# QaRL: Rollout-Aligned Quantization-Aware RL for Fast and Stable Training under Training–Inference Mismatch

**Hao Gu**[1] **Hao Wang**[2] **Jiacheng Liu**[1] **Lujun Li**[1] **Qiyuan Zhu**[1]
**Bei Liu**[1] **Binxing Xu**[3] **LEI WANG**[1] **Xintong Yang**[1] **Sida Lin**[1] **Sirui Han**[1*] **Yike Guo**[1*]

[1]The Hong Kong University of Science and Technology
[2]City University of Hong Kong [3]Zhejiang University
marcusguhao@gmail.com siruihan@ust.hk yikeguo@ust.hk

## Abstract

Large language model (LLM) reinforcement learning (RL) pipelines are often bottlenecked by rollout generation, making end-to-end training slow. Recent work mitigates this by running rollouts with quantization to accelerate decoding, which is the most expensive stage of the RL loop. However, these setups destabilize optimization by amplifying the training—inference gap: rollouts are operated at low precision, while learning updates are computed at full precision. To address this challenge, we propose QaRL (Rollout Alignment Quantization-Aware RL), which aligns training-side forward with the quantized rollout to minimize mismatch. We further identify a failure mode in quantized rollouts: long-form responses tend to produce repetitive, garbled tokens (error tokens). To mitigate these problems, we introduce TBPO (Trust-Band Policy Optimization), a sequence-level objective with dual clipping for negative samples, aimed to keep updates within the trust region. On Qwen3-30B-A3B MoE for math problems, QaRL outperforms quantized-rollout training by +5.5 while improving stability and preserving low-bit throughput benefits.

## 1 Introduction

Recent reasoning LLMs, such as OpenAI o1 (Jaech et al., 2024), have demonstrated the effectiveness of Chain-of-Thought (CoT) (Wei et al., 2022) for solving complex problems. More recently, DeepSeek-R1 (Guo et al., 2025) has shown that reinforcement learning with simple rule-based reward functions (RLVR), can induce emergent reasoning behaviors and yield gains in challenging domains such as math problem solving (Luo et al., 2025). A key driver behind these improvements is test-time scaling (Muennighoff et al., 2025): reasoning models often generate longer CoT, trade additional computation for accuracy. However, longer generations increase training-time cost, since RL continously samples long responses during optimization.

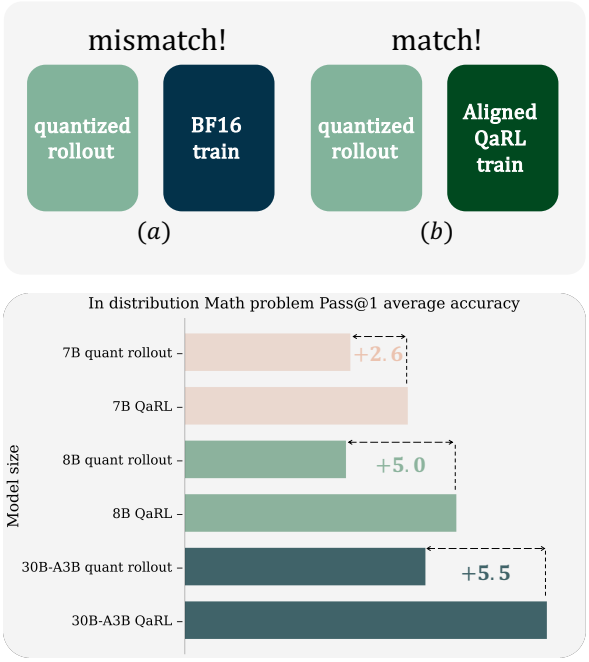


Figure 1: (a) Quantized rollout alone introduces mismatch. (b) Aligned QaRL alleviates it, improving performance across model sizes, including MoE.

Unlike SFT, which requires only a single forward pass, a standard RL step (Sheng et al., 2024; Hu et al., 2025) for LLMs involves three phases: (i) rollout to generate responses, (ii) a forward pass to compute token probabilities, and (iii) a backward pass to update model via policy gradients. Autoregressive decoding generates response token by token, making rollouts the dominant cost, comprising roughly 70% of RL training time. Consequently, quantizing rollout model is a natural way to accelerate RL optimization (Liu et al., 2025b). However, it also creates a training-inference mismatch challenge: responses are sampled from low bit rollout, while updates are performed by full precision learner.

To addresses both efficiency and stability, in this work, we study **quantization-aware RL**. By exe-

[*]Corresponding authors.



*Findings of the Association for Computational Linguistics: ACL 2026*, pages 17051–17064

cuting rollouts under low-bit quantization, we substantially reduce the dominant generation cost and accelerate end-to-end RL training. To mitigate the resulting training-inference mismatch, we further perform rollout-aligned quantization-aware training on the learner side, aligning learner's policy with the quantized behavior used for sampling.

Moreover, we find a critical failure mode in quantized rollouts: noise accumulates over long generations, producing off-trajectory repetitive and garbled tokens. These **error tokens** are typically assigned very low probability under the policy, driving the policy ratio (and mismatch reweighting) to extreme. Such outliers are not reliably controlled by standard PPO-style clipping, breaking the intended trust region behavior and destabilizing training. To address this, we introduce a trust-band control: we apply dual clipping tailored to negative samples and perform sequence-level objectives on both the policy ratio and the mismatch weight, drop entire responses that exceed the bands.

Extensive experiments on math, logic and code benchmarks demonstrate that QaRL matches BF16 training performance and outperform quantized rollout training. Even on unstable MoE models such as Qwen3-30B-A3B-Base, QaRL achieves an average math score of 51.2, close to BF16's 52.1 and surpassing quantized rollout's 45.7, while delivering a 1.3× training speedup over BF16. Our contributions are as follows:

1. We present a practical quantized aware RL pipeline for decoupled, hybrid RL systems, and introduce **QaRL** to minimize the mismatch between quantized rollouts and learner-side training.
2. We identify error tokens as the key cause of training instability under quantized rollouts, and propose **Trust Band Policy Optimization (TBPO)**, a sequence-level dual-clipping strategy that keeps updates within a trust region and stabilizes convergence.

## 2 Preliminaries

### 2.1 Group Relative Policy Optimization.

GRPO (Shao et al., 2024) is a PPO style (Schulman et al., 2017) policy gradient objective that removes the value function (critic) and uses group relative rewards as the baseline. Given a query $q$, we sample a group of $G$ responses $\{o_1, \ldots, o_G\}$ from the old policy $o_i \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$. Let $r_i$ be the scalar (verifiable) reward for $o_i$ and $|o_i|$ its length. GRPO defines a group-normalized advantage $A_i$ (broadcast to tokens as $A_{i,t} = A_i$) and the token-level importance ratio $R_{i,t}(\theta)$:

$$A_i = \frac{r_i - \text{mean}\{r_1, \ldots r_G\}}{\text{std}\{r_1, \ldots r_G\}},$$

$$R_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q_i)}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q_i)}$$

To prevent overly large updates and ensure the update remains within the trust region (Schulman et al., 2015), GRPO applies PPO-style clipping and defines the token-level clipped policy loss:

$$\mathcal{L}_{i,t}(\theta) = -\min\Big(R_{i,t}(\theta)\, A_i,\ \tilde{R}_{i,t}(\theta)\, A_i\Big),$$

where $\tilde{R}_{i,t} = \text{clip}(R_{i,t}, 1-\epsilon, 1+\epsilon)$ and $\epsilon$ controls the trust region. The GRPO optimization objective is defined as follows:

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{q,\, o\sim\pi_{\theta_{\text{old}}}} \left[ \frac{1}{G} \sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \mathcal{L}_{i,t}(\theta) \right].$$

### 2.2 Quantization

Quantization accelerates LLM inference by enabling low-bit GEMM operations, where weights and activations are compressed into low bit representations that modern GPUs can process directly via Tensor Core kernels.

**Integer quantization.** A full-precision tensor $W \in \mathbb{R}$ is mapped to $W_q \in \mathbb{Z}$ via scale $s$ and zero-point $z$:

$$\textbf{Quant}: \ W_q = \text{clamp}\left( \left\lfloor \frac{W}{s} \right\rceil + z,\ q_{\min}, q_{\max} \right),$$

$$\textbf{Dequant}: \ \hat{W} = s\,(W_q - z),$$

where $\lfloor \cdot \rceil$ denotes rounding. Quantized GEMM computes the integer matmul and rescales the result (accumulating in higher precision):

$$Y \approx s_x s_w \cdot \big((X_q - z_x)\,(W_q - z_w)\big). \quad (1)$$

**Floating-point quantization.** For low-bit floating formats (e.g., FP8/FP4), values are cast to reduced precision with scaling:

$$\textbf{Quant}: \ s = \frac{\max(|W|)}{\alpha},\ W_q = \big\lfloor W/s \big\rceil_{\text{FP}k},$$

$$\textbf{Dequant}: \ \hat{W} = s \cdot W_q,$$

where $\lfloor \cdot \rceil_{\text{FP}k}$ denotes casting to FP$k$ and $\alpha$ is the maximum finite representable value ($\alpha = 6$

for e2m1FP4, $\alpha = 448$ for e4m3FP8). Low-bit GEMM performs multiplication on quantized operands and rescales:

$$Y \approx (s_x s_w) \cdot \big(X_q W_q\big), \tag{2}$$

with $X_q, W_q$ as the low-bit floating representations. We denote x-bit weight and y-bit activation quantization as WxAy.

**Quantization-Aware Training (QAT).** To recover accuracy lost in low-bit conversion, QAT integrates quantization noise into the training loop (Liu et al., 2024). Specifically, it injects `fake quant` into the forward pass to simulate rounding, while master weights remain in full precision. The Straight-Through Estimator (STE) approximates gradients for non-differentiable backward pass.

**Fully Quantized Training (FQT).** Distinct from QAT, which employs `fake quant` to merely simulate quantization noise while maintaining high-precision arithmetic, Fully Quantized Training (or Low-Bit Training) fundamentally executes operations using actual low-bit data types (e.g., FP8, INT8) (Xi et al., 2024; Zhao et al., 2025). By leveraging low-precision hardware kernels for both forward and backward passes, this paradigm directly reduces the computational overhead and memory footprint of the training process itself.

### 2.3 Training–Inference Mismatch in RL

Modern LLM RL pipelines are typically hybrid systems: for throughput, rollouts are generated by a high-performance inference engine (e.g., vLLM, SGLang), while policy optimization are carried out by a training backend (e.g., FSDP/Megatron). Although both components load the same parameter $\theta_{\text{old}}$, the rollout and the training engine may produce different results for the same query ($\text{LLM}_{\text{rollout}}(q) \neq \text{LLM}_{\text{train}}(q)$). This occurs because they often implement different kernels (e.g., different attention kernels, batch variant operations). As a result, the rollout engine induces a **sampler policy** $\pi_{\text{sampler}}(\cdot \mid \theta_{\text{old}})$ that is not exactly the same distribution as the **learner policy** $\pi_{\text{learner}}(\cdot \mid \theta_{\text{old}})$. We refer to this discrepancy as **training–inference (learner–sampler) mismatch**.

Consider PPO's clipped objective:

$$\mathbb{E}_{a\sim\pi_{\theta_{\text{old}}}}\Big[\min\big(r_{\text{prox}}\,\hat{A}, \text{clip}(r_{\text{prox}}, 1-\epsilon, 1+\epsilon)\,\hat{A}\big)\Big]$$

where proximal importance sampling $r_{\text{prox}} = \frac{\pi(a|\theta)}{\pi(a|\theta_{\text{old}})}$ and $\hat{A}$ is an advantage estimate. In a hybrid system, however, tokens are sampled from $\pi_{\text{sampler}}(\cdot \mid \theta_{\text{old}})$, while $r_{\text{prox}}$ is computed using learner's distributions:

$$\mathbb{E}_{a\sim\pi_{\textbf{sampler}}(\theta_{\text{old}})}\Big[\min\big(\frac{\pi_{\textbf{learner}}(a \mid \theta)}{\pi_{\textbf{learner}}(a \mid \theta_{\text{old}})}\,\hat{A},\ \text{clip}(\frac{\pi_{\textbf{learner}}(a \mid \theta)}{\pi_{\textbf{learner}}(a \mid \theta_{\text{old}})}, 1-\epsilon, 1+\epsilon)\,\hat{A}\big)\Big]$$

Consequently, the trust region is enforced on the sampler distribution rather than the learner distribution, causing the update to deviate from its intended target. Drawing inspiration from Decoupled PPO (Hilton et al., 2022), where a mismatch between the behavior and proximal policies is corrected via importance sampling, we can reweight the updates by multiplying with the ratio:

$$w_{\text{mismatch}} = \frac{\pi_{\textbf{learner}}(a \mid \theta_{\text{old}})}{\pi_{\textbf{sampler}}(a \mid \theta_{\text{old}})} \tag{3}$$

Applying $w_{\text{mismatch}}$ yields

$$\mathbb{E}_{a\sim\pi_{\text{sampler}}(\theta_{\text{old}})}\Big[\frac{\pi_{\textbf{learner}}(a \mid \theta_{\text{old}})}{\pi_{\textbf{sampler}}(a \mid \theta_{\text{old}})} \cdot \min\big(r_{\text{prox}}\,\hat{A}, \text{clip}(r_{\text{prox}}, 1-\epsilon, 1+\epsilon)\,\hat{A}\big)\Big]. \tag{4}$$

Intuitively, this correction is conceptually orthogonal to PPO proximal ratio $r_{\text{prox}} = \frac{\pi_{\text{learner}}(a|\theta)}{\pi_{\text{learner}}(a|\theta_{\text{old}})}$, while $w_{\text{mismatch}}$ rectifies the distribution shift caused by the system-level training–inference mismatch. Together, they ensure policy optimization remains within the learner's trust region. Recent works have further explored this direction, such as **TIS** (Yao et al., 2025) (which truncates upper bound of $w_{\text{mismatch}}$) and **MIS** (Liu et al., 2025a) (reject samples with overly large $w_{\text{mismatch}}$ and apply sequence level $w_{\text{mismatch}}$).

## 3 Methods

### 3.1 Rollout-Aligned Quantization Aware Reinforcement Learning

To alleviate the decoding bottleneck during rollouts, a practical approach is to quantize the rollout model. By lowering the precision of weights and activations (e.g., W8A8 or W4A16), we can significantly accelerate rollout generation. However, quantizing only the rollout engine (we term this **Quantized Rollout Training**) introduces a more severe form of training–inference mismatch.

In this regime, the sampler policy $\pi_{\text{sampler}}$ becomes a quantized approximation $\pi_{\text{quant-sampler}}$, while the learner policy $\pi_{\text{learner}}$ typically remains

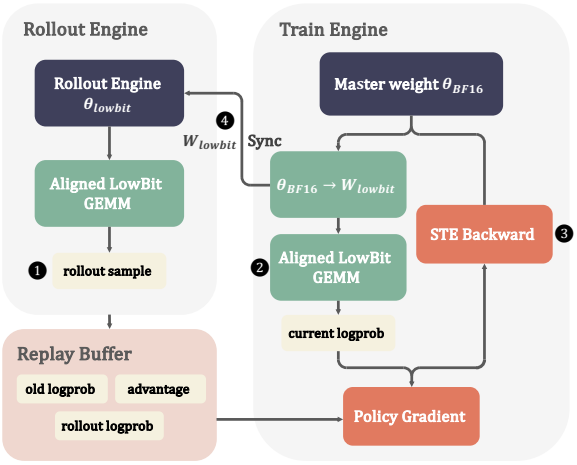


Figure 2: **Overview of the QaRL pipeline in a hybrid RL system.** ❶ The quantized rollout engine $\theta_{\text{lowbit}}$ generates samples. ❷ The training engine maintains $\theta_{\text{BF16}}$ master weights and performs rollout-aligned low-bit GEMM to compute current logprob. ❸ Policy gradients are computed using replay buffer data to update the model via STE. ❹ The updated low-bit weights $W_{\text{lowbit}}$ are synchronized to the rollout engine.

full-precision $\pi_{\text{BF16-learner}}$. The resulting mismatch makes the importance weight

$$w_{\text{mismatch}} = \frac{\pi_{\text{learner}}}{\pi_{\text{sampler}}} = \frac{\pi_{\text{BF16-learner}}}{\pi_{\text{quant-sampler}}}$$

drift far away from 1.00 as Fig. 4(d). When both sampler and learner operate in full precision, the mismatch is typically limited to implementation details (e.g. different kernels), so the distribution shift is mild and can often be compensated by reweighting with $w_{\text{mismatch}}$. In contrast, quantized rollout training introduces larger distribution gap and $w_{\text{mismatch}}$ become more extreme. This effect is especially pronounced for long responses, quantization-induced errors accumulate across decoding steps, making the divergence grow with response length and increasingly difficult to correct. To fundamentally resolve this mismatch, we align the learner with the quantized sampler by adopting **Quantization-Aware Reinforcement Learning (QaRL)**.

But building a quantized training pipeline in a decoupled hybrid RL stack is non-trivial: rollouts are generated by an inference engine, while policy optimization runs in a separate training backend. To address this challenge, we decompose our implementation into three components: **(1) low-bit rollout inference**, **(2) rollout aligned quantization aware training**, and **(3) weight synchronization from training to inference**. Concretely:

**1. Quantized inference in the rollout engine.** We deploy a low-bit rollout model in the inference engine to accelerate decoding. Empirically, for larger models (especially **MoE**), we observe that W4A16 can deliver higher throughput than W8A8 due to memory constraints. Therefore, we support both configurations to maximize rollout efficiency across different model regimes.

**2. Rollout aligned quantization aware training.** On the training side, we maintain master weights $\theta_{\text{BF16}}$ in high-precision. Standard QAT simulates quantization error by inserting fake-quant operators $\left[\text{dequant}[\text{quant}(\theta_{\text{BF16}})]\right]$, while computing in high precision. To minimize the gap between $\pi_{\text{quant-sampler}}$ and $\pi_{\text{quant-learner}}$, we perform on-the-fly quantization during the forward pass:

$$\hat{\mathbf{W}}_{\text{low-bit}} = \text{Quant}(\theta_{\text{BF16}})$$

Critically, rather than merely simulating quantization, we execute low-bit GEMM directly on these quantized tensors, thereby precisely mirroring the arithmetic behavior of the rollout engine. During the backward pass, gradients are computed via the Straight-Through Estimator (STE) and applied to the master weights $\theta_{\text{BF16}}$. Our approach lies between fake quant QAT and end-to-end low-precision FQT: we execute the forward pass in low bit while keeping the backward pass in full precision. We avoid FQT for larger quantization error introduced by 4-bit gradients.

**3. Weight updates from training to inference.** After each optimization step, the learner's parameters change, requiring the rollout engine to be refreshed accordingly. Since the training engine already materializes $\hat{\mathbf{W}}_{\text{low-bit}}$ for the aligned low-bit GEMM, we directly publish these low-bit tensors to the inference engine at each step. This avoids redundant re-quantization and ensures the sampler weights remain in the exact low-bit format learner uses during forward. The overall pipeline is illustrated in Fig. 2.

Notably, unlike prior work (Wasti et al., 2025; SGLang Team, 2025) that enforces strictly bitwise consistent on-policy execution, our approach tolerates discrepancy between the sampler and learner policies $\pi_{\text{quant-sampler}} \neq \pi_{\text{quant-learner}}$ and compensates for it using $w_{\text{mismatch}}$. Achieving bitwise alignment requires both batch and tensor-parallel invariance kernel, which is not a "free lunch", incurring an average 2× slower (He and Lab, 2025; Zhang et al., 2025). Consequently, rather than bitwise-

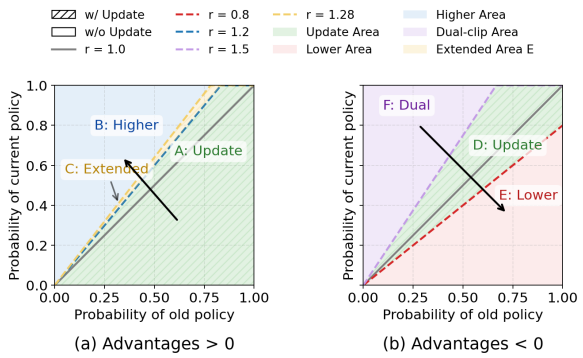


Figure 3: **Token level policy clipping regions.** Axes represent token probabilities under the old and current policies, with the slope defining the $r_{\text{prox}} = \text{prob}_{\text{current}}/\text{prob}_{\text{old}}$, with arrows indicating the direction of the policy update.

identical kernels, we focus on using aligned low-bit forward to ensure that optimization remains robustly within the intended trust region. As for master weight precision, previous work (Qi et al., 2025) claims that using the finer granularity of $\theta_{\text{FP16}}$ can mitigate mismatch. However, we empirically find that using $\theta_{\text{FP16}}$ with dynamic loss scaling causes gradient NaN underflow, and this phenomenon does not occur with $\theta_{\text{BF16}}$.

### 3.2 Trust Band Policy Optimization

Although rollout-aligned quantized training substantially mitigates instability arising from training–inference mismatch, we observe a remaining failure mode: under-trained quantized policy tend to produce **repetitive and garbled tokens** in long responses, which we term **error tokens**, illustrated in appendix table 6. This stems from the error-amplification dynamics of autoregressive decoding: an error token at step $t$ sends the model off-trajectory, causing subsequent tokens generated from a corrupted state and amplifying degradation.

This phenomenon interacts closely with the trust-region control in PPO-style objectives. Standard PPO clipping operates directionally: for positive advantages, it constrains only the upper bound to $(0, 1+\epsilon)$, while for negative advantages, it constrains only the lower bound to $(1-\epsilon, +\infty)$. These prevent the updated policy drift too far from the old policy in the update direction. Recently, several works have revisited this design. For example, DAPO (Yu et al., 2025) proposes clipping higher for $A > 0$ as `C:extended` region in Fig. 3(a), arguing that high-entropy which correspond to low-probability tokens, can encourage exploration.

**Dual Clipping.** In our setting, the more critical

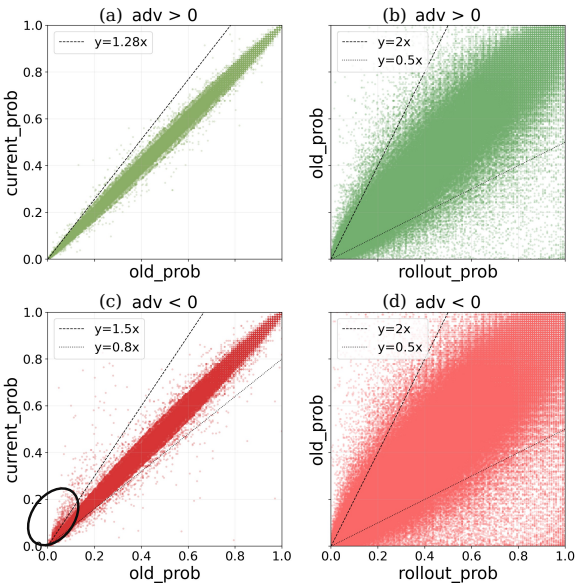


Figure 4: **Distribution of token level $r_{\text{prox}}$ and $w_{\text{mismatch}}$.** Lines indicate clipping boundaries.

issue emerges with negative samples. Empirically, error tokens are rare in positive trajectories but predominate in negative ones. As shown in Fig. 4(c), these negative error tokens are concentrated in the low-probability regions of both the old and current policies ($\pi_{\text{old},t}$ and $\pi_{\theta,t}$). In this regime, the PPO policy ratio becomes more sensitive to baseline probabilities:

$$r_{\text{prox},t} = \begin{cases} 0.80/0.77 \approx 1.03 & p_{\text{old}} = 0.77,\ p_\theta = 0.80 \\ 0.05/0.02 = 2.5 & p_{\text{old}} = 0.02,\ p_\theta = 0.05 \end{cases}$$

Here, identical absolute perturbations produce dramatically different relative changes depending on $\pi_{\text{old},t}$. For typical high-confidence tokens (e.g., $p_{\text{old}} = 0.77 \rightarrow p_\theta = 0.80$), the ratio remains modest at $\approx 1.07$. Conversely, for low-probability tail tokens (e.g., $p_{\text{old}} = 0.02 \rightarrow p_\theta = 0.05$), the ratio explodes to 2.5. And low-prob tokens are inherently more prone to probability shifts. This effect is directly visible in Fig. 5(b): when $\pi_{\text{old}}$ is small, negative-advantage tokens exhibit a much heavier tail of $r_{\text{prox}}$, producing extreme ratios. Under $A_t < 0$, standard PPO clipping is one-sided (it only enforces a lower bound), so these high $r_{\text{prox}}$ tail tokens can dominate the gradient magnitude, inflate variance, and break the intended trust region. To mitigate this failure mode, we leverage **a negative dual clipping scheme** (Ye et al., 2020): for $A < 0$, we not only enforce the lower bound but also impose an explicit upper bound as `F:Dual` region in Fig. 3(b), preventing exploding ratios from

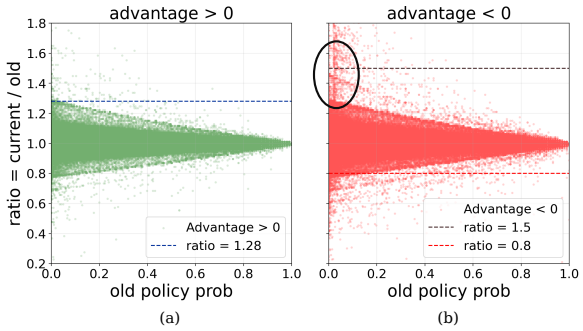


Figure 5: **Relationship between $r_{\text{prox}}$ and old policy probability.** Low-probability negative tokens are more prone to extreme $r_{\text{prox}}$.

low-probability error tokens from dominating the update.

**Sequence Level Objectives.** Furthermore, since RLVR optimize outcome rewards, GRPO's token-level importance sampling serves merely as a first-order approximation. Token-level clipping fails to rectify **mid-response errors**: once a single token deviates, the subsequent tokens no longer lie on the intended trajectory (Fig 6).

To remedy this, we return to a stricter trust-region view and treat each entire response as single action. Concretely, we apply sequence-level ratios and clipping (Zheng et al., 2025), mask out the entire responses from policy update when their sequence-level ratio exceeds the bounds. Another advantage of sequence-level ratio is its ability to discriminate error tokens and low-prob exploration tokens, which token-level bounds cannot resolve. Error tokens typically occur in clusters, causing high variance in $r_{\text{seq-prox}}$ that readily exceeds trust-region thresholds. In contrast, exploration tokens generate more stable sequence-level ratios, as the geometric averaging over the sequence reduces sensitivity to individual perturbations.

Formally, we treat the entire response $o = (o_1, \ldots, o_L)$ as a single action and defining its sequence-level probability as the geometric mean of the token probabilities. Consequently, we formulate $r_{\text{seq-prox}}$ and $w_{\text{seq-mismatch}}$:

$$\pi(o \mid q) \triangleq \exp\left(\frac{1}{L}\sum_{t=1}^{L}\log \pi(o_t \mid q, o_{<t})\right).$$

$$r_{\text{seq-prox}}(\theta) \triangleq \frac{\pi_{\text{current learner}}(o \mid q)}{\pi_{\text{old learner}}(o \mid q)},$$

$$w_{\text{seq-mismatch}}(\theta) \triangleq \frac{\pi_{\text{old learner}}(o \mid q)}{\pi_{\text{old sampler}}(o \mid q)}.$$

For mismatch weight, we truncate with a two-sided


Figure 6: **A mid response error propagates to future tokens.** Although the initial garbled tokens are clipped, the repetitive tokens induced by this error are not clipped by token-level objectives.

cap to control variance:

$$\tilde{w}(\theta) \triangleq \text{clip}(w(\theta), [-\log c, \log c]), \qquad (5)$$

where $c > 1$ is the TIS cap. For the proximal ratio, positive-advantage samples use the standard PPO-style bound $[0,\ 1+\epsilon_h]$, while negative samples are constrained by dual-sided band $[1-\delta_\ell,\ 1+\delta_h]$:

$$\tilde{r}(\theta) = \begin{cases} \text{clip}(r(\theta), 0, \quad 1+\epsilon_h) & \text{if } \hat{A} \geq 0 \\ \text{clip}(r(\theta), 1-\delta_l, 1+\delta_h) & \text{if } \hat{A} < 0 \end{cases}$$

Put everything together, the sequence-level surrogate (shared by all tokens in $o$) is

$$\mathcal{J}(\theta) = \mathbb{E}_{q,\, o\sim\pi_{\text{old sampler}}(\cdot|q)}\left[\tilde{w}(\theta)\cdot\tilde{r}(\theta)\cdot\hat{A}\right].$$

## 4 Experiments

**Training Setup.** We utilize OpenR1-Math-46K (Yan et al., 2025), a dataset of 46,000 mathematical problems, and conduct experiments on Qwen2.5-Math-1.5B/7B, Qwen3-8B-Base, and Qwen3-30B-A3B-Base (Yang et al., 2025). We employ GRPO with TIS for the BF16 RL baseline and quantized training, while GSPO is adopted for MoE models to enhance stability. The Muon optimizer (Jordan et al., 2024) exhibits significantly faster convergence than AdamW and is therefore used across all experiments; detailed hyperparameters are provided in Appendix B.

**Evaluation.** We evaluate on math benchmarks (AIME2024/2025, AMC, Math-500, OlympiadBench (He et al., 2024), Minerva) and out-of-distribution benchmarks (ARC-Challenge (Clark et al., 2018), GPQA-Diamond (Rein et al., 2024), LiveCodeBench, MMLU Pro).

### 4.1 Main Results

Table 1 and Fig. 7 demonstrate a consistent trend across model scales: quantized rollout training undermines stability and yields lower final accuracy

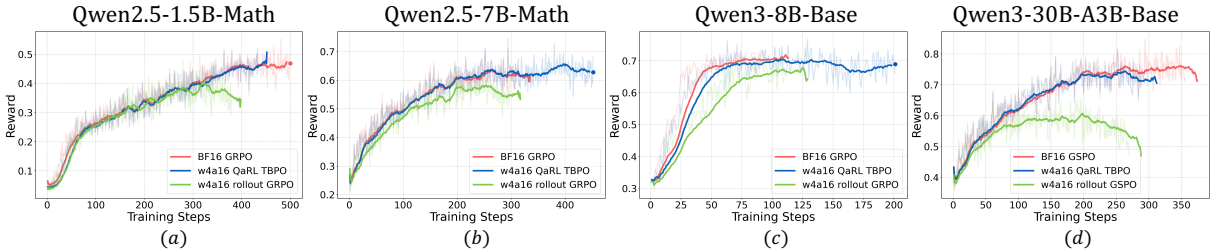


Figure 7: **Training reward curves across different models.** Our QaRL TBPO demonstrates stability over quantized rollout training, and converging to reward levels nearly identical to the full-precision BF16 baseline.

| Model | In-Distribution Performance | | | | | | Out-of-Distribution Performance | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | AIME 24/25 | AMC | MATH-500 | Minerva | Olympiad | Avg. | ARC-c | GPQA | MMLU-Pro | LiveCodeBench | Avg. |
| Qwen2.5-1.5B-Math | 4.5/2.8 | 26.5 | 50.8 | 21.6 | 20.3 | 21.0 | 11.7 | 12.4 | 10.4 | 2.7 | 9.3 |
| BF16 GRPO | 12.5/9.2 | 43.5 | 71.4 | 36.2 | 34.5 | 34.5 | 58.9 | 27.8 | 26.7 | 11.9 | 31.3 |
| w4a16 rollout GRPO | 7.9/6.4 | 38.1 | 65.3 | 30.2 | 28.9 | 29.3 | 50.1 | 19.8 | 21.2 | 4.3 | 23.8 |
| w4a16 QaRL TBPO | 12.5/10.4 | 46.6 | 69.8 | 31.9 | 32.6 | 33.9 | 57.6 | 21.1 | 25.3 | 8.1 | 28.0 |
| Qwen2.5-7B-Math | 15.0/6.4 | 46.2 | 67.4 | 32.9 | 23.9 | 33.4 | 62.6 | 28.5 | 32.1 | 8.0 | 32.8 |
| BF16 GRPO | 19.5/11.6 | 59.6 | 80.0 | 45.9 | 43.4 | 43.3 | 80.4 | 38.2 | 46.3 | 14.6 | 44.8 |
| w4a16 rollout GRPO | 19.1/9.6 | 54.6 | 79.1 | 42.8 | 40.6 | 40.9 | 73.9 | 31.6 | 40.0 | 8.7 | 38.5 |
| w4a16 QaRL TBPO | 19.5/13.3 | 58.1 | 81.8 | 43.4 | 45.3 | 43.5 | 81.1 | 37.1 | 45.8 | 13.9 | 44.4 |
| Qwen3-8B-Base | 7.9/9.6 | 46.3 | 74.2 | 42.7 | 39.3 | 36.6 | 44.9 | 31.2 | 49.5 | 23.0 | 37.1 |
| BF16 GRPO | 28.3/19.3 | 64.3 | 88.1 | 54.5 | 56.7 | 51.8 | 93.0 | 46.3 | 65.1 | 45.7 | 62.5 |
| w4a16 rollout GRPO | 20.0/12.5 | 53.0 | 82.2 | 50.7 | 45.1 | 43.9 | 91.6 | 43.9 | 61.5 | 41.4 | 59.5 |
| w4a16 QaRL TBPO | 26.6/16.9 | 62.2 | 83.6 | 52.5 | 51.7 | 48.9 | 92.3 | 45.0 | 63.8 | 43.4 | 61.0 |
| Qwen3-30B-A3B-Base | 15.4/7.9 | 49.0 | 67.4 | 31.2 | 38.1 | 34.8 | 61.3 | 34.8 | 52.5 | 28.5 | 44.2 |
| BF16 GSPO | 27.9/21.6 | 63.2 | 88.8 | 54.7 | 56.7 | 52.1 | 95.2 | 50.1 | 70.3 | 55.8 | 67.8 |
| w4a16 rollout GSPO | 22.0/18.7 | 55.4 | 84.0 | 47.4 | 47.1 | 45.7 | 89.3 | 42.4 | 65.3 | 47.9 | 61.2 |
| w4a16 QaRL TBPO | 27.5/22.0 | 62.9 | 87.2 | 51.4 | 56.1 | 51.2 | 96.6 | 48.2 | 68.0 | 55.4 | 67.05 |

Table 1: Main results on in-distribution math and out-of-distribution benchmarks. LiveCodeBench results are reported in $pass@4$, while all other metrics use $pass@1$.

than the BF16 RL baseline, whereas QaRL TBPO exhibits markedly improved stability, converging to BF16 comparable rewards. For instance, on the Qwen3-8B model, the average in-distribution math performance drops significantly from 51.8% (BF16) to 43.9% with quantized rollout training, while QaRL TBPO successfully maintain the performance to 48.9%, only 2.9% drop. Critically, QaRL TBPO recovers most of the degradation from quantized-rollout training, achieving near-baseline performance. Although quantized rollout training under GSPO still suffers from router shift at each forward pass, TBPO remains stable on MoE models, achieving an average math score of 51.2%, nearly matching the 52.1% baseline.

Beyond math, QaRL TBPO improves OOD performance vs quantized rollout training while matching BF16, demonstrating that gains arise from stable optimization and better generalization rather than overfitting.

## 4.2 Ablation

**Optimize objectives** We conduct ablations to validate TBPO's effectiveness in mitigating error-token interference under QaRL (Fig. 8). **Overall:** TBPO achieves stable learning, high reward, and tight KL control, while alternatives suffer from KL drift or low sample efficiency. **GSPO** is unstable—error tokens in negatives cause KL drift and collapse, degrading reward. **MIS GSPO:** Rejection sampling reduces data efficiency and limits reward ceiling. **Positive GRPO:** Discarding negatives loses exploration and limits performance. **On-policy GRPO:** Single updates per batch reduce error amplification but hurt efficiency. **Dual-clip GRPO:** Clipping bounds extreme ratios, yet tokens

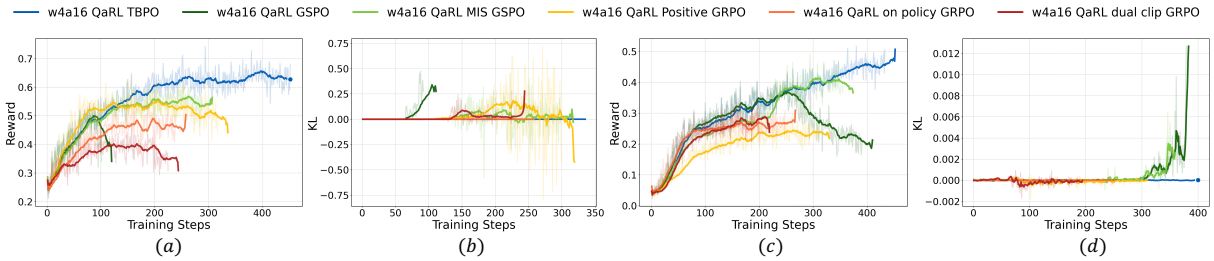


Figure 8: Training dynamics (Reward/KL) of Qwen2.5-Math 1.5B (a-b) and 7B (c-d) across different optimization objectives.

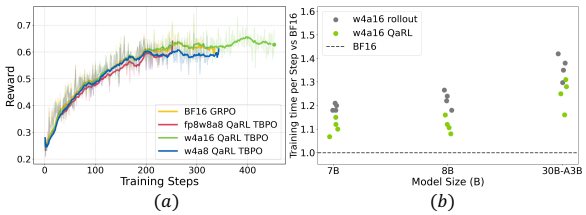


Figure 9: **(a)** Reward curves under different quant scheme. **(b)** Per-step training time speedup ratio.

after error tokens remain contaminated, causing incorrect learning. For comprehensive analysis of SAPO, please refer to Appendix D.4.

**Quantization Scheme** We further ablate quantization schemes in QaRL-TBPO, comparing FP8W8A8, W4A16, and W4A8 against the BF16 GRPO baseline (Fig. 9(a)). No substantial differences emerge: all bit-widths exhibit similar reward curves and nearly identical final rewards. This suggests that, once stabilized via TBPO, RL convergence is largely insensitive to the specific low-bit format, and performance gains are not tied to a particular quantization choice. We adopt W4A16 for most experiments due to its superior speed on larger models and broad hardware compatibility, whereas W4A8 requires more complex kernel support. Results of TBPO used on quantized rollout training are detailed in appendix D.2.

**Speed** Fig. 9(b) reports per-step training time normalized to BF16. Across 7B, 8B, and 30B-A3B MoE models, both QaRL and quantized rollout training achieve speedups over BF16. For large-scale MoE models, quantized rollout training delivers a 1.4× speedup, while QaRL achieves a 1.3× speedup, since QaRL incurs modest overhead from in-training quantization. Comparison of different quantization schemes are provided in appendix 11. We opted not to implement FP8 KV quantization, as KV quant currently fails to provide meaningful throughput benefits in vLLM.

## 5 Related Work

**Quantization.** Quantization accelerates and compresses LLMs (Liu et al., 2026) via post-training quantization (PTQ) or quantization-aware training (QAT). PTQ methods (Gu et al., 2025) such as GPTQ (Frantar et al., 2022) and AWQ (Lin et al., 2024) avoid retraining but often struggle on reasoning models, where long CoT generation induces distribution shift and error accumulation beyond static calibration. QAT methods (e.g., LLM-QAT (Liu et al., 2024), EfficientQAT (Chen et al., 2025), Bit-by-Bit (Xu et al., 2026)) train with quantization noise for better robustness, while fully quantized training (Wang et al., 2025b) executes low-bit arithmetic end-to-end to further improve robustness and throughput.

**Reinforcement Learning in LLMs.** Modern reasoning models (e.g., DeepSeek-R1, Qwen3 (Yang et al., 2025), Kimi-K2 (Moonshot AI, 2025)) commonly build on GRPO. Recent variants include DAPO (higher-bound clipping to promote exploration), GSPO (sequence-level importance sampling for MoE stability), and ASPO (Wang et al., 2025a) (asymmetric sampling for low-probability tokens) and other studies have explored the role of entropy in RL (Wang et al., 2026, 2025c). Building on this line, we focus on stabilizing RL under quantized rollouts and QAT by suppressing quantization-induced errors during generation.

## 6 Conclusion and Limitations

We propose QaRL to mitigate the severe training–inference mismatch in quantized-rollout RL via rollout-aligned training, and identify error tokens as a key driver of collapse under quantized rollouts, addressed by TBPO with a sequence-level objective and dual clipping on negative samples. Looking forward, we plan to explore fully quantized RL training, and to replace costly, low-utilization sequence-level optimization with token-

level approximations that retain stability while improving efficiency and sample usage.

## References


Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. 2025. Efficientqat: Efficient quantization-aware training for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10081–10100.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. 2025. Soft adaptive policy optimization. *arXiv preprint arXiv:2511.20347*.

Hao Gu, Lujun Li, Zheyu Wang, Bei Liu, Qiyuan Zhu, Sirui Han, and Yike Guo. 2025. Btc-llm: Efficient sub-1-bit llm quantization via learnable transformation and binary codebook. *arXiv preprint arXiv:2506.12040*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.

Horace He and Thinking Machines Lab. 2025. Defeating nondeterminism in llm inference. *Thinking Machines Lab: Connectionism*. Https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/.

Jacob Hilton, Karl Cobbe, and John Schulman. 2022. Batch size-invariance for policy optimization. *Advances in Neural Information Processing Systems*, 35:17086–17098.

Jian Hu, Xibin Wu, Wei Shen, Jason Klein Liu, Weixun Wang, Songlin Jiang, Haoran Wang, Hao Chen, Bin Chen, Wenkai Fang, Xianyu, Yu Cao, Haotian Xu, and Yiming Liu. 2025. OpenRLHF: A ray-based easy-to-use, scalable and high-performance RLHF framework. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 656–666, Suzhou, China. Association for Computational Linguistics.

Wei Huang, Yi Ge, Shuai Yang, Yicheng Xiao, Huizi Mao, Yujun Lin, Hanrong Ye, Sifei Liu, Ka Chun Cheung, Hongxu Yin, and 1 others. 2025. Qerl: Beyond efficiency–quantization-enhanced reinforcement learning for llms. *arXiv preprint arXiv:2510.11696*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. 2024. Muon: An optimizer for hidden layers in neural networks.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6:87–100.

Jiacai Liu, Yingru Li, Yuqian Fu, Jiawei Wang, Qian Liu, and Yu Shen. 2025a. When speed kills stability: Demystifying RL collapse from the training-inference mismatch.

Jiacheng Liu, Peng Tang, Wenfeng Wang, Yuhang Ren, Xiaofeng Hou, Pheng Ann Heng, Minyi Guo, and Chao Li. 2026. A survey on inference optimization techniques for mixture of experts models. *ACM Computing Surveys*, 58(10):1–37.

Liyuan Liu, Feng Yao, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. 2025b. Flashrl: 8bit rollouts, full power rl.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2024. Llm-qat: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 467–484.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2. Notion Blog.

Moonshot AI. 2025. Kimi-k2: Thinking and reasoning. https://moonshotai.github.io/Kimi-K2/thinking.html. Accessed: 2025-12-22.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. 2025. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332.

Penghui Qi, Zichen Liu, Xiangxin Zhou, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Defeating the training-inference mismatch via fp16. *arXiv preprint arXiv:2510.26788*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

SGLang Team. 2025. *Deterministic Inference*. Accessed: 2025-12-21.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Hao Wang, Hao Gu, Hongming Piao, Kaixiong Gong, Yuxiao Ye, Xiangyu Yue, Sirui Han, Yike Guo, and Dapeng Wu. 2026. Learning while staying curious: Entropy-preserving supervised fine-tuning via adaptive self-distillation for large reasoning models. *arXiv preprint arXiv:2602.02244*.

Jiakang Wang, Runze Liu, Lei Lin, Wenping Hu, Xiu Li, Fuzheng Zhang, Guorui Zhou, and Kun Gai. 2025a. Aspo: Asymmetric importance sampling policy optimization. *arXiv preprint arXiv:2510.06062*.

Wenjun Wang, Shuo Cai, Congkai Xie, Mingfa Feng, Yiming Zhang, Zhen Li, Kejing Yang, Ming Li, Jiannong Cao, and Hongxia Yang. 2025b. Infir2: A comprehensive fp8 training recipe for reasoning-enhanced language models. *arXiv preprint arXiv:2509.22536*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.

Zihan Wang, Xingle Xu, Hao Wang, Yiwen Ye, Yuchen Li, Linhao Wang, Hongze Tan, Peidong Wang, Shi Feng, Guoqing Chen, Jinghao Lin, Zijing Wang, Yiqun Zhang, Yongkang Liu, Xiaocui Yang, Tao Yan, Shengzhi Wang, Yuhang Wu, and Ge Yu. 2025c. A survey on entropy mechanism in large reasoning models. *TechRxiv*, 2025(1220).

Bram Wasti, Wentao Ye, Teja Rao, Michael Goin, Paul Zhang, Tianyu Liu, Natalia Gimelshein, Woosuk Kwon, Kaichao You, and Zhuohan Li. 2025. No more train-inference mismatch: Bitwise consistent on-policy reinforcement learning with vllm and torchtitan. Accessed: 2025-12-21.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Haocheng Xi, Han Cai, Ligeng Zhu, Yao Lu, Kurt Keutzer, Jianfei Chen, and Song Han. 2024. Coat: Compressing optimizer states and activation for memory-efficient fp8 training. *arXiv preprint arXiv:2410.19313*.

Binxing Xu, Hao Gu, Lujun Li, Hao Wang, Bei Liu, Jiacheng Liu, Qiyuan Zhu, Xintong Yang, Chao Li, Sirui Han, and 1 others. 2026. Bit-by-bit: Progressive qat strategy with outlier channel splitting for stable low-bit llms. *arXiv preprint arXiv:2604.07888*.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. 2025. Learning to reason under off-policy guidance. *Preprint*, arXiv:2504.14945.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Feng Yao, Liyuan Liu, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. 2025. Your efficient rl framework secretly brings you off-policy rl training.

Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, and 1 others. 2020. Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 6672–6679.

Yixin Ye, Yang Xiao, Tiantian Mi, and Pengfei Liu. 2025. Aime-preview: A rigorous and immediate evaluation framework for advanced mathematical reasoning. `https://github.com/GAIR-NLP/AIME-Preview`. GitHub repository.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Ziyang Zhang, Xinheng Ding, Jiayi Yuan, Rixin Liu, Huizi Mao, Jiarong Xing, and Zirui Liu. 2025. Deterministic inference across tensor parallel sizes that eliminates training-inference mismatch. *Preprint*, arXiv:2511.17826.

Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Huazuo Gao, Jiashi Li, Liyue Zhang, Panpan Huang, Shangyan Zhou, Shirong Ma, and 1 others. 2025. Insights into deepseek-v3: Scaling challenges and reflections on hardware for ai architectures. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture*, pages 1731–1745.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, and 1 others. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

# Appdenix

## A The Use of Large Language Models (LLMs)

A large language model was utilized for grammatical and stylistic refinement of the manuscript. Its role was strictly limited to text editing and polishing to enhance clarity. All research ideas, experimental design, and analytical content are the original work of the authors.

## B Experiment Setting

**Training.** We use `Verl` (Sheng et al., 2024) as the training framework and `vLLM` (Kwon et al., 2023) as the inference engine. All experiments are conducted on $8\times$ NVIDIA H800 GPUs.

**Evaluation Dataset.** We evaluate our approach on standard in distribution mathematical datasets including AIME2024, AIME2025 (Ye et al., 2025), AMC, Math-500 (Lightman et al., 2023), OlympiadBench (He et al., 2024) and Minerva. To further investigate the generalizability of quantized training, we extend evaluation to out-of-distribution benchmarks such a out-of-distribution benchmarks ARC-Challenge (Clark et al., 2018), GPQA-Diamond (Rein et al., 2024), LiveCodeBench (Jain et al., 2024), MMLU Pro (Wang et al., 2024).

**Hyperparameters.**

- seq_clip_ratio_high is $\epsilon_h$,
- neg_seq_clip_ratio_high is $\delta_h$ and neg_seq_clip_ratio_low is $\delta_l$,
- seq_tis_imp_ratio_cap is $c$ in equation 5.

## C Batch invariant kernel

The root cause of output inconsistency across different kernel implementations lies in the finite precision of floating point accumulators. Due to the non associativity of floating point addition, where $(a + b) + c = a + (b + c)$ the final result is sensitive to the order of operations. To ensure bit-wise reproducibility, it is necessary to enforce a fixed summation order regardless of the batch configuration, leading to the design of a Batch-Invariant Kernel. However, imposing a strict execution order often comes at the cost of performance, as it restricts the asynchronous parallelism and dynamic scheduling inherent to GPU architectures.

| Parameter Name | Value |
|---|---|
| trainer.nnodes | 1 |
| trainer.n_gpu_per_node | 8 |
| data.train_batch_size | 512 |
| data.max_prompt_length | 2048 |
| data.max_response_length | 16384 |
| rollout.n | 8 |
| rollout.temperature | 1.0 |
| rollout.top_p | 1.0 |
| val_kwargs.temperature | 0.6 |
| actor.ppo_mini_batch_size | 64 |
| actor.ppo_max_token_len_per_gpu | 22528 |
| optim.opt_type | Muon |
| optim.lr | 1e-6 |
| optim.weight_decay | 0.01 |
| actor.use_kl_loss | False |
| actor.seq_clip_ratio_high | 0.0004 |
| actor.seq_clip_ratio_low | 0.0003 |
| actor.neg_seq_clip_ratio_high | 0.0007 |
| actor.neg_seq_clip_ratio_low | 0.0003 |
| actor.seq_tis_imp_ratio_cap | 2 |

Table 2: Hyperparameters for Experiment.

# D More Experimental Results

## D.1 Speed on different quant scheme

Fig. 11 illustrates the per-step training latency of various quantization schemes on Qwen3-30B-A3B (MoE), normalized to the BF16 baseline (dashed line at 1.0). Our results show that efficiency gains become increasingly significant from W8 to W4. This trend underscores that MoE training is primarily memory/IO-bound; since MoE operators are almost inherently memory-bound during decoding, the weight bit-width directly dictates computational efficiency. Furthermore, lower precision reduces the memory footprint per model instance, minimizing the required GPU count and alleviating inter-GPU communication overhead—a benefit that becomes even more pronounced when intra-node interconnects (e.g., NVLink) are fully utilized. Consequently, we adopt W4 as the default configuration for our primary experiments.

## D.2 More Ablation on TBPO

To further evaluate the effectiveness of TBPO, particularly its capability to mitigate the negative impact of "error tokens" generated by quantized rollout engines, we conduct a ablation study against GRPO under different precision settings.

| Method | Qwen3-8B-Base MATH-500 | AIME25 | AMC | Avg. |
|---|---|---|---|---|
| BF16 GRPO | 88.1 | 19.3 | 64.3 | 57.2 |
| BF16 TBPO | 86.3 | 20.0 | 62.4 | 56.2 |
| w4a16 GRPO | 82.2 | 12.5 | 53.0 | 49.2 |
| w4a16 TBPO | 84.9 | 13.3 | 58.3 | 52.1 |

Table 3: Performance comparison between GRPO and TBPO under full-precision (BF16) and quantized (w4a16) rollout training settings.

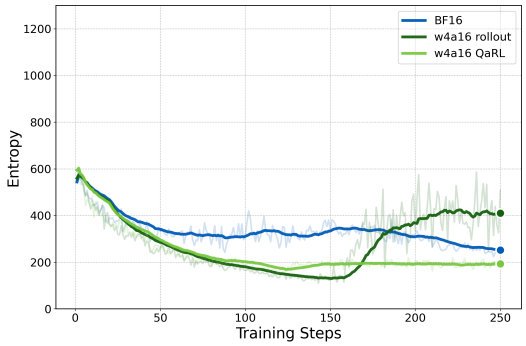


Figure 10: Comparison of RL training entropy

As demonstrated in Table 3, TBPO achieves performance on par with GRPO under BF16 precision. However, under w4a16 quantized rollout training, TBPO exhibits markedly superior robustness by effectively neutralizing error token interference during optimization, thereby delivering an average margin of 2.9 points over GRPO. While TBPO provides some degree of mitigation for the mismatch issues arising from quantized rollout training, the fundamental resolution lies in leveraging QAT/QaRL or fully quantized training regimes to eliminate systemic inconsistencies at their core.

## D.3 Entropy of quantized RL training

Whereas Huang et al. (2025) Fig. 5 posits that quantization errors during training may attenuate the

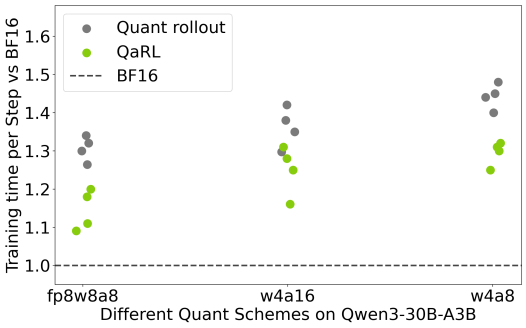


Figure 11: Different quant scheme speed on Qwen3-30B-A3B MoE model.

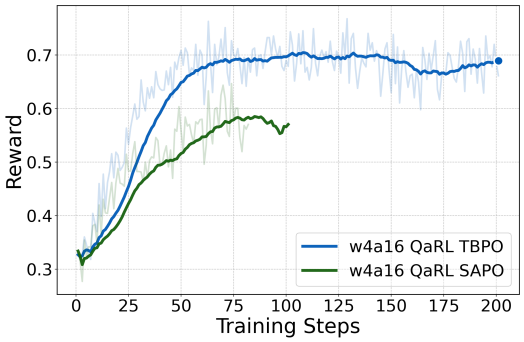


Figure 12: Comparison of SAPO on QaRL

entropy reduction in RL, our empirical findings present a more nuanced picture. Across both quantized rollout training and QAT/QaRL paradigms (Fig. 10), quantization discrepancies are progressively assimilated throughout the optimization process, yielding no discernible entropy elevation relative to the BF16 baseline. The entropy escalation observed in the latter stages of w4a16 quantized rollout training stems from the over-optimization of error tokens, which precipitates repetitive generation patterns.

### D.4 Comparsion of soft weighted methods SAPO

SAPO (Gao et al., 2025) introduces a soft adaptive weighting mechanism as a nuanced alternative to hard clipping, aiming to bolster optimization stability in MoE architectures. Through dynamic reweighting of token contributions—assigning diminishing importance to tokens whose advantage ratios deviate further from unity—SAPO adeptly discriminates between erroneous tokens (demanding substantial correction) and exploration tokens (that foster policy discovery). Nevertheless, as evidenced in Fig. 12, SAPO remains unable to surpass TBPO in the context of quantized RL training. We posit that this suboptimal performance stems from SAPO's continued assimilation of responses containing error tokens, despite attenuating their individual contributions to policy updates. Critically, such response sequences are globally off-distribution, rendering them intrinsically unsuitable for stable policy learning. This observation reaffirms the paramount importance of constraining sequence-level optimization objectives within the trust region, while simultaneously attesting to the elegance and principled simplicity inherent in TBPO's design.

## E Detail of TBPO

To effectively optimize the policy while maintaining training stability, we define the sequence-level importance weights and mismatch factors. Specifically, for a given query $q$ and its corresponding output sequence $o$ of length $L$, we introduce the sequence-level proximity ratio $r_{\text{seq-prox}}(\theta)$ and the sequence-level mismatch weight $w_{\text{seq-mismatch}}(\theta)$ as illustrated in Fig. 13.

Instead of using a simple product of token-level probabilities, which often leads to vanishing or exploding gradients in long-context scenarios, we employ the geometric mean of token-level ratios (formulated as the exponential of the average log-difference). This design ensures that the importance weights remain numerically stable across varying sequence lengths. $r_{\text{seq-prox}}(\theta)$ serves to constrain the policy update within a reliable trust region by measuring the drift from the previous learner, while $w_{\text{seq-mismatch}}(\theta)$ accounts for the distributional shift between the historical sampling policy and the current optimization baseline.

$$r_{\text{seq-prox}}(\theta) \triangleq \frac{\pi_{\text{current learner}}(o \mid q)}{\pi_{\text{old learner}}(o \mid q)} = \exp\left(\frac{1}{L}\sum_{t=1}^{L}(\log \pi_{\text{current learner}} - \log \pi_{\text{old learner}})(o_t \mid q, o_{<t})\right),$$

$$w_{\text{seq-mismatch}}(\theta) \triangleq \frac{\pi_{\text{old learner}}(o \mid q)}{\pi_{\text{old sampler}}(o \mid q)} = \exp\left(\frac{1}{L}\sum_{t=1}^{L}(\log \pi_{\text{old learner}} - \log \pi_{\text{old sampler}})(o_t \mid q, o_{<t})\right).$$

Figure 13: Sequence ratio and weight definition.

**GRPO token level clipping & weighting**

Response1: Use quantized rollout engine ~~to to to to to to to to to to to to to~~ accelerate RL wide sees gr $_1$ ~~ContributionsĠD±ĐμD~~

Response2: Train -inference mismatch is a primary cause of training collapse in RL

Response3: ~~Aha~~ $_2$ ! We can clip error token to ~~keep~~ optimization in trust region

**TBPO sequence level clipping & weighting**

Response1: ~~Use quantized rollout engineto to to to to to to to to to to to to accelerate RL wide sees gr ContributionsĠD±ĐμD~~

Response2: Train-inference mismatch is a primary cause of training collapse in RL

Response3: Aha! We can clip error token to keep optimization in trust region

Table 6: Comparison of clipping and weighting strategies. Gray highlights clipped tokens, while different colors represent varying weighting magnitudes (darker shades indicate weights further from 1). GRPO employs token-level clipping and weighting, whereas TBPO utilizes sequence-level granularity. In GRPO, the garbled tokens (marked by 1) are not clipped, while the exploration tokens (marked by 2) are clipped. In contrast, TBPO's sequence-level approach correctly clips the garbled tokens without erroneously clipping the exploration tokens.