

TensorBoard的使用



Manifestain

关注

0.135

2018.01.23 11:58:39

字数 1,211

阅读 10,525

TensorBoard是Tensorflow的可视化工具，它通过对Tensorflow程序运行过程中输出的日志文件进行可视化Tensorflow程序的运行状态。

使用TensorBoard展示数据，需要在执行Tensorflow计算图的过程中，将各种类型的数据（`summary` `protobuf`）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。

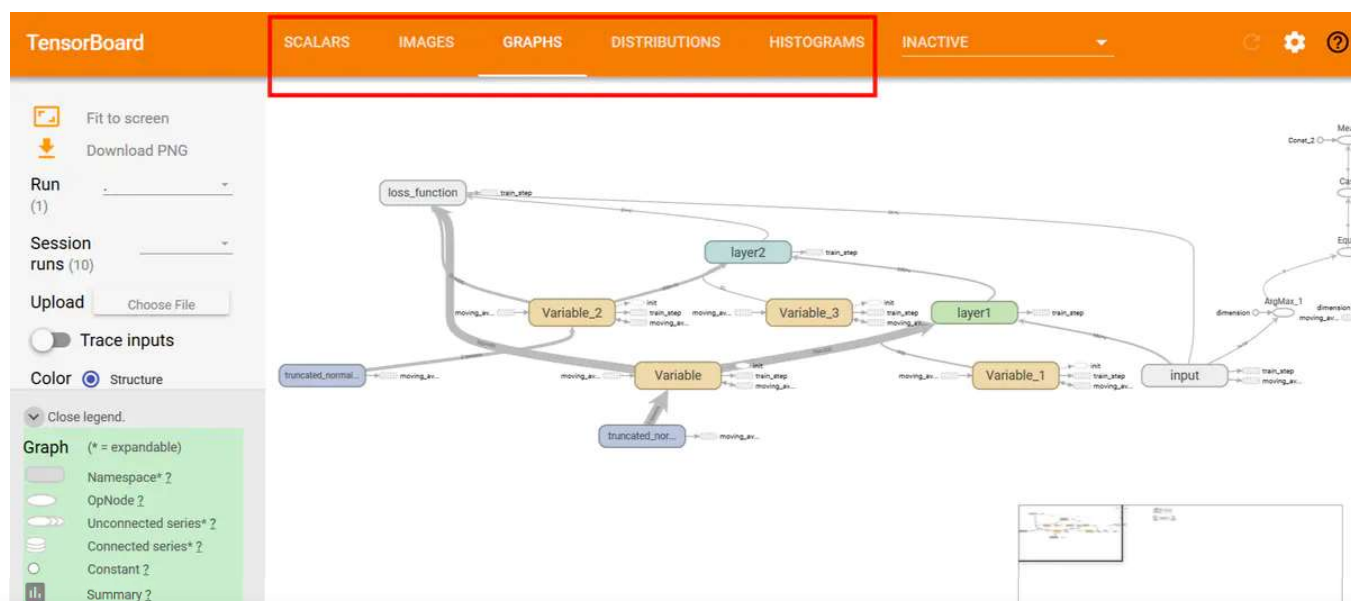
注：tensorflow --version 1.4.0

TensorBoard

当生成了日志文件后，在命令行中使用 `tensorboard --logdir=日志文件目录` 启动一个服务，在浏览器中使用 `http://DESKTOP-JGL4HV5:6006` 查看可视化结果。

这里有一点需要注意，日志文件目录要使用绝对路径，即从某个盘开始的路径（如果不行的话将 / 变成 // 再试试）。同时使用360浏览器可能无法显示。

下面红框中是可视化的顶部：



写下你的评论...

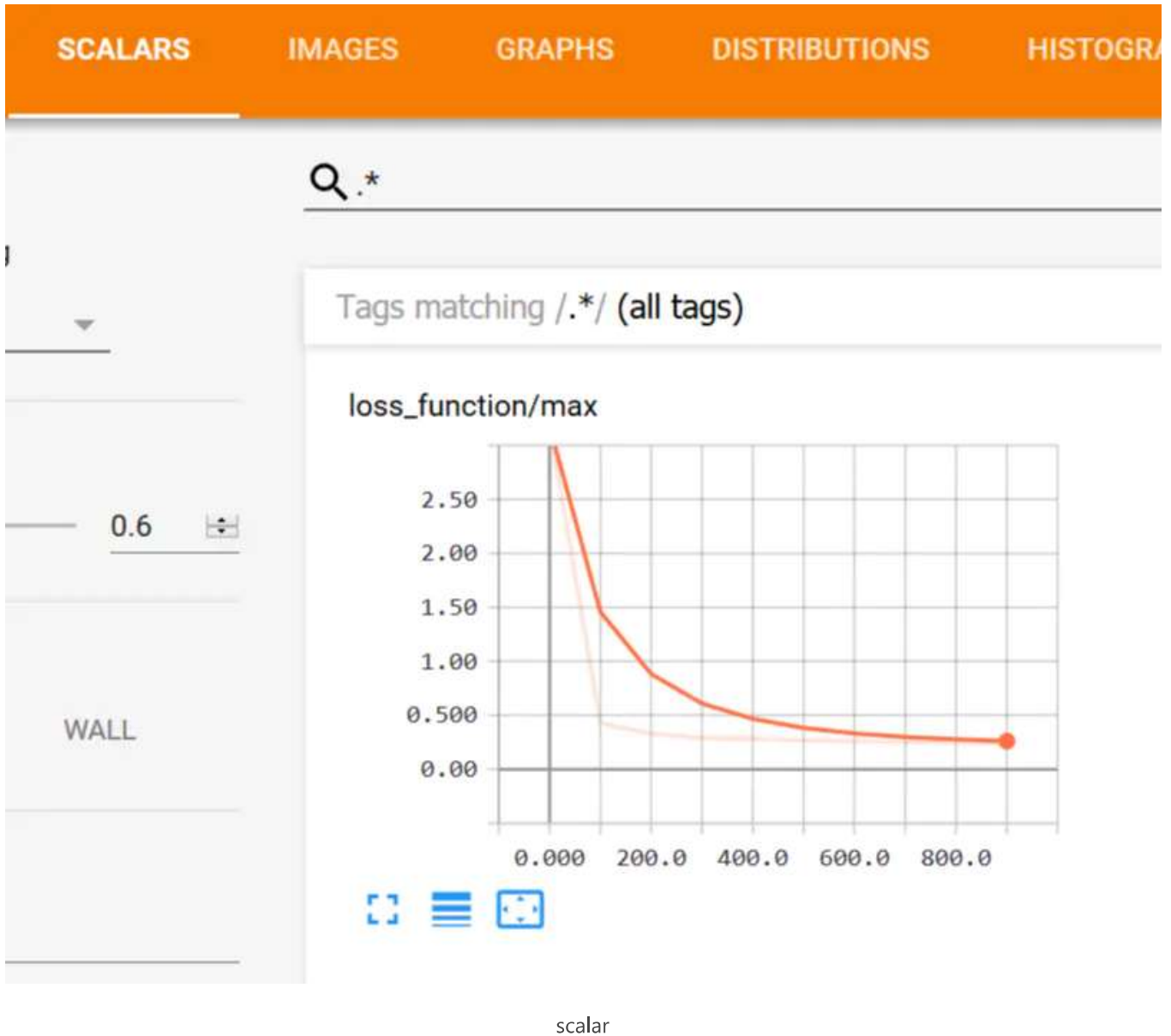
评论5

赞5

...

1. SCALARS, 对标量数据进行汇总和记录

使用方法: `tf.summary.scalar(tags, values, collections=None, name=None)`

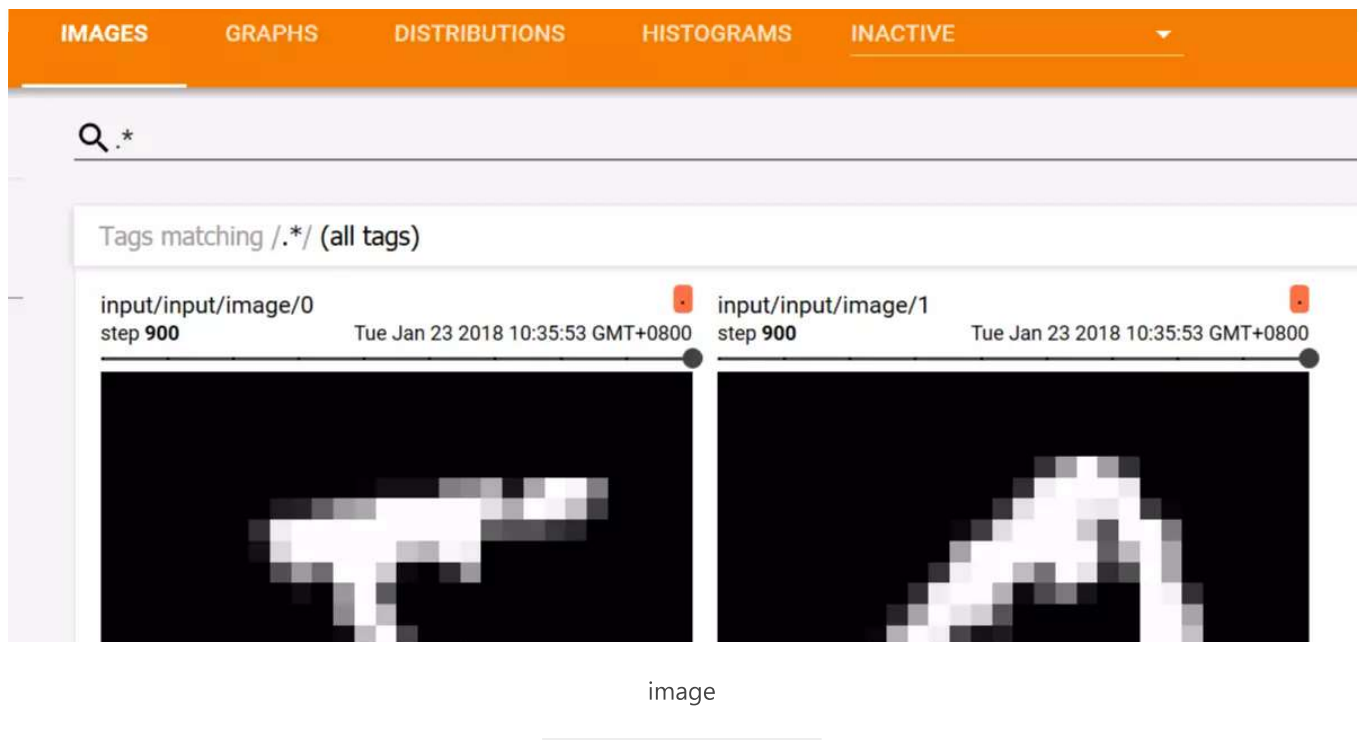


scalar

一般在刻画loss和accuracy时会用到, 可以计算标量的最大最小值或者标准差等

2. IMAGES, 汇总数据中的图像, 例如MNIST中可以将输入的向量还原成图片的像素矩阵

使用方法: `tf.summary.image(tag, tensor, max_images=3, collections=None, name=None)`



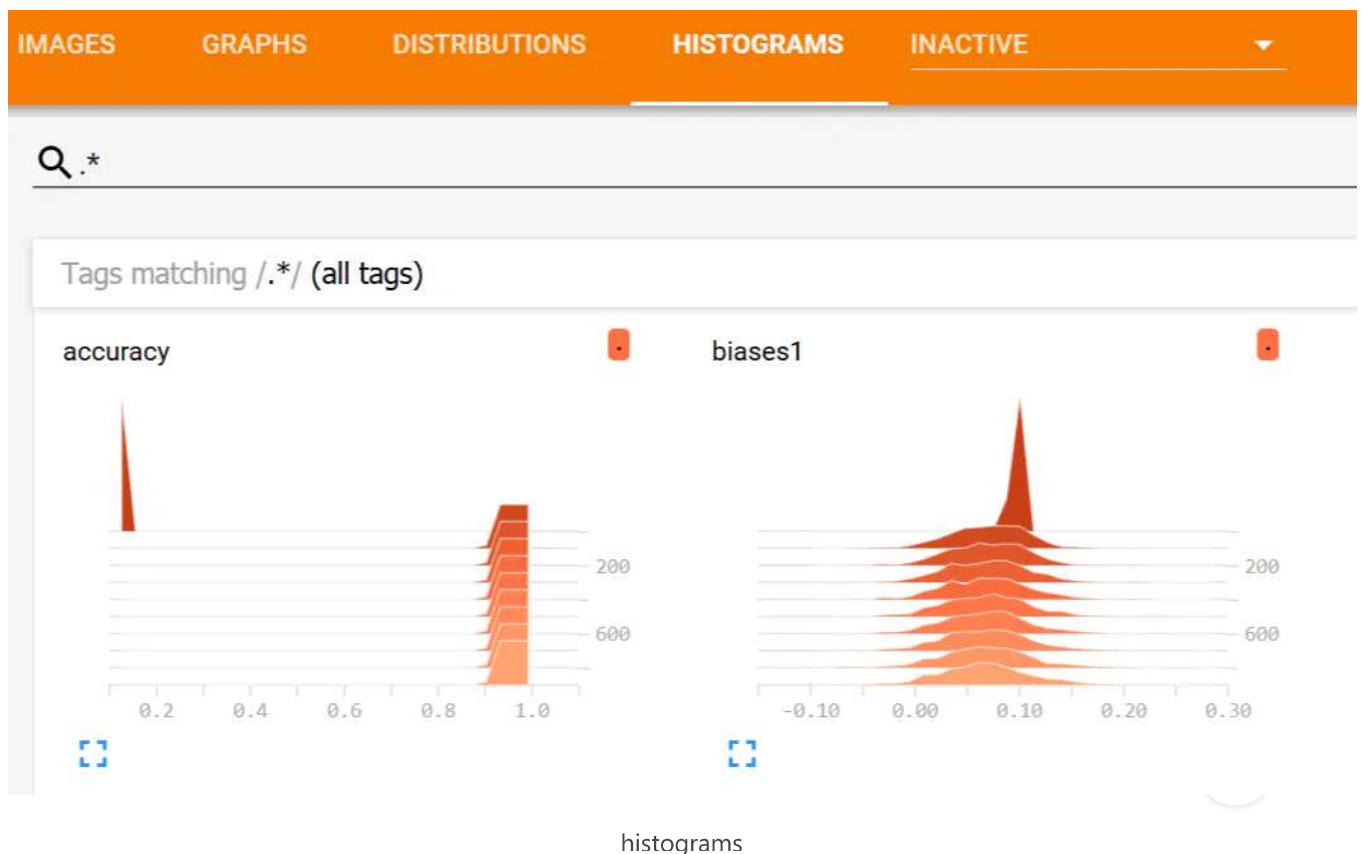
3. GRAPHS, 可视化Tensorflow计算图的结构及计算图上的信息

使用方法: `tf.summary.FileWriter(logdir, graph)`

其实这个方法是将当前summary protobuf写近日志文件中, 但是会自动生成计算图。

4. HISTOGRAMS, 记录变量的直方图(张量中元素的取值分布)

使用方法: `tf.summary.histogram(tag, values, collections=None, name=None)`



写下你的评论...

评论5 赞5 ...

计算图

计算图可以很好展现整个神经网络的结构。下来将计算图中的图标进行总结：

1. 边，计算图中的节点之间有两种不同的边：

实线：刻画了数据的传输，箭头代表方向

虚线：表达了计算之间的依赖关系

有些边上的箭头是 **双向** 的表示一个节点可能会修改另一个节点，同时边上还标注了张量的 **维度信息**，边上的 **粗细** 表示了两个节点之间传输的标量维度的总大小（不是传输的标量个数）。

2. 图，TensorBoard会智能的调整可视化效果图上的节点，将计算图分成了 **主图（Main Graph）** 和 **辅助图（Auxiliary nodes）**。也可以手动调整，对图中的节点进行移除（不会保存手工修改结果，刷新后还原）。

3. 节点，当点击可视化图中的节点时，界面右上角会弹出该节点的基本信息（输入、输出、依赖关系以及消耗时间和内存信息等）。

空心小椭圆：对应计算图上一个计算节点

矩形：对应了计算图上的一个命名空间

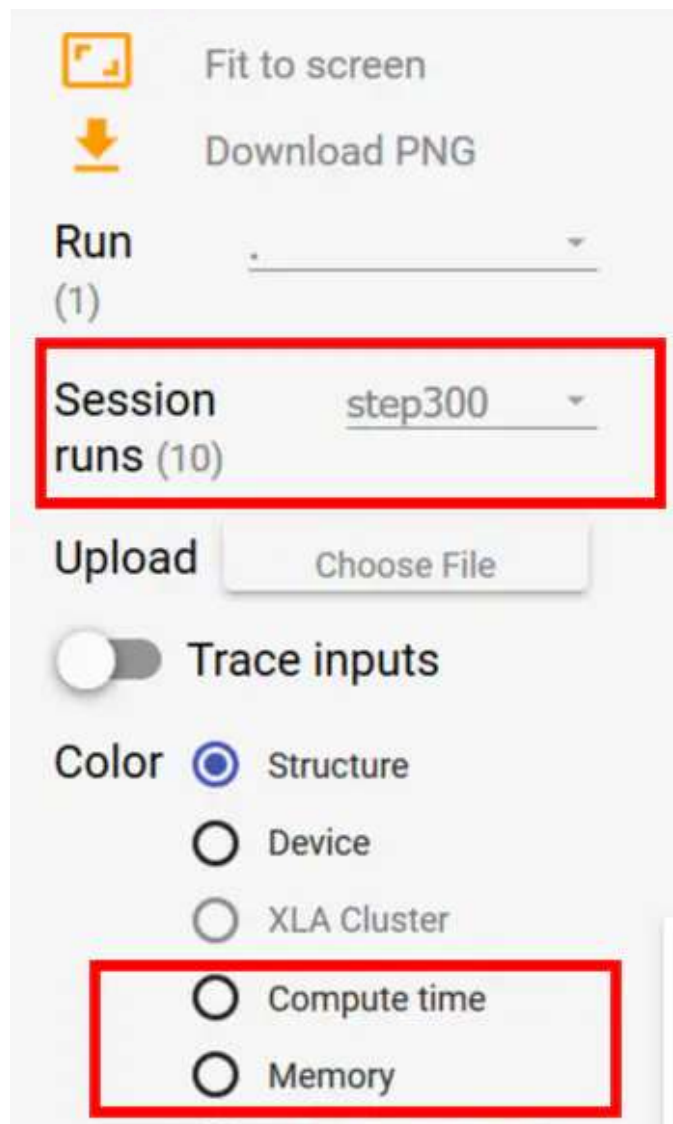
节点信息

刚说节点的基本信息中包含消耗时间和内存信息，可以通过以下方法将其添加到日志文件并进行展示。

```
1   for i in range(TRAINING_STEPS):
2       # 每1000次就在验证集上测试训练的模型精度
3       if i % 100 == 0:
4           # 配置运行时要记录的信息
5           run_options = tf.RunOptions(trace_level=tf.RunOptions.FULL_TRACE)
6           # 运行时记录运行信息的proto
7           run_metadata = tf.RunMetadata()
8           # 将配置信息和运行记录信息的proto传入运行过程，从而进行记录
9           validate_acc, sum = sess.run([accuracy, summ], feed_dict=validate_feed, options=run_option
10          # 将节点的运行信息写入日志文件
11          writer.add_run_metadata(run_metadata, 'step%03d' % i)
```

在TensorBoard可视化的Graphs页面中，左侧的 **Session runs** 会出现一个下拉菜单，记录了所有运行次数，选择一次运行后，**Color** 栏中会出现 **Compute time** 和 **Memory**，分别对应了计算节点的运行时间和消耗的内存。





GRAPHS的左边框

merge_all()

和Tensorflow类似，`tf.summary.histograms()`等函数不会立即执行，需要通过`sess.run()`来明确调用，当日志程序中定义写日志的操作比较多时，可以使用`summ = tf.summary.merge_all()`函数来整理所有的日志生成操作，最后只需要`sess.run(summ)`即可将定义中的所有日志生成操作一次执行。

TensorBoard的使用流程

1. 添加记录节点：`tf.summary.scalar/image/histogram()`等
2. 汇总记录节点：`merged = tf.summary.merge_all()`
3. 运行汇总节点：`summary = sess.run(merged)`，得到汇总结果
4. 日志书写器实例化：`summary_writer = tf.summary.FileWriter(logdir, graph=sess.graph)`，实例化时传入`graph`将当前计算图写入日志

写下你的评论...



评论5



赞5



5. 调用日志书写器实例对象 `summary_writer` 的 `add_summary(summary, global_step=i)` 方法将所有汇总日志写入文件

6. 调用日志书写器实例对象 `summary_writer` 的 `close()` 方法写入内存，否则它每隔120s写入一次

下面是一个完整的使用TensorBoard的代码：

```
1  # -*- coding:utf-8 -*-
2
3  import tensorflow as tf
4  from tensorflow.examples.tutorials.mnist import input_data
5
6  # 定义神经网络的神经元数目
7  INPUT_NODE = 784
8  LAYER1_NODE = 500
9  OUTPUT_NODE = 10
10
11 # 每次训练数据的个数
12 BATCH_SIZE = 100
13
14 # 衰减学习率的参数
15 LEARNING_RATE_BASE = 0.8
16 LEARNING_RATE_DECAY = 0.99
17
18 # 正则化项的系数
19 REGULARIZATION_RATE = 0.0001
20
21 # 滑动平均的参数
22 TRAINING_STEPS = 1000
23 MOVING_AVERAGE_DECAY = 0.99
24
25 # 定义神经网络和前向传播算法
26 def inference(input_tensor, avg_class, weights1, biases1, weights2, biases2):
27     if avg_class == None:
28         with tf.name_scope('layer1'):
29             layer1 = tf.nn.relu(tf.matmul(input_tensor, weights1) + biases1)
30         with tf.name_scope('layer2'):
31             output = tf.matmul(layer1, weights2) + biases2
32     else:
33         with tf.name_scope('layer1'):
34             layer1 = tf.nn.relu(tf.matmul(input_tensor, avg_class.average(weights1)) + avg_class.average(b
35         with tf.name_scope('layer2'):
36             output = tf.matmul(layer1, avg_class.average(weights2)) + avg_class.average(biases2)
37
38     tf.summary.histogram('weights1', weights1)
39     tf.summary.histogram('biases1', biases1)
40     tf.summary.histogram('weights2', weights2)
41     tf.summary.histogram('biases2', biases2)
42
43     return output
44
45 def train(mnist):
46     with tf.name_scope('input'):
47         x = tf.placeholder(tf.float32, [None, INPUT_NODE], name='x-input')
48         x_image = tf.reshape(x, [-1, 28, 28, 1])
49         tf.summary.image('input', x_image, 10)
```

写下你的评论...



评论5



赞5

...

```

52 # 定义神经网络的参数
53 weights1 = tf.Variable(tf.truncated_normal([INPUT_NODE, LAYER1_NODE], stddev=0.1))
54 biases1 = tf.Variable(tf.constant(0.1, shape=[LAYER1_NODE]))
55 weights2 = tf.Variable(tf.truncated_normal([LAYER1_NODE, OUTPUT_NODE], stddev=0.1))
56 biases2 = tf.Variable(tf.constant(0.1, shape=[OUTPUT_NODE]))
57
58 # 计算前向传播结果
59 y = inference(x, None, weights1, biases1, weights2, biases2)
60
61 # 使用带有滑动平均的模型计算前行传播结果
62 with tf.name_scope('moving_average'):
63     global_step = tf.Variable(0, trainable=False)
64     variable_average = tf.train.ExponentialMovingAverage(MOVING_AVERAGE_DECAY, global_step)
65     variables_averages_op = variable_average.apply(tf.trainable_variables())
66
67     average_y = inference(x, variable_average, weights1, biases1, weights2, biases2)
68
69 # 计算交叉熵和损失函数
70 with tf.name_scope('loss_function'):
71     cross_entropy = tf.nn.sparse_softmax_cross_entropy_with_logits(logits=y, labels=tf.argmax(y_, 1))
72     cross_entropy_mean = tf.reduce_mean(cross_entropy)
73     regularizer = tf.contrib.layers.l2_regularizer(REGULARIZATION_RATE)
74     regularization = regularizer(weights1) + regularizer(weights2)
75     loss = cross_entropy_mean + regularization
76
77     tf.summary.scalar('max', tf.reduce_max(loss))
78
79 # 使用衰减学习率
80 with tf.name_scope('train_step'):
81     learning_rate = tf.train.exponential_decay(
82         LEARNING_RATE_BASE,
83         global_step,
84         mnist.train.images.shape[0] / BATCH_SIZE,
85         LEARNING_RATE_DECAY
86     )
87
88 # 定义使用的优化方法
89 train_step = tf.train.GradientDescentOptimizer(learning_rate).minimize(loss, global_step=global_st
90
91 # 定义同时更新滑动平均值和参数的方法
92 with tf.control_dependencies([train_step, variables_averages_op]):
93     train_op = tf.no_op('train')
94
95 # 定义精度的计算
96 correct_prediction = tf.equal(tf.argmax(average_y, 1), tf.argmax(y_, 1))
97 accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
98
99 tf.summary.histogram('accuracy', accuracy)
100 summ = tf.summary.merge_all()
101
102
103 # 初始化会话并开始训练
104 with tf.Session() as sess:
105     tf.global_variables_initializer().run()
106     validate_feed = {x: mnist.validation.images, y_: mnist.validation.labels}
107     test_feed = {x: mnist.test.images, y_: mnist.test.labels}
108
109     writer = tf.summary.FileWriter('log/')
110     writer.add_graph(sess.graph)
111

```

```

114         if i % 100 == 0:
115             # 配置运行时要记录的信息
116             run_options = tf.RunOptions(trace_level=tf.RunOptions.FULL_TRACE)
117             # 运行时记录运行信息的proto
118             run_metadata = tf.RunMetadata()
119             # 将配置信息和运行记录信息的proto传入运行过程, 从而进行记录
120             validate_acc, sum = sess.run([accuracy, summ], feed_dict=validate_feed, options=run_option
121             # 将节点的运行信息写入日志文件
122             writer.add_run_metadata(run_metadata, 'step%03d' % i)
123             writer.add_summary(sum, i)
124
125             print('After %d training step(s), validation accuracy using average model is %g' % (i, val
126
127             # 用于生成下一次迭代的训练数据
128             xs, ys = mnist.train.next_batch(BATCH_SIZE)
129             sess.run(train_op, feed_dict={x: xs, y_: ys})
130
131             # 验证在测试集上的准确度
132             test_acc = sess.run(accuracy, feed_dict=test_feed)
133             print('After %d training step(S), test accuracy using average model is %g' % (TRAINING_STEPS, test
134
135     def main(argv=None):
136         mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
137         train(mnist)
138
139     if __name__ == '__main__':
140         tf.app.run()

```



5人点赞 >

Tensorflow学习笔记

"小礼物走一走，来简书关注我"

赞赏支持

还没有人赞赏，支持一下



Manifestain 凡心所向，素履皆往。 生如逆旅，一苇以航。
总资产4 (约0.45元) 共写了10.3W字 获得104个赞 共67个粉丝

关注

华为云

0元试用 [40+高配云产品]

云主机 | 安全 | 数据库 (企业)0元体验

立即体验



写下你的评论...


评论5
 赞5

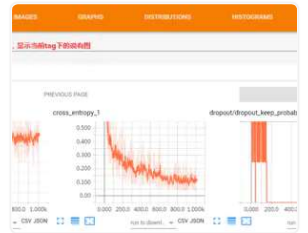
推荐阅读

更多精彩内容>

TensorBoard 的使用

本篇博客主要介绍下 tensorboard 的使用方法，tensorboard 是 tensorflow 中一个可...

 Yigit_dev 阅读 935 评论 0 赞 0



TensorFlow从入门到入门

简单线性回归 import tensorflow as tf import numpy # 创造数据 x_dat...

 CAICAI0 阅读 2,319 评论 0 赞 50

学习笔记TF048:TensorFlow 系统架构、设计理念、编程模型、API、作用域、批标准...

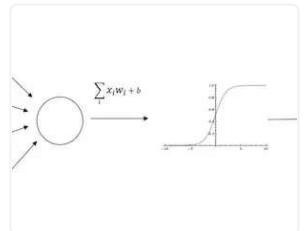
系统架构。自底向上，设备层、网络层、数据操作层、图计算层、API层、应用层。核心层，设备层、网络层、数据操作层、图...

 利炳根 阅读 3,076 评论 0 赞 16

深入浅出TensorFlow（二）：TensorFlow解决MNIST问题入门

作者 | 赵翼编辑 | NatalieAI前线出品 | ID: ai-front 2017年2月16日，Google正式对外发...

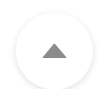
 AI前线 阅读 2,275 评论 0 赞 7



开发必备使用的插件【xcode8这些不能用了，记录下吧】

这种百度到的就不多说了。http://www.codeceo.com/article/10-ios-xcode-p...

 LD_X 阅读 154 评论 0 赞 0



写下你的评论...



评论5



赞5

