

# Ubuntu16.04 安装 NVIDIA 驱动

\_gcc4.8\_Anaconda2\_cuda9.0\_cudnn\_caffe\_Pytorch\_tensorflow\_

## Keras 教程

鲍春 2018.11.24

### 1.gcc 降级

由于安装 nvidia 显卡驱动的时候 gcc 版本过高会导致不成功，所以在安装驱动之前我们要对 gcc 进行降级，在这里我用的 gcc 版本为 gcc4.8，其实版本在 gcc5.3 以下就可以了。

(1)先看看系统用的 gcc 和 g++是什么版本

```
gcc -v
```

可以获得的信息如下

```
gcc version 5.2.1
```

(2)如果我们想使用 gcc4.8，首先看看有没有安装 gcc4.8

```
ls /usr/bin/gcc*
```

结果只有/usr/bin/gcc /usr/bin/gcc-4.4 两个，那么我们需要安装

```
sudo apt-get install gcc-4.8 gcc-4.8-multilib g++-4.8 g++-4.8-multilib
```

(3)安装好后输入以下指令：

```
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.8 40
```

```
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-5 50
```

(这里的 50 和 40 代表优先级)

接着输入：

```
sudo update-alternatives --config gcc
```

```
baochun@baochun:/$ sudo update-alternatives --config gcc
有 2 个候选项可用于替换 gcc (提供 /usr/bin/gcc)。
```

选择	路径	优先级	状态
0	/usr/bin/gcc-4.8	50	自动模式
1	/usr/bin/gcc-4.8	50	手动模式
* 2	/usr/bin/gcc-4.9	40	手动模式

```
要维持当前值[*]请按<回车键>，或者键入选择的编号： █
```

会看到如下的选项，有 2 个候选项可用于替换 gcc (提供 /usr/bin/gcc)。由于我之前换过 gcc，将 gcc-5 已经删了，如果是新的环境，默认的应该是 gcc-5,而不是 gcc-4.9,在这里只是示范一下。

要维持当前值[\*]请按回车键，或者键入选择的编号：

要想用哪个 gcc 就输入编号。

再用 `gcc -v` 来看一下版本是否改变了。

```
gcc -v
```

(4)同样也要设置一下 `g++` 的

```
sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-5 50
```

```
sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.9 40
```

如果想删除可选项的话可以键入以下指令：

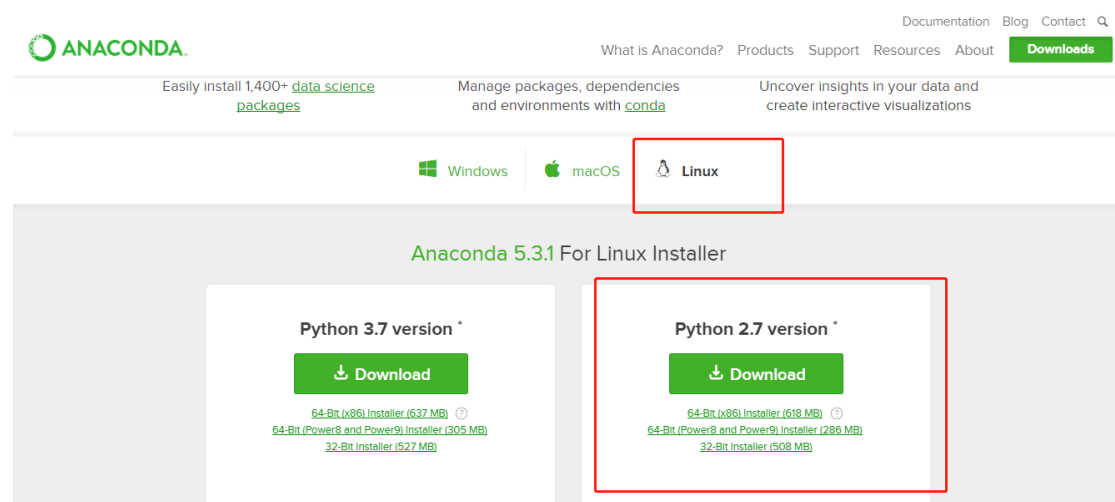
```
sudo update-alternatives --remove gcc /usr/bin/gcc-4.9
```

## 2. 安装 Anaconda

anaconda 里面集成了很多关于 python 的第三方库，通过 anaconda 安装会省很多事，不然以后需要自己手动安装一些依赖项，因此在这先将 anaconda 安装。

### (1) 下载

下载网址：<https://www.anaconda.com/download/#linux>



下载自己需要的版本，我选择的版本为：**Anaconda2-5.0.1-Linux-x86\_64.sh**

### (2) 安装

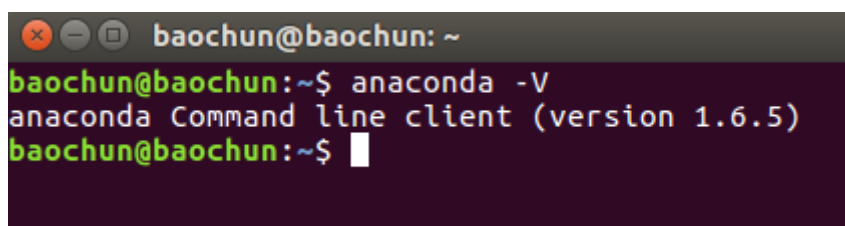
进入 **Anaconda2-5.0.1-Linux-x86\_64.sh** 的目录下，执行下列命令进行安装。

```
sudo ./Anaconda2-5.0.1-Linux-x86_64.sh
```

输入回车，弹出须知，再一直按 `Ctrl+F`，直到须知结束，输入 `yes` 继续。最后选择自己的安装路径，即可完成安装，会在 `home` 目录下出现一个 `anaconda2` 的文件夹。

### (3) 查看版本

```
anaconda -V
```

A terminal window with a dark background. The prompt is 'baochun@baochun: ~'. The user has entered 'anaconda -V' and the output is 'anaconda Command line client (version 1.6.5)'. The prompt is now 'baochun@baochun: ~\$' with a cursor.

### 3.安装 NVIDIA 驱动

#### (1)禁用 nouveau 驱动和相关的驱动包

用编辑器打开 blacklist.conf 配置文件

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

在文件的最后一行加入下面的命令，屏蔽有影响的驱动包（有的博客添加了 blacklist amd76x\_edac，其实不加也是可以安装成功的）

```
blacklist rivafb  
blacklist vga16fb  
blacklist nouveau  
blacklist nvidiafb  
blacklist rivatv
```

#### (2)卸载所有安装的 nvidia 驱动

如果之前没安装过 nvidia 驱动，可以不执行这条指令，但是执行以下也没关系。

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

卸载完以后，**重启**。

#### (3)安装显卡驱动

在 <http://www.geforce.cn/drivers> 上下载对应的显卡驱动包，如下图所示。



我选择的是(NVIDIA-Linux-x86\_64-390.67.run)（支持大部分显卡，我使用的是 GTX 1070）

### 安装需要的依赖

```
sudo apt update
```

```
sudo apt install dkms build-essential linux-headers-generic
```

### 安装驱动包

因为要关闭图形界面，最好手机拍照保存此教程，

首先，**Ctrl+Alt+F1** 进入命令提示符界面

然后，输入对应的 username 和 password 进入命令行。

最后，使用指令 **sudo service lightdm stop** 关闭图形界面，再利用 **cd** 指令进入下载好的驱动目录

```
sudo chmod 755 NVIDIA-Linux-x86_64-390.67.run
```

#修改权限（否则没有访问权限，无法进行指令安装）

```
sudo ./NVIDIA-Linux-x86_64-390.67.run --no-x-check --no-nouveau-check --no-opengl-files
```

#安装驱动

#--no-x-check 关闭 X 服务

#--no-nouveau-check 禁用 nouveau

#--no-opengl-files 不安装 OpenGL 文件

#...安装完成后

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

```
sudo reboot
```

判断显卡驱动是否安装成功

**nvidia-smi** #输入指令查看显卡信息

如果出现以下信息，说明安装成功

```
baochun@baochun: ~  
baochun@baochun:~$ nvidia-smi  
Sat Nov 24 11:17:19 2018  
+-----+  
| NVIDIA-SMI 390.67                  Driver Version: 390.67 |  
+-----+-----+  
| GPU Name Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |  
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |  
+-----+-----+  
| 0  GeForce GTX 1070    Off | 00000000:01:00.0 Off |          N/A         |  
| 0%   34C   P0      32W / 151W | 0MiB / 8119MiB |           0%    Default |  
+-----+-----+  
+-----+  
| Processes:                          GPU Memory |  
| GPU       PID   Type   Process name                      Usage |  
+-----+  
| No running processes found |  
+-----+  
baochun@baochun:~$
```

## 4.安装 CUDA

(1)进入 [CUDA9.0 官网](#)，下载 cuda9.0，如下图所示。

### CUDA Toolkit 9.0 Downloads

Select Target Platform ⓘ  
Click on the green buttons that describe your target platform. Only supported platforms will be shown.  
  
Operating System: Windows, Linux, Mac OSX  
Architecture ⓘ: x86\_64, ppc64le  
Distribution: Fedora, OpenSUSE, RHEL, CentOS, SLES, Ubuntu  
Version: 17.04, 16.04  
Installer Type ⓘ: runfile [local], deb [local], deb [network], cluster [local]

Download Installers for Linux Ubuntu 16.04 x86\_64  
The base installer is available for download below.  
There are 4 patches available. These patches require the base installer to be installed first.  
  
Base Installer: Download (1.6 GB) ⬆️  
Installation Instructions:  
1. Run `sudo sh cuda\_9.0.176\_384.81\_linux.run`  
2. Follow the command-line prompts  
  
Patch 1 (Released Jan 25, 2018): Download (112.8 MB) ⬆️

Related Links  
CUDA Quick Start Guide  
Release Notes  
EULA  
CUDA Toolkit Overview  
Installer Checksums  
Open Source Packages  
Legacy CUDA Toolkits

下载以后的.run 文件为 **cuda\_9.0.176\_384.81\_linux.run**。

(2)给 cuda 文件取消权限

**sudo chmod 777 cuda\_9.0.176\_384.81\_linux.run**

### (3)执行安装程序

```
sudo sh cuda_9.0.176_384.81_linux.run
```

### (4)配置环境变量

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

在文件末尾添加：

```
export PATH=/usr/local/cuda-9.0/bin:$PATH
```

```
export LD_LIBRARY_PATH=/usr/local/cuda9.0/lib64:$LD_LIBRARY_PATH
```

### (5)测试 cuda，最后输出 pass，则为正确安装。

```
cd /usr/local/cuda-9.0/samples/1_Utilities/deviceQuery
```

```
sudo make
```

```
sudo ./deviceQuery
```

## 5.安装 cuDNN

(1)把 cudnn 压缩包下载下来后，按照如下步骤安装即可：(版本为 cuDNN7.4.1)网址：

<https://developer.nvidia.com/rdp/cudnn-download>（需要注册一下账号）

### cuDNN Download

NVIDIA cuDNN is a GPU-accelerated library of primitives for deep neural networks.

☒ I Agree To the Terms of the [cuDNN Software License Agreement](#)

Note: Please refer to the [Installation Guide](#) for release prerequisites, including supported GPU architectures and compute capabilities, before downloading.

For more information, refer to the cuDNN Developer Guide, Installation Guide and Release Notes on the [Deep Learning SDK Documentation](#) web page.

[Download cuDNN v7.4.1 \(Nov 8, 2018\), for CUDA 10.0](#)

[Download cuDNN v7.4.1 \(Nov 8, 2018\), for CUDA 9.2](#)

[Download cuDNN v7.4.1 \(Nov 8, 2018\), for CUDA 9.0](#)

#### Library for Windows, Mac, Linux and Ubuntu (x86\_64 architecture)

[cuDNN Library for Windows 7](#)

[cuDNN Library for Windows 10](#)

[cuDNN Library for Linux](#)

[cuDNN Runtime Library for Ubuntu16.04 \(Deb\)](#)

[cuDNN Developer Library for Ubuntu16.04 \(Deb\)](#)

[cuDNN Code Samples and User Guide for Ubuntu16.04 \(Deb\)](#)

[cuDNN Runtime Library for Ubuntu14.04 \(Deb\)](#)

[cuDNN Developer Library for Ubuntu14.04 \(Deb\)](#)

### (2)删除原来的 cudnn 系统路径下的一些文件

```
sudo rm -rf /usr/local/cuda/include/cudnn.h
```

```
sudo rm -rf /usr/local/cuda/lib64/libcudnn*
```

#这里\*是通配符，libcudnn\*指的是名字中带有 libcudnn 的所有文件

(3)安装刚才解压的 cudnn 版本，在终端 cd 到刚解压的 cuda 文件夹，然后继续输入下面两个指令，这两个指令相当于把解压后的 cuda 文件夹下的一些文件拷到系统路径下面

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

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

#这里\*是通配符，lib\*指的是名字中带有lib的所有文件

(4)在系统路径下建立软链接（解压出来的lib64下面有3个so文件。分别是libcudnn.so和libcudnn.so.7以及libcudnn.so.7.4.1文件。并且这3个点so文件大小都一样。其实都是软连接！libcudnn.so链接到libcudnn.so.7，而libcudnn.so.7又链接到libcudnn.so.7.4.1。真正的文件只有libcudnn.so.7.4.1）

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

```
sudo chmod +r libcudnn.so.7.4.1
```

```
sudo ln -sf libcudnn.so.7.4.1 libcudnn.so.7
```

```
sudo ln -sf libcudnn.so.7 libcudnn.so
```

接下来设置环境变量：

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

在打开的文件中加入如下两句话

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

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

保存后，使环境变量立即生效，

```
source /etc/profile
```

## 6.安装 caffe

(1)首先安装一般依赖项

```
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
```

(2)再安装 BLAS 依赖项

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

(3)下载 caffe 的 github 包，解压为 caffe 文件夹，然后进入到源码目录

```
cd caffe
```

(4)编译 Caffe

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

**注意：**Makefile 和 Makefile.config 文件需要修改的地方：

1) Makefile

```

INCLUDE_DIRS += $(BUILD_INCLUDE_DIR) ./src ./include
ifneq ($(CPU_ONLY), 1)
    INCLUDE_DIRS += $(CUDA_INCLUDE_DIR)
    LIBRARY_DIRS += $(CUDA_LIB_DIR)
    LIBRARIES := cudart cublas curand
endif

LIBRARIES += glog gflags protobuf boost_system boost_filesystem m hdf5_serial hl hdf5_serial

# handle IO dependencies
USE_LEVELDB ?= 1

```

## 2) Makefile.config

需要使用 cuDNN 的朋友可以将 Makefile.config 中 USE\_CUDNN := 1 这一行之前的#号注释去掉

```

## Refer to http://caffe.berkeleyvision.org/installation.html
# Contributions simplifying and improving our build system are welcome!

# cuDNN acceleration switch (uncomment to build with cuDNN).
USE_CUDNN := 1

```

用到了 opencv, 记得打开 opencv 并选择 opencv 的版本

```

# uncomment to disable IO dependencies and corresponding data layers
USE_OPENCV := 1
# USE_LEVELDB := 0
# USE_LMDB := 0
# This code is taken from https://github.com/shir0/caffe-android-lib
# USE_HDF5 := 0

# uncomment to allow MDB_NOLOCK when reading LMDB files (only if necessary)
#     You should not set this flag if you will be reading LMDBs with any
#     possibility of simultaneous read and write
# ALLOW_LMDB_NOLOCK := 1

# Uncomment if you're using OpenCV 3
OPENCV_VERSION := 3

```

同时, 为了匹配 cuda9.0 的计算能力, 把 Makefile.config 中 CUDA\_ARCH 中的前两行去掉, 如下图所示(保留也行, 编译的时候会弹出警告)

```

# CUDA architecture setting: going with all of them.
# For CUDA < 6.0, comment the *_50 through *_61 lines for compatibility.
# For CUDA < 8.0, comment the *_60 and *_61 lines for compatibility.
# For CUDA >= 9.0, comment the *_20 and *_21 lines for compatibility.
CUDA_ARCH := -gencode arch=compute_30,code=sm_30 \
              -gencode arch=compute_35,code=sm_35 \
              -gencode arch=compute_50,code=sm_50 \
              -gencode arch=compute_52,code=sm_52 \
              -gencode arch=compute_60,code=sm_60 \
              -gencode arch=compute_61,code=sm_61 \
              -gencode arch=compute_61,code=compute_61

```

将 INCLUDE\_DIRS 和 LIBRARY 的地址改为以下地址。

```

# Whatever else you find you need goes here.
INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include /usr/include/hdf5/serial
LIBRARY_DIRS := $(PYTHON_LIB) /usr/local/lib /usr/lib /usr/lib/aarch64-linux-gnu /usr/lib/aarch64-linux-gnu/hdf5/serial

```

## (9)最后编译

**make all -j8**

(如果在编译过程中有错误, 可以先 **make clean**, 然后 **make all -j8**)

最后再测试一下:

**make test**



如果 caffe 正常配置好了就会出现以下界面，并且不会报错。

```
nvidia@tegra-ubuntu: ~/caffe
LD .build_release/src/caffe/test/test_common.o
LD .build_release/src/caffe/test/test_deconvolution_layer.o
LD .build_release/src/caffe/test/test_batch_reindex_layer.o
LD .build_release/src/caffe/test/test_slice_layer.o
LD .build_release/src/caffe/test/test_lrn_layer.o
LD .build_release/src/caffe/test/test_softmax_with_loss_layer.o
LD .build_release/src/caffe/test/test_split_layer.o
LD .build_release/src/caffe/test/test_sigmoid_cross_entropy_loss_layer.o
LD .build_release/src/caffe/test/test_protobuf.o
LD .build_release/src/caffe/test/test_image_data_layer.o
LD .build_release/src/caffe/test/test_mvn_layer.o
LD .build_release/src/caffe/test/test_power_layer.o
LD .build_release/src/caffe/test/test_concat_layer.o
LD .build_release/src/caffe/test/test_flatten_layer.o
LD .build_release/src/caffe/test/test_reduction_layer.o
LD .build_release/src/caffe/test/test_filler.o
LD .build_release/src/caffe/test/test_batch_norm_layer.o
LD .build_release/src/caffe/test/test_data_layer.o
LD .build_release/src/caffe/test/test_tanh_layer.o
LD .build_release/src/caffe/test/test_util_blas.o
LD .build_release/src/caffe/test/test_hinge_loss_layer.o
LD .build_release/src/caffe/test/test_rnn_layer.o
LD .build_release/src/caffe/test/test_lstm_layer.o
LD .build_release/src/caffe/test/test_argmax_layer.o
LD .build_release/src/caffe/test/test_embed_layer.o
LD .build_release/src/caffe/test/test_threshold_layer.o
LD .build_release/src/caffe/test/test_stochastic_pooling.o
LD .build_release/src/caffe/test/test_solver.o
LD .build_release/src/caffe/test/test_dummy_data_layer.o
LD .build_release/src/caffe/test/test_io.o
LD .build_release/src/caffe/test/test_softmax_layer.o
LD .build_release/src/caffe/test/test_hdf5data_layer.o
LD .build_release/src/caffe/test/test_gradient_based_solver.o
LD .build_release/src/caffe/test/test_solver_factory.o
LD .build_release/src/caffe/test/test_syncedmem.o
LD .build_release/src/caffe/test/test_inner_product_layer.o
LD .build_release/src/caffe/test/test_euclidean_loss_layer.o
LD .build_release/src/caffe/test/test_maxpool_dropout_layers.o
LD .build_release/src/caffe/test/test_upgrade_proto.o
LD .build_release/src/caffe/test/test_multinomial_logistic_loss_layer.o
LD .build_release/src/caffe/test/test_contrastive_loss_layer.o
LD .build_release/src/caffe/test/test_net.o
LD .build_release/src/caffe/test/test_infogain_loss_layer.o
LD .build_release/src/caffe/test/test_crop_layer.o
LD .build_release/src/caffe/test/test_pooling_layer.o
LD .build_release/src/caffe/test/test_convolution_layer.o
LD .build_release/src/caffe/test/test_filter_layer.o
LD .build_release/src/caffe/test/test_bias_layer.o
LD .build_release/src/caffe/test/test_internal_thread.o
LD .build_release/src/caffe/test/test_layer_factory.o
LD .build_release/src/caffe/test/test_scale_layer.o
LD .build_release/src/caffe/test/test_db.o
LD .build_release/src/caffe/test/test_spp_layer.o
LD .build_release/src/caffe/test/test_math_functions.o
LD .build_release/src/caffe/test/test_random_number_generator.o
LD .build_release/src/caffe/test/test_benchmark.o
LD .build_release/src/caffe/test/test_blob.o
LD .build_release/src/caffe/test/test_reshape_layer.o
LD .build_release/cuda/src/caffe/test/test_4m2col_kernel.o
nvidia@tegra-ubuntu: ~/caffe$
```

(注：由于安装成功后图未保存，这是我配置 TX2 的 caffe 成功时候的图，命令提示是一样的。)

## 7.安装 Pytorch

### (1)下载 PyTorch

由于直接用指令下载存在网络问题，所以我选择手动下载安装。首先下载所需版本 PyTorch。在 PyTorch 主页上点击“**Click here for previous versions of PyTorch**”，如图右下角所示。地址：<https://pytorch.org/get-started/previous-versions/>

## START LOCALLY

Select your preferences and run the install command. Please ensure that you have met the prerequisites below (e.g., `numpy`), depending on your package manager. Anaconda is our recommended package manager since it installs all dependencies. You can also [install previous versions of PyTorch](#). Note that LibTorch is only available for C++.

PyTorch Build	Stable		Preview		
Your OS	Linux		Mac	Windows	
Package	Conda	Pip	LibTorch	Source	
Language	Python 2.7	Python 3.5	Python 3.6	Python 3.7	C++
CUDA	8.0	9.0	9.2	None	

- [cu91/torch-0.3.1-cp27-cp27m-linux\\_x86\\_64.whl](#)

## PyTorch Linux binaries compiled with CUDA 9.0

- [cu90/torch-0.4.0-cp36-cp36m-linux\\_x86\\_64.whl](#)
- [cu90/torch-0.4.0-cp35-cp35m-linux\\_x86\\_64.whl](#)
- [cu90/torch-0.4.0-cp27-cp27mu-linux\\_x86\\_64.whl](#)
- [cu90/torch-0.4.0-cp27-cp27m-linux\\_x86\\_64.whl](#)
- [cu90/torch-0.3.1-cp36-cp36m-linux\\_x86\\_64.whl](#)
- [cu90/torch-0.3.1-cp35-cp35m-linux\\_x86\\_64.whl](#)
- [cu90/torch-0.3.1-cp27-cp27mu-linux\\_x86\\_64.whl](#)
- [cu90/torch-0.3.1-cp27-cp27m-linux\\_x86\\_64.whl](#)
- [cu90/torch-0.3.0.post4-cp36-cp36m-linux\\_x86\\_64.whl](#)

找到所需版本，如上图所示，下载 **torch-0.4.0-cp35-cp35m-linux\_x86\_64.whl**。其中，**0.4.0** 表示 **PyTorch** 版本，**cp35** 表示支持 **Python3.5**，**cuda90** 表示支持 **GPU** 版本，电脑有 GPU 的话须选择该项。

## (2)安装 PyTorch

进入 PyTorch 的下载目录，使用 `pip3` 命令安装：

```
pip3 install torch-0.4.0-cp35-cp35m-linux_x86_64.whl
```

## (3)安装 torchvision

安装 `torchvision` 比较简单，可直接使用 `pip3` 命令安装：

```
pip3 install torchvision
```

至此，`pytorch` 和 `torchvision` 已安装完毕。

## (4)更新 numpy

安装成功 `pytorch` 和 `torchvision` 后，输入：

```
python3
```

```
>>>import torch
```

可能会出现报错的情况，如下所示：

**ImportError: numpy.core.multiarray failed to import**

这是因为 numpy 的版本需要更新，直接使用 pip 更新 numpy：

```
pip3 install numpy
```

至此，PyTorch 安装成功：

```
baochun@baochun:~$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> import torchvision
>>> print(torch.cuda.is_available())
True
>>> █
```

最后验证 GPU 也可以用。

## 8.安装 tensorflow

### (1)装 python 对应版本的 pip 和依赖包

若 python 版本为 2.7，则输入如下命令：

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

若 python 版本为 3.x，则输入如下命令：

```
sudo apt-get install python3-pip python3-dev
```

### (2)升级 pip 版本

在装 tensorflow 之前，不管是不是最新的 pip 版本，都要更新一下，具体命令如下：

python 2.7 版本： **sudo pip install --upgrade pip**

python 3.x 版本： **sudo pip3 install --upgrade pip**

### (3)安装 TensorFlow

TensorFlow 可以安装 CPU 和 GPU 两种版本，

CPU 版本安装命令如下：

python 2.7 版本： **sudo pip install tensorflow**

python 3.x 版本： **sudo pip3 install tensorflow**

GPU 版本安装命令如下：

python 2.7 版本： **sudo pip install tensorflow-gpu**

python 3.x 版本： **sudo pip3 install tensorflow-gpu**

### (4)测试安装结果

进入 python 编译环境，导入 TensorFlow，做一个简单的加法运算，如下图所示。

```
baochun@baochun:~$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> session = tf.Session()
2018-11-24 12:24:27.944478: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions
that this TensorFlow binary was not compiled to use: AVX2 FMA
2018-11-24 12:24:28.025834: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:897] successful NUMA node re
ad from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2018-11-24 12:24:28.026466: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1405] Found device 0 with proper
ties:
name: GeForce GTX 1070 major: 6 minor: 1 memoryClockRate(GHz): 1.683
pciBusID: 0000:01:00.0
totalMemory: 7.93GiB freeMemory: 7.83GiB
2018-11-24 12:24:28.026483: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1484] Adding visible gpu devices
: 0
2018-11-24 12:24:28.223931: I tensorflow/core/common_runtime/gpu/gpu_device.cc:965] Device interconnect StreamE
xecutor with strength 1 edge matrix:
2018-11-24 12:24:28.223966: I tensorflow/core/common_runtime/gpu/gpu_device.cc:971] 0
2018-11-24 12:24:28.223972: I tensorflow/core/common_runtime/gpu/gpu_device.cc:984] 0: N
2018-11-24 12:24:28.224132: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1097] Created TensorFlow device
(/job:localhost/replica:0/task:0/device:GPU:0 with 7560 MB memory) -> physical GPU (device: 0, name: GeForce GT
X 1070, pci bus id: 0000:01:00.0, compute capability: 6.1)
>>> a = tf.constant(23)
>>> b = tf.constant(12)
>>> print(session.run(a+b))
35
>>>
```

## 9. 安装 Keras

```
sudo pip3 install tensorflow-gpu # GPU 加速版
```

```
sudo pip3 install keras
```

在终端中验证是否安装成功：

```
python3
```

```
>>> import tensorflow
```

```
>>> import keras
```

如果不报错，即配置成功！

```
baochun@baochun:~$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow
>>> import keras
Using TensorFlow backend.
>>>
```