

PR #7802 完整报告

verl-project/verl

[sglang, rollout] fix: forward video features to SGLang

合并时间: 2026-09-15 14:44

原文链接: <http://prhub.com.cn/verl-project/verl/pull/7802>

执行摘要

- 一句话: SGLang rollout 补齐视频特征转发, 修复视频 VLM 盲答
- 推荐动作: 值得精读, 尤其是维护 rollout/ 多模态链路的同学: 本案是“静默降级类 bug”的教科书样本——TODO 注释无告警、特征算完即丢、两个后端行为不一致, 三者叠加后表现为模型输出流畅但完全虚构。建议重点看三处设计决策: (1) 用 `format="processor_output"` 复用 verl 已算好的 processor 输出而不是让 SGLang 重新抽帧, 避免 fps 采样不一致这一隐形 desync; (2) 把后端差异收敛到 `LLMServerClient.generate` 的一处分支, 并用无条件传参 + `TypeError` 换取“漏改即失败”的可见性; (3) teacher/OPD 路径与 rollout 路径共用同一 payload, 避免修复只覆盖一半。阅读时建议同时关注作者未解决的遗留: 非 Qwen 视频家族的静默丢弃、多轮 ToolAgentLoop 重复传帧、以及缺失的单元测试 (作者愿意贡献 CPU-only mRoPE 对齐测试, 可作为后续 PR 的补位点)。

功能与动机

SGLang rollout 长期不支持视频输入: verl/workers/rollout/sglang_rollout/async_sglang_server.py 中 `# video_data=video_data` 被注释并留有 `TODO: support video input for sglang`, 且没有任何告警。作者在 PR body 中描述后果为 “SGLang therefore receives input_ids containing thousands of <|video_pad|> placeholder tokens but no visual features at all, and the model produces fluent, well-structured, entirely hallucinated answers”, 而 vLLM 会正常转发, 所以两个后端在视频任务上完全不可比 (作者实测 SGLang 改前 val acc $0.4968 \approx$ 语言先验水平, 改后 0.77186 vs vLLM 0.7868)。问题由两个缺口叠加: 一是类型不匹配, SGLang 的 video_data 只接受 path/URL/base64/VideoData/dict, 不接受 verl 传的 `list[tuple[torch.Tensor, dict]]`, 简单取消注释会在 `async_load_mm_data` 失败, 这正是 TODO 未完成的原因; 二是 verl 的 `VLContinuousTokenMixin.render_tokens_with_mm` 用 `build_multimodal_processor_inputs` 算出 `pixel_values_videos / video_grid_thw` 后只返回 `input_ids`, 基类 docstring 明确写着 “Pixel tensors are intentionally not returned here.”, 特征算完即丢。

实现拆解

1. 打开特征出口 (verl/utils/tokenizer/continuous_token.py) :
`ContinuousTokenBuilder.build_initial_tokens`、`render_tokens_with_mm` 以及 `VLContinuousTokenMixin` 的同名两个方法新增可选参数 `mm_inputs_out`; mixin 在

build_multimodal_processor_inputs 之后执行 mm_inputs_out.update({k: v for k, v in processor_output.items() if k != "input_ids"}) 回填。文本 builder 与 DeepSeekV4ContinuousTokenBuilder.build_initial_tokens 用 del mm_inputs_out 显式忽略；MiniMaxVLContinuousTokenBuilder.render_tokens_with_mm 透传给 super()，DeepSeekVL2ContinuousTokenBuilder.build_initial_tokens 接受并忽略——这两个 override 不改就会在调用方传关键字时抛 TypeError。作者故意无条件传参：漏改的 builder 首调即大声失败，而不是静默退化为盲答。

2. 构造 SGLang payload (verl/experimental/agent_loop/agent_loop.py)：新增静态方法 AgentLoopBase.build_sglang_video_payload(videos, mm_inputs)，返回 [{"format": "processor_output", "pixel_values_videos": ..., "video_grid_thw": ..., "second_per_grid_ts": ...}]；无视频或 processor 未产出 pixel_values_videos 时返回 None。AgentLoopOutput 增加 mm_processor_output 字段承载该 payload。选择透传 processor 输出而非让 SGLang 自行解码，是因为若无 video_metadata，SGLang 会退化到 fps=24 重新抽帧，作者实测 video_grid_thw 从 [[7, 30, 40]] 变成 [[2, 30, 40]]，等于把“无视频”换成“另一组帧”，同样静默破坏 rollout/actor 一致性。
3. 两个 agent loop 接入：single_turn_agent_loop.py 的 run() 中建 mm_inputs = {} → ct_build_initial_tokens(..., mm_inputs_out=mm_inputs) → build_sglang_video_payload(videos, mm_inputs) → server_manager.generate(mm_processor_output=...)，并把 payload 写入 AgentLoopOutput；tool_agent_loop.py 把 payload 暂存在 AgentData.mm_processor_output (_handle_pending_state 构建)，在 _handle_generating_state 每轮 generate 时透传，与其既有 video_data 的传递方式保持一致。
4. 后端分派与请求落地 (verl/workers/rollout/llm_server.py、async_sglang_server.py)：LLMServerClient.generate 先 kwargs.pop("mm_processor_output", None) (两个 server 的 remote 签名都不接受 **kwargs)，当 rollout.name == "sglang" 时用 payload 覆盖 video_data，vLLM 分支保持原始帧；这套写法与同函数既有的 vLLM-only priority 分派同构。async_sglang_server.py 删除 TODO 注释行，把 "video_data": video_data 放回 GenerateReqInput 请求字典——这是本次 bug 的直接落点。
5. 蒸馏 teacher (OPD) 路径与配套：AgentLoopBase._compute_teacher_logprobs 用 getattr(output, "mm_processor_output", None) 取值 (缺字段时安全 no-op)，经 TeacherManager.compute_teacher_logprobs_single 传入 client.generate；teacher client 复用同一个 LLMServerClient，因此 per-backend 分派自动生效。测试与部署配套：无用户可见配置变更；PR checklist 明确未新增单测 /CI，理由是端到端视频 rollout 需要视频 VLM 权重与 GPU，作者提出可贡献 CPU-only 的 mRoPE 对齐测试，最终以人工验证 (vLLM 输入不变性逐样本对比、与 SGLang MRotaryEmbedding.get_rope_index 的位置 id 逐元素比对、AST 检查 7 个 builder 实现签名) 替代。

关键文件：

- verl/experimental/agent_loop/agent_loop.py (模块 智能体循环；类别 source；类型 core-logic；符号 AgentLoopBase.build_sglang_video_payload, AgentLoopOutput.mm_processor_output, AgentLoopBase.ct_build_initial_tokens, AgentLoopBase._compute_teacher_logprobs)：新增

AgentLoopBase.build_sglang_video_payload 公共 helper 与

AgentLoopOutput.mm_processor_output 字段，并把 payload 贯通到蒸馏 teacher 路径，是整个修复的中枢。

- verl/utils/tokenizer/continuous_token.py (模块 分词构建; 类别 source; 类型 core-logic; 符号 VLContinuousTokenMixin.render_tokens_with_mm, VLContinuousTokenMixin.build_initial_tokens, ContinuousTokenBuilder.build_initial_tokens, MiniMaxVLContinuousTokenBuilder.render_tokens_with_mm) : 为 build_initial_tokens / render_tokens_with_mm 增加 mm_inputs_out 出口，是“特征算完即丢”这一根因的修复点; 同时需要适配 MiniMax / DeepSeekVL2 等 override 以免 TypeError。
- verl/workers/rollout/llm_server.py (模块 请求分派; 类别 source; 类型 core-logic; 符号 LLMServerClient.generate) : LLMServerClient.generate 是后端分派点: 对 SGLang 用 processor payload 覆盖 video_data, 对 vLLM 保留原始帧, 并 pop 掉 server 签名不接受的 mm_processor_output 关键字。
- verl/experimental/agent_loop/single_turn_agent_loop.py (模块 单轮循环; 类别 source; 类型 core-logic; 符号 SingleTurnAgentLoop.run) : 单轮 agent loop 的接入点: 创建 mm_inputs、传给 ct_build_initial_tokens、调用 helper 生成 payload, 并在 generate 与 AgentLoopOutput 中透传。
- verl/experimental/agent_loop/tool_agent_loop.py (模块 工具循环; 类别 source; 类型 core-logic; 符号 AgentData.mm_processor_output, ToolAgentLoop._handle_pending_state, ToolAgentLoop._handle_generating_state, ToolAgentLoop.run) : 多轮工具循环的接入点: AgentData 新增 mm_processor_output 暂存字段, 在 _handle_pending_state 构建、_handle_generating_state 每轮透传; 这是 reviewer 要求补齐的覆盖范围。
- verl/workers/rollout/sglang_rollout/async_sglang_server.py (模块 异步服务端; 类别 source; 类型 core-logic; 符号 AsyncSglangServer.generate) : bug 的直接落点: 把被注释掉的 video_data 真正放进 GenerateReqInput 请求, 并删除 TODO。
- verl/experimental/teacher_loop/teacher_manager.py (模块 教师蒸馏; 类别 source; 类型 core-logic; 符号 TeacherManager.compute_teacher_logprobs_single) : 蒸馏 teacher 调用链新增 mm_processor_output 参数并转发给 client.generate, 保证 SGLang teacher 不再丢视频。
- verl/utils/tokenizer/deepseek.py (模块 分词构建; 类别 source; 类型 configuration; 符号 DeepSeekV4ContinuousTokenBuilder.build_initial_tokens) : DeepSeekV4ContinuousTokenBuilder.build_initial_tokens 新增并忽略 mm_inputs_out, 保持签名兼容; 改由 del images, videos, audios, mm_inputs_out 统一声明忽略。

关键符号: AgentLoopBase.build_sglang_video_payload,

VLContinuousTokenMixin.render_tokens_with_mm,

VLContinuousTokenMixin.build_initial_tokens,

ContinuousTokenBuilder.build_initial_tokens, AgentLoopBase.ct_build_initial_tokens,

LLMServerClient.generate, AgentLoopBase._compute_teacher_logprobs,

TeacherManager.compute_teacher_logprobs_single, SingleTurnAgentLoop.run,

ToolAgentLoop._handle_pending_state, ToolAgentLoop._handle_generating_state, AsyncS
glangServer.generate

关键源码片段

[verl/experimental/agent_loop/agent_loop.py](#)

新增 AgentLoopBase.build_sglang_video_payload 公共 helper 与
AgentLoopOutput.mm_processor_output 字段，并把 payload 贯通到蒸馏 teacher 路径，是
整个修复的中枢。

```
class AgentLoopOutput(BaseModel):
    """Agent loop output."""

    prompt_ids: list[int]
    response_ids: list[int]
    response_mask: list[int]
    multi_modal_data: Optional[dict[str, Any]] = None
    """Multi-modal data for multi-modal tools."""
    mm_processor_kwargs: Optional[dict[str, Any]] = None
    """Processor/backend kwargs that must stay aligned across rollout and training paths."""
    # 新增字段：承载 SGLang 视频 payload（见 build_sglang_video_payload），
    # 同时供 OPD teacher 路径复用，保证 SGLang teacher 拿到的是 processor
    # 特征而不是原始帧；无视频时为 None。
    mm_processor_output: Optional[list[dict[str, Any]]] = None


class AgentLoopBase:
    @staticmethod
    def build_sglang_video_payload(videos, mm_inputs: dict[str, Any]) -> Optional[list[dict[str, Any]]]:
        """把已经算好的视频特征包装成 SGLang 的 ``video_data``。

        SGLang 无法接收原始帧，它的 ``video_data`` 只接受 path / URL / base64 字符串或 dict；
        ``mm_inputs`` 是 tokenization 期间通过 ``mm_inputs_out`` 钩子捕获的 processor 输出。
        打上 ``format="processor_output"`` 标签后，``BaseMultimodalProcessor`` 会原样透传该
        dict，
        避免 SGLang 自行重新抽帧、进而与 actor 侧的视频采样不一致。
        """
        # 没有视频、或该模型 processor 未产出视频像素时返回
        # None，调用方可以直接把返回值透传给 generate(...)
        if not videos or "pixel_values_videos" not in mm_inputs:
            return None
        payload = {
            "format": "processor_output",
            "pixel_values_videos": mm_inputs["pixel_values_videos"],
            "video_grid_thw": mm_inputs["video_grid_thw"],
        }
        # second_per_grid_ts 只有部分模型（如 Qwen2.5-VL）会产出，存在才带上
```

```

if "second_per_grid_ts" in mm_inputs:
    payload["second_per_grid_ts"] = mm_inputs["second_per_grid_ts"]
return [payload]

async def _compute_teacher_logprobs(
    self,
    output: AgentLoopOutput,
    prompt_ids: list[int],
    response_ids: list[int],
    validate: bool,
    sample_kwargs: Optional[dict[str, Any]] = None,
) -> None:
    """计算单条样本的 teacher logprobs (蒸馏 / OPD 路径) 。”
    if self.distillation_enabled and not validate:
        # 省略 routing_key 解析等与视频无关的逻辑
        teacher_ids, teacher_logprobs = await self.teacher_server_manager.compute_teacher_logprobs_single(
            sequence_ids=prompt_ids + response_ids,
            multi_modal_data=output.multi_modal_data,
            mm_processor_kwargs=output.mm_processor_kwargs,
            # getattr 兜底：不带该字段的 AgentLoopOutput 是安全 no-op，不会因属性缺失而报错
            mm_processor_output=getattr(output, "mm_processor_output", None),
            routing_key=routing_key,
        )
        output.extra_fields["teacher_ids"] = teacher_ids
        output.extra_fields["teacher_logprobs"] = teacher_logprobs

```

verl/utils/tokenizer/continuous_token.py

为 build_initial_tokens / render_tokens_with_mm 增加 mm_inputs_out 出口，是“特征算完即丢”这一根因的修复点；同时需要适配 MiniMax / DeepSeekVL2 等 override 以免 TypeError。

```

class VLContinuousTokenMixin:
    def render_tokens_with_mm(
        self,
        messages: list[dict[str, Any]],
        images: list[Any],
        *,
        videos: list[Any] | None = None,
        audios: list[Any] | None = None,
        add_generation_prompt: bool = True,
        tools: list[dict[str, Any]] | None = None,
        mm_inputs_out: dict[str, Any] | None = None, # 新增：回填 processor 中除 input_ids 外的特征
    ) -> list[int]:
        """通过 processor 渲染消息（包含全部模态）。

        ``mm_inputs_out`` 非空时，会接收 ``pixel_values_videos`` / ``video_grid_thw`` 等 processor 特征。

```

SGLang 在请求时需要这些特征，因为它的 ``video\_data`` 接不了原始帧；没有这个钩子，这些张量算完就会被直接丢掉。

```
"""
```

```
template_kwargs = dict(self.chat_template_kwargs)
if tools:
    template_kwargs["tools"] = tools
```

```
# 用 processor 的 chat template 渲染，保证占位文本与 legacy rollout 路径一致
text = apply_chat_template(
```

```
    self.processor,
    messages,
    tokenize=False,
    add_generation_prompt=add_generation_prompt,
    **template_kwargs,
)
```

```
# processor kwargs 是构造期捕获的常量，保证初始 prompt 与增量渲染保持一致
proc_kwargs = dict(self.mm_processor_kwargs or {})
```

```
processor_output = build_multimodal_processor_inputs(
    self.processor,
    text=[text],
    images=images if images else None,
    videos=videos if videos else None,
    audios=audios if audios else None,
    mm_processor_kwargs=proc_kwargs if proc_kwargs else None,
)
```

```
# 只回填 input_ids 之外的字段：token 本身仍按原路径返回，不改变既有调用方行为
if mm_inputs_out is not None:
```

```
    mm_inputs_out.update({k: v for k, v in processor_output.items() if k != "input_ids"})
return normalize_token_ids(processor_output["input_ids"])
```

```
def build_initial_tokens(
```

```
    self,
    messages: list[dict[str, Any]],
    *,
    tools: list[dict[str, Any]] | None = None,
    images: list[Any] | None = None,
    videos: list[Any] | None = None,
    audios: list[Any] | None = None,
    mm_inputs_out: dict[str, Any] | None = None,
```

```
) -> list[int]:
```

```
# 把 mm_inputs_out 一路透传到 render_tokens_with_mm，agent loop 只需要跟这一层打交道
return self.render_tokens_with_mm(
```

```
    messages,
    images,
    videos=videos,
    audios=audios,
    add_generation_prompt=True,
```

```

        tools=tools,
        mm_inputs_out=mm_inputs_out,
    )

```

verl/workers/rollout/llm_server.py

LLMServerClient.generate 是后端分派点：对 SGLang 用 processor payload 覆盖 video_data，对 vLLM 保留原始帧，并 pop 掉 server 签名不接受的 mm_processor_output 关键字。

```

class LLMServerClient:
    async def generate(
        self,
        request_id: str,
        prompt_ids: list[int],
        sampling_params: dict[str, Any],
        image_data: Optional[list[Any]] = None,
        video_data: Optional[list[Any]] = None,
        audio_data: Optional[list[Any]] = None,
        mm_processor_kwargs: Optional[dict[str, Any]] = None,
        **kwargs: Any,
    ) -> TokenOutput:
        # 省略 _acquire_server 的完整参数列表 (prompt_ids / sampling_params / image_data / ...
        )
        server_id, server = await self._acquire_server(...)
        try:
            multimodal_kwargs = {}
            if audio_data is not None:
                multimodal_kwargs["audio_data"] = audio_data
            if mm_processor_kwargs:
                multimodal_kwargs["mm_processor_kwargs"] = mm_processor_kwargs

            # SGLang 不能接收原始帧：它的 video_data 只接受 path / url / base64 或 processor_
            # output dict。
            # 因此对 SGLang 后端统一改发预计算好的 payload（模型 processor 未产出 pixel_values_
            # videos 时为
            # None，例如非 Qwen-VL 家族）——宁可丢视频，也不要给 SGLang
            # 传它无法解析的张量。
            # vLLM 继续使用原始帧、不会进入这个分支。两个 server 的签名都不接受
            # **kwargs，所以这里必须 pop。
            mm_processor_output = kwargs.pop("mm_processor_output", None)
            if self.config.actor_rollout_ref.rollout.name == "sglang":
                video_data = mm_processor_output

            # priority 只有 vLLM rollout server 支持；写法与上面的后端分派保持一致
            priority = kwargs.pop("priority", 0)
            priority_kwargs = (
                {"priority": priority}
                if priority != 0 and self.config.actor_rollout_ref.rollout.name == "vllm"
                else {}
            )

```

```

)
output: TokenOutput = await server.generate.remote(
    request_id=self._vllm_request_id(request_id), # 每轮都换新的 request_id
    prompt_ids=prompt_ids,
    sampling_params=sampling_params,
    image_data=image_data,
    video_data=video_data,
    **multimodal_kwargs,
    **priority_kwargs,
    **kwargs,
)
global_steps = output.extra_fields.get("global_steps")
output.extra_fields.setdefault("min_global_steps", global_steps)
output.extra_fields.setdefault("max_global_steps", global_steps)
return output
finally:
    self._release_server(server_id, request_id=request_id)

```

评论区精华

三条 reviewer 意见都来自 wuxibin89，且都在 author 回复后落实到代码：

1. 覆盖范围——“ToolAgentLoop should be adjusted as well, or can we put it in a common place?”；作者同意抽公共 helper，最终 build_sglang_video_payload 落到 AgentLoopBase，两个 loop 都改为单行调用。
 2. teacher 路径——“OPD teacher model should be adjusted as well: .
../teacher_manager.py#L128-L136”；作者确认
TeacherManager.compute_teacher_logprobs_single 转发的是原始 video_data，
SGLang teacher 会像 rollout 一样丢视频，随后把 payload 经
AgentLoopOutput.mm_processor_output → _compute_teacher_logprobs →
TeacherManager → client.generate 打通。
 3. 键名归属——“Is these keys: pixel_values_videos / video_grid_thw /
second_per_grid_ts specific to Qwen series model?”；作者回答这些确实是 Qwen2-VL
/ 2.5-VL / 3-VL 的 processor 输出名，但恰好就是 SGLang
collect_mm_items_from_processor_output (base_processor.py 的
ATTR_NAME_TO_MODALITY) 映射为 Modality.VIDEO 的键，生产端与消费端本就对齐；
非 Qwen 视频家族需两侧同时补键，当前行为是“视频被丢弃（与改前一致）而不是让服务
崩溃”，建议作为 follow-up，该跟进项在讨论中未关闭。作者还主动 flag 了
ToolAgentLoop 的 caveat：多轮场景每轮都会重发初始 prompt 的视频 payload（与原始
video_data 的传法一致），tool 返回的视频仍不支持（tool_agent_loop.py 会抛 "
Multimedia type 'video' is not currently supported"）；并声明两个红灯 CI 任务属于本
PR 未触碰的既有失败。
- ToolAgentLoop 覆盖与公共 helper 抽取 (design): payload 构造下沉为
AgentLoopBase.build_sglang_video_payload，SingleTurnAgentLoop 与
ToolAgentLoop 都改为单行调用；ToolAgentLoop 通过
AgentData.mm_processor_output 暂存、在 _handle_generating_state 每轮透传。

- OPD teacher (蒸馏) 路径同样丢弃视频 (correctness): payload 经 AgentLoopOutput.mm_processor_output → _compute_teacher_logprobs (getattr 兜底, 缺字段安全 no-op) → TeacherManager.compute_teacher_logprobs_single → client.generate, per-backend 分派自动生效。
- pixel_values_videos 等键是否 Qwen 专属 (design): 非 Qwen 视频家族需要两侧同时补键; 当前 payload 为 None 时对 SGLang 会清空 video_data, 即视频被丢弃 (与改前一致) 而不是让服务崩溃, 作者建议作为 follow-up, 讨论中未关闭该跟进项。
- 缺少自动化测试与 CI 红灯归属 (testing): 以人工验证替代自动化测试: vLLM 逐样本首字符差异 0/463、与 SGLang MRotaryEmbedding.get_rope_index 的位置 id 逐元素一致、AST 检查 7 个 builder 实现签名; 合入时没有新增任何测试。

风险与影响

- 风险:
 1. 行为变更显著但方向为正: SGLang 视频任务从“盲答”变为真正使用视觉特征, rollout 分布发生质变 (val acc 0.4968 → 0.77186, KL 0.093-0.114 → 0.013-0.016)。与改动前的 SGLang 实验基线不再可比, 跨版本对齐实验需重新跑; 同时需注意作者提示的 greedy 验证不可 bit 级复现 (同后端两次仅 6/469 字节一致), A/B 必须开 rollout.full_determinism=True。
 2. 缺少自动化测试覆盖: 没有单测或 CI 用例, LLMServerClient.generate 中“pop → 按后端覆盖 video_data”这条分支未来若被改动 (例如有人清理 kwargs 或调整调用顺序) 不会有任何测试报警; build_sglang_video_payload 的 None 语义与 payload 键集合也缺少回归保护。
 3. 多轮重复传特征: ToolAgentLoop 在每轮 _handle_generating_state 都携带初始 prompt 的 mm_processor_output, 与 prompt 已经扩展后的 token 序列配合时依赖 SGLang 每轮重新消费同一份特征, 作者已将其标为 caveat, 但缺少多轮视频的实际验证。
 4. 非 Qwen 视频家族静默降级: build_sglang_video_payload 在 processor 无 pixel_values_videos 时返回 None, LLMServerClient.generate 对 SGLang 无条件覆盖 video_data, 因此这类模型仍然是“静默丢视频”, 与 PR 力图消除的失败模式同类, 只是不会崩溃; 需要两侧同时注册新模态键才能真正支持。
 5. 内存与序列化开销: mm_inputs_out 与 payload 以引用方式持有 pixel_values_videos 等张量并挂在 AgentData / AgentLoopOutput 上, 多轮 rollout 与 teacher 调用会额外携带这些张量; max_frames=32 时量级可控, 但属于新增的常驻开销, 未见显式释放。
 6. vLLM 兼容性: 改动明确保持 vLLM 走原始帧分支, 作者用“差异样本 0/463 在首字符分叉”论证输入未变; 不过 mm_processor_output 会被 pop 并丢弃, 逻辑正确性依赖这一行 pop 不早于 vLLM 的 video_data 使用。- 影响: 用户侧: 所有使用 rollout.name=sglang 做视频 VLM RL 训练的用户直接受益, 视频问答从无效变为可用, vLLM 用户无感知; 采用 SGLang 作为蒸馏 teacher (OPD) 的视频任务同样被修复。系统侧: 在 tokenizer → agent loop → rollout client → SGLang server 之间新增了一条跨层数据契约 mm_processor_output, 四层都要保持签名兼容, 后续新增 VL builder 必须接受 mm_inputs_out 否则首调即 TypeError; 同时把“后端差异适配”集中到了

LLMServerClient.generate, 与既有的 priority 分派形成统一模式。团队侧: vLLM 与 SGLang 在视频任务上的对齐不再依赖各后端自行解码, 评测可比性恢复; 但非 Qwen 视频家族的支持仍需跟进, 且没有任何自动化测试守护这条链路。 - 风险标记: 核心 rollout 路径行为变更, 视频输入语义变化需重跑基线, 缺少自动化测试覆盖, 非 Qwen 视频家族静默降级, 多轮重复发送视频特征

关联脉络

- PR #2510 同主题 SGLang 视频输入支持 (body 以 DO NOT MERGE 开头, 2025-08-01 起停滞): PR body 明确对比: 该 PR 只改 sglang_rollout/sglang_rollout.py 的 batch 级 _batch_level_generate_sequences 路径, 转发原始 multi_modal_data["video"], 既未覆盖当前 main 上使用的 agent-loop generate 路径, 也未处理原始帧与 VideoDataInputItem 类型不匹配以及 SGLang 重新抽帧导致的偏差。
- PR #4513 搜索同类工作命中的另一个 SGLang 视频 PR: 作者按 "is:pr is:open sglang video" 搜索时一并命中, 用于确认没有重复工作; PR body 中将其与 #2510 并列作为差异说明的依据。
- PR #7643 [rollout] feat: continuous_token support Assistant Role in Merge and Encode Context: 同样改动 verl/utils/tokenizer/continuous_token.py 的 Continuous Token 渲染 / 合并路径; 本 PR 在 render_tokens_with_mm / build_initial_tokens 上新增 mm_inputs_out 参数, 需要与那里的 builder 签名和 VL mixin 结构保持兼容。
- PR #7858 [trainer] fix: pad teacher fields in minimal padding and support fsdp2 in forward topk loss: 同属 distillation teacher 数据链路: 本 PR 为 teacher 调用链新增 mm_processor_output 字段并贯通到 TeacherManager.compute_teacher_logprobs_single, 两者都在处理 teacher 侧字段的传递与对齐问题。