# PR #6848 完整报告

verl-project/verl

[fsdp, megatron, trainer] fix: enhance mem footprint for forward_kl_topk OPD

合并时间：2026-07-01 05:51

原文链接：http://prhub.com.cn/verl-project/verl/pull/6848

## 执行摘要

- 一句话：OPD top-k 蒸馏跳过 log_probs 和 PPO 损失，降低显存占用
- 推荐动作：本 PR 设计精巧，通过一个标志串联 trainer→engine→loss 的多个模块，是典型的"配置驱动优化"案例。值得精读，尤其关注 FSDP engine 中操作顺序的修复和 Megatron 中 _distillation_use_topk_active 的传递方式。建议在后续工作中增加 end-to-end GPU 测试。

## 功能与动机

OPD（Online Policy Distillation）训练中，当仅使用 top-k 蒸馏损失（forward_kl_topk）且不启用 policy gradient 和 task rewards 时，原本仍然会计算全词表 log_probs 和 PPO 损失，造成不必要的显存和计算开销，在显存紧张的场景下容易 OOM。PR 作者希望通过引入 distillation_only 标志来跳过这些冗余操作。

## 实现拆解

1. Trainer 侧设置 distillation_only 标志：在 verl/trainer/ppo/ray_trainer.py 和 verl/trainer/ppo/v1/trainer_base.py 的 _update_actor 中，当 distillation_use_topk=True 且 use_task_rewards=False 且 use_policy_gradient=False 时，将 distillation_only 设为 True，并通过 batch_td 或 extra_info 传递给 engine。
2. Loss 函数跳过 PPO 计算：verl/trainer/distillation/losses.py 中 distillation_ppo_loss 函数增加判断：若 use_task_rewards=False 且 use_policy_gradient=False，则直接跳过 ppo_loss 调用，避免不必要的 loss 计算。
3. FSDP engine 条件跳过 log_probs：verl/workers/engine/fsdp/transformer_impl.py 的 prepare_model_outputs 从 batch 中读取 distillation_only 标志，在 fused kernel 和 eager 分支中均跳过 log_probs 的计算，同时修复了在 eager 分支中先调用 logits_processor_func 再计算 log_probs 的顺序问题，防止 log_probs 的 inplace 操作污染 logits。
4. Megatron engine 实现同样跳过逻辑：verl/workers/engine/megatron/transformer_impl.py 的 _lm_head_logits_processor 方法新增 distillation_only 参数，在蒸馏模式下直接跳过 vocab_parallel_log_probs_from_logits 的调用，仅保留 top-k 蒸馏所需的输出。此外，在 forward_backward_batch 中记录 _distillation_use_topk_active，并在 optimizer_step 前根据该标志主动调用 empty_cache()，缓解显存紧张。

5. 配套测试：新增 tests/workers/test_megatron_distillation_only_on_cpu.py，通过 patch 的方式验证 Megatron engine 的 _lm_head_logits_processor 在蒸馏模式下不调用 vocab_parallel_log_probs_from_logits，同时蒸馏 key 被正确传播。修改 tests/workers/test_distillation_topk_symmetry_on_cpu.py 以覆盖两种 padding 模式。

关键文件：

- verl/workers/engine/megatron/transformer_impl.py（模块 Megatron 引擎；类别 source；类型 core-logic；符号 _lm_head_logits_processor, logits_processor）：核心引擎修改：新增 _lm_head_logits_processor 方法（将原来内联逻辑提取为独立方法），并添加 distillation_only 参数以跳过 log_probs 计算；optimizer_step 前根据蒸馏标志主动清空缓存。
- tests/workers/test_megatron_distillation_only_on_cpu.py（模块 测试；类别 test；类型 test-coverage；符号 _make_engine_stub, _make_logits_processor, _proc, _run_logits_processor）：新增的单元测试文件，覆盖 Megatron engine 的蒸馏路径，验证在 distillation_only=True 时 log_probs 未被调用，蒸馏 key 被正确返回。通过 patch 避免 Megatron TP 依赖，可在 CPU 上运行。
- verl/workers/engine/fsdp/transformer_impl.py（模块 FSDP 引擎；类别 source；类型 core-logic）：FSDP 引擎修改：从 batch 读取 distillation_only 标志，在 fused kernel 和 eager 分支跳过 log_probs 计算，并修复了 logits_processor_func 与 log_probs 计算的顺序。
- verl/trainer/distillation/losses.py（模块 损失函数；类别 source；类型 core-logic）：Loss 函数调整：当 use_task_rewards=False 且 use_policy_gradient=False 时跳过 PPO 损失计算，避免不必要的计算。
- verl/trainer/ppo/ray_trainer.py（模块 Trainer；类别 source；类型 core-logic）：Ray trainer 中设置 distillation_only 标志，连接蒸馏配置与 engine。
- verl/trainer/ppo/v1/trainer_base.py（模块 Trainer；类别 source；类型 core-logic）：V1 trainer 同样设置 distillation_only 标志，保持两个 trainer 版本行为一致。
- tests/workers/test_distillation_topk_symmetry_on_cpu.py（模块 测试；类别 test；类型 test-coverage；符号 test_distillation_outputs_emitted_in_both_padding_modes）：已有测试文件调整，增加对两种 padding 模式的蒸馏输出验证。

关键符号：_lm_head_logits_processor, prepare_model_outputs, distillation_ppo_loss, _update_actor

## 关键源码片段

### verl/workers/engine/megatron/transformer_impl.py

核心引擎修改：新增 _lm_head_logits_processor 方法（将原来内联逻辑提取为独立方法），并添加 distillation_only 参数以跳过 log_probs 计算；optimizer_step 前根据蒸馏标志主动清空缓存。

```
def _lm_head_logits_processor(
    self,
    logits, label, temperature, *,
```

```
    calculate_sum_pi_squared: bool,
    calculate_entropy: bool,
    distillation_use_topk: bool,
    distillation_only: bool, # 新增：蒸馏模式下跳过 log_probs
    logits_processor_func: Callable,
    batch: TensorDict,
    data_format: str,
):
    # 温度处理
    logits.div_(temperature.unsqueeze(dim=-1).to(logits.dtype))
    ret = {}
    if calculate_sum_pi_squared:
        ret["sum_pi_squared"] = vocab_parallel_sum_pi_squared(logits)
    if calculate_entropy:
        # 需要 clone 以保留 logits 给后续操作
        logits_bak = logits.clone()
        if self.engine_config.entropy_from_logits_with_chunking:
            entropy = vocab_parallel_entropy_with_chunking(logits, chunk_size=...)
        else:
            entropy = vocab_parallel_entropy(logits)
        ret["entropy"] = entropy
    # 核心选择：蒸馏模式跳过 log_probs
    if not distillation_only:
        ret["log_probs"] = vocab_parallel_log_probs_from_logits(logits, label)
    # 始终调用蒸馏 logits processor
    if distillation_use_topk and logits_processor_func is not None:
        outputs = logits_processor_func(
            student_logits=logits,
            data=data,
            data_format=data_format
        )
        ret.update(outputs)
    return ret
```

## verl/workers/engine/fsdp/transformer_impl.py

FSDP 引擎修改：从 batch 读取 distillation_only 标志，在 fused kernel 和 eager 分支跳过 log_probs 计算，并修复了 logits_processor_func 与 log_probs 计算的顺序。

```
def prepare_model_outputs(self, output, output_args, micro_batch: TensorDict, logits_processor_
func):
    # ... 读取标志
    distillation_only = tu.get_non_tensor_data(data=micro_batch, key="distillation_only", default=
    False)
    # ...
    if use_fused_kernels:
        log_probs = None
        if not distillation_only:
            log_probs = output.log_probs.squeeze(0) # (total_nnz,)
        entropy_rmpad = output.entropy.squeeze(0) if calculate_entropy else None
```

```
    else:
        logits_rmpad = output.logits.squeeze(0)
        logits_rmpad.div_(temperature_rmpad.clamp(min=1e-8).unsqueeze(-1).to(logits_rmpad.
        dtype))
        # 先调用蒸馏 processor，防止 log_probs 的 inplace 操作污染 logits
        if distillation_use_topk:
            outputs = logits_processor_func(student_logits=logits_rmpad.unsqueeze(0), data=micro_
            batch)
            # ... 存储蒸馏输出
        log_probs = None
        if not distillation_only:
            # 再计算 log_probs（可能 inplace 修改 logits_rmpad，但蒸馏已提取完毕）
            log_probs = logprobs_from_logits(logits_rmpad, labels=input_ids_rmpad_rolled, inplace_
            backward=True)
        # ...
```

### verl/trainer/distillation/losses.py

Loss 函数调整：当 use_task_rewards=False 且 use_policy_gradient=False 时跳过 PPO 损失计算，避免不必要的计算。

```
def distillation_ppo_loss(config, distillation_config, model_output, data, dp_group):
    # ...
    distillation_loss_config = distillation_config.distillation_loss
    distill_loss, distill_metrics = distillation_loss(...)

    # 只有需要时才计算 policy loss
    if not distillation_loss_config.use_task_rewards and not distillation_loss_config.use_policy_
    gradient:
        policy_loss = 0.0
        policy_metrics = {}
    else:
        policy_loss, policy_metrics = ppo_loss(config, model_output, data, dp_group)
        if not distillation_loss_config.use_task_rewards:
            policy_loss = 0.0
    # 合并损失
    policy_loss += distill_loss * distillation_loss_coef
    return policy_loss, policy_metrics
```

## 评论区精华

Gemini Code Assist 在 review 中指出 FSDP engine 中存在潜在的顺序问题：当 calculate_entropy=False 时，logprobs_from_logits 会 inplace 修改 logits_rmpad，而后续 logits_processor_func 可能使用被污染后的 logits 计算蒸馏损失。作者回复"fixed"，并在实际代码中将 logits_processor_func 的调用提前到 log_probs 计算之前，确保蒸馏损失使用原始 logits。讨论结论为问题已解决，无未解决疑虑。

- FSDP engine 中 logits_processor_func 与 log_probs 的顺序问题 (correctness): 作者修复：将 logits_processor_func 的调用提前到 log_probs 计算之前，确保蒸馏损失使用原始 logits。

# 风险与影响

- 风险：
  1. FSDP engine 顺序修复：虽然已修复，但如果未来有新的 inplace 操作加入，可能再次引入类似问题。
  2. distillation_only 默认 False，现有行为完全兼容，但若用户错误组合配置（如同时设置 use_policy_gradient=False 但通过其他路径依赖 log_probs）可能产生微妙错误。
  3. Megatron 中的 empty_cache调用增加了运行时开销，但在蒸馏场景下利大于弊。
  4. 单元测试仅覆盖 CPU 和 mock，缺少真实 GPU 的端到端蒸馏测试，可能遗漏分布式下的通信问题。 - 影响：影响范围：此变更主要影响使用 OPD 蒸馏且 use_policy_gradient=False、use_task_rewards=False 的用户，他们能观察到显存下降和训练吞吐提升。其他用户完全不受影响（distillation_only 默认 False）。系统层面：FSDP 和 Megatron 引擎均有修改，覆盖两大主流后端。团队：降低蒸馏训练的资源门槛，便于在更小的 GPU 集群上开展蒸馏实验。 - 风险标记：核心路径变更，缺少端到端 GPU 测试，distillation_only 标志依赖配置正确性

# 关联脉络

- PR #6867 [fully_async, doc] fix: ignore temperature config for teacher prompt_logprobs and warn when non-default value is set: 均涉及蒸馏训练相关的修复与优化，属于同一功能线。