PR #41574 完整报告

vllm-project/vllm

[Model] Fix Gemma4 MoE activation mismatch

合并时间：2026-05-05 12:34

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41574

执行摘要

- 一句话：修复 Gemma4 MoE 激活函数为 tanh 近似 GELU
- 推荐动作：该 PR 设计清晰、改动集中，已获得批准。建议团队在合入后验证 Gemma4 模型生成质量无明显波动，并确认 LoRA 相关测试通过。

功能与动机

Gemma4 的 config.json 中指定 hidden_activation="gelu_pytorch_tanh"，HuggingFace 参考实现使用 nn.GELU(approximate="tanh")，且同一文件中的密集 MLP Gemma4MLP 已强制使用 gelu_pytorch_tanh。MoE 分支原本因 MoEActivation 缺少该变体而退而使用精确 GELU，导致与官方规范不符。

实现拆解

1. 扩展 MoEActivation 枚举（activation.py）：新增 GELU_TANH = "gelu_tanh" 和 GELU_TANH_NO_MUL = "gelu_tanh_no_mul" 枚举成员，并添加字符串别名映射 _STR_ALIASES 使得 from_str("gelu_pytorch_tanh") 也能解析为 GELU_TANH。
2. 绑定自定义算子（activation.py）：在 _CUSTOM_OP_NAMES 查找表中将 GELU_TANH 映射到 "gelu_tanh_and_mul"，GELU_TANH_NO_MUL 同样映射到 "gelu_tanh_and_mul"；在 _WITHOUT_MUL 中添加 GELU_TANH 到 GELU_TANH_NO_MUL 的映射。
3. 实现门控和非门控调度（apply_moe_activation 函数）：门控分支调用 torch.ops._C.gelu_tanh_and_mul(output, input)；非门控分支调用 F.gelu(input, approximate="tanh")。
4. 注册到所有 Fused MoE 后端：在 fused_batched_moe.py、fused_humming_moe.py、fused_marlin_moe.py、fused_moe.py 的 _supports_activation 静态方法中，将 MoEActivation.GELU_TANH 和 MoEActivation.GELU_TANH_NO_MUL 加入允许列表。
5. 修复 Gemma4 模型调用（gemma4.py）：将 Gemma4MoE.__init__ 中传给 FusedMoE 的 activation 参数从 "gelu" 改为 "gelu_tanh"，使 MoE 层与密集层及 HuggingFace 参考实现一致。

关键文件：

- vllm/model_executor/layers/fused_moe/activation.py（模块 MoE 激活；类别 source；类型 data-contract；符号 MoEActivation.GELU_TANH, MoEActivation.GELU_TANH_NO_MUL, _STR_ALIASES, _CUSTOM_OP_NAMES）：核心

文件：新增 GELU_TANH 枚举、别名映射、自定义算子绑定、以及 apply_moe_activation 中的调度逻辑。

- vllm/model_executor/models/gemma4.py（模块 Gemma4 模型；类别 source；类型 data-contract；符号 Gemma4MoE.init）：修复目标文件：将激活参数从 "gelu" 改为 "gelu_tanh"，使 MoE 层与密集层及官方配置一致。
- vllm/model_executor/layers/fused_moe/fused_batched_moe.py（模块 MoE 后端；类别 source；类型 data-contract；符号 BatchedTritonExperts._supports_activation）：在 BatchedTriton 后端注册新激活，确保兼容性。
- vllm/model_executor/layers/fused_moe/fused_humming_moe.py（模块 MoE 后端；类别 source；类型 data-contract；符号 HummingExpertsBase._supports_activation）：在 Humming 后端注册新激活。
- vllm/model_executor/layers/fused_moe/fused_marlin_moe.py（模块 MoE 后端；类别 source；类型 data-contract；符号 MarlinExpertsBase._supports_activation）：在 Marlin 后端注册新激活。
- vllm/model_executor/layers/fused_moe/fused_moe.py（模块 MoE 后端；类别 source；类型 data-contract；符号 TritonExperts._supports_activation）：在基础 Fused MoE 后端注册新激活。

关键符号：MoEActivation.from_str, apply_moe_activation, Gemma4MoE.init, BatchedTritonExperts._supports_activation, HummingExpertsBase._supports_activation, MarlinExpertsBase._supports_activation, TritonExperts._supports_activation

关键源码片段

vllm/model_executor/layers/fused_moe/activation.py

核心文件：新增 GELU_TANH 枚举、别名映射、自定义算子绑定、以及 apply_moe_activation 中的调度逻辑。

```
# 关键片段：GELU_TANH 枚举定义与调度
class MoEActivation(Enum):
    # ... 其他成员
    GELU_TANH = "gelu_tanh" # tanh 近似 GELU，门控版本
    # ...
    GELU_TANH_NO_MUL = "gelu_tanh_no_mul" # 非门控版本

# 字符串别名，支持 HuggingFace 命名
_STR_ALIASES: dict[str, str] = {
    "gelu_pytorch_tanh": "gelu_tanh",
}

# 映射到自定义算子名称
_CUSTOM_OP_NAMES: dict[MoEActivation, str] = {
    MoEActivation.GELU_TANH: "gelu_tanh_and_mul",
    MoEActivation.GELU_TANH_NO_MUL: "gelu_tanh_and_mul",
    # ... 其他
}
```

```
# 无乘变体映射
_WITHOUT_MUL: dict[MoEActivation, MoEActivation] = {
    MoEActivation.GELU_TANH: MoEActivation.GELU_TANH_NO_MUL,
    # ...
}

def apply_moe_activation(activation, output, input):
    # ... 门控分支
    if activation == MoEActivation.GELU_TANH:
        torch.ops._C.gelu_tanh_and_mul(output, input) # 调用已注册的 C++ 算子
    # ... 非门控分支
    elif activation == MoEActivation.GELU_TANH_NO_MUL:
        output.copy_(F.gelu(input, approximate="tanh")) # 使用 PyTorch 的 tanh 近似
```

vllm/model_executor/models/gemma4.py

修复目标文件：将激活参数从 "gelu" 改为 "gelu_tanh"，使 MoE 层与密集层及官方配置一致。

```
# gemma4.py 中关键变更
self.experts = FusedMoE(
    num_experts=config.num_experts,
    top_k=config.top_k_experts,
    hidden_size=config.hidden_size,
    intermediate_size=...,
    renormalize=True,
    quant_config=quant_config,
    prefix=f"{prefix}.experts",
    custom_routing_function=routing_function,
    activation="gelu_tanh", # 修复前为 "gelu"
)
```

评论区精华

Gemini Code Assist 机器人提出 "gelu_tanh_and_mul" 自定义算子名称未在 activation.py 中注册，可能导致 LoRA 层查找失败。lucianommartins回应指出 custom_op_name 属性和 _CUSTOM_OP_NAMES 查找表在当前代码库中无任何消费（grep -rn "\.custom_op_name" vllm/ 返回零结果），属于历史残留；实际 MoE 激活通过 apply_moe_activation() 直接调用 torch.ops._C.gelu_tanh_and_mul，该算子已在 activation.py 中为 GeluAndMul(approximate="tanh") 注册，因此 LoRA 路径不会受影响。

- custom_op_name 注册不足可能影响 LoRA (correctness): lucianommartins 回应 custom_op_name 属性和 _CUSTOM_OP_NAMES 查找表在当前代码库中没有任何消费方（grep 无结果），属于历史残留。实际 MoE 激活通过 apply_moe_activation() 直接调用已注册的算子，LoRA 路径不受影响。

风险与影响

- 风险：低风险。变更集中在 MoE 激活枚举和允许列表的机械扩展，核心修复仅涉及 Gemma4 模型中的一个参数值更改。gelu_tanh_and_mul 自定义算子在 activation.py 中

已有注册（用于密集层），无需额外工作。数值差异约 2e-4（fp32），在 bf16 噪声范围内，对生成质量无实际影响。性能无变化，因为 gelu_tanh_and_mul 与 gelu_and_mul 速度相同。

- 影响：影响范围小：仅影响 Gemma4 模型用户，修复 MoE 层激活函数与官方规范一致性问题。不会影响其他模型或已有功能。向后兼容：from_str("gelu_pytorch_tanh") 通过别名支持。
- 风险标记：缺少 LoRA 兼容性显式验证，依赖已存在但未在 MoE 上下文测试的算子

关联脉络

- PR #39243 add: LFM2/2.5 Tool Parser: 无直接关联，但同为新增功能枚举变体，展示类似模式。