QCRL: QUANTIZATION-CONSISTENT REINFORCEMENT LEARNING FOR LLM REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Obtaining a capable W4A4 reasoning model through reinforcement learning with verifiable rewards remains challenging because direct W4A4 optimization suffers from rounding-induced reward instability and rollout-optimization policy mismatch. We propose Quantization-Consistent Reinforcement Learning (QCRL), a dual-view framework that combines two strategies, Quantization-Consistent Rollout and Quantization-Weight Dampening, to improve the final W4A4 policy. Specifically, the rollout strategy collects W4A16 completions from round-to-nearest and stochastic-rounding views of shared latent weights and calibrates their rewards through cross-view comparisons and within-view support to guide W4A4 policy optimization. The weight-dampening strategy stabilizes weights near their quantization centers after parameter updates. On GSM8K with Qwen2.5-3B, QCRL achieves 78.32% pass@1 under W4A4 evaluation, surpassing BF16 reinforcement learning followed by W4A4 post-training quantization by 2.28 percentage points and direct W4A4 reinforcement learning by 4.25 percentage points. These results support reinforcement learning for stronger W4A4 reasoning models while retaining compatibility with W4A16 rollout.

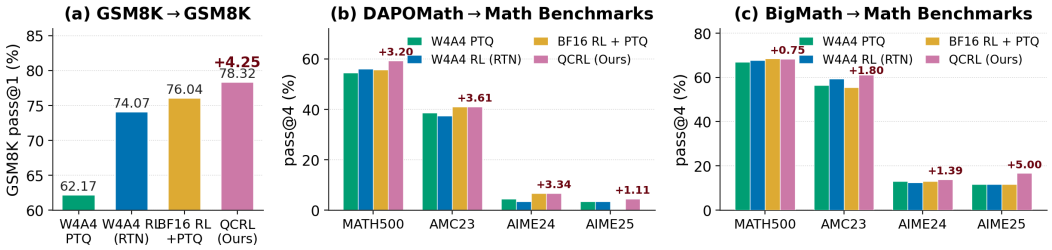

Figure 1: Main results across training settings. **(a)** GSM8K pass@1 on Qwen2.5-3B. **(b,c)** Pass@4 on four mathematical test sets after DAPOMath and BigMath training, respectively; annotations report QCRL gains over W4A4 RL (RTN).

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) improves the reasoning capabilities of large language models (LLMs) on tasks with checkable answers, including mathematics. Our goal is to retain this capability in a model whose quantized linear layers use 4-bit weights and 4-bit activations (W4A4). Recent low-precision RL methods explore reducing rollout cost and improving quantized policies through training Xi et al. (2026); Huang et al. (2026); Gu et al. (2026). We optimize the policy with the target W4A4 quantizer to improve the final reasoning model and compare its quality with that of a model obtained by applying post-training quantization (PTQ) Frantar et al. (2023); Ashkboos et al. (2024) after BF16 RL. We adopt stage-specific precision, using W4A16 behavior policies for rollout generation and W4A4 quantization for policy optimization and final evaluation.

Recently, QeRL Huang et al. (2026) showed that quantization noise can increase policy entropy and benefit exploration at W4A16. However, under W4A4 this benefit disappears. Direct W4A4

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

RLVR underperforms the simple baseline of PTQ after full-precision RL (Table 1), because aggressive rounding injects reward variation that obscures the true policy signal. In low-bit settings, the discrepancy between the quantized rollout policies and the policy used for optimization becomes more consequential, making rollout-level rewards less reliable for optimization. We measure two effects of decreasing precision. Fixed-trajectory log-probability deviation and update-direction conflict quantify numerical policy displacement relative to BF16 under teacher forcing, whereas excess reward disagreement quantifies rollout-outcome instability across quantized views (Figure 2). Nearly one-quarter of the rewards under W4A4 are inconsistent with their BF16 counterparts. This gap motivates a training design that preserves quantization-native exploration without external noise while stabilizing the rollout evidence used for W4A4 optimization.

We propose Quantization-Consistent Reinforcement Learning (QCRL) to improve the reasoning capability of the final W4A4 policy. QCRL constructs RTN and SR rollout views from the same latent weights and uses both views to train a W4A4 learner. Rollouts retain W4A16 precision for compatibility with weight-only inference acceleration, while policy optimization uses the target W4A4 quantizer. Quantization-Consistent Rollout (QCR) combines this dual-view sampling structure with reward calibration based on cross-view comparisons and within-view support. Quantization-Weight Dampening (QWD) complements QCR by stabilizing weight-quantization code assignments through periodic post-optimizer corrections.

For each prompt, QCR divides a fixed completion budget equally between the frozen RTN and SR views. It applies a cross-view penalty when a completion's reward falls below the opposite view's mean and adds within-view support to a correct completion when other completions from the same view are also correct. The resulting reward replaces the verifier reward as the input to group-based advantage estimation. This calibration adjusts training credit using view-conditioned outcome statistics; it does not change the verifier's correctness labels or identify a noise-free reward for each completion. The RL loss, clipping procedure, and optimizer remain unchanged.

QWD operates after parameter updates by detecting weights that have crossed a quantization decision boundary and gently pulling them back toward their current quantization centers. This reduces oscillatory switching and complements QCR's noise suppression, ensuring that policy gradients are computed under stable quantized representations.

We evaluate QCRL on language models from 1.5B to 7B across multiple arithmetic benchmarks. On GSM8K with Qwen2.5-3B, QCRL achieves 78.32% pass@1, outperforming BF16 RL + PTQ by 2.28 points and vanilla W4A4 RL by 4.25 points. Training dynamics reveal that QCRL initially raises policy entropy above the vanilla W4A4 level but later converges to the BF16 baseline's entropy, demonstrating a favorable balance between exploration and consolidation.

Our contributions are:

- We introduce stochastic rounding as a quantization-native exploration source for W4A4 RL and pair it with deterministic RTN rollouts to form a dual-view rollout structure over the same latent policy;
- We develop QCRL, comprising QCR for view-aware reward calibration and QWD for weight stabilization, which jointly enable robust W4A4 policy optimization under any group-based RLVR algorithm;
- We show that QCRL achieves strong W4A4 accuracy and, through entropy analysis, demonstrates effective exploration without sacrificing convergence.

2 Related Work

2.1 Reinforcement Learning for LLM Reasoning

Reinforcement learning has become a central post-training technique for improving LLM reasoning. DeepSeekMath Shao et al. (2024) introduced Group Relative Policy Optimization (GRPO), which removes the learned value model and estimates advantages from sampled response groups. DAPO Yu et al. (2025) further refines this value-model-free RLVR recipe through decoupled clipping and dynamic sampling, making large-scale reasoning RL more reproducible. A complementary line revisits value-based RLVR. VAPO Yue et al. (2025) addresses value-model bias, heterogeneous

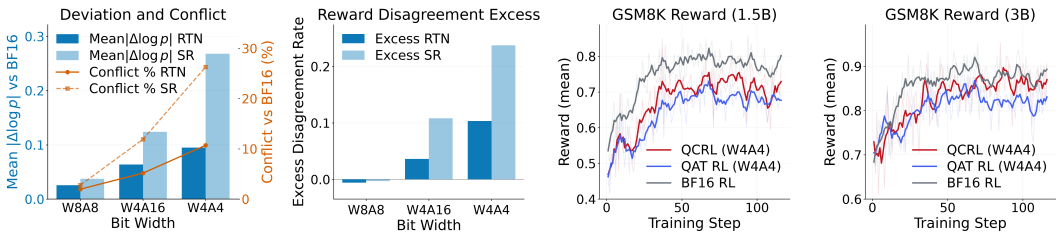

Figure 2: Quantization sensitivity and training dynamics. **(a)** Fixed-trajectory log-probability deviation and token-confidence conflict relative to BF16 under teacher forcing. **(b)** Sampling-noise-corrected excess reward disagreement for RTN and SR across precisions. **(c,d)** GSM8K reward trajectories of QCRL, W4A4 RL, and BF16 RL on Qwen2.5-1.5B and 3B.

response lengths, and sparse verifier rewards to improve the stability and efficiency of long-chain reasoning training. Together, these methods show that RLVR depends not only on the reward verifier but also on how rollouts are sampled, grouped, and credited, which becomes the central reliability issue when the rollout policy is aggressively quantized.

2.2 Quantization and Reinforcement Learning

Prior work Qiu et al. (2026); Li et al. (2026a) has explored quantization in reinforcement learning primarily from the perspective of rollout efficiency. QuaRL Krishnan et al. (2022) applies quantization to rollout policies for faster and more energy-efficient distributed RL training. More recent LLM-focused methods use quantization to accelerate the rollout bottleneck in RLVR. QeRL Huang et al. (2026) combines NVFP4 quantization with LoRA-based RL training and argues that adaptive quantization noise can improve exploration. QuRL Li et al. (2026b) uses a quantized rollout policy and addresses mismatch between quantized and full-precision policies through adaptive clipping and invariant scaling. QUADS Zhuge et al. (2026) stabilizes native NVFP4 W4A4 rollout for MoE RL through trainer-side W4A16 QAT and rollout-side residual activation compensation. These methods primarily target rollout acceleration and rollout-trainer alignment, often while retaining high-precision parameters or training states. QCRL follows a different route that retains the rollout-side benefit of low precision while directly optimizing and evaluating the output policy at W4A4, producing a quantized model with high downstream task capability. And unlike QeRL, we do not inject random Gaussian noise; instead, we use stochastic rounding (SR) to introduce noise intrinsic to the quantization process.

A separate line of quantization-aware training (QAT) NVIDIA et al. (2026); Liu et al. (2024); Song et al. (2026) is relevant to how the weights behave under this optimization. Nagel et al. (2022) shows that training a quantized network through a straight-through estimator induces weight oscillation near quantization boundaries, and propose a dampening penalty to suppress it. This work is set in supervised QAT with static per-tensor scales; QCRL adapts its dampening idea to dynamic per-group quantization under RL, which is the role of QWD.

3 Preliminaries

3.1 Group-Based RLVR Optimization

Let π_θ denote the trainable policy, p a prompt, and y_i is the i-th sampled completion. A verifier assigns each completion a response-level task reward R_i. Value-model-free, group-based RLVR methods sample G completions for the same prompt and use their relative outcomes to estimate credit. In the group-normalized estimator used in our experiments, the completion-level advantage is

$$\hat{A}_i = \frac{R_i - \mathrm{mean}(\{R_j\}_{j=1}^{G})}{\mathrm{std}(\{R_j\}_{j=1}^{G})}. \tag{1}$$

The group mean and standard deviation therefore provide a prompt-specific baseline for assigning relative credit.

Each completion is generated by a fixed quantized rollout policy. For token t of completion i, the policy update uses the importance ratio

$$r_{i,t}(\theta) = \exp\left(\ell_{i,t}^{\text{new}}(\theta) - \ell_{i,t}^{\text{beh}}\right), \tag{2}$$

where $\ell_{i,t}^{\text{beh}}$ is recorded under the quantized rollout policy that generated the completion and $\ell_{i,t}^{\text{new}}(\theta)$ is recomputed by the current W4A4 policy forward. When a rollout group contains multiple quantized views, each ratio is therefore defined relative to the completion's generating quantized rollout policy, which serves as its behavior policy.

After the advantages and importance ratios are defined, the RLVR backbone updates the trainable policy. Our experiments instantiate this backbone with the token-normalized clipped objective of DAPO Yu et al. (2025). With decoupled lower and upper bounds ϵ_{low} and ϵ_{high}, let

$$\bar{r}_{i,t}(\theta) = \text{clip}(r_{i,t}(\theta), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}}). \tag{3}$$

The policy objective is

$$\mathcal{J}(\theta) = \mathbb{E}_{\mathcal{B}}\left[\frac{1}{N_{\text{tok}}} \sum_{(p,\{y_i\})\in\mathcal{B}} \sum_{i=1}^{G} \sum_{t=1}^{|y_i|} \min\left(r_{i,t}(\theta)\hat{A}_i, \bar{r}_{i,t}(\theta)\hat{A}_i\right)\right], \tag{4}$$

where $\mathcal{B}$ is a training batch and $N_{\text{tok}} = \sum_{(p,\{y_i\})\in\mathcal{B}} \sum_i |y_i|$ is the number of completion tokens.

3.2 Quantization

We use NVIDIA NVFP4 NVIDIA et al. (2026) as the 4-bit format for all 4-bit weight and activation quantization in our experiments. Given a target quantization grid $\mathbb{P}$, RTN deterministically maps a scalar z to its closest representable value. SR instead samples one of the two adjacent values

$$\text{SR}(z) = \begin{cases} \lfloor z \rfloor_{\mathbb{P}}, & \text{w.p. } \dfrac{\lceil z \rceil_{\mathbb{P}} - z}{\lceil z \rceil_{\mathbb{P}} - \lfloor z \rfloor_{\mathbb{P}}}, \\ \lceil z \rceil_{\mathbb{P}}, & \text{otherwise}, \end{cases} \tag{5}$$

where $\lfloor \cdot \rfloor_{\mathbb{P}}$ and $\lceil \cdot \rceil_{\mathbb{P}}$ denote the adjacent grid values below and above z, respectively. SR Gupta et al. (2015) is unbiased, with $\mathbb{E}[\text{SR}(z)] = z$, and has been used in low-precision LLM training Chmiel et al. (2025); Xin et al. (2026). QCRL uses SR only to construct the stochastic quantized rollout policy; SR is not applied on the gradient path.

4 Method

We introduce Quantization-Consistent Reinforcement Learning (QCRL), a framework for reinforcement learning with verifiable rewards (RLVR) under 4-bit weight and activation quantization (W4A4). Rounding-dependent perturbations during W4A4 training can alter the distribution of verified outcomes across rollouts, leading to a mismatch between the continuous policy and the quantized rollout policy and thereby destabilizing policy updates. QCRL addresses these effects through two complementary modules: Quantization-Consistent Rollout (QCR) and Quantization-Weight Dampening (QWD). QCR calibrates completion-level rewards using reward statistics from RTN and SR rollouts, and QWD pulls latent weights toward their quantization centers, reducing oscillations between adjacent quantized levels.

4.1 Quantization-Consistent Rollout

QCR replaces the standard single-view rollout with a dual-view group jointly sampled from two quantized behavior policies. For each prompt, the rollout group is evenly split into completions from an RTN-quantized policy and those from an SR-quantized policy. QCR calibrates each task reward with two complementary signals: 1) a one-sided cross-view penalty for outcomes below the opposite view's average reward; and 2) a within-view reward rescaling term that amplifies successful outcomes when the remaining samples from the same view also succeed. The calibrated rewards are then fed into the backbone advantage estimator in place of the original task rewards, while the policy

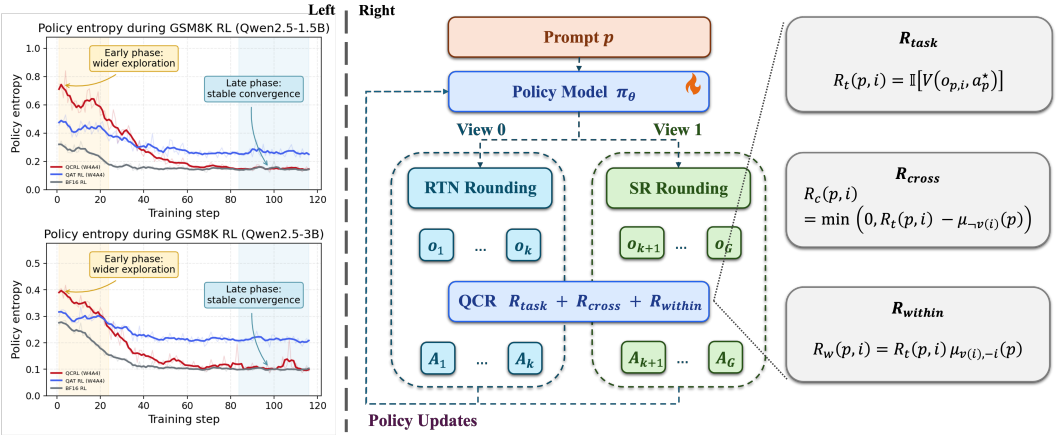

Figure 3: Overview of QCRL. **Left:** policy entropy on GSM8K (Qwen2.5-1.5B/3B); QCRL expands early exploration and later settles toward the BF16 range. **Right:** each prompt's rollout group is split across a nearest-rounding view (v=0, deterministic) and a stochastic-rounding view (v=1, exploratory). QCR combines the task reward with cross-view calibration and within-view support before the RLVR backbone estimates advantages; QWD subsequently stabilizes the updated quantized weights.

objective and clipping procedure remain unchanged. This design preserves full compatibility with the underlying RLVR algorithm.

Specifically, for each prompt we construct a dual-view rollout group of size G by sampling $G/2$ completions from each of two quantized views of the current policy: one using RTN quantization and the other using SR. Both views are frozen during rollout generation and serve as behavior policies. For every completion y_i, we record its task reward $R_{\text{task}}(p,i)$, view label $v(i)$ (0 for RTN and 1 for SR), and the token-level log-probabilities, which provide the view-specific reward statistics and behavior-policy probabilities needed for calibration and importance sampling.

We define three per-prompt statistics, the average reward $\mu_v(p)$ under view $v = 0$ (RTN) or 1 (SR), the average reward of the opposite view $\mu_{\neg v(i)}(p)$, and the leave-one-out average within the completion's view $\mu_{v(i),-i}(p)$ by

$$\begin{aligned}
\mu_v(p) &= \frac{1}{|\mathcal{I}_v(p)|} \sum_{j \in \mathcal{I}_v(p)} R_{\text{task}}(p,j), \\
\mu_{\neg v(i)}(p) &= \frac{1}{|\mathcal{V}| - 1} \sum_{\substack{u \in \mathcal{V} \\ u \neq v(i)}} \mu_u(p), \\
\mu_{v(i),-i}(p) &= \frac{1}{|\mathcal{I}_{v(i)}(p)| - 1} \sum_{\substack{j \in \mathcal{I}_{v(i)}(p) \\ j \neq i}} R_{\text{task}}(p,j),
\end{aligned} \tag{6}$$

Here, $\mathcal{I}_v(p)$ denotes the set of completion indices assigned to view $v \in \{0,1\}$, with 0 for RTN and 1 for SR, $|\mathcal{I}_v(p)| = G/2$, and $\mathcal{V} = \{0,1\}$. These statistics, computed from the collected rollout rewards and held fixed during policy optimization, directly define the two calibration components below.

Cross-view calibration. We penalize completions whose task rewards fall below the average reward of the alternative quantization view

$$R_c(p,i) = \min\big(0,\ R_{\text{task}}(p,i) - \mu_{\neg v(i)}(p)\big), \tag{7}$$

which yields a negative value equal to the reward shortfall when $R_{\text{task}}(p,i) < \mu_{\neg v(i)}(p)$ and is exactly zero otherwise. Intuitively, a below-reference completion incurs an additional penalty proportional to its shortfall, while a completion that matches or exceeds the opposite-view mean receives no cross-view adjustment.

Within-view support. For the within-view component, we assign a larger weight to a successful completion when other completions from the same view also succeed

$$R_w(p,i) = R_{\text{task}}(p,i)\ \mu_{v(i),-i}(p). \tag{8}$$

The leave-one-out average prevents the i-th completion from contributing to its own weight estimate. Consequently, this weight increases with the success rate of the remaining same-view completions.

Finally, we combine the task reward and the two calibration components into a single calibrated reward

$$R_{\text{qc}}(p,i) = \lambda_{\text{task}}\, R_{\text{task}}(p,i) + \lambda_{\text{cross}}\, R_c(p,i) + \lambda_{\text{within}}\, R_w(p,i). \tag{9}$$

We set unit weight to the task-reward term and smaller coefficients to the two calibration terms. The calibrated reward $R_{\text{qc}}(p,i)$ then replaces $R_{\text{task}}(p,i)$ as the input to the backbone advantage estimator.

QCR transforms the reward before advantage estimation and retains the policy-loss formulation, group normalization, clipping rules, and optimizer of the RLVR backbone. We instantiate it with DAPO in our experiments. The calibration conditions training credit on the quantization view that generated each completion. This separation allows the policy gradient to reflect rounding-dependent differences in outcome quality without altering the functional form of the RLVR objective. Appendix B.1 analyzes the calibration under binary rewards.

4.2 Quantization-Weight Dampening

In the RLVR STE training pipeline, the policy's continuous weights are updated by the optimizer, but their quantized discrete counterparts are used during training. A small continuous update can cross a quantization boundary, causing the quantized value to jump and creating a mismatch between the continuous weights and the model used for training. This mismatch can bias the reward signal and destabilize training.

To reduce this effect, we propose QWD, which pulls the continuous weights toward their current quantization centers after an optimizer step. Specifically, for each quantization group, we identify the weight with the largest absolute value, which determines the group's quantization scale, and keep it fixed, while the remaining weights are linearly interpolated toward their respective dequantized values. By doing so, the scale remains stationary during the interpolation, and the other weights are moved relative to a stable quantization grid.

Formally, let w_k denote the k-th continuous weight after an optimizer update, where we omit the step and layer indices for notational simplicity. Let $\bar{q}_k$ be its current quantization center, obtained by applying the same RTN quantizer used for the rollout policy. Define E as the set of weight indices that attain the maximum absolute value in their respective quantization groups. QWD keeps the weights in E unchanged and produces edited weights for the remaining indices according to

$$\widetilde{w}_k = \begin{cases} w_k, & k \in E, \\ (1-\gamma)\, w_k + \gamma\, \bar{q}_k, & k \notin E, \end{cases} \tag{10}$$

where the coefficient $\gamma \in [0,1]$ controls the interpolation strength. When $\gamma = 1$, the weight is fully pulled to its quantization center; when $\gamma = 0$, it remains unchanged. The edited weights replace the original continuous weights for the subsequent forward and backward passes, and the optimizer state is unaffected.

By fixing the largest weight in each group, QWD ensures that the group scale used to determine the quantization centers stays constant during the contraction. In the next forward pass, both the group scales and the quantized values are recomputed from the edited weights. In practice, we apply QWD at regular intervals, e.g., once every $P = 8$ optimizer steps, and schedule γ following a cosine ramp.

QWD does not alter the reward computation or the policy gradient, as it is a direct weight adjustment after the optimizer rather than an extra loss term, and is therefore compatible with any RLVR backbone. Compared to quantization-aware training methods that regularize weights toward quantization centers Nagel et al. (2022), QWD achieves a similar goal through a simple post-optimizer operation while additionally protecting the scales of dynamic groupwise quantization.

Table 1: Main GSM8K results (pass@1), trained and evaluated on GSM8K. Orig is the instruction-tuned model before any RL, and Orig + PTQ applies W4A4 post-training quantization to it directly. BF16 RL is full-precision RL training, and BF16 RL + PTQ quantizes that trained model post hoc. The "vs BF16 RL" column is the gap to full-precision RL, and "vs Orig + PTQ" compares each W4A4 method against the pre-RL PTQ baseline.

Model	Method	Eval. format	pass@1 (%)	vs BF16 RL (pts.)	vs Orig + PTQ (pts.)
Qwen2.5-3B	Orig	BF16	71.34	–	–
	Orig + PTQ	W4A4	62.17	–	–
	BF16 RL	BF16	82.79	–	–
	BF16 RL + PTQ	W8A8	81.05	−1.74	–
	W8A8 RL	W8A8	82.34	−0.45	–
	BF16 RL + PTQ	W4A16	79.08	−3.71	–
	W4A16 RL	W4A16	80.44	−2.35	–
	BF16 RL + PTQ	W4A8	79.30	−3.49	–
	W4A8 RL	W4A8	78.77	−4.02	–
	BF16 RL + PTQ	W4A4	76.04	−6.75	+13.87
	W4A4 RL (SR)	W4A4	71.34	−11.45	+9.17
	W4A4 RL (RTN)	W4A4	74.07	−8.72	+11.90
	QeRL (LoRA)	W4A4	73.84	−8.95	+11.67
	QeRL (Full)	W4A4	75.21	−7.58	+13.04
	QCRL	**W4A4**	**78.32**	**−4.47**	**+16.15**
Qwen2.5-1.5B	Orig	BF16	57.32	–	–
	Orig + PTQ	W4A4	44.05	–	–
	BF16 RL + PTQ	W4A4	56.79	–	+12.74
	W4A4 RL (RTN)	W4A4	53.60	–	+9.55
	QCRL	**W4A4**	**58.53**	–	**+14.48**
Qwen3-4B	Orig	BF16	90.30	–	–
	Orig + PTQ	W4A4	87.72	–	–
	BF16 RL + PTQ	W4A4	92.49	–	+4.77
	W4A4 RL (RTN)	W4A4	91.89	–	+4.17
	QCRL	**W4A4**	**93.33**	–	**+5.61**
Qwen3-1.7B	Orig	BF16	87.26	–	–
	Orig + PTQ	W4A4	83.47	–	–
	BF16 RL + PTQ	W4A4	85.37	–	+1.90
	W4A4 RL (RTN)	W4A4	84.99	–	+1.52
	QCRL	**W4A4**	**85.67**	–	**+2.20**

5 Experiments

5.1 Experimental Setup

Models and evaluation. We evaluate Qwen2.5-Instruct Team (2024) at 1.5B, 3B, and 7B parameters and Qwen3 Team (2025) at 1.7B and 4B parameters. For GSM8K Cobbe et al. (2021), we train on the training split and report pass@1 on all 1,319 test problems. We also train Qwen2.5-3B on DAPOMath Yu et al. (2025) and Qwen2.5-7B on BigMath Albalak et al. (2025), then evaluate both models on MATH500 Lightman et al. (2023), AMC23 Li et al. (2024), AIME24 Li et al. (2024), and AIME25 Dekoninck et al. (2026). DAPOMath results use pass@4, whereas BigMath results include pass@1 and pass@4 averaged over four evaluation seeds used for decoding.

Training protocol. In a WmAn setting, policy optimization and evaluation fake-quantize linear weights and inputs to m and n bits, while rollout generation quantizes weights to m bits and retains BF16 activations. We use symmetric quantization with groups of 16 values, and 4-bit quantization uses NVFP4 E2M1. All linear layers except the language-modeling head are quantized, while the head and master parameters remain in BF16. BF16 RL, direct quantized RL, and QCRL use the

Table 2: Evaluation of DAPOMath-trained Qwen2.5-3B models on multiple mathematical test sets (pass@1/pass@4).

Model	Method	Eval prec.	MATH500 (%)	AMC23 (%)	AIME24 (%)	AIME25 (%)	Avg. (%)
Qwen2.5-3B	Orig	BF16	54.34/62.40	32.23/54.22	3.33/5.56	0.83/3.33	22.68/31.38
	BF16 RL	BF16	55.62/64.00	34.64/55.42	5.28/10.00	3.33/10.00	24.72/34.86
	Orig + PTQ	W4A4	41.09/54.40	18.07/38.55	1.67/4.44	0.83/3.33	15.42/25.18
	BF16 RL + PTQ	W4A4	44.43/55.60	21.08/40.96	**2.50/6.67**	0.00/0.00	17.00/25.81
	W4A4 RL (RTN)	W4A4	42.76/56.00	18.07/37.35	1.67/3.33	0.83/3.33	15.83/25.00
	QeRL (LoRA)	W4A4	41.93/55.60	16.87/37.35	1.39/4.44	1.67/3.33	15.46/25.18
	QCRL	W4A4	**46.77/59.20**	**22.59/40.96**	**2.50**/4.44	**1.67/6.67**	**18.38/27.82**

DAPO objective Yu et al. (2025), AdamW, a learning rate of 1×10^{-6}, cosine scheduling, and a warmup ratio of 0.1. Experiments run on eight H20 GPUs, and model-specific rollout groups, batches, sequence limits, training durations, and seeds are listed in the supplementary material. We fixed the random seed, and training-seed robustness is reported in the supplementary material.

QCRL settings. QCR uses λ_{task}=1.0, λ_{c}=0.10, and λ_{w}=0.05. QWD uses $\gamma_{\max}$=0.01, P=8, and protection of the scale-defining coordinate in each quantization group.

5.2 Baselines

QeRL baselines. The QeRL-style LoRA baseline reproduces QeRL's low-rank trainable-state design Huang et al. (2026). Its base linear layers use the target W4A4 quantizer, while the LoRA branch remains in full precision in training. The QeRL-style Full baseline adds Adaptive Quantization Noise to our full-parameter W4A4 RL baseline.

5.3 Main Results

Table 1 reports GSM8K test pass@1 for four models trained on the GSM8K training split. At W8A8 and W4A16, quantized RL remains beneficial relative to quantizing the BF16 RL checkpoint afterward, exceeding its matched PTQ result. The sign reverses at W4A8 and W4A4, where quantized RL falls 0.53 and 1.97 points below the corresponding BF16 RL + PTQ result. Thus, the central W4A4 failure is not simply that fewer bits lower final accuracy. Below a precision-dependent threshold, optimizing through the quantized policy no longer preserves the advantage of training in the deployment format. QCRL changes this ordering at W4A4, reaching 78.32% and recovering a 2.28-point advantage over BF16 RL + PTQ. This recovery is consistent with the hypothesis that the target W4A4 policy remains trainable when rollout evidence is calibrated.

The different model results show the same ordering but not a constant gain. Relative to quantized training, QCRL gains 4.93 points on Qwen2.5-1.5B and 4.25 points on Qwen2.5-3B, compared with 1.44 points on Qwen3-4B and 0.68 points on Qwen3-1.7B. The larger recoveries occur on the Qwen2.5 checkpoints, which also suffer larger quantization losses when Orig is converted to W4A4. A useful interpretation is that QCRL has more room to help when the checkpoint is more sensitive to the target quantizer, rather than adding a fixed accuracy offset.

The training curves in Figure 2(c,d) show that the endpoint ranking is preceded by a sustained optimization difference. On Qwen2.5-3B, the two W4A4 methods are closer early, but QCRL repeatedly occupies the upper part of the low-precision trajectories and finishes nearer the BF16 reward range. The advantage is therefore not explained by a single late reward spike or an isolated checkpoint selection.

Figure 3 (left) provides a more specific view of that difference. For both the 1.5B and 3B models, QCRL begins with substantially higher policy entropy than BF16 RL and W4A4 RL. Its entropy contracts sharply during roughly the first third of training and approaches the BF16 trajectory in the late stage, whereas the direct W4A4 baseline remains on a higher plateau. Early RTN/SR variation broadens the response distribution when the policy is still searching, while the later contrac-

Table 3: Evaluation of BigMath-trained Qwen2.5-7B models. Each cell reports pass@1/pass@4 (%), averaged over multiple evaluation seeds.

Model	Method	Eval prec.	MATH500 (%)	AMC23 (%)	AIME24 (%)	AIME25 (%)	Avg. (%)
Qwen2.5-7B	Orig	BF16	65.81/68.85	41.42/63.55	8.26/14.72	5.42/17.50	30.23/41.16
	BF16 RL	BF16	67.22/69.60	44.13/61.14	8.89/15.83	7.50/20.00	31.94/41.64
	Orig + PTQ	W4A4	61.40/67.00	35.92/56.33	6.88/13.06	4.17/11.67	27.09/37.02
	BF16 RL + PTQ	W4A4	63.77/**68.40**	37.73/55.42	6.94/13.06	4.79/11.67	28.31/37.14
	W4A4 RL (RTN)	W4A4	63.21/67.60	40.06/59.34	6.60/12.50	4.79/11.67	28.67/37.78
	QCRL	**W4A4**	**64.64**/68.35	**41.19/61.14**	**8.61/13.89**	**5.83/16.67**	**30.07/40.01**

tion is consistent with QCR. Coupled with the higher reward trajectory, this pattern resembles an exploration-to-selection transition rather than persistent noise injection.

5.4 Different Training Corpora and Model Sizes

QCRL's W4A4 gains persist beyond GSM8K and across model scales. On DAPOMath-trained Qwen2.5-3B, QCRL achieves a 27.82% pass@4 macro-average, compared with 25.00% for direct W4A4 RL (RTN) and 25.81% for BF16 RL + W4A4 PTQ (Table 2). On BigMath-trained Qwen2.5-7B, it reaches 40.01%, versus 37.78% and 37.14% for the same baselines, respectively (Table 3). QCRL outperforms direct W4A4 RL on all four test sets in both settings, showing that its gains are not tied to a single benchmark.

5.5 Component Ablation

Table 4 separates the core QCR mechanism from the complementary QWD correction. Without either, W4A4 RL (RTN) falls below the BF16 RL + PTQ reference, confirming that standard quantization-aware RL is insufficient at W4A4. QCR alone supplies most of the gain, showing that dual-view reward shaping is the primary component. QWD alone improves over the baseline by a smaller margin, and adding it to QCR gives the best result.

Table 4: Component ablation of QCRL on GSM8K (Qwen2.5-3B, W4A4). RTN/SR-Mix uses the dual-view rollout only.

Config	Views	Cross	Within	QWD	pass@1 (%)
A0 (W4A4 RL)	RTN	×	×	×	74.07
A1 (RTN/SR-Mix)	RTN/SR	×	×	×	76.35
A2 (Cross-view only)	RTN/SR	✓	×	×	76.42
A3 (Within-view only)	RTN/SR	×	✓	×	76.95
A4 (QCR only)	RTN/SR	✓	✓	×	77.41
A5 (QWD only)	RTN	×	×	✓	74.64
A6 (QCRL)	RTN/SR	✓	✓	✓	**78.32**

6 Conclusion

We presented QCRL for improving the reasoning capability of models optimized and evaluated under W4A4 quantization. QCRL collects W4A16 rollouts from RTN and SR views of shared latent weights, adjusts training credit using view-conditioned reward statistics, and stabilizes quantized weights through QWD. Across the evaluated settings, the final W4A4 policy improves over direct W4A4 RL and achieves higher aggregate performance than BF16 RL followed by W4A4 PTQ. The component results support the contribution of dual-view sampling and the complete calibration design, while the QWD diagnostics show reduced quantization-code switching. The framework retains W4A16 rollout and provides an effective training route to higher-quality W4A4 reasoning models.

AI USE STATEMENT

Generative AI tools were used to assist with writing, formatting checks, code modifications, feedback on experimental design, and formulation of the reward-calibration analysis. The authors reviewed all AI-assisted content and take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The model, dataset, optimization, quantization, and evaluation settings needed to reproduce our experiments are described in Section 5 and Appendix A. Appendix A reports model-specific rollout-group sizes, batch sizes, sequence-length limits, training durations, random seeds, and the stage-specific precision used for rollout generation, policy optimization, and evaluation. All reported comparisons use matched training and evaluation protocols.

REFERENCES

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, et al. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models. *arXiv preprint arXiv:2502.17387*, 2025.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: outlier-free 4-bit inference in rotated llms. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pp. 100213–100240, 2024.

Brian Chmiel, Maxim Fishman, Ron Banner, and Daniel Soudry. Fp4 all the way: Fully quantized training of large language models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 91130–91151. Curran Associates, Inc., 2025. doi: 10.52202/085713-3049. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/8340b085045cf13f1f0b6c2c4cc0a89c-Paper-Conference.pdf.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms, 2026. URL https://arxiv.org/abs/2605.00674.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Elias Frantar, Roberto L. Castro, Jiale Chen, Torsten Hoefler, and Dan Alistarh. Marlin: Mixed-precision auto-regressive parallel inference on large language models. In *Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, PPoPP '25, pp. 239–251, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400714436. doi: 10.1145/3710848.3710871. URL https://doi.org/10.1145/3710848.3710871.

Hao Gu, Hao Wang, Jiacheng Liu, Lujun Li, Qiyuan Zhu, Bei Liu, Binxing Xu, Lei Wang, Xintong Yang, Sida Lin, et al. Qarl: Rollout-aligned quantization-aware rl for fast and stable training under training–inference mismatch. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17051–17064, 2026.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pp. 1737–1746. PMLR, 2015.

Wei Huang, Yi Ge, Shuai Yang, Yicheng Xiao, Huizi Mao, Yujun Lin, Hanrong Ye, Sifei Liu, Ka Chun Cheung, Hongxu Yin, Yao Lu, Xiaojuan Qi, Song Han, and Yukang Chen. QeRL: Beyond efficiency - quantization-enhanced reinforcement learning for LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zw8zxMJJlm.

Srivatsan Krishnan, Max Lam, Sharad Chitlangia, Zishen Wan, Gabriel Barth-maron, Aleksandra Faust, and Vijay Janapa Reddi. QuaRL: Quantization for fast and environmentally sustainable reinforcement learning. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=xwWsiFmUEs.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

Xiaocan Li, Shiliang Wu, and Zheng Shen. Decomposing mxfp4 quantization error for llm reinforcement learning: reducible bias, recoverable deadzone, and an irreducible floor, 2026a. URL https://arxiv.org/abs/2605.20402.

Yuhang Li, Reena Elangovan, Xin Dong, Priyadarshini Panda, and Brucek Khailany. QuRL: Low-precision reinforcement learning for efficient reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=eG0bpCwdKn.

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

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

Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. Overcoming oscillations in quantization-aware training. In *International Conference on Machine Learning*, pp. 16318–16330. PMLR, 2022.

NVIDIA, Felix Abecassis, Anjulie Agrusa, Dong Ahn, Jonah Alben, Stefania Alborghetti, Michael Andersch, Sivakumar Arayandi, Alexis Bjorlin, Aaron Blakeman, Evan Briones, Ian Buck, Bryan Catanzaro, Muya Chang, Jinhang Choi, Mike Chrzanowski, Eric Chung, Victor Cui, Steve Dai, Bita Darvish Rouhani, Carlo del Mundo, Deena Donia, Burc Eryilmaz, Henry Estela, Abhinav Goel, Oleg Goncharov, Yugi Guvvala, Robert Hesse, Russell Hewett, Herbert Hum, Ujval Kapasi, Brucek Khailany, Mikail Khona, Nick Knight, Alex Kondratenko, Ronny Krashinsky, Ben Lanir, Simon Layton, Michael Lightstone, Daniel Lo, Paulius Micikevicius, Asit Mishra, Tim Moon, Deepak Narayanan, Chao Ni, Abhijit Paithankar, Satish Pasumarthi, Ankit Patel, Mostofa Patwary, Ashwin Poojary, Gargi Prasad, Sweta Priyadarshi, Yigong Qin, Xiaowei Ren, Oleg Rybakov, Charbel Sakr, Sanjeev Satheesh, Stas Sergienko, Pasha Shamis, Kirthi Shankar, Nishant Sharma, Mohammad Shoeybi, Michael Siu, Misha Smelyanskiy, Darko Stosic, Dusan Stosic, Bor-Yiing Su, Frank Sun, Nima Tajbakhsh, Shelby Thomas, Przemek Tredak, Evgeny Tsykunov, Gandhi Vaithilingam, Aditya Vavre, Rangharajan Venkatesan, Roger Waleffe, Qiyu Wan, Hexin Wang, Mengdi Wang, Lizzie Wei, Hao Wu, Evan Wu, Keith Wyss, Ning Xu, Jinze Xue, Charlene Yang, Yujia Zhai, Ruoxi Zhang, Jingyang Zhu, and Zhongbo Zhu. Pretraining large language models with nvfp4, 2026. URL https://arxiv.org/abs/2509.25149.

Zhaopeng Qiu, Shuang Yu, Jingqi Zhang, Shuai Zhang, Xue Huang, Jingyi Yang, and Junjie Lai. Fp8-rl: A practical and stable low-precision stack for llm reinforcement learning, 2026. URL https://arxiv.org/abs/2601.18150.

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

Siqing Song, Chuang Wang, Yong Lang, Yi Yang, and Xu-Yao Zhang. Lbllm: Lightweight binarization of large language models via three-stage distillation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 35470–35484, 2026.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Haocheng Xi, Charlie Ruan, Peiyuan Liao, Yujun Lin, Han Cai, Yilong Zhao, Shuo Yang, Kurt Keutzer, Song Han, and Ligeng Zhu. Jet-rl: Enabling on-policy fp8 reinforcement learning with unified training and rollout precision flow, 2026. URL https://arxiv.org/abs/2601.14243.

Meng Xin, Sweta Priyadarshi, Jingyu Xin, Bilal Kartal, Aditya Vavre, Asma Kuriparambil Thekkumpate, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Ido Shahaf, Akhiad Bercovich, et al. Quantization-aware distillation for nvfp4 inference accuracy recovery. *arXiv preprint arXiv:2601.20088*, 2026.

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, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

Zhengyang Zhuge, Hao Yu, Xin Wang, Zheng Li, Yizhong Cao, Dayiheng Liu, and Jianwei Zhang. Quads: Stabilizing nvfp4 reinforcement learning for moe via quantization-error alignment across dual sides, 2026. URL https://arxiv.org/abs/2607.15810.

A Training Configurations and Deployment Accounting

A.1 Baselines

BF16 RL. BF16 RL serves as a higher-precision reference. Orig + PTQ quantizes the instruction-tuned checkpoint before RL and measures the starting W4A4 capability. BF16 RL + PTQ quantizes the BF16-trained checkpoint and provides the post-RL PTQ reference for direct W4A4 training.

Quantized RL. W4A4 RL (RTN) uses one fixed RTN W4A16 rollout policy with the shared W4A4 update and evaluation quantizer, without QCR or QWD. W4A4 RL (SR) uses one fixed SR snapshot and the bare verifier reward. RTN/SR-Mix matches QCRL's view split and behavior-likelihood bookkeeping but retains the bare verifier reward. The single-view controls compare rounding rules, and RTN/SR-Mix isolates dual-view sampling from QCR reward calibration.

A.2 Evaluation Protocol and Model-Specific Training Configurations

All experiments share the quantization, optimizer, QCR, and QWD settings described in the main paper unless otherwise noted. Unless a seed robustness run is stated explicitly, the random seed is fixed to 2027. Qwen2.5-3B GSM8K uses a rollout group of $G{=}4$. The per-device batch size is 4 with gradient accumulation of 8, giving a global prompt batch size of 256 across 8 H20 GPUs.

For Qwen2.5 models, the data-specific sequence budgets are listed in Table 5. Qwen3-series training and evaluation use a maximum generation length of 16,384.

Table 5: Sequence budgets for Qwen2.5 training.

Setting	Max. prompt	Max. completion	Max. sequence
GSM8K	256	2,048	2,500
DAPOMath	1,024	8,192	10,000
BigMath	1,024	4,096	5,500

BigMath uses $G{=}16$, split evenly between the RTN and SR views. Qwen2.5 7B BigMath is trained for 200 steps because of its data scale.

A.3 Stage-Specific Quantization Protocol

QCRL and its direct W4A4 RL baselines use stage-specific fake quantization. During rollout generation, RTN and SR snapshots use W4A16 weight-only fake quantization, while activations remain in BF16. During policy optimization, the weights and activations of all quantized linear layers are fake-quantized at W4A4; the language-modeling head remains in BF16. We use W4A16 rollout snapshots because an aligned W4A4 rollout and W4A4 update configuration exhibited training collapse. This protocol retains W4A4 optimization and W4A4 evaluation of the final model while using a higher-precision rollout policy.

B Additional Experiments and Diagnostics

B.1 Mechanism of View-Aware Reward Calibration

QCR assigns training credit using the outcomes of two quantized behavior views. The verifier reward remains a correctness label for the generated completion; calibration changes the reward used for advantage estimation. For binary rewards, write $R_i = R_{\text{task}}(p, i) \in \{0, 1\}$ and retain the view statistics from Equation 6. Equations 7 and 8 become

$$\begin{aligned} R_c(p, i) &= -(1 - R_i)\mu_{\neg v(i)}(p), \\ R_w(p, i) &= R_i \mu_{v(i),-i}(p). \end{aligned} \tag{11}$$

The cross-view term penalizes an incorrect completion in proportion to the success rate of the opposite view and leaves correct completions unchanged. The within-view term adds support to a correct completion in proportion to the success rate of its same-view peers and leaves incorrect completions unchanged. Together, these terms provide a view-conditioned adjustment to the learning signal.

Condition on a prompt and fixed quantized behavior snapshots. Let ρ_v denote the probability of a correct completion under view v, and assume completions are sampled independently from their respective views. With $\lambda_{\text{task}} = 1$, the expected calibrated reward for a completion from view v is

$$\mathbb{E}[R_{\text{qc}} \mid v] = \rho_v - \lambda_{\text{cross}}(1 - \rho_v)\rho_{\neg v} + \lambda_{\text{within}}\rho_v^2. \tag{12}$$

The cross-view contribution depends on the success probability of the alternative view, while the within-view contribution depends on repeated success within the generating view. These expectations characterize the reward transformation, not the expected normalized advantage or policy gradient.

Finite-sample variability remains in both statistics. For $n = G/2$ completions per view, the same assumptions give

$$\text{Var}(\mu_v) = \frac{\rho_v(1-\rho_v)}{n}, \qquad \text{Var}(\mu_{v,-i}) = \frac{\rho_v(1-\rho_v)}{n-1}. \tag{13}$$

At $G = 4$, the leave-one-out statistic contains only one other completion and can therefore be noisy. However, the experimental results demonstrate that QCRL still yields positive gains in this setting. We provide experimental results for both $G = 4$ and $G = 16$, and recommend using $G = 16$ for better training stability. The analysis is conditional on the frozen views; variability across newly sampled SR snapshots is an additional source of variation.

B.2 Robustness to Random Seeds

We retrain Qwen2.5-3B with QCRL and W4A4 RL(RTN) on GSM8K under random seeds 42, 2026, 2027, 2028 while holding the remaining training settings fixed. Table 6 reports the GSM8K test pass@1 of each run, together with the mean and standard deviation.

Table 6: Seed robustness of Qwen2.5-3B on GSM8K.

Seed	W4A4 RL pass@1 (%)	QCRL pass@1 (%)
2026	73.84	77.86
2027	74.07	78.32
2028	73.54	77.71
42	74.30	77.41
Mean $\pm$ std.	73.94 $\pm$ 0.32	77.83 $\pm$ 0.38

B.3 Sensitivity to QCR Coefficients

We evaluate the QCR reward coefficients on Qwen2.5-3B GSM8K. The default setting is λ_{cross}=0.10 and λ_{within}=0.05. Each non-default setting varies one coefficient while holding the other at its default value. Table 7 reports the resulting test pass@1.

Table 7: Sensitivity of Qwen2.5-3B W4A4 QCRL to the QCR reward coefficients on GSM8K.

λ_{cross}	λ_{within}	GSM8K pass@1 (%)
0.00	0.00	76.35
0.05	0.05	76.72
0.05	0.10	77.33
0.10	0.05	**78.32**
0.10	0.10	77.26

B.4 Sensitivity to Rollout Group Size

We evaluate QCRL with rollout group sizes $G \in \{4, 8, 16\}$ on Qwen2.5-3B DAPOMath. Each group is split evenly between the RTN and SR views, while the remaining training and evaluation settings are held fixed. Table 8 reports the resulting test pass@1 and pass@4.

Table 8: Sensitivity of Qwen2.5-3B W4A4 QCRL to the rollout group size on DAPOMath.

G	AIME24 (%)	AIME25 (%)	AMC23 (%)	MATH500 (%)	Avg (%)
4	2.50/4.44	1.67/6.67	22.59/40.96	46.77/59.20	18.38/27.82
8	2.22/6.67	1.67/6.67	22.89/39.76	47.88/57.40	18.67/27.63
16	2.22/5.56	1.67/3.33	24.10/45.78	48.44/58.20	19.11/28.22

B.5 Training-Time Effect of Quantization-Weight Dampening

Figure 4 compares matched training trajectories that differ only in whether QWD dampening is applied. Without dampening, the normalized rounding offset remains nearly constant at approximately 3.43×10^{-2}, whereas QWD progressively reduces it to about 2.92×10^{-2} by step 300, corresponding to an approximately 15% decrease. After the shared initial transient, QWD also lowers the residual sign-flip rate from roughly 6–7 $\times 10^{-5}$ to 3–4 $\times 10^{-5}$ and, more prominently, reduces the quantization-code flip rate from the order of 10^{-5} to the order of 10^{-6}. These trends are consistent with the intended action of QWD: contracting latent weights toward their current quantized values reduces both small-scale oscillation around quantization centers and discrete transitions between neighboring deployed quantization centers. The intermittent late-stage spikes show that sufficiently large optimizer updates can still induce boundary crossings, indicating that QWD dampens rather than completely eliminates quantization-induced switching.

B.6 Rollout Efficiency

We compare BF16 and W4A16 decode efficiency on NVIDIA H20 GPUs at batch sizes 1, 4, and 8. Each max completion length is 2,048. Because our H20 software stack does not provide a native NVFP4 Tensor Core path, NVFP4A16 is executed with the supported Marlin kernel Frantar et al. (2025). As shown in Table 9, W4A16 reduces memory usage from 6.17 to 2.66 GiB, a saving of 3.51 GiB (56.9%). Generation throughput improves by 14.0%, 12.1%, and 10.2% at batch sizes 1, 4, and 8, respectively.

Table 9: Model memory and throughput on NVIDIA H20 GPUs. Changes are relative to BF16.

Metric	BF16	W4A16	Change
Weight memory (GiB)	6.17	2.66	-56.9%
Generation throughput (tok/s)			
Batch Size = 1	383.8	437.6	$+14.0\%$
Batch Size = 4	1454.7	1630.7	$+12.1\%$
Batch Size = 8	3033.4	3342.8	$+10.2\%$

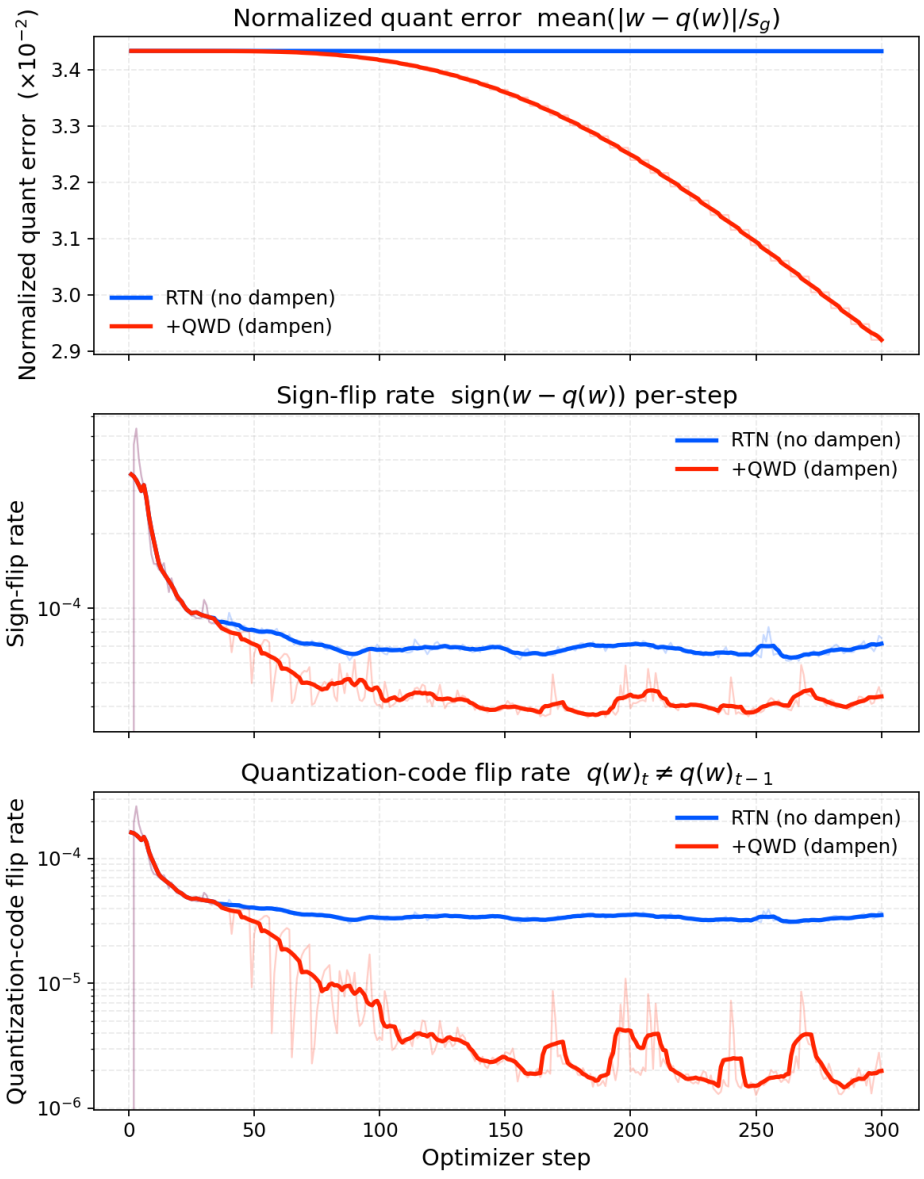

Figure 4: Training-time quantization diagnostics for matched W4A4 runs with and without Quantization-Weight Dampening (QWD). From top to bottom, the panels report the mean group-scale-normalized RTN rounding offset, the per-step sign-flip rate of the rounding residual, and the cross-step RTN quantization-code flip rate.