PR #43617 完整报告

vllm-project/vllm

Fix Qwen3-VL and Qwen3-omni-thinker accuracy degradation from deepstack inputs under torch.compile

合并时间：2026-05-28 06:34

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43617

执行摘要

- 一句话：修复 Qwen3-VL/Omni 在 torch.compile 下的精度退化
- 推荐动作：该 PR 值得精读，因为它揭示了一个常见的 torch.compile 陷阱：profile 阶段与 serving 阶段的输入结构不一致会导致编译图特化错误。设计上通过固定返回 tensor 而非 None 来保持图结构稳定的模式值得借鉴。合并前建议考虑的 device/dtype 问题可在后续 PR 中加固。

功能与动机

作者在 Issue #43602 中报告 Qwen3-VL-2B-Instruct 使用 vLLM 默认 compile 路径时 Geo3K 精度 (0.205) 明显低于 SGLang (0.288)，而 eager 模式 (0.291) 接近预期。PR body 指出根本原因是 'compile/profiling path: deepstack_input_embeds = None; real VL request path: deepstack_input_embeds = IntermediateTensors(...)'，导致编译图特化忽略视觉分支。

实现拆解

1. 消除提前返回：在 _get_deepstack_input_embeds 中，移除对 deepstack_input_embeds_num_tokens == 0 时的 return None 分支，改为检查 num_tokens 是否超出缓存容量，若超出则调用新辅助方法 _resize_deepstack_input_embeds 扩容，然后始终返回 IntermediateTensors（零张量或真实张量）。
2. 提取 resize 辅助方法：将 _set_deepstack_input_embeds 中已有的内联 resize 逻辑抽取为 _resize_deepstack_input_embeds 方法，被 _get_deepstack_input_embeds 和 _set_deepstack_input_embeds 共用，消除重复代码。
3. 双模型同步修改：在 qwen3_vl.py 和 qwen3_omni_moe_thinker.py 中做完全相同的修改，因为两个模型共享相同的 deepstack 机制和 bug。
4. 零张量语义：当没有视觉输入时，返回的 IntermediateTensors 包含全零张量，语义等同于无 deepstack 贡献，但保持了 tensor-based 输入结构，使编译图不因输入结构变化而重编译。

关键文件：

- vllm/model_executor/models/qwen3_vl.py（模块 模型执行；类别 source；类型 data-contract；符号 _resize_deepstack_input_embeds, _get_deepstack_input_embeds, _set_deepstack_input_embeds）：核心修复文件之一，修改了 _get_deepstack_input_embeds 的提前返回逻辑，并抽取 _resize_deepstack_input_embeds 方法；Qwen3-VL 模型的精度退化直接由此修复。
- vllm/model_executor/models/qwen3_omni_moe_thinker.py（模块 模型执行；类别 source；类型 data-contract；符号 _resize_deepstack_input_embeds, _get_deepstack_input_embeds, _set_deepstack_input_embeds）：Qwen3-Omni-Thinker 模型的同等修复，改动与 qwen3_vl.py 一致；确保 Omni 模型在 torch.compile 下的精度。

关键符号：_get_deepstack_input_embeds, _resize_deepstack_input_embeds, _set_deepstack_input_embeds

关键源码片段

vllm/model_executor/models/qwen3_vl.py

核心修复文件之一，修改了 _get_deepstack_input_embeds 的提前返回逻辑，并抽取 _resize_deepstack_input_embeds 方法；Qwen3-VL 模型的精度退化直接由此修复。

```
def _get_deepstack_input_embeds(
    self,
    num_tokens: int,
) -> IntermediateTensors | None:
    if not getattr(self, "deepstack_input_embeds", None):
        return None # 如果视觉 tower 被跳过（如纯文本模型）
    # 关键变更：不再因无有效 payload 而返回 None，而是确保 buffer 足够大
    if num_tokens > self.deepstack_input_embeds[0].size(0):
        self._resize_deepstack_input_embeds(num_tokens)
    # 始终返回 IntermediateTensors，零张量语义等价于无 deepstack 贡献
    return IntermediateTensors({
        f"deepstack_input_embeds_{idx}": self.deepstack_input_embeds[idx][
            :num_tokens]
        for idx in range(self.deepstack_num_level)
    })


def _resize_deepstack_input_embeds(self, num_tokens: int) -> None:
    # 新抽出的辅助方法：用零张量重新分配 buffer
    # 注：此处 device 从已有的 self.deepstack_input_embeds[0] 继承
    # 在首次 GPU 操作后该张量会在正确 device 上
    self.deepstack_input_embeds = [
        torch.zeros(
            num_tokens,
            self.config.text_config.hidden_size,
            device=self.deepstack_input_embeds[0].device,
            dtype=self.deepstack_input_embeds[0].dtype,
        )
```

```
        for _ in range(self.deepstack_num_level)
    ]


def _set_deepstack_input_embeds(self, deepstack_input_embeds: torch.Tensor) -> None:
    if not getattr(self, "deepstack_input_embeds", None):
        return
    num_tokens = deepstack_input_embeds.size(1)
    if num_tokens > self.deepstack_input_embeds[0].size(0):
        self._resize_deepstack_input_embeds(num_tokens) # 复用新方法
    for idx in range(self.deepstack_num_level):
        self.deepstack_input_embeds[idx][:num_tokens].copy_(
            deepstack_input_embeds[idx])
    self.deepstack_input_embeds_num_tokens = num_tokens
```

评论区精华

1. Gemini Code Assist 提出的 device/dtype 风险：机器人指出初始化时 deepstack_input_embeds 在 CPU 上创建，若 resize 条件不满足（如 text-only 请求 num_tokens ≤ max_num_batched_tokens），则返回 CPU 张量，后续 GPU 运算会因 device 不匹配崩溃。建议在 resize 条件中增加 device/dtype 检查，并使用 self.visual.device / self.visual.dtype。 - 结论：该风险在 PR 合并前未在代码中处理，但实际运行时 resize 在第一次 GPU 请求时被触发，且后续 resize 继承第一次 GPU 张量的 device/dtype，因此未暴露。建议后续 PR 加固。
2. Andakai 的分析补充：作者在 PR 评论中解释了 root cause：vLLM 的 pre-profile 阶段 encoder 和 decoder 分开 profiling，decoder dummy run 不含 deepstack_input_embeds，导致 torch.compile 无法捕获 deepstack 分支。

- device/dtype 不匹配风险：_resize_deepstack_input_embeds 应使用 self.visual 的 device/dtype (correctness): 作者未在本次 PR 中处理，但风险在现有逻辑下被规避：首次 GPU 请求触发 resize 后 buffer 已迁移到 GPU，后续 resize 继承正确 device/dtype。建议后续 PR 加固。
- root cause 分析：encoder 与 decoder 分开 profile 导致图特化 (design): 通过始终返回 IntermediateTensors 保持输入结构稳定，从而避免图重编译。

风险与影响

- 风险：
  1. device/dtype 不匹配风险（低概率，高影响）：如 review 评论指出，初始化时 deepstack_input_embeds 在 CPU 上分配，若第一个请求为 text-only（无视觉）且 num_tokens ≤ max_num_batched_tokens，_resize 不会被调用，返回 CPU 张量导致 GPU 崩溃。实际场景中，text-only 请求通常不会触发 deepstack 逻辑（因 getattr(self, "deepstack_input_embeds", None) 可能为 None），但若视觉 tower 存在但未提供视觉输入，则此风险存在。
  2. 性能影响：返回零张量比返回 None 增加了少量的 GPU memory 和计算开销，但 zero-backed IntermediateTensors 在 decoder 中贡献为零，且被 compiled graph 高

效吸收，实测性能反而略有提升（102.6s vs 116.8s）。

3. 回归风险：修改仅影响 deepstack 路径，不影响纯文本或非 deepstack 模型，回归概率低。 - 影响：影响范围：Qwen3-VL 和 Qwen3-Omni-Thinker 模型在 torch.compile 默认路径下的精度恢复，对所有使用这些模型的多模态用户有直接正面影响。影响程度：高（精度从接近随机恢复到接近 eager 水平，Geo3K 提升约 0.08~0.06）。性能：略有提升（约 12% 更快）。 - 风险标记：核心路径变更，涉及 torch.compile 编译图特化，缺少测试覆盖

关联脉络

- PR #43602 [Bug]: Qwen3-VL-2B-Instruct Geo3K accuracy score lower than SGLang with deterministic sampling: 直接关联的 bug report，本 PR 修复了该 issue 报告的核心问题。
- PR #43540 [Quantization] Fix Humming RoutedExperts import: 同为模型相关 bugfix，但与本 PR 无直接技术关联。
- PR #43361 [8/n] Migrate merge_attn_states, mamba, sampler to torch stable ABI (continued): 涉及 torch compile 和编译基础设施，与本 PR 的 torch.compile 问题有间接关联。