Machine Learning

PyTorch Installations

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / MSc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Virtual Environment

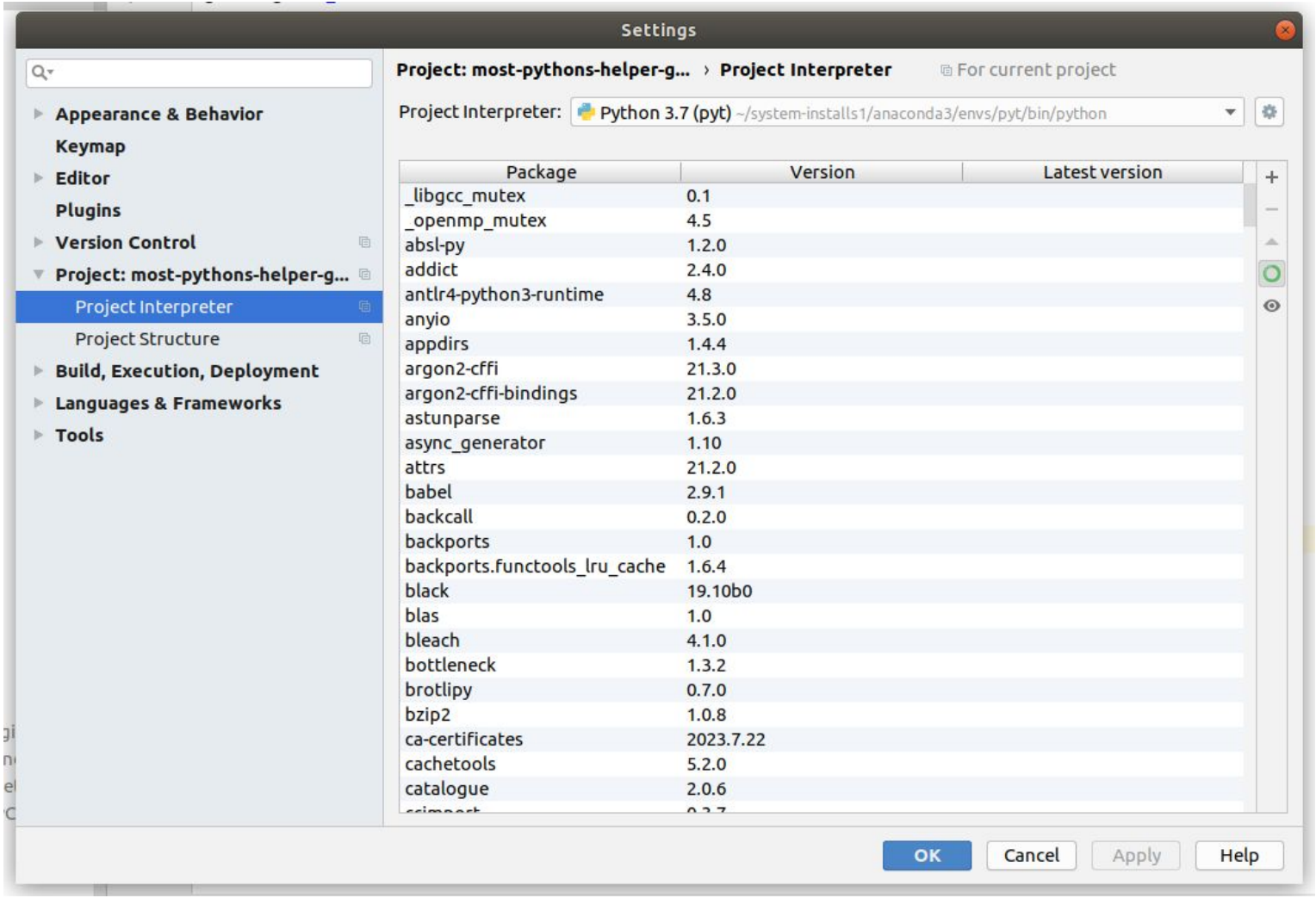
- A virtual environment is an isolated workspace in which you can install Python packages without affecting your global system setup.
  - Different projects may require different dependencies with different versions
  - If you installed everything on the global system, you will suffer!
- Examples
  - Python's Built-in **venv**
  - **virtualenv** is a third-party package that provides more features than venv.
  - **conda** environments not only manage Python packages + other languages.
    - Developed by Anaconda Inc
    - Cross-platform, but more commonly used in Linux and macOS.
    - Comes pre-installed with the **Anaconda and Miniconda** distributions
    - Follow installations steps for your OS

Conda Example

- conda **create** --name pyt
  - This will create a conda environment: find at: <>/anaconda3/envs
- **Activating** the Virtual Environment
  - conda activate pyt
- **Deactivating** the Virtual Environment
  - https://chat.openai.com/c/f26d07af-ce77-498f-99cc-09ffdee131b8conda%20deactivate

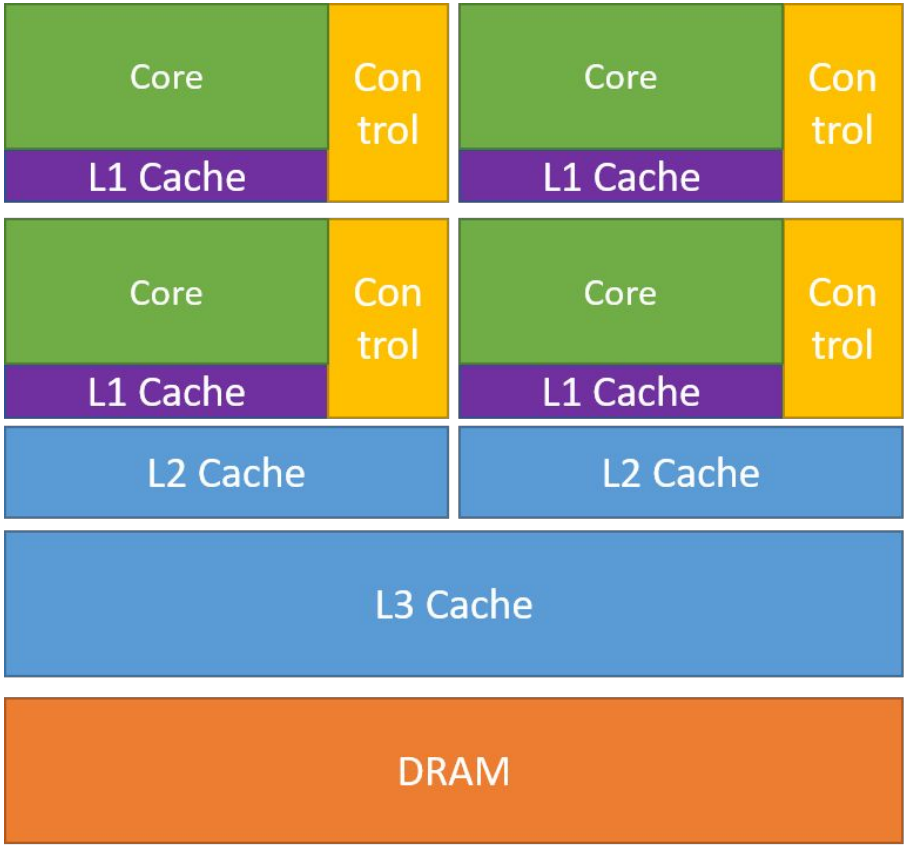
Conda with PyCharm

- IDEs are integrated with conda
  - File (menu) ⇒ settings ⇒ Project ⇒ Interpreter
  - Find the path of python executable in the acandonda
- Let’s demo

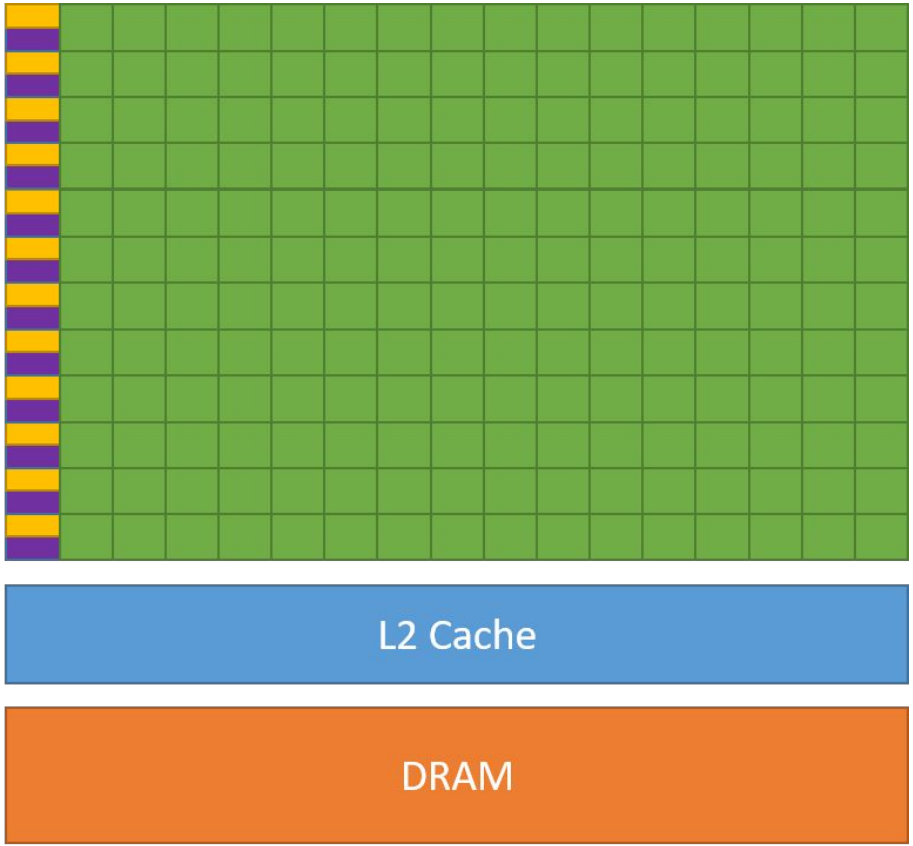

Settings
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Project: most-pythons-helper-g...
Project Interpreter
Project Structure
Build, Execution, Deployment
Languages & Frameworks
Tools
Project: most-pythons-helper-g... › Project Interpreter
For current project
Project Interpreter: Python 3.7 (pyt) ~/system-installs1/anaconda3/envs/pyt/bin/python
Package
Version
Latest version
_libgcc_mutex 0.1
_openmp_mutex 4.5
absl-py 1.2.0
addict 2.4.0
antlr4-python3-runtime 4.8
anyio 3.5.0
appdirs 1.4.4
argon2-cffi 21.3.0
argon2-cffi-bindings 21.2.0
astunparse 1.6.3
async_generator 1.10
attrs 21.2.0
babel 2.9.1
backcall 0.2.0
backports 1.0
backports.functools_lru_cache 1.6.4
black 19.10b0
blas 1.0
bleach 4.1.0
bottleneck 1.3.2
brotlipy 0.7.0
bzip2 1.0.8
ca-certificates 2023.7.22
cachetools 5.2.0
catalogue 2.0.6
OK
Cancel
Apply
Help

CPU vs GPU

- CPUs (Central Processing Unit) are jacks-of-all-trades, optimized for **single-threaded** performance and general-purpose / **sequential** tasks.
- GPUs (Graphics Processing Units) are specialized for highly **parallel** tasks like graphics **rendering** and matrix **operations**
  - Simpler memory hierarchy, optimized for high-throughput data access.
  - Specialized programming models (e.g., **CUDA**, OpenCL) required for computation.
  - Designed such that **more transistors** are devoted to data processing rather than data **caching** and **flow** control



CPU



GPU

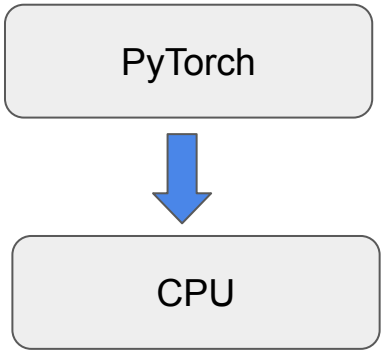
Img [src]

Check GPU (terminal)

- First, Check for NVIDIA GPU
- Use lspci
  - lspci is a command that prints detailed information about all PCI buses and devices
- As we want only to check nvidia, use grep command
  - Very useful one
  - short for “global regular expression print”, is a command used in searching and matching text files contained in the regular expressions.
- **lspci | grep -i nvidia**
  - Match with pattern (nvidia). -i means ignore case (lower/uppercase)
  - 01:00.0 VGA compatible controller: NVIDIA Corporation GP106M [**GeForce GTX 1060 Mobile**] (rev a1)
  - 01:00.1 Audio device: NVIDIA Corporation **GP106** High Definition Audio Controller (rev a1)

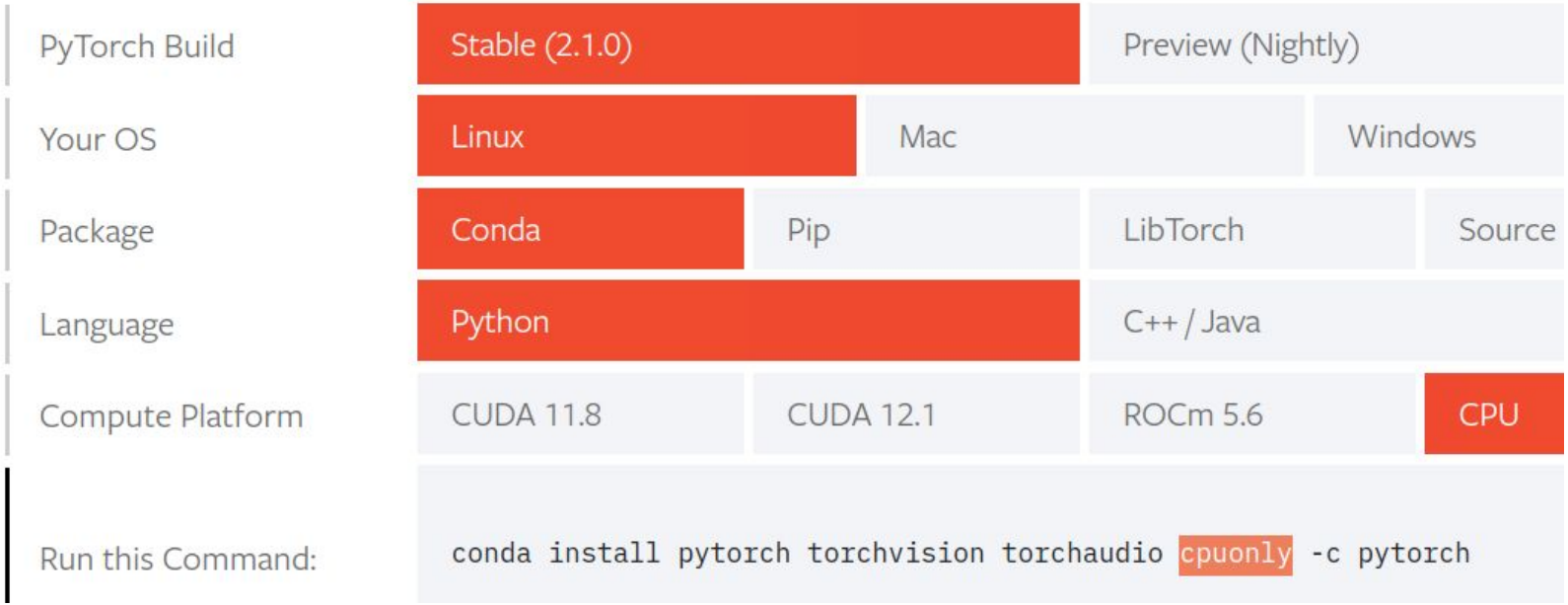
PyTorch Installations

- In practice, we always install frameworks to work with GPUs
  - However, this comes with several challenges
- An easier initial start is just to install to work with **CPUs**
  - You can even develop in this mode and launch later code on some GPU cloud



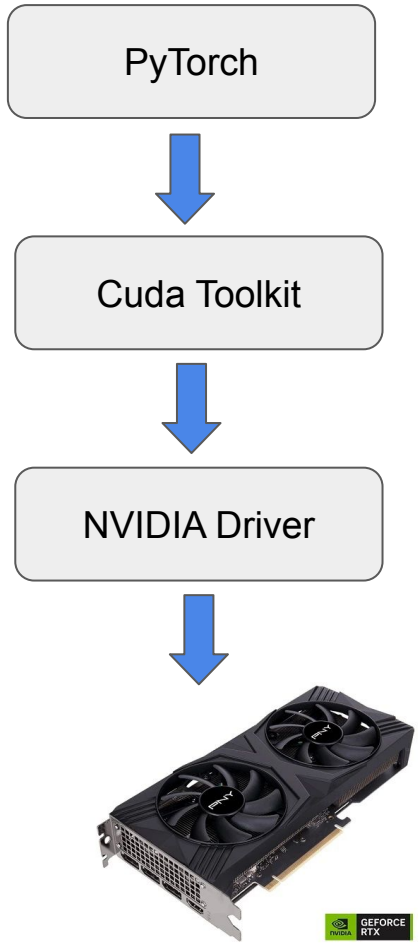
Pytorch CPU

- Go to: https://pytorch.org/get-started/locally/
  - Make different selections, including conda and cpu
  - E.g. inside your conda run: **conda install** pytorch torchvision torchaudio cpuonly -c pytorch



PyTorch Installations on GPU (Ubuntu)

- 3 steps in order
- 1) NVIDIA Driver
  - Installed on the OPERATING SYSTEM
- 2) Cuda toolkit
  - Installed on
    - Either install on OS locally and your framework use it
    - With your framework directly (in step 3) so no step 2
  - Must match a **minimum version** of the **driver**.
- 3) PyTorch
  - Installed on your conda / pip



- This page is updated with CUDA Toolkit and **Corresponding** Driver Versions

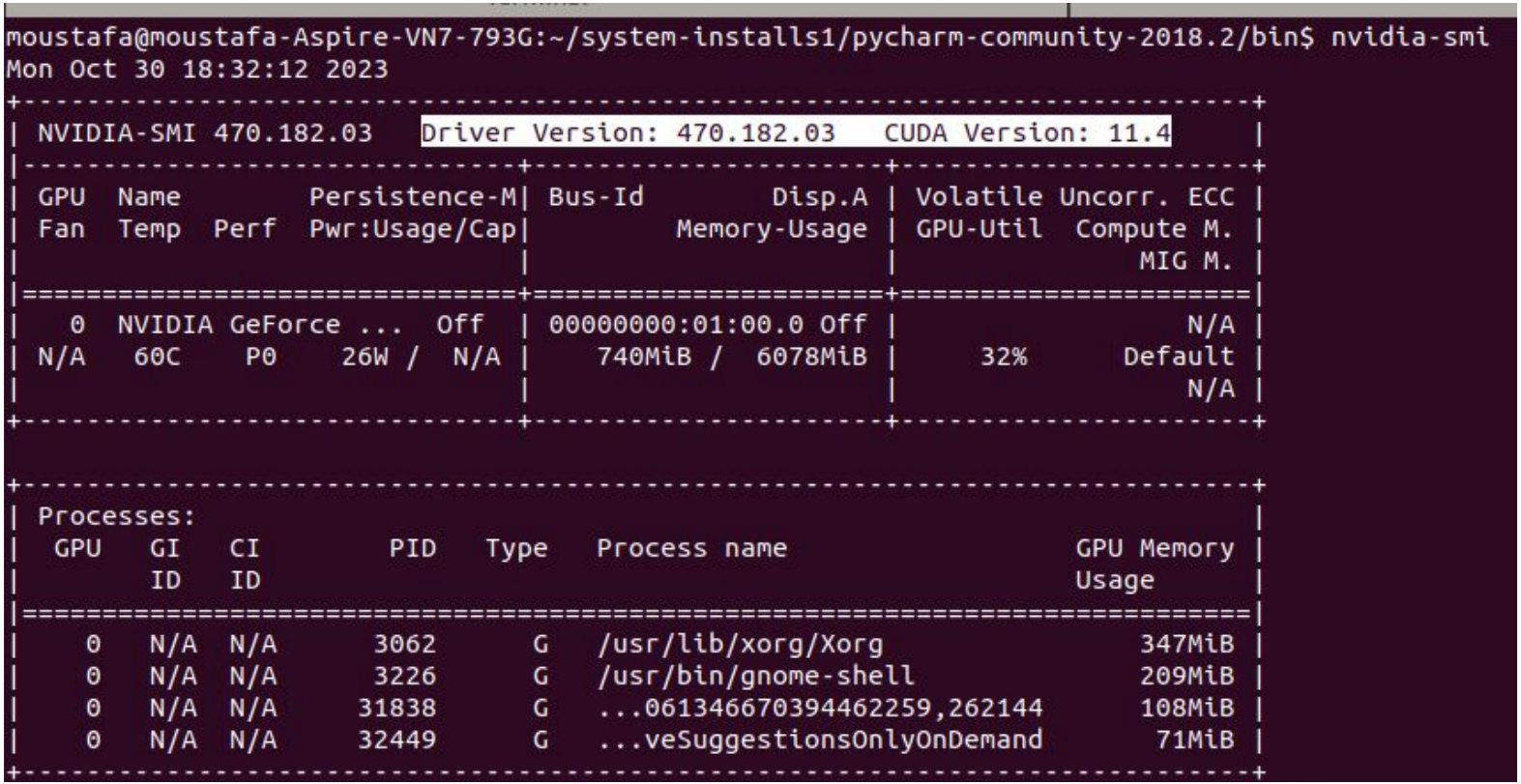
CUDA **Toolkit**	Linux x86_64 **Driver** Version	Windows x86_64 Driver Version
CUDA 12.3	>=545.23.06 | **ubuntu 20**+	>=545.84
CUDA 12.0	>=525.60.13 | ubuntu 18+	>=527.41
CUDA 11.7	>=515.48.07 | ubuntu 18+	>=516.31
CUDA 11.6	>=510.47.03 | ubuntu 18+	>=511.65
CUDA 11.4	>=470.82.01 | ubuntu 18+	>=472.50
CUDA 11.3	>=**465.19.01** | ubuntu 18+	>=465.89
CUDA 10.1	>= 418.39 | ubuntu 18+	>= 418.96
CUDA 9.0	>= 384.81 | **ubuntu 16+**	>= 385.54
CUDA 8.0	>= 367.48	>= 369.30

February 02, 2023: Deprecation of CUDA 11.6 and Python 3.7 Support

NVIDIA driver

- A software package that enables the operating system to communicate with NVIDIA GPUs. It includes
  - **CUDA Driver**: The interface between CUDA applications and the GPU
  - **Graphics Driver** for rendering 2D and 3D graphics.
  - **Display Driver**: Includes Control Panel GUI for configuring GPU settings / rendering
  - **nvidia-smi** tool: command-line utility to manage NVIDIA GPUs. Details
    - nvidia-smi
    - **watch** nvidia-smi (keep refreshing - very useful)
    - nvidia-smi -a (comprehensive details)
  - SDKs like **cuDNN**

nvidia-smi


moustafa@moustafa-Aspire-VN7-793G:~/system-installs1/pycharm-community-2018.2/bin$ nvidia-smi
Mon Oct 30 18:32:12 2023
NVIDIA-SMI 470.182.03 Driver Version: 470.182.03 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.
MIG M.
0 NVIDIA GeForce ... Off 00000000:01:00.0 Off N/A
N/A 60C P0 26W / N/A 740MiB / 6078MiB 32% Default
N/A
Processes:
GPU GI CI PID Type Process name GPU Memory
ID ID Usage
0 N/A N/A 3062 G /usr/lib/xorg/Xorg 347MiB
0 N/A N/A 3226 G /usr/bin/gnome-shell 209MiB
0 N/A N/A 31838 G ...061346670394462259,262144 108MiB
0 N/A N/A 32449 G ...veSuggestionsOnlyOnDemand 71MiB

Installing NVIDIA Driver: GUI

- From the GUI, go to your Ubuntu Software and Update
  - Click Additional Drivers
  - Select one and install
  - Restart after installation
- The higher the driver version, the higher cuda toolkit and PyTorch version
  - However, many of these listed version may fail to be installed
    - Due to your OS / Hardware / Driver code itself

Software & Updates

Ubuntu Software | Other Software | Updates | Authentication | Additional Drivers | Developer Options | Ubuntu Pro

NVIDIA Corporation: GP106M [GeForce GTX 1060 Mobile]

This device is using the recommended driver.

- (•) Using NVIDIA driver metapackage from nvidia-driver-470 (proprietary, tested)
- ( ) Using NVIDIA driver metapackage from nvidia-driver-530 (proprietary)
- ( ) Using NVIDIA driver metapackage from nvidia-driver-510 (proprietary)
- ( ) Using NVIDIA driver metapackage from nvidia-driver-390 (proprietary)
- ( ) Using NVIDIA Server Driver metapackage from nvidia-driver-470-server (proprietary)
- ( ) Using NVIDIA driver metapackage from nvidia-driver-525 (proprietary)
- ( ) Using NVIDIA Server Driver metapackage from nvidia-driver-515-server (proprietary)
- ( ) Using NVIDIA Server Driver metapackage from nvidia-driver-450-server (proprietary)
- ( ) Using NVIDIA Server Driver metapackage from nvidia-driver-418-server (proprietary)
- ( ) Using NVIDIA driver metapackage from nvidia-driver-515 (proprietary)
- ( ) Using NVIDIA Server Driver metapackage from nvidia-driver-525-server (proprietary)
- ( ) Using X.Org X server -- Nouveau display driver from xserver-xorg-video-nouveau (open source)

1 proprietary driver in use.

Revert | Apply Changes

A proprietary driver has private code that Ubuntu developers can't review or improve. Security and other updates are dependent on the driver vendor.

Close

Installing NVIDIA Driver: Terminal

- To check drivers for your card
  - sudo ubuntu-drivers list
    - driver : nvidia-driver-470-server - distro non-free
    - driver : nvidia-driver-515-server - distro non-free
    - driver : nvidia-driver-470 - distro non-free **recommended**
    - driver : nvidia-driver-525-server - distro non-free
- sudo apt update
- sudo apt install nvidia-**<driver number>**
  - sudo apt install nvidia-470
- Or sudo ubuntu-drivers install
  - will install the driver that is considered the **best match** for your hardware
- Restart after installation

Installing NVIDIA Driver: Troubleshooting

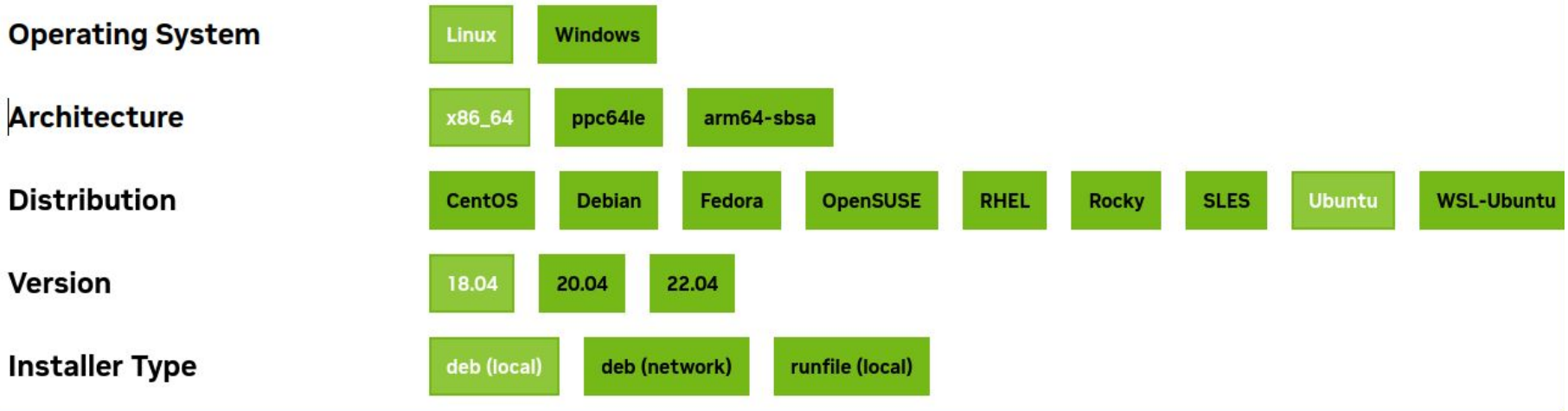
- Installing the driver can easily go wrong after restarting
  - Login page doesn't appear
  - Login loop problem: login returns you to login
- Don't rush to OS reinstallation: Google the error and get help for commands to run
- To run commands, you may Switch to Console Mode
  - Press **Ctrl + Alt + F1** to switch from the graphical interface to a console terminal.
    - You can also use F2, F3, etc., to switch to different terminals.
  - **Login** (numpad may not work)
  - **Stop the Display Manager**: sudo systemctl stop gdm
  - Uninstall Old Drivers:
    - sudo apt-get purge nvidia* **then** sudo apt-get autoremove **then** sudo apt-get autoclean
  - You may now restart or try installing another driver from the terminal
  - **Restart** the Display Manager: sudo systemctl start gdm
  - **Reboot:** sudo reboot
- For windows: Boot into Safe Mode (e.g. F8 or Shift F8)

CUDA Toolkit

- The **CUDA Toolkit** is a development environment for **building applications** (GPU-accelerated).
  - Include libraries, compiler, linker tools, debugging/ optimization, APIs, and runtime libraries.
- Below are some key components:
  - **CUDA Libraries**:
    - **cuBLAS**: for Linear Algebra / matrix operations.
    - **NCCL** (NVIDIA Collective **Communications** Library): For multi-GPU / multi-node
    - cuFFT, cuRAND, cuSPARSE, cuSOLVER, NPP, Thrust (parallel algorithms)
  - CUDA Compiler (**nvcc**) and Linker (nvlink)
  - Profiling and Debugging Tools: Nsight Compute / CUDA-GDB / CUDA-MEMCHECK
  - APIs: CUDA Runtime (high-level) and CUDA Driver (low-level) APIs
  - Utilities: Visual Profiler / Bandwidth Test Utilities
- Tip: Download .deb file and install it

CUDA Toolkit

- The official page will have the latest toolkit, but you can google a version
  - E.g. CUDA Toolkit 11.7 Update 1 Downloads
  - Make choices and select deb



CUDA Toolkit

- Follow the commands. Google the internet for errors

Installation Instructions:

```
$ wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/cuda-ubuntu1804.pin
$ sudo mv cuda-ubuntu1804.pin /etc/apt/preferences.d/cuda-repository-pin-600
$ wget https://developer.download.nvidia.com/compute/cuda/11.7.1/local_installers/cuda-repo-ubuntu1804-11-7-local_11.7.1-515.65.01-1_amd64.de
b
$ sudo dpkg -i cuda-repo-ubuntu1804-11-7-local_11.7.1-515.65.01-1_amd64.deb
$ sudo cp /var/cuda-repo-ubuntu1804-11-7-local/cuda-*-keyring.gpg /usr/share/keyrings/
$ sudo apt-get update
$ sudo apt-get -y install cuda
```

CUDA Toolkit: Troubleshooting

- Sometimes the cuda is not seen by something
- You need to add it to your .bashrc
- Add the following lines to your .bashrc
  - export **PATH**=/usr/local/cuda-<version>/bin${PATH:+:${PATH}}
  - export **LD_LIBRARY_PATH**=/usr/local/cuda-<version>/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
  - Tip: verify the paths
- Run **source ~/.bashrc** to refresh your **environment** variables.
  -

Installing PyTorch GPU

- Go to: https://pytorch.org/get-started/locally/
  - Now select the cuda version
  - However, this is always the latest
    - pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/**cu118**
  - Then google target version
  - Many versions here
    - conda install **pytorch==2.0.0** torchvision==0.15.0 torchaudio==2.0.0 **pytorch-cuda=11.7** -c pytorch -c nvidia
    - conda install **pytorch==1.13.1** torchvision==0.14.1 torchaudio==0.13.1 **pytorch-cuda=11.7** -c pytorch -c nvidia
    - conda install pytorch==**1.12**.1 torchvision==0.13.1 torchaudio==0.12.1 **cudatoolkit=11.3** -c pytorch
    - conda install pytorch==**1.9**.1 torchvision==0.10.1 torchaudio==0.9.1 **cudatoolkit=11.3** -c pytorch -c conda-forge
    - conda install pytorch==**1.13**.1 torchvision==0.14.1 torchaudio==0.13.1 **pytorch-cuda=11.6** -c pytorch -c nvidia
    - conda install pytorch==1.11.0 torchvision==0.12.0 torchaudio==0.11.0 **cudatoolkit=10.2** -c pytorch

Check PyTorch GPU

- **You can do so in the console**
- **import** torch
- print(torch.version.cuda)
- print(torch.cuda.**is_available**())
- print(torch.cuda.**device_count**())
- If nvidia-smi shows a GPU but device_count = 0, then some **mismatch** in installations

```
(test) moustafa@moustafa-Aspire-VN7-793G:~/system-installs1/pycharm-community-2018.2/bin$ conda list | grep pytorch
ffmpeg                    4.3                  hf484d3e_0    pytorch
pytorch                   1.12.0          py3.9_cuda11.3_cudnn8.3.2_0    pytorch
pytorch-mutex             1.0                        cuda    pytorch
torchaudio                0.12.0               py39_cu113    pytorch
torchvision               0.13.0               py39_cu113    pytorch
(test) moustafa@moustafa-Aspire-VN7-793G:~/system-installs1/pycharm-community-2018.2/bin$ python
Python 3.9.18 | packaged by conda-forge | (main, Aug 30 2023, 03:49:32)
[GCC 12.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> print(torch.version.cuda)
11.3
>>> print(torch.cuda.is_available())
True
>>> print(torch.cuda.device_count())
1
>>> exit()
```

Prepackaged Toolkit

- It seems latest PyTorch binaries come **pre-packaged** with their own copies of **CUDA libraries** such as cuBLAS, cuDNN, and NCCL.
  - This setup is designed to make it easier to get started with PyTorch without having to manually manage CUDA dependencies
- This suggests we don’t need to do step 2 explicitly (toolkit installation).
- So try step 1 and step 3. If still facing problems, add step 2

Upgrades

- Sometimes, you would like to make an upgrade to a different PyTroch version
- Don’t just start installations
- First, identify the corresponding toolkit and then the driver
- If your target driver is higher than the installed one, you will need to install a new driver (which may come with troubles)
    - First remove existing nvidia driver
- If you succeeded, then install the new toolkit
    - Better in a new conda env to not lose the old one

nvcc

- The NVIDIA CUDA Compiler (nvcc) is a command-line tool that serves as a wrapper around the **NVIDIA C/C++ compiler** (nvcc).
  - It compiles CUDA source files to generate object code or executables that can run on NVIDIA GPUs.
  - nvcc enables developers to include CUDA C/C++ code in the form of source code or device functions within host code, which is typically written in C/C++.
- Most of the time, you don’t need to worry about it
- If you want it and you don’t find it installed, you may install with
  - sudo apt install **nvidia-cuda-toolkit**

Suitable GPU

- Speed and GPU RAM are critical factors
- During my PhD/Work, I found 12 **GPU Ram (VRAM)** are a must for training
  - With nowadays transformers / LLMs, one may need 24+
- Don’t aim for a laptop to **train** Deep Networks
  - It will be very expensive (4k+) - hard to replace the GPU
  - You can use a cheap NVIDIA GPU to **do inference** or basic training cycle for validation
    - Then you need some cloud service that trains for you
  - Also, training on your local laptop will noisy and busy for hours/days to train
  - If you can afford it, you can build a desktop for 5k USD
    - Consider also machine RAM 16+ / SSD 1T / 4T HDD
- Educate yourself: here / here
- Tip: there are websites to evaluate if components work together

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."