

SOP INSTALASI OPENCV-DNN 4.8.0 MENGGUNAKAN CUDA BACKEND PADA LINUX UBUNTU 18.04

Pemagangan Mahasiswa Pusat Riset Telekomunikasi BRIN 2023

Pembimbing:

Arief Suryadi Satyawan M.T., D.Eng. (BRIN)

Dr. Eng. Munawir, S.Kom., M.T. (UPI)



Oleh:

Rahmawati (2008187)

UNIVERSITAS PENDIDIKAN INDONESIA

KAMPUS UPI DI CIBIRU

PROGRAM STUDI TEKNIK KOMPUTER

2023

SOP INSTALASI OPENCV-DNN 4.8.0 MENGGUNAKAN CUDA BACKEND PADA LINUX UBUNTU 18.04

Pada SOP ini akan menjelaskan 2 bagian yang berbeda. Bagian pertama akan membahas mengenai instalasi CUDA dan cuDNN karena untuk menginstal OpenCV CUDA membutuhkan CUDA dan cuDNN. Setelah itu, pada bagian kedua akan membahas mengenai langkah-langkah instalasi OpenCV menggunakan CUDA backend. SOP ini mengacu pada video penjelasan berikut:

<https://youtu.be/MpBweXzQXg0?si=GxD3CjtxWoFZrIWM>

Langkah-Langkah Instalasi CUDA dan cuDNN pada LINUX Ubuntu 18.04

Pada percobaan kali ini kita akan menginstal CUDA dan cuDNN di dalam conda environment agar tidak terpengaruh dengan project lain.

1. Langkah pertama ialah mengunduh Anaconda pada tautan berikut, <https://www.anaconda.com/download#downloads>. Pilih 64-Bit (x86) Installer.
2. Install Anaconda yang telah diunduh kemudian buka terminal dan buat Conda environment baru dengan mengikuti command berikut.
`conda create --name nama_env python=3.9`
Disini saya menggunakan python versi 3.9
3. Setelah environment terbuat maka aktifkan environment dengan command
`conda activate nama_env`
4. Setelah environment aktif langkah selanjutnya ialah menginstal CUDA dan cuDNN.
5. Sebelum menginstall CUDA dan cuDNN update dan upgrade semua packages yang ada dengan mengikuti command berikut.
`sudo apt-get update`
`sudo apt-get upgrade`
6. Kemudian install beberapa hal berikut
`sudo apt-get install build-essential cmake unzip pkg-config`
`sudo apt-get install gcc-6 g++-6`
`sudo apt-get install libxmu-dev libxi-dev libglu1-mesa libglu1-mesa-dev`
`sudo apt-get install libjpeg-dev libpng-dev libtiff-dev`
`sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev`
`sudo apt-get install libxvidcore-dev libx264-dev`
`sudo apt-get install libopenblas-dev libatlas-base-dev liblapack-dev gfortran`

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

```
sudo apt-get install python3-dev python3-tk python-imaging-tk
```

```
sudo apt-get install libgtk-3-dev
```

7. Tambahkan grafik driver.

```
sudo add-apt-repository ppa:graphics-drivers/ppa
```

```
sudo apt update
```

8. Pertama install NVIDIA Driver minimal versi 450 karena untuk CUDA versi 11.x minimum versi drivernya ialah 450. Informasi tersebut dapat dicek pada tautan berikut.

<https://docs.nvidia.com/deploy/cuda-compatibility/index.html>

Table 1. Example CUDA Toolkit 11.x Minimum Required Driver Versions (Refer to CUDA Release Notes)

CUDA Toolkit	Linux x86_64 Minimum Required Driver Version	Windows Minimum Required Driver Version
CUDA 12.x	>=525.60.13	>=527.41
CUDA 11.x	>= 450.80.02*	>=452.39*

Adapun command untuk instalasi CUDA Driver sebagai berikut.

```
sudo apt-get install nvidia-driver-450
```

9. Buat folder installers agar hal-hal yang perlu diinstall tidak berceceran.

```
cd ~
```

```
mkdir installers
```

```
cd installers/
```

10. Kemudian install CUDA Toolkit pada tautan berikut, <https://developer.nvidia.com/cuda-toolkit-archive>. Pada percobaan kali ini penulis menggunakan CUDA Toolkit 11.8.0.

CUDA Toolkit 11.8 Downloads

Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown. By downloading and using the software, you agree to fully comply with the terms and conditions of the [CUDA EULA](#).

Operating System	Linux	Windows							
Architecture	x86_64	ppc64le	arm64-sbsa	aarch64-jetson					
Distribution	CentOS	Debian	Fedora	KylinOS	OpenSUSE	RHEL	Rocky	SLES	Ubuntu
Version	WSL-Ubuntu								
Installer Type	deb (local)	deb (network)	runfile (local)						

Atau bisa saja dengan menyalin command di bawah

wget

```
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/cuda-ubuntu1804.pin
```

```
sudo mv cuda-ubuntu1804.pin /etc/apt/preferences.d/cuda-repository-pin-600
```

wget

```
https://developer.download.nvidia.com/compute/cuda/11.8.0/local_installers/cuda-repo-ubuntu1804-11-8-local_11.8.0-520.61.05-1_amd64.deb
```

```
sudo dpkg -i cuda-repo-ubuntu1804-11-8-local_11.8.0-520.61.05-1_amd64.deb
```

```
sudo cp /var/cuda-repo-ubuntu1804-11-8-local/cuda-*-keyring.gpg /usr/share/keyrings/
```

```
sudo apt-get update
```

```
sudo apt-get -y install cuda
```

11. Buka bacshrc script

```
nano ~/.bashrc
```

12. Tambahkan path berikut dibagian akhir bashrc file

```
# NVIDIA CUDA TOOLKIT
```

```
export PATH=/usr/local/cuda-11.8/bin${PATH:+:${PATH}}
```

```
export LD_LIBRARY_PATH=/usr/local/cuda-11.8/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

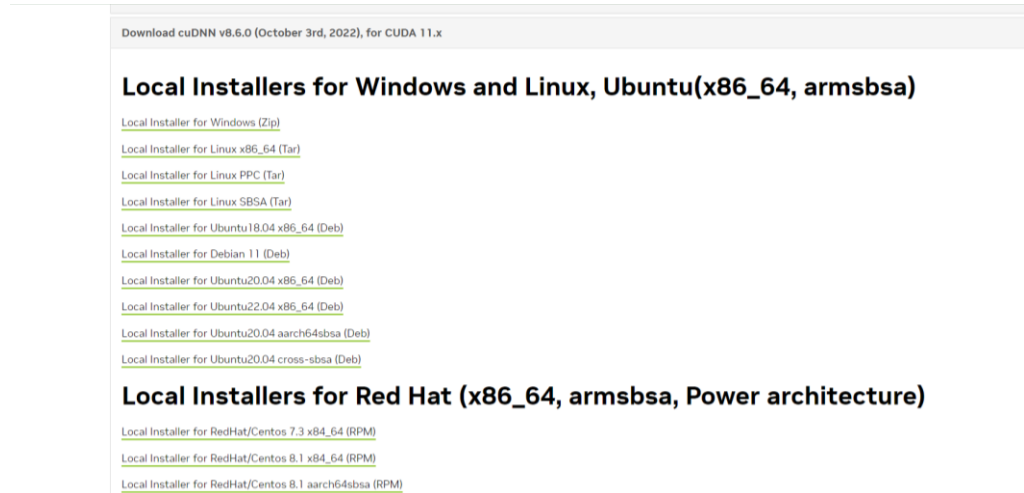
Tekan Ctrl+x, y, dan Enter untuk menyimpan perubahan

13. Kemudian jalankan command berikut untuk meng-update perubahan

```
source ~/.bashrc
```

14. Setelah CUDA berhasil terinstall langkah selanjutnya ialah mengunduh cuDNN. Pada percobaan kali ini penulis menggunakan cuDNN 8.6.0. Untuk mengunduh cuDNN dapat mengakses tautan berikut.

<https://developer.nvidia.com/rdp/cudnn-archive>



Pilih Local Installer for Linux x86_64 (Tar)

15. Copy folder zip cuDNN yang telah terunduh ke folder installers yang telah dibuat sebelumnya.

16. Setelah itu unzip folder cuDNN menggunakan command berikut

```
tar -xvf cudnn-linux-x86_64-8.6.0.163_cuda11-archive.tar.xz
```

17. Copy masing-masing folder include dan lib64 pada cuDNN ke dalam folder include dan lib64 pada direktori CUDA.

```
cd cuda
```

```
sudo cp -P lib64/* /usr/local/cuda/lib64/
```

```
sudo cp -P include/* /usr/local/cuda/include/
```

18. Setelah itu cek menggunakan 'conda list' apakah CUDA dan cuDNN telah terinstall. Jika telah terinstall maka kedua hal tersebut akan tertampilkan pada list. Selain itu, untuk mengetahui CUDA version bisa menggunakan command 'nvidia-smi' atau 'nvcc -version'.

Langkah-Langkah Instalasi OpenCV CUDA pada LINUX Ubuntu 18.04

Setelah CUDA dan cuDNN telah terinstall langkah selanjutnya ialah melakukan instalasi OpenCV CUDA. Pada percobaan kali ini penulis menggunakan OpenCV versi 4.8.0.

1. Langkah pertama, install packages yang akan diperlukan.

```
sudo apt-get update
```

```

sudo apt-get upgrade
sudo apt-get install build-essential cmake unzip pkg-config
sudo apt-get install libjpeg-dev libpng-dev libtiff-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
sudo apt-get install libv4l-dev libxvidcore-dev libx264-dev
sudo apt-get install libgtk-3-dev
sudo apt-get install libblas-dev liblapack-dev gfortran
sudo apt-get install python3-dev

```

2. Unduh dan unzip opencv 4.8.0

```

cd installers
wget -O opencv-4.8.0.zip https://github.com/opencv/opencv/archive/4.8.0.zip
unzip -q opencv-4.8.0.zip
mv opencv-4.8.0 opencv
rm -f opencv-4.8.0.zip

```

3. Unduh dan unzip opencv_contrib 4.8.0

```

wget -O opencv_contrib-4.8.0.zip https://github.com/opencv/opencv_contrib/archive/4.8.0.zip
unzip -q opencv_contrib-4.8.0.zip
mv opencv_contrib-4.8.0 opencv_contrib
rm -f opencv_contrib-4.8.0.zip

```

4. Setelah itu, install numpy

```

pip install numpy

```

5. Buat folder baru yang bernama build untuk menampung file-file yang akan digunakan ketika mem-build OpenCV package.

```

cd installers/opencv
mkdir build
cd build

```

6. Setelah itu, jalankan command berikut untuk mengkonfigurasi OpenCV dan men-set up CUDA pada OpenCV.

```

cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D WITH_TBB=ON \
-D ENABLE_FAST_MATH=1 \
-D CUDA_FAST_MATH=1 \

```

```

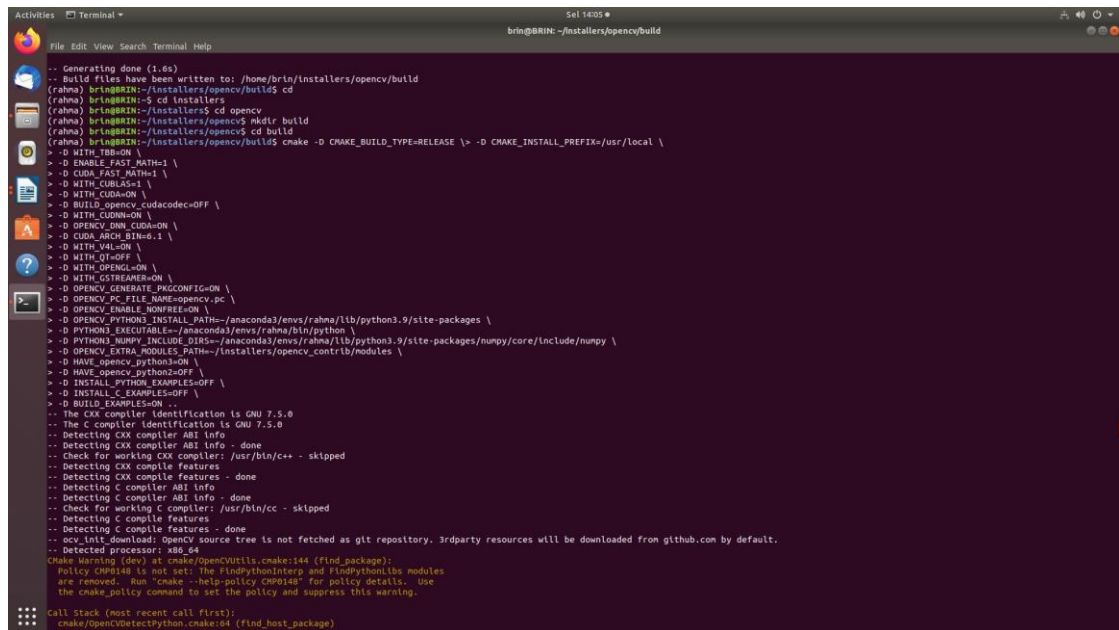
-D WITH_CUBLAS=1 \
-D WITH_CUDA=ON \
-D BUILD_opencv_cudacodec=OFF \
-D WITH_CUDNN=ON \
-D OPENCV_DNN_CUDA=ON \
-D CUDA_ARCH_BIN=6.1 \
-D WITH_V4L=ON \
-D WITH_QT=OFF \
-D WITH_OPENGL=ON \
-D WITH_GSTREAMER=ON \
-D OPENCV_GENERATE_PKGCONFIG=ON \
-D OPENCV_PC_FILE_NAME=opencv.pc \
-D OPENCV_ENABLE_NONFREE=ON \
-D
OPENCV_PYTHON3_INSTALL_PATH=~/.anaconda3/envs/rahma/lib/python3.9/site-
packages \
-D PYTHON3_EXECUTABLE=~/.anaconda3/envs/rahma/bin/python \
-D
PYTHON3_NUMPY_INCLUDE_DIRS=~/.anaconda3/envs/rahma/lib/python3.9/site-
packages/numpy/core/include/numpy \
-D OPENCV_EXTRA_MODULES_PATH=~/.installers/opencv_contrib/modules \
-D HAVE_opencv_python3=ON \
-D HAVE_opencv_python2=OFF \
-D INSTALL_PYTHON_EXAMPLES=OFF \
-D INSTALL_C_EXAMPLES=OFF \
-D BUILD_EXAMPLES=ON ..

```

Command WITH_CUDA=ON dan WITH_CUDNN=ON berfungsi untuk mensetting agar OpenCV dapat menggunakan CUDA pada backend-nya. Jika tidak ingin menggunakan CUDA, dua command di atas bisa di set OFF saja.

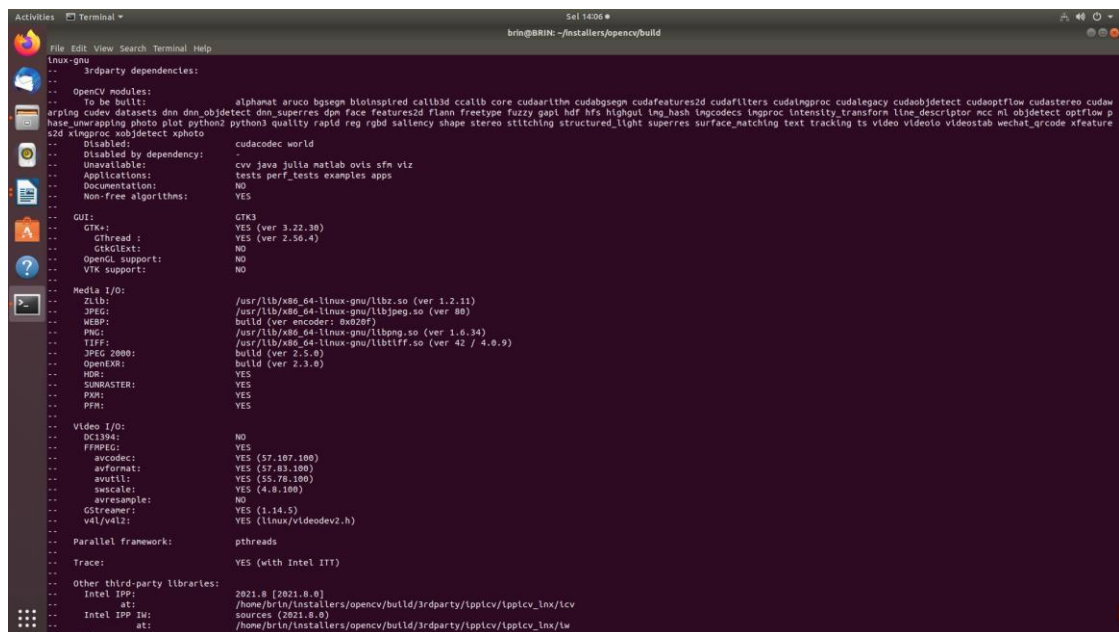
Sesuaikan CUDA_ARCH_BIN dengan jenis arsitektur GPU yang digunakan. Pada percobaan kali ini penulis menggunakan GPU NVIDIA GeForce GTX 106 sehingga

jenis arsitekturnya merupakan 6.1. Informasi mengenai hal ini dapat diakses pada tautan berikut <https://developer.nvidia.com/cuda-gpus>.



```
-- Generating done (1.6s)
-- Build files have been written to: /home/brin/installers/opencv/build
(rahma) brin@BRIN:~/installers/opencv/build$ cd
(rahma) brin@BRIN:~/installers$ cd installers
(rahma) brin@BRIN:~/installers$ cd opencv
(rahma) brin@BRIN:~/installers/opencv$ mkdir build
(rahma) brin@BRIN:~/installers/opencv$ cd build
(rahma) brin@BRIN:~/installers/opencv/build$ cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local \
> -D WITH_FFMPEG=ON \
> -D ENABLE_FAST_MATH=1 \
> -D CUDA_FAST_MATH=1 \
> -D WITH_CUBLAS=1 \
> -D WITH_CUDA=ON \
> -D BUILD_opencv_cudacodec=OFF \
> -D WITH_CUDNN=ON \
> -D OPENCV_DNN_CUDA=ON \
> -D CUDA_ARCH_BIN=6.1 \
> -D WITH_V4L=ON \
> -D WITH_QT=OFF \
> -D WITH_OPENGL=ON \
> -D WITH_GSTREAMER=ON \
> -D OPENCV_GENERATE_PKGCONFIG=ON \
> -D OPENCV_PC_FILE_NAME=opencv.pc \
> -D OPENCV_ENABLE_NONFREE=ON \
> -D OPENCV_PYTHON3_INSTALL_PATH=/anaconda3/envs/rahma/lib/python3.9/site-packages \
> -D PYTHON3_EXECUTABLE=/anaconda3/envs/rahma/bin/python \
> -D PYTHON3_NUMPY_INCLUDE_DIRS=/anaconda3/envs/rahma/lib/python3.9/site-packages/numpy/core/include/numpy \
> -D OPENCV_EXTRA_MODULES_PATH=/installers/opencv_contrib/modules \
> -D HAVE_opencv_python3=ON \
> -D HAVE_opencv_python2=OFF \
> -D INSTALL_PYTHON_EXAMPLES=OFF \
> -D INSTALL_C_EXAMPLES=OFF \
> -D BUILD_EXAMPLES=ON ..
-- The CXX compiler identification is GNU 7.5.0
-- The C compiler identification is GNU 7.5.0
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- dev link download: OpenCV source tree is not fetched as git repository. 3rdparty resources will be downloaded from github.com by default.
CMake Warning (dev) at cmake/OpenCVUtils.cmake:144 (find_package):
Policy CMP0148 is not set: The FindPythonInterp and FindPythonLibs modules
are removed. Run "cmake --help-policy CMP0148" for policy details. Use
the cmake_policy command to set the policy and suppress this warning.
Call Stack (most recent call first):
cmake/OpenCVDetectPython.cmake:64 (find_host_package)
cmake/OpenCVDetectPython.cmake:144 (find_host_package)
```

7. Pastikan pada hasil konfigurasi CUDA dan cuDNN keterangannya YES.



```
Linux-gnu
--
3rdparty dependencies:
--
OpenCV modules:
--
To be built:
  alphasat aruco bgsegm bioinspired call3d ccallb core cudaarithm cudabgsegm cudafeatures2d cudafilters cudalgpproc cudalegacy cudaobjdetect cudaoptflow cudastereo cudaw
arpring cudaw_daxtests dnn_dnn_objdetect dnn_superres dnn_face_features2d flann freetype fuzzy_gapi hdf hfs highgui img_hash imgcodecs imgproc intensity_transform line_descriptor mcl
objdetect optflow p
hase_unwrapping photo_plot python2 python3 quality rapid reg rgbd saliency shape stereo stitching structured_light superres surface_matching text tracking ts video videoleo videostab
wechat_gcode xfeature
s2d ximgproc xobjdetect xphoto
--
Disabled by dependency:
  cudacodec world
--
Unavailable:
  cvv java julia matlab ovis sfm viz
Applications:
  tests perf_tests examples apps
Documentation:
  NO
Non-free algorithms:
  YES
--
GUI:
  GTK3:
  YES (ver 3.22.30)
  GThread :
  YES (ver 2.56.4)
  GtGLExt:
  NO
  OpenGL support:
  NO
  VTK support:
  NO
--
Media I/O:
  ZLIB:
  /usr/lib/x86_64-linux-gnu/libz.so (ver 1.2.11)
  JPEG:
  /usr/lib/x86_64-linux-gnu/libjpeg.so (ver 80)
  HEIF:
  build (ver encoder: 0x200)
  PNG:
  /usr/lib/x86_64-linux-gnu/libpng.so (ver 1.6.34)
  TIFF:
  /usr/lib/x86_64-linux-gnu/libtiff.so (ver 42 / 4.0.9)
  JPEG 2000:
  build (ver 2.5.0)
  OpenEXR:
  build (ver 2.3.0)
  HDR:
  YES
  SUNRASTER:
  YES
  PXM:
  YES
  PFM:
  YES
--
Video I/O:
  DC1394:
  NO
  FFMpeg:
  YES
  avcodec:
  YES (57.107.100)
  avformat:
  YES (57.83.100)
  avutil:
  YES (55.78.100)
  swscale:
  YES (4.8.100)
  avresample:
  NO
  GStreamer:
  YES (1.14.5)
  v4l/v4l2:
  YES (linux/videodev2.h)
--
Parallel framework:
  pthreads
--
Trace:
  YES (with Intel ITT)
--
Other third-party libraries:
  Intel IPP:
    at:
    /home/brin/installers/opencv/build/3rdparty/ippicv/ippicv_linux/icv
sources (2021.8.0)
  Intel IPP IW:
    at:
    /home/brin/installers/opencv/build/3rdparty/ippicv/ippicv_linux/iw
sources (2021.8.0)
```



```
Activities Terminal
brin@BRIN: ~/installers/opencv/build

[100k] Linking CXX executable ../../bin/example_tutorial_gmp_detection
[100k] Building CXX object modules/python2/cvkernel/opencv_python2_d1/./src2/cv2_numpy.cpp.o
[100k] Built target example_tutorial_gmp_detection
[100k] Building CXX object modules/python2/cvkernel/opencv_python2_d1/./src2/cv2_convert.cpp.o
[100k] Building CXX object modules/python2/cvkernel/opencv_python2_d1/./src2/cv2_highgui.cpp.o
[100k] Linking CXX executable ../../bin/example_tutorial_gmp_registration
[100k] Built target example_tutorial_gmp_registration
[100k] Linking CXX shared module ../../lib/cv2.so
[100k] Linking CXX shared module ../../lib/python3/cv2.cpython-39-x86_64-linux-gnu.so
[100k] Built target opencv_python3
[100k] Built target opencv_python3
(rhna) brin@BRIN:~/installers/opencv/build$ sudo make install
[sudo] password for brin:
-- ocv_init_download: OpenCV source tree is not fetched as git repository. 3rdparty resources will be downloaded from github.com by default.
-- Detected processor: x86_64
-- Looking for cache: not found
-- Found ZLIB: /usr/lib/x86_64-linux-gnu/libz.so (found suitable version "1.2.11", minimum required is "1.2.3")
Cleaning INTERNAL cached variable: WEBP_INCLUDE_DIR
-- Could not find OpenJPEG (minimal suitable version: 2.0, recommended version >= 2.3.1). OpenJPEG will be built from sources
-- OpenJPEG: VERSION = 2.3.0, BUILD = opencv_4.8.0-openjp2-2.5.0
-- OpenJPEG libraries will be built from sources: libopenjp2 (version "2.5.0")
-- Found ZLIB: /usr/lib/x86_64-linux-gnu/libz.so (found version "1.2.11")
-- libvst: missing vst.h header (via INCLUDE_DIRS)
-- Found Intel IPP (ICV version): 2021.8.0 [2021.8]
-- at: /home/brin/installers/opencv/build/3rdparty/ippicv/ippicv_linux/icv
-- at: /home/brin/installers/opencv/build/3rdparty/ippicv/ippicv_linux/lw
CMake Warning (dev) at cmake/OpenCvUtils.cmake:144 (find_package):
Policy CMP0140 is not set: The find_cuda module is removed. Run cmake
-help-policy CMP0140 for policy details. Use the cmake_policy command to
set the policy and suppress this warning.

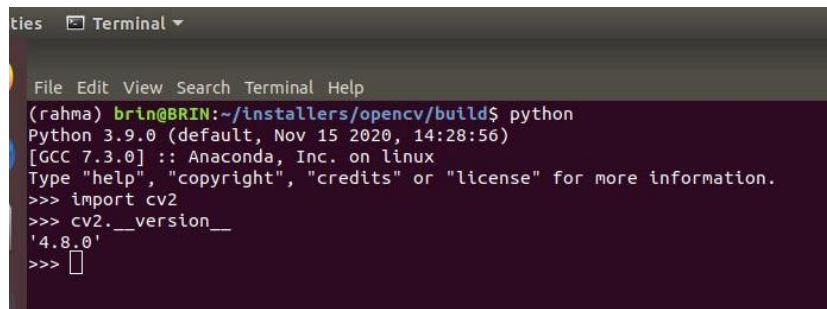
Call Stack (most recent call first):
cmake/OpenCvDetectCUDA.cmake:21 (find_host_package)
cmake/OpenCvFindLibraries.cmake:43 (include)
CMakeLists.txt:750 (include)

This warning is for project developers. Use -Wno-dev to suppress it.

-- CUDA detected: 11.8
-- CUDA_11XX target flags: -xcode;arch=compute_61;code=sm_61;-D_FORCE_INLINES
-- Could not find OpenBLAS include. Turning OpenBLAS_FOUND off
-- Could not find OpenBLAS lib. Turning OpenBLAS_FOUND off
-- Could not find Atlas (missing: Atlas_CPACK_INCLUDE_DIRS)
-- Could not find Java (missing: Java_JAR_EXECUTABLE Java_JAVAC_EXECUTABLE Java_JAVADOC_EXECUTABLE) (found version "11.0.19")
-- Could not find MATLAB (missing: MATLAB_INCLUDE_PATH MATLAB_INCLUDE_PATH2 MATLAB)
-- Could not find PyLint (missing: PYLINT_EXECUTABLE)
-- Could not find Flake8 (missing: FLAKE8_EXECUTABLE)
-- VTK is not found, please set -DVTK_DIR in cmake to VTK build directory, or to VTK install subdirectory with VTKConfig.cmake file
-- Checking for module 'gtk+-2.0'
-- No package 'gtk+-2.0' found
-- Checking for module 'libaresample'
-- No package 'libaresample' found
-- Checking for module 'libdc1394-2'

10. Untuk memastikan apakah library OpenCV telah terinstall atau belum bisa mengecek
di direktori anaconda3/envs/nama_env/lib/python3.9/site-packages. Jika terdapat
library CV2 maka OpenCV dengan CUDA telah terinstall. Untuk memastikan lebih
jelas dapat menjalankan comman berikut.

python
import cv2
cv2. version
```

A terminal window titled 'Terminal' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is '(rahma) brin@BRIN:~/installers/opencv/build\$'. The user has run 'python', showing 'Python 3.9.0 (default, Nov 15 2020, 14:28:56) [GCC 7.3.0] :: Anaconda, Inc. on linux'. Then they ran 'import cv2' and 'cv2.__version__', which outputs '4.8.0'.

```
(rahma) brin@BRIN:~/installers/opencv/build$ python
Python 3.9.0 (default, Nov 15 2020, 14:28:56)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'4.8.0'
>>> 
```

Jika terminal menampilkan versi OpenCV sesuai dengan yang telah di-build sebelumnya, yaitu versi 4.8.0, maka OpenCV dengan CUDA backend telah berhasil terinstall.

11. Untuk melihat lebih jelas, bisa menjalankan program test OpenCV DNN berikut untuk mengetahui GPU dan CPU yang digunakan. Jika menampilkan waktu GPU dan CPU maka OpenCV CUDA telah berhasil diinstall dan siap digunakan.

Source code test_DNN_CV.py

```
import numpy as np
```

```
import cv2 as cv
```

```
import time
```

```
npTmp = np.random.random((1024, 1024)).astype(np.float32)
```

```
npMat1 = np.stack([npTmp,npTmp],axis=2)
```

```
npMat2 = npMat1
```

```
cuMat1 = cv.cuda_GpuMat()
```

```
cuMat2 = cv.cuda_GpuMat()
```

```
cuMat1.upload(npMat1)
```

```
cuMat2.upload(npMat2)
```

```
start_time = time.time()
```

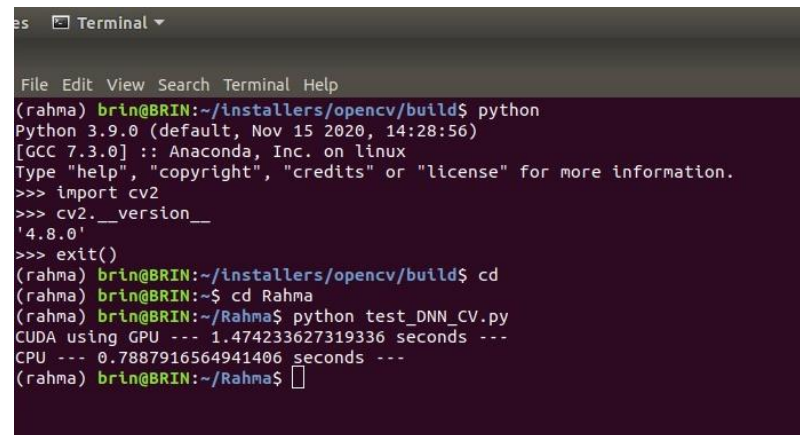
```
cv.cuda.gemm(cuMat1, cuMat2,1,None,0,None,1)
```

```
print("CUDA using GPU --- %s seconds ---" % (time.time() - start_time))
```

```
start_time = time.time()
```

```
cv.gemm(npMat1,npMat2,1,None,0,None,1)
```

```
print("CPU --- %s seconds ---" % (time.time() - start_time))
```



```
es Terminal ▾  
File Edit View Search Terminal Help  
(rahma) brin@BRIN:~/installers/opencv/build$ python  
Python 3.9.0 (default, Nov 15 2020, 14:28:56)  
[GCC 7.3.0] :: Anaconda, Inc. on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import cv2  
>>> cv2.__version__  
'4.8.0'  
>>> exit()  
(rahma) brin@BRIN:~/installers/opencv/build$ cd  
(rahma) brin@BRIN:~$ cd Rahma  
(rahma) brin@BRIN:~/Rahma$ python test_DNN_CV.py  
CUDA using GPU --- 1.474233627319336 seconds ---  
CPU --- 0.7887916564941406 seconds ---  
(rahma) brin@BRIN:~/Rahma$
```