Installation of GCC, Drivers, CUDA, CUDNN and TensorFlow GPU

Table of Contents

1. Installation of GCC Compiler

GCC is usually pre-installed on many Linux distributions. If it's not available or you need to update it, you can install it using your package manager.

Ubuntu/Debian:

```
sudo apt update
sudo apt install build-essential
```

2. Installation of GPU Drivers

> 💡 If You already have any driver install then which was creating problems , you can uninstall it and download another and can set up.

▼ 2.1 Removing existing drivers

To remove the current NVIDIA drivers, follow these steps:

1. **List Installed NVIDIA Packages**:

```
dpkg -l | grep nvidia
```

2. **Remove NVIDIA Packages**:

   Use the following command to remove them:

```
sudo apt-get --purge remove '*nvidia*'
```

3. **Check for Remaining NVIDIA Packages**:

```
bashCopy code
dpkg -l | grep nvidia
```

   Run:

   If there are still NVIDIA packages, remove them individually using `sudo apt-get --purge remove <package-name>`.

   4. Remove Additional NVIDIA Files:

   Remove any residual files or directories:

```
bashCopy code
sudo rm /etc/X11/xorg.conf
sudo rm -rf /usr/lib/nvidia*
sudo rm -rf /usr/local/cuda*
```

▼ 2.2 Installation of Nvidia Driver

1. **Install NVIDIA Driver 510**:

   You can search for the available driver versions:

```
apt search nvidia-driver
```

   Install the desired version, in this case, version 470:

```
sudo apt install nvidia-driver-470
```

2. **Reboot Your System**:

Restart your computer to apply the changes:

```
sudo reboot
```

4. Verify NVIDIA Driver Installation

After rebooting, verify that the NVIDIA driver is correctly installed:

1. **Check Driver Version**:

   ```
   nvidia-smi
   ```

   This should show you information about your GPU and the driver version. It should display version 470. But you can set up your desired GPU version like 510 GPU .

   Reboot Your System:

   Restart your computer to apply the changes:

   ```
   bashCopy code
   sudo reboot
   ```

3. CUDA and CUDNN Setup

> 💡 Before starting this setup, if you have already installed CUDA or CUdnn which is conflicting then it is better practice that to uninstall them and install again

▼ 3.1 Removing existing CUDA and CUDnn

`

1. Uninstall Existing CUDA Toolkit

To remove the installed CUDA Toolkit, follow these steps:

1. **List Installed CUDA Packages**:

   ```
   dpkg -l | grep cuda
   ```

   This will list all the installed CUDA packages.

2. **Remove CUDA Packages**:

   Remove each CUDA package by running:

   ```
   sudo apt-get --purge remove cuda-*
   ```

   You might need to remove any other residual packages related to CUDA:

   ```
   sudo apt-get --purge remove libcudnn*
   sudo apt-get --purge remove libnvinfer*
   ```

```
sudo apt-get --purge remove libnvinfer-plugin*
```

3. **Remove CUDA Configuration Files**:

   Remove any configuration files or directories that may have been created:

```
sudo rm -rf /usr/local/cuda*
```

Reboot Your System:

Restart your computer to apply the changes:

```
sudo reboot
```

Now start setup for CUDA and CUDNN.

1. Install Dependencies & pre-requisites

```
sudo apt install build-essential
sudo apt install gcc tar make git
sudo apt-get install freeglut3-dev build-essential libx11-dev libxmu-dev l
ibxi-dev libgl1-mesa-glx libglu1-mesa libglu1-mesa-dev
```

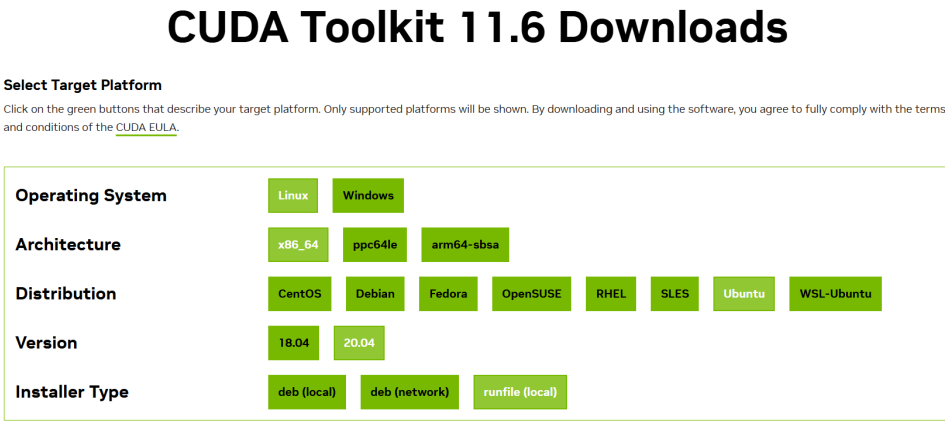
If cudnn.h file not found during testing cuda

```
sudo apt-get install libcudnn8-dev
```

If FreeImage is not set up correctly during testing cuda

```
sudo apt-get install libfreeimage3 libfreeimage-dev
```

> Download CUDA 11.6: CUDA Toolkit 11.6 Downloads | NVIDIA Developer



1. Install CUDA_toolkit_10.0 // Download the exact cuda version that you want to install, we are doing cuda 10.0 // allow read and write permission

```
wget https://developer.download.nvidia.com/compute/cuda/11.6.0/local_installers/cuda_11.6.0_510.39.01_linux.run
```

```
sudo sh cuda_11.6.0_510.39.01_linux.run
```

> when try to run this then uncheck all excluding cuda toolkit option. to check and uncheck use Enter.

Verify CUDA Toolkit Installation

You mentioned that the CUDA version shown is 12.2. Let's verify this installation and check its details.

Verify CUDA Toolkit

Run the following command to check the CUDA Toolkit version installed:

```
nvcc --version
```

You should see an output similar to:

```
vbnetCopy code
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2020 NVIDIA Corporation
Built on Wed_Jul_22_19:09:48_PDT_2020
Cuda compilation tools, release 12.2, V12.2.x
```

Check Environment Variables

Ensure that the environment variables for CUDA are correctly set. Open your `~/.bashrc` file:

```
bashCopy code
nano ~/.bashrc
```

Add the following lines if they are not already present, replacing `11.6` with your CUDA version if different:

```
export PATH=/usr/local/cuda-12.2/bin${PATH:+:${PATH}}
export LD_LIBRARY_PATH=/usr/local/cuda-12.2/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

Save(ctrl+o) and close(ctrl+x) the file, then reload it:

```
source ~/.bashrc
```

▼ 3.2 Download and Install CUDnn

first download Cudnn 8.9.7 https://developer.nvidia.com/rdp/cudnn-archive#a-collapse897-118

then download .deb package.

- First, change to the `Downloads` directory where your cuDNN `.deb` file is located. You're already in the correct directory based on your terminal output.

  ```
  cd ~/Downloads
  ```

- **Install the cuDNN Package**:

  Use `dpkg` to install the cuDNN package. Make sure to provide the correct file name including the version number you downloaded.

  ```
  sudo dpkg -i cudnn-local-repo-ubuntu2004-8.9.7.29_1.0-1_amd64.deb
  ```

  Replace `cudnn-local-repo-ubuntu2004-8.9.7.29_1.0-1_amd64.deb` with the exact name of the cuDNN `.deb` file you downloaded.

  Verify cuDNN Installation:

  After installation, you can verify if cuDNN is correctly installed by checking the installed version:

  ```
  dpkg -l | grep cudnn
  ```

Check cuda compiler

```
nvcc --version
```

Steps to Use cuDNN Samples with CUDA 11.6 to ensure its working or not

Test of Installation:

> Download: Install-Cuda-on-Ubuntu/cudnn_samples_v8-master.zip at main · muzahidai/Install-Cuda-on-Ubuntu · GitHub

1. Extract cuDNN Samples

1. **Navigate to the Downloads Directory**:

- Open a terminal and change to the `Downloads` directory where your cuDNN samples (`cudnn_samples_v8-master.zip`) are located.

```
cd ~/Downloads
```

2. **Extract the cuDNN Samples**:
   - Use the `unzip` command to extract the `cudnn_samples_v8-master.zip` file.

```
bashCopy code
unzip cudnn_samples_v8-master.zip
```

This command will extract the contents of `cudnn_samples_v8-master.zip` into a directory named `cudnn_samples_v8-master/` in your current location (`~/Downloads`).

2. Move the Samples to CUDA Directory

1. **Copy the Samples to CUDA Directory**:
   - Assuming CUDA is installed in `/usr/local/cuda-11.6/`, copy the extracted cuDNN samples directory there.

```
sudo cp -r cudnn_samples_v8-master/ /usr/local/cuda-11.6/
```

Replace `/usr/local/cuda-11.6/` with your actual CUDA installation directory if different.

3. Compile and Run mnistCUDNN Sample

1. **Navigate to mnistCUDNN Directory**:
   - Change directory to the mnistCUDNN sample within the cuDNN samples directory.

```
cd /usr/local/cuda-11.6/cudnn_samples_v8-master/mnistCUDNN/
```

2. **Compile and Run mnistCUDNN Sample**:
   - Clean previous builds, compile the sample, and run it to perform the digit classification test.

```
make clean
make
./mnistCUDNN
```

```
Output:
Test passed!
```

If cudnn.h file not found during make or performing classification test

```
sudo apt-get install libcudnn8-dev
```

If FreeImage is not set up correctly during performing classification test

```
sudo apt-get install libfreeimage3 libfreeimage-dev
```

Now perform classification again.

4. Anaconda, TensorFlow Installation

First Install Anaconda3-2023.03-1-Linux-x86_64.sh version then install tensorflow GPU.

Index of /anaconda/archive/ | 清华大学开源软件镜像站 | Tsinghua Open Source Mirror

Index of /anaconda/archive/ | 清华大学开源软件镜像站，致力于为国内和校内用户提供高质量的开源软件镜像、Linux 镜像源服务，帮助用户更方便地获取开源软件。本镜像站由清华大学 TUNA 协会负责运行维护。

https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/

💡 It is better practice to make a directory for every separate projects and environments

```
mkdir test
```

1. Create a folder first
2. Check directory is created successfully

   ```
   ls
   ```

3. Go to the directory

   ```
   cd test
   ```

4. Installing TensorFlow with GPU Support
   - Create a new Conda environment (optional but recommended):

     ```
     bashCopy code
     conda create -n dl python=3.8
     conda activate dl
     ```

     - Install TensorFlow GPU:

       ```
       pip install tensorflow-gpu==2.4.1
       ```

     Or,

if you are not sure which Tensor flow version is compatible with your python version then install-

For example, to install the latest TensorFlow GPU version compatible with Python 3.8:

```
conda install tensorflow-gpu
```

This command will install the latest available version of TensorFlow GPU that is compatible with your Python 3.8 environment.

Verifying Installation

- Check TensorFlow version:

```
python -c "import tensorflow as tf; print(tf.__version__)"
```

  - Verify GPU availability:

```
python
>>> import tensorflow as tf
>>> print("GPU is", "available" if tf.config.list_physical_devic
es('GPU'
```

5. Installing Jupyter Notebook

```
conda install jupyter notebook
```

6. Starting jupyter notebook

```
jupyter notebook
```

7. Test Code in to check jupyter notebook

```
import sys
import tensorflow.keras
import pandas as pd
import sklearn as sk
import tensorflow as tf

print(f"Tensor Flow Version: {tf.version}")
print(f"Keras Version: {tensorflow.keras.version}")
print()
print(f"Python {sys.version}")
print(f"Pandas {pd.version}")
print(f"Scikit-Learn {sk.version}")
gpu = len(tf.config.list_physical_devices('GPU'))>0
print("GPU is", "available" if gpu else "NOT AVAILABLE")
```