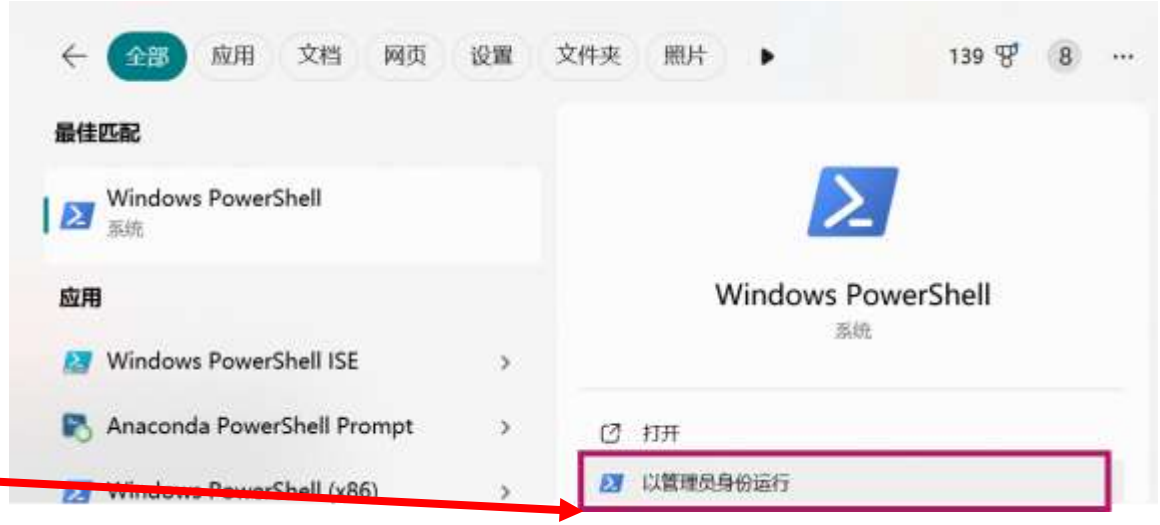
代码运行位置一览

⚠️<代码运行在哪里一览>

1.终端 powershell

2.终端 cmd

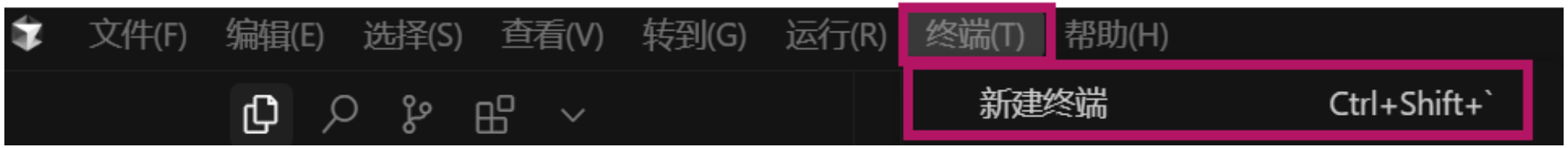
3.x64 Native Tools Command Prompt for VS 2022



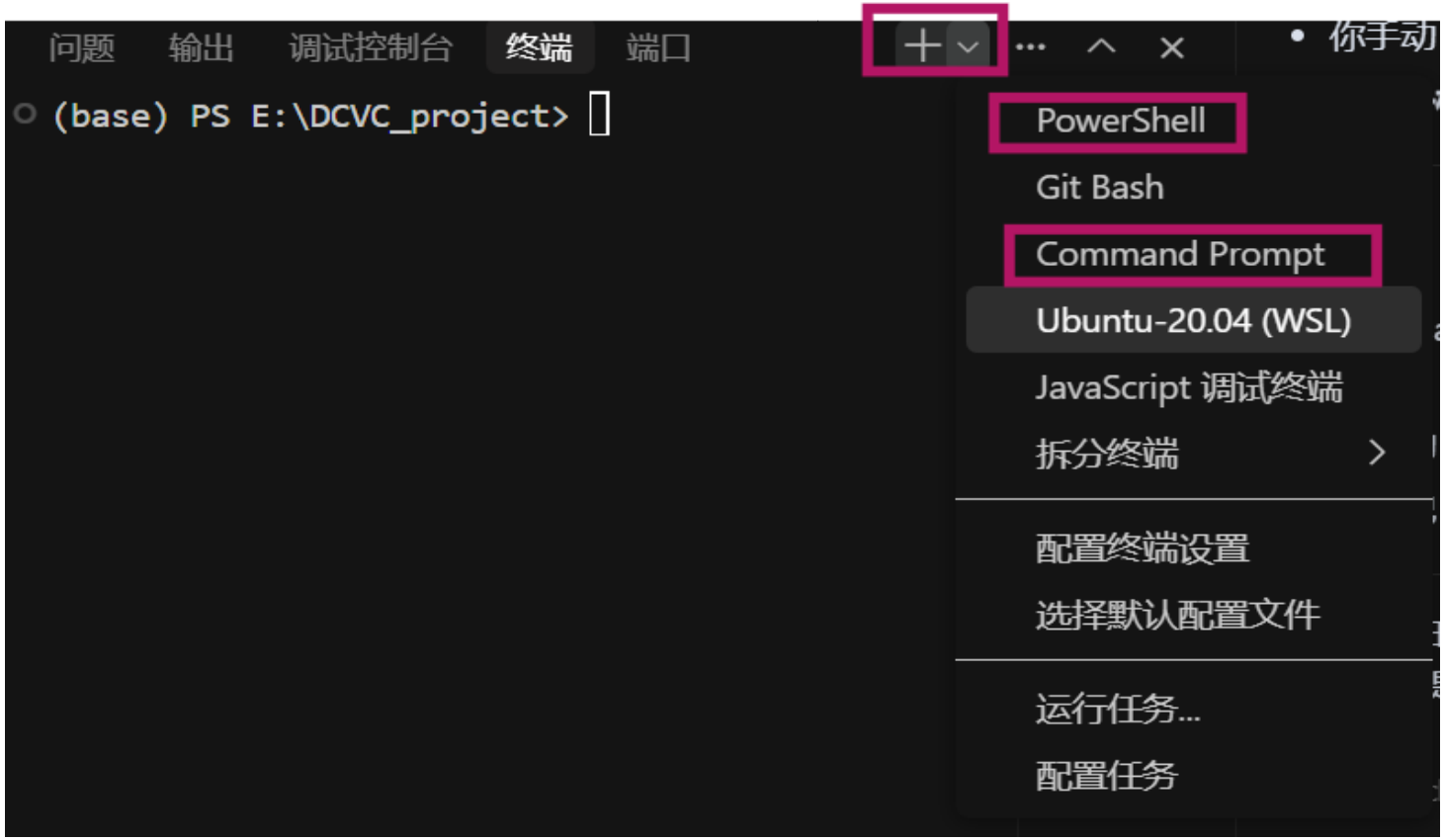
注：x64 Native Tools Command Prompt for VS 2022 只有在安装了VS2022之后才有

如果你有Cursor，cmd和powershell可按照如下方式打开：

方法1：



方法2：



详细安装教程，👀<请严格按照顺序执行>

🍥 *0.Visual Studio 2022(community)*

◆VS2022社区版本，注意在安装时一定要勾选C++桌面开发，默认配置即可。

🍥 *1.CUDA Toolkit-12.6安装*

1.Visual Studio 2022社区版和anconda请自己下载，注意安装路径，两个软件都比较大。：

https://visualstudio.microsoft.com/zh-hans/vs/community/

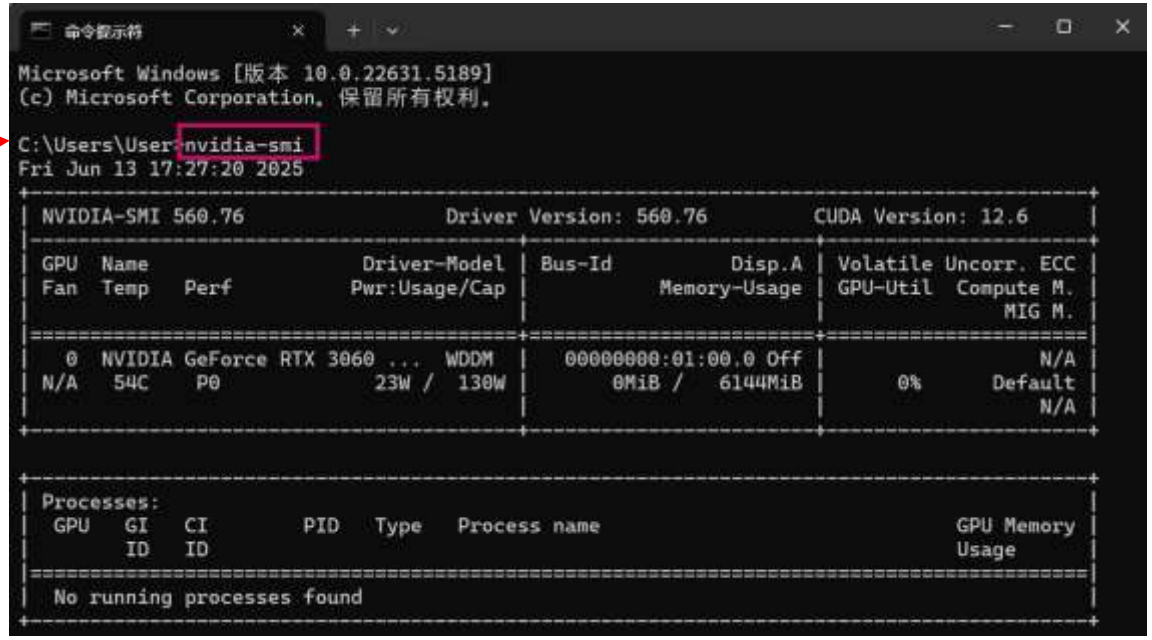
https://www.anaconda.com/

🐳 2. CUDA Toolkit-12.6安装

‼ 1.请先检查自己的电脑是否有GPU:

(1)打开cmd (2)输入nvidia-smi

有结果的情况下询问AI自己的电脑是否支持CUDA-12.6



CUDA Toolkit-12.6下载链接：

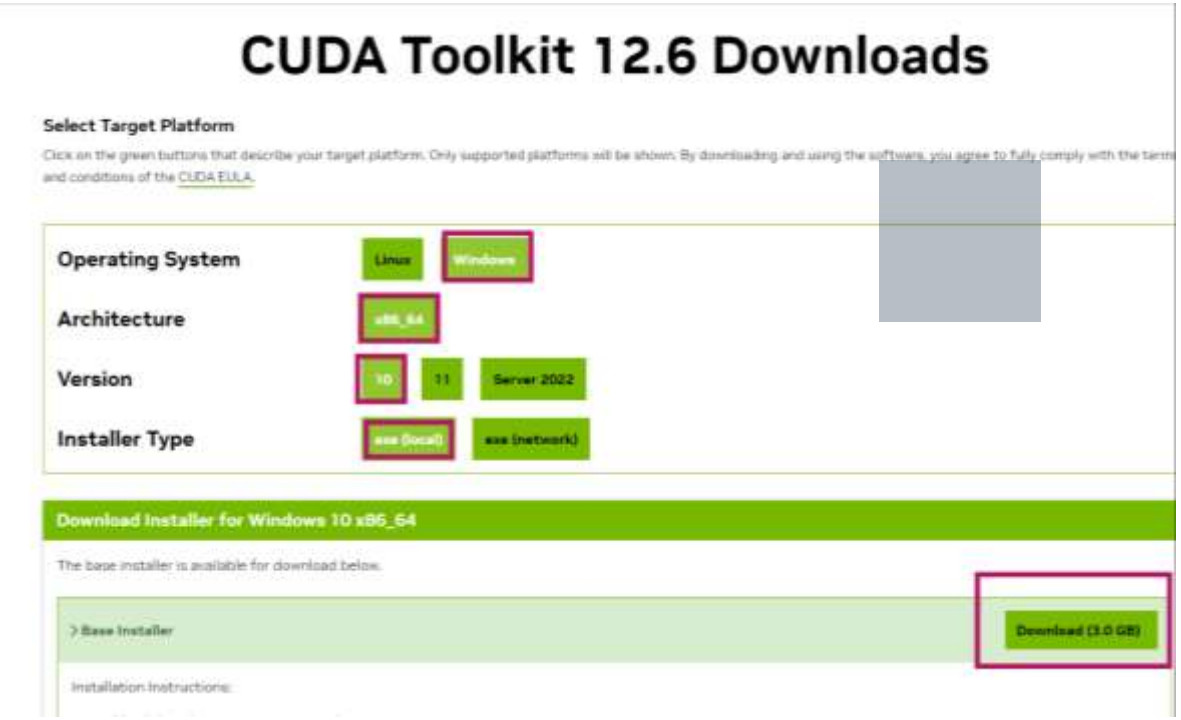
https://developer.nvidia.com/cuda-12-6-0-download-archive

具体下载内容请看下一页

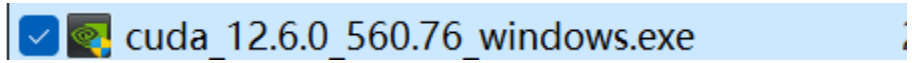
🔔注：这是dcvc官方推荐配置，如果电脑配置允许请尽量遵循本教程。否则会有不知名报错，难以解决。

CUDA Toolkit-12.6下载

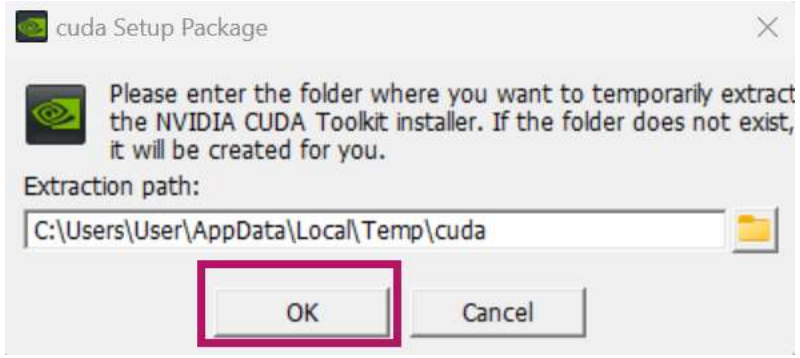
1.请按照如下配置下载



2.下完后，找到下载路径，双击运行此exe文件



3.这里直接点OK,这个是缓存，会自动删除



4.无脑下一步就好。

✳这一步选择自定义可以查看安装位置！

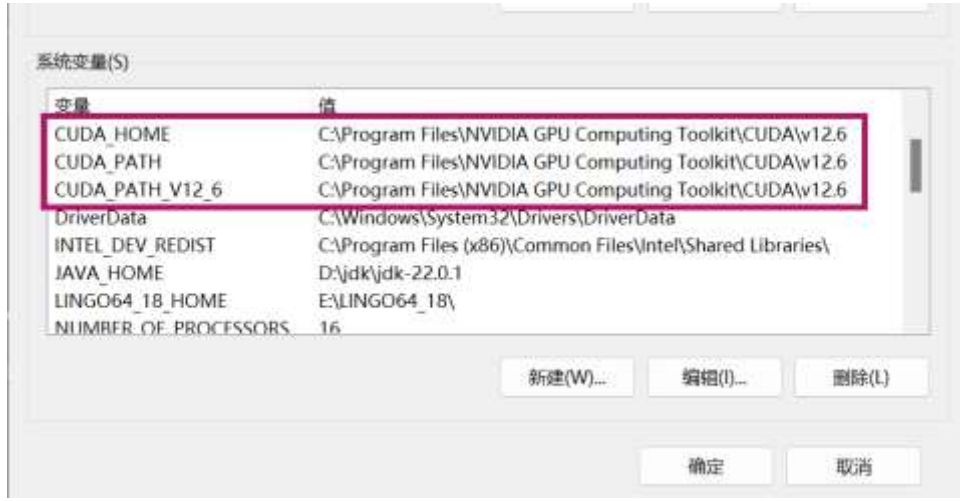
可以查看安装位置后返回安装精简版本！！！

CUDA Toolkit-12.6下载-配置环境变量

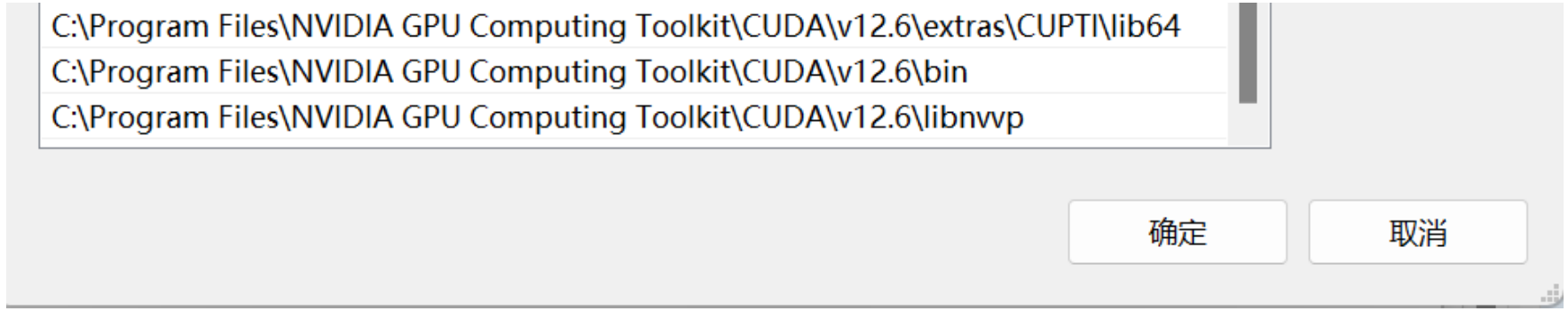
配置系统环境变量：

1.首先请找到自己的安装路径，在上一页第4步骤有写如何查看

2.确保有如下三个系统变量



3.在Path中添加这三个路径：



基于Anconda的dcvc环境配置 (基础)

请确保下载好anconda，<并配置清华大学镜像>,可以去b站找视频教程或问AI。

注：

1.以下内容我会用powershell终端演示，若有cursor或者VS2022/vscode都是一样的

2.环境名字统一为dcvc_0，你可以自定义别的名字

1. 创建环境dcvc_0, python解释器版本为3.12

- **conda create -n dcvc_0 python=3.12**
- **conda activate dcvc_0**

```
(base) PS C:\Users\User> conda activate dcvc_0
(dcvc_0) PS C:\Users\User>
```

注：命令行前面显示(dcvc_0)就表示环境激活成功了，后面的每次代码几乎都要在这个环境下运行！！！每次打开新项目新终端都要activate环境！！！

```
Windows PowerShell
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

安装最新的 PowerShell，了解新功能和改进！https://aka.ms/PSWindows

(base) PS C:\Users\User> conda create -n dcvc_0 python=3.12
Retrieving notices: done
Channels:
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
 - defaults
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/fastai
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/pytorch
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/bioconda
Collecting package metadata (repodata.json): / Retrying (Retry(total=2, connect=None, read=None, red
irect=None, status=None)) after connection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='re
po.anaconda.com', port=443): Read timed out. (read timeout=9.15)")': /pkgs/r/win-64/repodata.json.zs
t

done
Solving environment: done

## Package Plan ##

  environment location: D:\Anaconda\envs\dcvc_0

  added / updated specs:
    - python=3.12
done
#
# To activate this environment, use
#
#     $ conda activate dcvc_0
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) PS C:\Users\User>
```

清华大学镜像配置成功！！

环境创建成功！！

基于Anconda的dcvc环境配置 (基础)-pytorch安装

1.运行如下命令：

- pip install torch==2.6.0+cu126 torchvision==0.21.0+cu126 torchaudio==2.6.0+cu126 --index-url https://download.pytorch.org/whl/cu126

注：应该有一个2-3G的下载，因为本人有下载缓存，所以没有显示

<检测Cuda12.6+pytorch2.6.0是否安装成功>

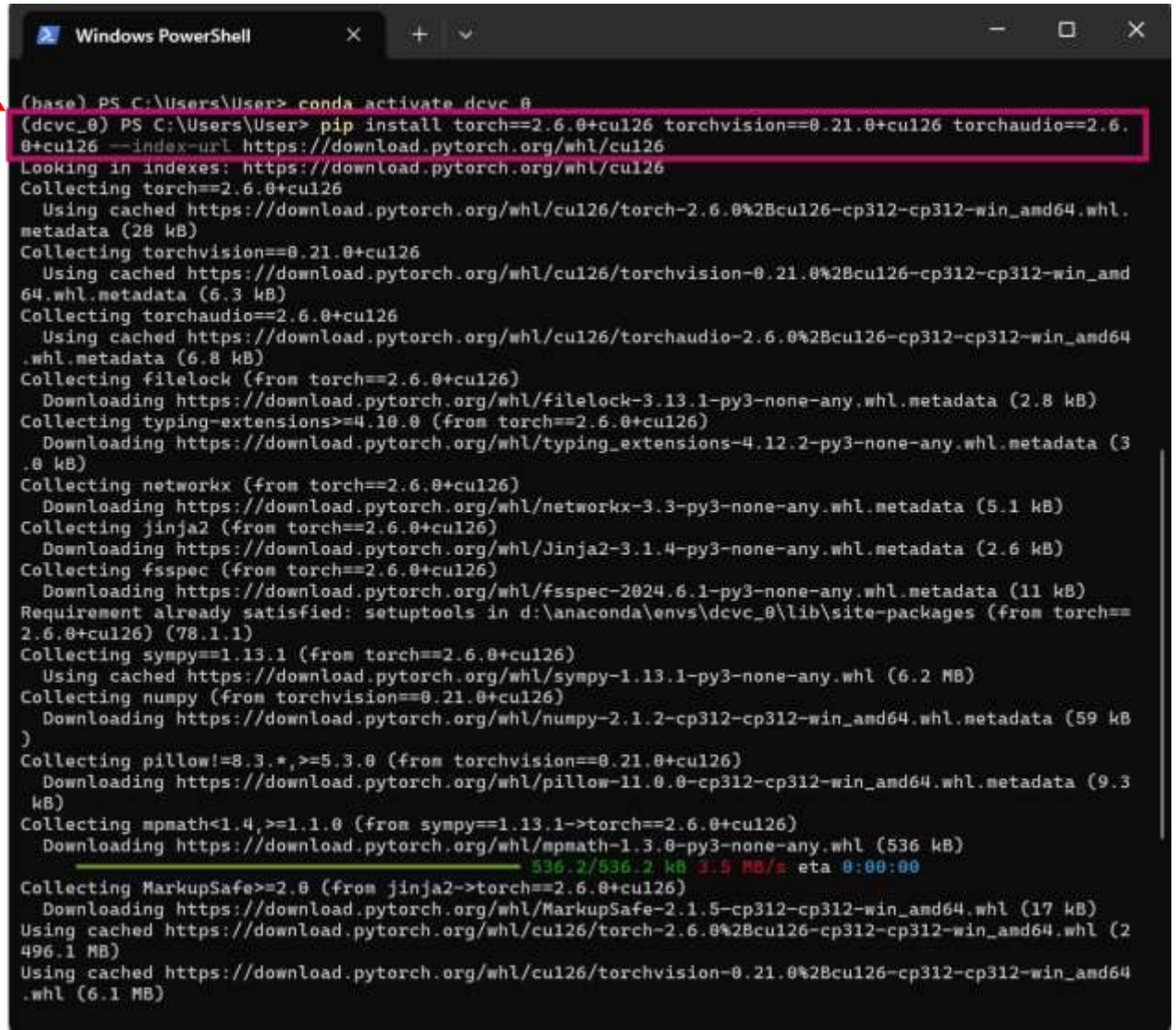
(1)创建一个py文件，在dcvc_0下运行如下内容。

```
import torch
import torchvision
import torchaudio
print(torch.__version__)
print(torchvision.__version__)
print(torchaudio.__version__)
print(torch.version.cuda)
```

注：随机打开一个项目创建一个py文件即可！注意要激活环境

- conda activate dcvc_0

(2) 创建py文件后，假设为1.py，cd到py文件所在的路径，然后使用< python 1.py >运行。详情参考下一页！



基于Anconda的dcvc环境配置 (基础)-检测CUDA 12.6+pytorch2.6是否安装成功

假设在E:\DCVC_project\farm_monitoring创建1.py
所以1.py真实路径为：E:\DCVC_project\farm_monitoring\1.py，复制如下内容

```
import torch
import torchvision
import torchaudio
print(torch.__version__)
print(torchvision.__version__)
print(torchaudio.__version__)
print(torch.version.cuda)
```

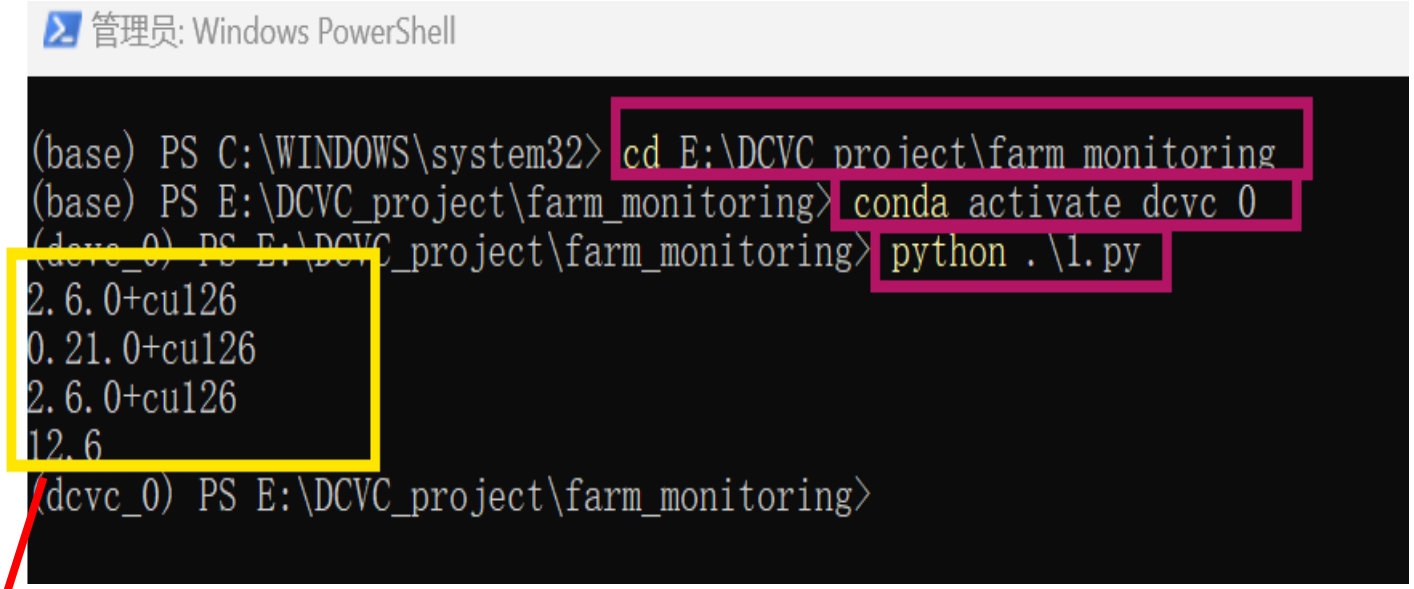
1.导航到E:\DCVC_project\farm_monitoring
powershell下是：
- **cd E:\DCVC_project\farm_monitoring**

cmd下是：

- **cd /d E:\DCVC_project\farm_monitoring**

2.激活环境（如果激活了请忽略）
- **conda activate dcvc_0**

3.运行1.py
- **python 1.py**



🎉 出现这个就表示成功了

😸基于Anconda的dcvc环境配置 (基础)-cmake,g++,ninja安装

1.Cmake

https://cmake.org/download/

Binary distributions:

Platform	Files
Windows x64 Installer:	cmake-4.0.2-windows-x86_64.msi
Windows x64 ZIP	cmake-4.0.2-windows-x86_64.zip
Windows i386 Installer:	cmake-4.0.2-windows-i386.msi
Windows i386 ZIP	cmake-4.0.2-windows-i386.zip
Windows ARM64 Installer:	cmake-4.0.2-windows-arm64.msi
Windows ARM64 ZIP	cmake-4.0.2-windows-arm64.zip
macOS 10.13 or later	cmake-4.0.2-macos-universal.dmg

2.ninja

https://github.com/ninja-build/ninja/releases

▾ Assets 7

ninja-linux-aarch64.zip	119 KB	May 11, 2024
ninja-linux.zip	120 KB	May 11, 2024
ninja-mac.zip	275 KB	May 11, 2024
ninja-win.zip	269 KB	May 11, 2024
ninja-winarm64.zip	248 KB	May 11, 2024
Source code (zip)		May 11, 2024
Source code (tar.gz)		May 11, 2024

3.g++

https://www.winlibs.com/

Help! I don't know which download to choose!

Don't worry. For most purposes the latest Windows 64-bit release version with MSVCRT runtime and POSIX threads is a good choice.

Download it here.

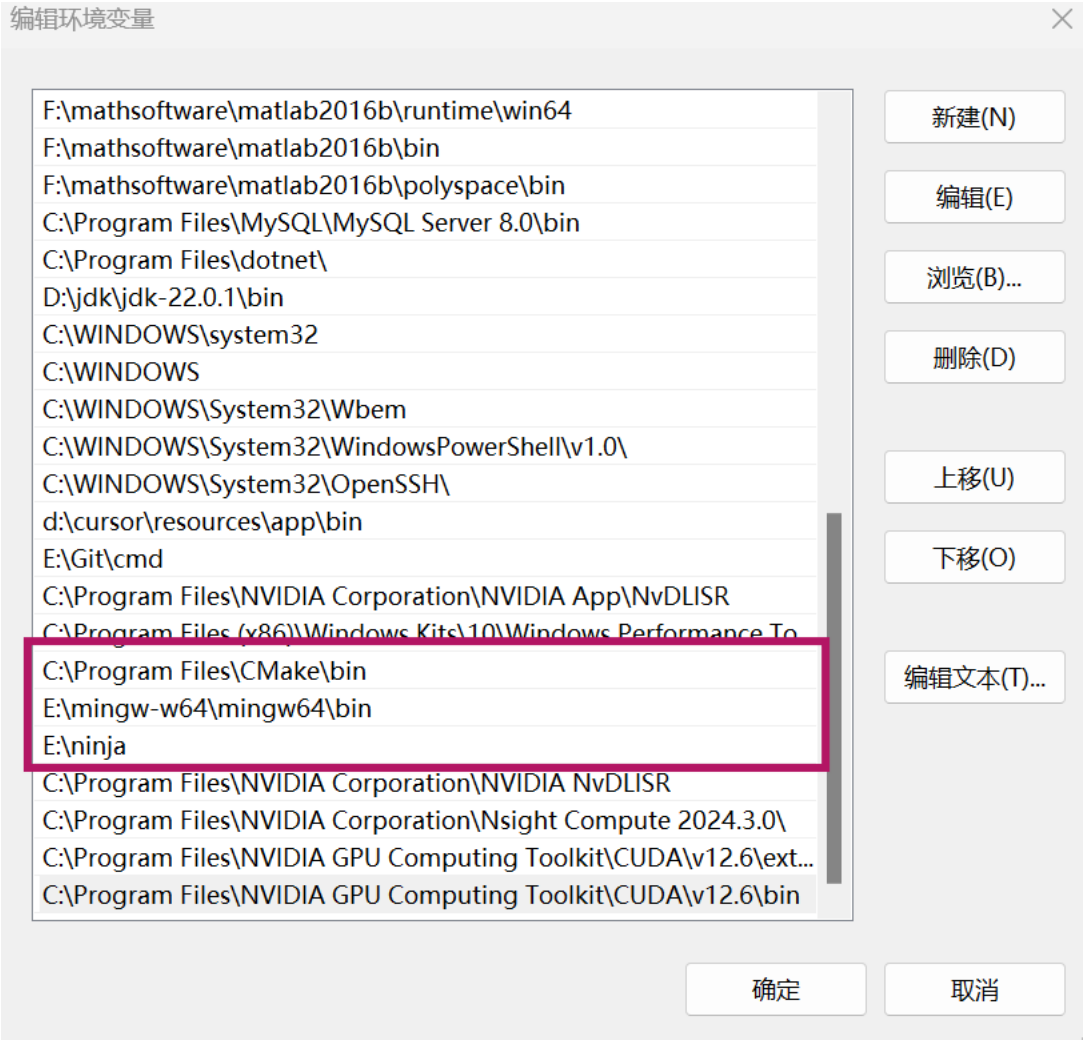
Release versions

UCRT runtime

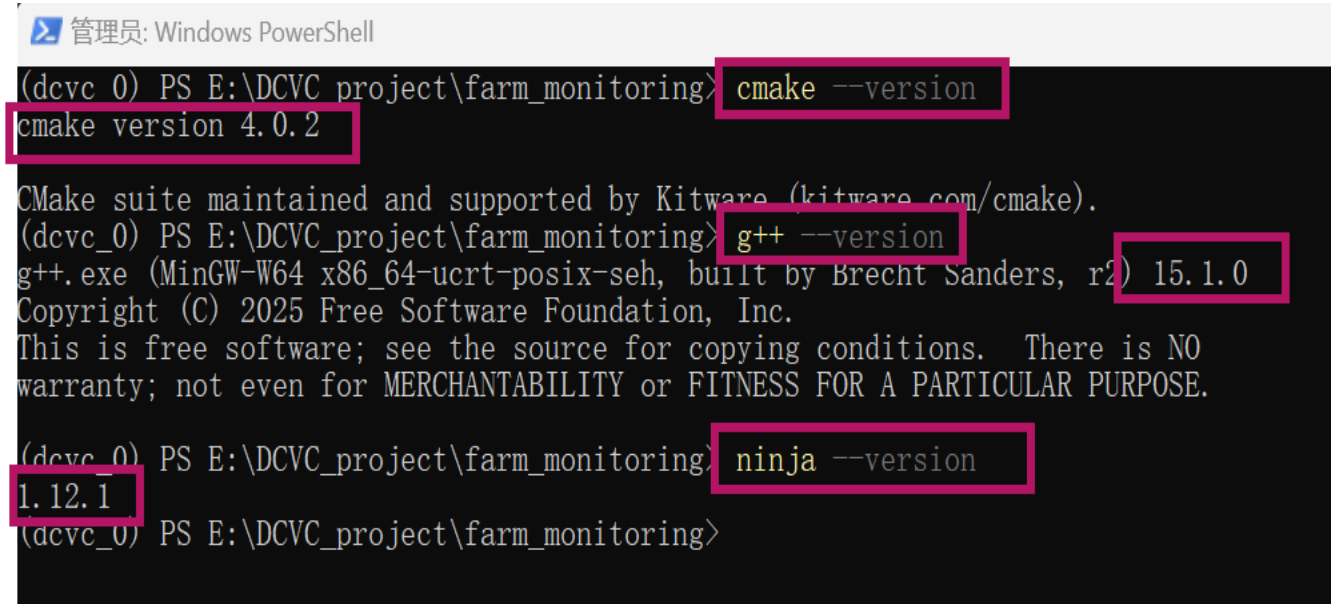
- GCC 15.1.0 (with **POSIX** threads) + MinGW-w64 13.0.0 UCRT - release 2 **(LATEST)**
  - Win32 (without LLVM/Clang/LLD/LLDB): 7-Zip archive* | Zip archive
  - Win64 (without LLVM/Clang/LLD/LLDB): 7-Zip archive* | Zip archive
- GCC 15.1.0 (with **POSIX** threads) + MinGW-w64 12.0.0 UCRT - release 1
  - Win32 (without LLVM/Clang/LLD/LLDB): 7-Zip archive* | Zip archive
  - Win64 (without LLVM/Clang/LLD/LLDB): 7-Zip archive* | Zip archive
- GCC 15.1.0 (with **MCF** threads) + MinGW-w64 12.0.0 UCRT - release 1
  - Win32 (without LLVM/Clang/LLD/LLDB): 7-Zip archive* | Zip archive
  - Win64 (without LLVM/Clang/LLD/LLDB): 7-Zip archive* | Zip archive
- GCC 14.3.0 (with **POSIX** threads) + MinGW-w64 12.0.0 UCRT - release 1
  - Win32 (without LLVM/Clang/LLD/LLDB): 7-Zip archive* | Zip archive
  - Win64 (without LLVM/Clang/LLD/LLDB): 7-Zip archive* | Zip archive
- GCC 14.2.0 (with **POSIX** threads) + LLVM/Clang/LLD/LLDB 19.1.7 + MinGW-w64 12.0.0 UCRT - release 3
  - Win32: 7-Zip archive* | Zip archive - without LLVM/Clang/LLD/LLDB: 7-Zip archive* | Zip archive
  - Win64: 7-Zip archive* | Zip archive - without LLVM/Clang/LLD/LLDB: 7-Zip archive* | Zip archive
- GCC 14.2.0 (with **POSIX** threads) + LLVM/Clang/LLD/LLDB 19.1.1 + MinGW-w64 12.0.0 UCRT - release 2
  - Win32: 7-Zip archive* | Zip archive - without LLVM/Clang/LLD/LLDB: 7-Zip archive* | Zip archive
  - Win64: 7-Zip archive* | Zip archive - without LLVM/Clang/LLD/LLDB: 7-Zip archive* | Zip archive

基于Anconda的dcvc环境配置 (基础)-cmake,g++,ninja安装验证

1.检查PATH中是否有如下三个！！



2.运行如下命令，如果输出版本号，则说明都安装成功并加入环境变量



🐳基于Anconda的dcvc环境配置 (基础)-requitements等

1.请尽量选择我们的版本🤗，也就是DCVC在E:\DCVC_project\farm_monitoring\external\DCVC的版本
链接：
优势：
- 🔶这个版本有VTM的使用，可以帮助理解视频压缩。
- 🔶本教程基于此版本。包括更详细的data配置，更完整的项目框架（后续待加入目标检测等）。

2.如果不考虑项目完整性，单纯想测试DCVC，也可以，注意路径搭配等，
具体查询：https://github.com/microsoft/DCVC/tree/main **查看README文件，里面非常详细。要说和我们版本的区别就是路径和大框架而已。咨询AI 解答**

>继续基于我们版本的安装：

🧠**接下来我们需要使用：**x64 Native Tools Command Prompt for VS 2022
原因：简单的说，有些dcvc需要的组件在VS2022里，所以需要用VS2022的终端环境。这个说法并不十分准确，可以详询AI。

有两种方法供选择：
1.在激活conda环境时同时激活x64 Native Tools Command Prompt for VS 2022 (推荐) ☑️
推荐原因：十分方便，没别的原因。

虽然但是，我们会先进行以下的教学，原因：方便理解我们在干什么。
2.直接在windows打开x64 Native Tools Command Prompt for VS 2022

🥚 x64 Native Tools Command Prompt for VS 2022 下安装 requitements等

1.打开x64 Native Tools Command Prompt for VS 2022

没有就是没有VS2022

2.激活环境，并导航到我们的项目目录

注：请通人性的变换路径。
本教程根路径为： E:\DCVC_project
本教程DCVC路径为： farm_monitoring\external\DCVC

- conda activate dcvc_0
- cd /d E:\DCVC_project\farm_monitoring\external\DCVC

📌 ps:x64 Native Tools Command Prompt for VS 2022是cmd类，所以cd要加/d

3.安装依赖：

路径为：E:\DCVC_project\farm_monitoring\external\DCVC\requirements.txt

- pip install -r requirements.txt

```
**********************************************************************
** Visual Studio 2022 Developer Command Prompt v17.14.4
** Copyright (c) 2025 Microsoft Corporation
**********************************************************************
[vcvarsall.bat] Environment initialized for: 'x64'

C:\Windows\System32>conda activate dcvc_0

(dcvc_0) C:\Windows\System32>
(dcvc_0) C:\Windows\System32>cd /d E:\DCVC_project\farm_monitoring\external\DCVC

(dcvc_0) E:\DCVC_project\farm_monitoring\external\DCVC>pip install -r requirements.txt
Requirement already satisfied: numpy>=1.20.0 in d:\anaconda\envs\dcvc_0\lib\site-packages (from -r requirements.txt (line 1)) (2.1.2)
Collecting scipy (from -r requirements.txt (line 2))
  Downloading scipy-1.15.3-cp312-cp312-win_amd64.whl.metadata (60 kB)
Collecting matplotlib (from -r requirements.txt (line 3))
```

🎉 成功！有报错那就AI吧！

管理员: x64 Native Tool...

```
━ 2.2/2.2 MB 3.0 MB/s eta 0:00:00
Downloading kiwisolver-1.4.8-cp312-cp312-win_amd64.whl (71 kB)
Downloading packaging-25.0-py3-none-any.whl (66 kB)
Downloading pyparsing-3.2.3-py3-none-any.whl (111 kB)
Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
Downloading six-1.17.0-py2.py3-none-any.whl (11 kB)
Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: six, scipy, pyparsing, pybind11, packaging, kiwisolver, fonttools, cycler, contourpy, colorama, tqdm, python-dateutil, bd-metric, matplotlib
Successfully installed bd-metric-0.9.0 colorama-0.4.6 contourpy-1.3.2 cycler-0.12.1 fonttools-4.58.3 kiwisolver-1.4.8 matplotlib-3.10.3 packaging-25.0 pybind11-2.13.6 pyparsing-3.2.3 python-dateutil-2.9.0.post0 scipy-1.15.3 six-1.17.0 tqdm-4.67.1
```

🐣x64 Native Tools Command Prompt for VS 2022 下安装 requitements等

安装剩余两个依赖(这里非常容易报错，多多咨询AI吧！实在不行咨询我)

(1)第一个依赖，路径：E:\DCVC_project\farm_monitoring\external\DCVC\src\cpp，本质就是路径下的setup.py文件

- cd ./src/cpp/
- pip install .

```
(dcvc_0) E:\DCVC_project\farm_monitoring\external\DCVC>cd ./src/cpp/

(dcvc_0) E:\DCVC_project\farm_monitoring\external\DCVC\src\cpp>pip install .
Processing e:\dcvc_project\farm_monitoring\external\dcvc\src\cpp
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: MLCodec_extensions_cpp
  DEPRECATION: Building 'MLCodec_extensions_cpp' using the legacy setup.py bdist_wheel mechanism, which will be removed
in a future version. pip 25.3 will enforce this behaviour change. A possible replacement is to use the standardized buil
d interface by setting the `--use-pep517` option, (possibly combined with `--no-build-isolation`), or adding a `pyproje
t.toml` file to the source tree of 'MLCodec_extensions_cpp'. Discussion can be found at https://github.com/pypa/pip/issu
es/6334
  Building wheel for MLCodec_extensions_cpp (setup.py) ... done
  Created wheel for MLCodec_extensions_cpp: filename=mlcodec_extensions_cpp-0.0.0-cp312-cp312-win_amd64.whl size=93970 s
ha256=2fd6ee14a426e6ce27a3086a0ca25b07390799b716385a11f556e145d3faa0d2
  Stored in directory: C:\Users\User\AppData\Local\Temp\pip-ephem-wheel-cache-5_n5al5q\wheels\39\02\87\3e0da9ad3409d968
054fd530fb79ba6183fe7e38be2da126c
Successfully built MLCodec_extensions_cpp
Installing collected packages: MLCodec_extensions_cpp
Successfully installed MLCodec_extensions_cpp-0.0.0
```

(1)第二个依赖，路径： E:\DCVC_project\farm_monitoring\external\DCVC\src\layers\extensions\inference

如果你刚安装完第一个依赖：

- cd ..
- cd layers/extensions/inference
- set DISTUTILS_USE_SDK=1
- pip install .

>详情见第二页

🐤x64 Native Tools Command Prompt for VS 2022 下安装 requitements等依赖

>安装第二个依赖详情:

管理员: x64 Native Tools Command Prompt for VS 2022 - "D:\Anaconda\condabin\conda.bat" activate dcvc_0

```
(dcvc_0) E:\DCVC_project\farm_monitoring\external\DCVC\src\cpp>cd ..

(dcvc_0) E:\DCVC_project\farm_monitoring\external\DCVC\src>cd layers/extensions/inference

(dcvc_0) E:\DCVC_project\farm_monitoring\external\DCVC\src\layers\extensions\inference>set DISTUTILS_USE_SDK=1

(dcvc_0) E:\DCVC_project\farm_monitoring\external\DCVC\src\layers\extensions\inference>pip install .
Processing e:\dcvc_project\farm_monitoring\external\dcvc\src\layers\extensions\inference
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: inference_extensions_cuda
  DEPRECATION: Building 'inference_extensions_cuda' using the legacy setup.py bdist_wheel mechanism, which will be
removed in a future version. pip 25.3 will enforce this behaviour change. A possible replacement is to use the stan
dardized build interface by setting the `--use-pep517` option, (possibly combined with `--no-build-isolation`), or
adding a `pyproject.toml` file to the source tree of 'inference_extensions_cuda'. Discussion can be found at https:
//github.com/pypa/pip/issues/6334
  Building wheel for inference_extensions_cuda (setup.py) ... done
  Created wheel for inference_extensions_cuda: filename=inference_extensions_cuda-0.0.0-cp312-cp312-win_amd64.whl s
ize=231948 sha256=42273217c6baac25a126d396838572ee799b288bbfc78b59560680af8ce5d8dd
  Stored in directory: C:\Users\User\AppData\Local\Temp\pip-ephem-wheel-cache-w5or6xb8\wheels\fd\64\73\9e0c932f923e
e64dc783d1a109a16af1b4f12488271c63c641
Successfully built inference_extensions_cuda
Installing collected packages: inference_extensions_cuda
Successfully installed inference_extensions_cuda-0.0.0

(dcvc_0) E:\DCVC_project\farm_monitoring\external\DCVC\src\layers\extensions\inference>
```

如果没有报错就是成功，太棒了，你已经成功安装好所有依赖🎉

本教程经过测试，严格按照流程来没有任何问题，如果最后一个依赖出现问题，尝试:

- 🔹 set CUDA_HOME=C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.6
- 🔹 set DISTUTILS_USE_SDK=1

🐤conda环境自动激活x64 Native Tools Command Prompt for VS 2022

这一路走来真的不容易，所以请完成最后一个配置：在我们的conda环境下自动激活x64 Native Tools Command Prompt for VS 2022 ，实现一劳永逸🐤

如何在当前终端（比如 Anaconda Prompt、PowerShell、CMD）下，自动获得 VS 2022 Developer Command Prompt 的编译环境，而不用每次手动打开专用窗口？

🧠 -VS Developer Command Prompt 的本质，就是**先运行一个批处理脚本（如 `vcvars64.bat`）来设置一堆环境变量**，让 cl.exe、link.exe、rc.exe、nmake 等编译工具能被系统找到。

2. 操作方法

A. 找到 `vcvars64.bat` 的路径