PR #7225 完整报告

verl-project/verl

[algo] fix: micro-batch normalization for distillation loss

合并时间：2026-08-03 11:27

原文链接：http://prhub.com.cn/verl-project/verl/pull/7225

执行摘要

- 一句话：修复蒸馏损失按微批划分的归一化差异
- 推荐动作：值得快速精读，重点学习两个点：一是 agg_loss() 的全局归一化机制与 DDP 梯度平均的关系，二是修复信息放置在分支之前的设计决策。PR body 中的定量对比（mb1 vs mb8、fixOff vs fixOn）是很有说服力的验证范例。若团队有蒸馏训练任务，建议尽早合入并留意是否需要补充 micro-batch 相关的回归测试。

功能与动机

Issue #7200 指出，当 loss_agg_mode=token-mean 且 use_policy_gradient=false 时，ActorConfig.global_batch_info 初始为空，agg_loss() 因拿不到 dp_size 与 batch_num_tokens 而退化为局部微批均值，使累积损失和梯度取决于微批划分方式。PR body 用固定 batch 的实验证明：修复前 mb=1 的 loss 比 mb=8 高约 8 倍；修复后损失与微批大小无关。

实现拆解

1. 变更入口：verl/trainer/distillation/losses.py 的 distillation_loss() 函数，这是蒸馏损失统一聚合入口。
2. 核心变更：在 loss_max_clamp 之后、use_policy_gradient 分支之前，向 config.global_batch_info 写入 data["dp_size"]、data["batch_num_tokens"]、data["global_batch_size"] 以及 config.loss_scale_factor。
3. 作用机制：监督分支调用 agg_loss(loss_mat=..., loss_mask=..., loss_agg_mode=..., **config.global_batch_info) 时，能按 masked_sum / batch_num_tokens * dp_size 计算全局 token 均值，与 DDP 的梯度平均约定一致；policy gradient 分支通过 for k,v in config.global_batch_info.items() 将同一份信息复制到 loss_config，保证两种损失使用同一归一化口径。
4. 关键点：赋值放在 if use_policy_gradient 之前，而不是监督分支内部，确保两个分支都能读取，避免 policy gradient 分支仍缺失全局批次信息。
5. 测试与实验：本 PR 未新增自动化测试；PR body 提供了 Qwen3-0.6B 学生 / Qwen3-1.7B 教师模型在 gsm8k 上的数值对比（RolloutSkip、固定 batch），显示 fixOff 与微批大小相关、fixOn 后损失对微批大小不变。

关键文件：

- verl/trainer/distillation/losses.py（模块 蒸馏损失；类别 source；类型 core-logic；符号 distillation_loss）：蒸馏损失聚合的核心函数所在文件，本次修复在此文件内完成，是 PR 唯一变更文件。

关键符号：distillation_loss

关键源码片段

verl/trainer/distillation/losses.py

蒸馏损失聚合的核心函数所在文件，本次修复在此文件内完成，是 PR 唯一变更文件。

```
# 该段位于 loss_max_clamp 之后、use_policy_gradient 分支之前，
# 保证监督与 policy gradient 两条路径都能拿到全局批次信息。
# 若此处为空，agg_loss() 会退化为局部微批均值，使损失依赖微批划分。
config.global_batch_info["dp_size"] = data["dp_size"]
config.global_batch_info["batch_num_tokens"] = data["batch_num_tokens"]
config.global_batch_info["global_batch_size"] = data["global_batch_size"]
config.global_batch_info["loss_scale_factor"] = config.loss_scale_factor

if loss_config.use_policy_gradient:
    # 将同一份全局批次信息复制到 loss_config，供 policy loss 使用，
    # 使两种损失共享同一归一化口径。
    for k, v in config.global_batch_info.items():
        loss_config.global_batch_info[k] = v
    # ... 计算 policy gradient 蒸馏损失 ...
else:
    # 监督路径直接把全局批次信息交给 agg_loss()，
    # agg_loss() 内部按 masked_sum / batch_num_tokens * dp_size 计算，
    # 归一化分母是全局 token 数，与 DDP 梯度平均约定一致。
    distillation_loss = agg_loss(
        loss_mat=distillation_losses,
        loss_mask=response_mask,
        loss_agg_mode=loss_agg_mode,
        **config.global_batch_info,
    )
```

评论区精华

ErenAta16 在 review 中提出关键质疑：最初的赋值出现在监督分支（else）内，而 policy gradient 分支在更早处复制 config.global_batch_info，若调用方未预填充该字典，PG 分支将拿不到全局批次信息，可能与监督分支行为不一致。结论：作者通过“Fix for pg branch”提交将赋值移到 if 分支之前，使两条路径都能看到全局批次信息，讨论已解决，wuxibin89 随后批准合并。

- global_batch_info 赋值位置是否覆盖 policy gradient 分支 (correctness): 作者将赋值移动到 if 分支之前（对应提交 “Fix for pg branch”），使两个分支都能读取全局批次信息，问题已解决。

风险与影响

- 风险：
  1. 缺少自动化测试：本次修改没有新增测试文件，仅依赖 PR 中的手工实验和现有蒸馏 e2e 测试，回归风险需靠后续测试补齐。
  2. 数据结构依赖：修复后 data 中必须存在 dp_size、batch_num_tokens、global_batch_size 三个字段，若某些调用路径未填充将触发 KeyError；按当前数据流这些字段由 DistillationData 提供，但兼容性风险仍存在。
  3. 行为变化影响存量实验：修复前使用小微批（如 mb=1）训练出的损失、梯度与修复后不同，已有蒸馏实验的数字与曲线可能无法直接对比。
  4. loss_scale_factor 被写入 global_batch_info 并通过 **kwargs 传入 agg_loss()，若用户配置了非默认 loss scale，蒸馏损失的幅度会相应变化，这是修复预期的一部分但也是可见行为变化。 - 影响：影响所有使用蒸馏且 loss_agg_mode=token-mean、use_policy_gradient=false 的训练任务：修复后标量 loss 与梯度不再依赖微批大小，目标函数符合配置的全局 token 均值语义。政策梯度蒸馏分支也会受影响（现在会拿到全局批次信息），但归一化口径更一致。对团队而言，正在跑蒸馏实验的用户需要重新校准损失曲线与超参，仓库代码影响面集中在一个函数。 - 风险标记：缺少测试覆盖，依赖 data 字段存在，行为变化影响存量实验

关联脉络

- 暂无明显关联 PR