

PR #6332 完整报告

verl-project/verl

[fsdp, fully_async] fix: fix CI import fast_pos_embed_interpolate in Qwen3-VL

合并时间: 2026-05-13 21:45

原文链接: <http://prhub.com.cn/verl-project/verl/pull/6332>

执行摘要

- 一句话: 修复 CI 中 Qwen3-VL 的 fast_pos_embed_interpolate 导入问题
- 推荐动作: 本次 PR 是紧急修复, 建议后续跟进:
 1. 将 fast_pos_embed_interpolate 抽取到共享工具模块 (如 verl/models/transformers/vision_utils.py), 消除重复。
 2. 考虑优化实现性能, 避免循环中的 tolist 模式。
 3. 由于 CI 目前不支持 Qwen3.5, 可评估是否需要增加对 Qwen3.5 的 CI 支持, 从根源避免类似问题。

功能与动机

PR body 指出: 在将 Qwen3-VL 适配到全异步训练时, 需要 fast_pos_embed_interpolate 函数。由于当前 CI 不支持 Qwen3.5 (qwen3_5.py 属于 Qwen3.5 模型模块), 从 verl/models/transformers/qwen3_5.py 导入该函数会导致 transformers 库报错。因此需要将函数移动到 Qwen3-VL 自己的模块中。

实现拆解

1. 新增函数定义: 在 verl/models/transformers/qwen3_vl.py 中新增 fast_pos_embed_interpolate 函数 (第 33-95 行), 代码逻辑与 qwen3_5.py 中的同名函数一致, 唯一修改点是将 self.pos_embed 相关的操作显式放到 grid_thw.device 上, 以避免 FSDP2 开启 cpu_offload 时 pos_embed 位于 CPU 导致设备不匹配。
2. 调整导入来源: 在 verl/models/transformers/monkey_patch.py 的 Qwen3-VL monkey patch 分支中, 将 fast_pos_embed_interpolate 的导入语句从 from verl.models.transformers.qwen3_5 import ... 改为 from verl.models.transformers.qwen3_vl import ...。这样 monkey patch 无需再依赖 qwen3_5 模块, 从而避免 CI 中因缺失 Qwen3.5 支持而失败。
3. 函数作用: 该函数实现了对视觉位置嵌入的双线性插值, 用于处理不同分辨率的图像 / 视频输入。它接收 grid_thw (时间、高度、宽度) 张量, 计算 4 个最近邻的索引和权重, 通过 self.pos_embed (一个 Embedding 层) 查找并加权求和, 最后按 merge_size 进行 permute 重排, 返回插值后的 position embedding。

关键文件:

- verl/models/transformers/qwen3_vl.py (模块 模型层; 类别 source; 类型 core-logic; 符号 fast_pos_embed_interpolate) : 新增了 fast_pos_embed_interpolate 函数定义, 是 PR 的核心变更。函数逻辑与 qwen3_5.py 中的副本一致, 但添加了设备对齐的修正。
- verl/models/transformers/monkey_patch.py (模块 模型层; 类别 source; 类型 other) : 修改了 fast_pos_embed_interpolate 的导入来源, 从 qwen3_5 改为 qwen3_vl, 解除对 qwen3_5 模块的依赖, 是解决 CI 导入错误的键。

关键符号: fast_pos_embed_interpolate

关键源码片段

verl/models/transformers/qwen3_vl.py

新增了 fast_pos_embed_interpolate 函数定义, 是 PR 的核心变更。函数逻辑与 qwen3_5.py 中的副本一致, 但添加了设备对齐的修正。

```
def fast_pos_embed_interpolate(self, grid_thw):
    """
    对 Qwen3-VL 的视觉位置嵌入进行快速双线性插值。
    grid_thw: (batch, 3) 每个图像的 (时间, 高度, 宽度) 网格数。
    返回插值后的 patch position embedding。

    注意: 此函数与 qwen3_5.py 中的版本逻辑相同,
    但显式从 grid_thw 获取 device, 避免 FSDP2 cpu_offload 时
    self.pos_embed 位于 CPU 导致设备不匹配。
    """
    grid_thw_list = grid_thw.tolist()
    grid_ts = [row[0] for row in grid_thw_list] # 时间维度
    grid_hs = [row[1] for row in grid_thw_list] # 高度维度
    grid_ws = [row[2] for row in grid_thw_list] # 宽度维度
    # 从 grid_thw 获取 device, 防止 FSDP2 cpu_offload 导致设备不匹配
    device = grid_thw.device

    idx_list = [[] for _ in range(4)]
    weight_list = [[] for _ in range(4)]

    for t, h, w in grid_thw_list:
        # 生成线性插值坐标
        h_idx = torch.linspace(0, self.num_grid_per_side - 1, h)
        w_idx = torch.linspace(0, self.num_grid_per_side - 1, w)

        h_idx_floor = h_idx.int()
        w_idx_floor = w_idx.int()
        h_idx_ceil = (h_idx.int() + 1).clip(max=self.num_grid_per_side - 1)
        w_idx_ceil = (w_idx.int() + 1).clip(max=self.num_grid_per_side - 1)

        dh = h_idx - h_idx_floor
        dw = w_idx - w_idx_floor

        # 计算 4 个最近邻索引
```

```

base_h = h_idx floor * self.num_grid_per_side
base_h_ceil = h_idx ceil * self.num_grid_per_side

indices = [
    (base_h[None].T + w_idx floor[None]).flatten(),
    (base_h[None].T + w_idx ceil[None]).flatten(),
    (base_h_ceil[None].T + w_idx floor[None]).flatten(),
    (base_h_ceil[None].T + w_idx ceil[None]).flatten(),
]

weights = [
    ((1 - dh)[None].T * (1 - dw)[None]).flatten(),
    ((1 - dh)[None].T * dw[None]).flatten(),
    (dh[None].T * (1 - dw)[None]).flatten(),
    (dh[None].T * dw[None]).flatten(),
]

for i in range(4):
    idx_list[i].extend(indices[i].tolist())
    weight_list[i].extend(weights[i].tolist())

idx_tensor = torch.tensor(idx_list, dtype=torch.long, device=device)
weight_tensor = torch.tensor(weight_list, dtype=self.pos_embed.weight.dtype, device=device)
pos_embeds = self.pos_embed(idx_tensor).to(device) * weight_tensor[:, :, None]
patch_pos_embeds = pos_embeds[0] + pos_embeds[1] + pos_embeds[2] + pos_embeds[3]

# 按实际 grid 尺寸分割
patch_pos_embeds = patch_pos_embeds.split([h * w for h, w in zip(grid_hs, grid_ws, strict=False)])

# 按 spatial_merge_size 重新排列
merge_size = self.config.spatial_merge_size
patch_pos_embeds_permute = []
for pos_embed, t, h, w in zip(patch_pos_embeds, grid_ts, grid_hs, grid_ws, strict=False):
    pos_embed = pos_embed.repeat(t, 1)
    pos_embed = (
        pos_embed.view(t, h // merge_size, merge_size, w // merge_size, merge_size, -1)
        .permute(0, 1, 3, 2, 4, 5)
        .flatten(0, 4)
    )
    patch_pos_embeds_permute.append(pos_embed)
patch_pos_embeds = torch.cat(patch_pos_embeds_permute)
return patch_pos_embeds

```

verl/models/transformers/monkey_patch.py

修改了 fast_pos_embed_interpolate 的导入来源，从 qwen3_5 改为 qwen3_vl，解除对 qwen3_5 模块的依赖，是解决 CI 导入错误的`关键`。

```
# 在 monkey_patch.py 的 qwen3_vl 分支中 (约第 426 行)      # 原来:      # from
verl.models.transformers.qwen3_5 import (      # fast_pos_embed_interpolate,      # )
# 现在改为直接从 qwen3_vl 导入:      from verl.models.transformers.qwen3_vl import (
    fast_pos_embed_interpolate, # 从本模块导入, 避免依赖 qwen3_5
forward_with_normal_backend,      patch_qwen3_vl_moe_sparse_moe_block_forward,
    qwen3_vl_base_forward,      )      # 后续 monkey patch 保持不变:
Qwen3VLMoeVisionModel.fast_pos_embed_interpolate = fast_pos_embed_interpolate
Qwen3VLVisionModel.fast_pos_embed_interpolate = fast_pos_embed_interpolate 调整后
, 只要 qwen3_vl 模块本身可导入, monkey patch 就不会因 qwen3_5 缺失而失败。
```

评论区精华

代码审查机器人 (gemini-code-assist[bot]) 提出了两个关注点:

1. 代码重复问题: 指出该函数是从 qwen3_5.py 复制而来, 导致重复代码。建议将其抽取到共享工具模块 (如 vision_utils.py), 以提高可维护性。
 2. 性能瓶颈: 指出在循环中使用 .tolist() 和 list.extend 会导致频繁的张量与 Python 列表转换, 建议使用 torch.cat 和 torch.stack 替代以提升性能。这些评论在 PR 合并前未得到进一步回复或解决, 最终 PR 依然被批准合并。
- 代码重复问题 (design): 未解决; PR 仍以当前方式合并, 但建议后续重构。
 - 性能优化建议 (performance): 未解决; PR 合并时未采纳该建议。

风险与影响

- 风险:
 1. 代码重复导致的维护风险: fast_pos_embed_interpolate 在 qwen3_vl.py 和 qwen3_5.py 中存在两份副本, 未来若修复 bug 或优化性能, 需同步两处修改, 容易遗漏。
 2. 性能风险: 当前实现中反复在 Python 列表与 PyTorch 张量间转换, 对于大规模 grid_thw 输入可能成为瓶颈, 影响训练效率。
 3. 回归风险低: 函数逻辑与原有实现完全一致, 仅修改了设备位置的计算方式, 对功能无负面影响。
- 影响:
 - 用户 / 开发者: 解决了 CI 中 Qwen3-VL 全异步训练脚本的导入错误, 使得相关功能可以在 CI 中正常测试。
 - 系统: 无直接影响, 属于代码组织调整。
 - 团队: 降低了维护便利性, 因为引入了重复代码; 但短期内避免了更复杂的重构。
 - 风险标记: 代码重复, 性能瓶颈风险

关联脉络

- PR #6131 [fsdp, fully_async] feat: add Qwen3-VL-30B-A3B fully async GRPO training script on geo3k: 该 PR 引入了 Qwen3-VL 全异步训练支持, 是本次修复的前置工作。本次

PR 解决了该 PR 中因导入 qwen3_5 导致的 CI 问题。