PR #35868 完整报告

sgl-project/sglang

[diffusion] Preserve PEFT LoRA semantics

合并时间：2026-08-21 22:58

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35868

执行摘要

- 一句话：新增 PEFT LoRA 语义支持，fail-closed 拒绝无法表示的语义
- 推荐动作：值得精读。关注 peft_adapter.py 的 fail-closed 设计、normalize_peft_keys 的前缀 / 槽位处理、scale_fused_sections 对融合 section 的逐层 alpha 实现，以及如何在不新增 CLI 的情况下读取 adapter_config.json。

功能与动机

PR body 说明这是对 #35774 的后续，目标是在现有 LoRA 格式转换之前先规范化常见的 PEFT 模型包装器和命名适配器槽位，应用 use_rslora 和 per-layer alpha_pattern，并对无法用 native diffusion LoRA 层表示的 DoRA 和辅助 / 运行时 PEFT 特性 fail-closed，避免静默运行普通 LoRA 数学产生错误结果。

实现拆解

1. 新增 pipelines_core/lora/peft_adapter.py，作为 PEFT 兼容层核心：load_peft_config（读取同目录 adapter_config.json）、get_peft_lora_alpha（校验 alpha 为正整数）、normalize_peft_keys（剥离 base_model.model. / peft_model.base_model.model. 前缀和命名槽位如 lora_A.default.weight → lora_A.weight）、_validate_peft_features（fail-closed 校验）、apply_peft_config（应用 rslora/alpha_pattern 缩放）、scale_fused_sections（处理融合投影 section 的逐层缩放）。
2. 在 format_adapter.py 的 normalize_lora_state_dict 中新增 adapter_config 参数，并在格式检测前先执行 normalize_peft_keys，格式转换后执行 apply_peft_config，使所有 LoRA 格式统一走 PEFT 语义归一化。
3. 在 pipeline.py 的 load_lora_adapter 中移除手写 json 读取逻辑，统一走 load_peft_config + get_peft_lora_alpha；_store_fused_lora_groups 引入 scale_fused_sections，使融合投影（如 attn.qkv 多 section）也能保留逐层 alpha。
4. 目录重构：lora_pipeline.py → pipelines_core/lora/pipeline.py，lora_format_adapter.py → pipelines_core/lora/format_adapter.py，并更新 pipelines_core/__init__.py、gpu_worker.py、各 pipeline 文件等共 17 处导入路径。
5. 测试配套：新增 test/unit/test_lora_peft.py（rslora 缩放保持 delta、unsupported fail-closed、非法 alpha fail-closed），扩展 test/unit/test_fused_lora_compose.py（逐层 alpha 的融合 section 数值一致性）。其中第 2、3 步是关键行为变化，第 4 步是纯组

织重构，PR body 说明相对 main 的生产文件 churn 为 +51/-31，其中 17 对行是目录移动的导入修改。

关键文件：

- python/sglang/multimodal_gen/runtime/pipelines_core/lora/peft_adapter.py（模块 PEFT 适配；类别 source；类型 core-logic；符号 load_peft_config, get_peft_lora_alpha, normalize_peft_keys, _validate_peft_features）：新增核心 PEFT 适配模块，包含键名规范化、配置校验、缩放计算与 fail-closed 校验，是本 PR 的核心实现。
- python/sglang/multimodal_gen/runtime/pipelines_core/lora/pipeline.py（模块 LoRA 管线；类别 source；类型 rename-or-move；符号 _store_fused_lora_groups, load_lora_adapter）：LoRA 加载入口与融合分组写入逻辑修改，接入 PEFT 配置读取和逐层 alpha 缩放。
- python/sglang/multimodal_gen/runtime/pipelines_core/lora/format_adapter.py（模块 格式转换；类别 source；类型 rename-or-move；符号 normalize_lora_state_dict）：格式转换入口接入 PEFT 键名规范化和配置应用，使所有 LoRA 格式统一支持 PEFT 语义。
- python/sglang/multimodal_gen/test/unit/test_lora_peft.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_peft_wrapper_slot_and_rslora_scaling_preserve_delta, test_unsupported_peft_runtime_semantics_fail_closed, test_invalid_peft_lora_alpha_fails_closed）：新增 PEFT 语义的针对性测试，覆盖 rslora 缩放、fail-closed 拒绝和不合法 alpha。
- python/sglang/multimodal_gen/test/unit/test_fused_lora_compose.py（模块 融合测试；类别 test；类型 test-coverage；符号 test_fused_sections_preserve_per_layer_alpha）：扩展融合 LoRA 组合测试，验证逐层 alpha 在融合投影 section 上的数值一致性。
- python/sglang/multimodal_gen/runtime/pipelines_core/__init__.py（模块 包导出；类别 source；类型 dependency-wiring）：导出入口调整，指向新的 LoRA pipeline 子包路径。

关键符号：load_peft_config, get_peft_lora_alpha, normalize_peft_keys, _validate_peft_features, apply_peft_config, scale_fused_sections, normalize_lora_state_dict, _store_fused_lora_groups, load_lora_adapter

关键源码片段

python/sglang/multimodal_gen/runtime/pipelines_core/lora/peft_adapter.py

新增核心 PEFT 适配模块，包含键名规范化、配置校验、缩放计算与 fail-closed 校验，是本 PR 的核心实现。

```
"""PEFT 检查点语义适配到 native diffusion LoRA 层。"""

# 正则匹配命名适配器槽位，例如 `lora_A.default.weight` 中的 `default`
_ADAPTER_SLOT = re.compile(r"(\.lora_[AB])\.([^.]+)\.weight$")
# 常见的 PEFT 包装前缀，按顺序尝试
_WRAPPER_PREFIXES = ("peft_model.base_model.model.", "base_model.model.")

def load_peft_config(weight_path: str) -> dict[str, Any]:
```

```
    """读取 LoRA 权重同目录下的 adapter_config.json，不存在时返回空字典。"""
    path = Path(weight_path).with_name("adapter_config.json")
    if not path.is_file():
        return {}
    with path.open(encoding="utf-8") as file:
        config = json.load(file)
    if not isinstance(config, dict):
        raise ValueError("PEFT adapter_config.json must contain a JSON object")
    return config


def get_peft_lora_alpha(config: Mapping[str, Any]) -> int | None:
    """校验并提取 lora_alpha，必须是正整数，否则抛出 ValueError。"""
    alpha = config.get("lora_alpha")
    if alpha is None:
        return None
    if (
        isinstance(alpha, bool)
        or not isinstance(alpha, (int, float))
        or alpha <= 0
        or isinstance(alpha, float)
        and not alpha.is_integer()
    ):
        raise ValueError("PEFT lora_alpha must be a positive integer")
    return int(alpha)


def normalize_peft_keys(state_dict: Mapping[str, torch.Tensor]) -> dict[str, torch.Tensor]:
    """剥离统一的 PEFT 包装前缀和命名适配器槽位，得到规范化键名。"""
    # 选择能匹配所有键的最长包装前缀，避免误删
    prefix = next(
        (
            prefix
            for prefix in _WRAPPER_PREFIXES
            if state_dict and all(name.startswith(prefix) for name in state_dict)
        ),
        "",
    )
    normalized: dict[str, torch.Tensor] = {}
    slots = set()
    has_bare_weights = False
    for name, tensor in state_dict.items():
        name = name.removeprefix(prefix)
        match = _ADAPTER_SLOT.search(name)
        if match is not None:
            slots.add(match.group(2)) # 记录命名的 adapter 槽位名
        elif name.endswith((".lora_A.weight", ".lora_B.weight")):
            has_bare_weights = True # 存在未命名的裸权重
        target = _ADAPTER_SLOT.sub(r"\1.weight", name) # 移除槽位名，如 lora_A.default.weight -
```

```
            > lora_A.weight
            if target in normalized:
                raise ValueError(
                    "LoRA checkpoint contains multiple PEFT adapter slots for "
                    f"the same tensor: {target!r}"
                )
            normalized[target] = tensor
        if len(slots) > 1:
            raise ValueError(
                f"LoRA checkpoint contains multiple PEFT adapter slots: {sorted(slots)}"
            )
        if slots and has_bare_weights:
            raise ValueError("LoRA checkpoint mixes named and unnamed PEFT adapter slots")
        return normalized
```

python/sglang/multimodal_gen/runtime/pipelines_core/lora/pipeline.py

LoRA 加载入口与融合分组写入逻辑修改，接入 PEFT 配置读取和逐层 alpha 缩放。

```
    def _store_fused_lora_groups(
        adapter: dict[str, torch.Tensor],
        to_merge_params: dict[Hashable, dict[Any, Any]],
        adapter_alpha: int | None,
        device: torch.device | str,
    ) -> None:
        """将延迟收拢的融合 lora_A/B 分组写回 adapter 字典。"""
        for a_key, a_parts in list(to_merge_params.items()):
            if not str(a_key).endswith(".lora_A"):
                continue
            base = str(a_key)[: -len(".lora_A")]
            b_key = f"{base}.lora_B"
            b_parts = to_merge_params.get(b_key)
            n = max(a_parts) + 1
            # 要求 A/B 分组的索引都是连续的 0..n-1，否则跳过不融合
            if (
                b_parts is None
                or set(a_parts) != set(range(n))
                or set(b_parts) != set(range(n))
            ):
                continue
            a_list = [a_parts[i] for i in range(n)]
            b_list = [b_parts[i] for i in range(n)]
            # 若存在逐层 alpha_pattern，缩放 B 部分后再拼接，保留 PEFT 语义
            scaled_b = scale_fused_sections(
                a_parts,
                b_parts,
                to_merge_params.get(f"{base}.alpha", {}),
                adapter_alpha,
            )
            a, b, fused_alpha = stack_or_compose_fused_lora(
```

```
            a_list, scaled_b or b_list, None if scaled_b else adapter_alpha
        )
        if scaled_b:
            # 逐层缩放后统一 alpha 不能再用 r_eff，改为总行数
            fused_alpha = a.shape[-2]
        adapter[str(a_key)] = a.to(device)
        adapter[b_key] = b.to(device)
        if fused_alpha is not None:
            adapter[f"{base}.alpha"] = torch.tensor(float(fused_alpha), device=device)
```

python/sglang/multimodal_gen/runtime/pipelines_core/lora/format_adapter.py

格式转换入口接入 PEFT 键名规范化和配置应用，使所有 LoRA 格式统一支持 PEFT 语义。

```
def normalize_lora_state_dict(
    state_dict: Mapping[str, torch.Tensor],
    logger: Optional[logging.Logger] = None,
    *,
    adapter_config: Mapping[str, Any] | None = None,
) -> Dict[str, torch.Tensor]:
    """Normalize any supported LoRA format into a single canonical layout."""
    log = logger or globals()["logger"]

    # 先剥离 PEFT 包装前缀和命名槽位，再做格式检测，保证检测逻辑不受 PEFT 包装干扰
    state_dict = normalize_peft_keys(state_dict)
    keys = list(state_dict.keys())
    log.info(
        "[LoRAFormatAdapter] normalize_lora_state_dict called, #keys=%d",
        len(keys),
    )
    if keys:
        log.info(
            "[LoRAFormatAdapter] before convert, sample keys (<=20): %s",
            ", ".join(_sample_keys(keys, 20)),
        )

    fmt = detect_lora_format_from_state_dict(state_dict)
    log.info("[LoRAFormatAdapter] detected format: %s", fmt)

    normalized = convert_lora_state_dict_by_format(state_dict, fmt, log)
    # 格式转换后再应用 PEFT 配置，将 use_rslora/alpha_pattern 等语义写入权重
    normalized = apply_peft_config(normalized, adapter_config or {})

    norm_keys = list(normalized.keys())
    if norm_keys:
        log.info(
            "[LoRAFormatAdapter] after convert, sample keys (<=20): %s",
            ", ".join(_sample_keys(norm_keys, 20)),
        )
```

```
    return normalized
```

评论区精华

本 PR 没有 review 评论。设计取舍在 body 中说明：PEFT 语义不支持的特性 fail before weight injection rather than silently running ordinary LoRA math；pipelines_core/lora/__init__.py 不 re-export 符号以避免隐式依赖与循环导入。Issue 评论只有 /tag-and-rerun-ci，无技术讨论。

- 暂无高价值评论线程

风险与影响

- 风险：行为风险：normalize_peft_keys 要求所有键共享同一个包装前缀，若检查点混用裸权重和命名槽位会直接报错；这是有意 fail-closed，但可能让原本勉强能跑的权重被拒绝。alpha_pattern 按正则匹配目标层，误匹配会导致缩放错误。兼容风险：lora_pipeline.py 等旧路径被移动，外部直接 from ...lora_pipeline import ... 的代码会失效，仓库内已同步，但下游用户若引用旧路径需更新。数值风险：scale_fused_sections 改变了融合 LoRA 的叠加方式（按层缩放后再拼接），测试验证了数值一致，但覆盖面限于测试中的配置。无性能影响：新增逻辑只在加载期执行，不影响推理。
- 影响：对用户：支持 PEFT 格式 LoRA（含 use_rslora、alpha_pattern），语义更准确；DoRA 等会得到明确报错而非错误结果。对系统：LoRA 加载路径统一，新增一个适配层，无性能影响。对团队：代码组织更清晰，为后续扩展更多 PEFT 特性奠定基础。影响范围限于 diffusion 模块的 LoRA 加载链路。
- 风险标记：核心路径变更，fail-closed 可能拒绝原本可用的权重，目录移动影响外部导入，alpha_pattern 正则匹配风险，融合 section 数值需回归验证

关联脉络

- PR #35774 [diffusion] Resolve LoRA weight sources deterministically: 本 PR 是其后续，延续 LoRA 权重加载的确定性处理，并新增 PEFT 语义支持。