PR #24701 完整报告

sgl-project/sglang

[FIX][1/2] fix step3-vl/deepseek-ocr image processor error

合并时间：2026-05-22 18:39

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24701

执行摘要

- 一句话：修复 Step3-VL 图像处理器 GPU Tensor 兼容性 bug
- 推荐动作：建议所有使用 Step3-VL 的用户升级到此 PR，以解决 JPEG 图像请求失败问题。设计上对 Tensor 输入的适配模式（get_image_size、函数重载）值得学习，可用于类似多模态处理器兼容性修复。

功能与动机

关联 Issue #24699：Step3-VL 和 DeepSeek-OCR 在使用 JPEG 图像并启用 GPU 解码时失败，因为 GPU 解码返回 torch.Tensor，而处理器预期 PIL.Image，导致 TypeError: cannot unpack non-iterable builtin_function_or_method object。

实现拆解

1. 类型扩展：新增 Step3Image = Union[Image.Image, torch.Tensor] 类型别名，更新 ImageWithPatches 类型定义。
2. GPUToTensor 改造：在 forward() 中添加 torch.Tensor 处理分支，校验 CHW 格式、单通道复制为三通道、uint8 转 float32 归一化，并返回连续张量。同时优化 PIL 分支：转换后主动移至 CUDA 设备。
3. 统一尺寸获取：新增 ImagePatcher.get_image_size() 方法，区分 PIL 和 Tensor 获取宽高的方式。
4. 核心函数适配：square_pad 使用 get_image_size 并针对 Tensor 采用 torch.nn.functional.pad；patch_crop、resize 等函数同步适配 Tensor 路径，确保裁剪和缩放操作兼容。
5. 安全 fallback：在 Step3VLImageProcessor 属性中设置 gpu_image_decode = False，阻止自动 GPU 解码导致的问题。

关键文件：

- python/sglang/srt/multimodal/processors/step3_vl.py（模块 图像处理；类别 source；类型 core-logic；符号 forward, get_image_size, square_pad, patch_crop）：唯一修改文件，包含核心修复：GPUToTensor 支持 Tensor、get_image_size、square_pad/tensor 适配

关键符号：GPUToTensor.forward, ImagePatcher.get_image_size, ImagePatcher.square_pad, ImagePatcher.patch_crop

关键源码片段

python/sglang/srt/multimodal/processors/step3_vl.py

唯一修改文件，包含核心修复：GPUToTensor 支持 Tensor、get_image_size、square_pad/tensor 适配

```
import math
from typing import Union

import numpy as np
import torch
from PIL import Image
from torchvision import transforms

Step3Image = Union[Image.Image, torch.Tensor]

class GPUToTensor(torch.nn.Module):
    def forward(
        self, raw_image: Union[np.ndarray, Image.Image, torch.Tensor]
    ) -> torch.Tensor:
        # 处理 Tensor 输入：来自 GPU 解码的 JPEG 图像
        if isinstance(raw_image, torch.Tensor):
            image_tensor = raw_image
            if image_tensor.ndim != 3:
                raise TypeError(
                    f"Expected CHW image tensor, got shape {tuple(image_tensor.shape)}"
                )
            # 单通道复制为三通道
            if image_tensor.shape[0] == 1:
                image_tensor = image_tensor.repeat(3, 1, 1)
            elif image_tensor.shape[0] != 3:
                raise TypeError(
                    f"Expected CHW image tensor with 1 or 3 channels, got shape {tuple(image_
                    tensor.shape)}"
                )
            # uint8 转换为 float32 并归一化
            if image_tensor.dtype == torch.uint8:
                image_tensor = image_tensor.to(torch.float32).div(255)
            elif not image_tensor.is_floating_point():
                image_tensor = image_tensor.to(torch.float32)
            return image_tensor.contiguous()
        # 处理 PIL Image 输入
        if isinstance(raw_image, Image.Image):
            image_tensor = transforms.ToTensor()(raw_image)
            if torch.cuda.is_available():
                image_tensor = image_tensor.to(torch.device("cuda"))
```

```
            return image_tensor
        # 处理 numpy 输入（包括灰度图复制通道）
        if raw_image.ndim == 2:
            raw_image = raw_image[:, :, None].repeat(3, -1)
        if torch.cuda.is_available():
            device = torch.device("cuda")
        else:
            device = torch.device("cpu")
        image_tensor = torch.from_numpy(raw_image).to(device)
        image_tensor = torch.permute(image_tensor, (2, 0, 1)).contiguous()
        if image_tensor.dtype == torch.uint8:
            image_tensor = image_tensor.to(torch.float32).div(255)
        return image_tensor

class ImagePatcher:
    def get_image_size(self, img: Step3Image) -> tuple[int, int]:
        # PIL Image 直接访问 .size 属性（width, height）
        if isinstance(img, Image.Image):
            return img.size
        # Tensor 输入：从 shape 获取宽度和高度（CHW 格式）
        if isinstance(img, torch.Tensor):
            if img.ndim != 3:
                raise TypeError(
                    f"Expected CHW image tensor, got shape {tuple(img.shape)}"
                )
            return int(img.shape[-1]), int(img.shape[-2])
        raise TypeError(f"Unsupported image type: {type(img)}")
```

评论区精华

Review 中主要讨论包括：

- gemini-code-assist[bot] 建议在 gpu_image_decode 属性后添加空白行以符合 PEP 8（已修复）。
- yuan-luo 询问能否将 DeepSeek-OCR 修复合并到同一 PR，作者同意但最终此 PR 仅完成 Step3-VL 部分。
- zhsurpass 建议在 DeepSeek-OCR/OCR2 模型上验证准确率。
- PEP 8 空白行建议 (style): 作者在后续 commit 中修复，已添加空白行。
- 是否合并 DeepSeek-OCR 修复 (design): 决定分两步，此 PR 仅包含 Step3-VL 修复，DeepSeek-OCR 修复将在后续 PR 完成。
- DeepSeek-OCR 验证建议 (testing): 未明确答复，但 Step2 修复预计会处理。

风险与影响

- 风险： 1) Tensor 分支的通道和类型转换逻辑可能与预期不完全一致，需关注边缘情况（如非标准通道数）。 2) square_pad 对 Tensor 使用 zero padding，与原 PIL 的 mode 扩展行为略有差异。 3) 仅修复 Step3-VL，DeepSeek-OCR 修复未包含，用户若使用该模型仍

可能遇到类似问题。 4) 缺少统一的测试覆盖，回归风险依赖基准测试。

- 影响：直接影响使用 Step3-VL 模型且开启 GPU 图像解码的用户，使 JPEG 请求恢复正常。PNG 请求不受影响。修复后 MMMU 基准测试通过，吞吐正常。不影响其他多模态模型。
- 风险标记：类型兼容风险，仅部分修复，缺少测试覆盖

关联脉络

- 暂无明显关联 PR