PR #28958 完整报告

sgl-project/sglang

Support nvidia/LocateAnything-3B

合并时间：2026-06-30 00:16

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28958

执行摘要

- 一句话：支持 nvidia/LocateAnything-3B 视觉定位模型
- 推荐动作：值得精读。本 PR 展示了在 SGLang 框架中集成新多模态模型的标准实践：利用已有组件（MoonViT、Qwen2）组装、注册 config/model_type、编写专属图像处理器，并适配多模态数据流水线。其 review 讨论涉及多图像拆分与 custom logit processor 集成的设计权衡，对后来者具有参考价值。特别是 LocateAnythingBoxGrammarLogitProcessor 的约束解码状态机实现，可作为自定义 logit 处理的参考范例。

功能与动机

本 PR 旨在响应 Feature Request #27674，为 SGLang 添加 nvidia/LocateAnything-3B 的原生支持。该模型是一个视觉接地 / 检测模型，覆盖目标检测、短语接地、场景文本检测、GUI 接地和指点等任务。PR body 指出模型架构复用 MoonViT、InternVL 投影器和 Qwen2 骨干，且无需 transformers fallback ，直接通过 SGLang 的 model_type 注册实现加载。

实现拆解

1. 配置与注册：在 python/sglang/srt/configs/locate_anything.py 中定义 LocateAnythingConfig，组合 MoonViTConfig（视觉）和 Qwen2Config（文本），并存储所有特殊 token ID（box_start/end、coord_start/end、none_token_id 等）。通过 configs/__init__.py、model_config.py 和 utils/hf_transformers/common.py 将其注册到全局架构映射中，使模型加载时能直接路由到 LocateAnythingForConditionalGeneration。
2. 核心模型实现：在 python/sglang/srt/models/locate_anything.py 中实现 LocateAnythingForConditionalGeneration，复用 MoonVitPretrainedModel（视觉 tower）和 Qwen2ForCausalLM（语言模型），并实现专属的 LocateAnythingMultiModalProjector（mlp1 投影器：先 patch merge 再 LayerNorm，与 Kimi-VL 的顺序不同）。get_image_feature 方法合并多图像特征并传入投影器；load_weights 执行权重前缀映射与 tied embedding 处理。额外提供可选的 LocateAnythingBoxGrammarLogitProcessor，在 <box>...</box> 块内约束解码为合法的 none / 2-d point / 4-d bbox 形式。
3. 多模态图像处理：在 python/sglang/srt/multimodal/processors/locate_anything.py 中编写 LocateAnythingImageProcessor，继承基础多模态处理器，通过覆盖 process_mm_data_async 将 prompt 中的 <image-N> 占位符替换为图像特征嵌入 token ，并复用 SGLang 标准的多模态数据加载流水线。

4. 多图像拆分适配：在 python/sglang/srt/managers/mm_utils.py 的 get_new_expanded_mm_items 中添加对 image_grid_hws 的识别——MoonViT 模型使用的网格格式为 [h, w] 而非标准的 [t, h, w]。新增 _is_rank2_grid 辅助函数，保护退化（一维）网格的回退，确保多图像请求能正确拆分为每图像单独项，以维持 RadixAttention 缓存粒度和 chunked-prefill 快速路径。
5. 测试与文档：新增 test/registered/unit/configs/test_locate_anything_config.py（配置单元测试）、test/registered/unit/models/test_locate_anything.py（投影仪和 box 语法处理器状态机测试，含前向形状接线测试）、test/registered/unit/managers/test_mm_utils_split.py（拆分器回归测试）。更新 docs_new/docs/supported-models/multimodal_language_models.mdx，为 LocateAnything-3B 增加一行支持记录，并注明需 --enable-custom-logit-processor 方可使用约束解码。

关键文件：

- python/sglang/srt/models/locate_anything.py（模块 模型层；类别 source；类型 core-logic；符号 LocateAnythingMultiModalProjector, forward, LocateAnythingForConditionalGeneration, get_image_feature）：核心模型实现，包含视觉编码器、多模态投影仪、语言骨干网络、盒子语法处理器等所有关键组件，是本 PR 的主要变更文件。
- python/sglang/srt/multimodal/processors/locate_anything.py（模块 图像处理；类别 source；类型 core-logic；符号 LocateAnythingImageProcessor, init, process_mm_data_async）：多模态图像处理器，负责将图像数据转换为模型需要的多模态输入格式，包括标记替换和特征提取流程。
- python/sglang/srt/configs/locate_anything.py（模块 配置定义；类别 source；类型 core-logic；符号 LocateAnythingConfig, init）：模型配置类，定义了视觉、文本子配置和所有特殊标记 ID，是模型加载和初始化的入口。
- python/sglang/srt/managers/mm_utils.py（模块 多模态工具；类别 source；类型 core-logic；符号 get_new_expanded_mm_items, _is_rank2_grid）：扩展了图像分片函数 get_new_expanded_mm_items 以支持 image_grid_hws 格式，确保多图像请求的正确拆分。
- test/registered/unit/models/test_locate_anything.py（模块 测试用例；类别 test；类型 test-coverage；符号 TestLocateAnythingProjector, test_merged_size_and_output_shape, test_forward_flattens_merged_patches, test_forward_handles_noncontiguous_input）：单元测试覆盖投影仪形状、布局语法处理器的约束解码状态机和图像特征提取路径，确保核心逻辑正确性。
- test/registered/unit/managers/test_mm_utils_split.py（模块 拆分测试；类别 test；类型 test-coverage；符号 TestGetNewExpandedMMItems, test_image_grid_hws_splits_per_image, test_image_grid_hws_tensor_splits_per_image, test_image_grid_thw_still_splits）：新增拆分器单元测试，覆盖 image_grid_hws 分割、边界案例和退化网格，确保多图像路径可靠。

关键符号：LocateAnythingMultiModalProjector.init, LocateAnythingMultiModalProjector.forward, LocateAnythingForConditionalGeneration.init, LocateAnythingForConditionalGeneration.get_image_feature,

LocateAnythingForConditionalGeneration.pad_input_ids, LocateAnythingForConditionalGeneration.load_weights, LocateAnythingBoxGrammarLogitProcessor.init, LocateAnythingBoxGrammarLogitProcessor.call, LocateAnythingConfig.init, LocateAnythingImageProcessor.init, LocateAnythingImageProcessor.process_mm_data_async, get_new_expanded_mm_items, _is_rank2_grid

关键源码片段

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

多模态图像处理器，负责将图像数据转换为模型需要的多模态输入格式，包括标记替换和特征提取流程。

```
# SPDX-License-Identifier: Apache-2.0
import re
from typing import Dict, List, Union

from sglang.srt.managers.schedule_batch import MultimodalProcessorOutput
from sglang.srt.models.locate_anything import LocateAnythingForConditionalGeneration
from sglang.srt.multimodal.processors.base_processor import (
    BaseMultimodalProcessor as SGLangBaseProcessor,
)
from sglang.srt.multimodal.processors.base_processor import MultimodalSpecialTokens

class LocateAnythingImageProcessor(SGLangBaseProcessor):
    models = [LocateAnythingForConditionalGeneration]
    # The LocateAnything HF processor is remote-code and does not support tensor inputs.
    gpu_image_decode = False

    def __init__(self, hf_config, server_args, _processor, *args, **kwargs):
        super().__init__(hf_config, server_args, _processor, *args, **kwargs)
        # The model's chat template emits numbered ``<image-N>`` placeholders.
        # The HF LocateAnythingProcessor expands each into
        # ``<img>`` + N×``<IMG_CONTEXT>`` + ``</img>`` and only the
        # ``<IMG_CONTEXT>`` (id 151665) run carries vision embeddings, so the
        # offset/embedding token id is ``image_token_index`` while the prompt-level
        # placeholder we split on is ``<image-N>``.
        self.mm_tokens = MultimodalSpecialTokens(
            image_token_id=hf_config.image_token_index,
            image_token_regex=re.compile(r"<image-\d+>"),
        ).build(_processor)

    async def process_mm_data_async(self, image_data, input_text, request_obj, *args,
    **kwargs):
        base_output = await self.load_mm_data(
            prompt=input_text,
            image_data=image_data,
            multimodal_tokens=self.mm_tokens,
```

```
        )
        mm_items, input_ids, _ = self.process_and_combine_mm_data(base_output, self.mm_
        tokens)
        return MultimodalProcessorOutput(
            input_ids=input_ids.tolist(),
            mm_items=mm_items,
            im_token_id=self.mm_tokens.image_token_id,
        )
```

python/sglang/srt/configs/locate_anything.py

模型配置类，定义了视觉、文本子配置和所有特殊标记 ID，是模型加载和初始化的入口。

```
# SPDX-License-Identifier: Apache-2.0
# Adapted from https://huggingface.co/nvidia/LocateAnything-3B/blob/main/configuration_
locateanything.py

from typing import Optional, Union
from transformers.configuration_utils import PretrainedConfig
from transformers.models.qwen2 import Qwen2Config
from sglang.srt.configs.kimi_vl_moonvit import MoonViTConfig

class LocateAnythingConfig(PretrainedConfig):
    model_type = "locateanything"

    def __init__(
        self,
        vision_config: Optional[Union[dict, MoonViTConfig]] = None,
        text_config: Optional[Union[dict, Qwen2Config]] = None,
        image_token_index: int = 151665,
        box_start_token_id: int = 151668,
        box_end_token_id: int = 151669,
        ref_start_token_id: int = 151672,
        ref_end_token_id: int = 151673,
        coord_start_token_id: int = 151677,
        coord_end_token_id: int = 152677,
        none_token_id: int = 4064,
        mlp_connector_layers: int = 2,
        **kwargs,
    ):
        # Build sub-configs from dict or use defaults
        if vision_config is None:
            vision_config = MoonViTConfig()
        elif isinstance(vision_config, dict):
            vision_config = MoonViTConfig(**vision_config)
        self.vision_config = vision_config

        if text_config is None:
            text_config = Qwen2Config()
        elif isinstance(text_config, dict):
```

```
            text_config = Qwen2Config(**text_config)
        self.text_config = text_config

        # Store all special token IDs used by the optional box grammar processor
        self.image_token_index = image_token_index
        self.box_start_token_id = box_start_token_id
        self.box_end_token_id = box_end_token_id
        self.ref_start_token_id = ref_start_token_id
        self.ref_end_token_id = ref_end_token_id
        self.coord_start_token_id = coord_start_token_id
        self.coord_end_token_id = coord_end_token_id
        self.none_token_id = none_token_id
        self.mlp_connector_layers = mlp_connector_layers

        super().__init__(**kwargs)
```

评论区精华

1. 多图像拆分器对 image_grid_hws 的缺失：JustinTong 指出 get_new_expanded_mm_items 只查找 image_grid_thw，而 MoonViT 模型使用 image_grid_hws，导致多图像请求不会被拆分，影响 RadixAttention 缓存粒度。作者随后扩展了拆分器支持两种格式，并增加了回归测试。
2. Box 语法处理器无法自动接入：JustinTong 发现 LocateAnythingBoxGrammarLogitProcessor 已定义并测试，但未被服务端自动关联，客户端需要手动通过 build_sampling_params 辅助函数配置 token ID 和服务端 flag。作者随后补全了该辅助函数和文档。
3. build_sampling_params 示例导致 500 错误：JustinTong 指出 docstring 示例将 custom_logit_processor 错误地放入 sampling_params，而它应是 GenerateReqInput 的顶级字段，会导致 msgspec 报错。作者修正了 docstring 并澄清了 API 契约。
4. 约束解码仅扫描 output_ids：作者确认 box 语法处理器只扫描 output_ids 而非 prompt+output，以避免引入 prompt 中未闭合 box 带来的复杂性，并在注释中明确这一设计决策。
5. 缺少端到端 CI 测试：JustinTong 指出模型缺少自动化的端到端推理测试，仅有人工验证。作者添加了前向形状接线测试（stub vision tower），但仍未覆盖完整 GPU 推理路径。

- 多图像拆分器对 image_grid_hws 的支持 (design): 扩展拆分器同时支持 image_grid_hws 和 image_grid_thw，并添加回归测试（test_mm_utils_split.py）。
- Box 语法处理器无法自动接入服务路径 (design): 添加 build_sampling_params 辅助函数，并在文档中注明需设置 --enable-custom-logit-processor。
- build_sampling_params 示例导致 500 错误 (correctness): 修正 docstring，明确 custom_params 放入 sampling_params，custom_logit_processor 作为顶级字段传递。
- 约束解码只扫描 output_ids 的取舍 (design): 保留 output_ids-only 扫描，在注释中明确这一设计决策和风险。
- 缺少拆分器回归测试 (testing): 作者添加了 test_mm_utils_split.py，覆盖 HWS/THW/ 退化网格等场景。

风险与影响

- 风险：
  1. 多图像拆分正确性：对 image_grid_hws 的支持是新加入的逻辑，虽然已添加单元测试，但实际多图像请求在生产中可能遇到非预期的网格格式（如更高维张量），若不匹配会导致拆分失败或错误。修复中已包含一维网格退化保护，但未覆盖更高维情况。
  2. 约束解码的 Prompt 忽略：LocateAnythingBoxGrammarLogitProcessor 只扫描 output_ids，若 prompt 中包含未闭合的 <box>（如 malformed few-shot），则约束不会生效，模型可能生成非法标记。作者已将此风险记录为已知限制。
  3. 缺少端到端 GPU 测试：仅有 CPU 单元测试，暂未将完整的 GPU 推理纳入 CI，若后续对 MoonViT 或 Qwen2 模型有修改可能造成回归而未被检测到。
  4. 投影器的 Reshape 使用：LocateAnythingMultiModalProjector.forward 使用 reshape（而非 view）处理非连续张量，虽然测试覆盖了非连续场景，但若未来输入布局变化可能导致尺寸错误。测试已验证基本非连续场景。 - 影响：用户侧：现在可以通过 python3 -m sglang.launch_server --model-path nvidia/LocateAnything-3B --trust-remote-code 加载该视觉接地模型，并通过设置 skip_special_tokens=False 获取结构化输出。高级用户可启用 --enable-custom-logit-processor 并使用 build_sampling_params 辅助函数来约束 box 内的解码格式。系统侧：新增约 1.1k 行代码，主要集中在模型定义、配置和处理器上；对 mm_utils.py 的修改（约 36 行）采用了向后兼容的方式（key 不存在时 fallback），未影响现有行为。维护团队：需要关注 MoonViT 和 Qwen2 模型的后续变更是否会破坏 LocateAnything 的复用部分；新增的拆分器测试可以捕获多图像拆分相关回归。
- 风险标记：多图像拆分依赖新逻辑，自定义处理器需单独启用，缺少端到端 CI 覆盖，投影仪使用了 reshape 而非 view

关联脉络

- PR #27674 [Feature] Support for nvidia/LocateAnything-3B: 关联的 Feature Issue，本 PR 即应此请求而实现。