PR #23811 完整报告

sgl-project/sglang

[Feature] Xiaomi MiMo-V2.5 day0 support

合并时间：2026-05-01 00:02

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23811

执行摘要

- 一句话：为 Xiaomi MiMo-V2.5 添加多模态与 EAGLE 推测解码支持
- 推荐动作：此 PR 是小米模型的完整集成，值得详细审阅，特别是 fused-qkv 装载模式、多模态处理器设计以及 VisionAttention 增强。合并后应关注 Gemma4 和其他多模态模型的回归测试。

功能与动机

XiaomiMiMo/MiMo-V2.5 是一个新的多模态大模型，支持图像、视频、音频理解。需要在 SGLang 中提供 day-0 支持以便用户部署推理。PR body 指出这包括注册模型架构、多模态处理器、推测解码集成等。

实现拆解

实现分以下主要步骤：

1. 模型架构实现：在 python/sglang/srt/models/mimo_v2.py 中新增 MiMoV2ForCausalLM 类，支持混合 SWA/ 全注意力、MoE、fused-qkv 装载。同时修改 mimo_v2_nextn.py 支持 MTP 推测解码。
2. 视觉模块：新增 python/sglang/srt/models/mimo_vl.py，包含 MiMoVLVisionConfig、MiMoVisionPatchEmbed、MiMoVisionBlock、MiMoVisionTransformer，复用上游 VisionAttention 并扩展 sink 和 window 支持。
3. 音频模块：新增 python/sglang/srt/models/mimo_audio.py，包含 MiMoAudioEncoder、MiMoAudioTokenizer 等，独立于模型主路径。
4. 多模态处理器：新增 python/sglang/srt/multimodal/processors/mimo_v2.py，处理图像、视频、音频的下载 / 解码 / 预处理，输出 MultimodalProcessorOutput。
5. 配置与集成：在 model_config.py 添加 MiMoV2 架构检测和 fused-qkv 期望 TP 大小推导；在 server_args.py 强制 attention TP 对齐；修改 mm_utils.py 支持列表特征的共享内存包装 / 解包；修改 VisionAttention 以支持 window_size 和 sink。
6. 测试：新增 test/registered/8-gpu-models/test_mimo_models.py 中的 GSM8K 和 MMMU 测试。

关键文件：

- python/sglang/srt/models/mimo_v2.py（模块 模型层；类别 source；类型 data-contract；符号 load_mimo_v2_qkv_proj_weight, MiMoV2ForCausalLM, pad_input_ids, get_image_feature）：核心模型文件，包含 MiMoV2ForCausalLM 主模型和 fused-qkv 装载逻辑
- python/sglang/srt/models/mimo_vl.py（模块 视觉模块；类别 source；类型 data-contract；符号 MiMoVLVisionConfig, MiMoVisionPatchEmbed, MiMoVisionBlock, MiMoVisionTransformer）：视觉 Transformer 实现，包括自定义 VisionConfig、PatchEmbed、VisionBlock
- python/sglang/srt/multimodal/processors/mimo_v2.py（模块 多模态处理；类别 source；类型 dependency-wiring；符号 ImageInput, VideoInput, AudioInput, VideoAudioInput）：多模态处理器，处理图像、视频、音频的输入预处理

关键符号：load_mimo_v2_qkv_proj_weight, get_mimo_v2_fused_qkv_expected_tp_size, MiMoProcessor.process_mm_data_async, DynamicFrequencyUpdate wrapper, MiMoVisionTransformer.forward

关键源码片段

python/sglang/srt/models/mimo_v2.py

核心模型文件，包含 MiMoV2ForCausalLM 主模型和 fused-qkv 装载逻辑

```
# python/sglang/srt/models/mimo_v2.py (excerpt)

def load_mimo_v2_qkv_proj_weight(
    name, param, loaded_weight, expected_fused_tp_size: Optional[int] = None
):
    # 检查点已经分片，直接装载
    if loaded_weight.shape == param.shape:
        default_weight_loader(param, loaded_weight)
        return

    # 验证维度兼容性
    if loaded_weight.ndim != param.ndim or loaded_weight.shape[1:] != param.shape[1:]:
        raise ValueError(f"qkv_proj weight {name}: unexpected shape ...")

    tp_size = get_attention_tp_size()
    tp_rank = get_attention_tp_rank()
    # 如果指定了期望 TP 大小，验证一致性
    if expected_fused_tp_size is not None and tp_size != expected_fused_tp_size:
        raise ValueError(f"MiMoV2 fused qkv_proj checkpoint is TP={expected_fused_tp_size}-"
                         f"interleaved; got attention tp_size={tp_size} while loading {name}.")
    # 检查完整融合形状
    fused_shape = (param.shape[0] * tp_size, *param.shape[1:])
    if tuple(loaded_weight.shape) != fused_shape:
        raise ValueError(...)
    # 按当前 TP 分片装载
    default_weight_loader(param, loaded_weight.chunk(tp_size, dim=0)[tp_rank])
```

python/sglang/srt/models/mimo_vl.py

视觉 Transformer 实现，包括自定义 VisionConfig、PatchEmbed、VisionBlock

```
# python/sglang/srt/models/mimo_vl.py (excerpt)

class MiMoVLVisionConfig(PretrainedConfig):
    model_type = "mimovl"
    base_config_key = "vision_config"

    def __init__(self, depth=28, hidden_size=1280, hidden_act="silu",
                 intermediate_size=4608, num_heads=32, in_channels=3,
                 patch_size=16, spatial_merge_size=2, temporal_patch_size=2,
                 tokens_per_second=2, window_size=128, out_hidden_size=2048,
                 fullatt_block_indexes=[7,15,23,31], initializer_range=0.02,
                 kv_channels=64, qk_channels=64, num_query_groups=4,
                 num_key_value_heads=8, vit_window_attn_types=None,
                 visual_token_window_size=64, **kwargs):
        super().__init__(**kwargs)
        self.depth = depth
        self.hidden_size = hidden_size
        # ... 具体赋值
        self.vit_window_attn_types = vit_window_attn_types or [-1] * depth
        self.visual_token_window_size = visual_token_window_size

class MiMoVisionPatchEmbed(nn.Module):
    def __init__(self, patch_size=16, temporal_patch_size=2, in_channels=3, embed_dim=1536):
        super().__init__()
        kernel_size = [temporal_patch_size, patch_size, patch_size]
        self.proj = nn.Conv3d(in_channels, embed_dim, kernel_size=kernel_size,
                              stride=kernel_size, bias=False)
        self.proj_weight_linear_format = None

    @torch.no_grad()
    def sync_proj_weight_linear_format(self):
        # 缓存线性格式权重以加速 forward
        self.proj_weight_linear_format = self.proj.weight.view(self.embed_dim, -1)

    def forward(self, hidden_states: torch.Tensor) -> torch.Tensor:
        # 使用线性层替代 conv3d
        target_dtype = self.proj.weight.dtype
        hidden_states = F.linear(hidden_states.to(dtype=target_dtype),
                                 self.proj_weight_linear_format)
        return hidden_states
```

评论区精华

Review 中主要讨论了以下问题：

- softmax_scale 移除风险（JustinTong0323）：VisionAttention 重构中丢弃了 softmax_scale 参数，导致 Gemma4 视觉注意力退化。Abatom 后续修复了此问题。
- VideoData 包装兼容性（JustinTong0323）：append_video 强制包装 VideoData 可能破坏下游处理器（moss_vl、qwen_vl 等）。Abatom 通过仅在有 preprocess_kwargs 时包装来修复。
- decord 死代码（JustinTong0323）：mimo_v2_omni.py 中 import decord 块从未使用，已移除。
- TP 强制硬编码（Abatom/JustinTong0323）：最初硬编码要求 TP=4，后改为从配置派生，支持 Pro 版本 TP=8。
- 测试稳定性：MMMU 测试因 lmms-eval 与推理解析器不兼容而被移除。
  - VisionAttention softmax_scale 丢失 (correctness): 已修复，添加了 softmax_scale=softmax_scale 到 fa_kwargs。
  - append_video VideoData 包装兼容性 (design): 已修复，添加条件检查。
  - decord 死代码 (other): 已移除。
  - TP 强制硬编码 vs 配置派生 (design): 通过 get_mimo_v2_fused_qkv_expected_tp_size 从 num_key_value_heads 派生。

风险与影响

- 风险：
  1. VisionAttention 回归：修改 VisionAttention 引入 window_size 参数和重构，可能影响其他依赖该模块的模型（如 Gemma4）。已通过修复 softmax_scale 缓解。
  2. 共享内存序列化：mm_utils.py 中 wrap/unwrap 支持列表特征，可能与其他已知使用 Tensor 特征的多模态模型不兼容。
  3. TP 强制约束：fused-qkv 检查点强制特定 TP 大小，限制了用户自由选择 TP 的能力，并可能影响未来量化版本（lukealonso 提及）。
  4. MTP 稳定性：多层 EAGLE 推测解码需要仔细调试，测试中发现 accept_length 属性命名不匹配等问题。
  5. 解耦性：音频编码器强依赖 CUDA，在 CPU 推理时会引发运行时错误。 - 影响：直接影响用户：只有使用 XiaomiMiMo/MiMo-V2.5 或 MiMo-V2.5-Pro 模型的用户。但 VisionAttention 和 mm_utils 的修改影响所有使用这些公共模块的多模态模型。团队需要确保这些通用变更不引入退化。 - 风险标记：VisionAttention 回归风险，共享内存序列化兼容性，TP 强制限制灵活性，MTP 稳定性，CUDA 硬依赖

关联脉络

- PR #24436 [Gemma 4] Adding MTP support: 共享 VisionAttention 和 MTP 推测解码机制，本 PR 修改了 VisionAttention 影响 Gemma4
- PR #23953 feat(constrained): two-phase reasoning grammar + --enable-strict-thinking: 引入了推理解析器，本 PR 为 MiMo-V2.5 应用了 --reasoning-parser mimo 和 --tool-call-parser mimo

- PR #23336 [SPEC V2][2/N] feat: adaptive spec support spec v2: 自适应推测解码框架，本 PR 使用多层 EAGLE，可能依赖该框架