PR #44429 完整报告

vllm-project/vllm

[Model] Add Gemma4 Unified (encoder-free) support

合并时间：2026-06-04 03:01

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/44429

执行摘要

- 一句话：新增 Gemma4 Unified 编码器无关多模态模型
- 推荐动作：值得精读，特别关注子类化父类避免分支的设计模式，以及量化条件处理和嵌入数据类型的讨论。后续需要跟进 PR#44340 的修复并验证音频回归。

功能与动机

支持 Gemma4 12B Unified 模型家族，其编码器无关设计无视觉和音频编码器，原始像素补丁和音频波形帧直接投影到语言模型空间，需要新的模型类和适配。

实现拆解

1. 新增模型文件vllm/model_executor/models/gemma4_unified.py：定义 Gemma4UnifiedVisionEmbedder（密集投影 + 分解位置嵌入）、Gemma4UnifiedProcessingInfo（软 token 配置）和 Gemma4UnifiedForConditionalGeneration（子类化父类并重构建模流水线）。
2. 修改基类gemma4_mm.py：在 Gemma4ProcessingInfo 中添加 _compute_audio_num_tokens 静态方法（精确模拟音频编码器序列长度计算）；调整量化塔条件（仅 BNB/torchao 量化非 64 维层）；修复 eoa_token_id 回退。
3. 修改 MTPgemma4_mtp.py：为 Gemma4MTPDecoderLayer 添加 attention_k_eq_v 和 num_global_key_value_heads 支持；在 drafter compute_logits 中应用 suppress_tokens。
4. 修复数据类型gemma4.py：将嵌入缩放器 normalizer 的数据类型改为 embed_tokens.weight.dtype，避免量化时属性缺失错误。
5. 注册与配置：更新 registry.py、config.py、model_arch_config_convertor.py 注册 Gemma4UnifiedForConditionalGeneration。
6. 推测解码：更新 speculative.py 和 llm_base_proposer.py 支持 Unified 模型作为助理。
7. GPU 运行器gpu_model_runner.py：在构建注意力组时跳过音频模态的双向注意力范围。
8. 测试：新增 tests/models/multimodal/processing/test_gemma4_unified.py 覆盖软 token 上限、批处理、视频帧等场景；更新 tests/models/registry.py。
9. 文档：更新 supported_models.md 和 mtp.md。

关键文件：

- vllm/model_executor/models/gemma4_unified.py（模块 模型定义；类别 source；类型 core-logic；符号 Gemma4UnifiedVisionEmbedder, init, _factorized_posemb, forward）：核心新增文件，实现 Gemma4UnifiedVisionEmbedder 和 Gemma4UnifiedForConditionalGeneration。
- tests/models/multimodal/processing/test_gemma4_unified.py（模块 测试；类别 test；类型 test-coverage；符号 test_gemma4_unified_image_schema_accepts_variable_patch_counts, test_gemma4_unified_image_batching_keeps_variable_patch_counts_unstacked, test_compute_num_soft_tokens_does_not_exceed_max_soft_tokens, test_get_mm_max_tokens_per_item_respects_configured_max_soft_tokens）：新增测试文件，覆盖软 token 上限、批处理、视频帧等关键处理逻辑。
- vllm/model_executor/models/gemma4_mm.py（模块 模型定义；类别 source；类型 data-contract；符号 _compute_audio_num_tokens）：关键修改，添加 _compute_audio_num_tokens 方法，调整量化塔条件，修复 eoa_token_id。
- vllm/model_executor/models/gemma4_mtp.py（模块 模型定义；类别 source；类型 data-contract）：添加 attention_k_eq_v 和 suppress_tokens 支持，增强推测解码兼容性。
- vllm/model_executor/models/gemma4.py（模块 模型定义；类别 source；类型 bugfix）：修复嵌入缩放器数据类型，避免量化时崩溃。
- vllm/model_executor/models/registry.py（模块 注册；类别 source；类型 configuration）：注册新模型到全局模型映射。
- tests/models/registry.py（模块 测试；类别 test；类型 test-coverage）：添加 Unified 模型到注册测试用例。

关键符号：Gemma4UnifiedVisionEmbedder.init, Gemma4UnifiedVisionEmbedder.forward, Gemma4UnifiedVisionEmbedder._factorized_posemb, Gemma4UnifiedProcessingInfo.get_mm_max_tokens_per_item, Gemma4ProcessingInfo._compute_audio_num_tokens, Gemma4MTPDecoderLayer.init

关键源码片段

vllm/model_executor/models/gemma4_unified.py

核心新增文件，实现 Gemma4UnifiedVisionEmbedder 和 Gemma4UnifiedForConditionalGeneration。

```
class Gemma4UnifiedVisionEmbedder(nn.Module):
    """编码器无关的视觉嵌入器，用于 Gemma4 Unified 变体。

    通过密集投影和分解式二维位置嵌入将原始像素补丁投影到语言模型空间。
    替代基于塔的 Gemma4 变体中使用的 SigLIP 视觉塔。

    处理流程: raw patches → LN₁ → Dense → LN₂ → +factorized_posemb → LN₃。
    """

    def __init__(self, config, quant_config=None):
        super().__init__()
        # 补丁维度 = (patch_size ** 2) * 3（RGB 通道）
```

```
        patch_dim = config.model_patch_size ** 2 * 3
        mm_embed_dim = config.mm_embed_dim

        # 第一个 LayerNorm，对 raw patches 进行归一化
        self.patch_ln1 = nn.LayerNorm(patch_dim)
        # 密集投影层，将补丁维度映射到多模态嵌入维度（支持张量并行）
        self.patch_dense = ColumnParallelLinear(
            patch_dim,
            mm_embed_dim,
            bias=True,
            quant_config=quant_config,
            gather_output=True,
        )
        self.patch_ln2 = nn.LayerNorm(mm_embed_dim)

        # 因子分解位置嵌入参数：(posemb_size, 2, mm_embed_dim)
        self.pos_embedding = nn.Parameter(
            torch.zeros(config.mm_posemb_size, 2, mm_embed_dim)
        )
        self.pos_norm = nn.LayerNorm(mm_embed_dim)

    def _factorized_posemb(self, positions_xy: torch.Tensor) -> torch.Tensor:
        """根据 x, y 坐标查询因子分解位置嵌入并求和。

        对每个坐标轴独立嵌入后相加，无效位置（`-1`，表示 padding）会被掩码掉。
        """
        clamped_pos = positions_xy.clamp(min=0).long() # 裁剪负值索引
        valid_mask = positions_xy != -1 # 有效位置掩码

        pos_embs = torch.zeros(
            *positions_xy.shape[:-1],
            self.pos_embedding.shape[-1],
            device=positions_xy.device,
            dtype=self.pos_embedding.dtype,
        )
        for i in range(2):
            axis_pe = self.pos_embedding[:, i, :][clamped_pos[..., i]]
            mask = valid_mask[..., i].unsqueeze(-1).to(axis_pe.dtype)
            pos_embs = pos_embs + (axis_pe * mask) # 求和 + 掩码
        return pos_embs

    def forward(
        self,
        pixel_values: torch.Tensor,
        pixel_position_ids: torch.Tensor,
    ) -> torch.Tensor:
        """前向传递：补丁归一化 -> 密集投影 -> LN -> 加位置嵌入 -> LN。
        """
        hidden_states = self.patch_ln1(pixel_values.to(self.pos_embedding.dtype))
```

```
        hidden_states, _ = self.patch_dense(hidden_states)
        hidden_states = self.patch_ln2(hidden_states)

        pos_embs = self._factorized_posemb(pixel_position_ids)
        hidden_states = hidden_states + pos_embs
        hidden_states = self.pos_norm(hidden_states)
        return hidden_states
```

vllm/model_executor/models/gemma4_mm.py

关键修改，添加 _compute_audio_num_tokens 方法，调整量化塔条件，修复 eoa_token_id。

```
@staticmethod
def _compute_audio_num_tokens(
    num_samples: int, sampling_rate: int, audio_seq_length: int
) -> int:
    """复制音频编码器的序列长度算术运算。

    模拟: mel 帧提取（Gemma4AudioFeatureExtractor 中的 _unfold）
    随后两个二维卷积下采样层（kernel=3, stride=2, 半因果 padding top=1, bottom=1），
    最终上限为 audio_seq_length。
    """
    # 帧长度：20ms 窗口
    frame_length = int(round(sampling_rate * 20.0 / 1000.0))
    # 帧移：10ms
    hop_length = int(round(sampling_rate * 10.0 / 1000.0))
    # unfold 需要 frame_length + 1 的帧大小
    frame_size_for_unfold = frame_length + 1
    # 左填充用于半因果卷积
    pad_left = frame_length // 2
    padded_samples = num_samples + pad_left
    # 计算 mel 帧数量
    num_mel_frames = (padded_samples - frame_size_for_unfold) // hop_length + 1
    if num_mel_frames <= 0:
        return 0
    t = num_mel_frames
    # 模拟两个二维卷积下采样：(t + 2 - 3) // 2 + 1
    for _ in range(2):
        t = (t + 2 - 3) // 2 + 1
    return min(t, audio_seq_length)
```

评论区精华

- 量化塔维度特殊处理(mgoin)：建议应由量化方法支持填充而非在此特殊区分，否则可能导致权重加载错误。未直接解决便合并。
- 嵌入权重数据类型(mgoin)：embed_tokens.weight 量化时可能不存在，同 PR#44340 问题。合并时未解决。
- 音频跳过可能回归(mgoin)：询问是否影响原始 Gemma4 音频行为，建议测试 E2B。作者未回应。

- 注册兼容性(mgoin)：要求设置 is_available_online=False，作者已改为 min_transformers_version。
- 量化塔维度条件特殊处理 (design): 作者未直接回复，PR 合并，作为遗留问题。
- 嵌入权重数据类型来源 (correctness): 作者未回复，PR 合并，可能需要后续修复。
- 音频模态跳过双向注意力可能影响回归 (correctness): 作者未回复，PR 合并，可能存在回归风险。
- 注册测试设置 is_available_online (testing): 作者改为设置 min_transformers_version。

风险与影响

- 风险：
  1. 量化条件特殊处理（gemma4_mm.py）：仅 BNB/torchao 量化塔层，其他量化方法下可能导致权重加载错误或精度损失。
  2. 嵌入权重数据类型（gemma4.py）：使用 embed_tokens.weight.dtype 在量化时可能无此属性，导致运行时错误。
  3. 音频注意力跳过（gpu_model_runner.py）：跳过音频的双向注意力范围可能影响原始 Gemma4 音频模型行为，缺乏回归测试。
  4. transformers 版本依赖（tests/models/registry.py）：依赖 5.8.0+，发布前需更新。 - 影响：用户：可使用 Gemma4 12B Unified 模型处理图像、视频、音频输入（编码器无关）。系统：增加约 466 行核心模型代码，注册和推测解码链接增强。团队：维护成本增加，但子类化模式便于后续扩展。兼容性：不影响现有塔式变体，但量化路径更改可能影响其他模型加载。
- 风险标记：量化塔条件特殊处理，嵌入权重数据类型风险，音频跳过可能回归，transformers 版本依赖

关联脉络

- 暂无明显关联 PR