

Deep learning in ROS guide

@ Date : Spring semester, 2023

@ Part : Automation part in Industrial AI & Automation class

@ Project #2

@ Instructor : prof. Young-Keun Kim

@ github : https://github.com/HanMinung/Robotarm_Automation

Deep learning in ROS guide

1. Python 3.9 installation
 2. CUDA installation
 3. cuDNN, pytorch installation
 4. Yolov5 implementation
- Reference

1. Python 3.9 installation

By entering the following command in the terminal, user can update the package list and install the necessary components.

```
sudo apt update  
sudo apt install software-properties-common
```

Add the PPA to the system's source list.

```
sudo add-apt-repository ppa:deadsnakes/ppa
```

Install python 3.9 version with below command in the terminal.

```
sudo apt install python3.9
```

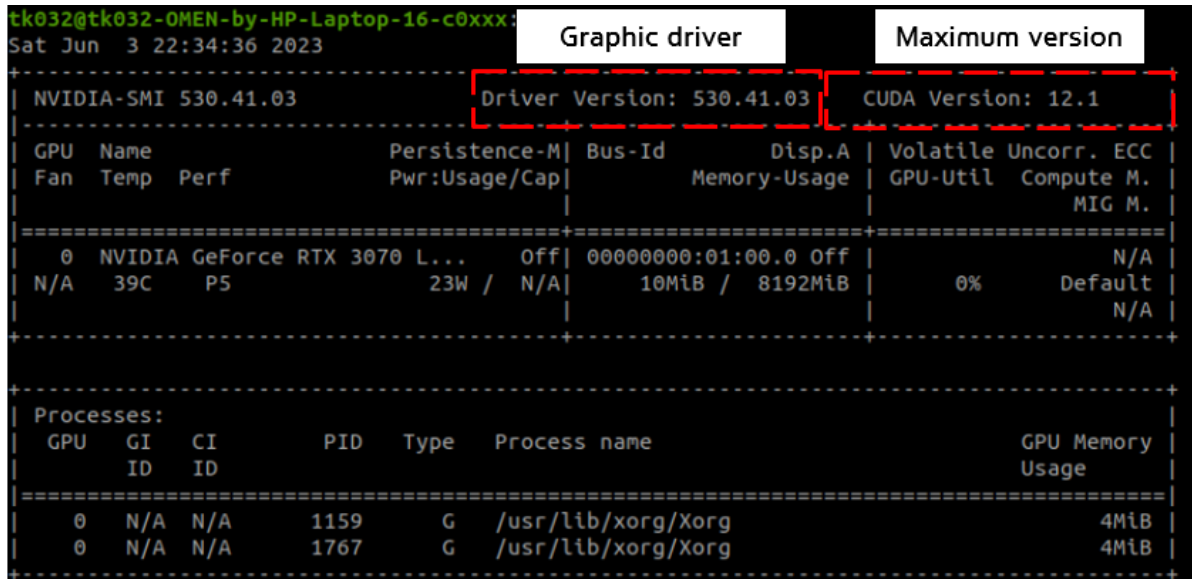
Version verification using below command in the terminal.

```
python3.9 --version
```

2. CUDA installation

1. Check the installation of the NVIDIA driver

```
nvidia-smi
```



The screenshot shows the output of the `nvidia-smi` command. At the top, it displays the system name and date. Below this, a table provides details about the NVIDIA-SMI version (530.41.03) and the installed driver version (530.41.03). A red dashed box highlights the 'Driver Version: 530.41.03' and 'CUDA Version: 12.1' fields. The table also lists GPU details for a GeForce RTX 3070, including fan speed, temperature, power usage, and memory usage. At the bottom, a section titled 'Processes:' lists running processes with their GPU IDs, PIDs, and names.

Graphic driver		Maximum version
NVIDIA-SMI 530.41.03	Driver Version: 530.41.03	CUDA Version: 12.1
GPU Name	Persistence-M	Bus-Id
Fan Temp Perf	Pwr:Usage/Cap	Disp.A
		Memory-Usage
		Volatile Uncorr. ECC
		GPU-Util Compute M.
		MIG M.
		N/A
		Default
		N/A

Processes:						
GPU	GI	CI	PID	Type	Process name	GPU Memory Usage
ID	ID	ID				
0	N/A	N/A	1159	G	/usr/lib/xorg/Xorg	4MiB
0	N/A	N/A	1767	G	/usr/lib/xorg/Xorg	4MiB

2. Check the compatibility between the NVIDIA driver and CUDA version in the link attached below.

link : [CUDA version](#)

CUDA Toolkit	Minimum Required Driver Version for CUDA Minor Version Compatibility*	
	Linux x86_64 Driver Version	Windows x86_64 Driver Version
CUDA 12.1.x	>=525.60.13	>=527.41
CUDA 12.0.x	>=525.60.13	>=527.41
CUDA 11.8.x	>=450.80.02	>=452.39
CUDA 11.7.x	>=450.80.02	>=452.39
CUDA 11.6.x	>=450.80.02	>=452.39
CUDA 11.5.x	>=450.80.02	>=452.39
CUDA 11.4.x	>=450.80.02	>=452.39
CUDA 11.3.x	>=450.80.02	>=452.39
CUDA 11.2.x	>=450.80.02	>=452.39
CUDA 11.1 (11.1.0)	>=450.80.02	>=452.39
CUDA 11.0 (11.0.3)	>=450.36.06**	>=451.22**

3. User can enter the following command to check your CPU architecture.

```
uname -m
```

4. cuda toolkit 다운

cuda toolkit installation link : [click here to download](#)

Operating System

LinuxWindows

Architecture

x86_64ppc64learm64-sbsaaarch64-jetson

Distribution

CentOSDebianFedoraKylinOSOpenSUSERHELRockySLESUbuntuWSL-Ubuntu

Version

18.0420.0422.04

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/12.1.1/local_installers/cuda_12.1.1_530.30.02_linux.run
$ sudo sh cuda_12.1.1_530.30.02_linux.run
```

wget

https://developer.download.nvidia.com/compute/cuda/12.1.1/local_installers/cuda_12.1.1_530.30.02_linux.run

sudo sh cuda_12.1.1_530.30.02_linux.run

5. Version verification after installing CUDA toolkit

- Installed version : CUDA 11.3

```
nvcc -V
```

```
tk032@tk032-OMEN-by-HP-Laptop-16-c0xxx:~$ nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2021 NVIDIA Corporation
Built on Mon_May__3_19:15:13_PDT_2021
Cuda compilation tools, release 11.3, V11.3.109
Build cuda_11.3.r11.3/compiler.29920130_0
```

3. cuDNN, pytorch installation

1. nvidia login for installing cuDNN

- link : [download link](#)

2. After logging in, install cuDNN compatible with the CUDA version.

- link : [download link](#)
- click archive cuDNN release

NVIDIA cuDNN is a GPU-accelerated library of primitives for deep neural networks.

☒ I Agree To the Terms of the [cuDNN Software License Agreement](#)

Note: Please refer to the [Installation Guide](#) for release prerequisites, including supported GPU architectures and compute capabilities, before downloading.

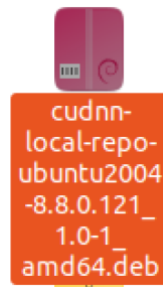
For more information, refer to the cuDNN Developer Guide, Installation Guide and Release Notes on the [Deep Learning SDK Documentation](#) web page.

Download cuDNN v8.9.2 (June 1st, 2023), for CUDA 12.x

Download cuDNN v8.9.2 (June 1st, 2023), for CUDA 11.x

[Archived cuDNN Releases](#)

3. Unzip downloaded file



4. Copy the unzipped file with below commands in the terminal

- command to copy : `cp [복사할 디렉토리/파일] [복사될 디렉토리/파일]`
- command to unzip : `tar xvzf [압축 파일명]`

```
tar xvzf cudnn-11.3-linux-x64-v8.2.1.32.tgz
sudo cp cuda/include/cudnn* /usr/local/cuda/include
sudo cp cuda/lib64/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-11.3/targets/x86_64-
linux/lib/libcudnn_adv_train.so.8.2.1 /usr/local/cuda-11.3/targets/x86_64-
linux/lib/libcudnn_adv_train.so.8

sudo ln -sf /usr/local/cuda-11.3/targets/x86_64-
linux/lib/libcudnn_ops_infer.so.8.2.1 /usr/local/cuda-11.3/targets/x86_64-
linux/lib/libcudnn_ops_infer.so.8

sudo ln -sf /usr/local/cuda-11.3/targets/x86_64-
linux/lib/libcudnn_cnn_train.so.8.2.1 /usr/local/cuda-11.3/targets/x86_64-
linux/lib/libcudnn_cnn_train.so.8

sudo ln -sf /usr/local/cuda-11.3/targets/x86_64-
linux/lib/libcudnn_adv_infer.so.8.2.1 /usr/local/cuda-11.3/targets/x86_64-
linux/lib/libcudnn_adv_infer.so.8
```

```
sudo ln -sf /usr/local/cuda-11.3/targets/x86_64-  
linux/lib/libcudnn_ops_train.so.8.2.1 /usr/local/cuda-11.3/targets/x86_64-  
linux/lib/libcudnn_ops_train.so.8
```

```
sudo ln -sf /usr/local/cuda-11.3/targets/x86_64-  
linux/lib/libcudnn_cnn_infer.so.8.2.1 /usr/local/cuda-11.3/targets/x86_64-  
linux/lib/libcudnn_cnn_infer.so.8
```

```
sudo ln -sf /usr/local/cuda-11.3/targets/x86_64-linux/lib/libcudnn.so.8.2.1  
/usr/local/cuda-11.3/targets/x86_64-linux/lib/libcudnn.so.8
```

- version verification of cuDNN with below command in the terminal

```
cat /usr/local/cuda/include/cudnn_version.h | grep CUDNN_MAJOR -A 2
```

5. pytorch version verification

- link : [click here to view](#)

6. pyTorch installation

```
# ROCm 5.1.1 (Linux only)  
pip install torch==1.12.1+rocm5.1.1 torchvision==0.13.1+rocm5.1.1 torchaudio==0.12.1 --extra-index-url http:  
# CUDA 11.6  
pip install torch==1.12.1+cu116 torchvision==0.13.1+cu116 torchaudio==0.12.1 --extra-index-url https://down:  
# CUDA 11.3  
pip install torch==1.12.1+cu113 torchvision==0.13.1+cu113 torchaudio==0.12.1 --extra-index-url https://down:  
# CUDA 10.2  
pip install torch==1.12.1+cu102 torchvision==0.13.1+cu102 torchaudio==0.12.1 --extra-index-url https://down:  
# CPU only  
pip install torch==1.12.1+cpu torchvision==0.13.1+cpu torchaudio==0.12.1 --extra-index-url https://download
```

```
pip install torch==1.12.1+cu113 torchvision==0.13.1+cu113  
torchaudio==0.12.1 --extra-index-url https://download.pytorch.org/whl/cu113
```

7. pytorch version verification

- Type below commands in the terminal for installed pytorch version verification

```
python  
  
import torch  
  
print(torch.__version__)
```

```
tk032@tk032-OMEN-by-HP-Laptop-16-c0xxx:~$ python3.9
Python 3.9.16 (main, Dec 7 2022, 01:11:51)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> print(torch.__version__)
1.12.1+cu113
```

4. YOLOv5 implementation

- Type below commands in the terminal

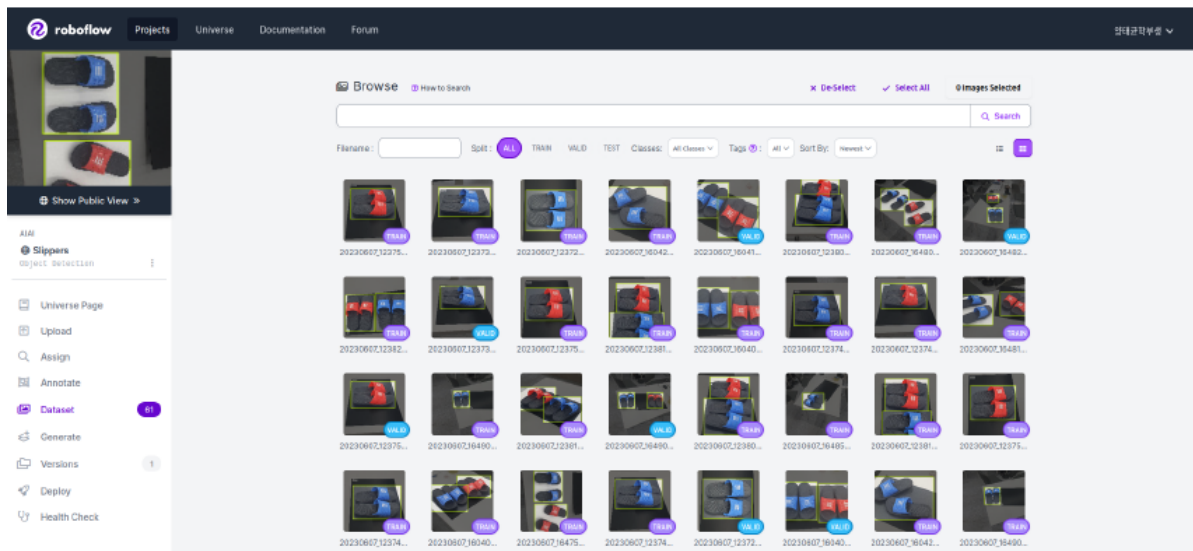
```
git clone https://github.com/ultralytics/yolov5 # clone

cd yolov5

pip install -r requirements.txt # install
```

- Training process

There are various methods for data labeling, but in this project, Roboflow was used to facilitate the process of data labeling. After the labeling, download the datasets.

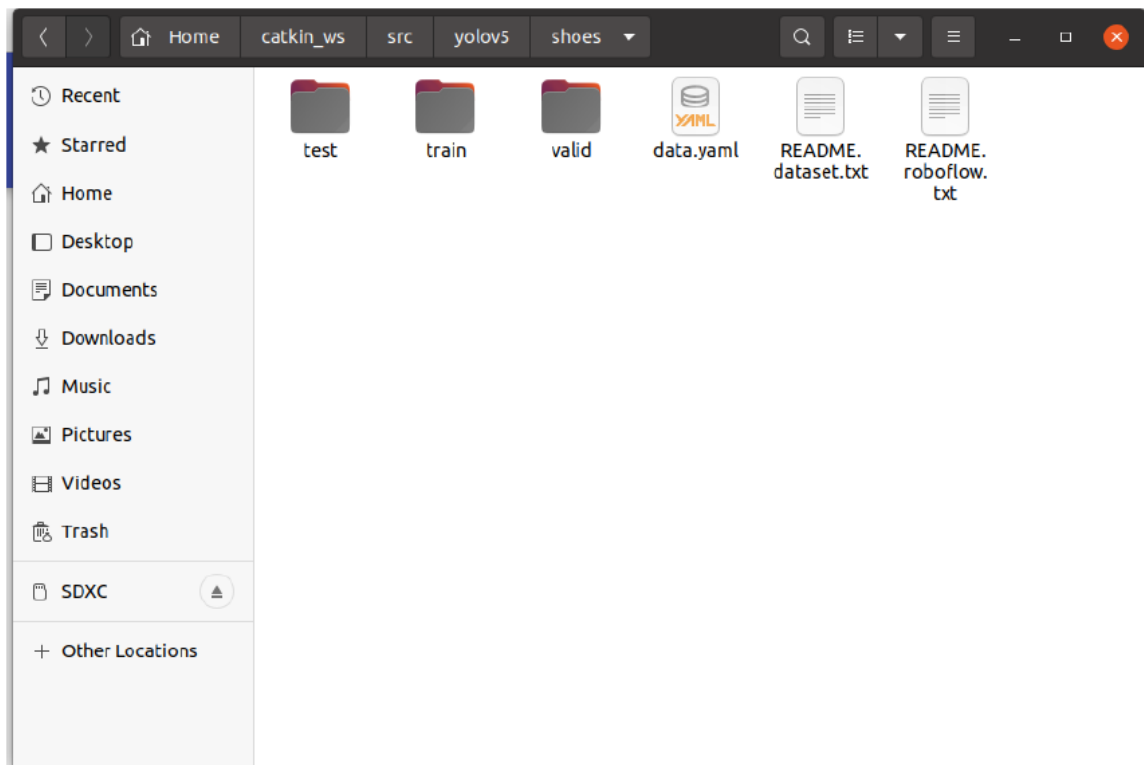


Modify the data.yaml file to match the data format you want to train. In this project, only shoe objects are detected, so set the class to 1.

```
*data.yaml
~/catkin_ws/src/yolov5/shoes

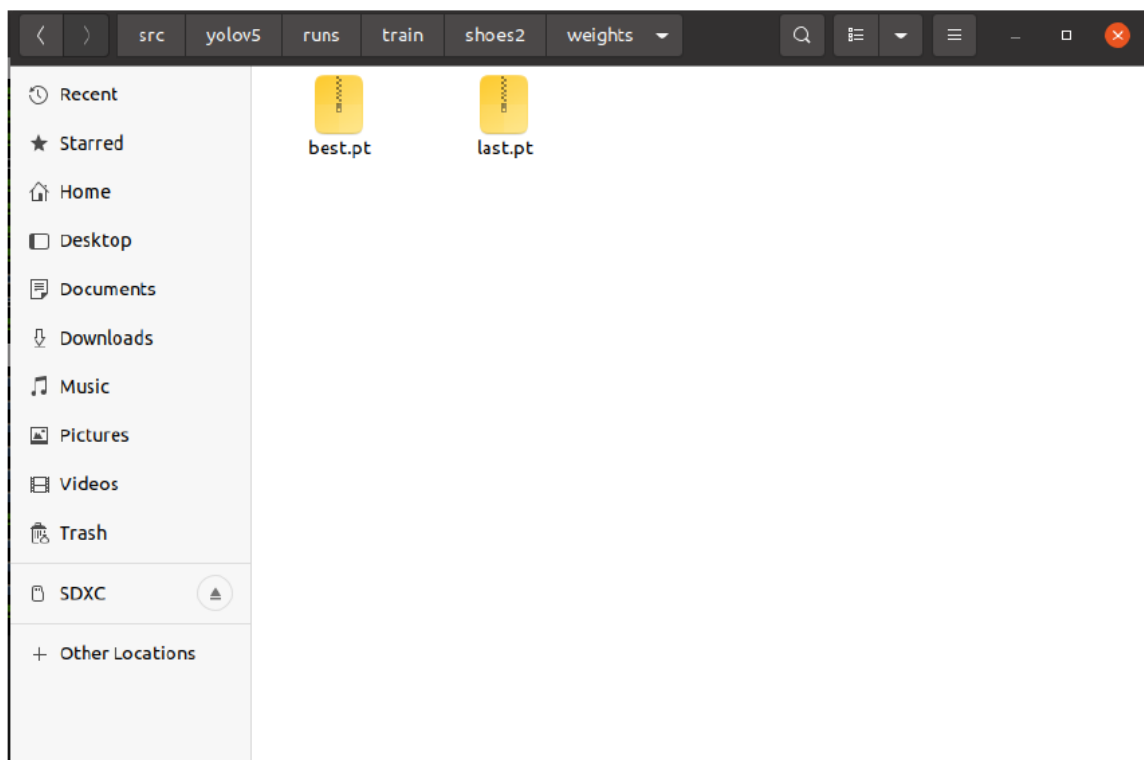
1 train: /home/tk032/catkin_ws/src/yolov5/shoes/train/images
2 val: /home/tk032/catkin_ws/src/yolov5/shoes/valid/images
3
4 nc: 1
5 names: ['SHOES']
```

train, val data location verification



```
python train.py --batch 16--epochs 10--data "자신의것\data.yaml"--weights yolov5s.pt
```

Train the model by specifying the batch size and number of epochs, matching the weights file and data.yaml file. The best.pt file in yolov5/runs/train/exp contains the trained weights.



- Reference

<https://jjeongil.tistory.com/2066>

https://ingu627.github.io/tips/install_cuda_linux/