按具体版本核对输入规模、硬件与数据库

完整操作教程 · 参考固定 · 往返恢复验收

先确认运行的是AlphaFold2还是AlphaFold3，再检查实际容器内的框架、输入实体和数据库，按失败阶段处理。

运行边界：示例代码需按安装版本和真实项目输入复核；未提供真实生物数据，语法或小样本检查不代表科学结果和性能验收。

“AlphaFold本地跑不动”需要先说清具体实现。AlphaFold2、AlphaFold3和基于它们的第三方入口，输入格式、数据库与资源条件不同。先保存版本、命令和第一条实质错误，再区分环境启动、数据搜索、模型推理或结果写出的失败。

先核对版本与输入合同

AlphaFold2常用FASTA并按单体或多聚体设置运行；AlphaFold3使用自己的JSON格式，可描述多种分子实体。不能把FASTA改后缀当成JSON，也不能把在线服务器导出的输入不经核对直接当作本地原生格式。记录每条链、拷贝数、分子类型和序列来源。

要确认的内容	记录什么	常见混淆
软件	仓库版本或提交、容器镜像	把不同AlphaFold入口当同一个程序
输入	链、长度、拷贝、配体及其他实体	只报最长蛋白链长度
资源阶段	搜索、特征化、推理、保存	把搜索慢都解释成显卡不够
数据库与参数	发布版本、路径、访问条件	下载了一部分就认为安装完成

在真正运行的环境里检查设备

下面命令应在任务实际使用的环境或容器内执行，不只在宿主机运行。NVIDIA路径下可查看nvidia-smi，框架设备则查看JAX。设备编号是该作业可见范围内的编号，应遵守平台分配，不自行占用其他用户的卡。

```
nvidia-smi
python -c "import sys,jax; print(sys.version); print(jax.__version__);
print(jax.devices())"
df -h .
free -h
```

宿主机能看到GPU，但容器中的JAX只列CPU时，应先核对容器映射、驱动与兼容构建。导入库失败或CUDA初始化错误属于环境问题，不能用缩短序列掩盖。主机RAM不足发生在读取或搜索时，也不等于显存不足。

检查AlphaFold3输入的基本规模

下例只盘点原生JSON中的蛋白链，不替代官方输入验证，也不计算所有分子token。保存为inspect_af3.py，执行python inspect_af3.py fold_input.json；脚本不打印具体序列。

```
import json, sys
from collections import Counter

with open(sys.argv[1], encoding="utf-8") as handle:
    job = json.load(handle)
assert isinstance(job, dict) and job.get("dialect") == "alphafold3"
types = Counter()
for entity in job["sequences"]:
    assert len(entity) == 1
    kind, data = next(iter(entity.items()))
    types[kind] += 1
    if kind == "protein":
        ids = data["id"] if isinstance(data["id"], list) else [data["id"]]
        print("protein_ids", ids, "length", len(data["sequence"]), "copies", len(ids))
print("entity_entries", dict(types))
print("model_seed_count", len(job.get("modelSeeds", [])))
```

同一链多个拷贝、核酸、配体和修饰都会影响任务，不能只用上述蛋白长度推断显存。报输入错误时查ID、实体定义和版本；进入推理才分配失败时，再记录完整规模与对应模型设置。

留一份失败登记：输入文件校验值、完整命令、开始时间、退出码和错误前后日志。向管理员报告时优先提供这些材料，并隐藏序列及敏感路径。重装环境前先保留证据，否则无法判断下一次成功来自哪个改动，也难以解释相同输入为何表现不同。

数据库和使用条件分别核对

代码、模型参数和数据库有各自条件，尤其AlphaFold3参数与输出有明确使用范围。按官方文件确认机构、用途及托管方式，租到GPU不自动获得参数或再分发权限。准备环境的人与使用者都应明确由谁提供经过允许使用的参数。

核对数据库清单、解压完成状态、实际挂载路径和工作盘额度；搜索输出、编译缓存与预测结果还需额外空间。确认失败阶段后再做单项调整，并验证原关口通过。小型输入完成只说明软件链可用，不证明正式复合物能放入内存，也不证明预测结构正确。

先确认究竟运行哪个AlphaFold

AlphaFold 2、AlphaFold 3本地代码库、AlphaFold Server和其他封装平台不是同一接口。提交前记录软件入口、版本或commit、运行容器、输入格式和输出目录；AF2通常经其脚本及配置/数据库目录执行，AF3源码仓库使用JSON输入并区分data pipeline与inference阶段。不要通过文件扩展名或网页标题猜版本，也不要把AF2的FASTA命令直接传给AF3。保存实际执行命令和帮助信息，升级后重新核对字段与模型参数位置。

拆分数据准备与GPU推理

AF3官方工作流允许分阶段运行：数据pipeline用于搜索MSA和模板，主要消耗CPU、内存、网络或数据库I/O；推理阶段需要GPU并读取模型参数。若输入已包含合法MSA和模板，可按当前源码支持的选项跳过数据pipeline；空MSA和空模板则是明确选择不使用这些信息，不能把“输入格式接受空值”解释成推荐策略。记录每个阶段是否执行、生成了哪些JSON字段，以及推理是否使用了同一输入版本。AF2常见流程也有搜索与模型推理阶段，但其命令、数据库和模型权重合同不同。

硬件检查要覆盖实际作业额度

nvidia-smi显示宿主机GPU不代表容器或调度任务可以使用该卡；需在作业环境里检查可见设备、显存额度、驱动与CUDA/JAX兼容情况。free -h显示节点总内存，也不等同作业cgroup上限。检查模型参数和数据库是否位于可读路径，权重许可允许的用途与存储方式是否满足机构规则。先提交一个官方小样例，记录data pipeline与推理的资源峰值和日志，再据此判断需要GPU、CPU、内存和高速工作盘的组合。

区分技术验收与结构质量评估

命令成功、输出CIF存在或脚本能读入JSON，只能证明相应步骤通过基本技术验收。结构结果还需检查序列与链映射、残基覆盖、置信度指标、模板/MSA状态及与研究问题相关的独立证据。AF2和AF3的指标、任务范围及结构输出不能简单横向等价；涉及配体或多分子互作时，应确认模型版本确实支持相应实体类型。不要把单次seed或高局部置信度写成实验验证，也不因GPU任务成功就推断模型结构正确。

数据存储与团队交付

把输入序列、JSON或FASTA、MSA/模板来源、软件版本、模型参数版本标识、完整命令、环境信息、日志、输出结构和分析摘要放在同一任务ID下。限制敏感序列和受限权重的访问，确认备份、共享目录及临时清理策略。交接时说明预测对象的序列版本、构建体是否截短或突变、链ID对应关系、使用了哪些搜索阶段及未解决的问题。给出准确的复跑条件，比只发送一个CIF文件更有帮助。

AF2与AF3硬件预算不要混用

AF2与AF3的代码、数据库布局、模型参数、推理流程和资源需求各有差异；即便都用GPU，也不能按一个软件的试跑时间直接估算另一个。分别记录数据准备CPU时间、模型推理GPU时间、峰值显存与RAM、数据库读取和临时目录。AF3运行固定输入时，seed数量、实体类型、序列长度和模板/MSA状态都会影响成本；AF2则需按其实际配置与模型数估算。硬件申请应基于目标软件与实际输入的试跑，而非“AlphaFold任务”这一笼统名称。

输入和输出按版本建立关联

每次预测为输入JSON或FASTA计算校验和，并将其与输出结构、log、summary和运行环境记录绑定。相同文件名并不代表序列或MSA相同；合作者重跑时应能验证输入哈希是否一致。AF3单个输入JSON可包含多个seed，但运行输出的目录规则也要按源码版本确认；批量目录中每个任务使用唯一名称，避免同名覆盖。AF2输出目录常含多模型结果与ranking文件，需按该版本说明解释，而不是套AF3的文件名。

云端与本地运行的隐私差别

序列若涉及未发表项目、专利或临床相关样本，应先确认机构的数据分级和平台政策。公开服务器、商业云服务、本地实验室服务器和共享集群的数据流向、日志保存、备份和访问人员并不相同。把数据放在“远程服务器”也不天然代表安全，仍需核实网络入口、账号权限、加密备份、临时文件清理和服务商处理边界。模型权重许可和数据许可是两项不同约束，计算方案应同时核对。

模型输出复核和再运行策略

若预测结果要用于构建设计或机制假设，选取关键区域与已知结构、保守位点、实验突变或交联等独立证据比较。多seed或重复运行应按预先规则汇总，不能事后只挑最符合预期的模型。结构明显不合理时先检查序列、实体类型、链映射、MSA/模板和日志，必要时做独立建模或实验验证。最终报告写清哪些结论由模型支持、哪些仍是待验证假设，避免把预测坐标当作实验测定结构。

选本地运行还是使用托管服务

先列出序列是否敏感、是否需要配体/复合物、任务数量、可用GPU和数据保留要求，再比较本地代码、机构集群和托管平台。不同方式可能在可预测输入类型、配额、隐私、结果可复现性、许可及算力费用上各有差异。不要假设托管服务与开源代码库接口相同，也不要假设本地部署就自动满足模型权重许可。对尚未公开的序列，先核对数据审批与存储边界，再决定运行位置。

一次任务可以用哪些指标估算资源

试跑记录序列数和长度、链数、seed数、是否搜索MSA/模板、CPU线程、峰值内存、GPU型号与峰值显存、阶段耗时和存储增长。对批量工作，按长度与实体复杂度分层，而不是只按任务个数估算。若推理阶段显存不足，先检查单任务配置、并行数和输入复杂度，并参考对应版本文档；数据搜索阶段慢则检查数据库位置和CPU/I/O。把测试样本和生产样本区别标记，避免把轻量样例的耗时直接线性外推。

安装不等于验证成功

AF2/AF3安装完成后，先运行官方提供的最小示例并检查退出码、阶段日志、输出结构及硬件使用。能import软件包或看到GPU设备，只能证明部分依赖可用；还需确认模型参数能加载、MSA/模板数据库能读取、输入schema通过，并且输出目录可写。验证过程不需要真实课题序列；小型公开示例可检查基础环境。真实样本结果仍要单独做序列、结构与科学解释审查。

记录结论能回答什么

方法部分说明使用的具体软件、版本、输入实体、MSA/模板来源、运行阶段和模型种子，并把置信度及未验证区域作为结果的一部分。结构模型可以帮助形成假设或设计实验，但不独立证明结合、功能或因果关系；结论需要实验或其他独立数据支持。

归档和复核提醒

AF3的外部MSA或分阶段运行要按当前JSON schema准备；AF2则按其自身配置文件和数据库合同运行。每次任务保存原始输入与校验和，避免同名文件更新后无法判断模型结果对应哪条序列。

若模型运行依赖外部数据库搜索，固定数据库发布日期与检索参数；只保存模型版本而不保存MSA来源，通常不足以复现同一预测输入。

配套练习包

本PDF对应的ZIP包含本文代码块、主题练习模板或合成数据、README和SHA256校验文件。请从同一教程页面下载练习包；先核对校验值，再在独立目录复现。

参考资料

AlphaFold2官方安装与数据库配置 · 来源资料

AlphaFold3官方运行与分阶段执行 · 来源资料

AlphaFold3输入格式 · 来源资料

AlphaFold3模型参数使用条款 · 来源资料

操作建议：先在合成小样本上验证流程，再依据项目参考、标签合同、存储策略与获配资源验收真实数据。