PR #5969 完整报告

verl-project/verl

[data, trainer] fix: batch padding for multi-trajectory

合并时间：2026-04-16 23:26

原文链接：http://prhub.com.cn/verl-project/verl/pull/5969

执行摘要

- 一句话：修复多轨迹训练中批次样本数不满足数据并行和 PPO 小批次可除性导致的错误。
- 推荐动作：该 PR 值得精读，特别是 padding_utils.py 中的设计决策：如何构造最小填充样本、处理多维位置 ID 和专家路由、以及通过 is_padding 标签隔离指标计算。这些设计对于理解分布式训练中的数据对齐和性能权衡有重要参考价值。同时，应关注 review 中提到的性能瓶颈和边缘情况，在实际部署中监控 I/O 开销和指标准确性。

功能与动机

根据 PR 描述，当前 tq_trainer 中的 AgentLoopWorkerTQ 已支持多轨迹功能，但在实际训练中，每个批次仍需进行样本（轨迹）级别的填充，以使样本数同时能被 dp_size 和 mini_batch_size 整除，否则会抛出错误。现有方法 pad_dataproto_to_divisor() 会复用前 pad_size 个现有样本，这会污染 GRPO 优势、梯度和指标，导致不一致。因此需要一种新的填充机制，确保填充样本使用独立 UID、最小化计算开销，并通过 is_padding 标签避免影响准确性指标。

实现拆解

1. 新增填充工具模块：创建 verl/trainer/ppo/padding_utils.py，包含四个核心函数：
   - build_padding_position_ids：根据源样本的维度和设备，构建与源样本形状一致的填充位置 ID。
   - build_padding_routed_experts：为 MoE 模型构建零值的路由专家张量，匹配源样本的每 token 专家形状。
   - construct_minimal_padding_template：基于一个真实样本，构造一个仅含一个提示 token 和一个响应 token 的最小填充模板，并设置 is_padding 标签。
   - upsample_batch_to_divisible_size：计算需要填充的样本数，循环调用 construct_minimal_padding_template 生成填充样本，并通过 tq.kv_batch_put 添加到批次中。
2. 修改主训练器逻辑：在 verl/trainer/main_ppo_sync.py 中：
   - 新增 _get_required_batch_multiple 方法，根据 dp_size、critic 和 actor 的 PPO 小批次大小（考虑 rollout.n）计算所需的最小公倍数。
   - 在 _balance_batch 方法中，调用 upsample_batch_to_divisible_size 进行批次上采样，替换原有的直接报错逻辑。

- 在 _compute_metrics 方法中，根据 is_padding 标签过滤填充样本，确保指标计算仅基于真实数据。

3. 测试与配置配套：本次改动未包含直接的测试文件变更，但 PR 描述中通过实验（使用 dp=2, batch_size=45, mini_batch_size=15, rollout.n=8 等参数）进行了验证，并附带了训练曲线截图。

关键文件：

- verl/trainer/ppo/padding_utils.py（模块 填充工具；类别 source；类型 core-logic；符号 build_padding_position_ids, build_padding_routed_experts, construct_minimal_padding_template, upsample_batch_to_divisible_size）：新增的核心填充工具模块，包含所有填充相关的辅助函数和主上采样逻辑，是本 PR 的技术核心。
- verl/trainer/main_ppo_sync.py（模块 训练器；类别 source；类型 entrypoint；符号 _get_required_batch_multiple）：主训练器文件，修改了批次平衡和指标计算逻辑，集成了填充工具，是变更的入口点。

关键符号：build_padding_position_ids, build_padding_routed_experts, construct_minimal_padding_template, upsample_batch_to_divisible_size, _get_required_batch_multiple

关键源码片段

verl/trainer/ppo/padding_utils.py

新增的核心填充工具模块，包含所有填充相关的辅助函数和主上采样逻辑，是本 PR 的技术核心。

```
def construct_minimal_padding_template(
    source_td: dict,
    source_tag: dict,
    eos_token_id: int,
) -> tuple[dict, dict]:
    """构造一个仅含一个提示token和一个响应token的最小文本填充模板。

    Args:
        source_td: 从TransferQueue检索的单个样本字典。
        source_tag: 该样本对应的标签字典。
        eos_token_id: 分词器的EOS token ID。

    Returns:
        一个元组 (template_sample, template_tag)，准备用于填充。
    """
    # 从现有样本复制模板
    template_sample = {}
    for key in source_td.keys():
        value = source_td[key]
        template_sample[key] = value.clone() if isinstance(value, torch.Tensor) else copy.
        deepcopy(value)

    # 从现有样本深拷贝标签模板
```

```
    template_tag = copy.deepcopy(source_tag)

    # 构建最小序列：一个提示token（EOS）和一个响应token（EOS）
    prompts = torch.full((1,), eos_token_id, dtype=torch.int64)
    input_ids = prompts.repeat(2) # 序列长度为2
    attention_mask = torch.ones_like(input_ids, dtype=torch.int64)
    response_mask = torch.zeros_like(input_ids) # 修复：形状与input_ids一致
    # 使用辅助函数构建与源样本兼容的位置ID
    position_ids = build_padding_position_ids(template_sample.get("position_ids"), attention_mask)
    # 为MoE模型构建零值路由专家张量
    routed_experts = build_padding_routed_experts(template_sample.get("routed_experts"), input_
    ids.size(0))

    # 更新字段并移除冗余部分
    template_sample.update(
        prompts=prompts,
        responses=prompts.clone(),
        input_ids=input_ids,
        attention_mask=attention_mask,
        position_ids=position_ids,
        num_turns=0,
        response_mask=response_mask,
        loss_mask=response_mask,
        rm_scores=torch.zeros_like(response_mask, dtype=torch.float32),
        rollout_log_probs=torch.zeros_like(response_mask, dtype=torch.float32),
    )
    # 为VLM模型保留空的多模态输入字典，确保兼容性
    if "multi_modal_inputs" in template_sample:
        template_sample["multi_modal_inputs"] = {}
    if routed_experts is not None:
        template_sample["routed_experts"] = routed_experts
    else:
        template_sample.pop("routed_experts", None)

    # 添加is_padding标签，以保护指标计算（如响应长度、分数、奖励）
    template_tag.update(is_padding=True, prompt_uid=str(uuid.uuid4()))
    return template_sample, template_tag
```

verl/trainer/main_ppo_sync.py

主训练器文件，修改了批次平衡和指标计算逻辑，集成了填充工具，是变更的入口点。

```
  def _get_required_batch_multiple(self, dp_size: int) -> int:
    """返回下游训练步骤（如critic、actor）所需的全局批次倍数。

    计算逻辑：
    1. 基础倍数为dp_size。
    2. 如果启用critic训练，批次需与critic的PPO小批次大小对齐（考虑rollout.n）。
    3. 如果actor更新已开始（超过critic预热步数），批次也需与actor的PPO小批次大小对齐。
    4. 使用math.lcm计算最小公倍数，确保批次可被所有相关大小整除。
```

```
    """
    required_multiple = dp_size

    # 如果启用critic训练，批次应与critic PPO小批次对齐
    if self.use_critic:
        critic_global_mini_batch_size = self.config.critic.ppo_mini_batch_size
        critic_global_mini_batch_size *= self.config.actor_rollout_ref.rollout.n
        required_multiple = math.lcm(required_multiple, critic_global_mini_batch_size)

    # 如果存在actor更新，批次也应与actor PPO小批次对齐
    if self.config.trainer.critic_warmup <= self.global_steps:
        actor_global_mini_batch_size = self.config.actor_rollout_ref.actor.ppo_mini_batch_size
        actor_global_mini_batch_size *= self.config.actor_rollout_ref.rollout.n
        required_multiple = math.lcm(required_multiple, actor_global_mini_batch_size)

    # 注意 lcm(a, b, c) == lcm(lcm(a, b), c)，因此这是最优的
    return required_multiple
```

评论区精华

1. 填充样本兼容性问题：gemini-code-assist[bot] 指出，原始实现中 response_mask 形状与 input_ids 不匹配、position_ids 可能为 1D 导致 VLM 模型崩溃，以及移除 multi_modal_inputs 和 routed_experts 字段会使填充样本与 VLM/MoE 模型不兼容。这些在后续提交中通过 build_padding_position_ids、build_padding_routed_experts 和保留空字典字段得到修复。
2. 性能瓶颈担忧：gemini-code-assist[bot] 提到 tq.kv_batch_put 是同步 I/O 操作，若频繁调用可能影响训练吞吐量，建议考虑异步或批量优化。此问题在 review 中未被直接解决，但 wuxibin89 的评论推动了工具函数的模块化，可能为后续优化铺路。
3. 指标计算边缘情况：gemini-code-assist[bot] 指出，如果批次全由填充样本组成（non_padding_mask.any() 为 False），指标计算会错误地基于填充数据。PR 通过 metrics_batch = batch.select_idxs(non_padding_mask) if non_padding_mask.any() else batch 处理，但 review 认为这仍可能导致误导，建议更稳健的处理（如跳过计算）。此问题在讨论中未进一步解决。
4. 代码组织建议：wuxibin89 建议将 _upsample_batch_to_divisible_size 及相关辅助函数移动到 verl/trainer/ppo 下的独立文件中，以便 fully_async 策略复用。这直接导致了第三次提交，将相关逻辑提取到 padding_utils.py 模块。

- 填充样本与 VLM/MoE 模型的兼容性 (correctness): 通过后续提交修复：response_mask 形状改为与 input_ids 一致；使用 build_padding_position_ids 处理多维位置 ID；为 multi_modal_inputs 保留空字典；使用 build_padding_routed_experts 构建零值专家张量。
- 性能瓶颈：同步 I/O 操作 (performance): 讨论中未直接解决，但 wuxibin89 的评论推动了工具函数的模块化，可能为后续异步优化铺路。
- 指标计算中的边缘情况 (correctness): PR 通过条件选择处理，但 review 认为应更稳健（如跳过计算）。此问题在讨论中未进一步解决。
- 代码组织与复用 (design): 直接采纳，第三次提交将相关逻辑提取到 padding_utils.py 模块。

风险与影响

- 风险：1. 性能风险：upsample_batch_to_divisible_size 中的 tq.kv_batch_put 是同步 I/O 操作，在训练循环中频繁调用可能成为瓶颈，影响训练吞吐量。 2. 兼容性风险：尽管修复了 VLM 和 MoE 相关字段，但填充样本的构造逻辑（如 multi_modal_inputs 设为空字典）可能仍与某些模型的前向传递不兼容，需在实际训练中验证。 3. 正确性风险：指标计算中，当批次全为填充样本时，现有逻辑可能仍会计算基于填充数据的指标，导致误导性结果。 4. 回归风险：修改了 _balance_batch 的核心逻辑，从直接报错改为上采样，若填充逻辑有误（如 UID 冲突、形状不匹配）可能导致训练失败或结果不准确。
- 影响：1. 对用户的影响：解决了多轨迹训练中因批次大小不可整除而导致的训练中断问题，提升了训练稳定性和灵活性，用户现在可以更自由地设置 rollout.n 等参数。 2. 对系统的影响：填充样本增加了额外的计算和内存开销，但由于采用最小序列（2 个 token），开销相对可控；同时，模块化的设计使 fully_async 等策略可以复用该工具，促进了代码复用。 3. 对团队的影响：引入了一个新的工具模块，需要团队成员熟悉其使用方式和限制；review 中的讨论揭示了 VLM/MoE 兼容性和性能方面的潜在问题，为后续优化提供了方向。
- 风险标记：同步 I/O 性能瓶颈，VLM/MoE 兼容性风险，指标计算边缘情况

关联脉络

- PR #5636 [trainer] feat: multi-trajectory support for AgentLoopWorkerTQ: PR #5969 修复了该 PR 引入的多轨迹功能中批次填充的 bug，两者共同完善了多轨迹训练流程。
- PR #5401 [trainer] fix: pad_dataproto_to_divisor pollution issue: PR #5969 的动机中提到避免使用 pad_dataproto_to_divisor()，因为其会污染数据，该 PR 可能涉及相关问题的早期修复。