**深度学习主机环境配置: Ubuntu16.04 + GeForce GTX 1070 + CUDA8.0 + cuDNN5.1 + TensorFlow**

最近在公司做深度学习相关的学习和实验，原来一直在自己的电脑上安装虚拟机跑，速度实在太慢，主机本身性能太弱，独显都没有，物理安装Ubuntu也没多大意义，所以考虑用公司性能最强悍的游戏主机(i7 6700+GTX 1070) 做实验。

原本以为整个一套安装下来会很顺利，一路火花带闪电的就完成了，没想到刚开始就掉坑了，然而这仅仅是悲剧的开始，路上坑还很多，刚从这个坑里爬出来，没走两步又掉另一个坑里了，下面慢慢絮叨一下爬坑过程。

**一. 安装Ubuntu16.04**

**1.1 下载Ubuntu ISO 镜像**

从[Ubuntu官网](https://www.ubuntu.com/download/alternative-downloads)下载ISO镜像，选择Ubuntu 16.04.1 Desktop (64-bit) ，LTS表示这是一个长期支持版（Long Term Support），一定要选64位的版本，因为很多深度学习框架都只支持64位。

**1.2 制作U盘**

找一个空的U盘，把刚下载到的ISO镜像写入到U盘中。注意，这里并不是把ISO文件解压到U盘中，而是使用工具把ISO文件刻录到U盘中，区别是后者可以引导系统启动，前者不能。

我使用的是UltraISO软碟通，使用方法网上教程很多，这里不再赘述。完成后就得到了一个可以引导启动的Ubuntu USB安装盘。

**1.3 硬盘分区**

主机原来的Window 10系统还是需要保留的，毕竟不能妨碍了小伙伴们平时最大的兴趣爱好。所以考虑安装双系统，在硬盘上找了一个比较空的分区，把资料复制到其他盘后，删除这个分区，空闲出来的空间留给Ubuntu用。

双系统安装时可能会遇到分区个数的问题，MBR类型的分区最多只能有4个主分区，而Ubuntu安装会占用两个主分区（一个挂载swap，一个挂载根），所以要保证原有主分区不超过2个，GPT类型分区的主分区没有个数限制。

**1.4 安装系统**

设置BIOS从U盘启动后安装Ubuntu，现在的系统安装都很傻瓜，看提示很快安装完成。

注意：系统安装完成，登录用户后只能看到桌面壁纸，右键菜单不停闪烁，且tty1-6均黑屏，接着看下面如何解决。

**二. 安装GTX1070驱动**

第一次安装驱动是在NVIDIA官网下载的，地址在[这里](http://www.nvidia.com/Download/index.aspx)，但是安装后第一次安装显卡驱动后登录，只能看到桌面壁纸。

其实有更简单的安装方法：

$ sudo add-apt-repository ppa:xorg-edgers/ppa -y

$ sudo apt-get update

# install the latest version

$ sudo apt-get install nvidia-current

* 1
* 2
* 3
* 4

查了一下说登录后只能看到桌面壁纸这种现象出现的原因：是由于显卡驱动没有合适安装、设定、启用造成的。

解决办法：

* 重启电脑，在开机启动项（GRUB菜单选择处）画面按e进入启动项编辑模式，在Linux那一行最后面空一格然后加入“acpi\_osi=linux nomodeset”，然后按F10使用编辑好的启动项启动（nomodest参数是告诉内核不要加载显卡驱动而用BIOS模式，直到图形界面运行，详细介绍看[这里](https://www.douban.com/note/343152353/)）。
* 再次开机后发现还是在不停的闪，但是这次tty1-6可以用了
* 按Ctrl+Alt+F1切换到控制台界面，输入用户名密码登录
* 安装NVIDIA显卡驱动，步骤：

sudo add-apt-repository ppa:graphics-drivers/ppa

sudo apt-get update

sudo apt-get install nvidia-367

sudo apt-get install mesa-common-dev

sudo apt-get install freeglut3-dev

* 1
* 2
* 3
* 4
* 5
* 之后重启主机

登录后桌面不闪了但仍然只有桌面和鼠标，这时候需要重新安装一下Unity桌面，步骤：

sudo apt-get update

sudo apt-get install --reinstall ubuntu-desktop

sudo apt-get install unity

* 1
* 2
* 3

安装完成后重启，一切正常。

**三. 下载和安装CUDA**

这里是CUDA 8的下载地址：<https://developer.nvidia.com/cuda-release-candidate-download>。我选择了Ubuntu16.04 runfile安装类型，大小在1.4G左右。

下载完成后，直接运行这个文件：

sudo sh cuda\_8.0.27\_linux.run --tmpdir=/opt/temp/

* 1

运行这个文件后，会遇到几个问题，让你选择yes或者no。其中一个是询问是否安装NVIDIA驱动程序的，这个要选择no，不然前面安装的驱动就白装了。

安装完成后，家目录下会生成一个NVIDIA\_CUDA-8.0\_Samples的文件夹，这里可以测试一下CUDA 8。

cd ~/NVIDIA\_CUDA-8.0\_Samples/1\_Utilities/deviceQuery

make && ./deviceQuery

./deviceQuery Starting...

CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "GeForce GTX 1070"

CUDA Driver Version / Runtime Version 8.0 / 8.0

CUDA Capability Major/Minor version number: 6.1

Total amount of global memory: 8105 MBytes (8499167232 bytes)

(15) Multiprocessors, (128) CUDA Cores/MP: 1920 CUDA Cores

GPU Max Clock rate: 1759 MHz (1.76 GHz)

Memory Clock rate: 4004 Mhz

Memory Bus Width: 256-bit

L2 Cache Size: 2097152 bytes

Maximum Texture Dimension Size (x,y,z) 1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)

Maximum Layered 1D Texture Size, (num) layers 1D=(32768), 2048 layers

Maximum Layered 2D Texture Size, (num) layers 2D=(32768, 32768), 2048 layers

Total amount of constant memory: 65536 bytes

Total amount of shared memory per block: 49152 bytes

Total number of registers available per block: 65536

Warp size: 32

Maximum number of threads per multiprocessor: 2048

Maximum number of threads per block: 1024

Max dimension size of a thread block (x,y,z): (1024, 1024, 64)

Max dimension size of a grid size (x,y,z): (2147483647, 65535, 65535)

Maximum memory pitch: 2147483647 bytes

Texture alignment: 512 bytes

Concurrent copy and kernel execution: Yes with 2 copy engine(s)

Run time limit on kernels: Yes

Integrated GPU sharing Host Memory: No

Support host page-locked memory mapping: Yes

Alignment requirement for Surfaces: Yes

Device has ECC support: Disabled

Device supports Unified Addressing (UVA): Yes

Device PCI Domain ID / Bus ID / location ID: 0 / 1 / 0

Compute Mode:

< Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 8.0, CUDA Runtime Version = 8.0, NumDevs = 1, Device0 = GeForce GTX 1070

Result = PASS

**四. 下载和安装cuDNN**

cuDNN是The NVIDIA CUDA® Deep Neural Network library的简称，它是一个使用GPU加速的深度神经网络库。下载地址：[点这里](https://developer.nvidia.com/cudnn)，需要登录一个Nvidia开发者账号，简单注册一个就行，我选择了cuDNN v5.1。

安装cuDNN非常简单：

tar -zxvf cudnn-8.0-linux-x64-v5.0-ga.tgz

* 1

得到一个cuda文件夹，把其中的文件拷贝到系统相应目录

sudo cp -a cuda/include/cudnn.h /usr/local/cuda/include/

sudo cp -a cuda/lib64/libcudnn\* /usr/local/cuda/lib64/

* 1
* 2

**五. 安装TensorFlow**

TensorFlow官方给出的文档中详细的介绍了各种安装方式：<https://github.com/jikexueyuanwiki/tensorflow-zh/blob/master/SOURCE/get_started/os_setup.md> 。   
最方便、快速的方式当然是直接安装已经编译好的包，其中也提供了开启GPU加速的版本，只不过只能在CUDA 7.5 和cuDNN v4环境下使用。既然环境不符合，不能省事儿，那就只好从源码安装。

**5.1 安装Bazel**

Bazel是一个项目构建工具，类似于Make的工具。Bazel原本是Google为其内部软件开发的特点量身定制的工具，想来TensorFlow原本就是Google内部的一个项目，使用Bazel再正常不过。

首先，安装一下后面会用到的几个依赖包

sudo apt-get update

sudo apt-get install python-pip python-numpy swig python-dev python-wheel

sudo apt-get install pkg-config zip g++ zlib1g-dev unzip

* 1
* 2
* 3

安装 Java JDK

sudo apt-get install default-jdk

* 1

在<https://github.com/bazelbuild/bazel/releases>中下载适合的最新稳定版。   
然后运行下面的命令

./bazel-0.3.2-installer-linux-x86\_64.sh --user

* 1

将执行路径output/bazel 添加到$PATH环境变量后bazel工具就可以使用了。

**5.2 编译TensorFlow**

克隆TensorFlow仓库

git clone --recurse-submodules https://github.com/tensorflow/tensorflow

* 1

配置TensorFlow的CUDA选项

$ ./configure

Do you wish to bulid TensorFlow with GPU support? [y/n] y

GPU support will be enabled for TensorFlow

* 1
* 2
* 3

configure会下载很多依赖的包，这些包也不是太大，但是网络不给力，几十K的包下载时经常中断，后果就是需要再次运行configure，从头下载，每次不是因为这个包断了就是那个包，最终试了可能有十几遍吧才全部成功，强烈谴责这种不做缓存的行为。

编译目标程序，开启GPU支持   
从源码树的根路径执行:

$ bazel build -c opt --config=cuda //tensorflow/cc:tutorials\_example\_trainer

$ bazel-bin/tensorflow/cc/tutorials\_example\_trainer --use\_gpu

# 大量的输出信息. 这个例子用 GPU 迭代计算一个 2x2 矩阵的主特征值 (major eigenvalue).

# 最后几行输出和下面的信息类似.

000009/000005 lambda = 2.000000 x = [0.894427 -0.447214] y = [1.788854 -0.894427]

000006/000001 lambda = 2.000000 x = [0.894427 -0.447214] y = [1.788854 -0.894427]

000009/000009 lambda = 2.000000 x = [0.894427 -0.447214] y = [1.788854 -0.894427]

注意, GPU 支持需通过编译选项 “–config=cuda” 开启.

创建 pip 包并安装

$ bazel build -c opt //tensorflow/tools/pip\_package:build\_pip\_package

* 1

这条命令运行时出现一个错误

~/download/tensorflow$ bazel build -c opt --config=cuda //tensorflow/tools/pip\_package:build\_pip\_package

ERROR: /home/fct/download/tensorflow/tensorflow/python/BUILD:1777:1: in cc\_library rule //tensorflow/python:tf\_session\_helper: non-test target '//tensorflow/python:tf\_session\_helper' depends on testonly target '//tensorflow/python:construction\_fails\_op' and doesn't have testonly attribute set.

ERROR: Analysis of target '//tensorflow/tools/pip\_package:build\_pip\_package' failed; build aborted.

INFO: Elapsed time: 0.134s

在tensorflow/python/BUILD 中找到tf\_cuda\_library，注释掉testonly = 1, 这一行解决

$ bazel-bin/tensorflow/tools/pip\_package/build\_pip\_package /tmp/tensorflow\_pkg

# .whl 文件的实际名字与你所使用的平台有关

$ pip install /tmp/tensorflow\_pkg/tensorflow-0.5.0-cp27-none-linux\_x86\_64.whl

**测试TensorFlow**

训练一个神经网络模型用来做测试，从源代码树根路径执行：

$ cd tensorflow/models/image/mnist

$ python convolutional.py

Succesfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.

Succesfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.

Succesfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.

Succesfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.

Extracting data/train-images-idx3-ubyte.gz

Extracting data/train-labels-idx1-ubyte.gz

Extracting data/t10k-images-idx3-ubyte.gz

Extracting data/t10k-labels-idx1-ubyte.gz

Initialized!

Epoch 0.00

Minibatch loss: 12.054, learning rate: 0.010000

Minibatch error: 90.6%

Validation error: 84.6%

Epoch 0.12

Minibatch loss: 3.285, learning rate: 0.010000

Minibatch error: 6.2%

Validation error: 7.0%

...

...

**常见问题**

如果在尝试运行一个TensorFlow程序时出现一下错误：

ImportError: libcudart.so.8.0: cannot open shared object file: No such file or directory

* 1

这个错是程序所依赖的动态库找不到导致的，只需要在环境变量LD\_LIBRARY\_PATH中添加   
/usr/local/cuda/lib64即可。我的做法是在~/.bashrc最后添加

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/usr/local/cuda/lib64

* 1