

## 1. 安装 NVIDIA 显卡驱动 (\*)

### 首先禁用 nouveau 驱动

查看 nouveau 信息

```
lsmod | grep nouveau
```

或者

```
lspci | grep nouveau
```

打开终端，先删除旧的驱动

```
sudo apt-get --purge nvidia*
```

在命令行输入：

```
sudo gedit /etc/modprobe.d/blacklist-nouveau.conf
```

在文件最后添加：

```
blacklist nouveau
```

```
options nouveau modeset=0
```

保存并退出

在命令行执行

```
sudo update-initramfs -u
```

```
lsmod | grep nouveau 或者 lspci | grep nouveau
```

若什么都没有显示，说明禁用成功。

重启

```
sudo reboot
```

重启之后按 `ctrl+alt+F1` 进入控制台，登陆后 `sudo /etc/init.d/lightdm`

`stop` 关闭图形界面

### 安装 NVIDIA 显卡驱动

驱动文件下载：<https://www.geforce.cn/drivers>

注意：为方便起见，先将 `.run` 文件放到 `home` 目录下：

`CTRL+ALT+F1` 进入 `ttf1` 控制台：

```
sudo service lightdm stop
```

```
sudo sh ./NVIDIA*-390.87.run -no-opengl-files #根据实际情况更
```

改 `.run` 文件版本

-----  
在 NVIDIA 驱动安装过程中，选项参照：

1) 选择 **Accept**

2) The distribution-provided pre-install script failed ... ..

---选择 **Continue installation**

3) Would you like to run the **nvidia-xconfig** utility to automatically update your X Configuration file so set the NVIDIA X driver will be used when you restart X? -----选择 **no**

4) Install 32-Bit compatibility libraries? ----选择 **no**  
-----

-  
最后重启图形环境：

```
sudo service lightdm start
```

打开终端，查看显卡信息：

```
nvidia-smi # 验证显卡信息说明驱动安装成功。
```

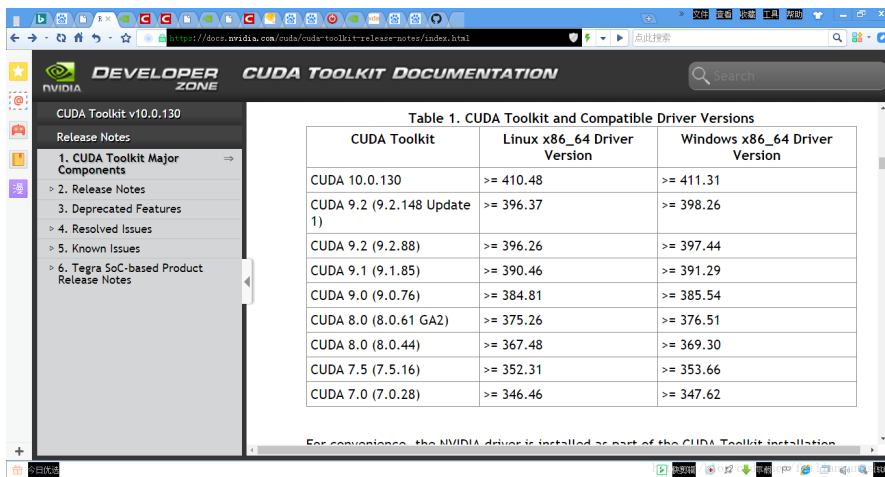
重启：

```
sudo reboot
```

## 2. 安装 CUDA (\*)

Cuda 下载地址：<https://developer.nvidia.com/cuda-toolkit-archive>

不同版本的 CUDA 对应不同显卡驱动的版本，在 CUDA 的官网上有详细的要求，摘录如下：



打开终端，cd 到 .run 文件目录：

```
sudo sh cuda_8.0.44_linux.run
```

选择如下所示

Do you accept the previously read EULA?

accept/decline/quit: **accept**

Install NVIDIA Accelerated Graphics Driver for Linux-x86\_64 384.81?

(y)es/(n)o/(q)uit: **n**

Install the CUDA 9.0 Toolkit?

(y)es/(n)o/(q)uit: **y**

Enter Toolkit Location

[ default is /usr/local/cuda-9.0 ]:

Do you want to install a symbolic link at /usr/local/cuda?

(y)es/(n)o/(q)uit: **y**

Install the CUDA 9.0 Samples?

(y)es/(n)o/(q)uit: **y**

在安装结束后可能会出现 missing recommended library 的情况

```
sudo apt-get install freeglut3-dev build-essential libx11-dev libxmu-dev  
libxi-dev libgl1-mesa-glx libglu1-mesa libglu1-mesa-dev
```

敲入上述命令就能解决了

配置系统环境变量：

```
sudo gedit ~/.bashrc
```

文末位置加入

```
# CUDA
```

```
export PATH=/usr/local/cuda-*/bin:$PATH
```

```
export LD_LIBRARY_PATH=/usr/local/cuda-
```

```
*/lib64:$LD_LIBRARY_PATH
```

使生效：

```
source ~/.bashrc
```

终端输入：

```
nvcc -V,
```

提示 CUDA Toolkit 版本信息表示安装成功。

安装结束后测试：

```
cd /usr/local/cuda-8.0/samples/1_Utilities/deviceQuery
```

```
sudo make
sudo ./deviceQuery
输出结果 Pass 表示通过测试
```

### 3. 安装 cudnn (\*)

Cudnn 下载地址: <https://developer.nvidia.com/rdp/cudnn-archive>  
按照官方文档的方式安装:

<https://docs.nvidia.com/deeplearning/sdk/cudnn-install/index.html#installlinux>

下载完 cudnn 后, 命令行输入文件所在的文件夹, 解压

```
tar -xzf cudnn-9.0-linux-x64-v7.tgz
```

解压后一级目录是 cuda, 里面包含两个文件夹: include 和 lib64

复制头文件

```
sudo cp cuda/include/cudnn.h /usr/local/cuda/include
```

复制动态库

```
sudo cp cuda/lib64/libcudnn* /usr/local/cuda/lib64
```

改变 mode

```
sudo chmod a+r /usr/local/cuda/include/cudnn.h
/usr/local/cuda/lib64/libcudnn*
```

建立连接:

```
cd /usr/local/cuda/lib64/
```

```
sudo rm -rf libcudnn.so libcudnn.so.5 #删除原有动态文件
```

```
sudo ln -s libcudnn.so.5.1.10 libcudnn.so.5 #生成软链接
```

```
sudo ln -s libcudnn.so.5 libcudnn.so #生成软链接
```

```
sudo ldconfig
```

若不成功, 可以按照如下方式添加环境变量:

```
cd ~
```

```
sudo gedit /etc/profile
```

添加: export

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64
```

```
source /etc/profile
```

### 4. 安装 opencv3.1 (\*)

参考及问题解决 [https://blog.csdn.net/weixin\\_42287851/article/details/80419646](https://blog.csdn.net/weixin_42287851/article/details/80419646)

为方便起见, 将 opencv-3.1.0.zip 文件拷贝到 **home** 文件夹下

```
unzip opencv-3.1.0.zip
```

命令行进入已解压的文件夹 opencv-3.1.0 目录下

```
cd /opencv-3.1.0
```

创建编译文件夹:

```
mkdir build
```

```
cd build
```

配置:

```
sudo apt install cmake
```

```
cmake -D CMAKE_BUILD_TYPE=Release -D CMAKE_INSTALL_PREFIX=/usr/local -D
```

```
OPENCV_EXTRA_MODULES_PATH=<path to opencv_contrib/modules/> ..
```

编译:

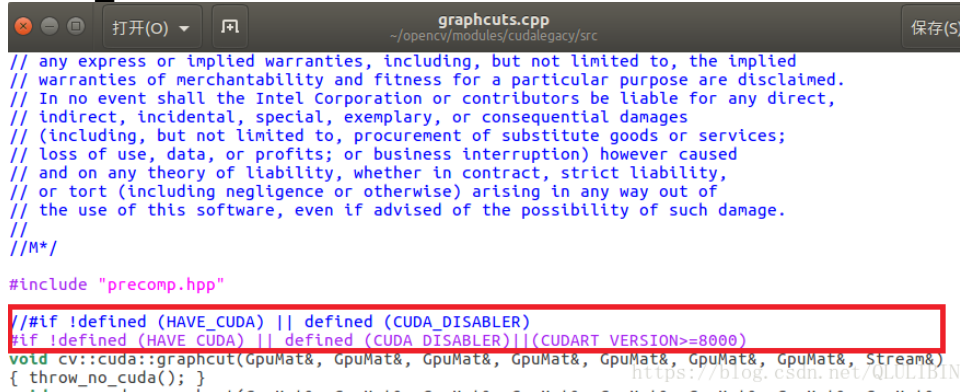
```
sudo make all-j8
```

在执行 `sudo make -j8` 命令编译到 93% 时可能会出现以下错误

这是因为 `opencv3.0` 与 `cuda8.0` 不兼容导致的。解决办法：修改

~/opencv/modules/cudalegacy/src/graphcuts.cpp 文件内容，如图所示：

其中，`#if !defined (HAVE_CUDA) || defined (CUDA_DISABLE) || (CUDA_VERSION >= 8000)` 是我们修改的



```
// any express or implied warranties, including, but not limited to, the implied
// warranties of merchantability and fitness for a particular purpose are disclaimed.
// In no event shall the Intel Corporation or contributors be liable for any direct,
// indirect, incidental, special, exemplary, or consequential damages
// (including, but not limited to, procurement of substitute goods or services;
// loss of use, data, or profits; or business interruption) however caused
// and on any theory of liability, whether in contract, strict liability,
// or tort (including negligence or otherwise) arising in any way out of
// the use of this software, even if advised of the possibility of such damage.
//
//M*/

#include "precomp.hpp"

#if !defined (HAVE_CUDA) || defined (CUDA_DISABLE) || (CUDA_VERSION >= 8000)
if !defined (HAVE_CUDA) || defined (CUDA_DISABLE) || (CUDA_VERSION >= 8000)
void cv::cuda::GraphCut(GpuMat&, GpuMat&, GpuMat&, GpuMat&, GpuMat&, GpuMat&, GpuMat&, Stream&)
{ throw_no_cuda(); }
```

安装

```
sudo make install
```

安装完成后通过查看 `opencv` 版本验证是否安装成功：

```
pkg-config --modversion opencv
```

检查是否安装成功

```
cd ~
```

```
python
```

```
import cv2      #没有报错，说明成功。
```

## 5. 安装 caffe (\*)

官网安装指南：<http://caffe.berkeleyvision.org/install apt.html>

安装依赖项：

```
sudo apt-get install libprotobuf-dev libleveldb-dev libsnappy-
dev libopencv-dev libhdf5-serial-dev protobuf-compiler
```

```
sudo apt-get install --no-install-recommends libboost-all-dev
```

```
sudo apt-get install libgflags-dev libgoogle-glog-dev liblmdb-
dev
```

BLAS: caffe 默认使用 atlas

```
sudo apt-get install libatlas-base-dev
```

或选择 `openblas`

```
sudo apt-get install libopenblas-dev
```

**Python** (可选)：如果需要编译 `pycaffe` 接口则需要先准备 `python`

将 `caffe` 安装包放到 `home` 目录下：

```
cd ~/caffe/python
```

```
sudo apt-get install python-pip python-dev build-essential
```

```
sudo pip install --upgrade pip
```

根据 `caffe/python` 目录下的 `requirements.txt` 安装 `python` 附加依赖库

```
for req in $(cat requirements.txt); do sudo pip install
$req; done
```

**安装 caffe:**

cd 到 caffe 文件夹, 复制并修改配置文件:

```
cp Makefile.config.example Makefile.config
```

打开 Makefile.config 并编辑:

```
sudo gedit Makefile.config
```

-----  
--

**根据情况更改文件, 简要说明如下:**

使用 cudnn

```
USE_CUDNN := 1
```

使用的 opencv 版本是 3

```
OPENCV_VERSION := 3
```

删除 CUDA\_ARCH := 的前两行, 避免 CUDA 报错

如果使用 MATLAB, MATLAB\_DIR:= /usr/local 这里改为 matlab 安装路径, 例如: MATLAB\_DIR:= /usr/local/MATLAB/R2015b

PYTHON 使用默认的 2.7

PYTHON\_INCLUDE :=这里默认是/user/lib/...

检查下默认地址是否包含 numpy, 因为 pip 可能会把 numpy 安装在了/usr/local/lib/... 需要检查后改一下, 若修改此处 PYTHON\_LIB 后也要追加/usr/local/lib/目录

使用 python 来编写 layer

```
WITH_PYTHON_LAYER := 1
```

将# Whatever else you find you need goes here.下面的

```
INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include LIBRARY_DIRS := $(PYTHON_LIB) /usr/local/lib /usr/lib
```

修改为:

```
INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include  
/usr/include/hdf5/serial/
```

```
LIBRARY_DIRS := $(PYTHON_LIB) /usr/local/lib /usr/lib /usr/lib/x86_64-  
linux-gnu /usr/lib/x86_64-linux-gnu/hdf5/serial
```

这是因为 ubuntu16.04 的文件包含位置发生了变化, 尤其是需要用到的 hdf5 的位置, 所以需要更改这一路径

同时打开 Makefile 在 LIBRARIES +=后把 hdf5 hl hdf5 替换成 hdf5\_serial hl hdf5\_serial, 并添加 opencv 部分, 防止报错。即将第一行改为第二行:

```
LIBRARIES += glog gflags protobuf boost_system boost_filesystem m  
hdf5 hl hdf5
```

```
LIBRARIES += glog gflags protobuf boost_system boost_filesystem m  
hdf5_serial hl hdf5_serial opencv_core opencv_highgui opencv_imgproc  
opencv_imgcodecs opencv_videoio
```

如果 gcc 没有降到 4.7 或 4.9, 就编辑 /usr/local/cuda/include/host\_config.h , 将其中的第 115 行注释掉

```
//#error -- unsupported GNU version! gcc versions later than 5 are  
not supported!
```

-----  
**安装 caffe:**

在 caffe 文件夹下:

```
make all -j8
make test
make runtest #所有都显示 pass 表示安装成功
```

若报错:

`make: *** [.build_release/lib/libcaffe.so.1.0.0] Error 1`

解决:

```
sudo gedit /etc/ld.so.conf.d/opencv.conf
# 添加 /usr/local/lib
```

## 6. 安装 anaconda

Anaconda 清华源下载地址: <https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/>

将安装包放在 home 文件夹

```
sudo bash Anaconda3-4.4.0-Linux-x86_64.sh
```

accept, 一路 yes 完成, 最后 vscode 安装选 no。安装路径默认为 /home/noah/anaconda3

```
sudo gedit /etc/profile
```

将 /home/noah/anaconda3/bin 添加到 path 中

```
export PATH="/home/noah/anaconda3/bin:$PATH"
```

创建一个名为 tensorflow 的 conda 环境 Python 3.6

```
conda create -n tfpy36 python=3.6
```

激活 conda 环境

```
source activate tfpy36
```

## 7. 安装 tensorflow-gpu 版

tensorflow 的各个版本: <https://pypi.org/project/tensorflow-gpu/#history>

Tensorflow 不同版本要求与 CUDA 及 CUDNN 版本对应关系

<https://blog.csdn.net/omodao1/article/details/83241074>

安装参考地址: <https://www.cnblogs.com/xuliangxing/p/7575586.html>

Version	Python version	Compiler	Build tools	cuDNN	CUDA
tensorflow_gpu-1.11.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.15.0	7	9
tensorflow_gpu-1.10.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.15.0	7	9
tensorflow_gpu-1.9.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.11.0	7	9
tensorflow_gpu-1.8.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.10.0	7	9
tensorflow_gpu-1.7.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.9.0	7	9
tensorflow_gpu-1.6.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.9.0	7	9
tensorflow_gpu-1.5.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.8.0	7	9
tensorflow_gpu-1.4.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.5.4	6	8
tensorflow_gpu-1.3.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.5	6	8
tensorflow_gpu-1.2.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.5	5.1	8
tensorflow_gpu-1.1.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.2	5.1	8
tensorflow_gpu-1.0.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.2	5.1	8

激活 conda 环境

```
source activate tfpy36
安装:
pip install tensorflow-gpu==1.2.0 #一定不要用 sudo
退出环境
source activate tfpy36
验证
source activate tfpy36
python
import tensorflow as tf #无报错表示成功
```

### 补充: gcc/g++降级

```
下载并安装 gcc/g++ 4.7
sudo apt-get install -y gcc-4.7
sudo apt-get install -y g++-4.7
链接 gcc/g++ 实现降级
cd /usr/bin
sudo rm gcc
sudo ln -s gcc-4.7 gcc
sudo rm g++
sudo ln -s g++-4.7 g++
查看是否连接到 4.7.x
ls -al gcc g++
gcc --version
g++ --version
```

## 8. 安装 matlab

如果 caffe 需要编译 matcaffe 接口则需要在编译 caffe 前先安装 matlab  
解压破解包

```
cd ~/matlab2016b_Linux
sudo apt-get install rar
rar x Matlab\ 2016b\ Linux64\ Crack.rar
```

挂载镜像:

```
mkdir mountMat #在 matlab2016b_Linux 中创建一个挂在文件夹
sudo mount -t auto -o loop ./R2016b_glnxa64_dvd1.iso
./mountMat #挂载后 mountMat 里面会有一个 install 文件
```

安装:

```
sudo mountMat/install
```

## 9. 安装 jsoncpp (\*)

### 1) 编译安装 ninja

```
mkdir ~/environment (该目录为自己新建的一个文件夹, 可以用不同的名字)
cd ~/environment
git clone git://github.com/ninja-build/ninja.git && cd ninja
git checkout release
./configure.py --bootstrap
sudo cp ~/environment/ninja/ninja /usr/bin
```

### 2) 编译 meson

```
cd ~/environment
wget https://github.com/mesonbuild/meson.git
tar -xf meson-*.tar.gz
```

```
cd meson-*
sudo python3 setup.py install
(使用 python3 安装时如果提示没有 setuptools, 用 sudo apt-get install
python3-pip 解决)
```

### 3) 编译 jsoncpp

```
cd ~/environment
wget https://github.com/open-source-parsers/jsoncpp.git
unzip jsoncpp-master.zip
cd jsoncpp-master
mkdir build-static
#BUILD_TYPE=debug #如果要编译 debug 版本, 注释后面行, 并取消当前行的
注释
```

```
BUILD_TYPE=release
#LIB_TYPE=shared #编译动态链接库版本
LIB_TYPE=static #编译静态链接库版本
/usr/local/bin/meson --buildtype ${BUILD_TYPE} --default-
library ${LIB_TYPE} . build-${LIB_TYPE}
ninja -v -C build-${LIB_TYPE} test
cd build-${LIB_TYPE}
sudo ninja install
```

### 10. 安装 curl (\*)

```
sudo apt-get install libcurl3 libcurl3-dev libcurl
```

### 11. 安装 zmq (\*)

```
cd ~/environment
wget
https://github.com/zeromq/libzmq/releases/download/v4.2.2/zerom
q-4.2.2.tar.gz
tar -zxf zeromq-4.2.2.tar.gz
cd zeromq-4.2.2
./configure --prefix=/home/ygy/zmq --without-libsodium
make
make install
```

### 12. 安装 libevent (\*)

```
wget http://libevent.org/libevent-2.0.22.tar.gz
tar -zxvf libevent-2.0.22-stable.tzr.gz
cd libevent-2.0.22-stable
./configure --prefix=/usr
make
sudo make install
检查是否 libevent 已经安装完毕
ls -al /usr/lib | grep libevent
```



**说明：凡大标题后打星号的（\*）均为必须安装的三方库**