# TensorFlow™源代码分析

## 学院：信息学院

## 姓名：杨旭

## 学号：2017104104

## 背景介绍

谷歌发布了全新人工智能系统TensorFlow。该系统可被用于语音识别或照片识别等多项机器深度学习领域，主要针对2011年开发的深度学习基础架构DistBelief进行了各方面的改进，它可在小到一部智能手机、大到数千台数据中心服务器的各种设备上运行。

TensorFlow是谷歌研发的第二代人工智能学习系统，而第一代的DistBelief比这个要早好多年。

DistBelief诞生于2011年，它是谷歌推出的第一代内部深度学习结构，能够帮助谷歌利用自家的数据中心构建大型的神经网络，主要应用于人工智能的开发，比如语音识别、图片搜索等等。

但是，DistBelief本身存在一些技术上的短板，对谷歌的人工智能发展仍有一些限制。如谷歌高级研究员Jeff Dean和技术主管Rajat Monga表示，DistBelief关注的重心是神经网络，而且与谷歌内部的基础架构联系紧密。也就是说，该系统“几乎不可能与外部共享研究代码”，而且使用起来比较难设置。

而TensorFlow正是基于第一代DistBelief进行开发的，其命名来源于本身的运行原理，Tensor（张量）意味着N维数组，Flow（流）意味着基于数据流图的计算，TensorFlow即为张量从图象的一端流动到另一端——将复杂的数据结构传输至人工智能神经网中进行分析和处理。

相比较之下，作为谷歌第二代人工智能系统，TensorFlow更快、更智能化，也更加灵活，可以更加轻松地应用于谷歌的新产品以及支持技术研究。按某些标准计算，TensorFlow的运行速度相当于DistBelief的3倍。

此外，TensorFlow一大亮点是支持异构设备分布式计算，它能够在各个平台上自动运行模型，从手机、单个CPU / GPU到成百上千GPU卡组成的分布式系统。也就是说，任何基于梯度的机器学习算法都能够受益于TensorFlow的自动分化（auto-differentiation）。

## 谷歌为什么要开源？

鉴于TensorFlow系统的强大功能，加上谷歌正计划在TensorFlow的基础上发布ImageNet计算机视觉模型，理论上使用谷歌的样本模型架构的话，就能很快地开始使用机器学习技术，因此，它可谓谷歌在计算机智能应用方面的杀手锏。不过，谷歌公司表示，开源该系统并不会危及公司的战略，而且还是会让公司保持优势。为什么呢？

通过谷歌最新公布的白皮书TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems可见，虽然该系统转为开源，但这次开源的TensorFlow是单机实现，其最有价值的分布式数据流实现，并没有开源。也就是说，谷歌此举还是留下了一些能保持自身独特性的东西，比如巨量的数据、可以运行该软件的计算机网络，以及一个庞大的可以调整算法的人工智能专家团队。

谷歌开源的做法，其实和Facebook开源人工智能研究的举措相类似。深度学习系统或软件不是一装上就能用的，它在发布前期需要通过使用者数据进行很多测试、调整，顾名思义就是一个学习的过程。Facebook开放人工智能研究就是因为缺乏自己的移动操作系统和通信渠道，没有什么途径来检测产品，唯有通过与其他公司组织合作，来完善创新。

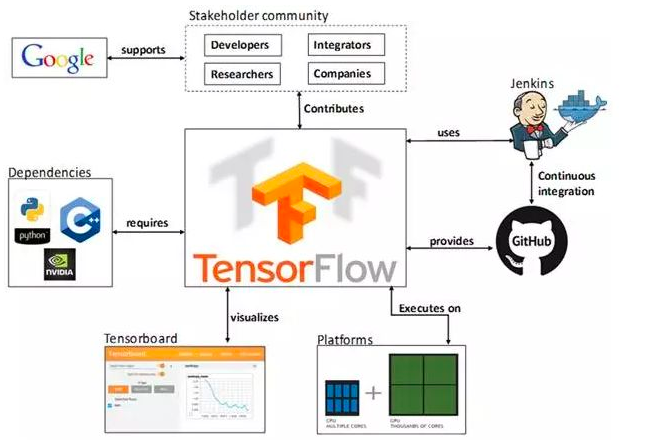
同样，如果谷歌没有工程师来为其进行数百万个参数的调整，那么谷歌这次放出的深度学习算法的用途就极为有限。专家表示，调整这些参数需要不断试错，初次接触的人要花费很长时间。谷歌也需要通过放出这套系统吸引更多研究人员，从而为其找到新的用武之地，并进行改进。

据报道，谷歌2011年展开了一个试图训练神经网络的Google Brain项目。Google Brain在当时拥有1.6万个多个微处理器，创建了一个有数十亿连接的神经网络，在这个项目研究中，该网络结果观看了千万数量级的YouTube图像，并从图像中自主辨识出了“猫”。研究人员表示：“在训练中，我们从未告诉它‘这是一只猫’。基本上可以说，它发明了猫这个概念。”

## TensorFlow™官方定义

TensorFlow™ 是一个开放源代码软件库，用于进行高性能数值计算。借助其灵活的架构，用户可以轻松地将计算工作部署到多种平台（CPU、GPU、TPU）和设备（桌面设备、服务器集群、移动设备、边缘设备等）。TensorFlow™ 最初是由 Google Brain 团队（隶属于 Google 的 AI 部门）中的研究人员和工程师开发的，可为机器学习和深度学习提供强力支持，并且其灵活的数值计算核心广泛应用于许多其他科学领域。

## TensorFlow™依赖视图



TF托管在github平台，有google groups和contributors共同维护。

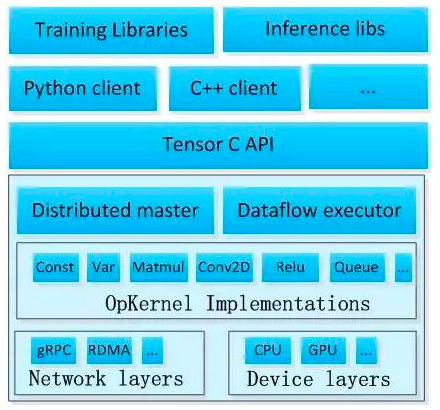
TF提供了丰富的深度学习相关的API，支持Python和C/C++接口。

TF提供了可视化分析工具Tensorboard，方便分析和调整模型。

TF支持Linux平台，Windows平台，Mac平台，甚至手机移动设备等各种平台。

## TensorFlow™系统架构

下图是TF的系统架构，从底向上分为设备管理和通信层、数据操作层、图计算层、API接口层、应用层。其中设备管理和通信层、数据操作层、图计算层是TF的核心层。



底层设备通信层负责网络通信和设备管理。设备管理可以实现TF设备异构的特性，支持CPU、GPU、Mobile等不同设备。网络通信依赖gRPC通信协议实现不同设备间的数据传输和更新。

第二层是Tensor的OpKernels实现。这些OpKernels以Tensor为处理对象，依赖网络通信和设备内存分配，实现了各种Tensor操作或计算。Opkernels不仅包含MatMul等计算操作，还包含Queue等非计算操作，这些将在第5章Kernels模块详细介绍。

第三层是图计算层（Graph），包含本地计算流图和分布式计算流图的实现。Graph模块包含Graph的创建、编译、优化和执行等部分，Graph中每个节点都是OpKernels类型表示。关于图计算将在第6章Graph模块详细介绍。

第四层是API接口层。Tensor C API是对TF功能模块的接口封装，便于其他语言平台调用。

第四层以上是应用层。不同编程语言在应用层通过API接口层调用TF核心功能实现相关实验和应用。

## TensorFlow™源码分析



我们以 TensorFlow 1.1.0 版本为例，看看 TensorFlow 的代码结构。 进入 tensorflow-1.1.0 目录，代码结构如下:

├── ACKNOWLEDGMENTS # TensorFlow 版权声明

├── ADOPTERS.md # 使用 TensorFlow 的人员或组织列表

├── AUTHORS # TensorFlow 作者的官方列表

├── BUILD

├── CONTRIBUTING.md # TensorFlow 贡献指导

├── ISSUE\_TEMPLATE.md # 提 ISSUE 的模板

├── LICENSE # 版权许可

├── README.md

├── RELEASE.md # 每次发版的 change log

├── WORKSPACE # 配置移动端开发环境

├── bower.BUILD

├── configure

├── models.BUILD

├── tensorflow # 主目录，后面分析的重点

├── third\_party # 第三方库，包括 eigen3(特征运算的库，包括 SVD、LU 分解等)、gpus(支持 cuda)、hadoop、jpeg、llvm、py、sycl

├── tools # 构建 cuda 支持

└── util

其中，最重要的源代码保存在 tensorflow 目录中。tensorflow 目录的结构如下:

├── BUILD

├── \_\_init\_\_.py

├── c

├── cc # 采用 C++进行训练的样例

├── compiler

├── contrib # 将常用功能封装在一起的高级 API

├── core # C++实现的主要目录

├── examples # 各种示例，本书后续讲的例子主要就在这个目录中

├── g3doc # 针对 C++、Python 版本的代码文档

├── go

├── java

├── opensource\_only # 声明目录

├── python # Python 实现的主要目录

├── stream\_executor # 流处理

├── tensorboard # App、Web 支持，以及脚本支持

├── tensorflow.bzl

├── tf\_exported\_symbols.lds

├── tf\_version\_script.lds

├── tools # 一些工具杂项

├── user\_ops

└── workspace.bzl

下面我们就简单介绍几个重点目录。

contirb

contrib 目录中保存的是将常用的功能封装成的高级 API。但是这个目录并不是官方支持的， 很有可能在高级 API 完善后被官方迁移到核心的 TensorFlow 目录中或去掉，现在有一部分包

(package)在 https://github.com/tensorflow/models 有了更完整的体现。这里重点介绍几个常用包。

● framework:很多函数(如get\_variables、get\_global\_step)都在这里定义，还有一些废弃的函数。

● layers:这个包主要有 initializers.py、layers.py、optimizers.py、regularizers.py、summaries.py 等文件。initializers.py 中主要是做变量初始化的函数。layers.py 中有关于层操作和权重 偏置变量的函数。optimizers.py 中包含损失函数和 global\_step 张量的优化器操作。 regularizers.py 中包含带有权重的正则化函数。summaries.py 中包含将摘要操作(见4.4.2 节可视化 API)添加到 tf.GraphKeys.SUMMARIES 集合中的函数。

● learn:这个包是使用 TensorFlow 进行深度学习的高级 API，包括完成训练模型和评估 模型、读取批处理数据和队列功能的 API 封装。

● rnn:这个包提供了额外的 RNN Cell，也就是对 RNN 隐藏层的各种改进，如 LSTMBlockCell、 GRUBlockCell、FusedRNNCell、GridLSTMCell、AttentionCellWrapper 等。

● seq2seq:这个包提供了建立神经网络 seq2seq 层和损失函数的操作。

● slim:TensorFlow-Slim (TF-Slim)是一个用于定义、训练和评估 TensorFlow 中的复杂 模型的轻量级库。在使用中可以将 TF-Slim 与 TensorFlow 的原生函数和 tf.contrib 中的其他包进行自由组合。TF-Slim 现在已经被逐渐迁移到 TensorFlow 开源的 Models1中， 这里包含了几种广泛使用的卷积神经网络图像分类模型的代码，可以从头训练模型或 者预训练模型开始微调。

Core

这个目录中保存的都是 C 语言的文件，是 TensorFlow 的原始实现。

├── BUILD

├── common\_runtime # 公共运行库

├── debug

├── distributed\_runtime # 分布式执行模块，含有 grpc session、grpc worker、 grpc master 等

├── example

├── framework # 基础功能模块

├── graph

├── kernels # 一些核心操作在 CPU、CUDA 内核上的实现

├── lib # 公共基础库

├── ops

├── platform # 操作系统实现相关文件

├── protobuf # .proto 文件，用于传输时的结构序列化

├── public # API 的头文件目录

├── user\_ops

└── util

Protocol Buffers 是谷歌公司创建的一个数据序列化(serialization)工具，可以用于结构化数据序列化，很适合作为数据存储或者 RPC 数据交换的格式。定义完协议缓冲区后，将生成.pb.h 和.pb.cc 文件，其中定义了相应的 get、set 以及序列化和反序列化函数。TensorFlow 的几个核心 proto 文件 graph\_def.proto、node\_def.proto、op\_def.proto 都保存在 framework 目录中。构图时先 构建 graph\_def，存储下来，然后在实际计算时再转成如图、节点、操作等的内存对象。

下面以 tensorflow-1.1.0/tensorflow/core/framework/node\_def.proto 为例来说明定义 proto 文件 的过程。node\_def.proto 定义中指定了设备(device)操作(op)以及操作的属性(attr)。代码 如下:

syntax = "proto3";

package tensorflow;

option cc\_enable\_arenas = true;

option java\_outer\_classname = "NodeProto";

option java\_multiple\_files = true;

option java\_package = "org.tensorflow.framework";

import "tensorflow/core/framework/attr\_value.proto";

message NodeDef {

string name = 1; # 操作的名称

string op = 2; # 操作的名称

repeated string input = 3; # 每个 input 指明了当前节点来自哪个节点的第几个张量，

# 格式是 node:index string device = 4; # 指定 device 方法

map<string, AttrValue> attr = 5; # 其中 AttrValue 又是另外一个协议缓冲区 };

framework 目录中还有 node\_def\_builder.h、node\_def\_builder.cc、node\_def\_util.h、node\_def\_ util\_test.cc 等文件，这都是为了在 C++里能操作上面代码中定义的 node\_def.proto 的 protobuf 结构。

## examples

examples 目录中给出了深度学习的一些例子，包括 MNIST、Word2vec、Deepdream、Iris、 HDF5 的一些例子，对入门非常有帮助。此外，这个目录中还有 TensorFlow 在 Android 系统上的移 动端实现，以及一些扩展为.ipynb 的文档教程，可以用 jupyter 打开

## g3doc

TensorFlow 的文档是用 Markdown 在维护的，并存放在 g3doc 中。g3doc 目录可以认为是 TensorFlow 的离线手册，非常好用。

g3doc/api\_docs 目录中的任何内容都是从代码中的注释生成的，不应该直接编辑。脚本 tools/docs/gen\_docs.sh是用来生成API文档的。如果无参数调用，它只重新生成Python API文 档(即操作的文档，包括用 Python 和 C ++定义的)。如果传递了-a，运行脚本时还会重新生成 C++ API 的文档。这个脚本必须从 tools/docs 目录调用，如果使用参数-a，需要安装 doxygen。

## TensorFlow的未来

2004年，Google创始人Larry Page和Sergey Brin曾预言道：“计算机将会被植入人类大脑，搜索会成为一种自主进行的行为。”14年之后的今天，Google没有将搜索功能植入大脑，但该公司将搜索机器变成一个人工智能大脑。“这是一种能够让研究人员的创意直接转化成产品的工具，以后研究人员或者就无须再为什么新想法重新编写代码了。”

这是谷歌未来的愿景，而TensorFlow就是通向这个未来的里程碑之一。开源TensorFlow能够加速谷歌在人工智能技术的部署，帮助该公司在人工智能发展日益重要的未来抢占更多的主导权。

不过当然，开源就存在着多方合作的可能，数据的资源共享、技术研究的交流碰撞……谷歌的发展同样也会带动业界的技术整体发展。多年以前人们就知道开源的重要性，正如开源硬件公司Arduino CEO Musto所言：“我们需要开源，我们需要社群的智慧。”

参考文献：

<https://blog.csdn.net/qq_25737169/article/details/79192211>

<https://github.com/hzy46/fast-neural-style-tensorflow>

<https://zhuanlan.zhihu.com/p/24383274>

<https://zhuanlan.zhihu.com/p/27697553>

《TensorFlow技术解析与实战》

<https://zhuanlan.zhihu.com/p/25646408>

<https://zhuanlan.zhihu.com/p/25927956>