

PR #53529 完整报告

vllm-project/vllm

[Test][Qwen3-VL] Cover compiled DeepStack input contract

合并时间: 2026-08-31 18:46

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/53529>

执行摘要

- 一句话: 新增 Qwen3-VL DeepStack 编译契约回归测试
- 推荐动作: 值得精读。它展示了一个高价值的测试模式: 用 `object.__new__` 构造裸模型实例 + 最小可编译解码器 + 复用真实私有方法, 把 `torch.compile` 图特化这一隐性正确性约束编码成可执行断言, 是“以测试锁定不稳定输入契约”的范例。PR body 对 #43617 与 #53516 冲突的处理方式 (锁定契约、质量争议单独调查) 也是处理“修复引入新回归”争议的务实参考。对从事 `torch.compile` / CUDA graph 特化或多模态模型维护的工程师尤其有借鉴意义。注意: 由于 #53516 尚未调查清楚, 该契约未来仍可能被重新审视。

功能与动机

PR body 明确指出动机: #43617 修复了一个正确性问题——`compile warmup` 可能以 `deepstack_input_embeds=None` 调用解码器, 使编译图特化为 `no-deepstack` 路径, 之后真实多模态请求会静默忽略 DeepStack 特征 (Geo3K 评测 2B 模型分数从 0.2047 回升至 0.2829)。该修复建立的“总是返回 zero-backed IntermediateTensors”契约此前没有任何测试保护, 且 #53516 报告恢复旧 guard 后人物检测指标反而上升 26.79 个点, 形成方向相反的评测结论。PR 的定位是: “This PR does not fix #53516. ... restoring it also reintroduces the compiled-graph correctness bug fixed by #43617. The test here protects the stable input contract while the underlying quality question is investigated separately.”即用回归测试锁定稳定契约, 质量争议另开调查, 避免以“恢复旧行为”的方式回退已被全量评测验证的正确性修复。

实现拆解

1. 测试骨架与编译环境: 新建 `tests/compile/test_deepstack_input_contract.py`, 引入 `support_torch_compile`、`VllmConfig` / `CompilationConfig` (`CompilationMode.VLLM_COMPILE`)、`set_current_vllm_config`、`set_forward_context` 等编译基础设施; 测试开头用 `monkeypatch.setenv("VLLM_DISABLE_COMPILE_CACHE", "1")` 关闭编译缓存, 确保每次运行真实触发重新编译、结果可复现。
2. 最小可编译解码器 `_DeepStackDecoder`: 一个仅含 `forward` 的 `nn.Module`, 用 `@support_torch_compile(dynamic_arg_dims={"x": 0, "deepstack_input_embeds": 0})` 声明动态维度并真实走 vLLM 编译链路; `forward` 在 `deepstack_input_embeds` 非 `None` 时把 `deepstack_input_embeds_0` 加到 `x` 上, 是真实 Qwen3-VL 解码层 DeepStack 注入路径的最小还原。

3. 绕过 `init__` 构造真实模型实例：用 `object.__new__(Qwen3VLForConditionalGeneration)` 创建“裸”实例，手工设置 `deepstack_input_embeds=[torch.zeros(2, 4, device="cuda")]`、`deepstack_input_embeds_num_tokens=0`、`deepstack_num_level=1`；既不加载权重，又直接复用真实的 `_get_deepstack_input_embeds / _set_deepstack_input_embeds` 方法，保证测试锁定的是真实契约而非 `mock`。
4. 契约执行流程 `_run_contract`：先以空 `payload`（零张量）调用编译后的模型模拟 `warmup`，再通过 `_set_deepstack_input_embeds` 注入全 7.0 的非零 `payload` 并再次调用；返回 `x`、`first`、`second` 三份结果用于数值对比。
5. 断言即回归保护：`torch.testing.assert_close(first, x)` 验证空 `payload` 语义等价于无贡献；`torch.testing.assert_close(second, x + 7)` 验证 `warmup` 后的编译图仍消费真实 `payload`。若旧 `num_tokens == 0` 早退恢复，`empty_payload` 为 `None`、编译图特化到 `no-deepstack` 路径，`second` 会等于 `x` 而非 `x + 7`，测试立即失败。配套验证：作者报告 `test_deepstack_input_contract.py` 通过、`test_qwen3_vl_mrope.py` 96 个用例通过、`ruff` 检查通过，并在 `stop_sign / cherry_blossom` 图像上确认编译与 `eager` 输出一致、恢复旧 `guard` 会同时改变两份输出。

关键文件：

- `tests/compile/test_deepstack_input_contract.py`（模块 编译契约；类别 `test`；类型 `test-coverage`；符号 `_DeepStackDecoder`，`_run_contract`，`test_qwen3_vl_deepstack_payload_after_empty_compile`）：本次唯一变更文件，将 #43617 建立的 `DeepStack` 输入契约编码为回归测试；若恢复旧 `num_tokens == 0` 提前返回 `None` 的逻辑，测试将失败。

关键符号：`test_qwen3_vl_deepstack_payload_after_empty_compile`，`_run_contract`，`_DeepStackDecoder.forward`

关键源码片段

`tests/compile/test_deepstack_input_contract.py`

本次唯一变更文件，将 #43617 建立的 `DeepStack` 输入契约编码为回归测试；若恢复旧 `num_tokens == 0` 提前返回 `None` 的逻辑，测试将失败。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project

import torch
from torch import nn

from vllm.compilation.decorators import support_torch_compile
from vllm.config import (
    CompilationConfig,
    CompilationMode,
    VllmConfig,
    set_current_vllm_config,
)
from vllm.forward_context import set_forward_context
```

```

from vllm.model_executor.models.qwen3_vl import (
    Qwen3VLForConditionalGeneration,
)
from vllm.sequence import IntermediateTensors

# 用极简解码器模拟 Qwen3-VL 解码层的 DeepStack 相加路径，并真实走 torch.compile
# 编译链路。dynamic_arg_dims 声明 x 与 deepstack_input_embeds 都是动态维度，
# 模拟真实解码中 batch/token 数可变的情况。
@support_torch_compile(
    dynamic_arg_dims={"x": 0, "deepstack_input_embeds": 0},
)
class _DeepStackDecoder(nn.Module):
    def __init__(self, *, vllm_config: VllmConfig, prefix: str = "") -> None:
        super().__init__()

    def forward(
        self,
        x: torch.Tensor,
        deepstack_input_embeds: IntermediateTensors | None,
    ) -> torch.Tensor:
        if deepstack_input_embeds is not None:
            # 与真实 Qwen3-VL 解码循环一致：把第 0 层 deepstack 特征加到 hidden
            # states 上；传入 None 时保持 x 不变。
            x = x + deepstack_input_embeds["deepstack_input_embeds_0"]
        return x

    def _run_contract(
        self,
        deepstack_buffer: Qwen3VLForConditionalGeneration,
    ) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor]:
        # VLLM_COMPILE 模式让上面的 _DeepStackDecoder 真实经历编译，模拟 vLLM 默认
        # serving 路径下的 compile warmup。
        config = VllmConfig(
            compilation_config=CompilationConfig(mode=CompilationMode.VLLM_COMPILE)
        )
        with set_current_vllm_config(config):
            model = _DeepStackDecoder(vllm_config=config).eval().cuda()

        x = torch.arange(8, dtype=torch.float32, device="cuda").reshape(2, 4)
        with torch.inference_mode(), set_forward_context(None, vllm_config=config):
            # 第 1 步：以空 payload（零张量）完成第一次编译调用，模拟 warmup 阶段。
            # 若 _get_deepstack_input_embeds 在 num_tokens == 0 时返回 None，编译图
            # 会特化为 no-deepstack 路径——这正是 #43617 修复的根因场景。
            empty_payload = deepstack_buffer._get_deepstack_input_embeds(num_tokens=2)
            first = model(x, empty_payload)

            # 第 2 步：注入已知非零 payload（全 7.0）后再次取 deepstack 特征。稳定
            # 契约下这里必须返回 IntermediateTensors 而不是 None。

```

```

    payload = torch.full((1, 2, 4), 7.0, device="cuda")
    deepstack_buffer._set_deepstack_input_embeds(payload)
    active_payload = deepstack_buffer._get_deepstack_input_embeds(num_tokens=2)
    assert active_payload is not None
    second = model(x, active_payload)

return x.cpu(), first.cpu(), second.cpu()

# 回归测试主体：绕过 __init__ 构造“裸” Qwen3VL 实例，手工布置 deepstack 缓冲
# 与计数，从而在不加载权重的前提下复用真实模型方法。
def test_qwen3_vl_deepstack_payload_after_empty_compile(monkeypatch):
    # 关闭编译缓存，保证每次运行都真实触发重新编译、结果可复现。
    monkeypatch.setenv("VLLM_DISABLE_COMPILE_CACHE", "1")

    deepstack_buffer = object.__new__(Qwen3VLForConditionalGeneration)
    # 缓冲区已存在但内容全零、token 计数为 0：等价于“当前没有真实 DeepStack 特征”。
    deepstack_buffer.deepstack_input_embeds = [torch.zeros(2, 4, device="cuda")]
    deepstack_buffer.deepstack_input_embeds_num_tokens = 0
    deepstack_buffer.deepstack_num_level = 1

    x, first, second = _run_contract(deepstack_buffer)
    # 空 payload 在语义上等价于无 DeepStack 贡献：输出应与输入完全一致。
    torch.testing.assert_close(first, x)
    # 关键断言：warmup 时编译过的图必须继续消费后续到来的真实 payload。若旧的
    # num_tokens == 0 提前返回 None 逻辑被恢复，second 会等于 x 而不是 x + 7，
    # 测试立即失败，从而锁定 #43617 建立的稳定输入契约。
    torch.testing.assert_close(second, x + 7)

```

评论区精华

PR 的 review 线程没有实质技术交锋：Isotr0py 直接 APPROVED，claude[bot] 仅提示 fork 仓库自动 review 被禁用，其余评论均为 /ci run、/ci retry 等 CI 触发指令。真正的讨论发生在 PR body 与两个关联 issue 之间：

- PR 作者明确划定边界：“This PR does not fix #53516... restoring it also reintroduces the compiled-graph correctness bug fixed by #43617. The test here protects the stable input contract while the underlying quality question is investigated separately.”
- 53516 提交者给出实测证据：恢复旧 guard 在人物检测任务上 +26.79 个点（42.86% → 45.54%，McNemar $\chi^2(cc)=23.36$ ，33 对 discordant 中 31/31 经人工盲评确认为“报告了真实存在的人”），要求恢复旧行为。
- 43617 则用 Geo3K 全量评测证明旧行为在 compile 路径下显著偏低（2B: 0.2047 → 0.2829；8B: 0.5225 → 0.5657），且指出 eager 路径不受影响。
- 结论：先保契约稳定性，测试锁定 #43617 行为，质量问题单独调查。这是“修复引入新回归”争议下的务实决策：不因单点质量报告而回退已被全量评测验证的正确性修复。

- 43617 契约 vs #53516 质量回退的取舍 (design): 接受 PR 的边界划定: 先保契约稳定, 质量争议单独开 issue 调查。
- 回归测试对旧 guard 的拦截能力 (testing): 测试设计成立, Isotr0py APPROVED, CI 通过; 作者另在 stop_sign 与 cherry_blossom 图像上确认编译与 eager 输出一致。

风险与影响

- 风险:
 1. 测试与 Qwen3-VL 私有方法强耦合: 直接调用 `_get_deepstack_input_embeds / _set_deepstack_input_embeds` 并依赖精确数值语义 (`second == x + 7`); 若后续实现调整 `reshape / 加法细节` (如 #49397 同类演进), 测试可能因数值细节失败而非契约被破坏, 需要维护者分辨假阳性。
 2. 环境依赖: 测试需要 CUDA 设备与完整 `torch.compile` 链路, 未看到针对 CPU / ROCm 的 `skip` 标记, 需确认 Buildkite GPU 任务覆盖该文件, 否则可能出现平台性失败。
 3. 契约锁定效应: 测试会阻塞任何“恢复旧 guard”的改动 (即 #53516 的诉求), 可能引发社区讨论; 由于 #53516 尚未定位根因, 该契约未来仍可能被重新审视, 测试需随之演进。
 4. 覆盖盲区: 只覆盖“缓冲区存在但 token 数为 0”的分支; #49397 引入的“vision tower 不可用时跳过 DeepStack 缓冲”分支没有对应的编译回归保护。- 影响: 用户侧无运行时行为影响, 纯测试变更。系统侧: 为 #43617 的 zero-backed 输入契约提供持续回归保护, 任何未来重构若让 `_get_deepstack_input_embeds` 在空 payload 时返回 `None`, CI 会立即拦截, 避免编译图特化正确性 bug 再次潜入。团队侧: 测试成为 #53516 争议的事实基准——修改该契约的 PR 必须同时更新或绕过此测试, 客观上提高了行为变更的决策门槛; 同时测试把 `torch.compile` 图特化的隐性约束文档化, 降低后续维护者的认知负担。总体影响程度为低到中: 变更面仅一个测试文件, 但语义上牵制未来编译路径的行为演进。- 风险标记: 仅测试变更, 无运行时功能覆盖, 测试与 Qwen3-VL 私有方法强耦合, 依赖 CUDA 与 `torch.compile` 环境, 未覆盖 vision tower 跳过分支

关联脉络

- PR #43617 Fix Qwen3-VL and Qwen3-omni-thinker accuracy degradation from deepstack inputs under torch.compile: 本 PR 锁定的 zero-backed 输入契约正是 #43617 建立的; PR body 明确引用其为根因修复, 测试的失败条件即恢复该 PR 移除的 `early return`。
- PR #53516 [Bug]: Qwen3-VL — #43617's removal of the `num_tokens == 0` early return costs 27 points of people detection under torch.compile (human-verified): 关联 issue (非 PR): 本 PR 的边界声明直接针对它——不修复该质量回退报告, 仅用测试锁定契约; 若后续调查有结论, 测试断言可能需要演进。
- PR #49397 Keep DeepStack contract for vision-enabled models while skipping buffers when vision tower is unavailable (标题按 PR body 描述整理): PR body 列为 related work: #49397 在 vision tower 不可用时跳过 DeepStack 缓冲但保持输入契约; 本测试未覆盖该分支, 与它共同构成“契约守护”链条。