Caffe 安装教程

深圳安法影像技术有限公司 周攀

一、 提前准备：

1、 软件准备：

Linux 系统：Ubuntu（16.04.3LTS）

CUDA：cuda_8.0.44_linux.run、cudnn5.1、Nvidia 驱动版本 375.66

OpenCV：opencv3.1

其他软件：UltraISO（制作 Linux 系统启动盘）

注：必须严格安装给出的系统和驱动版本号安装，否则会出现很多错误，上述版本已经完成了安装。

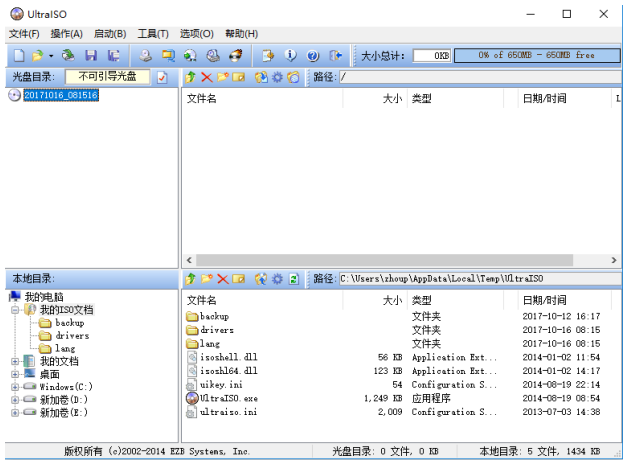
2、 硬件准备：

空 U 盘（16G）

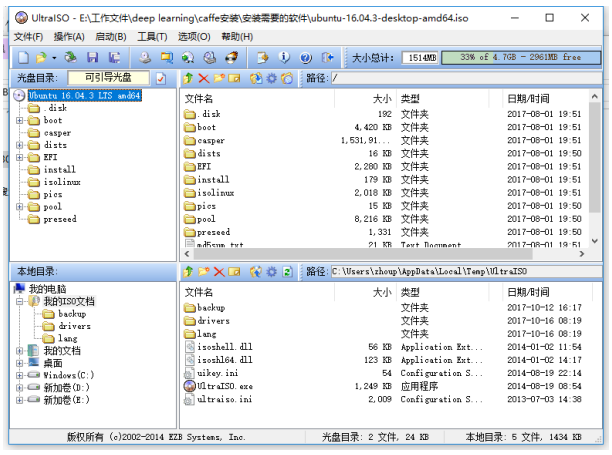
二、 安装步骤：

1、 将 U 盘制作为 Ubuntu 系统的启动盘：

文件->打开->选择下载好的 Ubuntu 系统



加载完后，出现如下界面：



启动->写入硬盘镜像->写入

2、 安装 window 和 Linux 双系统：

A、只有一块硬盘的电脑：

重新启动电脑，进入 BIOS 界面，设置 U 盘为第一启动项，进入 Ubuntu 安装界面。具体分区为（分区全部安装在同一硬盘上面）：

根分区（/）：100G

Boot 分区（/boot）：500M

交换分区：与电脑的内存条的大小相当

B、有两块硬盘的电脑：

重新启动电脑，进入 BIOS 界面，设置 U 盘为第一启动项，进入 Ubuntu 安装界面。具体分区为（分区全部安装在同一硬盘上面）：

根分区（/）：100G 安装于机械硬盘上面

Boot 分区（/boot）：500M 安装于主启动硬盘上面，一般为固态硬盘

交换分区：与电脑的内存条的大小相当 安装于机械硬盘上面

3、 安装依赖包。打开终端，输入：

```
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 libopenblas-dev liblapack-dev libatlas-base-dev
sudo apt-get install libgflags-dev libgoogle-glog-dev liblmdb-dev
sudo apt-get install git cmake build-essential
```

注：以上依赖包最好一个一个安装，务必保证所有安装包都安装上了。

有一定几率安装失败而导致后续步骤出现问题，所以要确保以上依赖包都已安装成功，验证方法就是重新运行安装命令，如验证 git cmake build-essential 是否安装成功共则再次运行以下命令：

```
sudo apt-get install git cmake build-essential
```

界面提示如下则说明已成功安装依赖包，否则继续安装直到安装成功。

```
yhao@yhao-X550VB:~$ sudo apt-get install git cmake build-essential
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
build-essential 已经是最新版 (12.1ubuntu2)。
cmake 已经是最新版 (3.5.1-1ubuntu3)。
git 已经是最新版 (1:2.7.4-0ubuntu1.1)。
下列软件包是自动安装的并且现在不需要了：
  lib32gcc1 libc6-i386
使用'sudo apt autoremove'来卸载它(它们)。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 94 个软件包未被升级。
```

4、 安装 Nvidia 驱动：

首先去官网 http://www.nvidia.com/Download/index.aspx?lang=en-us 查看适合自己显卡的驱动并下载：

驱动文件后缀名应当是以.run 结尾的。我们要把这个文件移动到/home/username/目录下，原因是下面我们要切换到文字界面下，如果放到~/下载下面，我们没有办法进入下载这个目录（没有中文输入法，且中文全部是乱码）。

注：最好安装给定版本的驱动，否者很有可能版本过新，导致错误。

在终端下输入： sudo gedit /etc/modprobe.d/blacklist.conf

输入密码后在最后一行加上 blacklist nouveau。这里是将 Ubuntu 自带的显卡驱动加入黑名单。在终端输入：

sudo update-initramfs -u

重启电脑。这里要尤其注意，安装显卡驱动要先切换到文字界面(按 Ctrl+Alt+F1~F6)。所以，启动电脑后，先进入文字界面。然后，输入命令：

sudo service lightdm stop

现在可以安装驱动了，先进入家目录 cd ~，然后：sudo sh NVIDIA-Linux-x86_64-375.20.run(具体改为实际下载的版本)，按照提示一步步来，完成后，再次重启电脑。

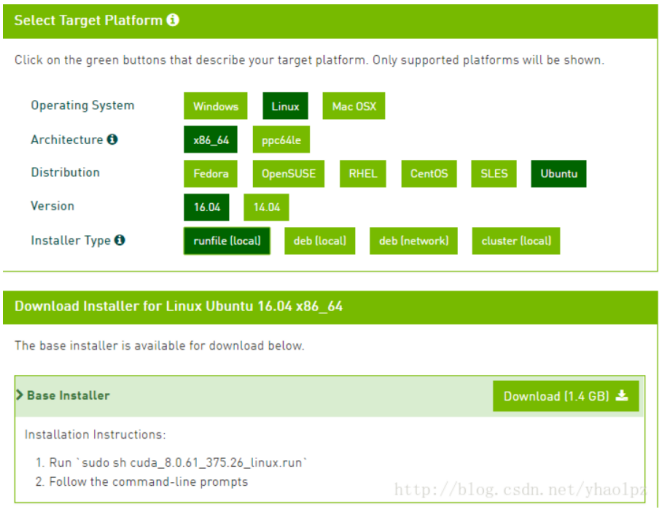
安装完成之后输入以下指令进行验证：sudo nvidia-smi，若列出了 GPU 的信息列表则表示驱动安装成功。如下所示：

```
[redacted]:~$ sudo nvidia-smi
[sudo] [redacted] 的密码:
Fri Dec  9 21:42:18 2016
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 375.20                 Driver Version: 375.20                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Quadro K620         Off  | 0000:03:00.0     On |                  N/A |
| 34%   40C    P8     1W /  30W |    198MiB /  2028MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    0      4420    G   /usr/lib/xorg/Xorg                              91MiB |
|    0      5029    G   compiz                                          98MiB |
|    0      5294    G   fcitx-qimpanel                                   5MiB |
+-----------------------------------------------------------------------------+
[redacted]:~$
```

5、 安装 CUDA：

进入 https://developer.nvidia.com/cuda-downloads ，依次选择 CUDA 类型然后下载即可。



打开终端，输入：

```
sudo sh cuda_8.0.61_375.26_linux.run
```

注：执行后会有一系列提示让你确认，但是注意，有个让你选择是否安装 nvidia 驱动时，一定要选择否。

6、 环境配置：

打开终端，输入：

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

打开后在文件最后加入以下四行内容：

```
export LD_LIBRARY_PATH=/usr/lib/x86_64-linux-gnu:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/lib/x86_64-linux-gnu:$LD_LIBRARY_PATH
export PATH=/usr/local/cuda-8.0/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
```

使该配置生效：

```
source ~/.bashrc
```

7、 验证 CUDA 8.0 是否安装成功：

分别执行以下命令：

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

若看到类似以下信息则说明 cuda 已安装成功：

```
---------------------------------------------------------------------------
./deviceQuery Starting...
 CUDA Device Query (Runtime API) version (CUDART static linking)
Detected 1 CUDA Capable device(s)
Device 0: "GeForce GT 740M"
  CUDA Driver Version / Runtime Version          8.0 / 8.0
  CUDA Capability Major/Minor version number:    3.5
  Total amount of global memory:                 2004 MBytes (2100953088 bytes)
  ( 2) Multiprocessors, (192) CUDA Cores/MP:     384 CUDA Cores
  GPU Max Clock rate:                            1032 MHz (1.03 GHz)
  Memory Clock rate:                             800 Mhz
```

```
  Memory Bus Width:                              64-bit
  L2 Cache Size:                                 524288 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(65536), 2D=(65536, 65536),
3D=(4096, 4096, 4096)
  Maximum Layered 1D Texture Size, (num) layers  1D=(16384), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(16384, 16384), 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 1 copy engine(s)
  Run time limit on kernels:                     No
  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 GT 740M
Result = PASS
```

8、 安装 cudnn：

登录官网：https://developer.nvidia.com/rdp/cudnn-download ，下载对应 cuda 版本且 linux 系统的 cudnn 压缩包（文件夹里面已经包含）。

下载完成后解压，得到一个 cudn 文件夹，该文件夹下 include 和 lib64 两个文件夹，命令行进入 cudn/include 路径下，然后进行以下操作：

```
sudo cp cudnn.h /usr/local/cuda/include/ #复制头文件
```

然后命令行进入 cudn/lib64 路径下，运行以下命令：

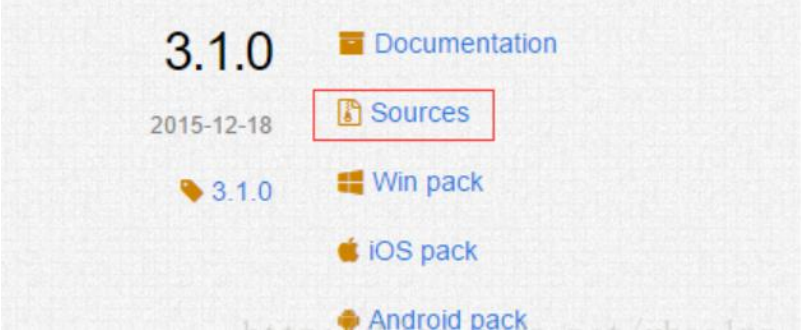
```
sudo cp lib* /usr/local/cuda/lib64/ #复制动态链接库
cd /usr/local/cuda/lib64/
sudo rm -rf libcudnn.so libcudnn.so.5 #删除原有动态文件
sudo ln -s libcudnn.so.5.1.10 libcudnn.so.5 #生成软衔接
sudo ln -s libcudnn.so.5 libcudnn.so #生成软链接
```

安装完成后可用 nvcc -V 命令验证是否安装成功，若出现以下信息则表示安装成功：

```
yhao@yhao-X550VB:~$ nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2016 NVIDIA Corporation
Built on Tue_Jan_10_13:22:03_CST_2017
Cuda compilation tools, release 8.0, V8.0.61
```

9、 安装 opencv3.1：

进入官网：http://opencv.org/releases.html ， 选择 3.1.0 版本的 source ， 下载 opencv-3.1.0.zip。



解压到你要安装的位置，命令行进入已解压的文件夹 opencv-3.1.0 目录下，修改 /opencv-3.1.0/modules/cudalegacy/src/graphcuts.cpp 文件内容，如图：

```
//
//M*/

#include "precomp.hpp"

//#if !defined (HAVE_CUDA) || defined (CUDA_DISABLER)
#if !defined (HAVE_CUDA) || defined (CUDA_DISABLER) || (CUDART_VERSION>=8000)

void cv::cuda::graphcut(GpuMat&, GpuMat&, GpuMat&, GpuMat&, GpuMat&, GpuMat&, GpuMat&, Stream&)
{ throw_no_cuda(); }
void cv::cuda::graphcut(GpuMat&, GpuMat&, GpuMat&, GpuMat&, GpuMat&, GpuMat&, GpuMat&, GpuMat&,
GpuMat&, GpuMat&, GpuMat&, Stream&) { throw_no_cuda(); }

void cv::cuda::connectivityMask(const GpuMat&, GpuMat&, const cv::Scalar&, const cv::Scalar&,
Stream&) { throw_no_cuda(); }
void cv::cuda::labelComponents(const GpuMat&, GpuMat&, int, Stream&) { throw_no_cuda(); }
```

打开终端，进入到 oepncv3.1 文件夹下，输入：

```
mkdir build # 创建编译的文件目录
cd build
cmake -D CMAKE_BUILD_TYPE=Release -D CMAKE_INSTALL_PREFIX=/usr/local ..
make -j8  #编译
```

编译成功后安装：

```
sudo make install #安装
```

安装完成后通过查看 opencv 版本验证是否安装成功：

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

10、安装 caffe：

首先在你要安装的路径下 clone，打开终端，输入 ：

```
git clone https://github.com/BVLC/caffe.git
```

进入 caffe ， 将 Makefile.config.example 文件复制一份并更名为 Makefile.config ，也可以在 caffe 目录下直接调用以下命令完成复制操作 ：

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

复制一份的原因是编译 caffe 时需要的是 Makefile.config 文件，而 Makefile.config.example 只是 caffe 给出的配置文件例子，不能用来编译 caffe。
然后修改 Makefile.config 文件，在 caffe 目录下打开该文件：

sudo gedit Makefile.config

修改 Makefile.config 文件内容：

1.应用 cudnn

将#USE_CUDNN := 1 修改成：USE_CUDNN := 1

2.应用 opencv 版本

将#OPENCV_VERSION := 3 修改为：OPENCV_VERSION := 3

3.使用 python 接口

将#WITH_PYTHON_LAYER := 1 修改为 WITH_PYTHON_LAYER := 1

4.修改 python 路径

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

然后修改 caffe 目录下的 Makefile 文件：

将：
NVCCFLAGS +=-ccbin=$(CXX) -Xcompiler-fPIC $(COMMON_FLAGS)
替换为：
NVCCFLAGS += -D_FORCE_INLINES -ccbin=$(CXX) -Xcompiler -fPIC $(COMMON_FLAGS)

将：
LIBRARIES += glog gflags protobuf boost_system boost_filesystem m hdf5_hl hdf5
改为：
LIBRARIES += glog gflags protobuf boost_system boost_filesystem m hdf5_serial_hl hdf5_serial

然后修改 /usr/local/cuda/include/host_config.h 文件 ：

将
#error-- unsupported GNU version! gcc versions later than 4.9 are not supported!
改为
//#error-- unsupported GNU version! gcc versions later than 4.9 are not supported!

开始编译了，在 caffe 目录下执行 ：

make all -j8

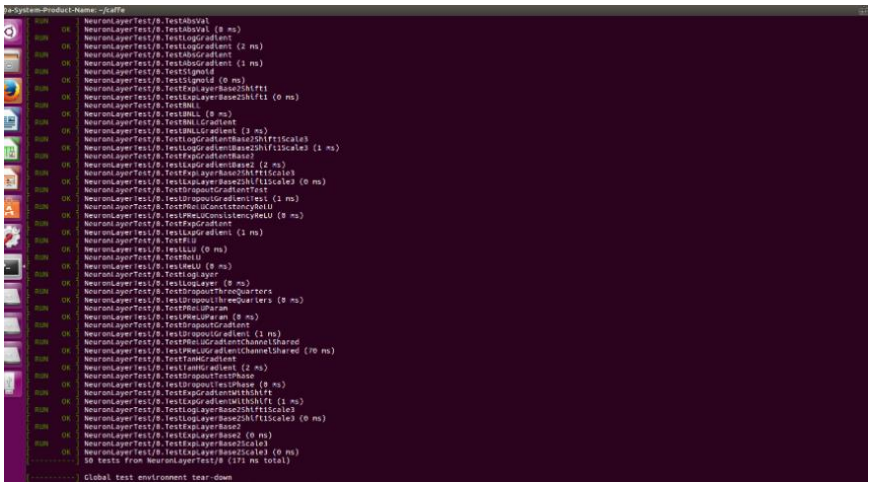
这是如果之前的配置或安装出错，那么编译就会出现各种各样的问题，所以前面的步骤一定要细心。编译成功后可运行测试：

sudo make runtest -j8

如出现错误，输入：

```
sudo cp /usr/local/cuda-8.0/lib64/libcudart.so.8.0 /usr/local/lib/libcudart.so.8.0 && sudo ldconfig
sudo cp /usr/local/cuda-8.0/lib64/libcublas.so.8.0 /usr/local/lib/libcublas.so.8.0 && sudo ldconfig
sudo cp /usr/local/cuda-8.0/lib64/libcurand.so.8.0 /usr/local/lib/libcurand.so.8.0 && sudo ldconfig
sudo cp /usr/local/cuda-8.0/lib64/libcudnn.so.5 /usr/local/lib/libcudnn.so.5 && sudo ldconfig
```

如果出现下面，说明前面安装正确。



11、MNIST 数据集测试

配置 caffe 完成后，我们可以利用 MNIST 数据集对 caffe 进行测试，过程如下：

1.将终端定位到 Caffe 根目录

```
cd ~/caffe
```

2.下载 MNIST 数据库并解压缩

```
./data/mnist/get_mnist.sh
```

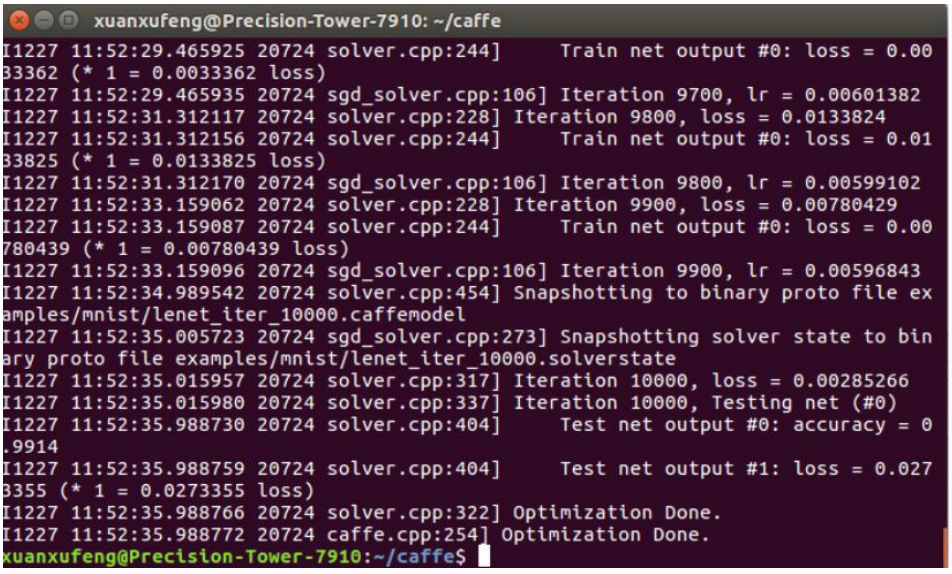
3.将其转换成 Lmdb 数据库格式

```
./examples/mnist/create_mnist.sh
```

4.训练网络

```
./examples/mnist/train_lenet.sh
```

训练的时候可以看到损失与精度数值，如下图：



可以看到最终训练精度是 0.9914。

12、可参考网站：

http://kangqingfei.cn/2016/08/25/Linux-%E5%AE%89%E8%A3%85Caffe/
http://blog.csdn.net/yhaolpz/article/details/71375762?locationNum=14&fps=1
http://www.linuxidc.com/Linux/2016-12/138870.htm
http://blog.csdn.net/u014696921/article/details/60140264