* 个人电脑情况：   
  Ubuntu16.04LTS 64bit   
  含有NVIDIA 显卡 实验室电脑Quadro K620

**配置步骤：**   
**第一步：基本步骤 #更新安装必要的软件包，去除不必要的包。**

sudo apt-get update

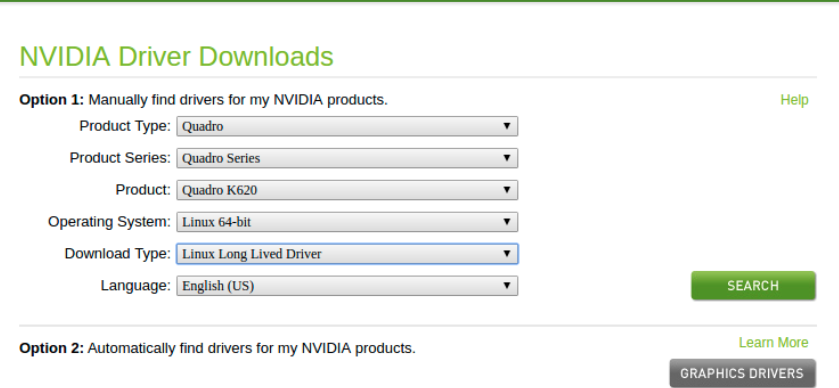
sudo apt-get upgrade

sudo apt-get install build-essential

sudo apt-get autoremove

**第二步：安装Git #后面获取caffe和其他配置文件用**

sudo apt-get install git

**第三步：查看电脑的显卡情况 驱动安装**   
查看显卡情况，运行：lspci | grep -i nvidia   
比如我的就显示：   
01:00.0 VGA compatible controller: NVIDIA Corporation GM107GL [Quadro K620] (rev a2)   
01:00.1 Audio device: NVIDIA Corporation Device 0fbc (rev a1)   
如果驱动没有装好，转到官网：<http://www.nvidia.com/Download/index.aspx?lang=en-us> 选择你电脑对应的信息，nvidia自动给出适合的型号，然后你可以下载安装包到本地。附一张我电脑的截图：   
   
点击SEARCH 就会给出适合的型号：   
   
然后记住版本号，这里我的是nvidia-375，然后在终端里安装就好。（注：我不采用进入lightdm的方法，感觉也可以

sudo add-apt-repository ppa:graphics-drivers/ppa #添加源

sudo apt-get update

sudo apt-get install nvidia-375 #你的版本号

sudo apt-get install mesa-common-dev

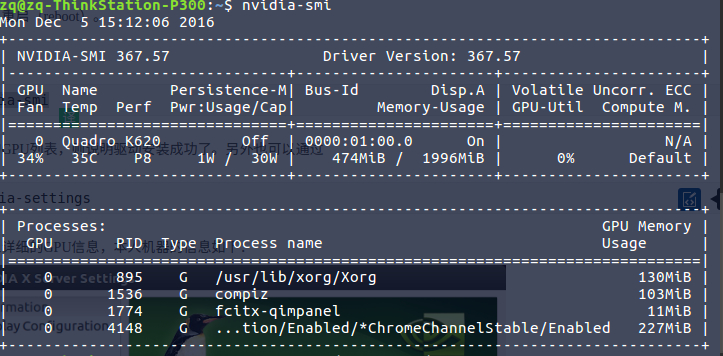
sudo apt-get install freeglut3-dev

安装完以上请重启电脑，可以选择运行

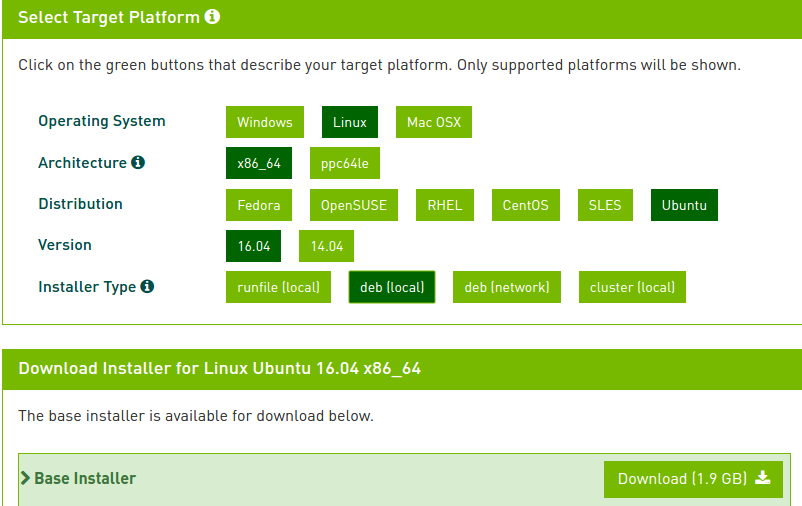
sudo reboot #重启电脑

重启电脑后，运行

nvidia-smi



如果出现了你的GPU列表，则说明驱动安装成功了。

**第四步：安装CUDA8.0**   
<https://developer.nvidia.com/cuda-downloads> 进入官网选择对应的安装包 下载deb包   
附图：   


下载好本地的deb包之后 运行：

sudo dpkg -i cuda-repo-ubuntu1404\*amd64.deb

sudo apt-get update

sudo apt-get install cuda

最后配置CUDA环境变量：   
声明环境变量：

export PATH=/usr/local/cuda-8.0/bin${PATH:+:${PATH}}

export LD\_LIBRARY\_PATH=/usr/local/cuda-8.0/lib64${LD\_LIBRARY\_PATH:+:${LD\_LIBRARY\_PATH}}

sudo gedit /etc/profile #保存

置环境变量和动态链接库，在命令行输入：

sudo gedit /etc/profile #打开文件

在打开的文件末尾加入：

export PATH = /usr/local/cuda/bin:$PATH

保存之后，创建链接文件：

sudo gedit /etc/ld.so.conf.d/cuda.conf

在打开的文件中添加如下语句

/usr/local/cuda/lib64

执行：

sudo ldconfig #保存

这里我提供了以下方法来检查CUDA是否安装成功：   
检查CUDA版本：

nvcc -V

编译CUDA Sample：

/usr/local/cuda/bin/cuda-install-samples-8.0.sh ~/cuda-samples #复制

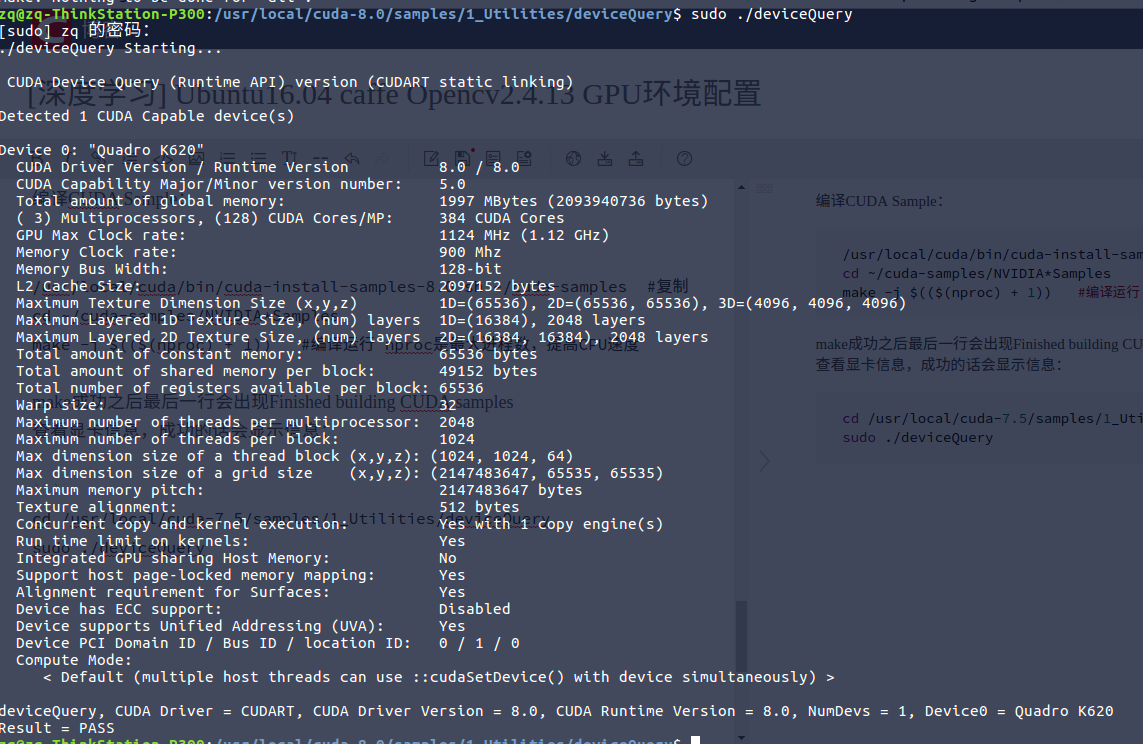
cd ~/cuda-samples/NVIDIA\*Samples

make -j $(($(nproc) + 1)) #编译运行 nproc是最大进程数，提高CPU速度

make成功之后最后一行会出现Finished building CUDA samples   
查看显卡信息，成功的话会显示信息：

cd /usr/local/cuda-7.5/samples/1\_Utilities/deviceQuery

sudo ./deviceQuery

附图：   


**第五步：安装OpenBLAS**

git clone https://github.com/xianyi/OpenBLAS.git

cd OpenBLAS

make -j $(($(nproc) + 1))

make install

这里提前声明，git caffe之后，编译之前，在Makefile.config里修改为以下内容：

BLAS := open

# Custom (MKL/ATLAS/OpenBLAS) include and lib directories.

# Leave commented to accept the defaults for your choice of BLAS

# (which should work)!

BLAS\_INCLUDE := /opt/OpenBLAS/include

BLAS\_LIB := /opt/OpenBLAS/lib

(optional)BLAS 加速设置   
事实上，在计算时候将所有CPU核心用完并不一定是一件好的事情。CPU核心使用的越多，数据通信的开销就会越大，性能反而会下降。OpenBLAS提供了很多方法修改占用CPU核心数。（这里：[https://github.com/xianyi/OpenBLAS#set-the-number-of-threads-with-environment-variables](https://github.com/xianyi/OpenBLAS" \l "set-the-number-of-threads-with-environment-variables)）   
我觉得导出环境变量的方式最方便。在控制台里输入：

export OPENBLAS\_NUM\_THREADS=4

**第六步：安装、编译OpenCV2.4.13**   
最开始我配置过OpenCV3.0.0 ,但是因为运行一些目标检测的demo有点问题，于是换成了2.4.13版本。需要配置OpenCV3.0.0的可以点这里 <https://pan.baidu.com/s/1jIdmcL4>   
下载这个压缩包解压。   
运行dependences.sh 和opencv\_install.sh就可以了。   
我这里就记录配置Opencv2.4.13的过程。首先，参见github上有人分享了脚本，<https://github.com/jayrambhia/Install-OpenCV> 将这个项目下载到本地   
运行

cd Ubuntu

chmod +x \*

./opencv\_latest.sh

**第八步：安装依赖项**   
1、安装google-glog   
下载地址：<https://pan.baidu.com/s/1slpjjeH>   
安装步骤：   
将glog‐0.3.3.tar.gz这个文件拷贝到主文件夹下，即/home/yourname 文件夹下，点击鼠标右键，选择提取到此处（即是解压），解压后终端输入：

cd /home/yourname/glog-0.3.3

./configure

make

sudo make install

安装成功。

2、安装其他依赖项：

sudo apt-get install -y libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev libboost-all-dev libhdf5-serial-dev libgflags-dev libgoogle-glog-dev liblmdb-dev protobuf-compiler protobuf-c-compiler protobuf-compiler

3、安装Python所需依赖项：

sudo apt-get install -y python-numpy python-scipy python-matplotlib python-sklearn python-skimage python-h5py python-protobuf python-leveldb python-networkx python-nose python-pandaspython-gflags Cython ipython

**第八步：安装cuDNN**   
首先从官网上下载cudnn安装包<https://developer.nvidia.com/cudnn>   
如果你以前没有注册过，注册一个帐号，填写下他们的问卷就可以下载了。   
下载到本地之后进行解压、配置：

tar xvf cudnn\*.tgz #cudnn下载的目录下执行

cd cuda

sudo cp \*/\*.h /usr/local/cuda/include/ #复制动态链接库

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

sudo ln -s /usr/local/lib/libcudnn.so.6.5.48 /usr/local/lib/libcudnn.so.5.0 #添加软链接，5.0都是版本号，根据你下载的版本来，若和我不一致，修改数字

sudo ln -s /usr/local/lib/libcudnn.so.5.0 /usr/local/lib/libcudnn.so

sudo chmod a+r /usr/local/cuda/lib64/libcudnn\* #添加权限

**第九步：caffe**   
前往 <https://github.com/BVLC/caffe> 下载caffe到本地，建议放在/home/yourname/下，然后进入caffe，启动终端 （cd caffe）

cp Makefile.config.example Makefile.config

gedit Makefile.config #打开配置文件

修改：（不加#的部分）   
USE\_CUDNN := 1

BLAS :=open   
BLAS\_INCLUDE := /opt/OpenBLAS/include   
BLAS\_LIB := /opt/OpenBLAS/lib

PYTHON\_INCLUDE := /usr/include/python2.7 /usr/lib/python2.7/dist-packages/numpy/core/include

PYTHON\_LIB := /usr/local/lib

WITH\_PYTHON\_LAYER := 1

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

修改之后保存关闭。   
修改Makefile

cd ~/caffe

gedit Makefile

进入文件，修改   
NVCCFLAGS += -ccbin=(*CXX*)−*Xcompiler*−*fPIC*(COMMON\_FLAGS)   
为 NVCCFLAGS += -D\_FORCE\_INLINES -ccbin=(*CXX*)−*Xcompiler*−*fPIC*(COMMON\_FLAGS)

说明：修改这一步是为了避免出现string.h ‘memcy’ was not declared in this scope这样的错误，这种错误通常是由于gcc版本太新而导致的。   
建立hdf5软连接：

cd /usr/lib/x86\_64-linux-gnu #hdf5文件目录，检查下是否和我一样

sudo ln libhdf5\_serial.so.10.1.0 libhdf5.so

sudo ln libhdf5\_serial\_hl.so.10.0.2 libhdf5\_hl.so

sudo ldconfig #使连接生效

python必要安装

cd python

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

编译caffe：

cd ~/caffe

make all -j4

make test -j4

make runtest -j4

make pycaffe -j4

编译过程中可能会出现nvcc warning: The ‘compute\_20’and ‘sm\_21’ architectures are ……这种警告，可以不管，只要编译过程中不报错就OK。   
（如果你不希望出现warning，可以进入Makefile.config文件修改CUDA\_ARCH的内容，结合你GPU的计算能力，注释掉一些行，当然修改之后要重新编译caffe），接下来运行：

cd ~/caffe/python

python

import caffe