**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 Memory |
|  GPU       PID   Type   Process name                             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

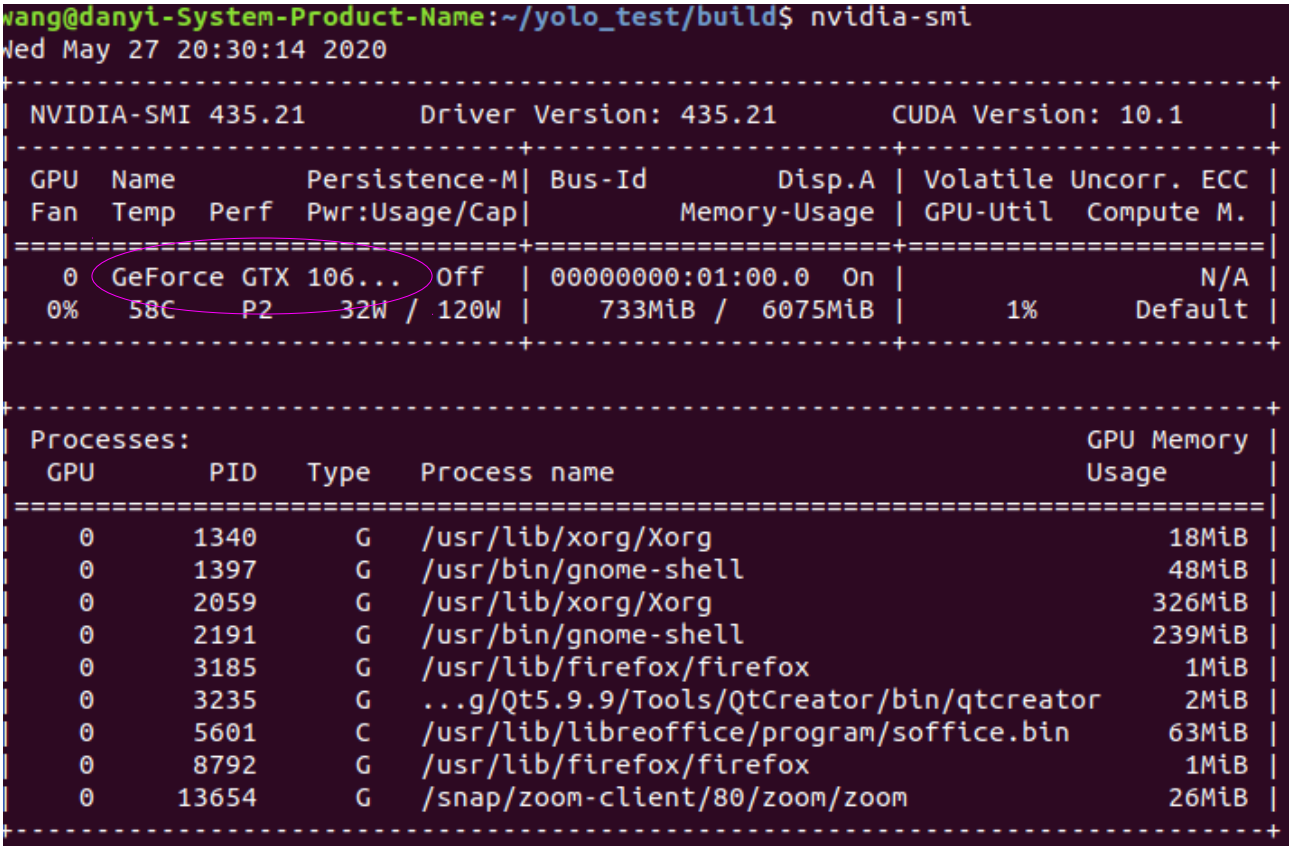
#### install openGL
sudo apt-get install build-essential libgl1-mesa-dev
sudo apt-get install freeglut3-dev
sudo apt-get install libglew-dev libsdl2-dev libsdl2-image-dev libglm-dev libfreetype6-dev
```
sudo apt-get install libgtkglext1-dev
```

##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:




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 architectute version’. I am using GeForce GTX 1060, so  CUDA_ARCH_BIN=6.1.But yours may different, 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 availible. 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()
```


Check the installation information of OpenGL, CUDA, cuDNN