

醍醐灌顶的酸爽

TENSORFLOW 实践

徘徊之秋叶

QQ: 809758521 | 南京.天印大道
2018 年 6 月 30 日

目录

前言	3
Python 2/3 Change.....	5
深度学习框架的对比.....	6
应用领域.....	20
部署	22
pip 安装	22
容器部署.....	23
基于 VirtualEnv 的安装	33
基于 Anaconda 的安装	33
INTEL DIGITS.....	36
CUDA	37
CUDA 虚拟化	37
基本使用.....	37
Command Line	37
Spyder	41
Jupyter notebook	42
通过 Anaconda 安装 Jupyter notebook	43
DOIT Simply	51
图形对象检测.....	51
街景识别.....	51
汽车类型识别.....	67
实时视频人脸识别.....	69
模型训练.....	69
数据基础.....	69
数据挖掘.....	69
C4.5	69
K-Means	69
SVM	76
Apriori	76
EM	76
PageRank.....	76
AdaBoost.....	76
kNN.....	76
Naïve Bayes	76
CART	76
中级应用.....	76
JAVA 开发.....	76
扩展.....	76
深度学习画图神器(TikZ).....	76
监督学习.....	77
非监督学习.....	77

数据处理与模型调优.....	77
NLP	77
语音识别.....	77
图像识别.....	77
参考	78

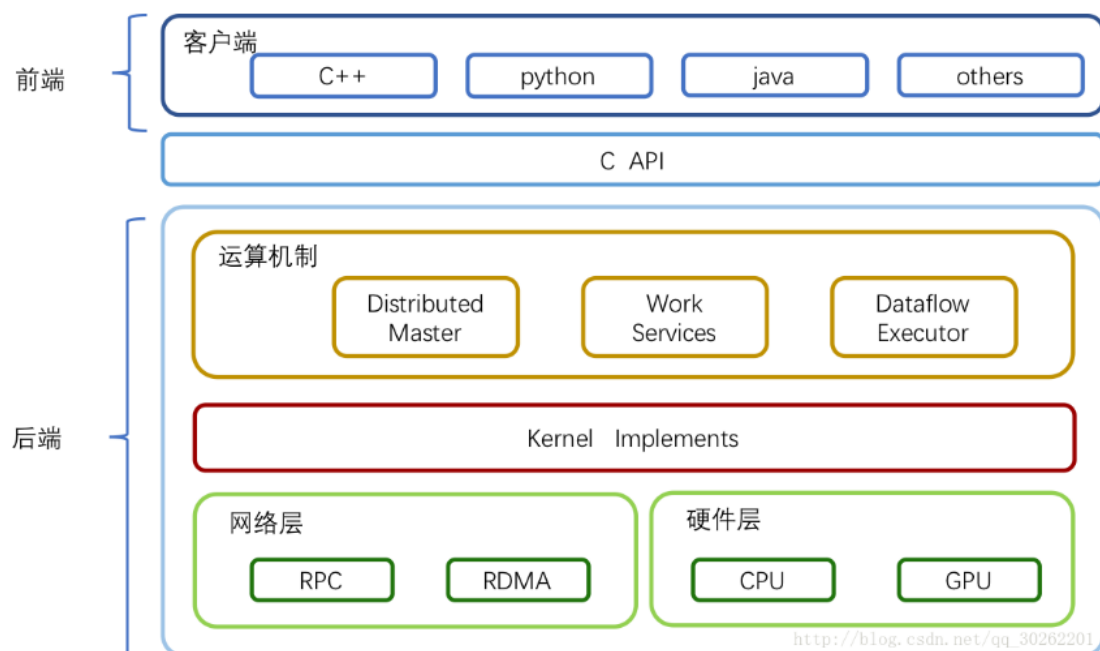
前言

层	功能	组件
视图层	计算图可视化	TensorBoard
工作流层	数据集准备、存储、加载	Keras/TF Slim
计算图层	计算图构造与优化 前向计算/后向计算	TensorFlow Core
高维计算层	高维数组处理	Eigen
数值计算层	矩阵计算/卷积计算	BLAS/cuBLAS/ cuRAND/cuDNN
网络层	通信	gRPC/RDMA
设备层	硬件	CPU/GPU

整个系统从底层到上层可分为七层：

- 最底层是硬件计算资源，支持 CPU、GPU；
- 支持两种通信协议；
- 数值计算层提供最基础的计算，有线性计算、卷积计算；
- 数据的计算都是以数组的形式参与计算；
- 计算图层用来设计神经网络的结构；
- 工作流层提供轻量级的框架调用
- 最后构造的深度学习网络可以通过 TensorBoard 服务端可视化

技术架构：

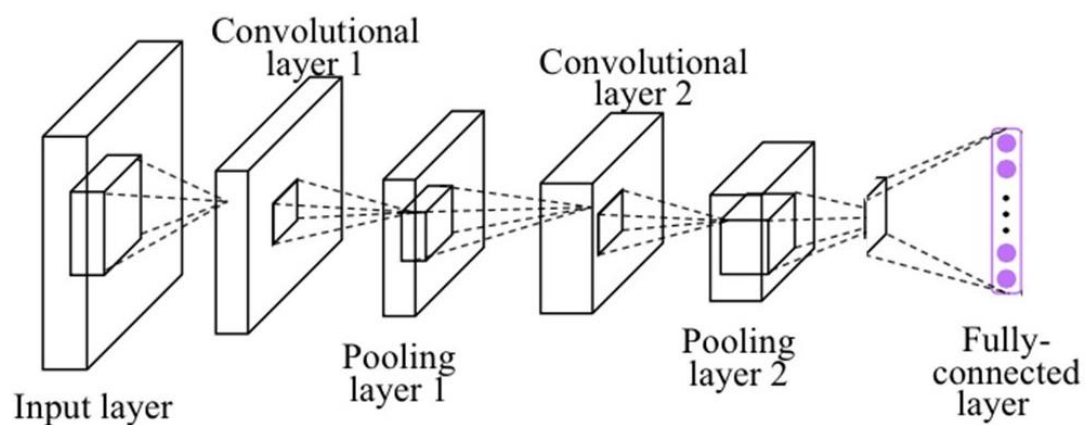
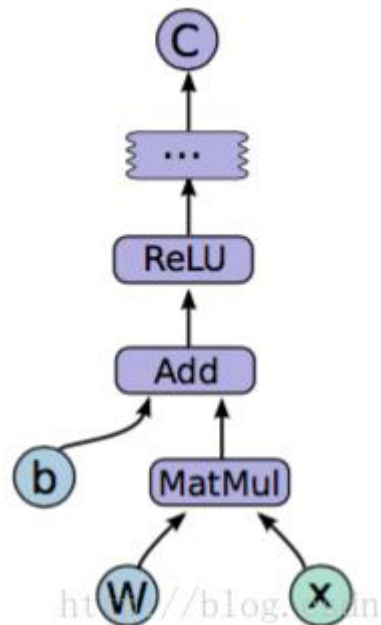


TensorFlow 是用数据流图(data flow graphs)技术来进行数值计算的。

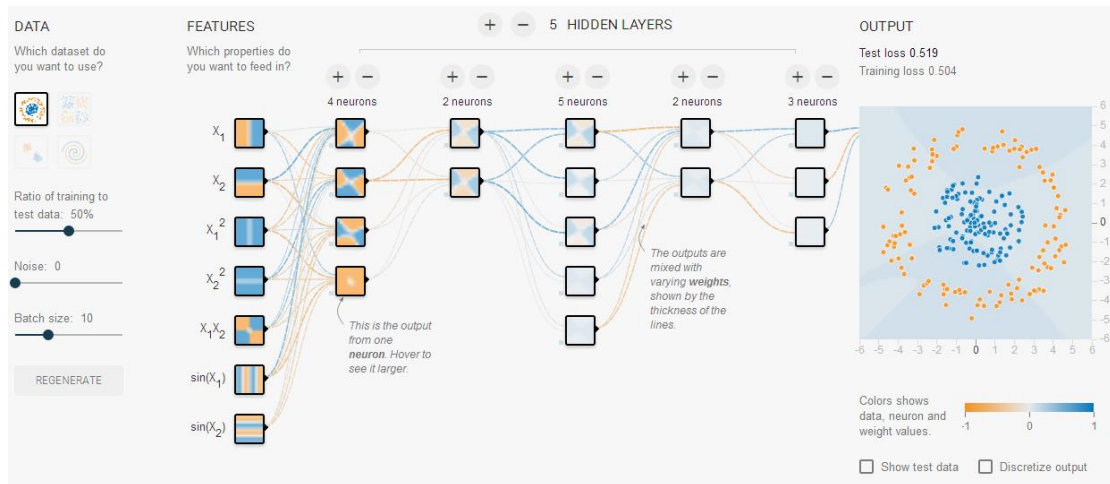
数据流图是描述有向图中的数值计算过程。

有向图中，节点通常代表数学运算，边表示节点之间的某种联系，它负责传输多维数据(Tensors)。

节点可以被分配到多个计算设备上，可以异步和并行地执行操作。因为是有向图，所以只有等到之前的入度节点们的计算状态完成后，当前节点才能执行操作。



<http://playground.tensorflow.org/> （动态模拟神经网络）



Tensorflow 分为 CPU、GPU 版本。官方推荐使用 Ubuntu

Tensorflow 暴露出来的两个弱点:

- 1、性能评测中比 caffe 等慢;
- 2、中间状态的值不可传递给外部。

Python 2/3 Change

1. `tf.mul` `tf.sub` `tf.neg` 已经废弃
分别可用 `tf.multiply` `tf.subtract` `tf.negative` 替代。

2. `xrange`

报错:

`NameError: name 'xrange' is not defined`

解决:

在 Python 3 中, `range()` 与 `xrange()` 合并为 `range()`。

3. `print` 函数

(Python3 中 `print` 为一个函数, 必须用括号括起来; Python2 中 `print` 为 class)

4. `Input`

通过 `input()` 解析用户的输入: (Python3 中 `input` 得到的为 `str`; Python2 的 `input` 的到的为 `int` 型, Python2 的 `raw_input` 得到的为 `str` 类型) 统一一下: Python3 中用 `input`, Python2 中用 `raw_input`, 都输入为 `str`

5. 整除

(没有太大影响) (Python3 中 `/` 表示真除, `%` 表示取余, `//` 表示地板除 (结果取整); Python2 中 `/` 表示根据除数被除数小数点位得到结果, `//` 同样表示地板除) 统一一下: Python3 中 `/` 表示真除, `%` 表示取余, `//` 结果取整; Python2 中带上小数点 `/` 表示真除, `%` 表示取余, `//` 结果取整

6. 字符串存储

原: 字符串以 8-bit 字符串存储

改为: 字符串以 16-bit Unicode 字符串存储

7. try except 语句的变化

原: try:

```
.....
except Exception, e :
.....
```

改为

```
try:
.....
except Exception as e :
.....
```

8. 打开文件

原: file(.....)

或 open(.....)

改为:

只能用 open(.....)

9. 从键盘录入一个字符串

原: raw_input("提示信息")

改为: input("提示信息")

10. bytes 数据类型

A bytes object is an immutable array. The items are 8-bit bytes, represented by integers in the range $0 \leq x < 256$.

bytes 可以看成是“字节数组”对象，每个元素是 8-bit 的字节，取值范围 0~255。

由于在 python 3.0 中字符串以 unicode 编码存储，当写入二进制文件时，字符串无法直接写入（或读取），必须以某种方式的编码为字节序列后，方可写入。

11. Python3 中可以用中文名做变量名称.

深度学习框架的对比

<https://blog.csdn.net/xiangxizhishi/article/details/76039214>

Google 近日发布了 TensorFlow 1.0 候选版，这第一个稳定版将是深度学习框架发展中的里程碑的一步。自 TensorFlow 于 2015 年底正式开源，距今已有一年多，这期间

TensorFlow 不断给人以惊喜。在这一年多时间，TensorFlow 已从初入深度学习框架大战的新星，成为了几近垄断的行业事实标准。

主流深度学习框架对比

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、DeepLearning4J、Lasagne、Neon 等等。然而 TensorFlow 却杀出重围，在关注度和用户数上都占据绝对优势，大有一统江湖之势。表 2-1 所示为各个开源框架在 GitHub 上的数据统计（数据统计于 2017 年 1 月 3 日），可以看到 TensorFlow 在 star 数量、fork 数量、contributor 数量这三个数据上都完胜其他对手。

究其原因，主要是 Google 在业界的号召力确实强大，之前也有许多成功的开源项目，以及 Google 强大的人工智能研发水平，都让大家对 Google 的深度学习框架充满信心，以至于 TensorFlow 在 2015 年 11 月刚开源的第一个月就积累了 10000+ 的 star。其次，TensorFlow 确实在很多方面拥有优异的表现，比如设计神经网络结构的代码的简洁度，分布式深度学习算法的执行效率，还有部署的便利性，都是其得以胜出的亮点。如果一直关注着 TensorFlow 的开发进度，就会发现基本上每星期 TensorFlow 都会有 1 万行以上的代码更新，多则数万行。产品本身优异的质量、快速的迭代更新、活跃的社区和积极的反馈，形成了良性循环，可以想见 TensorFlow 未来将继续在各种深度学习框架中独占鳌头。

框 架	机 构	支持语言	Stars	Forks	Contributors
TensorFlow	Google	Python/C++/Go/...	41628	19339	568
Caffe	BVLC	C++/Python	14956	9282	221
Keras	fchollet	Python	10727	3575	322
CNTK	Microsoft	C++	9063	2144	100
MXNet	DMLC	Python/C++/R/...	7393	2745	241
Torch7	Facebook	Lua	6111	1784	113
Theano	U. Montreal	Python	5352	1868	271
Deeplearning4J	DeepLearning4J	Java/Scala	5053	1927	101
Leaf	AutumnAI	Rust	4562	216	14
Lasagne	Lasagne	Python	2749	761	55
Neon	NervanaSystems	Python	2633	573	52

表 2-1 各个开源框架在 GitHub 上的数据统计

观察表 2-1 还可以发现，Google、Microsoft、Facebook 等巨头都参与了这场深度学习框架大战，此外，还有毕业于伯克利大学的贾扬清主导开发的 Caffe，蒙特利尔大学 Lisa Lab 团队开发的 Theano，以及其他个人或商业组织贡献的框架。另外，可以看到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。虽然有来自 R、Julia 等语言的竞争压力，但是 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。仅在数据挖掘工具链上，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。

表 2-1 和图 2-1 所示为对主流的深度学习框架 TensorFlow、Caffe、CNTK、Theano、Torch 在各个维度的评分，本书 2.2 节会对各个深度学习框架进行比较详细的介绍。

	模型设计	接 口	部 署	性 能	架构设计	总体评分
TensorFlow	80	80	90	90	100	88
Caffe	60	60	90	80	70	72

	模型设计	接 口	部 署	性 能	架构设计	总体评分
CNTK	50	50	70	100	60	66
Theano	80	70	40	50	50	58
Torch	90	70	60	70	90	76
MXNet	70	100	80	80	90	84
DeepLearning4J	60	70	80	80	70	72

续表

表 2-2 主流深度学习框架在各个维度的评分

图 2-1 主流深度学习框架对比图

各深度学习框架简介

在本节，我们先来看看目前各流行框架的异同，以及各自的特点和优势。

TensorFlow

TensorFlow 是相对高阶的机器学习库，用户可以方便地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 则会比较消耗资源，并且执行效率不高）。除了核心代码的 C++接口，TensorFlow 还有官方的 Python、Go 和 Java 接口，是通过 SWIG（Simplified Wrapper and Interface Generator）实现的，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。SWIG 支持给 C/C++ 代码提供各种语言的接口，因此其他脚本语言的接口未来也可以通过 SWIG 方便地添加。不过使用 Python 时有一个影响效率的问题是，每一个 mini-batch 要从 Python 中 feed 到网络中，这个过程在 mini-batch 的数据量很小或者运算时间很短时，可能会带来影响比较大的延迟。现在 TensorFlow 还有非官方的 Julia、Node.js、R 的接口支持，地址如下。

Julia: <http://github.com/malmaud/TensorFlow.jl>

Node.js: <http://github.com/node-tensorflow/node-tensorflow>

R: <http://github.com/rstudio/tensorflow>

TensorFlow 也有内置的 TF.Learn 和 TF.Slim 等上层组件可以帮助快速地设计新网络，并且兼容 Scikit-learn estimator 接口，可以方便地实现 evaluate、grid search、cross validation 等功能。同时 TensorFlow 不只局限于神经网络，其数据流式图支持非常自由的算法表达，当然也可以轻松实现深度学习以外的机器学习算法。事实上，只要可以将计算表示成计算图的形式，就可以使用 TensorFlow。用户可以写内层循环代码控制计算图分支的计算，TensorFlow 会自动将相关的分支转为子图并执行迭代运算。TensorFlow 也可以将计算图中的各个节点分配到不同的设备执行，充分利用硬件资源。定义新的节点只需要写一个 Python 函数，如果没有对应的底层运算核，那么可能需要写 C++或者 CUDA 代码实现运算操作。

在数据并行模式上，TensorFlow 和 Parameter Server 很像，但 TensorFlow 有独立的 Variable node，不像其他框架有一个全局统一的参数服务器，因此参数同步更自由。TensorFlow 和 Spark 的核心都是一个数据计算的流式图，Spark 面向的是大规模的数据，支持 SQL 等操作，而 TensorFlow 主要面向内存足以装载模型参数的环境，这样可以最大化计算效率。

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。相比于 Theano，TensorFlow 还有一个优势就是它极快的编译速度，在定义新网络结构时，Theano 通常需要长时间的编译，因此尝试新模型需要比较大的代价，而 TensorFlow 完全没有这个问题。TensorFlow 还有功能强大的可视化组件 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。TensorFlow 针对生产环境高度优化，它产品级的高质量代码和设计都可以保证在生产环境中稳定运行，同时一旦 TensorFlow 广泛地被工业界使用，将产生良性循环，成为深度学习领域的事实标准。

除了支持常见的网络结构（卷积神经网络（Convolutional Neural Network，CNN）、循环神经网络（Recurrent Neural Network，RNN））外，TensorFlow 还支持深度强化学习乃至其他计算密集的科学计算(如偏微分方程求解等)。TensorFlow 此前不支持 symbolic loop，需要使用 Python 循环而无法进行图编译优化，但最近新加入的 XLA 已经开始支持 JIT 和 AOT，另外它使用 bucketing trick 也可以比较高效地实现循环神经网络。TensorFlow 的一个薄弱地方可能在于计算图必须构建为静态图，这让很多计算变得难以实现，尤其是序列预测中经常使用的 beam search。

TensorFlow 的用户能够将训练好的模型方便地部署到多种硬件、操作系统平台上，支持 Intel 和 AMD 的 CPU，通过 CUDA 支持 NVIDIA 的 GPU（最近也开始通过 OpenCL 支持 AMD 的 GPU，但没有 CUDA 成熟），支持 Linux 和 Mac，最近在 0.12 版本中也开始尝试支持 Windows。在工业生产环境中，硬件设备有些是最新款的，有些是用了几年的老机型，来源可能比较复杂，TensorFlow 的异构性让它能够全面地支持各种硬件和操作系统。同时，其在 CPU 上的矩阵运算库使用了 Eigen 而不是 BLAS 库，能够基于 ARM 架构编译和优化，因此在移动设备（Android 和 iOS）上表现得很好。

TensorFlow 在最开始发布时只支持单机，而且只支持 CUDA 6.5 和 cuDNN v2，并且没有官方和其他深度学习框架的对比结果。在 2015 年年底，许多其他框架做了各种性能对比评测，每次 TensorFlow 都会作为较差的对照组出现。那个时期的 TensorFlow 真的不快，性能上仅和普遍认为很慢的 Theano 比肩，在各个框架中可以算是垫底。但是凭借 Google 强大的开发实力，很快支持了新版的 cuDNN（目前支持 cuDNN v5.1），在单 GPU 上的性能追上了其他框架。表 2-3 所示为 <https://github.com/soumith/convnet-benchmarks> 给出的各个框架在 AlexNet 上单 GPU 的性能评测。

表 2-3 各深度学习框架在 AlexNet 上的性能对比

目前在单 GPU 的条件下，绝大多数深度学习框架都依赖于 cuDNN，因此只要硬件计算能力或者内存分配差异不大，最终训练速度不会相差太大。但是对于大规模深度学习来说，巨大的数据量使得单机很难在有限的时间完成训练。这时需要分布式计算使 GPU 集群乃至 TPU 集群并行计算，共同训练出一个模型，所以框架的分布式性能是至关重要的。TensorFlow 在 2016 年 4 月开源了分布式版本，使用 16 块 GPU 可达单 GPU 的 15 倍提速，在 50 块 GPU 时可达到 40 倍提速，分布式的效率很高。目前原生支持的分布式深度学习框架不多，只有 TensorFlow、CNTK、DeepLearning4J、MXNet 等。不过目前 TensorFlow 的设计对不同设备间的通信优化得不是很好，其单机的 reduction 只能用 CPU 处理，分布式的通信使用基于 socket 的 RPC，而不是速度更快的 RDMA，所以其分布式性能可能还没有达到最优。

Google 在 2016 年 2 月开源了 TensorFlow Serving，这个组件可以将 TensorFlow 训练好的模型导出，并部署成可以对外提供预测服务的 RESTful 接口，如图 2-2 所示。有了这个组件，TensorFlow 就可以实现应用机器学习的全流程：从训练模型、调试参数，到打包模型，最后部署服务，名副其实是一个从研究到生产整条流水线都齐备的框架。这里引用 TensorFlow 内部开发人员的描述：“TensorFlow Serving 是一个为生产环境而设计的高性能的机器学习服务系统。它可以同时运行多个大规模深度学习模型，支持模型生命周期管理、算法实验，并可以高效地利用 GPU 资源，让 TensorFlow 训练好的模型更快捷方便地投入到实际生产环境”。除了 TensorFlow 以外的其他框架都缺少为生产环境部署的考虑，而 Google 作为广泛在实际产品中应用深度学习的巨头可能也意识到了这个机会，因此开发了这个部署服务的平台。TensorFlow Serving 可以说是一副王牌，将会帮 TensorFlow 成为行业标准做出巨大贡献。

图 2-2 TensorFlow Serving 架构

TensorBoard 是 TensorFlow 的一组 Web 应用，用来监控 TensorFlow 运行过程，或可视化 Computation Graph。TensorBoard 目前支持五种可视化：标量（scalars）、图片（images）、音频（audio）、直方图（histograms）和计算图（Computation Graph）。TensorBoard 的 Events 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

是验证集上的准确率 (accuracy); Image Dashboard 则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, 并且支持缩放拖曳和查看节点属性。TensorBoard 的可视化效果如图 2-3 和图 2-4 所示。

图 2-3 TensorBoard 的 loss 标量的可视化

图 2-4 TensorBoard 的模型结构可视化

TensorFlow 拥有产品级的高质量代码, 有 Google 强大的开发、维护能力的加持, 整体架构设计也非常优秀。相比于同样基于 Python 的老牌对手 Theano, TensorFlow 更成熟、更完善, 同时 Theano 的很多主要开发者都去了 Google 开发 TensorFlow (例如书籍 Deep Learning 的作者 Ian Goodfellow, 他后来去了 OpenAI)。Google 作为巨头公司有比高校或者个人开发者多得多的资源投入到 TensorFlow 的研发, 可以预见, TensorFlow 未来的发展将会是飞速的, 可能会把大学或者个人维护的深度学习框架远远甩在身后。

Caffe

官方网站: <http://caffe.berkeleyvision.org/>

GitHub: <http://github.com/BVLC/caffe>

Caffe 全称为 Convolutional Architecture for Fast Feature Embedding, 是一个被广泛使用的开源深度学习框架 (在 TensorFlow 出现之前一直是深度学习领域 GitHub star 最多的项目), 目前由伯克利视觉学中心 (Berkeley Vision and Learning Center, BVLC) 进行维护。Caffe 的创始人是加州大学伯克利的 Ph.D. 贾扬清, 他同时也是 TensorFlow 的作者之一, 曾工作于 MSRA、NEC 和 Google Brain, 目前就职于 Facebook FAIR 实验室。Caffe 的主要优势包括如下几点。

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。

训练速度快, 能够训练 state-of-the-art 的模型与大规模的数据。

组件模块化, 可以方便地拓展到新的模型和学习任务上。

Caffe 的核心概念是 Layer, 每一个神经网络的模块都是一个 Layer。Layer 接收输入数据, 同时经过内部计算产生输出数据。设计网络结构时, 只需要把各个 Layer 拼接在一起构成完整的网络 (通过写 protobuf 配置文件定义)。比如卷积的 Layer, 它的输入就是图片的全部像素点, 内部进行的操作是各种像素值与 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) 的运

算，即从输入数据计算输出结果，也就是模型的预测过程；另一种是反向（backward）的运算，从输出端的 gradient 求解相对于输入的 gradient，即反向传播算法，这部分也就是模型的训练过程。实现新 Layer 时，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（当需要运行在 GPU 时）代码，对普通用户来说还是非常难上手的。正如它的名字 Convolutional Architecture for Fast Feature Embedding 所描述的，Caffe 最开始设计时的目标只针对于图像，没有考虑文本、语音或者时间序列的数据，因此 Caffe 对卷积神经网络的支持非常好，但对时间序列 RNN、LSTM 等支持得不是特别充分。同时，基于 Layer 的模式也对 RNN 不是非常友好，定义 RNN 结构时比较麻烦。在模型结构非常复杂时，可能需要写非常冗长的配置文件才能设计好网络，而且阅读时也比较费力。

Caffe 的一大优势是拥有大量的训练好的经典模型（AlexNet、VGG、Inception）乃至其他 state-of-the-art（ResNet 等）的模型，收藏在它的 Model Zoo（<http://github.com/BVLC/caffe/wiki/Model-Zoo>）。因为知名度较高，Caffe 被广泛地应用于前沿的工业界和学术界，许多提供源码的深度学习的论文都是使用 Caffe 来实现其模型的。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。虽然 Caffe 主要是面向学术圈和研究者的，但它的程序运行非常稳定，代码质量比较高，所以也很适合对稳定性要求严格的生产环境，可以算是第一个主流的工业级深度学习框架。因为 Caffe 的底层是基于 C++ 的，因此可以在各种硬件环境编译并具有良好的移植性，支持 Linux、Mac 和 Windows 系统，也可以编译部署到移动设备系统如 Android 和 iOS 上。和其他主流深度学习库类似，Caffe 也提供了 Python 语言接口 pycaffe，在接触新任务，设计新网络时可以使用其 Python 接口简化操作。不过，通常用户还是使用 Protobuf 配置文件定义神经网络结构，再使用 command line 进行训练或者预测。Caffe 的配置文件是一个 JSON 类型的 .prototxt 文件，其中使用许多顺序连接的 Layer 来描述神经网络结构。Caffe 的二进制可执行程序会提取这些 .prototxt 文件并按其定义来训练神经网络。理论上，Caffe 的用户可以完全不写代码，只是定义网络结构就可以完成模型训练了。Caffe 完成训练之后，用户可以把模型文件打包制作成简单易用的接口，比如可以封装成 Python 或 MATLAB 的 API。不过在 .prototxt 文件内部设计网络结构可能会比较受限，没有像 TensorFlow 或者 Keras 那样在 Python 中设计网络结构方便、自由。更重要的是，Caffe 的配置文件不能用编程的方式调整超参数，也没有提供像 Scikit-learn 那样好用的 estimator 可以方便地进行交叉验证、超参数的 Grid Search 等操作。Caffe 在 GPU 上训练的性能很好（使用单块 GTX 1080 训练 AlexNet 时一天可以训练上百万张图片），但是目前仅支持单机多 GPU 的训练，没有原生支持分布式的训练。庆幸的是，现在有很多第三方的支持，比如雅虎开源的 CaffeOnSpark，可以借助 Spark 的分布式框架实现 Caffe 的大规模分布式训练。

Theano

官方网址：<http://www.deeplearning.net/software/theano/>

GitHub: <http://github.com/Theano/Theano>

Theano 诞生于 2008 年，由蒙特利尔大学 Lisa Lab 团队开发并维护，是一个高性能的符号计算及深度学习库。因其出现时间早，可以算是这类库的始祖之一，也一度被认为是深度学习研究和应用的重要标准之一。Theano 的核心是一个数学表达式的编译器，专门为处理大规模神经网络训练的计算而设计。它可以将用户定义的各种计算编译为高效的底层代

码，并链接各种可以加速的库，比如 BLAS、CUDA 等。Theano 允许用户定义、优化和评估包含多维数组的数学表达式，它支持将计算装载到 GPU（Theano 在 GPU 上性能不错，但是 CPU 上较差）。与 Scikit-learn 一样，Theano 也很好地整合了 NumPy，对 GPU 的透明让 Theano 可以较为方便地进行神经网络设计，而不必直接写 CUDA 代码。Theano 的主要优势如下。

集成 NumPy，可以直接使用 NumPy 的 ndarray，API 接口学习成本低。

计算稳定性好，比如可以精准地计算输出值很小的函数（像 $\log(1+x)$ ）。

动态地生成 C 或者 CUDA 代码，用以编译成高效的机器代码。

因为 Theano 非常流行，有许多人为它编写了高质量的文档和教程，用户可以方便地查找 Theano 的各种 FAQ，比如如何保存模型、如何运行模型等。不过 Theano 更多地被当作一个研究工具，而不是当作产品来使用。虽然 Theano 支持 Linux、Mac 和 Windows，但是没有底层 C++ 的接口，因此模型的部署非常不方便，依赖于各种 Python 库，并且不支持各种移动设备，所以几乎没有在工业生产环境的应用。Theano 在调试时输出的错误信息非常难以看懂，因此 DEBUG 时非常痛苦。同时，Theano 在生产环境使用训练好的模型进行预测时性能比较差，因为预测通常使用服务器 CPU（生产环境服务器一般没有 GPU，而且 GPU 预测单条样本延迟高反而不如 CPU），但是 Theano 在 CPU 上的执行性能比较差。

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，编译复杂模型的时间非常久。此外，Theano 在导入时也比较慢，而且一旦设定了选择某块 GPU，就无法切换到其他设备。目前，Theano 在 CUDA 和 cuDNN 上不支持多 GPU，只在 OpenCL 和 Theano 自己的 gpuarray 库上支持多 GPU 训练，速度暂时还比不上 CUDA 的版本，并且 Theano 目前还没有分布式的实现。不过，Theano 在训练简单网络（比如很浅的 MLP）时性能可能比 TensorFlow 好，因为全部代码都是运行时编译，不需要像 TensorFlow 那样每次 feed mini-batch 数据时都得通过低效的 Python 循环来实现。

Theano 是一个完全基于 Python（C++/CUDA 代码也是打包为 Python 字符串）的符号计算库。用户定义的各种运算，Theano 可以自动求导，省去了完全手工写神经网络反向传播算法的麻烦，也不需要像 Caffe 一样为 Layer 写 C++ 或 CUDA 代码。Theano 对卷积神经网络的支持很好，同时它的符号计算 API 支持循环控制（内部名 scan），让 RNN 的实现非常简单并且高性能，其全面的功能也让 Theano 可以支持大部分 state-of-the-art 的网络。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，其中有大名鼎鼎的 Keras，Keras 对神经网络抽象得非常合适，以至于可以随意切换执行计算的后端（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、激活函数等）连接起来，但是设计新模块或者新的 Layer 就不太方便了。除 Keras 外，还有学术界非常喜爱的 Lasagne，同样也是 Theano 的上层封装，它对神经网络的每一层的定义都非常严谨。另外，还有 scikit-neuralnetwork、nolearn 这两个基于 Lasagne 的上层封装，它们将神经网络抽象为兼容 Scikit-learn 接口的 classifier 和 regressor，这样就可以方便地使用 Scikit-learn 中经典的 fit、transform、score 等操作。除此之外，Theano 的上层封装库还有 blocks、deepy、pylearn2 和

Scikit-theano，可谓是一个庞大的家族。如果没有 Theano，可能根本不会出现这么多好用的 Python 深度学习库。同样，如果没有 Python 科学计算的基石 NumPy，就不会有 SciPy、Scikit-learn 和 Scikit-image，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。虽然 Theano 非常重要，但是直接使用 Theano 设计大型的神经网络还是太烦琐了，用 Theano 实现 Google Inception 就像用 NumPy 实现一个支持向量机(SVM)。且不说很多用户做不到用 Theano 实现一个 Inception 网络，即使能做到但是否有必要花这个时间呢？毕竟不是所有人都是基础科学工作者，大部分使用场景还是在工业应用中。所以简单易用是一个很重要的特性，这也就是其他上层封装库的价值所在：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 粒度设计网络。

Torch

官方网址：<http://torch.ch/>

GitHub：<http://github.com/torch/torch7>

Torch 给自己的定位是 LuaJIT 上的一个高效的科学计算库，支持大量的机器学习算法，同时以 GPU 上的计算优先。Torch 的历史非常悠久，但真正得到发扬光大是在 Facebook 开源了其深度学习的组件之后，此后包括 Google、Twitter、NYU、IDIAP、Purdue 等组织都大量使用 Torch。Torch 的目标是让设计科学计算算法变得便捷，它包含了大量的机器学习、计算机视觉、信号处理、并行运算、图像、视频、音频、网络处理的库，同时和 Caffe 类似，Torch 拥有大量的训练好的深度学习模型。它可以支持设计非常复杂的神经网络的拓扑图结构，再并行化到 CPU 和 GPU 上，在 Torch 上设计新的 Layer 是相对简单的。它和 TensorFlow 一样使用了底层 C++加上层脚本语言调用的方式，只不过 Torch 使用的是 Lua。Lua 的性能是非常优秀的（该语言经常被用来开发游戏），常见的代码可以通过透明的 JIT 优化达到 C 的性能的 80%；在便利性上，Lua 的语法也非常简单易读，拥有漂亮和统一的结构，易于掌握，比写 C/C++简洁很多；同时，Lua 拥有一个非常直接的调用 C 程序的接口，可以简便地使用大量基于 C 的库，因为底层核心是 C 写的，因此也可以方便地移植到各种环境。Lua 支持 Linux、Mac，还支持各种嵌入式系统（iOS、Android、FPGA 等），只不过运行时还是必须有 LuaJIT 的环境，所以工业生产环境的使用相对较少，没有 Caffe 和 TensorFlow 那么多。

为什么不简单地使用 Python 而是使用 LuaJIT 呢？官方给出了以下几点理由。

LuaJIT 的通用计算性能远胜于 Python，而且可以直接在 LuaJIT 中操作 C 的 pointers。

Torch 的框架，包含 Lua 是自洽的，而完全基于 Python 的程序对不同平台、系统移植性较差，依赖的外部库较多。

LuaJIT 的 FFI 拓展接口非常易学，可以方便地链接其他库到 Torch 中。Torch 中还专门设计了 N-Dimension array type 的对象 Tensor，Torch 中的 Tensor 是一块内存的视图，同时一块内存可能有许多视图（Tensor）指向它，这样的设计同时兼顾了性能（直接面向内存）和便利性。同时，Torch 还提供了不少相关的库，包括线性代数、卷积、傅里叶变换、绘图和统计等，如图 2-5 所示。

图 2-5 Torch 提供的各种数据处理的库

Torch 的 nn 库支持神经网络、自编码器、线性回归、卷积网络、循环神经网络等，同时支持定制的损失函数及梯度计算。Torch 因为使用了 LuaJIT，因此用户在 Lua 中做数据预处理等操作可以随意使用循环等操作，而不必像在 Python 中那样担心性能问题，也不需要学习 Python 中各种加速运算的库。不过，Lua 相比 Python 还不是那么主流，对大多数用户有学习成本。Torch 在 CPU 上的计算会使用 OpenMP、SSE 进行优化，GPU 上使用 CUDA、cutorch、cunn、cuDNN 进行优化，同时还有 cuda-convnet 的 wrapper。Torch 有很多第三方的扩展可以支持 RNN，使得 Torch 基本支持所有主流的网络。和 Caffe 类似的是，Torch 也是主要基于 Layer 的连接来定义网络的。Torch 中新的 Layer 依然需要用户自己实现，不过定义新 Layer 和定义网络的方式很相似，非常简便，不像 Caffe 那么麻烦，用户需要使用 C++ 或者 CUDA 定义新 Layer。同时，Torch 属于命令式编程模式，不像 Theano、TensorFlow 属于声明性编程（计算图是预定义的静态的结构），所以用它实现某些复杂操作（比如 beam search）比 Theano 和 TensorFlow 方便很多。

Lasagne

官网网址：<http://lasagne.readthedocs.io/>

GitHub：<http://github.com/Lasagne/Lasagne>

Lasagne 是一个基于 Theano 的轻量级的神经网络库。它支持前馈神经网络，比如卷积网络、循环神经网络、LSTM 等，以及它们的组合；支持许多优化方法，比如 Nesterov momentum、RMSprop、ADAM 等；它是 Theano 的上层封装，但又不像 Keras 那样进行了重度的封装，Keras 隐藏了 Theano 中所有的方法和对象，而 Lasagne 则是借用了 Theano 中很多的类，算是介于基础的 Theano 和高度抽象的 Keras 之间的一个轻度封装，简化了操作同时支持比较底层的操作。Lasagne 设计的六个原则是简洁、透明、模块化、实用、聚焦和专注。

Keras

官方网址：<http://keras.io>

GitHub：<http://github.com/fchollet/keras>

Keras 是一个崇尚极简、高度模块化的神经网络库，使用 Python 实现，并可以同时运行在 TensorFlow 和 Theano 上。它旨在让用户进行最快速的原型实验，让想法变为结果的这个过程最短。Theano 和 TensorFlow 的计算图支持更通用的计算，而 Keras 则专精于深度学习。Theano 和 TensorFlow 更像是深度学习领域的 NumPy，而 Keras 则是这个领域的 Scikit-learn。它提供了目前为止最方便的 API，用户只需要将高级的模块拼在一起，就可以设计神经网络，它大大降低了编程开销（code overhead）和阅读别人代码时的理解开销。

(cognitive overhead)。它同时支持卷积网络和循环网络，支持级联的模型或任意的图结构的模型（可以让某些数据跳过某些 Layer 和后面的 Layer 对接，使得创建 Inception 等复杂网络变得容易），从 CPU 上计算切换到 GPU 加速无须任何代码的改动。因为底层使用 Theano 或 TensorFlow，用 Keras 训练模型相比于前两者基本没有什么性能损耗（还可以享受前两者持续开发带来的性能提升），只是简化了编程的复杂度，节约了尝试新网络结构的时间。可以说模型越复杂，使用 Keras 的收益就越大，尤其是在高度依赖权值共享、多模型组合、多任务学习等模型上，Keras 表现得非常突出。Keras 所有的模块都是简洁、易懂、完全可配置、可随意插拔的，并且基本上没有任何使用限制，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 也包括绝大部分 state-of-the-art 的 Trick，包括 Adam、RMSProp、Batch Normalization、PReLU、ELU、LeakyReLU 等。同时，新的模块也很容易添加，这让 Keras 非常适合最前沿的研究。Keras 中的模型也都是在 Python 中定义的，不像 Caffe、CNTK 等需要额外的文件来定义模型，这样就可以通过编程的方式调试模型结构和各种超参数。在 Keras 中，只需要几行代码就能实现一个 MLP，或者十几行代码实现一个 AlexNet，这在其他深度学习框架中基本是不可能完成的任务。Keras 最大的问题可能是目前无法直接使用多 GPU，所以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。Keras 的编程模型设计和 Torch 很像，但是相比 Torch，Keras 构建在 Python 上，有一套完整的科学计算工具链，而 Torch 的编程语言 Lua 并没有这样一条科学计算工具链。无论从社区人数，还是活跃度来看，Keras 目前的增长速度都已经远远超过了 Torch。

MXNet

官网网址: <http://mxnet.io>

GitHub: <http://github.com/dmlc/mxnet>

MXNet 是 DMLC (Distributed Machine Learning Community) 开发的一款开源的、轻量级、可移植的、灵活的深度学习库，它让用户可以混合使用符号编程模式和指令式编程模式来最大化效率和灵活性，目前已经是 AWS 官方推荐的深度学习框架。MXNet 的很多作者都是中国人，其最大的贡献组织为百度，同时很多作者来自 cxxnet、minerva 和 purine2 等深度学习项目，可谓博采众家之长。它是各个框架中率先支持多 GPU 和分布式的，同时其分布式性能也非常高。MXNet 的核心是一个动态的依赖调度器，支持自动将计算任务并行化到多个 GPU 或分布式集群（支持 AWS、Azure、Yarn 等）。它上层的计算图优化算法可以让符号计算执行得非常快，而且节约内存，开启 mirror 模式会更加省内存，甚至可以在某些小内存 GPU 上训练其他框架因显存不够而训练不了的深度学习模型，也可以在移动设备（Android、iOS）上运行基于深度学习的图像识别等任务。此外，MXNet 的一个很大的优点是支持非常多的语言封装，比如 C++、Python、R、Julia、Scala、Go、MATLAB 和 JavaScript 等，可谓非常全面，基本主流脚本语言全部都支持了。在 MXNet 中构建一个网络需要的时间可能比 Keras、Torch 这类高度封装的框架要长，但是比直接用 Theano 等要快。MXNet 的各级系统架构（下面为硬件及操作系统底层，逐层向上为越来越抽象的接口）如图 2-6 所示。

图 2-6 MXNet 系统架构

DIGITS

官方网址: <http://developer.nvidia.com/digits>

GitHub: <http://github.com/NVIDIA/DIGITS>

DIGITS (Deep Learning GPU Training System) 不是一个标准的深度学习库, 它可以算是一个 Caffe 的高级封装 (或者 Caffe 的 Web 版培训系统)。因为封装得非常重, 以至于你不需要 (也不能) 在 DIGITS 中写代码, 即可实现一个深度学习的图片识别模型。在 Caffe 中, 定义模型结构、预处理数据、进行训练并监控训练过程是相对比较烦琐的, DIGITS 把所有这些操作都简化为在浏览器中执行。它可以算作 Caffe 在图片分类上的一个漂亮的用户可视化界面 (GUI), 计算机视觉的研究者或者工程师可以非常方便地设计深度学习模型、测试准确率, 以及调试各种超参数。同时使用它也可以生成数据和训练结果的可视化统计报表, 甚至是网络的可视化结构图。训练好的 Caffe 模型可以被 DIGITS 直接使用, 上传图片到服务器或者输入 url 即可对图片进行分类。

CNTK

官方网址: <http://cntk.ai>

GitHub: <http://github.com/Microsoft/CNTK>

CNTK (Computational Network Toolkit) 是微软研究院 (MSR) 开源的深度学习框架。它最早由 start the deep learning craze 的演讲人创建, 目前已经发展成一个通用的、跨平台的深度学习系统, 在语音识别领域的使用尤其广泛。CNTK 通过一个有向图将神经网络描述为一系列的运算操作, 这个有向图中子节点代表输入或网络参数, 其他节点代表各种矩阵运算。CNTK 支持各种前馈网络, 包括 MLP、CNN、RNN、LSTM、Sequence-to-Sequence 模型等, 也支持自动求解梯度。CNTK 有丰富的细粒度的神经网络组件, 使得用户不需要写底层的 C++ 或 CUDA, 就能通过组合这些组件设计新的复杂的 Layer。CNTK 拥有产品级的代码质量, 支持多机、多 GPU 的分布式训练。

CNTK 设计是性能导向的, 在 CPU、单 GPU、多 GPU, 以及 GPU 集群上都有非常优异的表现。同时微软最近推出的 1-bit compression 技术大大降低了通信代价, 让大规模并行训练拥有了很高的效率。CNTK 同时宣称拥有很高的灵活度, 它和 Caffe 一样通过配置文件定义网络结构, 再通过命令行程序执行训练, 支持构建任意的计算图, 支持 AdaGrad、RmsProp 等优化方法。它的另一个重要特性就是拓展性, CNTK 除了内置的大量运算核, 还允许用户定义他们自己的计算节点, 支持高度的定制化。CNTK 在 2016 年 9 月发布了对强化学习的支持, 同时, 除了通过写配置文件的方式定义网络结构, CNTK 还将支持其他语言的绑定, 包括 Python、C++ 和 C#, 这样用户就可以用编程的方式设计网络结构。CNTK 与 Caffe 一样也基于 C++ 并且跨平台, 大部分情况下, 它的部署非常简单。PC 上支持 Linux、Mac 和 Windows, 但是它目前不支持 ARM 架构, 限制了其在移动设备上的发挥。图 2-7 所示为 CNTK 目前的总体架构图。

图 2-7 CNTK 的总体架构图

CNTK 原生支持多 GPU 和分布式，从官网公布的对比评测来看，性能非常不错。在多 GPU 方面，CNTK 相对于其他的深度学习库表现得更突出，它实现了 1-bit SGD 和自适应的 mini-batching。图 2-8 所示为 CNTK 官网公布的在 2015 年 12 月的各个框架的性能对比。在当时，CNTK 是唯一支持单机 8 块 GPU 的框架，并且在分布式系统中可以超越 8 块 GPU 的性能。

图 2-8 CNTK 与各个框架的性能对比

Deeplearning4J

官方网址: <http://deeplearning4j.org/>

GitHub: <http://github.com/deeplearning4j/deeplearning4j>

Deeplearning4J (简称 DL4J) 是一个基于 Java 和 Scala 的开源的分布式深度学习库，由 Skyrmind 于 2014 年 6 月发布，其核心目标是创建一个即插即用的解决方案原型。埃森哲、雪弗兰、博斯咨询和 IBM 等都是 DL4J 的客户。DL4J 拥有一个多用途的 n-dimensional array 的类，可以方便地对数据进行各种操作；拥有多种后端计算核心，用以支持 CPU 及 GPU 加速，在图像识别等训练任务上的性能与 Caffe 相当；可以与 Hadoop 及 Spark 自动整合，同时可以方便地在现有集群（包括但不限于 AWS, Azure 等）上进行扩展，同时 DL4J 的并行化是根据集群的节点和连接自动优化，不像其他深度学习库那样可能需要用户手动调整。DL4J 选择 Java 作为其主要语言的原因是，目前基于 Java 的分布式计算、云计算、大数据的生态非常庞大。用户可能拥有大量的基于 Hadoop 和 Spark 的集群，因此在这类集群上搭建深度学习平台的需求便很容易被 DL4J 满足。同时 JVM 的生态圈内还有数不胜数的 Lipary 的支持，而 DL4J 也创建了 ND4J，可以说是 JVM 中的 NumPy，支持大规模的矩阵运算。此外，DL4J 还有商业版的支持，付费用户在出现问题时可以通过电话咨询寻求支持。

Chainer

官方网址: <http://chainer.org>

GitHub: <http://github.com/pfnet/chainer>

Chainer 是由日本公司 Preferred Networks 于 2015 年 6 月发布的深度学习框架。Chainer 对自己的特性描述如下。

Powerful: 支持 CUDA 计算，只需要几行代码就可以使用 GPU 加速，同时只需少许改

动就可以运行在多 GPU 上。

Flexible: 支持多种前馈神经网络，包括卷积网络、循环网络、递归网络，支持运行中动态定义的网络（Define-by-Run）。

Intuitive: 前馈计算可以引入 Python 的各种控制流，同时反向传播时不受干扰，简化了调试错误的难度。

绝大多数的深度学习框架是基于“Define-and-Run”的，也就是说，需要首先定义一个网络，再向网络中 feed 数据（mini-batch）。因为网络是预先静态定义的，所有的控制逻辑都需要以 data 的形式插入网络中，包括像 Caffe 那样定义好网络结构文件，或者像 Theano、Torch、TensorFlow 等使用编程语言定义网络。而 Chainer 则相反，网络是在实际运行中定义的，Chainer 存储历史运行的计算结果，而不是网络的结构逻辑，这样就可以方便地使用 Python 中的控制流，所以无须其他工作就可以直接在网络中使用条件控制和循环。

Leaf

官方网址: <http://autumnai.com/leaf/book>

GitHub: <http://github.com/autumnai/leaf>

Leaf 是一个基于 Rust 语言的直观的跨平台的深度学习乃至机器智能框架，它拥有一个清晰的架构，除了同属 Autumn AI 的底层计算库 Collenchyma，Leaf 没有其他依赖库。它易于维护和使用，并且拥有非常高的性能。Leaf 自身宣传的特点是为 Hackers 定制的，这里的 Hackers 是指希望用最短的时间和最少的精力实现机器学习算法的技术极客。它的可移植性非常好，可以运行在 CPU、GPU 和 FPGA 等设备上，可以支持有任何操作系统的 PC、服务器，甚至是没有操作系统的嵌入式设备，并且同时支持 OpenCL 和 CUDA。Leaf 是 Autumn AI 计划的一个重要组件，后者的目标是让人工智能算法的效率提高 100 倍。凭借其优秀的设计，Leaf 可以用来创建各种独立的模块，比如深度强化学习、可视化监控、网络部署、自动化预处理和大规模产品部署等。

Leaf 拥有最简单的 API，希望可以最简化用户需要掌握的技术栈。虽然才刚诞生不久，Leaf 就已经跻身最快的深度学习框架之一了。图 2-9 所示为 Leaf 官网公布的各个框架在单 GPU 上训练 VGG 网络的计算时间（越小越好）的对比（这是和早期的 TensorFlow 对比，最新版的 TensorFlow 性能已经非常好了）。

图 2-9 Leaf 和各深度学习框架的性能对比（深色为 forward，浅色为 backward）

DSSTNE

GitHub: <http://github.com/amznlabs/amazon-dsstne>

DSSTNE (Deep Scalable Sparse Tensor Network Engine) 是亚马逊开源的稀疏神经网络框架, 在训练非常稀疏的数据时具有很大的优势。DSSTNE 目前只支持全连接的神经网络, 不支持卷积网络等。和 Caffe 类似, 它也是通过写一个 JSON 类型的文件定义模型结构, 但是支持非常大的 Layer (输入和输出节点都非常多); 在激活函数、初始化方式及优化器方面基本都支持了 state-of-the-art 的方法, 比较全面; 支持大规模分布式的 GPU 训练, 不像其他框架一样主要依赖数据并行, DSSTNE 支持自动的模型并行 (使用数据并行需要在训练速度和模型准确度上做一定的 trade-off, 模型并行没有这个问题)。

在处理特征非常多 (上亿维) 的稀疏训练数据时 (经常在推荐、广告、自然语言处理任务中出现), 即使一个简单的三个隐层的 MLP (Multi-Layer Perceptron) 也会变成一个有非常多参数的模型 (可能高达上万亿)。以传统的稠密矩阵的方式训练方法很难处理这么多的模型参数, 更不必提超大规模的数据量, 而 DSSTNE 有整套的针对稀疏数据的优化, 率先实现了对超大稀疏数据训练的支持, 同时在性能上做了非常大的改进。

在 DSSTNE 官方公布的测试中, DSSTNE 在 MovieLens 的稀疏数据上, 在单 M40 GPU 上取得了比 TensorFlow 快 14.8 倍的性能提升 (注意是和老版的 TensorFlow 比较), 如图 2-10 所示。一方面是因为 DSSTNE 对稀疏数据的优化; 另一方面是 TensorFlow 在数据传输到 GPU 上时花费了大量时间, 而 DSSTNE 则优化了数据在 GPU 内的保留; 同时 DSSTNE 还拥有自动模型并行功能, 而 TensorFlow 中则需要手动优化, 没有自动支持。

应用领域

1. 文件分类
 - 文档文件分类
 - 音乐文件分类
 - 视频文件分类
2. 反欺诈
3. 自然语言处理 (NLP)
4. 图像处理
5. 智能安防

案例一

6月15日11时30分, 南京交警五大队民警根据南京交警指挥中心指令和前期研判的结果, 在北京西路云南路路口将一辆号牌为苏 AA8T**的白色别克轿车成功查获。

经现场核实, 该车驾驶人路某 (男, 38岁, 南京人), 因吸食冰毒其驾驶证早已被注销。没想到的是, 这辆车居然还是一辆正在营业的“网约车”。

经向车上乘客李某了解, 其是通过滴滴平台打到该车, 准备去珠江路办事已产生9元车资, 而滴滴平台显示该车的驾驶人是贾某, 明显与实际驾驶人不符, 涉嫌非法营运“网约车”。

路某交代, 这辆车是朋友的, 自己因吸食冰毒被警方查获, 其驾驶证于2015年9月就已经被注销, 随后一直没有重新取得驾驶证, 属于无证驾驶。经过进一步查询发现, 路某曾于2018年3月因无证驾驶被交警一大队查获并处罚, 如今已是二次无证驾驶, 随后南京交警五大队对路某无证驾驶的违法行为处以1000元罚款并处15日行政拘留。

案例二

5月29日9时30分根据前期分析研判,南京交警一大队民警早已在指定地点等候一辆号牌为苏 A1AA**的轿车出现,随后该车及驾驶人朱某宁(男,32岁,南京人)在太平门路口被南京交警一大队民警成功拦截。

面对现场民警的询问,驾驶人自称叫朱某龙(男,31岁,南京人),并拿出相应的驾驶证及身份证进行辩解。

根据前期调查的信息得知,朱某龙是朱某宁的亲弟弟,早在几年前已经离开南京。由于朱某宁和朱某龙的驾驶证照片非常相似,交警一大队立即通知朱某宁的妻子赴现场进行辨认,最终,驾驶人不得不承认自己就是重点“失驾”人员朱某宁,其2014年因醉酒驾车已被吊销驾驶证,为了逃避查处,非法使用他人驾驶证企图蒙混过关。

6. 广告
7. 时间序列分析(Time Series Analysis)
8. 聊天机器人(bot)
9. 机器人写诗歌
10. 机器翻译
11. 垃圾邮件过滤
12. 手写识别
13. 癌症筛选
14. 质量检测
15. 设备维护
16. 消费习惯分析
17. 行人识别
18. 棋类竞技
19. 量化交易
- 2015 年股灾
20. 智能物联网
21. 智能制造
22. 语音识别
23. 生物特征身份验证
24. OCR、票据识别、证件识别、车牌识别
25. 电影推荐
26. 情感分析
27. 员工行为分析
28. 比赛预测
29. 小娜、小艾、小菊

部署

pip 安装

Tensorflow 1.8 下载

<https://pypi.org/project/tensorflow/#files>

安装 Ubuntu 16.04 desktop

开启远程管理:

```
$ sudo apt install -y openssh-server
```

```
$ systemctl status ssh
```

修改服务器 ip 地址之后, 执行:

```
$ sudo ifconfig ens160 down
```

```
$ sudo ifconfig ens160 up
```

```
$ ifconfig
```

SecureCRT 连接 Ubuntu, 更新下系统:

```
$ sudo apt update
```

```
$ sudo apt upgrade -y
```

安装 pip

```
$ sudo apt-get install python-pip python-dev
```

```
tensorflow@tensorflow-vm:~$ pip -V
```

```
pip 8.1.1 from /usr/lib/python2.7/dist-packages (python 2.7)
```

升级 pip:

```
tensorflow@tensorflow-vm:~$ sudo pip -V
```

```
pip 8.1.1 from /usr/local/lib/python2.7/dist-packages/pip (python 2.7)
```

```
tensorflow@tensorflow-vm:~$ sudo pip install --upgrade pip
```

```
tensorflow@tensorflow-vm:~$ sudo pip -V
```

```
pip 10.0.1 from /usr/local/lib/python2.7/dist-packages/pip (python 2.7)
```

Pip 升级之后, 运行 pip:

```
ImportError: cannot import name main
```

修改 pip 文件

```
tensorflow@tfvm:/usr/bin$ cat pip
```

```
#!/usr/bin/python
```

```
# GENERATED BY DEBIAN
```

```
import sys
```

```
# Run the main entry point, similarly to how setuptools does it, but because
# we didn't install the actual entry point from setup.py, don't use the
# pkg_resources API.
from pip import __main__
if __name__ == '__main__':
    sys.exit(__main__._main())
```

第二种改法:

```
# -*- coding: utf-8 -*-
import re
import sys
from pip._internal import main as _main
if __name__ == '__main__':
    sys.argv[0] = re.sub(r'(-script\.pyw?|\\.exe)?$', '', sys.argv[0])
    sys.exit(_main())
```

```
sudo pip install --upgrade https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.8.0-
cp27-none-linux_x86_64.whl
```

安装 tensorflow (CPU)

```
tensorflow@tensorflow-vm:~$ sudo pip install --upgrade
https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.8.0-cp27-none-
linux_x86_64.whl
```

安装过程中, 一并安装了: wheel、protobuf、six、numpy、setuptools

```
$ pip install jupyter
```

```
$ jupyter notebook
```

```
$ jupyter notebook --ip=192.168.100.66
```

```
$ jupyter notebook --ip=192.168.100.66 --allow-root
```

```
$ jupyter notebook list
```

<http://192.168.100.66:8888>

容器部署

Github 地址:

<https://github.com/tensorflow/tensorflow/tree/master/tensorflow/tools/docker>

TensorFlowCPUImage 标识了 Docker 容器。指定下列值之一：

- gcr.io/tensorflow/tensorflow，这是 TensorFlow CPU binary image。
- gcr.io/tensorflow/tensorflow:latest-devel，这是最新的 TensorFlow CPU 二进制镜像加源代码。
- gcr.io/tensorflow/tensorflow:version，它是 TensorFlow CPU 二进制镜像的指定版本（例如，1.1.0rc1）。
- gcr.io/tensorflow/tensorflow:version-devel，它是 TensorFlow GPU 二进制镜像的源代码的指定版本（例如，1.1.0rc1）。

We currently maintain two Docker container images:

- tensorflow/tensorflow - TensorFlow with all dependencies - CPU only!
- tensorflow/tensorflow:latest-gpu - TensorFlow with all dependencies and support for NVidia CUDA

Note: We store all our containers on Docker Hub.

```
[root@Docker-GP ~]# docker search tensor --no-trunc
```

INDEX	NAME	DESCRIPTION	STARS
OFFICIAL	AUTOMATED		
docker.io	docker.io/tensorflow/tensorflow	Official docker images for deep learning framework TensorFlow (http://www.tensorflow.org)	928
docker.io	docker.io/jupyter/tensorflow-notebook	Jupyter Notebook Scientific Python Stack w/ Tensorflow from https://github.com/jupyter/docker-stacks	66
docker.io	docker.io/xblaster/tensorflow-jupyter	Dockerized Jupyter with tensorflow	50 [OK]
docker.io	docker.io/romilly/rpi-docker-tensorflow	Tensorflow and Jupyter running in docker container on Raspberry Pi 3B	18
docker.io	docker.io/bitnami/tensorflow-serving	Bitnami Docker Image for TensorFlow Serving	10 [OK]
docker.io	docker.io/floydhub/tensorflow	tensorflow	10 [OK]
docker.io	docker.io/tensorflow/tf_grpc_server	Server for TensorFlow GRPC Distributed Runtime	6
docker.io	docker.io/opensciencengrid/tensorflow-gpu	TensorFlow GPU set up for OSG	4
docker.io	docker.io/tensorflow/tf_grpc_test_server	Testing server for GRPC-based distributed runtime in TensorFlow	3
docker.io	docker.io/eboraas/tensorflow	TensorFlow with Jupyter Notebook, including CPU optimizations	2 [OK]
docker.io	docker.io/hytssk/tensorflow	tensorflow image with matplotlib.pyplot.imshow() enabled.	2 [OK]
docker.io	docker.io/abhishek404/tensorflow-gpu	Tensorflow GPU image	1
docker.io	docker.io/bitnami/tensorflow-inception	Bitnami Docker Image for TensorFlow Inception	1 [OK]

docker.io docker.io/chaneyk/tensorflow Tensorflow Releases with GPU Support
 1

docker.io docker.io/mikebirdgeneau/r-tensorflow RStudio and Tensorflow
 1 [OK]

docker.io docker.io/tensorlayer/tensorlayer
 https://github.com/tensorlayer/tensorlayer 1

docker.io docker.io/andreleoni/cnn-tensorflow Container for convlutional network
 with Python 3.6 + Tensorflow 0

docker.io docker.io/aretelabs/tensorflow 0

docker.io docker.io/davidchiu/tensorflow09 tensorflow09 with GPU support
 0

docker.io docker.io/djpetti/rpinets-tensorflow Tensorflow container that is ready to be
 used with RPinets. 0 [OK]

docker.io docker.io/fluxcapacitor/prediction-tensorflow 0

docker.io docker.io/mediadesignpractices/tensorflow Tensorflow w/ CUDA (GPU) +
 extras 0 [OK]

docker.io docker.io/opensciencenetwork/tensorflow TensorFlow image with some OSG
 additions 0 [root@Docker-GP ~]# docker pull
 docker.io/tensorflow/tensorflow
 Using default tag: latest
 Trying to pull repository docker.io/tensorflow/tensorflow ...
 latest: Pulling from docker.io/tensorflow/tensorflow
 297061f60c36: Pull complete
 e9ccef17b516: Pull complete
 dbc33716854d: Pull complete
 8fe36b178d25: Pull complete
 686596545a94: Pull complete
 ed66f2c5f3d9: Pull complete
 8405b6c3f141: Pull complete
 070615ca3a03: Pull complete
 306ac2321f8e: Pull complete
 c30111bc1e74: Pull complete
 7aa552c3f7f7: Pull complete
 4db41af3662a: Pull complete
 bf5fbadacf01: Pull complete
 Digest: sha256:1cc84937252fcc6e8521901cbbbe180c7a93300792aceb0da8a53a5728360320a
 Status: Downloaded newer image for docker.io/tensorflow/tensorflow:latest

Running the container
 Run non-GPU container using
 \$ docker run -it -p 8888:8888 tensorflow/tensorflow

For GPU support install NVidia drivers (ideally latest) and nvidia-docker. Run using
 \$ nvidia-docker run -it -p 8888:8888 tensorflow/tensorflow:latest-gpu

Note: If you would have a problem running nvidia-docker you may try the old method we have used. But it is not recommended. If you find a bug in nvidia-docker, please report it there and try using nvidia-docker as described above.

\$ # The old, not recommended way to run docker with gpu support:

\$ export CUDA_SO=\$(ls /usr/lib/x86_64-linux-gnu/libcuda.* | xargs -l{} echo '-v {}:{}'.')

\$ export DEVICES=\$(ls /dev/nvidia* | xargs -l{} echo '--device {}:{}'.')

\$ docker run -it -p 8888:8888 \$CUDA_SO \$DEVICES tensorflow/tensorflow:latest-gpu

```
[root@Docker-GP ~]# docker run -it -p 8888:8888 tensorflow/tensorflow
```

```
[I 03:23:53.508 NotebookApp] Writing notebook server cookie secret to /root/.local/share/jupyter/runtime/notebook_cookie_secret
```

```
[W 03:23:53.528 NotebookApp] WARNING: The notebook server is listening on all IP addresses and not using encryption. This is not recommended.
```

```
[I 03:23:53.537 NotebookApp] Serving notebooks from local directory: /notebooks
```

```
[I 03:23:53.537 NotebookApp] 0 active kernels
```

```
[I 03:23:53.537 NotebookApp] The Jupyter Notebook is running at:
```

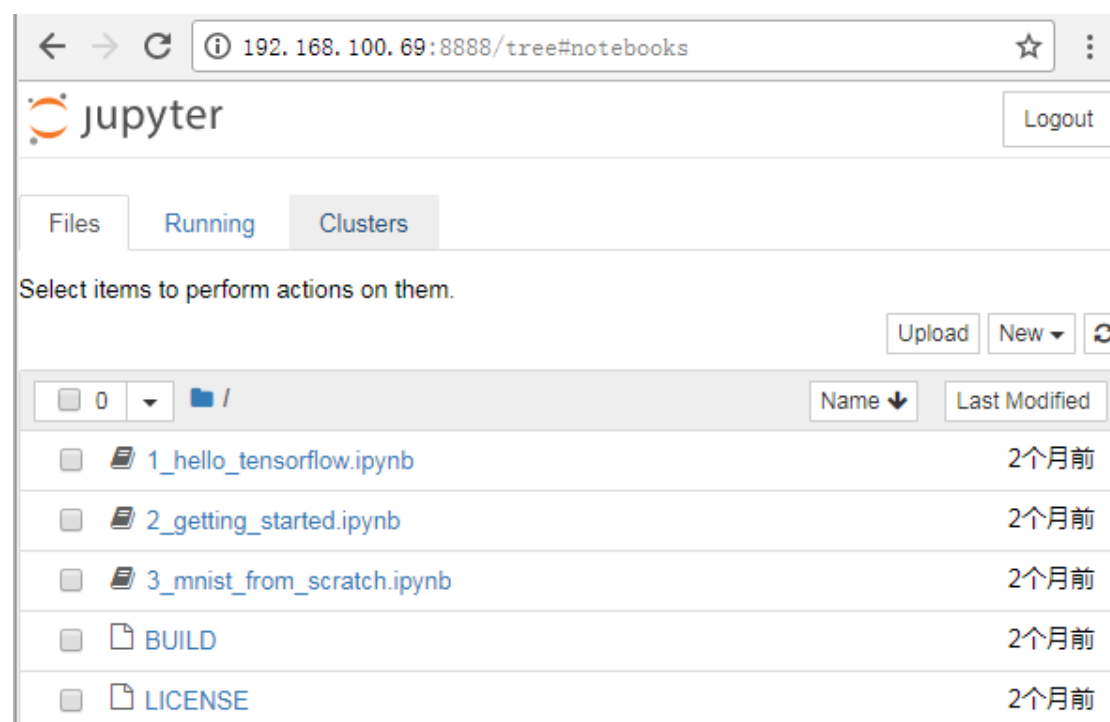
```
[I 03:23:53.537 NotebookApp] http://[all ip addresses on your system]:8888/?token=03e16d7d42944430a08789307676035f3e0661c9033f43d0
```

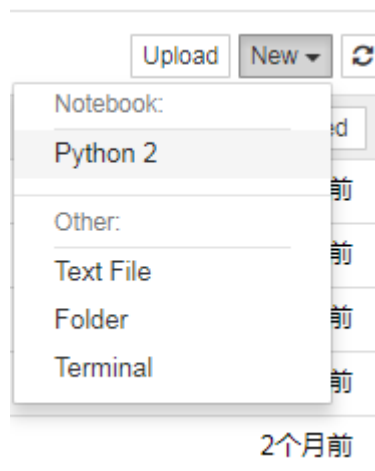
```
[I 03:23:53.537 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```

```
[C 03:23:53.538 NotebookApp]
```

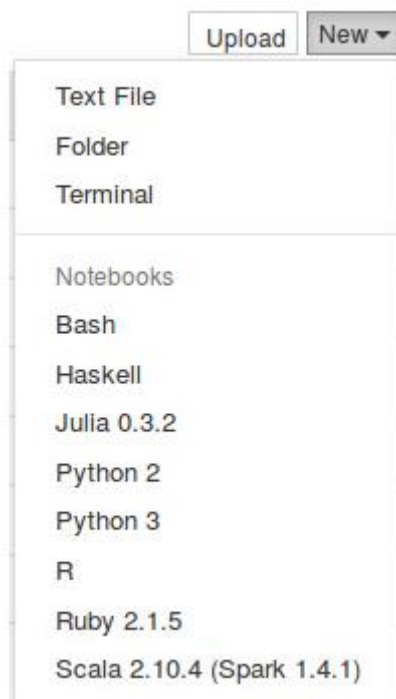
Copy/paste this URL into your browser when you connect for the first time, to login with a token:

<http://localhost:8888/?token=03e16d7d42944430a08789307676035f3e0661c9033f43d0>





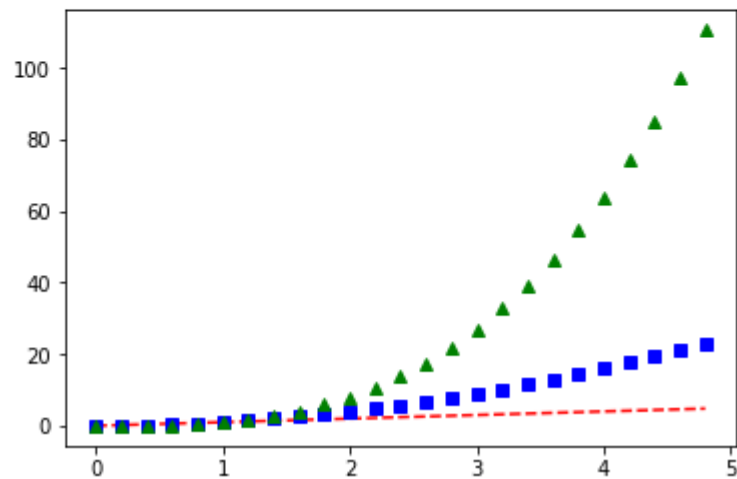
环境被简化了，看看隔壁老王的 jupyter:



```
In [23]: import numpy as np
import matplotlib.pyplot as plt

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```










```
import numpy as np
import matplotlib.pyplot as plt

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```

File Edit View Insert Cell Kernel Widgets Help

 Run Code ▾

In [1]: 1+6

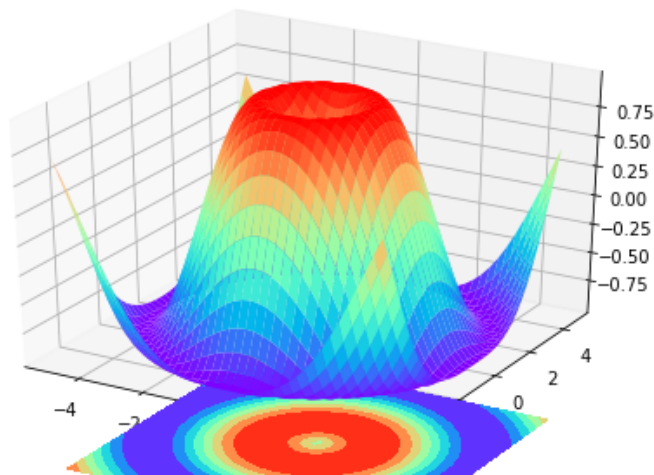
Out[1]: 7

In [2]: 2*7 -9

Out[2]: 5

In [14]: import math;
print str(math.sin(4)) +' * ' + str(math.sqrt(4))
-0.756802495308 * 2.0

```
In [22]: import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig=plt.figure()
ax=Axes3D(fig)
x=np.arange(-5,5,0.25)
y=np.arange(-5,5,0.25)
x,y=np.meshgrid(x,y)
r=np.sqrt(x**2+y**2)
z=np.sin(r)
#高度
ax.plot_surface(x,y,z,rstride=1,cstride=1,cmap=plt.get_cmap('rainbow'))
#填充rainbow颜色
ax.contourf(x,y,z,zdir='z',offset=-2,cmap='rainbow')
#绘制3D图形,zdir表示从哪个坐标轴上压下去
plt.show()
#显示图片
```



```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig=plt.figure()
ax=Axes3D(fig)
x=np.arange(-5,5,0.25)
y=np.arange(-5,5,0.25)
x,y=np.meshgrid(x,y)
r=np.sqrt(x**2+y**2)
z=np.sin(r)
#高度
ax.plot_surface(x,y,z,rstride=1,cstride=1,cmap=plt.get_cmap('rainbow'))
```

```
#填充 rainbow 颜色
ax.contourf(x,y,z,zdir='z',offset=-2,cmap='rainbow')
#绘制 3D 图形,zdir 表示从哪个坐标轴上压下去
plt.show()
#显示图片
```

打开一个 Terminal:

```
# python
Python 2.7.12 (default, Dec 4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
/usr/local/lib/python2.7/dist-packages/h5py/__init__.py:36: FutureWarning: Conversion of
precreated. In future, it will be treated as np.float64 == np.dtype(float).type`.
    from ._conv import register_converters as _register_converters
>>> hello= tf.constant('Hello, TensorFlow!')
>>> sess =tf.Session()
>>> print sess.run(hello)
Hello, TensorFlow!
>>> a=tf.constant(10)
>>> b=tf.constant(32)
>>> print sess.run(a+b)
42
>>> █
```

```
# python
Python 2.7.12 (default, Dec 4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
/usr/local/lib/python2.7/dist-packages/h5py/__init__.py:36: FutureWarning: Conversion of the
second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be
treated as `np.float64 == np.dtype(float).type`.
    from ._conv import register_converters as _register_converters
>>> hello= tf.constant('Hello, TensorFlow!')
>>> sess =tf.Session()
>>> print sess.run(hello)
Hello, TensorFlow!
>>> a=tf.constant(10)
>>> b=tf.constant(32)
>>> print sess.run(a+b)
42
>>>
```

```
[root@Docker-GP ~]# docker ps -a
[root@Docker-GP ~]# docker image ls
[root@Docker-GP ~]# docker start f8a59e6eb3e3
[root@Docker-GP ~]# docker stop f8a59e6eb3e3
[root@Docker-GP ~]# docker rm f8a59e6eb3e3
```

Tensorflow 生成散点图测试:

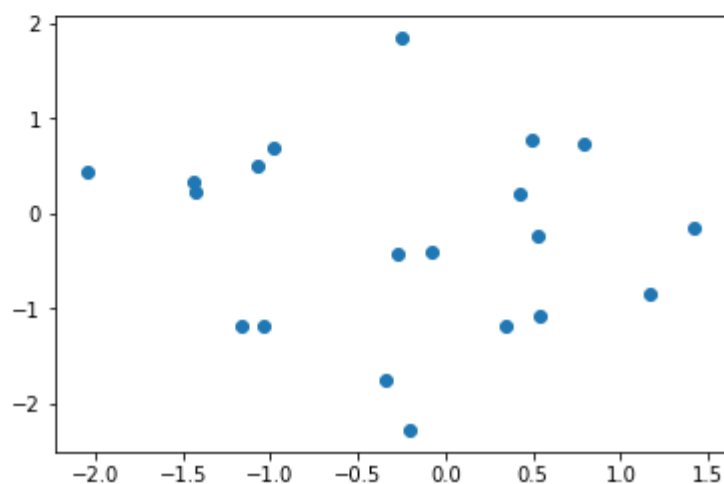

```
In [1]: import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

#表示直接在浏览器中显示matplotlib图表
%matplotlib inline

a = tf.random_normal([2,20]) #定义2x20的随机数矩阵 |
sess = tf.Session() #启动一个tensorflow会话
out = sess.run(a) #用在sess会话里执行a, 结果放out里
x, y = out

plt.scatter(x, y) #用pyplot创建一系列散列点, 坐标为x和y
plt.show()
```

/usr/local/lib/python2.7/dist-packages/h5py/__init__.py:36:
np.floating is deprecated. In future, it will be treated as
from ._conv import register_converters as _register_converters



```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
```

```
#表示直接在浏览器中显示 matplotlib 图表
%matplotlib inline
```

```

a = tf.random_normal([2,20]) #定义 2x20 的随机数矩阵
sess = tf.Session() #启动一个 tensorflow 会话
out = sess.run(a) # 用在 sess 会话里执行 a，结果放 out 里
x, y = out

plt.scatter(x, y) #用 pyplot 创建一系列散列点，坐标为 x 和 y
plt.show()

```

基于 VirtualEnv 的安装

基于 Anaconda 的安装

Anaconda 是一个集成许多第三方科学计算库的 Python 科学计算环境,Anaconda 使用 conda 作为自己的包管理工具,同时具有自己的计算环境,类似 Virtualenv.

和 Virtualenv 一样,不同 Python 工程需要的依赖包,conda 将他们存储在不同的地方。TensorFlow 上安装的 Anaconda 不会对之前安装的 Python 包进行覆盖.

- 安装 Anaconda
- 建立一个 conda 计算环境
- 激活环境,使用 conda 安装 TensorFlow
- 安装成功后,每次使用 TensorFlow 的时候需要激活 conda 环境

ANACONDA 下载

<https://www.anaconda.com/download/>

python 第一步——Annaconda 基础使用及 pycharm、spyder

<https://blog.csdn.net/u013818990/article/details/79329319#%E6%96%B0%E5%BB%BA%E4%B8%80%E4%B8%AApython%E7%8E%AF%E5%A2%83>

创建 python 3.6 测试环境:

```
tensor@tfvm:~$ conda create --name test_py3 python=3.6
```

```
Solving environment: done
```

```
## Package Plan ##
```

```
environment location: /opt/anaconda2/envs/test_py3
```

```
added / updated specs:
```

```
- python=3.6
```

The following packages will be downloaded:

package	build	
----- -----		
ca-certificates-2018.03.07	0	124 KB
setuptools-39.2.0	py36_0	551 KB
python-3.6.5	hc3d631a_2	29.4 MB
pip-10.0.1	py36_0	1.8 MB
sqlite-3.24.0	h84994c4_0	1.8 MB
wheel-0.31.1	py36_0	62 KB
certifi-2018.4.16	py36_0	142 KB

	Total:	33.8 MB

The following NEW packages will be INSTALLED:

```
ca-certificates: 2018.03.07-0
certifi:         2018.4.16-py36_0
libedit:         3.1.20170329-h6b74fdf_2
libffi:          3.2.1-hd88cf55_4
libgcc-ng:       7.2.0-hdf63c60_3
libstdcxx-ng:    7.2.0-hdf63c60_3
ncurses:         6.1-hf484d3e_0
openssl:         1.0.2o-h20670df_0
pip:             10.0.1-py36_0
python:          3.6.5-hc3d631a_2
readline:        7.0-ha6073c6_4
setuptools:      39.2.0-py36_0
sqlite:          3.24.0-h84994c4_0
tk:              8.6.7-hc745277_3
wheel:           0.31.1-py36_0
xz:              5.2.4-h14c3975_4
zlib:            1.2.11-ha838bed_2
```

Downloading and Extracting Packages

```
ca-certificates-2018 | 124 KB | ##### | 100%
setuptools-39.2.0    | 551 KB | ##### | 100%
python-3.6.5         | 29.4 MB | ##### | 100%
pip-10.0.1           | 1.8 MB | ##### | 100%
sqlite-3.24.0        | 1.8 MB | ##### | 100%
wheel-0.31.1         | 62 KB | ##### | 100%
certifi-2018.4.16    | 142 KB | ##### | 100%
```

Preparing transaction: done

Verifying transaction: done

Executing transaction: done

#

To activate this environment, use:

> [source activate test_py3](#)

#

To deactivate an active environment, use:

> [source deactivate](#)

#

【Windows 下使用没有 source，譬如：\$ activate test_py3、\$ deactivate】

创建 python 2.7 测试环境:

tensor@tfvm:~\$ conda create --name test_py2 python=2.7

Solving environment: done

Package Plan

environment location: /opt/anaconda2/envs/test_py2

added / updated specs:

- python=2.7

The following packages will be downloaded:

package	build	
certifi-2018.4.16	py27_0	142 KB
setuptools-39.2.0	py27_0	583 KB
Total:		726 KB

The following NEW packages will be INSTALLED:

ca-certificates:	2018.03.07-0
certifi:	2018.4.16-py27_0
libedit:	3.1.20170329-h6b74fdf_2
libffi:	3.2.1-hd88cf55_4
libgcc-ng:	7.2.0-hdf63c60_3
libstdcxx-ng:	7.2.0-hdf63c60_3
ncurses:	6.1-hf484d3e_0
openssl:	1.0.2o-h20670df_0
pip:	10.0.1-py27_0

```
python:          2.7.15-h1571d57_0
readline:        7.0-ha6073c6_4
setuptools:      39.2.0-py27_0
sqlite:          3.24.0-h84994c4_0
tk:              8.6.7-hc745277_3
wheel:           0.31.1-py27_0
zlib:            1.2.11-ha838bed_2
```

Proceed ([y]/n)? y

Downloading and Extracting Packages

```
certifi-2018.4.16 | 142 KB | ##### | 100%
setuptools-39.2.0 | 583 KB | ##### | 100%
```

Preparing transaction: done

Verifying transaction: done

Executing transaction: done

#

To activate this environment, use:

> [source activate test_py2](#)

#

To deactivate an active environment, use:

> [source deactivate](#)

#

【Windows 下使用没有 source，譬如：\$ activate test_py3、\$ deactivate】

INTEL DIGITS

英特尔® 深度学习 SDK 教程：安装指南：

<https://software.intel.com/zh-cn/articles/intel-deep-learning-sdk-tutorial>

英特尔® 深度学习 SDK 教程：英特尔® 深度学习 SDK 训练工具入门指南

<https://software.intel.com/zh-cn/articles/intel-deep-learning-sdk-tutorial-getting-started-with-intel-deep-learning-sdk-training-tool>

CUDA

CUDA 虚拟化

基本使用

Command Line

测试:

```
tensorflow@tensorflow-vm:~$ python
Python 2.7.12 (default, Dec  4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> hello=tf.constant('Hello, TensorFlow!')
>>> sess=tf.Session()
>>> print sess.run(hello)
Hello, TensorFlow!
>>> a=tf.constant(10)
>>> b= tf.constant(32)
>>> print sess.run(a+b)
42
>>> matrix1=tf.constant([[3.,3.]])
>>> matrix2=tf.constant([[2.],[2.]])
>>> product=tf.matmul(matrix1,matrix2)
>>> sess=tf.Session()
【大小写敏感】
>>> result=sess.run(product)
>>> print result
[[12.]]
>>> sess.close()
>>> with tf.Session() as sess:
...     result=sess.run([product])
...     print result
...
【回车两次】
[array([[12.]], dtype=float32)]
```

>>>

一、 矩阵的加法

定义2 设有两个 $m \times n$ 矩阵 $A=(a_{ij}), B=(b_{ij})$,
那末矩阵 A 与 B 的和记作 $A+B$, 规定为

$$A+B=\begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn} \end{pmatrix}$$

同型矩阵才能相加，两个 $m \times n$ 的矩阵的结果是 $m \times n$ 的矩阵。

矩阵的加法同理就好理解了。

如果设矩阵 D 为矩阵 A 与 B 的差，

那么： $D=A-B=$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} - \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{pmatrix} \\ = \begin{pmatrix} a_{11}-b_{11} & a_{12}-b_{12} & \cdots & a_{1n}-b_{1n} \\ a_{21}-b_{21} & a_{22}-b_{22} & \cdots & a_{2n}-b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1}-b_{m1} & a_{m2}-b_{m2} & \cdots & a_{mn}-b_{mn} \end{pmatrix}$$

矩阵线性代数教材上的各种定义都太过复杂了。尝试一个浅显的解释：

小明今天要做饭，消耗 2 斤肉，1 斤蔬菜。肉每斤 20 元，蔬菜每斤 5 元，则一共需多少花费？

这个问题的答案很简单：

$$20 \times 2 + 5 \times 1 = 45$$

我们用向量相乘的方法写出来：

$$(20 \ 5) \begin{pmatrix} 2 \\ 1 \end{pmatrix} = 45$$

如果小明第二天有另一种做饭的方法，需要消耗 1 斤肉，4 斤蔬菜，那么这两种方法的花费各是多少呢？我们显然需要另算这第二种方法的花费。把这个做饭方式写在第二个矩阵（向量是宽度或长度为 1 的矩阵）里：

$$(20 \ 5) \begin{pmatrix} 2 & 1 \\ 1 & 4 \end{pmatrix} = (45 \ 40)$$

小明家附近还有另一个菜市场，那里肉每斤 15 元，蔬菜每斤 10 元。那么，小明如果去这个菜市场，花费又是多少呢（分别计算上述两种做饭方式）？我们把这另外的一种价格写进第一个矩阵里：

$$\begin{pmatrix} 20 & 5 \\ 15 & 10 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 45 & 40 \\ 40 & 55 \end{pmatrix}$$

这样我们看到了一个矩阵乘法的例子。在左边的这个矩阵的每一行，都代表了一种价目表；在右边的矩阵的每一列，都代表了一种做饭方式。那么所有可能的组合所最终产生的花费，则在结果矩阵中表示出来了。

小明有一天成为了餐厅大厨，小红做掌柜兼管算账。我们假设物价不变。小红发现，如果今天买 10 斤肉花了 A 元，明天买 20 斤肉就得花 2A 元。如果买一斤肉要花 C 元，买 1 斤菜要花 D 元，那么买一斤肉和一斤菜就要花 (C+D) 元。每天小明汇报今日的材料消耗之后，小红便会将材料消耗转为需要花的钱数。如果材料消耗翻倍，花的钱数也翻倍。另外，如果去不同的菜市场，也会得到不同的花钱数量。

小明每月送来一张长列表，里面是每日的材料消耗；而经过小红的处理，这张列表会转为每日，在不同的菜市场购买这些材料的花费。材料消耗翻倍，花费也翻倍。我们管这种从材料列表转为开销表的过程，就叫做一个线性映射。这也即是矩阵乘法的意义。

三、矩阵与矩阵相乘

定义4 设 $A=(a_{ij})$ 是一个 $m \times s$ 矩阵, $B=(b_{ij})$ 是一个 $s \times n$ 矩阵, 那末规定矩阵 A 与矩阵 B 的乘积是一个 $m \times n$ 矩阵 $C=(c_{ij})$, 其中

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{is}b_{sj} = \sum_{k=1}^s a_{ik}b_{kj} \\ (i=1,2,\cdots,m; j=1,2,\cdots,n)$$

并把此乘积记作 $C=AB$

结果是 $m \times n$ 的矩阵

矩阵乘法满足的运算规律:

- (1) 结合律: $(AB)C = A(BC)$;
- (2) 分配律: $A(B+C) = AB + AC$,
 $(B+C)A = BA + CA$;
- (3) $\lambda(AB) = (\lambda A)B = A(\lambda B)$
- (4) $AE = EA = A$;

$$\begin{pmatrix} a_1 & & & \\ & a_2 & & \\ & & \ddots & \\ & & & a_n \end{pmatrix}_{n \times n} \begin{pmatrix} b_1 & & & \\ & b_2 & & \\ & & \ddots & \\ & & & b_n \end{pmatrix}_{n \times n} \\ = \begin{pmatrix} a_1 b_1 & & & \\ & a_2 b_2 & & \\ & & \ddots & \\ & & & a_n b_n \end{pmatrix}_{n \times n}$$

矩阵的加法:

Spyder

Python 2,Anaconda 2,spyder 2

Python 3,Anaconda 3,spyder 3

Ubuntu 16.04 desktop python 2 安装 spyder

```
tensorflow@tfvm:/usr/bin$ sudo apt install spyder
```

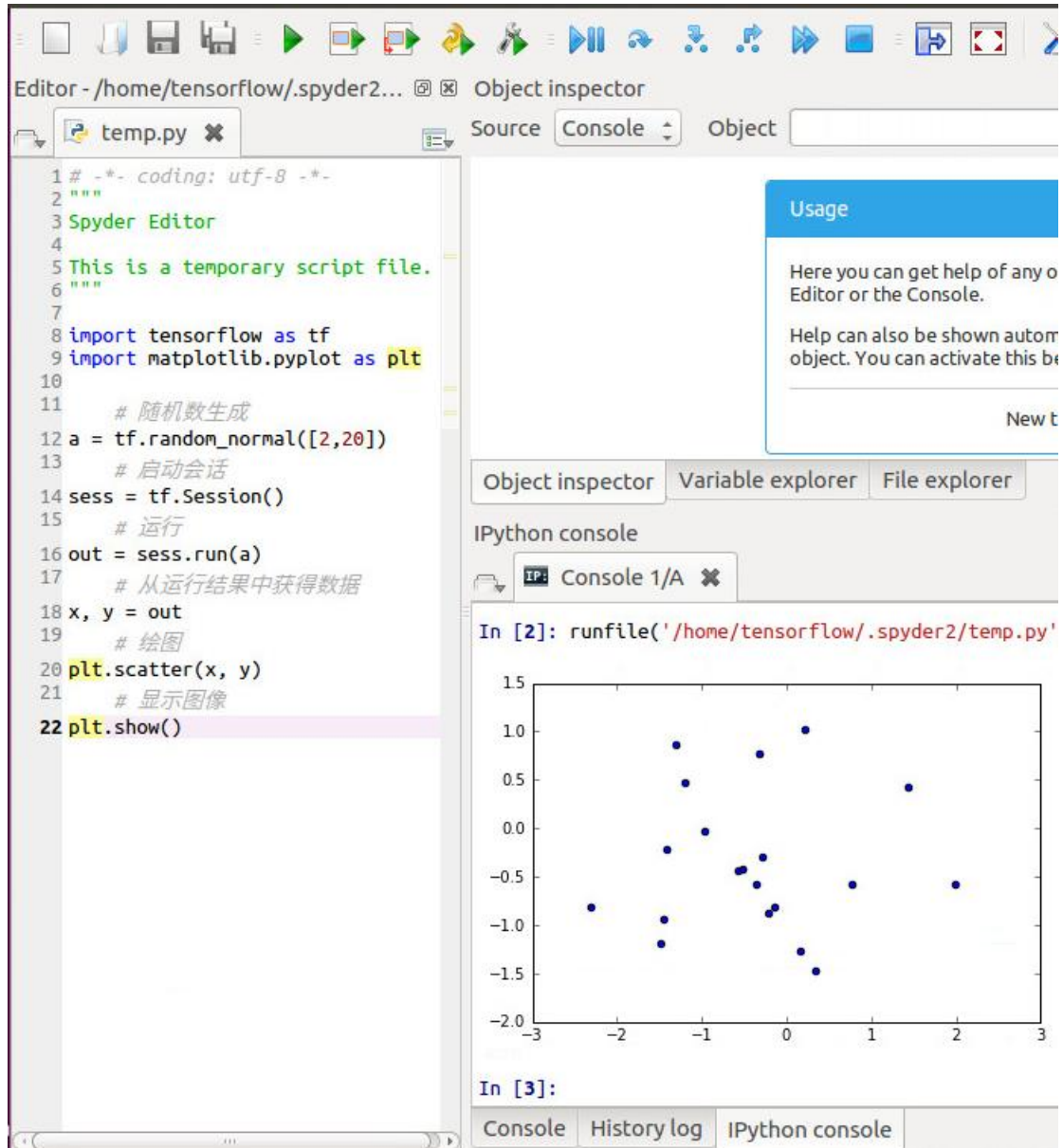
```
tensorflow@tfvm:/usr/bin$ spyder
```

Spyder: cannot connect to X server

【需要 X Windows 支持】

```
tensorflow@tfvm:/usr/bin$ spyder
```





Anaconda3 自带 spyder

tensorflow@tensorflow-vm:~\$ which spyder

/home/tensorflow/anaconda3/bin/spyder

\$ sudo apt install -y spyder3

启动:

\$ spyder3

Jupyter notebook

Jupyter 啥东东? 涨姿势的桀纣。

快速入门参考:

<https://www.cnblogs.com/nxld/p/6566380.html>

Jupyter 是软件（开发环境）套装。

Jupyter Notebook（此前被称为 IPython notebook）是一个交互式笔记本，支持运行 40 多种编程语言。

Jupyter Notebook 的本质是一个 Web 应用程序，便于创建和共享文学化程序文档，支持实时代码，数学方程，可视化和 markdown。用途包括：数据清理和转换，数值模拟，统计建模，机器学习等等。

数据挖掘领域中最热门的比赛 Kaggle 里的资料都是 Jupyter 格式。

```
tensorflow@tfvm:/usr/bin$ sudo pip install jupyter
```

通过 Anaconda 安装 Jupyter notebook

通过套件 anaconda 使用 jupyter notebook，进而调用 python 2 的 tensorflow 能保持与操作系统的 python 2 保持相对的独立性。

在此处下载:

<https://mirrors.tuna.tsinghua.edu.cn/help/anaconda/>

https://mirrors.tuna.tsinghua.edu.cn/anaconda/miniconda/Miniconda2-4.5.4-Linux-x86_64.sh

```
tensorflow@tfvm:~$ chmod +x Miniconda2-4.5.4-Linux-x86_64.sh
```

```
tensorflow@tfvm:~$ ./Miniconda2-4.5.4-Linux-x86_64.sh
```

【建议还是现在全版的 Anaconda】

如果在安装过程中没有选择将 Anaconda 的目录加入的 PATH 中，需要手工添加：

```
tensorflow@tfvm:~/anaconda2/bin$ export PATH=/home/tensorflow/anaconda2/bin:$PATH
```

anaconda2 使用自己的 python 2，所以在 anaconda2 中需要安装自身的 tensorflow。

查询安装信息

```
$ conda info
```

```
tensorflow@tfvm:~$ conda info
```

Current conda install:

```
platform : linux-64
```

```
conda version : 4.3.30
```

```
conda is private : False
```

```
conda-env version : 4.3.30
```

```
conda-build version : 3.10.5
  python version : 2.7.15.final.0
  requests version : 2.18.4
  root environment : /home/tensorflow/anaconda2 (writable)
default environment : /home/tensorflow/anaconda2
  envs directories : /home/tensorflow/anaconda2/envs
                    /home/tensorflow/.conda/envs
  package cache : /home/tensorflow/anaconda2/pkg
                 /home/tensorflow/.conda/pkg
  channel URLs : https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free/linux-64
                https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free/noarch
                https://repo.continuum.io/pkg/main/linux-64
                https://repo.continuum.io/pkg/main/noarch
                https://repo.continuum.io/pkg/free/linux-64
                https://repo.continuum.io/pkg/free/noarch
                https://repo.continuum.io/pkg/r/linux-64
                https://repo.continuum.io/pkg/r/noarch
                https://repo.continuum.io/pkg/pro/linux-64
                https://repo.continuum.io/pkg/pro/noarch
config file : /home/tensorflow/.condarc
netrc file : None
offline mode : False
user-agent : conda/4.3.30 requests/2.18.4 CPython/2.7.15 Linux/4.13.0-45-generic
debian/stretch/sid glibc/2.23
UID:GID : 1000:1000
```

查询当前已经安装的库

```
$ conda list
```

安装库(**代表库名称)

```
$ conda install **
```

更新库

```
$ conda update **
```

Anaconda 仓库镜像

官方下载更新工具包的速度很慢,所以继续添加清华大学 TUNA 提供的 Anaconda 仓库镜像,在终端或 cmd 中输入如下命令进行添加

```
$ conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free/
$ conda config --set show_channel_urls yes
```

```
$ conda install numpy #测试是否添加成功
```

之后会自动在用户根目录生成“.condarc”文件，Ubuntu 环境下路径为~/ .condarc，Windows 环境下路径为 C:\用户\your_user_name\.condarc

channels:

- <https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free/>
- defaults

show_channel_urls: yes

如果要删除镜像，直接删除“.condarc”文件即可

```
tensorflow@tfvm:~/anaconda2/bin$ anaconda search -t conda tensorflow
```

Using Anaconda API: <https://api.anaconda.org>

Packages:

Name	Version	Package Types	Platforms	Builds
GlaxoSmithKline/tensorflow	0.12.0	conda		linux-64

py27hb0d0e74_0

: TensorFlow is a machine learning library

HCC/tensorflow	1.7.0	conda	linux-64	py34_1, py27_1, py27_0, py36_0, np113py35_0, np113py27_0, np113py36_0, py35_0, py35_1
----------------	-------	-------	----------	--

: Computation using data flow graphs for scalable machine learning.

Run 'anaconda show <USER/PACKAGE>' to get installation details

```
tensorflow@tfvm:~/anaconda2/bin$ anaconda show HCC/tensorflow
```

Using Anaconda API: <https://api.anaconda.org>

Name: tensorflow

Summary: Computation using data flow graphs for scalable machine learning.

Access: public

Package Types: conda

Versions:

- + 0.12.1
- + 1.0.0
- + 1.3.1
- + 1.4.0
- + 1.5.0
- + 1.7.0

To install this package with conda run:

```
conda install --channel https://conda.anaconda.org/HCC tensorflow
```

```
tensorflow@tfvm:~/anaconda2/bin$ conda install --channel https://conda.anaconda.org/HCC tensorflow
```

Solving environment: done

Package Plan

environment location: /opt/anaconda2

added / updated specs:

- tensorflow

The following packages will be downloaded:

package	build		
----- -----			
tensorflow-1.4.0	py27_0	33.7 MB	HCC
libprotobuf-3.5.2	h6f1eeef_0	4.2 MB	
protobuf-3.5.2	py27hf484d3e_0	603 KB	
libgcc-7.2.0	h69d50b8_2	304 KB	

Total:	38.7 MB		

The following NEW packages will be INSTALLED:

libgcc: 7.2.0-h69d50b8_2
libprotobuf: 3.5.2-h6f1eeef_0
protobuf: 3.5.2-py27hf484d3e_0
tensorflow: 1.4.0-py27_0 HCC

Fetching package metadata

Solving package specifications: .

Package plan for installation in environment /home/tensorflow/anaconda2:

The following NEW packages will be INSTALLED:

absl-py: 0.1.10-py27_0 HCC
astor: 0.6.2-py27_0 defaults
backports.weakref: 1.0rc1-py27_0
<https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free>
gast: 0.2.0-py27_0 defaults
grpcio: 1.12.0-py27hdbcaa40_0 defaults
libgcc: 5.2.0-0 <https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free>
libprotobuf: 3.5.2-h6f1eeef_0 defaults
markdown: 2.6.9-py27_0

```

https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free
mock: 2.0.0-py27_0
https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free
pbr: 1.10.0-py27_0
https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free
protobuf: 3.5.2-py27hf484d3e_0 defaults
tensorboard: 1.7.0-np113py27_0 HCC
tensorflow: 1.7.0-np113py27_0 HCC
termcolor: 1.1.0-py27_0
https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free

```

The following packages will be UPDATED:

```

anaconda: 5.2.0-py27_3 defaults --> custom-py27_0
https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free

```

The following packages will be SUPERSEDED by a higher-priority channel:

```

bleach: 2.1.3-py27_0 defaults --> 1.5.0-py27_0
https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free
conda-env: 2.6.0-h36134e3_1 defaults --> 2.6.0-0
https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free
futures: 3.2.0-py27h7b459c0_0 defaults --> 3.1.1-py27_0
https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free
html5lib: 1.0.1-py27h5233db4_0 defaults --> 0.9999999-
py27_0 https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free

```

The following packages will be DOWNGRADED:

```

numpy: 1.14.3-py27hcd700cb_1 defaults --> 1.13.3-py27hdbf6ddf_4
defaults

```

Proceed ([y]/n)? y

```

conda-env-2.6. 100% |#####| Time: 0:00:00 776.95 kB/s
libgcc-5.2.0-0 100% |#####| Time: 0:00:00 4.80 MB/s
libprotobuf-3. 100% |#####| Time: 0:00:01 2.20 MB/s
anaconda-custo 100% |#####| Time: 0:00:00 6.44 MB/s
astor-0.6.2-py 100% |#####| Time: 0:00:00 5.22 MB/s
futures-3.1.1- 100% |#####| Time: 0:00:00 32.15 MB/s
gast-0.2.0-py2 100% |#####| Time: 0:00:00 10.18 MB/s
markdown-2.6.9 100% |#####| Time: 0:00:00 480.50 kB/s
termcolor-1.1. 100% |#####| Time: 0:00:00 17.43 MB/s
absl-py-0.1.10 100% |#####| Time: 0:00:02 43.35 kB/s

```



```
backports.weak 100% |#####| Time: 0:00:00 16.58 MB/s
html5lib-0.999 100% |#####| Time: 0:00:00 13.13 MB/s
protobuf-3.5.2 100% |#####| Time: 0:00:00 3.94 MB/s
bleach-1.5.0-p 100% |#####| Time: 0:00:00 30.15 MB/s
grpcio-1.12.0- 100% |#####| Time: 0:00:00 4.46 MB/s
pbr-1.10.0-py2 100% |#####| Time: 0:00:00 14.49 MB/s
mock-2.0.0-py2 100% |#####| Time: 0:00:00 14.59 MB/s
numpy-1.13.3-p 100% |#####| Time: 0:00:00 5.27 MB/s
tensorboard-1. 100% |#####| Time: 0:00:12 261.62 kB/s
tensorflow-1.7 100% |#####| Time: 0:00:45 884.91 kB/s
```

在浏览器中浏览 anaconda 下的 tensorflow channel:

<https://anaconda.org/search?q=tensorflow>

安装时指定版本号:

```
conda install --channel https://conda.anaconda.org/conda-forge tensorflow=1.0.0
```

```
tensor@tfvm:/opt$ conda install --channel https://conda.anaconda.org/anaconda
tensorflow=1.8.0
```

Solving environment: done

Package Plan

environment location: /opt/anaconda2

added / updated specs:

- tensorflow=1.8.0

The following packages will be downloaded:

package	build			
absl-py-0.2.2	py27_0	132 KB	anaconda	
certifi-2018.4.16	py27_0	142 KB	anaconda	
astor-0.6.2	py27_0	41 KB	anaconda	
_tfflow_180_select-3.0	eigen	2 KB	anaconda	
tensorflow-1.8.0	h7b2774c_0	3 KB	anaconda	
gast-0.2.0	py27_0	15 KB	anaconda	
termcolor-1.1.0	py27_1	7 KB	anaconda	
markdown-2.6.11	py27_0	102 KB	anaconda	
ca-certificates-2018.03.07	0	124 KB	anaconda	
protobuf-3.5.2	py27hf484d3e_0	603 KB	anaconda	
bleach-1.5.0	py27_0	21 KB	anaconda	
pbr-4.0.4	py27_0	114 KB	anaconda	

tensorboard-1.8.0		py27hf484d3e_0	3.0 MB	anaconda
tensorflow-base-1.8.0		py27h5f64886_0	40.0 MB	anaconda
conda-4.5.4		py27_0	1.0 MB	anaconda
html5lib-0.9999999		py27_0	183 KB	anaconda
libprotobuf-3.5.2		h6f1eeef_0	4.2 MB	anaconda
mock-2.0.0		py27h0c0c831_0	100 KB	anaconda
backports.weakref-1.0.post1		py27h0df1112_0	8 KB	anaconda
grpcio-1.12.1		py27hdbcaa40_0	1.7 MB	anaconda
openssl-1.0.2o		h20670df_0	3.4 MB	anaconda

Total:		55.0 MB		

The following NEW packages will be INSTALLED:

_tfflow_180_select:	3.0-eigen	anaconda
absl-py:	0.2.2-py27_0	anaconda
astor:	0.6.2-py27_0	anaconda
backports.weakref:	1.0.post1-py27h0df1112_0	anaconda
gast:	0.2.0-py27_0	anaconda
grpcio:	1.12.1-py27hdbcaa40_0	anaconda
libprotobuf:	3.5.2-h6f1eeef_0	anaconda
markdown:	2.6.11-py27_0	anaconda
mock:	2.0.0-py27h0c0c831_0	anaconda
pbr:	4.0.4-py27_0	anaconda
protobuf:	3.5.2-py27hf484d3e_0	anaconda
tensorboard:	1.8.0-py27hf484d3e_0	anaconda
tensorflow:	1.8.0-h7b2774c_0	anaconda
tensorflow-base:	1.8.0-py27h5f64886_0	anaconda
termcolor:	1.1.0-py27_1	anaconda

The following packages will be REMOVED:

anaconda:	5.2.0-py27_3
-----------	--------------

The following packages will be UPDATED:

ca-certificates:	2018.03.07-0	--> 2018.03.07-0	anaconda
certifi:	2018.4.16-py27_0	--> 2018.4.16-py27_0	anaconda
conda:	4.5.4-py27_0	--> 4.5.4-py27_0	anaconda
openssl:	1.0.2o-h20670df_0	--> 1.0.2o-h20670df_0	anaconda

The following packages will be DOWNGRADED:

bleach:	2.1.3-py27_0	--> 1.5.0-py27_0	anaconda
---------	--------------	------------------	----------

```
html5lib:          1.0.1-py27h5233db4_0    --> 0.9999999-py27_0  anaconda
```

```
tensorflow@tfvm:~$ python -V
Python 2.7.15 :: Anaconda custom (64-bit)
```

错误提示:

```
ImportError: No module named google.protobuf
```

```
tensor@tfvm:/opt$ conda install protobuf
```

```
ImportError: cannot import name pywrap_tensorflow
```

这个问题是 PATH 的问题，安装 tensorflow 之后没有更新 PATH 设置，重启之后正常。

```
tensor@tfvm:/opt$ sudo reboot
```

确认 tensorflow 版本:

```
tensor@tfvm:~$ conda list tensorflow
```

```
# packages in environment at /opt/anaconda2:
```

```
#
```

# Name	Version	Build	Channel
tensorflow	1.8.0	h7b2774c_0	anaconda
tensorflow-base	1.8.0	py27h5f64886_0	anaconda

```
tensor@tfvm:~$ anaconda --version
```

```
anaconda Command line client (version 1.6.14)
```

```
tensor@tfvm:~$ conda --v
```

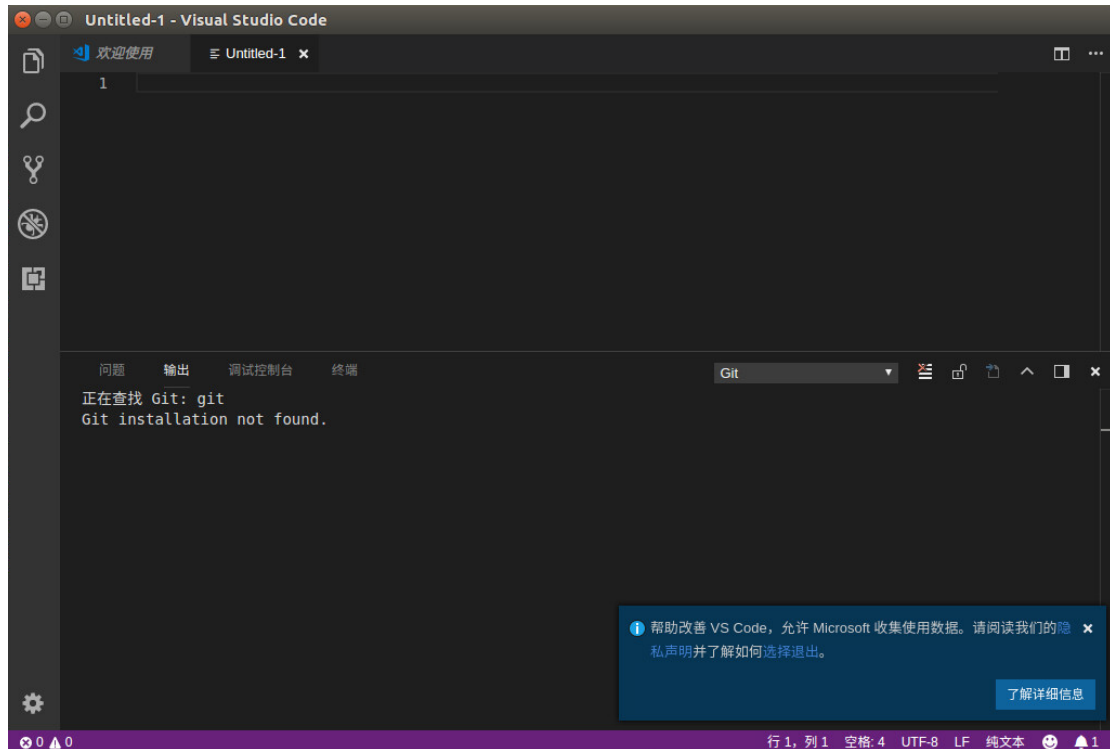
```
conda 4.5.4
```

```
tensorflow@tfvm:~/anaconda2/bin$ jupyter notebook --ip=192.168.100.66
```

```
tensorflow@tfvm:~/anaconda2/bin$ jupyter notebook list
```

Anaconda 2 中还包括了 VSCode (微软推出的免费的轻量级的编辑器)，启动命令

```
tensorflow@tfvm:~$ code
```



<https://blog.csdn.net/u011258217/article/details/78693564>

DOIT Simply

图形对象检测

Python 3、anaconda、conda 4.5.4、tensorflow 1.8、keras 2.1.6、ImageAI 2.0.1

街景识别

识别对象：car、person、handbag 以及动物等物体。

```
zebra : 60.902196168899536
kite : 63.8006865978241
teddy bear : 74.13351535797119
giraffe : 94.53464150428772
```

参考：

<https://mp.weixin.qq.com/s/Moi1-yGLOJOAbHYxYL6Hjw>

在 test_py3 环境下重复过程（ImageAI 官网提供的安装包是 python 3 的）：

```
tensor@tfvm:~$ source activate test_py3
```

```
(test_py3) tensor@tfvm:~$ conda list tensorflow
```

```
# packages in environment at /opt/anaconda2/envs/test_py3:
```

```
#
```

```
# Name                               Version                               Build  Channel
```

```
没有 tensorflow
```

```
(test_py3) tensor@tfvm:~$ mkdir ~/.pip
```

```
(test_py3) tensor@tfvm:~$ vi ~/.pip/pip.conf
```

```
[global]
```

```
index-url = https://pypi.tuna.tsinghua.edu.cn/simple
```

```
(test_py3) tensor@tfvm:~$ conda install --channel https://conda.anaconda.org/anaconda tensorflow=1.8.0
```

```
Solving environment: done
```

```
## Package Plan ##
```

```
environment location: /opt/anaconda2/envs/test_py3
```

```
added / updated specs:
```

```
- tensorflow=1.8.0
```

The following packages will be downloaded:

package	build			
six-1.11.0	py36h372c433_1	21 KB	anaconda	
mkl-2018.0.3	1	198.7 MB	anaconda	
blas-1.0	mkl	6 KB	anaconda	
astor-0.6.2	py36_0	42 KB	anaconda	
tensorflow-base-1.8.0	py36h5f64886_0	40.1 MB	anaconda	
grpcio-1.12.1	py36hdbcaa40_0	1.7 MB	anaconda	
markdown-2.6.11	py36_0	104 KB	anaconda	
ca-certificates-2018.03.07	0	124 KB	anaconda	
tensorboard-1.8.0	py36hf484d3e_0	3.1 MB	anaconda	
libfortran-ng-7.2.0	hdf63c60_3	1.2 MB	anaconda	
certifi-2018.4.16	py36_0	142 KB	anaconda	
absl-py-0.2.2	py36_0	135 KB	anaconda	
intel-openmp-2018.0.3	0	705 KB	anaconda	
numpy-1.14.5	py36hcd700cb_0	94 KB	anaconda	

gast-0.2.0		py36_0	15 KB	anaconda
mkl_fft-1.0.1		py36h3010b51_0	140 KB	anaconda
werkzeug-0.14.1		py36_0	423 KB	anaconda
mkl_random-1.0.1		py36h629b387_0	373 KB	anaconda
openssl-1.0.2o		h20670df_0	3.4 MB	anaconda
termcolor-1.1.0		py36_1	7 KB	anaconda
bleach-1.5.0		py36_0	22 KB	anaconda
tensorflow-1.8.0		h57681fa_0	3 KB	anaconda
numpy-base-1.14.5		py36hdbf6ddf_0	4.1 MB	anaconda
protobuf-3.5.2		py36hf484d3e_0	610 KB	anaconda
html5lib-0.999999		py36_0	176 KB	anaconda

Total:			255.5 MB	

The following NEW packages will be INSTALLED:

_tfflow_180_select:	3.0-eigen	anaconda
absl-py:	0.2.2-py36_0	anaconda
astor:	0.6.2-py36_0	anaconda
blas:	1.0-mkl	anaconda
bleach:	1.5.0-py36_0	anaconda
gast:	0.2.0-py36_0	anaconda
grpcio:	1.12.1-py36hdbcaa40_0	anaconda
html5lib:	0.999999-py36_0	anaconda
intel-openmp:	2018.0.3-0	anaconda
libgfortran-ng:	7.2.0-hdf63c60_3	anaconda
libprotobuf:	3.5.2-h6f1eef_0	anaconda
markdown:	2.6.11-py36_0	anaconda
mkl:	2018.0.3-1	anaconda
mkl_fft:	1.0.1-py36h3010b51_0	anaconda
mkl_random:	1.0.1-py36h629b387_0	anaconda
numpy:	1.14.5-py36hcd700cb_0	anaconda
numpy-base:	1.14.5-py36hdbf6ddf_0	anaconda
protobuf:	3.5.2-py36hf484d3e_0	anaconda
six:	1.11.0-py36h372c433_1	anaconda
tensorboard:	1.8.0-py36hf484d3e_0	anaconda
tensorflow:	1.8.0-h57681fa_0	anaconda
tensorflow-base:	1.8.0-py36h5f64886_0	anaconda
termcolor:	1.1.0-py36_1	anaconda
werkzeug:	0.14.1-py36_0	anaconda

The following packages will be UPDATED:

ca-certificates:	2018.03.07-0	--> 2018.03.07-0	anaconda
------------------	--------------	------------------	----------

```
certifi:          2018.4.16-py36_0          --> 2018.4.16-py36_0  anaconda
openssl:         1.0.2o-h20670df_0        --> 1.0.2o-h20670df_0 anaconda
```

Proceed ([y]/n)? y

Downloading and Extracting Packages

```
six-1.11.0          | 21 KB | ##### | 100%
mkl-2018.0.3        | 198.7 MB | ##### | 100%
blas-1.0            | 6 KB | ##### | 100%
astor-0.6.2         | 42 KB | ##### | 100%
tensorflow-base-1.8. | 40.1 MB | ##### | 100%
grpcio-1.12.1       | 1.7 MB | ##### | 100%
markdown-2.6.11     | 104 KB | ##### | 100%
ca-certificates-2018 | 124 KB | ##### | 100%
tensorboard-1.8.0   | 3.1 MB | ##### | 100%
libgfortran-ng-7.2.0 | 1.2 MB | ##### | 100%
certifi-2018.4.16    | 142 KB | ##### | 100%
absl-py-0.2.2        | 135 KB | ##### | 100%
intel-openmp-2018.0. | 705 KB | ##### | 100%
numpy-1.14.5         | 94 KB | ##### | 100%
gast-0.2.0           | 15 KB | ##### | 100%
mkl_fft-1.0.1        | 140 KB | ##### | 100%
werkzeug-0.14.1      | 423 KB | ##### | 100%
mkl_random-1.0.1     | 373 KB | ##### | 100%
openssl-1.0.2o       | 3.4 MB | ##### | 100%
termcolor-1.1.0      | 7 KB | ##### | 100%
bleach-1.5.0         | 22 KB | ##### | 100%
tensorflow-1.8.0     | 3 KB | ##### | 100%
numpy-base-1.14.5    | 4.1 MB | ##### | 100%
protobuf-3.5.2       | 610 KB | ##### | 100%
html5lib-0.9999999   | 176 KB | ##### | 100%
```

Preparing transaction: done

Verifying transaction: done

Executing transaction: done

```
(test_py3) tensor@tfvm:~$ sudo apt install python3-pip
```

```
(test_py3) tensor@tfvm:~$ pip install numpy
```

Looking in indexes: <https://pypi.tuna.tsinghua.edu.cn/simple>

Requirement already satisfied: numpy in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (1.14.5)

mkl-random 1.0.1 requires cython, which is not installed.

mkl-fft 1.0.0 requires cython, which is not installed.

```
(test_py3) tensor@tfvm:~$ pip install scipy
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting scipy
  Downloading
https://pypi.tuna.tsinghua.edu.cn/packages/a8/0b/f163da98d3a01b3e0ef1cab8dd2123c34aee2
bafbb1c5bffa354cc8a1730/scipy-1.1.0-cp36-cp36m-manylinux1_x86_64.whl (31.2MB)
    100% |████████████████████████████████████████████████████████████████████████████████| 31.2MB 1.5MB/s
Requirement          already          satisfied:          numpy>=1.8.2          in
/opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from scipy) (1.14.5)
mkl-random 1.0.1 requires cython, which is not installed.
mkl-fft 1.0.0 requires cython, which is not installed.
Installing collected packages: scipy
Successfully installed scipy-1.1.0
```

```
(test_py3) tensor@tfvm:~$ pip install opencv-python
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting opencv-python
  Downloading
https://pypi.tuna.tsinghua.edu.cn/packages/e6/d1/732afb3a056d7e7f3af08f3fcb67a7c1ceedd6b
e941f8e3907da0400c36e/opencv_python-3.4.0.14-cp36-cp36m-manylinux1_x86_64.whl
(24.8MB)
    100% |████████████████████████████████████████████████████████████████████████████████| 24.8MB 421kB/s
Requirement          already          satisfied:          numpy>=1.11.3          in
/opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from opencv-python) (1.14.5)
mkl-random 1.0.1 requires cython, which is not installed.
mkl-fft 1.0.0 requires cython, which is not installed.
Installing collected packages: opencv-python
Successfully installed opencv-python-3.4.0.14
```

```
(test_py3) tensor@tfvm:~$ pip install pillow
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting pillow
  Downloading
https://pypi.tuna.tsinghua.edu.cn/packages/5f/4b/8b54ab9d37b93998c81b364557dff9f61972c0
f650efa0ceaf470b392740/Pillow-5.1.0-cp36-cp36m-manylinux1_x86_64.whl (2.0MB)
    100% |████████████████████████████████████████████████████████████████████████████████| 2.0MB 16.8MB/s
mkl-random 1.0.1 requires cython, which is not installed.
mkl-fft 1.0.0 requires cython, which is not installed.
Installing collected packages: pillow
Successfully installed pillow-5.1.0
```

```
(test_py3) tensor@tfvm:~$ pip install matplotlib
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
```



```
Downloading
https://pypi.tuna.tsinghua.edu.cn/packages/49/b8/89dbd27f2fb171ce753bb56220d4d4f6dbc5fe
32b95d8edc4415782ef07f/matplotlib-2.2.2-cp36-cp36m-manylinux1_x86_64.whl (12.6MB)
100% |██████████████████████████████████████████| 12.6MB 3.3MB/s
```

Downloading
<https://pypi.tuna.tsinghua.edu.cn/packages/f7/d2/e07d3ebb2bd7af696440ce7e754c59dd546ffe1bbe732c8ab68b9c834e61/cycler-0.10.0-py2.py3-none-any.whl>

```
Downloading https://pypi.tuna.tsinghua.edu.cn/packages/69/a7/88719d132b18300b4369fbffa741841cfd36d1e637e1990f27929945b538/kiwisolver-1.0.1-cp36-cp36m-manylinux1_x86_64.whl (949kB)
100% |██████████████████████████████████████████| 952kB 24.2MB/s
```

```
Downloading
https://pypi.tuna.tsinghua.edu.cn/packages/dc/83/15f7833b70d3e067ca91467ca245bae0f6fe56
ddc7451aa0dc5606b120f2/pytz-2018.4-py2.py3-none-any.whl (510kB)
100% |██████████████████████████████████████████| 512kB 11.9MB/s
```

Collecting python-dateutil>=2.1 (from matplotlib)

[illegible]

Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 (from matplotlib)

[illegible]

mkl-random 1.0.1 requires cython, which is not installed.

mkl-fft 1.0.0 requires cython, which is not installed.

Installing collected packages: cyciler, kiwisolver, pytz, python-dateutil, pyparsing, matplotlib

```
pip install h5pySuccessfully installed cycler-0.10.0 kiwisolver-1.0.1 matplotlib-2.2.2 pyparsing-2.2.0
python-dateutil-2.7.3 pytz-2018.4
```

```
(test py3) tensor@tfvm:~$ pip install h5py
```

Looking in indexes: <https://pypi.tuna.tsinghua.edu.cn/simple>

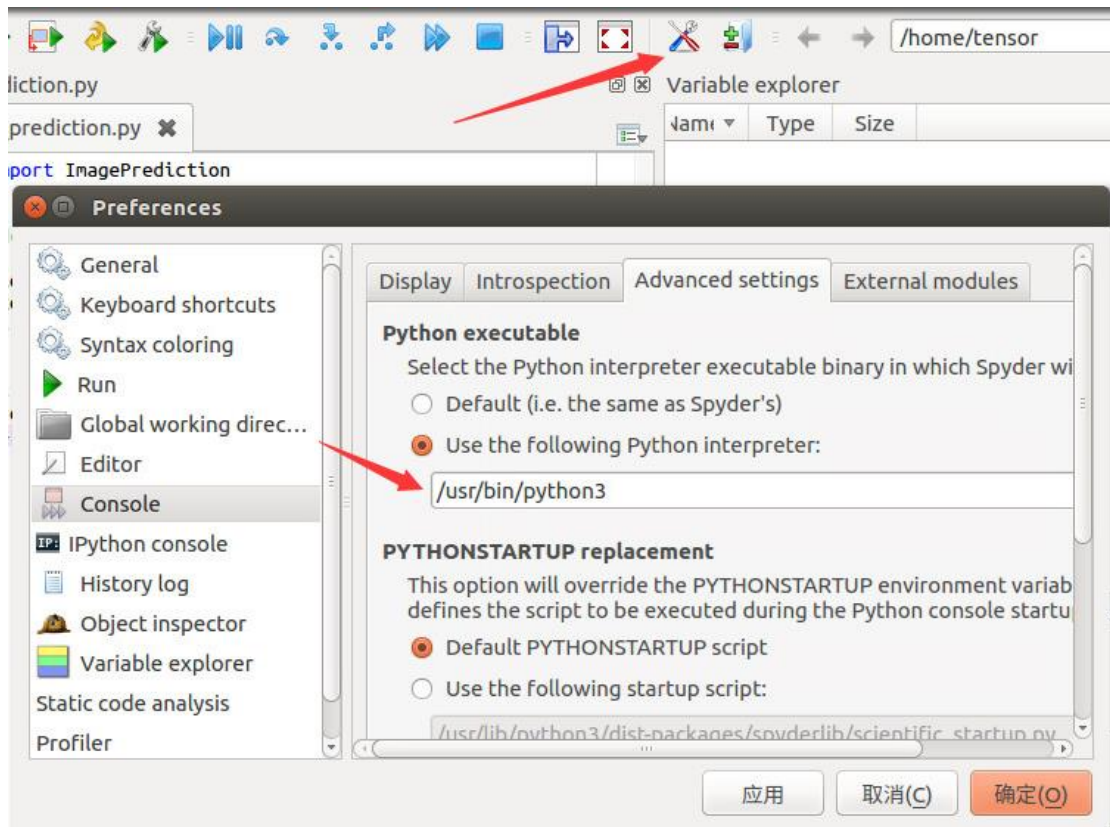
Collecting h5py


```
(test_py3) tensor@tfvm:~$ pip install
https://github.com/OlafenwaMoses/ImageAI/releases/download/2.0.1/imageai-2.0.1-py3-none-
any.whl
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting imageai==2.0.1 from
https://github.com/OlafenwaMoses/ImageAI/releases/download/2.0.1/imageai-2.0.1-py3-none-
any.whl
  Downloading https://github.com/OlafenwaMoses/ImageAI/releases/download/2.0.1/imageai-
2.0.1-py3-none-any.whl (137kB)
    100% |██████████████████████████████████████████████████████████████████████████| 143kB 94kB/s
mkl-random 1.0.1 requires cython, which is not installed.
mkl-fft 1.0.0 requires cython, which is not installed.
Installing collected packages: imageai
Successfully installed imageai-2.0.1
```

```
(test_py3) tensor@tfvm:~$ pip install cython
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting cython
  Downloading
https://pypi.tuna.tsinghua.edu.cn/packages/6f/79/d8e2cd00bea8156a995fb284ce7b6677c49ecc
d2d318f73e201a9ce560dc/Cython-0.28.3-cp36-cp36m-manylinux1_x86_64.whl (3.4MB)
    100% |██████████████████████████████████████████████████████████████████████████| 3.4MB 9.5MB/s
Installing collected packages: cython
Successfully installed cython-0.28.3
```

```
(test_py3) tensor@tfvm:~$ pip -V
pip 10.0.1 from /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages/pip (python 3.6)
```

```
(test_py3) tensor@tfvm:~$ sudo apt install spyder3
(test_py3) tensor@tfvm:~$ spyder3
```



在环境 test_py3 下测试成功:

下载 tensorflow-1.8.0-cp36-cp36m-manylinux1_x86_64.whl 之后 pip 安装:

```
(test_py3) tensor@tfvm:~$ pip install tensorflow-1.8.0-cp36-cp36m-manylinux1_x86_64.whl
```

Looking in indexes: <https://pypi.tuna.tsinghua.edu.cn/simple>

Requirement already satisfied: tensorflow==1.8.0 from file:///home/tensor/tensorflow-1.8.0-cp36-cp36m-manylinux1_x86_64.whl in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (1.8.0)

Requirement already satisfied: absl-py>=0.1.6 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (0.2.2)

Requirement already satisfied: wheel>=0.26 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (0.31.1)

Requirement already satisfied: protobuf>=3.4.0 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (3.5.2)

Requirement already satisfied: termcolor>=1.1.0 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (1.1.0)

Requirement already satisfied: six>=1.10.0 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (1.11.0)

Requirement already satisfied: grpcio>=1.8.6 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (1.12.1)

Requirement already satisfied: gast>=0.2.0 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (0.2.0)

Requirement already satisfied: numpy>=1.13.3 in

/opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (1.14.5)

Requirement already satisfied: tensorboard<1.9.0,>=1.8.0 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (1.8.0)

Requirement already satisfied: astor>=0.6.0 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (0.6.2)

Requirement already satisfied: setuptools in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from protobuf>=3.4.0->tensorflow==1.8.0) (39.2.0)

Requirement already satisfied: werkzeug>=0.11.10 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorboard<1.9.0,>=1.8.0->tensorflow==1.8.0) (0.14.1)

Requirement already satisfied: html5lib==0.9999999 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorboard<1.9.0,>=1.8.0->tensorflow==1.8.0) (0.9999999)

Requirement already satisfied: markdown>=2.6.8 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorboard<1.9.0,>=1.8.0->tensorflow==1.8.0) (2.6.11)

Requirement already satisfied: bleach==1.5.0 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorboard<1.9.0,>=1.8.0->tensorflow==1.8.0) (1.5.0)

(test_py3) tensor@tfvm:~\$ pip install tensorflow-1.8.0-cp36-cp36m-manylinux1_x86_64.whl

Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple

Requirement already satisfied: tensorflow==1.8.0 from file:///home/tensor/tensorflow-1.8.0-cp36-cp36m-manylinux1_x86_64.whl in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (1.8.0)

Requirement already satisfied: astor>=0.6.0 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (0.6.2)

Requirement already satisfied: absl-py>=0.1.6 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (0.2.2)

Requirement already satisfied: gast>=0.2.0 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (0.2.0)

Requirement already satisfied: protobuf>=3.4.0 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (3.5.2)

Requirement already satisfied: numpy>=1.13.3 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (1.14.5)

Requirement already satisfied: six>=1.10.0 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (1.11.0)

Requirement already satisfied: termcolor>=1.1.0 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (1.1.0)

Requirement already satisfied: grpcio>=1.8.6 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (1.12.1)

Requirement already satisfied: tensorboard<1.9.0,>=1.8.0 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (1.8.0)

Requirement already satisfied: wheel>=0.26 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorflow==1.8.0) (0.31.1)

Requirement already satisfied: setuptools in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from protobuf>=3.4.0->tensorflow==1.8.0) (39.2.0)

Requirement already satisfied: werkzeug>=0.11.10 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorboard<1.9.0,>=1.8.0->tensorflow==1.8.0) (0.14.1)

Requirement already satisfied: html5lib==0.9999999 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorboard<1.9.0,>=1.8.0->tensorflow==1.8.0) (0.9999999)

Requirement already satisfied: markdown>=2.6.8 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorboard<1.9.0,>=1.8.0->tensorflow==1.8.0) (2.6.11)

Requirement already satisfied: bleach==1.5.0 in /opt/anaconda2/envs/test_py3/lib/python3.6/site-packages (from tensorboard<1.9.0,>=1.8.0->tensorflow==1.8.0) (1.5.0)

代码:

```
(test_py3) tensor@tfvm:~$ cat FirstDetection.py
from imageai.Detection import ObjectDetection
import os
```

```
execution_path = os.getcwd()
```

```
detector = ObjectDetection()
detector.setModelTypeAsRetinaNet()
detector.setModelPath(os.path.join(execution_path,"resnet50_coco_best_v2.0.1.h5"))
detector.loadModel()
detections=detector.detectObjectsFromImage(input_image=os.path.join(execution_path,"image.
jpg"),output_image_path=os.path.join(execution_path,"imagenew.jpg"))
```

```
for eachObject in detections:
    print(eachObject["name"] + " : " + eachObject["percentage_probability"])
```

```
(test_py3) tensor@tfvm:~$ python FirstDetection.py
/opt/anaconda2/envs/test_py3/lib/python3.6/site-packages/h5py/__init__.py:36:
FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is
deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
```

```
from _conv import register_converters as _register_converters
Using TensorFlow backend.
```

```
2018-06-24 17:49:39.152047: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU
supports instructions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX
```

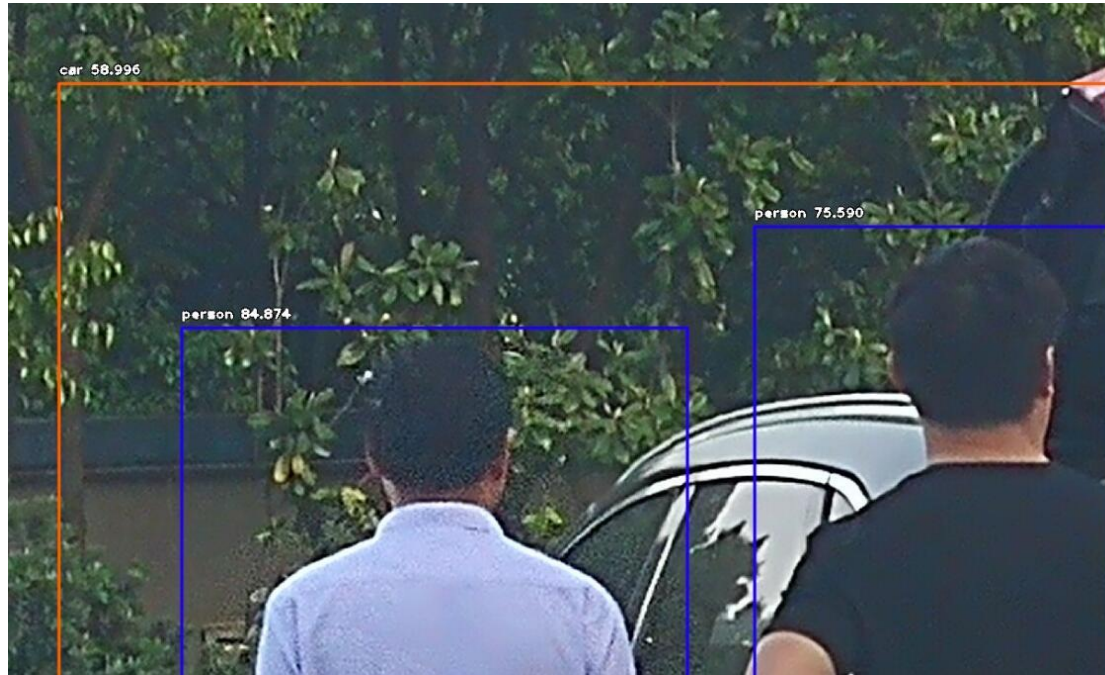
```
car : 80.99603056907654
```

```
car : 63.76085877418518
```

```
handbag : 52.554577589035034
```

person : 84.87399816513062
person : 75.58995485305786
car : 65.04697799682617
car : 71.75977826118469
person : 87.2457504272461
person : 81.74251914024353
car : 58.996182680130005





```
(test_py3) tensor@tfvm:~$ cp 2018spring.jpg image.jpg
```

```
(test_py3) tensor@tfvm:~$ python FirstDetection.py
```

```
/opt/anaconda2/envs/test_py3/lib/python3.6/site-packages/h5py/__init__.py:36:
```

```
FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
```

```
from ._conv import register_converters as _register_converters
```

```
Using TensorFlow backend.
```

```
2018-06-24 17:53:34.431756: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX
```

```
person : 82.31648802757263
```

```
person : 82.685387134552
```

```
person : 82.50718712806702
```

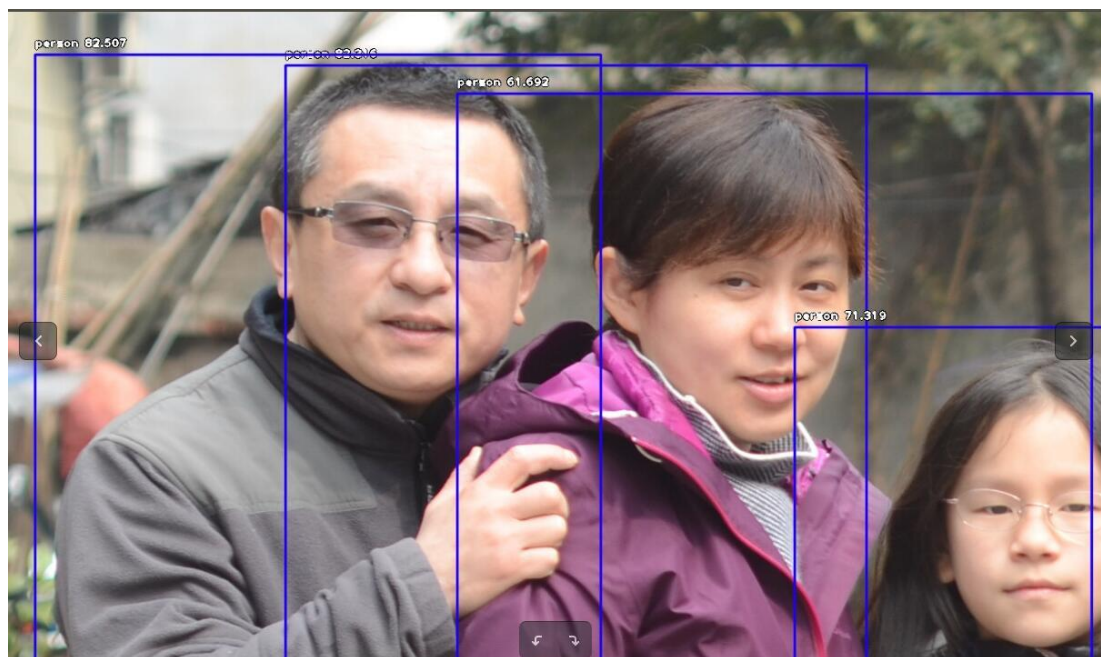
```
person : 94.79280710220337
```

```
person : 96.5410828590393
```

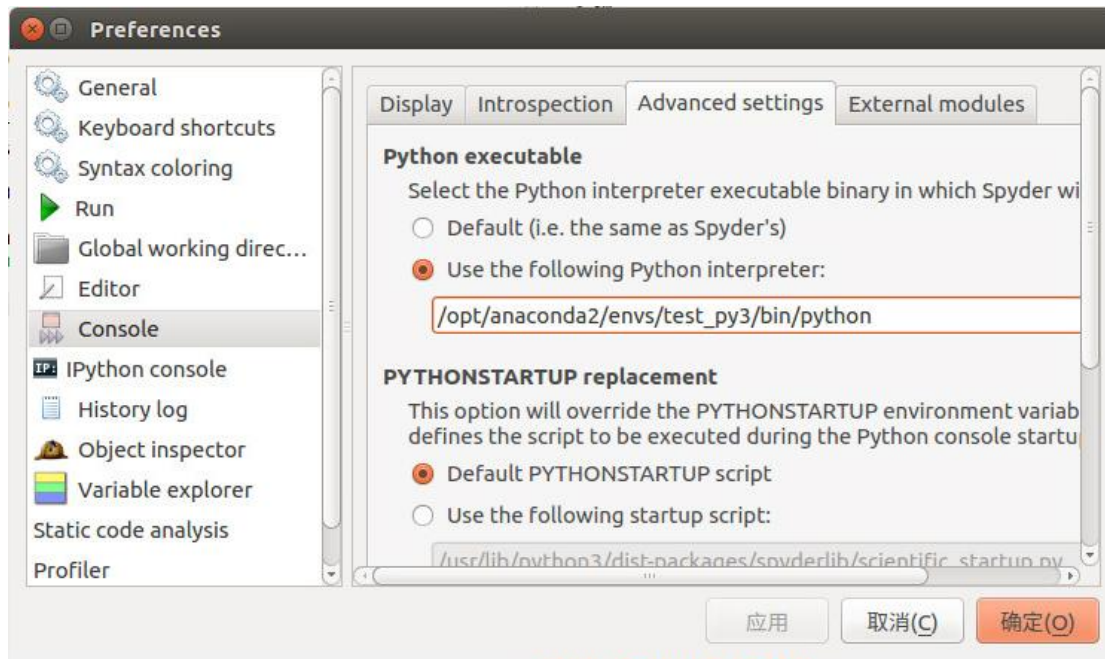
```
person : 71.31900787353516
```

```
person : 96.81103229522705
```

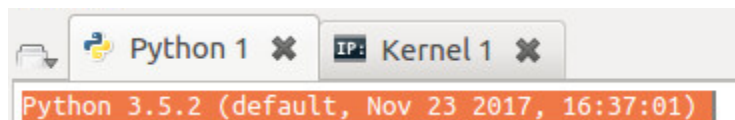
```
person : 61.69199347496033
```

回过头来整 spyder3 的环境，spyder3 在 ubuntu 的 python 3（python 3.5.2）的环境下运行。

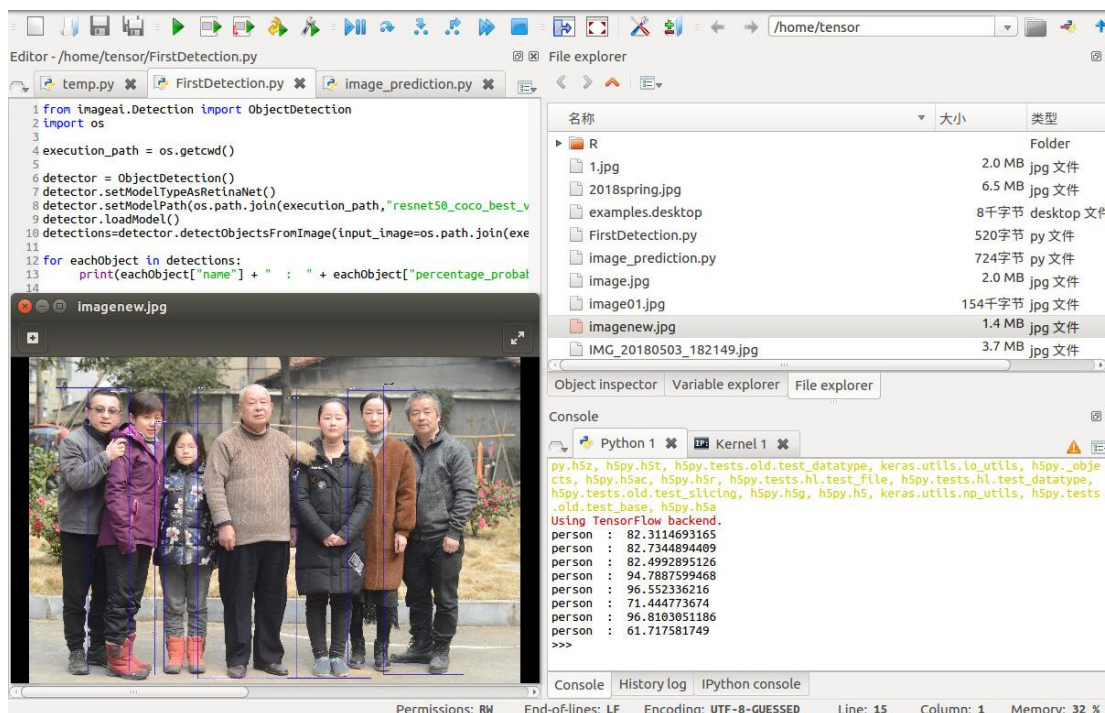


tensor@tfvm:~\$ pip3 -V
pip 8.1.1 from /usr/lib/python3/dist-packages (python 3.5)



使用 pip3 安装一系列依赖:
tensor@tfvm:~\$ pip3 install tensorflow-1.8.0-cp35-cp35m-manylinux1_x86_64.whl

成功运行:



调整 jupyter 环境:

jupyter Untitled44 Last Checkpoint: 16 分钟前 Autosave Failed!

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
In [8]: from imageai.Detection import ObjectDetection
import os
from PIL import Image
import matplotlib.pyplot as plt

execution_path = os.getcwd()

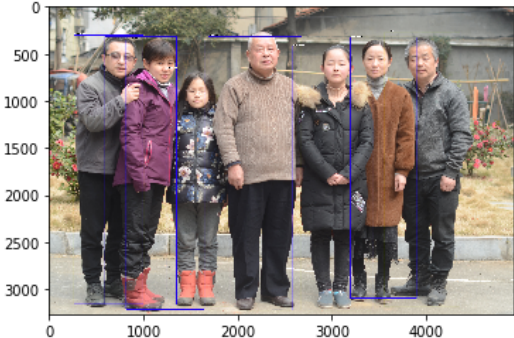
detector = ObjectDetection()
detector.setModelTypeAsRetinaNet()
detector.setModelPath(os.path.join(execution_path, "resnet50_coco_best_v2.0.1.h5"))
detector.loadModel()
detections=detector.detectObjectsFromImage(input_image=os.path.join(execution_path, "image.jpg"),

for eachObject in detections:
    print(eachObject["name"] + " : " + eachObject["percentage_probability"])
img=Image.open(os.path.join(execution_path, "imagenew.jpg"))

plt.figure("检测")
plt.imshow(img)
plt.show()

print(img.size)
print(img.mode)
print(img.format)
```

```
person : 82.31648802757263
person : 82.685387134552
person : 82.50718712806702
person : 94.79280710220337
person : 96.5410828590393
person : 71.31900787353516
person : 96.81103229522705
person : 61.69199347496033
```



(4928, 3264)
RGB
JPEG

```
from imageai.Detection import ObjectDetection
import os
from PIL import Image
import matplotlib.pyplot as plt
```



```
execution_path = os.getcwd()

detector = ObjectDetection()
detector.setModelTypeAsRetinaNet()
detector.setModelPath(os.path.join(execution_path,"resnet50_coco_best_v2.0.1.h5"))
detector.loadModel()
detections=detector.detectObjectsFromImage(input_image=os.path.join(execution_path,"image.
jpg"),output_image_path=os.path.join(execution_path,"imagenew.jpg"))

for eachObject in detections:
    print(eachObject["name"] + " : " + eachObject["percentage_probability"])
img=Image.open(os.path.join(execution_path,"imagenew.jpg"))

plt.figure("检测")
plt.imshow(img)
plt.show()

print(img.size)
print(img.mode)
print(img.format)
```

汽车类型识别



代码:

```
tensor@tfvm:~$ cat image_prediction.py
```

```

from imageai.Prediction import ImagePrediction
import os

execution_path = os.getcwd()
print("NOW path:"+execution_path)

prediction = ImagePrediction()
prediction.setModelTypeAsResNet()
prediction.setModelPath(os.path.join(execution_path,
"resnet50_weights_tf_dim_ordering_tf_kernels.h5"))

#prediction.setModelPath(os.path.join(execution_path, "resnet50_coco_best_v2.0.1.h5"))

print(os.path.join(execution_path, "resnet50_weights_tf_dim_ordering_tf_kernels.h5"))

prediction.loadModel()

predictions, probabilities = prediction.predictImage(os.path.join(execution_path, "1.jpg"),
result_count=5 )
for eachPrediction, eachProbability in zip(predictions, probabilities):
    print(eachPrediction + " : " + eachProbability)

```

```

tensor@tfvm:~$ python3 image_prediction.py
/home/tensor/.local/lib/python3.5/site-packages/h5py/__init__.py:36: FutureWarning:
Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In
future, it will be treated as `np.float64 == np.dtype(float).type`.

```

```

    from ._conv import register_converters as _register_converters
NOW path:/home/tensor
/home/tensor/resnet50_weights_tf_dim_ordering_tf_kernels.h5
convertible : 53.22014093399048
sports_car : 34.547680616378784
pickup : 3.907627612352371
minivan : 2.6654161512851715
car_wheel : 1.8447380512952805

```

```

convertible : 折篷车；敞篷车
sports_car : 运动轿车
pickup : 皮卡、轻型货车
minivan : （尤指载客的）小型面包车
car_wheel : 汽车轮子？

```

实时视频人脸识别

<http://www.cnblogs.com/neo-T/p/>

模型训练

<https://www.cnblogs.com/qcloud1001/p/7677661.html>

云加社区：

<http://www.cnblogs.com/qcloud1001/tag/%E4%BA%BA%E5%B7%A5%E6%99%BA%E8%83%BD/>

数据基础

数据分析

概率论

线性代数及矩阵

凸优化

数据挖掘

C4.5

K-Means

算法简介

k-means 算法是一种聚类算法，所谓聚类，即根据相似性原则，将具有较高相似度的数据对象划分至同一类簇，将具有较高相异度的数据对象划分至不同类簇。聚类与分类最大的区别在于，聚类过程为无监督过程，即待处理数据对象没有任何先验知识，而分类过程为有监督过程，即存在有先验知识的训练数据集。

k-means 算法中的 **k** 代表类簇个数，**means** 代表类簇内数据对象的均值（这种均值是一种对类簇中心的描述），因此，**k-means** 算法又称为 **k-均值** 算法。**k-means** 算法是一种基于划分的聚类算法，以距离作为数据对象间相似性度量的标准，即数据对象间的距离越小，则它们的相似性越高，则它们越有可能在同一个类簇。数据对象间距离的计算有很多种，**k-means** 算法通常采用欧氏距离来计算数据对象间的距离。

k-means 算法优缺点分析

- 优点:
 - 算法简单易实现;
- 缺点:
 - 需要用户事先指定类簇个数 K ;
 - 聚类结果对初始类簇中心的选取较为敏感;
 - 容易陷入局部最优;
 - 只能发现球型类簇;

k-means 算法改进方法

初始类簇中心的选取, 可以通过 k-means++ 算法进行改进。

代码实现 (jupyter anaconda3 python 3 tensorflow 1.8):

```
# -*- coding: utf-8 -*-
import numpy as np
from numpy.linalg import cholesky
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import tensorflow as tf
from random import choice, shuffle
from numpy import array
#####Sachin Joglekar 的基于 tensorflow 写的一个 kmeans 模板#####
def KMeansCluster(vectors, noofclusters):
    """
    K-Means Clustering using TensorFlow.
    `vectors` 应该是一个  $n \times k$  的二维的 NumPy 的数组, 其中  $n$  代表着  $K$  维向量的数目
    'noofclusters' 代表了待分的集群的数目, 是一个整型值
    """

    noofclusters = int(noofclusters)
    assert noofclusters < len(vectors)
    #找出每个向量的维度
    dim = len(vectors[0])
    #辅助随机地从可得的向量中选取中心点
    vector_indices = list(range(len(vectors)))
    shuffle(vector_indices)
    #计算图
    #我们创建了一个默认的计算流的图用于整个算法中, 这样就保证了当函数被多次调用
    #时, 默认的图并不会被从上一次调用时留下的未使用的 OPS 或者 Variables 挤满
    graph = tf.Graph()
    with graph.as_default():
        #计算的会话
```

```

sess = tf.Session()
##构建基本的计算的元素
##首先我们需要保证每个中心点都会存在一个 Variable 矩阵
##从现有的点集合中抽取出一部分作为默认的中心点
centroids = [tf.Variable((vectors[vector_indices[i]]))
               for i in range(noofclusters)]
##创建一个 placeholder 用于存放各个中心点可能的分类的情况
centroid_value = tf.placeholder("float64", [dim])
cent_assigns = []
for centroid in centroids:
    cent_assigns.append(tf.assign(centroid, centroid_value))
##对于每个独立向量的分属的类别设置为默认值 0
assignments = [tf.Variable(0) for i in range(len(vectors))]
##这些节点在后续的操作中会被分配到合适的值
assignment_value = tf.placeholder("int32")
cluster_assigns = []
for assignment in assignments:
    cluster_assigns.append(tf.assign(assignment,
                                     assignment_value))

##下面创建用于计算平均值的操作节点
#输入的 placeholder
mean_input = tf.placeholder("float", [None, dim])
#节点/OP 接受输入，并且计算 0 维度的平均值，譬如输入的向量列表
mean_op = tf.reduce_mean(mean_input, 0)
##用于计算欧几里得距离的节点
v1 = tf.placeholder("float", [dim])
v2 = tf.placeholder("float", [dim])
euclid_dist = tf.sqrt(tf.reduce_sum(tf.pow(tf.subtract(v1, v2), 2)))
##这个 OP 会决定应该将向量归属到哪个节点
##基于向量到中心点的欧几里得距离
#Placeholder for input
centroid_distances = tf.placeholder("float", [noofclusters])
cluster_assignment = tf.argmin(centroid_distances, 0)
##初始化所有的状态值
##这会帮助初始化图中定义的所有 Variables。Variable-initializer 应该定
##义在所有的 Variables 被构造之后，这样所有的 Variables 才会被纳入初始化
init_op = tf.global_variables_initializer()
#初始化所有的变量
sess.run(init_op)
##集群遍历
#接下来在 K-Means 聚类迭代中使用最大期望算法。为了简单起见，只让它执行固
#定的次数，而不设置一个终止条件
noofiterations = 20
for iteration_n in range(noofiterations):

```



```

##期望步骤
##基于上次迭代后算出的中心点的未知
##the _expected_ centroid assignments.
#首先遍历所有的向量
for vector_n in range(len(vectors)):
    vect = vectors[vector_n]
    #计算给定向量与分配的中心节点之间的欧几里得距离
    distances = [sess.run(euclid_dist, feed_dict={
        v1: vect, v2: sess.run(centroid)})
        for centroid in centroids]
    #下面可以使用集群分配操作，将上述的距离当做输入
    assignment = sess.run(cluster_assignment, feed_dict = {
        centroid_distances: distances})
    #接下来为每个向量分配合适的值
    sess.run(cluster_assigns[vector_n], feed_dict={
        assignment_value: assignment})

##最大化的步骤
#基于上述的期望步骤，计算每个新的中心点的距离从而使集群内的平方和最
小

for cluster_n in range(noofclusters):
    #收集所有分配给该集群的向量
    assigned_vects = [vectors[i] for i in range(len(vectors))
        if sess.run(assignments[i]) == cluster_n]
    #计算新的集群中心点
    new_location = sess.run(mean_op, feed_dict={
        mean_input: array(assigned_vects)})
    #为每个向量分配合适的中心点
    sess.run(cent_assigns[cluster_n], feed_dict={
        centroid_value: new_location})

#返回中心节点和分组
centroids = sess.run(centroids)
assignments = sess.run(assignments)
return centroids, assignments

#####生成测试数据#####
sampleNo = 10;#数据数量
mu =3
# 二维正态分布
mu = np.array([[1, 5]])
Sigma = np.array([[1, 0.5], [1.5, 3]])
R = cholesky(Sigma)
srcdata= np.dot(np.random.randn(sampleNo, 2), R) + mu

```

```

plt.plot(srcdata[:,0],srcdata[:,1],'bo')
#####kmeans 算法计算#####
k=4
center,result=KMeansCluster(srcdata,k)
print(center)
#####利用 seaborn 画图#####

res={"x":[],"y":[],"kmeans_res":[]}
for i in range(len(result)):
    res["x"].append(srcdata[i][0])
    res["y"].append(srcdata[i][1])
    res["kmeans_res"].append(result[i])
pd_res=pd.DataFrame(res)
sns.lmplot("x","y",data=pd_res,fit_reg=False,size=5,hue="kmeans_res")
plt.show()

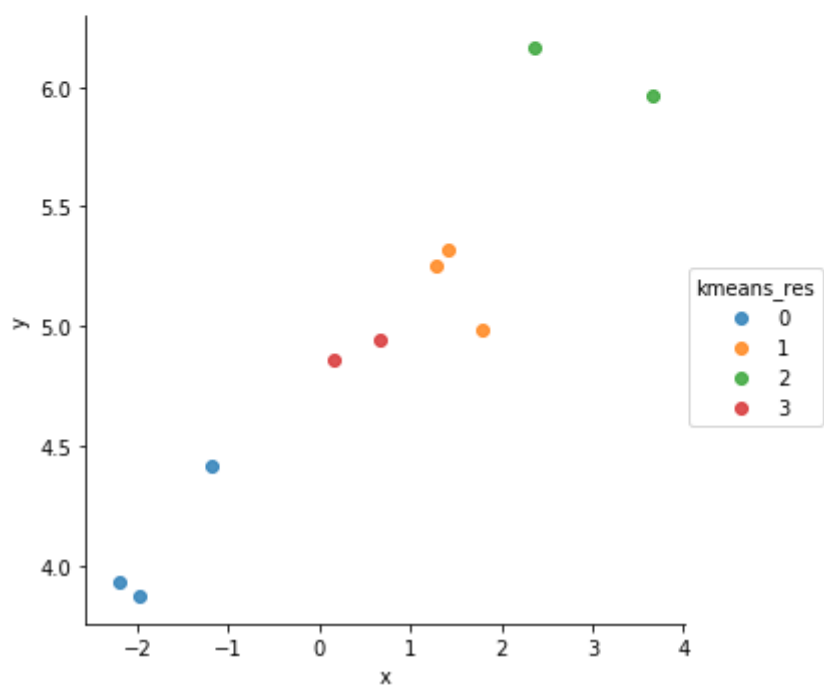
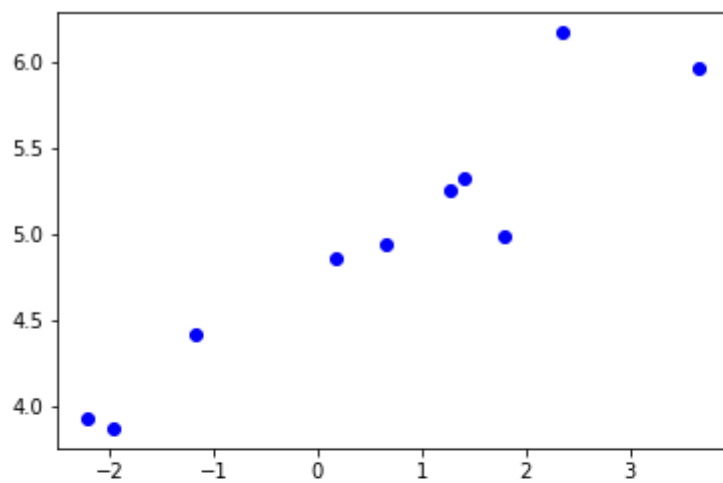
```

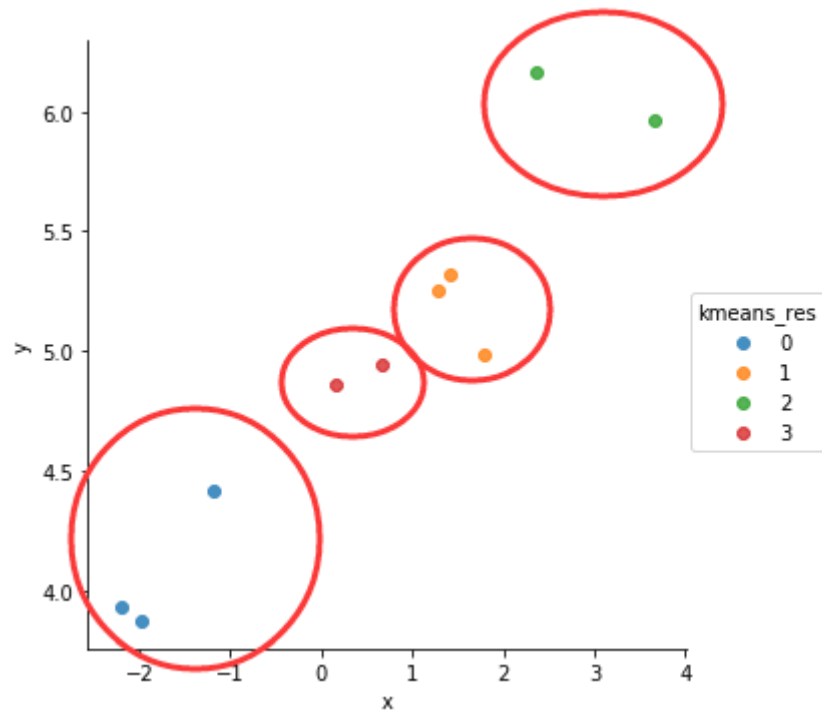
运行结果：

```

[array([-1.78417623,  4.07541895]), array([1.4949894,  5.1860218]), array([3.00592971,
6.06591892]), array([0.4123759 , 4.90223598])]

```





SVM

Apriori

EM

PageRank

AdaBoost

kNN

Naïve Bayes

CART

中级应用

JAVA 开发

扩展

深度学习画图神器(TikZ)

<https://blog.csdn.net/zhaoyu106/article/details/53763293>

1、TikZ 的官网：内含很多示例代码

<http://www.texample.net/tikz/>

3、LateX 在线编辑工具

4、

<https://www.overleaf.com>

3、TikZ 快速入门文档

<http://cremeronline.com/LaTeX/minimaltikz.pdf>

TeXstudio

Geogebra

在 MikTeX 的官网下载免费的 MikTeX 编译包（150Mb）并安装。或在 ctex.org 下载 ctex 套装（203Mb 或 1.3Gb）（含 MikTeX 及 WinEdt）

监督学习

非监督学习

数据处理与模型调优

NLP

语音识别

图像识别

参考

官网:

<https://www.tensorflow.org/>

【好像访问不了】

Tensorflow 官方文档中文版（极客学院）:

<http://wiki.jikexueyuan.com/project/tensorflow-zh/>

tensorflow 教程——莫烦 Python

<https://morvanzhou.github.io/tutorials/machine-learning/tensorflow/>

英文教程:

<http://learningtensorflow.com/>

工具——云算子:

http://www.yunsuanzi.com/cgi-bin/matrix_multiplication.py

清华大学镜像源

<https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/>

<https://mirrors.tuna.tsinghua.edu.cn>

镜像源汇总:

<http://www.mamicode.com/info-detail-2279504.html>

tensorflow 1.8 下载地址:

<https://pypi.org/project/tensorflow/1.8.0/#files>

TF 中文社区——完整教程

<http://www.tensorfly.cn/>

<http://www.tensorfly.cn/tfdoc/tutorials/overview.html>

AI 之路

<https://blog.csdn.net/u014380165/article/category/6829229>

干货！从基础到进阶，长文解析微软量子计算概念和算法（上）

<https://www.leiphone.com/news/201806/MJTy0Bt3VlBaIQi.html>

雷锋网：

<https://www.leiphone.>

手把手教你如何用 TensorFlow 实现基于 DNN 的文本分类

<https://www.leiphone.com/news/201704/HE5REPagOVlzEEI0.html>

不是你无法入门自然语言处理（NLP），而是你没找到正确的打开方式

<https://blog.csdn.net/amds123/article/details/72860143>

北风 AI 工程师课程体系：

http://www.ibeifeng.com/job_AIpage.html?f_kwid=1413964267&v_kwid=69261677679&v_adid=18129782104&hmsr=com-bdss&hmpl=p3181-ai&hmcu=mk005-ai.tensorflow&hmkw=69261677679_tensorflow%c5%e0%d1%b5&hmci=18129782104&matchtype=1&adposition=cl1&pagenum=1#python-prg

Tensorflow 快速入门 1--实现 K-Means 聚类算法

<https://blog.csdn.net/yhhyhhyhhyh/article/details/54429034>

我读 ResNet

https://blog.csdn.net/xuanwu_yan/article/details/52495471

欲想了解更多信息，请扫码入群：



Knowage
241837271

首页 成员 设置

编辑资料



群介绍 本群创建于2018/3/15：报
表、JASPER、Birt、R、Python、OLAP、数据挖掘
(DATA MINING)、机器学习 (Machine
Learning)、GREENPLUM、ETL、ESB、大数据、云
平台、go、kubernate

群标签 [行业交流](#) [IT/互联网](#)



Knowage

扫一扫二维码，加入群聊。

如果您觉得受益想请鄙人吃个葱油饼，不妨打赏：



推荐使用微信支付

