PR #6276 完整报告

verl-project/verl

[data, rollout] feat: add audio data support

合并时间：2026-05-12 23:09

原文链接：http://prhub.com.cn/verl-project/verl/pull/6276

执行摘要

- 一句话：新增音频数据管道支持
- 推荐动作：建议仔细审查键名不一致问题，并在后续 PR 中统一；值得关注 build_multimodal_processor_inputs 的设计，它为多模态输入提供了统一入口，可被未来模态复用。

功能与动机

从 #6118 拆分，为训练框架提供通用音频数据管道，避免与 Qwen3-Omni 等模型特定实现耦合，驱动后续多模态 RL 训练能力。

实现拆解

1. 在 verl/utils/dataset/rl_dataset.py 中添加 audio_key 配置和 _extract_audio_info/_process_multi_modal_info 方法，从数据集中解析音频并构建多模态处理器输入。
2. 在 verl/utils/tokenizer.py 中新增 get_processor_token_id 和 build_multimodal_processor_inputs，统一处理器调用逻辑并支持音频采样率注入。
3. 在 verl/experimental/agent_loop/agent_loop.py 中引入 mm_processor_kwargs 配置，并将 process_vision_info 泛化为 process_multi_modal_info，使音频载荷可沿 agent loop 传递。
4. 更新 verl/workers/rollout/schemas.py 以及 LLMServerClient、vLLM 和 TRT-LLM 服务器以在 generate 请求中传输 audio_data 和 mm_processor_kwargs；TRT-LLM 路径因不支持音频而显式抛出异常。
5. 在 verl/utils/model.py 中增加 _pad_last_dim_and_cat 函数，支持变长多模态输入的正确拼接；补充 extract_multi_modal_inputs 对 _VARLEN_MULTI_MODAL_KEYS 的处理。
6. 新增两个 CPU 测试文件，覆盖数据集音频解析和服务器契约。

关键文件：

- verl/experimental/agent_loop/agent_loop.py（模块 Agent 循环；类别 source；类型 core-logic；符号 _get_mm_processor_kwargs, process_multi_modal_info, _compute_position_ids）：核心变更文件，添加 mm_processor_kwargs 支持，将 process_vision_info 泛化为 process_multi_modal_info 以处理音频，并实现 _get_mm_processor_kwargs 推断采样率。

- verl/utils/dataset/rl_dataset.py（模块 数据集；类别 source；类型 core-logic；符号 _extract_audio_info, _process_multi_modal_info, process_multi_modal_info）：音频数据入口，添加 audio_key 配置、_extract_audio_info 和 _process_multi_modal_info 方法，使数据集能解析并传递音频载荷。
- verl/utils/tokenizer.py（模块 分词器；类别 source；类型 core-logic；符号 get_processor_token_id, _split_videos_and_metadata, build_multimodal_processor_inputs）：新增 get_processor_token_id 和 build_multimodal_processor_inputs，为所有多模态处理器（含音频）提供统一调用入口，支持采样率注入。
- verl/utils/model.py（模块 模型工具；类别 source；类型 data-contract；符号 _pad_last_dim_and_cat, _format_tensor_shapes）：新增 _pad_last_dim_and_cat 函数，支持变长多模态输入（如音频特征）的正确拼接，增强了多模态提取的鲁棒性。
- tests/utils/test_audio_input_support_on_cpu.py（模块 测试；类别 test；类型 test-coverage；符号 test_build_messages_replaces_audio_placeholder, test_build_multimodal_processor_inputs_includes_audio_sampling_rate, AudioProcessor, init）：新增 CPU 测试，覆盖 RLHFDataset 的音频构建、build_multimodal_processor_inputs 的采样率注入、以及 extract_multi_modal_inputs 的变长合并。
- tests/experimental/agent_loop/test_audio_server_contract_on_cpu.py（模块 测试；类别 test；类型 test-coverage；符号 _load_module_ast, test_async_server_manager_generate_accepts_audio_and_mm_kwargs, test_async_server_manager_generate_forwards_audio_and_mm_kwargs, test_fully_async_server_manager_generate_forwards_audio_and_mm_kwargs）：新增 CPU 测试，使用 AST 解析验证 LLMServerClient、fully_async_rollouter 和 vLLM 服务器接口包含 audio_data 和 mm_processor_kwargs 参数。

关键符号：_extract_audio_info, _process_multi_modal_info, build_multimodal_processor_inputs, get_processor_token_id, _pad_last_dim_and_cat

关键源码片段

verl/utils/tokenizer.py

新增 get_processor_token_id 和 build_multimodal_processor_inputs，为所有多模态处理器（含音频）提供统一调用入口，支持采样率注入。

```
def build_multimodal_processor_inputs(
    processor,
    *,
    text,
    images=None,
    videos=None,
    audio=None,
    mm_processor_kwargs=None,
    return_tensors: str = "pt",
):
```

```
    # 初始化处理器参数，优先使用 mm_processor_kwargs 中传递的值
    processor_kwargs = dict(mm_processor_kwargs or {})
    # 如果提供了音频且未指定采样率，从 processor 的 feature_extractor 中推断
    if audio is not None and "sampling_rate" not in processor_kwargs:
        sampling_rate = getattr(
            getattr(processor, "feature_extractor", None), "sampling_rate", None
        )
        if sampling_rate is not None:
            processor_kwargs["sampling_rate"] = int(sampling_rate)

    # 分离视频数据与元数据（如果视频是 (value, metadata) 元组列表）
    videos, video_metadata = _split_videos_and_metadata(videos)
    processor_kwargs.setdefault("return_tensors", return_tensors)

    if video_metadata is not None:
        processor_kwargs.setdefault("video_metadata", video_metadata)
        processor_kwargs.setdefault("do_sample_frames", False)

    # 构建最终处理器调用参数
    processor_inputs = {
        "text": text,
        "images": images,
        "videos": videos,
        **processor_kwargs,
    }
    if audio is not None:
        processor_inputs["audio"] = audio

    # 调用 processor，返回 processor-specific 格式的结果
    return processor(**processor_inputs)
```

verl/utils/model.py

新增 _pad_last_dim_and_cat 函数，支持变长多模态输入（如音频特征）的正确拼接，增强了多模态提取的鲁棒性。

```
def _pad_last_dim_and_cat(values: list[torch.Tensor], key: str) -> torch.Tensor:
    # 如果列表为空，直接报错
    if not values:
        raise ValueError(f"Cannot merge empty multi-modal input list for key {key!r}.")

    # 内部函数：格式化所有形状用于错误消息
    def _format_tensor_shapes(values: list[torch.Tensor]) -> str:
        return ", ".join(str(tuple(value.shape)) for value in values)

    rank = values[0].dim()
    if rank < 2:
        raise RuntimeError(
            f"Cannot pad multi-modal input {key!r} with rank {rank}; shapes: {_format_tensor_
            shapes(values)}"
```

```
        )

    # 检查除了 batch 维度 (0) 和时间维度 (-1) 外的中间维度是否匹配
    middle_shape = values[0].shape[1:-1]
    for value in values:
        if value.dim() != rank or value.shape[1:-1] != middle_shape:
            raise RuntimeError(
                f"Cannot pad multi-modal input {key!r}; expected matching rank and non-batch/non-time "
                f"dimensions, got shapes: {_format_tensor_shapes(values)}"
            )

    max_len = max(value.shape[-1] for value in values)
    # 如果所有值的时间维度已经相同，直接 cat
    if all(value.shape[-1] == max_len for value in values):
        return torch.cat(values, dim=0)

    # 否则填充到最大长度再 cat
    padded_values = []
    for value in values:
        if value.shape[-1] == max_len:
            padded_values.append(value)
            continue
        padded_value = value.new_zeros((*value.shape[:-1], max_len))
        padded_value[..., : value.shape[-1]] = value
        padded_values.append(padded_value)

    return torch.cat(padded_values, dim=0)
```

评论区精华

Review 中 gemini-code-assist[bot] 突出两个问题：

1) vLLM 的 TokensPrompt 构造中，若 mm_processor_kwargs 触发 TypeError，fallback 路径未删除该键，可能导致后续失误； 2) agent loop 中使用复数键（images/videos/audios）而 rollout 服务器使用单数键（image/video/audio），键名不一致可能导致音频数据丢失。PR 已合并，上述问题尚未得到解决。

- vLLM mm_processor_kwargs 兼容性风险 (correctness): PR 已合并，未看到作者回复或修复；风险仍然存在。
- Agent loop 与 rollout 服务器键名不一致 (design): PR 已合并，未统一；需要后续 PR 对齐。

风险与影响

- 风险： 1) 键名不一致风险：agent loop 与 rollout 服务器分别使用复数 / 单数键，导致音频数据可能无法正确传递。 2) vLLM 版本兼容性：mm_processor_kwargs 在旧版 vLLM 中不受支持，fallback 逻辑未移除该键，可能引发二次错误。 3) TRT-LLM 不支持音频：显式异常会中断训练，影响混合后端部署。 4) 变长多模态输入处理：新引入的

_pad_last_dim_and_cat 对形状校验严格，极端情况下可能触发 RuntimeError。

- 影响：影响范围：修改涉及 dataset、agent loop、rollout worker 和测试模块，共 21 个文件。影响程度：中，新功能对文本 / 图像仅路径保持向后兼容，仅音频处理新增。但由于键名不一致风险，音频功能可能存在数据传递缺陷。
- 风险标记：键名不一致，vLLM 兼容性，TRT-LLM 不支持，变长输入处理

关联脉络

- PR #6118 Original mixed PR being split: 本 PR 从 #6118 拆分，提取通用音频数据管道部分，#6118 是原始混合 PR。
- PR #6277 Qwen3-Omni thinker follow-up: 依赖本 PR 的后续 PR #6277 将添加 Qwen3-Omni thinker 模型特定支持。
- PR #3297 Older draft Omni PR: #3297 是更早的 WIP/Draft Omni 模型特定 PR，与本 PR 有重叠但方法不同。