# 9种深度学习框架的比较分析

Jiechao Cheng

Nov. 28th, 2016

针对目前比较热门的几种深度学习框架 (Caffe, CNTK, GraphLab, GraphX, MXNet, Petuum, TensorFlow, Theano, Torch)，本文分别从相关的背景、特性、优势和不足展开。

# 1.Caffe

## 背景：

Caffe全称Convolutional Architecture for Fast Feature Embedding，是第一个主流的工业级深度学习工具，诞生于2013年底，由UC Berkely的Yangqing Jia老师编写和维护的出色的卷积神经网络实现。它是一个清晰、高效、开源的深度学习框架，将Matlab实现的快速卷积网络移植到了C和C++平台上，用于图像分类。Caffe是目前最流行的深度学习框架之一，在计算机视觉领域依然是最流行的工具包。

## 特性：

开发者：Berkeley Vision and Learning Center

底层语言：C++

接口语言：命令行、Python、Matlab

运行方式：CPU、GPU

操作系统：Windows、Linux、Mac OS X

安装方式：需要配置相关Python库，需要CUDA支持，需要OpenCV支持

支持的深度学习网络：支持大部分先进的网络

集成的数据集：CIFAR10、ILSVRC12、MNIST

## 优势：

(a) 表达方便：定型模型和优化办法的表达用的是纯文本办法表示，而不是代码

(b) 速度快：对于科研来说，接近工业化的速度处理大规模数据，拥有当前最牛掰的算法

(c) 模块化：具有新任务和设置需要的灵活性和扩展性

(d) 开放性：科学研究和应用程序可调用同样的代码，参考模型，并且结果可重现

(e) 社区性：学术研究、启动原型和工业应用领域的人以BSD-2项目的形式共同研讨开发

(f) 易用性：网络开发和构建过程简单，实现了多种接口，跨平台，Python界面相当有用

(g) 适用性：适合前馈网络和图像处理，适合微调已有的网络

(h) 流行性：由出色的卷积神经网络实现，在计算机视觉领域依然很流行

## 不足：

(a) 虽有很多扩展，但由于遗留的架构问题，不够灵活且对递归网络和语言建模支持很差

(b) 基于层的网络结构，扩展性差，对于新增加的层，需要自行实现前后向传播和梯度更新

(c) 运用于大型网络 (GoogLeNet、ResNet) 时过于繁琐

(d) 支持pycaffe接口，仅用来辅助命令行接口；使用pycaffe时必须用protobuf定义模型

(e) 需要用C++ / CUDA编写新的GPU层

(f) 不适用于文本、声音或时间序列数据等其他类型的深度学习应用

# 2.CNTK

## 背景：

CNTK (Computational Network Toolkit) 是微软的开源深度学习框架，是一个统一的深度学习工具包，该工具包通过一个有向图将神经网络描述为一系列计算步骤。在有向图中，叶节点表示输入值或网络参数，其他节点表示该节点输入之上的矩阵运算。CNTK通过符号化分析流程图来计算梯度下降训练算法中所用到的梯度值。虽然CNTK遵循一个比较宽松的许可协议，却并未采用ASF 2.0、BSD或MIT等一些较为传统的许可协议。

## 特性：

开发者：Microsoft

底层语言：C++

接口语言：Python、C/C++、命令行

运行方式：CPU、GPU

操作系统：Windows、Linux

安装方式：安装相关依赖库并编译源码

支持的深度学习网络：前馈DNN网络、卷积网络、循环网络

## 优势：

(a) 通过细粒度的构件块而不需要使用低层次的语言就能创建新的、复杂的层类型

(b) 跨平台，大部分情况下部署非常简单

(c) 适合并行计算，支持大规模GPU

(d) 实现了跨多GPU和服务器自动分化和并行化的随机梯度下降 (误差反向传播) 学习

## 不足：

(a) 不支持ARM架构，这限制了它在移动设备上的能力

(b) 现阶段只支持一种学习方法：Mini-batch随机梯度下降法

# 3.GraphLab

## 背景：

GraphLab 是由CMU Select 实验室在2010 年提出的一个基于图像处理模型的开源图计算框架，框架使用C++语言开发实现。该框架是面向机器学习（ML）的流处理并行计算框架，可以运行在多处理机的单机系统、集群或是亚马逊的EC2 等多种环境下。GraphLab 自成立以来就是一个发展很迅速的开源项目，其用户涉及的范围也相当广泛，全球有2 000 多个企业、机构使用GraphLab。

## 特性：

开发者：CMU Select

底层语言：C++

接口语言：C++、Java、Python

运行方式：CPU、GPU

操作系统：Windows、Linux、Mac OS X

安装方式：安装相关依赖库并编译源码

支持的深度学习网络： CNN、RNN

集成的数据集：CIFAR10、ILSVRC12、MNIST

数据模型：Graph结构

执行模型：执行过程抽象成Gather、Apply、Scatter三个步骤

抽象类型：图并行抽象

并行机制：对顶点切分，顶点是最小并行粒度和通信粒度，边是算法中数据依赖性的表示

## 优势：

(a) API接口统一：对于多核处理器和分布式集群环境，一次编写程序即可高效地运行

(b) 高性能：优化C++执行引擎，在大量多线程操作和同步I/O操作之间能很好地平衡

(c) 可伸缩性强：能够智能地选择存储和计算的节点，算法设计优良

(d) 集成HDFS：内置支持HDFS，能直接读取HDFS中数据或者将数据直接写入HDFS

(e) 工具集功能强大：在自身提供的API 接口之上实现了大量开箱即用的工具集

(f) 框架高度抽象：可以高效并行执行具有稀疏的计算依赖性强的迭代性算法

(g) 计算过程中数据保持高度一致

(h) 适用于大规模机器学习任务，也适用于许多数据挖掘方面的计算任务

## 不足：

(a) 抽象成图不直接：深度学习多层结构中，已有算法移植到基于图抽象上需要不少工作

(b) 基于图的抽象有时还会导致程序不正确，或者陷入次优化

# 4.GraphX

## 背景：

GraphX最先是Berkeley AMPLAB的一个分布式图计算框架项目，后来整合到Spark中成为一个核心组件。GraphX是从表到图、允许图与表的交互，融合了图并行以及数据并行的优势，是Spark中用于图 (如社交网络) 和图并行计算 (如PageRank、协同过滤) 的API，可以认为是GraphLab (C++) 和Pregel (C++) 在Spark (Scala) 上的重写及优化。GraphX最大的贡献是，在Spark之上提供一栈式数据解决方案，可以方便且高效地完成图计算的一整套流水作业。

## 特性：

开发者：Berkeley AMPLAB

底层语言：Java

接口语言：Java、C#、Python、Scala

运行方式：CPU、GPU

操作系统：Windows、Linux、Mac OS X

安装方式：安装相关依赖库并编译源码

支持的深度学习网络：CNN

集成的数据集：CIFAR10、MNIST

## 优势：

(a) 底层是基于Spark，天然就是一个分布式的图处理系统。

(b) 基于Spark平台提供对图计算和图挖掘简洁易用且丰富的接口，极大方便分布式图处理

(c) 在Spark之上提供一栈式数据解决方案，完成图计算的一整套流水作业方便且高效

(d) 从整个图处理流水线视角（图构建、图合并、最终结果查询）看，性能非常具有竞争性

(e) 借助Spark一体化流水线处理、社区活跃度以及快速改进速度，有竞争优势

(f) 操作灵活且执行效率高

(g) 代码很简洁，只有3千多行；而在此之上实现的 Pregel 模式，只有20 多行

## 不足：

(a) 与GraphLab相比，运行速度有一定差距

(b) 在单机上的计算性能不如GraphLab等计算框架

# 5.MXNet

## 背景：

MXNet是一个兼具效率和灵活性的开源深度学习框架，由李沐和陈天奇等各路英雄豪杰打造，是分布式机器学习通用工具包DMLC 的重要组成部分。它注重灵活性和效率，文档也非常的详细，同时强调提高内存使用的效率，甚至能在智能手机上运行诸如图像识别等任务。MXNet允许使用者将符号编程和命令式编程相结合，以追求效率和生产力的最大化。

## 特性：

开发者：分布式 (深度) 机器学习社区

底层语言：C++

接口语言：C++、Python、Julia、Matlab、JavaScript、R、Scala

运行方式：CPU、GPU

操作系统：Linux、Mac OS X、Windows、AWS、Android、iOS

安装方式：需要Python和CUDA支持

支持的深度学习网络：随机储存器/动态贝叶斯网络、卷积网络、循环网络

软件证书：Apache 2.0

## 优势：

(a) 强调提高内存使用的效率，甚至能在智能手机上运行诸如图像识别等任务

(b) 整合了各种编程方法的优势，最大限度地提高灵活性和效率

(c) 对“云计算”友好，直接兼容S3、HDFS和Azure

(d) 有预训练模型，表现、记忆效果好

(e) 对R的支持很出色，是唯一支持所有R函数的构架

(f) 符号执行和自动求导，运行依赖引擎，内存节省

## 不足：

(a) 迁移学习和微调网络可能实现，但和Lasagne、Keras 相比不太容易

(b) 编辑现存训练好的网络更困难，因此使用特定域的自定义数据也有点难

# 6.Petuum

## 背景：

Petuum由CMU计算机学院Eric Xing教授创建，专门为机器学习而设计。Petuum可使用户从复杂繁琐的分布式系统编程和调试中解脱出来，将更多精力集中在优化模型和算法上；提供了包括协同过滤、稀疏逻辑回归、主题模型、矩阵分解，以及深度学习等重要的大规模机器学习算法。Petuum可高效地运行于各种硬件环境下，包括实验室集群服务器和云计算平台，如Amazon EC2、Google GCE。

## 特性：

开发者：CMU Eric Xing

底层语言：C++

接口语言：C++、Java

运行方式：CPU、GPU

操作系统： Linux、Ubuntu

安装方式：安装相关依赖库并编译源码

支持的深度学习网络：支持大部分先进的网络

集成的数据集：CIFAR10、ILSVRC12、MNIST

## 优势：

(a) 在不损失模型性能的前提下，可在更少的硬件上更快地运行更大模型

(b) 高性能，最大化系统性能

(c) 可编程性，提供了简单易用的编程接口，用户可实现自己的机器学习算法

(d) 丰富的机器学习库

(e) 运行于集群服务器和云计算平台，可支持几百台机器

(f) 支持Hadoop生态系统，如YARN和HDFS

(g) 充分利用机器学习算法的特征，如迭代性，容错性，参数收敛的不均匀性等

## 不足：

(a) 目前主要适用于几十台机器的集群，在更大集群上处理有风险

(b) 调度器设计有点复杂

# 7.TensorFlow

## 背景：

TensorFlow是谷歌发布的第二代开源机器学习系统，被用在Google搜索、图像识别以及Gmail邮箱的深度学习框架中。TensorFlow是一个利用数据流图 (Data Flow Graphs) 进行数值计算的开源软件库，图中的节点 (Nodes) 代表数学运算操作，图中的边 (Edges) 表示节点之间相互流通的多维数组，即张量 (Tensors)。这种灵活的架构可以让使用者可以多样化地将计算部署在台式机、服务器或者移动设备的一个或多个CPU上，而且无需重写代码。

## 特性：

开发者：Google Brain Team

底层语言：C++、Python

接口语言：Python、C/C++

运行方式：CPU、GPU

操作系统：Linux、Mac OS X

安装方式：需要安装Python支持

支持的深度学习网络：随机储存器/动态贝叶斯网络、卷积网络、循环网络

集成的数据集：CIFAR10、ILSVRC12、MNIST

软件证书：Apache 2.0

## 优势：

(a) 理想的RNN API和实现，非常好用的网络结构可视化工具，有利于分析训练网络

(b) 使用了向量运算的符号图方法，使得新网络的指定变得相当容易，支持快速开发

(c) 编译过程比Theano快，简单把符号张量操作映射到已经编译好的函数调用

(d) 支持使用ARM/NEON指令实现model decoding

(e) 架构清晰，采用了模块化设计，支持多种前端和执行平台

(f) 通过符号化分析流程图来计算梯度下降训练算法中所用到的梯度值

(g) 为并行计算而设计，支持 CUDA，大规模GPU支持才是真正的设计点

(h) 自动求导 (微分)，因此每次尝试新的神经网络设计就要手动自动编写新的反向传播算法

(i) 不止用于深度学习，还有支持强化学习和其他算法的工具

(j) 同时支持数据并行和模型并行，与Theano类似的计算图抽象化

(k) 支持Python以及NumPy库

## 不足：

(a) 速度比其他框架慢，运行时间是最新深度学习工具的四倍，内存占用较大 (如相比Torch)

(b) 支持的层没有Torch和Theano丰富，特别是没有时间序列的卷积

(c) 卷积也不支持动态输入尺寸，这些功能在NLP中非常有用

(d) 不支持双向RNN和3D卷积，同时公共版本的图定义也不支持循环和条件控制

(e) RNN实现不理想，因为必须要使用Python循环且无法进行图编译优化

(f) 目前不支持所谓的内联矩阵运算，必须要复制矩阵才能对其进行运算

(g) 复制非常大的矩阵会导致成本全面偏高

(h) 计算图纯粹基于Python，所以速度较慢，也不适合Java和Scala用户群

(i) 比Torch笨重许多，也更难理解

(j) 已预定型的模型不多

# 8.Theano

## 背景：

Theano 2008年诞生于蒙特利尔理工学院，主要开发语言是Python。Theano派生出了大量深度学习Python软件包，和NumPy一样，是处理多维数组的开源学习库，最著名的包括Blocks和Keras。Theano是深度学习模型的极佳选择，它允许使用者有效地定义、优化和评估涉及多维数组的数学表达式，同时支持GPU和高效符号分化操作，很容易用 Lasagne/Keras 实现新网络或者编辑现存网络。Theano可与其他学习库配合使用，非常适合数据探索和研究活动。

## 特性：

开发者：Université de Montréal

底层语言：Python

接口语言：Python、C++

运行方式：CPU、GPU

操作系统：Windows、Linux

安装方式：需要配置相关Python库 需要CUDA支持

支持的深度学习网络：支持大部分先进的网络，网络需要自己实现，缺少现成案例

集成的数据集：无

软件证书：BSD证书

预训练模型：通过Lasagne的model zoo

## 优势：

(a) 非常灵活，适合做学术研究实验，且对递归网络和语言建模有较好的支持

(b) 对于全局最优搜索的后阶段梯度准确率要求高的搜索方法，能给出更有效的结果

(c) 第一个使用符号张量图描述模型的架构，引领了符号图在编程网络中使用的趋势

(d) 支持更多的平台

(e) 在其上有可用的高级工具：Blocks、Keras等

(f) 支持 CUDA，自动微分

(g) 与NumPy紧密相关

(h) 透明地使用GPU (执行数据密集型计算比CPU快140多倍)

(i) 速度和稳定性的优化 (即使输入的x非常小也可以得到log(1+x)正确结果)

(j) 动态生成C代码 (表达式计算更快)

(k) 广泛的单元测试和自我验证 (多种错误类型的检测和判定)

(l) API支持循环控制，容易且高效地实现RNN

(m) 计算图是良好的抽象化方式，计算图与RNN匹配良好

(n) 高级的包装界面 (Keras、Lasagne) 减少了使用时的麻烦

## 不足：

(a) 速度较慢，编译过程慢，但同样采用符号张量图的TensorFlow无此问题

(b) 启动时间长，缺乏底层接口，Python解释器很低效，对工业用户缺少吸引力

(c) 开发者难改进，因为底层代码是Python，C/CUDA代码被打包在Python字符串中

(d) 原始的Theano级别偏低，对已预定型模型的支持不够完善

(e) 错误信息可能没有帮助

(f) 大型模型的编译时间可能较长

(g) 比Torch笨重许多，更难理解

# 9.Torch

## 背景：

Torch由NYU最初开发，Facebook力推的深度学习框架，主要开发语言是C和Lua，Lua是上世纪九十年代早期在巴西开发的多范例脚本语言，曾被用来开发视频游戏。Torch有较好的灵活性和速度，支持机器学习算法。Torch诞生已经有十年之久，但是真正起势得益于2015年Facebook开源了大量Torch的深度学习模块和扩展，谷歌DeepMind等大型科技公司也使用Torch的某些版本，由内部团队专门负责定制自己的深度学习平台。Ubuntu平台最容易搭建，CentOS需要仔细调整OS包。最新版Torch7虽然功能强大，但其设计并不适合在两个群体中大范围普及，即主要依赖Python的学术界，以及普遍使用Java的企业软件工程师。

## 特性：

开发者：Facebook

底层语言：用快速的脚本语言LuaJIT开发，底层是C/CUDA实现

接口语言：Lua语言

运行方式：CPU、GPU

操作系统：Windows、Linux、Mac OS X

安装方式：需要安装Lua语言支持

支持的深度学习网络：支持大部分先进的网络

## 优势：

(a) 有较好的灵活性和速度

(b) 实现并优化了基本的计算单元，使用者可以很简单地在此基础上实现自己的算法

(c) 核心计算单元使用C或者CUDA做了很好的优化，使用Lua构建了常见模型

(d) 支持全面的卷积网络操作

(e) 支持时间卷积：输入长度可变，对NLP非常有用，而TF和Theano都不支持

(f) 支持3D卷积：对视频识别很有用，而TF不支持

(g) 通过很多非官方的扩展支持大量的RNN，同时网络的定义方法有很多种

(h) 大量模块化组件，容易组合，有很多已预定型的模型

(i) 很容易编写自己的层类型并在GPU上运行

(j) 多数学习库的代码也是Lua，容易读取

## 不足：

(a) 接口为Lua语言，通常需要自己编写定型代码 (即插即用相对少)，需要一点时间学习

(b) 目前没有Python接口

(c) 与Caffe一样，基于层的网络结构，扩展性不好，对于新增加的层，需要自己实现

(d) RNN没有官方支持，不太适合递归神经网络

(e) 以图层的方式定义网络，这种粗粒度的方式使得其对新图层类型的扩展缺乏足够支持

(f) 模型运行需要LuaJIT的支持，虽对性能影响不大，但对集成造成很大障碍

(g) 吸引力不如Caffe、CNTK、TensorFlow等直接支持C++的框架

# 参考

深度学习框架的评估与比较

<http://www.infoq.com/cn/news/2016/01/evaluation-comparison-deep-learn>

Caffe、TensorFlow、MXnet三个开源库对比

<http://chenrudan.github.io/blog/2015/11/18/comparethreeopenlib.html>

CVPR 2015深度学习回顾：ConvNet、Caffe、Torch及其他

<http://www.csdn.net/article/1970-01-01/2825395>

torch7怎么样？和theano和caffe相比如何？顺便问下实现cnn麻烦吗？

<https://www.zhihu.com/question/34789475>

深度学习简析,TensorFlow,Torch,Theano,Mxn

<http://lchiffon.github.io/2015/11/16/long.html>

<http://tensorflow.org/>

<http://mxnet.readthedocs.org/en/latest/index.html>

<http://caffe.berkeleyvision.org/>

[[caffe]的项目架构和源码解析](http://chenrudan.github.io/blog/2015/05/07/cafferead.html)

[如何评价Tensorflow和其它深度学习系统](http://weibo.com/p/1001603907610737775666)

[Imagenet Winners Benchmarking](https://github.com/soumith/convnet-benchmarks)

[Blobs, Layers, and Nets: anatomy of a Caffe model](http://caffe.berkeleyvision.org/tutorial/net_layer_blob.html)

[Deep Learning in a Single File for Smart Devices](http://mxnet.readthedocs.io/en/latest/how_to/smart_device.html#demo-apk-download)

MXNet设计和实现简介

TensorFlow vs. Theano vs. Torch comparison