

# rtx4070 final

## ubuntu 20.04 설치

1. 윈도우로 접속 해서 검색 창에 “하드 디스크 파티션 만들기 및 포맷” 눌러서 들어간 다음 100gb 짜리 할당해서 새 “새 단순 볼륨” 누르면 새 볼륨 생성됨.
2. 윈도우 종류 후 우분투 usb 꽂은 다음에 pc 키면서 f2 누르면 BIOS 모드로 진입 가능 그 후에 ubuntu 들어있는 usb를 1번으로 설정한 후에 save&exit 종료.
3. 그럼 ubuntu 설치 듯. install ubuntu 누르고 뒤에서 2번째까지 끝까지 그대로 누르고 **뒤에서 두 번째 까지 가서 “something else” 선택 후 넘어가기**
4. 마지막에서 100gb 할당 되어있는거 “-” 눌러서 free space로 만들어주고 그 100gb짜리 free space 누르고 “+”눌러서 mount point 눌러서 “/” 선택후 ok (**이 과정 2번 반복**)
5. 마지막으로 install now 누르면 어떤 창이 하나 뜨는데 continue를 누를 수 있음 이 창 말고 다른 창이 나오면 문제가 일어난거임 continue 누를 수 있는 창이면 걱정 ㄴ ㄴ
6. seoul 선택하고 continue
7. 위에 이름이랑 이름포함 위에서 3개 칸 전부다 kuuve 로 맞춰주고 password 스페이스바로 설정.
8. 마지막으로 좀 중요한데 restart now 누르고 usb를 빼야 부팅할때 usb로 부팅이 안되어서 잘 빼야하는데 이건 어디까지나 내 기준인데 화면 꺼지자 마자 바로 usb 뺏음 그래야 좀 덜 깨지는 듯.

## Yolov5 사용하기 위한 환경 설정

### 한글 설치하기

setting → region&language에서 한글 설치

부팅 한 다음에 위에서 한 한글 설치 마무리

Korean(한글) 써있는 거 추가

환경 설정 버튼 눌러서 Add하고 → alt\_r apply → ok

(한/영 키 눌러서 한영 바꾸려고 설정 해주는 과정)

### nvidia 그래픽 카드 잡아주기.

```
$ sudo apt update  
$ sudo add-apt-repository ppa:graphics-drivers/ppa  
$ sudo apt update  
$ sudo apt upgrade  
$ ubuntu-drivers devices non-free recommended nvidia-drive
```

위에 명령어를 치면 우리에게 어떤 버전을 설치하라고 추천을 해준다.

```
$ sudo apt-get install nvidia-driver-(추천해준 숫자)
```

```
$ sudo reboot
```

부팅 한 다음에

```
$ nvidia-smi 이용해서 제대로 설치 되었는지 확인 하기.
```

rtx기준 535 설치하면 됨

## Opencv 설치

- 서론으로 우분투를 밀지 않고 환경 설정을 했지만 밀게 된 가장 큰 이유가 이 친구 때문이다. 우리가 ROS를 설치 할 때 OPENCV가 떨려서 설치가 된다. 그 친구는 버전이 4.X.X 대로 설치 되지만 우리가 이용하려면 버전 3.X.X 대로 설치해야해서 우분투를 밀고 ROS를 깔기 전에 환경 설정을 썩 다 한 이후에 하는 것이다.

```
$ sudo apt-get install python2.7-dev python3-dev python-numpy python3-numpy
```

```
$ sudo apt-get -y install qt4-default
```

→ 위에 qt4를 설치하면 오류가 날 것 이다. 이 오류를 해결하려면

Install QT4 on Ubuntu 20.04

 <https://www.skyer9.pe.kr/wordpress/?p=4587>

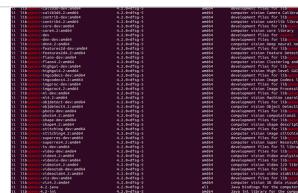
여기 들어가면 그냥 썩 다 따라하면 오류가 나지 않을 것이다.

그 이후에 본격적으로 opencv 설치 하려면

[몽돌] OpenCV 3.4.5 installation in Ubuntu 20.04 ( OpenCV 3.4.5를 Ubuntu 20.04에 설치해보기 )

참조 블로그 : <https://ghostweb.tistory.com/823> 현재 구성입니다. Ubuntu 20.04 OpenCV 3.4.5 Google에서 "How to install OpenCV in Ubuntu 20.04" 라고 검색을 하시면 여러 웹사이트가 나오는 것을 확인할 수 있습니다. 방법은 크게 2가지로 나눠집니다. 1. Installing OpenCV from the Ubuntu Repository. 2. Installing OpenCV

 <https://wth-mongdol.tistory.com/223>



### #빌드, 소스 관련

```
$ sudo apt install -y build-essential cmake pkg-config git
```

여기서부터 따라하면 된다. 바로 밑에 써 놓은게 다 그거임.

#빌드, 소스 관련

```
$ sudo apt install -y build-essential cmake pkg-config git
```

# 이미지 관련

```
$ sudo apt install -y libjpeg-dev libtiff5-dev libpng-dev libjasper-dev
```

##여기서 아마 오류가 발생 할 것이다.

E: Unable to locate package libjasper-dev라고. 이걸 해결하려면

Unable to locate package libjasper-dev

I want to install opencv in ubuntu 17.04 and I know that the jasper library is removed from  
ubuntu 17.04  
what should I do to complete install opencv correctly ???  
<https://stackoverflow.com/questions/44468081/unable-to-locate-package-libjasper-dev>



```
$ sudo add-apt-repository 'deb http://security.ubuntu.com/ubuntu xenial-security main'  
$ sudo apt update  
$ sudo apt install libjasper1 libjasper-dev
```

답변 중에 이거 3개 써져있는 답변으로 해결하면 됨.

# 동영상 및 카메라 관련

```
$ sudo apt install -y libavcodec-dev libavformat-dev libswscale-dev libdc1394-22-dev libxvidcore-dev libx264-dev  
libxine2-dev libv4l-dev v4l-utils libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev
```

# GUI, qt ( qt Creator -> 정말 많이 씁니다. ), 최적화, Python 3 관련

```
$ sudo apt install -y libgtk-3-dev libatlas-base-dev libeigen3-dev gfortran
```

```
$ sudo apt install -y python3-dev python3-numpy libtbb2 libtbb-dev
```

# 이때 Qt 버전을 꼭 명심해야 합니다. 자세한 내용은 "OpenCV 3 qt version" 이라고 구글링 하시면 됩니다.

```
$ sudo apt-get -y install qt5-default
```

여기까지 끝난다음 opencv 폴더 자체를 만들어서 진행 할 것이다.

```
$ mkdir opencv  
$ cd opencv
```

opencv 폴더 안에서

```
$ wget -O opencv-3.4.5.zip https://github.com/opencv/opencv/archive/3.4.5.zip  
$ wget -O opencv_contrib-3.4.5.zip https://github.com/opencv/opencv_contrib/archive/3.4.5.zip
```

받아온 집들을 압축을 풀어주면

```
$ unzip opencv-3.4.5.zip  
$ unzip opencv_contrib-3.4.5.zip
```

그 후에 압축 풀어서 만들어진 폴더에 들어가서 build 폴더 만들어서 들어가준다.

```
$ cd opencv-3.4.5  
$ mkdir build && cd build  
$ cmake \ -D CMAKE_BUILD_TYPE=RELEASE \ -D CMAKE_INSTALL_PREFIX=/usr/local \ -D WITH_TBB=OFF \ -D WITH_IPP=OFF \ -D WITH_1394=OFF \ -D BUILD_WITH_DEBUG_INFO=OFF \ -D BUILD_DOCS=OFF \ -D INSTALL_C_EXAMPLES=ON \ -D INSTALL_PYTHON_EXAMPLES=ON \ -D BUILD_EXAMPLES=OFF \ -D BUILD_TESTS=OFF \ -D BUILD_PERF_TESTS=OFF \ -D WITH_QT=ON \ -D WITH_GTK=OFF \ -D WITH_OPENGL=ON \ -D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-3.4.5/modules \ -D WITH_V4L=ON \ -D WITH_FFMPEG=ON \ -D WITH_XINE=ON \ -D BUILD_NEW_PYTHON_SUPPORT=ON \ -D OPENCV_GENERATE_PKGCONFIG=ON ..\
```

그 후에 사용가능 코어 확인

\$ nproc **이게 진짜 중요한거다.**

\$ make -j숫자 ( or make 만 ) 실행을 해줍니다.

# 이제 설치를 해줍니다.

\$ sudo make install

# 설치 후, 버전 확인

\$ pkg-config --modversion opencv

3.4.5 출력 되는지 확인하기.

## Python 3.8 설치

opencv 설치하던 터미널은 종료하고 다시 home에서 터미널 창을 끔다.

\$ sudo apt update

\$ sudo apt install software-properties-common

\$ sudo add-apt-repository ppa:deadsnakes/ppa

\$ sudo apt install python3.8

\$ which python3.8

- output: /usr/bin/python3.8

\$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python3.8 3

- output: update-alternatives: using /usr/bin/python3.8 to provide /usr/bin/python (python) in auto mode

\$ python -V으로 설치확인

## Cuda 11.8 설치

### CUDA Toolkit Archive

Previous releases of the CUDA Toolkit, GPU Computing SDK, documentation and developer drivers can be found using the links below. Please select the release you want from the list below, and be sure to check [www.nvidia.com/drivers](http://www.nvidia.com/drivers) for more recent production drivers appropriate for your hardware configuration.

 <https://developer.nvidia.com/cuda-toolkit-archive>

## CUDA Toolkit 11.8 Downloads

Home

### 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
	WSL-Ubuntu								
Version	18.04	20.04	22.04						
Installer Type	deb (local)	deb (network)	runfile (local)						

### Download Installer for Linux Ubuntu 20.04 x86\_64

The base installer is available for download below.

#### Base Installer

Installation Instructions:

```
$ wget https://developer.download.nvidia.com/compute/cuda/11.8.0/local_installers/cuda_11.8.0_520.61.05_linux.run  
$ sudo sh cuda_11.8.0_520.61.05_linux.run
```

이거 이용해서 설치하면 설치 하면 나오는데

아래 화면 좀 늦게 나오는데 참을성을 갖고 기다릴것.

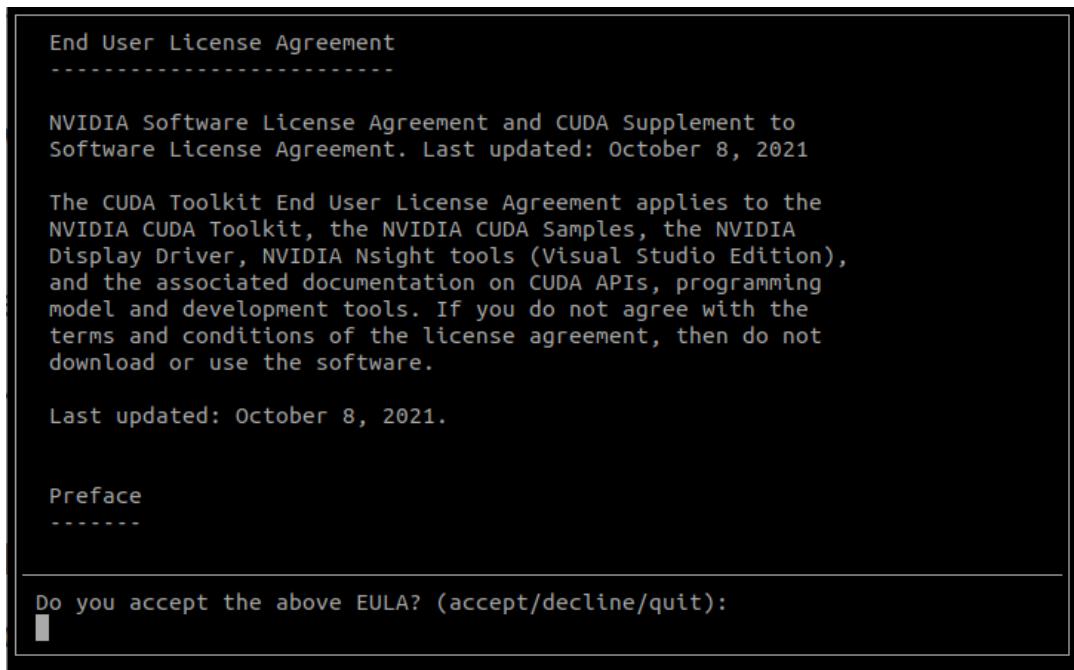
```
Existing package manager installation of the driver found. It is strongly recommended that you remove this before continuing.
```

**Abort**

Continue

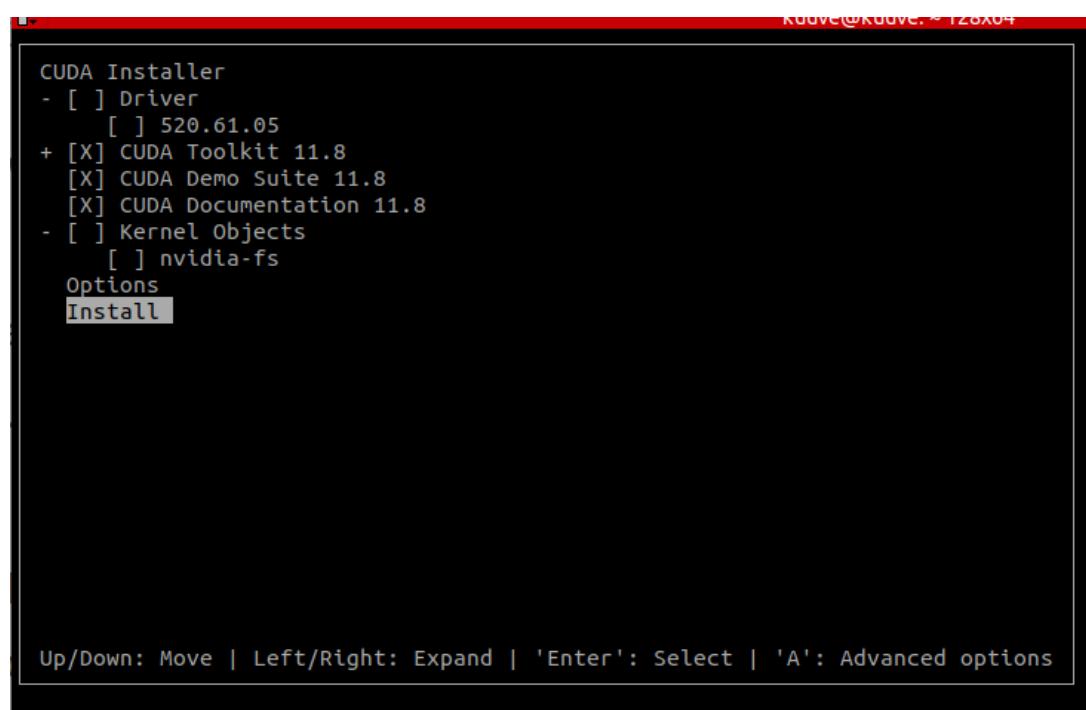
Up/Down: Move | 'Enter': Select

continue 눌러서 가면 됨 엔터



그 다음 accept 또는 누름 됨

Driver 항목에서 스페이스바 눌러서 체크 해제하고, install 항목에서 엔터를 누릅니다.



이게 좀 중요한데 driver 항목에서 스페이스 눌러서 x표로 체크되어 있는거 해제하고 install 항목에서 엔터 누름 끝~  
아래 summary 화면 좀 늦게 나오는데 참고성을 갖고 기다릴것.

=summary= 뜨면 설치 성공한것.

그 이후에

```
$ sudo sh -c "echo 'export PATH=$PATH:/usr/local/cuda-11.8/bin' >> /etc/profile"
```

```

$ sudo sh -c "echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda-11.8/lib64' >> /etc/profile"
$ sudo sh -c "echo 'export CUDADIR=/usr/local/cuda-11.8' >> /etc/profile"
$ source /etc/profile
$ nvcc -V
이거 까지하면 완전 끝난 거임.

```

## cuDNN 8.9.0 for cuda 11.x/cuDNN Library for Linux(x86\_64) 설치

NVIDIA cuDNN Archive

Explore and download past releases from cuDNN GPU-accelerated primitive library for deep neural networks for your development work.

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

[Download cuDNN v8.9.0 \(April 11th, 2023\), for CUDA 11.x](#)

## Local Installers for Windows and Linux, Ubuntu(x86\_64, armsbsa)

[Local Installer for Windows \(Zip\)](#)

[Local Installer for Linux x86\\_64 \(Tar\)](#)

```

$ cd Downloads (다운로드에 들어가서)
$ tar -xf cudnn-linux-x86_64-8.9.0.131_cuda11-archive.tar.xz

```

압축해제한 파일을 품으로 옮겨준 다음에 폴더 이름 cuda로 이름 변경

cd 하고 home 가서 아래 명령어 실행

```

$ sudo cp cuda/include/cudnn* /usr/local/cuda/include
$ sudo cp cuda/lib/libcudnn* /usr/local/cuda/lib64
$ sudo chmod a+r /usr/local/cuda/include/cudnn.h /usr/local/cuda/lib64/libcudnn*

```

```

$ sudo ln -sf /usr/local/cuda-11.8/targets/x86_64-linux/lib/libcudnn_adv_train.so.8.9.0 /usr/local/cuda-11.8/targets/x86_64-linux/lib/libcudnn_adv_train.so.8

```

```

$ sudo ln -sf /usr/local/cuda-11.8/targets/x86_64-linux/lib/libcudnn_ops_infer.so.8.9.0 /usr/local/cuda-11.8/targets/x86_64-linux/lib/libcudnn_ops_infer.so.8

```

```

$ sudo ln -sf /usr/local/cuda-11.8/targets/x86_64-linux/lib/libcudnn_cnn_train.so.8.9.0 /usr/local/cuda-11.8/targets/x86_64-linux/lib/libcudnn_cnn_train.so.8

```

```

$ sudo ln -sf /usr/local/cuda-11.8/targets/x86_64-linux/lib/libcudnn_adv_infer.so.8.9.0 /usr/local/cuda-11.8/targets/x86_64-linux/lib/libcudnn_adv_infer.so.8

```

```
$ sudo ln -sf /usr/local/cuda-11.8/targets/x86_64-linux/lib/libcudnn_ops_train.so.8.9.0
/usr/local/cuda-11.8/targets/x86_64-linux/lib/libcudnn_ops_train.so.8
```

```
$ sudo ln -sf /usr/local/cuda-11.8/targets/x86_64-linux/lib/libcudnn_cnn_infer.so.8.9.0
/usr/local/cuda-11.8/targets/x86_64-linux/lib/libcudnn_cnn_infer.so.8
```

```
$ sudo ln -sf /usr/local/cuda-11.8/targets/x86_64-linux/lib/libcudnn.so.8.9.0
/usr/local/cuda-11.8/targets/x86_64-linux/lib/libcudnn.so.8
```

```
$ sudo ldconfig
$ cd
$ ldconfig -N -v
$ (sed 's/:/ /' <<< $LD_LIBRARY_PATH) 2>/dev/null | grep libcudnn
$ sudo apt install python3-pip (파이썬 3라서)
```

## Tensorrt 8.6 GA for Linux x86\_64 and CUDA 11.x TAR Package

Table 1. List of Supported Features per Platform

	Linux x86-64	Windows x64	Linux ppc64le	Linux AArch64
	<b>8.6.x</b>	<b>8.6.x</b>	<b>8.5.x</b>	<b>8.6.x</b>
<u>Supported NVIDIA CUDA® versions</u>	<a href="#">12.1 update 1<sup>1</sup></a> <a href="#">12.0 update 1</a> <a href="#">11.8</a> <a href="#">11.7 update 1<sup>2</sup></a> <a href="#">11.6 update 2<sup>3</sup></a> <a href="#">11.5 update 2<sup>4</sup></a> <a href="#">11.4 update 4<sup>5</sup></a> <a href="#">11.3 update 1<sup>6</sup></a> <a href="#">11.2 update 2<sup>7</sup></a> <a href="#">11.1 update 1<sup>8</sup></a> <a href="#">11.0 update 1<sup>9</sup></a>	<a href="#">12.1 update 1<sup>10</sup></a> <a href="#">12.0 update 1</a> <a href="#">11.8</a> <a href="#">11.7 update 1<sup>11</sup></a> <a href="#">11.6 update 2<sup>12</sup></a> <a href="#">11.5 update 2<sup>13</sup></a> <a href="#">11.4 update 4<sup>14</sup></a> <a href="#">11.3 update 1<sup>15</sup></a> <a href="#">11.2 update 2<sup>16</sup></a> <a href="#">11.1 update 1<sup>17</sup></a> <a href="#">11.0 update 1<sup>18</sup></a>	<a href="#">11.8</a> 	<a href="#">12.0 update 1 (SBSA)</a> <a href="#">11.4 (JetPack)</a>
<u>Supported cuBLAS versions</u>	12.1.3.1 12.0.2.224 11.11.3.6 11.10.3.66 11.9.2.110 11.7.4.6 11.6.5.2 11.5.1.109 11.4.1.1043 11.3.0.106 11.2.0.252	12.1.3.1 12.0.2.224 11.11.3.6 11.10.3.66 11.9.2.110 11.7.4.6 11.6.5.2 11.5.1.109 11.4.1.1043 11.3.0.106 11.2.0.252	11.11.3.6	12.0.2.224 11.6.5.2
<u>Supported cuDNN versions</u>	<a href="#">cuDNN 8.9.0</a>	<a href="#">cuDNN 8.9.0</a>	<a href="#">cuDNN 8.6.0</a>	<a href="#">cuDNN 8.9.0</a>
TensorRT Python API	Yes	Yes	Yes	Yes
NvUffParser	Yes	Yes	Yes	Yes
NvOnnxParser	Yes	Yes	Yes	Yes
Loops	Yes	Yes	Yes	Yes

<https://developer.nvidia.com/nvidia-tensorrt-8x-download>

# NVIDIA TensorRT 8.x Download

NVIDIA TensorRT is a platform for high performance deep learning inference.

TensorRT works across all NVIDIA GPUs using the CUDA platform.

Please review [TensorRT online documentation](#) for more information, including the [installation guide](#).

I Agree To the Terms of the [NVIDIA TensorRT License Agreement](#)

Please download the version compatible with your development environment.

TensorRT 8.6 GA

## Documentation

- [Online Documentation](#)

## TensorRT 8.6 GA for x86\_64 Architecture

### Debian, RPM, and TAR Install Packages for Linux

- [TensorRT 8.6 GA for Linux x86\\_64 and CUDA 11.0, 11.1, 11.2, 11.3, 11.4, 11.5, 11.6, 11.7 and 11.8 TAR Package](#)

```
$ cd Downloads
```

```
$ tar xzvf TensorRT-8.6.1.6.Linux.x86_64-gnu.cuda-11.8.tar.gz
```

```
$ export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/usr/local/cuda-11.8/lib64:/usr/local/cuda/extras/CUPTI/lib64:/home/kuuve/TensorRT-8.6.1.6/lib"
```

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

- bashrc 진입해서
- export LD\_LIBRARY\_PATH=\$LD\_LIBRARY\_PATH:/home/kuuve/TensorRT-8.6.1.6/lib
- 이걸 맨 밑에 추가하고 저장

```
$ source ~/.bashrc (이거 꼭 해야함.)
```

cd 하고 tensorrt 파일 home으로 옮기기!

```
$ cd TensorRT-8.6.1.6/python  
$ python -m pip install tensorrt-8.6.1-cp38-none-linux_x86_64.whl
```

이건 python3 아님

```
$ cd TensorRT-8.6.1.6/uff  
$ sudo pip install uff-0.6.9-py2.py3-none-any.whl
```

```

$ which convert-to-uf

$ cd TensorRT-8.6.1.6/graphsurgeon

$ python -m pip install graphsurgeon-0.4.6-py2.py3-none-any.whl      ( 여기서 cd로 나가지 않고 그대로 진행한다.)

$ cd TensorRT-8.6.1.6/lib

$ sudo cp ./libnvinfer_builder_resource.so.8.6.1 /usr/lib

$ python

$ import tensorrt

or

$ import tensorrt as trt

아무것도 뜨지 않으면 설치 성공.

```

### **nvinfer 오류 발생하니까 미리 해놓는거**

```

$ cd TensorRT-8.6.1.6

$ sudo mkdir -p /usr/local/cuda-originals/cuda-11.8

$ sudo cp -r /usr/local/cuda-11.8/targets/ /usr/local/cuda-originals/cuda-11.8/

$ sudo cp -r /home/kuuve/TensorRT-8.6.1.6/include/* /usr/local/cuda-11.8/targets/x86_64-linux/include/

$ sudo cp -r /home/kuuve/TensorRT-8.6.1.6/targets/x86_64-linux-gnu/lib/* /usr/local/cuda-11.8/targets/x86_64-linux/lib/

```

## **Anaconda 설치**

기존에 설치하던 터미널창 끄고 새로 키기

```

$ wget https://repo.anaconda.com/archive/Anaconda3-2021.05-Linux-x86\_64.sh

$ sudo bash Anaconda3-2021.05-Linux-x86_64.sh

1. more 나오면 enter 꾸욱 누르고 있기

2. yes 누르기

3. /home/kuuve/anaconda3/.

4. 복불하고 enter하면 이 경로에 아나콘다가 생김

5. yes 누르기

6. $ source /home/kuuve/anaconda3/bin/activate (base)가 앞에 뜨면 아나콘다 접속한 것

7. $ conda init

8. $ source ~/.bashrc

9. $ sudo gedit ~/.bashrc

10. export PATH="/home/kuuve/anaconda3/bin:$PATH" 맨 밑에 복불하고 저장

11. 아나콘다 사용 시 ros 쓸 수 없음, ros쓸 때는 주석을 해줘야 함

$ conda info 설치 확인

```

## **Yolov5 git clone**

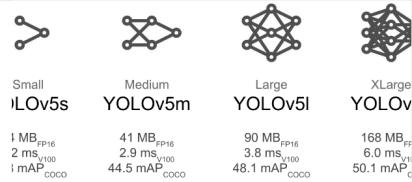
블로그 참조해서 yolov5 환경 설정

<https://github.com/wang-xinyu/tensorrtx>

### yolov5 학습 튜토리얼 1

Custom 데이터로 YoloV5 모델 학습하기 2편 - [AI/Self-Study] - yolov5 학습 튜토리얼 2 (+ mAP, IoU 설명 / Test 와 Inference) 1. 환경 세팅 1) YoloV5 깃헙 레포지토리 clone \$ git clone <https://github.com/ultralytics/yolov5> 2) PyTorch 깔기 아니콘다 가상환경을 만들어주고 CUDA 버전 설정하기 3) yolov5 학습하기

👉 <https://lynnshin.tistory.com/47>



\$ conda create -n yolov5 python=3.8

y 치기

\$ conda activate yolov5

\$ git clone [GitHub - ultralytics/yolov5: YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite](https://github.com/ultralytics/yolov5)

위 명령어 치면 설치 완료

\$ cd yolov5/

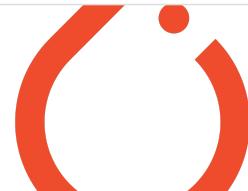
위 명령어 통해서 설치 확인

## PyTorch 설치

### PyTorch

An open source machine learning framework that accelerates the path from research prototyping to production deployment.

👉 <https://pytorch.org/get-started/previous-versions/>



\$ [conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia](https://pytorch.org/get-started/previous-versions/)

위에거 설치할때 멈칫멈칫 오래걸려도 이해해주기^o^

\$ cd yolov5

\$ pip install -r requirements.txt

(코드 실행 안 될 때 다시 하면 됨)

가상 환경에서 나가기 위해서

\$ gedit ~/.bashrc (하고 export까지 주석 처리 하기)

\$ source ~/.bashrc (하면 아나콘다 나가짐)

## ROS noetic 설치

ros 설치는 python 2에서 해야한다.

\$ which python2

```
$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python2 150
$ update-alternatives --list python
python 2가 생김
$ sudo update-alternatives --config python
숫자 0 or 1 python 버전 2나오는걸로 숫자 누르기
터미널 나갔다와서 python -V 하면 2 뜰거임 (python 버전 바꿀 때는 터미널 꺼다 키)
그 다음 ros 설치하면 끝
```

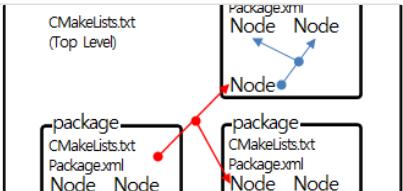
[ROS] - ROS1 설치 (Ubuntu20.04, ROS Noetic)  
ROS1은 Ubuntu 22.04를 지원하지 않는다. 우리는 ROS1 Noetic을 사용할 것이기에, Ubuntu 20.04 버전을 준비해준다. 주요 설치 코드(ex) slam-gmapping Package Install 패키지 리스트 보기 (<https://velog.io/@deep-of-machine/ROS-ROS1-설치-Ubuntu20.04-ROS-Noetic>)



## catkin\_ws 작업공간 만들기

[Jetson ROS] 2. ROS catkin workspace  
ROS를 사용하기 위해 필요한 catkin workspace를 구성하는 단계이다. catkin 이외에도 ROS내에서 사용 가능한 workspace 형태가 있다고 알고 있지만 사용해본적이 없고 또 catkin이 가장 대중적인 방법이기도 하기에 catkin을 소개하겠다. 우선 catkin이란 공통된 source를 가지는 ROS 패키지

<https://junk-research-note.tistory.com/8>



마지막에 패키지 경로에 넣는 것

```
$ gedit ~/.bashrc
마지막줄에
source ~/catkin_ws/devel/setup.bash
추가후 저장하고 나와서
$ source ~/.bashrc
```

## alias 단축기 넣기

```
$ gedit ~/.bashrc
alias l='ls -CF' 밑으로
```

```
alias cs='cd catkin_ws/src'
alias cw='cd catkin_ws'
alias gb='gedit ~/.bashrc'
alias cm='cd ~/catkin_ws && catkin_make'
alias sb='source ~/.bashrc'
alias eb='nano ~/.bashrc'
```

이거 그대로 복복하고 저장하고 나와서

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

- ubuntu termintor ctrl + shift + e 안먹을 경우에 사용하면 됨.

Ctrl + Shift + E 명령어가 안 먹히는 경우(Terminator) · snowdeer's Code Holic

매일매일 공부하는 snowdeer입니다.

👉 <https://snowdeer.github.io/mac-os/2020/09/22/ctrl-shift-e-key-on-ubuntu-20p04/>