

AI Workshop

머신러닝을 여행하는 예술가를 위한 안내서

DIGITS 환경 설치 방법

DADASMIMI

송하윤

머신러닝을 여행하는 예술가를 위한 안내서 - 실습

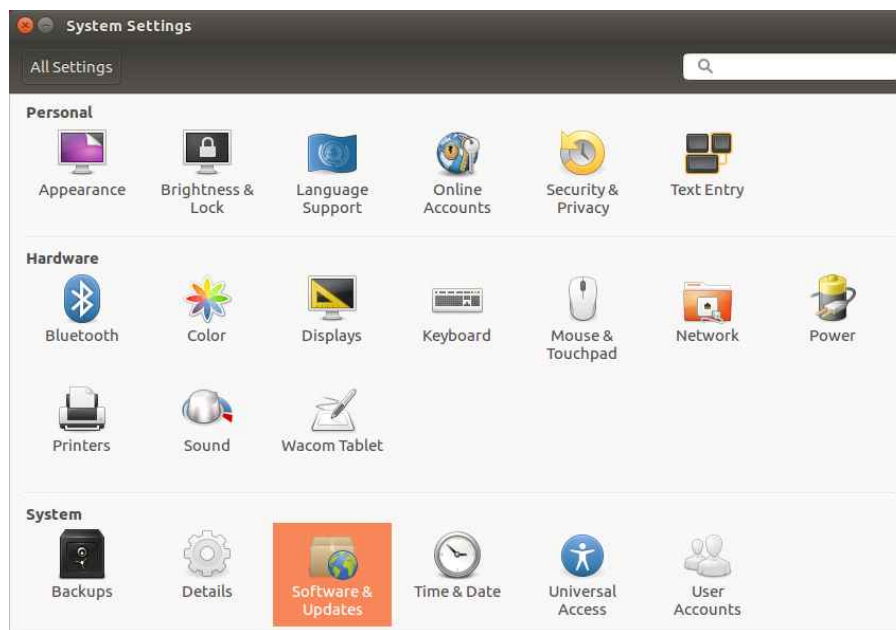
1. 지도학습을 위한

DIGITS 환경 설치 방법입니다.

DIGITS 환경은 Ubuntu 16.04 OS 환경에서 설치하도록 합니다.

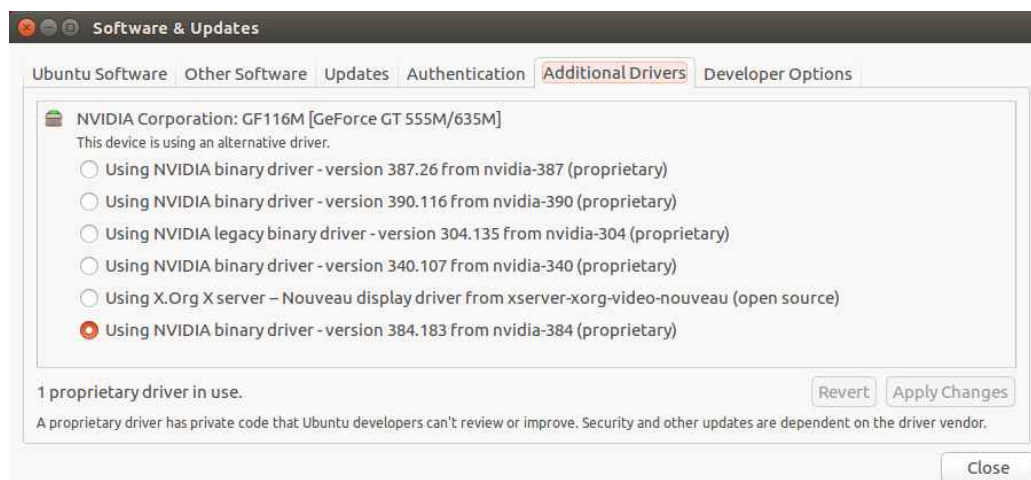
## 1. NVIDIA 드라이버 설치하기

System Settings에서 Software & Updates 아이콘을 클릭합니다.



Additional Drivers 탭에서 그래픽카드에 맞는 드라이버를 선택에 선택 후 Apply Changes 버튼을 눌러 설치해줍니다.

설치가 완료되면 재부팅 해줍니다.



## 2. NVIDIA Cuda 설치하기

<https://developer.nvidia.com/cuda-toolkit-archive>

위의 경로로 이동하여 그래픽카드에 맞는 cuda toolkit을 선택하여 설치 해 줍니다.

[Home](#) > [High Performance Computing](#) > [CUDA Toolkit](#) > [CUDA Toolkit Archive](#) > [CUDA Toolkit 10.1 original Archive](#)

### CUDA Toolkit 10.1 original Archive

**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	18.10	18.04	16.04	14.04		
Installer Type	runfile (local)	deb (local)	deb (network)	cluster (local)		

**Download Installer for Linux Ubuntu 16.04 x86\_64**

The base installer is available for download below.

Base Installer

Download [2.4 GB]

Installation Instructions:

1. Run `sudo sh cuda_10.1.105_418.39_linux.run`
2. Follow the command-line prompts

The CUDA Toolkit contains Open-Source Software. The source code can be found [here](#).  
The checksums for the installer and patches can be found in [Installer Checksums](#).  
For further information, see the [Installation Guide for Linux](#) and the [CUDA Quick Start Guide](#).

## 3. Cudnn 설치하기

<https://developer.nvidia.com/cudnn>

위의 경로로 이동하여 Cuda 버전에 맞는 Cudnn을 설치 해 줍니다.

Cudnn 설치가 완료되면 재부팅 해줍니다.

#### 4. 실습 자료 다운받기

<https://github.com/hayun-dadasmimi/ai-workshop>

위의 링크로 이동하여 실습 자료를 다운로드 합니다.

압축 폴더를 압축 해제하여 내부의 'ThunderlMages' 폴더를 바탕화면 혹은 원하는 경로로 이동해 줍니다.

경로는 대소문자를 구분하며 바탕화면에 위치했다고 가정했을 시  
/home/username/Desktop/ThunderlMages/  
의 경로를 가지게 됩니다.

#### 5. Protobuf 빌드하기

Ctrl + Alt + T 커맨드를 입력하여 터미널을 열어줍니다.

터미널에 아래의 명령어를 순서대로 입력하여 Protobuf를 빌드해줍니다.

\*주의 Protobuf가 제대로 빌드되지 않으면 이후 Caffe 및 Digits 빌드에 실패하게 됩니다.

```
sudo apt-get install autoconf automake libtool curl make g++ git python-dev python-setuptools unzip
```

```
export PROTOBUF_ROOT=~/.protobuf
```

```
git clone https://github.com/google/protobuf.git $PROTOBUF_ROOT -b '3.2.x'
```

```
cd $PROTOBUF_ROOT
```

```
./autogen.sh
```

```
./configure
```

```
make "-j$(nproc) "
```

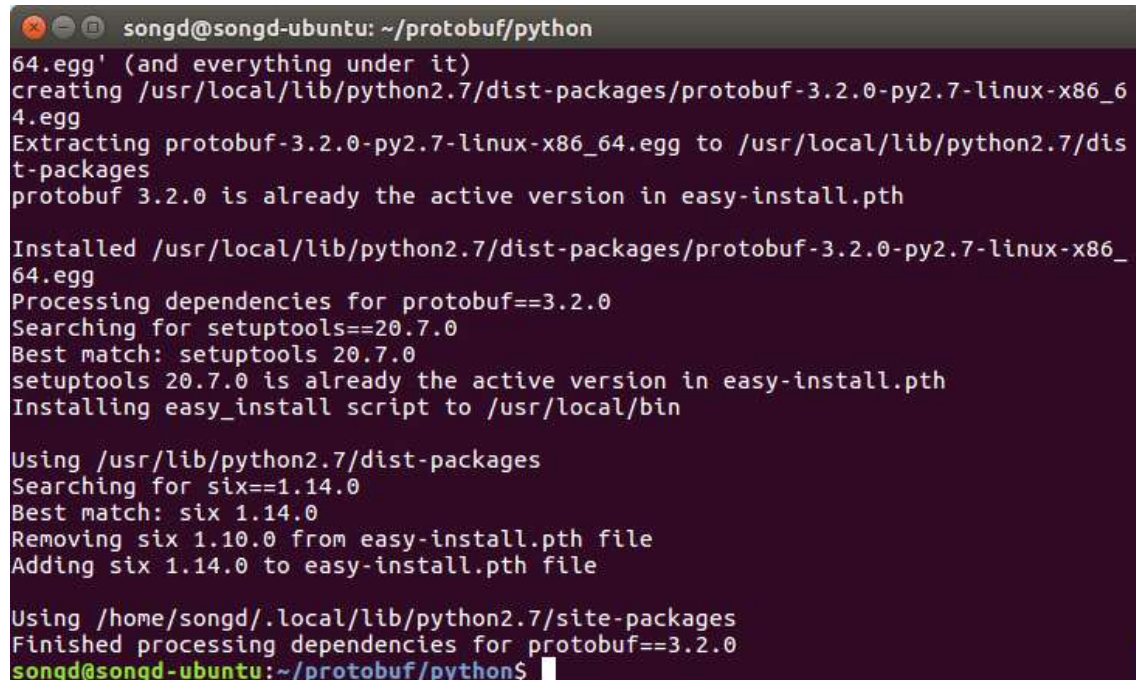
```
make install
```

```
ldconfig
```

```
cd python
```

```
python setup.py install --cpp_implementation
```

터미널의 마지막에 Finished processing dependencies for protobuf 문자가 출력되면 성공입니다.



```
songd@songd-ubuntu: ~/protobuf/python
64.egg' (and everything under it)
creating /usr/local/lib/python2.7/dist-packages/protobuf-3.2.0-py2.7-linux-x86_64.egg
Extracting protobuf-3.2.0-py2.7-linux-x86_64.egg to /usr/local/lib/python2.7/dist-packages
protobuf 3.2.0 is already the active version in easy-install.pth

Installed /usr/local/lib/python2.7/dist-packages/protobuf-3.2.0-py2.7-linux-x86_64.egg
Processing dependencies for protobuf==3.2.0
Searching for setuptools==20.7.0
Best match: setuptools 20.7.0
setuptools 20.7.0 is already the active version in easy-install.pth
Installing easy_install script to /usr/local/bin

Using /usr/lib/python2.7/dist-packages
Searching for six==1.14.0
Best match: six 1.14.0
Removing six 1.10.0 from easy-install.pth file
Adding six 1.14.0 to easy-install.pth file

Using /home/songd/.local/lib/python2.7/site-packages
Finished processing dependencies for protobuf==3.2.0
songd@songd-ubuntu:~/protobuf/python$
```

## 6. Caffe 빌드하기

터미널에 아래의 명령어를 순서대로 입력하여 Caffe를 빌드해줍니다.

```
sudo apt-get install --no-install-recommends build-essential cmake
git gfortran libatlas-base-dev libboost-filesystem-dev
libboost-python-dev libboost-system-dev libboost-thread-dev
libgflags-dev libgoogle-glog-dev libhdf5-serial-dev libleveldb-dev
liblmdb-dev libopencv-dev libsnappy-dev python-all-dev
python-dev python-h5py python-matplotlib python-numpy
python-opencv python-pil python-pip python-pydot python-scipy
python-skimage python-sklearn
```

```
export Caffe_ROOT=~/.caffe
```

```
git clone https://github.com/NVIDIA/caffe.git $Caffe_ROOT -b
```

'caffe-0.15'

```
sudo pip install -r $CAFFE_ROOT/python/requirements.txt
```

\*위의 명령에서 오류가 발생하면 아래의 명령을 입력합니다.

```
cat $CAFFE_ROOT/python/requirements.txt | xargs -n1 sudo pip install
```

```
cd $CAFFE_ROOT
```

```
mkdir build
```

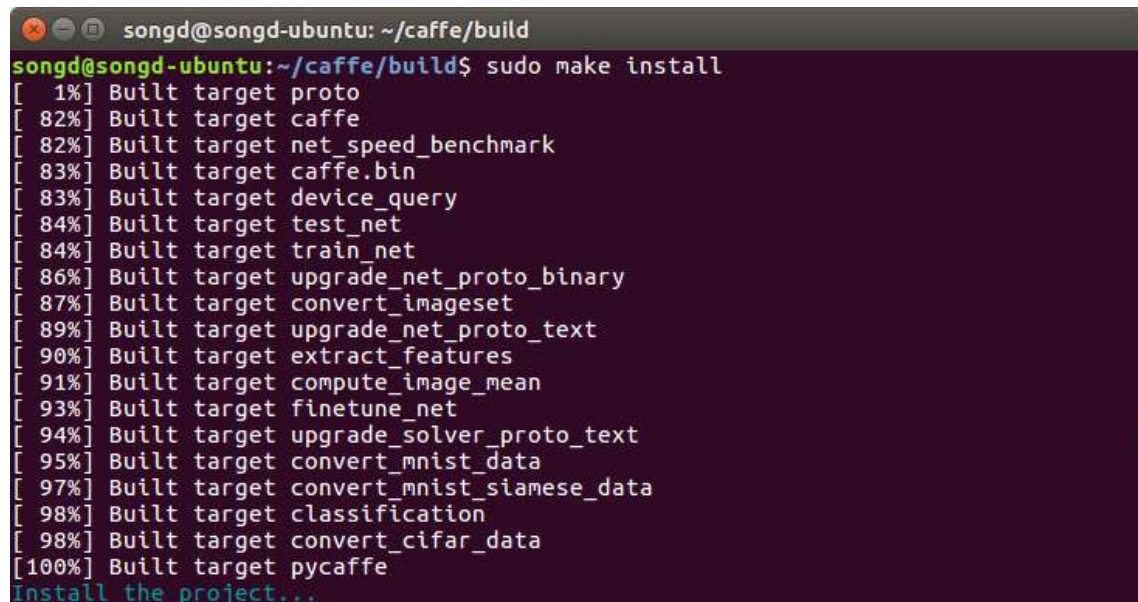
```
cd build
```

```
cmake ..
```

```
make -j$(nproc) "
```

```
make install
```

Caffe Install 중 [100%] Build target pycaffe 문자가 출력되면 성공입니다.

A terminal window titled 'songd@songd-ubuntu: ~/caffe/build' showing the output of the command 'sudo make install'. The output is a list of build targets with their progress percentages, ranging from 1% to 100%. The final target is 'pycaffe' at 100%. Below the list, it says 'Install the project...'.

```
songd@songd-ubuntu: ~/caffe/build
songd@songd-ubuntu:~/caffe/build$ sudo make install
[ 1%] Built target proto
[ 82%] Built target caffe
[ 82%] Built target net_speed_benchmark
[ 83%] Built target caffe.bin
[ 83%] Built target device_query
[ 84%] Built target test_net
[ 84%] Built target train_net
[ 86%] Built target upgrade_net_proto_binary
[ 87%] Built target convert_imageset
[ 89%] Built target upgrade_net_proto_text
[ 90%] Built target extract_features
[ 91%] Built target compute_image_mean
[ 93%] Built target finetune_net
[ 94%] Built target upgrade_solver_proto_text
[ 95%] Built target convert_mnist_data
[ 97%] Built target convert_mnist_siamese_data
[ 98%] Built target classification
[ 98%] Built target convert_cifar_data
[100%] Built target pycaffe
Install the project...
```

## 7. DIGITS 빌드하기

아래의 명령을 순서대로 입력하여 DIGITS를 빌드해줍니다.

```
CUDA_REPO_PKG=http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64/cuda-repo-ubuntu1604_8.0.61-1_amd64.deb
```

```
ML_REPO_PKG=http://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1604/x86_64/nvidia-machine-learning-repo-ubuntu1604_1.0.0-1_amd64.deb
```

```
wget "$CUDA_REPO_PKG" -O /tmp/cuda-repo.deb && sudo dpkg -i /tmp/cuda-repo.deb && rm -f /tmp/cuda-repo.deb
```

```
wget "$ML_REPO_PKG" -O /tmp/ml-repo.deb && sudo dpkg -i /tmp/ml-repo.deb && rm -f /tmp/ml-repo.deb
```

```
sudo apt-get update
```

```
sudo apt-get install --no-install-recommends git graphviz python-dev python-flask python-flaskext.wtf python-gevent python-h5py python-numpy python-pil python-pip python-scipy python-tk
```

```
DIGITS_ROOT=~/.digits
```

```
git clone https://github.com/NVIDIA/DIGITS.git $DIGITS_ROOT
```

```
sudo pip install -r $DIGITS_ROOT/requirements.txt
```

```
sudo pip install -e $DIGITS_ROOT
```

설치가 완료되면 `./digits-devserver` 명령을 통해 DIGITS를 실행 할 수 있습니다.

DIGITS가 실행 된 후 인터넷창에 `http://localhost:5000/` 주소를 입력 해 DIGITS 메인페이지로 이동할 수 있습니다.

## 부록

<https://github.com/NVIDIA/DIGITS/blob/master/docs/GettingStarted.md>

위의 링크에서 DIGITS를 이용해 MNIST 손글씨 데이터를 이미지 분류 실습을 할 수 있습니다.

DIGITS 환경 설치가 완료되었습니다.

머신러닝을 여행하는 예술가를 위한 안내서 - 실습 문서에서

1. 지도학습 실습을 진행할 수 있습니다.