深層学習を用いた物体認識 RT コンポーネント

名城大学メカトロニクス工学科

ロボットシステムデザイン研究室

2017 年 11 月 27 日

目次

1. はじめに

1.1. コンポーネントの概要

本コンポーネントは予め構築した深層学習モデルを用いて，取得した画像を深層学習の処理に掛け物体認識を行う．また，深層学習を行うために画像処理分野に適した Caffe を使用し，深層学習の処理に高速性を求めるため NVIDIA を積んだ PC を使用する．

Caffe：処理の高速性とモジュール性を考慮した深層学習のオープンソース．主に画像処理分野において利用される．

1.2. 開発環境

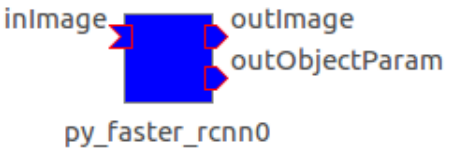
本 RTC の開発環境を表 1 に示す．

表 1　本 RTC 開発環境

OS	Ubuntu 16.04
CPU	Core i7(3.6GHz-4.2GHx)
コア数	4 コア/ 8 スレッド
メインメモリ	32GB
GPU	GeForce GTX 1080Ti
RT ミドルウェア	OpenRTM-aist-1.1.2
Caffe	py-faster-rcnn
CUDA	CUDA8.0
cuDNN	cuDNN_v5

2. RTC 仕様

2.1. インターフェース仕様

RTC の名称	
py_faster_rcnn	

入力ポート		
名称	データ型	説明
inImage	TimedCameraImage	カメラでキャプチャ下の RGB 画像を入力する

出力ポート		
名称	データ型	説明
outImage	TimedCameraImage	物体認識を描画した RGB 画像を出力する
outObjectParam	TimedObjectParam	認識した物体名と認識した物体の長方形描画パラメータ(5.1 参照)

コンフィグレーションパラメータ			
名称	データ型	デフォルト値	説明
mode	string	cpu	cpu, gpu 選択項目
net	string	zf	選択項目 zf　：　5 層の畳み込み層 vgg16：13 層の畳み込み層
dataset	string	voc	選択項目 voc：pascal_voc2007 coco：coco2014(5.2 参照)
recognitionRate	double	0.8	対象物の認識可能度合い

2.2. 認識できる対象物

pascal_voc2007 の認識可能対象物を表 2 に示し, coco2014 の認識可能対象物を表 3 に示す.

表 2 pascal_voc2007 の認識可能対象物

Aeroplane	Bicycle	Bird	Boat
Bottle	Bus	Car	Cat
Chair	Cow	Diningtable	Dog
Horse	Motorbike	Person	pottedplant
Sheep	Sofa	Train	Tvmonitor

表 3 coco2014 の認識可能対象物

Person	Bicycle	Car	Motorcycle
Airplane	Bus	Train	Truck
Boat	Traffic light	Fire hydrant	Stop sign
Parking meter	Bench	Bird	Cat
Dog	Horse	Sheep	cow
Elephant	Bear	Zebra	Giraffe
Backpack	Umbrella	Handbag	Tie
Suitcase	Frisbee	Skis	Snowboard
Sports ball	Kite	Baseball bat	Baseball glove
Skateboard	Surfboard	Tennis racket	Bottle
Wine glass	Cup	Fork	Knife
Spoon	Bowl	Banana	Apple
Sandwich	Orange	Broccoli	carrot
Hot dog	Pizza	Donut	Cake
Chair	Couch	Potted plant	Bed
Dining table	Toilet	Tv	Laptop
Mouse	Remote	Keyboard	Cellphone
Microwave	Oven	Toaster	Sink
Refrigerator	Book	Clock	Vase
Scissors	Teddy bear	Hair drier	toothbrush

2.3. 性能評価

認識対象物を鉢植えとし認識結果画像を図 1 に示す. 加えて, CPU, GPU で各ネットワークの計測を行い認識率の 10 回平均と 1 フレーム毎の処理時間を表 4 に示す.

図 1　認識結果画像

表 4　鉢植え計測結果

認識対象物 ： 鉢植え				
	CPU		GPU	
	zf	vgg16	zf	vgg16
認識率(%)	97.54	97.88	97.51	98.97
1 フレーム毎の処理時間 (sec)	4.8	15.3	0.3	0.43

3. RTC の導入・動作確認

3.1. Caffe インストール前準備

Caffe に必要なライブラリを事前にインストール.

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install build-essential cmake git pkg-config libjpeg8-dev ¥
libjasper-dev libpng12-dev libgtk2.0-dev ¥
libavcodec-dev libavformat-dev libswscale-dev libv4l-dev gfortran ¥
libtiff5-dev libatlas-base-dev -no-install-recommends libboost-all-dev ¥
ibgflags-dev libgoogle-glog-dev liblmdb-dev python2.7-dev
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python get-pip.py
$ sudo pip install cython easydict numpy
```

3.2. OpenCV インストール

OpenCV とは，画像や動画の処理をするのに様々な機能が実装されている．本 RTC でも画像を入力，加工する際に使用している．CUDA を入れる前に OpenCV を入れることで OpenCV の中の CUDA ライブラリのコンパイルをする必要がなく容易に入れることができる．CUDA を先にインストールした場合 cmake に CUDA のパスを通す必要がある(5.3 参照)．

```
$ cd
$ git clone https://github.com/Itseez/opencv.git
$ cd opencv
$ git checkout 3.0.0
$ cd
$ git clone https://github.com/Itseez/opencv_contrib.git
$ cd opencv_contrib
$ git checkout 3.0.0
$ cd ~/opencv
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE ¥
        -D CMAKE_INSTALL_PREFIX=/usr/local ¥
        -D INSTALL_C_EXAMPLES=ON ¥
        -D INSTALL_PYTHON_EXAMPLES=ON ¥
        -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules ¥
        -D BUILD_EXAMPLES=ON ..
```

```
$ make
$ sudo make install
$ sudo ldconfig
$ python
>>> import cv2
>>> cv2.__version__
```

3.3. GPU 環境化

GPU で Deep Learning を行うためには NVIDIA ドライバと CUDA, cuDNN をインストールする必要がある. NVIDIA ドライバ, CUDA インストールを失敗するとログイン画面ループや GUI が機能しなくなることがある. 以下に適切な NVIDIA ドライバから GPU 環境化での Caffe のインストールの手順を示す(5.4 参照).

3.3.1. NVIDIA ドライバ・CUDA インストール

事前準備として, NVIDIA ドライバと CUDA8.0, cuDNN_v5 for CUDA8.0 を NVIDIA の公式サイトからダウンロードする(5.5 参照). そして, Ubuntu はデフォルトの状態として nouveau グラフィックボードが有効となっているため無効にする必要がある.

NVIDIA ドライバ	http://www.nvidia.co.jp/Download/index.aspx?lang=jp
CUDA8.0	https://developer.nvidia.com/cuda-80-ga2-download-archive
cuDNN_v5 for CUDA8.0	https://developer.nvidia.com/rdp/cudnn-archive

NVIDIA ドライバ・CUDA のインストール前準備

```
$ sudo apt-get update
$ sudo apt-get install -y build-essential wget
$ sudo gedit /etc/modprobe.d/blacklist-nouveau.conf
  開いたファイルに下記の情報を書き込み, 保存する.
  blacklist nouveau
  options nouveau modeset=0
$ sudo update-initramfs -u
$ sudo reboot
```

セキュアブート無効後(5.6 参照), CUI(ctrl + alt + f1)でログインし, 実行権限追加

```
$ chmod a+x NVIDIA-Linux-x86_64-384.90.run
$ chmod a+x cuda_8.0.61_375.26_linux.run
```

X Server を停止(5.7 参照)

```
$ sudo service lightdm stop
```

opengl なしで NVIDIA ドライバをインストール

```
$ sudo ./NVIDIA-Linux-x86_64-384.90.run -no-opengl-files
```

CUDA を NVIDIA ドライバインストールなしでインストール

```
$ sudo ./cuda_8.0.61_375.26_linux.run --silent --no-opengl-libs --toolkit
```

環境変数追加

```
$ echo  'export PATH=/usr/local/cuda/bin:$PATH' >> ~/.bashrc
$ echo  'export
LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH' >>~/.bashrc
$ echo  'export CUDA_PATH=/usr/local/cuda' >> ~/.bashrc
$ source ~/.bashrc
$ sudo reboot
```

cuDNN_v5 をインストール

```
$ tar xvzf cudnn-8.0-linux-x64-v5.0-ga.tgz
$ sudo cp cuda/include/cudnn.h /usr/local/cuda/include
$ sudo cp cuda/lib64/* /usr/local/cuda/lib64
```

以上で NVIDIA ドライバ・CUDA・cuDNN インストール完了である.

3.3.2. Caffe(GPU)インストール

```
$ git clone https://github.com/BVLC/caffe.git
$ cd caffe
$ cp Makefile.config.example Makefile.config
$ gedit Makefile.config
```

Makefile.config 修正

5, 21 行目コメントアウト外す. 94, 95 行目パス追加.

```
5  USE_CUDNN := 1（コメントアウト外す）
...
21 OPENCV_VERSION := 3（コメントアウト外す）
...
94 INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include
/usr/local/include/hdf5/serial（パス追加）
```

```
95 LIBRARY_DIRS := $(PYTHON _LIB) /usr/local/lib /usr/lib /usr/lib/x86_64-linux-gnu /usr/lib/x86_64-linux-gnu/hdf5/serial（パス追加）
```

コンパイル

```
$ make all
$ make test
$ make runtest
```

Python で Caffe を使うための環境構築

```
$ sudo apt-get install python-dev python-pip python-numpy python-skimage gfortran
$ sudo pip install -r ~/caffe/python/requirements.txt
$ make pycaffe
$ gedit ~/.bashrc
```

環境設定

```
export PYTHONPATH=~/caffe/python/:$PYTHONPATH
```

.bashrc 更新

```
$ source ~/.bashrc
```

Caffe 動作確認

```
$ python
>>> import caffe
```

caffe が読み込まれれば Caffe の動作確認 OK

3.3.3. py-faster-rcnn(GPU)の導入

```
$ cd
$ git clone --recursive https://github.com/rbgirshick/py-faster-rcnn.git
```

Cython コンパイル

```
$ cd py-faster-rcnn/lib
$ make
```

Ubuntu16.04, CUDA8.0 使用する場合, py-faster-rcnn を github から更新する必要がある(5.8 参照).

```
$ cd ../caffe-fast-rcnn
$ git remote add caffe https://github.com/BVLC/caffe.git
$ git fetch caffe
$ git merge caffe/master
$ gedit include/caffe/layers/python_layer.hpp
```

include/caffe/layers/python_layer.hpp の修正

29 行目をコメントアウトする.

```
29 self_.attr("phase")=static_cast(this->phase_); (コメントアウト)
```

Makefile.config コピー

```
$ cp Makefile.config.example Makefile.config
$ gedit Makefile.config
```

Makefile.config を修正

5, 21, 87, 99 行目コメントアウト外す. 90, 91 パス追加.

```
5  USE_CUDNN := 1 (コメントアウト外す)
...
21 OPENCV_VERSION := 3 (コメントアウト外す)
...
92 WITH_PYTHON_LAYER := 1 (コメントアウト外す)
...
95 INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include
/usr/include/hdf5/serial (パス追加)
96 LIBRARY_DIRS := $(PYTHON _LIB) /usr/local/lib /usr/lib /usr/lib/x86_64-
linux-gnu /usr/lib/x86_64-linux-gnu/hdf5/serial (パス追加)
...
108 USE_PKG_CONFIG := 1 (コメントアウト外す)
```

コンパイル

```
$ make clean
$ make all && make pycaffe
```

深層学習モデルをダウンロード後, py-faster-rcnn(GPU)動作確認

```
$ cd ..
$ ./data/scripts/fetch_faster_rcnn_models.sh
$ ./tools/demo.py
```

以上で完了とする.

3.4. CPU 環境化

GPU 環境に適していない PC で扱う場合は以下の手順で行う.

3.4.1. Caffe(CPU) インストール

```
$ cd
$ git clone https://github.com/BVLC/caffe.git
$ cd caffe
$ cp Makefile.config.example Makefile.config
$ gedit Makefile.cofig
```

Makefile.config 修正

8, 21 行目コメントアウト外す.

```
8  CPU_ONLY := 1（コメントアウト外す）
21 OPENCV_VERSION := 3（コメントアウト外す）
```

コンパイル

```
$ make all
$ make test
$ make runtest
```

Python で Caffe を使うための環境構築

```
$ sudo apt-get install python-dev python-pip python-numpy python-skimage
gfortran
$ sudo pip install -r ~/caffe/python/requirements.txt
$ make pycaffe
$ gedit ~/.bashrc
```

環境設定

```
export PYTHONPATH=~/caffe/python/:$PYTHONPATH
```

./bashrc 更新

```
$ source ~/.bashrc
```

Caffe 動作確認

```
$ python
>>> import caffe
```

caffe が読み込まれれば Caffe の動作確認 OK

3.4.2. py-faster-rcnn(CPU)の導入

Github から「py-faster-rcnn」ダウンロード

```
$ mkdir py-faster-rcnn-cpu
$ cd py-faster-rcnn-cpu
$ git clone --recursive https://github.com/rbgirshick/py-faster-rcnn.git
$ cd py-faster-rcnn/lib
$ gedit setup.py
```

CPU モード用に setup.py を修正する.
58, 90, 125～141 行目コメントアウト.

```
58  # CUDA = locate_cuda()
...
90  # self.set_executable('compiler_so', CUDA['nvcc'])
...
125 #Extension('nms.gpu_nms',
126 #    ['nms/nms_kernel.cu', 'nms/gpu_nms.pyx'],
127 #    library_dirs=[CUDA['lib64']],
128 #    libraries=['cudart'],
129 #    language='c++',
130 #    runtime_library_dirs=[CUDA['lib64']],
131    # this syntax is specific to this build system
132    # we're only going to use certain compiler args with nvcc and not with
133    # gcc the implementation of this trick is in customize_compiler()
below
134 #    extra_compile_args={'gcc': ["-Wno-unused-function"],
135 #                        'nvcc': ['-arch=sm_35',
136 #                                 '--ptxas-options=-v',
137 #                                 '-c',
138 #                                 '--compiler-options',
139 #                                 "'-fPIC'"]},
140 #    include_dirs = [numpy_include, CUDA['include']]
141 #),
```

Cython module のコンパイル(5.9 参照)

```
$ make
```

ディレクトリ移動後 Makefile.config.example をコピーする.

```
$ cd ../caffe-fast-rcnn
$ cp Makefile.config.example Makefile.config
$ gedit Makefile.config
```

Makefile.config を修正

8, 21, 87,行目コメントアウト外す．91, 92 行目パス追加.

```
8  CPU_ONLY := 1（コメントアウト外す）
...
21 OPENCV_VERSION := 3（コメントアウト外す）
...
92 WITH_PYTHON_LAYER := 1（コメントアウト外す）
...
95 INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include
/usr/local/include/hdf5/serial（パス追加）
...
96 LIBRARY_DIRS := $(PYTHON _LIB) /usr/local/lib /usr/lib /usr/lib/x86_64-
linux-gnu /usr/lib/x86_64-linux-gnu/hdf5/serial（パス追加）
```

Caffe コンパイル

```
$ make clean
$ make -j8 && make pycaffe
$ gedit ../lib/fast_rcnn/nms_wrapper.py
```

lib/fast_rcnn/nms_wrapper.py を修正

9 行目コメントアウト.

```
9 from nms.gpu_nms import gpu_nms（コメントアウト）
```

深層学習モデルをダウンロード

```
$ cd ..
$ ./data/scripts/fetch_faster_rcnn_models.sh
```

認識対象物を鉢植えとし動作確認(5.10 参照)

```
$ ./tools/demo.py --cpu
```

以上で完了とする.

4. py_faster_rcnnRTC の導入

環境構築完了後, py_faster_rcnn に関連した RTC である「WebCamera」「ImageViwer」を github よりダウンロードし, py_faster_rcnnRTC を workspace 内に置く(5.11 参照).

加えて, py_faster_rcnn で認識した対象物のパラメータを視覚するための RTC「Show_ObjectParam」を workspace 内に置く.

RTC	ダウンロードサイト
WebCamera	https://github.com/rsdlab/WebCamera
ImageViewer	https://github.com/rsdlab/ImageViewer

py-faster-rcnn から py_faster_rcnnRTC に必要なライブラリフォルダをコピーする(5.12 参照).

```
$ cd
$ cp -r py-faster-rcnn/lib/fast_rcnn workspace/py_faster_rcnnRTC
$ cp -r py-faster-rcnn/lib/nms workspace/py_faster_rcnnRTC
$ cp -r py-faster-rcnn/lib/utils workspace/py_faster_rcnnRTC
$ cp -r py-faster-rcnn/lib/rpn workspace/py_faster_rcnnRTC
$ gedit workspace/py_faster_rcnnRTC/fast_rcnn/config.py
```

workspace/py_faster_rcnnRTC/fast_rcnn/config.py の修正

```
 190  __C.ROOT_DIR =
osp.abspath(osp.join(osp.dirname(__file__),'../../..', 'py-faster-
rcnn'))
```

Github から WebCameraRTC, ImageViewerRTC ダウンロード

```
$ cd workspace
$ git clone https://github.com/rsdlab/WebCamera
$ git clone https://github.com/rsdlab/ImageViewer
```

WebCameraRTC, ImageViewerRTC のコンパイル

```
$ cd WebCamera/
$ mkdir build
$ cd build
$ cmake ..
$ make
```

```
$ cd ../../ImageViewer
$ mkdir build
$ cd build
$ cmake ..
$ make
```

コンポーネント起動

```
$ cd
$ cd workspace/WebCamera/build/src
$ ./WebCameraComp
$ cd
$ cd workspace/ImageViewer/build/src
$ ./ImageViewerComp
$ cd
$ cd workspace/py_faster_rcnnRTC
$ python py_faster_rcnn.py
$ cd workspace/Show_ObjectParam
$ python Show_ObjectParam.py
```

eclipse を開き, RT System Editor で図 2 のように繋げる.

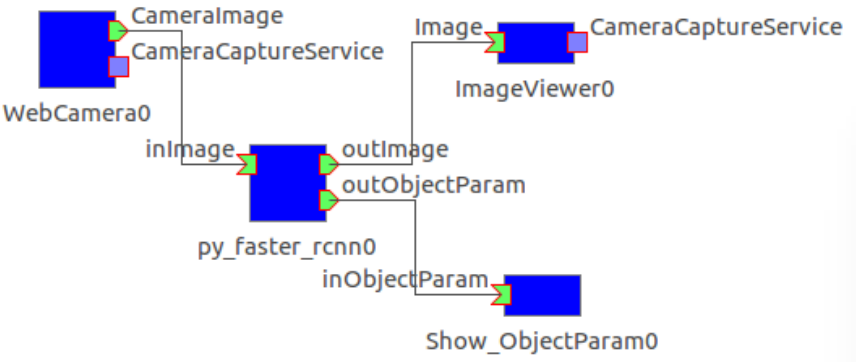


図 2　RT System Editor で RTC の連携

コンポーネントを実行する前に WebCameraRTC のコンフィグレーションの変更を行う.
camera.yml 配置ディレクトリ: ~/workspace/WebCamera/camera.yml

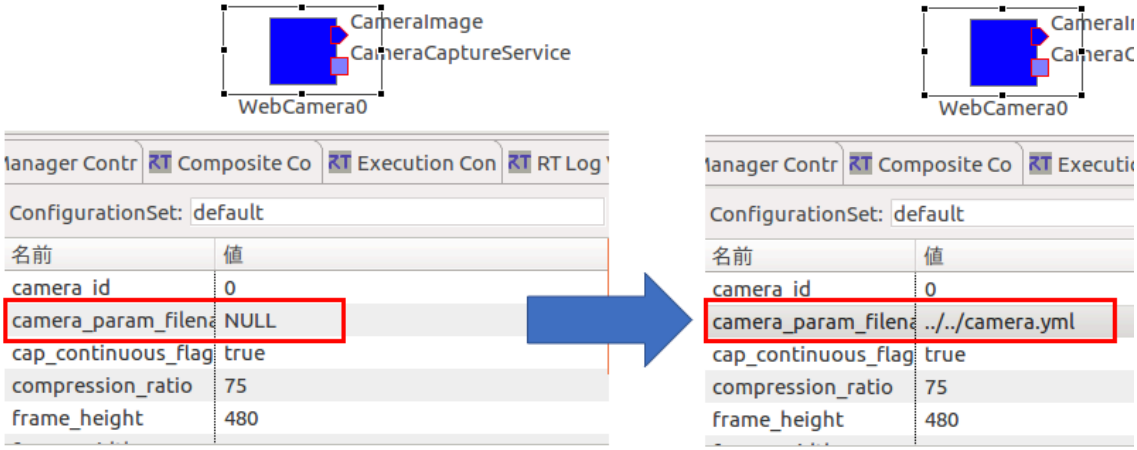


図 3　WebCameraRTC のコンフィグレーション変更

全体のコンポーネントを実行後, ImageViewerRTC と Show_ObjectParamRTC に以下のような結果が得られる.

図 4　ImageViewerRTC 出力結果

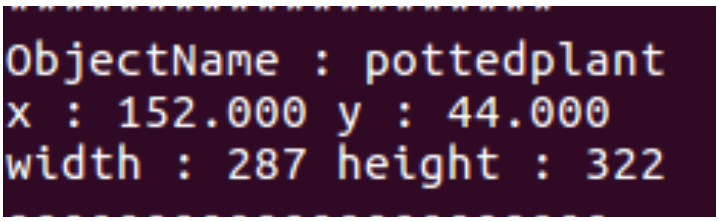


図 5 Show_ObjectParamRTC 出力結果

5. 注意・補足事項

5.1. 本 RTC の ObjectParamOutput は独自 IDL を用いている．詳細については別紙「物体認識インターフェース」を参照する．

5.2. coco データベースを用いる場合は，画像データと注釈ファイルをダウンロードを行い，学習させる必要がある．学習後の caffemodel ファイルを生成してからデータセットを行うようにする．Coco データベースの生成方法については別紙「py-faster-rcnn データベースの追加方法」を参照する．

5.3. CUDA を先にインストールした場合の OpenCV に CUDA のパスを通す必要がある．

```
$ mkdir opencv
$ cd opencv
$ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.2.0.zip
$ unzip opencv.zip
$ cd opencv-3.2.0/
$ mkdir build
$ cd build
```

cmake で CUDA 連携を ON にする

```
$ cmake -D CMAKE_BUILD_TYPE=RELEASE ¥
      -D CMAKE_INSTALL_PREFIX=/usr/local ¥
      -D INSTALL_C_EXAMPLES=OFF ¥
      -D INSTALL_PYTHON_EXAMPLES=ON ¥
      -D WITH_CUDA=ON ¥
      -D ENABLE_FAST_MATH=1 ¥
      -D CUDA_FAST_MATH=1 ¥
      -D WITH_CUBLAS=1 ..
$ make all
$ sudo make install
$ sudo /bin/bash -c  ‘echo  “/usr/local/lib” >
/etc/ld.so.conf.d/opencv.conf’
$ sudo ldconfig
```

5.4. Ubuntu の場合グラフィックボードの設定の際, [システム設定]→[ソフトウェアとアップデート]→[追加のドライバ]でグラフィックボードを切り替えることができるが, NVIDIA ドライバが適してインストールできないので行う必要がない.

5.5. CUDA インストーラには「.deb」「.run」ファイルがあるが, runfile を選択する. deb ファイルは，デフォルトとして NVIDIA ドライバを自動的にインストールしてしまうため GPU に適さないドライバをインストールされることがある.

5.6. Ubuntu14.04 の場合はセキュアブートを無効にせず NVIDIA ドライバのインストールを行えるが, Ubuntu16.04 の場合セキュアブートが有効時に NVIDIA ドライバのインストールを行うと以下のエラーが発生する

```
ERROR : Unable to load the ‘nvidia -drm’ kernel module
```

Ubuntu16.04 で NVIDIA ドライバをインストールする際，セキュアブートを無効にしなければインストールできない．再起動後, UEFI BIOS Utility を起動しセキュアブートを無効にする．以下に ASUS マザーボードの場合のセキュアブート無効までの手順を示す.

PC 起動時に F2 を押し, UEFI BIOS Utilitiy を起動

1. Advaced Mode を起動し, [Boot]タブの[Secure Boot]を選択
2. [Boot]タブの画面に戻り，[CSM(Compatibility Support Module)]を選択
3. [Launch CSM]の項目を[Dinabled]に設定する
4. すべての設定が完了後, [Exit]メニューの Save Changes & Reset を実行する

5.7. X Server を停止後, NVIDIA ドライバをインストール試合と NVIDIA ドライバインストールの開始が行われない．また，X Server を停止する際は必ず CUI 上で行う必要がある.

5.8. $ git merge する際, github アカウントを作成し，コンソール上に以下を入力し，サイド実行すると良い.

```
$ git config --global user.name アカウント名
$ git config --global user.email メールアドレス
$ git merge -X theirs caffe/master
```

nano エディタが開いたら, ctrl+X でそのまま閉じる.

また, include/caffe/layers/python_layer.hpp を編集せず，動作確認を行うと以下のようなエラーが発生する.

```
Traceback (most recent call last):
File “./tools/demo.py”,line 134,in
net = caffe.Net(prototxt,caffemodel,caffe.TEST)
AttributeError:can’t set attrubute
```

5.9. 以下のようなエラーが出る場合は，pip バージョンを 8 程度に上げるとよい.

エラー内容

```
  File "setup.py", line 10, in <module>
    from setuptools import setup

ImportError: No module named setuptools
```

pip をアップグレード

```
$ pushd ~/Downloads
$ sudo apt-get install curl
$ curl "https://bootstrap.pypa.io/get-pip.py" -o "get-pip.py"
$ sudo python get-pip.py
$ pip install --upgrade pip
```

setuptools, cython などをインストール

```
$ sudo pip install --upgrade pip
$ sudo pip install --upgrade setuptools
$ sudo pip install numpy
$ sudo pip install h5py
$ sudo pip install --upgrade cython
```

py-faster—rcnn/lib に戻ってコンパイル

```
$ popd
$ make
```

5.10. demo.py 起動もしくはコンポーネント起動後

```
[libprotobuf ERROR google/protobuf/descriptor_database.cc:57] File already
exists in database: caffe.proto
[libprotobuf FATAL google/protobuf/descriptor.cc:1080] CHECK failed:
generated_database_->Add(encoded_file_descriptor, size):
terminate called after throwing an instance of
 'google::protobuf::FatalException' what(): CHECK failed:
generated_database_->Add(encoded_file_descriptor,size):
Aborted
```

このようなエラーが発生した場合，以下のように.bash の PYTHONPATH の修正を行う

```
$ gedit ~/.bashrc
```

./bashrc 修正を行う

```
export PYTHONPATH=~/py-faster-rcnn-cpu/py-faster-rcnn/caffe-fast-
rcnn/python/:$PYTHONPATH
```

5.11. py_faster_rcnnRTC のディレクトリ設定

py_faster_rcnnRTC を「~/workspace/py_faster_rcnn」に置く必要がある．Workspace 以外のディレクトリに置く場合は「~/workspace/py_faster_rcnn/fast_rcnn/config.py」の 190 行目に変更を加える必要がある．

5.12. py_faster_rcnnRTC を起動するために rbgirshick 様から提供されているライブラリを使用しています．そちらのライセンスに関しては以下のファイルの指示に従ってください．

URL：https://github.com/rbgirshick/py-faster-rcnn/blob/master/LICENSE

6. 参考資料

- NVIDIA・CUDA インストール関連

Ubuntu14.04 に NVIDIA ドライバーをインストールしたら GUI ログインできなくなったときの話

URL：https://qiita.com/tanakatsu1080/items/c97c4ea3b1d349e2f718

Ubuntu16.04 に NVIDIA ドライバをインストールする方法

URL：http://blog.livedoor.jp/tsuchiyano/archives/52183842.html

GitHubGist wangrouchui/install NVIDIA Driver and CUDA.md

URL：https://gist.github.com/wangruohui/df039f0dc434d6486f5d4d098aa52d07

- py-faster-rcnn 導入関連

GitHub rbgirshick/py-faster-rcn

URL：https://github.com/rbgirshick/py-faster-rcnn

GitHub rbgirshick/py-faster-rcnn issues：support 1080P and cudnn v5?

URL：https://github.com/rbgirshick/py-faster-rcnn/issues/237

py-faster-rcnn 覚え書き

URL:https://www.cs.gunma-u.ac.jp/~nagai/wiki/index.php?py-faster-rcnn%20%B3%D0%A4%A8%BD%F1%A4%AD

Ubuntu16.04 + GT720M + Cuda8.0 + py-faster-rcnn(caffe)

URL：http://blog.csdn.net/nicky_lyu/article/details/53181434

Compiling and Running Faster R-CNN on Ubuntu (CPU Mode)

URL:https://chunml.github.io/ChunML.github.io/project/Running-Faster-RCNN-Ubuntu/

- OpenCV インストール関連

Ubuntu に OpenCV3.2 と contrib をインストールする

URL：http://shibafu3.hatenablog.com/entry/2017/03/28/164125

Ubuntu16.04 に OpenCV をインストール(CUDA8.0 + python3.5)

URL：http://mashup.hatenablog.com/entry/2017/05/16/200000

- Caffe インストール関連

Caffe 公式サイト

URL：http://caffe.berkeleyvision.org/

Ubuntu14.04 に Caffe をインストール(GPU 編)

URL：https://qiita.com/TD72/items/bcb243ee02760ea1d8bb

Caffe をインストールしてサンプルを動かすまで

URL：https://qiita.com/J-Holliday/items/1787a4b83826261c88ec