## TensorFlow 内核剖析 编程模型

刘光聪

技术规划部

2017-08-01



- 1 架构概述
- 2 编程模型
- ③ 训练模型
- 4 实战 Mnist
- 5 文献

# 架构概述

- 系统架构
- ② 图控制

架构概述



架构概述 系统架构

#### What's TensorFlow

#### TensorFlow 是一个使用数据流图用于数值计算的开源软件库

- 跨平台:支持多 CPU/GPU 运算;台式机/服务器/移动设备
- 分布式:支持本地/分布式运行模式
- 多语言: 支持 Python, C++, Java, Go 等多种编程语言接口
- 通用性:支持构建复杂的网络模型,支持数据/模型并发模式
- 可扩展: 支持 OP 扩展, Kernel 扩展, Device 扩展
- 可视化: 使用 TensorBoard 可视化计算图
- 自动微分:自动构造反向的计算子图,完成梯度的计算

架构概述 ○●○○○○○○ 系统架构

# 系统架构

Client C++ Python Java Others
C API
Distributed Runtime  Distributed Work Dataflow Executor
Kernel Implements
Network Layer  RPC RDMA  Device Layer  GPU CPU

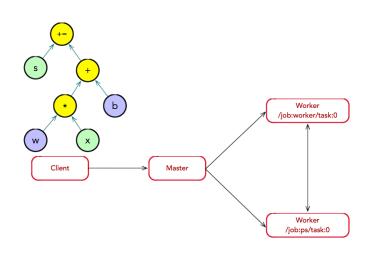
### 设计原则

- 延迟计算:图的构造与执行分离,并推迟计算图的执行过程
- 原子 OP: OP 是最小的抽象计算单元, 支持构造复杂的网络模型
- 抽象设备:支持 CPU, GPU, ASIC 多种异构设备类型
- 抽象任务:基于任务的 PS. 支持优化算法和网络模型的扩展

图控制

架构概述 ○○○●0000

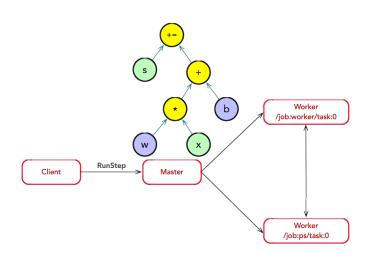
#### 构造计算图



图控制

架构概述 ○○○0●000

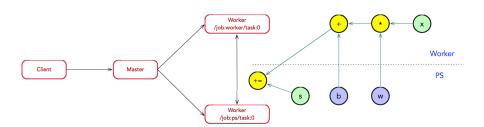
#### 执行计算图



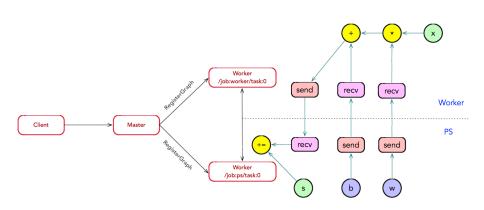


0000000

# 图分解



#### 注册图

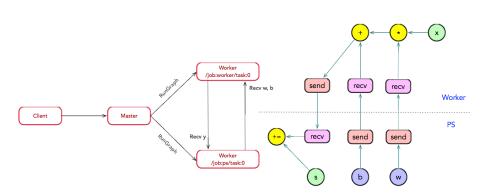




架构概述 图控制

#### 执行图

0000000





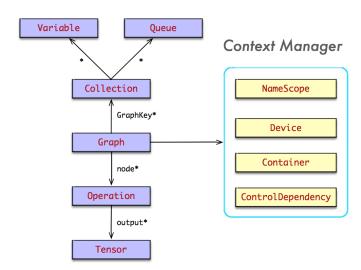
# 编程模型

- 计算图
- 2 变量
- 3 会话
- 图构造与执行



计算图

### $Graph = Set\{OP\} + Set\{Tensor\}$

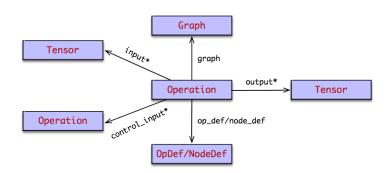




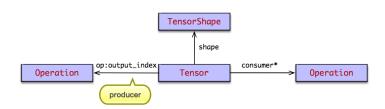
刘光聪 (ZTE) TensorFlow Internals 2017.08 13/50

计算图

#### OP: 抽象计算



### Tensor: 承载数据





计算图

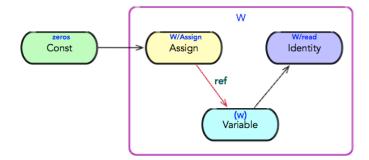
有向边

#### 两种类型

- 普通边: 承载 Tensor, 且表示执行依赖关系
- 控制依赖边: 不承载 Tensor, 仅表示执行依赖关系

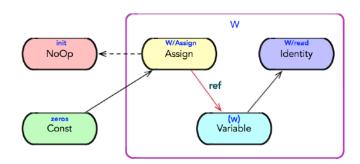


#### 初始化模型



变量

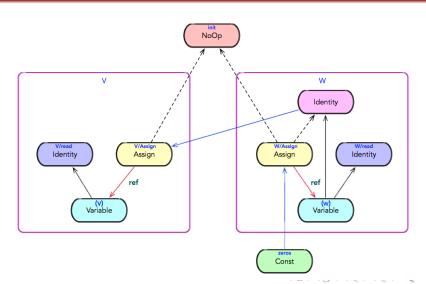
# 探秘 init\_op



19 / 50

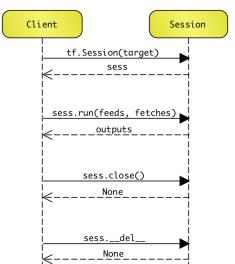
变量

#### 初始化依赖



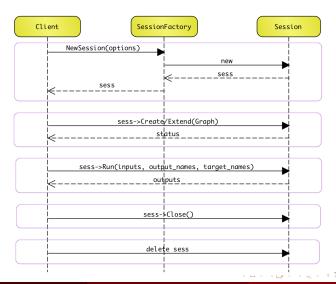
会话

# 生命周期: Python



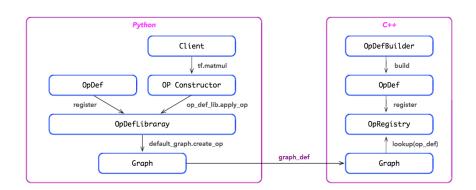
会话

### 生命周期: C++



图构造与执行

## 图构造与传递



#### 实例: OP 构造器

#### **OP** Constructor

```
def zeros_like(tensor, name=None):
    gen_array_ops._zeros_like(tensor, name=name)
tensor = tf.constant([1, 2], name="n1")
zeros = tf.zeros_like(tensor, name="n2")
```

#### Code Generator

```
def _zeros_like(dtype, shape=None, name=None):
    return _op_def_lib.apply_op("ZerosLike", x=x, name=name)
```



### 实例: 构造 OP

#### **OpDef Repository**

#### Graph



### 实例:图构造

```
node {
  name: "n1"
  op: "Const"
  attr {
    key: "dtype"
                                  constant/n1
   value { type: DT_INT32 }
  attr {
                                         src_output=0
    kev: "value"
    value {
      tensor {
        dtype: DT_INT32
        tensor_shape { dim { size: 2 }
        tensor_content: "\001\000\000\000\002\000\000\000"
                                zeros_like/n2
node {
  name: "n2"
  op: "ZerosLike"
                                         src_output=0
  input: "node1"
  attr {
    kev: "T"
    value { type: DT_INT32 }
```

图构造与执行

### 实例:图执行

#### **Graph Execution**



# 训练模型

- 优化算法
- ② 计算梯度
- 3 应用梯度



#### 优化器

```
class Optimizer(object):
  """Add operations to minimize loss by updating var_list.
  def minimize(self, loss, var_list):
   grads_and_vars = self.compute_gradients(loss, var_list)
    return self.apply_gradients(grads_and_vars)
```

# 计算梯度

```
def computea_gradients(loss, var_list):
 grads = gradients(loss, var_list, grad)
  return list(zip(grads, var_list))
def gradients(loss, var_list, grads=1):
 ops\_and\_arads = \{\}
  for op in reversed_graph(loss).topological_sort():
    arad = op.grad_fn(grad)
    ops_and_qrads[op] = qrad
  return [ops_and_grads.get(var) for var in var_list]
```

### 梯度函数

#### @ops.RegisterGradient("op\_name")

def grad\_func(op, grad):

"""construct gradient subgraph for an op type.

Returns:

A list of gradients, one per each input of op.

return cons\_grad\_subgraph(op, grad)

$$(y_1, y_2, ..., y_m) = f(x_1, x_2, ..., x_n)$$

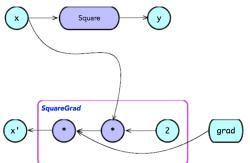
$$(\partial L/\partial x_1, \partial L/\partial x_2, ..., \partial L/\partial x_n) = g(x_1, x_2, ..., x_n; \partial L/\partial y_1, \partial L/\partial y_2, ..., \partial L/\partial y_n)$$

$$\frac{\partial L}{\partial x_i} = \sum_{i=1}^m \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial x_i} \text{ for } i = 1, 2, ..., n$$

→ロト→団ト→重ト→重・釣り○

### 例子: SquareGrad 函数

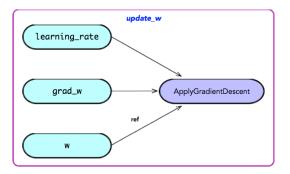
```
@ops.RegisterGradient("Square")
def SquareGrad(op, grad):
    x = op.inputs[0]
    with ops.control_dependencies([grad.op]):
        return grad * (2.0 * x)
```



→ フェータ は マンスで 対光彩 (ZTE) TensorFlow Internals 2017.08 31/50

# 应用梯度

```
def apply_gradients(grads_and_vars, learning_rate):
    for (grad, var) in grads_and_vars:
        apply_gradient_descent(learning_rate, grad, var)
```





应用梯度

# RunStep 过程



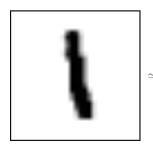
- 提出问题
- ② 单层感知器
- 3 多层感知器
- 调整超参
- ⑤ 卷积神经网络

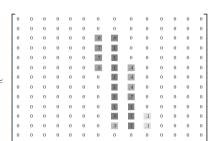


的概述 编程模型 训练模型 **实战 Mnist** 文献 ○○○○○○ ○○○○○○○○○○ ○○○○○○○ ○○○○○○○○

提出问题

#### 图片: 28x28 = 784



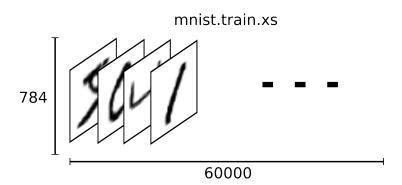


训练模型 000000 实战 Mnist 0●000000000

文献 0000

提出问题

### 训练数据集输入: [100(batch\_size), 784]





刘光聪 (ZTE)

提出问题

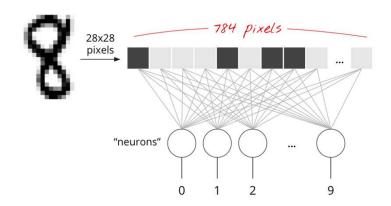
# 训练数据集输出: [100(batch\_size), 10]

# mnist.train.ys 10 60000



刘光聪 (ZTE)

#### 单层网络模型





# 占位符

```
# inputs
x = tf.placeholder("float", [None, 784])
# labels
t = tf.placeholder("float", [None, 10])
```

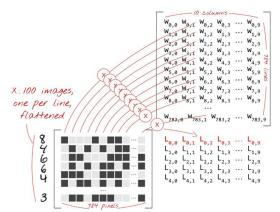
- tf.placeholder 定义了一个占位 OP
- None 表示未确定的样本数目 (batch\_size)
- Session.run 时提供 feed\_dict 提供一个批次的样本数据

```
# train parameters
W = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))
# init_op
init_op = tf.global_variables_initializer()
```

- 变量:使用 tf. Variable 定义变量,常用于定义模型的权重和偏置
- 初始化:使用初始化 init\_op 初始化所有全局变量

# 模型:线性和

y = tf.nn.softmax(tf.matmul(x, W) + b)



#### 模型: Softmax

y = tf.nn.softmax(tf.matmul(x, W) + b)

Predictions | Images | Weights | Biases | 
$$X[100, 10]$$
 |  $X[100, 784]$  |  $X[1$ 

单层感知器

#### 损失函数

Cross entropy: 
$$-\sum Y_i'.log(Y_i)$$
 this is a "6" 
$$computed\ probabilities$$
 0.1 0.2 0.1 0.3 0.2 0.1 0.9 0.2 0.1 0.1 0.1 0.1 0.1 2.3 4.5 6.7 8.9

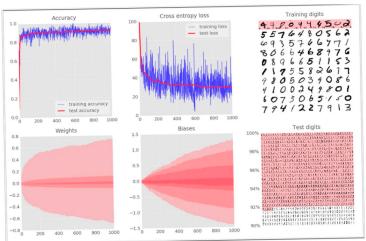
# 执行训练

```
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(t, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
with tf.Session() as sess:
  sess.run(init_op)
  for step in range(1000):
   batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train_step, feed_dict={x: batch_xs, t: batch_ys})
    if step % 10 == 0:
      acc, loss = sess.run([accuracy, cross_entropy],
        feed_dict={x: batch_xs, t: batch_ys})
    if step % 100 == 0:
      acc, loss = sess.run([accuracy, cross_entropy],
        feed_dict={x: mnist.test.images, t: mnist.test.labels})
```

刘光聪 (ZTE) TensorFlow Internals

单层感知器

#### 精度:约为92%









文献

文献

# 文献

- TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, Google Inc.
- TensorFlow: A System for Large-Scale Machine Learning, Google Inc.
- tensorflow.org



致谢

Q&A



文献 ○●00

致谢

# 联系我

- Github: https://github.com/horance-liu
- 简书: http://www.jianshu.com/users/49d1f3b7049e



文献 ○○●○ 致谢

致谢

# **Thanks**

