

PR #48291 完整报告

vllm-project/vllm

Add Cosmos3 Edge Reasoner model

合并时间: 2026-07-14 16:14

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/48291>

执行摘要

- 一句话: 新增 Cosmos3 Edge Reasoner 多模态模型
- 推荐动作: 建议阅读该 PR 以了解如何将复杂多模态模型集成到 vLLM 中, 特别是:
 1. 如何通过继承现有模型 (Qwen3-VL、Nemotron-H、SigLIP2) 快速实现新模型。
 2. 配置类中子配置的组合与字段同步技巧。
 3. Checkpoint 加载方式从 `secondary_weights` 到 `allow_patterns_overrides` 的迁移原因和具体做法。对于计划集成类似多模态模型的开发者, 本 PR 提供了很好的参考范例。

功能与动机

根据 PR 描述, 该 PR 旨在将尚未正式发布的 Cosmos3 Edge Reasoner 模型集成到 vLLM 中。Cosmos3 是一个混合 Transformer 模型, 由一个 Reasoner 和一个 Generator Tower 组成, 支持多模态输入。PR 引入了自定义模型层、处理器和 checkpoint 映射, 使用户可以加载并使用该模型进行推理。

实现拆解

1. 配置层: 在 `vllm/transformers_utils/configs/cosmos3_edge.py` 中定义了 `Cosmos3EdgeTextConfig`、`Cosmos3EdgeProjectorConfig`、`Cosmos3EdgeVisionConfig` 和 `Cosmos3EdgeConfig`。文本配置继承自 `NemotronHConfig`, 将 `num_hidden_layers` 翻倍用于混合注意力模式 (*-), 并在 `to_dict` 中还原; 同时通过 `_normalize_mrope_parameters` 统一 M-RoPE 的参数字段。视觉配置继承 `Siglip2VisionConfig`, 投影配置定义了空间合并大小和中间维度。顶层 `Cosmos3EdgeConfig` 将三个子配置组合, 并从投影配置同步字段到视觉配置 (如 `spatial_merge_size`、`out_hidden_size`), 以兼容 Qwen3-VL 的处理流程。
2. 处理器: 在 `vllm/transformers_utils/processors/cosmos3_edge.py` 中实现了 `Cosmos3EdgeImageProcessor`, 继承自 `Siglip2ImageProcessor`。其主要工作是在 `_resize_image` 方法中: 确保图像的宽高能被 `merge_size * patch_size` 整除, 控制像素总数在 `[min_pixels, max_pixels]` 区间内, 并尽可能保持宽高比。此外还提供了 `round_by_factor`、`ceil_by_factor`、`floor_by_factor` 辅助函数。`Cosmos3EdgeImageProcessor` 与 `Qwen3VLProcessor` 组合, 构成完整的 `Cosmos3EdgeProcessor`。

3. 模型主体：在 `vllm/model_executor/models/cosmos3_edge.py` 中新增了核心模型类。
`Cosmos3EdgeVisionEncoder` 继承自 `Siglip2VisionTransformer`，其 `forward` 方法将 (T, H, W) 的 `grid_thw` 展开为帧独立的 HW 序列，通过 `torch.repeat_interleave` 和载入 `cu_seqlens` 后调用父类前向。`patch_merging_by_param` 函数将每个 `merge_size` 的 `patch` 块沿通道合并。`Cosmos3EdgeProcessingInfo` 和 `Cosmos3EdgeForConditionalGeneration` 继承自 `Qwen3VLProcessingInfo` 和 `Qwen3VLForConditionalGeneration`，复用了 `Qwen3-VL` 的 dummy 输入构建、多模态处理器注册等逻辑。视觉模型在初始化时会创建 `Cosmos3EdgeVisionEncoder`、`Cosmos3EdgePatchMerger` 以及 `M-RoPE` 投影层。
4. 基站模型修改：`vllm/model_executor/models/nemotron_h.py` 的 `NemotronHModel` 增加了 `decoder_layer_types` 参数，允许从外部传递解码器层类型映射，而非固定使用 `ALL_DECODER_LAYER_TYPES`。`Cosmos3EdgeTextConfig` 通过 `hybrid_override_pattern` 生成全注意力模式，并传递有效的层类型集合。
5. `Cosmos3` 加载修复：`vllm/model_executor/models/cosmos3.py` 中移除了 `secondary_weights` 的使用，改为类级别的 `allow_patterns_overrides = ['[tv]*er/*.safetensors']`，以适配 `Diffusers` 格式 checkpoints 的统一加载。这一修改也惠及原 `Cosmos3` 模型。
6. 注册与入口：在 `vllm/transformers_utils/configs/__init__.py`、`vllm/transformers_utils/processors/__init__.py`、`vllm/transformers_utils/config.py` 以及 `vllm/model_executor/models/registry.py` 中添加了 `Cosmos3Edge` 的模型和处理器注册项。
7. 测试：`tests/models/multimodal/processing/test_cosmos3_edge.py` 测试了图像和视频的预处理流程，验证 `grid_thw`、像素值、placeholder 数量和令牌 ID 的正确性。
`tests/models/multimodal/test_mapping.py` 新增了 `test_cosmos3_edge_checkpoint_weights_mapper` 测试权重映射逻辑。
`tests/models/registry.py` 也更新了注册表测试。

关键文件：

- `vllm/model_executor/models/cosmos3_edge.py`（模块 模型主体；类别 `source`；类型 `data-contract`；符号 `Cosmos3EdgeVisionEncoder`, `dtype`, `forward`, `patch_merging_by_param`）：核心模型文件，定义了视觉编码器、patch 合并、处理信息以及主生成类，是 PR 的核心变更。
- `vllm/transformers_utils/processors/cosmos3_edge.py`（模块 图像处理器；类别 `source`；类型 `dependency-wiring`；符号 `round_by_factor`, `ceil_by_factor`, `floor_by_factor`, `Cosmos3EdgeImagesKwargs`）：图像处理器，基于 `SigLIP2` 处理器实现了适应 `Cosmos3 Edge` 的缩放逻辑，是模型预处理的关键。
- `vllm/transformers_utils/configs/cosmos3_edge.py`（模块 配置层；类别 `source`；类型 `core-logic`；符号 `_normalize_mrope_parameters`, `Cosmos3EdgeTextConfig`, `init`, `to_dict`）：配置类层次结构，定义了文本、投影和视觉子配置，并完成字段同步，是模型初始化的基础。
- `vllm/model_executor/models/nemotron_h.py`（模块 文本模型；类别 `source`；类型 `data-contract`；符号 `init`）：为 `NemotronHModel` 增加 `decoder_layer_types` 参数，使得

Cosmos3 Edge 可以传递自定义层类型集合。

- `vllm/model_executor/models/cosmos3.py` (模块 模型加载; 类别 source; 类型 data-contract; 符号 init) : 修复 Cosmos3 模型权重加载, 移除 `secondary_weights`, 改为 `allow_patterns_overrides`, 以适配 Diffusers 格式。
- `tests/models/multimodal/processing/test_cosmos3_edge.py` (模块 处理测试; 类别 test; 类型 test-coverage; 符号 processor, `_assert_image_outputs`, `_assert_video_outputs`, `test_process_images`) : 图像 / 视频处理测试, 验证预处理流程的正确性。
- `tests/models/multimodal/test_mapping.py` (模块 映射测试; 类别 test; 类型 test-coverage; 符号 `test_cosmos3_edge_checkpoint_weights_mapper`) : 权重映射测试, 验证 checkpoint 键映射的正确性。
- `vllm/model_executor/models/registry.py` (模块 模型注册; 类别 source; 类型 data-contract) : 注册 `Cosmos3EdgeForConditionalGeneration` 到模型注册表。
- `vllm/transformers_utils/configs/__init__.py` (模块 配置入口; 类别 source; 类型 core-logic) : 导出 `Cosmos3EdgeConfig` 供其他模块引用。
- `vllm/transformers_utils/processors/__init__.py` (模块 处理器入口; 类别 source; 类型 core-logic) : 注册 `Cosmos3EdgeProcessor` 到处理器映射。
- `vllm/transformers_utils/config.py` (模块 配置映射; 类别 source; 类型 core-logic) : 将 `Cosmos3Edge` 添加到信任的远程代码列表。
- `tests/models/registry.py` (模块 注册测试; 类别 test; 类型 test-coverage) : 更新注册表测试以包含 `Cosmos3 Edge`。

关键符号: `Cosmos3EdgeVisionEncoder.forward`, `patch_merging_by_param`, `Cosmos3EdgePatchMerger.init`, `Cosmos3EdgeImageProcessor._resize_image`, `Cosmos3EdgeImageProcessor._preprocess`, `_normalize_mrope_parameters`, `Cosmos3EdgeTextConfig.init`, `Cosmos3EdgeTextConfig.to_dict`, `Cosmos3EdgeConfig.init`, `NemotronHModel.init`

关键源码片段

`vllm/model_executor/models/cosmos3_edge.py`

核心模型文件, 定义了视觉编码器、patch 合并、处理信息以及主生成类, 是 PR 的核心变更。

```
class Cosmos3EdgeVisionEncoder(Siglip2VisionTransformer):
    """Adapts Cosmos (T, H, W) metadata to vLLM packed SigLIP2."""

    @property
    def dtype(self) -> torch.dtype:
        # 从 patch_embedding 权重获取 dtype
        return self.embeddings.patch_embedding.weight.dtype

    def forward(
        self,
        pixel_values: torch.Tensor,
        grid_thw: torch.Tensor,
```

```

) -> torch.Tensor:
    # SigLIP2 对每一帧独立 attention, 所以将 (T, H, W) 展开为 T 个 HW 序列
    grid_thw_cpu = grid_thw.to(device='cpu')
    spatial_shapes = torch.repeat_interleave(
        grid_thw_cpu[:, 1:],
        grid_thw_cpu[:, 0],
        dim=0,
    )
    lengths_cpu = spatial_shapes.prod(dim=-1).to(torch.int32)
    lengths = lengths_cpu.to(device=pixel_values.device, non_blocking=True)
    cu_seqlens = torch.zeros(lengths.numel() + 1, dtype=torch.int32, device=pixel_values.device)
    cu_seqlens[1:] = lengths.cumsum(dim=0)
    max_seqlen = lengths_cpu.max().reshape(1)
    return super().forward(
        pixel_values_packed=pixel_values,
        spatial_shapes=spatial_shapes,
        cu_seqlens=cu_seqlens,
        max_seqlen=max_seqlen,
    )[0]

```

vllm/transformers_utils/configs/cosmos3_edge.py

配置类层次结构，定义了文本、投影和视觉子配置，并完成字段同步，是模型初始化的基础。

```

class Cosmos3EdgeTextConfig(NemotronHConfig):
    """Cosmos3 Edge 文本配置，基于 Nemotron-H 层。
    将客户端的 num_hidden_layers 翻倍传给父类以构造全注意力模式。"""
    model_type = 'cosmos3_edge_text'
    has_no_defaults_at_init = True
    ignore_keys_at_rope_validation = {'mrope_section', 'mrope_interleaved'}

    def __init__(self, num_hidden_layers=28, hidden_act='relu2', rms_norm_eps=1e-5, **kwargs):
        super().__init__(
            num_hidden_layers=2 * num_hidden_layers,
            hybrid_override_pattern='*- ' * num_hidden_layers,
            mlp_hidden_act=hidden_act,
            layer_norm_epsilon=rms_norm_eps,
            **kwargs,
        )

    def to_dict(self):
        config_dict = super().to_dict()
        config_dict['num_hidden_layers'] = self.num_hidden_layers // 2
        config_dict['hidden_act'] = config_dict.pop('mlp_hidden_act')
        config_dict['rms_norm_eps'] = config_dict.pop('layer_norm_epsilon')
        config_dict.pop('hybrid_override_pattern', None)
        return config_dict

```

评论区精华

Review 中 Isotr0py 提出了多项改进建议并得到作者积极响应：

- 正则映射可读性：Isotr0py 建议将复杂的 regex 映射改为前缀 / 子串映射，作者回复已改进可读性。
- 冗余 patch：Isotr0py 指出部分 patch（如 `_normalize_mrope_parameters` 的调用）在 `cosmos3_edge.py` 中已存在，但实际已在 `configs/cosmos3_edge.py` 中完成，作者确认并移除。
- 未使用导入：Isotr0py 指出一些导入未使用，作者已删除。
- 权重加载方式：Isotr0py 对 `secondary_weights` 的移除提出疑问，作者解释由于最新权重加载机制的变化，需改用类级别 `allow_patterns_overrides` 才能正确加载视觉编码器权重，并同步修复了 Cosmos3 的加载问题。
- 处理器注册顺序：Isotr0py 要求按字母顺序插入，作者照做。
- 正则映射可读性 (design)：作者回复已改进可读性。
- 冗余 patch 移除 (correctness)：作者移除冗余补丁。
- 权重加载方式变更 (design)：接受 `allow_patterns_overrides` 方案。
- 未使用导入清理 (cleanup)：作者删除未使用的导入。
- 处理器注册顺序 (style)：作者照做。

风险与影响

- 风险：
 - 新模型稳定性：模型尚未正式发布，checkpoint 可能在后续变更，需持续跟踪官方更新。
 - 依赖内部接口：大量复用 Qwen3-VL 和 Nemotron-H 的内部类（如 `Qwen3VLForConditionalGeneration`、`NemotronHModel`），若这些基类发生接口变更可能影响到 Cosmos3 Edge 模型。
 - 图像缩放逻辑：`_resize_image` 中的像素裁剪算法涉及像素上限 / 下限的循环调整，可能在极端宽高比或分辨率下产生非预期行为，建议增加更多边界测试。
 - Checkpoint 加载：`allow_patterns_overrides` 模式变更可能影响原有 Cosmos3 模型的加载，需确保兼容性。
 - 测试覆盖有限：目前仅处理测试和映射测试，缺少端到端推理测试和性能 benchmark。
- 影响：
 - 用户影响：用户现在可以在 vLLM 中使用 `nvidia/Cosmos3-Edge` 模型，进行包含图像和视频的多模态推理。需注意该模型尚未公开发布，仅早期访问。
 - 系统影响：新增约 1500 行代码，扩展了模型库的规模，但未修改核心调度或执行路径。对现有模型无负面影响。
 - 团队影响：维护者需关注 NVIDIA 后续发布的官方配置和权重，可能需要对映射逻辑进行调整。代码中使用了 `# type: ignore` 和 `typing.Unpack` 等类型提示，可能对静态类型检查产生干扰。
 - 风险标记：新模型尚未发布，依赖 Qwen3-VL 和 Nemotron-H 内部接口，图像缩放算法边界情况，权重加载模式变更影响原 Cosmos3 模型，缺少端到端推理测试

关联脉络

- 暂无明显关联 PR