

PR #6659 完整报告

verl-project/verl

[model] fix: fix vocab_weights device and shard handling for qwen3_vl.py

合并时间: 2026-07-10 22:57

原文链接: <http://prhub.com.cn/verl-project/verl/pull/6659>

执行摘要

- 一句话: 修复 Qwen3-VL 多模态模型 FSDP2 下 vocab_weights 的 device 与 shard 处理
- 推荐动作: 值得精读, 尤其是使用 Qwen3-VL 多模态模型的用户。该 PR 体现了一个典型的设计决策: 在分布式训练中, 即使并行度 =1, FSDP2 仍可能产生 DTensor 类型的参数, 因此在涉及自定义 forward 路径时需显式进行 DTensor 检测与全量张量提取, 确保与后续运算的类型兼容。

功能与动机

修复 Qwen3-VL 多模态模型在使用 FSDP2 (含 `torch.compile`) 或单卡训练时, 因 `vocab_weights` 为 `DTensor` 或 CPU 张量导致的 `RuntimeError: aten.embedding.default: got mixed torch.Tensor and DTensor`。该问题直接关联 Issue #4756, 且与 Qwen3.5 的已有修复逻辑一致。

实现拆解

1. 导入 DTensor: 在文件顶部新增 `from torch.distributed.tensor import DTensor`。
2. 添加 vocab_weights 的 shard 检测与设备对齐: 在 `forward_with_torch_backend` 函数中, 将 `self.lm_head.weight` 赋值给本地变量 `vocab_weights`。若其为 `DTensor` 类型, 则调用 `.full_tensor()` 获取完整张量, 并 `.to(hidden_states.device)` 将其移至与 `hidden_states` 相同的设备。
3. 复用转换后的张量: 将 `fused_linear_for_ppo.forward` 的 `vocab_weights` 参数从直接使用 `self.lm_head.weight` 改为使用处理后的 `vocab_weights`。
4. 无配置 / 测试变更: 未修改任何配置、测试文件或部署流程。

关键文件:

- `verl/models/transformers/qwen3_vl.py` (模块 模型层; 类别 source; 类型 data-contract; 符号 `forward_with_torch_backend`): 唯一修改的文件, 在 `forward_with_torch_backend` 中添加了 `DTensor` 检测与设备对齐逻辑, 修复了 Qwen3-VL 多模态 GRPO 训练的核心错误。

关键符号: `forward_with_torch_backend`

关键源码片段

verl/models/transformers/qwen3_vl.py

唯一修改的文件，在 `forward_with_torch_backend` 中添加了 DTensor 检测与设备对齐逻辑，修复了 Qwen3-VL 多模态 GRPO 训练的核心错误。

```
def forward_with_torch_backend(
    self: "Qwen3VLForConditionalGeneration",
    input_ids: torch.LongTensor = None,
    labels: Optional[torch.LongTensor] = None,
    temperature: float = 1.0,
    shift_labels: Optional[torch.LongTensor] = None,
    **kwargs,
) -> "Qwen3VLCausalLMOutputForPPO":
    from verl.utils.experimental.torch_functional import FusedLinearForPPO

    outputs = self.model(input_ids, **kwargs)
    hidden_states = outputs[0]

    # ... 中间 labels rolling 逻辑省略 ...

    # 修复：处理 vocab_weights 可能为 DTensor 或位于 CPU 的情况
    # 在 FSDP2（或 torch.compile）下，self.lm_head.weight 可能为 DTensor，
    # 而 hidden_states 是普通 CUDA Tensor，直接混合使用会触发
    # "mixed torch.Tensor and DTensor" 错误。
    # 这里将 DTensor 展开为普通 Tensor 并确保与 hidden_states 同设备。
    vocab_weights = self.lm_head.weight
    if isinstance(vocab_weights, DTensor):
        vocab_weights = vocab_weights.full_tensor().to(hidden_states.device)

    fused_linear_for_ppo = FusedLinearForPPO()
    log_probs, entropy = fused_linear_for_ppo.forward(
        hidden_states=hidden_states,
        vocab_weights=vocab_weights, # 使用转换后的权重
        input_ids=rolled_labels,
        temperature=temperature,
    )
    return Qwen3VLCausalLMOutputForPPO(
        log_probs=log_probs,
        entropy=entropy,
        hidden_states=outputs.hidden_states,
    )
```

评论区精华

核心争议：问题根源与复现条件

- Luosuu质疑：该 PR 修复的是分布式张量问题，但作者描述仅在单卡场景下复现，而近期分布式 Qwen3.5 FSDP 运行正常，不确定此修复的实际作用。

- ocp8深入调查后确认：单卡时 `self.lm_head.weight` 实际存储为 `DTensor`（因为 `FSDP2` 默认将权重保存为 `DTensor`，即使并行度 $=1$ ），而 `hidden_states` 是普通 `torch.Tensor`，两者类型不匹配导致错误。该修复正是通过统一类型来解决此问题，且逻辑与 `Qwen3.5` 中的已有处理一致。
- 结论：确认修复有效，CI 通过后合并。
 - 修复的必要性和复现条件 (correctness): 确认修复有效：`FSDP2` 即使并行度 $=1$ 也会将权重存储为 `DTensor`，导致与普通 `Tensor` 的 `hidden_states` 类型不匹配。

风险与影响

- 风险：低风险。变更仅涉及 `forward_with_torch_backend` 一个函数，增加一行 `DTensor` 检测与转换，该逻辑已在 `Qwen3.5` 模型中验证有效，不会引入回归。但需要注意，`full_tensor()` 会在分布式环境下触发 `all-gather` 通信，可能带来少量通信开销，但属于正确性必需操作。
- 影响：直接影响：修复了 `Qwen3-VL` 多模态模型在 `FSDP2`（含单卡 / 多卡）下 `GRPO` 训练时因权重张量类型 / 设备不匹配导致的 `RuntimeError`，用户可正常使用多模态 RL 训练。
间接影响：无；代码仅针对 `forward_with_torch_backend` 路径。
- 风险标记：缺少测试覆盖，核心路径变更

关联脉络

- 暂无明显关联 PR