1 TensorFlow 简介

TensorFlow 是一款通过数据流图进行数值计算的开源库. 在数据流图中, 节点代表数学运算, 边代表连接了节点的多维数组 (张量). 灵活的架构设计让你可以在台式机, 服务器, 甚至移动设备的 CPU, GPU 上面部署运算节点. TensorFlow 最早是被 Google Brain 的研究者和工程师开发出来的用于机器学习和深度神经网络的研究. 但是这个系统也同样适用于其他领域.

TensorFlow 具有很多非常好的特性 [1]:

- **深度灵活性**

  TensorFlow 不是一个僵化的神经网络库, 只要你能够把你的运算表达为数据流图, 你就可以使用 TensorFlow. TensorFlow 提供了现成的神经网络中常见的图, 用户也可以在 TensorFlow 的基础上自己编写库. 编写自定义的复合运算就像写一个 Python 函数那么简单. 如果底层的数据算子没有提供, 你也可以用 C++ 自己写一个.

- **可移植性**

  台式机, 服务器, 甚至移动设备, 都可以部署, 移植 TensorFlow 的程序. 既可以在 CPU 上运行, 也能在 GPU 运行. 如果你想部署到云端, 没问题. Docker 很容易支持 TensorFlow.

- **连接科研与产品**

  研究员通过 TensorFlow 研究新的算法, 产品团队用 TensorFlow 训练模型并把模型集成在产品中提供给用户. 学术界的科学研究和工业界的产品部署可以更加快速地迭代.

- **自动求导**

  如果你使用了基于梯度的学习算法, TensorFlow 的自动求导会给你带来很大的便利. 作为用户, 你只需要自己定义你的预测模型的计算架构和目标函数, 只需要加上数据, TensorFlow 可以自动为你计算导数.

- **语言选择**

  TensorFlow 提供了方便易用的 Python 接口, 也支持 C++, Java, Go 等语言.

- **性能最大化**

  你可以把你的计算单元部署到 CPU, GPU 等不同的设备上面, 并使用线程, 队列, 异步计算来最大化你的硬件使用效率. TensorFow 为你提供一切便利.

TensorFlow 使用数据流图来规划计算流程, 它可以将计算映射到不同的硬件和操作系统平台. 凭借着统一的架构, TensorFlow 可以方便地部署到各种平台, 大大简化了真实场景中应用机器学习的难度. 使用 TensorFlow 我们不需要给大规模的模型训练和小规模的应用部署开发两套不同的系统, 避免了同时维护两套程序的成本, TensorFlow 给训练和预测的共同部分提供了一个恰当的抽象. TensorFlow 的计算可以表示为有状态的数据流图, 对于大规模的神经网络训练, TensorFlow 可以让用户简单的实现并行计算, 同时使用不同的硬件资源进行训练, 同步或异步的更新全局共享的模型参数和状态. 将一个串行的 TensorFlow 算法改造成并行的成本也是非常低的, 通常只需要对小部分代码进行改写. 相比于 DistBelief, TensorFlow 的计算模型更简洁灵活, 计算性能显著提升, 同时支持更多的异构计算系统. 大量 Google 内部的 DistBelief 用户转向了 TensorFlow, 他们使用 TensorFlow 进行各种研究和产品开发, 包括在手机上跑计算机视觉模型, 或是训练有数百亿参数, 数千亿数据的神经网络模型. 虽然绝大多数的 TensorFlow 应用都在机器学习及深度学习领域, 但 TensorFlow 抽象出的数据流图也可以应用在通用数值计算和符号计算上, 比如分形图计算或者偏微分方程数值求解. table 1 所示为 TensorFlow 的主要技术特性 [2].

2 TensorFlow 编程模型简介

2.1 核心概念

TensorFlow 中的计算可以表示为一个有向图 (directed graph), 或称计算图 (computation graph), 其中每一个运算操作 (operation) 将作为一个节点 (node), 节点与节点之间的连接成为边 (edge). 这个计算图描述了数据的计算流程, 它也负责维护和更新状态, 用户可以

表 1. TensorFlow 的主要技术特性

编程模型	dataflow-like model (数据流模型)
语言	• Python, C++, Go, Rust, Haskell, Java • 还有非官方的 Julia, JavaScript, R 的支持
部署	code-once, run everywhere (一次便携, 各处运行)
计算资源	• CPU (Linux, Mac, Windows, Android, iOS) • GPU (Linux, Mac, Windows) • TPU (Tensor Processing Unit, 张量计算单)
实现方式	• Local Implementation (单机实现) • Distributed Implementation (分布式实现)
平台支持	• Google Cloud Platform (谷歌云平台) • Hadoop File System (Hadoop 分布式文件系统)
数学表达	• Math Graph Expression (数学计算图表达) • Auto Differentiation (自动微分)
优化	• Common Subexpression Elimination (共同子图消除) • Asynchronous Kernel Optimization (异步核优化) • Communication Optimization (通信优化) • Model Parallelism (模型并行) • Data Parallelism (数据并行) • Pipeline (流水线)

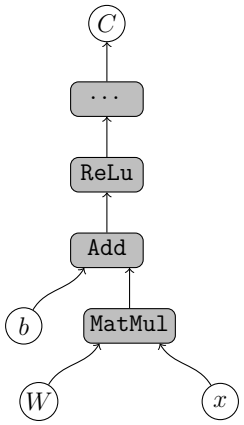
对计算机的分支进行条件控制或循环操作. 用户可以使用 Python, C++, Go, Java 等几种语言设计这个数据计算的有向图. 计算图中每一个节点可以有任意多个输入和任意多个输出, 每一个节点描述了一种运算操作, 节点可以算是运算操作的实例化 (instance). 在计算图的边中流动 (flow) 的数据被称为张量 (tensor), 故得名 TensorFlow. 而 tensor 的数据类型, 可以是事先定义的, 也可以根据计算图的结构推断得到. 有一类特殊的边没有数据流动, 这种边是依赖控制 (control dependencies), 作用是让它的起始节点执行完之后再执行目标节点, 用户可以使用这样的边进行灵活的条件控制, 比如限制内存使用的最高峰值. 下面是用 Python 设计并执行计算图的示例. 计算图示例如 fig. 1 所示.

23333

```python
import tensorflow as tf


b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784, 100], -1, 1))
x = tf.placeholder(name = 'x')
relu = tf.nn.relu(tf.matmul(W, x) + b)
C = [...]
sess = tf.Session()
for step in range(0, 10):
    input = ... construct 100-D input array ...
    d23
    result = sess.run(C, feed_dict = {x:input})
    print(step, result)
```

图 1. 计算图示例



一个运算操作代表了一种类型的抽象运算, 比如矩阵乘法或者向量加法. 运算操作可以有自己的属性, 但是所有属性必须预先设置, 或者能在创建计算图时被推断出来. 通过设置运算操作的属性可以用来支持不同的 tensor 元素类型, 比如让向量加法支持浮点数