

PR #56446 完整报告

vllm-project/vllm

[Bugfix] Align vLLM YaRN with Transformers and stop re-scaling max_model_len

合并时间: 2026-09-11 21:01

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56446>

执行摘要

- 一句话: 对齐 vLLM 与 Transformers 的 YaRN, 修正 YaRN 变体 max_model_len 二次缩放
- 推荐动作: 值得精读, 尤其是三个设计决策: (1) 为什么用预计算 attention_factor 而不是 mscale: 0.7 (上游只在 mscale_all_dim 为真时才看 mscale, 否则会得到 1.1386 而非 1.0970); (2) 为什么把修复从“给别名打补丁”改为“对整个 YaRN 家族套用上游规则”, 并顺势删除重复类; (3) 为什么把 apply_yarn_scaling 重拼为 attention_factor 而不是保留别名——结果是 RoPE 字典的每个键都成为 Transformers 键, ignore_keys_at_rope_validation 这类 workaround 随之消失。阅读时应同时留意作者自述的验证缺口: Model evaluation — Not yet run — needs a GPU run before merge 并未在合并前补齐。若下游确实在生产使用 TeleChat3、sarvam-105b 或任何带 mscale/mscale_all_dim 的 YaRN 模型, 建议自行补一次长上下文精度与输出一致性回归, 并确认新默认 max_model_len 与现有部署配置兼容。

功能与动机

PR body 指出两个可直接观察的错误推导: Tele-AI/TeleChat3-36B-Thinking (rope_type telechat3-yarn) 的 max_model_len 从 131,072 被算成 32,768 的反面——旧代码把它推到 131,072, 而实际 config 已经是 32,768×4; sarvamai/sarvam-105b (deepseek_yarn, max_position_embeddings = 131072, factor = 40) 被推导出 5,242,880, 远超其 cos/sin cache 长度, 请求可以索引到 cache 之外。作者把根因归结为“与 transformers.modeling_rope_utils._compute_yarn_parameters 的漂移 (drift)”, 并说明这正是每个 vendor 别名都需要单独打补丁的原因, 因此选择一次性对齐上游规则: 上游从不对 YaRN 把 max_position_embeddings 乘 factor, 上游也只有一份 YaRN 实现而非每个 vendor 一个类。PR 还明确 supersede 了 #55116 (把 telechat3-yarn 加入排除元组) 与 #56279 (加入 deepseek_yarn), 并致谢 @leex404 与 @pushkar-sarvam。

实现拆解

- 修正派生上下文长度 (vllm/config/model.py::_get_and_verify_max_len): 把 factor 累积的排除元组从 ("su", "longrope", "llama3") 扩展为再包含 "yarn"、"deepseek_yarn"、"deepseek_llama_scaling", 并删除原先 if rope_type == "yarn": derived_max_model_len = rp["original_max_position_embeddings"] 的特例。理由是与上游一致: 对 YaRN 家族而言 max_position_embeddings 本身就是最终上下文长度, 再乘 factor 会高估上限。影响面

覆盖所有依赖 `derived_max_model_len` 的启动校验、KV cache 容量估算与 `--max-model-len` 默认值。

模型	rope_type	变更前推导值	变更后推导值
Tele-AI/TeleChat3-36B-Thinking	telechat3-yarn	131,072	32,768
sarvamai/sarvam-105b	deepseek_yarn	5,242,880	131,072

1. 把 `vendor` 别名归一化为标准 YaRN (`vllm/transformers_utils/config.py::_patch_legacy_rope_type`)：新增分支，把 `rope_type == "telechat3-yarn"` 改写为 `"yarn"`，并预计算 `attention_factor = 0.07 * math.log(factor) + 1.0`（同时 `pop` 掉遗留的 `type` 键），为此新增 `import math`。这样 TeleChat3 会自动落入所有基于 `"yarn"` 的判断（包括第 1 步的排除元组），也是删除重复 RoPE 类的前提。
2. 删除重复实现并收敛分派逻辑 (`vllm/model_executor/layers/rotary_embedding/telechat3_scaling_rope.py` 删除、`rotary_embedding/__init__.py`)：整文件删除 57 行的 `TeleChat3RoPEScaledRotaryEmbedding`（其 `get_mscales` 只在系数上把 0.1 换成 0.07），同步移除导入与 `get_rope` 里 `scaling_type == "telechat3-yarn"` 分支；`yarn` 分支的 `extra_kwargs` 白名单改为 `beta_fast/beta_slow/mscale/mscale_all_dim/attention_factor/truncate`，并删掉 `extra_kwargs.pop("apply_yarn_scaling", None)` 这个 `mrope` 专用 workaround。
3. 统一 YaRN attention 缩放语义 (`yarn_scaling_rope.py`、`mrope.py`、`deepseek_scaling_rope.py`、`common.py`)：`YaRNScalingRotaryEmbedding.__init__` 采用与 `_compute_yarn_parameters` 相同的优先级——`attention_factor` 优先，其次在 `mscale` 与 `mscale_all_dim` 同时为真时取两者比值，最后回退 `yarn_get_mscales(factor)`；`MRotaryEmbedding` 采用同一套解析以覆盖 `mrope+yarn`（如 Qwen2.5-VL 系）；`DeepseekScalingRotaryEmbedding` 去掉 `mscale` 上的 `* attn_factor`；`common.py::yarn_get_mscales(scale, mscale=1)` 增加系数形参。所有 RoPE 类的 `extrapolation_factor/attn_factor` 被删除，`inv_freq_mask` 不再乘以 `extrapolation_factor`（默认 1，等价）。
4. Mistral YaRN 参数映射改写 (`vllm/transformers_utils/configs/mistral.py::_remap_mistral_yarn_args`)：从映射表移除 `apply_scale` → `apply_yarn_scaling`，改为 `if not yarn_config.pop("apply_scale", True): rope_parameters["attention_factor"] = 1.0`；同时删除 `packaging.version` 与 `TRANSFORMERS_VERSION` 导入以及 `ignore_keys_at_rope_validation` 的版本分支 hack。作者核对后指出 `mistralai/Mistral-Large-3` 上游算 1.3583 而 `vLLM` 算 1.0，改后两边都写 1.0 且 `vLLM` 数值不变。
5. 测试配套：`tests/test_config.py` 新增参数化用例 `test_get_and_verify_max_len_yarn_is_already_scaled`（覆盖 `yarn/deepseek_yarn/deepseek_llama_scaling`，并以 `linear` 作为仍然乘 `factor` 的对照组，全部为直接调用不走网络）；`tests/transformers_utils/test_config.py` 新增 `test_patch_legacy_rope_type_normalizes_telechat3_yarn` 验证归一化。作者声明的测试计划还包括 `tests/kernels/core/test_mrope.py`、`test_pos_encoding.py` 与 `pre-commit run --all-files`，并指出新用例在无修复时会失败（`assert 1310720 == 32768`）。

关键文件：

- `vllm/config/model.py`（模块 模型配置；类别 source；类型 core-logic；符号 `_get_and_verify_max_len`）：核心 bugfix 所在：`_get_and_verify_max_len` 的 YaRN 排除元组扩展与 "yarn" 特例删除，直接决定 TeleChat3 / sarvam-105b 的默认 `max_model_len`。
- `vllm/transformers_utils/config.py`（模块 配置适配；类别 source；类型 data-contract；符号 `_patch_legacy_rope_type`）：`_patch_legacy_rope_type` 新增 telechat3-yarn → yarn 归一化并预计算 `attention_factor`，是删除重复 RoPE 类、让 TeleChat3 落入所有 "yarn" 判断的前提。
- `vllm/model_executor/layers/rotary_embedding/yarn_scaling_rope.py`（模块 位置编码；类别 source；类型 data-contract；符号 `YaRNScalingRotaryEmbedding.init`, `YaRNScalingRotaryEmbedding._compute_inv_freq`）：`YaRNScalingRotaryEmbedding` 的 `mscale` 解析重写为与 Transformers `_compute_yarn_parameters` 逐字对齐，是本次参数契约变更的核心落点。
- `vllm/model_executor/layers/rotary_embedding/telechat3_scaling_rope.py`（模块 位置编码；类别 source；类型 deletion；符号 `TeleChat3RoPEScaledRotaryEmbedding`, `init`, `get_mscale`）：整文件删除 57 行的 `TeleChat3RoPEScaledRotaryEmbedding`，是“去重复实现、统一 YaRN”这一设计方向的直接体现。
- `vllm/model_executor/layers/rotary_embedding/__init__.py`（模块 编码分派；类别 source；类型 data-contract；符号 `get_rope`）：`get_rope` 分派表：移除 telechat3-yarn 分支与导入，yarn 的 `extra_kwargs` 白名单改为 Transformers 键，是全链路收敛的入口。
- `vllm/model_executor/layers/rotary_embedding/mrope.py`（模块 位置编码；类别 source；类型 data-contract；符号 `MRotaryEmbedding.init`）：`MRotaryEmbedding` 复用同一套 YaRN `mscale` 解析，影响 mrope+yarn（如 Qwen2.5-VL 系）模型的 attention 缩放。
- `vllm/model_executor/layers/rotary_embedding/deepseek_scaling_rope.py`（模块 位置编码；类别 source；类型 cleanup；符号 `DeepseekScalingRotaryEmbedding.init`, `DeepseekScalingRotaryEmbedding._compute_inv_freq`）：`DeepseekScalingRotaryEmbedding` 去掉 `attn_factor/extrapolation_factor` 乘子，直接作用于 DeepSeek/Kimi 等走 `deepseek_yarn` 的 MLA 模型。
- `vllm/transformers_utils/configs/mistral.py`（模块 模型适配；类别 source；类型 configuration；符号 `_remap_mistral_yarn_args`）：`_remap_mistral_yarn_args` 把 `apply_scale: false` 改写为显式 `attention_factor = 1.0`，并删除版本相关的 `ignore_keys_at_rope_validation` 分支。
- `vllm/model_executor/layers/rotary_embedding/common.py`（模块 位置编码；类别 source；类型 data-contract；符号 `yarn_get_mscale`）：`yarn_get_mscale` 增加 `mscale` 形参，是 YaRN/DeepSeek 两类实现共享的缩放公式基础改动。
- `tests/test_config.py`（模块 配置测试；类别 test；类型 test-coverage；符号 `test_get_and_verify_max_len_yarn_is_already_scaled`）：新增参数化回归用例 `test_get_and_verify_max_len_yarn_is_already_scaled`，直接锁定「YaRN 家族不再乘 factor」这一行为契约。

- tests/transformers_utils/test_config.py (模块 配置测试; 类别 test; 类型 test-coverage ; 符号 test_patch_legacy_rope_type_normalizes_telechat3_yarn) : 新增 test_patch_legacy_rope_type_normalizes_telechat3_yarn, 验证 vendor 别名的归一化与 attention_factor 预计算。

关键符号: _get_and_verify_max_len, _patch_legacy_rope_type, YaRNScalingRotaryEmbedding.init, MRotaryEmbedding.init, DeepseekScalingRotaryEmbedding.init, yarn_get_mscale, _remap_mistral_yarn_args, get_rope, TeleChat3RoPEScaledRotaryEmbedding (删除) , test_get_and_verify_max_len_yarn_is_already_scaled, test_patch_legacy_rope_type_normalizes_telechat3_yarn

关键源码片段

vllm/config/model.py

核心 bugfix 所在: `_get_and_verify_max_len` 的 YaRN 排除元组扩展与 "yarn" 特例删除, 直接决定 TeleChat3 / sarvam-105b 的默认 `max_model_len`。

```
# vllm/config/model.py —— _get_and_verify_max_len() 中 RoPE 缩放相关的核心片段
# 语义: YaRN 家族 (yarn / deepseek_yarn / deepseek_llama_scaling) 的
# max_position_embeddings 已经是最终上下文长度, 不能再乘一次 factor。
```

```
if rope_parameters is not None and "gemma3" not in hf_config.model_type:
    scaling_factor = 1.0
    for rp in rope_parameters.values():
        # 加载 HF config 时已经过 patch_rope_parameters, 这里无需再看 "type" 键
        rope_type = rp["rope_type"]

        # 与 Transformers 的 _compute_yarn_parameters 保持一致:
        # 上游从不把 max_position_embeddings 乘以 factor,
        # 因此整个 YaRN 家族都排除在 factor 累积之外。
        # 旧实现只排除精确字符串 "yarn", 导致 telechat3-yarn /
        # deepseek_yarn 的 max_position_embeddings 被二次缩放。
        if rope_type not in (
            "su",
            "longrope",
            "llama3",
            "yarn",
            "deepseek_yarn",
            "deepseek_llama_scaling",
        ):
            # rope_type == "default" 不定义 factor;
            # 这里假设所有层类型共享同一个 scaling factor
            scaling_factor = rp.get("factor", scaling_factor)
```

```
if scaling_factor is None:
    # 用户在 config 里显式写了 null 时回退到 1.0, 并提示可能是模型侧 bug
    logger.warning_once(
```

```

        "The model's RoPE configuration has a null scaling factor ..."
    )
    scaling_factor = 1.0

    # 在循环外统一乘一次，避免多层类型重复相乘
    derived_max_model_len *= scaling_factor

    # 未显式指定 max_model_len 时以它作为默认值；
    # LongRoPE 单独回退到 original_max_position_embeddings，避免短序列性能退化
    if max_model_len is None or max_model_len == -1:
        if rope_parameters is not None and any(
            rp["rope_type"] == "longrope" for rp in rope_parameters.values()
        ):
            max_model_len = int(
                getattr(
                    hf_config, "original_max_position_embeddings", derived_max_model_len
                )
            )
        else:
            max_model_len = int(derived_max_model_len)

```

vllm/transformers_utils/config.py

`_patch_legacy_rope_type` 新增 `telechat3-yarn` → `yarn` 归一化并预计算 `attention_factor`，是删除重复 RoPE 类、让 TeleChat3 落入所有 `"yarn"` 判断的前提。

```

# vllm/transformers_utils/config.py —— _patch_legacy_rope_type() 新增分支
def _patch_legacy_rope_type(rope_parameters: dict[str, Any]) -> None:
    """把模型侧遗留的 rope_type 归一化为 Transformers 认识的标准值。"""

    if rope_parameters["rope_type"] == "mrope":
        # 历史写法: mrope 等价于 default
        rope_parameters["rope_type"] = "default"
        logger.warning("Replacing legacy rope_type 'mrope' with 'default'")

    elif rope_parameters["rope_type"] == "telechat3-yarn":
        # TeleChat3 是 YaRN 的 vendor 副本: 只把 attention scaling 系数
        # 从 0.1 改成 0.07。这里预先算出 attention_factor，配置就等价于
        # 标准 YaRN，因此 vLLM 不必再维护一份重复的 RoPE 类，
        # TeleChat3 也会自动落入所有基于 "yarn" 的判断分支。
        #
        # 注意: 写成 mscale=0.7 无法 round-trip —— 上游只有在
        # mscale_all_dim 同时为真时才读 mscale，会算出 1.1386
        # 而不是预期的 0.07 * log(4) + 1 = 1.0970。
        factor = rope_parameters["factor"]
        rope_parameters["rope_type"] = "yarn"
        rope_parameters["attention_factor"] = 0.07 * math.log(factor) + 1.0
        # 清掉旧式 "type" 键，避免下游重复解析
        rope_parameters.pop("type", None)
        logger.warning("Replacing rope_type 'telechat3-yarn' with 'yarn'")

```

之后仍会继续处理 interleaved sliding attention 的嵌套 rope_parameters

vllm/model_executor/layers/rotary_embedding/yarn_scaling_rope.py

YaRNScalingRotaryEmbedding 的 mscale 解析重写为与 Transformers
_compute_yarn_parameters 逐字对齐，是本次参数契约变更的核心落点。

```
class YaRNScalingRotaryEmbedding(RotaryEmbedding):
```

```
    """RotaryEmbedding extended with YaRN method.
```

```
    Credits to Peng et al. github.com/jquesnelle/yarn
```

```
    """
```

```
    def __init__(
```

```
        self,
```

```
        head_size: int,
```

```
        rotary_dim: int,
```

```
        max_position_embeddings: int,
```

```
        base: float,
```

```
        is_neox_style: bool,
```

```
        scaling_factor: float,
```

```
        dtype: torch.dtype,
```

```
        *,
```

```
        beta_fast: int = 32,
```

```
        beta_slow: int = 1,
```

```
        mscale: float | None = None,
```

```
        mscale_all_dim: float | None = None,
```

```
        attention_factor: float | None = None,
```

```
        truncate: bool = True,
```

```
    ) -> None:
```

```
        self.scaling_factor = scaling_factor
```

```
        self.beta_fast = beta_fast
```

```
        self.beta_slow = beta_slow
```

```
        self.truncate = truncate
```

```
    # attention 缩放的优先级与 Transformers 的 _compute_yarn_parameters 一致:
```

```
    # 1) 显式 attention_factor 优先 (Mistral 的 apply_scale: false 会映射成 1.0) ;
```

```
    # 2) 否则当 mscale 与 mscale_all_dim 同时为真时取两者比值;
```

```
    # 3) 最后回退到标准 YaRN 的 yarn_get_mscale(factor)。
```

```
    # 旧的 attn_factor / apply_yarn_scaling 已删除 —— 它们不是 Transformers
```

```
    # 的 RoPE 参数，任何 HF config 都无法携带。
```

```
    if attention_factor is not None:
```

```
        self.mscale = float(attention_factor)
```

```
    elif mscale and mscale_all_dim:
```

```
        self.mscale = float(
```

```
            yarn_get_mscale(self.scaling_factor, mscale)
```

```
            / yarn_get_mscale(self.scaling_factor, mscale_all_dim)
```

```
        )
```

```
    else:
```

```
        self.mscale = float(yarn_get_mscale(self.scaling_factor))
```

```

super().__init__(
    head_size, rotary_dim, max_position_embeddings, base, is_neox_style, dtype
)

def _compute_inv_freq(self, scaling_factor: float) -> torch.Tensor:
    pos_freqs = self.base ** (
        torch.arange(0, self.rotary_dim, 2, dtype=torch.float) / self.rotary_dim
    )
    inv_freq_extrapolation = 1.0 / pos_freqs
    inv_freq_interpolation = 1.0 / (scaling_factor * pos_freqs)

    low, high = yarn_find_correction_range(
        self.beta_fast,
        self.beta_slow,
        self.rotary_dim,
        self.base,
        self.max_position_embeddings,
        self.truncate,
    )
    # 不再乘以 extrapolation_factor: 该旋钮在 vLLM 中无人设置,
    # 删除后与原实现等价 (乘 1)
    inv_freq_mask = 1 - yarn_linear_ramp_mask(
        low, high, self.rotary_dim // 2, dtype=torch.float
    )
    inv_freq = (
        inv_freq_interpolation * (1 - inv_freq_mask)
        + inv_freq_extrapolation * inv_freq_mask
    )
    return inv_freq

```

评论区精华

本 PR 没有任何行内 review 评论 (`review_comments_count = 0`)，讨论信息几乎全部来自 PR body 与合并记录：

hmellor (PR body, 作者自评风险) : Model evaluation — Not yet run — needs a GPU run before merge. ... This touches attention scaling and derived context length, so TeleChat3 and an MLA model on the deepseek_yarn path should be evaluated before this lands; treat it as unvalidated end-to-end until those results are posted here.

hmellor (PR body, 关于打包清理的正当性) : The exact-string check was a symptom of drift from transformers.modeling_rope_utils._compute_yarn_parameters, which is why each vendor alias needed its own special case.

hmellor (PR body, 关于 mscale 无法 round-trip) : TeleChat3 uses a precomputed attention_factor rather than mscale: 0.7, which looks natural but does not round-trip: upstream only consults mscale when mscale_all_dim is also truthy, so it would compute 1.1386 where vLLM computed 1.0970.

claude[bot]: This pull request is from a fork — automated review is disabled. 即自动 review 未生效，没有被机器审查覆盖。

审阅记录仅两条空 body 的 APPROVED (noooooop、Isotr0py)，随后由作者本人 merged_by: hmellor 合并，Issue 评论只有 /ci run 与 Buildkite 触发回执。也就是说，“合并前端到端必须验证”这一自我要求没有在 PR 内留下任何结果记录，属于未闭环的讨论。

- 合并前端到端精度验证缺口（作者自述） (testing): 作者只提供了单元级等价性论证（`cos_sin_cache torch.equal`、`mscale = 0.07*log(4)+1 = 1.097040605278392`、`inv_freq` 与 `ROPE_INIT_FUNCTIONS["yarn"]` 逐位一致），自己在未补评测的情况下合并；该验证缺口未闭环。
- attention_factor 预计算 vs `mscale=0.7` 的 round-trip 陷阱 (design): 采用 `attention_factor` 作为唯一表达，成为 `apply_scale: false` (→1.0) 与 TeleChat3 (→0.07 系数) 共用的语义出口。
- 把 per-vendor 特例泛化为整个 YaRN 家族 (design): 泛化方案被采纳并合并，两个既有 PR 被取代；但本质上仍是枚举式白名单，未注册的第三方别名仍可能被二次缩放。
- 删除 `attn_factor` / `extrapolation_factor` / `apply_yarn_scaling` 三个参数 (design): 三个 kwargs 被移除，RoPE 字典的每个键都成为具有上游含义的 Transformers 键；对下游直接实例化这些类的代码构成潜在破坏性变更。
- 自动化 review 未生效 (other): 本 PR 没有机器审查覆盖，只有两名人工审阅者给出无正文的 APPROVED。

风险与影响

- 风险：
 1. 核心路径语义变更 (`_get_and_verify_max_len`)：默认 `max_model_len` 对 TeleChat3-36B-Thinking (131,072 → 32,768) 与 sarvam-105b (5,242,880 → 131,072) 大幅下调。这是修复越界索引的有意行为变更，但对已依赖旧默认值的部署（例如按旧值配置显存预算或直接暴露长上下文）会表现为启动报错或需要显式 `--max-model-len`，属于用户可见的兼容性变化。
 2. 白名单枚举仍有遗漏风险：修复方式是继续枚举“已知 YaRN 别名”，而非按前缀 / 语义识别 YaRN 家族。任何未注册的新 vendor 别名（第三方私有 config 常见）仍会落入 `scaling_factor = rp.get("factor")` 分支并被二次缩放。
 3. RoPE 参数契约变更：`extrapolation_factor`、`attn_factor`、`apply_yarn_scaling` 三个 kwargs 从 YaRNScalingRotaryEmbedding、MRotaryEmbedding、DeepseekScalingRotaryEmbedding 中删除。仓库内已无调用方（作者声明为 dead），但下游 fork、插件或自定义模型若直接实例化这些类并传参，将直接 TypeError。
 4. attention 缩放数值路径变化：`mscale/mscale_all_dim` 从“被忽略”变为“真正生效”，携带这两个键的 HF config (`mrope+yarn`、`deepseek_yarn` 路径) 输出数值会变化。作者只做了分析与 `torch.equal` 级别的单元对比，没有端到端精度评测。
 5. Mistral 分支的版本条件被删除：原先按 `TRANSFORMERS_VERSION >= 5.3.0.dev0` 决定是否写 `ignore_keys_at_rope_validation`，现在统一改为 `attention_factor`。对新旧 Transformers 版本的交叉兼容性缺少显式测试覆盖。

6. `telechat3-yarn` 只在校验前的 `config patch` 阶段被归一化：若有路径绕过 `_patch_legacy_rope_type`（例如直接构造 `rope_parameters` 的第三方代码），`get_rope` 会抛出 `Unknown RoPE scaling type telechat3-yarn`，因为专用类与分支都已删除。
7. 测试层次偏单元：两个新增用例都是直接调用、无网络、无 GPU，验证的是推导值与归一化，无法覆盖“`cos/sin cache` 是否被越界索引”“`attention scaling` 是否与预期精度一致”这类端到端后果。 - 影响：用户侧：TeleChat3-36B-Thinking 与 `sarvam-105b` 的默认上下文长度被修正到与 HF config 一致的安全值，越界索引 `cos/sin cache` 的隐患被消除；Mistral-Large-3 的 `attention factor` 语义与上游对齐（vLLM 数值不变），TeleChat3 不再需要 `vendor` 专用 RoPE 类。系统侧：`get_rope` 的 `yarn` 分支、`MRotaryEmbedding` 与 `DeepseekScalingRotaryEmbedding` 的参数集统一到 Transformers 词汇表，`vllm/transformers_utils/configs/mistral.py` 少了一个版本判断分支与一个校验 `workaround`，`_ROPE_DICT` 相关实现去掉了重复类；代价是 `mrope+yarn`（Qwen2.5-VL 类）与 `deepseek_yarn`（DeepSeek/Kimi MLA 类）的数值路径发生变化。团队侧：后续处理新的 YaRN `vendor` 别名有了固定套路——在 `_patch_legacy_rope_type` 中预计算 `attention_factor` 归一化为 `yarn`，而不是新增子类；同时维护 `_get_and_verify_max_len` 的排除元组成为隐性约定。若生产上运行受影响的模型，需要重新确认 `--max-model-len`、显存预算与长上下文评测基线。 - 风险标记：核心路径变更，默认 `max_model_len` 下调，端到端精度未验证，RoPE 参数契约变更，枚举白名单仍有遗漏风险，缺少下游迁移说明

关联脉络

- PR #55116 被本 PR 取代：telechat3-yarn 排除元组修复（标题未在上下文中提供）：PR body 写明本 PR `supersedes #55116` —— 该 PR 只把 `telechat3-yarn` 加入排除元组，本 PR 泛化为整个 YaRN 家族并删除其专用 RoPE 类，覆盖了同一问题。
- PR #56279 被本 PR 取代：deepseek_yarn max_model_len 修复（标题未在上下文中提供）：PR body 写明本 PR `supersedes #56279` —— 该 PR 为 `deepseek_yarn` 单独加排除项，本 PR 用上游规则一次性覆盖 `yarn/deepseek_yarn/deepseek_llama_scaling`。
- PR #56214 [Model] Support DeepSeek-V4.1-Flash: 同一功能线：本 PR 修改的 `deepseek_scaling_rope.py` 与 `_get_and_verify_max_len` 直接影响 DeepSeek MLA 系列（含 V4 家族）的 RoPE 缩放与派生上下文长度，且该历史 PR 也改动了 `vllm/v1/core/kv_cache_utils.py` 等与上下文 /KV 容量相关的路径。
- PR #56433 [ROCm][Bugfix] Fix AITER preshuffled FP8 block-scale kernel: 同一类问题：两者都修正 MLA/DeepSeek 路径上的静默数值错误（前者是 FP8 kernel 精度、本 PR 是 YaRN `attention scaling` 与上下文长度），说明该路径近期是精度回归高发区，值得合起来看。