Tangram资源评估：先核模式与训练基因，再估映射矩阵

稀疏h5ad不等于映射全程低内存。本文用输入维度估算Tangram dense张量下限，核对训练基因、映射模式和密度先验，再用小试跑判断资源。

Tangram把单细胞表达映射到空间位置。评估内存时不能只看h5ad文件大小：当前实现会将训练基因上的表达矩阵转为float32，并创建“细胞×空间位置”的映射矩阵。这个矩阵本身可能比输入文件大很多。下面给出检查方法；实际模式名称以锁定版本的源代码为准，因为部分文档写cells、源代码文档字符串则写cell，但当前实现的模式检查使用cells。

先定映射单位与输入尺度

选择	适用思路	需核验
cells	逐细胞映射，矩阵较大	显存与细胞数×位置数
clusters	按注释聚合后映射	cluster_label在obs中且解释粒度合适
constrained	有可信细胞总数约束	target_count及位置密度先验来源
density_prior	空间位置的相对密度	RNA比例不等于真实细胞数

参考与空间AnnData要用相容的基因标识和可比的表达尺度，不要一份用原始counts、另一份用log归一化值。避免重复或大小写混杂的基因名；标记基因应覆盖目标细胞类型且在空间数据中有信号。tg.pp_adatas会筛掉不共有或全零的训练基因，并把共同基因顺序写入uns；调用后要检查training_genes数量和顺序。它默认会把基因名转小写，若名称区分大小写，明确设置gene_to_lowercase并核验ID体系。

先估显存下限

令C为参考细胞数、S为空间位置数、G为实际训练基因数。仅表达输入和映射矩阵的float32体积约为4×(C×G+S×G+C×S)字节；优化器状态、梯度和临时副本会继续增加峰值。这只是下限估算，不能直接当作可用显存配置。cluster模式把C换成细胞群数，通常能明显减小映射矩阵，但改变了映射单位，也会改变分析解释。

```
import inspect, json, time
from importlib.metadata import version
from pathlib import Path
import anndata as ad, numpy as np, torch, tangram as tg
from scipy import sparse

scd = ad.read_h5ad("single-cell.h5ad")
spd = ad.read_h5ad("spatial.h5ad")
for obj in (scd, spd):
    assert obj.n_obs > 0 and obj.n_vars > 0
    assert obj.obs_names.is_unique and obj.var_names.is_unique
    vals = obj.X.data if sparse.issparse(obj.X) else np.asarray(obj.X)
    assert np.isfinite(vals).all() and (vals >= 0).all()
genes = Path("training-genes.txt").read_text().splitlines()
assert genes and len(set(genes)) == len(genes)
assert set(genes) <= set(scd.var_names) and set(genes) <= set(spd.var_names)
tg.pp_adatas(scd, spd, genes=genes, gene_to_lowercase=False)
assert list(scd.uns["training_genes"]) == list(spd.uns["training_genes"])
C, S, G = scd.n_obs, spd.n_obs, len(scd.uns["training_genes"])
print({"tangram-sc":version("tangram-sc"), "torch":version("torch"),
       "api":str(inspect.signature(tg.map_cells_to_space)),
       "source":inspect.getsourcefile(tg.map_cells_to_space),
       "C":C, "S":S, "G":G, "lower_bound_GiB":4*(C*G+S*G+C*S)/1024**3})
mode = "cells"  # ■■■■■■■■■■mode■■
device = "cuda:0" if torch.cuda.is_available() else "cpu"
if device.startswith("cuda"): torch.cuda.reset_peak_memory_stats(0)
t0 = time.perf_counter()
mapped = tg.map_cells_to_space(scd, spd, mode=mode, device=device,
            num_epochs=100, random_state=101, density_prior="rna_count_based")
if device.startswith("cuda"): torch.cuda.synchronize()
report = {"mode":mode, "device":device, "seconds":time.perf_counter()-t0,
          "cuda_allocated_peak":torch.cuda.max_memory_allocated(0) if
device.startswith("cuda") else None}
out = Path("tangram-probe-001"); out.mkdir()
mapped.write_h5ad(out/"mapping.h5ad")
(out/"resource.json").write_text(json.dumps(report, indent=2), encoding="utf-8")
```

代码在当前上游实现中使用mode="cells"；先查看打印的Tangram版本、函数签名和源文件位置，再在该版本源代码中核对模式检查。文档与实现不一致时，不要靠反复试参数猜接口，锁定兼容的release或commit并记录。100轮只是短资源探测，不能用于判断映射是否收敛。

按顺序试跑与检查

先在代表性子集上核输入和输出，再在完整数据上测峰值；子集的训练时间不能简单按比例外推，因为映射矩阵随C×S增长。改变细胞抽样、训练基因数或空间位置数时一次只改一项，并记录抽样规则。GPU峰值可读PyTorch统计，但不等于整卡占用；同时用系统工具记录进程RSS和设备整体显存。若单细胞模式超出预算，可评估clusters模式，但需保留足够细的细胞类型标签，不要只为省内存把不同亚群合并。

density_prior="rna_count_based"按空间位置RNA分子数形成先验，不是实测细胞数；单细胞分辨率数据可评估uniform。constrained模式需要有依据的target_count和归一化密度先验，不能把spot数直接当细胞数。映射后检查训练分数、低分基因及稀疏度差异，并把细胞注释投影到空间坐标查看已知标记和组织结构。分数平台或图像合理都不是独立验证，仍要结合实验设计和独立标记。

保存输入文件哈希、软件版本、训练基因表、模式、先验、随机种子、C/S/G、显存与RAM峰值和输出文件。只有这些实测显示现有设备无法完成目标规模时，再按C×S×G规模评估计算资源。

参考资料

Tangram官方映射教程 官方文档
Tangram官方模式实现与内存数据转换 官方文档
Tangram官方README：cluster映射与内存建议 官方文档