PR #32921 完整报告

sgl-project/sglang

[diffusion][model] Add native SANA-Video T2V support

合并时间：2026-08-12 10:07

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32921

执行摘要

- 一句话：新增 SANA-Video 原生 T2V 支持，绕过 Diffusers 通用后端
- 推荐动作：值得精读。该 PR 是 SGLang Diffusion 原生支持 SANA-Video 的完整范例，重点学习：(1) 如何用 param_names_mapping 将 Diffusers 权重映射到 SGLang 的合并投影层；(2) SanaVideoTextEncodingStage 如何复现 Diffusers 的 complex human instruction 与 prompt window 语义；(3) registry 中模型检测器的互斥注册模式。若后续要接新的视频 DiT 模型，可参照此 PR 的 pipeline 划分与潜在形状对齐逻辑。

功能与动机

SGLang Diffusion 此前通过通用 Diffusers 后端运行 SANA-Video，但该路径无法充分利用 SGLang 的调度、KV 缓存和 offload 能力。PR body 明确说明要 'Add native SGLang Diffusion support for the official Efficient-Large-Model/SANA-Video_2B_480p_diffusers text-to-video checkpoint instead of routing it through the generic Diffusers backend'，即通过原生实现获得可控的执行路径和性能优化空间。

实现拆解

1. 3D Transformer 原生实现（python/sglang/multimodal_gen/runtime/models/dits/sana_video.py）：新增 SanaVideoTransformerBlock、SanaVideoLinearAttention、SanaVideoCrossAttention、GLUMBTempConv、SanaVideoRotaryPosEmbed，覆盖 interleaved temporal/height/width RoPE、ReLU 线性注意力、packed QKV 与 KV 投影、时间维度 GLUMB 卷积，并通过 param_names_mapping 将 Diffusers checkpoint 权重映射到合并后的投影层。
2. T2V Pipeline 编排（python/sglang/multimodal_gen/runtime/pipelines/sana_video.py）：新增 SanaVideoPipeline（继承 LoRAPipeline 与 ComposedPipelineBase），通过 create_pipeline_stages 挂载 InputValidationStage、SanaVideoTextEncodingStage（实现 asymmetric 正 / 负 prompt 编码，包含 complex human instruction 拼接和 select_sana_video_prompt_window 截断窗口逻辑）以及标准 timestep/latent/denoising/decoding 阶段。
3. 配置体系搭建（configs/pipeline_configs/sana_video.py、configs/models/dits/sana_video.py、configs/sample/sana_video.py）：定义 SanaVideoPipelineConfig（task_type=T2V、flow_shift=8.0、enable_autocast=False、WanVAE 仅加载 decoder、Gemma2 文本编码器）、SanaVideoArchConfig（

patch_size=(1,2,2)、20 层、head_dim=112 等架构参数及权重映射正则）和 SanaVideoSamplingParams（480p、81 帧、16 fps、50 步、guidance 6.0 与默认 negative prompt）。

4. 注册与隔离（registry.py）：在 registry 中注册官方 checkpoint 和 detector，并通过排除条件确保与已有 SANA 图像检测器互不重叠（'sana-video'/'sana_video' 从 SANA 图片模型检测器中排除）。
5. 测试与文档配套：新增 test/unit/test_sana_video.py（4 个单元测试覆盖注册解析、潜在形状与帧对齐、prompt 窗口截断、RoPE 输出形状）；在 gpu_cases.py 增加单卡 T2V CI 用例（sana_video_2b_t2v，关闭 perf/consistency/input-reference 检查）；修改 testcase_configs.py、test_utils.py、component_accuracy/hooks.py 以支持该模型；更新 docs/docs/sglang-diffusion/compatibility_matrix.mdx 记录兼容性。

关键文件：

- python/sglang/multimodal_gen/runtime/models/dits/sana_video.py（模块 扩散模型；类别 source；类型 core-logic；符号 apply_interleaved_rotary_emb, SanaVideoRotaryPosEmbed, GLUMBTempConv, SanaVideoLinearAttention）：3D SANA-Video Transformer 的原生实现，包含核心的 interleaved RoPE、ReLU 线性注意力、GLUMB 时间卷积与 packed QKV/KV 投影，是精度对齐的关键载体。
- python/sglang/multimodal_gen/runtime/pipelines/sana_video.py（模块 扩散流程；类别 source；类型 dependency-wiring；符号 select_sana_video_prompt_window, SanaVideoTextEncodingStage, _normalize_text, _encode_negative_text）：定义 T2V pipeline 的编排逻辑与 SanaVideoTextEncodingStage，复现 Diffusers 的 prompt 指令拼接、窗口截断和正负样本编码，是行为对齐的入口。
- python/sglang/multimodal_gen/configs/pipeline_configs/sana_video.py（模块 参数配置；类别 source；类型 configuration；符号 sana_video_postprocess_text, SanaVideoPipelineConfig, post_init, adjust_num_frames）：Pipeline 配置的组装点，定义 task_type、flow_shift、VAE/ 文本编码器组合、潜在形状计算与帧数对齐，是 T2V 行为契约的声明。
- python/sglang/multimodal_gen/configs/models/dits/sana_video.py（模块 模型配置；类别 source；类型 data-contract；符号 SanaVideoArchConfig, post_init, SanaVideoConfig）：定义 3D Transformer 的架构参数与 Diffusers 权重映射规则（param_names_mapping），是权重加载正确性的基础。
- python/sglang/multimodal_gen/configs/sample/sana_video.py（模块 采样参数；类别 source；类型 configuration；符号 SanaVideoSamplingParams）：定义官方 480p 采样默认值（81 帧、16 fps、50 步、guidance 6.0）与默认 negative prompt，直接影响用户开箱即用的生成效果。
- python/sglang/multimodal_gen/registry.py（模块 模型注册；类别 source；类型 dependency-wiring）：注册 SANA-Video checkpoint 与 detector，并调整通用 SANA 检测器排除条件，避免与图片模型重叠。
- python/sglang/multimodal_gen/test/unit/test_sana_video.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_sana_video_registry_resolution, test_sana_video_pipeline_latent_shape_and_frame_alignment,

test_select_sana_video_prompt_window_keeps_first_and_tail_tokens,test_sana_video_rotary_embeddings_follow_video_token_order）：4 个单元测试覆盖配置解析、潜在形状与帧对齐、prompt 窗口截断、RoPE 输出形状，是核心逻辑的回归保障。

- python/sglang/multimodal_gen/test/server/gpu_cases.py（模块 GPU 用例；类别 test；类型 test-coverage）：新增单卡 GPU CI 用例 sana_video_2b_t2v，保证官方 checkpoint 可端到端运行。

关键符号：apply_interleaved_rotary_emb, SanaVideoRotaryPosEmbed.forward, GLUMBTempConv.forward, SanaVideoLinearAttention.forward, SanaVideoCrossAttention.forward, SanaVideoPipeline.create_pipeline_stages, SanaVideoTextEncodingStage.forward, select_sana_video_prompt_window, SanaVideoPipelineConfig.adjust_num_frames, SanaVideoPipelineConfig.prepare_latent_shape

关键源码片段

python/sglang/multimodal_gen/runtime/models/dits/sana_video.py

3D SANA-Video Transformer 的原生实现，包含核心的 interleaved RoPE、ReLU 线性注意力、GLUMB 时间卷积与 packed QKV/KV 投影，是精度对齐的关键载体。

```
# python/sglang/multimodal_gen/runtime/models/dits/sana_video.py
# SANA-Video 3D Transformer 的核心注意力实现。
# 关键点：线性注意力对 query/key 先做 ReLU，再施加 interleaved RoPE，
# 最后将 scores 与归一化因子相乘；中间计算保持 FP32 以避免精度损失。
class SanaVideoLinearAttention(nn.Module):
    """Diffusers-compatible ReLU linear attention with packed QKV."""

    def __init__(self, query_dim, num_heads, head_dim, bias):
        super().__init__()
        self.num_heads = num_heads
        self.head_dim = head_dim
        self.inner_dim = num_heads * head_dim
        # 将 Diffusers 中独立的 to_q / to_k / to_v 合并为单个投影层，
        # 权重映射由 SanaVideoArchConfig.param_names_mapping 负责。
        self.to_qkv = MergedColumnParallelLinear(
            query_dim,
            [self.inner_dim, self.inner_dim, self.inner_dim],
            bias=bias,
            gather_output=True,
        )
        self.norm_q = RMSNorm(self.inner_dim, eps=1e-5)
        self.norm_k = RMSNorm(self.inner_dim, eps=1e-5)
        self.to_out = nn.ModuleList(
            [nn.Linear(self.inner_dim, query_dim, bias=True), nn.Identity()]
        )

    def forward(self, hidden_states, rotary_emb):
        original_dtype = hidden_states.dtype
```

```
        batch_size, sequence_length, _ = hidden_states.shape
        qkv, _ = self.to_qkv(hidden_states)
        query, key, value = qkv.split(self.inner_dim, dim=-1)
        query = self.norm_q(query).view(
            batch_size, sequence_length, self.num_heads, self.head_dim
        )
        key = self.norm_k(key).view(
            batch_size, sequence_length, self.num_heads, self.head_dim
        )
        value = value.view(batch_size, sequence_length, self.num_heads, self.head_dim)

        query = F.relu(query)
        key = F.relu(key)
        query_rotate = apply_interleaved_rotary_emb(query, *rotary_emb)
        key_rotate = apply_interleaved_rotary_emb(key, *rotary_emb)

        query = query.permute(0, 2, 3, 1)
        key = key.permute(0, 2, 3, 1)
        # 线性注意力的核心矩阵乘在 FP32 下进行，避免低精度累积误差。
        query_rotate = query_rotate.permute(0, 2, 3, 1).float()
        key_rotate = key_rotate.permute(0, 2, 3, 1).float()
        value = value.permute(0, 2, 3, 1).float()

        # 因果线性注意力的归一化：用未旋转的 key 统计求和作为分母。
        normalizer = 1.0 / (
            key.sum(dim=-1, keepdim=True).transpose(-2, -1) @ query + 1e-15
        )
        scores = value @ key_rotate.transpose(-1, -2)
        hidden_states = (scores @ query_rotate) * normalizer
        hidden_states = hidden_states.flatten(1, 2).transpose(1, 2)
        # 输出回落到原始精度，保持与 Diffusers 行为一致。
        hidden_states = hidden_states.to(original_dtype)
        return self.to_out[0](hidden_states)
```

python/sglang/multimodal_gen/runtime/pipelines/sana_video.py

定义 T2V pipeline 的编排逻辑与 SanaVideoTextEncodingStage，复现 Diffusers 的 prompt 指令拼接、窗口截断和正负样本编码，是行为对齐的入口。

```
# python/sglang/multimodal_gen/runtime/pipelines/sana_video.py
# SANA-Video 的 prompt 编码需要严格复现 Diffusers 的语义：
# 先拼接 'complex human instruction'，编码后只保留 BOS 和尾部窗口。
def select_sana_video_prompt_window(tensor, max_sequence_length):
    """Keep the BOS token and the final prompt window, matching Diffusers."""
    if tensor.shape[1] < max_sequence_length:
        raise ValueError(
            f"Encoded prompt has {tensor.shape[1]} tokens, expected at least "
            f"{max_sequence_length}"
        )
    if max_sequence_length == 1:
```

```
        return tensor[:, :1]
    # BOS 保留在首位，其余取最后 max_sequence_length - 1 个 token。
    return torch.cat([tensor[:, :1], tensor[:, -(max_sequence_length - 1):]], dim=1)


class SanaVideoTextEncodingStage(TextEncodingStage):
    """Apply SANA-Video's asymmetric positive/negative prompt encoding."""

    @torch.no_grad()
    def forward(self, batch, server_args):
        assert batch.prompt is not None
        self.tokenizers[0].padding_side = "right"
        all_indices = list(range(len(self.text_encoders)))
        max_sequence_length = batch.max_sequence_length or 300
        prompt = self._normalize_text(batch.prompt)
        prompt_list = [prompt] if isinstance(prompt, str) else prompt
        # 与 Diffusers pipeline 默认完全一致的增强指令前缀。
        enhanced_prompt = [
            SANA_VIDEO_COMPLEX_HUMAN_INSTRUCTION + item for item in prompt_list
        ]
        instruction_tokens = len(
            self.tokenizers[0].encode(SANA_VIDEO_COMPLEX_HUMAN_INSTRUCTION)
        )
        encoded_length = instruction_tokens + max_sequence_length - 2
        positive_outputs = list(
            self.encode_text(
                enhanced_prompt,
                server_args,
                encoder_index=all_indices,
                return_attention_mask=True,
                max_length=encoded_length,
            )
        )

        # 对 embedding、mask 等输出统一做窗口截断，保持 BOS + 尾部语义。
        for output_index in (0, 1, 3):
            positive_outputs[output_index] = [
                select_sana_video_prompt_window(tensor, max_sequence_length)
                for tensor in positive_outputs[output_index]
            ]
        positive_outputs[4] = [
            [int(value) for value in mask.sum(dim=1).tolist()]
            for mask in positive_outputs[1]
        ]

        self._append_positive_text_outputs(batch, *positive_outputs)
        if batch.do_classifier_free_guidance:
            negative_outputs = self._encode_negative_text(batch, server_args, all_indices)
            self._append_negative_text_outputs(batch, positive_outputs[0], *negative_outputs)
```

```
        return batch
```

python/sglang/multimodal_gen/configs/pipeline_configs/sana_video.py

Pipeline 配置的组装点，定义 task_type、flow_shift、VAE/ 文本编码器组合、潜在形状计算与帧数对齐，是 T2V 行为契约的声明。

```
# python/sglang/multimodal_gen/configs/pipeline_configs/sana_video.py
# SANA-Video T2V 的 pipeline 配置契约。
@dataclass
class SanaVideoPipelineConfig(PipelineConfig):
    task_type: ModelTaskType = ModelTaskType.T2V
    should_use_guidance: bool = False
    flow_shift: float | None = 8.0
    # 线性注意力刻意在 FP32 下累积分数，因此关闭 autocast。
    enable_autocast: bool = False

    dit_config: DiTConfig = field(default_factory=SanaVideoConfig)
    vae_config: VAEConfig = field(default_factory=WanVAEConfig)
    vae_tiling: bool = False
    vae_sp: bool = False
    vae_precision: str = "fp32"
    vae_decode_precision: str = "fp32"

    text_encoder_configs: tuple[EncoderConfig, ...] = field(
        default_factory=lambda: (Gemma2Config(),)
    )
    text_encoder_precisions: tuple[str, ...] = field(default_factory=lambda: ("bf16",))
    text_encoder_extra_args: list[dict] = field(
        default_factory=lambda: [
            {
                "padding": "max_length",
                "return_attention_mask": True,
                "add_special_tokens": True,
            }
        ]
    )

    def __post_init__(self) -> None:
        # T2V 只用 VAE 解码器，不加载编码器。
        self.vae_config.load_encoder = False
        self.vae_config.load_decoder = True

    def adjust_num_frames(self, num_frames: int) -> int:
        # 帧数必须对齐 VAE 的时间压缩率：(n - 1) % ratio == 0。
        temporal_scale = self.vae_config.arch_config.temporal_compression_ratio
        if num_frames < 1:
            raise ValueError("num_frames must be positive")
        return ((num_frames - 1) // temporal_scale) * temporal_scale + 1
```

```
def prepare_latent_shape(self, batch, batch_size, num_frames):
    # 潜在张量为 5D：batch、通道、帧、高、宽，空间维度按 VAE 压缩率缩小。
    spatial_scale = self.vae_config.arch_config.spatial_compression_ratio
    return (
        batch_size,
        self.dit_config.arch_config.num_channels_latents,
        num_frames,
        batch.height // spatial_scale,
        batch.width // spatial_scale,
    )
```

评论区精华

该 PR 的 review 评论仅有两条来自 gemini-code-assist[bot] 的自动告警，提示 'The consumer version of Gemini Code Assist on GitHub has been sunset. All code review activity has officially ceased.'，没有实质性的技术讨论记录。从提交历史看，开发者 BBuf 在合并前主动修复了 SANA-Video 的 import 顺序问题（commit abfcc85），并两次合并 main 分支保持同步，说明代码质量把关主要发生在 PR 提交前的自检和 CI 阶段。

- Gemini Code Assist 自动评论失效 (other): 无实质技术讨论；代码质量主要依赖 PR 提交者的自检、组件精度验证和 CI。

风险与影响

- 风险：
  1. 新增源码路径无删除、高风险逻辑集中：3D Transformer 实现（513 行）和 pipeline 编排（152 行）为全新代码，线性注意力的 FP32 累积、interleaved RoPE 的切片索引（0::2 / 1::2）和 prompt window 截断（保留 BOS + 尾部窗口）均是精度敏感逻辑，若与 Diffusers 行为细节存在偏差，可能出现生成质量下降但不易察觉。
  2. 注册顺序与检测器互斥：registry.py 中 SANA-Video 需在通用 SANA 之前注册，且通用 SANA detector 增加了排除条件；未来若新增其他含 'sana' 子串的模型，可能引入检测冲突。
  3. 单卡 CI 覆盖有限：新增的 GPU 用例关闭了 perf、consistency 与 input-reference 检查，仅验证可运行性；组件精度验证依赖人工在 RTX 5090 上执行，未固化为自动门槛。
  4. 兼容性约束：依赖 diffusers.models.embeddings.PixArtAlphaTextProjection，diffusers 版本升级可能影响行为；vae_precision/vae_decode_precision 固定为 fp32，在多卡或低显存场景可能有性能与显存压力（当前单卡峰值 16.9 GB）。 - 影响：对用户而言，SANA-Video 2B 480p 现在可获得原生 SGLang Diffusion 路径，支持 LoRA pipeline 组合、显存 offload 等能力，并可按官方 480p 默认参数直接生成视频。对系统而言，新增一个完整的 T2V pipeline 与 3D DiT 实现，registry 的模型识别逻辑更复杂但隔离清晰。对团队而言，该 PR 为后续 SANA 系列视频模型（如 SANA-Video-WM）提供了可复用的架构模板和权重映射范式。 - 风险标记：核心逻辑全新实现，精度依赖手工验证，检测器互斥逻辑需持续维护，GPU CI 用例关闭关键检查项

关联脉络

- PR #34401 Fix model-driven DiT layerwise offload auto policy: 同为 multimodal_gen 下 diffusion 模型配置与 runtime 的联动修改，涉及模型部署配置与平台适配，与 SANA-Video 共享 pipeline config 基础设施。
- PR #31590 Add Cosmos3 Edge and Distilled checkpoints support: 同为 multimodal_gen 新增视频生成模型（T2V/I2V/T2I）的原生支持，涉及 pipeline 配置、模型实现与 registry 注册，是 SANA-Video 的同类先例。
- PR #34315 [diffusion] LTX-2: mount the bit-exact fused modulate at the 8 bare adaLN sites: 同为 diffusion 模型的原生实现优化，注重与 Diffusers 的位精确对齐，与 SANA-Video 的组件精度对齐思路一致。