PR #42662 完整报告

vllm-project/vllm

[LoRA][Gemma4] Support vision tower LoRA

合并时间：2026-08-13 22:07

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42662

执行摘要

- 一句话：Gemma4 多模态 LoRA 运行时管道与 token 计数接口
- 推荐动作：值得精读，重点看三处设计：一是 get_mm_lora_token_counts 如何用 (tower_tokens, connector_tokens | None) 统一不同模态的计数差异；二是 models/gemma4_mm.py 中图像 padding 与 LoRA mapping 对齐的取舍（禁用 bucketing 换正确性）；三是 model_manager.py 按模态最大预算 size Punica wrapper 的做法。建议同时关注 taroshi 反馈的 Gemma4ClippableLinear 无法包装问题是否在后续 PR 修复，以及 anshulkulhari7 指出的 stub 返回 None 的健壮性缺口。

功能与动机

issue #40693 中用户使用 unsloth 微调 gemma-4-E4B-it 后启动 vLLM 0.19.1 报错 ValueError: Gemma4ForConditionalGeneration does not support LoRA yet.。PR body 说明在 #43798 之后 Gemma4-MM 视觉线性层已通过 Transformers 后端路径转换，因此本 PR 不再重实现 vision tower，而是补齐运行时 LoRA 映射与 token 计数：为多模态模型引入可按 modality 分别上报 tower 与 connector token 数的接口，并用各模态最大预算来 size LoRA 包装器。

实现拆解

实现按 5 步拆解：

1. 接口层新增统一计数入口：vllm/model_executor/models/interfaces.py 在 SupportsMultiModal 上新增 get_mm_lora_token_counts(modality, mm_kwargs, num_mm_embeds) 默认实现，内部委托给旧的 get_num_mm_encoder_tokens / get_num_mm_connector_tokens，并把非 int 的 connector 计数归一为 None。这样未覆盖多模态差异的模型无需改动即可兼容新管道。
2. Gemma4-MM 按模态实现计数与编码调整：vllm/model_executor/models/gemma4_mm.py 新增 get_mm_lora_token_counts 重载——image 用 vision_config.default_output_length（或 mm_processor_kwargs 中的 max_soft_tokens）乘以 pooling_kernel_size**2 得到 tower token 数；video 用 _VIDEO_MAX_SOFT_TOKENS，必要时从 mm_kwargs.pixel_values_videos 形状推导；audio 用 input_features_padded 按 batch_size * ceil(features/4) 估算。同时 _process_image_input 在 enable_tower_connector_lora 开启时禁用分辨率 bucketing，把所有图像 padding 到最大 patch 数（超限抛 ValueError），并跳过内存感知分块，保证

一次 encoder 调用与 tower LoRA mapping 对齐。

3. LoRA 管理器按模态最大预算分配：vllm/lora/model_manager.py 的 _maybe_init_mm 遍历 mm_budget.mm_max_toks_per_item，对每个 modality 调用 get_mm_lora_token_counts(modality, mm_kwargs=None, num_mm_embeds=mm_budget.get_encoder_budget())，tower 取最大值、connector 取非 None 最大值来创建 Punica wrapper，替代原先单一 get_num_mm_encoder_tokens 调用。
4. V1 执行器与 V2 runner 路径同步改造：vllm/v1/worker/gpu_model_runner.py 的 _execute_mm_encoder 与 vllm/v1/worker/gpu/mm/lora.py 的 set_active_mm_loras 都改为一次调用返回 (tower_tokens, connector_tokens)，并用 all(count is not None ...) 判断是否设置 connector mapping，避免继承 stub 的模型误入 connector 分支。
5. 测试、CI 与文档配套：新增 tests/lora/test_gemma4_tp.py 端到端测试（单卡、TP2、TP4，使用真实 vision LoRA adapter EpochEcho/gemma4-e2b-it-lora-pokemon）；tests/lora/conftest.py 增加 gemma4_vision_lora_files fixture；.buildkite/test_areas/lora.yaml 把该测试加入 CI；docs/models/supported_models.md 将 Gemma4 行的 LoRA 支持标记为可用；tests/v1/worker/test_gpu_model_runner.py 同步更新 mock 以匹配新 API。

关键文件：

- vllm/model_executor/models/gemma4_mm.py（模块 模型层；类别 source；类型 data-contract；符号 get_mm_lora_token_counts, _process_image_input）：实现核心：新增 get_mm_lora_token_counts 按 image/video/audio 模态计算 tower/connector token 数；_process_image_input 在 MM LoRA 开启时统一 padding 并禁用 bucketing，保证 encoder 调用与 LoRA mapping 对齐。
- tests/lora/test_gemma4_tp.py（模块 测试；类别 test；类型 test-coverage；符号 generate_and_test, test_gemma4_lora, test_gemma4_lora_tp2, test_gemma4_lora_tp4）：新增端到端 LoRA 测试，覆盖单卡、TP2、TP4 三种配置，使用真实 Gemma4 vision LoRA adapter 验证文本输出前缀匹配，是功能正确性的关键验证。
- vllm/model_executor/models/interfaces.py（模块 接口层；类别 source；类型 data-contract；符号 get_mm_lora_token_counts）：在 SupportsMultiModal 上新增 get_mm_lora_token_counts 默认实现，成为所有多模态模型接入 tower/connector LoRA 的统一契约。
- vllm/lora/model_manager.py（模块 LoRA 管理；类别 source；类型 data-contract；符号 _maybe_init_mm）：LoRA 管理器初始化改为按模态取 tower/connector token 预算最大值来创建 Punica wrapper，是运行时正确性的核心配置逻辑。
- vllm/v1/worker/gpu_model_runner.py（模块 执行器；类别 source；类型 data-contract；符号 _execute_mm_encoder）：V1 执行器 _execute_mm_encoder 改用 get_mm_lora_token_counts，并借助 connector_token_counts 全非 None 判断是否设置 connector mapping，避免 stub 误入。
- vllm/v1/worker/gpu/mm/lora.py（模块 编码器路径；类别 source；类型 core-logic；符号 set_active_mm_loras）：V2 runner 路径 set_active_mm_loras 同步切换到新 API，并修复了 connector 守卫的运算符优先级问题，是 V2 路径正确性的关键。

- tests/v1/worker/test_gpu_model_runner.py（模块 测试；类别 test；类型 test-coverage；符号 test_set_active_mm_loras_builds_tower_and_connector_mappings）：更新 mock 测试以匹配 get_mm_lora_token_counts 新签名，确保执行器逻辑的单元测试同步。
- tests/lora/conftest.py（模块 测试；类别 test；类型 test-coverage；符号 gemma4_vision_lora_files）：新增 gemma4_vision_lora_files session 级 fixture，下载真实 Gemma4 vision LoRA adapter 供端到端测试使用。
- .buildkite/test_areas/lora.yaml（模块 CI 配置；类别 config；类型 configuration）：将 test_gemma4_tp.py 加入 LoRA CI 测试区域，但该测试文件头注释声明不触发 CI，两者存在不一致。
- docs/models/supported_models.md（模块 文档；类别 docs；类型 documentation）：更新 Gemma4 在 supported_models 中的 LoRA 支持标记为用户可见的公开承诺。

关键符号：get_mm_lora_token_counts, get_num_mm_encoder_tokens, get_num_mm_connector_tokens, _maybe_init_mm, _execute_mm_encoder, set_active_mm_loras, _process_image_input

关键源码片段

vllm/model_executor/models/gemma4_mm.py

实现核心：新增 get_mm_lora_token_counts 按 image/video/audio 模态计算 tower/connector token 数；_process_image_input 在 MM LoRA 开启时统一 padding 并禁用 bucketing，保证 encoder 调用与 LoRA mapping 对齐。

```
    def get_mm_lora_token_counts(
        self,
        *,
        modality: str,
        mm_kwargs: "MultiModalKwargsItem | None",
        num_mm_embeds: int,
    ) -> tuple[int, int | None]:
        # 返回 (tower_tokens, connector_tokens)，供 MM LoRA 构建 adapter
        # mapping。Gemma4 的图像 / 视频共享 vision tower，但 token 预算不同：
        # 图像按 default_output_length（或用户覆盖的 max_soft_tokens）计算，
        # 视频按 _VIDEO_MAX_SOFT_TOKENS 计算；音频则按打包后的特征帧估算。
        if modality in ("image", "video"):
            vision_config = self.config.vision_config
            pooling_k2 = vision_config.pooling_kernel_size**2

            if modality == "image":
                pixel_values_key = "pixel_values"
                max_soft_tokens = vision_config.default_output_length
                mm_processor_kwargs = getattr(
                    getattr(self, "multimodal_config", None),
                    "mm_processor_kwargs",
                    None,
                )
                # 用户可通过 mm_processor_kwargs 覆盖 soft token 数量，
```

```
            # 但只接受白名单 _SUPPORTED_SOFT_TOKENS 内的取值。
            if isinstance(mm_processor_kwargs, Mapping):
                val, _ = _get_max_soft_tokens(mm_processor_kwargs)
                if isinstance(val, int) and val in _SUPPORTED_SOFT_TOKENS:
                    max_soft_tokens = val
        else:
            pixel_values_key = "pixel_values_videos"
            max_soft_tokens = _VIDEO_MAX_SOFT_TOKENS

        # image 的 tower token 数是确定的；video 需要从 mm_kwargs 的
        # 张量形状读实际 patch 数，否则用最小 soft token 估算。
        tower_tokens = max_soft_tokens * pooling_k2 if modality == "image" else None
        connector_tokens = num_mm_embeds
        if tower_tokens is None and mm_kwargs is not None:
            field = mm_kwargs.get(pixel_values_key)
            if field is not None:
                data = field.data
                if isinstance(data, torch.Tensor) and data.ndim >= 2:
                    tower_tokens = int(math.prod(data.shape[:-1]))

        if tower_tokens is None:
            min_soft_tokens = min(_SUPPORTED_SOFT_TOKENS)
            tower_tokens = (
                math.ceil(num_mm_embeds / min_soft_tokens)
                * max_soft_tokens
                * pooling_k2
            )

    if modality == "audio":
        # 音频塔与连接器共享同一 token 预算，默认取 num_mm_embeds；
        # 若提供 padding 后的特征，则按每 4 帧打包 1 个 token 重新估算。
        tower_tokens = num_mm_embeds
        connector_tokens = num_mm_embeds

        if mm_kwargs is not None:
            field = mm_kwargs.get("input_features_padded")
            if field is not None:
                data = field.data
                if isinstance(data, torch.Tensor) and data.ndim >= 2:
                    batch_size = math.prod(data.shape[:-2])
                    audio_tokens = batch_size * math.ceil(data.shape[-2] / 4)
                    tower_tokens = audio_tokens
                    connector_tokens = audio_tokens

    return tower_tokens, connector_tokens
```

vllm/lora/model_manager.py

LoRA 管理器初始化改为按模态取 tower/connector token 预算最大值来创建 Punica wrapper，是运行时正确性的核心配置逻辑。

```
mm_budget = MultiModalBudget(vllm_config, mm_registry)
limit_per_prompt = max(mm_budget.mm_max_items_per_prompt.values())
max_lora_tokens = mm_budget.get_encoder_budget()
# 对每个 modality 分别询问 LoRA token 预算，避免单一模态的
# 计数口径掩盖其他模态（如图像 / 视频 / 音频）的更大需求。
lora_token_counts_by_modality = [
    self.model.get_mm_lora_token_counts(
        modality=modality,
        mm_kwargs=None,
        num_mm_embeds=max_lora_tokens,
    )
    for modality in mm_budget.mm_max_toks_per_item
]
num_encoder_tokens = max(
    tower_tokens for tower_tokens, _ in lora_token_counts_by_modality
)

# Tower wrappers 统一用最大 tower token 预算，保证任何模态
# 的 encoder 输出都能被 Punica wrapper 容纳。
tower_punica_wrapper = get_punica_wrapper(
    num_encoder_tokens,
    max_batches=self.max_num_seqs * limit_per_prompt,
    device=self.device,
    lora_config=self.lora_config,
)
for prefix in self.mm_mapping.tower_model:
    self.punica_wrapper_mapping[prefix] = tower_punica_wrapper

# Use wrapper for connector if present.
if self.mm_mapping.connector:
    connector_tokens = max(
        (
            connector_tokens
            for _, connector_tokens in lora_token_counts_by_modality
            if connector_tokens is not None
        ),
        default=None,
    )
    if connector_tokens is not None:
        connector_punica_wrapper = get_punica_wrapper(
            connector_tokens,
            max_batches=self.max_num_seqs * limit_per_prompt,
            device=self.device,
            lora_config=self.lora_config,
        )
        for prefix in self.mm_mapping.connector:
```

```
                self.punica_wrapper_mapping[prefix] = connector_punica_wrapper
        else:
            logger.warning_once(
                "Connector LoRA support disabled: model does not implement "
                "get_num_mm_connector_tokens(). This method is required to "
                "determine the connector's token budget for LoRA operations."
            )
```

评论区精华

核心讨论集中在接口设计与正确性上：

- depthfirst-app 指出 vllm/v1/worker/gpu/mm/lora.py 中 or not mm_mapping.connector and all(...) 存在 Python 运算符优先级问题（and 优先于 or），导致 connector 计数含 None 时早退守卫失效。最终版本已改为 or not all(count is not None ...)，问题修复。
- chatgpt-codex-connector[bot] 提出两点 P2 建议：一是在量化层（如 GGUF 替换 weight 属性）上不应直接读 weight；二是 MM LoRA 容量不应被 max_num_batched_tokens 截断，而应使用 mm_budget.get_encoder_budget()。最终版本均体现：早期 vision tower 重实现被移除，max_lora_tokens 直接取 get_encoder_budget()。
- gemini-code-assist 曾报告 zip(strict=True) 在 Python 3.9 不兼容，但该问题位于早期 vision tower 实现中，随 #43798 路线精简后已不适用。
- 用户 taroshi 实测发现 vision_tower.encoder.layers.0~15 的 self_attn.o_proj 与 mlp.down_proj（Gemma4ClippableLinear）无法被任何 LoRA 层包装而被忽略，意味着这些层的 LoRA 权重不会生效，该问题在 PR 内未见明确结论。
- 用户 anshulkulhari7 高度认可新接口（对可变长度音频塔模型是 O(1) 形状读取），但指出 hasattr 门控会放过只继承 SupportsMultiModal stub 的模型，未实现时返回 None 并直接喂给 get_punica_wrapper，存在崩溃风险；他正基于此接口推进 Ultravox connector-LoRA（#45771、#45944、#45697）。
- gemini-code-assist: zip(strict=True) 在 Python 3.9 不兼容 (correctness): 最终版本按 #43798 路线移除了 vision tower 重实现，该问题随代码删除而消失；但历史评论仍指向已删除的 diff。
- codex: 量化线性层上不应直接读 weight (correctness): 该问题针对早期 vision tower 重实现，最终代码改为走 Transformers 后端路径，不再直接构造 vLLM 量化线性层。
- codex: V2 MM LoRA 应使用新 count API (design): 最终版本同步更新了 vllm/v1/worker/gpu/mm/lora.py 的 set_active_mm_loras，问题已解决。
- codex: MM LoRA 容量不应被 max_num_batched_tokens 截断 (design): 最终 model_manager.py 直接使用 mm_budget.get_encoder_budget()，未做 min 截断，符合建议。
- depthfirst-app: mm/lora.py 守卫条件运算符优先级 bug (correctness): 最终版本改为 or not all(count is not None ...)，守卫语义正确。
- jeejeelee: 建议拆分 vision tower 支持为独立 PR (design): PR 最终按此建议聚焦运行时 mapping 与 token 计数，vision tower 转换交由 #43798 完成。

- anshulkulhari7: hasattr 门控放行 stub 模型 (design): PR 内未看到针对该问题的新增守卫；后续消费者（#45771、#45944、#45697）会基于此接口继续演进。
- taroshi: 部分 Gemma4 视觉层 LoRA 无法被包装 (correctness): PR 内未见修复或回复；该问题可能使部分层 LoRA 不生效，需要后续 PR 处理。

风险与影响

- 风险：主要风险点：
- LoRA 覆盖不完整：taroshi 实测多组 Gemma4ClippableLinear（o_proj、down_proj）无法被包装而静默忽略，用户微调这些层时适配器行为不完整，可能产生错误输出而难以察觉。
- 新接口默认实现返回 None 的连锁问题：get_mm_lora_token_counts 默认实现委托给 get_num_mm_encoder_tokens，而该方法是带 ... 的 stub，所有 SupportsMultiModal 模型都能通过 hasattr 检查。一旦某模型未真正实现且 enable_tower_connector_lora=True，model_manager.py 中 max(tower_tokens for ...) 会收到 None 并抛 TypeError。虽然该功能为实验性开关，但默认路径缺少显式校验。
- 图像批处理显存上升：_process_image_input 在 MM LoRA 启用时放弃分辨率 bucketing 与内存感知分块，所有图像 padding 到最大 patch 并整批编码，高分辨率或大批次场景峰值显存明显增加；超限图像直接抛 ValueError，可能影响在线服务稳定性。
- 测试配置矛盾：tests/lora/test_gemma4_tp.py 文件头注明“不会在 CI 触发，仅本地验证”，但 .buildkite/test_areas/lora.yaml 已将其加入 CI，两者不一致可能导致 CI 行为与预期不符（或文件头注释过时）。
- 早期 review 指向已删除代码：量化层读取 weight、zip(strict=True) 等评论针对的是早期 vision tower 重实现，最终版本已移除该实现，历史讨论可能误导后来阅读者。
- 影响：影响范围：
- 用户侧：Gemma4（gemma-4-E2B-it 等）用户可直接加载经 unsloth 等工具微调的 vision LoRA，issue #40693 得到修复；但部分视觉层 LoRA 不生效问题会影响效果一致性。
- 系统侧：get_mm_lora_token_counts 成为多模态 LoRA 的新契约，V1 主执行器与 V2 runner 路径（mm/lora.py）同步改造，未来任何多模态模型接入 tower/connector LoRA 都必须遵循该接口；model_manager 的 wrapper 配置逻辑从单一模态预算升级为跨模态最大预算。
- 团队侧：该接口为 Ultravox 等可变长度音频塔模型的 connector-LoRA 铺路（#45771、#45944、#45697），属于 multi-modality + LoRA 能力线的关键前置。
- 风险标记：部分视觉层 LoRA 权重被静默忽略，新接口 stub 返回 None 可能引发崩溃，MM LoRA 下图像批处理显存峰值上升，测试文件与 CI 配置不一致，早期 review 指向已删除代码

关联脉络

- PR #43798 Convert Gemma4-MM vision linear layers through Transformers backend: PR body 明确依赖此前置：Gemma4-MM 视觉线性层已走 Transformers 后端路径，本 PR 不再重实现 vision tower。

- PR #48215 [Model][LoRA] Add tower/connector LoRA support for Ultravox: 同属多模态 tower/connector LoRA 功能线，与本 PR 引入的计数接口和 wrapper 机制高度相关。
- PR #45771 Ultravox connector-LoRA consumer (staged by anshulkulhari7): anshulkulhari7 在评论中表示正基于 get_mm_lora_token_counts 接口推进 Ultravox connector-LoRA，等待本 PR 合并。
- PR #45944 Ultravox connector-LoRA consumer (staged by anshulkulhari7): 评论中提到的同一工作线，依赖本 PR 的新接口。
- PR #45697 Ultravox connector-LoRA consumer (staged by anshulkulhari7): 评论中提到的同一工作线，依赖本 PR 的新接口。