PR #23151 完整报告

sgl-project/sglang

[Diffusion] add per-step rollout options for SDE and trajectory capture

合并时间：2026-04-24 23:26

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23151

执行摘要

- 一句话：添加逐步骤 SDE 过滤与轨迹捕获选项
- 推荐动作：值得精读，尤其是 scheduler_rl_mixin.py 中按步骤选择 SDE 类型的设计，和 rollout_denoising_mixin.py 中轨迹收集过滤的实现。测试中的严格位精确断言也是良好实践。

功能与动机

PR body 中指出：需要支持按步骤粒度的 SDE rollout，允许部分步骤执行随机 SDE/CPS（贡献真实 log-prob），其余步骤回退到确定性 ODE（贡献 0）。同时修复了变量命名不一致问题，并加强了 ODE 路径的位精确测试。

实现拆解

1. 定义新参数：在 io_struct.py 的 RolloutRequest 和 sampling_params.py 的 SamplingParams 中新增 rollout_sde_step_indices: Optional[List[int]] 和 rollout_return_step_indices: Optional[List[int]] 字段，默认 None 保持向后兼容。
2. 入口层提取与传递：将 rollout_api.py 中的 rollout_generate 函数的采样参数构建逻辑提取为 _build_sampling_kwargs 函数，将新字段加入字典，并在返回前过滤掉 None 值。
3. 核心 SDE/ODE 分支过滤：在 scheduler_rl_mixin.py 的 flow_sde_sampling 方法中，通过读取 batch.rollout_sde_step_indices 和 batch._rollout_loop_step_index 决定实际使用的SDE类型。若当前步骤不在允许列表中，则回退为ODE，跳过噪声注入，log-prob为零。 # scheduler_rl_mixin.py 关键片段 sde_step_indices = getattr(batch, " rollout_sde_step_indices", None) loop_step_index = getattr(batch, " _rollout_loop_step_index", None) if (sde_type != "ode" and sde_step_indices is not None and loop_step_index is not None and loop_step_index not in sde_step_indices): effective_sde_type = "ode" else: effective_sde_type = sde_type # 后续根据 effective_sde_type 执行对应分支
4. 轨迹收集过滤：在 rollout_denoising_mixin.py 的 _maybe_append_dit_trajectory_step 方法中新增 step_index 参数，与 batch.rollout_return_step_indices 比对，仅当步骤在列表中才追加轨迹。同时统一了 _rollout_dit_env_state 更名为 _rollout_denoising_env_state，删除不再使用的 sanitize_dit_env_kwargs 调用，并移除未实现的 sanitize_denoising_env_kwargs 钩子。 # rollout_denoising_mixin.py 关键片段 def_maybe_append_dit_trajectory_step(self, batch, latents, timestep_value,

```
step_index ):     if not batch.rollout or not batch.rollout_return_dit_trajectory:
return     state = getattr(batch, "_rollout_denoising_env_state", None)     if state is
None:         return     return_step_indices = getattr(batch, "
rollout_return_step_indices", None)     if return_step_indices is not None and
step_index not in return_step_indices:         return
state["step_latents"].append(latents.detach())
state["step_timesteps"].append(timestep_value.detach().cpu())
```

5. 测试覆盖加强：新增 test_timestep_filters_gate_sde_and_trajectory 单元测试验证过滤正确性，并强化 ODE 位精确测试使用 assertEqual(diff, 0.0) 而非 torch.equal，报告具体差值。同时新增 TestBuildSamplingKwargs 测试参数传递路径。

关键文件：

- python/sglang/multimodal_gen/test/unit/test_scheduler_rollout_unit.py（模块 Rollout 测试；类别 test；类型 test-coverage；符号 test_timestep_filters_gate_sde_and_trajectory, _mock_variance_noise, _DummyDit）：新增逐步骤过滤测试，验证 SDE gating 和轨迹 gating 的完整性；加强 ODE 位精确测试为绝对相等断言。
- python/sglang/multimodal_gen/runtime/post_training/rollout_denoising_mixin.py（模块 去噪混合器；类别 source；类型 core-logic；符号 _maybe_finalize_dit_env_collection, _maybe_finalize_denoising_env_collection）：核心混合类，实现轨迹收集的过滤逻辑和变量统一。
- python/sglang/multimodal_gen/test/unit/test_rollout_api.py（模块 Rollout API；类别 test；类型 test-coverage；符号 TestBuildSamplingKwargs, _make_request, test_step_index_filters_forwarded, test_step_index_filters_default_dropped_as_none）：新增 TestBuildSamplingKwargs 测试类，验证参数传递和默认值行为。
- python/sglang/multimodal_gen/runtime/entrypoints/post_training/rollout_api.py（模块 Rollout API；类别 source；类型 entrypoint；符号 rollout_generate, _build_sampling_kwargs）：入口层提取 _build_sampling_kwargs 函数，包含新字段映射，简化 rollout_generate。
- python/sglang/multimodal_gen/runtime/post_training/scheduler_rl_mixin.py（模块 调度器强化；类别 source；类型 core-logic；符号 flow_sde_sampling）：核心 SDE 过滤逻辑，根据 rollout_sde_step_indices 和 loop_step_index 动态选择 effective_sde_type。

关键符号：test_timestep_filters_gate_sde_and_trajectory, _maybe_append_dit_trajectory_step, _maybe_finalize_denoising_env_collection, _build_sampling_kwargs, flow_sde_sampling, test_step_index_filters_forwarded, test_step_index_filters_default_dropped_as_none, test_sampling_params_exposes_filters_via_req_getattr

关键源码片段

python/sglang/multimodal_gen/runtime/post_training/rollout_denoising_mixin.py

核心混合类，实现轨迹收集的过滤逻辑和变量统一。

```
def _maybe_append_dit_trajectory_step(self, batch, latents, timestep_value, step_index):
    # 仅在启用 rollout 且要求返回轨迹时执行
    if not batch.rollout or not batch.rollout_return_dit_trajectory:
        return
    state = getattr(batch, "_rollout_denoising_env_state", None)
    if state is None:
        return
    # 如果指定了步骤过滤，仅在 step_index 在列表中时追加
    return_step_indices = getattr(batch, "rollout_return_step_indices", None)
    if return_step_indices is not None and step_index not in return_step_indices:
        return
    state["step_latents"].append(latents.detach())
    state["step_timesteps"].append(timestep_value.detach().cpu())
```

python/sglang/multimodal_gen/runtime/post_training/scheduler_rl_mixin.py

核心 SDE 过滤逻辑，根据 rollout_sde_step_indices 和 loop_step_index 动态选择 effective_sde_type。

```
# scheduler_rl_mixin.py 中 flow_sde_sampling 的过滤逻辑
sde_step_indices = getattr(batch, "rollout_sde_step_indices", None)
loop_step_index = getattr(batch, "_rollout_loop_step_index", None)
if (
    sde_type != "ode"
    and sde_step_indices is not None
    and loop_step_index is not None
    and loop_step_index not in sde_step_indices
):
    effective_sde_type = "ode"
else:
    effective_sde_type = sde_type

# 后续根据 effective_sde_type 分支
if effective_sde_type == "sde":
    # fp32 转换和噪声注入 ...
    pass
elif effective_sde_type == "cps":
    # ...
    pass
elif effective_sde_type == "ode":
    prev_sample = sample + dt * model_output
    # log-prob 为零
    pass
```

评论区精华

无实质性讨论。Gemini Code Assist 机器人评论表示无反馈，维护者 mickqian 直接批准。所有变更在内部已充分审查。

- 暂无高价值评论线程

风险与影响

- 风险：主要风险在于新参数与现有系统的兼容性。测试验证了默认 None 时行为不变，且 ODE 路径保持位精确相等。引入的 loop_step_index 机制避免了与 scheduler._step_index 的偏移问题。代码变更集中在扩散模块内，不影响其他模块。
- 影响：影响范围限于扩散模型的 rollout 训练 / 评估场景。用户可通过新参数精细控制去噪过程的随机性和轨迹记录。这为 FlowGRPO 等基于 rollout 的训练方法提供必要支持。变更向后兼容，无 breaking change。
- 风险标记：新参数 None 默认保障兼容性，ODE 路径位精确测试覆盖

关联脉络

- 暂无明显关联 PR