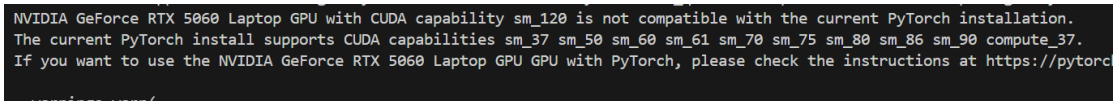
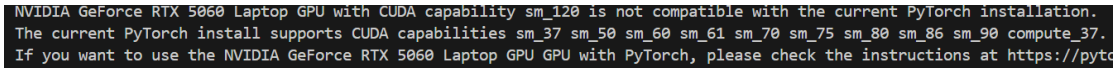
记录一个有趣的 bug

CUDA 和 pytorch 版本不匹配



我理解的 PyTorch，CUDA，GPU 版本关系，以及常见问题处理方法（需要的代码放在此，CUDAtest.py）：

1. PyTorch 下载： PyTorch 是用某个版本的 CUDA 工具链（CUDA Toolkit）编译出来的，在下载时可以选择（“compute platform”，比如当前的 CUDA12.6，12.8，13.0，13.0 是目前最高版本）
   当你使用
   （python）import torch；print(torch.__version__)
   （cmd）python -c "import torch; print(torch.__version__)"
   查询 PyTorch 版本号时，输出会返回主版本号和编译用的 CUDA 版本号。
   示例：
   2.3.1+cu121 表示此版本是用 CUDA12.1 编译的。
2. 为什么你不需要下载 CUDA Toolkit？这同时涉及“1”中的一个疑问：用 CUDA 编译出来什么意思？
   当你通过 pip 安装 PyTorch 时，你安装的是 **PyTorch 官方预编译好的二进制包**，它已经内置了运行所需的 CUDA 库（如 cuBLAS、cuDNN）不依赖你本地是否安装了 CUDA Toolkit。
   而回到 1，PyTorch 官方团队用 CUDA Toolkit 12.1 编译时，他们链接了 CUDA 12.1 的库，这也是 CUDA 版本号的含义。
3. PyTorch 支持的 CUDA 算力 compute capability，GPU 对应的 CUDA 算力



a. 什么是 CUDA 算力？

Compute Capability 是 NVIDIA 为每一代 GPU 架构定义的版本号，格式为 X.Y（如 7.5、8.9、12.0），用于标识该 GPU 的：

- 流处理器（SM）结构
- 支持的指令集
- 共享内存/寄存器容量
- 最大线程数等硬件特性

在 PyTorch 中通常写作 sm_xx，例如我的 RTX 5060 Laptop，算力为 sm_120

b. PyTorch 需要特定算力

PyTorch 的底层计算（如矩阵乘、卷积）依赖 **CUDA C++ 编写的高性能 kernel**。这些 kernel 必须：

1. 针对目标 GPU 架构编译（因为不同 SM 结构差异大）
2. 使用对应架构支持的指令

看不懂没关系，知道每个特定版本的 PyTorch 依赖特定的 SM 结构，只能对应一个范围的 sm_即可

c. PyTorch 支持的算力范围**完全取决于两个因素：**

1：编译时使用的 CUDA Toolkit 版本

- CUDA Toolkit 中的编译器 `nvcc` 只支持到某个最大算力
- 例如：

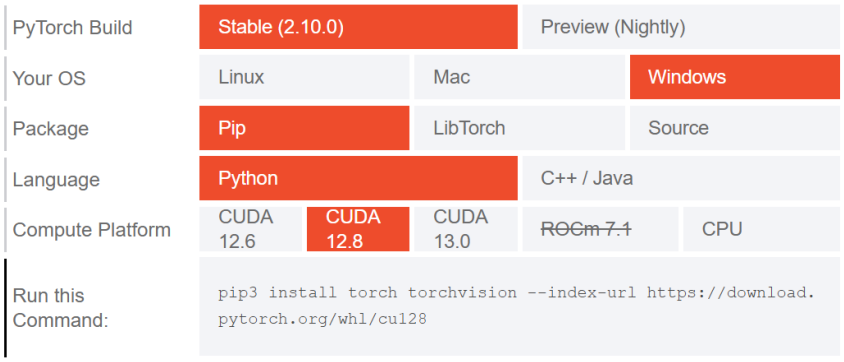
表格

CUDA Toolkit	最高支持的 Compute Capability
11.8	8.6（Ampere）
12.1	9.0（Ada Lovelace）
12.8+	12.0（Blackwell）

PyTorch 无法支持比其编译所用 CUDA Toolkit 更高的算力。

2：编译时显式指定的架构列表,这个主要是团队的事情

因此，让我们回到本节初始的案例：目前我们的 pytorch 版本只支持 sm_37 到 sm_90，而我们的显卡算力为 sm_120，因此需要更新 PyTorch——使用更高的 CUDA 算力版本，查表知至少需 12.8，在 PyTorch 官网选择合适的下载即可。



(11 封私信 / 16 条消息) 如何解决 PyTorch 版本和 CUDA 版本不匹配的关系 - 知乎

详细介绍：显卡算力过高导致 PyTorch 不兼容的救赎指南 - yangykaifa - 博客园

我电脑目前的 pytorch 版本：2.7.1+cu118（主版本号+用 CUDA 11.8 编译的）

sm_ 是 **Streaming Multiprocessor** 的缩写，它是 **NVIDIA GPU 架构中的核心计算单元**。而 sm_xx（如 sm_75、sm_86、sm_120）是 NVIDIA 为不同代 GPU 定义的 **“计算能力”（Compute Capability）标识符**。

PyTorch 能支持的最高 GPU 计算能力（sm_xx），受限于编译它时所用的 CUDA Toolkit 版本。

CUDA: compute unified devices architecture