比较批大小、显存和训练耗时后选择GPU租赁方案

用同一输入和新进程比较batch、训练时间与显存峰值，保留失败日志，并区分资源试跑和正式模型验收。

scVI能在GPU上训练，但batch越大不一定越快，也不一定适合当前显卡。batch size会影响每个优化步骤的显存、每轮步数和梯度更新，因此应在相同输入、模型和训练设置下做小规模对照，再看显存余量、耗时和训练曲线。本文给出一套记录方法，用来帮助估算GPU资源；候选参数是否科学仍要按研究设计评估。

先固定输入和模型设置

使用相同的AnnData对象、细胞和基因集合、counts层、批次字段、数据拆分、模型结构、随机种子和训练轮数。scvi-tools的SCVI.train支持batch_size参数；先确认表达层为适用的原始计数，并按数据实际结构注册批次。下方代码假设obs中有名为batch的技术批次列；没有该列或不需要校正时，删去batch_key参数。病例组别不能未经设计说明就拿来代替技术批次。记录输入细胞数、基因数、稀疏格式、软件版本、CUDA和显卡型号，避免每种batch实际训练了不同数据。

候选值可以从较小batch开始，再逐步增大，例如64、128、256只是试验点，不是推荐值。先用短跑淘汰明显OOM或速度差的设置；短跑只能筛选资源范围，不能证明模型已经收敛。最终候选仍应在完整预定训练轮数上检查ELBO/损失轨迹、早停与验证指标。

每个候选值放进新进程测量

不同候选batch最好各自启动独立Python进程，避免前一次模型、CUDA缓存或对象残留干扰峰值统计。训练前记录显卡空闲显存；开始训练前重置PyTorch峰值统计，训练结束后同步CUDA，再读取峰值。保存batch值、耗时、退出码、日志、峰值allocated与reserved。前者是PyTorch张量占用口径，后者是缓存分配器管理总量；nvidia-smi还会显示CUDA上下文、其他进程等整卡用量，三者不要混为一个数字。

```
import os, time, torch, scanpy as sc, scvi
adata = sc.read_h5ad("pilot-counts.h5ad")
scvi.model.SCVI.setup_anndata(adata, layer="counts", batch_key="batch")
batch_size = int(os.environ.get("BATCH_SIZE", "128"))
model = scvi.model.SCVI(adata)
torch.cuda.synchronize()
torch.cuda.reset_peak_memory_stats()
t0 = time.perf_counter()
model.train(max_epochs=5, batch_size=batch_size,
            accelerator="gpu", devices=1)
torch.cuda.synchronize()
print({"batch_size": batch_size,
       "seconds": time.perf_counter() - t0,
       "peak_allocated_GiB": torch.cuda.max_memory_allocated()/2**30,
       "peak_reserved_GiB": torch.cuda.max_memory_reserved()/2**30})
```

上面5轮仅作为短跑示例，并非正式模型训练建议。可将代码存成同一个脚本，在每个新进程只改变环境变量BATCH_SIZE，并用不同日志名和资源记录文件；不要在循环内重建数据后改变筛选或拆分。若模型创建、AnnData读取也大量占显存，应同时记录训练前基线，明确比较的是纯训练峰值还是整个进程的峰值。

怎么判断更合适的batch

观察	可能解释	下一步
更大batch省时且显存有余量	吞吐提升，尚未碰到资源上限	完整训练并核对指标
显存接近上限或出现OOM	单步激活/缓存压力过高	退回较小batch，检查余量与失败日志
GPU利用低、训练仍慢	可能在CPU、数据读取或预处理等待	同时看主机内存、I/O和DataLoader
速度变快但损失/验证表现改变	更新步数和优化轨迹改变	按预定模型目标比较，不只看耗时

选型不是单纯挑最大batch。例：某候选每轮快10%，但留给其他任务的显存余量过小，或验证表现不稳定，就未必适合共享GPU服务器。保留代表性数据规模、候选值、每轮耗时、峰值和模型指标，按完整训练的成功率和研究目标决策。若主机内存先耗尽，换更大显存卡也不能解决CPU侧瓶颈。

把试验结果转换成配置需求

至少留出显存余量给CUDA上下文、数据传输和峰值波动，并检查同卡是否有其他进程。最终配置沟通时提供模型、输入规模、PyTorch/CUDA/scvi-tools版本、最大候选batch、allocated/reserved峰值、整卡观察和完整训练耗时。本文未在真实项目数据上测试；显卡型号无法替代同环境实测，batch参数也不能绕开统计和模型质量检查。

参考资料

scvi-tools SCVI.train API（stable官方文档） · 官方资料

PyTorch CUDA内存管理说明（2.15官方文档） · 官方资料

Understanding CUDA Memory Usage（PyTorch 2.15官方文档） · 官方资料