

**1. Before install opencv you need :** (This step is *only for GPU users.*)

- An NVIDIA GPU
- The CUDA drivers for that particular GPU installed
- CUDA Toolkit and cuDNN configured and installed

1.1 We need to add an apt-get repository so that we can install NVIDIA GPU drivers.

This can be accomplished in your terminal:

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

```
sudo apt-get update
```

1.2 choose one drive version :

```
ubuntu-drivers devices
```

```
wang@danyi-System-Product-Name:~/yolo_test/build$ ubuntu-drivers devices
== /sys/devices/pci0000:00/0000:00:01.0/0000:01:00.0 ==
modalias : pci:v000010DEd00001C03sv00001458sd0000371Abc03sc00i00
vendor    : NVIDIA Corporation
model     : GP106 [GeForce GTX 1060 6GB]
driver    : nvidia-driver-390 - distro non-free
driver    : nvidia-driver-415 - third-party free
driver    : nvidia-driver-440 - third-party free recommended
driver    : nvidia-driver-410 - third-party free
driver    : nvidia-driver-435 - distro non-free
driver    : xserver-xorg-video-nouveau - distro free builtin
```

1.3 install your NVIDIA graphics driver:

```
sudo apt-get install nvidia-driver-435
```

1.4 reboot your system :

```
sudo reboot now
```

1.5 Once you are back at your terminal, run the nvidia-smi command to query your GPU and check its status:

```
wang@danyi-System-Product-Name:~/yolo_test/build$ nvidia-smi
Wed May 27 20:30:14 2020

+-----+
| NVIDIA-SMI 435.21      Driver Version: 435.21      CUDA Version: 10.1  |
+-----+
| GPU   Name               Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+
|  0   GeForce GTX 106...    Off   | 00000000:01:00.0 On   |           N/A       |
|  0%   58C    P2      32W / 120W |  733MiB /  6075MiB |           1%      Default |
+-----+-----+

Processes:
+-----+
| GPU    PID    Type   Process name                      GPU Memory |
|                               Usage      |
+-----+-----+
|  0     1340   G     /usr/lib/xorg/Xorg                 18MiB     |
|  0     1397   G     /usr/bin/gnome-shell               48MiB     |
|  0     2059   G     /usr/lib/xorg/Xorg                 326MiB    |
|  0     2191   G     /usr/bin/gnome-shell               239MiB    |
|  0     3185   G     /usr/lib/firefox/firefox           1MiB      |
|  0     3235   G     ...g/Qt5.9.9/Tools/QtCreator/bin/qtcreator 2MiB      |
|  0     5601   C     /usr/lib/libreoffice/program/soffice.bin 63MiB     |
|  0     8792   G     /usr/lib/firefox/firefox           1MiB      |
|  0    13654   G     /snap/zoom-client/80/zoom/zoom     26MiB     |
+-----+-----+
```

## 1.6 download CUDA 10.1.

(the cuda version depends on your Driver version, see the red circle on the image above.)

The following commands will both *download* and *install* CUDA 10.1 right from your terminal

```
wget https://developer.nvidia.com/compute/cuda/10.1/Prod/local_installers/
cuda_10.1.105_418.39_linux
mv cuda_10.1.105_418.39_linux cuda_10.1.105_418.39_linux.run
chmod +x cuda_10.1.105_418.39_linux.run
sudo ./cuda_10.1.105_418.39_linux.run --override
```

**Note:** As you follow these commands take note of the line-wrapping due to long URLs/filenames.

## 1.7 update bash profile

```
vi ~/.bashrc
```

Insert the following lines at the bottom of the profile:

```
# NVIDIA CUDA Toolkit
export PATH=/usr/local/cuda-10.2/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-10.2/lib64
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
```

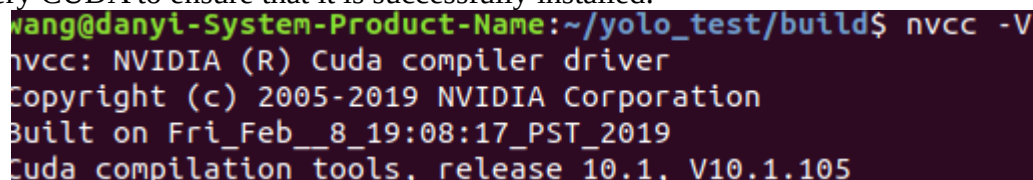
```
#CUDA
```

```
export PATH=/usr/local/cuda-10.2/bin${PATH:+:${PATH}}
export LD_LIBRARY_PATH=/usr/local/cuda-10.2/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

```
then:
```

```
source ~/.bashrc
```

## 1.8 query CUDA to ensure that it is successfully installed:



```
wang@danyi-System-Product-Name:~/yolo_test/build$ nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2019 NVIDIA Corporation
Built on Fri Feb  8 19:08:17 PST 2019
Cuda compilation tools, release 10.1, V10.1.105
```

## 1.9 Go ahead and download **cuDNN v7.6.4 for CUDA 10.1** from the following link:

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

1. Download cuDNN v7.6.4 (September 27, 2019), for CUDA 10.1
2. cuDNN Library for Linux
3. And then allow the .zip file to download (you may need to create an account on NVIDIA's website to download the cuDNN files)

## 1.10 install cuDNN:

```
tar -zxf cudnn-10.1-linux-x64-v7.6.4.38.tgz
tar -zxf cudnn-10.2-linux-x64-v8.0.1.13.tgz
tar -zxf cudnn-10.2-linux-x64-v7.6.5.32.tgz
cd cudnn-10.1
cd cuda
sudo cp cudnn.h /usr/local/cuda/include/
sudo cp cuda/include/cudnn.h /usr/local/cuda/include/

sudo cp cuda/lib64/libcudnn* /usr/local/cuda/lib64/

sudo chmod a+r /usr/local/cuda/include/cudnn.h
```

```
sudo chmod a+r /usr/local/cuda/lib64/libcudnn*
```

```
sudo cp -P lib64/* /usr/local/cuda/lib64/  
sudo cp -P include/* /usr/local/cuda/include/  
cd ~  
cat /usr/local/cuda/include/cudnn.h | grep CUDNN_MAJOR -A 2
```

## 2. Install opencv from source

## start with installing OpenCV 4.2.0 (version in my computer) on Ubuntu 18.04. OpenCV uses intensively third-party software libraries. These must be installed on Ubuntu before OpenCV can be set up.

```
sudo apt-get update  
sudo apt-get upgrade
```

```
sudo apt-get install build-essential cmake git 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 libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev  
sudo apt-get install libfaac-dev libmp3lame-dev libtheora-dev  
sudo apt-get install libavresample-dev libvorbis-dev  
sudo apt-get install libopencore-amrnb-dev libopencore-amrwb-dev  
sudo apt-get install libgtk2.0-dev libcanberra-gtk*  
sudo apt-get install x264 libxvidcore-dev libx264-dev libgtk-3-dev  
sudo apt-get install python3-dev python3-numpy python3-pip  
sudo apt-get install python3-testresources  
sudo apt-get install libtbb2 libtbb-dev libdc1394-22-dev  
sudo apt-get install libv4l-dev v4l-utils  
cd /usr/include/linux  
sudo ln -s -f ../libv4l1-videodev.h videodev.h  
cd ~  
sudo apt-get install libxine2-dev  
sudo apt-get install software-properties-common  
sudo add-apt-repository "deb http://security.ubuntu.com/ubuntu xenial-security main"  
sudo apt-get update  
sudo apt-get install libjasper-dev  
sudo apt-get install libopenblas-dev libatlas-base-dev libblas-dev  
sudo apt-get install liblapack-dev gfortran  
sudo apt-get install libhdf5-dev protobuf-compiler  
sudo apt-get install libprotobuf-dev libgoogle-glog-dev libgflags-dev
```

#### install openGL

```
sudo apt-get install build-essential libgl1-mesa-dev  
sudo apt-get install freeglut3-dev  
sudo apt-get install libglew-dev libsdl2-dev libsdl2-image-dev libglm-dev libfreetype6-dev  
sudo apt-get install libgtkglext1-dev
```

##Download OpenCV.

```
cd ~  
wget -O opencv.zip https://github.com/opencv/opencv/archive/4.3.0.zip  
wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/4.3.0.zip
```

```
unzip opencv.zip
unzip opencv_contrib.zip
```

```
mv opencv-4.3.0 opencv
mv opencv_contrib-4.3.0 opencv_contrib
```

```
cd opencv
mkdir build
cd build
```

##Build Make

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D OPENCV_EXTRA_MODULES_PATH=~/.opencv_contrib/modules \
-D BUILD_TIFF=ON \
-D WITH_FFMPEG=ON \
-D WITH_GSTREAMER=ON \
-D WITH_TBB=ON \
-D BUILD_TBB=ON \
-D WITH_EIGEN=ON \
-D WITH_V4L=ON \
-D WITH_LIBV4L=ON \
-D WITH_VTK=OFF \
-D WITH_OPENGL=ON \
-D OPENCV_ENABLE_NONFREE=ON \
-D INSTALL_C_EXAMPLES=OFF \
-D INSTALL_PYTHON_EXAMPLES=OFF \
-D BUILD_NEW_PYTHON_SUPPORT=ON \
-D OPENCV_GENERATE_PKGCONFIG=ON \
-D BUILD_TESTS=OFF \
-D BUILD_EXAMPLES=OFF \
-D WITH_CUDA=ON \
-D ENABLE_FAST_MATH=ON \
-D CUDA_FAST_MATH=ON \
-D WITH_CUDNN=ON \
-D OPENCV_DNN_CUDA=ON \
-D ENABLE_FAST_MATH=1 \
-D CUDA_FAST_MATH=1 \
-D WITH_CUBLAS=ON \
-D WITH_GTK_2_X=ON \
-D CUDA_ARCH_BIN=7.5 ..
```

NOTE: The ABOVE value of CUDA\_ARCH\_BIN depends on which GPU you are using, so ensure you know your GPU model ahead of time.

**IT IS VERY IMPORTANT!**

Failing to correctly set your CUDA\_ARCH\_BIN variable can result in OpenCV still compiling but failing to use your GPU for inference (making it troublesome to diagnose and debug).

###use the nvidia-smi command:

```
wang@danyi-System-Product-Name:~/yolo_test/build$ nvidia-smi
Wed May 27 20:30:14 2020
```

NVIDIA-SMI 435.21		Driver Version: 435.21		CUDA Version: 10.1	
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile Uncorr. ECC
Fan	Temp	Perf Pwr:Usage/Cap		Memory-Usage	GPU-Util Compute M.
0	GeForce GTX 106...	Off	00000000:01:00.0	On	N/A
0%	58C	P2 32W / 120W	733MiB / 6075MiB		1% Default

Processes:					GPU Memory Usage
GPU	PID	Type	Process name		
0	1340	G	/usr/lib/xorg/Xorg		18MiB
0	1397	G	/usr/bin/gnome-shell		48MiB
0	2059	G	/usr/lib/xorg/Xorg		326MiB
0	2191	G	/usr/bin/gnome-shell		239MiB
0	3185	G	/usr/lib/firefox/firefox		1MiB
0	3235	G	...g/Qt5.9.9/Tools/QtCreator/bin/qtcreator		2MiB
0	5601	C	/usr/lib/libreoffice/program/soffice.bin		63MiB
0	8792	G	/usr/lib/firefox/firefox		1MiB
0	13654	G	/snap/zoom-client/80/zoom/zoom		26MiB

I am using GeForce GTX 1060.

**You can find your NVIDIA GPU architecture version for your particular GPU using this page:**

<https://developer.nvidia.com/cuda-gpus>

### GeForce and TITAN Products

GPU	Compute Capability
NVIDIA TITAN RTX	7.5
Geforce RTX 2080 Ti	7.5
Geforce RTX 2080	7.5
Geforce RTX 2070	7.5
Geforce RTX 2060	7.5
NVIDIA TITAN V	7.0
NVIDIA TITAN Xp	6.1
NVIDIA TITAN X	6.1
GeForce GTX 1080 Ti	6.1
GeForce GTX 1080	6.1
GeForce GTX 1070	6.1
GeForce GTX 1060	6.1
GeForce GTX 1050	6.1
GeForce GTX TITAN X	5.2

### GeForce Notebook Products

GPU	Compute Capability
Geforce RTX 2080	7.5
Geforce RTX 2070	7.5
Geforce RTX 2060	7.5
GeForce GTX 1080	6.1
GeForce GTX 1070	6.1
GeForce GTX 1060	6.1
GeForce GTX 980	5.2
GeForce GTX 980M	5.2
GeForce GTX 970M	5.2
GeForce GTX 965M	5.2
GeForce GTX 960M	5.0
GeForce GTX 950M	5.0
GeForce 940M	5.0
GeForce 930M	5.0

The 'Compute Capability' for your GPU IS YOUR 'NVIDIA GPU architecture version'. I am using GeForce GTX 1060, so `CUDA_ARCH_BIN=6.1`. But yours may differ, check your version before `cmake` your `opencv`.

##Make OpenCV

speed things up by using all your cores in your machine working simultaneously. The command `nproc` gives you the number of cores available. In my machine is 8.

```
make -j8
sudo make install
sudo ldconfig
```

#####check the installation in Python 3

```
wang@danyi-System-Product-Name:~$ python3
Python 3.6.9
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'4.2.0'
>>> cv2.getBuildInformation()
```

```
>>> cv2.getBuildInformation()
General configuration for OpenCV 4.2.0 =====
Version control:          unknown
Extra modules:           Location (extra): /home/wang/opencv_contrib/mod
Platform:                Linux 4.15.0-99-generic x86_64
CMake:                   Linux 4.15.0-99-generic x86_64
CMake generator:        Unix Makefiles
CMake build tool:       /usr/bin/make
Configuration:          RELEASE
CPU/HW features:        Baseline:
SSE SSE2 SSE3          requested: SSE3
Dispatched code generation: SSE4_1 SSE4_2 FP16 AVX AVX2 AVX512_SKX
requested: SSE4_1 SSE4_2 AVX FP16 AVX2 AVX51
2_SKX SSE4_1 (14 files): + SSE3 SSE4_1 SSE4_2 (1 files): + SSE3 SSE4_1 POPCNT SSE4_2 AVX AVX2 (4
files): + SSE3 SSE4_1 POPCNT SSE4_2 AVX AVX2 (27 files): + SSE3 SSE4_1 POPCNT SSE4_2 FP16 FMA3 AVX AVX2
AVX512_SKX (3 files): + SSE3 SSE4_1 POPCNT SSE4_2 FP16 F
MA3 AVX AVX2 AVX_512F AVX512_COMMON AVX512_SKX C/C++:
Built as dynamic libs?: YES
C++ Compiler:          /usr/bin/c++ (ver 7.5.0)
C++ flags (Release):   -fsigned-char
-fsigned-char -W -Wall -Werror=return-type -Werror=non-virtual-dtor -Werror=address -Werror=sequence-point -Wformat -Werror=format-security -Wmissing-declarations -Wundef -Winit-self -Wpointer-arith -Wshadow
-Wsign-promo -Wuninitialized -Winit-self -Wsuggest-override -Wno-delete-non-virtual-dtor -Wno-comment -Wimplicit-fallthrough=3 -Wno-strict-overflow -fdiagnostics-show-option -Wno-long-long -pthread -fo
mit-frame-pointer -ffunction-sections -fdata-sections -msse -msse2 -msse3 -fvisibility-hidden -fvisibility-inlines-hidden -O3 -DNDEBUG -DNDEBUG
C++ flags (Debug):     -fsigned-char -ffast-mat
h -W -Wall -Werror=return-type -Werror=non-virtual-dtor -Werror=address -Werror=sequence-point -Wformat -Werror=format-security -Wmissing-declarations -Wundef -Winit-self -Wpointer-arith -Wshadow -Wsign-p
romo -Wuninitialized -Winit-self -Wsuggest-override -Wno-delete-non-virtual-dtor -Wno-comment -Wimplicit-fallthrough=3 -Wno-strict-overflow -fdiagnostics-show-option -Wno-long-long -pthread -fomit-frame-p
ointer -ffunction-sections -fdata-sections -msse -msse2 -msse3 -fvisibility-hidden -fvisibility-inlines-hidden -g -O0 -DDEBUG -DDEBUG
C Compiler:           /usr/bin/cc
C flags (Release):    -fsigned-char -ffast-math -W -Wall -Werror=return-type -Werror=non-virtual-dtor -Werror=address -Werror=sequence-point -Wformat -Werror=format-security -Wmissing-declarations -Wmissing-prot
otypes -Wstrict-prototypes -Wundef -Winit-self -Wpointer-arith -Wshadow -Wuninitialized -Winit-self -Wno-comment -Wimplicit-fallthrough=3 -Wno-strict-overflow -fdiagnostics-show-option -Wno-long-long -pthre
ad -fomit-frame-pointer -ffunction-sections -fdata-sections -msse -msse2 -msse3 -fvisibility-hidden -O3 -DNDEBUG -DNDEBUG
C flags (Debug):      -fsigned-char -ffast-math -W -Wall -Werror=ret
urn-type -Werror=non-virtual-dtor -Werror=address -Werror=sequence-point -Wformat -Werror=format-security -Wmissing-declarations -Wmissing-prototypes -Wstrict-prototypes -Wundef -Winit-self -Wpointer-arith
-Wshadow -Wuninitialized -Winit-self -Wno-comment -Wimplicit-fallthrough=3 -Wno-strict-overflow -fdiagnostics-show-option -Wno-long-long -pthread -fomit-frame-pointer -ffunction-sections -fdata-sections
-msse -msse2 -msse3 -fvisibility-hidden -g -O0 -DDEBUG -DDEBUG
Linker flags (Release): -Wl,-gc-sections
Linker flags (Debug):  -Wl,-gc-sections
ccache:
NO
Precompiled headers:   NO
Extra dependencies:    m pthread /usr/lib/x86_64-linux-gnu/libGL.so /usr/lib/x86_64-linux-gnu/libGLU.so cudart_static -lpthread dl rt nppc nppial np
ppicom nppidei nppif nppig nppim nppist nppisu nppitc npps cublas cudnn cuFFT -L/usr/local/cuda-10.1/lib64 -L/usr/lib/x86_64-linux-gnu/lib3rdpartydependencies:
aruco bgsegm bioinspired cal3d ccalib core_cudaarithm cudabgsegm cudacodec cudafeatures2d cudafilters cudaimgproc cudalegacy cudaobjdetect cudaoptflow cudastereo cudawarping cudexd
cudatasets dnn dnn_objdetect dnn_superres dpm face_features2d flann freetype fuzzy gapi hdf hfs highgui img_hash imgcodecs imgproc line_descriptor ml_objdetect optflow phase_unwrapping photo plot python2 pyth
on3_quality reg rgbd_saliency shape stereo stitching structured_light superres surface_matching text tracking ts video videoio videostab xfeatures2d ximgproc xobjdetect xphoto
Disabled:
world
Disabled by dependency: -
Unavailable:          cnn_3dobj cv java js matlab ovis sfm viz
Applications:         perf_tests apps
Documentation:
NO
Non-free algorithms:  YES
GUI:                  GTK+:
YES (ver 2.24.32)
GThread:              YES (ver 2.56.4)
GtkGLExt:             YES (ver 1.
2.0)
OpenGL support:       YES (/usr/lib/x86_64-linux-gnu/libGL.so /usr/lib/x86_64-linux-gnu/libGLU.so)
Media I/O:            ZLib:
/usr/lib/x86_64-linux-gnu/libz.so (ve
r 1.2.11)
JPEG:                 /usr/lib/x86_64-linux-gnu/libjpeg.so (ver 80)
WEBP:                 build (ver encoder: 0x020e)
PNG:                  /usr/lib/x86_64-li
nux-gnu/libpng.so (ver 1.6.34)
TIFF:                 build (ver 42 - 4.0.10)
JPEG 2000:           /usr/lib/x86_64-linux-gnu/libjasper.so (ver 1.900.1)
OpenEXR:              YES
build (ver 2.3.0)
HDR:                 YES
SUNRASTER:           YES
PXM:                 YES
PFM:                 YES
Video I/O:           Video I/O:
DC1394:
YES (55.78.100)
swscale:             YES (4.8.100)
avresample:          YES (3.7.0)
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:            sources (2019.0.0)
at:                  /home/wang/opencv/build
/3rdparty/ippcv/ippcv_lnx/iw
Lapack:              NO
Eigen:               NO
Custom HAL:          NO
Protobuf:            build (3.5.1)
NVIDIA GPU arch:    61
NVIDIA PTX archs:
cuDNN:               YES (ver 7.6.4)
OpenCL:              YES (no extra features)
Include path:        /home/wang/opencv/3rdparty/include/opencv/1.2
Link libraries:      Dynamic load
Python 2:            Interpreter:
/usr/bin/python2.7 (ver 2.7.17)
Libraries:          /usr/lib/x86_64-linux-gnu/libpython2.7.so (ver 2.7.17)
numpy:              /home/wang/.local/lib/python2.7/site-packages/nu
mpy/core/include (ver 1.16.6)
install path:       lib/python2.7/dist-packages/cv2/python-2.7
Python 3:           Interpreter:
/usr/bin/python3 (ver 3.6.9)
Libraries:          /usr/lib/x86_64-linux-gnu/libpython3.6m.so (ver 3.6.9)
numpy:              /home/wang/.local/lib/python3.6/site-packages/numpy/core/include (ver 1.18.3)
install path:       lib/python3.6/dist-packages/cv2/python-3.6
Python (for build): /usr/bin/python2.7
Java:
NO
Java wrappers:       NO
Java tests:          NO
Install to:          /usr/local
```

Check the installation information of OpenGL, CUDA, cuDNN