Deep Learning Practice with Caffe: Prerequisites

Kye-Hyeon Kim

Computer Vision and Deep Learning Software Group (CVDS)

Intel

Summary

- Before starting this class, please complete following requirements
  - Install Ubuntu + Docker + Caffe
    - Follow instructions in pp. 3~14
    - They should work properly with tests in pp. 13 and 18, + pp. 14 for GPU mode
    - If your machine has NVIDIA GPU, preparing GPU version is strongly recommended
  - Download some large-scale image databases (pp. 15~18)
  - Make your PC accessible from anywhere
    (or prepare all the above in your laptop and bring it)
    - Follow instructions in pp. 19~23
    - Test in pp. 22 should work properly with your public IP address

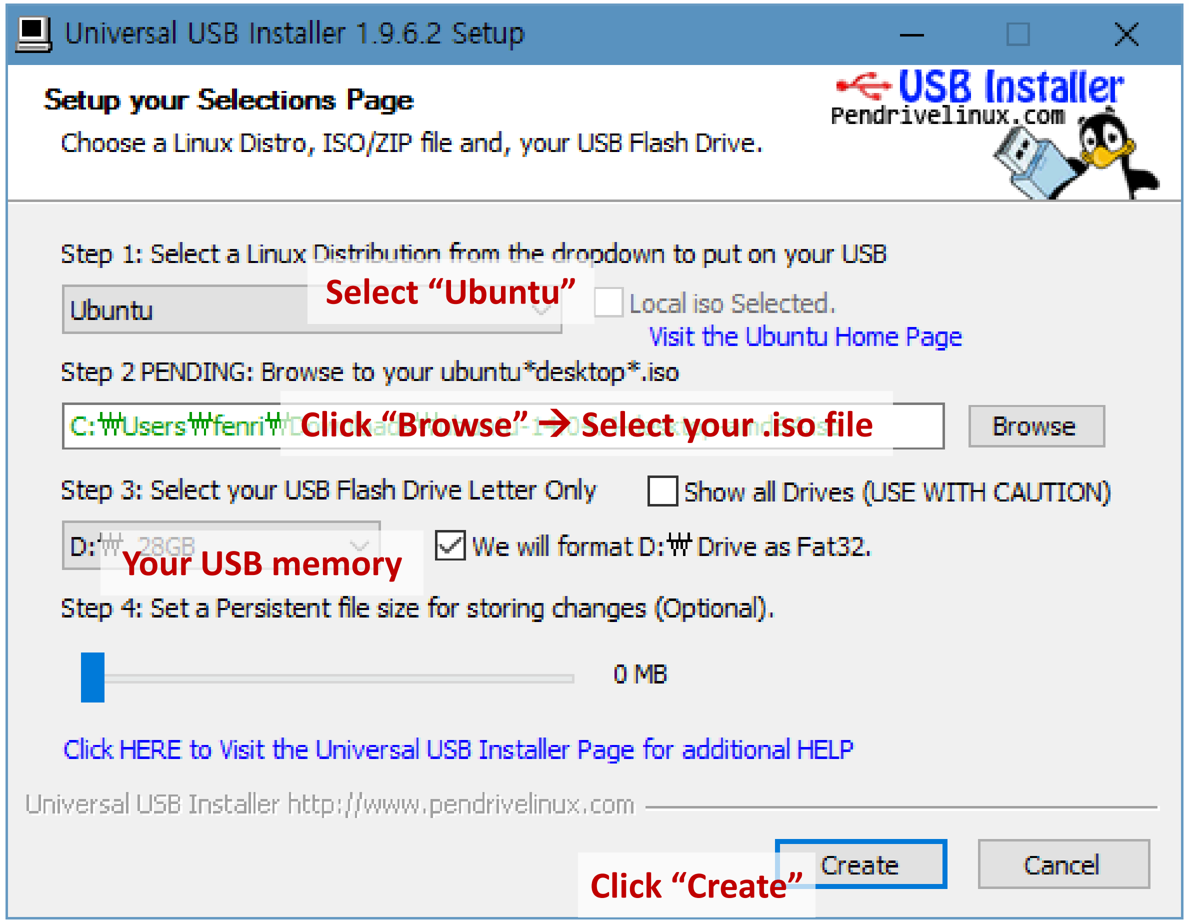
Installation

Installation: Overview

- Pre-requisites
    - 1 PC with…
        - NVIDIA GPU: optional, but strongly recommended for practice in this class
        - 1 free disk to install Ubuntu (≥ 128GB): Internal or USB disk
    - 1 bootable USB memory (≥ 8GB)
- Recommended method: Ubuntu + Docker + Pre-built Caffe image
    - Install Ubuntu 14.04
    - Install NVIDIA driver & CUDA-7.5 (Skip for CPU-only mode)
    - Install Docker
    - Install NVIDIA Docker (Skip for CPU-only mode)
    - Build Caffe image & Create Caffe virtual machine

Installation: Ubuntu

- Download Ubuntu Desktop 14.04, 64-bit
  - http://ftp.neowiz.com/ubuntu-releases/14.04.5/ubuntu-14.04.4-desktop-amd64.iso
- Download Universal USB Installer
  - http://www.pendrivelinux.com/universal-usb-installer-easy-as-1-2-3/
- Insert your USB memory & Run Universal USB Installer
  - Set as shown in the right panel



Universal USB Installer 1.9.6.2 Setup

Setup your Selections Page

Choose a Linux Distro, ISO/ZIP file and, your USB Flash Drive.

USB Installer Pendrivelinux.com

Step 1: Select a Linux Distribution from the dropdown to put on your USB

Ubuntu — Select "Ubuntu" — ☐ Local iso Selected.

Visit the Ubuntu Home Page

Step 2 PENDING: Browse to your ubuntu*desktop*.iso

C:₩Users₩fenri₩ — Click "Browse" → Select your .iso file — Browse

Step 3: Select your USB Flash Drive Letter Only — ☐ Show all Drives (USE WITH CAUTION)

D:₩ 28GB — Your USB memory — ☑ We will format D:₩ Drive as Fat32.

Step 4: Set a Persistent file size for storing changes (Optional).

0 MB

Click HERE to Visit the Universal USB Installer Page for additional HELP

Universal USB Installer http://www.pendrivelinux.com

Click "Create" — Create — Cancel

Installation: Ubuntu

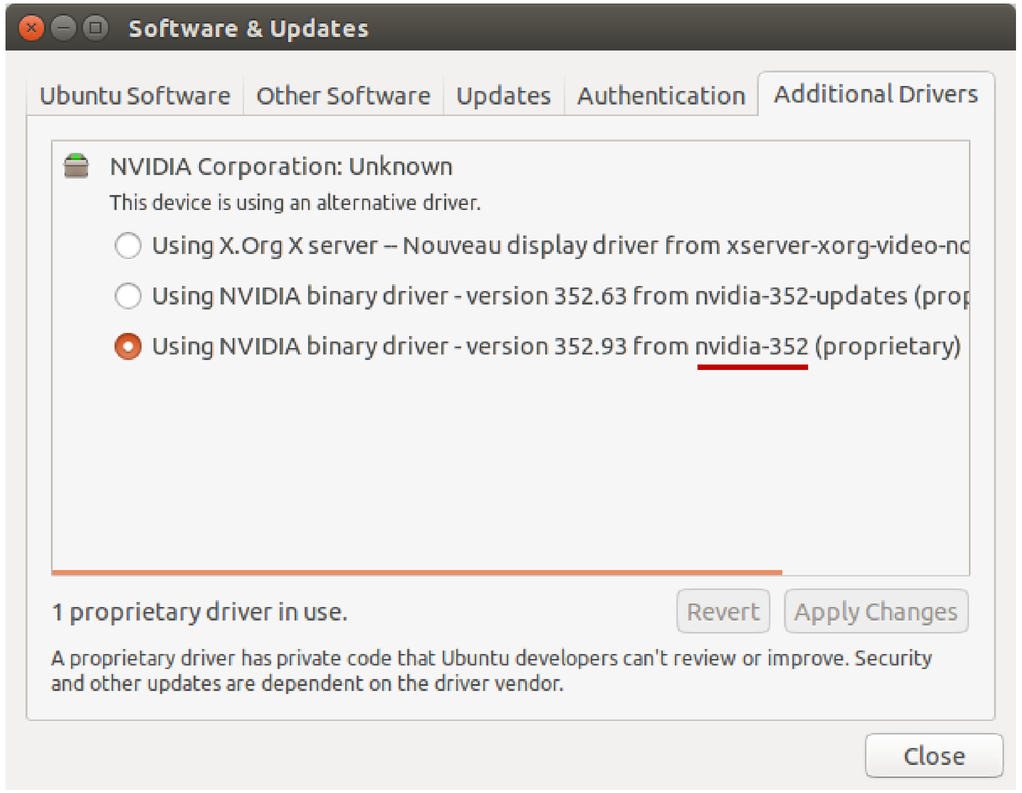
- Install Ubuntu using your USB memory
  - Language: Select "English"
  - Installation Type: Select "Something else" → Choose your free disk
  - Keyboard Layout: Select "English (US)"

Installation: NVIDIA Driver (GPU mode)

- Install NVIDIA driver
  - Open terminal (<Ctrl + Alt + t>)

```
$ sudo apt-get update && sudo apt-get upgrade
```

  - System Settings
    → "Software & Updates"
    → "Additional Drivers" tab
    → Select "nvidia-352"
- If it doesn't work or causes some problems with CUDA, install manufacturer's driver
  - https://developer.nvidia.com/cuda-downloads
  - Select Linux → x86_64 → Ubuntu → 14.04 → runfile (local)
  - You will probably be faced with many troubles



NVIDIA Corporation: Unknown
This device is using an alternative driver.
( ) Using X.Org X server – Nouveau display driver from xserver-xorg-video-no
( ) Using NVIDIA binary driver - version 352.63 from nvidia-352-updates (prop
[✓ Using NVIDIA binary driver - version 352.93 from nvidia-352 (proprietary)]

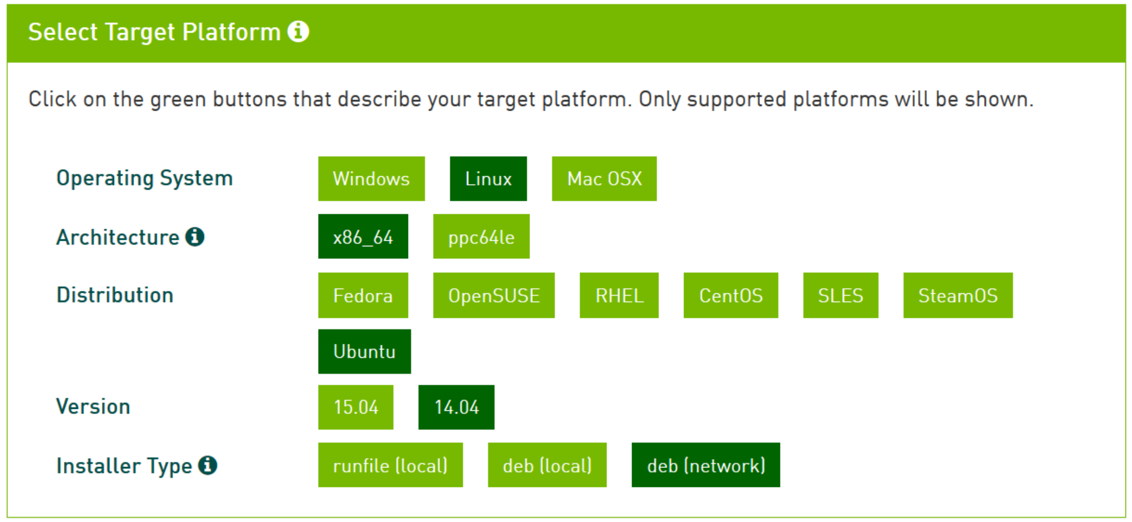
1 proprietary driver in use.

A proprietary driver has private code that Ubuntu developers can't review or improve. Security and other updates are dependent on the driver vendor.

Installation: CUDA (GPU mode)

- Open Firefox & Download CUDA 7.5
  - https://developer.nvidia.com/cuda-downloads
  - Select Linux → x86_64 → Ubuntu → 14.04 → deb (network)
- Open terminal
  & Follow installation instructions

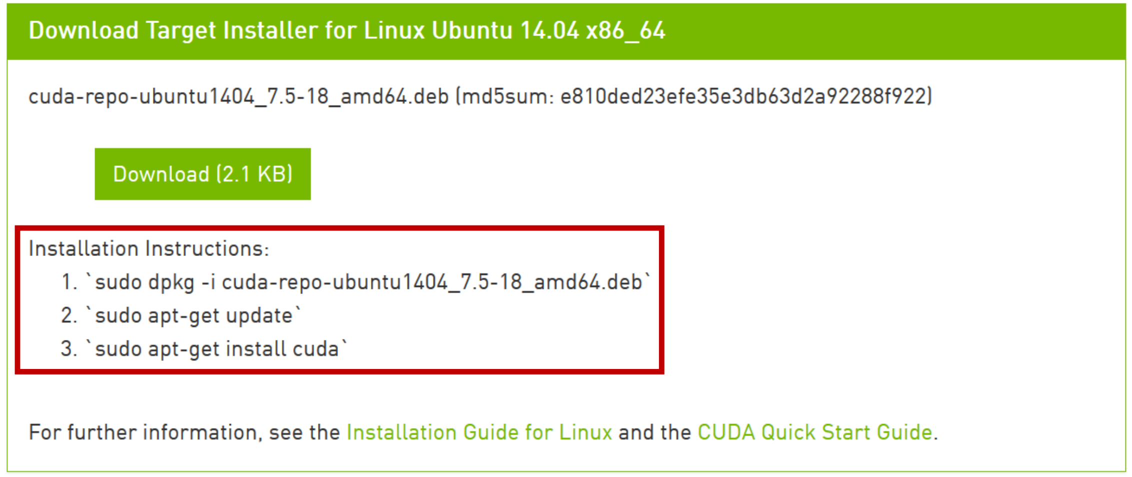
```
$ cd /home/<your username>/Downloads
$ sudo dpkg -i cuda-repo-ubuntu1404_7.5-18_amd64.deb
$ sudo apt-get update
$ sudo apt-get install cuda
// Restart after installation
$ sudo init 6
```



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 SteamOS [✓ Ubuntu]
Version	15.04 [✓ 14.04]
Installer Type	runfile (local) deb (local) [✓ deb (network)]



Download Target Installer for Linux Ubuntu 14.04 x86_64

cuda-repo-ubuntu1404_7.5-18_amd64.deb (md5sum: e810ded23efe35e3db63d2a92288f922)

Download (2.1 KB)

Installation Instructions:
1. `sudo dpkg -i cuda-repo-ubuntu1404_7.5-18_amd64.deb`
2. `sudo apt-get update`
3. `sudo apt-get install cuda`

For further information, see the Installation Guide for Linux and the CUDA Quick Start Guide.

Installation: Docker

- Open terminal & Follow installation instructions in https://docs.docker.com/engine/installation/linux/ubuntulinux/
- Test whether Docker is properly installed

```
$ sudo docker run hello-world
```

```
Hello from Docker!
This message shows that your installation appears to be
working correctly.
...
```

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificates

$ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 \
                   --recv-keys 58118E89F3A912897C070ADBF76221572C52609D

$ sudo bash -c \
    'echo "deb https://apt.dockerproject.org/repo ubuntu-trusty main" > \
     /etc/apt/sources.list.d/docker.list'

$ sudo apt-get update
$ sudo apt-get purge lxc-docker
$ apt-cache policy docker-engine
$ sudo apt-get update
$ sudo apt-get install linux-image-extra-$(uname -r)
$ sudo apt-get install apparmor
$ sudo apt-get update
$ sudo apt-get install docker-engine
$ sudo service docker start
```

Installation: NVIDIA Docker (GPU mode)

- Follow installation instructions for "Ubuntu distributions" in https://github.com/NVIDIA/nvidia-docker/wiki

```
$ wget -P /tmp https://github.com/NVIDIA/nvidia-docker/releases/download/v1.0.0-rc.3/nvidia-docker_1.0.0.rc.3-1_amd64.deb
$ sudo dpkg -i /tmp/nvidia-docker*.deb && rm /tmp/nvidia-docker*.deb
```

- Test whether nvidia-docker is properly installed

```
$ sudo nvidia-docker run --rm nvidia/cuda:7.5-devel nvcc --version
$ sudo nvidia-docker run --rm nvidia/cuda:7.5-devel nvidia-smi
```

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2015 NVIDIA Corporation
Built on Tue_Aug_11_14:27:32_CDT_2015
Cuda compilation tools, release 7.5, V7.5.17
```

- Numbers can be different

```
+------------------------------------------------------+
¦ NVIDIA-SMI 352.93     Driver Version: 352.93         ¦
|-------------------------------+----------------------+----------------------+
¦ GPU  Name        Persistence-M¦ Bus-Id        Disp.A ¦ Volatile Uncorr. ECC ¦
¦ Fan  Temp  Perf  Pwr:Usage/Cap¦         Memory-Usage ¦ GPU-Util  Compute M. ¦
...
```

- Numbers can be different, but you should see a table like this
- Otherwise, it means that NVIDIA driver is not properly installed

Installation: Caffe (GPU mode)

- Open gedit & Write Dockerfile

```
$ gedit Dockerfile
```

```
FROM kaixhin/cuda-caffe
RUN apt-get update
RUN apt-get install -y x11-apps python-tk tk-dev vim
RUN pip uninstall -y matplotlib
RUN pip install matplotlib
ENV DISPLAY :0
RUN echo "export PATH=:/root/caffe/build/tools:\${PATH}" >> ~/.bashrc
RUN echo "export LD_LIBRARY_PATH=:/root/caffe/build/lib:\${LD_LIBRARY_PATH}" >> ~/.bashrc
RUN cp /root/caffe/Makefile.config.example /root/caffe/Makefile.config
RUN echo "USE_CUDNN := 1" >> /root/caffe/Makefile.config
RUN cd /root/caffe
RUN git pull origin master
RUN make clean
RUN make -j"$(nproc)" all && make pycaffe
```

- Build image named caffe-img

```
$ sudo nvidia-docker build -t caffe-img .
```

- Permission settings for GUI

```
$ echo "xhost +SI:localuser:root" >> ~/.profile
$ xhost +SI:localuser:root
```

Installation: Caffe (CPU-only mode)

- Open gedit & Write Dockerfile

```
$ gedit Dockerfile
```

```
FROM kaixhin/caffe
RUN apt-get update
RUN apt-get install -y x11-apps python-tk tk-dev vim
RUN pip uninstall -y matplotlib
RUN pip install matplotlib
ENV DISPLAY :0
RUN echo "export PATH=:/root/caffe/build/tools:\${PATH}" >> ~/.bashrc
RUN echo "export LD_LIBRARY_PATH=:/root/caffe/build/lib:\${LD_LIBRARY_PATH}" >> ~/.bashrc
RUN cp /root/caffe/Makefile.config.example /root/caffe/Makefile.config
RUN echo "CPU_ONLY := 1" >> /root/caffe/Makefile.config
RUN cd /root/caffe
RUN git pull origin master
RUN make clean
RUN make -j"$(nproc)" all && make pycaffe
```

- Build image named caffe-img

```
$ sudo docker build -t caffe-img .
```

- Permission settings for GUI

```
$ echo "xhost +SI:localuser:root" >> ~/.profile
$ xhost +SI:localuser:root
```

GPU vs. CPU: Only two lines are different!

```
FROM kaixhin/cuda-caffe
...
RUN echo "USE_CUDNN := 1" >> ...
```

```
FROM kaixhin/caffe
...
RUN echo "CPU_ONLY := 1" >> ...
```

Installation: Caffe

CPU-only mode: Replace nvidia-docker → docker

- Create virtual machine named caffe

```
$ sudo nvidia-docker run -tid \
                       -v /tmp/.X11-unix:/tmp/.X11-unix \
                       -v /tmp/.docker.xauth:/tmp/.docker.xauth \
                       -e XAUTHORITY=/tmp/.docker.xauth \
                       --name caffe caffe-img
```

t: Enable terminal mode
i: Get standard input (interactive mode)
d: Run on background

Options for GUI

- Open terminal on the VM

```
$ sudo nvidia-docker exec -ti caffe bash
```

```
$ sudo nvidia-docker exec -ti caffe bash   // Start terminal
    root@...:~/caffe#        // Now you are in VM as root
    ... do some work ...
    root@...:~/caffe# exit  // End terminal
$ sudo nvidia-docker stop caffe          // Power-off VM
$ sudo nvidia-docker start caffe         // Power on VM
$ sudo nvidia-docker commit caffe caffe_160819  // Backup VM
$ sudo nvidia-docker rm caffe            // Remove VM
$ sudo nvidia-docker rmi caffe-img       // Remove image
```

Installation: Caffe

- Test Caffe on the VM

```
~/caffe# ./data/cifar10/get_cifar10.sh
~/caffe# ./examples/cifar10/create_cifar10.sh
~/caffe# ./examples/cifar10/train_quick.sh
```

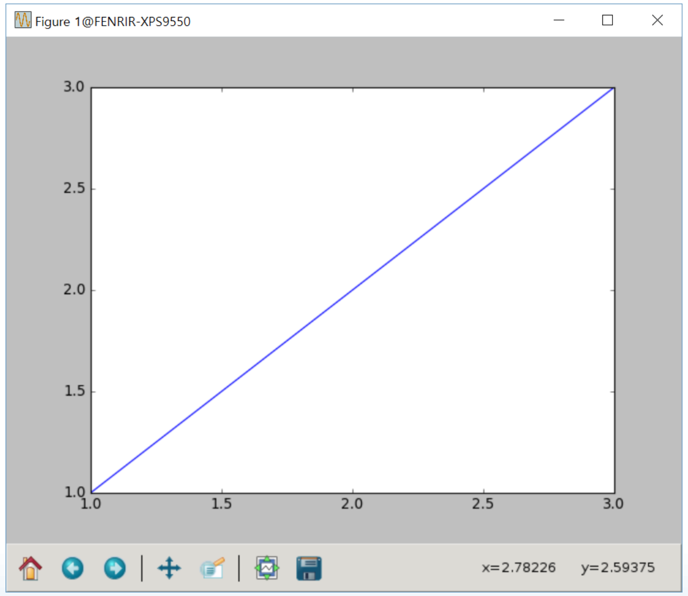
CPU-only mode: Edit solver_mode: **GPU** → solver_mode: **CPU**
in ./examples/cifar10/cifar10_quick_solver.prototxt

```
libdc1394 error: Failed to initialize libdc1394
...
I08… solver.cpp:317] Iteration 5000, loss = 0.584047
I08… solver.cpp:337] Iteration 5000, Testing net (#0)
I08… solver.cpp:404]     Test net output #0: accuracy = 0.7587
I08… solver.cpp:404]     Test net output #1: loss = 0.723281 (* 1 = 0.723281 loss)
I08… solver.cpp:322] Optimization Done.
I08… caffe.cpp:254] Optimization Done.
```

- Numbers can be different
- In CPU mode, every 100-iteration takes around 1 minute or more
- In GPU mode, every 100-iteration should be done in a few seconds, and the whole training process should be finished in several minutes. Otherwise, it means that CUDA doesn't work, mostly because NVIDIA driver is not properly installed

- Test GUI on the VM

```
~/caffe# python
>>> import matplotlib.pyplot as plt
>>> plt.plot([1,2,3], [1,2,3])
>>> plt.show()
```



- UI can be different, but you should see a figure like this

Datasets

Datasets: ILSVRC-2012 Dataset

- Benchmark DB for image classification and localization
- http://www.image-net.org/challenges/LSVRC/2012/nonpub-downloads
- Download:
  - Development kit (Task 1 & 2) (2.5MB)
  - Validation images (all tasks) (6.3GB)
  - Validation bounding box annotations (all tasks) (2.2MB)
  - Training images (Task 1 & 2) (optional, 138GB)
  - Training bounding box annotations (Task 1 & 2 only) (optional, 20MB)

Datasets: VOC-2007 Dataset

- Benchmark DB for object detection, classification, segmentation
- http://host.robots.ox.ac.uk/pascal/VOC/voc2007/
- Download:
  - training/validation data (450MB)
  - development kit code and documentation (250KB)
  - annotated test data (430MB)

Datasets: LFW Dataset

- Benchmark DB for face recognition
- http://vis-www.cs.umass.edu/lfw/
- Download:
  - All images as gzipped tar file (173MB)
  - pairsDevTrain.txt, pairsDevTest.txt, peopleDevTrain.txt, peopleDevTest.txt

Remote Access

Remote Access: Overview

- Recommended method: X11vnc + Port-forwarding
- You can choose any other options you prefer
  - TeamViewer (beware of hacking), Chrome remote desktop, ...
  - Your option should be able to display the plot in pp. 14 remotely

Remote Access: X11vnc

- Your PC: Install X11vnc server

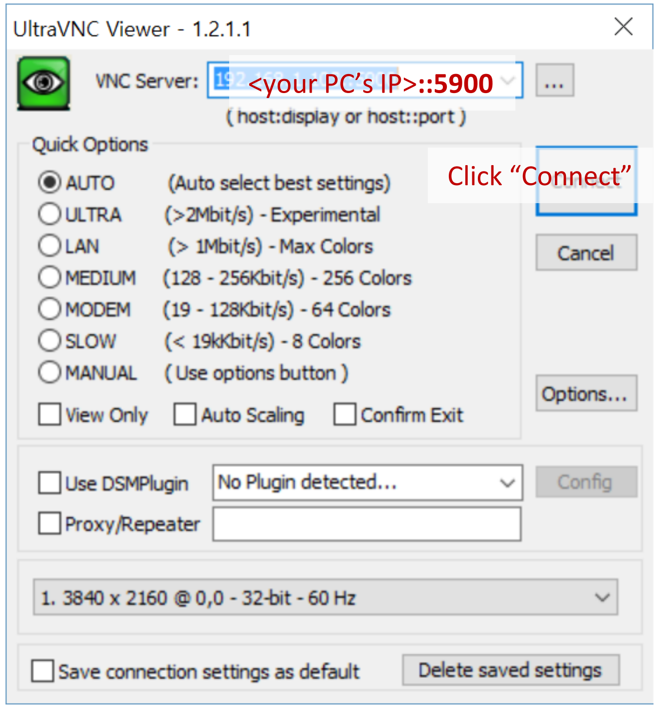
```
$ sudo apt-get install x11vnc

// Set VNC password & Press 'y' to save it
$ x11vnc -storepasswd

// Run server, you will see 'PORT=5900'
$ x11vnc -forever -shared -bg \
       -o ~/.vnc/x11vnc.log -rfbauth ~/.vnc/passwd
```

  - You should relaunch the server when your PC is restarted
  - For auto launch at startup, see http://uni2u.meximas.com/2014/12/22/ubuntu-14-04-and-vnc/

- Your laptop: Install VNC client
  - UltraVNC http://www.uvnc.com/downloads/ultravnc/116-download-ultravnc-1211.html
  - Sufficient to install viewer only
  - Any other clients are fine

Remote Access: X11vnc

- Your laptop: Connect to your PC



UltraVNC Viewer - 1.2.1.1

VNC Server: <your PC's IP>::5900
(host:display or host::port)

Quick Options

- [✓ AUTO] (Auto select best settings)
- ULTRA (>2Mbit/s) - Experimental
- LAN (> 1Mbit/s) - Max Colors
- MEDIUM (128 - 256Kbit/s) - 256 Colors
- MODEM (19 - 128Kbit/s) - 64 Colors
- SLOW (< 19kKbit/s) - 8 Colors
- MANUAL (Use options button)

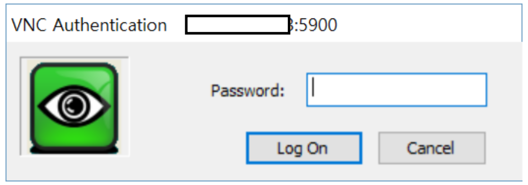
☐ View Only ☐ Auto Scaling ☐ Confirm Exit

☐ Use DSMPlugin: No Plugin detected...

☐ Proxy/Repeater

1. 3840 x 2160 @ 0,0 - 32-bit - 60 Hz

☐ Save connection settings as default

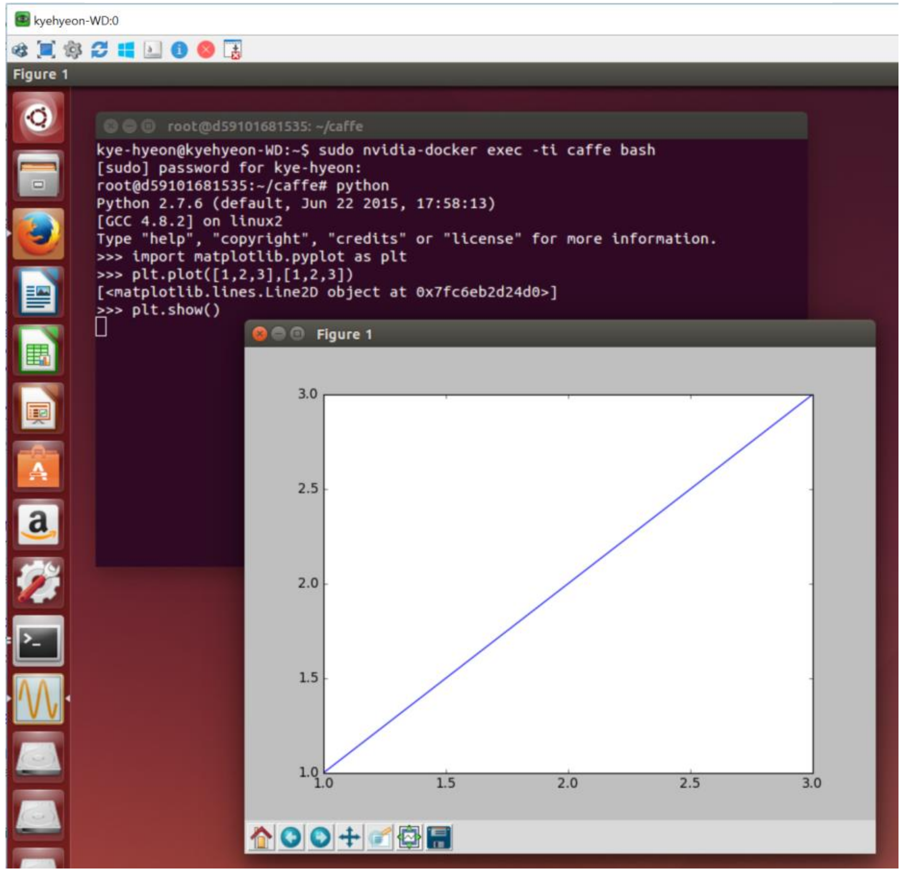


VNC Authentication [redacted]:5900

Password:

You will see this popup
if the connection is made successfully.
Input your VNC password

Check whether VNC displays a plot drawn in your Caffe VM



kyehyeon-WD:0

root@d59101681535: ~/caffe

```
kye-hyeon@kyehyeon-WD:~$ sudo nvidia-docker exec -ti caffe bash
[sudo] password for kye-hyeon:
root@d59101681535:~/caffe# python
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import matplotlib.pyplot as plt
>>> plt.plot([1,2,3],[1,2,3])
[<matplotlib.lines.Line2D object at 0x7fc6eb2d24d0>]
>>> plt.show()
```

Remote Access: Port Forwarding

- It requires if your PC has a private IP address (e.g., 192.168.xxx.xxx) and your router has a public IP
- Set up your router to forward packets to your PC via an arbitrary port
    - Search with your router's model name and/or manufacturer and "port forwarding" and follow instructions
        - e.g., Search with "iptime port forwarding" → http://ngee.tistory.com/224
    - Set up port-forwarding to <your PC's private IP> and port 5900
        - e.g., <your router's public IP>::30123 → 192.168.xxx.xxx::5900
        - Now you can access your PC from anywhere via <your router's public IP>::30123
    - Also set up another port to forward to <your PC's private IP> and port 22 (back-up plan)
- Repeat VNC test in pp. 22 through your public IP and port
    - If it also works well, you are finished ☺