Caffe

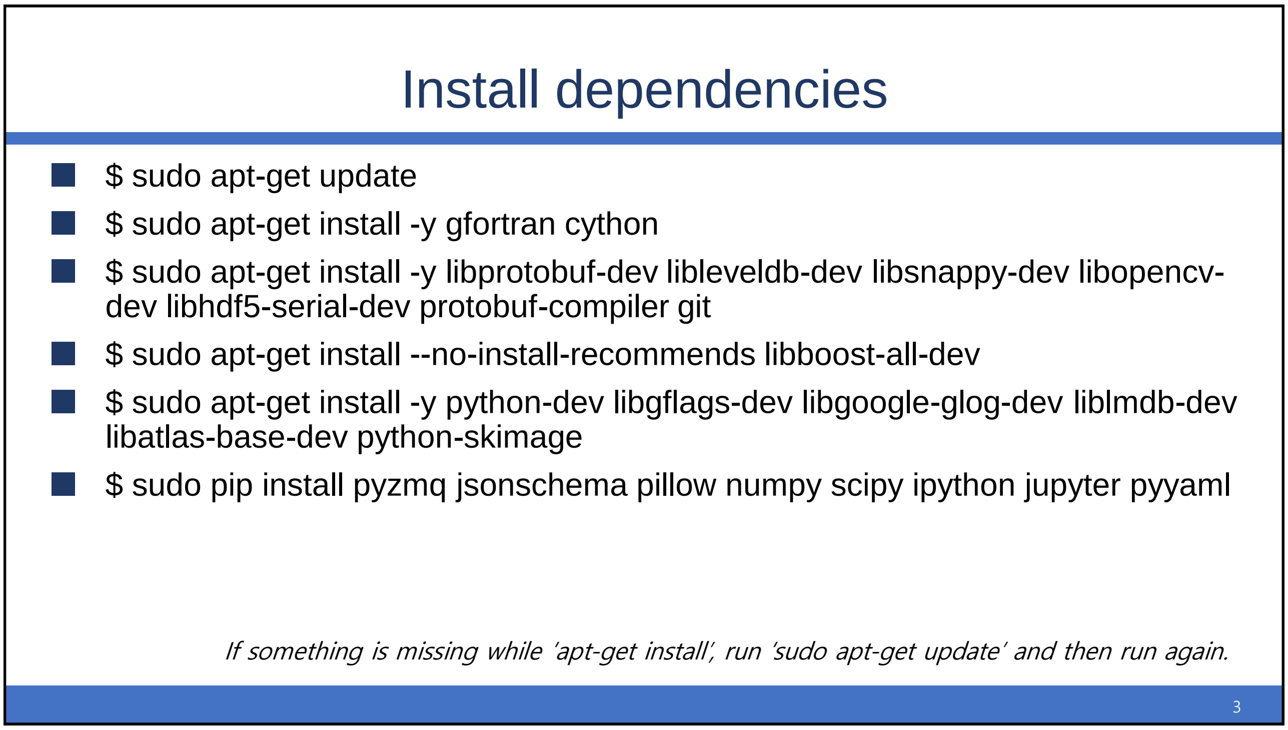
Caffe V1 on Raspberry Pi

- Convolutional Architecture for Fast Feature Embedding -

Aug. 2019

Ando Ki, Ph.D.
adki@future-ds.com

Contents

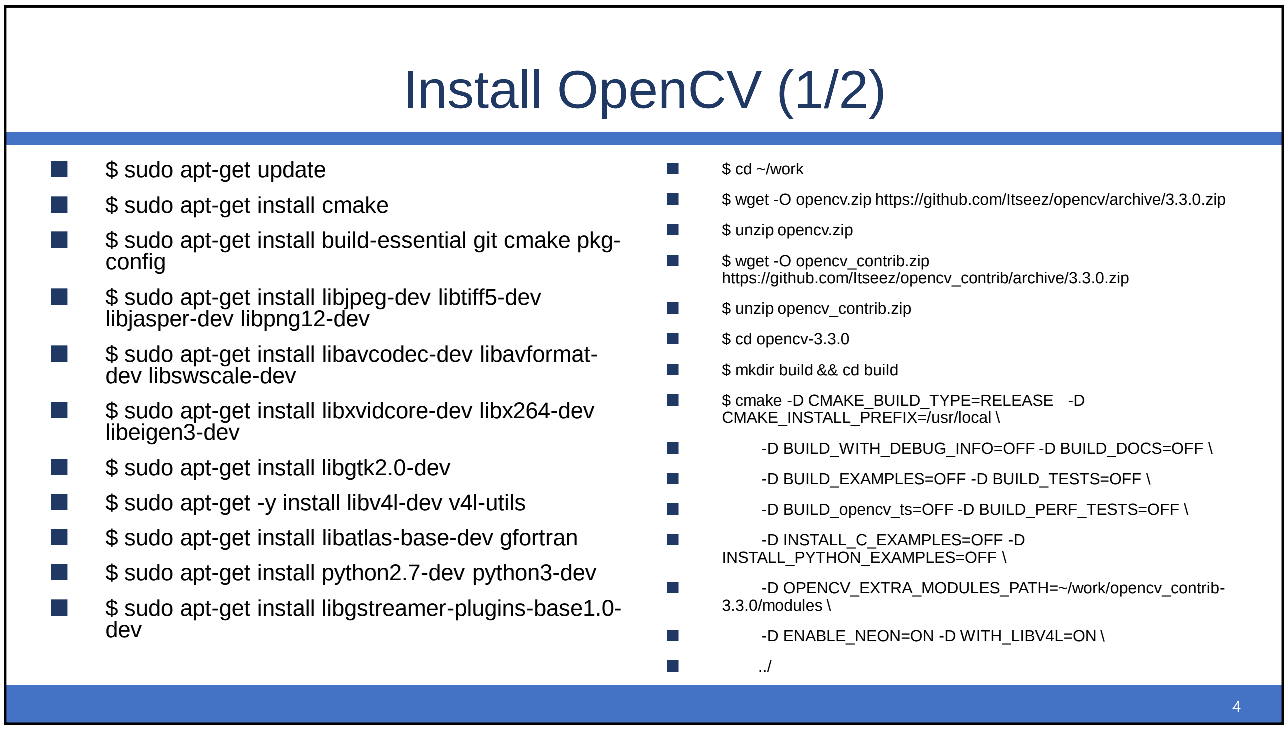


Install dependencies

- $ sudo apt-get update
- $ sudo apt-get install -y gfortran cython
- $ sudo apt-get install -y libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev libhdf5-serial-dev protobuf-compiler git
- $ sudo apt-get install --no-install-recommends libboost-all-dev
- $ sudo apt-get install -y python-dev libgflags-dev libgoogle-glog-dev liblmdb-dev libatlas-base-dev python-skimage
- $ sudo pip install pyzmq jsonschema pillow numpy scipy ipython jupyter pyyaml

If something is missing while 'apt-get install', run 'sudo apt-get update' and then run again.

3

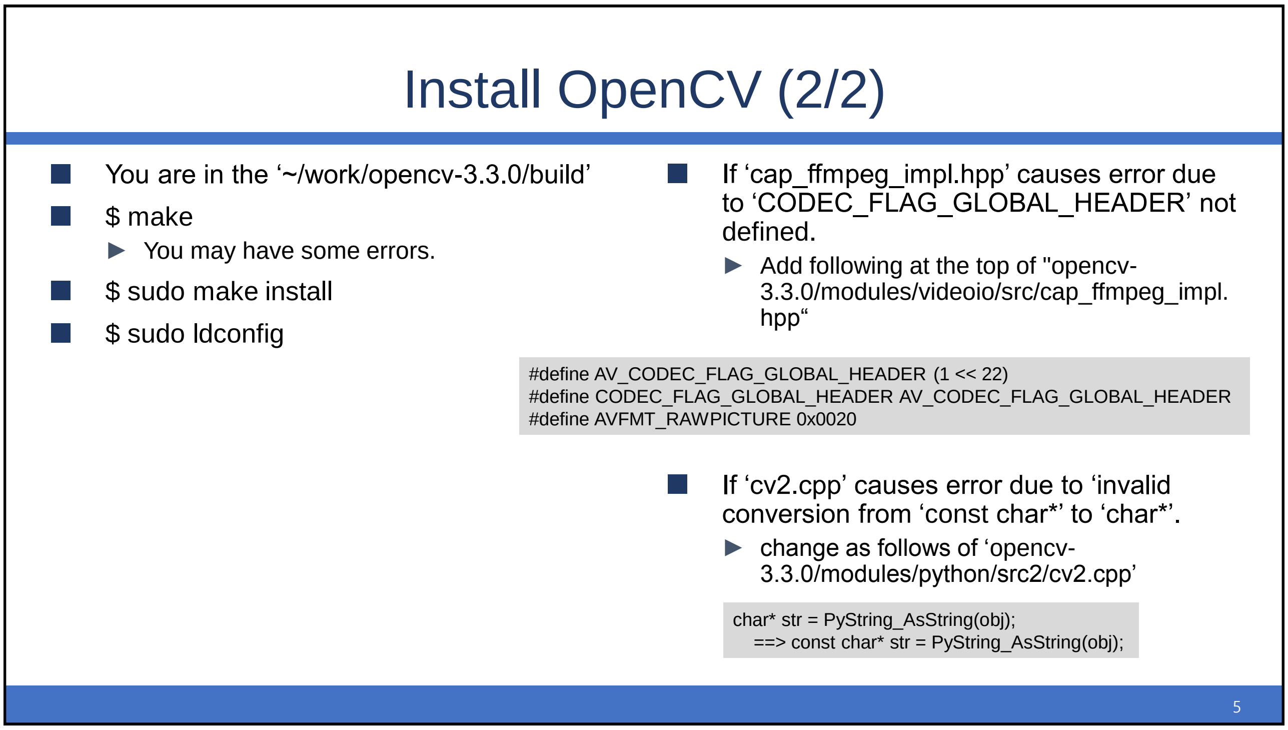


Install OpenCV (1/2)

- $ sudo apt-get update
- $ sudo apt-get install cmake
- $ sudo apt-get install build-essential git cmake pkg-config
- $ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
- $ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
- $ sudo apt-get install libxvidcore-dev libx264-dev libeigen3-dev
- $ sudo apt-get install libgtk2.0-dev
- $ sudo apt-get -y install libv4l-dev v4l-utils
- $ sudo apt-get install libatlas-base-dev gfortran
- $ sudo apt-get install python2.7-dev python3-dev
- $ sudo apt-get install libgstreamer-plugins-base1.0-dev

```
$ cd ~/work
$ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.3.0.zip
$ unzip opencv.zip
$ wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.3.0.zip
$ unzip opencv_contrib.zip
$ cd opencv-3.3.0
$ mkdir build && cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE   -D CMAKE_INSTALL_PREFIX=/usr/local \
        -D BUILD_WITH_DEBUG_INFO=OFF -D BUILD_DOCS=OFF \
        -D BUILD_EXAMPLES=OFF -D BUILD_TESTS=OFF \
        -D BUILD_opencv_ts=OFF -D BUILD_PERF_TESTS=OFF \
        -D INSTALL_C_EXAMPLES=OFF -D INSTALL_PYTHON_EXAMPLES=OFF \
        -D OPENCV_EXTRA_MODULES_PATH=~/work/opencv_contrib-3.3.0/modules \
        -D ENABLE_NEON=ON -D WITH_LIBV4L=ON \
        ../
```

4



Install OpenCV (2/2)

- You are in the '~/work/opencv-3.3.0/build'
- $ make
  - You may have some errors.
- $ sudo make install
- $ sudo ldconfig

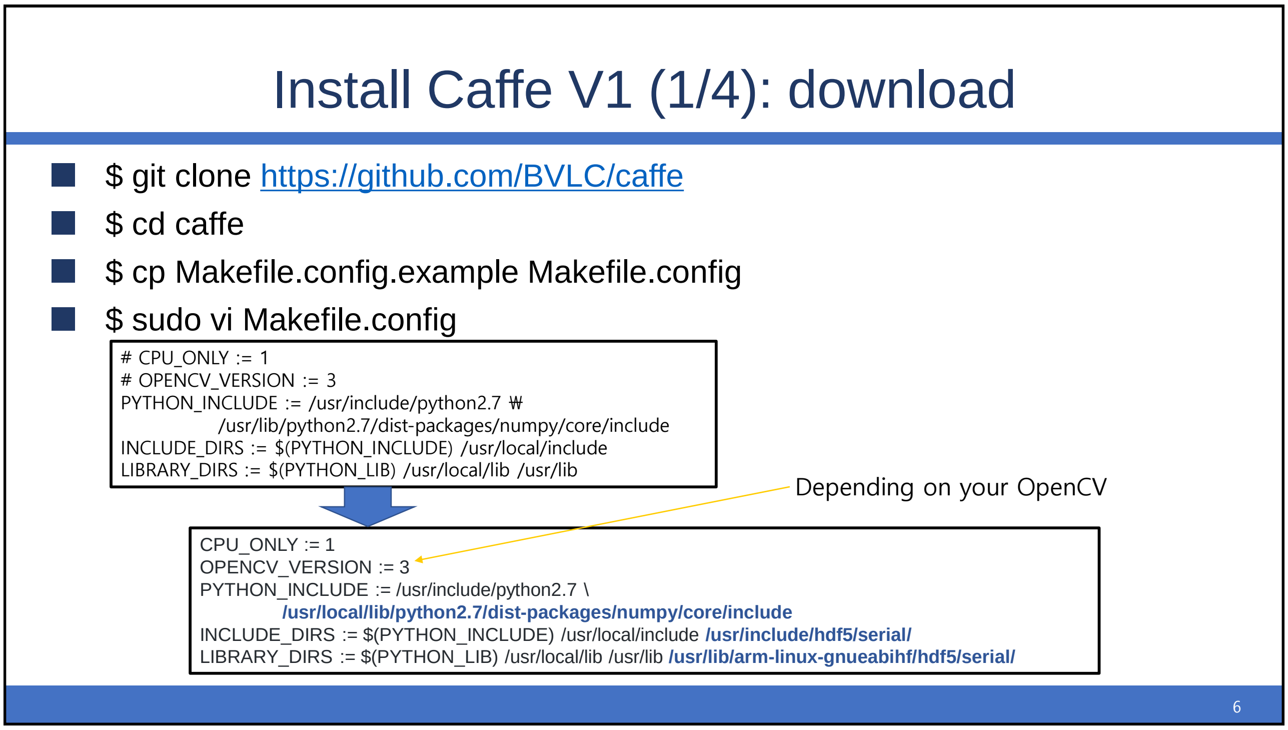
- If 'cap_ffmpeg_impl.hpp' causes error due to 'CODEC_FLAG_GLOBAL_HEADER' not defined.
  - Add following at the top of "opencv-3.3.0/modules/videoio/src/cap_ffmpeg_impl.hpp"

```
#define AV_CODEC_FLAG_GLOBAL_HEADER (1 << 22)
#define CODEC_FLAG_GLOBAL_HEADER AV_CODEC_FLAG_GLOBAL_HEADER
#define AVFMT_RAWPICTURE 0x0020
```

- If 'cv2.cpp' causes error due to 'invalid conversion from 'const char*' to 'char*'.
  - change as follows of 'opencv-3.3.0/modules/python/src2/cv2.cpp'

```
char* str = PyString_AsString(obj);
   ==> const char* str = PyString_AsString(obj);
```

5



Install Caffe V1 (1/4): download

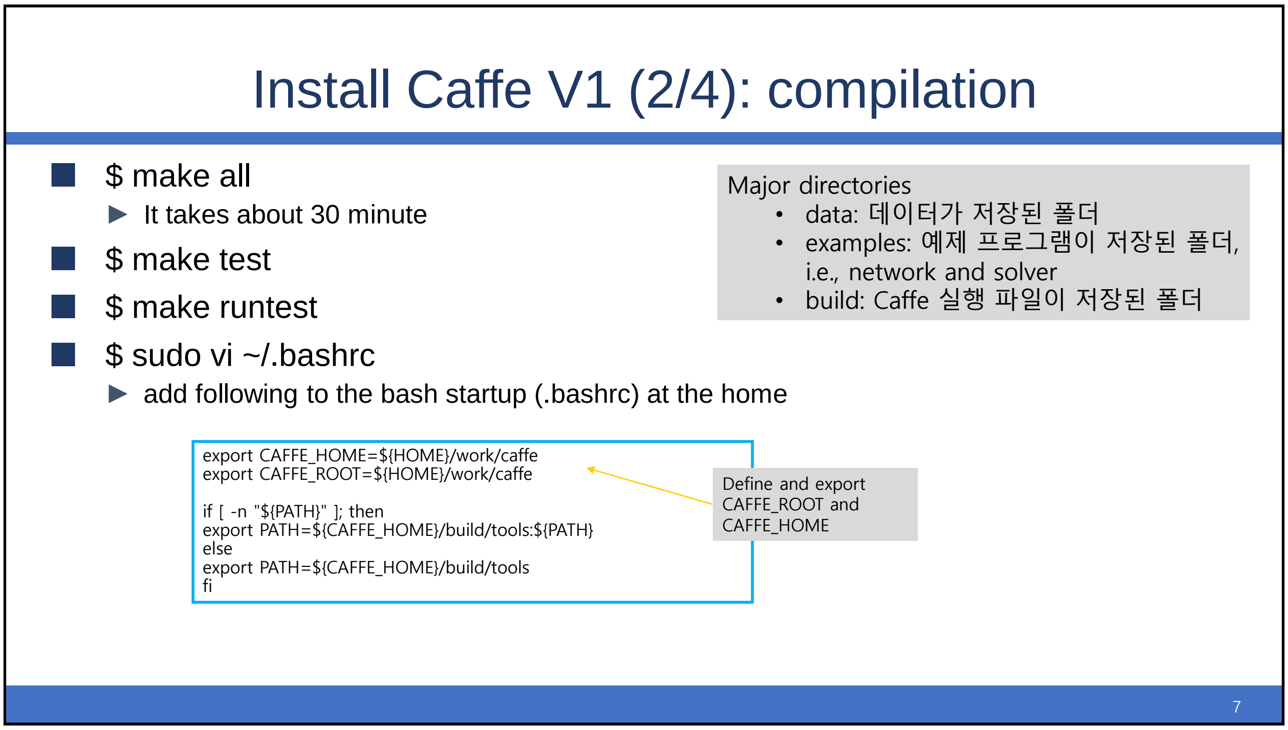
- $ git clone https://github.com/BVLC/caffe
- $ cd caffe
- $ cp Makefile.config.example Makefile.config
- $ sudo vi Makefile.config

```
# CPU_ONLY := 1
# OPENCV_VERSION := 3
PYTHON_INCLUDE := /usr/include/python2.7 ₩
              /usr/lib/python2.7/dist-packages/numpy/core/include
INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include
LIBRARY_DIRS := $(PYTHON_LIB) /usr/local/lib /usr/lib
```

Depending on your OpenCV

```
CPU_ONLY := 1
OPENCV_VERSION := 3
PYTHON_INCLUDE := /usr/include/python2.7 \
            /usr/local/lib/python2.7/dist-packages/numpy/core/include
INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include /usr/include/hdf5/serial/
LIBRARY_DIRS := $(PYTHON_LIB) /usr/local/lib /usr/lib /usr/lib/arm-linux-gnueabihf/hdf5/serial/
```

6



Install Caffe V1 (2/4): compilation

- $ make all
  - It takes about 30 minute
- $ make test
- $ make runtest
- $ sudo vi ~/.bashrc
  - add following to the bash startup (.bashrc) at the home

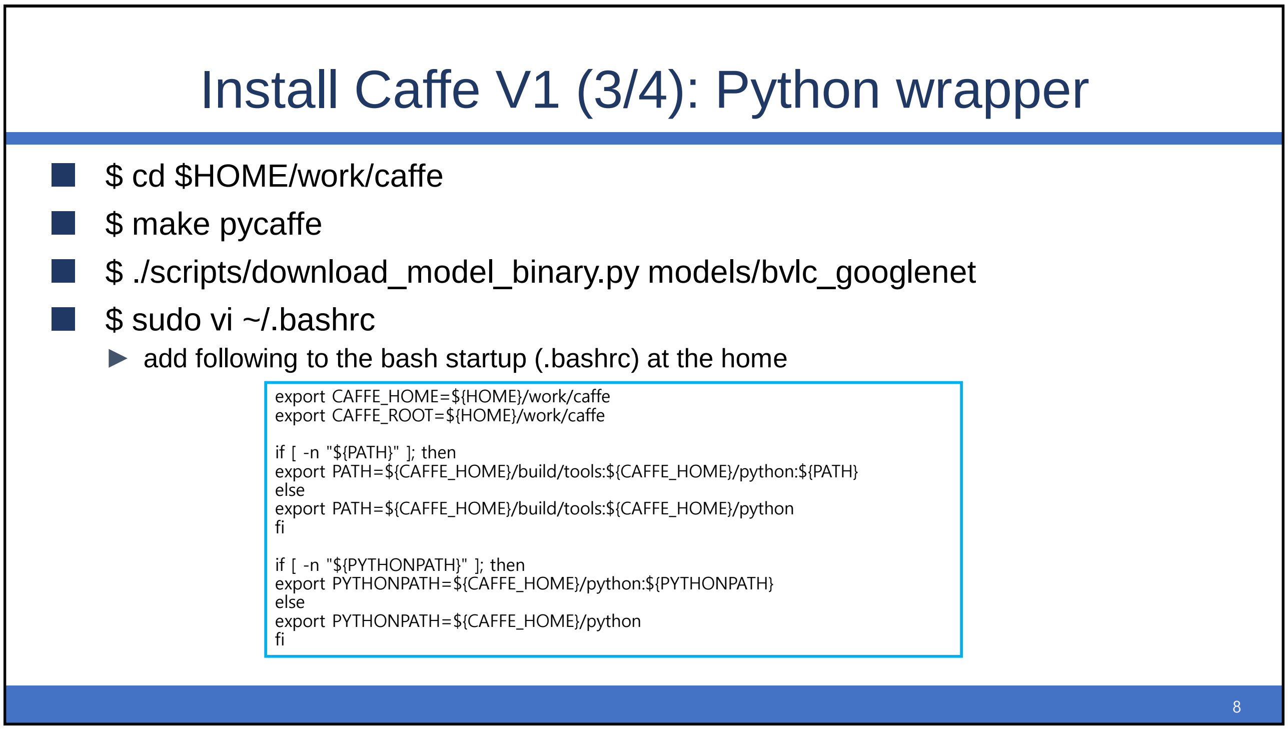
Major directories
- data: 데이터가 저장된 폴더
- examples: 예제 프로그램이 저장된 폴더, i.e., network and solver
- build: Caffe 실행 파일이 저장된 폴더

```
export CAFFE_HOME=${HOME}/work/caffe
export CAFFE_ROOT=${HOME}/work/caffe

if [ -n "${PATH}" ]; then
export PATH=${CAFFE_HOME}/build/tools:${PATH}
else
export PATH=${CAFFE_HOME}/build/tools
fi
```

Define and export CAFFE_ROOT and CAFFE_HOME

7



Install Caffe V1 (3/4): Python wrapper

- $ cd $HOME/work/caffe
- $ make pycaffe
- $ ./scripts/download_model_binary.py models/bvlc_googlenet
- $ sudo vi ~/.bashrc
  - add following to the bash startup (.bashrc) at the home

```
export CAFFE_HOME=${HOME}/work/caffe
export CAFFE_ROOT=${HOME}/work/caffe

if [ -n "${PATH}" ]; then
export PATH=${CAFFE_HOME}/build/tools:${CAFFE_HOME}/python:${PATH}
else
export PATH=${CAFFE_HOME}/build/tools:${CAFFE_HOME}/python
fi

if [ -n "${PYTHONPATH}" ]; then
export PYTHONPATH=${CAFFE_HOME}/python:${PYTHONPATH}
else
export PYTHONPATH=${CAFFE_HOME}/python
fi
```

8

Install Caffe V1 (4/4): Protobuf installation

- Install required packages for Python
  - $ cd ~/caffe_v1/caffe/python
  - $ sudo apt-get install python-pip
  - $ sudo pip install -r requirements.txt

- Not work
  - $ cd $HOME/work/caffe
  - $ cd python
  - $ python setup.py build
  - $ python setup.py google_test
  - $ sudo python setup.py install

9

Caffe command line options

- $ /home/pi/work/caffe/build/tools/caffe

```
usage: caffe <command> <args>

commands:
  train          train or finetune a model
  test           score a model
  device_query   show GPU diagnostic information
  time           benchmark model execution time

 Flags from tools/caffe.cpp:
   -gpu (Optional; run in GPU mode on given device IDs separated by ','.Use '-gpu all' to run on all available GPUs. The effective
        training batch size is multiplied by the number of devices.) type: string default: ""
   -iterations (The number of iterations to run.) type: int32 default: 50
   -level (Optional; network level.) type: int32 default: 0
   -model (The model definition protocol buffer text file.) type: string default: ""
   -phase (Optional; network phase (TRAIN or TEST). Only used for 'time'.) type: string default: ""
   -sighup_effect (Optional; action to take when a SIGHUP signal is received: snapshot, stop or none.) type: string default:
        "snapshot"
   -sigint_effect (Optional; action to take when a SIGINT signal is received: snapshot, stop or none.) type: string default: "stop"
   -snapshot (Optional; the snapshot solver state to resume training.) type: string default: ""
   -solver (The solver definition protocol buffer text file.) type: string default: ""
   -stage (Optional; network stages (not to be confused with phase), separated by ','.) type: string default: ""
   -weights (Optional; the pretrained weights to initialize finetuning, separated by ','. Cannot be set simultaneously with snapshot.)
        type: string default: ""
```

- '**.caffemodel**' file of shapshot: a output at a specific interval while training; a binary containing the current stat of the weights for each layer of the network.
- '**.solverstate**' file of snapshot: a binary contains the information required to continue training the model from where it last stopped.

(10)

Testing Python wrapper

```
$ source ~/.bashrc
$ python
>>> import caffe
>>> print caffe.__version__
1.0.0
>>> quit()
```

㈜퓨쳐디자인시스템
34051 대전광역시 유성구 문지로 193, KAIST 문지캠퍼스, F723호
(042) 864-0211~0212 / contact@future-ds.com / www.future-ds.com

Future Design Systems, Inc.
Faculty Wing F723, KAIST Munji Campus, 193 Munji-ro, Yuseong-gu, Daejeon 34051, Korea
+82-042-864-0211~0212 / contact@future-ds.com / www.future-ds.com

FUTURE
Design Systems