PR #32341 完整报告

sgl-project/sglang

[diffusion][model] Add LingBot-Video MoE 30B T2V support

合并时间：2026-08-07 17:57

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32341

执行摘要

- 一句话：新增 LoRA 支持的 30B MoE 文生视频模型，复用 fused_experts
- 推荐动作：值得精读。核心设计决策：1）直接复用 srt fused_experts 而非在 diffusion 侧自研 MoE GEMM，注意 gate_up_interleaved/inplace/routed_scaling_factor 三个对齐上游语义的关键开关；2）fp32 敏感模块清单 LINGBOT_VIDEO_FP32_MODULES 与 should_keep_in_fp32 的精度管理方式；3）热路径优化教训——权重打包一次加载、RoPE 设备端一次构建、B>1 合并为单次 batched attention；4）srt TP group 桥接的权宜设计及其后续框架级替代方向。阅读顺序建议：先看 runtime/layers/moe.py，再看 lingbot_video_moe.py 的 attention 与 position ids，最后看 pipeline 组装。

功能与动机

PR body 明确提出 "Add native SGLang support for LingBot-Video MoE 30B...a DeepSeek-V3-style MoE text-to-video model (128 experts, 30B total / ~3B active per token, 48 MoE layers)"，并强调 "This is the first MoE DiTin multimodal_gen, reusing SGLang's fused_experts Triton kernel for the expert GEMMs"。关联 issue #32336 提出模型接入需求。PR 同时说明该 DiT 训练于结构化 JSON caption，raw 自然语言 prompt 会 out-of-distribution，因此 prompt 字段需携带 rewriter 的结构化输出。

实现拆解

实现分 6 步拆解：

1. MoE 层落地：新增 runtime/layers/moe.py，实现 LingBotVideoRouter（fp32 计算、sigmoid + e_score_correction_bias、group-limited top-k）、LingBotVideoGroupedExperts（w13_weight 与 w2 打包为单个 Parameter）、LingBotVideoSparseMoeBlock（128 路由专家 + 1 共享专家）。关键设计是 _run_sglang_triton_experts 直接调用 srt 的 fused_experts，并通过 MoeRunnerConfig(gate_up_interleaved=False, inplace=False, apply_router_weight_on_input=False, routed_scaling_factor=None) 对齐上游权重布局与缩放语义——router 已预缩放分数，内核不得二次缩放。评审后 w13 改为加载时一次打包，消除每层 forward 中的 torch.cat 热路径。
2. DiT 主模型：新增 runtime/models/dits/lingbot_video_moe.py，LingBotVideoTransformer3DModel 实现 joint self-attention（video;text 拼接），LingBotVideoAttention 用 ColumnParallelLinear/RowParallelLinear + USPAttention，

RoPE 采用 NDRotaryEmbedding + _apply_rotary_emb（neox 风格关闭），位置编号由 _joint_position_ids 在设备上一次性生成（支持 batch 内变长文本，padding 由 mask 隔离）。AdaLN 通过 scale_shift_table 与 token 级 temb6 调制，fp32 敏感模块由 LINGBOT_VIDEO_FP32_MODULES 清单决定（含 router、各 norm、time_embedder 等），实现 should_keep_in_fp32 供 offload 与精度策略使用。

3. 配置与注册：新增 configs/models/dits/lingbot_video_moe.py（LingBotVideoMoEArchConfig，48 层、128 专家、top-8、moe_intermediate_size=768、n_group=4、topk_group=2、routed_scaling_factor=2.5）、configs/pipeline_configs/lingbot_video_moe.py（T2V 任务、WanVAEConfig、flow_shift=3.0、Qwen3-VL 文本编码器、get_decode_scale_and_shift 读取 vae 的 latents_mean/std）、configs/sample/lingbot_video_moe.py（默认 81 帧 480×480 与结构化负面 prompt），并在 registry.py 与各 __init__.py 完成注册导出。
4. Pipeline 组装：LingBotVideoTextEncodingStage 继承 TextEncodingStage，实现 check_inputs（帧数 4n+1、宽高 16 对齐）、_compute_crop_start（模板前缀 token 数缓存一次）、_encode_prompt（crop 模板前缀、B=1 时去掉右侧 padding）；LingBotVideoPipeline 按 InputValidationStage → 文本编码 → 标准 latent/timestep/denoising（_flow_shift_kwarg 注入 shift）→ 标准 decoding 的顺序组合，整体继承 LoRAPipeline 以支持后续权重更新。
5. srt 桥接：runtime/distributed/parallel_state.py 新增 _sync_srt_tp_group/_clear_srt_tp_group，在 diffusion 侧初始化 TP group 后同步给 srt 的 parallel_state，并在 destroy_model_parallel 时清理；gpu_worker.py 发布独立的 SrtMoeBridgeArgs（而非污染 diffusion ServerArgs）到 srt runtime context，让 fused_experts 拿到所需运行参数。这是 PR 自述的 pragmatic workaround，后续可在框架层由 worker 统一初始化 srt parallel_state 替代。
6. 测试与性能基线：test/unit/test_lingbot_video_moe.py（367 行）覆盖 MoE 配置解析、router bias 只影响选路不影响权重、B>1 注意力样本隔离、2D mask/varlen 元数据透传、文本编码 crop+trim；test/server/testcase_configs.py、gpu_cases.py 增加服务器端 case，perf_baselines/5090.json 与 h100.json 写入性能基线。

关键文件：

- python/sglang/multimodal_gen/runtime/models/dits/lingbot_video_moe.py（模块 DiT 模型；类别 source；类型 data-contract；符号 should_keep_in_fp32, LingBotVideoRMSNorm, make_joint_position_ids, _joint_position_ids）：DiT 主模型：joint attention、NDRotaryEmbedding、AdaLN 调制、3D joint position ids、fp32 敏感模块清单。首个 MoE DiT 的模型侧实现，578 行。
- python/sglang/multimodal_gen/runtime/layers/moe.py（模块 MoE 层；类别 source；类型 core-logic；符号 LingBotVideoMLP, LingBotVideoRouter, _group_limited_topk, LingBotVideoGroupedExperts）：首次在 diffusion 运行时引入 MoE 层：DeepSeek-V3 风格路由、group-limited top-k、复用 srt fused_experts Triton 内核，评审热路径优化（w13 打包）的核心文件。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/lingbot_video_moe/text_encoding.py（模块 文本编码；类别 source；类型 data-contract

；符号 LingBotVideoTextEncodingStage, check_inputs, apply_text_to_template,_compute_crop_start）：LingBotVideoTextEncodingStage：Qwen3-VL 提示编码、PROMPT_TEMPLATE 前缀 crop 缓存、B=1 去 padding，pipeline 正确性的关键 stage。

- python/sglang/multimodal_gen/configs/models/dits/lingbot_video_moe.py（模块 模型配置；类别 source；类型 data-contract；符号 is_blocks, LingBotVideoMoEArchConfig, LingBotVideoMoEConfig）：LingBotVideoMoEArchConfig：30B 架构参数（48 层、128 专家、top-8）、identity param_names_mapping、FSDP shard 条件，权重加载契约所在。
- python/sglang/multimodal_gen/configs/pipeline_configs/lingbot_video_moe.py（模块 管道配置；类别 source；类型 core-logic；符号 _qwen3vl_postprocess_text, LingBotVideoMoEPipelineConfig, get_model_deployment_config, get_pos_prompt_embeds）：LingBotVideoMoEPipelineConfig：T2V 任务装配（Qwen3-VL 编码器、WanVAE、flow_shift、CFG 默认值），latents 反归一化契约。
- python/sglang/multimodal_gen/runtime/pipelines/lingbot_video_moe.py（模块 管道组装；类别 source；类型 core-logic；符号 _flow_shift_kwarg, LingBotVideoPipeline, create_pipeline_stages）：LingBotVideoPipeline 组装入口：InputValidation → 文本编码 → 标准 latent/timestep/denoising/decoding 阶段，继承 LoRAPipeline。
- python/sglang/multimodal_gen/test/unit/test_lingbot_video_moe.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_moe_path_resolves_moe_configs, test_arch_config_defaults_without_mlp_only_layers, test_router_bias_shifts_selection_but_not_gate_weights, test_attention_isolates_samples_across_batch）：367 行单元测试：覆盖 MoE 配置解析、router bias 语义、B>1 注意力隔离、2D mask/varlen 透传、文本编码 crop/trim，是本 PR 正确性保障的主体。
- python/sglang/multimodal_gen/runtime/distributed/parallel_state.py（模块 并行状态；类别 source；类型 core-logic；符号 _sync_srt_tp_group, _clear_srt_tp_group）：srt TP group 桥接：_sync_srt_tp_group/_clear_srt_tp_group 让 fused_experts 复用 diffusion 侧初始化的并行组，跨运行时协作的关键改动。

关键符号：should_keep_in_fp32, LingBotVideoRMSNorm.forward, _joint_position_ids, LingBotVideoAttention.forward, LingBotVideoBlock.forward, LingBotVideoSparseMoeBlock.forward, _run_sglang_triton_experts, LingBotVideoRouter.forward, _group_limited_topk, LingBotVideoTextEncodingStage.forward, _encode_prompt, _compute_crop_start, LingBotVideoPipeline.create_pipeline_stages, _flow_shift_kwarg, _sync_srt_tp_group, _clear_srt_tp_group

关键源码片段

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

DiT 主模型：joint attention、NDRotaryEmbedding、AdaLN 调制、3D joint position ids、fp32 敏感模块清单。首个 MoE DiT 的模型侧实现，578 行。

```
class LingBotVideoAttention(nn.Module):
```

```
    """LingBot-Video MoE 的 joint attention：video;text 拼接后统一做注意力。"""

    def __init__(self, hidden_size, num_heads, norm_eps, qkv_bias, out_bias, prefix="",
                 supported_attention_backends=None, quant_config=None):
        super().__init__()
        self.num_heads = num_heads
        self.head_dim = hidden_size // num_heads
        tp_size = get_tp_world_size()
        self.local_num_heads = divide(num_heads, tp_size)

        # q/k/v/out 全部走并行线性层，为后续 TP/SP 扩展预留接口
        self.to_q = ColumnParallelLinear(hidden_size, hidden_size, bias=qkv_bias,
                                         gather_output=False, quant_config=quant_config)
        self.to_k = ColumnParallelLinear(hidden_size, hidden_size, bias=qkv_bias,
                                         gather_output=False, quant_config=quant_config)
        self.to_v = ColumnParallelLinear(hidden_size, hidden_size, bias=qkv_bias,
                                         gather_output=False, quant_config=quant_config)
        self.norm_q = LingBotVideoRMSNorm(self.head_dim, norm_eps)
        self.norm_k = LingBotVideoRMSNorm(self.head_dim, norm_eps)
        self.to_out = RowParallelLinear(hidden_size, hidden_size, bias=out_bias,
                                        input_is_parallel=True, quant_config=quant_config)
        self.attn = USPAttention(num_heads=self.local_num_heads, head_size=self.head_dim,
                                 dropout_rate=0, softmax_scale=None, causal=False,
                                 supported_attention_backends=supported_attention_backends,
                                 skip_sequence_parallel=False, quant_config=quant_config)

    def forward(self, x, freqs_cis, attention_mask=None, attn_mask_meta=None):
        cos, sin = freqs_cis
        q, _ = self.to_q(x)
        k, _ = self.to_k(x)
        v, _ = self.to_v(x)
        # 先做 head 维 RMSNorm，再 reshape 出头维度
        q = self.norm_q(q.unflatten(2, (self.local_num_heads, self.head_dim)))
        k = self.norm_k(k.unflatten(2, (self.local_num_heads, self.head_dim)))
        v = v.unflatten(2, (self.local_num_heads, self.head_dim))

        B, S, H, D = q.shape
        # 将 batch 展平为一条序列做一次 RoPE 调用，避免 per-sample 循环与设备同步；
        # 样本间隔离由 attention_mask / attn_mask_meta 保证
        q = _apply_rotary_emb(q.reshape(1, B * S, H, D), cos, sin, is_neox_style=False).reshape(B,
        S, H, D)
        k = _apply_rotary_emb(k.reshape(1, B * S, H, D), cos, sin, is_neox_style=False).reshape(B,
        S, H, D)
        out = self.attn(q, k, v, attn_mask=attention_mask, attn_mask_meta=attn_mask_meta)
        out = out.flatten(2)
        out, _ = self.to_out(out)
        return out
```

python/sglang/multimodal_gen/runtime/layers/moe.py

首次在 diffusion 运行时引入 MoE 层：DeepSeek-V3 风格路由、group-limited top-k、复用 srt fused_experts Triton 内核，评审热路径优化（w13 打包）的核心文件。

```
class LingBotVideoSparseMoeBlock(nn.Module):
    """DeepSeek-V3 风格 MoE 块：128 路由专家 + 1 共享专家，复用 srt 的 fused_experts。"""

    def __init__(self, hidden_size, intermediate_size, num_experts, top_k, score_func,
                 norm_topk_prob, n_group, topk_group, routed_scaling_factor, n_shared_experts):
        super().__init__()
        self.hidden_size = hidden_size
        self.num_experts = num_experts
        self.top_k = top_k
        self.intermediate_size = intermediate_size
        self.router = LingBotVideoRouter(hidden_size, num_experts, top_k, score_func,
                                         norm_topk_prob, n_group, topk_group,
                                         routed_scaling_factor)
        # w13_weight 已在加载时打包为 [E, 2I, H]；forward 热路径不再执行 torch.cat((w1, w3))，
        # 避免每层每次调用物化约 768 MiB 临时张量
        self.experts = LingBotVideoGroupedExperts(num_experts, hidden_size, intermediate_size)
        self.shared_experts = None
        if n_shared_experts is not None and n_shared_experts > 0:
            self.shared_experts = LingBotVideoMLP(hidden_size, intermediate_size * n_shared_
            experts)

    def _run_sglang_triton_experts(self, tokens, top_scores, top_indices):
        # 直接复用 srt 的 Triton 融合 MoE 内核，与 LLM 运行时共用同一套专家 GEMM 实现
        from sglang.srt.layers.moe.moe_runner import MoeRunnerConfig
        from sglang.srt.layers.moe.moe_runner.triton_utils.fused_moe import fused_experts
        from sglang.srt.layers.moe.topk import StandardTopKOutput

        topk_output = StandardTopKOutput(
            topk_weights=top_scores.float(),
            topk_ids=top_indices.to(torch.int32),
            router_logits=torch.empty(0, device=tokens.device),
        )
        # Router 已预缩放 topk 分数，fused_experts 不能再乘 routed_scaling_factor；
        # gate_up_interleaved=False 与 inplace=False 是对齐上游权重的关键开关
        runner_config = MoeRunnerConfig(
            num_experts=self.num_experts,
            num_local_experts=self.num_experts,
            hidden_size=self.hidden_size,
            intermediate_size_per_partition=self.intermediate_size,
            top_k=self.top_k,
            activation="silu",
            is_gated=True,
            inplace=False,
            apply_router_weight_on_input=False,
            routed_scaling_factor=None,
            gate_up_interleaved=False,
```

```
        )
        return fused_experts(tokens.contiguous().bfloat16(),
                             self.experts.w13_weight.bfloat16(),
                             self.experts.w2.bfloat16(), topk_output, runner_config).type_as(tokens)

    def forward(self, hidden_states):
        b = hidden_states.shape[0]
        tokens = hidden_states.reshape(-1, self.hidden_size)
        top_indices, top_scores = self.router(tokens)
        out = self._run_sglang_triton_experts(tokens, top_scores, top_indices)
        out = out.reshape(b, -1, self.hidden_size)
        if self.shared_experts is not None:
            out = out + self.shared_experts(hidden_states)
        return out
```

python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/lingbot_video_moe/text_encoding.py

LingBotVideoTextEncodingStage：Qwen3-VL 提示编码、PROMPT_TEMPLATE 前缀 crop 缓存、B=1 去 padding，pipeline 正确性的关键 stage。

```
@torch.no_grad()
def _encode_prompt(self, prompt, device, dtype):
    text_encoder = self.text_encoders[0]
    if text_encoder is None or self.tokenizers[0] is None:
        raise ValueError("`text_encoder` and `processor` are required for encode_prompt().")

    inputs = self._build_prompt_inputs(prompt)
    inputs = inputs.to(device)
    outputs = text_encoder(**inputs, output_hidden_states=self.hidden_state_skip_layer is not
    None)
    # 取倒数第 hidden_state_skip_layer+1 层的 hidden states，与上游逐字对齐
    if self.hidden_state_skip_layer is not None:
        prompt_embeds = outputs.hidden_states[-(self.hidden_state_skip_layer + 1)]
    else:
        prompt_embeds = outputs.last_hidden_state

    prompt_mask = inputs["attention_mask"]
    # 模板前缀（system 指令等）的 token 数只计算一次并缓存，避免每次编码都重复 tokenize 前缀
    crop_start = self._compute_crop_start()
    if crop_start > 0:
        prompt_embeds = prompt_embeds[:, crop_start:]
        prompt_mask = prompt_mask[:, crop_start:]

    # B=1 时去掉右侧 padding 再交给 DiT 推理，减少冗余计算量
    if prompt_embeds.shape[0] == 1:
        true_len = int(prompt_mask[0].sum().item())
        prompt_embeds = prompt_embeds[:, :true_len]
        prompt_mask = prompt_mask[:, :true_len]
```

```
    return prompt_embeds.to(dtype=dtype), prompt_mask
```

评论区精华

核心讨论集中在两轮 CHANGES_REQUESTED 后的架构收敛：

- MoE 热路径性能：mickqian 指出 "torch.cat((self.experts.w1, self.experts.w3), dim=1) runs in every MoE block forward. At the current BF16 shape, it materializes [128, 1536, 2048] (768 MiB) per invocation"，并要求 "pack/load w13 in the fused-kernel layout once"；同时批评 "text_lens.detach().cpu().tolist() synchronizes the GPU on every denoising step"。作者修复后 5090 生成时间从 110s 降到 105s，且输出 byte-identical。
- 注意力与 RoPE 原生化：mickqian 要求 "adapt to USPAttention and native rotary embedding modules"，作者将手写 complex64 RoPE + SDPA 替换为 NDRotaryEmbedding + _apply_rotary_emb + USPAttention，q/k/v/out 全部改走 ColumnParallelLinear/RowParallelLinear，为多卡 TP 铺路。
- stage 拆分：mickqian 明确 "we'd prefer using native split stages instead of aggregated stage"，作者放弃 monolithic BeforeDenoisingStage，改为独立文本编码 stage + 框架标准 latent/timestep/decoding 阶段。
- ServerArgs 桥接范围：作者回应 mickqian "move to adjust logic of ServerArgs" 时说明，3 个参数是 srt fused_experts 内部旋钮（非用户 flag），因此放入 gpu_worker 的 SrtMoeBridgeArgs 而非扩散到公共 ServerArgs。
- MoE 热路径性能：per-forward w1/w3 拼接与 CPU 同步 (performance): 作者按建议将 w13 打包为单参数加载一次，RoPE 改为设备端一次性构建，B>1 注意力合并为单次 batched 调用；5090 生成 110s → 105s，输出 byte-identical。
- 改用 USPAttention 与原生 rotary embedding (design): 作者替换为 NDRotaryEmbedding + _apply_rotary_emb + USPAttention，q/k/v/out 改用 ColumnParallelLinear/RowParallelLinear，为后续 TP/SP 扩展铺路。
- 拆分 aggregated stage 为原生 split stages (design): 作者拆分为 LingBotVideoTextEncodingStage + 标准 latent/timestep/denoising/decoding 阶段，复用框架标准组件，PR body 中也同步更新了该方案。
- srt ServerArgs 桥接范围 (design): 桥接参数收敛到 worker 初始化处，srt knobs 不再泄漏到公共 ServerArgs；作者确认仅 fused_experts 读取这些参数。
- 与上游 diffusers 的 BF16 输出差异 (question): 作者断言为预期的 BF16 推理级差异而非正确性 bug，场景内容、prompt 符合度与整体质量等价，并附三份对照视频。

风险与影响

- 风险：
  1. 跨运行时耦合：runtime/layers/moe.py 直接 import srt 内部 API（MoeRunnerConfig、StandardTopKOutput、fused_experts），srt 侧 MoE 接口演进会直接破坏 diffusion 路径；_sync_srt_tp_group 对 srt 全局 _TP 的写入是隐式副作用，若 srt 侧已初始化 TP group 则不会覆盖，但生命周期管理（destroy 时按对象同一性清理）依赖实现细节。

2. BF16 数值差异：PR body 明确承认与 diffusers 输出存在镜头运动与轨迹差异，根因是 fused_experts vs torch._grouped_mm 的累加顺序差异与 SDP dispatch 差异，跨 48 层 × 40 步累积。对要求逐帧对齐的评测场景（如 benchmark 一致性验证）可能造成困扰。
3. 显存与 offload 边界：fp32 敏感模块（router、全部 RMSNorm）在 --dit-layerwise-offload 场景下的驻留策略未在 PR 中说明；_run_sglang_triton_experts 强制 tokens.contiguous().bfloat16() 会额外复制。
4. 路由正确性：e_score_correction_bias 参与选路但不参与门控权重（测试已覆盖），_group_limited_topk 中 group 内 top-2 求和与 n_group 整除假设若未来配置变化可能越界。
5. MVP 边界：仅 T2V base 单卡，TP/SP/FSDP、refiner、prompt rewriter、B>1 长 prompt 均未覆盖，USPAttention 的 varlen 元数据路径在 B>1 时依赖 attn_mask_meta 正确性，尚未有端到端多请求测试。 - 影响：对用户：robbyant/lingbot-video-moe-30b-a3b 可通过 sglang serve 一键使用，32GB 显卡配合 --dit-layerwise-offload 可运行，H100 70s 完成 40 步 241 帧生成（快于 diffusers 85s）。对系统：multimodal_gen 首次打通与 srt MoE 内核的复用通道，确立了 "srt bridge + 独立 MoE 层 " 的可复制模式；registry 新增一类模型，后续 refiner（同架构 1080p DiT）可直接复用 MoE 层。对团队：23 个 commit、20 文件、1666 行新增，跨 diffusion 模型、分布式并行、srt MoE 三个子系统，合并者 mickqian 深度参与代码提交（batched masked USP attention 支持），协作模式为后续大型模型接入提供了参照。 - 风险标记：跨运行时耦合，热路径性能风险，BF16 数值差异，单卡 MVP 范围，首次 MoE 内核复用

关联脉络

- PR #33923 [Diffusion] Route zimage and hunyuanvideo attention through USPAttention: 同为 diffusion 注意力基础设施向 USPAttention 收敛的演进，本 PR 的 LingBot 注意力直接基于 USPAttention 实现，共享同一抽象。
- PR #33707 Derive H3 attention admission from backend capabilities: multimodal_gen 注意力后端能力准入机制，本 PR 的 LingBotVideoAttention 也依赖 supported_attention_backends 选择后端，属于同一注意力后端演进脉络。
- PR #32667 [Diffusion] Add K/V-gather sequence parallel attention: SP 注意力能力建设，本 PR Future Work 中的多卡 SP 计划正是基于这类 SP 注意力基础设施，且都改动了 multimodal_gen 注意力层。
- PR #33849 [diffusion] gate fast VAE paths by quality: 本 PR 的 pipeline 解码阶段复用标准 decoding stage 与 VAE，该 PR 引入的 quality 门控 VA E 路径会影响 LingBot 视频解码行为，两者同属 diffusion pipeline 演进。