

<http://guanghan.info/blog/en>

## Ning

# Build Personal Deep Learning Rig: GTX 1080 + Ubuntu 16.04 + CUDA 8.0RC + CuDnn 7 + Tensorflow/Mxnet/Caffe/Darknet (<http://guanghan.info/blog/en/my-works/building-our-personal-deep-learning-rig-gtx-1080-ubuntu-16-04-cuda-8-0rc-cudnn-7-tensorflowmxnetcaffedarknet/>)

16/07/20 at 10.13pm / by Guanghan Ning (<http://guanghan.info/blog/en/author/admin/>) / 18 Comments (<http://guanghan.info/blog/en/my-works/building-our-personal-deep-learning-rig-gtx-1080-ubuntu-16-04-cuda-8-0rc-cudnn-7-tensorflowmxnetcaffedarknet/>)

My intern at TCL is over soon. Before going back to the campus for graduation, I have decided to build myself a personal deep learning rig. I guess I cannot really rely on the machines either in the company or in the lab, because ultimately the workstation is not mine, and the development environment may be messed up (It already happened once) . With a personal rig, I can conveniently use teamviewer to login my deep learning workstation at any time. And I got the chance to build everything from scratch.

In this post, I will go through the whole process of the building a deep learning PC, including the hardware and the software. Upon sharing it with you, I hope it will be helpful to researchers and engineers with the same needs. Since I am building the rig with **GTX 1080, Ubuntu 16.04, CUDA 8.0RC, CuDnn 7**, everything is pretty up-to-date. Here is an overview of this article:

### Hardware

- 1. Pick Parts
- 2. Build the workstation

### Software

## Categories

My thoughts

(<http://guanghan.info/blog/en/category/my-thoughts/>)

My Works

(<http://guanghan.info/blog/en/category/my-works/>)

## Archives

July 2016

(<http://guanghan.info/blog/en/2016/07/>)

March 2016

(<http://guanghan.info/blog/en/2016/03/>)

February 2016

(<http://guanghan.info/blog/en/2016/02/>)

December 2015

(<http://guanghan.info/blog/en/2015/12/>)

March 2015

2017/12/4 下午2:28  
(<http://guanghan.info>)

- 3. Operating System Installation

- Preparing bootable installation USB drives
- Build systems

- 4. Deep Learning Environment Installation

- Remote Control
  - teamviewer
- Bundle Management
  - anaconda
- Development Environment
  - python IDE
- GPU-optimized Environment
  - CUDA
  - CuDnn
- Deep Learning Frameworks
  - Tensorflow
  - Mxnet
  - Caffe
  - Darknet

- 5. Docker for Out-of-the-Box Deep Learning Environment

- Install Docker
- Install NVIDIA-Docker
- Download Deep Learning Docker Images
- Share Data between Host and Container
- Learn Simple Docker Commands

## Links

Author' Info

(<http://guanghan.info/>)

## Preface

I am sharing thoughts with you in this blog.

If something echoes back, I feel most lucky;

if it is to some extent understood... more than satisfied;

if it drains down into the ground, I have nothing to regret.

## Hardware:

### 1. Pick Parts

I recommend using **PcPartPicker** (<https://pcpartpicker.com/>) to pick your parts. It helps you find the source where you can buy your part with the lowest price available, and it checks the compatibility of the selected parts for you. They also have a **youtube channel** ([https://www.youtube.com/channel/UCXQpf5LtNI7dbmZlxFIO\\_w](https://www.youtube.com/channel/UCXQpf5LtNI7dbmZlxFIO_w)) where they offer videos that demonstrate the building process.

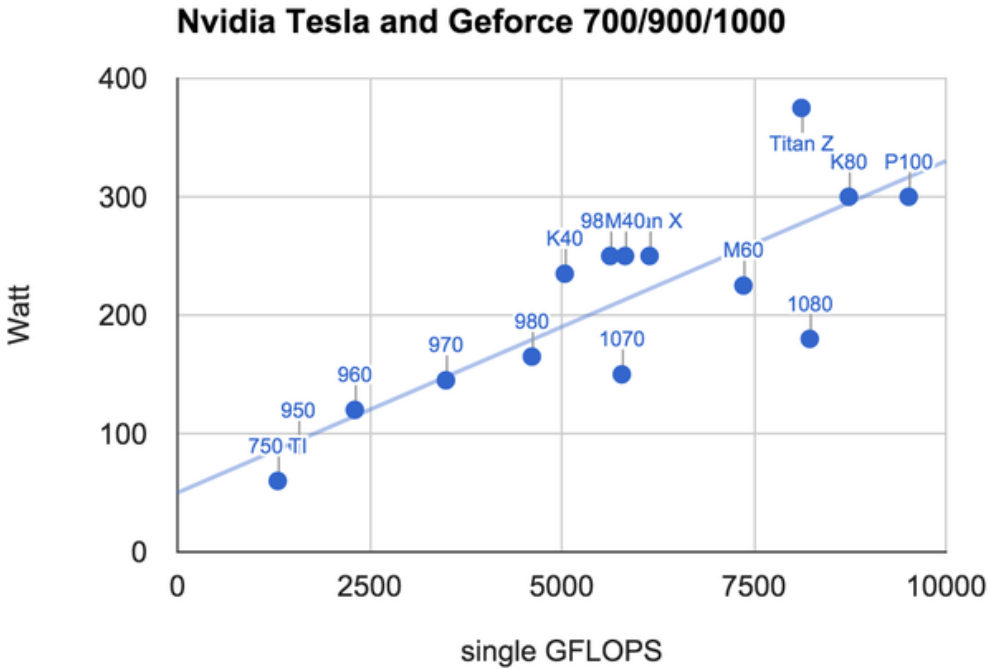
In my case, I used their build article as reference, and created a build list myself, which can be found [[here \(https://pcpartpicker.com/user/quietning/saved/#view=YP6v6h\)](https://pcpartpicker.com/user/quietning/saved/#view=YP6v6h)]. Here are the parts that I used to build the workstation.



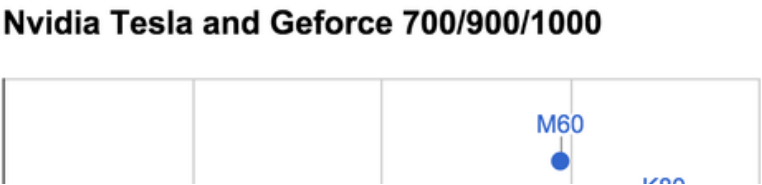


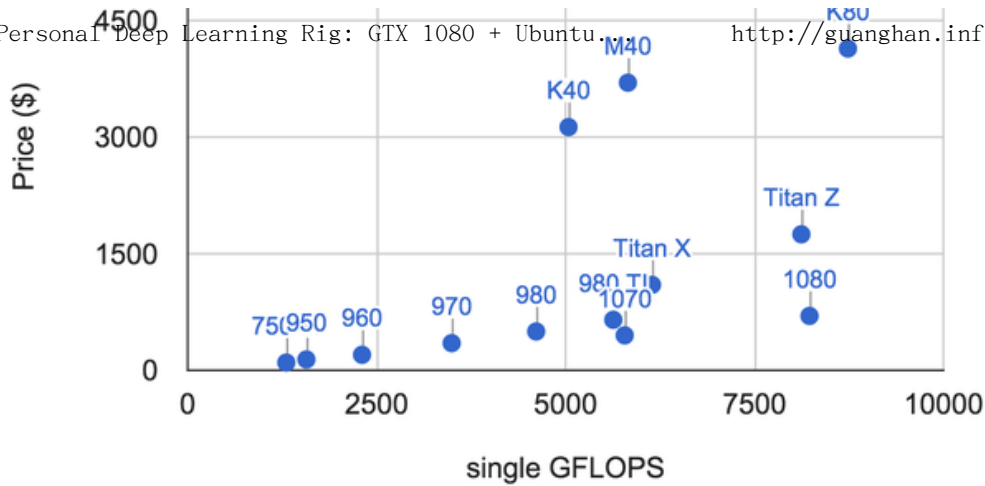
([http://guanghan.info/blog/en/wp-content/uploads/2016/07/IMG\\_20160707\\_191958-Copy.jpg](http://guanghan.info/blog/en/wp-content/uploads/2016/07/IMG_20160707_191958-Copy.jpg))

Since we are doing deep learning research, a good GPU is necessary. Therefore, I choose the recently released GTX 1080. It was quite hard to buy, but if you notice the bundles in newegg, some people are gathering this to sell in [GPU + motherboard] or [GPU + Power] bundles. Market, you know. It is better buying the bundle than buying it at a raised price, though. Anyway, a good GPU will make the training or finetuning process much faster. Here are some figures to show the advantage of GTX 1080 over some other GPUs, with respect to performance, price, and power efficiency (saves you electricity daily and the money to buy the appropriate PC power supply).



([http://guanghan.info/blog/en/wp-content/uploads/2016/07/gtx\\_1.png](http://guanghan.info/blog/en/wp-content/uploads/2016/07/gtx_1.png))





([http://guanghan.info/blog/en/wp-content/uploads/2016/07/gtx\\_2.png](http://guanghan.info/blog/en/wp-content/uploads/2016/07/gtx_2.png))

Note that GTX 1080 has only 8GB memory, compared to 12GB of TITAN X. You may be richer or more generous to yourself, therefore considering using stacked GPUs. Then remember to choose another motherboard that has more PCIs.

## 2. Build from Parts

As of building from the parts, I followed the tutorial of **[this video]** (<https://www.youtube.com/watch?v=bHF2eEnXP6I>). Although the parts are slightly different, the building process is quite similar. I have no previous experience building by my own, but with this tutorial I was able to make it work within 3 hours. (It should take you less time, but I was extremely cautious, you know.)



# Software:

## 3. Installation of Operating Systems

It is very common to use Ubuntu for deep learning research. But sometimes you may need another operating system working as well. For example, if you are also a VR developer, having a GTX 1080, you may want a Win10 for VR development with Unity or whatsoever. Here I introduce the installation of both Win10 and Ubuntu. If you are only interested in the Ubuntu installation, you can skip installing windows.

### 3.1 Preparing bootable installation USB drives

It is very convenient to install operating systems with USB disks, as we all have them. Because the USB disks will be formatted, you won't want that happen to your portable hard disk. Or if you have writable DVDs, you can use them to install operating systems and save them for future use, if you can find them again by then.

Since it is well illustrated in the official website, you can go to [Windows 10 page (<https://www.microsoft.com/en-us/software-download/windows10/>)] to learn how to make the USB drive. As of Ubuntu, you can similarly download the ISO and create USB installation media or burn it to a DVD. If you are now using Ubuntu system, follow [this tutorial (<http://www.ubuntu.com/download/desktop/create-a-usb-stick-on-ubuntu>)] from Ubuntu official website. If you are current using Windows, follow [this tutorial (<http://www.ubuntu.com/download/desktop/create-a-usb-stick-on-windows>)] instead.

### 3.2 Installing Systems

It is highly recommended that you install windows first for a dual-system installation. I will skip win10 installation as detailed guide can be found here: [Windows 10 page (<https://www.microsoft.com/en-us/software-download/windows10/>)] . One thing to note is that you will need the activation key. You can find the tag on the bottom of your laptop, if it has been installed windows 7 or windows 10 upon purchasing.

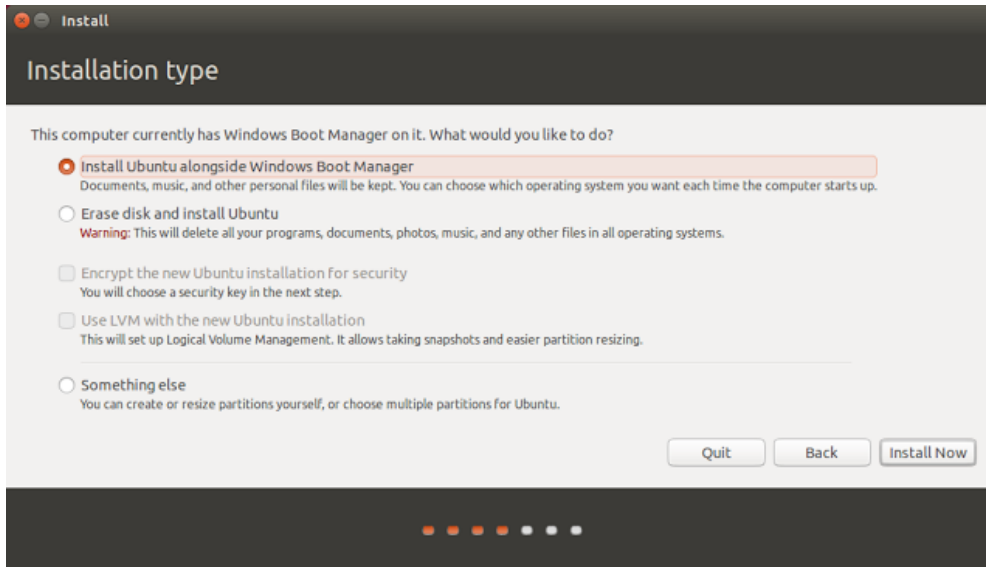
Installing Ubuntu16.04 was a little tricky for me, which was kind of a surprise. It was mainly because I did not have the GTX 1080 driver pre-installed at the very beginning. I will share my story with you, in case you encounter the same problems.

#### Installing Ubuntu :

First things first, insert the boot USB for installation. Nothing is showing on my LG screen, except that it says frequency is too high. But the screen is okay, as is tested on another laptop. I tried to connect the PC



with a TV which was showing, but only the desktop with no tool panel. I figured out it was the problem of the NVIDIA driver. So I went to BIOS and set the integrated graphics as default and restart. Remember to switch the HDMI from the port on GTX1080 to that on the motherboard. Now the display works well. I successfully installed Ubuntu following its prompt guides.



([http://guanghan.info/blog/en/wp-content/uploads/2016/07/installing\\_ubuntu.png](http://guanghan.info/blog/en/wp-content/uploads/2016/07/installing_ubuntu.png))

In order to use GTX1080, go to [this page (<http://www.nvidia.com/download/driverResults.aspx/104284/en-us>)] to get the NVIDIA display driver for Ubuntu. Upon installing this driver, make sure that GTX1080 is on the motherboard.

Now it shows “You appear to be running an X server.. “. I followed this link (<http://askubuntu.com/questions/149206/how-to-install-nvidia-ru>) to solve this problem and installed the driver. I quote it here:

- Make sure you are logged out.
- Hit CTRL+ALT+F1 and login using your credentials.
- Kill your current X server session by typing `sudo service lightdm stop` or `sudo stop lightdm`
- Enter runlevel 3 by typing `sudo init 3` and install your \*.run file.
- You might be required to reboot when the installation finishes. If not, run `sudo service lightdm start` or `sudo start lightdm` to start your X server again.

After installing the driver, we can now restart and set the GTX1080 as default in BIOS. We are good to go.

Some other small problems I encountered are listed here, in case they are helpful:

- Problem: When I restart, I couldn't find the option to choose windows.
- Solver: In ubuntu, **sudo gedit /boot/grub/grub.cfg**, add following lines:

```
1. menuentry 'Windows 10'
```

```
set root="hd0:msdos1"
```

```
3. chainloader +1
```

```
4. }
```

- Problem: Ubuntu does not support wireless adapter Belkin N300, which is commonly sold in Bestbuy.
- Solver: Follow instructions in this link (<https://ubuntuforums.org/showthread.php?t=1515747>), the problem will be solved.
- Problem: Upon installing teamviewer, it says "dependencies not met"
- Solver: Refer to this link (<http://askubuntu.com/questions/362951/installed-teamviewer-using-a-64-bits-system-but-i-get-a-dependency-error/363083>).

## 4. Deep Learning Environment

### 4.1 Installation of Remote Control (TeamViewer):

```
dpkg -i teamviewer_11.0.xxxxx_i386.deb
```

### 4.2 Installation of Package Management System (Anaconda):

Anaconda is an easy-to-install free package manager, environment manager, Python distribution, and collection of over 720 open source packages offering free community support. It can be used to create virtual environments, where each environment will not mess up with each other. It is helpful when we use different deep learning frameworks at the same time, and the configurations are different. Using it to install packages is convenient as well. It can be easily installed, follow this (<https://docs.continuum.io/anaconda/install#linux-install>).

Some commands to start using virtual environment:

- source activate [virtualenv]
- source deactivate

### 4.3 Installation of Development Environment (Python IDE):

#### 4.3.1 Spyder vs Pycharm?

Spyder

- Advantage : matlab-like , easy to review intermediate results.

Pycharm :

- Advantage : modular coding , more complete IDE for web development frameworks and cross-platform.

In my personal philosophy, I regard them to be merely tools. Each tool will be used when it comes in handy. I will use IDEs for the construction of the backbone for the project. For example, use pycharm for the framework construction. After that, I will just modify code with VIM. It is not that VIM is so powerful and showy, but because it is the single text editor that I want to really master. As of text editors, there is no need we should master two. For special occasions, where we need to frequently

check IO, directories, etc, we might want to use spyder instead

### 4.3.2 Installation:

- spyder :
  - You do not need to install spyder, as is included in anaconda.
- pycharm
  - Download from the official website (<https://www.jetbrains.com/pycharm/>). Just unzip.
  - Set anaconda to be the project interpreter for pycharm, dealing with package management. Follow this ([https://docs.continuum.io/anaconda/ide\\_integration#pycharm](https://docs.continuum.io/anaconda/ide_integration#pycharm)).
- vim
  - sudo apt-get install vim
  - The configuration that I currently use: Github (<https://github.com/Guanghan/VimIDE>)
- Git
  - sudo apt install git
  - git config --global user.name "Guanghan Ning"
  - git config --global user.email "guanghan.ning@gmail.com"
  - git config --global core.editor vim
  - git config --list

## 4.4 Installation of GPU-Optimized Computing Environment (CUDA and CuDNN)

### 4.4.1 CUDA

- Install CUDA 8.0 RC (<https://developer.nvidia.com/cuda-release-candidate-download>)
  - There are two reasons to choose the 8.0 version over 7.5:
    - CUDA 8.0 will give a performance gain for GTX1080 (Pascal), compared to CUDA 7.5.
    - It seems that ubuntu 16.04 does not support CUDA 7.5 because you cannot find it to download on the official website. Therefore CUDA 8.0 is the only choice.
- CUDA starter Guide ([http://developer.download.nvidia.com/compute/cuda/8.0/secure/rc1/docs/sidebar/CUDA\\_Quick\\_Start\\_Guide.pdf?autho=1468531210\\_b9ce6047a5b7cb575fde7a6ffd6ad729&file=CUDA\\_Quick\\_Start\\_Guide.pdf](http://developer.download.nvidia.com/compute/cuda/8.0/secure/rc1/docs/sidebar/CUDA_Quick_Start_Guide.pdf?autho=1468531210_b9ce6047a5b7cb575fde7a6ffd6ad729&file=CUDA_Quick_Start_Guide.pdf))
- CUDA Installer Guide ([http://developer.download.nvidia.com/compute/cuda/8.0/secure/rc1/docs/sidebar/CUDA\\_Installation\\_Guide\\_Linux.pdf?autho=1468531209\\_7b8d97cef95dffcb18e2fecb656b8a85&file=CUDA\\_Installation\\_Guide\\_Linux.pdf](http://developer.download.nvidia.com/compute/cuda/8.0/secure/rc1/docs/sidebar/CUDA_Installation_Guide_Linux.pdf?autho=1468531209_7b8d97cef95dffcb18e2fecb656b8a85&file=CUDA_Installation_Guide_Linux.pdf))
  - sudo sh cuda\_8.0.27\_linux.run
  - Follow the command-line prompts
  - As part of the CUDA environment, you should add the following in the ~/.bashrc file of your home folder.
    - export CUDA\_HOME=/usr/local/cuda-8.0
    - export LD\_LIBRARY\_PATH=\${CUDA\_HOME}/lib64
    - PATH=\${CUDA\_HOME}/bin:\${PATH}



## ■ export PATH

- check if CUDA is installed (Remember to restart the terminal):

- nvcc -version

#### 4.4.2 Cudnn ( CUDA Deep Learning Library )

- install cudnn (<https://developer.nvidia.com/cudnn>)
  - Version : Cudnn v5.0 for CUDA 8.0RC
- User Guide ([http://developer.download.nvidia.com/compute/machine-learning/cudnn/secure/v5/prod/cudnn\\_library.pdf?auth=1468531134\\_f12a2097cf581a5659608091857f7326&file=cudnn\\_library.pdf](http://developer.download.nvidia.com/compute/machine-learning/cudnn/secure/v5/prod/cudnn_library.pdf?auth=1468531134_f12a2097cf581a5659608091857f7326&file=cudnn_library.pdf))
- Install Guide ([http://developer.download.nvidia.com/compute/machine-learning/cudnn/secure/v5/prod/cudnn\\_install-8.txt?auth=1468531135\\_e4d639fa518ee9ba983b48e36602f028&file=cudnn\\_install-8.txt](http://developer.download.nvidia.com/compute/machine-learning/cudnn/secure/v5/prod/cudnn_install-8.txt?auth=1468531135_e4d639fa518ee9ba983b48e36602f028&file=cudnn_install-8.txt))
  - Choice one: (Add CuDNN path to environment variables)
    - Extract folder "cuda"
    - cd <installpath>
    - export LD\_LIBRARY\_PATH=`pwd`:LD\_LIBRARY\_PATH
  - Choice two: (Copy the files of CuDNN to CUDA folder. If CUDA is working alright, it will automatically find CUDNN by relative path)
    - tar xvfz cudnn-8.0.tgz
    - cd cudnn
    - sudo cp include/cudnn.h /usr/local/cuda/include
    - sudo cp lib64/libcudnn\* /usr/local/cuda/lib64
    - sudo chmod a+r /usr/local/cuda/include/cudnn.h /usr/local/cuda/lib64/libcudnn\*

#### 4.5 Installation of Deep Learning Frameworks :

##### 4.5.1 Tensorflow / keras

- Install tensorflow first
  - Install with anaconda ([https://github.com/tensorflow/tensorflow/blob/master/tensorflow/g3doc/get\\_started/os\\_setup.md#anaconda-installation](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/g3doc/get_started/os_setup.md#anaconda-installation))
    - conda create -n tensorflow python=3.5
  - Install Tensorflow using Pip in the environment ([https://www.tensorflow.org/versions/r0.9/get\\_started/os\\_setup.html#anaconda-installation](https://www.tensorflow.org/versions/r0.9/get_started/os_setup.html#anaconda-installation)) (It does NOT supports cuda 8.0 at the moment. I will update this when binaries for CUDA 8.0 come out)
    - source activate tensorflow
    - sudo apt install python3-pip
    - export TF\_BINARY\_URL=[https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow-0.9.0-cp35-cp35m-linux\\_x86\\_64.whl](https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow-0.9.0-cp35-cp35m-linux_x86_64.whl)
    - pip3 install --upgrade \$TF\_BINARY\_URL
  - Install Tensorflow directly from source ([https://www.tensorflow.org/versions/r0.9/get\\_started/os\\_setup.html#installing-from-sources](https://www.tensorflow.org/versions/r0.9/get_started/os_setup.html#installing-from-sources))
    - install bazel (<http://www.bazel.io/docs/install.html>)
      - install jdk 8
      - uninstall jdk 9
    - sudo apt-get install python-numpy swig python-dev

- **configure**
  - **build with bazel**
    - `bazel build -c opt --config=cuda //tensorflow/cc:tutorials_example_trainer`
    - `bazel-bin/tensorflow/cc/tutorials_example_trainer --use_gpu`
- **Install keras**
  - download it at: <https://github.com/fchollet/keras/tree/master/keras> (<https://github.com/fchollet/keras/tree/master/keras>)
  - cd to the Keras folder and run the install command:
    - `sudo python setup.py install`
  - Change the default backend (<http://keras.io/backend/>) from theano to tensorflow
- **Use conda to activate/deactivate the virtual environment**
  - `source activate tensorflow`
  - `source deactivate`

## 4.5.2 Mxnet

- **Create a virtual environment for Mxnet**
  - `conda create -n mxnet python=2.7`
  - `source activate mxnet`
- **Follow the official website ([http://mxnet-mli.readthedocs.io/en/latest/how\\_to/build.html#building-on-ubuntu-debian](http://mxnet-mli.readthedocs.io/en/latest/how_to/build.html#building-on-ubuntu-debian)) to install mxnet**
  - `sudo apt-get update`
  - `sudo apt-get install -y build-essential git libatlas-base-dev libopencv-dev`
  - `git clone --recursive https://github.com/dmlc/mxnet` (<https://github.com/dmlc/mxnet>)
  - `edit make/config.mk`
    - `set cuda= 1, set cudnn= 1, add cuda path`
  - `cd mxnet`
  - `make clean_all`
  - `make -j4`
    - One problem I encountered was , "gcc version later than 5.3 not supported!" My gcc version was 5.4, and I had to remove it.
      - `apt-get remove gcc g++`
      - `conda install -c anaconda gcc=4.8.5`
      - `gcc --version`
- **Python package install ([http://mxnet-mli.readthedocs.io/en/latest/how\\_to/build.html#python-package-installation](http://mxnet-mli.readthedocs.io/en/latest/how_to/build.html#python-package-installation)) for mxnet**
  - `conda install -c anaconda numpy=1.11.1`
  - **Method 1:**
    - `cd python; sudo python setup.py install`
    - `sudo apt-get install python-setuptools`
  - **Method 2:**
    - `cd mxnet`
    - `cp -r ../mxnet/python/mxnet .`
    - `cp ../mxnet/lib/libmxnet.so mxnet/`
  - **Quick test :**
    - `python example/image-classification/train_mnist.py`
  - **GPU test:**

### 4.5.3 Caffe

- Follow this detailed guide: **Caffe Ubuntu 16.04 or 15.10 Installation Guide** (<https://github.com/BVLC/caffe/wiki/Ubuntu-16.04-or-15.10-Installation-Guide>)
- OpenCV is needed. For Installation of Opencv 3.1, refer to this link: **Ubuntu 16.04 or 15.10 OpenCV 3.1 Installation Guide** (<https://github.com/BVLC/caffe/wiki/Ubuntu-16.04-or-15.10-OpenCV-3.1-Installation-Guide>)

### 4.5.4 Darknet

- This is the easiest of all to install. Just type “make”, and that’s it.

## 5. Docker for Out-of-the-Box Deep Learning Environment

I used to have caffe, darknet, mxnet, tensorflow all installed correctly in Ubuntu 14.04 and TITAN-X (cuda7.5). And I have done projects with these frameworks, all turning out working well. It is therefore safer to use these pre-built environments than adventuring with latest versions, if you want to focus on the deep learning research instead of being potentially bothered by peripheral problems you may encounter. Then you should consider isolate each framework with its own environment using docker. These docker images can be found in DockerHub (<https://hub.docker.com/>).

### 5.1 Install Docker

Unlike virtual machines, a docker image is built with layers. Same ingredients are shared among different images. When we download a new image, existing components won’t be re-downloaded. It is more efficient and convenient compared to the replacement of the whole virtual machine image. Docker containers are like the run-time of docker images. They can be committed and used to update docker images, just like Git.

To install docker on Ubuntu 16.04, we follow instructions on the official website (<https://docs.docker.com/engine/installation/linux/ubuntu/linux/>).

### 5.2 Install NVIDIA-Docker

Docker containers are both hardware-agnostic and platform agnostic, but docker does not natively support NVIDIA GPUs with containers. (The hardware is specialized, and driver is needed.) To solve this problem, we need the nvidia-docker to mount the devices and driver files when starting the container on the target machine. In this way, the image is

The installation of NVIDIA-Docker can be found here (<https://github.com/NVIDIA/nvidia-docker>).

### 5.3 Download Deep Learning Docker Images

I have collected some pre-built docker images from the Docker Hub.

They are listed here:

- cuda-caffe (<https://hub.docker.com/r/kaixhin/cuda-caffe/>)
- cuda-mxnet (<https://hub.docker.com/r/kaixhin/cuda-mxnet/>)
- cuda-keras-tensorflow-jupyter (<https://hub.docker.com/r/lukovkin/dockerfile-cuda-tensorflow-keras-jupyter/>)

More can be easily found on docker hub.

### 5.4 Share Data between Host and Container

For computer vision researchers, it will be awkward not to see results. For instance, after adding some Picasso style to an image, we would definitely want to the output images from different epoches. Check out this page (<https://github.com/rocker-org/rocker/wiki/Sharing-files-with-host-machine>) quickly to share data between the host and the container. In a shared directory, we can create projects. On the host, we can start coding with text editors or whatever IDEs we prefer. And then we can run the program in the container. The data in the shared container can be viewed and processed with the GUI of the host Ubuntu machine.

### 5.5 Learn Simple Docker Commands

Don't be overwhelmed if you are new to docker. It does not need to be systematically studied unless you want to in the future. Here are some simple commands for you to use to start dealing with docker. Usually they are sufficient if you consider Docker a tool, and want to use it solely for a deep learning environment.

#### How to check the docker images ?

- docker images : Check all the docker images that you have.

#### How to check the docker containers ?

- docker ps -a : Check all the containers that you have.
- docker ps: Check containers that are running

#### How to exit a docker container ?

- (Method 1) In the terminal corresponding the current container:
  - exit
- (Method 2) Use [Ctrl + Alt + T] to open a new terminal, or use [Ctrl + Shift + T] to open a new terminal tab :
  - docker ps -a : Check the containers you have.
  - docker ps: Check the running container(s).
  - docker stop [container's ID]: Stop this container.

## How to remove a docker image ?

- `docker rmi [docker_image_name]`

## How to remove a docker container ?

- `docker rm [docker_container_name]`

## How to create our own docker image, based on one that is from someone else ? ( Update a container created from an image and commit the results to an image. )

- load image , open a container
- do some changes in the container
- commit to the image
  - `docker commit -m "Message: Added changes" -a "Author: Guanghan" 0b2616b0e5a8 ning/cuda-mxnet`

## Copy data between host and the docker container :

- `docker cp foo.txt mycontainer:/foo.txt`
- `docker cp mycontainer:/foo.txt foo.txt`

## Open a container from a docker image :

- If the container is to be saved because it is probably to be committed:
  - `docker run -it [image_name]`
- If the container is only for temporary use:
  - `docker run --rm -it [image_name]`

◀ **YOLO CPU Running Time Reduction:  
Basic Knowledge and Strategies**  
(<http://guanghan.info/blog/en/my-works/yolo-cpu-running-time-reduction-basic-knowledge-and-strategies/>)

**18 Comments on "Build Personal Deep Learning Rig: GTX 1080 + Ubuntu 16.04 + CUDA 8.0RC + CuDnn 7 + Tensorflow/Mxnet/Caffe/Darknet"**

Powered by [OneAll Social Login](#)

Notify of

☐ new follow-up comments

☐ Email

>




So helpful! Thank you!



0  |  Reply - Share

Hide Replies ^


Guanghan Ning

1 year 3 months ago 

(<http://guanghan.info>) You are very welcome, and good luck with your research!

0  |  Reply - Share

wergerg


1 year 3 months ago 

傻B才买1080 , 性能只差20%,价格贵一倍,搞半天最后tf根本不支持8.0....

-45  |  Reply - Share

Hide Replies ^


Guanghan Ning

1 year 3 months ago 

(<http://guanghan.info>) tf支持8.0 , 不过安装起来麻烦一点。  
NVIDIA官网有更详细的安装说明 : <http://www.nvidia.com/object/gpu-accelerated-applications-tensorflow-installation.html>  
新的titan x不是出了么 , 不喜欢1080没关系 , 有更好的选择——显存至少上去了吧 ?

8  |  Reply - Share

Paul Vilevac


1 year 3 months ago 

What is the intended target of this project? It's quite a nice model for a single machine with hardware accelerated processing. I what impact be to efficient result generation by spending the same funding on a PI and docker swarm. If only I had more time to play.

0  |  Reply - Share

Hide Replies ^

Guanghan Ning

1 year 2 months ago 


(<http://guanghan.info>) I'd say it's nice and flexible to have a machine at hand to test new ideas. In deep learning research, when you think of a new layer, or a new training scheme, you can implement and try it right away. For a certain purpose though, one may want to use a more powerful machine with multiple GPUs to train. That is usually from the lab or the company.



panovr

1 year 2 months ago

你好，全新安装的Ubuntu 16.04能够识别出GTX 1080的硬件吗？之前看过一个帖子，说他安装的时候出现了不能识别的情况。

0  |  Reply - Share

Hide Replies ^

Guanghan Ning

1 year 2 months ago

(<http://guanghan.info>) 你好，全新安装的ubuntu应该是不能马上识别GTX1080了，我记得我使用集成显卡登入ubuntu，然后安装了最新的驱动。

0  |  Reply - Share

Anonymous

1 year 2 months ago

With Ubuntu 16.04, You may need to disable the secure boot in bios and then install the standard driver of gtx 1080 from Nvidia

1  |  Reply - Share

Buklik

1 year 2 months ago

Hi! Thank you for your article! A question – with Ubuntu 14.04, CUDA 8.0, Cudnn v5 and GTX 1080 will work as well?

0  |  Reply - Share

Enhao Song

1 year 2 months ago

I just bought a desktop with Gtx 1080. These are really helpful information. Thank you!

0  |  Reply - Share

SH Ho

1 year 1 month ago

Hi Ning

Thank you for such a superbly written summary! It's really helpful!

So it's been a few months since you built this machine, are you happy with it? Do you feel you need to provide extra cooling to the CPU or the case?


Is Ubuntu 16.04 stable and playing nice w/ the HW?

How is your experience w/ GTX1080? Are you glad you got it or do you think it's inadequate or overkill for most ML and TF datasets?

Thank you and I look forward to your reply!



4  |  Reply - Share

Jimmy Kumseong Moon


11 months 29 days ago 

Wow! it's all I wanted to know so far.

I found a used GTX 1080 GDDR5x only and have to find the other parts again.

0  |  Reply - Share


Jimmy Kumseong Moon

11 months 29 days ago 

Oops, Sorry. I forgot to say thanks a billion!

0  |  Reply - Share

STYLIANOS IORDANIS

11 months 28 days ago 

Thank you so much! incredible article

Can you elaborate on how you managed to successfully install mxnet on anaconda 2.7?

It has become impossible for me to execute:

```
import mxnet
```

I get:

```
File "/home/steliox/mxnet/python/mxnet/__init__.py", line 7, in
```

```
from .base import MXNetError
```

```
File "/home/steliox/mxnet/python/mxnet/base.py", line 43, in
```

```
_LIB = _load_lib()
```

```
File "/home/steliox/mxnet/python/mxnet/base.py", line 35, in _load_lib
```

```
lib = ctypes.cdll.LoadLibrary(lib_path[0])
```

```
File "/home/steliox/anaconda2/lib/python2.7/ctypes/__init__.py", line 440, in
```

```
LoadLibrary
```

```
return self._dlltype(name)
```

```
File "/home/steliox/anaconda2/lib/python2.7/ctypes/__init__.py", line 362, in
```


```
__init__
```

```
self._handle = _dlopen(self._name, mode)
```

```
OSError: /home/steliox/anaconda2/bin/../lib/libgomp.so.1: version `GOMP_4.0'
not found (required by /home/steliox/mxnet/python/mxnet/../lib/libmxnet.so)
```



2  |  Reply - Share

J. Lin


10 months 4 days ago 

Hi. Just wondering. Is it possible to use the monitor if we are doing deep

learning on the GTX 1080? Will it slow down the processing time of the deep learning library?

1  |  Reply - Share


Alston

4 months 20 days ago 

Thanks for sharing. BTW, ssh reverse tunnel could do the better job for remote control! It works even with dynamic IP, only if you have a proxy server.

1  |  Reply - Share

J Yang

1 month 5 days ago 

I train my model on GTX 1080Ti + cuda 7.5 + caffe , the caffe can be run, but the converged loss value is larger than on GTX 980Ti. How to solve it ?

0  |  Reply - Share