PR #6715 完整报告

verl-project/verl

[model] feat: support Gemma4 multimodal models in RL (GRPO)

合并时间：2026-06-14 16:21

原文链接：http://prhub.com.cn/verl-project/verl/pull/6715

执行摘要

- 一句话：支持 Gemma4 多模态模型 RL 训练
- 推荐动作：值得精读。变更模式清晰体现了“能力检测优于模型名称判断”的设计原则，对于新增加多模态模型支持有参考价值。可以学习如何用 hasattr 替代 model name 硬编码来保持扩展性。

功能与动机

Gemma4 是多模态模型的重要成员，但 verl 的 VLM 管线假设处理器具有 get_rope_index（M-RoPE）并执行严格 per-image-token 校验，Gemma4 使用标准 1D RoPE，因此无法直接训练。Issue #6341 请求支持 Gemma 3/4 多模态训练，#6030 已添加了 Gemma4 FSDP SFT 支持，本 PR 在此基础上扩展至 RL 训练。

实现拆解

1. 注册 Gemma4Processor（verl/utils/tokenizer.py）：在 hf_processor 的 match 分支中新增 case "Gemma4Processor"，由于 Gemma4 使用标准 1D RoPE（无 get_rope_index），因此不绑定该方法；同时将其 validate_inputs 替换为无操作，绕过该处理器因重构后文本缺少图像占位符而触发的严格校验。
2. 扩展变长多模态键集（verl/utils/model.py）：将 mm_token_type_ids 加入 _VARLEN_MULTI_MODAL_KEYS 集合，使该张量在 per-sample 变长序列下能被正确填充成 batch（而非 torch.cat 失败）。Qwen 在更早阶段已消费 mm_token_type_ids 用于 M-RoPE，不会进入该路径。
3. 回退 1D 位置编码（verl/experimental/agent_loop/agent_loop.py）：在 _compute_position_ids 中，将原先 processor is None 的条件扩展为 processor is None or not hasattr(processor, "get_rope_index")，使缺少 M-RoPE 的处理器（如 Gemma4）直接使用 mask 计算标准 1D 位置编码，而非调用不存在的 get_rope_index。

关键文件：

- verl/utils/tokenizer.py（模块 工具；类别 source；类型 core-logic；符号 hf_processor）：核心入口：注册 Gemma4Processor，绕过其严格的 per-image-token 校验。没有这个变更，Gemma4 在 rollout 后处理阶段因校验失败而崩溃。
- verl/utils/model.py（模块 工具；类别 source；类型 data-contract；符号 _VARLEN_MULTI_MODAL_KEYS）：数据契约变更：将 mm_token_type_ids 加入变长多模

态键集，使其在 batch 时被正确填充而非引发错误。

- verl/experimental/agent_loop/agent_loop.py（模块 Agent 循环；类别 source；类型 core-logic；符号 _compute_position_ids）：位置编码计算的关键回退路径：当处理器没有 get_rope_index（即非 M-RoPE）时，回退到标准 1D 位置编码。

关键符号：hf_processor, _compute_position_ids

关键源码片段

verl/utils/tokenizer.py

核心入口：注册 Gemma4Processor，绕过其严格的 per-image-token 校验。没有这个变更，Gemma4 在 rollout 后处理阶段因校验失败而崩溃。

```
# verl/utils/tokenizer.py 中 hf_processor 函数的多模态处理器匹配分支
match processor.__class__.__name__:
    case "Qwen2VLProcessor":
        from transformers.models.qwen2_vl import Qwen2VLModel
        model_class = Qwen2VLModel
    case "Qwen2_5_VLProcessor":
        from transformers.models.qwen2_5_vl import Qwen2_5_VLModel
        model_class = Qwen2_5_VLModel
    case "Qwen3VLProcessor":
        from transformers.models.qwen3_vl import Qwen3VLModel
        model_class = Qwen3VLModel
    case "Glm4vImageProcessor":
        from transformers.models.glm4v import Glm4vModel
        model_class = Glm4vModel
    case "MllamaProcessor":
        pass # Mllama 也没有 get_rope_index，但不需要额外处理
    case "Gemma4Processor":
        # Gemma4 使用标准 1D RoPE，因此不需要绑定 get_rope_index。
        # 同时禁用其严格的 per-image-token 校验（Qwen 处理器没有这个校验），
        # 因为 verl 重构后的文本会去除图像占位符，导致校验失败。
        processor.validate_inputs = lambda *args, **kwargs: None
    case _:
        raise ValueError(f"Unsupported processor type: {processor.__class__.__name__}")

# 仅对绑定了 get_rope_index 的模型执行后续绑定
if model_class is not None:
    processor.get_rope_index = types.MethodType(model_class.get_rope_index, processor)
    if hasattr(model_class, "get_vision_position_ids"):
        processor.get_vision_position_ids = types.MethodType(model_class.get_vision_position_ids,
        processor)
```

verl/utils/model.py

数据契约变更：将 mm_token_type_ids 加入变长多模态键集，使其在 batch 时被正确填充而非引发错误。

```
# verl/utils/model.py 中定义变长多模态键的集合
```

```
# 这些键对应的张量在 per-sample 变长时会被 pad 成 batch，而非直接 torch.cat。
_VARLEN_MULTI_MODAL_KEYS = {
    "input_features",
    "feature_attention_mask",
    "mm_token_type_ids", # Gemma4 使用该键传递 token 类型掩码，需加入变长处理
}
```

verl/experimental/agent_loop/agent_loop.py

位置编码计算的关键回退路径：当处理器没有 get_rope_index（即非 M-RoPE）时，回退到标准 1D 位置编码。

```
# verl/experimental/agent_loop/agent_loop.py 中 _compute_position_ids 方法

def _compute_position_ids(
    self,
    input_ids,
    attention_mask,
    multi_modal_inputs,
    mm_processor_kwargs: Optional[dict[str, Any]] = None,
) -> torch.Tensor:
    """计算多模态输入的位置编码。"""
    # 文本 -only 或 非 M-RoPE 多模态（如 Gemma4）-> 标准 1D 位置编码
    if self.processor is None or not hasattr(self.processor, "get_rope_index"):
        # 使用 attention_mask 计算 1D 位置编码
        return compute_position_id_with_mask(attention_mask) # (1, seq_len)

    # 以下为 M-RoPE 处理逻辑（Qwen 系列）：
    multi_modal_kwargs = {
        "image_grid_thw": multi_modal_inputs.get("image_grid_thw"),
        "video_grid_thw": multi_modal_inputs.get("video_grid_thw"),
    }
    # ... 后续 M-RoPE 位置编码计算不变
```

评论区精华

审阅者 Luosuu 在 verl/utils/tokenizer.py 中问：“禁用这个（Gemma4Processor 的 validate_inputs）是否有风险？”作者 dbuos 回复“据我所知没有风险”。该对话未产生其他分支或后续讨论，审阅最终批准。

- 禁用 validate_inputs 的风险 (question): 作者 dbuos 回复“no afaik”，审阅者接受该回答并最终批准合并。

风险与影响

- 风险：
  1. 禁用 validate_inputs 可能在 Gemma4 处理器未来更新或不同配置下遗漏输入合法性检查，但图像特征仍由处理器无条件生成，风险有限。

2. mm_token_type_ids 被加入变长键集后，如果其他非 M-RoPE 模型未来也生成该字段但语义不同，可能被错误填充，但目前该字段仅出现在 Gemma4 且路径安全。
3. 无配套测试：PR body 明确声明因多模态需要 GPU 和门控检查点而未新增 CI 测试，回归依赖人工实验。

- 影响：
  1. 用户 / 模型支持：Gemma4（如 google/gemma-4-E2B-it）现可直接用于 GRPO/PPO 训练，需使用 SDPA attention（因 FlashAttention 不支持 head_dim=512）。
  2. 已有功能：Qwen-VL、GLM-4V、MIlama 等已有处理器路径完全不变（通过能力检测隔离）。
  3. 系统：无 API 变更、新配置项或依赖变化。影响范围局限于 agent_loop 和 tokenizer，不涉及核心 trainer 或 rollout 框架。 - 风险标记：缺少测试覆盖

关联脉络

- PR #6030 [fsdp, sft] feat: add Gemma4 FSDP SFT support: PR body 明确提及本 PR 补全了 #6030 在 RL 侧的支持，两者共同为 Gemma4 提供了完整的训练能力（SFT + GRPO/PPO）。
- PR #6710 [trainer] feat: add unify trainer abstraction for sync and async training: PR body 指出本 PR 的修改在子函数层面，因此也覆盖了 #6710 引入的新 AgentLoopWorkerTQ（其覆盖了 orchestrator 但继承相同的子函数）。
- PR #6341 gemma3/4 processor support request: 该 issue 是提出支持 Gemma3/4 多模态训练的功能请求，本 PR 是对该请求的响应。