

R2-Gaussian 复现踩坑总结与完美复现流程

核心结论：复现核心问题集中在 **CUDA 版本不匹配**、**编译依赖缺失**、**环境变量未正确配置**，通过统一 CUDA 11.6 版本、安装对应编译器、显式配置环境变量可解决所有问题。

一、核心踩坑总结

（一）CUDA 相关坑（最关键）

1. 系统 CUDA 与 PyTorch CUDA 版本冲突

- 系统默认 nvcc 版本为 10.1，而安装的 PyTorch 是 cu118 版本，编译时检测到版本不匹配直接报错。
- 解决关键：通过 conda 安装对应版本的 cuda-nvcc 和 cudatoolkit-dev，而非依赖系统 CUDA。

2. CUDA 工具包混淆

- 混淆点：cudatoolkit（仅运行时库）、cudatoolkit-dev（含编译所需头文件）、cuda-nvcc（CUDA 编译器）三者缺一不可。
- 踩坑表现：仅安装 cudatoolkit 时，编译提示 `cuda_runtime.h: No such file or directory`，因缺少头文件；未安装 cuda-nvcc 时，无法找到 nvcc 编译器。

3. 环境变量未优先指向 conda 环境

- 系统 CUDA（10.1）优先级高于 conda 环境的 CUDA（11.6），导致编译时始终调用旧版本。
- 关键环境变量：CUDA_HOME（指定 CUDA 根目录）、CUDACXX（指定 nvcc 路径）、CPLUS_INCLUDE_PATH（指定头文件路径）必须显式配置。

（二）编译器版本坑

1. GCC 版本过高不兼容

- CUDA 11.6 不支持 GCC 9+，系统默认 GCC 版本过高导致编译报错 `unsupported GNU version! gcc versions later than 8 are not supported`。
- 解决：通过 conda 安装 GCC 8.5.0，并指定为默认编译器。

（三）子模块编译坑

1. torch 模块未被识别

- 编译simple-knn和xray-gaussian-rasterization-voxelization时，提示ModuleNotFoundError: No module named 'torch'。
- 原因：编译隔离模式导致依赖无法被识别，需添加--no-build-isolation参数。

2. 旧编译缓存干扰

- 多次编译失败后，残留的 build 目录、.so 文件导致后续编译仍沿用错误配置，需手动清理。

二、完美复现命令流程（一步到位）

前置准备

- 显卡：NVIDIA GeForce RTX 3060（支持 CUDA 11.6+）
- 系统：Linux（Ubuntu 衍生版）
- Anaconda 已安装并配置国内源（加速下载）

步骤 1：创建并激活 conda 环境

```
# 创建python3.9环境（项目依赖兼容版本）
conda create -n r2_gaussian python=3.9 -y
# 激活环境
conda activate r2_gaussian
```

步骤 2：安装 CUDA 相关依赖（关键统一版本）

```
# 安装CUDA 11.6编译器、开发包（含头文件和库）
conda install cudatoolkit-dev=11.6 cuda-nvcc=11.6 -c conda-forge -c nvidia/label/cuda-11.6.0 -y
# 安装兼容的GCC/G++ 8.5.0（避免版本过高）
conda install -c conda-forge gcc=8.5.0 gxx=8.5.0 -y
```

步骤 3：配置环境变量（永久生效 + 临时生效）

3.1 临时生效（当前终端，快速测试）

```
# 指定CUDA根目录（conda环境路径）
export CUDA_HOME=$CONDA_PREFIX
# 指定nvcc编译器路径
export CUDACXX=$CONDA_PREFIX/bin/nvcc
# 配置C/C++头文件路径（优先conda环境）
export C_INCLUDE_PATH="$CONDA_PREFIX/include:$C_INCLUDE_PATH"
export CPLUS_INCLUDE_PATH="$CONDA_PREFIX/include:$CPLUS_INCLUDE_PATH"
# 配置库文件路径
export LIBRARY_PATH="$CONDA_PREFIX/lib:$LIBRARY_PATH"
export LD_LIBRARY_PATH="$CONDA_PREFIX/lib:$LD_LIBRARY_PATH"
# 指定编译器为conda安装的GCC/G++
export CC=$CONDA_PREFIX/bin/gcc
export CXX=$CONDA_PREFIX/bin/g++
```

3.2 永久生效（所有终端，推荐）

```
# 将环境变量写入bash配置文件
echo "export CUDA_HOME=$CONDA_PREFIX" >> ~/.bashrc
echo "export CUDACXX=$CONDA_PREFIX/bin/nvcc" >> ~/.bashrc
echo "export C_INCLUDE_PATH=$CONDA_PREFIX/include:$C_INCLUDE_PATH" >> ~/.bashrc
echo "export CPLUS_INCLUDE_PATH=$CONDA_PREFIX/include:$CPLUS_INCLUDE_PATH" >>
~/.bashrc
echo "export LIBRARY_PATH=$CONDA_PREFIX/lib:$LIBRARY_PATH" >> ~/.bashrc
echo "export LD_LIBRARY_PATH=$CONDA_PREFIX/lib:$LD_LIBRARY_PATH" >> ~/.bashrc
echo "export CC=$CONDA_PREFIX/bin/gcc" >> ~/.bashrc
echo "export CXX=$CONDA_PREFIX/bin/g++" >> ~/.bashrc
# 生效配置
source ~/.bashrc
# 重新激活环境（确保变量生效）
conda activate r2_gaussian
```

步骤 4：安装 PyTorch 及核心依赖

```
# 安装PyTorch 2.1.2+cu118（与CUDA 11.6兼容， minor版本差异可忽略）
pip install torch==2.1.2+cu118 torchvision==0.16.2+cu118 --extra-index-url https://download.pytorch.org/whl/cu118
# 安装项目其他依赖（从requirements.txt）
pip install -r requirements.txt
```

步骤 5：编译子模块（simple-knn + xray-gaussian-rasterization-voxelization）

5.1 编译 simple-knn

```
# 进入子模块目录
cd r2_gaussian/submodules/simple-knn
# 清理旧编译缓存（避免残留干扰）
rm -rf build/ dist/ *.so simple_knn.egg-info/
# 编译安装（--no-build-isolation避免torch未被识别）
pip install -e . --no-build-isolation --force-reinstall
# 验证编译成功
python -c "import simple_knn; print('simple-knn编译成功! ')"
# 返回项目根目录
cd ../../../../
```

5.2 编译 xray-gaussian-rasterization-voxelization

```
# 编译安装（同样禁用build隔离）
pip install -e r2_gaussian/submodules/xray-gaussian-rasterization-voxelization --no-build-isolation
```

步骤 6：安装 TIGRE（可选，根据项目需求）

```
# 进入TIGRE Python目录（假设已下载TIGRE-2.3）
cd TIGRE-2.3/Python
# 安装（禁用build隔离）
pip install . --no-build-isolation
# 返回项目根目录
cd ../../
```

三、关键工具包区别说明

工具包名称	作用	缺失后果	安装渠道
cuda-toolkit	提供 CUDA 运行时库	运行 PyTorch CUDA 版本时报错缺 少 CUDA 库	conda-forge
cuda-toolkit-dev	提供 CUDA 编译所 需头文件（如 cuda_ runtime.h）	编译 C++ 扩展时提示 头文件缺失	conda-forge
cuda-nvcc	提供 CUDA 编译器（- nvcc）	无法编译 CUDA 相关 扩展（如 simple-knn ）	nvidia/label/cuda-1 1.6.0
gcc/g++（8.5.0）	提供 C/C++ 编译器	编译器版本不兼容 ，CUDA 编译报错	conda-forge

四、验证环境正确性

```
# 验证nvcc版本（应输出11.6）
nvcc -V
# 验证GCC版本（应输出8.5.0）
gcc --version
# 验证PyTorch CUDA可用性
python -c "import torch; print('PyTorch CUDA版本:', torch.version.cuda); print('CUDA是否可用:', torch.cuda.is_available())"
# 验证子模块安装成功
python -c "import simple_knn; import xray_gaussian_rasterization_voxelization; print('所有子模块安装成功! ')"
```

要不要我帮你整理一份**一键执行的环境配置脚本**？直接运行即可完成环境变量配置和依赖安装，无需手动逐行输入命令。

（注：文档部分内容可能由 AI 生成）