FreeQuant: Bridging Training-Inference Mismatch in Quantized RL via Near-Zero-Overhead Calibration

Anonymous ACL submission

Abstract

Reinforcement Learning for LLMs suffers from a computational bottleneck during rollout generation. While deploying quantized inference engines accelerates this phase, it introduces a training-inference mismatch between the low-precision behavior policy and the high-precision target policy, often destabilizing training. We propose FreeQuant, a framework that alleviate this discrepancy via near-zero-overhead calibration. By observing that the RL training loop inherently necessitates a full-precision forward pass, FreeQuant seamlessly reuses these intermediate activations for on-the-fly calibration. Extensive experiments demonstrate that FreeQuant successfully preserves the training stability and terminal performance of full-precision RL while fully retaining the throughput acceleration of quantized rollout.

1 Introduction

Reinforcement Learning (RL) like PPO (Schulman et al., 2017) or GRPO (DeepSeek-AI et al., 2025) has become the cornerstone for enhancing capabilities of Large Language Models (LLMs). However, RL scaling presents significant computational and memory challenges. The rollout generation phase, where the policy interacts with the environment to sample trajectories, constitutes a severe computational bottleneck. This bottleneck is exceptionally pronounced when training large-scaled Mixture-of-Experts models long-horizon tasks, where the sequential autoregressive generation severely limit the overall training throughput.

To mitigate this bottleneck, RL training frameworks commonly introduce quantization in the inference engine, utilizing quantization formats (e.g., INT4/FP8) to accelerate rollout generation. Unfortunately, this optimization comes at a cost: it triggers a severe *Training-Inference Mismatch* (He and Lab, 2025; Yao et al., 2025; Li et al., 2026). A discrepancy emerges between the low-precision behavior policy used for rollout generation and the high-precision target policy maintained for policy update. This discrepancy often leads to training instability and suboptimal convergence. While Post-Training Quantization (PTQ) techniques are standard solutions for reducing these quantization errors, they are ill-suited for the dynamic nature of RL. As the policy's parameters are updated each turn, performing frequent recalibration imposes prohibitive computational overhead, entirely negating the efficiency gains of quantization.

To break this dilemma, we propose FreeQuant. Our core insight stems from the characteristic of the RL training loop: the log-probability (LogProb) computation phase, which immediately precedes the policy update, inherently necessitates a high-precision forward pass over the recently sampled in-domain trajectories. Building upon this observation, we introduce a activation-reuse mechanism. Instead of executing separate forward passes for PTQ, FreeQuant seamlessly reuses the real-time activations generated during the LogProb computation, performing on-the-fly calibration of the weight quantization parameters for the subsequent rollout step. This design alleviates the discrepancy between the inference engine and the training framework with near-zero overhead.

We conduct extensive experiments across multiple settings. The results demonstrate that FreeQuant significantly accelerates rollout throughput while maintaining the training stability and terminal performance of full-precision RL training.

2 Preliminary and Motivation

2.1 Off-Policy Discrepancy in Reinforcement Learning

The RL training pipeline of LLMs (Sheng et al., 2024; Zhu et al., 2025) iteratively alternates among four distinct phases. (1) *Rollout Generation Phase*, the model is typically deployed in an inference

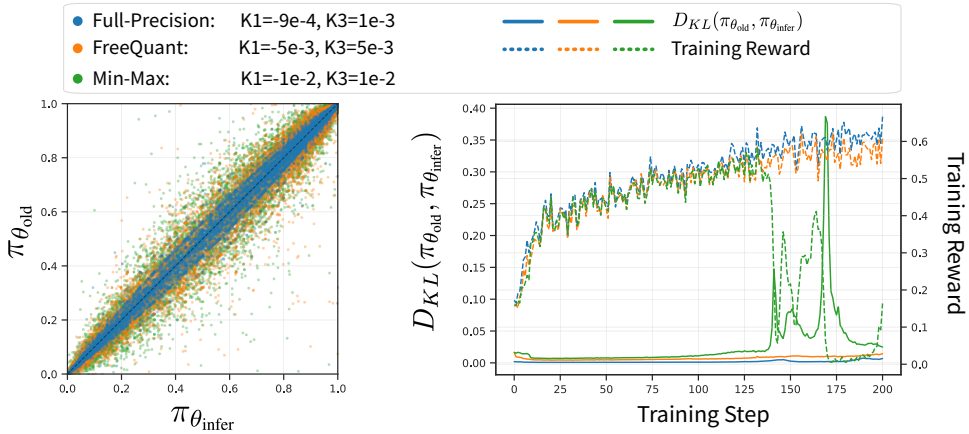

Figure 1: (Left) Token-level Training-Inference Mismatch between $\pi_{\theta_{\text{infer}}}$ and $\pi_{\theta_{\text{old}}}$. Each point is a token sampled during rl training. From top to down (Full-Precision $\rightarrow$ FreeQuant $\rightarrow$ Min-Max), the KL divergence estimated by K1 and K3 estimator becomes larger. (Right) The training dynamics of $D_{KL}(\pi_{\theta_{\text{old}}}, \pi_{\theta_{\text{infer}}})$ and training reward. Min-Max suffers from training instability because of the larger Training-Inference Mismatch.

engine utilizing low-precision formats to maximize throughput. We denote this behavior policy as $\pi_{\theta_{\text{infer}}}$, from which trajectories $\tau \sim \pi_{\theta_{\text{infer}}}(\cdot)$ are sampled. (2) *LogProb Computation Phase*, those trajectories are forwarded through the training framework to calculate the proximal probabilities before update, denoted as $\pi_{\theta_{\text{old}}}(\tau)$ (Hilton et al., 2022). (3) *Policy Update Phase*, the target policy π_θ is updated over multiple optimization steps using policy gradients. (4) *Weight Synchronization Phase* the target policy is synchronized to inference engine for the next round rollout generation.

Formally, the general form of RL surrogate objective can be expressed as follows:

$$\mathcal{J}(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta_{\text{infer}}}} \left[\frac{\pi_\theta(\tau)}{\pi_{\theta_{\text{old}}}(\tau)} \cdot \frac{\pi_{\theta_{\text{old}}}(\tau)}{\pi_{\theta_{\text{infer}}}(\tau)} \cdot R(\tau) \right],$$

where $R(\tau)$ is the reward. This decomposition intuitively reveals two different types of discrepancy between the target policy to optimize and the behavior policy for rollout generation.

Algorithmic Off-Policy: The first ratio term, $\frac{\pi_\theta(\tau)}{\pi_{\theta_{\text{old}}}(\tau)}$, represents the standard distributional shift resulting from the iterative parameter updates during the Policy Update Phase. This discrepancy is typically mitigated by clipping mechanisms in algorithms like PPO (Schulman et al., 2017).

Training-Inference Mismatch: The second ratio term, $\frac{\pi_{\theta_{\text{old}}}(\tau)}{\pi_{\theta_{\text{infer}}}(\tau)}$, captures the gap between inference engines (e.g., vLLM) and training frameworks (e.g., Megatron). Though utilizing identical weights, quantization is commonly adopted in the inference engine to improve throughput. A standard symmetric uniform k-bit quantization function for a given weight tensor x is defined as:

$$q = \text{clip}\left(\lfloor \frac{x}{s} \rceil + z, q_{\min}, q_{\max} \right), \; Q_s(x) = s \cdot (q - z),$$

where s denotes the quantization scaling factor, $z = 2^{k-1}$ represents the zero point, $[q_{\min} = 0, q_{\max} = 2^k - 1]$ is the range of the target low-precision data type, and q is the quantized parameter. Evidently, the quantization reconstruction $Q_s(x)$ inherently incurs precision loss. This underlying quantization error causes the inference policy distribution $\pi_{\theta_{\text{infer}}}$ to diverge significantly from its full-precision counterpart $\pi_{\theta_{\text{old}}}$. While techniques like Truncated Importance Sampling (TIS) (Yao et al., 2025) are employed to bound $\frac{\pi_{\theta_{\text{old}}}(\tau)}{\pi_{\theta_{\text{infer}}}(\tau)}$, they merely serve as a post-hoc correction rather than addressing the underlying quantization error, still leading to training instability as shown in Figure 1.

2.2 Quantization and the PTQ Dilemma

As shown above, the core determinant of the quantization error is the optimal derivation of the scaling factor s, which heavily relies on the underlying activation distribution (Frantar et al., 2023a). The most direct approach is Post-Training Quantization (PTQ), which aims at reconstructing the high-precision policy's output space.

However, the dynamic nature of RL differs from the traditional scenarios of quantization paradigms, where the policy updates frequently. This creates a challenging dilemma for existing quantization methods. *Min-Max PTQ* is the most trivial method without calibration, simply computing the scale from the extreme values of weights: $s = \frac{\max(x) - \min(x)}{q_{\max} - q_{\min}}$. However, it completely ignores the activation distribution. Since LLM activations are notoriously plagued by massive out-

Algorithm 1 FreeQuant

Input: Initial full-precision weights $\boldsymbol{W}^{(0)}$; Initial scaling factors $S^{(0)}$; A quantization method deriving optimal quantization from activations $f(\cdot)$; Total RL iterations T.
Output: Optimized full-precision weights $\boldsymbol{W}^{(T)}$
for $t \leftarrow 0$ **to** $T-1$ **do**
  `// Phase 1: Rollout Generation`
  Sample $\tau^{(t)} \sim \pi^{(t)}_{\theta_{\text{infer}}}$ in the inference engine
  `// Phase 2: LogProb Computation`
  Forward $\tau^{(t)}$ using full-precision weights to compute log-probabilities $\pi_{\theta_{\text{old}}}(\tau^{(t)})$
  Cache the activations $\mathcal{A}^{(t)}$ **for all target weights**
  `// Phase 3: Policy Update`
  Compute RL optimization objective $\mathcal{J}(\theta)$
  Update full-precision weights $\theta^{(t+1)}$ via gradient ascent
  $\theta^{(t+1)}_{\text{old}} \leftarrow \theta^{(t+1)}$
  `// Phase 4: Weight Synchronization`
  for $\boldsymbol{W}^{(t+1)}$ *in* $\theta^{(t+1)}$ **do**
    Derive the optimal quantization weight with the cached activation $s, q^{(t+1)} \leftarrow f(\boldsymbol{A}^{(t)}, \boldsymbol{W}^{(t+1)})$
    Sync quantized weights $q^{(t+1)}$ and scale s to the inference engine to update $\pi^{(t+1)}_{\theta_{\text{infer}}}$
  end
end

liers (Sun et al., 2024), treating all weights equally yields poor precision. *Calibration-based PTQ* achieves high precision by deriving s dynamically based on the input activations to directly optimize the quantization error (Xiao et al., 2024; Frantar et al., 2023b; Lin et al., 2026a). Specifically, the quantization error (Frantar et al., 2023a) for a linear layer $\boldsymbol{W} \in \mathbb{R}^{o \times h}$ is defined as:

$$\mathcal{L}(s) = ||\boldsymbol{A}(Q_s(\boldsymbol{W}) - \boldsymbol{W})^T||_F^2, \qquad (1)$$

where $\boldsymbol{A} \in \mathbb{R}^{t \times h}$ denotes the input activations of calibrated data. Yet, executing additional forward passes over calibration data introduces prohibitive computational overhead, negating the latency reductions that quantization was intended to provide.

3 FreeQuant

3.1 Calibration via Activation Reuse

As discussed in Section 2.2, the primary bottleneck hindering the direct application of calibration-based PTQ in RL is the computational overhead of calibration. However, we observe an opportunity within the LogProb Computation Phase, where a "free" full-precision forward pass is conduct over the exact in-domain "calibration" data. Building upon this insight, we propose FreeQuant, a novel framework that seamlessly alleviates the Training-Inference Mismatch with near-zero computational overhead. Specifically, we use the pre-forward hook mechanism in Pytorch to collect these intermediate activations. Subsequently during Weight Synchronization Phase, we can apply any calibration-based PTQ methods f, deriving the optimal scale s and quantized weight q based on collected activations. We present the entire FreeQuant pipeline in Algorithm 1.

3.2 Implementation: Near-Zero Overhead via NeUQI

This FreeQuant pipeline is *agnostic* to the chosen quantization method f, and there is a trade-off between quantization error and computation overhead. For instance, optimizing the objective in Equation 1 using algorithms like GPTQ (Frantar et al., 2023a) requires computing and inverting the full Hessian matrix, which is costly.

In this work, we favor latency in order to compress the calibration overhead to a minimum, maximizing the superiority of quantized RL training. Therefore, we integrates NeUQI (Lin et al., 2026b) as the underlying quantization method. Similar to other PTQ methods, NeUQI also aims at optimizing the quantization loss:

$$\begin{aligned}\mathcal{L}(s) &= ||\boldsymbol{A}(Q_s(\boldsymbol{W}) - \boldsymbol{W})^T||_F^2 \\ &= \sum_i (Q_s(\boldsymbol{W}_i) - \boldsymbol{W}_i)\boldsymbol{H}(Q_s(\boldsymbol{W}_i) - \boldsymbol{W}_i)^T \\ &\approx \sum_{i=1}^{h}\sum_{j=1}^{o} \boldsymbol{H}_{i,i}(Q_s(\boldsymbol{W}_{j,i}) - \boldsymbol{W}_{j,i})^2,\end{aligned}$$

where $\boldsymbol{H} = \boldsymbol{A}^T\boldsymbol{A}$ is the Hessian matrix, and the last approximation applied is *diagonal approximation*. The loss then becomes a piecewise quadratic functions of s. And NeUQI proposes a $O(oh2^k \log(h2^k))$ algorithm, where h is the input size, o is the output size and k is the quantization bit-width. Combining with other optimization technique, such as communication-computation overlap and multi-processing quantization, we demonstrate a near-zero overhead compared with the Minmax quantization baseline.

4 Experiments

4.1 Experimental Settings

Models and Datasets. We employ MoE models of varying scales for evaluation: Qwen3-30B-A3B, Qwen3.5-35B-A3B, and Qwen3-Next-80B-A3B (Yang et al., 2025; Team, 2026). Following standard practices for MoE quantization (Team et al., 2026), we exclusively quantize the expert components (i.e. Up and Down Projections). We

Metric	Full-Precision	Min-Max	FreeQuant	QAT
Qwen3-30B-A3B				
AIME25	**26.25**	17.50	22.71	23.33
AIME24	**39.58**	26.67	33.33	30.42
Time (s)	459.76	**300.35**	305.47	310.09
Qwen3-Next-80B-A3B				
AIME25	**78.83**	78.75	78.33	76.25
AIME24	**88.33**	85.42	**88.33**	86.67
Time (s)	2126.81	**1517.08**	1538.58	2174.97
Qwen3-5-35B-A3B				
AIME25	**85.17**	83.75	85.00	81.67
AIME24	**92.42**	89.17	92.08	91.25
Time (s)	21755.53	1919.07	**1915.49**	2282.91

Table 1: Comparison between different mixed-precision training methods.

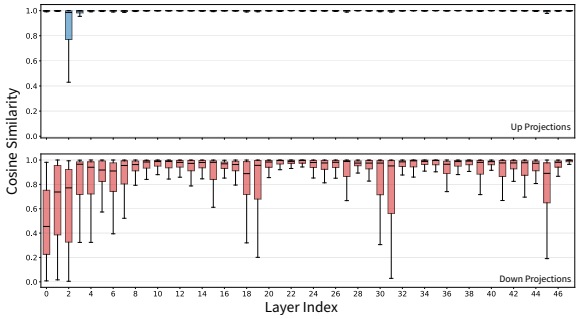

Figure 2: Cosine similarity of layer-wise activations between consecutive RL iterations.

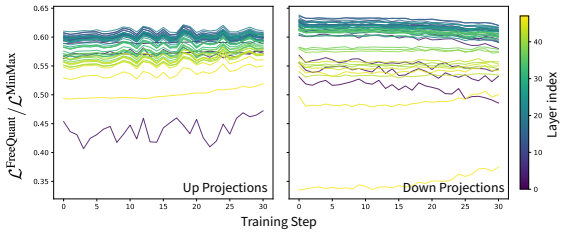

Figure 3: Ratio of quantization loss between FreeQuant and MinMax (i.e. $\mathcal{L}^{\text{FreeQuant}}/\mathcal{L}^{\text{MinMax}}$).

adopt DAPO (Yu et al., 2025) augmented with TIS (Yao et al., 2025) as our base RL algorithm. We utilize the DAPO-Math-17k dataset as the training corpus, while using AIME24 and AIME25 benchmarks (Avg@8) as the test set.

Baselines. We establish the training pipeline with `slime` (Zhu et al., 2025). We comprehensively benchmark three other baselines: **Full-Precision:** The standard, `BF16` RL pipeline. **Min-Max:** A trivial PTQ approach as formulated in Section 2.2. **QAT:** An approach orthogonal to PTQ that integrates fake quantization nodes into the computational graph to simulate `INT4` parameter space during the policy update (Jacob et al., 2017).

4.2 Analysis

Main Results. As shown in Table 1, while Full-Precision establishes the empirical upper bound for terminal accuracy, it incurs prohibitive computational costs. Conversely, naive Min-Max quantization dramatically accelerates rollouts but suffers from precision-induced instability. FreeQuant successfully resolves the performance-efficiency bottleneck. It almost matches the Full-Precision upperbound with a marginal $\sim 1\%$ time overhead compared to the Min-Max, effectively offering a free lunch for scaled RL training.

Validity of Lagged Calibration. We want to discuss whether the activations from the previous iteration can approximate the activation distribution of the updated behaviour policy. We visualize the cosine similarity of layer-wise activations between consecutive RL iterations across the MoE expert layers in Figure 2. The results show that the similarities remain remarkably high. Especially for up projections, it exhibit near-perfect alignment (~ 0.99). This phenomenon is not coincidental. DAPO explicitly enforce trust regions to strictly bound the magnitude of policy updates, ensuring the transition from $\pi_\theta^{(t)}$ to $\pi_\theta^{(t+1)}$ is highly localized and smooth. Consequently, the data distribution shift is incremental, yielding $\boldsymbol{A}^{(t)} \approx \boldsymbol{A}^{(t+1)}$ as a highly reliable assumption.

Superiority of Dynamic Calibration over Min-Max. To further illustrate why dynamic calibration is indispensable, we compare the layer-wise quantization loss under FreeQuant ($\mathcal{L}^{\text{FreeQuant}}$) and Min-Max ($\mathcal{L}^{\text{MinMax}}$). As depicted in Figure 3, applying FreeQuant reduces the quantization error to approximately 60% of the Min-Max baseline, with deeper network layers exhibiting even more pronounced benefits.

5 Conclusion

We introduced FreeQuant to alleviate the training-inference mismatch in quantized RL. Our core philosophy is to repurpose the "free-lunch" activations from the mandatory log-probability computation phase for near-zero-overhead dynamic calibration. Future work will explore more advanced strategies to exploit these real-time activations for low-bit training and distribution shift mitigation.

Limitations

This work primarily focuses on weight-only quantization and is empirically evaluated on text-based alignment and long-horizon reasoning tasks. However, the core methodological philosophy of FreeQuant—repurposing the mandatory log-probability computation phase to collect "free-lunch" activations—is inherently applicable to more aggressive quantization dimensions, such as activation and KV cache quantization. Furthermore, our framework is inherently generalizable to broader reinforcement learning environments, including multimodal or multi-agent RL, provided that their training loops necessitate a high-precision forward pass over the sampled trajectories.

6 Ethical considerations

This work utilizes Large Language Models (LLMs) to assist in writing refinement and code generation for result visualization. All models and datasets employed in this study are publicly available and widely used within the research community. There are no ethical concerns regarding data privacy or copyright infringement associated with the resources used in this research.

References

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint*. ArXiv:2501.12948 [cs].

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023a. Gptq: Accurate post-training quantization for generative pre-trained transformers. *Preprint*, arXiv:2210.17323.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023b. Gptq: Accurate post-training quantization for generative pre-trained transformers. *Preprint*, arXiv:2210.17323.

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. *Preprint*, arXiv:2110.00641.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2017. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Preprint*, arXiv:1712.05877.

Yingru Li, Jiacai Liu, Jiawei Xu, Yuxuan Tong, Ziniu Li, Qian Liu, and Baoxiang Wang. 2026. Trust region masking for long-horizon llm reinforcement learning. *Preprint*, arXiv:2512.23075.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2026a. Awq: Activation-aware weight quantization for llm compression and acceleration. *Preprint*, arXiv:2306.00978.

Li Lin, Xinyu Hu, and Xiaojun Wan. 2026b. Neuqi: Near-optimal uniform quantization parameter initialization for low-bit llms. *Preprint*, arXiv:2505.17595.

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

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*.

Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. 2024. Massive activations in large language models. *Preprint*, arXiv:2402.17762.

Kimi Team, Yifan Bai, Yiping Bao, Y. Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, and 181 others. 2026. Kimi k2: Open agentic intelligence. *Preprint*, arXiv:2507.20534.

Qwen Team. 2026. Qwen3.5-omni technical report. *Preprint*, arXiv:2604.15804.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2024. Smoothquant: Accurate and efficient post-training quantization for large language models. *Preprint*, arXiv:2211.10438.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *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.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. 2025. slime: An llm post-training framework for rl scaling. https://github.com/THUDM/slime. GitHub repository. Corresponding author: Xin Lv.

A Experimental Details

Models and Datasets. To evaluate the efficacy of MyMethod, we employ a suite of open-weight Mixture-of-Experts (MoE) models of varying scales: Qwen3-30B-A3B, Qwen3.5-35B-A3B, and Qwen3-Next-80B-A3B (Yang et al., 2025; Team, 2026). We adopt DAPO (Yu et al., 2025) augmented with Truncated Importance Sampling (TIS) (Yao et al., 2025) as our base RL framework. This combination robustly handles policy optimization by applying dual clipping mechanisms that simultaneously mitigate the *Algorithmic Off-Policy* shift and the *Training-Inference Mismatch*. For the training corpus, we utilize the DAPO-Math-17k dataset. The terminal reasoning capabilities of the models are evaluated using the Avg@8 (Pass@8) accuracy on the challenging AIME24 and AIME25 benchmarks.

Quantization Settings and Baselines. We establish a rigorous mixed-precision training paradigm utilizing the `slime` framework (Zhu et al., 2025). To maintain gradient fidelity, the training framework operates entirely in `BF16`, whereas the inference engine executes rollout generation in `INT4` to maximize throughput. The fundamental quantization scheme employed at the inference stage is standard symmetric uniform quantization. We comprehensively benchmark MyMethod against three representative paradigms:

- **Full-Precision (`BF16`):** The standard, unquantized RL pipeline. This serves as the strict empirical upper bound for training stability and terminal performance.
- **Min-Max PTQ:** A trivial `INT4` Post-Training Quantization approach (as formulated in Section 2.2) that calculates scaling factors based solely on the extreme minimum and maximum values of the weights, completely omitting activation-based calibration.
- **Quantization-Aware Training (QAT):** An approach orthogonal to PTQ that integrates fake quantization nodes into the computational graph and utilizes the Straight-Through Estimator (STE) to approximate gradients. This explicitly simulates `INT4` parameter space constraints during the policy update's forward and backward passes.

Implementation Details. All experiments are conduct on 2×8 A100 GPUs. To rigorously evaluate system efficiency, we optimize the training hyperparameters for each model scale to maximize throughput and minimize latency while strictly preventing Out-of-Memory (OOM) errors. These configurations are detailed in Table 2.

To realize the lagged calibration of MyMethod, we attach `torch.forward_hook` to the target layers, enabling the zero-overhead caching of local activations during the mandatory `BF16` LogProb computation. Crucially, during the weight synchronization phase, we execute a distributed quantization strategy. Local activations are directly utilized to compute local quantization parameters, which are subsequently aggregated globally via an `all-reduce` operation. This design significantly curtails both computational latency and communication bottlenecks. Notably, this distributed calibration requires no additional synchronization across Pipeline Parallelism (PP) or Expert Parallelism (EP) ranks; activation aggregation via `all-gather` and `all-reduce` is exclusively required across Data Parallelism (DP) and row-parallel Tensor Parallelism (TP) groups. We implement the NeUQI quantization kernels for FreeQuant, and reuse the Min-Max kernels in slime for both Min-Max and QAT baselines.

	Qwen3-30B-A3B		Qwen-Next-80B-A3B		Qwen3.5-35B-A3B	
	Full	Mixed	Full	Mixed	Full	Mixed
tensor-model-parallel-size	4	4	2	2	2	2
pipeline-model-parallel-size	1	1	4	4	1	1
context-parallel-size	1	1	1	1	4	4
2						
expert-model-parallel-size	8	8	8	8	8	8
expert-tensor-parallel-size	1	1	1	1	1	1
rollout-num-gpus-per-engine	8	1	8	1	8	1
max-tokens-per-gpu	32768	32768	16384	16384	4096	8192

Table 2: Hyper-parameters for different methods.

B NeUQI Technique

Tthe optimal scale for each row of weight in $\mathcal{L}(s)$ can be solved independently. Therefore, it suffices to consider a single row only. Let $x \in \mathbb{R}^h$ denote the weight vector of one row, and let $b \in \mathbb{R}^h$ denote

the corresponding error coefficient, where $b_i = H_{i,i}$. Then,

$$\mathcal{L}_{\text{row}}(s) = \sum_{i=1}^{h} b_i \big(Q_s(x_i) - x_i\big)^2.$$

Following the same derivation as in NeUQI, $\big(Q_s(x_i) - x_i\big)^2$ is a piecewise quadratic function of s. The breakpoints occur when $x_i/s + z$ reaches the midpoints of consecutive quantization levels in $[q_{\min}, q_{\max}]$. Since there are 2^k quantization levels, each $\big(Q_s(x_i) - x_i\big)^2$ contains $2^k - 1$ breakpoints. Consequently, $\mathcal{L}_{\text{row}}(s)$ is also a piecewise quadratic function of s, where the set of breakpoints is given by the union of the breakpoints of all $\big(Q_s(x_i) - x_i\big)^2$. Therefore, by applying Algorithm 1 in NeUQI, the quadratic coefficients of each segment can be computed efficiently, which enables obtaining the exact optimum of $\mathcal{L}_{\text{row}}(s)$ in $O(2^k h \log(2^k h))$ time.