TensorFlow初识

Yasaka 陈博

TensorFlow概要

- 由Goole Brain开源，设计初衷是加速机器学习的研究
- 2015年11月在GitHub上开源
- 2016年4月分布式版本
- 2017年发布了1.0版本，趋于稳定
- Google希望让这个优秀的工具得到更多的应用，从整体上提高深度学习的效率

Google大量成功项目

- Android系统
- Chromium浏览器
- Go编程语言
- JavaScript引擎V8
- Protobuf数据交换框架
- Bazel编译工具
- OCR工具Tesseract

TensorFlow相关链接

- TensorFlow官方网址：www.tensorflow.org
- GitHub网址：github.com/tensorflow/tensorflow
- 模型仓库网址：github.com/tensorflow/models

支持语言

- Python
- C++
- Go
- Java
- 后端使用C++、CUDA

TensorFlow

- TensorFlow实现的算法可以在众多异构的系统上方便地移植，比如Android手机、iphone、
- 普通的CPU服务器、大规模GPU集群

- 除了执行深度学习算法，TensorFlow还可以用来实现很多其他算法，包括线性回归、逻辑回归、随机森林等
- TensorFlow建立的大规模深度学习模型应用场景也非常广，包括语音识别、自然语言处理、计算机视觉、机器人控制、信息抽取、药物研发、分子活动预测

在Google的应用

- 为了研究超大规模的深度神经网络，Google在2011年启动了Google Brain项目
- 比如Google Search中的搜索结果排序
- Google Photos中的图片标注
- Google Translate中的自然语言处理，都依赖建立的深度学习模型
- 2016年已经有超过2000个项目使用了TensorFlow建立的深度学习模型

核心概念

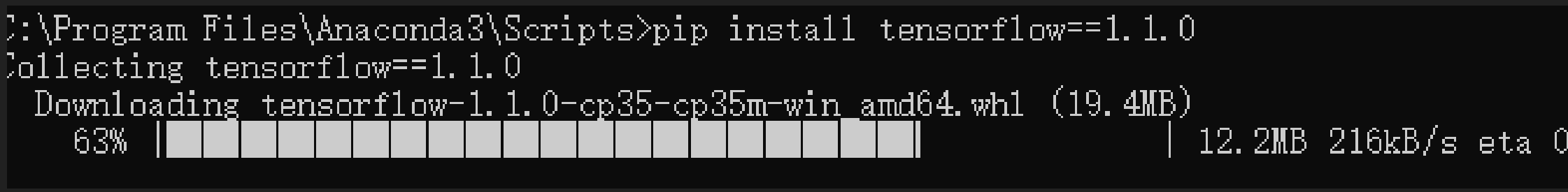
- TensorFlow中的计算可以表示为一个有向图（Directed Graph）
- 或者称计算图（Computation Graph）
- 其中每一个运算操作（operation）将作为一个节点（node）
- 计算图描述了数据的计算流程，也负责维护和更新状态
- 用户通过python，c++，go，Java语言设计这个这个数据计算的有向图
- 计算图中每一个节点可以有任意多个输入和任意多个输出
- 每一个节点描述了一种运算操作，节点可以算是运算操作的实例化（instance）
- 计算图中的边里面流动（flow）的数据被称为张量（tensor），故得名TensorFlow

代码流程

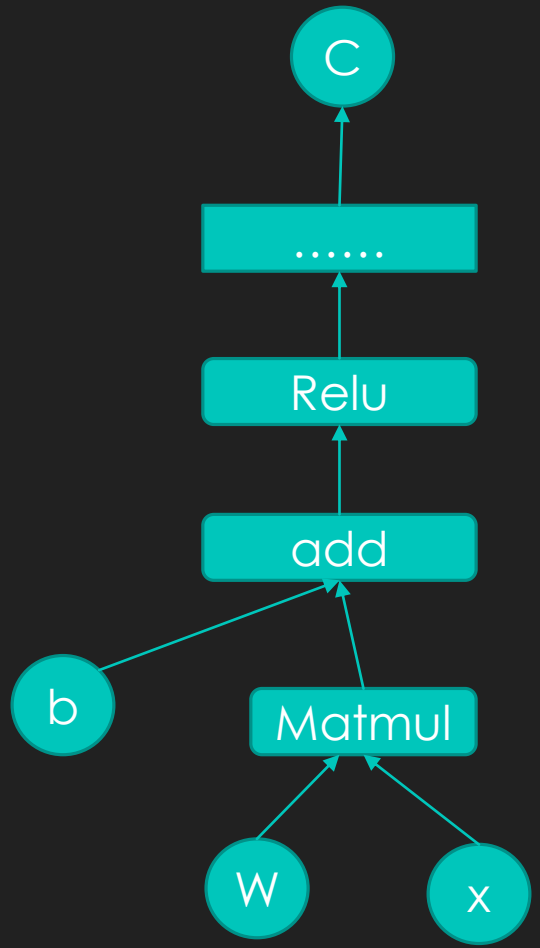
- Import tensorflow as tf
- b = tf.Variable(tf.zeros([100]))
- W = tf.Variable(tf.random_uniform([784,100], -1, 1))
- x = tf.placeholder(name="x")
- relu = tf.nn.relu(tf.matmul(W, x) + b)
- Cost = […]
- Sess = tf.Session()
- for step in range(0, 10):
- input = …construct 100-D input array…
- result = sess.run(cost, feed_dict={x: input})
- print(step, result)

安装tensorflow

- pip install tensorflow==1.1.0



有向无环图DAG



其他深度学习框架

- 深度学习研究的热潮持续高涨，各种开源深度学习框架也层出不穷，其中包括TensorFlow、Caffe、Keras、CNTK、Torch7、MXNet、Leaf、Theano、DeepLearning4、Lasagne、Neon
- TensorFlow杀出重围，在关注度、用户数上都占据绝对优势，大有一统江湖之势
- TensorFlow在Star数量，Fork数量，Contributor数量这三个数据上都完胜其他对手，主要是Google在业界的号召力确实强大，Google强大的人工智能研发水平，让大家对Google的深度学习框架充满信息

Python独领风骚

- 各大主流框架基本都支持Python，目前python在科学计算和数据挖掘领域可以说是独领风骚
- Python的各种库实在是太完善了，web开发，数据可视化，数据预处理，数据库连接，爬虫等无所不能，有一个完美的生态环境
- 数据挖掘工具链上，python就有Numpy，Scipy，Pandas，Scikit-Learn，XGBoost等组件
- 做数据采集和预处理都非常方便，并且之后的模型训练阶段可以和TensorFlow等基于python的深度学习框架完美衔接

TensorFlow

- TensorFlow是相对高阶的机器学习库，用户可以方便的用它设计神经网络结构，而不必为了追求高效率的实现亲自写C++或CUDA代码。
- 它和Theano一样都支持自动求导，用户不需要再通过反向传播求解梯度
- 其核心代码和Caffe一样是用C++编写的，使用C++简化了线上部署的复杂度，并让手机这种内存和CPU资源都紧张的设备可以运行复杂模型（Python则会比较耗费资源，并且执行效率不高）
- 有C++接口，也有python、Go、Java接口，这样用户就可以在一个硬件配置较好的机器中用python进行实验，并在资源比较紧张的嵌入式环境或需要低延迟的环境中用C++部署模型
- TensorFlow也有内置的TF.Learn和TF.Slim等上层组件可以帮助快速的设计新网络，并且兼容Scikit-Learn estimator接口，可以方便的实现evaluate、grid search、cross validation等功能。
- TensorFlow不只局限于神经网络，器数据流式图支持非常自由的算法表达，可以轻松实现深度学习以外的机器学习算法，只要可以将计算表示成计算图的形式，就可以使用TensorFlow

TensorFlow

- TensorFlow的另外一个重要特点是它灵活的移植性，可以将同一份代码几乎不经过修改就轻松地部署到有任意数量CPU或GPU的PC、服务器或者移动设备上
- TensorFlow还有强大的可视化组件TensorBoard，能可视化网络结构和训练过程，对于观察复杂的网络结构和监控长时间、大规模的训练很有帮助

TensorBoard

- TensorBoard是TensorFlow的一组Web应用，用来监控TensorFlow运行过程，或可视化目前支持5种可视化：标量（scalars）、图片（images）、音频（audio）、直方图（histograms）、计算图（computation graph）
- TensorBoard的Event Dashboard可以用来持续地监控运行时的关键指标，比如loss，学习速率（learning rate）或者是验证集上的准确率（accuracy）
- Image Dashboard可以展示训练过程中用户设定保存的图片，比如某个训练中间结果用Matplotlib等绘制plot出来的图片
- Graph Explorer则可以完全展示一个TensorFlow的计算图，并且支持缩放拖拽和查看节点属性

Caffe

- Convolutional Architecture for Fast Feature Embedding
- 一个被广泛使用的开源深度学习框架，由伯克利视觉中心进行维护
- 特点一：容易上手，网络结构都是以配置文件形式定义，不需要用代码设计网络
- 特点二：训练速度快，能够训练state-of-the-art的模型与大规模的数据
- 特点三：组件模块化，可以方便地拓展到新地模型和学习任务上
- Caffe核心概念是Layer，每一个神经网络地模块都是一个Layer。Layer接收输入数据，同时经过内部计算产生输出数据。设计网络，需要把各个Layer拼接在一起构成完整地网络（通过写protobuf配置文件定义）
- 比如卷积Layer，它的输入就是图片全部像素点，内部进行的操作是各种像素的值与Layer参数的convolution操作，最后输出的是所有卷积核filter的结果。
- 每一个Layer需要定义两种运算，一种是正向forward运算，从输入数据计算出输出结果

Caffe

- 另一种是反向backward运算，从输出端的gradient求解相对于输入的gradient，即反向传播算法，这部分也是模型的训练过程
- 实现Layer的时候，需要将正向和反向两种计算过程的函数都实现，这部分计算需要用户自己写C++或者CUDA（当运行在GPU时候）代码，对于普通用户来说非常难于上手
- Caffe最初设计目标只针对图像，因此对卷积神经网络的支持非常好
- Caffe优势是有大量训练好的经典模型（AlexNet、VGG、Inception）乃至ResNet模型，收藏与它的Model Zoo，github.com/BVLC/caffe/wiki/Model-Zoo，在计算机视觉领域Caffe应用尤其多，可以用来做人脸识别、图片分类、位置检测、目标追踪等
- 底层基于C++，可以在各种硬件环境编译并具有很好的移植性，支持Linux、Mac和Windows系统，可以编译部署到移动设备系统Android和IOS上
- 也提供了pycaffe支持python语言，设计网络可以使用python接口简化操作，不过还是使用protobuf配置文件定义神经网络结构，再用command Line进行训练和或者预测。

Caffe

- 配置文件是一个类似JSON的prototxt文件，其中使用许多顺序连接的Layer来描述神经网络结构
- 在prototxt文件中设计网络结构比较受限，没有像TensorFlow或者Keras那样在python中设计网络结构方便、自由。
- 配置文件不能用编程方式调整超参数，也没有像sklearn那样好用的estimator可以方便进行交叉验证、超参数的Grid Search
- Caffe在GPU上性能很好，一个GTX 1080训练AlexNet一天可以训练上百万张图片，但是目前仅支持单机多GPU训练，原生没有支持分布式训练，不过有CaffeOnSpark，借助雅虎开源的技术结合spark分布式框架实现Caffe大规模分布式训练

Theano

- 与sklearn一样，Theano很好的整合了Numpy
- 因为Theano非常流行，有许多人写了高质量文档和教程，用户方便查找Theano的各种FAQ，比如如何保持模型，如何运行模型等，不过它更多被当作一个研究工具，而不是当作产品来使用
- Theano在单GPU上执行效率不错，性能和其他框架类似，但是运算时需要将用户的python代码转换成CUDA代码，再编译为二进制可执行文件，编译复杂模型时间非常久
- 不过，Theano在训练简单网络比如很浅的MLP时性能可以比TensorFlow好，因为全部代码都是运行时编译，不需要像TensorFlow那样每次喂给mini-batch数据时候都得通过低效的python循环来实现
- Theano是一个完全基于Python的符合计算库，不需要像Caffe一样为Layer写C++或CUDA代码，用户定义各种运算，Theano可以自动求导，省去了完全手工写神经网络反向传播的麻烦，也不需要像Caffe一样为Layer写C++或者CUDA代码

Theano

- 如果没有Theano，可能根本不会出现这么多好的python深度学习框架，就如没有python的科学计算的基石Numpy，就不会有Scipy、Sklearn、Skimage，可以说Theano就是深度学习界的Numpy，是其他各类python深度学习库的基石
- 但是直接使用Theano来设计大型的神经网络还是太繁琐了，用Theano实现Google Inception就像用Numpy实现一个支持向量机SVM
- 事实上，不需要 总是从最基础的tensor粒度开始设计网络，而是从更上层的Layer粒度来设计网络

Keras

- Theano派生出了大量基于它的深度学习库，包括一系列的上层封装，其中大名鼎鼎的Keras对神经网络抽象得非常合适，以致于可以随意切换执行计算得后端（目前同时支持Theano和TensorFlow）
- Keras比较适合在探索阶段快速地尝试各种网络结构，组件都是可插拔的模块，只需要将一个个组件，比如卷积层、激活函数连接起来，但是设计新的模块或者新的Layer就不太方便了
- 它是高度模块化、极简的神经网络库，用python实现，同时运行在TensorFlow和Theano上，意在让用户进行最快的原型试验，让想法变为结果的过程最短
- Theano和TensorFlow的计算图支持更通用的计算，而Keras专精于深度学习
- Theano和TensorFlow更像深度学习领域的Numpy，而Keras是深度学习的sklearn
- 神经网络、损失函数、优化器、初始化方法、激活函数、和正则化等模块都是可以自由组合的
- Keras中的模型都是在python中定义的，不信Caffe和CNTK等需要额外的配置文件来定义模型

Keras

- 通过编程方式调试模型结构和各种超参数
- 几行代码就可以实现一个MLP，十几行代码实现一个AlexNet，这在其它框架里面是不可能完成的任务
- Keras问题在于目前无法直接使用多GPU，所有对大规模的数据处理速度没有其他支持多GPU和分布式的框架快