PR #33859 完整报告

sgl-project/sglang

[diffusion] fix: crop GLM-Image output to requested size

合并时间：2026-08-25 16:50

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33859

执行摘要

- 一句话：GLM-Image 输出按请求尺寸中心裁剪，D32 对齐后复原
- 推荐动作：值得精读。核心价值在于两点：一是 dataclasses.replace() 与 init=False 字段的坑及其修复方式；二是以专用 DecodingStage 子类承载模型特有后处理的模式，对 diffusion 子系统后续扩展有参考意义。建议 diffusion 管线与 pipeline 架构相关工程师阅读 glm_image.py 和 diffusion_generator.py 的改动。

功能与动机

PR body 明确说明：GLM-Image requires its generation resolution to be divisible by 32，用户请求 1280x720 时实际按 1280x736 生成，而"the decoded output currently keeps the aligned dimensions instead of the dimensions requested by the user"，即最终交付尺寸与请求不一致。该 PR 的目标是把输出恢复为用户请求的尺寸，而不是让用户接受对齐后画布。

实现拆解

本 PR 的变更入口是 GLM-Image 的采样参数与解码管线，整体按 5 步完成：

1. 数据契约扩展：python/sglang/multimodal_gen/configs/sample/glmimage.py 的 GlmImageSamplingParams 新增 requested_width / requested_height 两个 init=False 字段，在 _adjust() 中对齐到 D32 网格之前先记录用户请求尺寸。
2. AR 阶段保存请求尺寸：glm_image.py 中 GlmImageAR.forward() 在宽度 / 高度对齐前把请求尺寸写入 batch（requested_width / requested_height），并调整图编辑路径——先 load_image 原图、记录原图尺寸，再统一 resize_glm_image_to_alignment()，保证隐式继承编辑图片尺寸的场景也能拿到真实请求画布。
3. 新增解码裁剪阶段：glm_image.py 新增 center_crop_glm_image_output() 工具函数与 GlmImageDecodingStage（继承 DecodingStage），在 D32 画布上完成 VAE 解码后对最终帧与 trajectory_decoded 做中心裁剪；python/sglang/multimodal_gen/runtime/pipelines/glm_image.py 将 add_standard_decoding_stage() 替换为 GlmImageDecodingStage 的 stage factory。
4. 入口层克隆保护：diffusion_generator.py 提取 _replace_sampling_params_for_prompt()，修复 dataclasses.replace() 会重置 init=False 字段的问题，把 requested_width / requested_height 带回每个 prompt 的克隆体，并保留 _explicit_fields 供

InputValidationStage 识别显式传参。

5. 响应与 CI 配套：openai/image_api.py 的 _get_response_resize() 回退逻辑优先读取 requested_width / requested_height，上报真实输出尺寸；perf_baselines_npu.json 中的阶段名从 DecodingStage 改为 GlmImageDecodingStage 以匹配 Ascend CI 预期；同时新增 / 更新 test_glm_image_ar.py、test_sampling_params.py、test_openai_image_api.py 三个单元测试文件。

关键文件：

- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/glm_image.py（模块 图像管线；类别 source；类型 data-contract；符号 center_crop_glm_image_output, GlmImageDecodingStage, forward）：核心实现文件：新增 center_crop_glm_image_output 与 GlmImageDecodingStage，并改造 GlmImageAR.forward 在 D32 对齐前保存请求尺寸、调整图编辑路径的缩放时机。
- python/sglang/multimodal_gen/runtime/entrypoints/diffusion_generator.py（模块 生成入口；类别 source；类型 core-logic；符号 _replace_sampling_params_for_prompt）：修复离线 /CLI 入口 per-prompt 克隆时 dataclasses.replace() 丢弃 init=False 字段的问题，提取 _replace_sampling_params_for_prompt 统一处理。
- python/sglang/multimodal_gen/configs/sample/glmimage.py（模块 采样参数；类别 source；类型 dependency-wiring；符号 GlmImageSamplingParams, _adjust, requested_width, requested_height）：数据契约变更源头：新增 requested_width / requested_height 字段并在 _adjust() 中对齐前保存用户请求尺寸。
- python/sglang/multimodal_gen/runtime/pipelines/glm_image.py（模块 管线装配；类别 source；类型 dependency-wiring；符号 GlmImagePipeline.create_pipeline_stages, GlmImageDecodingStage）：管线装配处把标准 add_standard_decoding_stage() 替换为 GlmImageDecodingStage，是解码裁剪生效的接线点。
- python/sglang/multimodal_gen/runtime/entrypoints/openai/image_api.py（模块 图像接口；类别 source；类型 entrypoint；符号 _get_response_resize）：OpenAI 兼容响应在无法检查输出图片时优先读取 requested 尺寸，保证返回真实最终输出尺寸。
- python/sglang/multimodal_gen/test/unit/test_glm_image_ar.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_forward_preserves_implicit_edit_image_size, test_center_crop_restores_requested_size, test_decoding_stage_crops_outputs_and_trajectory）：核心测试文件：覆盖请求尺寸保留、中心裁剪、解码阶段对输出与轨迹的裁剪、隐式编辑图尺寸等场景。
- python/sglang/multimodal_gen/test/unit/test_sampling_params.py（模块 参数测试；类别 test；类型 test-coverage；符号 test_per_prompt_clone_preserves_glm_image_crop_size）：验证 per-prompt 克隆在 dataclasses.replace 后仍保留 GLM-Image 裁剪尺寸。
- python/sglang/multimodal_gen/test/unit/test_openai_image_api.py（模块 接口测试；类别 test；类型 test-coverage；符号 test_response_resize_prefers_requested_size_over_generation_canvas）：验证 OpenAI 响应尺寸回退优先使用请求尺寸而非生成画布。
- python/sglang/multimodal_gen/test/server/ascend/perf_baselines_npu.json（模块 NPU 基线；类别 config；类型 configuration）：阶段名从 DecodingStage 改为 GlmImageDecodingStage，匹配 Ascend CI 对阶段名的预期，否则性能基准测试失败。

关键符号：center_crop_glm_image_output, GlmImageDecodingStage.forward, GlmImageAR.forward, _replace_sampling_params_for_prompt, GlmImageSamplingParams._adjust, _get_response_resize

关键源码片段

python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/glm_image.py

核心实现文件：新增 center_crop_glm_image_output 与 GlmImageDecodingStage，并改造 GlmImageAR.forward 在 D32 对齐前保存请求尺寸、调整图编辑路径的缩放时机。

```
def center_crop_glm_image_output(
    frames: torch.Tensor,
    target_width: int | None,
    target_height: int | None,
) -> torch.Tensor:
    """Center-crop decoded GLM-Image pixels back to the requested canvas."""
    # 任一目标尺寸缺失（例如未走 GLM-Image 路径）时直接返回原帧
    if None in (target_width, target_height):
        return frames

    decoded_height, decoded_width = frames.shape[-2:]
    # 请求尺寸大于解码画布属于非法状态，提前报错避免静默越界
    if target_width > decoded_width or target_height > decoded_height:
        raise ValueError(
            "Cannot crop GLM-Image output to a canvas larger than the decoded "
            f"image: requested {target_width}x{target_height}, decoded "
            f"{decoded_width}x{decoded_height}"
        )
    # 尺寸恰好一致时无需裁剪，避免无谓拷贝
    if (target_width, target_height) == (decoded_width, decoded_height):
        return frames

    # 按左右、上下各半的方式居中取窗；切片本身不连续，
    # 用 .contiguous() 保证后续消费方拿到内存连续的张量
    left = (decoded_width - target_width) // 2
    top = (decoded_height - target_height) // 2
    return frames[
        ..., top : top + target_height, left : left + target_width
    ].contiguous()

# 新增的解码阶段：在 D32 对齐画布上解码，再把结果恢复为用户请求尺寸。
# 解码输出帧与 trajectory_decoded 都应用同一中心裁剪，保证轨迹与最终结果一致
# （对应 test_decoding_stage_crops_outputs_and_trajectory 的断言）。
class GlmImageDecodingStage(DecodingStage):
    """Decode on the D32 canvas, then restore the user-requested dimensions."""

    @torch.no_grad()
    def forward(self, batch: Req, server_args: ServerArgs):
```

```
        # 先复用父类 DecodingStage 的 VAE 解码逻辑（在 batch.width/height
        # 即对齐后画布上执行），再对返回的帧与 trajectory 统一中心裁剪
        decoded = super().forward(batch, server_args)

        cropped_frames = center_crop_glm_image_output(
            decoded, batch.requested_width, batch.requested_height
        )
        # 轨迹与最终输出使用同一裁剪入口，此处仅示意调用方式，
        # 输出结构的具体组装顺序以仓库最终实现为准
        return cropped_frames
```

python/sglang/multimodal_gen/runtime/entrypoints/diffusion_generator.py

修复离线 /CLI 入口 per-prompt 克隆时 dataclasses.replace() 丢弃 init=False 字段的问题，提取 _replace_sampling_params_for_prompt 统一处理。

```
def _replace_sampling_params_for_prompt(
    sampling_params_orig: SamplingParams,
    prompt: str,
    output_file_name: str | None,
    image_path: str | list[str] | None,
) -> SamplingParams:
    """Clone per-prompt parameters without losing model-internal state."""
    # dataclasses.replace() 会重置 init=False 字段；这里先替换显式参数
    sampling_params = dataclasses.replace(
        sampling_params_orig,
        prompt=prompt,
        output_file_name=output_file_name,
        image_path=image_path,
    )

    # 把 GLM-Image 的请求画布尺寸带回克隆体，否则解码阶段拿不到
    # requested_width/requested_height，中心裁剪会直接跳过
    for field_name in ("requested_width", "requested_height"):
        if hasattr(sampling_params_orig, field_name):
            setattr(
                sampling_params,
                field_name,
                getattr(sampling_params_orig, field_name),
            )

    # dataclasses.replace() 同样会丢弃非字段属性；恢复 _explicit_fields
    # 让 InputValidationStage 能识别用户显式传入的 width/height 等参数
    sampling_params._explicit_fields = getattr(
        sampling_params_orig, "_explicit_fields", set()
    ) | {"prompt", "output_file_name", "image_path"}
    return sampling_params
```

python/sglang/multimodal_gen/configs/sample/glmimage.py

数据契约变更源头：新增 requested_width / requested_height 字段并在 _adjust() 中对齐前保存用户请求尺寸。

```
@dataclass
class GlmImageSamplingParams(SamplingParams):
    negative_prompt = ""

    num_frames: int = 1
    guidance_scale: float = 1.5
    num_inference_steps: int = 30

    # 在对齐到 D32 生成网格之前，先保存用户请求的画布尺寸；
    # init=False 让这些字段不参与构造参数，但保留在 dataclass 实例上，
    # 供解码阶段做中心裁剪使用
    requested_width: int | None = field(default=None, init=False)
    requested_height: int | None = field(default=None, init=False)

    def _adjust(self, server_args):
        requested_width = self.width
        requested_height = self.height
        if self.width is not None and self.height is not None:
            # 首次 adjust 时记录原始请求尺寸，之后重复 adjust 不覆盖
            if self.requested_width is None:
                self.requested_width = requested_width
            if self.requested_height is None:
                self.requested_height = requested_height
            self.width, self.height = align_glm_image_resolution(
                self.width, self.height
            )
            if (self.width, self.height) != (
                requested_width,
                requested_height,
            ):
                logger.warning(
                    "GLM-Image requires dimensions divisible by %s; adjusted "
                    "requested resolution from %sx%s to %sx%s",
                    GLM_IMAGE_RESOLUTION_ALIGNMENT,
                    requested_width,
                    requested_height,
                    self.width,
                    self.height,
                )
        super()._adjust(server_args)
```

评论区精华

评审中主要有 3 处代码风格 / 简化建议和 1 处 CI 基线问题，全部已解决：

- ping1jing2 在 image_api.py 建议把回退逻辑简化为 width = sampling_params.requested_width or sampling_params.width，最终实现采纳。
- ping1jing2 批评 _replace_sampling_params_for_prompt 签名中使用了 *（关键字专属参数分隔符）："i personly think this is not a good code style, we are supposed to avoid using * here"，最终改为显式位置参数签名。
- ping1jing2 建议 center_crop_glm_image_output 的 None 判断写成 if None in (target_width, target_height)，已采纳。
- OrangeRedeng 在 issue 评论中指出 Ascend CI 仍期望名为 DecodingStage 的阶段，要求把 perf_baselines_npu.json 改为 GlmImageDecodingStage，已通过后续提交（a92e1f11）修复。
- image_api 回退逻辑简化建议 (style): 已采纳，最终代码按建议实现。
- 函数签名中避免使用 * (style): 已按建议去除，改为显式位置参数签名。
- None 判断表达风格 (style): 已采纳，最终代码使用该写法。
- Ascend CI 期望 DecodingStage 阶段名 (testing): 已通过提交 a92e1f11 修复，并在后续 merge 中解决冲突。

风险与影响

- 风险：
  1. 裁剪尺寸冲突：center_crop_glm_image_output() 在请求尺寸大于解码画布时抛 ValueError；若某请求路径未设置 requested_width/height（为 None）则跳过裁剪，但同 batch 内部分请求有、部分没有时可能出现尺寸不一致的解码输出。
  2. 图编辑行为变化：forward() 从"先 resize 再 load"改为"先 load 原图记录尺寸、再 resize"，对未对齐的输入源图，隐式尺寸来自输入图；若用户同时显式传入更大的 width/height，裁剪阶段可能触发上述 ValueError。
  3. dataclasses 克隆坑：dataclasses.replace() 丢弃 init=False 字段的问题只在 diffusion_generator.py 的离线 /CLI 入口被兜住，HTTP generations 等其他入口若直接 clone sampling_params，仍可能丢失请求尺寸。
  4. CI 基线耦合：perf_baselines_npu.json 依赖 GlmImageDecodingStage 阶段名，后续若恢复标准 DecodingStage 或再次改名，需同步更新，否则 Ascend CI 会失败。
  5. 轨迹裁剪契约：trajectory_decoded 的裁剪依赖解码阶段返回结构，若未来解码输出格式调整，裁剪逻辑需同步适配。 - 影响：用户侧：GLM-Image 文生图与图生图的最终输出尺寸与请求一致（如 1280x720 不再被 1280x736 替代），OpenAI 兼容响应返回真实最终尺寸。系统侧：GLM-Image 解码管线新增专用 GlmImageDecodingStage 替换标准阶段，仅影响 GlmImagePipeline，不影响其他 diffusion 模型；离线 /CLI 的 per-prompt 参数克隆逻辑被封装复用。团队侧：NPU Ascend CI 性能基线随之更新，后续新增 diffusion 模型可参考"对齐画布解码 + 中心裁剪"的模式。 - 风险标记：数据契约变更（requested_width/requested_height），解码管线核心路径替换，NPU CI 基线依赖阶段名，dataclasses 克隆易丢失隐式字段

关联脉络

- PR #32999 [diffusion] GLM-Image D32 resolution alignment（PR body 提及的前置 PR）: PR body 明确说明本分支是 #32999 的后续，rebase 后只保留输出裁剪改动，二者属于同一功能线。
- PR #36035 [Diffusion] Add component-scoped quantization overrides: 同属 diffusion/multimodal_gen 运行时（loader、server_args、entrypoint）的配套演进，体现该子系统持续扩展 per-prompt/ 组件级配置的改动模式。