PR #25403 完整报告

sgl-project/sglang

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

合并时间：2026-05-24 01:36

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25403

执行摘要

- 一句话：修复 DeepSeek-OCR GPU 解码 tensor 类型不兼容
- 推荐动作：值得精读。此 PR 展示了如何优雅地在一个关键路径上解决 PIL.Image 与 torch.Tensor 的兼容问题，通过类型受限的统一函数（Union[Image.Image, torch.Tensor]）避免散乱的条件判断，是处理多模态预处理中 GPU 解码类型不一致的范本。同时 review 中的性能建议（复用函数结果）反映了代码审阅的最佳实践。

功能与动机

Issue #24699 报告：DeepSeek-OCR 在 GPU 图像解码启用后，预处理代码调用 image.size 时收到 torch.Tensor（其 size 是方法而非属性），导致 TypeError: 'builtin_function_or_method' object is not subscriptable。本 PR 是修复的第二部分（第一部分 #24701 已修复 Step3-VL），专门解决 DeepSeek-OCR 的 tokenize_with_images 路径。

实现拆解

1. 类型别名定义：在 deepseek_ocr.py 顶部新增 DeepseekOCRImage = Union[Image.Image, torch.Tensor]，为后续函数签名提供统一类型。
2. 核心工具函数：新增四个静态函数实现 PIL/Tensor 双路径互斥分支：
   - get_image_size(img)：返回 (width, height)，PIL 直接取 .size，Tensor 从 CHW 形状提取。
   - resize_image(img, size)：PIL 使用 img.resize，Tensor 使用 TF.resize 并保持 BICUBIC 插值和 antialias。
   - crop_image(img, box)：PIL 使用 img.crop，Tensor 直接切片 img[:, upper:lower, left:right]。
   - pad_image(img, target_size, fill_color)：PIL 使用 ImageOps.pad，Tensor 手动缩放并居中填充，支持 uint8 和浮点 fill_color。
3. 改造 ImageTransform：在 __call__ 中增加 Tensor 分支 —— 若输入是 Tensor 且为 uint8 则转为 float32 并除以 255，然后使用已导入的 TF.normalize 替代原 torchvision.transforms.Normalize，消除冗余导入。
4. 改造 tokenize_with_images：将 image.size 替换为 get_image_size(image)，避免直接访问属性；同时将两次 get_image_size 调用合并为一次并复用结果（采纳 review 建议）。整个变更控制在单文件内，无新增测试，但 PR 提供手动性能对比。

关键文件：

- python/sglang/srt/configs/deepseek_ocr.py（模块 图像处理；类别 source；类型 core-logic；符号 get_image_size, resize_image, crop_image, pad_image）：唯一变更文件，新增四个统一图像处理函数并改造 ImageTransform 和 tokenize_with_images，实现对 GPU 解码 tensor 的支持，解决核心 bug。

关键符号：get_image_size, resize_image, crop_image, pad_image, ImageTransform.call

关键源码片段

python/sglang/srt/configs/deepseek_ocr.py

唯一变更文件，新增四个统一图像处理函数并改造 ImageTransform 和 tokenize_with_images，实现对 GPU 解码 tensor 的支持，解决核心 bug。

```
# 统一类型别名，标注输入可为 PIL.Image 或 torch.Tensor
DeepseekOCRImage = Union[Image.Image, torch.Tensor]


def get_image_size(img: DeepseekOCRImage) -> Tuple[int, int]:
    """返回 (宽,高) 元组，兼容 PIL.Image 和 torch.Tensor (CHW)。"""
    if isinstance(img, Image.Image):
        return img.size
    if isinstance(img, torch.Tensor):
        if img.ndim != 3:
            raise TypeError(f"Expected CHW image tensor, got shape {tuple(img.shape)}")
        # Tensor 形状为 (C, H, W)，宽为 W，高为 H
        return int(img.shape[-1]), int(img.shape[-2])
    raise TypeError(f"Unsupported image type: {type(img)}")


def resize_image(img: DeepseekOCRImage, size: Tuple[int, int]) -> DeepseekOCRImage:
    """缩放到 (宽,高)，PIL 和 Tensor 均使用 BICUBIC 插值。"""
    if isinstance(img, Image.Image):
        return img.resize(size, Image.BICUBIC)
    # Tensor 路径，TF.resize 期望 (H,W)
    return TF.resize(
        img, [size[1], size[0]],
        interpolation=InterpolationMode.BICUBIC, antialias=True
    ).contiguous()


def crop_image(img: DeepseekOCRImage, box: Tuple[int, int, int, int]) -> DeepseekOCRImage:
    """裁剪 box=(左,上,右,下)，Tensor 直接切片。"""
    if isinstance(img, Image.Image):
        return img.crop(box)
    left, upper, right, lower = box
    return img[:, upper:lower, left:right].contiguous()

def pad_image(img: DeepseekOCRImage, target_size: Tuple[int, int], fill_color: Tuple[int, int, int]
```

```
) -> DeepseekOCRImage:
    """居中填充到 target_size，替代 PIL 的 ImageOps.pad。"""
    if isinstance(img, Image.Image):
        return ImageOps.pad(img, target_size, color=fill_color)
    # Tensor 路径：手动缩放并填充
    _, h, w = img.shape
    target_w, target_h = target_size
    scale = min(target_w / w, target_h / h)
    new_w, new_h = int(w * scale), int(h * scale)
    resized = TF.resize(img, [new_h, new_w], interpolation=InterpolationMode.BICUBIC, antialias=
    True)
    # 构造填充底色，支持 uint8 或 float
    if img.dtype == torch.uint8:
        fill_tensor = torch.tensor(list(fill_color), device=img.device, dtype=torch.uint8).view(3, 1, 1)
    else:
        fill_tensor = torch.tensor([c / 255.0 for c in fill_color], device=img.device, dtype=img.
        dtype).view(3, 1, 1)
    result = fill_tensor.expand(3, target_h, target_w).clone()
    pad_left, pad_top = (target_w - new_w) // 2, (target_h - new_h) // 2
    result[:, pad_top:pad_top + new_h, pad_left:pad_left + new_w] = resized
    return result.contiguous()


class ImageTransform(object):
    # ... 初始化不变
    def __call__(self, img):
        if isinstance(img, torch.Tensor):
            x = img
            if x.dtype == torch.uint8:
                x = x.to(torch.float32).div(255) # 归一化到 [0,1]
            elif not x.is_floating_point():
                x = x.to(torch.float32)
            if self.normalize:
                x = TF.normalize(x, self.mean, self.std) # 复用顶层导入的 TF
            return x
        else:
            # PIL 路径沿用原 Transform（转为 Tensor 并 normalize）
            return self.transform(img)
```

评论区精华

来自 gemini-code-assist[bot] 的三条 review 评论均已采纳：

- PIL resize 插值对齐：建议PIL路径也使用Image.BICUBIC以匹配Tensor路径，已修改。
- normalize 复用已导入的 TF：建议使用 TF.normalize 替代局部 import torchvision.transforms，已修改。
- 缓存 get_image_size 结果：建议将两次连续调用合并为一次并复用，已修改。

- PIL 路径的插值方法应与 Tensor 一致使用 BICUBIC (correctness): 已在后续提交中修改为 Image.BICUBIC。
- 避免重复导入 torchvision.transforms，使用 TF.normalize (performance): 已采纳，改用 TF.normalize。
- 缓存 get_image_size 结果避免重复调用 (performance): 已修改，先调用一次得到 img_w, img_h 再使用。

风险与影响

- 风险:
  1. Tensor 形状假设：工具函数假设 Tensor 为 CHW 格式（img.ndim == 3），若上游传入 BHWC 或其它格式将抛出 TypeError，但当前所有调用来源均符合 CHW。
  2. GPU 内存占用：Tensor 路径下中间结果保留在 GPU 上，可能增加显存消耗，但相比模型权重很小。
  3. 缺少单元测试：当前无自动化测试覆盖 GPU 路径和边缘情况（如非 uint8 输入、异常形状），回归风险依赖 CI 的手动脚本。
  4. PIL 路径保持兼容：修改未影响原有 PIL 路径，通过 isinstance 分支保证。
- 影响:
  - 用户：DeepSeek-OCR 用户现在可以正常使用 JPEG 图像（GPU 解码）并获得 ~16% 延迟优化；PNG 等原生 PIL 路径无影响。
  - 系统：预处理阶段 GPU 利用率升高，但整体吞吐提升；无配置或 API 变更。
  - 团队：代码复用性提升，未来其他模型遇到类似 tensor/PIL 混合问题时可直接复用这些工具函数。
  - 风险标记：缺少测试覆盖，GPU 路径边界条件，显存占用增加

关联脉络

- PR #24701 [FIX][1/2] fix step3-vl image processor error: 同一 issue #24699 的第一部分修复，针对 Step3-VL 模型，与本 PR 共享相同的根因和设计模式。
- PR #24699 [Bug] Step3-VL and DeepSeek-OCR2 fail on JPEG image requests when GPU image decoding returns tensors: 关联 Issue，提供了完整的 bug 描述和复现步骤，是此 PR 的直接触发因素。