INT8-RL: TRAINING–INFERENCE CO-OPTIMIZATION FOR QUANTIZED ROLLOUTS ON ASCEND

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive rollout generation is a major cost of reinforcement learning (RL) post-training for large language models. INT8 inference can reduce this cost, but activation outliers and training–rollout numerical mismatch can degrade policy optimization. We present INT8-RL, a framework for INT8 rollouts on Ascend that combines quantization-aware training, jointly learned smoothing and weight scales, reproducible hash-based rounding, and selective precision fallback. Experiments on openPangu-7B and Qwen3-30B MoE show substantial reward recovery over the corresponding INT8 baselines. On openPangu-7B with DAPO-Math-17K, the combined W8A8 configuration improves reward from 0.252 to 0.305, compared with 0.309 for BF16. Profiling shows a 19.7% reduction in per-token decode latency. A separate long-sequence ablation that disables training-side fake quantization reduces step time by 18.7%; this measures implementation headroom rather than the end-to-end gain of the complete QAT pipeline. The results identify quantization adaptation and training overhead as coupled requirements for practical INT8 RL.

1 INTRODUCTION

Reinforcement learning (RL) improves large language model reasoning, but autoregressive rollout generation makes post-training expensive (Chu et al., 2025; Yu et al., 2025). Quantized rollouts reduce inference cost using formats such as INT8, FP8, and NVFP4 (Xi et al., 2026; Huang et al., 2025; Li et al., 2026). Their practical benefit depends on preserving useful training trajectories while limiting the overhead of adapting the policy to low precision. Figure 1 illustrates the stages of an RL post-training iteration.

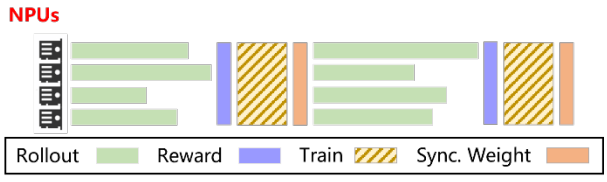

Figure 1: Stages of an RL post-training iteration. Autoregressive rollout generation contributes a substantial part of the iteration cost, motivating low-precision inference.

INT8 RL faces two coupled challenges. Activation outliers reduce the resolution of W8A8 quantization, while quantization and backend arithmetic can make training and rollout evaluate different token distributions. Policy updates continually change the weights and activation distributions, limiting the suitability of fixed quantization parameters. Quantization-aware training (QAT) can adapt to these changes, but its fake-quantization operations may offset rollout savings.

We present INT8-RL, which couples QAT with INT8 rollout execution on Ascend. Jointly learned smoothing factors and weight scales adapt the quantization grid as the policy evolves. Content-derived hashing makes rounding reproducible for identical quantizer inputs, and selective precision fallback protects sensitive layers. Updated quantization parameters are exported with each policy refresh.

Our contributions are: (i) joint adaptation of smoothing and weight scales during RL; (ii) reproducible rounding combined with model-specific precision fallback; and (iii) an evaluation on dense and MoE

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.

models that separates reward recovery, rollout latency, and training overhead. On openPangu-7B with DAPO-Math-17K, the combined W8A8 configuration improves reward from 0.252 to 0.305, versus 0.309 for BF16. Decode latency falls by 19.7%, while the complete QAT pipeline retains additional training costs.

2 Related Work

RL post-training. RL post-training spans preference alignment (Bai et al., 2022), generalization (Chu et al., 2025), group-based optimization (Zhang et al., 2025a), search-assisted reasoning (Jin et al., 2025), and cross-domain reasoning (Cheng et al., 2025). We study the numerical execution of policy training and rollout.

Quantized rollouts and numerical alignment. QeRL combines NVFP4 with low-rank adaptation and explores quantization noise (Huang et al., 2025); QuRL addresses policy-ratio clipping and small weight updates under INT8/FP8 rollout (Li et al., 2026); Jet-RL unifies FP8 training and rollout precision (Xi et al., 2026). FP16 execution (Qi et al., 2025) and deterministic tensor-parallel kernels (Zhang et al., 2025b) further address numerical mismatch. Our focus is INT8 QAT with joint scale adaptation, reproducible rounding, and selective fallback.

Low-precision training and model compression. FP8-LM, large-scale FP8 training, and COAT examine training precision and memory costs (Peng et al., 2023; Fishman et al., 2024; Xi et al., 2024). Complementary approaches target KV caches (Taneja & Shingvi, 2026), layer-wise precision (Kim et al., 2025), OpenPangu deployment (Luo et al., 2025), weight PTQ (Yan et al., 2026; Huang et al., 2024), and MoE pruning (Lasby et al., 2025). Their deployment benefits do not directly establish behavior during RL updates.

Smoothing and quantization-aware adaptation. PTQ and QAT provide different adaptation routes (Hasan, 2024). SmoothQuant redistributes activation range into weights through an equivalent channel-wise transformation (Xiao et al., 2023). We jointly adapt smoothing and quantization scales during RL, when weights and rollout-induced activation distributions evolve.

3 Method

3.1 Policies and Quantized Execution

Let c denote a prompt and $y = (y_1, \ldots, y_T)$ a generated response, with prefix $h_t = (c, y_{<t})$. We write θ for the trainable model parameters and ϕ for the quantization parameters, including smoothing factors and weight scales. At rollout iteration k, the behavior policy $\mu_k(\cdot \mid h_t)$ is the distribution used by the inference engine to sample the next token. We denote the training-side policy by $\pi^{\mathrm{tr}}_{\theta,\phi}(\cdot \mid h_t)$.

Floating-point parameter storage does not imply a BF16 forward policy: QAT inserts fake quantizers into training computation. Even at the same checkpoint and quantization settings, backend differences can leave

$$\pi^{\mathrm{tr}}_{\theta_k,\phi_k}(\cdot \mid h_t) \neq \mu_k(\cdot \mid h_t). \tag{1}$$

Our components address this mismatch and quantization-induced task degradation; importance weighting is a separate statistical correction.

3.2 Jointly Learned Smoothing and Weight Scales

Fake quantization. For a scalar v, a positive quantization step Δ, an integer zero point z, and integer bounds $q_{\min}$ and $q_{\max}$, define

$$u = v/\Delta + z, \tag{2}$$

$$q(v) = \mathrm{clip}(\mathcal{R}(u), q_{\min}, q_{\max}), \tag{3}$$

$$\widehat{v} = \Delta\big(q(v) - z\big), \tag{4}$$

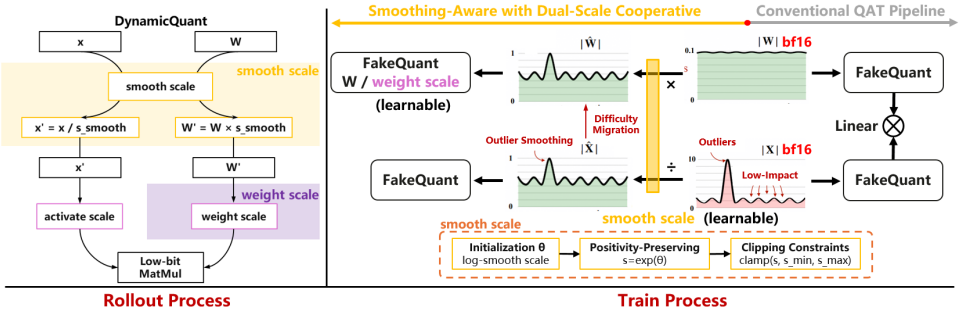

Figure 2: Joint adaptation of smoothing factors and weight quantization scales. The training path simulates quantization of the reparameterized operands, while the rollout path executes the corresponding low-precision computation. Smoothing preserves the unquantized linear mapping but changes the quantization error of its operands.

where $\mathcal{R}$ is the rounding operator. The fake-quantized value lies on the deployment grid but is represented in floating point during training. QAT-W8 applies this transformation to weights; QAT-W8A8 also simulates activation quantization. W8A16 and W8A8 denote the corresponding rollout operand precisions, subject to the specified fallback mask.

Rounding is non-differentiable, so we use a straight-through estimator (STE). Under the usual identity surrogate for rounding and an identity derivative inside the clipping interval,

$$\frac{\partial \widehat{v}}{\partial v} \approx \mathbb{I}\{q_{\min} \le u \le q_{\max}\}. \tag{5}$$

Treating the zero point as fixed, the corresponding step-size surrogate is

$$\frac{\partial \widehat{v}}{\partial \Delta} \approx \begin{cases} q_{\min} - z, & u < q_{\min}, \\ \mathcal{R}(u) - z - v/\Delta, & q_{\min} \le u \le q_{\max}, \\ q_{\max} - z, & u > q_{\max}. \end{cases} \tag{6}$$

Gradients sum over elements sharing a scale. This STE updates quantization steps with the RL objective; it approximates the discrete quantizer derivative.

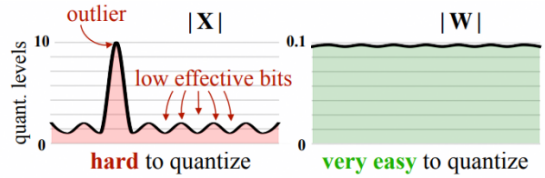

Figure 3: Effect of activation outliers on uniform quantization. Large values widen the quantization range and reduce resolution for smaller values. Channel-wise smoothing redistributes this range across activations and weights.

Equivalent channel-wise reparameterization. Consider a linear layer with input $X \in \mathbb{R}^{N \times d_{\text{in}}}$ and weights $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. Let $S = \text{diag}(s)$ for a positive vector $s \in \mathbb{R}^{d_{\text{in}}}$. We reparameterize the operands as

$$\widetilde{X} = XS^{-1}, \qquad \widetilde{W} = WS, \tag{7}$$

which preserves the unquantized linear transformation:

$$\widetilde{X}\widetilde{W}^{\top} = XS^{-1}SW^{\top} = XW^{\top}. \tag{8}$$

Increasing a smoothing factor reduces the corresponding activation magnitude and increases its weight magnitude, trading their quantization ranges (Figure 3).

To learn positive factors, we introduce unconstrained log-domain parameters β_j and write

$$s_j = \exp(\alpha\beta_j), \qquad \alpha > 0, \tag{9}$$

where α controls the parameterization scale. Jointly learning s and the weight steps adapts the quantization grid to the smoothed weights (Figure 2).

The W8A8 training forward for the layer can then be written as

$$\widehat{Y} = \mathrm{FQ}_a(XS^{-1})\ \mathrm{FQ}_w(WS)^\top, \tag{10}$$

with any bias added in the implemented output precision. The W8A16 counterpart retains the activation operand in 16-bit precision. As described in our implementation, an exponential moving average of smoothed weight magnitudes provides a reference for scale updates, and scale-gradient normalization controls variation in update magnitude across layers.

3.3 Reproducible Hash-Based Rounding

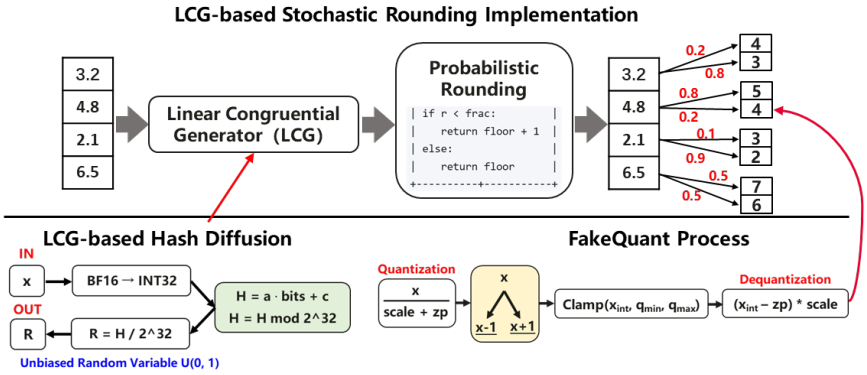

Figure 4: Ideal stochastic rounding and its reproducible hash-based implementation. Randomized rounding selects neighboring grid points using the fractional part of the normalized input. The implementation derives its threshold from a deterministic input hash; reproducibility is distinct from the unbiasedness guarantee of independent uniform randomness.

Nearest rounding introduces a deterministic error for each off-grid input. Its aggregate error need not vanish for the values encountered during training. Conventional stochastic rounding instead chooses between neighboring integers. For a normalized value u, let $n = \lfloor u \rfloor$ and $\delta = u - n$. With $U \sim \mathrm{Uniform}[0, 1)$ independent of u,

$$\mathcal{R}_{\mathrm{SR}}(u; U) = n + \mathbb{I}\{U < \delta\}. \tag{11}$$

Before clipping, this ideal randomized operator satisfies

$$\mathbb{E}_U[\mathcal{R}_{\mathrm{SR}}(u; U) \mid u] = n + \delta = u. \tag{12}$$

This expectation applies before clipping and with representable neighboring grid points.

Independent random-number streams can cause training and inference to round the same value differently. We instead derive a reproducible threshold from the input's bit representation (Figure 4). Denote this representation by $\mathrm{bits}(v)$ and define the 32-bit affine hash

$$H(v) = \big(a\,\mathrm{bits}(v) + b\big) \bmod 2^{32}, \qquad r(v) = H(v)/2^{32}. \tag{13}$$

Replacing U in Eq. equation 11 with $r(v)$ makes the scalar result reproducible when bit patterns, quantizer settings, and integer arithmetic match. It does not ensure equality of complete backend computations.

Scope of the unbiasedness argument. Equation equation 12 requires independent uniform randomness. A fixed content-derived hash is deterministic conditional on its input, so pointwise unbiasedness does not follow. We retain SR as the experimental shorthand for reproducible hash-based rounding and evaluate it empirically.

3.4 Selective Precision Fallback and Weight Refresh

Layer selection. We select a fallback set $\mathcal{F}$ through layer-wise error analysis and retain its layers in floating point. The evaluated openPangu-7B mask restores `o_proj` and `down_proj`; Qwen3-30B MoE restores attention layers. Other eligible layers use INT8 operands.

These masks are model-specific and fixed in the reported experiments; they do not establish online adaptive selection. W8A8 with fallback therefore denotes mixed-precision execution.

Refreshing rollout weights. After each policy update, we regenerate rollout weights from the latest parameters and quantization state, including $W_k S_k$ and its quantization steps. This refresh tracks the learned parameterization but cannot remove backend arithmetic differences.

Residual policy mismatch. QAT, rounding, and fallback change policy computation; importance weighting adjusts samples from a different behavior policy. Truncating importance weights limits extreme contributions at the cost of bias. Our ablations isolate quantization components and do not separately measure the benefit of this correction.

4 Experiments

4.1 Experimental Setup

We evaluate openPangu-7B and Qwen3-30B MoE as representative dense and mixture-of-experts architectures, with model-family background provided by Chen et al. (2025) and Yang et al. (2025). The experiments use examples from GSM8K (Cobbe et al., 2021) and DAPO-Math-17K (Yu et al., 2025). The reported GSM8K setting uses approximately 1.32K examples and a response-length limit of 512 tokens; the DAPO-Math-17K setting uses approximately 1.7K examples and a limit of 1,024 tokens. Experiments run on a single Ascend server with 16 NPUs and 64 GB of memory per device. The quality experiments use GRPO with verl v0.7.0 and vLLM/vLLM Ascend v0.13.0. All compared configurations are reported under shared training and evaluation settings unless specified otherwise.

Configurations and metrics. We compare BF16 training and rollout, weight-only QAT (QAT-W8), and weight-and-activation QAT (QAT-W8A8). WS denotes learnable weight scales, SS denotes learnable smoothing factors, SR denotes hash-based rounding, and MP denotes selective mixed-precision fallback. We retain *Reward* and *Pearson* as the metric names used in the experimental records. Reward measures task performance under the reported reward procedure; Pearson is a correlation diagnostic for the compared training and rollout quantities. Correlation alone does not establish equality of probability distributions.

4.2 Main Results

Table 1 and Figure 5 summarize the main results. On GSM8K, adding SR and MP to QAT-W8/W8A8 raises openPangu-7B reward from 0.451 to 0.595 (BF16: 0.588), and Qwen3-30B MoE from 0.782 to 0.822 (BF16: 0.836). Recovery thus depends on the model; the latter retains a 0.014 absolute gap.

On DAPO-Math-17K, the openPangu-7B QAT-W8A8 baseline reaches 0.252. Adding MP, MP+SS, and MP+SS+SR yields 0.280, 0.299, and 0.305, respectively. The final W8A8 result is 0.004 below BF16 (1.29% relative); W8A16 reaches 0.303. These nested configurations support the combined method without isolating additive component effects.

Reward and correlation capture different behavior. Figure 6 shows that reward and Pearson correlation need not improve together. On DAPO, adding SS and then SR to MP increases reward from 0.280 to 0.299 to 0.305, while Pearson falls from 0.9909 to 0.9872 to 0.9825. W8A16 instead reaches Pearson 0.9985 with reward 0.303. Correlation is therefore a supplementary numerical diagnostic, not a substitute for task performance or policy divergence.

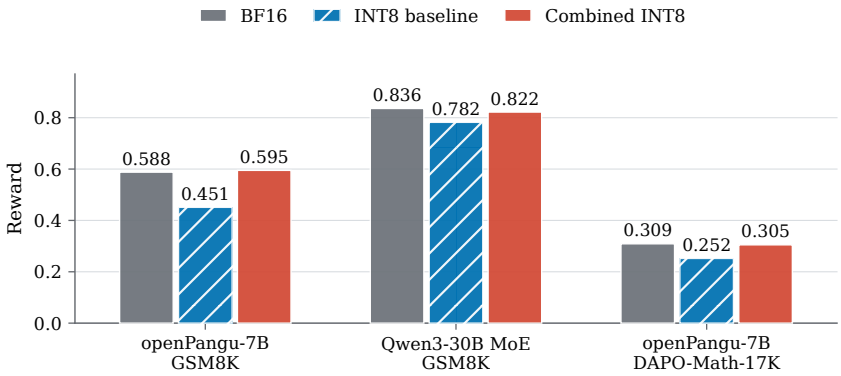

Figure 5: Task reward for BF16, the INT8 baseline, and combined INT8 settings from Table 1. GSM8K combines QAT-W8 with SR and MP; DAPO combines QAT-W8A8 with MP, SS, and SR. All INT8 bars use W8A8 rollout.

Table 1: Task reward and correlation under BF16 and INT8 rollouts. SR, MP, and SS denote hash-based rounding, selective precision fallback, and learnable smoothing, respectively. A dash in a metric column indicates an unreported result.

Quantization Setting					openPangu-7B				Qwen3-30B MoE	
Training	Rollout	SR	MP	SS	GSM8K		DAPO-Math-17K		GSM8K	
					Reward	Pearson	Reward	Pearson	Reward	Pearson
BF16	BF16	–	–	–	0.588	0.9994	0.309	0.9986	0.836	0.9992
QAT-W8	W8A8	–	–	–	0.451	0.9723	–	–	0.782	0.9903
QAT-W8	W8A8	✓	–	–	0.543	0.9613	–	–	0.816	0.9908
QAT-W8	W8A8	–	✓	–	0.496	0.9854	–	–	0.812	0.9977
QAT-W8	W8A8	✓	✓	–	**0.595**	0.9862	–	–	**0.822**	**0.9978**
QAT-W8A8	W8A8	–	–	–	–	–	0.252	0.9802	–	–
QAT-W8A8	W8A8	–	✓	–	–	–	0.280	0.9909	–	–
QAT-W8A8	W8A8	–	✓	✓	–	–	0.299	0.9872	–	–
QAT-W8A8	W8A8	✓	✓	✓	–	–	**0.305**	0.9825	–	–
QAT-W8	W8A16	✓	✓	✓	–	–	0.303	**0.9985**	–	–

4.3 Component Ablations

Learnable scales. Figure 7 and Appendix Table A1 compare the scale mechanisms. WS raises GSM8K reward from 0.444 to 0.451 under QAT-W8; SS raises DAPO reward from 0.252 to 0.277 under QAT-W8A8. Because these comparisons use different datasets and QAT settings, they do not isolate a joint-scale interaction.

Hash-based rounding. The dedicated rounding ablation (Appendix Table A2) reports GSM8K reward gains from 0.451 to 0.550 on openPangu-7B and from 0.782 to 0.816 on Qwen3-30B MoE. Pearson changes in opposite directions across models. These observations support SR empirically without identifying whether its benefit arises from bias reduction or optimization effects. The original SR-only records differ between tables; Appendix A documents this discrepancy.

Selective fallback. Appendix Table A3 reports fallback gains. Under QAT-W8, MP raises reward from 0.451 to 0.496 on openPangu-7B and from 0.782 to 0.812 on Qwen3-30B MoE. Under QAT-W8A8, openPangu-7B improves from 0.414 to 0.488. Measuring the associated efficiency cost also requires the fraction of computation retained in higher precision.

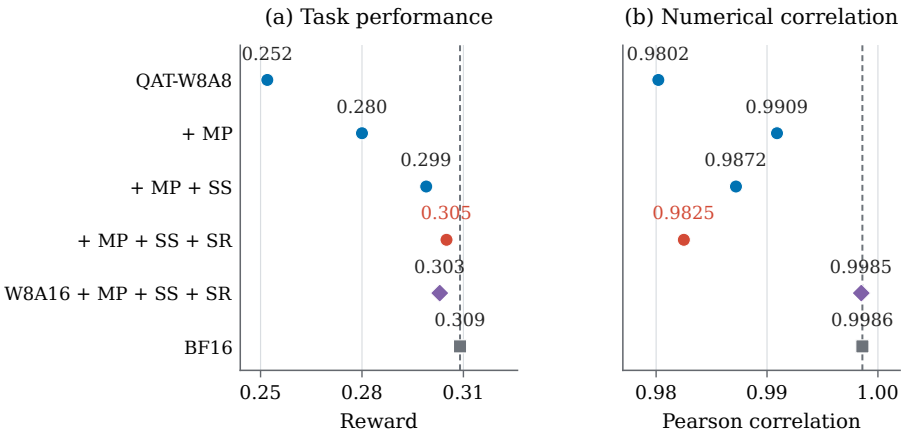

Figure 6: DAPO ablations on openPangu-7B: reward and Pearson correlation. The first four rows use QAT-W8A8/W8A8; the fifth uses QAT-W8/W8A16. Dashed lines mark BF16. Reward and correlation need not improve together; horizontal axes are zoomed.

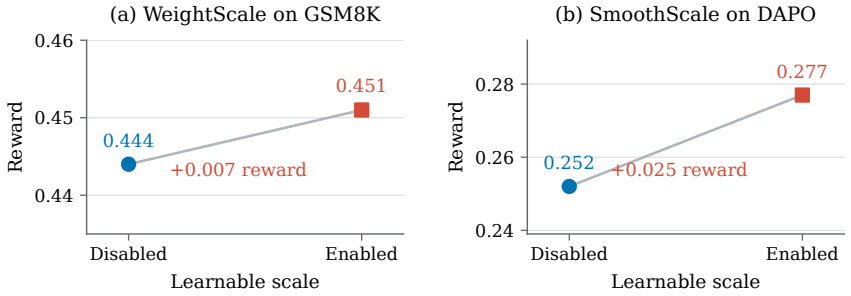

Figure 7: Scale ablations on openPangu-7B with W8A8 rollout: WS under QAT-W8/GSM8K and SS under QAT-W8A8/DAPO. Segments compare discrete settings, not training steps. Vertical ranges differ; the two experiments do not isolate a joint-scale interaction.

Combined effects of rounding and fallback. Figure 8 uses the four configurations in Table 1. Enabling MP adds 0.045 reward without SR and 0.052 with SR on openPangu-7B; corresponding MoE gains are 0.030 and 0.006. These descriptive differences suggest model-dependent component effects, without establishing statistical interactions.

4.4 System Efficiency

Measured rollout latency. Figure 9(a,b) reports openPangu-7B latency. W8A8 reduces prefill from 100.5 to 92.4 ms (8.06%) and per-token decode from 26.4 to 21.2 ms (19.7%; approximately 1.25×). End-to-end gains also depend on batching, sequence lengths, and training cost.

Training overhead and optimization headroom. QAT adds fake-quantization kernels and memory traffic. To assess optimization headroom, we disable training-side fake quantization while retaining W8A8 rollout. At response length 10K and batch size 16, openPangu-7B step time changes from 272.48 s (BF16) to 221.52 s:

$$\text{step-time reduction} = \frac{272.48 - 221.52}{272.48} \approx 18.7\%, \qquad \text{speedup} = \frac{272.48}{221.52} \approx 1.23\times. \tag{14}$$

This ablation changes the training forward computation. Its 18.7% time reduction (Figure 9(c)) is an implementation reference, not a measured gain of the complete QAT method or evidence of preserved reward. Realizing it with QAT requires reducing training overhead.

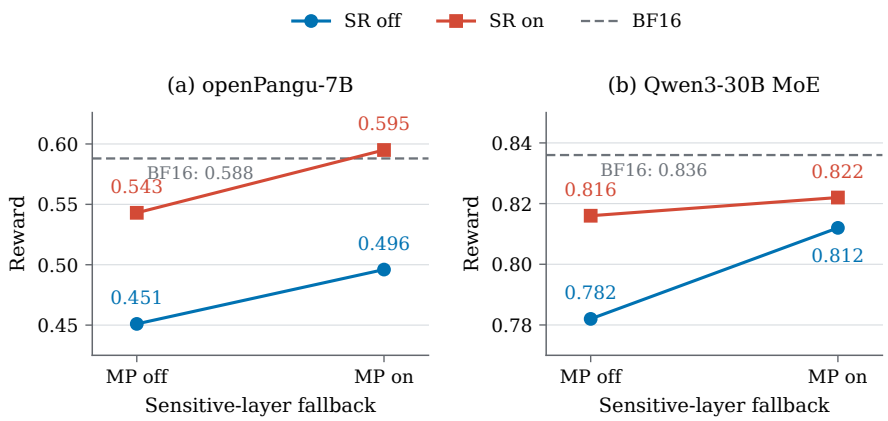

Figure 8: SR/MP comparisons on GSM8K with QAT-W8/W8A8, using Table 1. Dashed lines mark BF16. Segments connect discrete configurations, not training steps; panels use different vertical ranges.

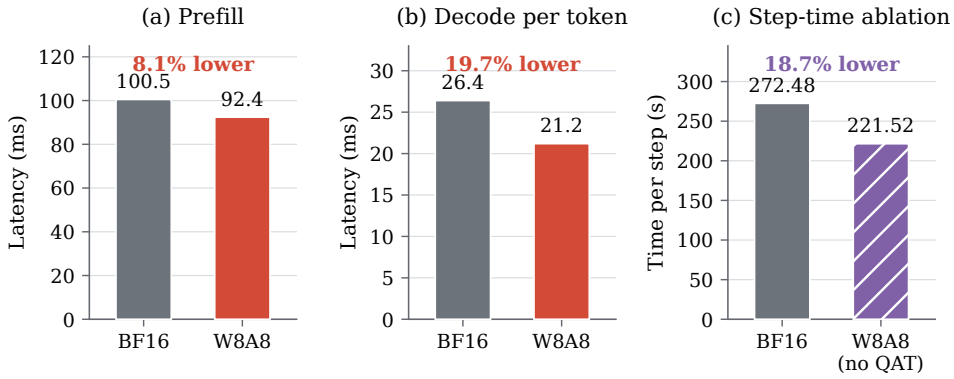

Figure 9: openPangu-7B timings: prefill, per-token decode, and a separate step-time ablation. Panel (c) disables training-side fake quantization while retaining W8A8 rollout (10K response length; batch size 16). Its 18.7% reduction is not the measured speedup of the complete QAT method.

5 Conclusion

INT8-RL combines learned smoothing and weight scales, reproducible rounding, and selective fallback to recover reward under INT8 rollouts. Experiments on dense and MoE models show that reward and numerical correlation can diverge, while faster INT8 decoding does not directly establish full-pipeline speedup.

The evidence is limited to the reported models, mathematical-task subsets, and execution settings. It does not establish multi-seed robustness, broad generalization, or adaptive mask selection. Priorities are matched comparisons with quantized-rollout and QAT-alignment methods, complete policy-mismatch diagnostics, and lower-overhead QAT under matched quality settings.

References

Yuntao Bai, Andy Jones, Kamal Ndousse, et al. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *arXiv preprint arXiv:2204.05862*, 2022. URL https://arxiv.org/abs/2204.05862.

Hanting Chen, Yasheng Wang, Kai Han, et al. Pangu Embedded: An Efficient Dual-System LLM Reasoner with Metacognition. *arXiv preprint arXiv:2505.22375*, 2025. URL https://arxiv.

org/abs/2505.22375.

Zhoujun Cheng, Shibo Hao, Tianyang Liu, et al. Revisiting Reinforcement Learning for LLM Reasoning from a Cross-Domain Perspective. *arXiv preprint arXiv:2506.14965*, 2025. URL https://arxiv.org/abs/2506.14965.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, et al. SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-Training. *arXiv preprint arXiv:2501.17161*, 2025. URL https://arxiv.org/abs/2501.17161.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Maxim Fishman, Brian Chmiel, Ron Banner, et al. Scaling FP8 Training to Trillion-Token LLMs. *arXiv preprint arXiv:2409.12517*, 2024. URL https://arxiv.org/abs/2409.12517.

Jahid Hasan. Optimizing Large Language Models through Quantization: A Comparative Analysis of PTQ and QAT Techniques. *arXiv preprint arXiv:2411.06084*, 2024. URL https://arxiv.org/abs/2411.06084.

Wei Huang, Yangdong Liu, Haotong Qin, et al. BiLLM: Pushing the Limit of Post-Training Quantization for LLMs. *arXiv preprint arXiv:2402.04291*, 2024. URL https://arxiv.org/abs/2402.04291.

Wei Huang, Yi Ge, Shuai Yang, et al. QeRL: Beyond Efficiency – Quantization-Enhanced Reinforcement Learning for LLMs. *arXiv preprint arXiv:2510.11696*, 2025. URL https://arxiv.org/abs/2510.11696.

Bowen Jin, Hansi Zeng, Zhenrui Yue, et al. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. *arXiv preprint arXiv:2503.09516*, 2025. URL https://arxiv.org/abs/2503.09516.

Jihoon Kim, Minseok Seo, and Xuan Truong Nguyen. Mixed INT4-INT8 LLM Quantization via Progressive Layerwise Assignment with Dynamic Sensitivity Estimation. In *2025 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5. IEEE, 2025. doi: 10.1109/ISCAS56072.2025.11043378. URL https://ieeexplore.ieee.org/document/11043378.

Mike Lasby, Ivan Lazarevich, Nish Sinnadurai, et al. REAP the Experts: Why Pruning Prevails for One-Shot MoE Compression. *arXiv preprint arXiv:2510.13999*, 2025. URL https://arxiv.org/abs/2510.13999.

Yuhang Li, Reena Elangovan, Xin Dong, et al. QuRL: Efficient Reinforcement Learning with Quantized Rollout. *arXiv preprint arXiv:2602.13953*, 2026. URL https://arxiv.org/abs/2602.13953.

Yilun Luo, Huaqing Zheng, Haoqian Meng, et al. Post-Training Quantization of OpenPangu Models for Efficient Deployment on Atlas A2. *arXiv preprint arXiv:2512.23367*, 2025. URL https://arxiv.org/abs/2512.23367.

Houwen Peng, Kan Wu, Yixuan Wei, et al. FP8-LM: Training FP8 Large Language Models. *arXiv preprint arXiv:2310.18313*, 2023. URL https://arxiv.org/abs/2310.18313.

Penghui Qi, Zichen Liu, Xiangxin Zhou, et al. Defeating the Training-Inference Mismatch via FP16. *arXiv preprint arXiv:2510.26788*, 2025. URL https://arxiv.org/abs/2510.26788.

Maanas Taneja and Purab Shingvi. GPU-Accelerated INT8 Quantization for KV Cache Compression in Large Language Models. *arXiv preprint arXiv:2601.04719*, 2026. URL https://arxiv.org/abs/2601.04719.

Haocheng Xi, Han Cai, Ligeng Zhu, et al. COAT: Compressing Optimizer States and Activation for Memory-Efficient FP8 Training. *arXiv preprint arXiv:2410.19313*, 2024. URL https://arxiv.org/abs/2410.19313.

Haocheng Xi, Charlie Ruan, Peiyuan Liao, et al. Jet-RL: Enabling On-Policy FP8 Reinforcement Learning with Unified Training and Rollout Precision Flow. *arXiv preprint arXiv:2601.14243*, 2026. URL https://arxiv.org/abs/2601.14243.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 38087–38099. PMLR, 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

Xianglong Yan, ChengZhu Bao, Zhiteng Li, et al. D^2Quant: Accurate Low-bit Post-Training Weight Quantization for LLMs. *arXiv preprint arXiv:2602.02546*, 2026. URL https://arxiv.org/abs/2602.02546.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Qiying Yu, Zheng Zhang, Ruofei Zhu, et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. In *Advances in Neural Information Processing Systems*, volume 38, pp. 113222–113244, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Kaichen Zhang, Yuzhong Hong, Junwei Bao, et al. GVPO: Group Variance Policy Optimization for Large Language Model Post-Training. *arXiv preprint arXiv:2504.19599*, 2025a. URL https://arxiv.org/abs/2504.19599.

Ziyang Zhang, Xinheng Ding, Jiayi Yuan, et al. Deterministic Inference across Tensor Parallel Sizes That Eliminates Training-Inference Mismatch. *arXiv preprint arXiv:2511.17826*, 2025b. URL https://arxiv.org/abs/2511.17826.

APPENDIX

A DETAILED COMPONENT ABLATIONS

The following tables retain the original reported observations. Table A1 lists scale ablations; Tables A2 and A3 give the dedicated rounding and fallback experiments.

Table A1: Learnable-scale ablations on openPangu-7B. WS denotes learnable weight scales and SS denotes learnable smoothing. The GSM8K and DAPO-Math-17K comparisons use different QAT configurations.

Quantization Setting				GSM8K		DAPO-Math-17K	
Training	**Rollout**	**WS**	**SS**	**Reward**	**Pearson**	**Reward**	**Pearson**
BF16	BF16	–	–	0.588	0.9994	0.309	0.9986
BF16	W8A8	–	–	0.411	0.9613	–	–
QAT-W8	W8A8	–	–	0.444	0.9478	–	–
QAT-W8	W8A8	✓	–	**0.451**	**0.9723**	–	–
QAT-W8A8	W8A8	–	–	–	–	0.252	0.9802
QAT-W8A8	W8A8	–	✓	–	–	**0.277**	**0.9858**

Analysis of learnable scales. Table A1 separates weight-scale learning from activation smoothing. On GSM8K, enabling WS under QAT-W8 increases reward from 0.444 to 0.451 and Pearson correlation from 0.9478 to 0.9723, absolute gains of 0.007 and 0.0245. On DAPO-Math-17K, enabling SS under QAT-W8A8 raises reward from 0.252 to 0.277 and correlation from 0.9802 to 0.9858. The latter comparison supports learnable smoothing when both weights and activations are quantized, although reward remains 0.032 below BF16. Both comparisons favor adapting the quantization parameters in their respective settings. Because the datasets and training precisions differ, however, the table does not establish that SS is intrinsically more effective than WS or quantify the benefit of learning both jointly.

Table A2: Hash-based rounding ablation on GSM8K for openPangu-7B and Qwen3-30B MoE. SR denotes the reproducible hash-based rounding implementation.

Quantization Setting			openPangu-7B		Qwen3-30B MoE	
Training	**Rollout**	**SR**	**Reward**	**Pearson**	**Reward**	**Pearson**
BF16	BF16	–	0.588	0.9994	0.836	0.9992
QAT-W8	W8A8	–	0.451	0.9723	0.782	0.9903
QAT-W8	W8A8	✓	**0.550**	0.9514	**0.816**	0.9928
QAT-W8A8	W8A8	✓	0.492	0.9281	–	–

Analysis of hash-based rounding. Under QAT-W8 with W8A8 rollout, Table A2 reports reward increases of 0.099 on openPangu-7B (0.451 to 0.550) and 0.034 on Qwen3-30B MoE (0.782 to 0.816) after enabling SR. These results support reward recovery across both evaluated architectures. The correlation response differs: Pearson decreases from 0.9723 to 0.9514 on openPangu-7B but increases from 0.9903 to 0.9928 on Qwen3-30B MoE. Thus, improved reward need not coincide with stronger numerical correlation. The QAT-W8A8 row reports reward 0.492, but this table contains no corresponding SR-disabled control, so it cannot isolate the SR effect under that training precision. These aggregate observations do not establish unbiased rounding or statistical significance; the discrepancies with the main-results table are documented below.

Table A3: Selective precision fallback on GSM8K. The openPangu-7B mask restores `o_proj` and `down_proj`; the Qwen3-30B MoE mask restores attention layers.

Quantization Setting			openPangu-7B		Qwen3-30B MoE	
Training	**Rollout**	**MP**	**Reward**	**Pearson**	**Reward**	**Pearson**
BF16	BF16	–	–	–	0.836	0.9992
QAT-W8	W8A8	–	0.451	0.9723	0.782	0.9903
QAT-W8	W8A8	✓	**0.496**	**0.9856**	**0.812**	**0.9977**
QAT-W8A8	W8A8	–	0.414	0.9656	–	–
QAT-W8A8	W8A8	✓	**0.488**	**0.9804**	–	–

Analysis of selective precision fallback. Table A3 shows reward and correlation gains for each reported fallback comparison. Under QAT-W8, openPangu-7B reward increases from 0.451 to 0.496 and Pearson from 0.9723 to 0.9856; Qwen3-30B MoE improves from 0.782 to 0.812 and from 0.9903 to 0.9977, respectively. For openPangu-7B under QAT-W8A8, fallback raises reward from 0.414 to 0.488, an absolute gain of 0.074, and correlation from 0.9656 to 0.9804. The reported reward gain is larger than the 0.045 gain under QAT-W8, suggesting that the selected mask may be particularly useful when activations are also quantized. This remains a configuration-specific observation. Since fallback retains higher-precision computation, assessing its quality–efficiency trade-off requires the retained computation fraction and matched latency measurements; the table alone does not establish an optimal mask.

Consistency across reported tables. The original records contain repeated-configuration discrepancies. The openPangu SR-only row is 0.543/0.9613 (reward/Pearson) in Table 1 and 0.550/0.9514 in Table A2; Qwen SR-only Pearson is 0.9908 versus 0.9928. OpenPangu MP-only Pearson is 0.9854 versus 0.9856 in Tables 1 and A3. Values are preserved rather than averaged. The SR/MP interaction plot consistently uses Table 1; configuration, checkpoint, and seed provenance must be reconciled before interpreting these as matched repeats.

B Use of Large Language Models

We used ChatGPT primarily to assist with writing and language polishing, including improving sentence clarity, readability, and manuscript organization. It also provided auxiliary support for literature cross-checking, clarifying mathematical exposition, and visualizing the reported experimental results. The authors remain responsible for the scientific content, interpretations, and conclusions of the paper.