**Doc Processing Installation Guide**

Date: 20/09/2018

Ref doc: Doc Processing Installation Guide

**Table of Contents**

[I.Global Architecture 3](#__RefHeading___Toc1153_249089512)

[1. About Doc Processing 3](#__RefHeading___Toc1155_249089512)

[2. Architecture Schema 4](#__RefHeading___Toc1157_249089512)

[3. Infrastructure 4](#__RefHeading___Toc1159_249089512)

[II.Installation Guide 5](#__RefHeading___Toc1163_249089512)

[1. Installation the dependencies 5](#__RefHeading___Toc1165_249089512)

[2. Installation Python 5](#__RefHeading___Toc1167_249089512)

[3. Installation Python 3.5 5](#__RefHeading___Toc1169_249089512)

[4. Installation OpenCV 3.2 5](#__RefHeading___Toc1171_249089512)

[5. Installation Caffe 7](#__RefHeading___Toc1173_249089512)

[6. Installation Faster RCNN 10](#__RefHeading___Toc1175_249089512)

[7. Installation Tesseract 11](#__RefHeading___Toc1177_249089512)

[8. Installation clstm 11](#__RefHeading___Toc1179_249089512)

[9. Install rentree\_signnee 12](#__RefHeading___Toc917_2105605421)

**Revision History**

This section lists the history of revisions made to the document:

| Date | Version | Change Reference | Issued by |
| --- | --- | --- | --- |
| 27 |  |  |  |
| 20/09/2018 | 0 | Initial version | M. Cuong NGUYEN |
| 10/10/2018 | 1.0 | Version 1.0 | M. Cuong NGUYEN |
|  |  |  |  |
|  |  |  |  |

# Global Architecture

## 1. About Doc Processing

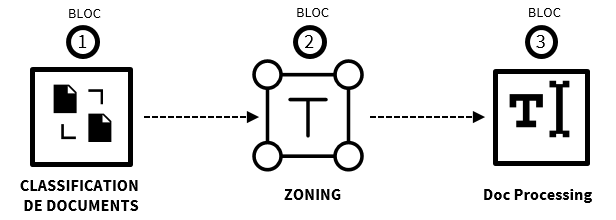
Document Processing is an area of AI that aims to convert handwritten text and / or electronic documents (scanned images of a document) into structured data.

Eligibility criteria:

* Image in a document or document scanned, faxed, scanned, ...
* Structure of the standard document
* Text preferably typed (by handwriting)
* To be studied on a case-by-case basis: presence of signature, checkbox

## 2. Architecture Schema

**Architecture :**



* 1. Global architecture
* Document classification: to determine the nature of the document (Which document is processed, which model will be applied.)
* Zoning: to automatically recognize the necessary text fields (Selecting the image areas to evaluate)
* Doc Processing: recognize optical characters, validate the content of the zone (content present, extracting customer data)

## 3. Infrastructure

**Production:**

|  |  |
| --- | --- |
| OS type | Ubuntu Server 16.04 |
| CPU | G2x2xlarge |
| Memory | 4 Go |
| Disk | 60 Go |

**Pre-required software:**

|  |  |
| --- | --- |
| Name | Version |
| Caffe | V1.0 |
| Cuda Toolkit | 8.0 |
| Python | 3.5 |
| Open CV | 3.2 |
| Tesseract | 3.04 |
| Faster RCNN | V1.0 |
| Clstm | V |

**List of application components :**

* Doc processing web-service client
* Doc processing web-service server

# Installation Guide

## 1. Installation the dependencies

sudo apt-get update

sudo apt-get upgrade

sudo apt-get install -y build-essential cmake git pkg-config

sudo apt-get install -y libprotobuf-dev libleveldb-dev libsnappy-dev libhdf5-serial-dev protobuf-compiler

sudo apt-get install -y libatlas-base-dev

sudo apt-get install -y --no-install-recommends libboost-all-dev

sudo apt-get install -y libgflags-dev libgoogle-glog-dev liblmdb-dev

## 2. Installation Python

sudo apt-get install -y python-pip

## 3. Installation Python 3.5

sudo apt-get install -y python3-dev

sudo apt-get install -y python3-numpy python3-scipy

## 4. Installation OpenCV 3.2

sudo apt-get install --assume-yes build-essential cmake git

sudo apt-get install --assume-yes pkg-config unzip ffmpeg qtbase5-dev python-dev python3-dev python-numpy python3-numpy

sudo apt-get install --assume-yes libopencv-dev libgtk-3-dev libdc1394-22 libdc1394-22-dev libjpeg-dev libpng12-dev libtiff5-dev libjasper-dev

sudo apt-get install --assume-yes libavcodec-dev libavformat-dev libswscale-dev libxine2-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev

sudo apt-get install --assume-yes libv4l-dev libtbb-dev libfaac-dev libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev

sudo apt-get install --assume-yes libvorbis-dev libxvidcore-dev v4l-utils python-vtk

sudo apt-get install --assume-yes liblapacke-dev libopenblas-dev checkinstall

sudo apt-get install --assume-yes libgdal-dev

In order to install the NVIDIA Cuda Toolkit with CUDNN library, see [https://github.com/BVLC/caffe/wiki/Ubuntu-16.04-or-15.10-Installation-Guide#the-gpu-support-prerequisites](https://github.com/BVLC/caffe/wiki/Ubuntu-16.04-or-15.10-Installation-Guide" \l "the-gpu-support-prerequisites)

Or

----------------------------------------- TO BE TEST-------------------------------------------------------

|  |
| --- |
| #!/bin/bash |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # install CUDA Toolkit v8.0 |
|  |

|  |
| --- |
| # instructions from https://developer.nvidia.com/cuda-downloads (linux -> x86\_64 -> Ubuntu -> 16.04 -> deb (network)) |
|  |

|  |
| --- |
| CUDA\_REPO\_PKG="cuda-repo-ubuntu1604\_8.0.61-1\_amd64.deb" |
|  |

|  |
| --- |
| wget http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86\_64/${CUDA\_REPO\_PKG} |
|  |

|  |
| --- |
| sudo dpkg -i ${CUDA\_REPO\_PKG} |
|  |

|  |
| --- |
| sudo apt-get update |
|  |

|  |
| --- |
| sudo apt-get -y install cuda |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # install cuDNN v6.0 |
|  |

|  |
| --- |
| CUDNN\_TAR\_FILE="cudnn-8.0-linux-x64-v6.0.tgz" |
|  |

|  |
| --- |
| wget http://developer.download.nvidia.com/compute/redist/cudnn/v6.0/${CUDNN\_TAR\_FILE} |
|  |

|  |
| --- |
| tar -xzvf ${CUDNN\_TAR\_FILE} |
|  |

|  |
| --- |
| sudo cp -P cuda/include/cudnn.h /usr/local/cuda-8.0/include |
|  |

|  |
| --- |
| sudo cp -P cuda/lib64/libcudnn\* /usr/local/cuda-8.0/lib64/ |
|  |

|  |
| --- |
| sudo chmod a+r /usr/local/cuda-8.0/lib64/libcudnn\* |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # set environment variables |
|  |

|  |
| --- |
| export PATH=/usr/local/cuda-8.0/bin${PATH:+:${PATH}} |
|  |

export LD\_LIBRARY\_PATH=/usr/local/cuda-8.0/lib64\${LD\_LIBRARY\_PATH:+:${LD\_LIBRARY\_PATH}}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Download the latest source archive for OpenCV 3.2 from <https://github.com/opencv/opencv/archive/3.2.0.zip>

Enter the unpacked directory. Execute:

mkdir build

cd build/

cmake -D CMAKE\_BUILD\_TYPE=RELEASE -D CMAKE\_INSTALL\_PREFIX=/usr/local -D FORCE\_VTK=ON -D WITH\_TBB=ON -D WITH\_V4L=ON -D WITH\_QT=ON -D WITH\_OPENGL=ON -D WITH\_CUBLAS=ON -D CUDA\_NVCC\_FLAGS="-D\_FORCE\_INLINES" -D WITH\_GDAL=ON -D WITH\_XINE=ON -D BUILD\_EXAMPLES=ON ..

make -j $(($(nproc) + 1))

sudo make install

sudo /bin/bash -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'

sudo ldconfig

sudo apt-get update

## 5. Installation Caffe

<https://github.com/BVLC/caffe/wiki/Install-Caffe-on-EC2-from-scratch-(Ubuntu,-CUDA-7,-cuDNN-3)>

export LC\_ALL="en\_US.UTF-8"

export LC\_CTYPE="en\_US.UTF-8"

sudo dpkg-reconfigure locales

sudo pip install opencv-python

*sudo apt-get install python-skimage*

then pip install requirements

warning no more the 4 threads

add to Makefile:

INCLUDE\_DIRS += $(PYTHON\_INCLUDE) /usr/local/include /usr/include/hdf5/serial/

Delete "-gencode arch=compute\_20,code=sm\_20 " in Makefile

# USE\_CUDNN := 1

#OPENCV\_VERSION := 3

Add two lines in the end of Makefile.config.example

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/

> Installing Caffe

Go to the <https://github.com/BVLC/caffe> and download the zip archive. Unpack it to ~/bin/ or any other location. Enter the caffe-master directory in the terminal window. Note that only the version 1.0RC5 compiles well at this moment, so download the 1.0RC5 version from <https://github.com/BVLC/caffe/archive/rc5.zip>. If you download the 1.0 version from <https://github.com/BVLC/caffe/archive/1.0.zip>, you need to edit the file /src/caffe/util/blocking\_queue.cpp. After the line 89, add the new line that contains the following:

template class BlockingQueue<Datum\*>;

Copy the Makefile.config.example to Makefile.config like this:

cp Makefile.config.example Makefile.config

and open it for editing (with a text editor). I use the kate editor for this purpose, so the command that I execute goes as follows. You first need to install the kate editor with:

sudo apt-get install kate

and then you can edit the configuration file with:

kate ./Makefile.config &

The following line in the configuration file tells the program to use CPU only for the computations.

CPU\_ONLY := 1

WITH\_PYTHON\_LAYER := 1

CPU\_ONLY option is enabled for a computer without any NVIDIA graphics card and it is typical for the installation of Caffe inside the typical virtual machine. (Notice that there is a special type of virtual machine inside the Ubuntu host machine that can access the physical NVIDIA graphics card directly. See <https://github.com/NVIDIA/nvidia-docker>)

The default option value is to use GPU and CPU computation. Change the line if needed, by commenting it out (# CPU\_ONLY := 1) if you have an NVIDIA graphics card with the proprietary driver, CUDA toolkit and CUDNN installed. Jump to the end of this guide to read about how to install the GPU support prerequisites.

The Makefile.config should contain the following lines, so find them and fill them in.

PYTHON\_INCLUDE := /usr/include/python2.7 /usr/lib/python2.7/dist-packages/numpy/core/include

(for some Ubuntu 16.04 users, the path may be different)

PYTHON\_INCLUDE := /usr/include/python2.7 /usr/local/lib/python2.7/dist-packages/numpy/core/include

WITH\_PYTHON\_LAYER := 1

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 /usr/lib/x86\_64-linux-gnu/hdf5/serial

(For ways to create an isolated Python environment, explore the topic of virtual environments here: <http://docs.python-guide.org/en/latest/dev/virtualenvs/>)

If your CUDA is 8.0, find this in Makefile.config..

If your CUDA is 8.0, find this in Makefile.config..

CUDA\_DIR := /usr/local/cuda

And replace it with this..

CUDA\_DIR := /usr/local/cuda-8.0

Now lets continue with the instructions for the Ubuntu 15.10 first, followed by the instructions for Ubuntu 16.04 users.

Execute the additional commands:

find . -type f -exec sed -i -e 's^"hdf5.h"^"hdf5/serial/hdf5.h"^g' -e 's^"hdf5\_hl.h"^"hdf5/serial/hdf5\_hl.h"^g' '{}' \;

cd /usr/lib/x86\_64-linux-gnu

sudo ln -s libhdf5\_serial.so.8.0.2 libhdf5.so

sudo ln -s libhdf5\_serial\_hl.so.8.0.2 libhdf5\_hl.so

The above commands are no longer needed on Ubuntu 16.04 after a certain system update. If your Ubuntu 16.04 still needs this step to be performed, take the following considerations.

The file versions for libhdf5\_serial.so and libhdf5\_serial\_hl.so are different and so the last two lines will need to be altered. Visit /usr/lib/x86\_64-linux-gnu/ and list the relevant contents of that directory by using the command such as ls -l | grep hdf5. The versions of libhdf5 that need to be linked to are 10.1.0 and 10.0.2 respectively:

find . -type f -exec sed -i -e 's^"hdf5.h"^"hdf5/serial/hdf5.h"^g' -e 's^"hdf5\_hl.h"^"hdf5/serial/hdf5\_hl.h"^g' '{}' \;

cd /usr/lib/x86\_64-linux-gnu

sudo ln -s libhdf5\_serial.so.10.1.0 libhdf5.so

sudo ln -s libhdf5\_serial\_hl.so.10.0.2 libhdf5\_hl.so

Now for both platforms lets return to the unpacked Caffe directory caffe-master and enter these commands:

cd python

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

NOTE: If the Ubuntu operating system was updated, perhaps the Python layer needs to be updated and recompiled, because the Python module no longer works. Perform this step again in that case.

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

In case of any problems, try:

for req in $(cat requirements.txt); do sudo -H pip install $req --upgrade; done

The default Python version is 2. You can edit the Makefile.conf to enable the Python 3, but this will fail during the linking phase: boost\_python3 cannot be found on Ubuntu 16.04. Instead, this file should be /usr/lib/x86\_64-linux-gnu/libboost\_python-py35.so.1.58.0. This requires further testing.

The next step is to build Caffe:

cd ..

(now you are in caffe-master directory)

The build process will fail in Ubuntu 16.04. Edit the Makefile with an editor such as

kate ./Makefile

and replace this line:

NVCCFLAGS += -ccbin=$(CXX) -Xcompiler -fPIC $(COMMON\_FLAGS)

with the following line

NVCCFLAGS += -D\_FORCE\_INLINES -ccbin=$(CXX) -Xcompiler -fPIC $(COMMON\_FLAGS)

Also, open the file CMakeLists.txt and add the following line:

# ---[ Includes

set(${CMAKE\_CXX\_FLAGS} "-D\_FORCE\_INLINES ${CMAKE\_CXX\_FLAGS}")

make all

make test

make runtest

make pycaffe -should be finished already, so you can omit this one

make distribute

Note that the build process can be sped up by appending -j $(($(nproc) + 1)) to the above commands, which distributes the build across the available processors on your system. For example:

make all

can become

make all -j $(($(nproc) + 1))

In order to make the Python work with Caffe, open the file ~/.bashrc for editing in your favorite text editor. There, add the following line at the end of file:

export PYTHONPATH=/path/to/caffe-master/python:$PYTHONPATH

You can also execute that same line immediately in the terminal as a command for immediate effects, or in general execute:

source ~/.bashrc

The binary models can be download with the following script. In caffe-master directory,

cd scripts

./download\_model\_binary.py ../models/bvlc\_alexnet/

./download\_model\_binary.py ../models/bvlc\_googlenet/

./download\_model\_binary.py ../models/bvlc\_reference\_caffenet/

./download\_model\_binary.py ../models/bvlc\_reference\_rcnn\_ilsvrc13/

./download\_model\_binary.py ../models/finetune\_flickr\_style/

For more models, see <https://github.com/BVLC/caffe/wiki/Model-Zoo> \* For most Linux programs compiled from source, you can attempt to build a package that can be installed and uninstalled with a single click.

sudo apt-get install checkinstall

Now, when you execute the:

sudo checkinstall

## 6. Installation Faster RCNN

Clone the Faster R-CNN repository

# Make sure to clone with --recursive

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

We'll call the directory that you cloned Faster R-CNN into FRCN\_ROOT

*Ignore notes 1 and 2 if you followed step 1 above.*

**Note 1:** If you didn't clone Faster R-CNN with the --recursive flag, then you'll need to manually clone the caffe-fast-rcnn submodule:

git submodule update --init --recursive

**Note 2:** The caffe-fast-rcnn submodule needs to be on the faster-rcnn branch (or equivalent detached state). This will happen automatically *if you followed step 1 instructions*.

Build the Cython modules

cd $FRCN\_ROOT/lib

make

Build Caffe and pycaffe

cd $FRCN\_ROOT/caffe-fast-rcnn

cp Makefile.config.example Makefile.config

insert two lines in the end of Makefile.config.example

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/

remove two lines:

-gencode arch=compute\_20,code=sm\_20 \

-gencode arch=compute\_20,code=sm\_21 \

#comment line USE\_CUDNN

#comment line CPU\_ONLY=1

#comment line OPENCV\_VERSION := 3

WITH\_PYTHON\_LAYER := 1

# Now follow the Caffe installation instructions here:

# http://caffe.berkeleyvision.org/installation.html

# If you're experienced with Caffe and have all of the requirements installed

# and your Makefile.config in place, then simply do:

make -j8 && make pycaffe

Download pre-computed Faster R-CNN detector

cd $FRCN\_ROOT

./data/scripts/fetch\_faster\_rcnn\_models.sh

./data/scripts/ilsvrc12/get\_ilsvrc\_aux.sh

This will populate the $FRCN\_ROOT/data folder with faster\_rcnn\_models. See data/README.md for details. These models were trained on VOC 2007 trainval.

## 7. Installation Tesseract

sudo apt-get install tesseract-ocr

pip install pytesseract

## 8. Installation clstm

cd ~

# Ubuntu 15.04, 16.04 / Debian 8, 9

sudo apt-get install scons libprotobuf-dev protobuf-compiler libpng-dev libeigen3-dev swig

git clone https://github.com/tmbdev/clstm.git

cd clstm

scons

sudo scons install

pip install .

$ pip install git+https://github.com/jbaiter/clstm.git@cythons

## 9. Install program

pip install flask

sudo pip install tornado

sudo pip install easydict

sudo pip install FuzzySet

sudo pip install nltk

#sudo pip install sklearn

sudo pip install scikit-learn==0.19.1

sudo pip install imutils

sudo pip install openpyxl

sudo pip install SOAPpy

sudo pip install requests

pip install unidecode

sudo pip install pdf2image

sudo apt install poppler-utils

sudo apt-get install imagemagick

python

import nltk

nltk.download(‘punkt’)

add the following lines to ~./bashrc

export CUDA\_HOME=/usr/local/cuda

export CUDA\_ROOT=/usr/local/cuda

export PATH=$PATH:$CUDA\_ROOT/bin

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:$CUDA\_ROOT/lib64:$CUDA\_ROOT/extras/CUPTI/lib64

alias python=python2