PR #40588 完整报告

vllm-project/vllm

[Models][Gemma3/Gemma4] Support hidden_act variants in gated MLP

合并时间：2026-05-09 02:29

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/40588

执行摘要

- 一句话：支持 Gemma3/4 门控 MLP 的 hidden_act 变体
- 推荐动作：此 PR 值得精读，特别是 activation.py 中的注册表扩展方式和模型层调用 get_act_and_mul_fn 的解耦设计，适合作为 vLLM 中支持模型配置变体的范例。

功能与动机

作者（Preferred Networks）开发了可以作为 Gemma3/Gemma4 变体的新模型，希望这些模型能通过 transformers 配置在 vLLM 上运行，而无需硬编码激活函数。

实现拆解

1. 在 vllm/model_executor/layers/activation.py 的 _ACTIVATION_REGISTRY 中添加了 "swish": lambda: nn.SiLU() 作为别名。
2. 在同一文件的 _ACTIVATION_AND_MUL_REGISTRY 中添加了 "gelu_pytorch_tanh": lambda: GeluAndMul(approximate="tanh") 和 "swish": lambda: SiluAndMul()，使门控 MLP 路由能够识别这些激活函数。
3. 修改 vllm/model_executor/models/gemma3.py 和 vllm/model_executor/models/gemma4.py 中的 Gemma3MLP.__init__ 和 Gemma4MLP.__init__：移除对 hidden_activation 的硬编码检查，改为 self.act_fn = get_act_and_mul_fn(hidden_activation)，同时将导入从 GeluAndMul 改为 get_act_and_mul_fn。
4. 新增测试文件 tests/model_executor/test_gemma_hidden_act.py，包含三个测试：验证 get_act_and_mul_fn 对三种激活名返回正确类型、验证 get_act_fn 支持 swish 别名、验证 Gemma3MLP 和 Gemma4MLP 实例化时能正确设置 act_fn 并能执行前向传播。

关键文件：

- vllm/model_executor/layers/activation.py（模块 激活层；类别 source；类型 data-contract；符号 _ACTIVATION_REGISTRY, _ACTIVATION_AND_MUL_REGISTRY, get_act_and_mul_fn, get_act_fn）：核心基础层变更：在激活注册表中添加 gelu_pytorch_tanh、swish 条目，使 get_act_and_mul_fn 和 get_act_fn 能正确解析这些名字。
- vllm/model_executor/models/gemma3.py（模块 模型定义；类别 source；类型 data-contract；符号 Gemma3MLP）：Gemma3 MLP 层移除 hidden_activation 硬编码检

查，改为通过 get_act_and_mul_fn 动态解析。

- vllm/model_executor/models/gemma4.py（模块 模型定义；类别 source；类型 data-contract；符号 Gemma4MLP）：Gemma4 MLP 层做相同改造，与 gemma3.py 一致。
- tests/model_executor/test_gemma_hidden_act.py（模块 测试验证；类别 test；类型 test-coverage；符号 test_get_act_and_mul_fn_supports_gemma_hidden_act_aliases, test_get_act_fn_supports_swish_alias, test_gemma_mlp_supports_hidden_act_variants）：新增的测试文件，覆盖了三种激活变体在 get_act_and_mul_fn 和两个 MLP 类中的正确行为。

关键符号：get_act_and_mul_fn, get_act_fn, Gemma3MLP.init, Gemma4MLP.init

关键源码片段

vllm/model_executor/layers/activation.py

核心基础层变更：在激活注册表中添加 gelu_pytorch_tanh、swish 条目，使 get_act_and_mul_fn 和 get_act_fn 能正确解析这些名字。

```
# vllm/model_executor/layers/activation.py
# 在 _ACTIVATION_REGISTRY 中添加 swish 别名（与 silu 相同）
_ACTIVATION_REGISTRY = LazyDict({
    "gelu": lambda: GELU(),
    "gelu_fast": lambda: FastGELU(),
    "gelu_new": lambda: NewGELU(),
    "gelu_pytorch_tanh": lambda: _get_gelu_pytorch_tanh(),
    "relu": lambda: nn.ReLU(),
    "relu2": lambda: ReLUSquaredActivation(),
    "silu": lambda: nn.SiLU(),
    "swish": lambda: nn.SiLU(), # 新增：swish 与 SiLU 等价
    "quick_gelu": lambda: QuickGELU(),
    "tanh": lambda: nn.Tanh(),
    "sigmoid": lambda: nn.Sigmoid(),
    "xielu": lambda: XIELU(),
})

# 在 _ACTIVATION_AND_MUL_REGISTRY 中添加 gelu_pytorch_tanh 和 swish
_ACTIVATION_AND_MUL_REGISTRY: LazyDict[nn.Module] = LazyDict({
    "gelu": lambda: GeluAndMul(),
    "gelu_pytorch_tanh": lambda: GeluAndMul(approximate="tanh"), # 新增
    "silu": lambda: SiluAndMul(),
    "swish": lambda: SiluAndMul(), # 新增：swish 映射到 SiluAndMul
    "geglu": lambda: GeluAndMul(),
    "swigluoai": lambda: SwigluOAIAndMul(),
})
```

vllm/model_executor/models/gemma3.py

Gemma3 MLP 层移除 hidden_activation 硬编码检查，改为通过 get_act_and_mul_fn 动态解析。

```
# vllm/model_executor/models/gemma3.py
# 修改后的 Gemma3MLP.__init__
class Gemma3MLP(nn.Module):
    def __init__(
        self,
        hidden_size: int,
        intermediate_size: int,
        hidden_activation: str,
        quant_config: QuantizationConfig | None = None,
        prefix: str = "",
    ) -> None:
        super().__init__()
        self.gate_up_proj = MergedColumnParallelLinear(
            hidden_size, [intermediate_size] * 2, bias=False,
            quant_config=quant_config, prefix=f"{prefix}.gate_up_proj",
        )
        self.down_proj = RowParallelLinear(
            intermediate_size, hidden_size, bias=False,
            quant_config=quant_config, prefix=f"{prefix}.down_proj",
        )
        # 移除了 hidden_activation 的硬编码检查和 ValueError
        # 现在通过 get_act_and_mul_fn 动态解析
        self.act_fn = get_act_and_mul_fn(hidden_activation)

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        gate_up, _ = self.gate_up_proj(x)
        x = self.act_fn(gate_up)
        x, _ = self.down_proj(x)
        return x
```

tests/model_executor/test_gemma_hidden_act.py

新增的测试文件，覆盖了三种激活变体在 get_act_and_mul_fn 和两个 MLP 类中的正确行为。

```
# tests/model_executor/test_gemma_hidden_act.py
import pytest
import torch

from vllm.model_executor.layers.activation import (
    GeluAndMul, SiluAndMul, get_act_and_mul_fn, get_act_fn,
)
from vllm.model_executor.models.gemma3 import Gemma3MLP
from vllm.model_executor.models.gemma4 import Gemma4MLP

@pytest.mark.parametrize(
    ("activation_name", "expected_type"),
    [
        ("gelu_pytorch_tanh", GeluAndMul),
        ("silu", SiluAndMul),
        ("swish", SiluAndMul), # swish 应返回与 silu 相同的类型
```

```
    ],
)
def test_get_act_and_mul_fn_supports_gemma_hidden_act_aliases(
    activation_name: str,
    expected_type: type[torch.nn.Module],
    default_vllm_config,
) -> None:
    # 验证 get_act_and_mul_fn 返回正确的模块类型
    assert isinstance(get_act_and_mul_fn(activation_name), expected_type)

def test_get_act_fn_supports_swish_alias() -> None:
    # 验证 swish 作为 get_act_fn 的别名也返回 SiLU
    assert isinstance(get_act_fn("swish"), torch.nn.SiLU)

@pytest.mark.parametrize("mlp_cls", [Gemma3MLP, Gemma4MLP])
@pytest.mark.parametrize(
    ("activation_name", "expected_type"),
    [
        ("gelu_pytorch_tanh", GeluAndMul),
        ("silu", SiluAndMul),
        ("swish", SiluAndMul),
    ],
)
def test_gemma_mlp_supports_hidden_act_variants(
    mlp_cls: type[torch.nn.Module],
    activation_name: str,
    expected_type: type[torch.nn.Module],
    default_vllm_config,
    dist_init,
) -> None:
    # 使用指定的 hidden_activation 实例化 MLP
    mlp = mlp_cls(
        hidden_size=16,
        intermediate_size=32,
        hidden_activation=activation_name,
    )
    # 验证 act_fn 类型正确
    assert isinstance(mlp.act_fn, expected_type)
    # 验证前向传播能正常执行
    assert mlp(torch.randn(3, 16)).shape == (3, 16)
```

评论区精华

审核人 mgoin 批准了该 PR，评论 "LGTM! Thanks for the generalization"。机器人审核（Claude、Gemini）未提出具体反馈，整个审核过程无实质性讨论。

- Activation alias support (design): PR 被接受，无修改要求。

风险与影响

- 风险：风险很低。核心变更是在激活注册表中添加别名，并用 get_act_and_mul_fn 替换硬编码，这两个都是 vLLM 中已有的测试良好模式。若配置中使用了尚未注册的激活名，get_act_and_mul_fn 会抛出 ValueError，行为清晰可预测。同时，测试覆盖了所有新增 / 变更的路径，包括三种激活名在两种模型上的实例化和前向传播。向后兼容性：所有现有使用 gelu_pytorch_tanh 的模型配置不受影响。
- 影响：对用户而言，现在可以为 Gemma3 和 Gemma4 模型指定 hidden_activation 为 silu 或 swish（以及原有的 gelu_pytorch_tanh），从而扩大了适配的 checkpoint 范围。对系统内部，激活函数解析集中到 activation.py，模型定义更简洁，便于后续维护。对团队，此 PR 提供了一个可复用的模式：未来为其他模型添加激活变体时只需注册即可。
- 风险标记：低风险，测试覆盖完整

关联脉络

- PR #39582 [Model] Gemma4 quantized MoE weight loading and KV cache spec merge: PR body 中提及这是唯一相关的 Gemma PR，但与激活变体无关。