A guide to building Caffe from source with CUDA and Python support

This tutorial has been put together in order to assist the students of the Autonomous Drones Lab to set up Caffe quickly and easily. If you come across problems with these instructions, need assistance or want to suggest improvements, please address your requests to maayantamari@mail.tau.ac.il.

Environment/ Prerequisites

- Ubuntu 16.04
- NVIDIA GPU
- CUDA 9.0
- Python 3.5/3.6
- Git

Pre-installation steps

Step 1: Install dependencies

Since this guide covers installing Caffe with a Python interface, we will need to install dependencies for Caffe, and dependencies for PyCaffe. This step covers Caffe dependencies. There is another section for PyCaffe dependencies. If you do not need the Python interface, you can skip all the steps regarding Python.

General dependencies:

```
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
```

BLAS:

Install **ATLAS** by

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

Alternatively, install **OpenBLAS** by

```
sudo apt-get-install libopenblas-dev
```

Boost:

Make sure that you are using version 1.55 or newer.

```
dpkg -s libboost-dev | grep Version
```

If not, please update **boost** before proceeding.

PYTHON_DEV:

python_dev provides the Python headers for building the **pycaffe** interface

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

Additional dependencies:

The next few dependencies are from the **commonly encountered build issues** page on GitHub. I find it easier to check them beforehand than encounter make errors later on.

GLOG:

Check that **libgoogle-glog-dev** is installed.

```
dpkg -s libgoogle-glog-dev | grep Version
```

If it is not installed, you can install it by running

```
sudo apt-get install libgoogle-glog-dev
```

GFLAGS:

Check that **libgflags-dev** is installed.

```
dpkg -s libgflags-dev | grep Version
```

If it is not, you can install it by running

```
sudo apt-get install libgflags-dev
```

HDF5:

Check if **libhdf5-dev** is installed.

```
dpkg -s libhdf5-dev | grep Version
```

If it is not, you can install it by running

```
sudo apt-get install -y libhdf5-dev
```

If you installed it now, remember that you will need to adjust the make file later on.

LMDB:

Check if **liblmdb-dev** is installed.

```
dpkg -s liblmdb-dev | grep Version
```

If it is not, you can install it by running

```
sudo apt-get install liblmdb-dev
```

Installation steps

Step 1: Clone Caffe from GitHub

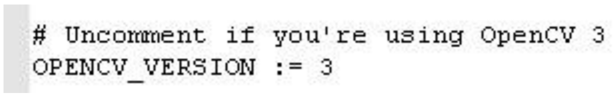
```
git clone https://github.com/BVLC/caffe.git
cd caffe
```

Step 2: Adjust the Makefile

Inside the folder, you will find a **makefile.config.example** file. We will use that with some adjustments. You will need to uncomment the lines that are relevant to your system.

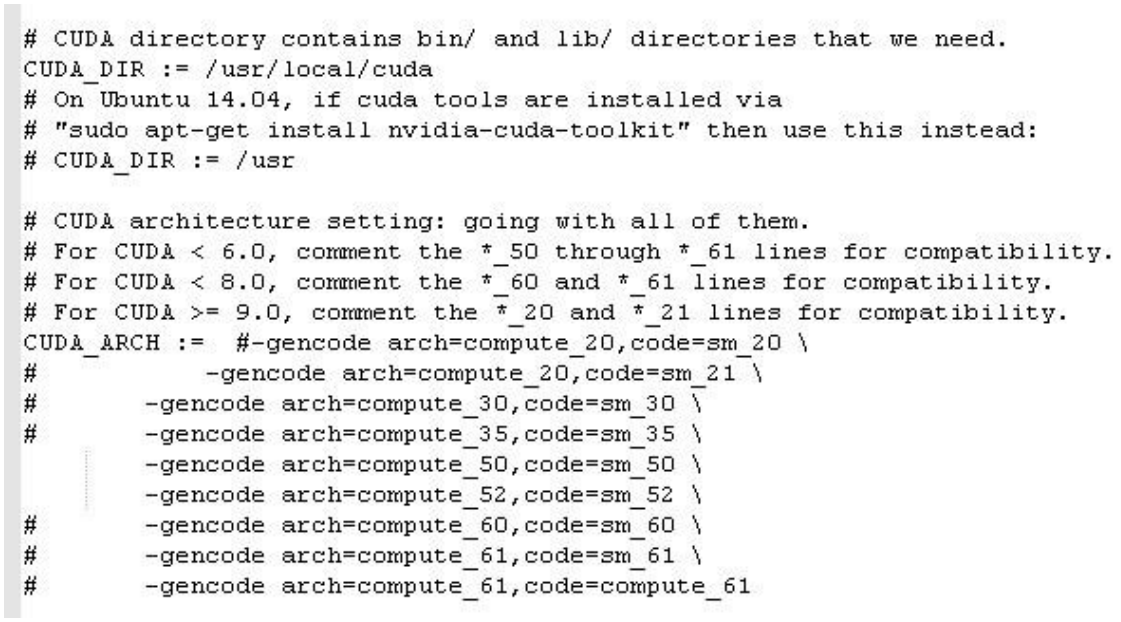
Makefile adjustments:

- If you want to use Caffe with OpenCV, you can uncomment this line.



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

- For CUDA compatibility, first check the **CUDA_DIR** path. CUDA should be installed in **usr/local**. You also need to comment the architectures that are not relevant to your CUDA version. If you know your specific architecture, you can comment everything else, but if you do not, just comment as explained in the Makefile example.



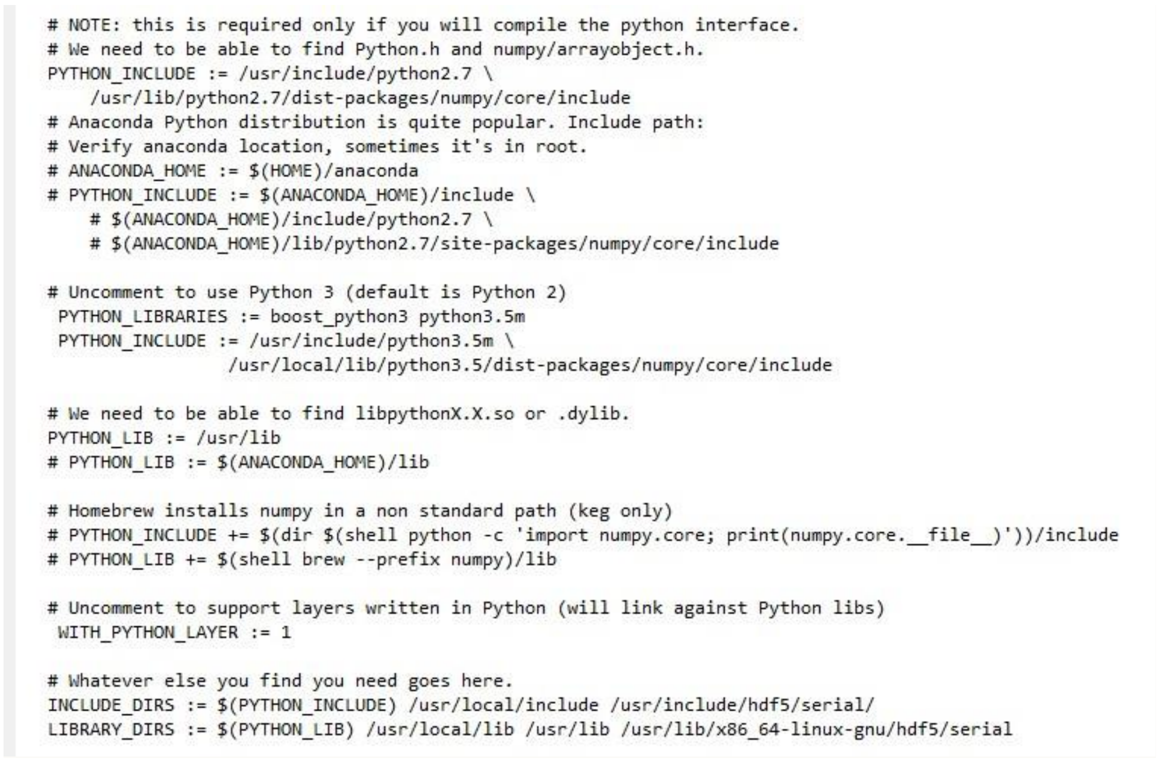
```
# CUDA directory contains bin/ and lib/ directories that we need.
CUDA_DIR := /usr/local/cuda
# On Ubuntu 14.04, if cuda tools are installed via
# "sudo apt-get install nvidia-cuda-toolkit" then use this instead:
# CUDA_DIR := /usr

# 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_20,code=sm_20 \
#           -gencode arch=compute_20,code=sm_21 \
#       -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
```

- For Python compatibility, you will need to make sure that all paths are correct for your version of Python.

  Note 1: If you are using Python 3, you can comment the PYTHON_INCLUDE lines of python2.7. I did not notice it at the time of installation, and it did not cause any problem to leave in uncommented.

  Note 2: If you installed libhdf5_dev add the path to **hdf5/serial** to **INCLUDE_DIRS** and **LIBRARY_DIRS**. You can see it in the image below.



```
# NOTE: this is required only if you will compile the python interface.
# We need to be able to find Python.h and numpy/arrayobject.h.
PYTHON_INCLUDE := /usr/include/python2.7 \
    /usr/lib/python2.7/dist-packages/numpy/core/include
# Anaconda Python distribution is quite popular. Include path:
# Verify anaconda location, sometimes it's in root.
# ANACONDA_HOME := $(HOME)/anaconda
# PYTHON_INCLUDE := $(ANACONDA_HOME)/include \
    # $(ANACONDA_HOME)/include/python2.7 \
    # $(ANACONDA_HOME)/lib/python2.7/site-packages/numpy/core/include

# Uncomment to use Python 3 (default is Python 2)
 PYTHON_LIBRARIES := boost_python3 python3.5m
 PYTHON_INCLUDE := /usr/include/python3.5m \
                 /usr/local/lib/python3.5/dist-packages/numpy/core/include

# We need to be able to find libpythonX.X.so or .dylib.
PYTHON_LIB := /usr/lib
# PYTHON_LIB := $(ANACONDA_HOME)/lib

# Homebrew installs numpy in a non standard path (keg only)
# PYTHON_INCLUDE += $(dir $(shell python -c 'import numpy.core; print(numpy.core.__file__)'))/include
# PYTHON_LIB += $(shell brew --prefix numpy)/lib

# Uncomment to support layers written in Python (will link against Python libs)
 WITH_PYTHON_LAYER := 1

# 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/x86_64-linux-gnu/hdf5/serial
```

Save your modified file to a new file and call it **makefile.config**

Step 3: Make the build

Execute

```
make all
```

Note: If you have an issue regarding **opencv_imgcodecs opencv_videoio** like this one:

```
/usr/bin/ld: cannot find -lopencv_imgcodecs
/usr/bin/ld: cannot find -lopencv_videoio
collect2: error: ld returned 1 exit status
Makefile:579: recipe for target '.build_release/lib/libcaffe.so.1.0.0-rc5'
failed
make: *** [.build_release/lib/libcaffe.so.1.0.0-rc5] Error 1
```

you should open your Makefile (NOT Makefile.config!) with some text editor, locate the LIBRARIES += line. It should be somewhere around line 164 (it was in line 164 for the person who wrote the build issues page. For me it was in line 181).

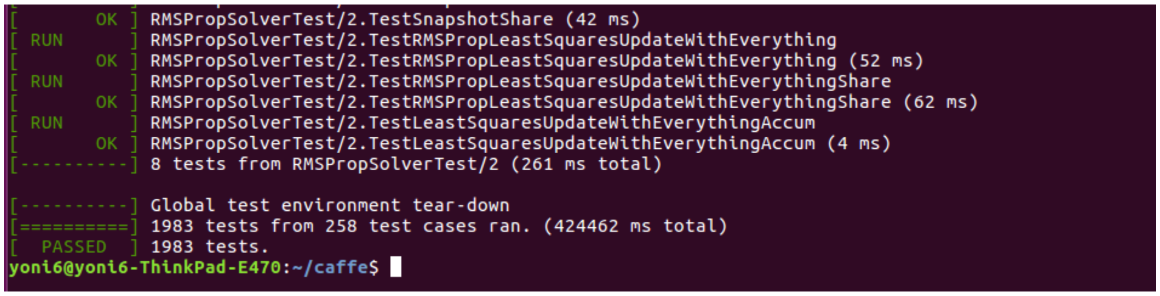
Add opencv_imgcodecs below it.

```
LIBRARIES += glog gflags protobuf leveldb snappy \
  lmdb boost_system hdf5_hl hdf5 m \
  opencv_core opencv_highgui opencv_imgproc opencv_imgcodecs
```

If make all finished without errors, you can continue to make the test.

```
make test
make runtest
```

At the end of **make runtest** you should see a test passed indicator.



```
[       OK ] RMSPropSolverTest/2.TestSnapshotShare (42 ms)
[ RUN      ] RMSPropSolverTest/2.TestRMSPropLeastSquaresUpdateWithEverything
[       OK ] RMSPropSolverTest/2.TestRMSPropLeastSquaresUpdateWithEverything (52 ms)
[ RUN      ] RMSPropSolverTest/2.TestRMSPropLeastSquaresUpdateWithEverythingShare
[       OK ] RMSPropSolverTest/2.TestRMSPropLeastSquaresUpdateWithEverythingShare (62 ms)
[ RUN      ] RMSPropSolverTest/2.TestLeastSquaresUpdateWithEverythingAccum
[       OK ] RMSPropSolverTest/2.TestLeastSquaresUpdateWithEverythingAccum (4 ms)
[----------] 8 tests from RMSPropSolverTest/2 (261 ms total)

[----------] Global test environment tear-down
[==========] 1983 tests from 258 test cases ran. (424462 ms total)
[  PASSED  ] 1983 tests.
yoni6@yoni6-ThinkPad-E470:~/caffe$ 
```

Step 4: Set up the Python interface

To install the PyCaffe interface, execute

```
make pycaffe
```

In order to be able to import Caffe into Python, you will need to install some Python dependencies. This can be done easily with pip. All of the dependencies are listed a **requirements.txt** file inside the Caffe folder.

To install the dependencies:

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

Step 5: Add environment variables to .bashrc

After completing the installation, to be able to import the Caffe module into Python, add the module directory to your PYTHONPATH

For example, open **bashrc** with **nano**:

```
nano ~/.bashrc
```

Add to the end of the file:

```
export PYTHONPATH=/path/to/caffe/python:$PYTHONPATH
```

Save your changes and exit. Do not forget to close the terminal or source the changes you made with

```
source ~/.bashrc
```

Step 6: Verify the installation from python

Open Python and try to import Caffe

```
Python 3.6.5 |Anaconda, Inc.| (default, Apr 29 2018, 16:14:56)
[GCC 7.2.0] on linux
Type "help", "copyright", "credits" or "license" for more
information.
```

```
>>> import caffe
```

Sources:

This guide is based mostly on the official Caffe installation guide:

https://caffe.berkeleyvision.org/install_apt.html

In addition, the following link:

https://github.com/BVLC/caffe/wiki/Commonly-encountered-build-issues