Lecture 0
Introduction to JAX

明玉瑞 Yurui Ming

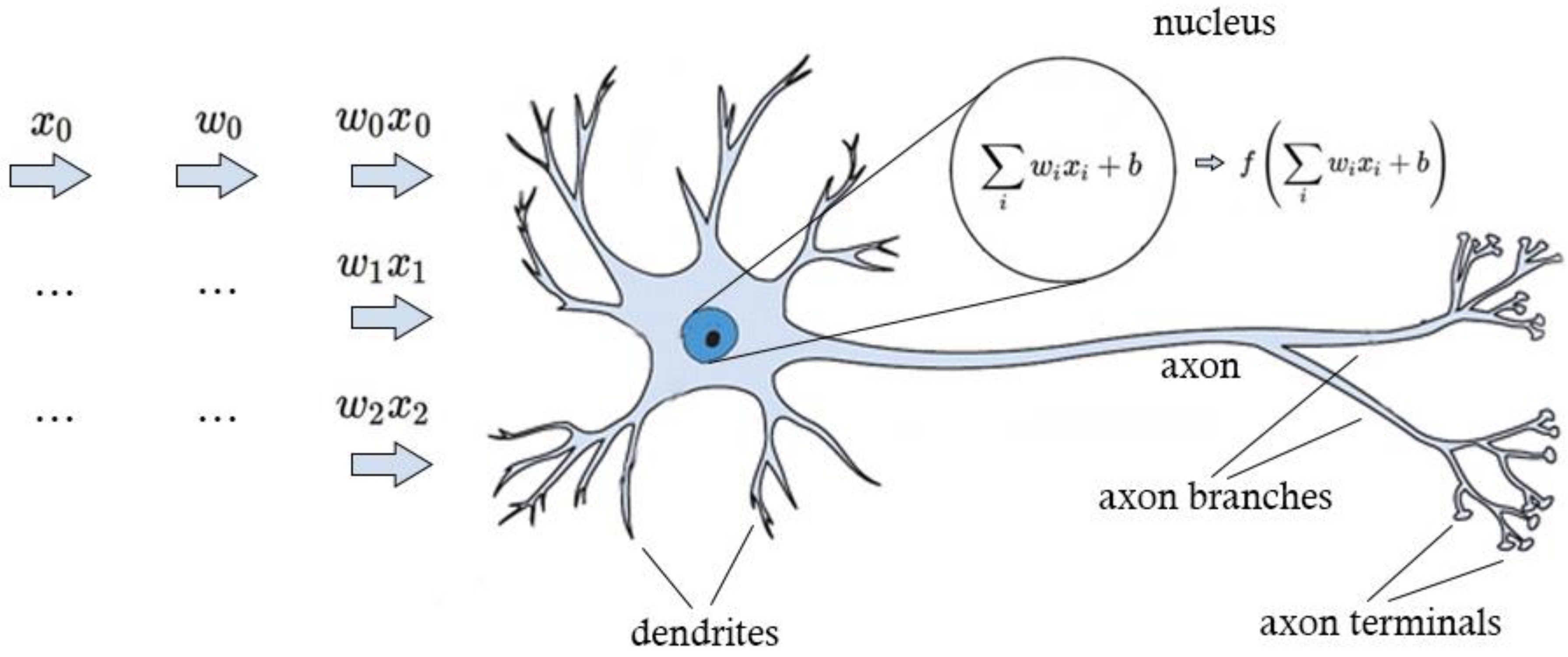
yrming@gmail.com

声明
Disclaimer

- 本讲义在准备过程中由于时间所限，所用材料来源并未规范标示引用来源。所引材料仅用于教学所用，作者无意侵犯原著者之知识版权，所引材料之知识版权均归原著者所有；若原著者介意之，请联系作者更正及删除。

  The time limit during the preparation of these slides incurs the situation that not all the sources of the used materials (texts or images) are properly referenced or clearly manifested. However, all materials in these slides are solely for teaching and the author is with no intention to infringe the copyright bestowed on the original authors or manufacturers. All credits go to corresponding IP holders. Please address the author for any concern for remedy including deletion.

Motivation



$$y = \varphi(\sum_{i=1}^{n} w_ix_i + b) = \varphi(\mathbf{w}^T\mathbf{x} + b)$$

Motivation

- 许多复杂的数学运算可以被分解为一系列标量运算。

  Many complex math operations can be decomposed into a series of scalar operations.

- 一些运算以向量形式表示时，非常简洁且易于理解。

  Vectorization of scalar operations can result in succinct representation which can be easily understood.

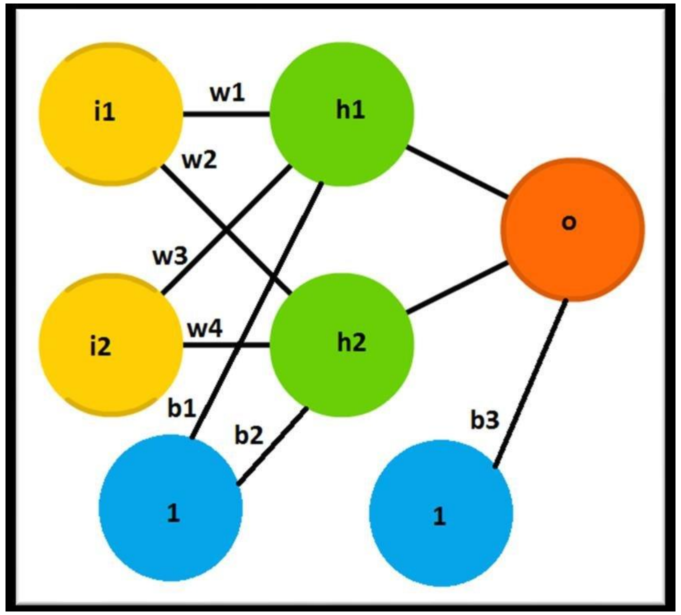
- 现代许多硬件可以原生地执行向量运算。

  Many contemporary hardware are capable of doing vectorized operations.

Motivation

- Python的数值运算库Numpy在一定程度上解决了上述问题。

  The numerical computation library Numpy of Python to some extent some this problem.



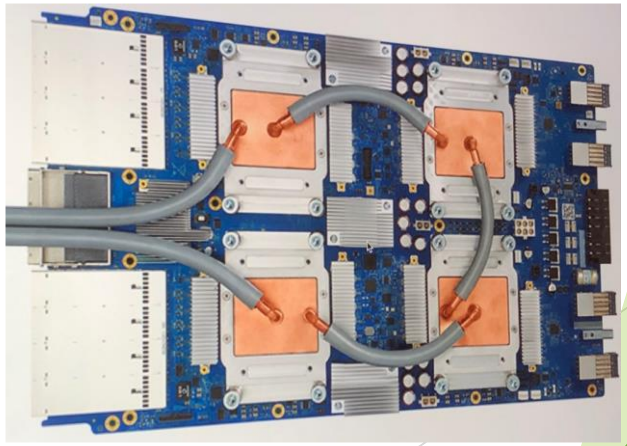
```
import numpy as np

def predict(params, inputs):
    for W, b in params:
        outputs = np.dot(inputs, W) + b
        outputs = np.tanh(outputs)
    return outputs

def loss(params, batch):
    inputs, targets = batch
    preds = predict(params, inputs)
    return np.sum((preds – targets) ** 2)
```

Motivation

- 上述利用原生的Numpy的方式能从面上解决问题，但并非最优，因为缺乏如下一些特性：
  - 基于GPU或TPU的硬件加速
  - 优化的自动微分机制
  - 编译优化的聚合、内存布局、重整
  - 向量化的批处理操作
  - 数据与操作的并行化

- Utilizing the original Numpy can solve the problem superficially, however, due to lacks of features below, it is not the optimal way:
  - Accelerator hardware (GPU/TPU)
  - Fast optimization via autodiff
  - Optimized compilation with fusion, memory layout, remat, …
  - Vectorized batching of operations
  - Parallelization of data & computations

Motivating JAX



GPU/TPU autodiff

batching compilation parallelization

Motivating JAX

```python
import numpy as np

def predict(params, inputs):
    for W, b in params:
        outputs = np.dot(inputs, W) + b
        outputs = np.tanh(outputs)
    return outputs

def loss(params, batch):
    inputs, targets = batch
    preds = predict(params, inputs)
    return np.sum((preds - targets) ** 2)
```

```python
import jax.numpy as jnp

def predict(params, inputs):
    for W, b in params:
        outputs = jnp.dot(inputs, W) + b
        outputs = jnp.tanh(outputs)
    return outputs

def loss(params, batch):
    inputs, targets = batch
    preds = predict(params, inputs)
    return jnp.sum((preds - targets) ** 2)
```

Jax

- JAX是一个专为高性能数值计算而设计的Python库。它支持的主要特性包括：

  JAX is a Python library designed for high-performance numerical computing. Among its key ingredients it supports:

  - JIT-compilation:

    JAX programs are just-in-time (JIT) compiled by JAX’s underlying mechanism called XLA, which results in faster CPU code, and transparent GPU and Cloud TPU acceleration.

  - Vectorization:

    JAX implements SIMD programming paradigm via automatic vectorization, e.g., vmap, pmap.

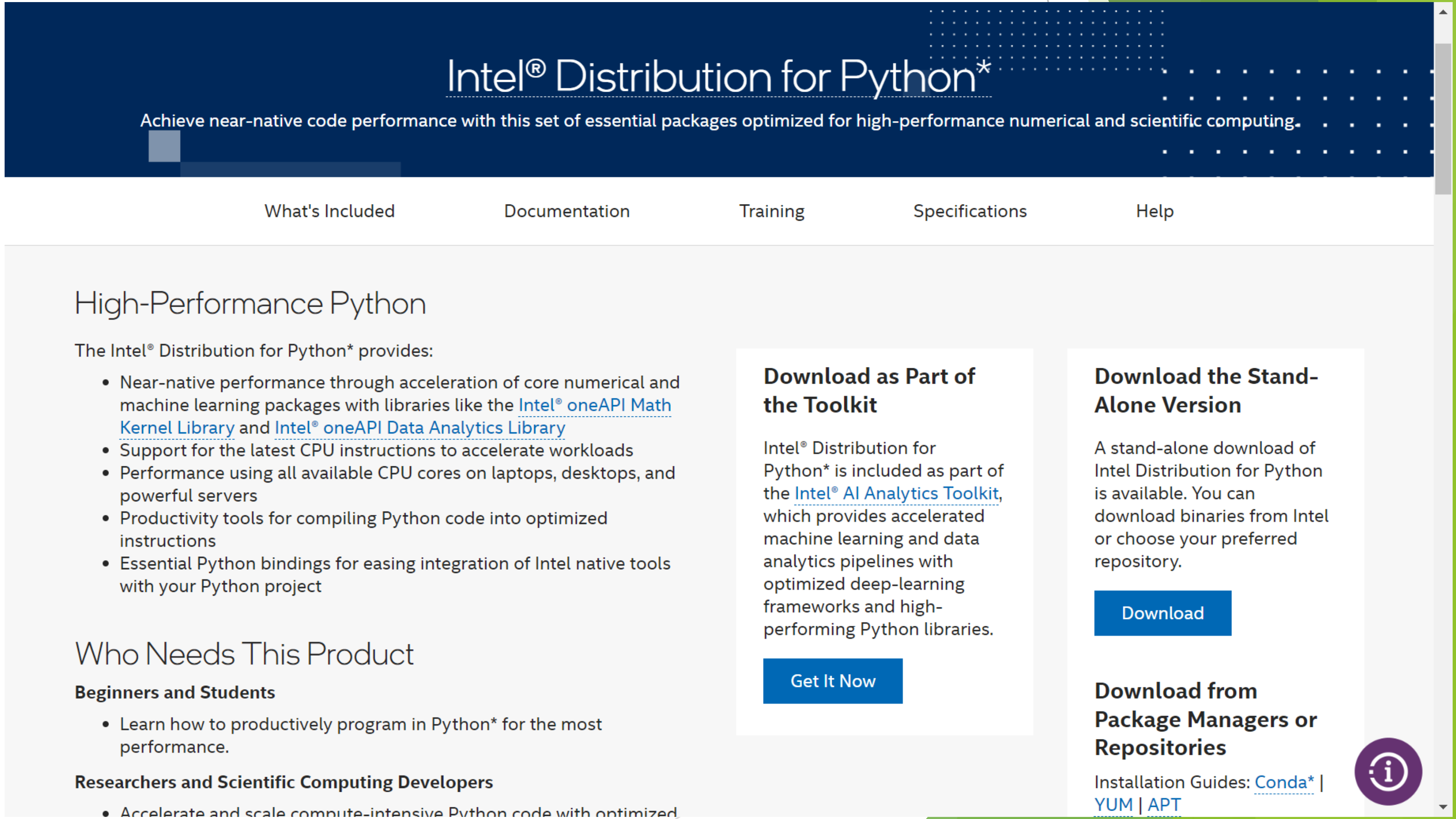
  - Differentiation:

    JAX extends automatic differentiation of arbitrary numerical functions, such as forward differentiation, reverse differentiation, etc. One can even easily compose these differentiations. The provided functions are grad, hessian, jacfwd, jacrev, etc.

Installation

- JAX原生支持Linux平台，依赖于Python，并且与Python发行版本相关；在Windows平台的安装依赖于社区的工作，且对GPU的支持有些滞后。下面介绍基于Anaconda环境的安装。
- 对于基于Intel CPU的电脑，由于Intel针对该公司的CPU做过Python及相关库的优化，因此推荐使用Intel的Python发行版本：
- 网址为：

https://www.intel.com/content/www/us/en/developer/tools/oneapi/distribution-for-python.html



Intel® Distribution for Python*

Achieve near-native code performance with this set of essential packages optimized for high-performance numerical and scientific computing.

High-Performance Python

The Intel® Distribution for Python* provides:

- Near-native performance through acceleration of core numerical and machine learning packages with libraries like the Intel® oneAPI Math Kernel Library and Intel® oneAPI Data Analytics Library
- Support for the latest CPU instructions to accelerate workloads
- Performance using all available CPU cores on laptops, desktops, and powerful servers
- Productivity tools for compiling Python code into optimized instructions
- Essential Python bindings for easing integration of Intel native tools with your Python project

Who Needs This Product

Beginners and Students

- Learn how to productively program in Python* for the most performance.

Researchers and Scientific Computing Developers

- Accelerate and scale compute-intensive Python code with optimized

Download as Part of the Toolkit

Intel® Distribution for Python* is included as part of the Intel® AI Analytics Toolkit, which provides accelerated machine learning and data analytics pipelines with optimized deep-learning frameworks and high-performing Python libraries.

Download the Stand-Alone Version

A stand-alone download of Intel Distribution for Python is available. You can download binaries from Intel or choose your preferred repository.

Download from Package Managers or Repositories

Installation Guides: Conda* | YUM | APT

Installation

- JAX is only natively supported on the Linux variant platforms. JAX depends on the Python and its release version due to the compilation process. Installation on Windows platform is supported by the community however advanced features such as GPU are of limited support. The following instructions are based on the Anaconda toolset.
- Intel has optimized Python to exert potentials of their CPUs by exploiting some exclusive features. So if your computer is installed with an Intel CPU, use the Python release from Intel. Just install the standalone version from the below website is enough. You can search something like "Intel python installation standalone" to find the source:

  https://www.intel.com/content/www/us/en/developer/tools/oneapi/distribution-for-python.html
- For AMD CPU, one can install the software directly from the official website of Anaconda:

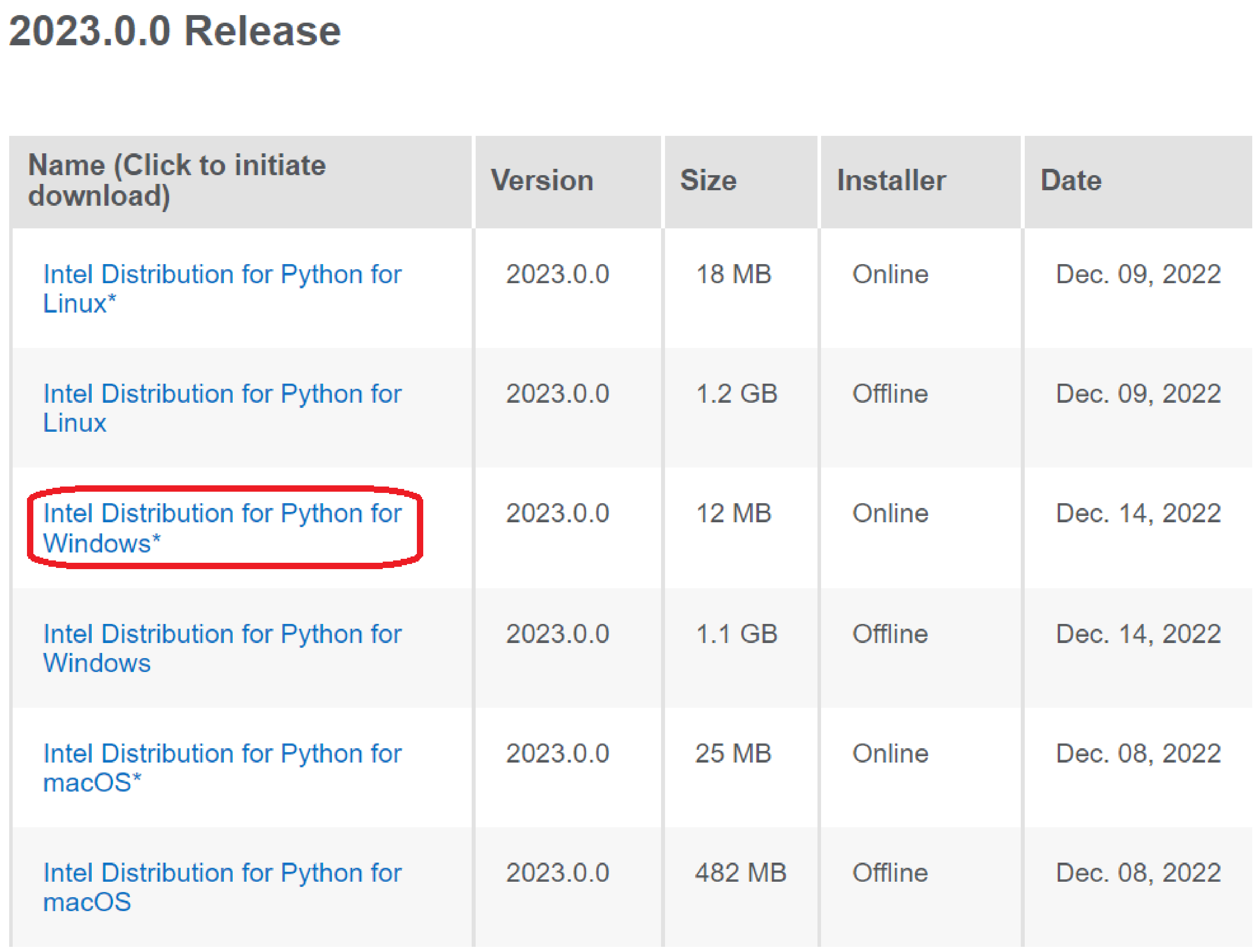
  https://www.anaconda.com/

Installation

- 我们以Intel的Python发行版本为例进行讲解，首先下载并安装Python：

  We exemplify the process by installing the Python shipped by Intel, first download and install it from Intel's website:

https://www.intel.com/content/www/us/en/developer/articles/tool/oneapi-standalone-components.html#python



2023.0.0 Release

Name (Click to initiate download)	Version	Size	Installer	Date
Intel Distribution for Python for Linux*	2023.0.0	18 MB	Online	Dec. 09, 2022
Intel Distribution for Python for Linux	2023.0.0	1.2 GB	Offline	Dec. 09, 2022
Intel Distribution for Python for Windows*	2023.0.0	12 MB	Online	Dec. 14, 2022
Intel Distribution for Python for Windows	2023.0.0	1.1 GB	Offline	Dec. 14, 2022
Intel Distribution for Python for macOS*	2023.0.0	25 MB	Online	Dec. 08, 2022
Intel Distribution for Python for macOS	2023.0.0	482 MB	Offline	Dec. 08, 2022

Installation

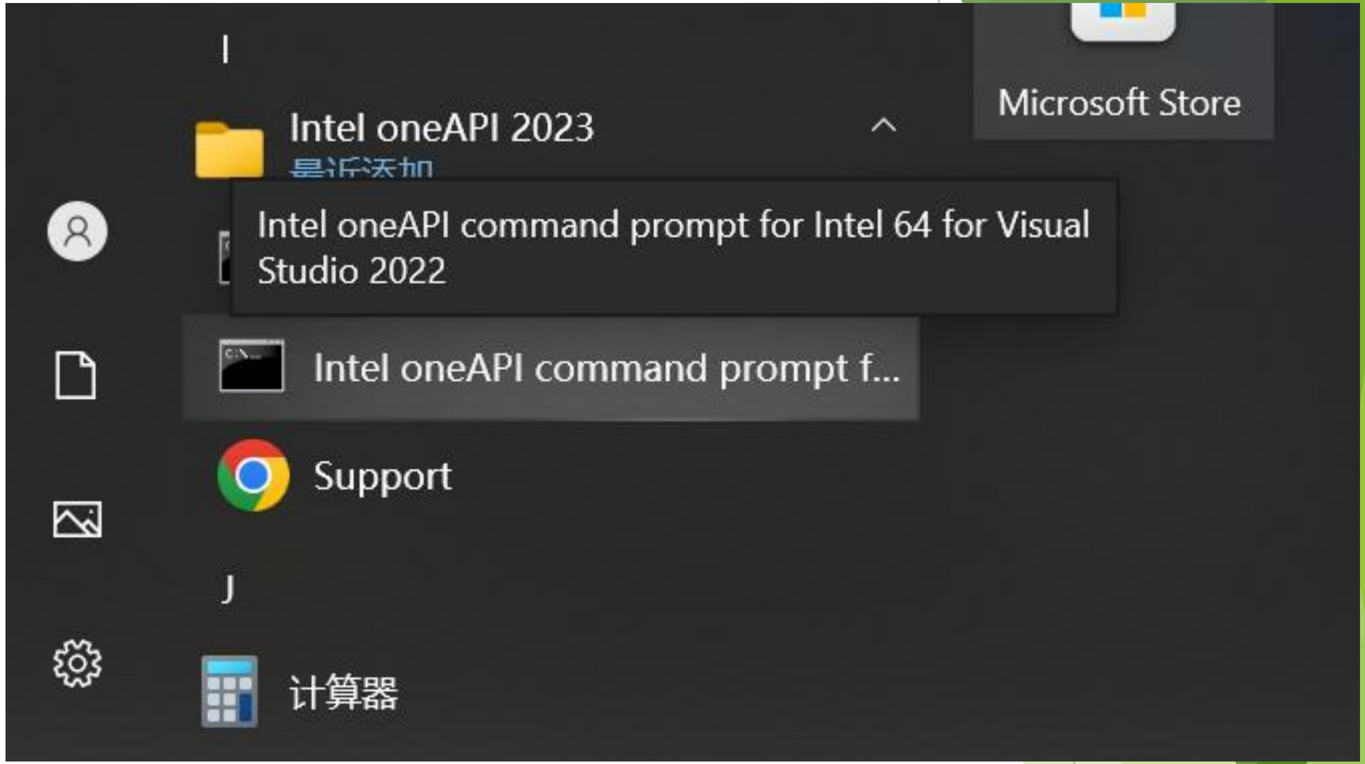
- 启动安装后的Intel Python环境，一般启动64位的。查看Python版本：

  Launch the Intel Python environment, usually the 64-bit command prompt. Check the Python version:

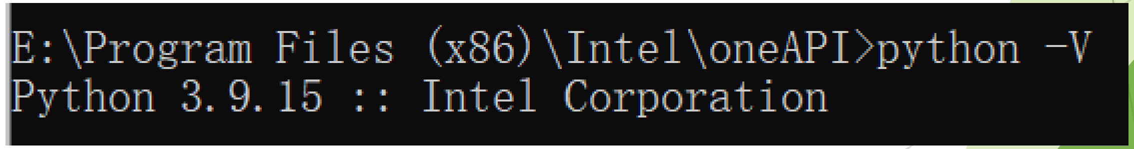
  python -V

- 可以看出作者所安装的Python的版本为3.9。

  It is manifest that the python version of the author’s installation is 3.9.



I
Intel oneAPI 2023
最近添加
Microsoft Store
Intel oneAPI command prompt for Intel 64 for Visual Studio 2022
Intel oneAPI command prompt f...
Support
J
计算器



```
E:\Program Files (x86)\Intel\oneAPI>python -V
Python 3.9.15 :: Intel Corporation
```

Installation

- 下一步，我们需要找社区支持的Windows安装的JAXLib的源。经过简单搜索即可知其维护在Github上：

  The next step is to find out the JAXLib source which provides the package could be installed on Windows. A quick search reveals its maintenance via Github:

  https://github.com/cloudhan/jax-windows-builder

  相应的源的网址如下：

  The address of the source is as follows:

  https://whls.blob.core.windows.net/unstable/index.html

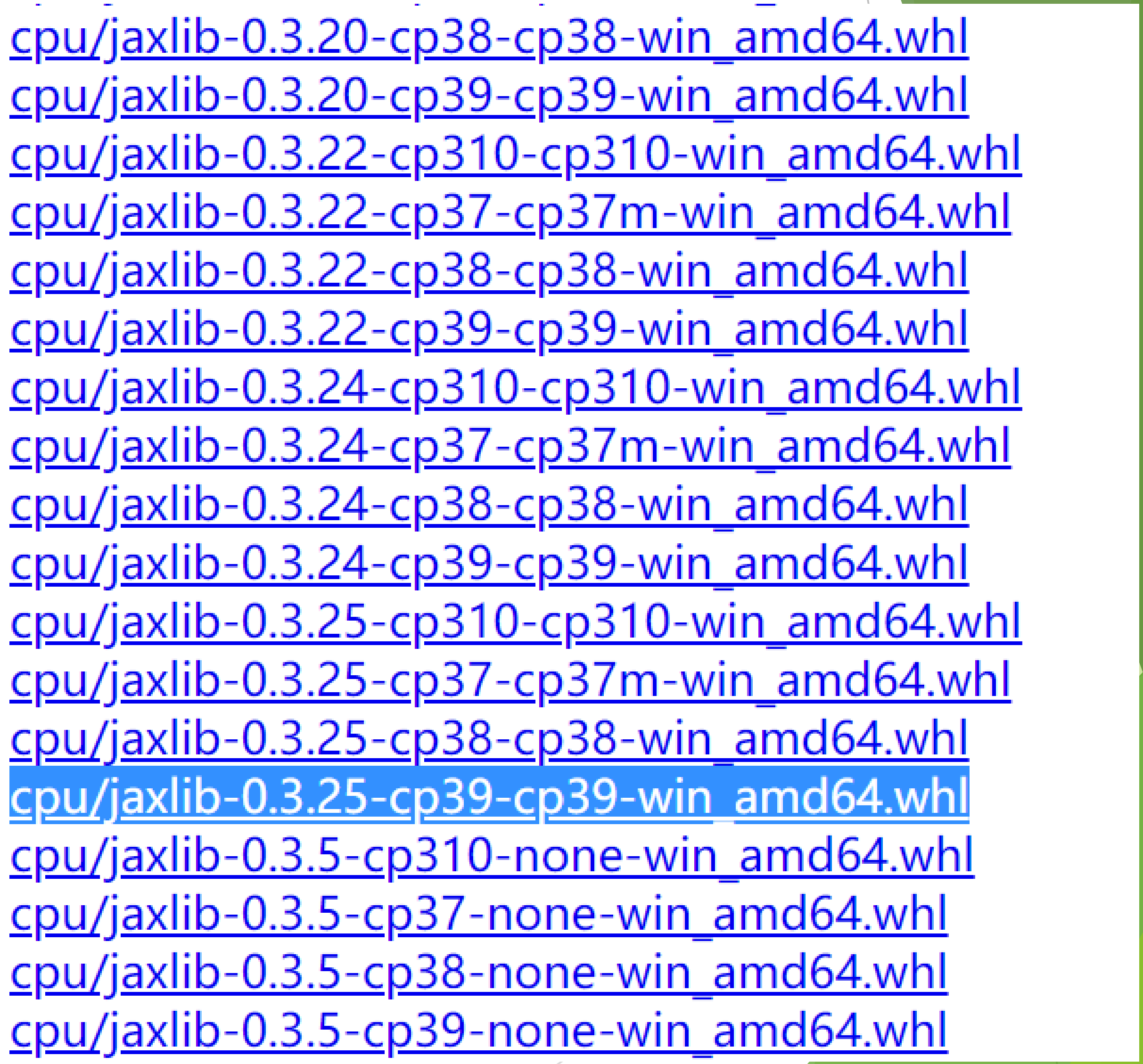
- 从源上列出的包中，我们可以知道对应的Python的版本。读者可以安装CPU版本或GPU版本（可能需要回退最新的显卡驱动），但均推荐创建一个新的虚拟环境进行安装。

Installation

- 查看社区支持的Windows的JAXLib的最新版本并下载到本地。读者根据实际情况，决定安装仅CPU版本或者是支持GPU的：

  Check and download the latest community-supported version of JAXLib. Users can decide on which one to install, the CPU version or the version with GPU supported.

https://whls.blob.core.windows.net/unstable/index.html


cpu/jaxlib-0.3.20-cp38-cp38-win_amd64.whl
cpu/jaxlib-0.3.20-cp39-cp39-win_amd64.whl
cpu/jaxlib-0.3.22-cp310-cp310-win_amd64.whl
cpu/jaxlib-0.3.22-cp37-cp37m-win_amd64.whl
cpu/jaxlib-0.3.22-cp38-cp38-win_amd64.whl
cpu/jaxlib-0.3.22-cp39-cp39-win_amd64.whl
cpu/jaxlib-0.3.24-cp310-cp310-win_amd64.whl
cpu/jaxlib-0.3.24-cp37-cp37m-win_amd64.whl
cpu/jaxlib-0.3.24-cp38-cp38-win_amd64.whl
cpu/jaxlib-0.3.24-cp39-cp39-win_amd64.whl
cpu/jaxlib-0.3.25-cp310-cp310-win_amd64.whl
cpu/jaxlib-0.3.25-cp37-cp37m-win_amd64.whl
cpu/jaxlib-0.3.25-cp38-cp38-win_amd64.whl
cpu/jaxlib-0.3.25-cp39-cp39-win_amd64.whl
cpu/jaxlib-0.3.5-cp310-none-win_amd64.whl
cpu/jaxlib-0.3.5-cp37-none-win_amd64.whl
cpu/jaxlib-0.3.5-cp38-none-win_amd64.whl
cpu/jaxlib-0.3.5-cp39-none-win_amd64.whl

Installation

- 在创建虚拟环境之前，我们首先更新一下conda软件包：

  First we update the conda before creating the virtual environment:

  ```
  conda update conda
  ```

- 如果安装的是Intel的Python发行版，那么通过如下命令创建名为jax的虚拟环境：

  If we installed the Python software shipped by Intel, then enter the following command to create a new environment named jax:

  ```
  conda create -n jax --clone base
  ```

- 激活虚拟环境：

  Acitvate conda environment:

  ```
  conda activate jax
  ```

Installation

- 注意，如果我们使用的是Intel的CPU，但不是安装的Intel的Python发行版，例如通过Anaconda官网上安装，则推荐将Intel加入为源，然后再创建名为jax的虚拟环境：

  If you are using Intel’s CPU, but not installed the Python shipped by Intel, for example, you did it via downloading from Anaconda’s official website, then it is recommended to add Intel as the channel first, following by creating a virtual environment named jax:

  ```
  conda config --add channels intel
  ```

  ```
  conda create -n jax intelpython3_core
  ```

- 注意，使用参数intelpython3_core表明仅安装Intel维护的上游当前Python版本的核心功能，如想安装全部功能，则可以用参数intelpython3_full。同时，可以用显示指定版本，例如安装3.7的全部功能：

  Notably, the parameter intelpython3_core indicates to install the core function of the up-stream python libraries maintained by Intel. To install the full packages, use the parameter intelpython3_full. You can simultaneously specify a particular Python version, such as 3.7:

  ```
  conda create -n jax intelpython3_full python=3.7
  ```

Installation

- 接下来，在虚拟环境中安装之前下载的JAXLib安装包，例如：

  Installed the downloaded JAXLib in the virtual environment, for example:

  ```
  pip install cpu/jaxlib-0.3.25-cp39-cp39-win_amd64.whl --use-deprecated legacy-resolver
  ```

- 安装好JAXLib库之后，再安装对应的JAX库。注意，安装的JAX库的版本，须与安装的JAXLib的版本一致：

  We need to install JAX after installing JAXLib. Notably, the version of JAX to be installed must match the version of the already installed JAXLib:

  ```
  pip install jax==0.3.25
  ```

- 执行下面命令，若无报错，则说明JAX及JAXLib安装成功：

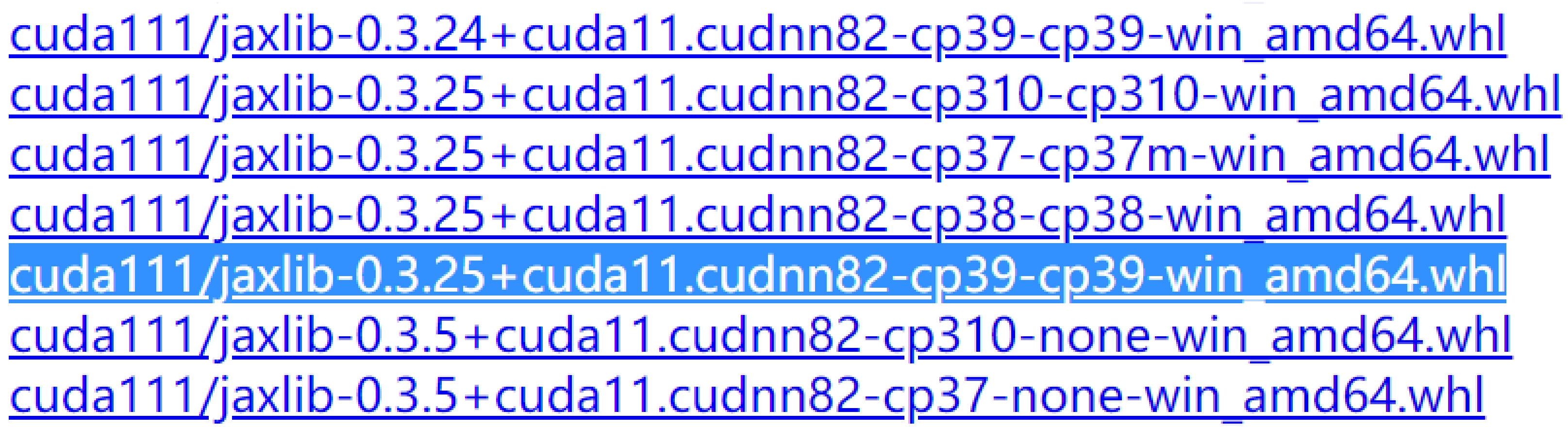
  Execute the following command, no error output indicates the successful installation of JAX and JAXLib:

  ```
  python -c "import jax,jaxlib"
  ```

Installation

- 上面安装的是CPU版本，如果读者的电脑安装有nVidia公司的显卡，则可以考虑安装GPU版本的JAXLib。如果要安装GPU版本，首先要去社区维护的网站上，确认最新版本对应的CUDA与cuDNN的版本：

  If your computer is installed with the GPU card from nVidia, then to install the GPU version of JAXLib might be a better option. If the GPU version is intended to be installed, the first step is to figure out the corresponding CUDA and cuDNN versions of the latest JAXLib release maintained by the community:

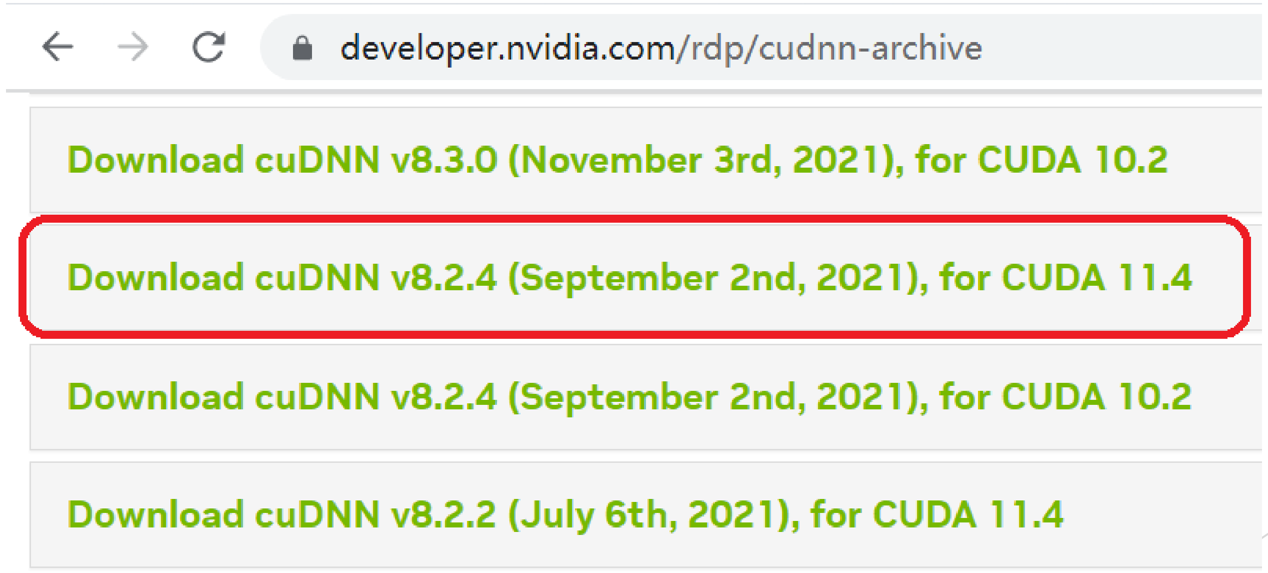


cuda111/jaxlib-0.3.24+cuda11.cudnn82-cp39-cp39-win_amd64.whl
cuda111/jaxlib-0.3.25+cuda11.cudnn82-cp310-cp310-win_amd64.whl
cuda111/jaxlib-0.3.25+cuda11.cudnn82-cp37-cp37m-win_amd64.whl
cuda111/jaxlib-0.3.25+cuda11.cudnn82-cp38-cp38-win_amd64.whl
cuda111/jaxlib-0.3.25+cuda11.cudnn82-cp39-cp39-win_amd64.whl
cuda111/jaxlib-0.3.5+cuda11.cudnn82-cp310-none-win_amd64.whl
cuda111/jaxlib-0.3.5+cuda11.cudnn82-cp37-none-win_amd64.whl

Installation

- 由于当前的最新版本对应的CUDA版本是11。这里11是大版本号，虽然小版本不明确，但是我们可以由cuDNN8.2支持的CUDA11的最大的小版本确定下载哪个合适：

  The latest release requires CUDA version 11. Note here 11 is the major version and we are not sure the minor version. However, from the maximum minor version of CUDA11 supported by cuDNN8.2, which is required by the latest JAXLib release, we can decide on which version of CUDA to download:


developer.nvidia.com/rdp/cudnn-archive

Download cuDNN v8.3.0 (November 3rd, 2021), for CUDA 10.2

Download cuDNN v8.2.4 (September 2nd, 2021), for CUDA 11.4

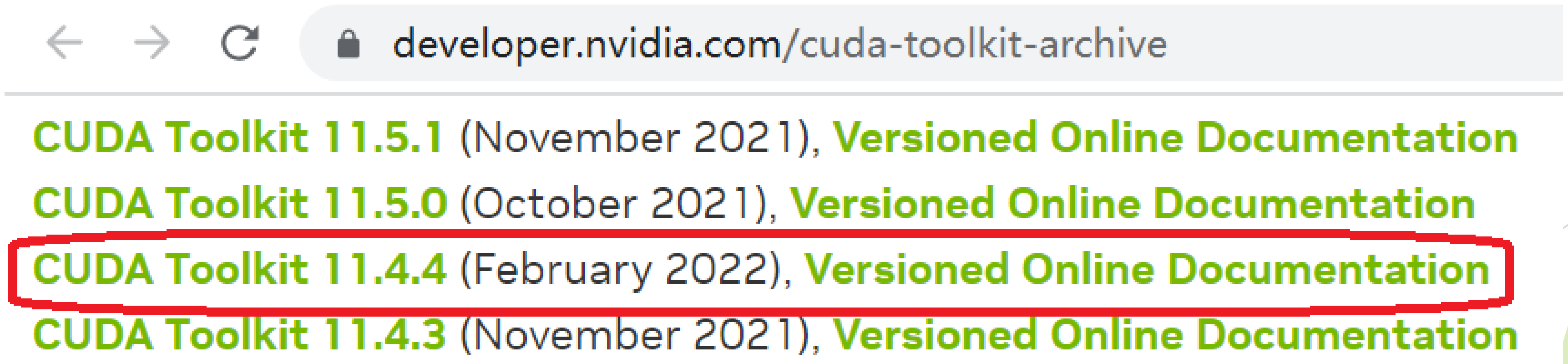
Download cuDNN v8.2.4 (September 2nd, 2021), for CUDA 10.2

Download cuDNN v8.2.2 (July 6th, 2021), for CUDA 11.4

Installation

- 注意：（1）尽管CUDA可以匿名下载，但cuDNN的下载需要注册；（2）从上面截图我们知道，我们可以下载cuDNN的最大版本为8.2.4，其对应的CUDA的版本11.4。通过网站查找比较，CUDA可以下载的最大版本为11.4.4。

  Note, (1) Although CUDA could be downloaded in an anonymously way, nVidia requires registration for downloading cuDNN. (2) From snipped figure in the last slide, we know the maximal valid cuDNN version is 8.2.4, which corresponds to the CUDA version 11.4. After search and compare, it is clear to us the maximal CUDA version is 11.4.4.

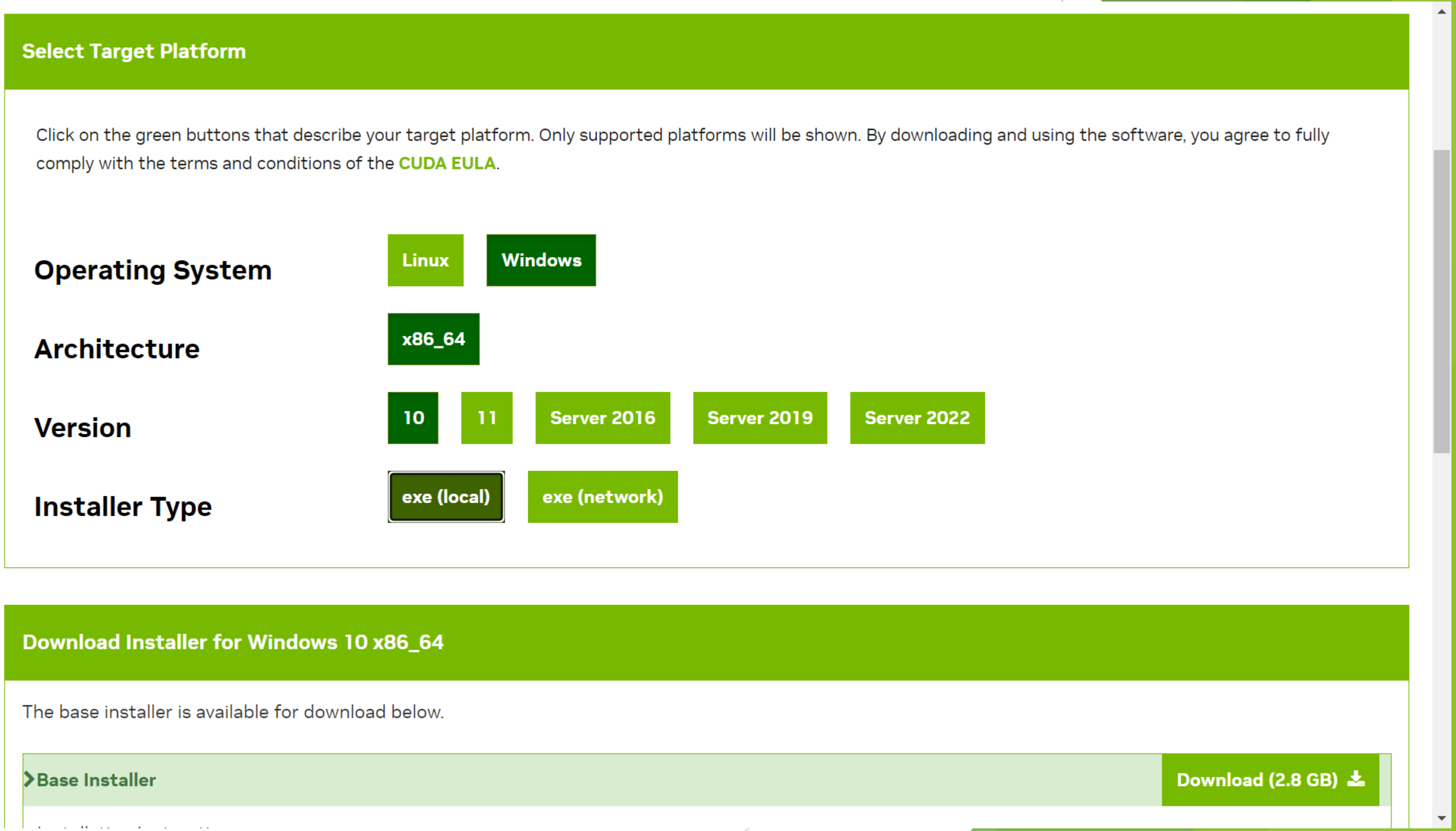

developer.nvidia.com/cuda-toolkit-archive

CUDA Toolkit 11.5.1 (November 2021), Versioned Online Documentation
CUDA Toolkit 11.5.0 (October 2021), Versioned Online Documentation
CUDA Toolkit 11.4.4 (February 2022), Versioned Online Documentation
CUDA Toolkit 11.4.3 (November 2021), Versioned Online Documentation

Installation

- 在下载CUDA或cuDNN时，系统会近可能地提供匹配的选项供读者选择后下载：

  The system will provide the best matched options for users to customize their downloads:



Select Target Platform

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]
Version	[✓ 10] 11 Server 2016 Server 2019 Server 2022
Installer Type	[✓ exe (local)] exe (network)

Download Installer for Windows 10 x86_64

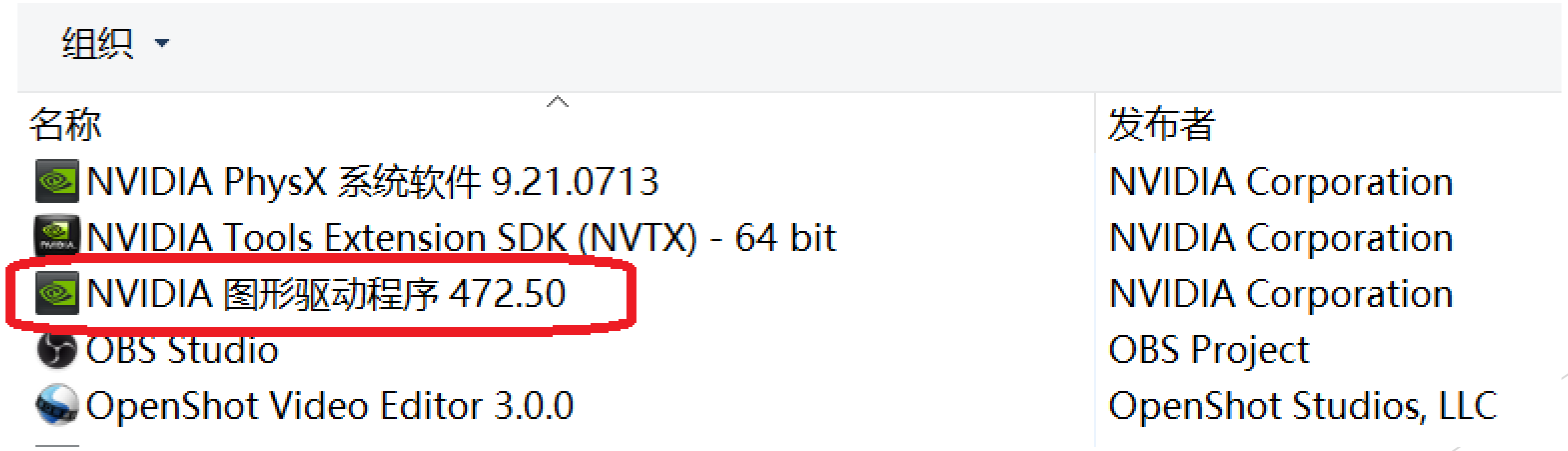
The base installer is available for download below.

Base Installer Download (2.8 GB)

Installation

- 对于CUDA，读者直接选择默认安装就好。注意由于安装CUDA时，需要安装对应的驱动程序。如果显卡的驱动程序比CUDA11.4.4对应的驱动程序要新，则有可能发生安装失败的情况。因此，推荐安装之前将显卡驱动首先卸载掉：

  For CUDA, the user can choose the default options to install it. However, since installing CUDA will simultaneously install the underlying driver, so a installed driver with newer version might incur installation failure. Therefore, it is recommended to uninstall the GPU driver before launch the installation:

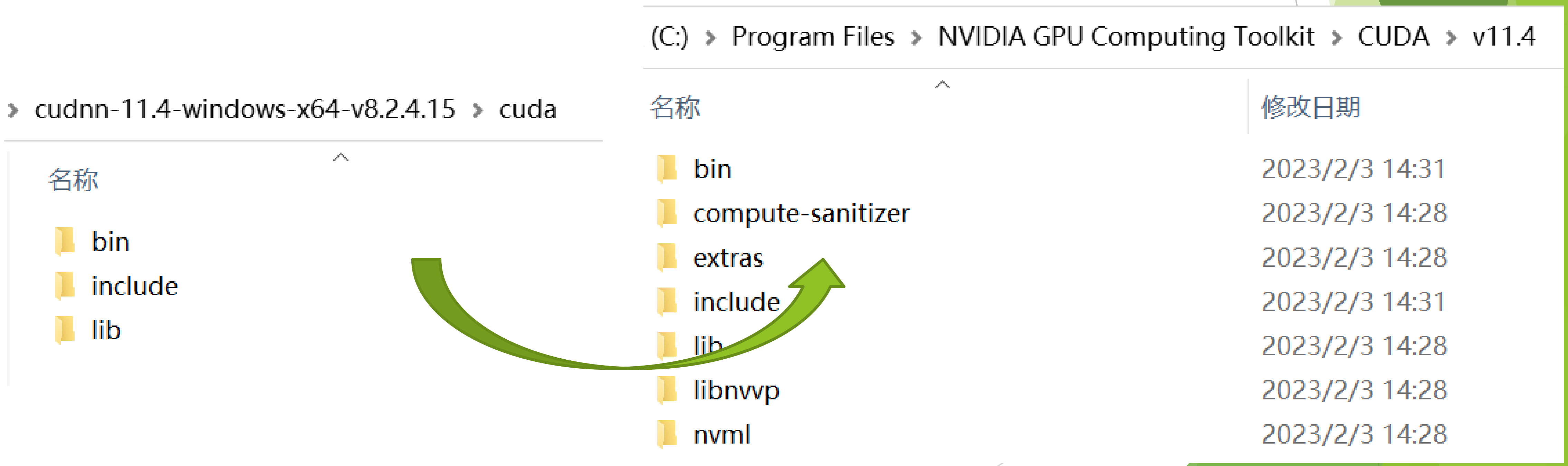


名称	发布者
NVIDIA PhysX 系统软件 9.21.0713	NVIDIA Corporation
NVIDIA Tools Extension SDK (NVTX) - 64 bit	NVIDIA Corporation
NVIDIA 图形驱动程序 472.50	NVIDIA Corporation
OBS Studio	OBS Project
OpenShot Video Editor 3.0.0	OpenShot Studios, LLC

Installation

- 对于cuDNN，并不需要显示安装，只需要下载后解压，将解压后的内容拷贝到对应的CUDA文件夹中即可：

  There is no explicit installation for cuDNN. You only need to unzip the downloaded file and copy the unzipped content to the corresponding folder of CUDA, and that is it:



› cudnn-11.4-windows-x64-v8.2.4.15 › cuda

名称
bin
include
lib

(C:) › Program Files › NVIDIA GPU Computing Toolkit › CUDA › v11.4

名称	修改日期
bin	2023/2/3 14:31
compute-sanitizer	2023/2/3 14:28
extras	2023/2/3 14:28
include	2023/2/3 14:31
lib	2023/2/3 14:28
libnvvp	2023/2/3 14:28
nvml	2023/2/3 14:28

Installation

- 同理，在虚拟环境中安装之前下载的JAXLib安装包，然后安装JAX：

  Installed the downloaded JAXLib in the virtual environment, then JAX:

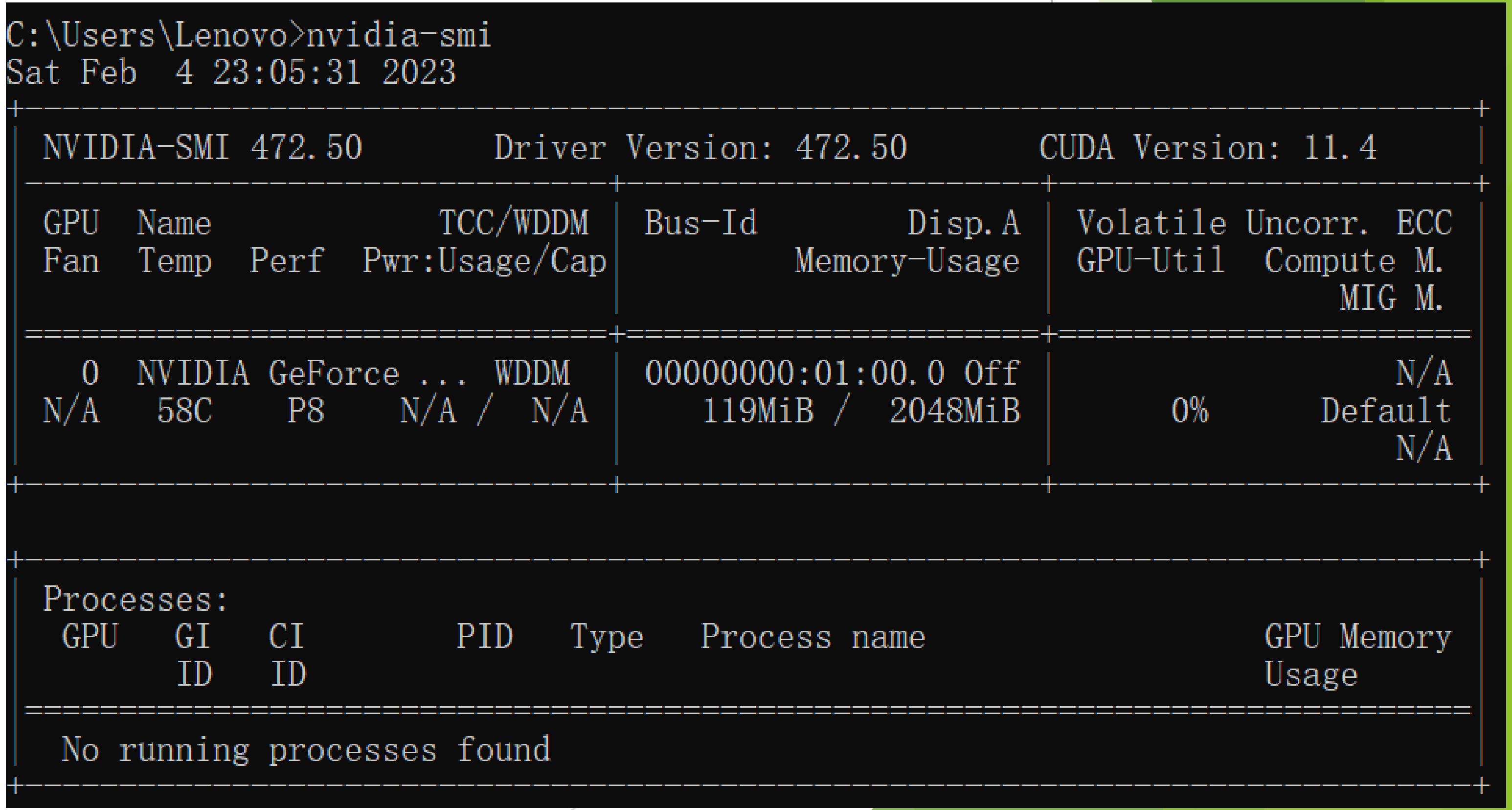
  pip install jaxlib-0.3.25+cuda11.cudnn82-cp39-cp39-win_amd64.whl --use-deprecated legacy-resolver

  pip install jax==0.3.25

- 我们首先测试一下CUDA安装成功：

  We first test the successful installation of CUDA:

  nvidia-smi



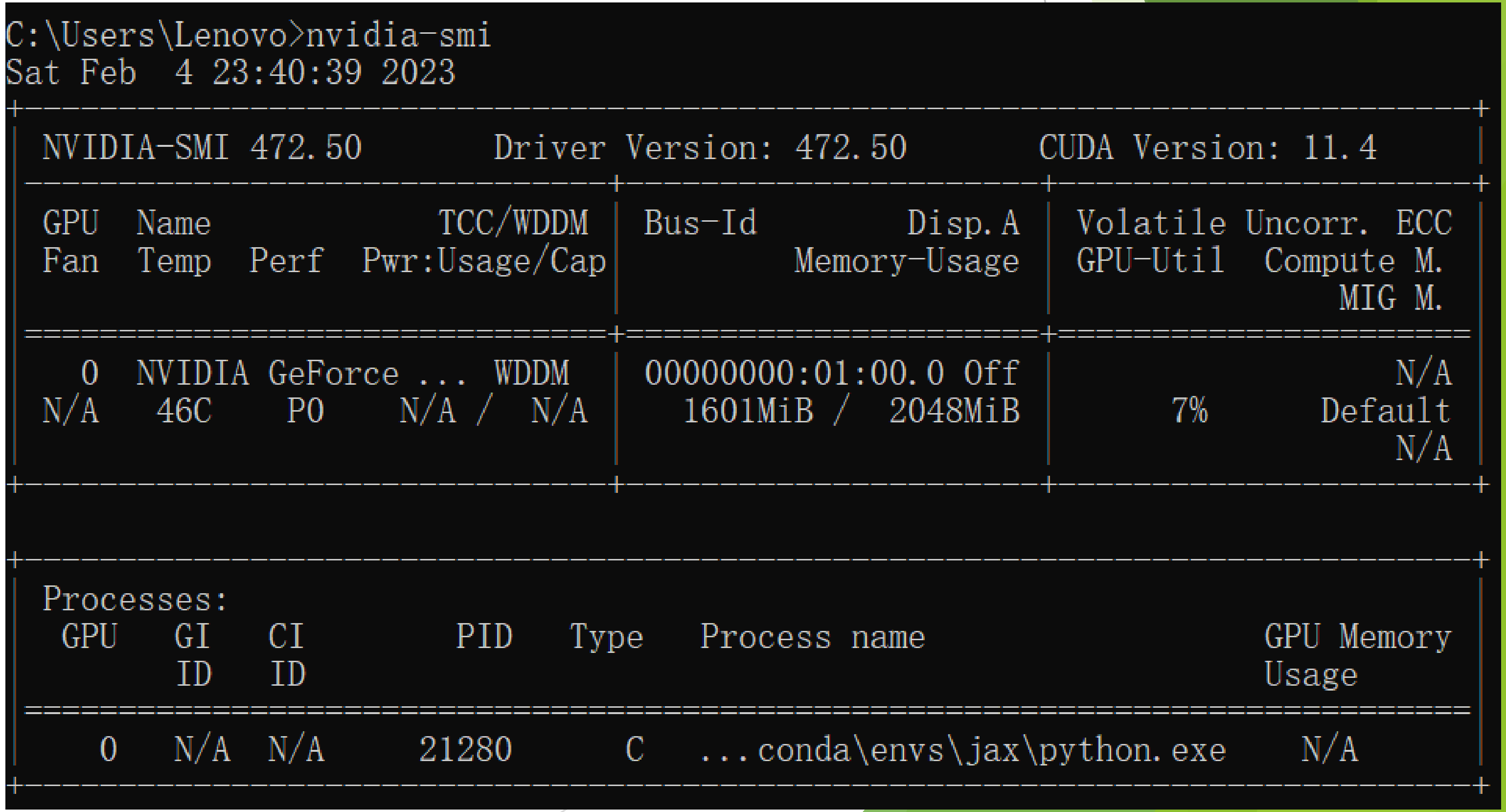
```
C:\Users\Lenovo>nvidia-smi
Sat Feb  4 23:05:31 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 472.50       Driver Version: 472.50       CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ... WDDM  | 00000000:01:00.0 Off |                  N/A |
| N/A   58C    P8    N/A /  N/A |    119MiB /  2048MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Installation

- 执行下面命令，若无报错，则说明JAX及JAXLib安装成功：若打印出结果并且GPU内存使用率上升，则证明python通过JAX库利用GPU在进行计算：

  Execute the following command, no error output indicates the successful installation of JAX and JAXLib. If there are print-outs and meantime the used memory ratio of GPU is increased, it shows the computation of Python code via JAX is backed by GPU:

```
python -c "exec(\"import jax.numpy as jnp\nfor i in range(10000000): print(jnp.sin(float(i)))\")"
```



```
C:\Users\Lenovo>nvidia-smi
Sat Feb  4 23:40:39 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 472.50       Driver Version: 472.50       CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ... WDDM  | 00000000:01:00.0 Off |                  N/A |
| N/A   46C    P0    N/A /  N/A |   1601MiB /  2048MiB |      7%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A     21280      C   ...conda\envs\jax\python.exe    N/A      |
+-----------------------------------------------------------------------------+
```