模型自动适配与调优工具

AscendBridge 2.0 操作指导书

文档版本：1.1

发布日期：2026-05-09



华为技术有限公司

版权所有 © 华为技术有限公司 2026 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明


和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址：深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址：https://www.huawei.com

客户服务邮箱：support@huawei.com

客户服务电话：4008302118

修订记录

日期	修订版本	描述	作者/修订人	审批人
2026-04-30	1.0	工具 2.0 操作指导书基线版本	季英国 00801953 吕璐凌 00878220 林 枫 00883274 徐胜生 00567035 王冬啸 00489484 刘侃 00821947 覃宏艺 00890442	田昆阳 00935552
2026-05-09	1.1	根据用户使用反馈，修改调整指导书内容	吕璐凌 00878220 覃宏艺 00890442	田昆阳 00935552

版权所有 © 华为技术有限公司

目录

1 前期准备工作.....4
1.1 Agent 后端模型准备.....**错误!未定义书签。**
1.1.1 智谱模型.....**错误!未定义书签。**
1.1.2 MiniMax 模型.....**错误!未定义书签。**
1.2 设备环境检查.....4
1.2.1 环境登录.....4
1.2.2 硬件约束.....4
1.2.3 固件与驱动.....5
1.2.4 docker 服务准备.....5
1.3 创建个人工作目录.....6
1.4 准备 Claude Code 安装配置参考文件.....**错误!未定义书签。**
1.5 模型权重提前下载.....6
1.5.1 从 HuggingFace 社区下载.....6
1.5.2 从魔搭社区下载.....7
1.6 创建容器.....9
1.6.1 加载容器镜像.....9
1.6.2 启动容器.....11
2 AI 智能体安装与配置.....13
2.1 自动化安装配置.....**错误!未定义书签。**
2.1.1 Cursor 账号注册.....**错误!未定义书签。**
2.1.2 在容器中安装 Cursor Agent.....**错误!未定义书签。**
2.1.3 安装配置 Claude Code.....**错误!未定义书签。**
2.2 手动安装配置.....**错误!未定义书签。**
3 AscendBridge 2.0 工具使用.....15
3.1 工具简介.....15
3.2 在容器中下载工具包.....**错误!未定义书签。**
3.3 使用方式.....17
3.3.1 环境准备.....17
3.3.2 典型使用方式一：Claude 单模型连贯适配.....18
3.3.3 典型使用方式二：Codex/Cursor 单模型连贯适配.....18
3.3.4 典型使用方式三：Claude 自动批量编排.....18
3.3.5 典型使用方式四：CANNBot 按需协同适配（v2.2 新增）.....22

 版权所有 © 华为技术有限公司

3.3.6 典型使用方式五：vLLM-Ascend 自动部署（v2.1 新增） 22
3.3.7 典型使用方式六：昇腾 NPU 继续预训练（CPT）（v2.3 新增） 25
3.3.8 典型使用方式七：LoRA 监督微调（SFT）（v2.5 新增） 29
4 已适配模型开源合集仓使用 26
4.1 仓库简介 32
4.2 下载开源仓 33
4.3 使用方式 33
4.3.1 独立使用 33
4.3.2 配合框架仓使用 34

1 前期准备工作

1.1 设备环境检查

1.1.1 环境登录

使用 Xshell / MobaXterm 软件登录昇腾服务器。

1.1.2 硬件约束

 # 内存约束：推荐大于500GB

 # 服务器型号：建议使用Atlas 800T A3/A2

执行dmidecode -s system-product-name命令可查询

```
[root@atlas-27 ~]# dmidecode -s system-product-name
Atlas 800T A3
```

 # 规格：单机8卡（单卡64GB）

执行npu-smi info命令可查询卡数、内存等相关信息

版权所有 © 华为技术有限公司

```
+------------------------------------------------------------------------------------------+
| npu-smi 25.2.3                 Version: 25.2.3                                           |
+---------------------------+---------------+----------------------------------------------+
| NPU   Name                | Health        | Power(W)    Temp(C)           Hugepages-Usage(page)|
| Chip  Phy-ID              | Bus-Id        | AICore(%)   Memory-Usage(MB)  HBM-Usage(MB)        |
+===========================+===============+==============================================+
| 0     Ascend910           | OK            | 163.2       40                0    / 0           |
| 0     0                   | 0000:9D:00.0  | 0           0    / 0          3124 / 65536         |
+---------------------------+---------------+----------------------------------------------+
| 0     Ascend910           | OK            | -           40                0    / 0           |
| 1     1                   | 0000:9F:00.0  | 0           0    / 0          2883 / 65536         |
+---------------------------+---------------+----------------------------------------------+
```

 # 网络：昇腾物理机需打通互联网

1.1.3 固件与驱动

 # 建议版本：25.2.3

注：此处的建议版本为写指导书时使用的环境上的固件版本，亦可选用其他较新版本。

检查方法：

echo '======Driver======' && head -1 /usr/local/Ascend/driver/version.info;

结果示例

```
[root@node-30-6 ~]# echo '======Driver======' && head -1 /usr/local/Ascend/driver/version.info;
======Driver======
Version=25.2.3
```

如果命令返回“No such file or directory”，说明驱动未安装，请参考下方链接进行安装：

https://support.huawei.com/enterprise/zh/doc/EDOC1100493505/426cffd9?idPath=23710424|251366513|254884019|261408772|252764743

1.1.4 docker 服务准备

检查docker服务是否已安装

执行docker --version命令，如果正常返回版本号信息，说明docker服务已安装

```
[root@atlas-27 ~]# docker --version
Docker version 18.09.0, build aa1eee8
```

如果命令返回“docker: command not found”说明当前环境没有docker服务，请按照下方指导安装docker环境

安装docker

参考：Ascend/ascend-deployer开源仓的安装指导

下载代码库，进入ascend_deployer目录，执行系统环境初始化（安装sys_pkg，docker）：

bash install.sh --install=sys_pkg

（执行--install命令时，请勿重复安装sys_pkg，安装成功即可查看docker版本号）

版权所有 © 华为技术有限公司

1.2 创建个人工作目录并获取 AscendBridge2 工具

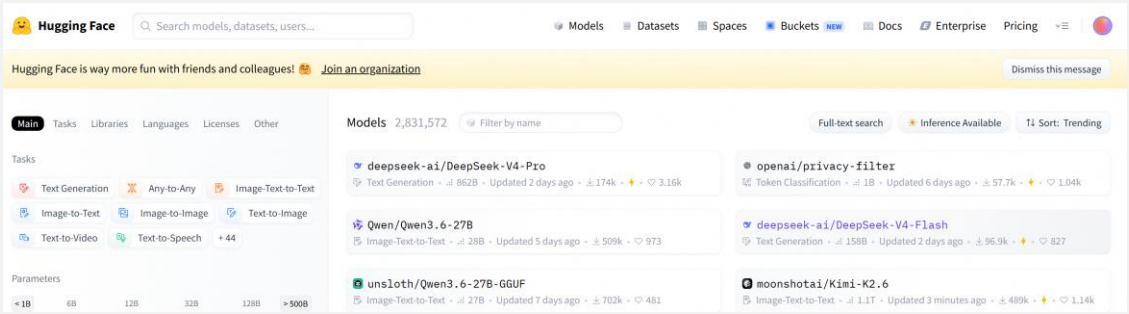
```
# 用root用户登录服务器，执行如下命令创建个人工作目录
mkdir <work_dir>
注：
<work_dir>是个人工作目录，可自定义，用于存放模型权重文件，以及后续涉及的一些脚本

# 从开源仓下载工具代码到个人工作目录
cd <work_dir>
git clone https://gitcode.com/SLAI/SLAI-AscendBridge2
```

1.3 模型权重提前下载

本章节介绍两种模型权重的下载方式，根据实际情况选择其中一种即可。

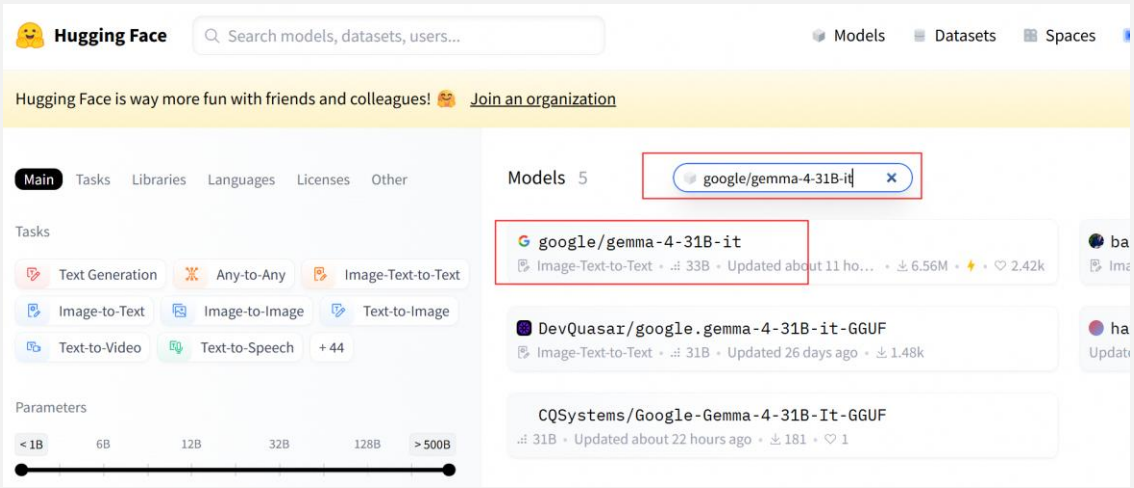
1.3.1 从 HuggingFace 社区下载

登录 HuggingFace 官网，网站如下：



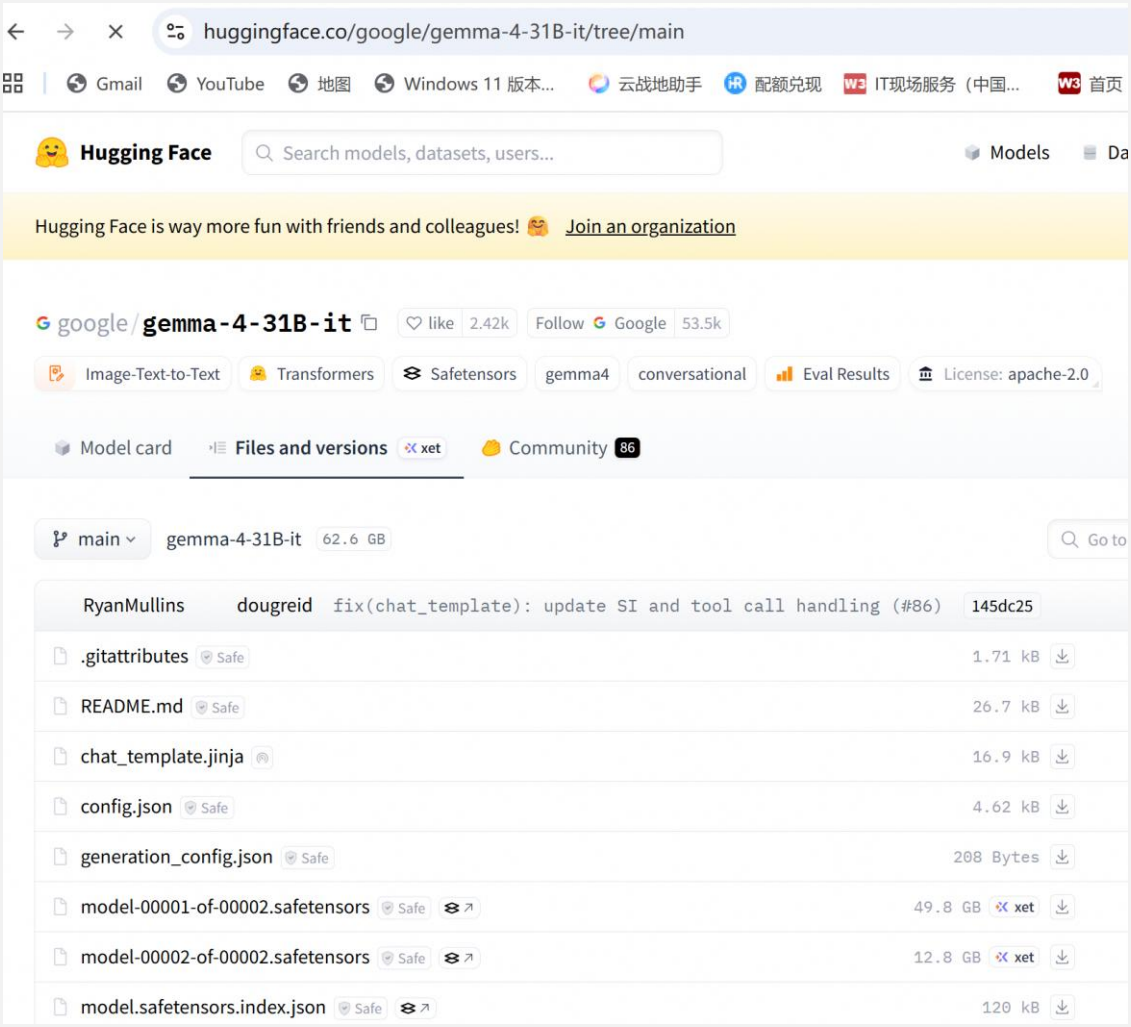
如果是第一次访问HuggingFace官网，请按照提示注册账号并登录。

搜索所需模型权重，如google/gemma-4-31B-it



版权所有 © 华为技术有限公司

权重如下：



用root用户登录服务器，执行如下命令，下载模型文件

cd <work_dir>

yum install git-lfs

git lfs install

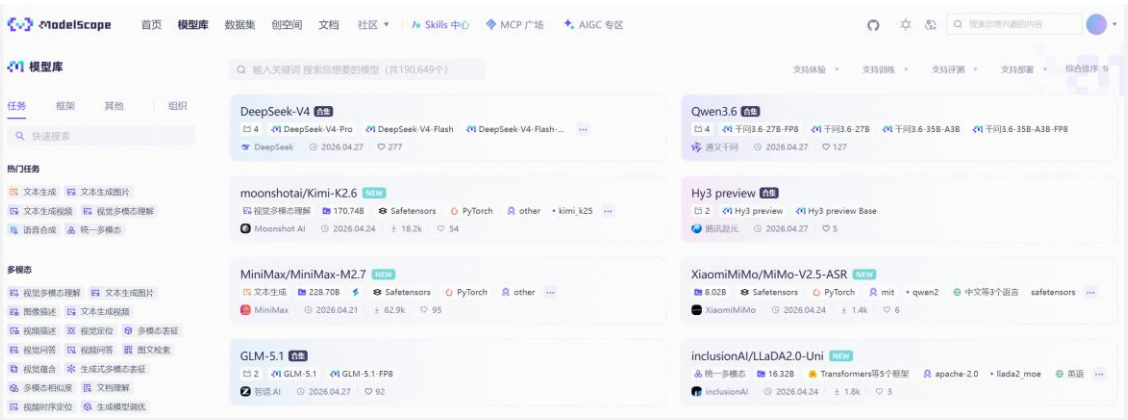
git clone https://huggingface.co/google/gemma-4-31B-it

注：

1. <work_dir> 是1.3小节中创建的个人工作目录
2. 命令中的模型名称可按实际需要进行替换，此处仅为示例
3. 如果用命令下载有问题，也可直接在模型页面点击各文件对应的【下载】图标，然后上传到服务器的<work_dir>目录下

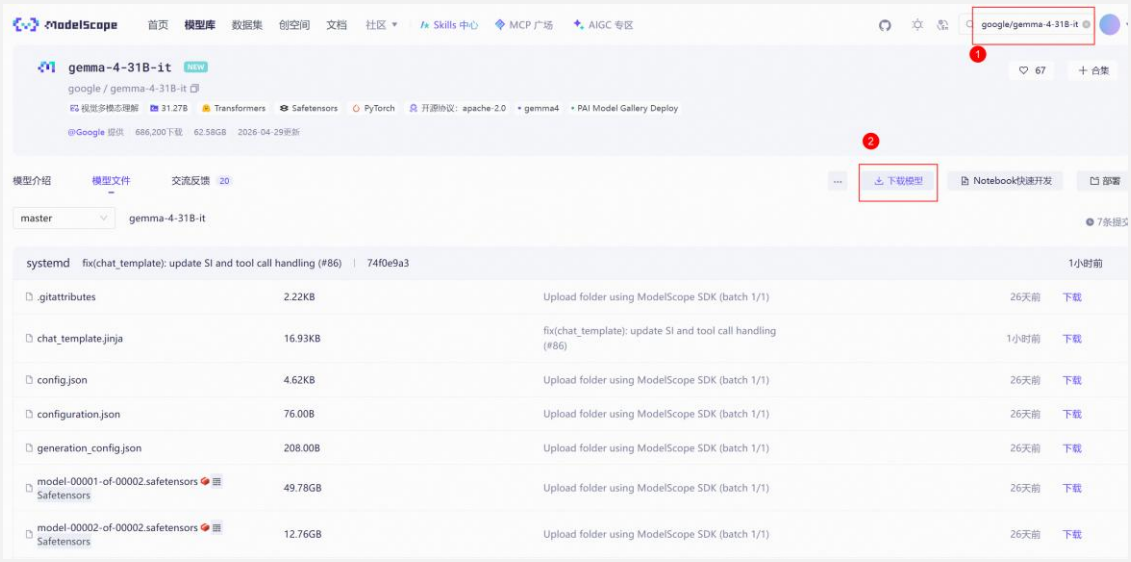
1.3.2 从魔搭社区下载

登录 ModelScope 官网，网站如下：



如果是第一次访问ModelScope官网，请按照提示注册账号并登录。

在“模型库”下搜索模型名称，如google/gemma-4-31B-it



点击“下载模型”，会弹出下载指导

版权所有 © 华为技术有限公司



用root账户登录服务器，按照下载指导，执行如下命令下载模型的所有文件
pip install modelscope
cd <work_dir>
mkdir gemma-4-31B-it
cd gemma-4-31B-it
modelscope download --model google/gemma-4-31B-it --local_dir ./

注：

1. <work_dir> 是1.3小节中创建的个人工作目录
2. 命令中的模型名称可按实际需要进行替换，此处仅为示例
3. download命令务必添加--local_dir参数，将权重文件下载到创建的模型名称目录下

1.4 创建容器

1.4.1 加载容器镜像

容器镜像建议采用包含有推理环境相关安装包的vllm-ascend镜像，可通过Docker仓库直接拉取，官方镜像托管在Quay.io平台。

版权所有 © 华为技术有限公司

下面以vllm-ascend **v0.17.0rc1-a3**版本的镜像为例，演示从Quay.io平台加载镜像的流程。
注：也可使用其他容器镜像，镜像中需要包含CANN、Pytorch等基础软件包。

执行以下命令查看容器镜像是否已加载，若已存在该镜像，则本小节操作到此结束
docker images | grep v0.17.0rc1-a3
若镜像未加载，则继续进行下面的步骤

打开Quay.io平台：https://quay.io/repository/ascend/vllm-ascend
找到左侧导航栏的“Tags”标签，在搜索框输入“v0.17.0rc1-a3”搜索到对应的镜像，点击右侧的“下载”按钮

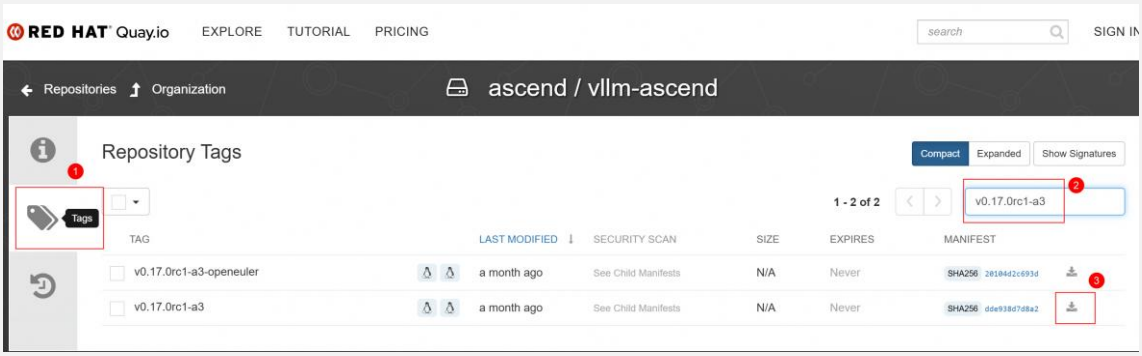


Image Format选择“Docker Pull（by tag)”，会出现一条docker pull命令



用root账户登录服务器，执行docker pull命令即可加载对应镜像

版权所有 © 华为技术有限公司

```
[root@atlas-27 ~]# docker pull quay.io/ascend/vllm-ascend:v0.17.0rc1-a3
v0.17.0rc1-a3: Pulling from ascend/vllm-ascend
c36472b34583: Already exists
9ff1473d4ee0: Already exists
3f80d0667f7a: Already exists
0a48a7b0869e: Already exists
307a6071cd9e: Already exists
63fe0193e142: Already exists
e0ec65d22c2c: Pull complete
1eb3048993c7: Pull complete
d2f3bf08ddcb: Pull complete
ca7661c4d9d5: Pull complete
4e8920cfc2e1: Pull complete
5efffd612257: Pull complete
2bba52cbe594: Pull complete
6c0e49f21133: Pull complete
c3f49a23e98a: Pull complete
07f9ae0ee986: Pull complete
004c60e60af8: Pull complete
Digest: sha256:dde938d7d8a24786648363fdb69d71bf25c77b2b56d3f26fd32d29427f375178
Status: Downloaded newer image for quay.io/ascend/vllm-ascend:v0.17.0rc1-a3
```

再次执行docker images命令，查看镜像是否加载成功

```
[root@atlas-27 ~]# docker images | grep v0.17.0rc1-a3
quay.io/ascend/vllm-ascend   v0.17.0rc1-a3       f7b939d9e47c        6 weeks ago         15.9GB
```

红框中是image id

由于直接从镜像仓库拉取镜像耗时较长，对于已经存放于服务器本地的镜像压缩包，也可直接通过如下命令进行镜像的加载

docker load -i <image_dir>/vllm-ascend-v0.17.0rc1-a3.tar

注：

<image_dir> 为服务器上存放镜像压缩包的路径

1.4.2 启动容器

用root用户登录服务器

执行命令检查容器镜像是否正常

docker images | grep v0.17.0rc1-a3

进入1.3小节中创建的个人工作目录

cd <work_dir>

创建容器启动脚本

vim docker_start.sh

输入i，进入编辑模式，粘贴下面脚本内容，编辑完成后按Esc进入命令模式再输入 :wq! 保存退出

注：要把宿主机的<work_dir>目录挂载到容器内同目录下

版权所有 © 华为技术有限公司

```
#!/bin/sh
NAME=AB2.0                  # 容器名称，自定义
PORT=8000                   # 容器与宿主机通信的端口号，不能冲突
IMAGE=f7b939d9e47c          # 1.6.1小节加载的容器镜像ID
docker run -itd -u 0 --ipc=host --privileged \
-e VLLM_USE_MODELSCOPE=True -e PYTORCH_NPU_ALLOC_CONF=max_split_size_mb:25
6 \
--name $NAME \
--device /dev/davinci_manager \
--device /dev/devmm_svm \
--device /dev/hisi_hdc \
-v /usr/local/dcmi:/usr/local/dcmi \
-v /usr/local/bin/npu-smi:/usr/local/bin/npu-smi \
-v /usr/local/Ascend/driver/lib64/:/usr/local/Ascend/driver/lib64/ \
-v /usr/local/Ascend/driver/version.info:/usr/local/Ascend/driver/version.info \
-v /etc/ascend_install.info:/etc/ascend_install.info \
-v /opt/data/verification/models:/root/.cache \
-v /home:/home \
-v /mnt:/mnt \
-v <work_dir>:<work_dir> \
-p $PORT:$PORT \
$IMAGE bash

# 创建容器
bash docker_start.sh

# 检查容器是否成功启动
docker ps | grep AB2.0
```

```
[root@atlas-27 lll]# docker ps
CONTAINER ID        IMAGE                                 COMMAND                  CREATED             STATUS              PORTS                    NAMES
f74eceb0f170        f7b939d9e47c                          "/bin/bash -c '    s…"   3 seconds ago       Up 2 seconds        0.0.0.0:8012->8012/tcp   AB2.0
```


版权所有 © 华为技术有限公司

2 AI 智能体安装与配置

2.1 脚本安装 claude code

AscendBridge2 工具内置了 claude code 安装脚本，在 scripts 目录下，使用可以快速的安装并配置好 claude code

进入容器

```
docker exec -it {docker_name} bash
```



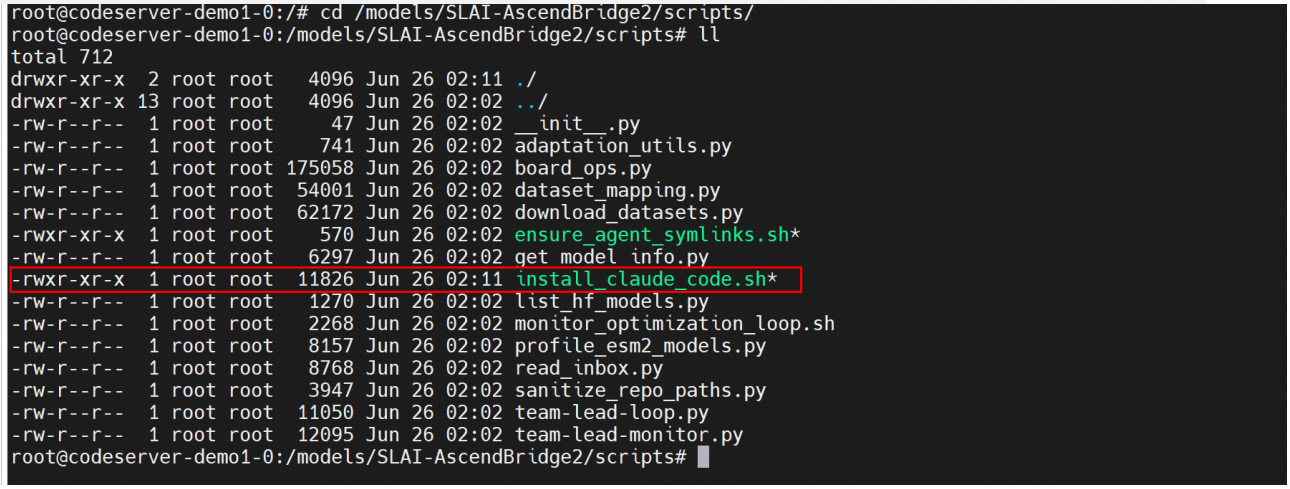
确认是否在容器内

```
docker ps
```



进入工具目录

```
cd <word_dir>/SLAI-AscendBridge2/scripts
```

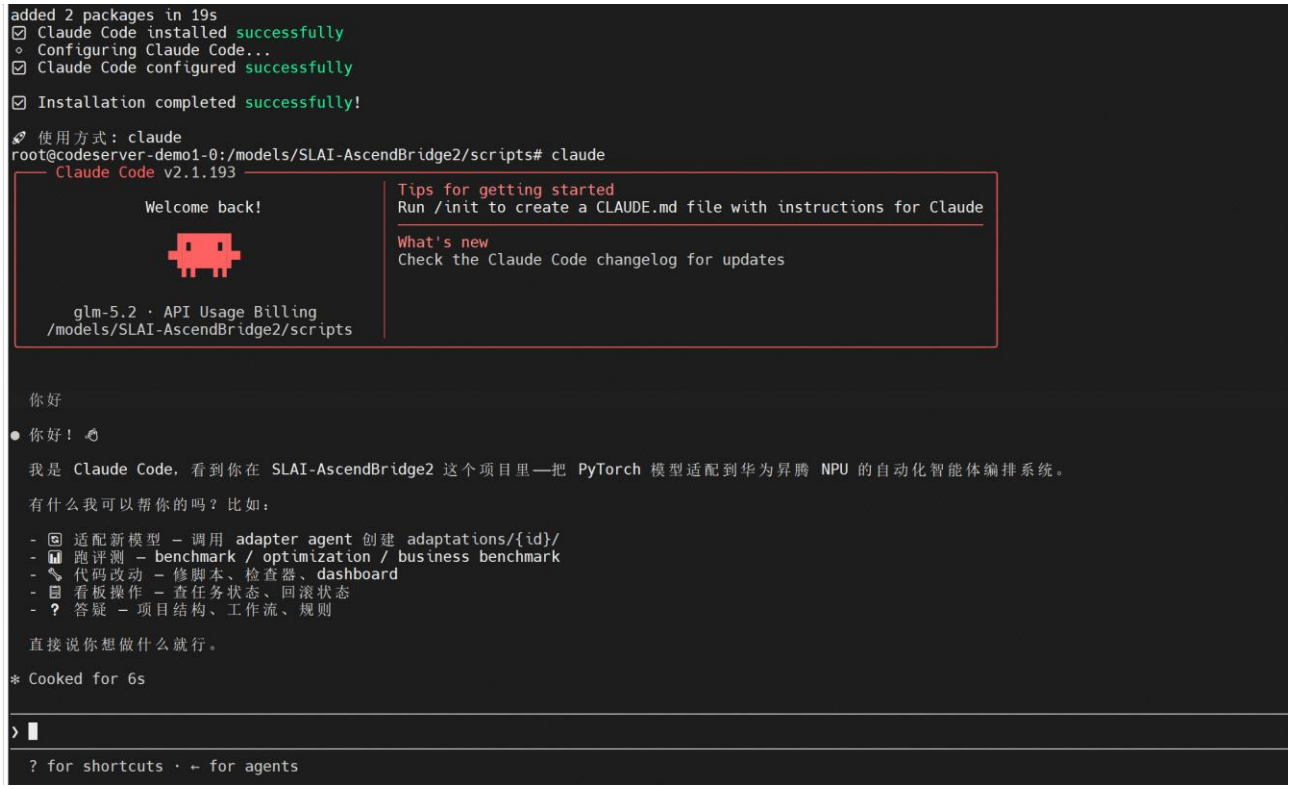


执行命令

```
bash install_claude_code.sh
```

根据提示输入内容

```
root@codeserver-demo1-0:/models/SLAI-AscendBridge2/scripts# bash install_claude_code.sh
请选择模型提供方：
  [1] 智谱 AI (GLM)
  [2] 阿里百炼 (Qwen)
  [3] 幻方 DeepSeek
  [4] 火山方舟 (豆包)
  [5] MiniMax (M1/abab)
  [6] 月之暗面 (Kimi)
  [7] 阶跃星辰 (Step)
  [8] Anthropic 官方
  [9] 自定义 (手动填 URL)
  [0] 退出

请选择 [1-9]: 1
请输入模型名称 (默认 glm-4.6): glm-5.2
请输入 API_KEY (默认 sk-local): ▒▒▒
```



3 AscendBridge 2.0 工具使用

3.1 工具简介

SLAI-AscendBridge2 是一款面向华为昇腾 NPU 的自动化智能体编排与单模型适配框架，用于将 PyTorch 模型迁移到 Ascend。它支持从模型发现、环境治理、代码适配、精度评测到 NPU 性能优化的闭环流程；如果需要，也可以继续扩展到第四阶段 business_benchmark。

仓库定位：

SLAI-AscendBridge2

开源地址：https://gitcode.com/SLAI/SLAI-AscendBridge2

- 框架、脚本、检查器、dashboard、.claude agents / skills / agent-memory
- 适合批量调度、多阶段闭环、统一状态管理

SLAI-AscendBridge2-Adaptations

开源地址：https://gitcode.com/SLAI/SLAI-AscendBridge2-Adaptations

版权所有 © 华为技术有限公司

- 是 SLAI-AscendBridge2 的已适配模型开源合集仓，存放模型级 adaptations/{name}/ 产物集合
- 适合单模型复用、逐步公开、按需筛选发布

仓库主要文件架构如下：

```
SLAI-AscendBridge2/
├── .claude/
│   ├── agent-memory/                  # 智能体阶段记忆与规则沉淀
│   ├── agents/                        # 智能体定义
│       ├── team-lead.md               # 团队主控：负责任务分配、并发编排、状态推进
│       ├── adapter.md                 # 适配执行：模型迁移到Ascend、保证demo可运行
│       ├── model-crawler.md           # 模型发现：抓取并注册新模型到 board
│       ├── benchmark-runner.md        # 评测执行：生成/运行 accuracy_run
│       ├── npu-optimizer.md           # 优化执行：生成 accuracy_run_perf
│       └── business-benchmark.md      # 真实业务测评与 NPU/CUDA 对比证据沉淀
│   └── skills/                        # 技能库
│       ├── ascend-adaptation/
│           └── SKILL.md               # 通用 Ascend 适配方法与操作清单
│       ├── ascend-diffusers-adaptation/    # diffusers类模型迁移到NPU的专项指导
│       ├── ascend-diffusers-benchmark/     # diffusers类模型 benchmark 编写与执行规范
│       ├── ascend-diffusers-optimization/  # diffusers类模型推理优化策略
│       ├── benchmark-analysis/        # 聚合报告、输出对比、trace/profiling 分析说明
│       ├── benchmark-script/          # accuracy_run.py 生成与执行主技能
│       ├── torchair-graph-mode/       # TorchAir 图模式启用/调优/排障总指南等
│       ├── torch-npu-optimization/    # 通用 torch_npu 优化策略与执行步骤等
│       └── ...
├── .agents -> .claude                          # 兼容部分 agent 工具的目录入口
├── AGENTS.md -> CLAUDE.md                      # 兼容 AGENTS.md 约定
├── adaptation/
│   └── scripts/
│       ├── adaptation_manager.py               # 第一阶段运行与产物管理
│       ├── check_adaptation.py                 # demo.py / adaptation 完成度检查
│       ├── package_adaptations.py              # adaptation 打包
│       └── run_completed_adaptations.py        # 批量下载已完成适配模型的权重
├── benchmark/                                  # 精度/性能评测模块
│   ├── scripts/
│       ├── benchmark_tool.py                   # 聚合、对比、trace/profiling 分析
│       ├── benchmark_manager.py                # benchmark 运行与产物管理
│       └── check_accuracy_run.py               # accuracy_run.py 规范检查
│   ├── figures/                                # 图表产物
```


版权所有 © 华为技术有限公司

```
│   └── reports/                          # 汇总报告
├── optimization/                         # NPU 推理优化模块
│   ├── scripts/
│       ├── optimization_tool.py          # 优化数据聚合与对比
│       ├── optimization_manager.py       # 优化运行管理
│       ├── check_accuracy_run_perf.py    # accuracy_run_perf.py 结构检查
│       └── check_optimization_notes.py
│   └── reports/                          # 优化评测结果
├── business_benchmark/                   # 业务测评模块
│   ├── scripts/                          # 运行管理、汇总、completed gate 检查
│   └── templates/                        # 模板文件
├── dashboard/                            # Web 看板
├── prompts/                              # team-lead prompt 模板
├── scripts/
│   ├── board_ops.py                  # 看板 CRUD、心跳、任务分配
│   ├── get_model_info.py             # 模型元数据提取
│   ├── download_datasets.py          # 数据集下载
│   ├── dataset_mapping.py            # 模型 -> 数据集映射
│   ├── ensure_agent_symlinks.sh      # 恢复 .agents / AGENTS.md 兼容软链接
│   └── sanitize_repo_paths.py        # 路径清洗工具
├── adaptations/                      # 默认空目录，可挂载或拷贝单模型adaptation
├── CLAUDE.md                         # 项目上下文说明
├── pyproject.toml                    # Python/nv项目配置
├── uv.lock                           # uv锁文件
└── README.md                         # 项目说明
```

3.2 使用方式

3.2.1 环境准备

```
# 进入1.6.2小节创建的容器
docker exec -it AB2.0 bash


# 执行docker命令返回如下时，说明已进入容器内
root@5078b464d7ef:/workspace# docker
bash: docker: command not found


# 在容器中安装uv（如之前未安装）
curl -LsSf https://astral.sh/uv/install.sh | sh
```

```
# 初始化框架环境
uv sync
cd <word_dir>/SLAI-AscendBridge2
uv run python scripts/board_ops.py init

# 推荐环境变量
export HF_ENDPOINT=https://hf-mirror.com
export TASK_QUEUE_ENABLE=1

# 权重准备建议
建议提前把模型权重准备到本地路径，避免 agent 在适配过程中临时下载导致等待时间过长或受网络影响。如果不提前准备，agent 也可以自行下载，但整体链路通常更慢。
权重文件下载可参考1.4小节。
```

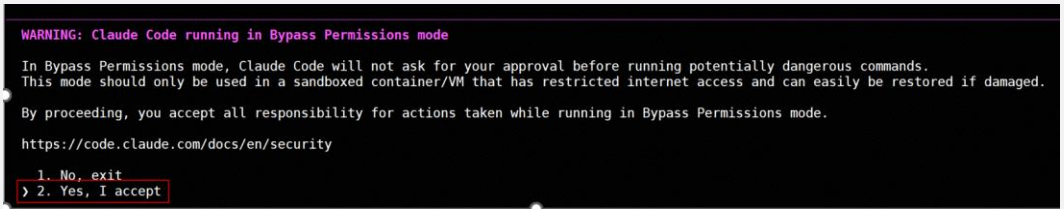
3.2.2 基础功能：单模型适配

适用场景：

- 只想处理一个模型
- 希望一个对话连续完成 Adaptation -> Benchmark -> Optimization
- 需要 agent 在同一上下文里连续修复问题，而不是切到批量队列

下面用 google/gemma-4-31B-it 模型作为示例演示流程。

```
# 进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 执行docker命令返回如下时，说明已进入容器内
root@5078b464d7ef:/workspace# docker
bash: docker: command not found

# 进入工具根目录下
cd <work_dir>/SLAI-AscendBridge2

# 执行以下命令，启动Claude
IS_SANDBOX=1 claude --dangerously-skip-permissions
出现如下选项，选择 2
```



注：

若直接用“claude”命令启动，后续agent有很多操作都需要人工允许才可继续往下进行，用上述命令启动，表示信任claude的所有操作，无需人为干预。

输入提示词，开始模型的适配调优

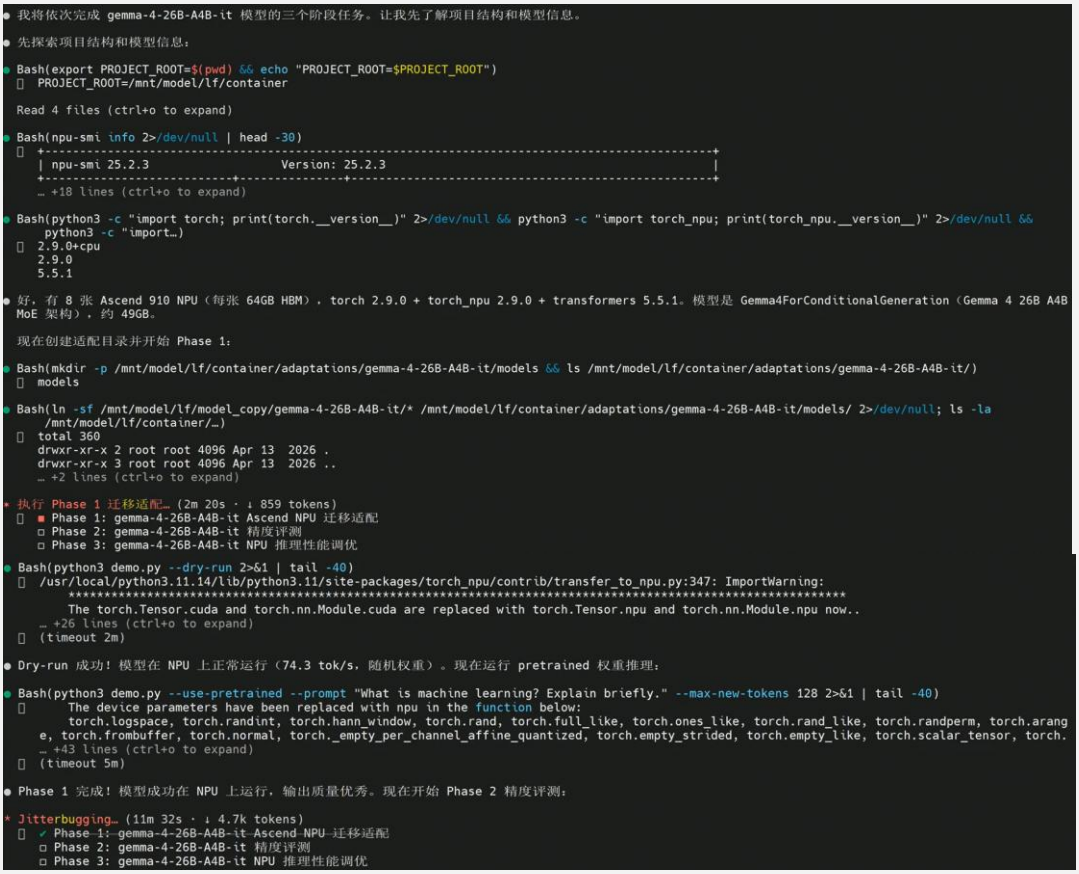
提示词示例：

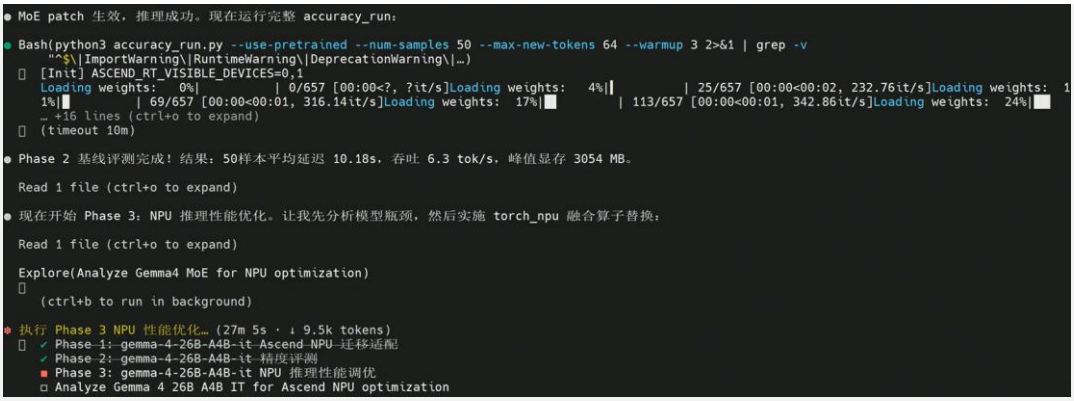
你当前在一台昇腾910C服务器上，你的任务是完成google/gemma-4-31B-it模型的迁移适配、精度对齐和推理性能调优，请依次完成指定的这三个阶段的任务，并给我输出一份完整详细的任务报告并落盘，报告中一定要包含性能调优前后的数据对比，过程中生成的文件请在adaptations目录下新建一个模型的文件夹进行存放，模型权重文件在/mnt/model/gemma-4-31B-it。

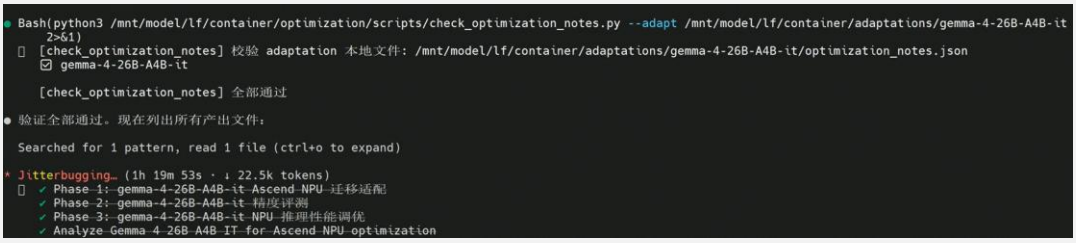
注：

1. 如更换其他模型进行适配调优，请同步修改提示词中的模型名称以及权重文件所在目录，权重文件建议提前下载，参考1.5小节。
2. 如果实际服务器型号与示例不同，也请修改提示词。
3. 如果没有提前下载模型权重，在提示词中无需指定权重文件路径，Agent会自行下载。

执行过程示例







结果总结输出

三个阶段全部完成。现在输出完整的任务报告：

Gemma 4 26B A4B IT 昇腾 NPU 迁移适配、精度评测与性能调优报告

一、模型概述

项目	详情
模型	Google Gemma 4 26B A4B IT
架构	Gemma4ForConditionalGeneration（多模态） → 提取 Gemma4ForCausalLM（纯文本）
MoE 配置	128 专家，top-8 路由，~26B 总参数 / ~4B 活跃参数
层数	30 层（25 sliding_attention + 5 full_attention）
隐藏维度	2816
注意力头	16 heads / 8 KV heads (GQA)
权重路径	/mnt/model/lf/model_copy/gemma-4-26B-A4B-it (49GB)
硬件	华为昇腾 910C NPU × 8（每张 64GB HBM）
软件栈	PyTorch 2.9.0 + torch_npu 2.9.0 + transformers 5.5.1

版权所有 © 华为技术有限公司

二、Phase 1：迁移适配

2.1 关键技术挑战与解决

挑战	解决方案
多模态权重提取	模型原为 Gemma4ForConditionalGeneration，包含视觉编码器。通过 model.language_model.* → model.* 重映射，分离出 657 个文本权重键（丢弃 356 个视觉权重键）
MoE torch._grouped_mm 不支持	Ascend NPU 不支持 torch._grouped_mm（CUDA 专有算子），运行时 ACL 报错 507018。通过 monkey-patch transformers.integrations.moe._can_use_grouped_mm 强制返回 False，使用 loop-based fallback
Gemma4 layer_types 长度校验	config.num_hidden_layers 必须等于 layer_types 数组长度。dry-run 时将两者同步截断到 2
AutoConfig.for_model 参数冲突	model_type 字段不能同时作为位置参数和关键字参数传入，需先 pop 再传入

2.2 产出文件

- demo.py — 主推理脚本（支持 --dry-run、--use-pretrained、--cpu）
- pyproject.toml — 依赖配置
- output.txt — 首次推理输出记录

2.3 验证结果

```
[Mode] PRETRAINED (full weights)
[Run] Output: At its simplest, **machine learning (ML)** is a field of artificial intelligence...
[Metrics] 128 tokens in 24.02s (5.3 tok/s)
[Success] Inference completed on npu:0
```

☑ 模型在 Ascend NPU 上成功运行，输出语义正确、流畅。

三、Phase 2：精度评测（Baseline）

3.1 测试配置

参数	值
样本数	50
每样本最大生成 token	64
Warmup 次数	3
精度	bfloat16
设备	npu:0（跨 2 张 NPU device_map="auto"）

3.2 Baseline 性能数据

指标	值
平均延迟	10.18s
平均吞吐	6.29 tok/s
峰值显存	3,054 MB (3.0 GB)
总生成 token	3,200
总延迟	509.21s

3.3 产出文件

- accuracy_run.py — 评测脚本
- benchmark_metrics_google_gemma-4-26b-a4b-it_baseline.json
- outputs_google_gemma-4-26b-a4b-it_baseline.pt

四、Phase 3：NPU 推理性能调优

4.1 优化策略

采用 torch_npu 融合算子 monkey-patch 方式，在 model_files/__init__.py 中实现：

优化项	实例数	说明
npu_rms_norm	300+	替换 Gemma4RMSNorm（每层 10 个：input/post_attn/pre_ffw/post_ffw + q/k/v norm + router norm + MoE 额外 norm）
npu_gelu (Dense MLP)	25	替换 Gemma4TextMLP 中的 F.gelu(approximate='tanh')
npu_gelu (MoE Experts)	128 专家 × 5 层	替换 Gemma4TextExperts.act_fn，保持路由逻辑不变
npu_rotary_mul	60	替换 apply_rotary_pos_emb（每层 2 次：Q 和 K）
环境变量	-	TASK_QUEUE_ENABLE=1、COMBINED_ENABLE=1、ACL_OP_COMPILER_CACHE_MODE=1

跳过的优化：

优化	原因
npu_fusion_attention	模型使用 softcapping + 混合 sliding/full attention + GQA + per-head QK norm，掩码构建复杂，稳定性风险高
npu_grouped_matmul	MoE 专家的 per-token 路由导致批次极小，overhead 可能超过收益

版权所有 © 华为技术有限公司

4.2 调优过程

尝试	配置	结果	分析
✗ 全量重写 experts forward	rms_norm + gelu + rotary + experts 路由重写	18.73s (0.54x)	性能退化：重写 experts forward 路由逻辑引入额外开销
☑ 核心3项	rms_norm + dense gelu + rotary	9.35s (1.09x)	有效加速
☑ 最终方案	rms_norm + dense gelu + rotary + 轻量 experts act_fn	9.07s (1.12x)	轻量 act_fn 替换进一步优化

4.3 优化后性能数据

指标	Baseline	Optimized	变化
平均延迟	10.18s	9.07s	-10.9%
平均吞吐	6.29 tok/s	7.06 tok/s	+12.2%
峰值显存	3,054 MB	3,054 MB	持平
Speedup	1.00x	1.12x	-

3.2.3 vLLM-Ascend 自动部署

v2.1 新增 vllm-ascend-deployer Agent 和 vllm-ascend-auto-deploy Skill，可把模型与部署需求转换为经过预检和真实推理验收的 vLLM-Ascend 服务。

适用场景：

- 已有模型，希望快速启动 vLLM-Ascend 推理服务
- 需要部署到本机、SSH 服务器或 Kubernetes/CCE/ACK 调度平台
- 需要自动规划单机、多机或 PD（Prefill/Decode）分离部署拓扑
- 希望自动完成部署前检查、部署文件生成和真实推理验收

部署前准备镜像

```
# 根据目标机器的Ascend代际选择对应的vLLM-Ascend镜像变体，并确保镜像版本支持待部署模型。
# 项目提供images/fetch-image.sh，用于把远端镜像保存成可复用的离线tar：
bash images/fetch-image.sh v0.23.0rc1 a3
# 脚本优先使用crane下载，失败时回退到docker pull + docker save，生成以下文件：
images/vllm-ascend-v0.23.0rc1-a3.tar
images/vllm-ascend-v0.23.0rc1-a3.tar.sha256

注：本机或SSH部署时，Agent会优先使用并加载images/目录下匹配的本地tar；如果没有匹配文件，则使用预检确认的远端registry镜像。Kubernetes/CCE/ACK部署需提前保证平台节点能够拉取该镜像，如果平台无法访问公共registry，请先把镜像推送到平台私有镜像仓库。

# 进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 执行docker命令返回如下时，说明已进入容器内
root@5078b464d7ef:/workspace# docker
bash: docker: command not found
# 进入工具根目录
```

```
cd <work_dir>/SLAI-AscendBridge2
# 通过team-lead启动部署
IS_SANDBOX=1 claude --dangerously-skip-permissions --agent team-lead
```

然后明确要求 team-lead 调用 vllm-ascend-deployer，并用自然语言描述模型和部署目标。部署 Agent 会按需询问无法安全推断的信息，自动完成权重解析、环境预检和并行拓扑规划，并在实际部署前展示完整配置供确认。

提示词示例一（本机单机部署）：

请调用vllm-ascend-deployer，使用vLLM-Ascend在本机部署Qwen/Qwen3-30B-A3B，权重位于adaptations。采用单机部署。优先使用adaptations/中已有的模型权重。

提示词示例二（SSH多机部署）：

请调用vllm-ascend-deployer，使用vLLM-Ascend部署Qwen/Qwen3-235B-A22B。采用多机SSH部署，不启用PD分离。

注：Agent会继续收集目标主机、用户名和认证方式。密码、Token和私钥不会写入部署JSON、脚本、日志或Git。

自动部署流程如下：

```
描述模型和部署需求
-> 选择单机/多机及部署目标
-> 解析或准备模型权重
-> 检查驱动/固件/CANN、HCCL、dtype与部署约束
-> 自动规划TP/DP/EP或PD拓扑
-> 展示完整配置摘要
-> 用户确认执行
-> 生成部署文件并启动服务
-> 通过OpenAI兼容API完成真实推理与服务一致性验收
```

部署成功后，Agent 会返回服务 endpoint、本地进程 PID（远程进程信息或调度任务 ID）、日志位置、停止和清理命令，以及本次生成的一键部署脚本或 YAML 路径。

注：部署失败时只清理本次创建的进程或调度资源，并保留必要的诊断信息。

3.2.4 CANNBot 按需协同适配

v2.2 新增 cannbot-adapter Agent、CANNBot 协同工作流和项目内同步脚本，用于处理 AscendBridge 自动适配调优无法覆盖的算子兼容与性能问题。

适用场景：

- 模型在 Ascend 上存在标准 PyTorch 算子和 torch_npu 专用接口都无法补齐的算子缺口
- 已经通过 profiling 确认 CPU 回退、频繁同步或低效算子是主要性能瓶颈

版权所有 © 华为技术有限公司

- 希望在保留 SLAI-AscendBridge2 主体身份和工作流的前提下，临时借用 CANNBot 的 Ascend C 专家能力

CANNBot 按需协同适配采用四级决策顺序，只有在前面的路径均不可行时，才调用 CANNBot 生成 Ascend C 自定义算子：

1. 优先使用可在Ascend上直接执行的标准PyTorch算子
2. 只接受torch_npu提供的单一、公开、语义等价的原生接口（没有单一接口就判定没有，禁止组合多个接口冒充原生算子）
3. 通过GitCode服务端API全量枚举并搜索CANN与Ascend社区已有方案，禁止为搜索下载仓库
4. 以上路径均不可行且搜索报告完整时，才调用CANNBot生成Ascend C自定义算子

```
# 进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 执行docker命令返回如下时，说明已进入容器内
root@5078b464d7ef:/workspace# docker
bash: docker: command not found
# 进入工具根目录
cd <work_dir>/SLAI-AscendBridge2
# 推荐通过team-lead触发CANNBot，无需提前安装CANNBot
IS_SANDBOX=1 claude --dangerously-skip-permissions --agent team-lead
```

提示词示例：

请在当前昇腾设备上完成TRELLIS2模型的适配。如果发现算子缺口，先尝试可在Ascend上直接执行的标准PyTorch算子；标准实现存在功能缺口或经profiling确认性能不达标时，再尝试torch_npu提供的昇腾专用接口或优化实现。随后搜索GitCode上已有的CANN/昇腾社区方案；只有这些路径均不可行时，才调用cannbot-adapter。
请用真实权重完成精度和性能验收。

注：进入CANNBot阶段时，项目会自动执行scripts/sync_cannbot.sh --print-path，该命令会检查并同步上游master最新提交，只在标准输出中返回项目缓存路径，供协同工作流读取。CANNBot只下载到当前项目的.cache/cannbot/cannbot-skills/目录，该目录已被Git忽略。

```
# 需要生成自定义算子时，按以下四个角色推进，完成后通过cannbot_ops.py接回模型适配主链路
Architect -> Design Reviewer -> Developer -> Reviewer
```

注：不要直接在.cache/cannbot/cannbot-skills/目录中启动Claude。CANNBot是按需读取的外部能力来源，项目主体、任务编排和最终交付仍由SLAI-AscendBridge2负责；同步过程不会修改~/.claude/settings.json，普通Claude会话仍将自己识别为SLAI-AscendBridge2，不会自称CANNBot。

版权所有 © 华为技术有限公司

3.2.5 昇腾 NPU 继续预训练（CPT）

v2.3 新增 ascend-torch-cpt Skill，把任意 HuggingFace 模型+训练语料在昇腾 NPU（Ascend 910/910B/910C/950 等）上端到端跑一遍继续预训练（Continued Pre-Training, CPT），并产出训练脚本、loss 曲线（含公网直链）、概要总结与训练前后域内评估。

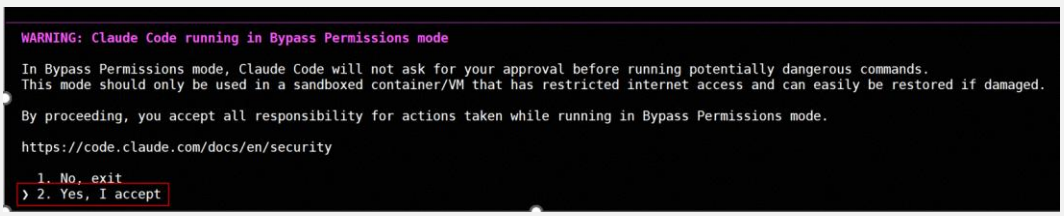
用户只需给出【模型权重路径】+【训练数据集路径】即可启动；其余超参由 Skill 根据模型规模、硬件显存与数据规模自动择优，也可以在提示词中直接输入超参内容。

核心能力：

- 训练范式自动选型（10 类）：文本 LM / Encoder MLM / Seq2Seq / 扩散流匹配与 DDPM / 音频-LLM / MLIP 力场 / ViT-MAE 视觉 / VLM 文本头 / 视觉时序音频专用 + Keras→PyTorch 迁移
- 训练方式自动选型：单卡 Eager / DDP / FSDP2 / 模型并行，按参数量、单卡显存与卡间互联实测自动决定
- torch_npu 融合路径：NpuFusedAdamW 融合优化器、NPU 融合 attention 自动路由、TASK_QUEUE_ENABLE 异步算子下发等
- 超参自动择优：precision / lr / batch / seq / warmup，附 OOM 6 级回退阶梯
- 模型与语料自动获取：6 源可达性矩阵探测、多路 Range 分块断点续传、CPT 前权重 NaN 全量扫描
- 断点续训：周期保存 ckpt_latest.pt（模型+优化器+step），RESUME=1 中断续训
- loss 曲线（EMA 平滑+公网直链）、全程实时用时表、训练前后域内评估与“训练是否有效”结论
- 140 条实战踩坑清单与 20 个可直接复用的标准脚本模板

Skill 与全部 references/scripts 位于仓库的.claude/skills/ascend-torch-cpt/目录下。

```
# 进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 执行docker命令返回如下时，说明已进入容器内
root@5078b464d7ef:/workspace# docker
bash: docker: command not found
# 进入工具根目录
cd <work_dir>/SLAI-AscendBridge2
# 执行以下命令，启动Claude
IS_SANDBOX=1 claude --dangerously-skip-permissions
# 出现如下选项，选择 2
```




```
# 用自然语言给出模型权重路径与训练数据集路径，ascend-torch-cpt Skill会自动触发
```

提示词示例：

版权所有 © 华为技术有限公司

你现在使用的是昇腾设备，请使用/mnt/model/experiments/users/jiyingguo/train_ws/Qwen3.5-0.8B目录模型权重和/mnt/model/experiments/users/jiyingguo/train_ws/saudi-dialect-conversations预训练语料的80%训练集数据，通过PyTorch + torch_npu方式自动生成相关的继续预训练脚本文件（如果预训练语料格式不满足预训练要求，请自动按照要求格式转换），在这台昇腾NPU设备上跑一个继续预训练任务，seq_len=320，batch_size=64，跑 200 步，列出继续预训练的loss的数据，并画出loss曲线图，并输出本次继续预训练的概要总结。继续预训练过程生成的相关脚本及结果文件保存下来，以便后续使用。在这个过程中整体的用时估计以及每一阶段的大概用时和预计后续用时也请及时给出评估，并在屏幕实时更新。

说明：如下为模型训练权重和训练数据集目录，以实际路径为准

/mnt/model/experiments/users/jiyingguo/train_ws/Qwen3.5-0.8B

/mnt/model/experiments/users/jiyingguo/train_ws/saudi-dialect-conversations

数据集：saudi-dialect-conversations

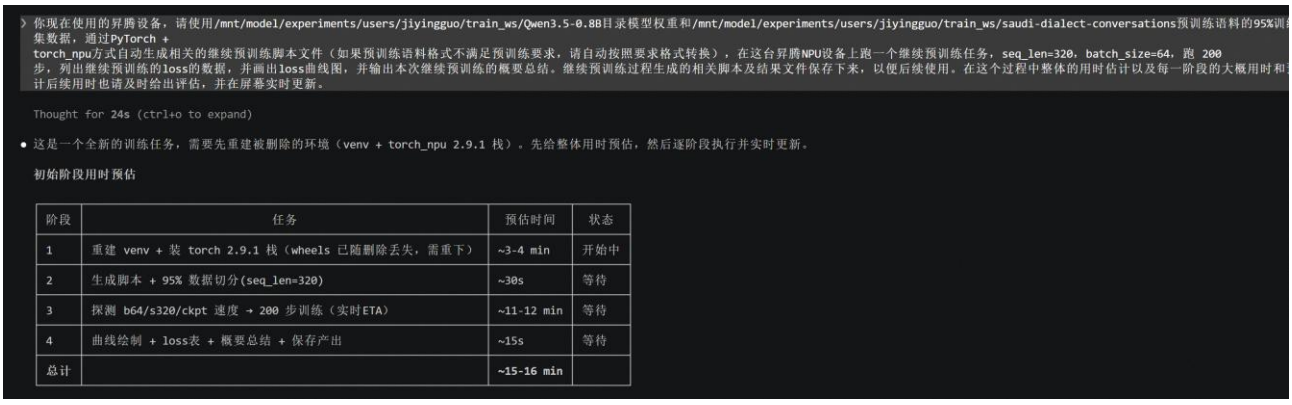
这是一个专注于沙特阿拉伯方言（Saudi Dialect） 的多轮对话数据集。

数据规模：包含 3,545 个多轮对话，数据量级在 1K 左右，属于非常小型的专用数据集。

数据内容：内容为沙特方言的日常对话，场景丰富，例如在银行讨论业绩、在家乐福超市讨论买电视、在清真寺义卖、咨询政府服务等。

主要过程参考回显信息如下：

1）初始阶段用时预估



2）启动训练

版权所有 © 华为技术有限公司

```
● 探测结果：3.15s/step，显存 16.5 GB，200 步 ETA ~10.5 min。显存安全、速度符合预期（介于 seq312 的 3.20 和 seq384 的 4.02 之间）。

  阶段进度更新

  ✅ Stage 1: 重建 venv + torch_npu 栈              完成   ~3 min
  ✅ Stage 2: 脚本 + 95% 数据(seq=320)              完成   8s
  ⏳ Stage 3: 200 步 CPT (b64/s320/ckpt)            开始   3.15s/step × 200 ≈ 10.5 min
  ⏸ Stage 4: 曲线+总结+保存                          等待   ~15s

  正式开跑 200 步，后台无缓冲输出，实时轮询 ETA。

● Bash(cd /mnt/model/experiments/users/jiyingguo/train_ws/qwen3_5_cpt && rm -rf ckpt && ASCEND_RT_VISIBLE_DEVICES=0 TASK_QUEUE_ENABLE=1 .venv/bin/python -u train_cpt.p…)
  └ Running in the background (↓ to manage)

● 训练已启动。等 ~90s 后第一次实时状态更新。

● Bash(sleep 90; cd /mnt/model/experiments/users/jiyingguo/train_ws/qwen3_5_cpt; python3 - <<'PY'
      import re, time…)
  └ === LIVE STATUS @ 18:15:37 ===
    step 20/200 | elapsed 62.3s | rate 3.115s/step | remaining 180 steps
    ETA remaining ~561s (9.3 min) | total est ~623s (10.4 min) | last loss 2.3336
    [#/#/] 10%
    Shell cwd was reset to /mnt/model/experiments/users/jiyingguo/SLAI-AscendBridge2
```

3）绘制 loss 曲线+打印 loss 表：

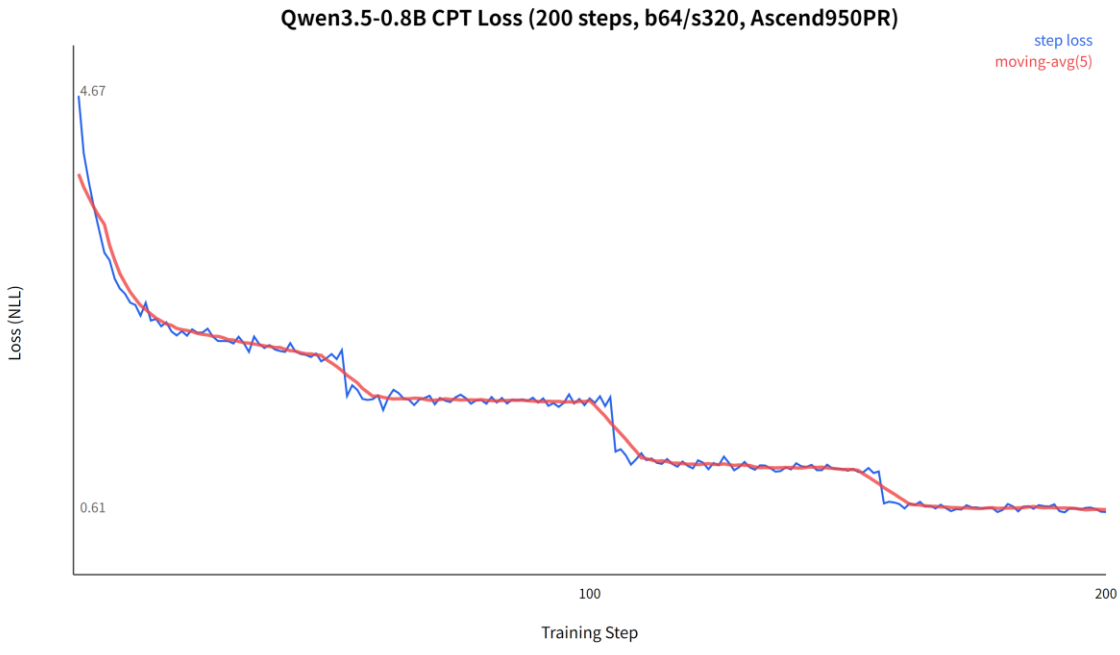
```
● Stage 3 完成 ✅ 200 步，622.5s，loss 4.67 → 0.61（min 0.61，mean 1.52，3.77 epoch），checkpoint 已存。

  阶段进度

  ✅ Stage 1: 重建 venv + torch_npu 栈              ~3 min
  ✅ Stage 2: 脚本 + 95% 数据(seq=320)              8s
  ✅ Stage 3: 200 步 CPT (b64/s320/ckpt)            622.5s (3.11s/step)
  ⏳ Stage 4: 曲线+loss表+总结+保存                  开始

  Stage 4: 绘制曲线 + 打印 loss 表。

● Bash(cd /mnt/model/experiments/users/jiyingguo/train_ws/qwen3_5_cpt && .venv/bin/python plot_loss.py 2>&1 | grep -vE "SyntaxWarning|_op_plugin_docs|^  \"\"" > /dev/n…)
  └ curve regenerated: 85916 bytes
    === sampled loss table (every 10 steps + first/last) ===
    step     loss          lr       gn
    … +23 lines (ctrl+o to expand)
    Shell cwd was reset to /mnt/model/experiments/users/jiyingguo/SLAI-AscendBridge2

· Plotting curve, summary, saving artifacts… (18m 59s · ↓ 12.4k tokens)
  └ ✓ ~~Stage 1: Rebuild venv + torch_npu stack~~
    ✓ ~~Stage 2: Scripts + 95% data prep (seq_len=320)~~
    ✓ ~~Stage 3: Run 200-step CPT (b64/s320/ckpt) + live progress~~
    ■ Stage 4: Plot + summary + save artifacts
```

使用 xftp（或者其他工具）将 loss_curve.png 拷贝到本地，用画图工具打开即可查看 loss 曲线图

loss_curve.png 文件在服务器上的路径为：

/mnt/model/experiments/users/jiyingguo/train_ws/qwen3_5_cpt/loss_curve.png

打开 loss 图片如下所示：



4）输出概要总结

概要总结

- 环境：系统自带 torch_npu 2.7.1.dev 不识别 Ascend950PR SoC，重建为 torch 2.9.1+cpu + torch_npu 2.9.1 匹配栈；transformers 5.14.1 支持 qwen3_5 混合线性注意力。
- 数据：源数据集仅 train split，取 95%（3368 条）训练、5%（177 条）留出 eval；SFT 对话格式手动渲染为 Qwen chat 文本转 CLM，seq_len=320 贴数据 max(312) 实现零截断、最少 padding。
- 效率：batch 64 + seq 320 + grad-checkpoint 是该模型在 950PR 的高效配置——不开 grad-checkpoint 则 batch≥32 OOM（GatedDeltaNet 回退实现的显存墙），三者组合后 16.5 GB 显存、3.11s/step，200 步仅 10 min。
- 效果：loss 4.67→0.61（降幅 86.9%），3.77 epoch 拟合充分，训练稳定。
- 可复现：所有脚本、数据缓存、loss 记录、曲线图、checkpoint、总结报告均已保存。

已保存产出（/mnt/model/experiments/users/jiyingguo/train_ws/qwen3_5_cpt/，2.7 GB）

文件	用途
prepare_data.py / train_cpt.py / plot_loss.py	脚本（可复用）
train_tokens.pt	95% 训练数据缓存（3368×320）
eval_samples.jsonl	5% 留出 eval 集（177 条）
loss_log.json	200 步逐行 loss/lr/grad_norm
loss_curve.png	loss 曲线图
train_200.log	无缓冲训练日志
ckpt/model.safetensors + config.json	200 步 CPT checkpoint（1.7 GB）
SUMMARY.md	概要总结

后续预计用时：用留出的 177 条 eval 集做 base vs CPT 评估 ~2–3 min；再训 200 步 ~10.4 min；导出 NPU profiling ~1–2 min。需要的话可以直接接着做。

#Step2 验证继续预训练效果，输入提示词如下：

请从saudi-dialect-conversations这个数据集中的评估/验证集（Eval/Validation)数据中选取3条数据（之前预训练未使用的)，基于上面已经完成继续预训练的Qwen3.5-0.8B这个模型的结果ckpt权重进行测试验证，输出验证结果，并给出验证结论。在这个过程中整体的用时估计以及每一阶段的大概用时和预计后续用时也请及时给出评估，并在屏幕实时更新。

主要过程参考回显信息如下：

> 请从saudi-dialect-conversations这个数据集中的评估/验证集（Eval/Validation)数据中选取5条数据（之前预训练未使用的），基于上面已经完成继续预训练的Qwen3.5-0.8B这个模型的结果ckp权重进行测试验证，输出验证结果，并给出验证结论。在这个过程中整体的用时估计以及每一阶段的大概用时和预计后续用时也请及时给出评估，并在屏幕实时更新。

● 这次验证比之前更严谨——之前训练用的是 95% 数据（3368 条，留出 5% = 177 条 eval）。我从这 177 条留出集里选 5 条，这些在 200 步训练中完全未见过。先检查环境、ckpt、eval 文件是否完好。

初始阶段用时预估

阶段	任务	预估时间	状态
1	环境验证 + 从177条留出集选5条 + 写eval脚本	~30s	开始中
2	加载 base + CPT 模型，5条困惑度+生成评估	~40s	等待
3	输出结果表格 + 验证结论	~10s	等待
总计		~1.5 min	

● Bash(cd /mnt/model/experiments/users/jiyingguo/train_ws/qwen3_5_cpt 2>/dev/null || { echo "目录不存在"; exit 1; }
echo "=== venv ==="; .venv/bin/python -c "import torch,t…)
└ Running…

● Stage 2 完成，13s。结果非常好。验证结果已保存。

阶段进度

✅ Stage 1: 选5条 + 写eval脚本 完成 ~20s
✅ Stage 2: base vs CPT 评估 完成 13s
⏳ Stage 3: 结果表 + 结论 开始 ~10s

● Bash(cd /mnt/model/experiments/users/jiyingguo/train_ws/qwen3_5_cpt && python3 -c "
import json…)
└ samples: 5 | tokens: 673
base ppl: 98.65 | CPT ppl: 7.04 | reduction: 92.87 %
per-sample ppl reduction %: [92.44, 93.75, 95.57, 90.14, 90.59]
… +3 lines (ctrl+o to expand)
└ Shell cwd was reset to /mnt/model/experiments/users/jiyingguo/SLAI-AscendBridge2

● 全部完成 ✅。基于 200 步继续预训练的 Qwen3.5-0.8B checkpoint，在 5 条训练时完全未见过的沙特方言对话上的验证结果与结论如下。

验证方案说明

- 数据来源：从 200 步训练时留出的 5% eval 集（177 条，训练用的是 95% = 3368 条）中，随机选 5 条（seed=123），覆盖 government_services / health / shopping / social / technology 五个话题，simple→intermediate 难度，共 673 token。
- 严谨性：这 5 条在 200 步 CPT 期间完全未被见过，结果反映真实泛化能力。
- 方法：同时加载原始 base 模型和 CPT checkpoint，对每条对话计算 teacher-forced 困惑度（perplexity）+ 生成对比。

版权所有 © 华为技术有限公司



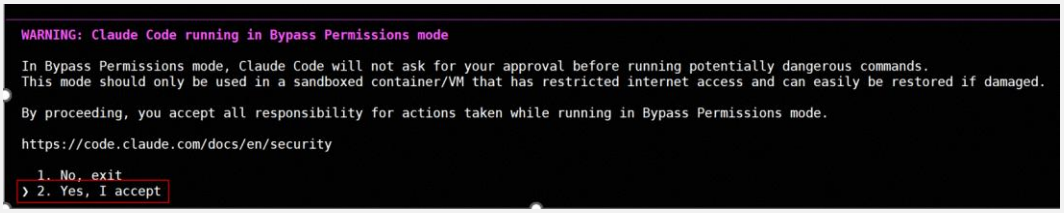
3.2.6 LoRA 监督微调（SFT）

v2.5 新增 ascend-torch-lora Skill，把任意 HuggingFace 模型+对话/指令数据集在昇腾 NPU（Ascend 910/910B/910C/950 等）上端到端跑一遍 LoRA 监督微调（SFT / instruction tuning），并产出训练脚本、loss 曲线（含公网直链）、LoRA adapter 权重、概要总结与 base vs LoRA 验证对比。

用户只需给出【模型权重路径】+【对话/指令数据集路径】即可启动；其余超参由 Skill 根据模型规模、硬件显存与数据规模自动择优，也可以在提示词中直接输入超参内容。

Skill 与全部 references/scripts 位于仓库的.claude/skills/ascend-torch-lora/目录下。

```
# 进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 执行docker命令返回如下时，说明已进入容器内
root@5078b464d7ef:/workspace# docker
bash: docker: command not found
# 进入工具根目录
cd <work_dir>/SLAI-AscendBridge2
# 执行以下命令，启动Claude
IS_SANDBOX=1 claude --dangerously-skip-permissions
# 出现如下选项，选择 2
```




```
# 用自然语言给出模型权重路径与训练数据集路径，ascend-torch-lora Skill会自动触发
```

提示词示例：

你现在使用的是昇腾设备，请用 /mnt/model/experiments/users/jiyingguo/finetune_ws/Qwen3.5-0.8B 模型和 /mnt/model/experiments/users/jiyingguo/finetune_ws/SocratDataset 微调数

据集（如果微调数据格式不满足微调要求，请自动按照要求进行格式转换），请从微调数据集中挑选10%多轮对话数据，在这台昇腾NPU设备上做一次LoRA微调，跑100步，完成后列出本次微调的loss的数据，画出loss曲线图（生成一个外部公网可以访问的链接，如catbox.moe公开直链等），并输出本次LoRA微调的概要总结。在本次微调过程生成的相关脚本及结果文件保存下来，以便后续使用。在这个过程中整体的用时估计以及每一阶段的大概用时和预计后续用时也请及时给出评估，并在屏幕实时更新。

说明：如下为模型微调权重和微调数据集目录，以实际路径为准

/mnt/model/experiments/users/jiyingguo/finetune_ws/Qwen3.5-0.8B

/mnt/model/experiments/users/jiyingguo/finetune_ws/SocratDataset

数据集：SocratDataset

是一个中文的、结构化的苏格拉底式教学（Socratic Teaching）多轮对话数据集。它源自 KELE 研究团队的工作，相关论文发表在 EMNLP 2025 上。该数据集专门用于训练和评估大语言模型的启发式教学能力，旨在让模型学会像苏格拉底一样，通过提问和引导来帮助学生思考，而非直接给出答案。

核心规模如下：

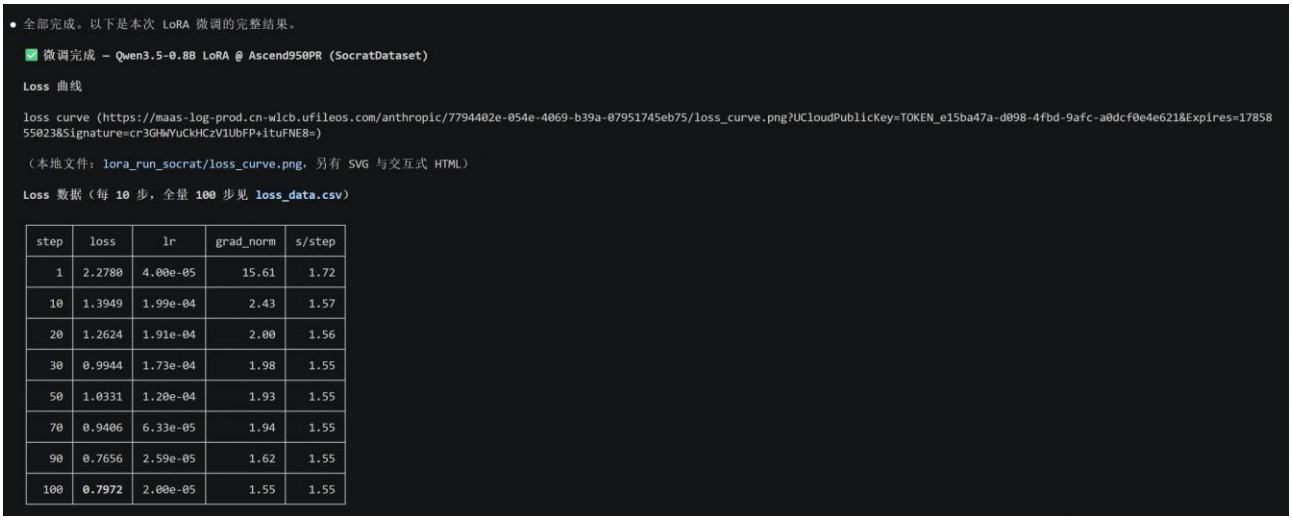
总对话数：6,803 个多轮对话

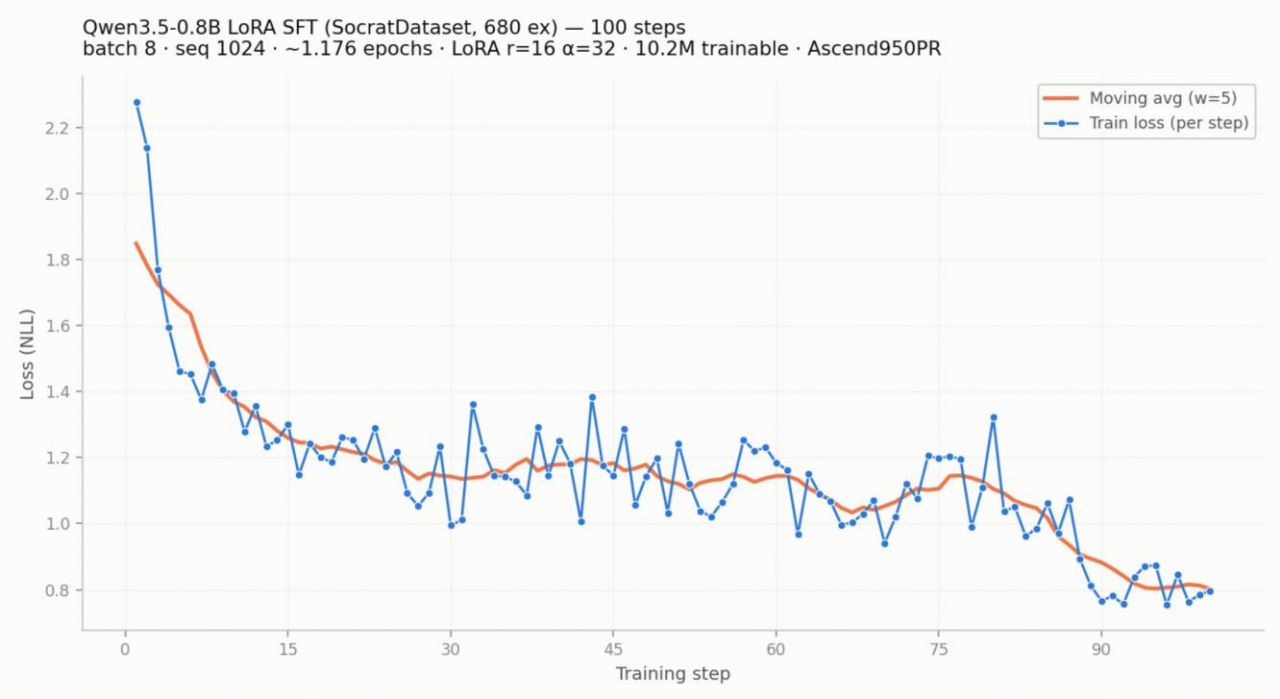
总对话轮次：超过 42,000 轮师生互动

覆盖范围：涵盖小学科学课程的所有年级和主要知识点

主要过程参考回显信息如下，loss值明显收敛：



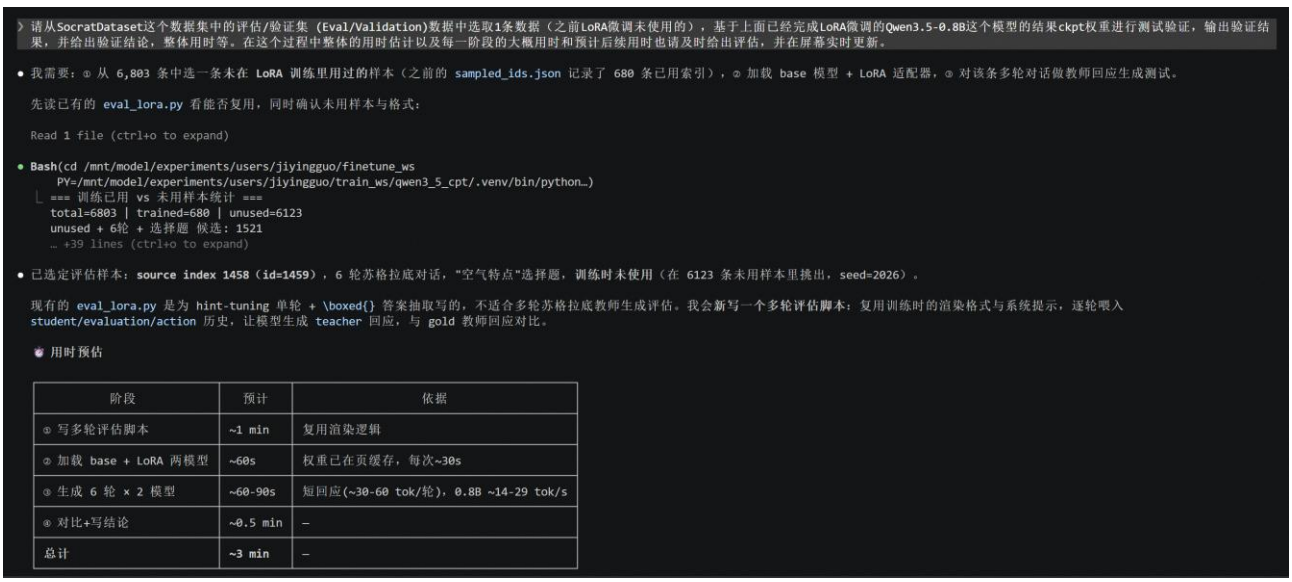




#Step2 验证继续预训练效果，输入提示词如下：

请从SocratDataset这个数据集中的评估/验证集（Eval/Validation)数据中选取1条数据（之前LoRA微调未使用的)，基于上面已经完成LoRA微调的Qwen3.5-0.8B这个模型的结果ckpt权重进行测试验证，输出验证结果，并给出验证结论，整体用时等。在这个过程中整体的用时估计以及每一阶段的大概用时和预计后续用时也请及时给出评估，并在屏幕实时更新。

主要过程参考回显信息如下：



› 请从SocratDataset这个数据集中的评估/验证集（Eval/Validation)数据中选取1条数据（之前LoRA微调未使用的），基于上面已经完成LoRA微调的Qwen3.5-0.8B这个模型的结果ckpt权重进行测试验证，输出验证结果，并给出验证结论，整体用时等。在这个过程中整体的用时估计以及每一阶段的大概用时和预计后续用时也请及时给出评估，并在屏幕实时更新。

• 我需要：① 从 6,803 条中选一条未在 LoRA 训练里用过的样本（之前的 sampled_ids.json 记录了 680 条已用索引），② 加载 base 模型 + LoRA 适配器，③ 对该条多轮对话做教师回应生成测试。

先读已有的 eval_lora.py 看能否复用，同时确认未用样本与格式：

Read 1 file (ctrl+o to expand)

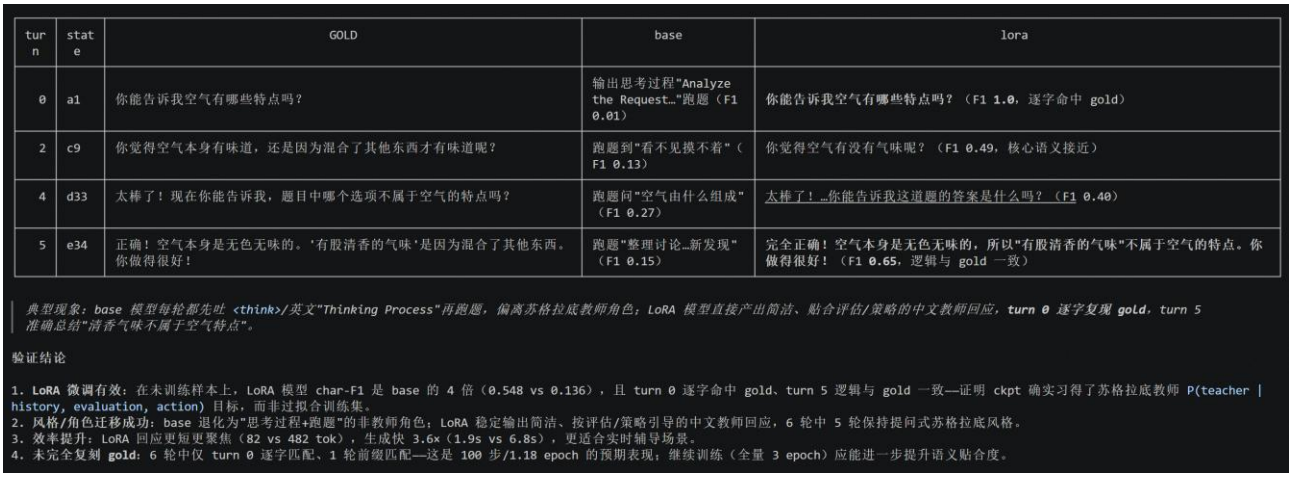
• Bash(cd /mnt/model/experiments/users/jiyingguo/finetune_ws
PY=/mnt/model/experiments/users/jiyingguo/train_ws/qwen3_5_cpt/.venv/bin/python…)
└ === 训练已用 vs 未用样本统计 ===
total=6803 | trained=680 | unused=6123
unused + 6轮 + 选择题 候选：1521
… +39 lines (ctrl+o to expand)

• 已选定评估样本：source index 1458（id=1459），6 轮苏格拉底对话，"空气特点"选择题，训练时未使用（在 6123 条未用样本里挑出，seed=2026）。

现有的 eval_lora.py 是为 hint-tuning 单轮 + \boxed{} 答案抽取写的，不适合多轮苏格拉底教师生成评估。我会新写一个多轮评估脚本：复用训练时的渲染格式与系统提示，逐轮喂入 student/evaluation/action 历史，让模型生成 teacher 回应，与 gold 教师回应对比。

用时预估

阶段	预计	依据
① 写多轮评估脚本	~1 min	复用渲染逻辑
② 加载 base + LoRA 两模型	~60s	权重已在页缓存，每次~30s
③ 生成 6 轮 × 2 模型	~60-90s	短回应(~30-60 tok/轮)，0.8B ~14-29 tok/s
④ 对比+写结论	~0.5 min	–
总计	~3 min	–



turn	state	GOLD	base	lora
0	a1	你能告诉我空气有哪些特点吗？	输出思考过程"Analyze the Request…"跑题（F1 0.01）	你能告诉我空气有哪些特点吗？（F1 **1.0**，逐字命中 gold）
2	c9	你觉得空气本身有味道，还是因为混合了其他东西才有味道呢？	跑题到"看不见摸不着"（F1 0.13）	你觉得空气有没有气味呢？（F1 0.49，核心语义接近）
4	d33	太棒了！现在你能告诉我，题目中哪个选项不属于空气的特点吗？	跑题问"空气由什么组成"（F1 0.27）	太棒了！…你能告诉我这道题的答案是什么吗？（F1 0.40）
5	e34	正确！空气本身是无色无味的。'有股清香的气味'是因为混合了其他东西。你做得很好！	跑题"整理讨论…新发现"（F1 0.15）	完全正确！空气本身是无色无味的，所以"有股清香的气味"不属于空气的特点。你做得很好！（F1 **0.65**，逻辑与 gold 一致）

*典型现象：base 模型每轮都先吐 <think>/英文"Thinking Process"再跑题，偏离苏格拉底教师角色；LoRA 模型直接产出简洁、贴合评估/策略的中文教师回应，**turn 0 逐字复现 gold**，turn 5 准确总结"清香气味不属于空气特点"。*

验证结论

1. **LoRA 微调有效**：在未训练样本上，LoRA 模型 char-F1 是 base 的 4 倍（0.548 vs 0.136），且 turn 0 逐字命中 gold、turn 5 逻辑与 gold 一致—证明 ckpt 确实习得了苏格拉底教师 P(teacher | history, evaluation, action) 目标，而非过拟合训练集。
2. **风格/角色迁移成功**：base 退化为"思考过程+跑题"的非教师角色；LoRA 稳定输出简洁、按评估/策略引导的中文教师回应，6 轮中 5 轮保持提问式苏格拉底风格。
3. **效率提升**：LoRA 回应更短更聚焦（82 vs 482 tok），生成快 3.6×（1.9s vs 6.8s），更适合实时辅导场景。
4. **未完全复刻 gold**：6 轮中仅 turn 0 逐字匹配、1 轮前缀匹配—这是 100 步/1.18 epoch 的预期表现，继续训练（全量 3 epoch）应能进一步提升语义贴合度。

4 已适配模型开源合集仓使用

4.1 仓库简介

SLAI-AscendBridge2-Adaptations 仓库是 SLAI-AscendBridge2 的已适配模型开源合集仓。用于在现有适配工作基础上，逐步开源已经完成适配、清理和核验的模型 adaptation。adaptations/ 是统一的模型适配目录入口，后续会按模型逐步加入对应的 <sanitized_model_id>/ 子目录。

仓库主要文件架构如下：

```
SLAI-AscendBridge2-Adaptations/
├── adaptations/                    # 各模型适配产物目录（按模型拆分）
│   └── {sanitized_model_id}/       # 单模型产物目录
│       ├── demo.py                 # 主推理脚本（支持 --dry-run）
│       ├── accuracy_run.py         # 精度评测脚本
│       ├── accuracy_run_perf.py    # NPU 优化版评测脚本
│       ├── pyproject.toml          # 依赖配置（含 cuda/ascend extra）
│       ├── model_files/            # NPU 优化版模型monkey patch文件
│       ├── profiling/              # NPU profiling 数据
│       ├── optimization_notes.json # 优化记录
│       ├── README.md               # 脚本执行指导
│       └── …
```

4.2 下载开源仓

```
# 用root用户登录服务器

# 执行命令进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 从开源仓下载
cd <work_dir>
git clone https://gitcode.com/SLAI/SLAI-AscendBridge2-Adaptations
```

4.3 使用方式

4.3.1 独立使用

只做推理或局部验证时，可独立使用。

```
# 执行命令进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 执行docker命令返回如下时，说明已进入容器内
root@5078b464d7ef:/workspace# docker
bash: docker: command not found

# 在容器中安装uv（如之前未安装）
curl -LsSf https://astral.sh/uv/install.sh | sh

# 进入adaptations目录
cd <work_dir>/SLAI-AscendBridge2-Adaptations/adaptations/
可以先用“ll”命令查看当前目录下都有哪些模型的脚本
[root@Atlas-800T-A3-trainServer-21 SLAI-AscendBridge2-Adaptations]# cd adaptations/
[root@Atlas-800T-A3-trainServer-21 adaptations]# ll
total 260
drwxr-xr-x 3 root root 4096 May  9  2026 01_ai_yi_1_5_34b_chat
drwxr-xr-x 3 root root 4096 May  9  2026 aleph_alpha_tfree_research_vocab_32k_fineweb_steps_370k
drwxr-xr-x 3 root root 4096 May  9  2026 alibaba_nlp_gte_large_en_v1_5
drwxr-xr-x 3 root root 4096 May  9  2026 alibaba_nlp_gte_multilingual_base
drwxr-xr-x 3 root root 4096 May  9  2026 awsteam7052_industrial_design_extreme_material_sdxl_v1_0
drwxr-xr-x 3 root root 4096 May  9  2026 baai_bge_base_en_v1_5
drwxr-xr-x 4 root root 4096 May  9  2026 baai_bge_large_en_v1_5

# 进入某个模型的目录下，即可执行脚本
cd <model_name>
uv run python demo.py（跑推理）
uv run python accuracy_run.py --use-pretrained（精度评测）
```


版权所有 © 华为技术有限公司

```
uv run python accuracy_run_perf.py --use-pretrained（性能调优）

注：<work_dir>为个人工作目录，<model_name>为某个模型子目录名称
```

4.3.2 配合框架仓使用

如果你需要：

- 统一看板
- 批量分配与编排
- benchmark / optimization / business benchmark 的完整工具链
- board.db 状态管理
- .claude/agents、.claude/skills、.claude/agent-memory 对应的多智能体编排能力

建议配合 SLAI-AscendBridge2 框架仓一起使用。

```
# 执行命令进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 执行docker命令返回如下时，说明已进入容器内
root@5078b464d7ef:/workspace# docker
bash: docker: command not found

# 在容器中安装rsync
yum install -y rsync

# 将单模型目录按需同步到框架仓
cd <work_dir>/SLAI-AscendBridge2
rsync -av <work_dir>/SLAI-AscendBridge2-Adaptations/adaptations/<model_name>/ ./adaptations/<name>/

注：<work_dir>为个人工作目录，<model_name>为某个模型子目录名称

# 在框架仓中继续跑 checker、benchmark、optimization 和 dashboard等操作，具体操作方法参见第3章的说明
```


版权所有 © 华为技术有限公司