# PR #34992 完整报告

sgl-project/sglang

[Diffusion] Reuse SRT SigLIP in Pi0.5

合并时间：2026-08-17 19:33

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34992

## 执行摘要

- 一句话：Pi0.5 复用 SRT SigLIP 视觉塔，删除自建实现并提速 8.8%
- 推荐动作：值得精读。三个设计决策尤其值得关注：① SRT SigLIP 通过 use_data_parallel 把 TP 切分降级为全 rank 复制，规避 diffusion 场景下切分导致的语义漂移；② _candidate_weight_keys 用迭代替换实现多 checkpoint 命名映射叠加，是权重兼容层的通用手法；③ device property 改从 patch_embedding 获取，为 layerwise offload 场景提供稳定设备来源。建议同时阅读依赖 PR #34988、#34991 与同方向的 #35004，形成完整视图。

## 功能与动机

PR body 明确了目标：'replace the Pi0.5 Hugging Face PaliGemma/SigLIP runtime modules and attention monkey patch with direct native assembly around sglang.srt.models.siglip.SiglipVisionModel'，并且 'This avoids the abandoned 347-line diffusion-specific SigLIP implementation and reduces pi05_core.py by 68 net lines'。此前 Pi0.5 平行维护一套自建 SigLIP 视觉实现，与 SRT 原生实现重复，维护成本高且性能有差距；迁移后 Transformers 只承担配置与 tokenization 的职责。

## 实现拆解

1. 移除 Pi0.5 自建 SigLIP（python/sglang/multimodal_gen/runtime/models/vlas/pi05_core.py，+46/-114）：删除 Pi05SiglipAttention（基于 LocalAttention 的包装）、patch_siglip_vision_attention_to_native、siglip_vision_forward_with_openpi_dtype 以及 patch_native_attention_after_dtype_finalize 调用链；import 从 HF 运行时建模类（modeling_gemma、modeling_paligemma）收紧为配置类（configuration_gemma），并引入 sglang.srt.models.siglip.SiglipVisionModel 与 LayerwiseOffloadableModuleMixin。
2. 扩展 SRT SigLIP 契约（python/sglang/srt/models/siglip.py，+45/-9）：为 SiglipVisionEmbeddings、SiglipMLP、SiglipEncoderLayer、SiglipEncoder、SiglipVisionTransformer、SiglipVisionModel 全线增加 act_layer、flatten_batch、use_data_parallel 三个参数。use_data_parallel=True 时强制 tp_size=1、tp_rank=0、关闭 VocabParallelEmbedding 的 enable_tp，实现视觉塔在 TP 下复制而非切分；同时把 device property 改为从 patch_embedding.weight.device 获取，forward 中把 embeddings 输出对齐到 post_layernorm.weight.dtype。

3. Pi0.5 原生组装：PaliGemmaModelWithPiGemma 放弃继承 HF PaliGemmaModel，改为纯 nn.Module，直接组合 Pi05SiglipVisionModel（qkv_backend='sdpa'、flatten_batch=False、use_data_parallel=True）、新写的 PaliGemmaMultiModalProjector 与 PiGemmaModel；Pi05SiglipVisionModel 通过 LayerwiseOffloadableModuleMixin 声明 layer_names = ['vision_model.encoder.layers'] 接入逐层 offload。
4. checkpoint 名称映射（python/sglang/multimodal_gen/runtime/models/vlas/pi05_policy.py）：_candidate_weight_keys 新增 .self_attn.out_proj. → .self_attn.proj. 键名映射，并把原来只对原始 key 的 if old in key 单次替换改为对 candidates 列表逐项迭代替换，支持多映射叠加，最终 dict.fromkeys 去重保序。
5. 测试与文档配套：python/sglang/multimodal_gen/test/unit/test_pi05_runtime_helpers.py 用 test_pi05_siglip_reuses_srt_model_with_layerwise_groups 和 test_pi05_siglip_checkpoint_names_map_to_srt_layers 替换原 fake attention 测试，锁定 TP 复制、flatten_batch=False、GELU tanh、offload 分组与键名映射；docs/cookbook/vla/OpenPI/Pi0.5.mdx 补充运行期全部为 SGLang 原生模块、Transformers 仅用于配置与 tokenization 的说明。

关键文件：

- python/sglang/multimodal_gen/runtime/models/vlas/pi05_core.py（模块 前缀编码；类别 source；类型 core-logic；符号 Pi05SiglipVisionModel, PaliGemmaModelWithPiGemma, PaliGemmaMultiModalProjector, get_image_features）：本 PR 核心：删除 Pi0.5 自建 SigLIP（约 347 行的 attention 包装与 dtype 转换逻辑），改为直接组装 SRT SiglipVisionModel，净减 68 行，并重写 PaliGemma 前缀编码器的原生组装。
- python/sglang/srt/models/siglip.py（模块 视觉模型；类别 source；类型 data-contract；符号 SiglipVisionEmbeddings.init, SiglipMLP.init, SiglipEncoderLayer.init, SiglipVisionTransformer.init）：SRT 侧契约扩展：为 SigLIP 全栈注入 act_layer、flatten_batch、use_data_parallel 参数，并改进 device 来源与 dtype 对齐，是跨模块复用成立的前提。
- python/sglang/multimodal_gen/test/unit/test_pi05_runtime_helpers.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_pi05_siglip_reuses_srt_model_with_layerwise_groups, test_pi05_siglip_checkpoint_names_map_to_srt_layers）：用真实 SRT 模型构建 + 键名映射测试替换原有 fake attention 测试，锁定 TP 复制、flatten_batch、GELU tanh 与 layerwise 分组行为，防止后续回归。
- python/sglang/multimodal_gen/runtime/models/vlas/pi05_policy.py（模块 权重映射；类别 source；类型 data-contract；符号 _candidate_weight_keys）：checkpoint 兼容层：把 HF 风格的 out_proj 映射到 SRT 风格 proj，并将键名替换改为迭代叠加，保证旧权重在原生组装下可 strict 加载。
- docs/cookbook/vla/OpenPI/Pi0.5.mdx（模块 使用文档；类别 docs；类型 documentation）：文档明确 Transformers 仅为配置 /tokenization、运行时全部为 SGLang 原生模块，防止后续误用 HF 建模路径。

关键符号：Pi05SiglipVisionModel.init, PaliGemmaModelWithPiGemma.get_image_features, PaliGemmaMultiModalProjector.forward, SiglipVisionModel.forward, SiglipVisionModel.device, SiglipMLP.init, Pi05PolicyModel._candidate_weight_keys

## 关键源码片段

### python/sglang/multimodal_gen/runtime/models/vlas/pi05_core.py

本 PR 核心：删除 Pi0.5 自建 SigLIP（约 347 行的 attention 包装与 dtype 转换逻辑），改为直接组装 SRT SiglipVisionModel，净减 68 行，并重写 PaliGemma 前缀编码器的原生组装。

```
# Pi0.5 的 SigLIP 视觉塔：直接复用 SRT 原生 SiglipVisionModel，
# 叠加 LayerwiseOffloadableModuleMixin 以支持逐层 offload 到 CPU。
class Pi05SiglipVisionModel(SiglipVisionModel, LayerwiseOffloadableModuleMixin):
    layerwise_offload_dit_group_enabled = False
    # 声明逐层 offload 的层路径，供 layerwise 内存管理器按层迁移
    layer_names = ['vision_model.encoder.layers']


# PaliGemma 多模态投影层：把 SigLIP 的 patch 特征投影到 text 侧维度
class PaliGemmaMultiModalProjector(nn.Module):
    def __init__(self, config):
        super().__init__()
        self.linear = nn.Linear(
            config.vision_config.hidden_size,
            config.vision_config.projection_dim,
            bias=True,
        )

    def forward(self, image_features: torch.Tensor) -> torch.Tensor:
        return self.linear(image_features)


# 组装后的 PaliGemma 前缀编码器：不再继承 HF PaliGemmaModel，
# 改为原生组件直连，视觉塔完全走 SRT 模型。
class PaliGemmaModelWithPiGemma(nn.Module):
    def __init__(self, config, *, tensor_parallel: bool = False):
        super().__init__()
        self.config = config
        self.vision_tower = Pi05SiglipVisionModel(
            config.vision_config,
            # 从配置读取激活函数名构造激活工厂，替代原来硬编码的 QuickGELU
            act_layer=partial(get_act_fn, config.vision_config.hidden_act),
            qkv_backend='sdpa',
            flatten_batch=False, # 保留 batch/ 图像维度，不做压平
            use_data_parallel=True, # TP 下复制视觉塔，避免切分后语义漂移
        )
        self.multi_modal_projector = PaliGemmaMultiModalProjector(config)
        self.language_model = PiGemmaModel(
```

```
            config.text_config,
            tensor_parallel=tensor_parallel,
        )

    def get_image_features(
        self, pixel_values: torch.Tensor
    ) -> BaseModelOutputWithPooling:
        vision_features = self.vision_tower(pixel_values)
        image_features = self.multi_modal_projector(vision_features)
        return BaseModelOutputWithPooling(
            last_hidden_state=vision_features,
            pooler_output=image_features,
        )
```

## python/sglang/srt/models/siglip.py

SRT 侧契约扩展：为 SigLIP 全栈注入 act_layer、flatten_batch、use_data_parallel 参数，并改进 device 来源与 dtype 对齐，是跨模块复用成立的前提。

```
class SiglipMLP(nn.Module):
    def __init__(
        self,
        config,
        act_layer: Callable[[], nn.Module] = QuickGELU,
        quant_config: Optional[QuantizationConfig] = None,
        prefix: str = '',
        use_data_parallel: bool = False,
    ):
        super().__init__()
        # 复用模式下强制 tp_size=1、tp_rank=0，把 MLP 权重完整复制到每个 rank，
        # 避免 TP 切分后与 diffusion 的 sequence parallel 语义冲突。
        tp_size = 1 if use_data_parallel else get_parallel().tp_size
        tp_rank = 0 if use_data_parallel else get_parallel().tp_rank
        self.fc1 = ColumnParallelLinear(
            config.hidden_size,
            config.intermediate_size,
            quant_config=quant_config,
            prefix=add_prefix('fc1', prefix),
            tp_size=tp_size,
            tp_rank=tp_rank,
        )
        # 激活函数由 act_layer 工厂创建，Pi0.5 传入配置对应的 GELU tanh
        self.act = act_layer()
        self.fc2 = RowParallelLinear(
            config.intermediate_size,
            config.hidden_size,
            quant_config=quant_config,
            prefix=add_prefix('fc2', prefix),
            tp_size=tp_size,
            tp_rank=tp_rank,
```

```
        )


class SiglipVisionModel(nn.Module):
    @property
    def device(self) -> torch.device:
        # 从 patch_embedding 取设备而不是 encoder 首层：
        # layerwise offload 时 encoder 层可能已迁到 CPU，embeddings 常驻 GPU。
        return self.embeddings.patch_embedding.weight.device

    def forward(self, pixel_values: torch.Tensor) -> torch.Tensor:
        # 把 pixel_values 统一搬到视觉塔设备，并将 patch embedding 输出
        # 对齐到 post_layernorm 的 dtype，保证后续 encoder 计算精度稳定
        hidden_states = self.embeddings(pixel_values.to(self.device)).to(
            self.post_layernorm.weight.dtype
        )
```

## python/sglang/multimodal_gen/runtime/models/vlas/pi05_policy.py

checkpoint 兼容层：把 HF 风格的 out_proj 映射到 SRT 风格 proj，并将键名替换改为迭代叠加，保证旧权重在原生组装下可 strict 加载。

```
@staticmethod
def _candidate_weight_keys(key: str) -> list[str]:
    if key.startswith('model.'):
        key = key[len('model.'):]
    if key.startswith('PaligemmaWithExpert.'):
        key = key.replace('PaligemmaWithExpert.', 'paligemma_with_expert.', 1)
    if key.startswith('action_time_mlp_in.'):
        key = key.replace('action_time_mlp_in.', 'time_mlp_in.', 1)
    elif key.startswith('action_time_mlp_out.'):
        key = key.replace('action_time_mlp_out.', 'time_mlp_out.', 1)
    if key.startswith('state_proj.'):
        return []
    if key == 'paligemma_with_expert.gemma_expert.lm_head.weight':
        return []

    candidates = [key]
    replacements = {
        '.vision_tower.vision_model.': '.vision_tower.',
        # HF 风格的 self_attn.out_proj 映射到 SRT 风格的 self_attn.proj
        '.self_attn.out_proj.': '.self_attn.proj.',
        '.paligemma.language_model.': '.paligemma.model.language_model.',
        '.paligemma.vision_tower.': '.paligemma.model.vision_tower.',
        '.paligemma.multi_modal_projector.': (
            '.paligemma.model.multi_modal_projector.'
        ),
    }
    # 对所有候选键逐一替换，保证多个映射可以叠加生效
    #  （例如 out_proj 与 vision_tower 路径变换同时命中）。
```

```
    for old, new in replacements.items():
        for candidate in list(candidates):
            if old in candidate:
                candidates.append(candidate.replace(old, new))

    if key in {
        'paligemma_with_expert.paligemma.lm_head.weight',
        'paligemma_with_expert.paligemma.model.lm_head.weight',
    }:
        candidates.append(
            'paligemma_with_expert.paligemma.model.language_model.'
            'embed_tokens.weight'
        )
    # dict.fromkeys 保留顺序并去重，避免叠加替换产生重复候选键
    return list(dict.fromkeys(candidates))
```

## 评论区精华

本 PR 未产生任何 review 评论（PR comments 与 review comments 均为 0），设计决策主要反映在 PR body 的验证矩阵与 commit 演进中。PR body 给出的正确性验证值得注意：真实 lerobot/pi05_base checkpoint（14 GB，H100）strict 权重加载通过、fused QKV 与投影张量逐元素一致、resident 与 layerwise offload 的视觉输出逐位一致、Gemma3 SigLIP TP1/TP2 与既有基线逐位一致。commit 史中的 fix(diffusion): align SigLIP encoder input dtype 与 fix(diffusion): preserve batched Pi0.5 vision encoding 反映复用过程中重点解决了输入 dtype 对齐与批量图像维度保持两个工程问题；perf: streamline SRT vision SDPA reshapes 对应依赖 PR #34991 的性能改动。

- 暂无高价值评论线程

## 风险与影响

- 风险：
  1. SRT SigLIP 行为变更：SiglipVisionModel.forward 新增 .to(self.post_layernorm.weight.dtype)，对 bf16 checkpoint 语义等价，但若存在 embeddings 与后层异 dtype 的旧 checkpoint，数值路径会改变；PR body 仅验证了 Gemma3 的 TP1/TP2 逐位一致，覆盖范围有限。
  2. checkpoint 映射全局生效：pi05_policy.py 的 .self_attn.out_proj. → .self_attn.proj. 是对全部候选键的全局替换，若 Pi0.5 语言模型侧（PiGemma）未来出现 self_attn.out_proj 命名也会被改写；当前测试只覆盖 vision tower 路径。
  3. TP 复制显存开销：use_data_parallel=True 使视觉塔权重在每个 rank 完整复制，TP 规模越大显存占用越高，这是换取 diffusion 语义一致性的显式取舍，部署配置时需留意。
  4. 依赖未合并 PR：#34992 依赖 #34988（共享 SRT SigLIP 集成）与 #34991（vision SDPA reshape 优化），若上游数值或接口行为变化，本 PR 的正确性与性能结论需要重验。

5. layerwise offload 交互：device property 改由 patch_embedding 提供，但 forward 中 pixel_values.to(self.device) 与 post_layernorm 的所在设备需一致；layer_names 只含 encoder.layers，post_layernorm 不在 offload 列表，风险较低但值得关注。 - 影响：对用户与系统：Pi0.5 视觉编码路径性能提升约 8.8%（H100、3 相机图像、100 次迭代中位数），代码量净减 68 行，Transformers 运行时依赖收敛为配置 /tokenization。对 SRT 侧：siglip.py 新增参数默认值向后兼容，既有 Gemma3 等 SigLIP 用户不受影响（有逐位验证）。对团队：该模式可复用于其他 diffusion 模型（如 #35004 复用 CLIP），减少多模态模型的双份实现维护成本。影响程度中等偏强，横跨 multimodal_gen 与 srt/models 两个子系统，但影响面集中在 Pi0.5 与 SigLIP 模型族。 - 风险标记：跨模块核心重构，SRT SigLIP 行为变更，checkpoint 映射全局生效，TP 复制显存开销，依赖未合并 PR

## 关联脉络

- PR #34988 Shared SRT SigLIP integration (dependency): PR body 声明本 PR 依赖 #34988 提供共享 SRT SigLIP 集成，本 PR 的复用建立在其之上。
- PR #34991 Generic vision SDPA reshape optimization (dependency): PR body 声明依赖 #34991 的通用 vision SDPA reshape 优化，对应 commit 'perf: streamline SRT vision SDPA reshapes'。
- PR #35004 [Diffusion] Reuse SRT CLIP encoder blocks: 同一重构方向：将 SRT CLIP 编码器复用到 diffusion 路径，消除扩散模型专属实现；merge 冲突涉及 test_srt_siglip_reuse.py，两条改动共享 srt/models 下的编码器文件。