

Native Installation of Integrated Vision System App on Linux System

Prepared by: Yap Jun Kang

Contents

1. Enabling GPU Support for the App	3
1.1. Nvidia Driver Installation	3
1.2. CUDA Toolkit Installation	4
1.3. cuDNN Installation	5
2. Installation of Required Application	6
2.1. Anaconda Installation	6
2.2. GPU Access Verification	7
2.3. Troubleshooting for GPU Access.....	8
2.4. Application Installation	9
2.5. Mosquito Installation.....	9
2.6. PostgreSQL Installation	10
3. Launching the Application	10
3.1. Application Start-up.....	10

1. Enabling GPU Support for the App

1.1. Nvidia Driver Installation

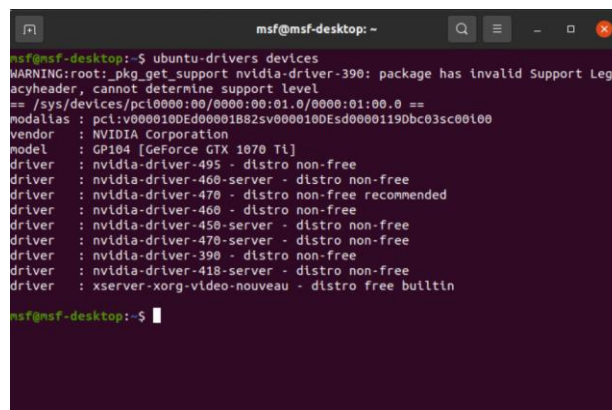
- 1) This document shows the installation step for the Integrated Vision Inspection System App installation on Ubuntu 20.04.
- 2) Before any installation, start by updating the software by using the following command.

```
$ sudo apt update
```

```
$ sudo apt upgrade
```

- 3) Next, install the Nvidia driver using the following commands.
- 4) The following command will show all the available Nvidia drivers for installation.

```
$ ubuntu-drivers devices
```



```
msf@msf-desktop: ~  
msf@msf-desktop:~$ ubuntu-drivers devices  
WARNING:root: pkg_get_support nvidia-driver-390: package has invalid Support Legend  
acyheader, cannot determine support level  
== /sys/devices/pcl0000:00/0000:00:01.0/0000:01:00.0 ==  
modalias : pci:v000010DEd00001B82sv000010DEsd00001190bc03sc00l00  
vendor    : NVIDIA Corporation  
model     : GP104 [GeForce GTX 1070 Ti]  
driver    : nvidia-driver-495 - distro non-free  
driver    : nvidia-driver-466-server - distro non-free  
driver    : nvidia-driver-470 - distro non-free recommended  
driver    : nvidia-driver-466 - distro non-free  
driver    : nvidia-driver-450-server - distro non-free  
driver    : nvidia-driver-470-server - distro non-free  
driver    : nvidia-driver-390 - distro non-free  
driver    : nvidia-driver-418-server - distro non-free  
driver    : xserver-xorg-video-nouveau - distro free builtin  
msf@msf-desktop:~$
```

- 5) If you are not sure which drivers to install, use the command for auto install

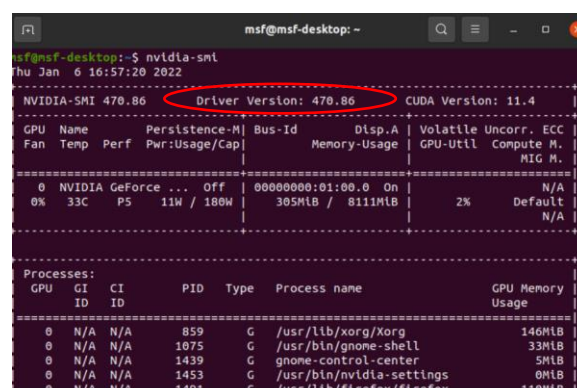
```
$ sudo ubuntu-drivers autoinstall
```

- 6) Otherwise, run the following command for the version to install by replacing the drivers shown in the previous command

WARNING: Do not install the third-party drivers unless it's the last choice as it can take very long to install and it may not even work

```
$ sudo apt install <driver version>
```

- 7) Once installation is done, reboot the PC and check the driver version using the command `$ nvidia-smi`



```
msf@msf-desktop: ~  
msf@msf-desktop:~$ nvidia-smi  
Thu Jan 6 16:57:20 2022  
+-----+  
| NVIDIA-SMI 470.86      Driver Version: 470.86      CUDA Version: 11.4      |  
+-----+-----+-----+-----+-----+-----+  
| GPU  Name            Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |  
| Fan  Temp  Perf    Pwr:Usage/Cap|  Memory-Usage | GPU-Util  Compute M. |  
|-----+-----+-----+-----+-----+-----+  
|   0   NVIDIA GeForce ... Off      | 00000000:01:00:0 | On      | N/A |  
| 0%   33C    P5     11W / 180W    | 305MiB / 811MiB | 2%      | Default |  
|-----+-----+-----+-----+-----+-----+  
| Processes:                                                       GPU Memory |  
|  GPU   GI    CI          PID    Type   Process name          Usage   |  
|-----+-----+-----+-----+-----+-----+  
|   0   N/A  N/A         859      G   /usr/lib/xorg/Xorg          146MiB |  
|   0   N/A  N/A         1075     G   /usr/bin/gnome-shell       33MiB  |  
|   0   N/A  N/A         1439     G   gnome-control-center       5MiB   |  
|   0   N/A  N/A         1453     G   /usr/bin/nvidia-settings   0MiB   |  
|   0   N/A  N/A         1491     G   /usr/lib/firefox/firefox   110MiB |  
+-----+-----+-----+-----+-----+-----+  
msf@msf-desktop:~$
```

1.2. CUDA Toolkit Installation

- 1) Before installing CUDA Toolkit, check if Nvidia Drivers have been installed.
- 2) When installing CUDA toolkit, it is important to determine whether the installation should include a driver installation.
- 3) Next, follow this [link](#) to check what is the desired CUDA toolkit and cuDNN version needed by TensorFlow.
- 4) This application uses TensorFlow 2.7 thus uses CUDA toolkit 11.2 and cuDNN 8.1.
- 5) Simply search on the web for CUDA toolkit <version number> installation for the download.
- 6) Choose the correct option and start the installation using the command shown on the website.

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 sbsa

Distribution

CentOS Debian Fedora OpenSUSE RHEL SLES Ubuntu WSL-Ubuntu

Version

20.04 18.04 16.04

Installer Type

runfile (local) deb (local) deb (network)

Download Installer for Linux Ubuntu 20.04 x86_64

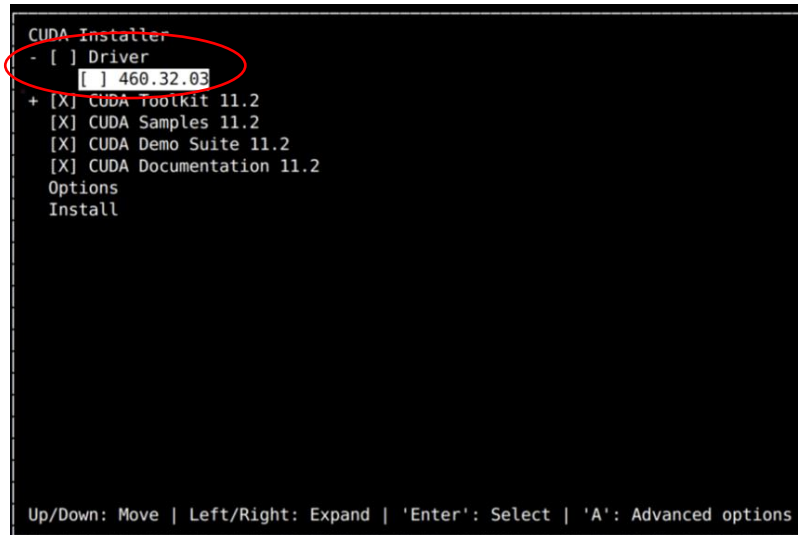
The base installer is available for download below.

Base Installer

Installation Instructions:

```
$ wget https://developer.download.nvidia.com/compute/cuda/11.2.0/local_installers/cuda_11.2.0_460.27.04_linux.run
$ sudo sh cuda_11.2.0_460.27.04_linux.run
```

- 7) It is recommended to use a [runfile](#) installation instead of a deb file, as it allows to choose what to install and not to install and most of the time the Nvidia driver that comes with this installation will not be compatible with most devices.
- 8) After running the commands, select continue and uncheck the Nvidia driver installation if one has been installed previously.



- 9) Next, go to file explorer and navigate to the Home directory and look for the bashrc file
- 10) Open the file and scroll to the bottom and paste in the following lines to add CUDA toolkit to the environment PATH.

```
export PATH=/usr/local/cuda-11.2/bin${PATH:+:${PATH}} export
LD_LIBRARY_PATH=/usr/local/cuda-11.2/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
export CUDA_HOME=/usr/local/cuda-11.2 export CUDADIR=/usr/local/cuda-11.2
```

- 11) Save the file and run the command `$ source ./bashrc`
- 12) Next, reboot the system once more.
- 13) Once rebooted, use the command `<nvcc -V>` to check if the right version of CUDA toolkit has been installed.

1.3. cuDNN Installation

- 1) Visit this [link](#) for cuDNN installation, an Nvidia account is required for this installation.
- 2) Once logged in, look for the correct version for the TensorFlow version that is being used.

Download cuDNN v8.3.1 [November 22nd, 2021], for CUDA 10.2
Download cuDNN v8.3.0 [November 3rd, 2021], for CUDA 11.5
Download cuDNN v8.3.0 [November 3rd, 2021], for CUDA 10.2
Download cuDNN v8.2.4 [September 2nd, 2021], for CUDA 11.4
Download cuDNN v8.2.4 [September 2nd, 2021], for CUDA 10.2
Download cuDNN v8.2.2 [July 6th, 2021], for CUDA 11.4
Download cuDNN v8.2.2 [July 6th, 2021], for CUDA 10.2
Download cuDNN v8.2.1 [June 7th, 2021], for CUDA 11.x
Download cuDNN v8.2.1 [June 7th, 2021], for CUDA 10.2
Download cuDNN v8.2.0 [April 23rd, 2021], for CUDA 11.x
Download cuDNN v8.2.0 [April 23rd, 2021], for CUDA 10.2
Download cuDNN v8.1.1 [February 26th, 2021], for CUDA 11.0,11.1 and 11.2
Download cuDNN v8.1.1 [February 26th, 2021], for CUDA 10.2
Download cuDNN v8.1.0 [January 26th, 2021], for CUDA 11.0,11.1 and 11.2
Download cuDNN v8.1.0 [January 26th, 2021], for CUDA 10.2

- 3) In this application, TensorFlow 2.7 is used, thus cuDNN of version 8.1 is used for CUDA Toolkit 11.2.
- 4) Choose whichever installation method that is desired, however it is recommended to install it through as followed.
- 5) Download the three files as shown below.

Library for Windows and Linux, Ubuntu(x86_64, armsbsa, PPC architecture)

cuDNN Library for Linux (aarch64sbsa)
 cuDNN Library for Linux (x86_64)
 cuDNN Library for Linux (PPC)
 cuDNN Library for Windows (x86)
 cuDNN Runtime Library for Ubuntu20.04 x86_64 (Deb)
 cuDNN Developer Library for Ubuntu20.04 x86_64 (Deb)
 cuDNN Code Samples and User Guide for Ubuntu20.04 x86_64 (Deb)
 cuDNN Runtime Library for Ubuntu20.04 aarch64sbsa (Deb)
 cuDNN Developer Library for Ubuntu20.04 aarch64sbsa (Deb)
 cuDNN Code Samples and User Guide for Ubuntu20.04 aarch64sbsa (Deb)
 cuDNN Cross-compile Library for Ubuntu20.04 aarch64sbsa (Deb)
 cuDNN Developer Cross-compile Library for Ubuntu20.04 aarch64sbsa (Deb)
 cuDNN Runtime Library for Ubuntu18.04 x86_64 (Deb)
 cuDNN Developer Library for Ubuntu18.04 x86_64 (Deb)
 cuDNN Code Samples and User Guide for Ubuntu18.04 x86_64 (Deb)
 cuDNN Runtime Library for Ubuntu16.04 x86_64 (Deb)
 cuDNN Developer Library for Ubuntu16.04 x86_64 (Deb)
 cuDNN Code Samples and User Guide for Ubuntu16.04 x86_64 (Deb)

- 6) Once downloaded, run the following command for cuDNN installation.

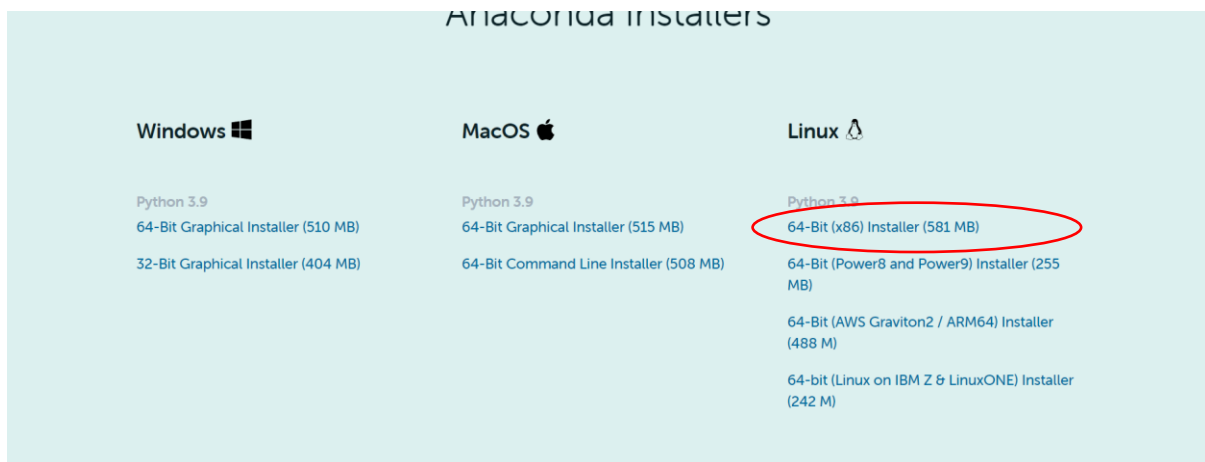
```
$ sudo dpkg -i libcudnn8_8.1.0.77-1+cuda11.2_amd64.deb
$ sudo dpkg -i libcudnn8-dev_8.1.0.77-1+cuda11.2_amd64.deb
$ sudo dpkg -i libcudnn8-samples_8.1.0.77-1+cuda11.2_amd64.deb
```

- 7) Remember to check the file name for the above command so that the command can be run without errors.
- 8) Finally, reboot the system once more.

2. Installation of Required Application

2.1. Anaconda Installation

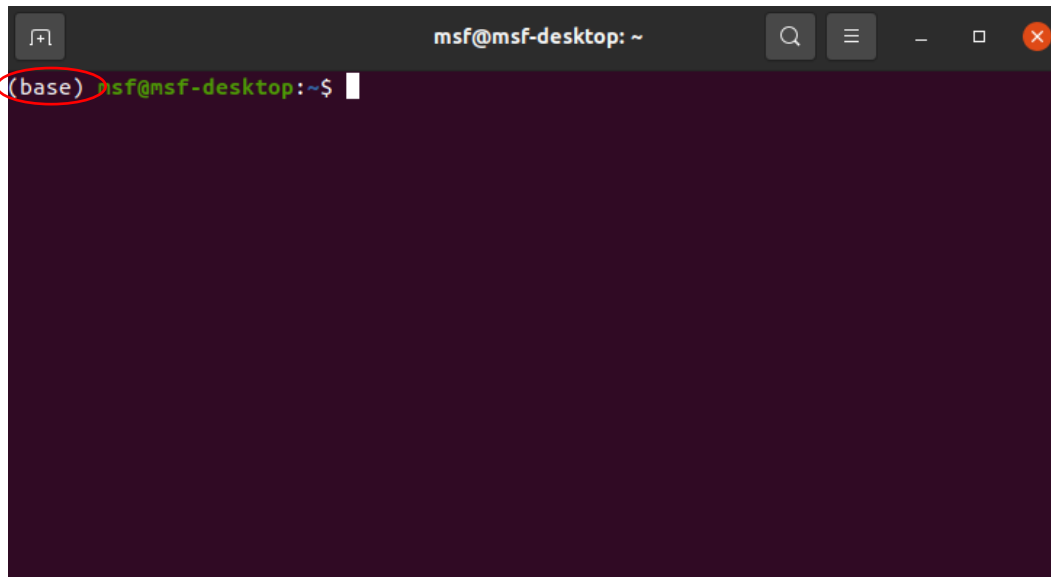
- 1) To install Anaconda, visit this link and choose Linux system and install the file as shown in the following image.



- 2) Once the file has been downloaded, follow the instructions in this [link](#) for a more detailed instruction.
- 3) Run the following commands for Anaconda installation.

```
$ bash ~/Downloads/Anaconda3-2021.11-Linux-x86_64.sh
```

- 4) Remember to check the file name for the above command just as mentioned in the cuDNN installation.
- 5) Once the installation is done. Use the command `$ source ./bashrc` to access Anaconda.



2.2. GPU Access Verification

- 1) Before proceeding to other application installation, it is important to verify whether TensorFlow will be able to access GPU.
- 2) First and foremost, run the following command for some prerequisite files for the app installation.

```
$ sudo apt install -y git make libpq-dev protobuf-compiler psmisc python3-icu libpq-dev ffmpeg libsm6 libxext6 libxrender-dev
```

- 3) Next, pull the application files from GitHub using the following command.

```
$ git clone https://github.com/msf4-0/Integrated-Vision-Inspection-System.git
```

```
$ cd Integrated-Vision-Inspection-System
```

- 4) Next, run the following command to create a virtual environment named <cvsystem> in Anaconda.

```
$ conda create --name cvsystem python=3.8
```

```
$ conda activate cvsystem
```

```
msf@msf-desktop: ~
(base) msf@msf-desktop:~$ conda activate cvsystem
(cvsystem) msf@msf-desktop:~$
```

- 5) Once the virtual environment is created and activated, run the following command to check if TensorFlow has GPU access.

```
$ python -c "import tensorflow as tf; print(\"Num GPUs Available: \", len(tf.config.list_physical_devices('GPU')))"
```

- 6) If it returns GPU devices = 1, skip the next section. and proceed with the installation, if GPU devices = 0, proceed to the next step to troubleshoot the installation of Nvidia driver, CUDA Toolkit and cuDNN.

```
msf@msf-desktop: ~
(base) msf@msf-desktop:~$ conda activate cvsystem
(cvsystem) msf@msf-desktop:~$ python -c "import tensorflow as tf; print(\"Num GPUs Available: \", len(tf.config.list_physical_devices('GPU')))"
2022-02-03 08:56:05.834370: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:939] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-02-03 08:56:05.854358: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:939] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-02-03 08:56:05.854666: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:939] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
Num GPUs Available: 1
(cvsystem) msf@msf-desktop:~$
```

2.3. Troubleshooting for GPU Access

- 1) To troubleshoot, run the command `$ nvidia-smi` and `$ nvcc -V`.
- 2) The outputs should be similar as bellow, be sure to the check the versions for CUDA Toolkit as well.


```
msf@msf-desktop: ~  
(base) msf@msf-desktop:~$ nvcc -V  
nvcc: NVIDIA (R) Cuda compiler driver  
Copyright (c) 2005-2020 NVIDIA Corporation  
Built on Mon_Nov_30_19:08:53_PST_2020  
Cuda compilation tools, release 11.2, V11.2.67  
Build cuda_11.2.r11.2/compiler.29373293_0  
(base) msf@msf-desktop:~$
```

- 3) If the installation of Nvidia drivers and CUDA Toolkit is fine as indicated by the above command. Be sure the proper PATH has been added during the installation of CUDA Toolkit as well.
- 4) If the above has been done, check if cuDNN installation is done properly.
- 5) However, in the event where it is not clear which issue it is, the installation can be reset by running the following commands.

```
$ sudo apt-get --purge remove "*cuda*" "*cublas*" "*cufft*" "*cufile*" "*curand*" \ "*cusolver*" "*cusparse*" "*gds-tools*" "*npp*" "*nvjpeg*" "nsight*"  
$ sudo apt-get --purge remove "*nvidia*"  
$ sudo apt-get autoremove
```

- 6) The following command is to purge all Nvidia driver files, CUDA files as well as cuDNN files and restart the required installation for the GPU support mentioned at the start of the document.

2.4. Application Installation

- 1) To install the required files for the application, be sure to keep the virtual environment created in Anaconda active first.
- 2) Next, run the following command to install the required files.

```
$ pip install -U pip  
$ pip install -r requirements_no_hash.txt  
$ python src/lib/machine_learning/tfod_installation.py
```

2.5. Mosquito Installation

- 1) To install mosquito broker, follow the following link for more detailed installation step.
- 2) To install mosquito, open terminal and run the following command.

```
$ sudo apt-add-repository ppa:mosquitto-dev/mosquitto-ppa  
$ sudo apt-add-repository ppa:mosquitto-dev/mosquitto-ppa
```

```
$ sudo apt-get update
$ sudo apt-get install mosquitto
$ sudo apt-get install mosquitto-clients
$ sudo apt clean
```

2.6. PostgreSQL Installation

- 1) To install PostgreSQL, visit this [link](#) for a more detailed step of installing and setting up PostgreSQL.
- 2) Run the following command for PostgreSQL, mind the version of PostgreSQL in the command.

```
$ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
$ wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
$ sudo apt-get update
$ sudo apt-get -y install postgresql
```

- 3) Once the installation is done, setup PostgreSQL by giving it a password so that the database can be access by the app.
- 4) First sign in as root in the terminal. `$ sudo su`
- 5) Use the following commands to set a new password for PostgreSQL. This password will be needed when first sign in into the app.

```
$ sudo passwd postgres
$ psql -c "ALTER USER postgres WITH PASSWORD 'secure_password_here';"
$ exit
```

- 6) Next, use the following command to reset PostgreSQL for the new password to take effect.

```
$ sudo systemctl restart postgresql
```

- 7) Run the next command and enter the password to verify that PostgreSQL has been setup properly.

```
$ sudo su - postgres
$ psql
```

3. Launching the Application

3.1. Application Start-up

- 1) To start up the app, change the directory using the following command.

```
$ cd Integrated-Vision-Inspection-System
```

- 2) Next, activate the virtual environment created previously.

```
$ conda activate cvsystem
```

- 3) To start the app, use the command `$ streamlit run src/app.py`, the app will be launch in the browser.
- 4) Once start up, a username and password will be prompted. The default username and password will be 'admin'
- 5) Once logged in, scroll to the bottom of the page and enter the PostgreSQL password set previously.
- 6) If successful, the application is ready to be used.