

Deep Learning with Python

Seonho Oh

Published
with GitBook



Table of Contents

Introduction	0
Python 개발환경 구축	1
개발환경	1.1
파이썬 설치	1.2
필수 패키지 설치	1.3
딥러닝 환경 구축	2
공통	2.1
CUDA Toolkit	2.1.1
cuDNN Library	2.1.2
딥러닝 라이브러리	2.2
Theano (with Lasagne, nolearn)	2.2.1
TensorFlow	2.2.2
Caffe	2.2.3
Torch	2.2.4
기타	3
OpenBLAS	3.1
OpenCV	3.2
Boost	3.3
DLib	3.4
OpenFace	3.5

Deep Learning with Python

Python 을 이용한 개발환경 구축부터 딥러닝까지...

Python 개발환경 구축

- 개발환경
- 파이썬 설치
- 필수 패키지 설치

개발환경

아래의 환경에서 딥러닝을 위한 각종 도구를 설치하고 실습해 볼 것이다.

Option	Value
Operating System	Ubuntu 14.04.3 LTS
Architecture	x86_64
CPU	/ Intel Core i7-6700 3.4GHz
GPU	NVIDIA GeForce 650 Ti / NVIDIA GeForce Titan X
RAM	16GB / 32 GB

개발 환경 구성에 앞서, 먼저 설치된 소프트웨어 패키지 업데이트를 수행한다. 그리고 컴파일을 위한 도구 모음과 `git` 을 설치한다.

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install build-essential git
```

구현 및 실습은 `Jupyter Nootbook` 을 이용할 것이다.

파이썬 설치

기본적으로 Ubuntu 14.04 LTS 에는 2.7.6 버전이 설치되어 있다. 따라서 별도로 설치할 필요는 없다. 또한 여기서는 2.7.6 버전을 기준으로 설명할 것이다.

파이썬 패키지 설치 준비

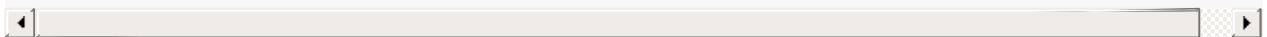
pip

패키지 설치에 앞서 파이썬으로 작성된 패키지 소프트웨어를 설치, 관리하는 패키지 관리 시스템인 **pip**를 먼저 설치한다. [Python Package Index \(PyPI\)](#)에서 많은 파이썬 패키지를 찾을 수 있다. 파이썬 2.7.9 이후 버전과 파이썬 3.4 이후 버전은 **pip**를 기본적으로 포함한다. **setuptools**와 **wheel**를 함께 설치한다.

```
$ curl https://bootstrap.pypa.io/get-pip.py -o - | sudo python
```

설치된 버전은 다음과 같이 확인할 수 있다.

```
$ pip --version # check pip version
pip 8.0.2 from /usr/local/lib/python2.7/dist-packages (python 2.7)
```



이후 파이썬 패키지 설치에 아래와 같이 할 수 있다.

```
$ pip install some-package-name
```

setuptools

pip 설치시에 함께 설치되지만, 최신 버전을 설치하고자 한다면 아래와 같이 하면 된다.

```
$ curl https://bootstrap.pypa.io/ez_setup.py -o - | sudo python
```

필수 패키지

- Jupyter
- NumPy
- SciPy
- Matplotlib
- Pandas

Jupyter

Jupyter는 웹 환경에서 실행 코드와 문서를 함께 작성하면서 실행 결과 및 가시화 결과 (차트 등)을 확인할 수 있는 도구이다.

설치 방법은 아래와 같다.

```
$ sudo pip install jupyter
```

실행은 터미널에서 `jupyter notebook` 을 입력하면 된다.

원격서버 설정하기

설정 파일 초기화

```
$ jupyter notebook --generate-config
```

`~/.jupyter/jupyter_notebook_config.py` 파일의 아래 내용을 수정한다.

```
# The IP address the notebook server will listen on.
c.NotebookApp.ip = 'your id address' # 서버 IP

# The directory to use for notebooks and kernels.
c.NotebookApp.notebook_dir = u'directory path' # 시작 디렉토리

# Whether to open in a browser after starting. The specific browser
# platform dependent and determined by the python standard library
# module, unless it is overridden using the --browser (NotebookApp
# configuration option.
c.NotebookApp.open_browser = False # 시작시 웹 브라우저 실행 여부
```

NumPy

NumPy는 과학용 계산에 특화된 파이썬 패키지이다.

요구사항

- OpenBLAS
- Cython

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install python-dev # for cython

sudo pip install cython
```

여기에서는 OpenBLAS 를 쓰는 것으로 가정한다.

설치

pip 로 설치할 경우, BLAS 설정이 안되어 있으므로 직접 저장소를 복제하여 설치한다.


```
# apt-get으로 설치된 것이 있다면 삭제
$ sudo apt-get remove python-numpy
# pip로 설치된 것이 있다면 삭제
$ sudo pip uninstall numpy

$ git clone https://github.com/numpy/numpy.git
$ cd numpy
# 현재 stable version은 1.10.4
$ git checkout v1.10.4
$ cp site.cfg.example site.cfg
$ vi site.cfg

# OpenBLAS 관련 설정 주석 풀기
[openblas]
libraries = openblas
library_dirs = /opt/OpenBLAS/lib
include_dirs = /opt/OpenBLAS/include
runtime_library_dirs = /opt/OpenBLAS/lib

# 컴파일 및 설치
$ python setup.py build --fcompiler=g95
$ sudo python setup.py install
```

OpenBLAS 바인딩 여부는 아래와 같이 확인할 수 있다.

```
$ cd .. # 동일한 디렉토리에서 하면 안 되므로, 디렉토리를 바꿔준다.
$ python
>> import numpy as np
>> np.__config__.show()
```

SciPy

SciPy는 과학용 계산에 필요한 기능들을 가진 파이썬 패키지이다.

요구사항

- [OpenBLAS](#)

설치

Numpy 와 마찬가지로 `apt-get install python-scipy` 로 설치했다면 OpenBLAS 설정이 안되어 있으므로 삭제한다. 또한 OpenBLAS 를 설치하기 전에 SciPy 를 먼저 설치한 경우에도 삭제 후, 재설치한다.

```
# OpenBLAS 바인딩이 안된 설치는 삭제
$ sudo apt-get remove python-scipy
$ sudo pip uninstall scipy

# 재설치
sudo pip install scipy
```

Numpy 와 동일하게 바인딩 여부를 확인할 수 있다.

```
$ python
>> import scipy
>> scipy.__config__.show()
```

Matplotlib

Matplotlib은 파이썬용 가시화 라이브러리로 MATLAB 과 유사한 인터페이스를 제공한다.

Dependencies

```
$ sudo apt-get install libpng-dev libfreetype6-dev
```

Install

```
$ sudo pip install matplotlib
```

Pandas

Pandas는 효과적인 데이터분석을 위한 파이썬 패키지이다.

```
$ sudo pip install pandas
```

딥러닝 개발환경 구축

- 공통
 - [CUDA Toolkit](#)
 - [cuDNN Library](#)
- 딥러닝 라이브러리
 - [Theano \(with Lasagne, nolearn\)](#)
 - [TensorFlow](#)
 - [Caffe](#)
 - [Torch](#)

NVIDIA CUDA Toolkit

다운로드

<https://developer.nvidia.com/cuda-downloads>에서 받을 수 있다. 최신 버전은 7.5 이다.

다음과 같이 Target Platform 을 선택하고 다운로드 한다.

Option	Value
Operating System	Linux
Architecture	x86_64
Distribution	Ubuntu
Version	14.04
Installer Type	deb (local)

아래와 같이 받을 수도 있다.

```
$ curl http://developer.download.nvidia.com/compute/cuda/7.5/Prod/
```

설치

아래와 같이 입력 후, 나머지는 화면에서 Next 만 누르면 된다.

```
$ sudo dpkg -i cuda-repo-ubuntu1404-7-5-local_7.5-18_amd64.deb
$ sudo apt-get update
$ sudo apt-get install cuda
```

경로 설정

CUDA 라이브러리와 실행 파일을 위한 경로를 .bashrc 파일에 추가해야 한다.

```
export CUDA_ROOT=/usr/local/cuda-7.5
export LD_LIBRARY_PATH=${CUDA_ROOT}/lib64:${LD_LIBRARY_PATH}
export PATH=${CUDA_ROOT}/bin:${PATH}
```

드라이버 설치 확인

아래 명령을 통해 확인할 수 있다.

```
$ nvidia-smi
```

오류 메시지가 나올 경우에는 재부팅 후 다시 시도하자.

설치 후 설정

기본적으로 TensorFlow 는 CUDA 7.0 을 지원하므로, 7.5 를 사용하기 위해서는 약간의 trick이 필요하다. 아래의 명령을 이용하여 /usr/local/cuda/lib64 의 *.so.7.5 에 해당하는 모든 링크를 *.so.7.0 으로 만들어준다.

```
$ for f in *.so.7.5; do sudo ln -s $f ${f%.5}.0; done

# 확인
$ ls -al *.7.?
```

NVIDIA cuDNN Library

다운로드

<https://developer.nvidia.com/rdp/cudnn-download>에서 받을 수 있는데, NVIDIA Accelerated Computing Developer Program 가입이 필요하다. 가입 신청을 하면 메일로 결과를 통보해주는데 시간이 좀(하루 정도) 걸린다.

우리는 Ubuntu 14.04.3 LTS 에 설치할 것이므로 cuDNN v4 Library for Linux 를 선택하면 된다.

최신버전은 v4인데 TensorFlow 설치 문서에서는 v2를 지원한다고 한다. 그러나 최신 버전이어도 관계는 없는듯 하다. v2는 [cuDNN Archive](#)에서 받을 수 있다.

설치

v4 (또는 v3)는 다음과 같이 설치하면 된다.

```
$ tar xvzf cudnn-7.0-linux-x64-v4.0-prod.tgz
$ sudo mv cuda/include/cudnn.h /usr/local/cuda/include
$ sudo mv cuda/lib64/libcudnn* /usr/local/cuda/lib64
$ sudo chmod a+r /usr/local/cuda/lib64/libcudnn* # 권한 설정 (안해도 될 듯)
```

v2의 경우는 아래와 같이 설치하면 된다.

```
$ tar xvzf cudnn-6.5-linux-x64-v2.tgz
$ sudo cp cudnn-6.5-linux-x64-v2/cudnn.h /usr/local/cuda/include
$ sudo cp cudnn-6.5-linux-x64-v2/libcudnn* /usr/local/cuda/lib64
$ sudo chmod a+r /usr/local/cuda/lib64/libcudnn*
```

Theano

수식 및 행렬 연산을 위한 패키지로 `deep learning`에 유용한 기능을 제공한다. 또한 `Theano`를 기반으로 보다 쉽게 사용할 수 있는 `Lasagne`나 `Keras` 등의 패키지도 있다. 보다 자세한 내용은 [공식문서](#)를 참고하자.

요구사항

- Python 2 > 2.6 or Python 3 >= 3.3
- g++ >= 4.2, python-dev
- [NumPy](#) >= 1.7.1
- [SciPy](#) >= 0.11
- [BLAS](#)
- (optional)nose >= 1.3.0 (Theano test-suite 실행에 필요)

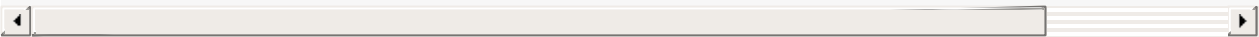
설치

설치는 아래와 같이 하면 된다.

```
sudo pip install Theano
```

`GitHub`에서 최신 버전(bleeding-edge)을 설치하고 싶다면 아래와 같이 하자.

```
sudo pip install --upgrade --no-deps git+git://github.com/Theano/Theano.git
```



사용 설정

[config – Theano Configuration](#)을 참고하자.

GPU 사용시 요구사항

- [NVIDIA CUDA Toolkit](#) (cuDNN과 호환되는 버전)
- [NVIDIA cuDNN Library](#) >= v3

Theano 설정은 `~/.theanorc` 에서 변경할 수 있다.

```
[global]
floatX = float32
device = gpu0
```

Lasagne

Lasagne는 neural network 를 만들고 학습하기 위한 Theano 기반의 가벼운 패키지이다.

보다 자세한 내용은 [공식문서](#)를 참고하자.

요구사항

- Python
- pip
- C compiler
- NumPy
- SciPy
- BLAS
- Theano

설치

아래와 같이 하면 의존 패키지와 함께 설치된다. 앞서 과정을 모두 수행했다면, 첫 번째 줄은 생략해도 된다.

```
$ sudo pip install -r https://raw.githubusercontent.com/Lasagne/Lasagne/master/requirements.txt
$ pip install Lasagne==0.1
```

nolearn

`nolearn`은 `scikit-learn` 과 연동되며, `Theano` , `Lasagne` 를 기반으로 `machine learning` 에 유용한 여러 모듈을 담고 있는 패키지이다. 보다 자세한 내용은 [공식문서](#)를 참고하자.

`pip install nolearn` 로 간단히 설치할 수 있으나, 버전이 구버전이 설치되므로 최신버전을 설치하기 위해서는 `Git` 을 이용한다.

설치

아래와 같이 하면 의존 패키지와 함께 설치된다. 앞서 과정을 모두 수행했다면, 첫 번째 줄은 생략해도 된다.

```
$ sudo pip install -r https://raw.githubusercontent.com/dnouri/nolearn/master/requirements.txt
$ sudo pip install git+https://github.com/dnouri/nolearn.git@master
```



TensorFlow

공식 홈페이지의 [Download and Setup](#)을 참고하자.

요구사항


1. NVIDIA CUDA Toolkit ≥ 7.0
2. NVIDIA cuDNN Library $\geq v2$
3. Python 2.7 (or 3.3+)
4. (Optional)Bazel

TensorFlow 설치

먼저 `pip` 와 `python-dev` 가 설치되어 있지 않다면 설치한다. `pip` 를 이용하여 아래와 같이 설치할 수 있다. 우리는 `GPU` 를 사용할 것이므로, 두 번째 방법으로 설치한다.

```
# Ubuntu/Linux 64-bit, CPU only:
$ sudo pip install --upgrade https://storage.googleapis.com/tensorflow/tensorflow-1.0.1-cp27-none-linux_x86_64.whl

# Ubuntu/Linux 64-bit, GPU enabled:
$ sudo pip install --upgrade https://storage.googleapis.com/tensorflow/tensorflow-1.0.1-cp27-cp27abi-linux_x86_64.whl
```



Caffe

Prerequisites

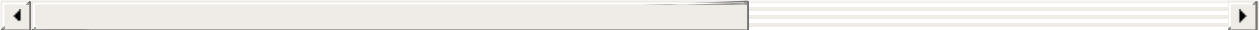
- [Boost](#)
- [OpenCV](#)

Clone GitHub repository

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

General dependencies

```
$ sudo apt-get install libprotobuf-dev libleveldb-dev libsnappy-dev
```



Remaining dependencies

```
$ sudo apt-get install libgflags-dev libgoogle-glog-dev liblmdb-dev
```

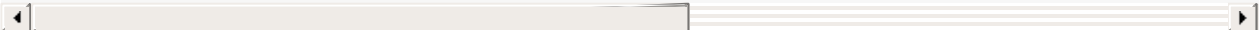


Python dependencies

```
$ cd python
$ for req in $(cat requirements.txt); do pip install $req; done
```

충돌을 방지하기 위해 파이썬과 `numpy` 경로 관련 부분을 수정한다.

```
$ sudo ln -s /usr/include/python2.7/ /usr/local/include/python2.7
$ sudo ln -s /usr/local/lib/python2.7/dist-packages/numpy/core/inc
```



Compilation

```
$ cp Makefile.config.example Makefile.config
# Adjust Makefile.config (for example, if using Anaconda Python, on
$ make pycaffe
$ make all
$ make test
$ make runtest
```

Makefile.config 은 아래의 내용을 수정한다.

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

...

# open for OpenBlas
BLAS := open
# Custom (MKL/ATLAS/OpenBLAS) include and lib directories.
# Leave commented to accept the defaults for your choice of BLAS
# (which should work)!
BLAS_INCLUDE := /opt/OpenBLAS/include
BLAS_LIB := /opt/OpenBLAS/lib

...

PYTHON_INCLUDE := /usr/local/include/python2.7 \
                /usr/local/include/python2.7/numpy
```

Install

파이썬 패키지 설치

```
$ sudo ln -s /home/${USER}/Downloads/setup/caffe/python/caffe/ /usr
```

바이너리 파일 복사

```
$ sudo ln -s /home/${USER}/Downloads/setup/caffe/build/tools/caffe
$ sudo ln -s /home/${USER}/Downloads/setup/caffe/build/lib/libcaffe
$ sudo ln -s /home/${USER}/Downloads/setup/caffe/build/lib/libcaffe
```

Test

python/caffe/io.py 수정

```
@@ -254,8 +254,14 @@ class Transformer:
        ms = (1,) + ms
        if len(ms) != 3:
            raise ValueError('Mean shape invalid')
-        if ms != self.inputs[in_][1:]:
-            raise ValueError('Mean shape incompatible with inputs')
+        #if ms != self.inputs[in_][1:]:
+        #    raise ValueError('Mean shape incompatible with inputs')
+        if ms != self.inputs[in_] :
+            print(self.inputs[in_])
+            in_shape = self.inputs[in_][1:]
+            m_min, m_max = mean.min(), mean.max()
+            normal_mean = (mean - m_min) / (m_max - m_min)
+            mean = resize_image(normal_mean.transpose((1,2,0)), in_shape)
        self.mean[in_] = mean

    def set_input_scale(self, in_, scale):
```

python/classifiy.py 수정

```
@@ -12,7 +12,7 @@ import glob
import time

import caffe
-
+import pandas as pd

def main(argv):
```

```

    pycaffe_dir = os.path.dirname(__file__)
@@ -86,6 +86,16 @@ def main(argv):
    help="Image file extension to take as input when a directory
        "is given as the input file."
    )
+   parser.add_argument(
+       "--print_results",
+       action='store_true',
+       help="Write output text to stdout rather than serializing"
+   )
+   parser.add_argument(
+       "--labels_file",
+       default=os.path.join(pycaffe_dir, "../data/ilsvrc12/synset_
+       help="Readable label definition file."
+   )
    args = parser.parse_args()

    image_dims = [int(s) for s in args.images_dim.split(',')]
@@ -126,9 +136,21 @@ def main(argv):

    # Classify.
    start = time.time()
+   scores = classifier.predict(inputs, not args.center_only).flat
    predictions = classifier.predict(inputs, not args.center_only)
    print("Done in %.2f s." % (time.time() - start))

+   if args.print_results:
+       with open(args.labels_file) as f:
+           labels_df = pd.DataFrame([{'synset_id':l.strip().split('
+           labels = labels_df.sort('synset_id')['name'].values
+
+           indices = (-scores).argsort()[:5]
+           predictions = labels[indices]
+
+           meta = [(p, '%.5f' % scores[i]) for i,p in zip(indices, pr
+           print meta
+
    # Save
    print("Saving results into %s" % args.output_file)
    np.save(args.output_file, predictions)

```



테스트는 아래와 같이 하면 된다.

```
python python/classify.py --print_results examples/images/cat.jpg 1
```



Reference

- [Ubuntu Installation](#)
- [Installation](#)

Torch

설치

```
curl -s https://raw.githubusercontent.com/torch/ezinstall/master/install.sh | sh
git clone https://github.com/torch/distro.git ~/torch --recursive
cd ~/torch; ./install.sh
```

bash 설정 반영

```
source ~/.bashrc
```

실행은 터미널에서 `th` 를 입력하면 된다.

Lua 패키지 관리

```
$ luarocks install image
$ luarocks list
```

Reference

<http://torch.ch/docs/getting-started.html>

OpenBLAS

Ubuntu 14.04 LTS 에 기본적으로 설치된 BLAS (Basic Linear Algebra Subprograms)는 속도가 느리기 때문에, OpenBLAS 를 설치한다.

OpenBLAS 를 컴파일하기 위해서는 gfortran 이 필요하므로 아래와 같이 의존 패키지를 설치한다.

```
$ sudo apt-get install gfortran
```

아래와 같이 설치한다.

```
# GitHub에서 저장소를 복제한다.
$ git clone https://github.com/xianyi/OpenBLAS.git
$ cd OpenBLAS

# 현재 공사중이다. 최신 릴리즈 버전으로 checkout
$ git checkout v0.2.15

# -j 옵션은 동시에 처리할 job의 수, 보통 코어 또는 논리 프로세서 수로 설정
# NO_AFFINITY와 USE_OPENMP는 OpenMP를 사용하기 위해 설정 (설정하지 않을 경
$ make FC=gfortran -j8 NO_AFFINITY=1 USE_OPENMP=1
$ sudo make PREFIX=/opt/OpenBLAS install

# 라이브러리 설정
$ echo "/opt/OpenBLAS/lib" | sudo tee /etc/ld.so.conf.d/openblas.conf
$ sudo ldconfig

# OpenBLAS 라이브러리의 우선순위 설정 - SciPy 모듈 사용시 필요
$ sudo update-alternatives --install /usr/lib/libblas.so.3 libblas
```

OpenCV

OpenCV Documentation의 [Installation in Linux](#)를 참고하자.

요구사항

- GCC 4.4.x or later
- CMake 2.8.7 or higher
- Git
- GTK+2.x or higher, including headers (libgtk2.0-dev)
- pkg-config
- Python 2.6 or later and Numpy 1.5 or later with developer packages (python-dev, python-numpy)
- ffmpeg or libav development packages: libavcodec-dev, libavformat-dev, libswscale-dev
- [optional] libtbb2 libtbb-dev
- [optional] libdc1394 2.x
- [optional] libjpeg-dev, libpng-dev, libtiff-dev, libjasper-dev, libdc1394-22-dev

의존 패키지 설치

```
$ sudo apt-get install build-essential
$ sudo apt-get install cmake git libqt4-dev libgtk2.0-dev pkg-confi
$ sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev
```

다운로드, 컴파일, 설치

GitHub 의 OpenCV 저장소에서 복제한다.

```
$ git clone https://github.com/Itseez/opencv.git
$ git clone https://github.com/Itseez/opencv_contrib.git
```

컴파일을 임시 디렉토리를 생성한다.

```
$ cd opencv
$ mkdir build
$ cd build
```

CUDA_ARCH_BIN , CUDA_ARCH_PTX 는 <https://developer.nvidia.com/cuda-gpus>의 Compute Capability 를 참고하여 본인의 GPU 에 따라 바꿔준다.

```
$ cmake -D CMAKE_BUILD_TYPE=Release -D CMAKE_INSTALL_PREFIX=/usr/local \
-D BUILD_TIFF=ON -D BUILD_EXAMPLES=OFF \
-D CUDA_GENERATION=Auto -D CUDA_ARCH_BIN=5.2 -D CUDA_ARCH_PTX=5.2 \
-D CUDA_FAST_MATH=1 -D WITH_CUBLAS=1 \
-D ENABLE_FAST_MATH=1 \
-D WITH_OPENGL=ON \
-D WITH_TBB=ON \
-D WITH_QT=ON -D WITH_OPENGL=ON ..

$ make -j8 # 컴파일
$ sudo make install # 설치
$ echo "/usr/local/lib" | sudo tee -a /etc/ld.so.conf.d/opencv.conf
$ sudo ldconfig
```

Reference

- [Install OpenCV 3.0 and Python 2.7+ on Ubuntu](#)

Boost

Boost 1.60.0 의 Caffe 와의 호환성 문제로 인해 1.59.0 을 설치한다.

```
$ curl -O http://jaist.dl.sourceforge.net/project/boost/boost/1.59.0/boost_1_59_0.tar.gz
$ tar xvf boost_1_59_0.tar.gz
$ cd boost_1_59_0
$ ./bootstrap.sh --with-python=python
$ sudo ./b2 -j8 install
```

DLib

```
curl -O http://dlib.net/files/dlib-18.18.tar.bz2  
tar xvfz dlib-18.18.tar.bz2
```

```
sudo python setpy.py install --yes USE_AVX_INSTRUCTIONS
```

OpenFace

OpenFace

OpenFace is a `Python` and `Torch` implementation of face recognition with deep neural networks and is based on the CVPR 2015 paper [FaceNet: A Unified Embedding for Face Recognition and Clustering](#) by Florian Schroff, Dmitry Kalenichenko, and James Philbin at Google.

Dependencies

- OpenCV (for python)
- DLib (for python)
- Torch
- numpy, pandas, scipy, scikit-learn, scikit-image (python dependencies)
- dpnn nn optim csvigo cunn fblualib torchx (Torch dependencies)

```
# clone repository
$ git clone --recursive https://github.com/cmusatyalab/openface.git

# install the dependencies (Torch)
$ luarocks install dpnn nn optim csvigo cunn fblualib torchx

# install OpenFace
$ cd openface
$ sudo python setup.py install
```

Test

```
$ cd models
$ ./get-models.sh
$ cd ..
$ ./run-tests.sh
```

Demo

```
$ python ./demos/compare.py images/examples/{lennon*,clapton*}
```

fix permission

Error

```
OpenFace: `openface_server.lua` subprocess has died.
```

```
$ cd /usr/local/lib/python2.7/dist-packages/openface  
$ sudo chmod +r *
```

OpenFace

FAQ

```
ImportError: libcudart.so.7.5: cannot open shared object file: No  
such file or directory
```

```
sudo ldconfig /usr/local/cuda/lib64
```