工欲善其事，必先利其器

> 有一个英语词汇叫做Handy，讲的是便利的，易使用的，当你有一个良好的环境配置时候，编程将变得handy，随手打开即可编程，一点都不复杂，所以配置好的环境，是未来学习快速进步的必要保障。

首先来了解一下深度学习框架

深度学习框架

1、深度学习框架是什么

在深度学习初始阶段，每个深度学习研究者都需要写大量的重复代码。

为了提高工作效率，他们就将这些代码写成了框架放到网上让所有研究者一起使用。

作一个简单的比喻，一套深度学习框架就是一套积木，各个组件就是某个模型或算法的一部分，你可以自己设计如何使用积木去堆砌符合你数据集的积木。

> 📌 **思考题**
>
> 自行了解张量和基于张量的各种操作。

2、为什么需要深度学习框架

显然是为了降低使用门槛。 深度学习对硬件环境的依赖很高，对于开发者有较高的门槛，深度学习计算框架的出现，屏蔽了大量硬件环境层面的开发代价，使研究者和开发人员可以专注于算法的实现，

快速迭代。

TensorFlow和pytorch

这么多的框架，我们应该如何选择呢（好吧直接就TensorFlow和pytorch了）

1. TensorFlow

开发语言

基于python编写，通过C/C++引擎加速，是Google开源的第二代深度学习框架。

编程语言

Python是处理TensorFlow的最方便的客户端语言。不过，JavaScript、C++、Java、Go、C#和Julia也提供了实验性的交互界面。

优点

（不讲人话的版本）

处理循环神经网RNN非常友好。其用途不止于深度学习，还可以支持增强学习和其他算法。

内部实现使用了向量运算的符号图方法，使用图graph来表示计算任务，使新网络的指定变得相当容易，支持快速开发。TF使用静态计算图进行操作。也就是说，我们首先定义图，然后运行计算，如果需要对架构进行更改，我们将重新训练模型。TF选择这种方法是为了提高效率，但是许多现代神经网络工具能够在不显著降低学习速度的情况下，同时兼顾到在学习过程中进行改进。在这方面，TensorFlow的主要竞争对手是Pythorch。

（讲人话啊喂！！）

- 谷歌爸爸一撑腰，研究代码两丰收
- 新版TensorFlow API（STFW) 较简洁
- 天生和谷歌云兼容
- 有良好的推断支持
- 功能十分强大!

缺点

（不讲人话的版本）

目前TensorFlow还不支持“内联（inline）”矩阵运算，必须要复制矩阵才能对其进行运算，复制庞大的矩阵会导致系统运行效率降低，并占用部分内存。

TensorFlow不提供商业支持，仅为研究者提供的一种新工具，因此公司如果要商业化需要考虑开源协议问题。

（讲人话！！）

- API不稳定
- 学习成本高
- 开发成本高
- 会出现前面版本存在的功能后面版本直接没了

2.pytorch

开发语言

Facebook用Lua编写的开源计算框架，支持机器学习算法。Tensorflow之后深入学习的主要软件工具是PyTorch。

Facebook于2017年1月开放了Torch的Python API — PyTorch源代码。

编程语言

PyTorch完全基于Python。

（直接说人话吧）

优点

- 上手容易
- 代码简洁
- 有较好的灵活性和速度
- 发展快速，现在已经支持TPU
- API相对稳定
- 里面附带许多开源模型代码可以直接调用

缺点

- 没有Keras API 那么简洁
- 一些功能难以实现

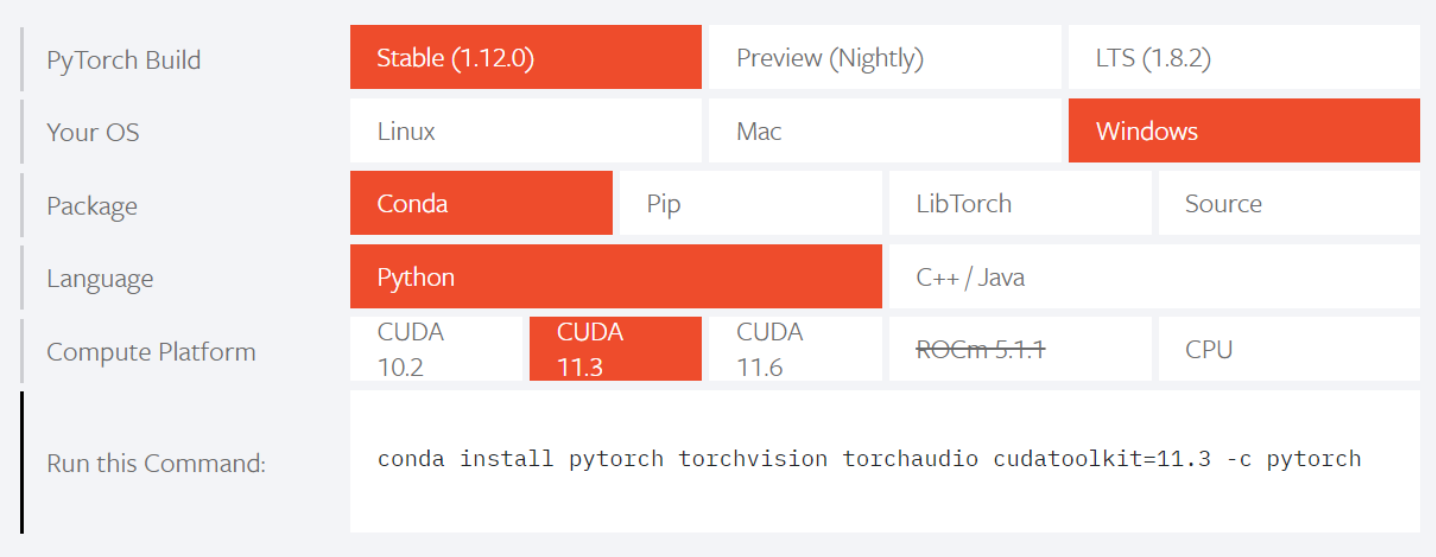
安装

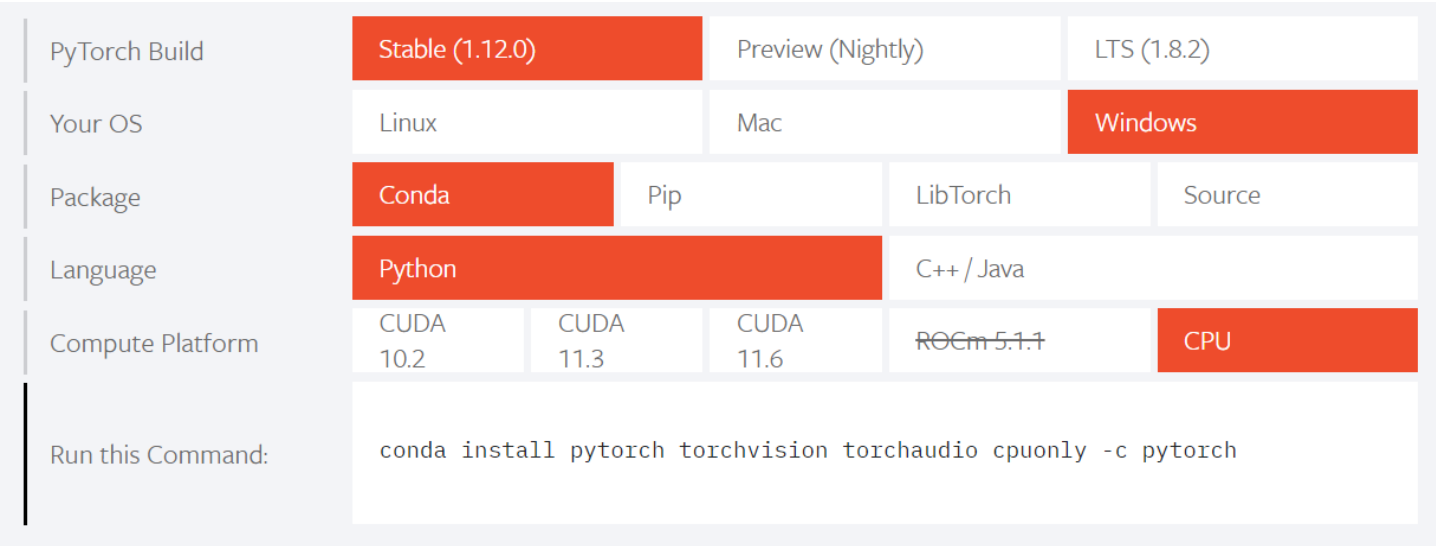
Pytorch

官网如下

https://pytorch.org/

PyTorch





选择Conda或者Pip安装皆可

最后选择CUDA版本或者CPU版本运行指令就好了

Tip：conda换源

如果你使用conda安装pytorch太慢或者失败，不妨换个下载源试试

在cmd 命令行中，输入添加以下命令：

```
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/f
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/m
```

```
conda config --set show_channel_urls yes
```

TensorFlow

https://tensorflow.google.cn/

TensorFlow

An end-to-end open source machine learning platform for everyone. Discover TensorFlow's flexible ecosystem of tools, libraries and community resources.

安装 TensorFlow 2

我们在以下 64 位系统上测试过 TensorFlow 并且这些系统支持 TensorFlow：

- Python 3.6–3.9
- Ubuntu 16.04 或更高版本
- Windows 7 或更高版本（含 C++ 可再发行软件包）
- macOS 10.12.6 (Sierra) 或更高版本（不支持 GPU）

下载软件包

使用 Python 的 pip 软件包管理器安装 TensorFlow。

> ★ TensorFlow 2 软件包需要使用高于 19.0 的 `pip` 版本（对于 macOS 来说，则需要高于 20.3 的 pip 版本）。

```
# Requires the latest pip
$ pip install --upgrade pip

# Current stable release for CPU and GPU
$ pip install tensorflow

# Or try the preview build (unstable)
$ pip install tf-nightly
```

官方软件包支持 Ubuntu、Windows 和 macOS。

有关支持 CUDA® 的卡，请参阅 GPU 指南。

📌 **教程**

在Windows上配置pytorch！（CPU和GPU版）

Windows 下 PyTorch 入门深度学习环境安装与配置 CPU GPU 版

最新 TensorFlow 2.8 极简安装教程

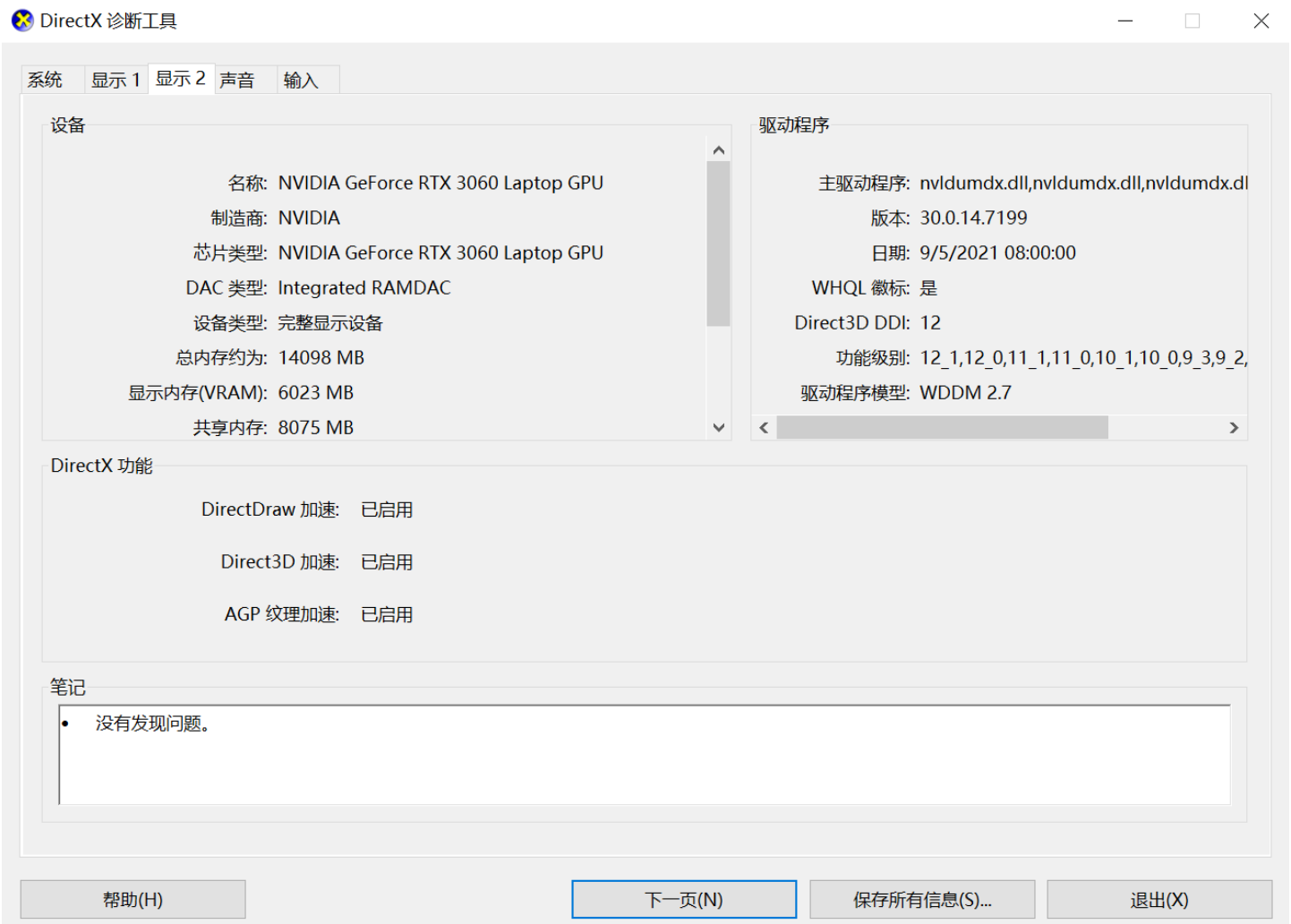
📌 **思考题：为什么需要CUDA版本？？？**

cuda版本需要额外配置，我们将这个任务留给聪明的你！！！

Tips：Windows和Linux如何查看显卡信息

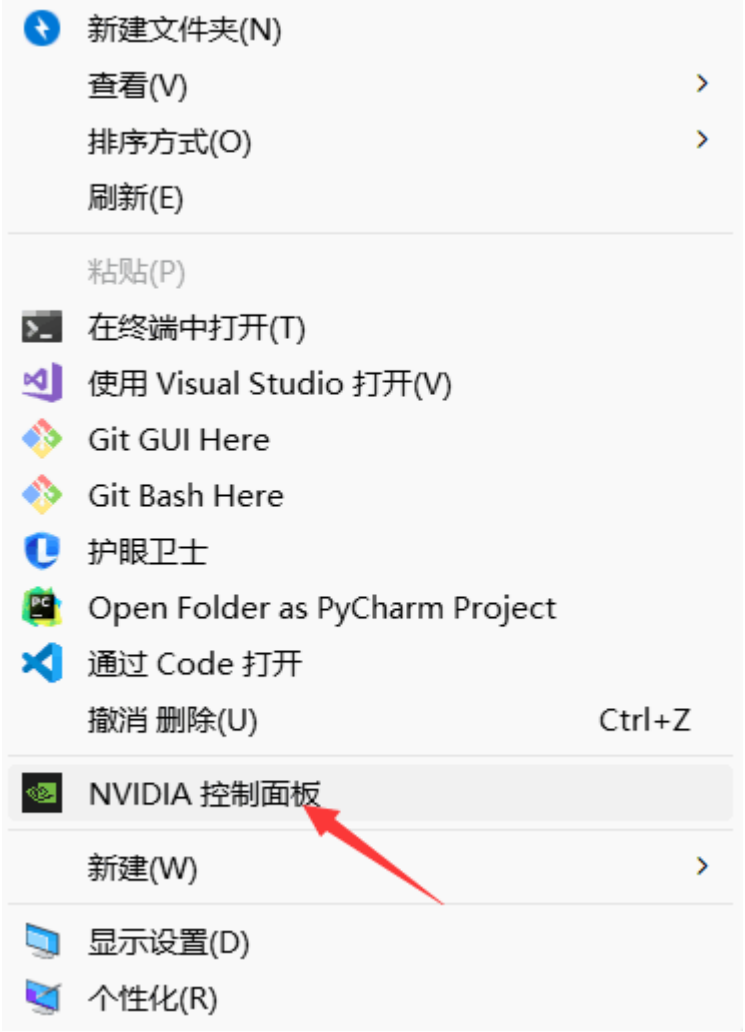
windows

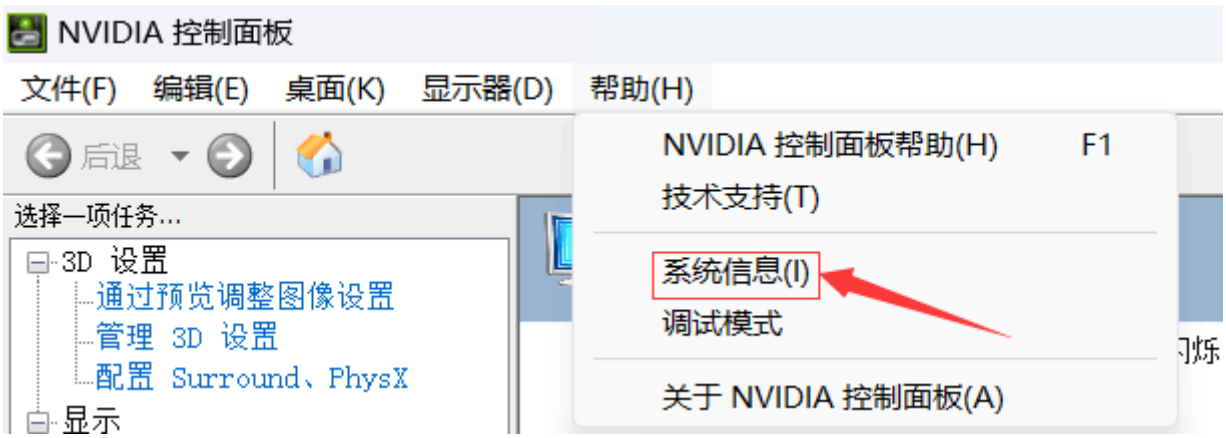
同时按下键盘的win+r键，打开cmd，键入 `dxdiag` 然后回车
系统、显卡、声卡以及其他输入设备的信息都在这里了。（给出我的界面）

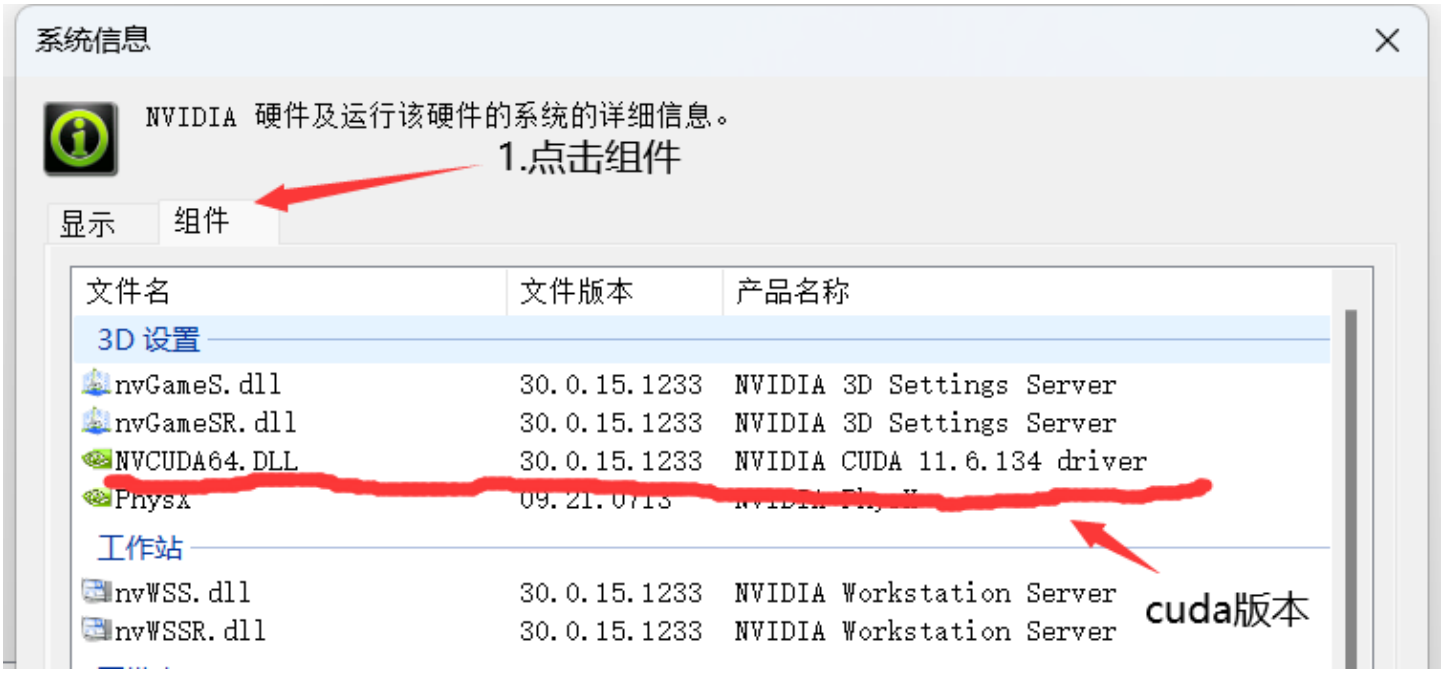


cuda版本查看

桌面空白位置摁下右键


新建文件夹(N)
查看(V)
排序方式(O)
刷新(E)
粘贴(P)
在终端中打开(T)
使用 Visual Studio 打开(V)
Git GUI Here
Git Bash Here
护眼卫士
Open Folder as PyCharm Project
通过 Code 打开
撤消 删除(U) Ctrl+Z
NVIDIA 控制面板
新建(W)
显示设置(D)
个性化(R)


NVIDIA 控制面板
文件(F) 编辑(E) 桌面(K) 显示器(D) 帮助(H)
后退
选择一项任务...
3D 设置
通过预览调整图像设置
管理 3D 设置
配置 Surround、PhysX
显示
NVIDIA 控制面板帮助(H) F1
技术支持(T)
系统信息(I)
调试模式
关于 NVIDIA 控制面板(A)



linux

打开bash键入

```
1 nvidia-smi
```

很多人会混淆的东西（非常重要）

1. cuda driver version / cuda runtime version

通常大家所指的cuda是位于/usr/local下的cuda

```
(base) ➜ ~ ls /usr/local
bin  cuda  cuda-11.6  etc  games  include  lib  libdata  man  opencv345  sbin  share  src
```

当然可以看到cuda是cuda-11.6所指向的软链接（类似windows的快捷方式），所以我们如果要切换cuda版本只需要改变软链接的指向即可。

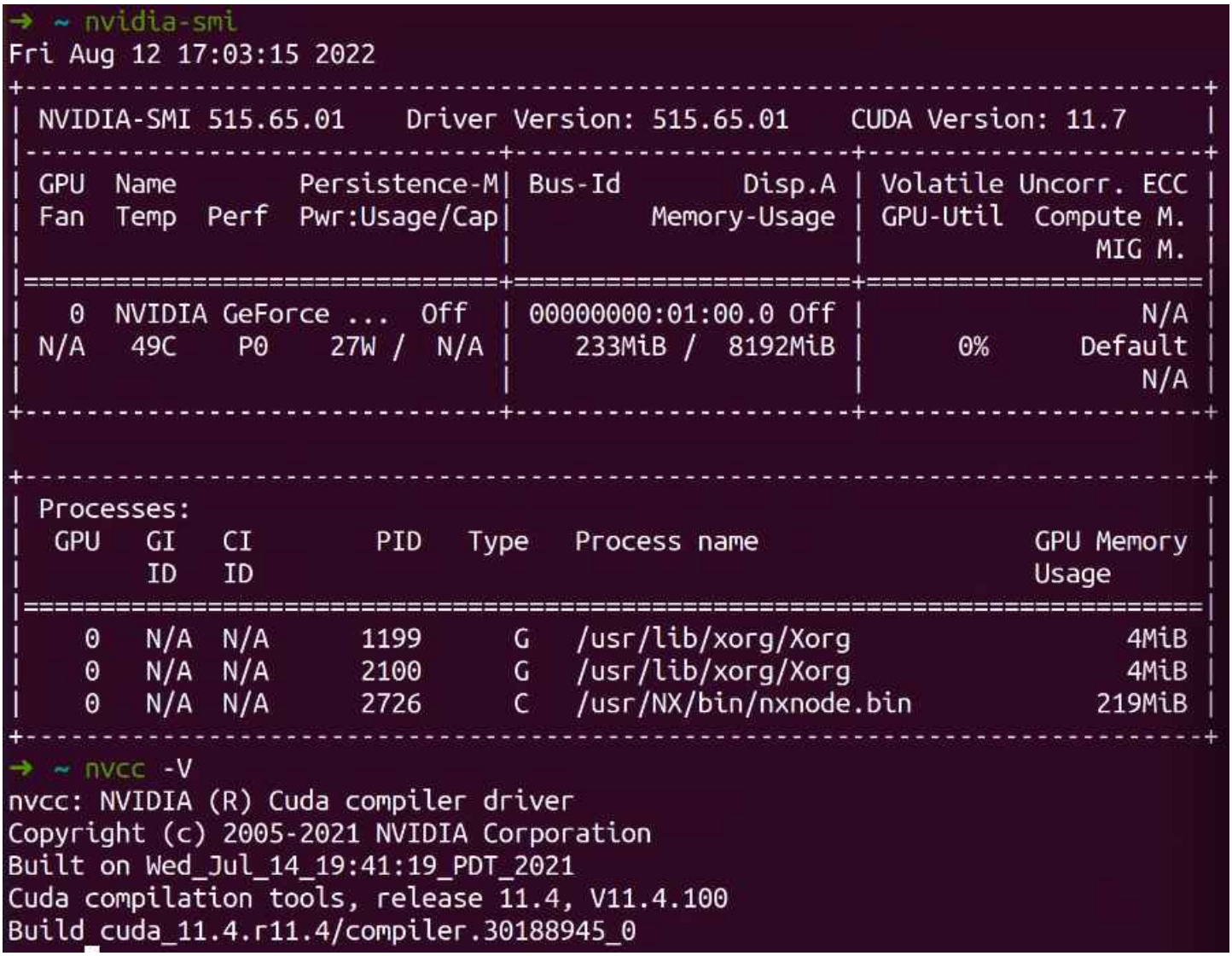
```
(base) ➜ ~ ll /usr/local | grep cuda
lrwxrwxrwx  1 root staff    21 Jun  5 13:55 cuda -> /usr/local/cuda-11.6/
drwxr-sr-x 16 root staff  4096 Jun  5 13:56 cuda-11.6
```

cuda driver version是cuda 的驱动版本。

cuda runtimer version是我们实际很多时候我们实际调用的版本。

二者的版本是可以不一致的。如下图所示：

```
➜ build ./cv_cuda
Device 0:  "NVIDIA GeForce RTX 3070 Laptop GPU"  7982Mb, sm_86, Driver/Runtime ver.11.70/11.40
CUDA Device(s) Number: 1
CUDA Device(s) Compatible: 1
```

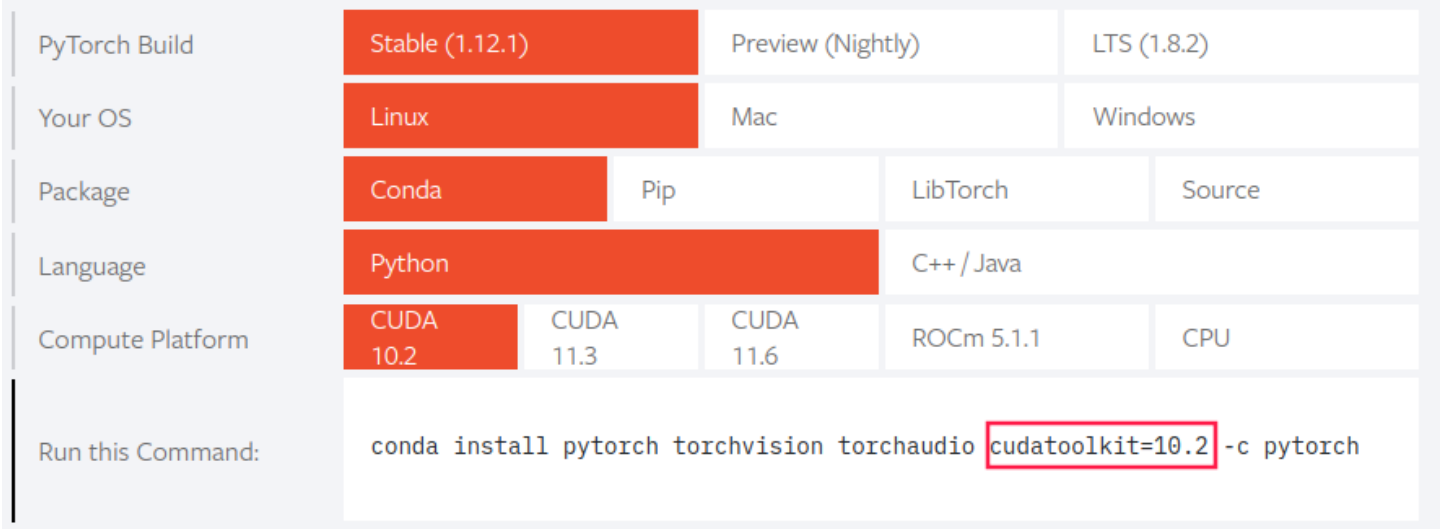


一般来讲cuda driver是向下兼容的。所以cuda driver version >= cuda runtime version 就不会太大问题。

如果我们用C++写CUDA，具体的说就是编写以.cu为后缀的文件。就是用nvcc（cuda编译器）去编译的，nvcc是cuda runtime api的一部分。cuda runtime 只知道自身构建时的版本，并不知道是否GPU driver的版本，甚至不知道是否安装了GPU driver。

2. Pytorch/tensorflow 使用的cuda版本

以pytorch为例，可以看到在安装过程中我们选择的cuda版本是10.2



那么这个cudatookit10.2 和nvidia-smi的11.7 以及 nvcc -V 的11.4三者有什么区别呢?

pytorch实际只需要cuda的链接文件，即.so文件，这些链接文件就都包含的cudatookkit里面。并不需要cuda的头文件等其他东西，如下所示

```
(base) ➜  ~ ls /usr/local/cuda
bin                 EULA.txt  lib64                    nsightee_plugins         nvvm     share    tools
compute-sanitizer   extras    libnvvp                  nsight-systems-2021.5.2  README   src      version.json
DOCS                include   nsight-compute-2022.1.1  nvml                     samples  targets
(base) ➜  ~ ls /usr/local/cuda/src
cusparse_fortran.c         cusparse_fortran.h  fortran_common.h  fortran_thunking.c
cusparse_fortran_common.h  fortran.c           fortran.h         fortran_thunking.h
```

所以我们如果想让使用pytorch-cuda版本，我们实际上不需要/usr/local/cuda。只需要在安装驱动的前提下，在python里面安装cudatookit即可。

但是有一种情况例外，就是你要用C++ CUDA 编写核函数给pytorch当做插件。这种情况下就需要/usr/local/cuda以及nvcc，cudatookit，而且后面两个版本很多时候需要保持严格一致。

3. Cudnn

Cudnn 是一些链接文件，你可以理解成是为了给cuda计算加速的东西。同样的我们也可以用以下命令查看/usr/local/cuda的cudnn：

```
(base) ➜  ~ ls /usr/local/cuda/lib64 | grep cudnn*
libcudnn_adv_infer.so
libcudnn_adv_infer.so.8
libcudnn_adv_infer.so.8.4.1
libcudnn_adv_infer_static.a
```

以及pytorch的cuda环境的cudnn

```
(py37) ➜  ~ conda list | grep cuda
cudatoolkit               11.6.0               hecad31d_10    conda-forge
pytorch                   1.12.0          py3.7_cuda11.6_cudnn8.3.2_0    pytorch
pytorch-mutex             1.0                        cuda    pytorch
```