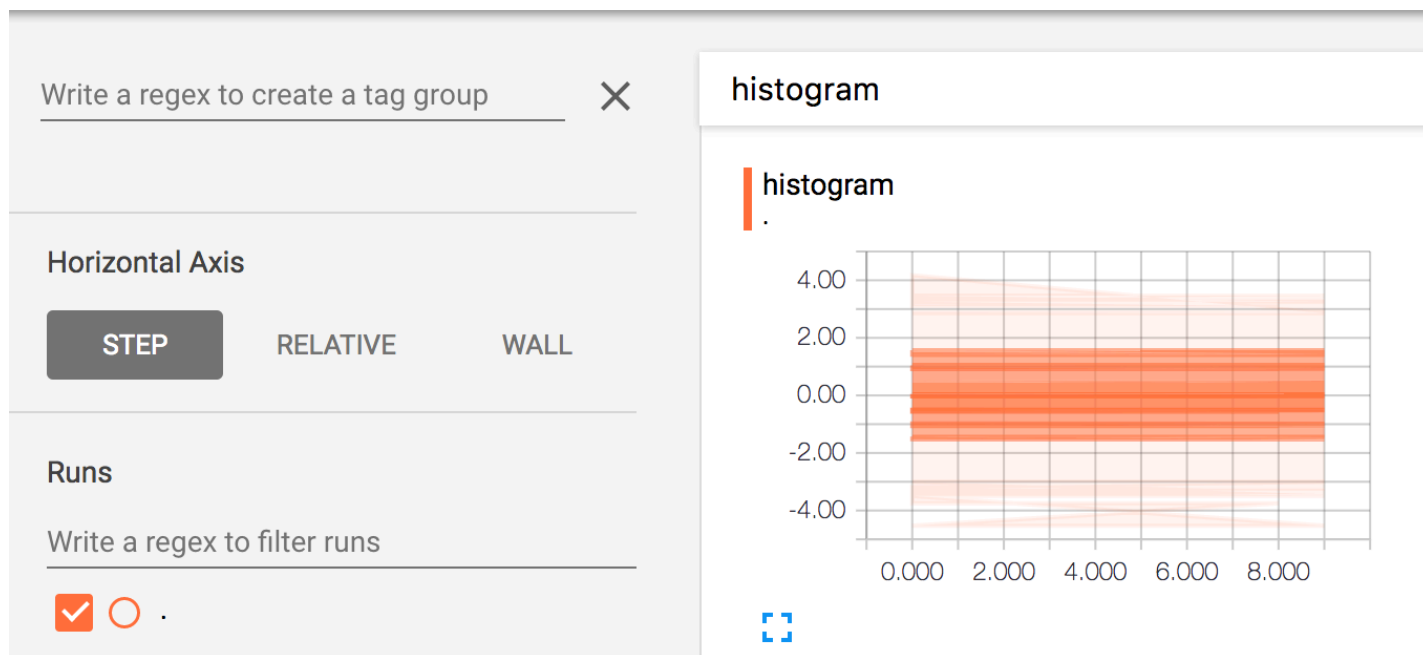
writer.close()
```

上述代码，我们使用了 `tf.summary.histogram()` 方法建立了柱状图摘要，其使用方法类似于scalar摘要。在HISTOGRAMS面板中显示如下。



3.4 DISTRIBUTIONS

DISTRIBUTIONS面板与HISTOGRAMS面板相关联，当我们创建了柱状图摘要之后，在DISTRIBUTIONS面板也可以看到图像，上述HISTOGRAMS面板对应的在DISTRIBUTIONS面板中显示如下：



DISTRIBUTIONS面板可以看做HISTOGRAMS面板的压缩后的图像。

3.5 IMAGES

IMAGES面板用来展示训练过程中的图像，例如我们需要对比原始图像与算法处理过的图像的差异时，我们可以使用IMAGES面板将其展示出来。

建立IMAGES摘要的函数是：`tf.summary.image()`，具体如下：

```
tf.summary.image(name, tensor, max_outputs=3, collections=None)
```

这里的参数 `tensor` 必须是4D的shape= `[batch_size, height, width, channels]` 的图片。参数 `max_outputs` 代表从一个批次中摘要几个图片。

举例，我们使用随机数初始化10张rgba的图片，并选取其中的前5张作为摘要：

```
import tensorflow as tf

with tf.Graph().as_default() as graph:
    var = tf.Variable(tf.random_normal([10, 28, 28, 4], mean=0., stddev=1.), name='var')
    image = tf.summary.image('image', var, 5)

with tf.Session(graph=graph) as sess:
    writer = tf.summary.FileWriter('./graphs')
    writer.add_graph(graph)

    sess.run(tf.global_variables_initializer())
    summaries, _ = sess.run([image, var])
    writer.add_summary(summaries)

writer.close()
```

在IMAGES面板中显示如下：

Write a regex to create a tag group
X

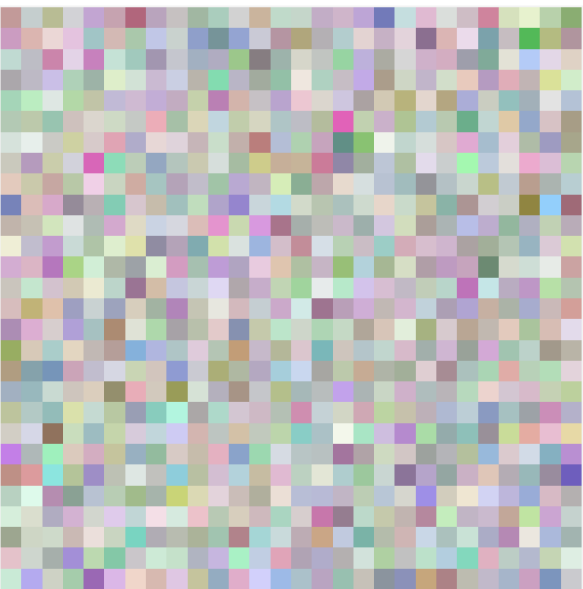
Runs
Write a regex to filter runs
☒ ☐ .



TOGGLE ALL RUNS

./graphs


image
5

image/image/0
step 9 (Wed Aug 16 2017 08:55:46 GMT+0800 (CST))



image/image/1
step 9 (Wed Aug 16 2017 08:55:46 GMT+0800 (CST))



3.6 AUDIO

AUDIO面板与IMAGES面板类似，只不过是图像换成了音频。音频数据比图像数据少一个维度。创建AUDIO摘要的方法是：`tf.summary.audio()`，具体如下：

```
tf.summary.audio(name, tensor, sample_rate, max_outputs=3, collections=None)
```

这里的参数 `tensor` 必须是3D的`shape=[batch_size, frames, channels]` 或2D的`shape=[batch_size, frames]` 的音频。`sample_rate` 参数。参数 `max_outputs` 代表从一个批次中摘要几个音频。

下面我们生成10段采样率为22050的长度为2秒的噪声，并选取其中的5个进行采样：

```
import tensorflow as tf

with tf.Graph().as_default() as graph:
    var = tf.Variable(tf.random_normal([10, 44100, 2], mean=0., stddev=1.), name='var')
    audio = tf.summary.audio('video', var, 22050, 5)

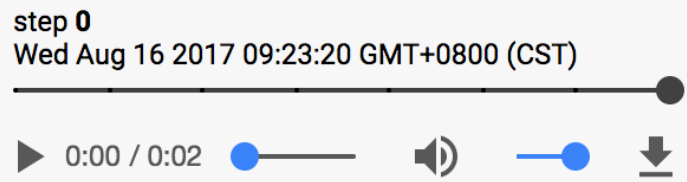
with tf.Session(graph=graph) as sess:
    writer = tf.summary.FileWriter('./graphs')
    writer.add_graph(graph)

    sess.run(tf.global_variables_initializer())
    summaries, _ = sess.run([audio, var])
    writer.add_summary(summaries)

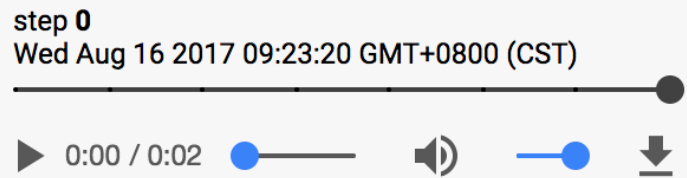
    writer.close()
```

在AUDIO面板中可以播放、下载采样的音频，显示效果如下：

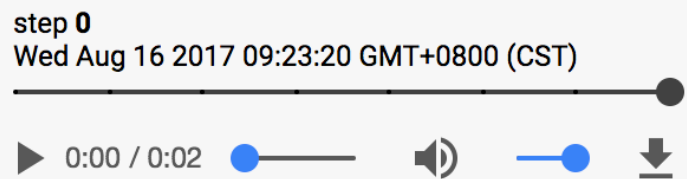
video/audio/0



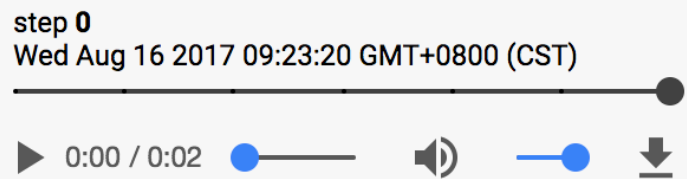
video/audio/1



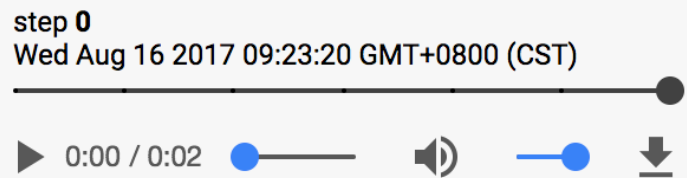
video/audio/2



video/audio/3



video/audio/4



3.7 TEXT

TEXT面板用来可视化多行字符串，使用 `tf.summary.text()` 方法进行。

```
tf.summary.text(name, tensor, collections=None)
```

举例：


```
import tensorflow as tf

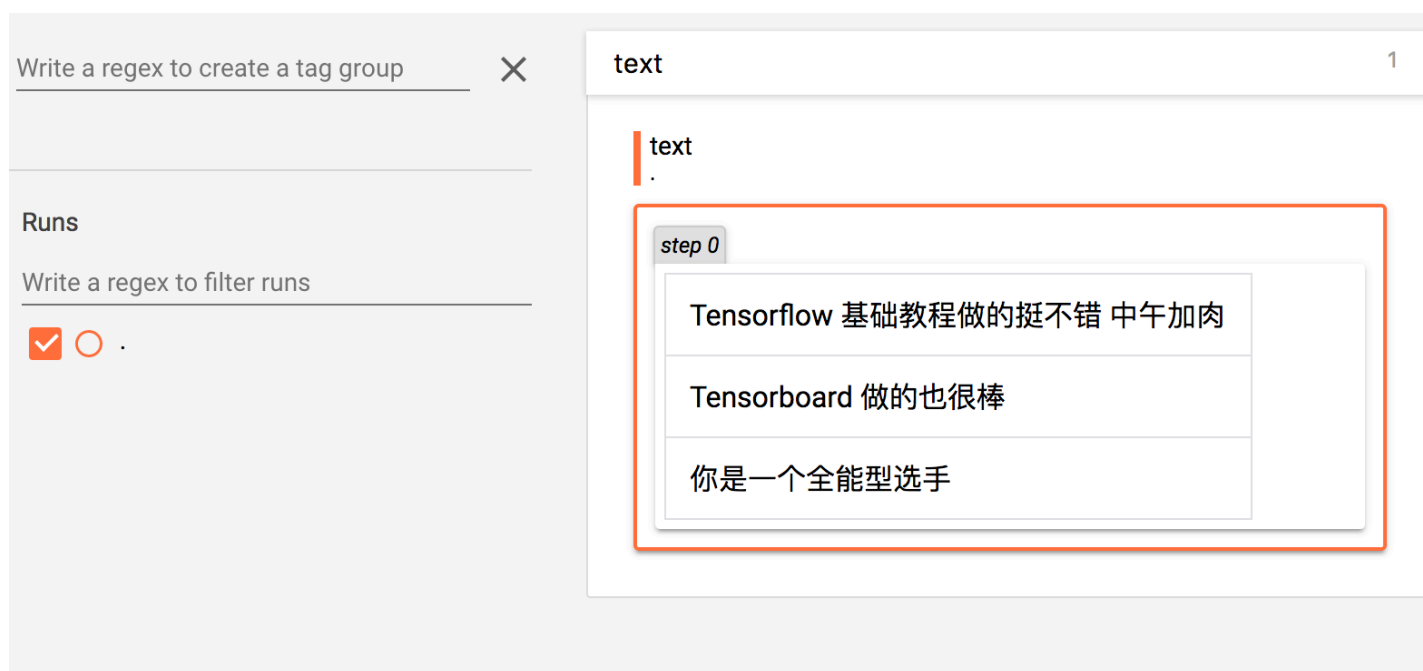
with tf.Graph().as_default() as graph:
    var = tf.constant(['Tensorflow 基础教程做的挺不错 中午加肉',
                      'Tensorboard 做的也很棒',
                      '你是一个全能型选手'])
    text = tf.summary.text('text', var)

with tf.Session(graph=graph) as sess:
    writer = tf.summary.FileWriter('./graphs')
    writer.add_graph(graph)

    sess.run(tf.global_variables_initializer())
    summaries, _ = sess.run([text, var])
    writer.add_summary(summaries)

writer.close()
```

可视化效果如下：



3.8 EMBEDDINGS

EMBEDDINGS面板一般用来可视化词向量(word embedding)。在自然语言处理、推荐系统中较为常见。embedding的可视化与其它类型的可视化完全不同，需要用到tensorboard插件并配合检查点操作。具体教程我们在之后的内容中详述，这里，我们可以查看一下手写数字可视化后的效果：



Points: 10000 | Dimension: 784

