

PR #50411 完整报告

vllm-project/vllm

[Model] Fused mm preprocess normalisation on the Device

合并时间: 2026-08-06 18:11

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/50411>

执行摘要

- 一句话: 图像归一化搬上 GPU, Qwen2VL 预处理提速近一倍
- 推荐动作: 值得精读。设计上有三个可借鉴点: 用 eval 模式 batch_norm 折叠 rescale + normalize 为一次仿射变换并自动检测恒等路径; 归一化参数「配置文件 → HF 实例 → 开关默认值 → 快速失败」的分级解析回退链; 模型能力标记 + 前端能力约束的开关解析模式 (`_resolve_mm_device_do_normalize`)。review 讨论中关于「读处理器类默认值不严谨」与「fail-fast vs 静默回退」的取舍对后续多模态预处理开发有参考价值。

功能与动机

PR body 中作者 noooop 给出两条动机: `normalize` 在 GPU 上计算时可融合进后续算子, 即使不融合也几乎不耗时; 图像数据从入口到引擎再到 GPU 全程以 `uint8` 传输, 直到输入归一化前才转成 `bf16`, 把归一化搬上设备与现有传输路径天然契合, 且可推广到所有视觉模型。实测数据: `Qwen2VLImageProcessor` 对 `1024x1024` 图像预处理速度接近翻倍 (`12 ms` → `6 ms`); `vllm bench` 平均 TTFT 从 `3809.70 ms` 降至 `3459.52 ms` (Rust 前端基线 `3078.26 ms`); PPL 测试开 / 关该特性与 `Transformers` 的差异完全一致 (约 `0.036%`)。

实现拆解

1. 配置契约扩展: `MultiModalConfig` 新增 `mm_device_do_normalize` 字段 (默认 `True`) 并纳入 `compute_hash`; `ModelConfig` 新增同名 `InitVar` 与 `_resolve_mm_device_do_normalize` 决策方法——未显式指定时跟随模型能力标记 (Rust 前端强制 `False`), 用户显式开启但不被支持时回退并输出 `warning`; `EngineArgs` 暴露 `--mm-device-do-normalize` CLI 参数 (默认 `None` 走模型能力推断)。
2. 处理器参数读取重构: `vllm/transformers_utils/processor.py` 将 `get_processor_cls_name_from_config` 拆分为返回完整配置 dict 的 `get_processor_config` 与原有包装函数, 供 `FusedInputNorm.from_model_config` 读取 `preprocessor_config.json` 中的 `image_mean`、`image_std`、`rescale_factor`、`do_rescale`、`do_normalize`; 任一字段缺失时回退到 HF `image_processor` 实例属性。
3. 核心归一化模块: `vllm/model_executor/models/vision.py` 新增 `FusedInputNorm`, 将 $(input * rescale_factor - mean) / std$ 折叠为一次 `F.batch_norm` (`weight = 1/std`、`bias = -mean/std`、`running_var = 1`、`eps = 0`), 并自动检测恒等场景跳过计算。
4. 模型接入: `Qwen2VLForConditionalGeneration` 与 `Qwen2_5_VLForConditionalGeneration` 声明 `supports_mm_device_do_normalize =`

True, 在 `_mark_tower_model` 上下文内构造 `self.input_norm`, 并在 `_process_image_input / _process_video_input` 中对图像与视频帧 `pixel_values` 做设备端归一化, `image_embeds` 分支统一 `.type(self.visual.dtype)`。

5. 配套改动: `MultiModalConfig.merge_mm_processor_kwargs` 在开启时强制注入 `do_normalize=False`、`do_rescale=False` (推理期 `inference_kwargs` 仍可覆盖); 测试新增 `test_mm_device_do_normalize` 数值一致性用例与 Qwen 系列 PPL 双模式参数化; 文档更新 `docs/design/mm_processing.md`。

关键文件:

- `vllm/model_executor/models/vision.py` (模块 视觉模型; 类别 source; 类型 core-logic; 符号 `FusedInputNorm`, `FusedInputNorm.from_model_config`, `FusedInputNorm.forward`, `FusedInputNorm.identity`): 新增 `FusedInputNorm` 核心模块, 用 `F.batch_norm` 实现设备端 `rescale + normalize` 等价仿射变换, 含 `identity` 快速路径、`from_model_config` 分级参数解析与 `forward` 实现, 是本 PR 的心脏。
- `vllm/config/model.py` (模块 配置层; 类别 source; 类型 configuration; 符号 `_resolve_mm_device_do_normalize`): 新增 `mm_device_do_normalize` `InitVar` 与 `_resolve_mm_device_do_normalize` 决策方法, 处理默认值推断、Rust 前端强制关闭与模型能力校验, 是整套开关逻辑的入口。
- `vllm/config/multimodal.py` (模块 配置层; 类别 source; 类型 configuration; 符号 `merge_mm_processor_kwargs`, `compute_hash`): `MultiModalConfig` 新增 `mm_device_do_normalize` 字段并纳入 `compute_hash`, 同时 `merge_mm_processor_kwargs` 在开启时强制注入 `do_normalize=False / do_rescale=False`, 构成本特性的数据契约与处理器行为联动。
- `vllm/model_executor/models/qwen2_vl.py` (模块 视觉模型; 类别 source; 类型 core-logic; 符号 `_process_image_input`, `_process_video_input`, `init`): `Qwen2VLForConditionalGeneration` 声明 `supports_mm_device_do_normalize = True`, 在 `_mark_tower_model` 内构造 `input_norm`, 并在图像 / 视频处理路径接入设备端归一化, 是本特性第一个落地模型。
- `vllm/model_executor/models/qwen2_5_vl.py` (模块 视觉模型; 类别 source; 类型 core-logic; 符号 `_process_image_input`, `_process_video_input`, `init`): `Qwen2.5-VL` 镜像接入同一套 `FusedInputNorm` 机制, 验证该模式可平移至同类视觉模型, 扩大特性覆盖范围。
- `vllm/transformers_utils/processor.py` (模块 处理器工具; 类别 source; 类型 refactor; 符号 `get_processor_config`, `get_processor_cls_name_from_config`): 将 `get_processor_cls_name_from_config` 拆出 `get_processor_config` (返回完整配置 dict), 供 `FusedInputNorm` 从 `preprocessor_config.json` 读取归一化参数, 是参数解析回退链的基础设施。
- `vllm/engine/arg_utils.py` (模块 启动参数; 类别 source; 类型 configuration; 符号 `create_model_config`, `add_cli_args`): `EngineArgs` 新增 `mm_device_do_normalize` 字段与 `--mm-device-do-normalize` CLI 参数 (默认 None 交 `ModelConfig` 推断), 打通用户入口。

- vllm/model_executor/models/interfaces.py (模块 模型接口; 类别 source; 类型 data-contract; 符号 SupportsMultiModal) : SupportsMultiModal 协议新增 supports_mm_device_do_normalize 类变量 (默认 False) , 定义模型能力标记契约, 是默认关闭、按模型开启机制的基础。
- tests/models/multimodal/processing/test_qwen2_vl.py (模块 测试; 类别 test; 类型 test-coverage; 符号 test_mm_device_do_normalize) : 新增 test_mm_device_do_normalize 数值一致性测试: 对比处理器端归一化结果与 FusedInputNorm 设备端结果, 用 torch.testing.assert_close 保证等价。
- tests/models/multimodal/generation_ppl_test/test_qwen.py (模块 测试; 类别 test; 类型 test-coverage; 符号 test_ppl) : Qwen 系列 PPL 测试参数化 mm_device_do_normalize 开 / 关两档并固化 HF PPL 基线值, 验证默认开启不引入精度回退。
- docs/design/mm_processing.md (模块 设计文档; 类别 docs; 类型 documentation) : 补充 mm_device_do_normalize 的设计说明与优化指南, 并对齐 DarkLight1337 提出的文档更新要求。

关键符号: FusedInputNorm.from_model_config, FusedInputNorm.forward, FusedInputNorm.identity, _resolve_mm_device_do_normalize, MultiModalConfig.merge_mm_processor_kwargs, MultiModalConfig.compute_hash, get_processor_config, get_processor_cls_name_from_config, Qwen2VLForConditionalGeneration._process_image_input, Qwen2VLForConditionalGeneration._process_video_input, Qwen2_5_VLForConditionalGeneration._process_image_input, Qwen2_5_VLForConditionalGeneration._process_video_input, test_mm_device_do_normalize

关键源码片段

vllm/config/model.py

新增 mm_device_do_normalize InitVar 与 _resolve_mm_device_do_normalize 决策方法, 处理默认值推断、Rust 前端强制关闭与模型能力校验, 是整套开关逻辑的入口。

```
def _resolve_mm_device_do_normalize(
    self, mm_device_do_normalize: bool | None
) -> bool:
    if mm_device_do_normalize is None:
        # 未显式指定时: Rust 前端暂不支持设备端归一化, 强制关闭;
        # 否则跟随模型能力标记 supports_mm_device_do_normalize 决定默认值。
        if envs.VLLM_USE_RUST_FRONTEND:
            logger.debug(
                "VLLM_USE_RUST_FRONTEND is set. "
                "Rust frontend does not currently support mm_device_do_normalize, "
                "forcing mm_device_do_normalize = False."
            )
            mm_device_do_normalize = False
        else:
```

```

mm_device_do_normalize = self._model_info.supports_mm_device_do_normalize
logger.debug(
    "mm_device_do_normalize is %s by default.",
    "enabled" if mm_device_do_normalize else "disabled",
)
else:
    # 用户显式开启但 Rust 前端或模型不支持时，强制回退并告警，
    # 避免用户困惑为什么开关不生效。
    if mm_device_do_normalize and envs.VLLM_USE_RUST_FRONTEND:
        logger.warning(
            "VLLM_USE_RUST_FRONTEND is set. "
            "Rust frontend does not currently support mm_device_do_normalize, "
            "forcing mm_device_do_normalize = False."
        )
        mm_device_do_normalize = False

    if (
        mm_device_do_normalize
        and not self._model_info.supports_mm_device_do_normalize
    ):
        logger.warning(
            "Model does not support mm_device_do_normalize, "
            "forcing mm_device_do_normalize = False."
        )
        mm_device_do_normalize = False

    logger.debug(
        "mm_device_do_normalize is %s.",
        "enabled" if mm_device_do_normalize else "disabled",
    )

return mm_device_do_normalize

```

vllm/model_executor/models/qwen2_vl.py

Qwen2VLForConditionalGeneration 声明 `supports_mm_device_do_normalize = True`，在 `_mark_tower_model` 内构造 `input_norm`，并在图像 / 视频处理路径接入设备端归一化，是本特性第一个落地模型。

`__init__` 中，在 `_mark_tower_model` 上下文内构造：

```

with self._mark_tower_model(vllm_config, {"image", "video"}):
    self.visual = Qwen2VisionTransformer(
        config.vision_config,
        norm_eps=getattr(config, "rms_norm_eps", 1e-6),
        quant_config=quant_config,
        prefix=maybe_prefix(prefix, "visual"),
    )
    self.input_norm = FusedInputNorm.from_model_config(self.model_config)

def _process_image_input(

```

```

self, image_input: Qwen2VLImageInputs
) -> tuple[tuple[torch.Tensor, ...]:
    grid_thw = image_input["image_grid_thw"]
    assert grid_thw.ndim == 2

    if image_input["type"] == "image_embeds":
        image_embeds = image_input["image_embeds"].type(self.visual.dtype)
    else:
        pixel_values = image_input["pixel_values"]
        # 此前在 CPU 上完成的 rescale + normalize 被搬到此处,
        # 输入仍是 uint8 语义的原始像素, FusedInputNorm 在 GPU 上完成等价变换。
        pixel_values = self.input_norm(pixel_values, self.visual.dtype)

        if self.use_data_parallel:
            return run_dp_sharded_mrope_vision_model(
                self.visual, pixel_values, grid_thw.tolist(), rope_type="rope_3d"
            )
        else:
            image_embeds = self.visual(pixel_values, grid_thw=grid_thw)

    # 按每张图的 patch 数切分拼接后的 embedding
    merge_size = self.visual.spatial_merge_size
    sizes = (grid_thw.prod(-1) // merge_size // merge_size).tolist()
    return image_embeds.split(sizes)

```

视频路径 `_process_video_input` 采用相同模式, 对 `pixel_values_videos` 同样调用 `self.input_norm`。

评论区精华

review 中围绕参数读取来源、fail-fast 与静默回退、模块形态、Rust 前端兼容性展开了多轮交锋。DarkLight1337 质疑直接读 `processor` 类默认值不严谨, Isotr0py 建议改读 `preprocessor_config.json` 并指出自定义 `remote code` 难以导入; noooop 主张参数缺失时快速失败以避免用错 `mean/std` (Qwen2-VL 用 OPENAI_CLIP、Qwen3.5 用 IMAGENET_STANDARD)。DarkLight1337 还要求对被强制关闭的场景输出告警、构造器增加 `dtype` 参数, 并在 issue 中质疑端到端收益、指出部分收益来自引擎批处理 MM 输入。所有讨论最终落地: 三级参数解析回退链、保留 `assert`、`_resolve_mm_device_do_normalize` 中打 `warning`、`FusedInputNorm` 增加 `dtype` 与 `identity` 快速路径, 两位 reviewer 最终均 APPROVED。

- 归一化参数读取来源: `processor` 类 vs `preprocessor_config.json` vs HF 实例 (design): 最终实现为分三级解析: 优先 `get_processor_config` 读取配置文件, 字段缺失再回退到 HF `image_processor` 实例属性, 仍缺失则按 `do_rescale/do_normalize` 开关套用恒等默认值, 最后 `assert` 快速失败。
- 参数缺失时快速失败还是静默回退 (design): 保留 `assert`, 但同时扩展默认值兜底逻辑 (`do_rescale=False` 时 `factor` 置 1、`do_normalize=False` 时 `mean/std` 取恒等值), 只在参数完全无法解析时才失败。

- 模块形态与放置位置 (nn.Identity 与 _mark_tower_model) (design): 全部采纳, 最终演化为 FusedInputNorm nn.Module, 自动检测恒等场景跳过计算。
- Rust 前端兼容性与端到端收益验证 (design): Rust 前端暂不支持, VLLM_USE_RUST_FRONTEND 下强制 mm_device_do_normalize=False 并打 warning, 形成已知限制。
- 被强制关闭时需提示用户 (documentation): 采纳, _resolve_mm_device_do_normalize 对不支持场景输出 logger.warning。
- merge_mm_processor_kwargs 强制覆写 do_normalize/do_rescale 的边界情况 (correctness): merge_mm_processor_kwargs 开启时强制注入 do_normalize=False、do_rescale=False, 但推理期 inference_kwargs 仍有更高优先级可覆盖; from_model_config 按开关应用默认值兜底。
- 端到端收益质疑与批处理 MM 输入的方向 (question): noooooo 用 TTFT 数据回应; 批处理 MM 输入被确认为后续优化方向, 与本 PR 正交可叠加。

风险与影响

- 风险:
 1. 行为变更范围: mm_device_do_normalize 默认 True, 凡声明 supports_mm_device_do_normalize = True 的模型 (当前为 Qwen2-VL、Qwen2.5-VL) 默认启用设备端归一化, 数值等价但执行路径有差异, 其他架构默认关闭, 回归面可控。
 2. Rust 前端不兼容: VLLM_USE_RUST_FRONTEND 下强制关闭并告警, 是明确的已知限制; 双前端行为不一致可能造成用户困惑。
 3. 配置契约扩展: 新字段纳入 MultiModalConfig.compute_hash, 影响计算图 hash; EngineArgs、ModelConfig、MultiModalConfig 三处需同步维护。
 4. 参数解析失败风险: from_model_config 中的 assert 对个别缺失 image_mean/image_std 的模型或自定义 remote code processor 可能导致启动失败, 虽有「配置文件 → HF 实例 → 开关默认值」三级兜底但覆盖面未完全验证。
 5. 数值一致性依赖: 等价性依赖 F.batch_norm 的 eps=0 语义与 float32 计算顺序, 仅 Qwen 两代模型有 assert_close 与 PPL 双模式测试, 其他视觉模型缺测试覆盖。
 6. GPU 负载转移: 预处理从 CPU 移到 GPU, GPU 侧增加一次轻量 batch_norm (可与后续算子融合), CPU 释放约一半预处理耗时, 对 CPU 受限的视觉部署有利。- 影响: 对用户: Qwen2-VL / Qwen2.5-VL 用户默认获得更快的图像预处理与更低平均 TTFT (约 9% 改善), PPL 数值不变; 可通过 --mm-device-do-normalize 显式关闭。使用 Rust 前端的用户该开关自动失效。对系统: 缓解 CPU 预处理瓶颈, 将归一化计算转移至 GPU (开销可融合, 几乎为 0)。对团队: 多模态配置契约扩展 (EngineArgs → ModelConfig → MultiModalConfig + hash), 新增模型若想启用该优化需声明 supports_mm_device_do_normalize = True, 并保持 processor 配置可解析。- 风险标记: 默认开启改变视觉模型行为, Rust 前端暂不支持, 配置 hash 契约扩展, 归一化参数断言可能误伤, 缺少其他视觉模型覆盖

关联脉络

- PR #51089 [Feature] Parse request priority from HTTP header: 同属 Python/Rust 双前端行为差异脉络：本 PR 的 `mm_device_do_normalize` 在 `VLLM_USE_RUST_FRONTEND` 下被强制关闭，review 中 `Isotr0py` 也提到 Rust 前端已有 `normalize+to_tensor` 融合，后续可在 Rust 侧补齐该优化。
- PR #45254 [MM][CG] Support ViT full CUDA graph for Ernie-4.5-VL image inference: 同属多模态视觉编码器设备端性能优化方向：ViT CUDA Graph 与本 PR 的设备端归一化都在减少 CPU 预处理负担、加速视觉 prefill。
- PR #48413 [Bugfix][MM] Fix MiniCPM-V placeholder replacement and image processor loading on Transformers v5: 同属多模态预处理管线与 processor 加载逻辑维护，说明 MM 预处理链路的稳定与性能是持续演进的主题。