GSPO：从token级到序列级优化的范式转变

大家好，我是居丽叶。

今天来看看Qwen团队前段时间提出的**GSPO算法**。

这是一种强化学习后训练的新方法，可以看作是 DeepSeek 提出的 **GRPO** 的改良版。这个算法很可能成为后续的面试热点，也是回答"最近有没有关注前沿技术"这类问题的好素材。

> 🏕️ **GSPO比GRPO好在哪里？一句话总结：**GSPO的提出正是为了解决 GRPO 在长序列和 MoE 模型训练中容易失稳的问题。GSPO 将优化粒度从 **token 级** 提升到 **序列级**，基于序列似然来定义重要性比率，并在序列层面执行奖励与优化。这样一来，训练信号更加一致，显著提升了训练的稳定性与效率，尤其在大规模 MoE 架构中优势突出。

本文的目录如下：

为什么 GRPO 会不稳定?

在理解 GSPO 之前，我们先看看 GRPO 存在哪些缺陷。

在 Qwen3 和 Minimax-m1 的技术报告中，研究者们都指出：使用 GRPO 训练大模型时，常常会出现灾难性且不可逆的模型崩溃。其根本原因在于**重要性比率的误用**：

- **高方差噪声**：GRPO 在 token 级别计算重要性比率，用于修正新旧策略的偏差。但单个 token 的比率波动往往很大，训练中会引入高方差噪声；随着序列长度增加，这种噪声在链式乘积中逐步累积，问题愈发严重。
- **clip 机制放大问题**：在高方差的权重基础上再做 clip，会强行截断有效梯度，破坏梯度方向的一致性，相当于把本就抖动的信号进一步“压扁”，从而放大了训练不稳定性。
- **MoE 模型尤为脆弱**：在 MoE 架构中，同一个查询在不同训练轮次可能激活不同专家，导致重要性比率 $w_{i,t}(\theta)$ 出现剧烈震荡，更容易触发训练崩溃。

GSPO（Group Sequence Policy Optimization）的核心创新在于：基于整个回复序列的似然来计算重要性比率，并将同一提示下多个回答的奖励做归一化后作为优势值，从而保证**奖励**与**优化**在**序列级别**上保持一致。

实验结果表明，GSPO 在训练稳定性、效率和性能方面都显著优于 GRPO，并且能够稳定 MoE 模型的训练，而无需借助诸如 Routing Replay 这类复杂的稳定化技巧（后文会详细介绍）。

回忆下PPO和GPRO

定义策略模型为 π_θ ， $x \in \mathcal{D}$ 为查询集合，给定查询 x 和响应 y ，其似然表示为 $\pi_\theta(y|x) = \Pi_{t=1}^{|y|}\pi_\theta(y_t|x, y_{<t})$ ，其中 $|y|$ 表示响应中的token数，定义查询-响应对的奖励为 $r(x, y) \in [0, 1]$.

- **PPO**：使用旧策略 π_{old} 的样本更新当前策略 π_θ ，利用clip将策略更新限制在旧策略附近，目标函数为：

$$\mathcal{J}_{\mathrm{PPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\theta_{\mathrm{old}}}(\cdot|x)} \left[\frac{1}{|y|} \sum_{t=1}^{|y|} \min\left(w_t(\theta)\hat{A}_t,\ \mathrm{clip}\left(w_t(\theta),\, 1-\varepsilon,\, 1+\varepsilon\right)\hat{A}_t \right) \right]$$

其中token y_t 的重要性比率为 $w_t(\theta) = \dfrac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\theta_{\mathrm{old}}}(y_t \mid x, y_{<t})}$， 衡量新旧策略对同一个token的生成概率比率。优势 $\hat{A}_t$ 是由评估模型计算得到， ϵ 是clip范围。**PPO比较的是当下行动和过去的行动哪个更优**，PPO的核心问题在于过于依赖评估模型。

- **GRPO**：为同一查询生成一组回复，计算组内回复的相对优势，省去了评估模型：

$$\mathcal{J}_{\mathrm{GRPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, \{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\mathrm{old}}}(\cdot|x)} \left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \min\left(w_{i,t}(\theta)\hat{A}_{i,t},\ \mathrm{clip}\left(w_{i,t}(\theta),\, 1-\varepsilon,\, 1+\varepsilon\right)\hat{A}_{i,t} \right) \right]$$

其中 G 表示组的大小，重要性比率 $w_{i,t}(\theta) = \dfrac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_{\theta_{\mathrm{old}}}(y_{i,t} \mid x, y_{i,<t})}$ ，优势 $\hat{A}_{i,t} = \hat{A}_i = \dfrac{r(x, y_i) - \mathrm{mean}\left(\{r(x, y_i)\}_{i=1}^{G}\right)}{\mathrm{std}\left(\{r(x, y_i)\}_{i=1}^{G}\right)}$ ，也就是所有一个回复中的所有token有相同的优势值。**GRPO比较的是组内的相对优势，即高于平均水平的响应优势值为正。**GRPO虽然省略了评估模型，但仍然使用token级重要性采样。

重要性比率：token-level or sequence-level?

在大模型**强化学习训练**中，由于模型规模庞大、MoE 模型存在稀疏性、响应长度不断增长，往往需要采用 **大 batch 训练**。为了提高样本效率，大 batch 会被拆分成 mini-batch 做梯度更新，这就引入了 **off-policy 训练** —— 当前的响应来自旧策略 $\pi_{\theta_{old}}$ ，而非当前正在优化的策略 π_θ 。

因此，PPO 和 GRPO 都需要依赖 **clip 机制** 来缓解 off-policy 带来的偏差。然而，即使引入了 clip，GRPO 在长序列任务上仍然可能发生模型崩溃，根源就在于 **重要性比率的不合理定义**。

回顾：重要性采样的原理

重要性采样的目标是纠正分布不匹配。设**行为分布**为 π_{beh} ，**目标分布**为 π_{tar} ，有：

$$\mathbb{E}_{z\sim\pi_{\text{tar}}}\left[f(z)\right]=\mathbb{E}_{z\sim\pi_{\text{beh}}}\left[\frac{\pi_{\text{tar}}(z)}{\pi_{\text{beh}}(z)}f(z)\right]$$

当 $\frac{\pi_{\text{tar}}(z)}{\pi_{\text{beh}}(z)}$ 能准确反映分布差异时，这个估计是无偏的。

在实践中，我们通过从行为分布 π_{beh} 采样 N 个样本，并利用重要性比率加权，来近似估计目标分布下的期望。

而在GRPO中，单个 token 的重要性比率定义为：$w_{i,t}(\theta)=\frac{\pi_\theta(y_{i,t}\mid x,y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}\mid x,y_{i,<t})}$，这种定义有两个问题：

1. 它依赖于 **下一个 token 的概率分布**，并不能准确修正分布不匹配；
2. 单个 token 的比率波动极大，直接在梯度中引入了高方差噪声。

再加上 clip 机制在高方差权重上进行非线性压缩，会进一步放大噪声，最终可能导致训练彻底崩溃。实验也表明，这种崩溃具有不可逆性，即使回退 checkpoint、调整超参或扩充数据也难以恢复。

核心原则：优化目标应与奖励的单位一致

奖励是针对 **完整序列** 的，因此重要性比率也应该在序列层面定义。

正确的做法是使用序列级权重：$\frac{\pi_\theta(y|x)}{\pi_{\theta_{old}}(y|x)}$，它能直接衡量新旧策略在整个序列上的差异，从而与序列级奖励保持一致。GSPO 正是基于这一原则，把重要性比率和优化目标统一到序列级别。

GSPO

优化目标

GSPO 将优化粒度从 **token 级** 提升到 **序列级**，其目标函数为：

$$\mathcal{J}_{\text{GSPO}}(\theta)=\mathbb{E}_{x\sim\mathcal{D},\{y_i\}_{i=1}^{G}\sim\pi_{\text{old}}(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}\min\left(s_i(\theta)\hat{A}_i,\text{clip}\left(s_i(\theta),1-\epsilon,1+\epsilon\right)\hat{A}_i\right)\right]$$

其中优势依旧基于组内相对奖励：

$$\hat{A}_i=\frac{r(x,y_i)-\text{mean}\left(\{r(x,y_i)\}_{i=1}^{G}\right)}{\text{std}\left(\{r(x,y_i)\}_{i=1}^{G}\right)}$$

重要性比率定义在**序列似然**上，并引入长度归一化：

$$s_i(\theta)=\left(\frac{\pi_\theta(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)}\right)^{\frac{1}{|y_i|}}=\exp\left(\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\log\frac{\pi_\theta(y_{i,t}|x,y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|x,y_{i,<t})}\right)$$

由此可以看到：

- **序列级 clip**：GSPO 对完整响应计算重要性比率并进行裁剪，从而削弱离群样本对梯度估计的影响。

- **长度归一化**：在 log 比率上除以 $|y_i|$，消除了长短序列之间的尺度差异，避免长序列天然劣势，同时减少了方差。
- **尺度差异**：由于 GSPO 在**序列级别**裁剪，裁剪范围与 GRPO 的 **token 级别**存在数量级上的差别。

梯度计算

GSPO目标函数的梯度为（省略了clip）：

$$\nabla_\theta \mathcal{J}_{\text{GSPO}}(\theta) = \nabla_\theta \mathbb{E}_{x\sim\mathcal{D},\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \left[\frac{1}{G}\sum_{i=1}^{G} s_i(\theta)\widehat{A}_i\right] \tag{8}$$

$$= \mathbb{E}_{x\sim\mathcal{D},\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \left[\frac{1}{G}\sum_{i=1}^{G} s_i(\theta)\widehat{A}_i \cdot \nabla_\theta \log s_i(\theta)\right] \tag{9}$$

$$= \mathbb{E}_{x\sim\mathcal{D},\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \left[\frac{1}{G}\sum_{i=1}^{G} \left(\frac{\pi_\theta(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)}\right)^{\frac{1}{|y_i|}} \widehat{A}_i \cdot \frac{1}{|y_i|}\sum_{t=1}^{|y_i|} \nabla_\theta \log \pi_\theta(y_{i,t}|x, y_{i,<t})\right]. \tag{10}$$

公式8到9的原理是：

$$\nabla_\theta f(\theta) = f(\theta)\nabla_\theta log f(\theta)$$

与之对比的是GRPO目标函数的梯度为（$\hat{A}_{i,t} = \hat{A}_i$）：

$$\nabla_\theta \mathcal{J}_{\text{GRPO}}(\theta) = \nabla_\theta \mathbb{E}_{x\sim\mathcal{D},\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|} w_{i,t}(\theta)\hat{A}_{i,t}\right]$$

$$= \mathbb{E}_{x\sim\mathcal{D},\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \left[\frac{1}{G}\sum_{i=1}^{G}\hat{A}_i \cdot \frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\left(\frac{\pi_\theta(y_{i,t}|x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|x, y_{i,<t})}\nabla_\theta \log \pi_\theta(y_{i,t}|x, y_{i,<t})\right)\right]$$

GSPO和GRPO的根本区别在于对token的对数似然梯度加权方式不同。

GRPO：

- 同一响应中所有 token 共享相同优势值 $\hat{A}_i$，但每个 token 的权重 $w_{i,t}(\theta)$ 差异很大。
- 在奖励为正时，部分 token 的权重可能接近 0，几乎不更新；奖励为负时，某些 token 的权重可能过大，导致更新过度。
- 这种 **token 间差异** 会在训练过程中不断累积，带来不可预测性，容易导致发散。

GSPO：

- 同一响应内的所有 token 使用相同的序列级权重 $s_i(\theta)$。
- 对于优秀的响应，所有 token 都会得到同等强度的正向更新，从而保证优化方向一致；对于差的响应，所有 token 都会被整体削弱。
- 这种一致性消除了 token 间的差异，使训练更稳定可控。

🎹 **GRPO 的更新粒度仍然停留在 token 级别。**
虽然 GRPO 的奖励和优势函数都是基于 **完整序列**的，但优化时的单位却是单个 token，因为在梯度项 $\nabla_\theta \log \pi_\theta(y_{i,t}|x, y_{i,<t})$ 前面的权重是 token 级的重要性比率 $w_{i,t}(\theta)$ 。

这种**粒度不匹配**带来了两个问题：

1. **同一响应内的不一致性**：一个序列的所有 token 共享相同的优势值 $\hat{A}_i$ ，但不同 token 的重要性比率可能差异很大。结果是奖励信号和优化方向被割裂。
2. **高方差噪声累积**：token 级别的比率波动过大，不仅不能有效修正新旧策略的分布差异，反而引入额外噪声；随着序列长度增加，这种噪声会不断累积，最终可能导致训练发散。

在 **MoE 模型** 中，这一问题尤为突出，因为相同查询在不同训练轮次可能激活不同专家，使得重要性比率 $w_{i,t}(\theta)$ 出现剧烈震荡，更容易触发模型崩溃。

Token-level变体

在多轮强化学习场景中，有时需要比序列级更精细的优势调整（因为前面某一步可能对后续产生深远影响）。为此，作者提出了 **GSPO 的 token-level 变体**。

其优化目标为：

$$\mathcal{J}_{\text{GSPO-token}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\left(s_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\left(s_{i,t}(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_{i,t}\right)\right]$$

其中 token 级的重要性比率定义为： $s_{i,t}(\theta) = \text{sg}\left[s_i(\theta)\right] \cdot \frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\text{sg}\left[\pi_\theta(y_{i,t} \mid x, y_{i,<t})\right]}$

$sg[\cdot]$ 表示取数值但停止梯度传播（即Pytorch 中的detach），进一步推导可得 GSPO-token 的梯度为：

$$\begin{aligned}
\nabla_\theta \mathcal{J}_{\text{GSPO-token}}(\theta) &= \nabla_\theta \mathbb{E}_{x\sim\mathcal{D},\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}s_{i,t}(\theta)\hat{A}_{i,t}\right] \\
&= \mathbb{E}_{x\sim\mathcal{D},\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}s_i(\theta)\cdot\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\hat{A}_{i,t}\frac{\nabla_\theta\pi_\theta(y_{i,t}|x, y_{i,<t})}{\pi_\theta(y_{i,t}|x, y_{i,<t})}\right] \\
&= \mathbb{E}_{x\sim\mathcal{D},\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}\left(\frac{\pi_\theta(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)}\right)^{\frac{1}{|y_i|}}\cdot\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\hat{A}_{i,t}\nabla_\theta\log\pi_\theta(y_{i,t}|x, y_{i,<t})\right].
\end{aligned}$$

这样设计GSPO-token有以下好处：

- **数值等价性**：对比GSPO和GSPO-token的梯度、优化目标和clip条件，作者发现当一个序列中所有token的优势都相同时（即 $\hat{A}_{i,t} = \hat{A}_i$ ），二者在数值上完全一致。具体表现为：
  - 序列重要性比率通过**长度归一化**将不同长度响应的 $s_i(\theta)$ 控制在统一的数值范围内。

- $\frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\text{sg}\left[\pi_\theta(y_{i,t} \mid x, y_{i,<t})\right]} = 1$，因此有 $s_{i,t}(\theta) = s_i(\theta)$。

- **细粒度优势控制**： $s_{i,t}(\theta)$ 可以对每个token进行定制化调整，拥有比序列级重要性比率更细粒度的优势控制能力。
- **稳定性和灵活性平衡**：GSPO-token既保持了为每个token定制优势的灵活性，还通过梯度停止 $sg[\cdot]$ 拥有GSPO的稳定性。

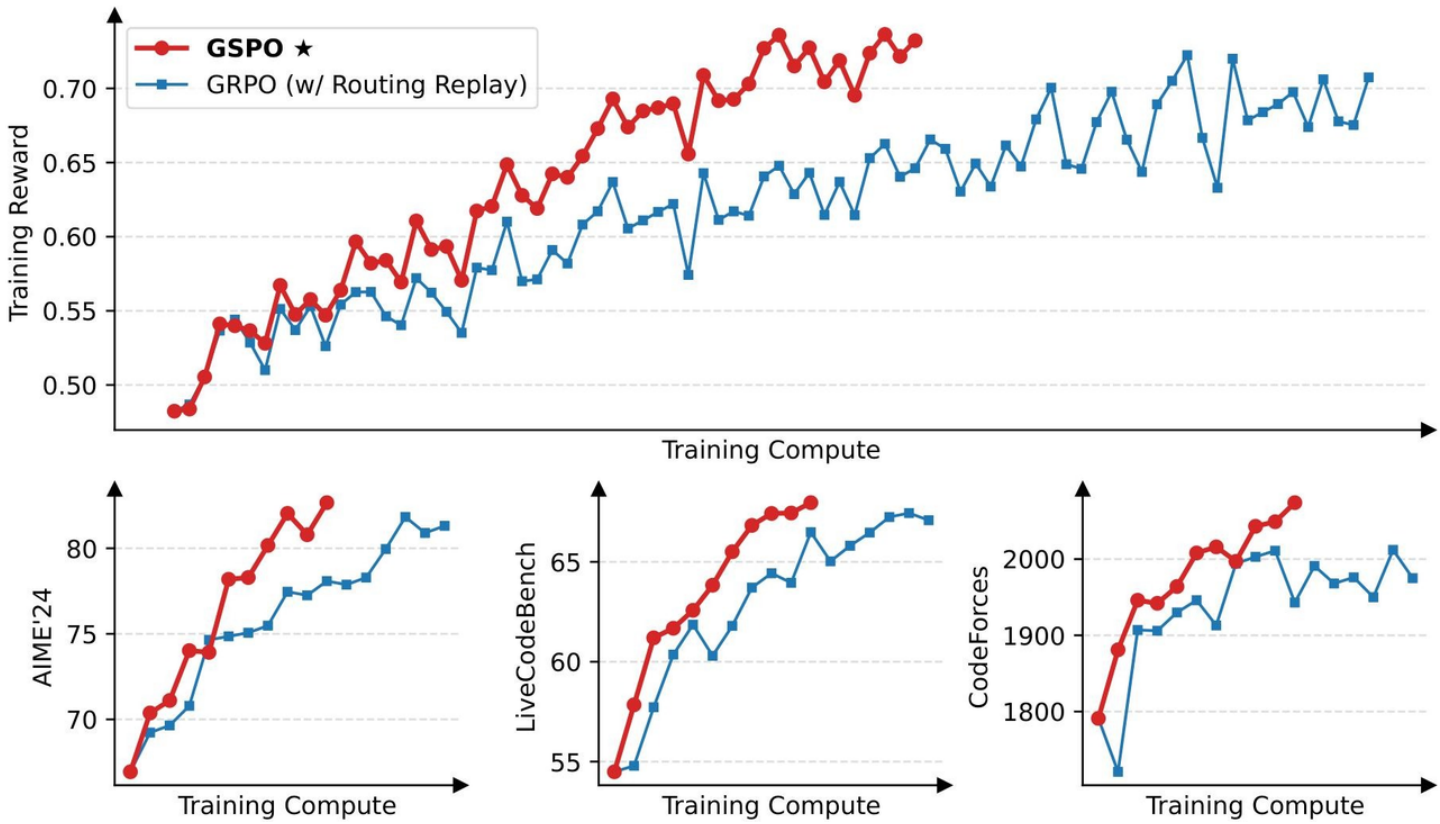
实验验证

训练稳定性与效率

研究者在从 **Qwen3-30B-A3B-Base** 微调得到的冷启动模型上开展实验，评估任务包括 AIME’24（数学）、LiveCodeBench（编程）和 CodeForces（编程）。训练设置为：每个 batch 划分为 4 个 mini-batch，GSPO 的 clip 范围为 $3e-4$和$4e-4$，GRPOclip范围为 0.2和0.27 （clip-higher），并在 GRPO 中启用了 **Routing Replay**。

结果表明：

- **GSPO 训练稳定**，性能能够随算力提升、查询集更新、生成长度延长而持续爬升；
- **GSPO 效率更高**，在相同算力和查询消耗下，取得了更高的训练奖励和下游任务表现。



Clip 行为对比

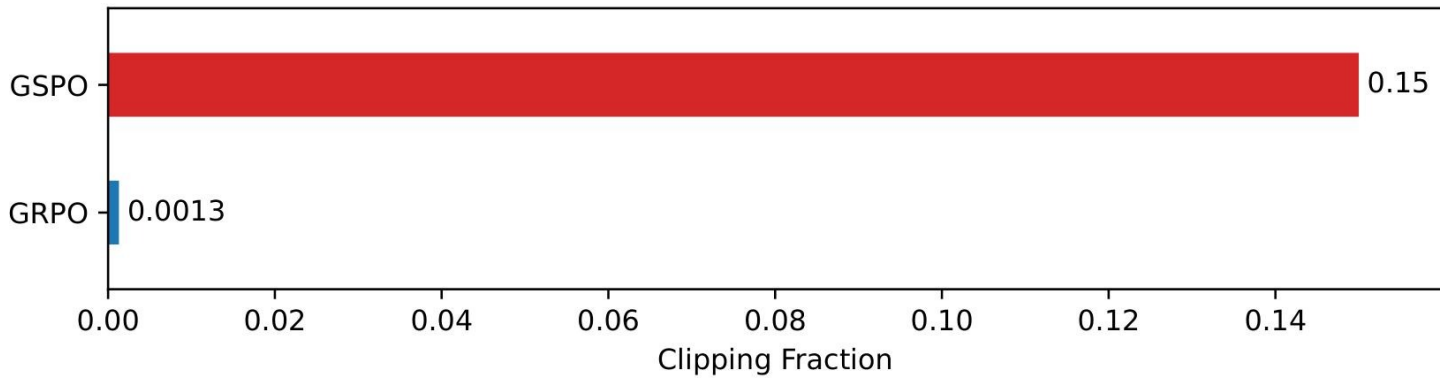
如下图所示，GSPO 会 **clip 掉更多数据（约 15% token）**，并且调整 clip 范围不会改变数量级差异。

这意味着：

- GRPO 的 token-level 更新中包含大量高方差噪声，即使保留了更多数据，训练信号仍然低效；

- GSPO 的序列级更新虽然“更严格”，丢弃的数据更多，但保留下来的训练信号更可靠，因此整体样本利用率更高。

换句话说，**GRPO 在噪声里屎海遨游，而 GSPO 提供了更干净的学习信号**。

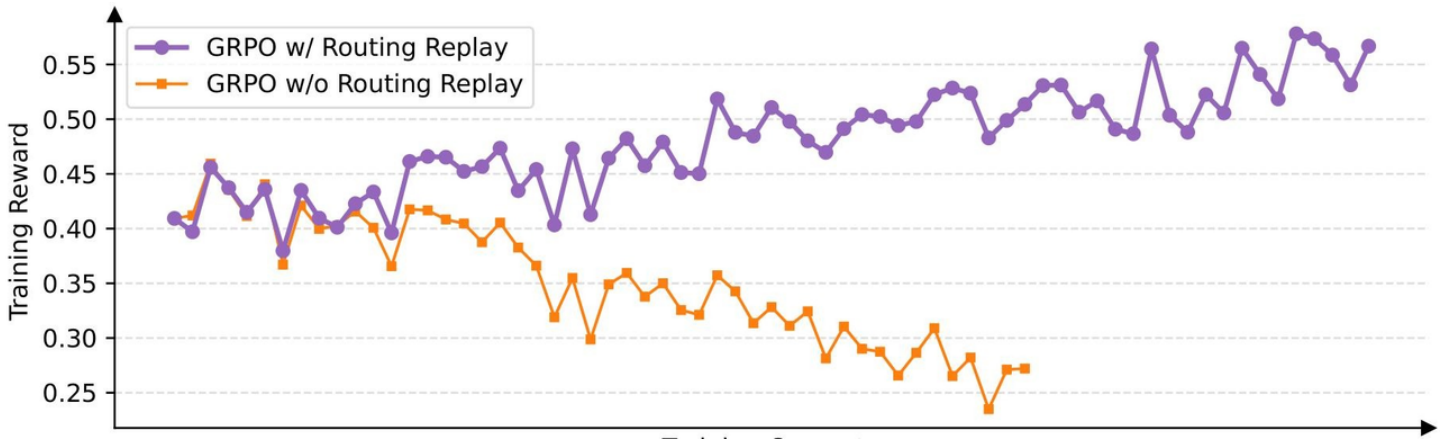


MoE模型的稳定性

MoE 模型的稀疏性使 GRPO 更容易失稳。实验发现，在 Qwen3-30B-A3B-Base 上，大约 10% 的查询会在新旧策略中激活不同的专家路径，在更深的 MoE 中比例更高。

- **GRPO 的应对方案**：通过 **Routing Replay** 强制新策略沿用旧策略的专家路径，保证一致性。但这增加了显存与通信开销，相当于“给 MoE 套上镣铐”。
- **GSPO 的根本改进**：只关注序列似然 $\pi_\theta(y(i)|x)$，而对单个 token 的似然不敏感。由于 MoE 保持整体语言建模能力，序列似然不会随专家切换剧烈波动，从根源上解决了训练震荡问题。

因此，**GSPO 在 MoE 模型中无需额外稳定化技巧，就能自然收敛**。



实际工程

在实际系统中，训练引擎（如 Megatron-LM）与推理引擎（如 SGLang、vLLM）往往存在精度差异。

- 使用 **GRPO** 时，需要在训练引擎中重新计算旧策略的 token-level 似然，代价高昂；
- **GSPO** 基于序列似然，对微小精度差异更具容忍度，可以直接使用推理引擎返回的序列似然，免去重算开销。

这一点在 **部分 rollout、多轮 RL、训练-推理分离**等场景下，能带来显著的效率提升。