AUDITING DYNAMIC SENSITIVITY-SELECTIVE QUANTIZATION FOR LOW-PRECISION LLM REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

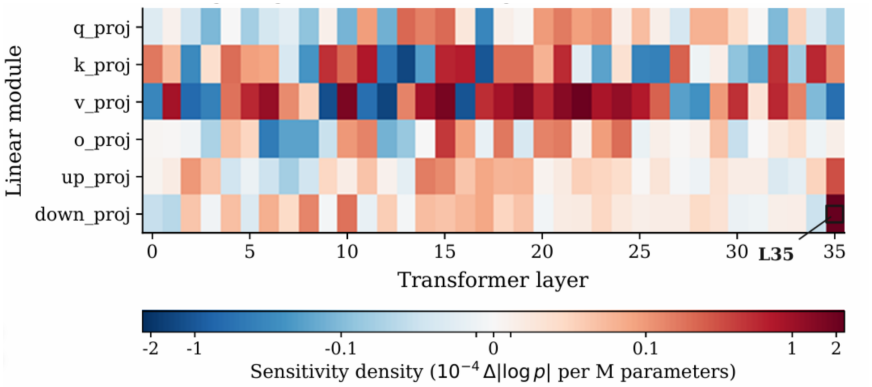
Reinforcement learning has become a critical post-training phase for large language models, where the rollout stage dominates the computational and memory overhead. Low-precision rollout reduces the cost of reinforcement learning for large language models but introduces a rollout-training mismatch that can destabilize optimization and even lead to collapse. Selective high-precision protection mitigates this mismatch, but the modules that benefit most from such protection can change as the policy evolves. Consequently, a protection set determined through one-time calibration may become suboptimal during training. We formulate selective protection as a time-varying allocation problem under a fixed high-precision budget, periodically assessing the sensitivity of each module to low-precision quantization and reconfiguring the protection strategy based on their current benefit-to-cost ratios. This allocation mechanism complements Quantization-Aware Training (QAT): while QAT adapts weights to low-precision computation, our method adapts precision assignments to the evolving module sensitivities. In the three-seed Infer-NVFP4 study, Dynamic lowers the mean step-120 mismatch and yields the highest mean validation accuracy, although one seed later develops a severe excursion. With QAT, all observed trajectories stay below the post-hoc mismatch threshold of 0.1 through step 172; Dynamic has the lowest mean peak mismatch over the common horizon and the highest mean step-170 accuracy among the QAT variants. These results support checkpoint-dependent allocation in the evaluated setting without implying that it eliminates seed sensitivity or is uniformly superior.

1 INTRODUCTION

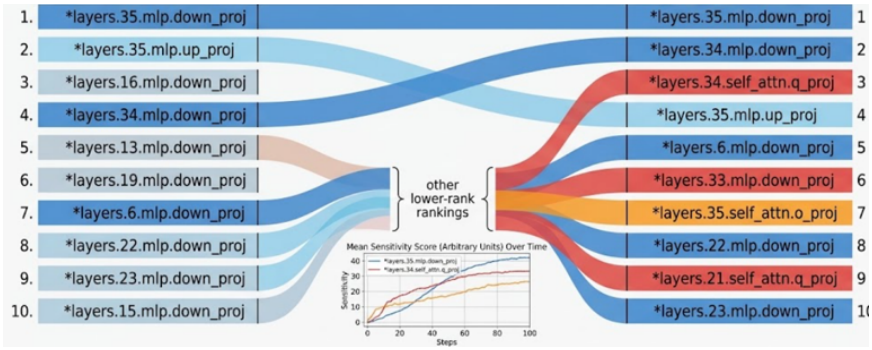
Reinforcement learning has become a central component of large-language-model (LLM) post-training (Ouyang et al., 2022; Yu et al., 2025). Its rollout stage can dominate end-to-end training time, motivating lower-precision generation (Qiu et al., 2026; Xi et al., 2026). In RL, however, the policy updated by the trainer must remain aligned with the policy that generates trajectories. Quantizing the rollout path perturbs this alignment, and recent FP8 RL studies report that precision-induced rollout–training mismatch can destabilize optimization under long rollouts (Qiu et al., 2026; Xi et al., 2026). If the mismatch grows, token-level importance weights become distorted. Training length alone therefore does not characterize low-precision stability.

Post-training quantization compresses pretrained models without full retraining. GPTQ (Frantar et al., 2023), SmoothQuant (Xiao et al., 2023), ZeroQuant (Yao et al., 2022), and OmniQuant (Shao et al., 2024) develop complementary weight, activation, and calibration strategies for LLM compression. Selective and mixed-precision methods further exploit heterogeneous sensitivity: HAWQ-V2 assigns precision across layers (Dong et al., 2020), while LLM.int8(), AWQ, and SpQR reserve higher precision or extra protection for outlier dimensions, salient channels, or outlier weights (Dettmers et al., 2022; Lin et al., 2024; Dettmers et al., 2024). QAT methods such as LLM-QAT (Liu et al., 2024) instead adapt the model during training. In RL, however, both the model and rollout distribution evolve, so a protected set chosen at step zero can become stale; applying QAT uniformly may also be expensive or insufficient for an aggressively quantized rollout.

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.



(a) Sensitivity differs across layers and module types.



(b) The most sensitive modules change during RL.

Figure 1: Empirical motivation for dynamic selection. (a) A checkpoint-level heatmap shows the signed, cost-normalized sensitivity density across transformer layers and linear-module types. (b) The top-10 ranking changes between steps 0 and 100: six modules persist, while four are replaced and attention projections enter the Top-10 sensitivity ranking. Together, the observations motivate module-wise protection and checkpoint-dependent reallocation.

Motivating observations. Our sensitivity probes reveal two properties that make a one-time allocation questionable. First, sensitivity is heterogeneous across both transformer depth and linear-module type: at the same checkpoint, the signed sensitivity density varies substantially among `q_proj`, `k_proj`, `v_proj`, `o_proj`, `up_proj`, and `down_proj` modules (Figure 1a). Second, sensitivity is not stationary during RL. Comparing the ten most sensitive modules at steps 0 and 100, only six remain in the list; four initial MLP projections are replaced, including by attention projections (Figure 1b). The first observation motivates selective rather than uniform protection, while the second motivates recomputing the allocation as the policy evolves.

Guided by these observations, we test whether a fixed protection budget should track the policy during training. Dynamic sensitivity-selective quantization (DynSens) toggles candidate linear modules in teacher-forced probes and normalizes each marginal benefit by its protection cost. A deterministic density-greedy allocator then updates the protected set at 20-step boundaries, whereas Static selection allocates the same budget only at initialization. We study this allocation mechanism both on an inference-only low-precision path and together with W4A16 fake-QAT, asking whether weight adaptation and precision-allocation adaptation address different parts of the mismatch.

Our contributions center on the method design. First, we formulate sensitivity-based selective quantization as a time-varying allocation problem under a fixed high-precision budget, allowing the protected set to follow an RL policy whose rollout distribution changes during optimization. Second, we introduce a checkpoint-dependent sensitivity probe and a cost-normalized deterministic allocator that updates the protected modules at segment boundaries without changing the budget. Third, we combine adaptive precision allocation with QAT: QAT adapts the weights to low-precision computation, while the selector adapts module-wise precision assignments as the policy evolves.

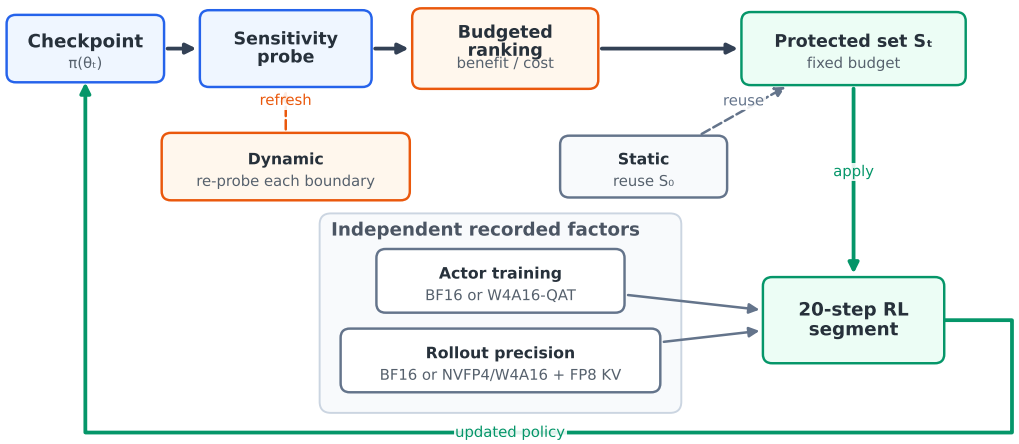


Figure 2: The allocation loop. A checkpoint-dependent probe produces a protected set for the next 20-step segment. Static selection keeps the step-0 set, while Dynamic re-estimates it at each boundary. Actor training mode and rollout precision remain separate recorded factors.

2 Related Work

LLM quantization and selective precision. Post-training methods use second-order information, activation smoothing, fine-grained kernels, or learned calibration to reduce quantization error (Frantar et al., 2023; Xiao et al., 2023; Yao et al., 2022; Shao et al., 2024). Selective schemes instead protect outlier dimensions, salient channels, vulnerable weights, or sensitive layers (Dettmers et al., 2022; Lin et al., 2024; Dettmers et al., 2024; Dong et al., 2020). These methods motivate non-uniform precision, but do not reallocate a fixed protection budget as an RL policy evolves.

Quantization-aware training and FP4. LLM-QAT adapts a model to low-bit operators during training (Liu et al., 2024), while NVFP4 targets very low precision at scale (NVIDIA et al., 2025). We treat W4A16 fake-QAT as a separate training bundle; because its runs also change fused-slice and MBridge settings, they do not isolate a causal QAT–Dynamic interaction.

RL systems and mismatch. DAPO provides the underlying scalable RL setting (Yu et al., 2025). Jet-RL and FP8-RL study rollout cost and numerical mismatch between training and generation (Xi et al., 2026; Qiu et al., 2026). Their remedies align precision across the system; we allocate high-precision protection among rollout linear modules and update that allocation at RL checkpoints.

3 Dynamic Sensitivity-Selective Quantization

3.1 Fixed-budget allocation

Let π_{θ_t} denote the actor after update t and $\widetilde{\pi}_t$ the rollout policy that collects the next batch. During a rollout segment, modules in S_t are protected in BF16 and the remaining candidate modules follow the low-precision path. We keep the protection budget B fixed across segments and use the following additive surrogate for the allocation problem:

$$\max_{z_i \in \{0,1\}} \sum_{i \in \mathcal{L}} z_i v_i(t) \quad \text{s.t.} \quad \sum_{i \in \mathcal{L}} z_i n_i \le B, \qquad S_t = \{i : z_i = 1\}, \tag{1}$$

where $\mathcal{L}$ is the candidate set, $v_i(t)$ is the estimated protection benefit, and n_i is the parameter count. Attention and MLP projections have different sizes, so raw-benefit ranking can spend the budget on large modules and fixed-count selection does not hold protected memory constant. We instead rank

benefit per protected parameter:

$$\rho_i(t) = \frac{v_i(t)}{\max(n_i, 1)}, \qquad i_1, \ldots, i_{|\mathcal{L}|} = \mathrm{sort}_{i \in \mathcal{L}}\big(\rho_i(t), v_i(t)\big) \text{ descending.} \tag{2}$$

Because BF16 cost is $2n_i$ bytes, this is also benefit per protected byte. The deterministic selector scans the order once, accepts any module that fits, and breaks ties by fixed layer and module order. It is a density-greedy approximation, not an exact knapsack solver. We use $B = 503{,}316{,}480$ selectable parameters; `lm_head` and `*gate_proj` remain protected outside this budget.

3.2 Checkpoint-dependent sensitivity

At each selection boundary, the probe loads the current checkpoint and evaluates a fixed teacher-forced batch. Its BF16 token log probabilities provide the reference. The same inputs are then evaluated with the protected set from the preceding segment, yielding the mixed-precision baseline error $E_{\mathrm{mix}}(t)$, defined as the mean absolute token-log-probability difference from that reference. Each candidate is toggled once while all other assignments and inputs remain unchanged. If S_t^- denotes the set active immediately before reselection and $E_i^{\mathrm{toggle}}(t)$ is the error after toggling module i, the recorded marginal value is

$$v_i(t) = \begin{cases} E_{\mathrm{mix}}(t) - E_i^{\mathrm{toggle}}(t), & i \notin S_t^-, \\ E_i^{\mathrm{toggle}}(t) - E_{\mathrm{mix}}(t), & i \in S_t^-. \end{cases} \tag{3}$$

For an unprotected module, the toggle restores BF16; for a protected module, it removes BF16 protection. A positive $v_i(t)$ therefore indicates benefit in either case. Candidate probes can be sharded, but their rows are merged before one deterministic ranking. The score is a forward sensitivity proxy, not a ground-truth training signal.

The candidate set contains six linear projections in each of 36 transformer layers, for 216 candidates. Probe construction, truncation, and token aggregation are specified in the supplementary appendix.

Static computes S_0 once. Dynamic holds the allocation fixed within each 20-step segment and repeats the probe and budgeted selection at the next boundary. Thus the protected parameter count remains fixed while its module-wise placement follows the evolving policy.

3.3 Cost-aware adaptive reselection

The policy above scans all 216 candidates at each boundary; we call it *Fixed-Full Dynamic*. Stateful Active-Set Incremental Reselection (SAIR) instead probes the preceding protected set, near-cutoff candidates, and rotating sentinels. It accepts this local ranking only when deterministic guards pass and otherwise reconstructs the full scan. SAIR preserves the checkpoint, restart, and budget protocol; the active-set construction and guard thresholds appear in the supplementary appendix.

3.4 Precision branches

We compare three precision configurations. *Infer-NVFP4* keeps actor optimization in BF16; rollout uses the NVFP4/W4A16 path with an FP8 KV cache. *W4A16-QAT* adds group-16 W4A16 fake quantization to actor training, with fused-slice QAT and MBridge enabled; rollout and KV precision match Infer-NVFP4. The *BF16* control uses full precision throughout. Because the QAT branch changes multiple runtime flags together, comparisons reflect the full configuration bundle.

3.5 Observables and censoring

Our primary stability signal is the signed rollout–training log-probability gap

$$m_t = \frac{1}{N_t} \sum_{j=1}^{N_t} \frac{1}{|R_{t,j}|} \sum_{k \in R_{t,j}} \left[\log \pi_{\mathrm{rollout},t}(y_{j,k} \mid y_{j,<k}) - \log \pi_{\mathrm{train},t}(y_{j,k} \mid y_{j,<k}) \right], \tag{4}$$

where $R_{t,j}$ contains valid response tokens. The metric averages within sequences and then across sequences; it is signed and is neither a KL divergence nor an absolute error. We also report validation

macro accuracy and per-step wall time. The post-hoc reference levels $m_t \geq 0.1$ and $m_t \geq 1$ denote mild and severe excursions.

All scalar trajectories reach the common horizon $H = 120$; figures extend to step 172 where observations exist. Threshold events, sample-limit endpoints, and external interruptions are distinct, and missing observations are not imputed. Full endpoint rules appear in the supplementary appendix.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We train Qwen3-8B Base (Yang et al., 2025) on DAPO-17K (Yu et al., 2025) with eight H100 GPUs. The 21-run matrix combines None, Static, and Dynamic selectors with two low-precision actor branches and a BF16 control across three seeds. Twenty runs yield scalar trajectories through step 120; QAT-None seed 33 stops during initialization and is not imputed. Training follows DAPO/GRPO with token-mean loss, no KL reward or loss, and a nominal 180-update horizon. The supplementary appendix specifies seeds, batching, sequence lengths, sampling, and software versions.

Table 1 summarizes the three precision configurations. Rollout and KV precision are fixed across all low-precision rows; only the actor-side settings differ.

Table 1: Precision configurations. Group 16 is used for W4A16 fake quantization.

Branch	Actor optimization	Rollout	KV cache	Selectors
Infer-NVFP4	BF16, no fake quant.	NVFP4/W4A16	FP8	None/Static/Dynamic
W4A16-QAT	W4A16 fake-QAT	NVFP4/W4A16	FP8	None/Static/Dynamic
BF16	BF16, no fake quant.	BF16	BF16	None

Dynamic runs use 20-step checkpoint/restart segments; Static and QAT-None are uninterrupted, while formal Infer-None is segmented. Selector comparisons therefore concern the recorded implementation bundles. Validation runs every 10 steps. We use $H = 120$ for the common stability comparison, step 170 for aligned QAT/BF16 quality, and step 172 only for trajectory display. Values are means and sample standard deviations over available seeds: $n = 3$ except QAT-None ($n = 2$). Missing observations are not imputed.

4.2 MAIN STABILITY COMPARISON

This experiment tests whether Dynamic selection controls rollout–training mismatch where static or no protection fails, and whether it provides additional benefit when combined with QAT.

Table 2 reports the step-120 comparison. In Infer-NVFP4, Dynamic reduces the incidence and severity of mismatch excursions but does not eliminate them: one of three Dynamic seeds crosses $m = 0.1$ by step 120 and none crosses $m = 1$, compared with two severe crossings for None and three for Static. Its mean m_{120} is 0.217, versus 2.871 for None and 3.114 for Static, while its mean validation accuracy is 0.413, versus 0.306 and 0.227. The new seed also exposes residual variability: Dynamic seed 33 crosses 0.1 at step 105 and later becomes severe. This seed-specific behavior is consistent with a finite-rollout hypothesis in which a small number of high-advantage or heavy-importance-weight trajectories steer an update into a region that is more sensitive to quantization perturbations. We treat this as a mechanism hypothesis rather than an identified cause.

All observed QAT trajectories remain below both thresholds through step 120. The per-seed peak signed gap, averaged within each configuration, is 0.0146 for Dynamic, 0.0291 for Static, and 0.0326 for None, so Dynamic roughly halves the average peak even when QAT already keeps the gap small. Endpoint mismatch is similar across the three QAT selectors, whereas Dynamic has the highest mean step-120 validation accuracy (0.455, compared with 0.447 for None and 0.443 for Static).

Beyond step 120, every observed QAT trajectory remains below $m = 0.1$ through the step-172 display window. Dynamic has the smallest observed maximum (0.022, compared with 0.039 for

Table 2: Three-seed summary at the common step-120 horizon. Values are mean ± sample standard deviation. All configurations use $n = 3$ except QAT-None ($n = 2$), whose seed-33 attempt produced no scalar trajectory.

Actor branch	Selector	$m \geq 0.1$	$m \geq 1$	m_{120}	Val. macro$_{120}$	Step time (s)
Infer-NVFP4	None	2/3	2/3	2.871 ± 3.079	0.306 ± 0.123	336.2 ± 18.3
Infer-NVFP4	Static	3/3	3/3	3.114 ± 1.222	0.227 ± 0.130	333.2 ± 16.0
Infer-NVFP4	Dynamic	1/3	0/3	0.217 ± 0.314	0.413 ± 0.050	339.0 ± 19.7
W4A16-QAT	None	0/2	0/2	$7.91\times10^{-3} \pm 2.12\times10^{-3}$	0.447 ± 0.003	370.5 ± 3.0
W4A16-QAT	Static	0/3	0/3	$7.11\times10^{-3} \pm 5.71\times10^{-4}$	0.443 ± 0.006	414.1 ± 9.1
W4A16-QAT	Dynamic	0/3	0/3	$7.56\times10^{-3} \pm 6.65\times10^{-4}$	0.455 ± 0.020	410.4 ± 3.7
BF16	None	0/3	0/3	$4.36\times10^{-4} \pm 3.39\times10^{-5}$	0.472 ± 0.012	313.7 ± 1.9

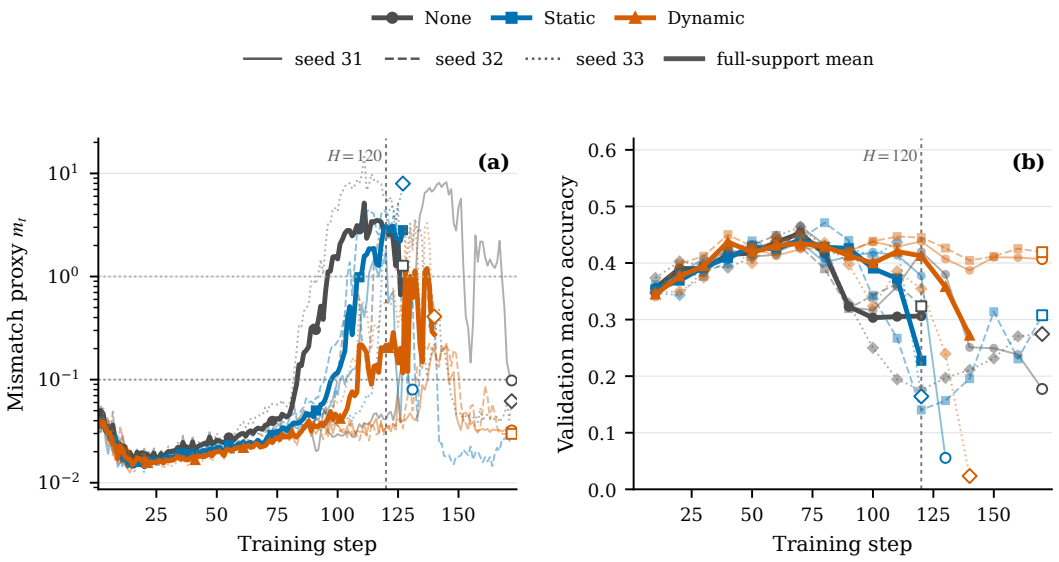


Figure 3: Inference-only selector ablation. (a) Mismatch m_t on a log scale. (b) Validation macro accuracy. Thin lines denote seeds; thick lines are full-support means. The vertical line marks $H = 120$ and horizontal guides mark $m = 0.1$ and $m = 1$.

None and 0.055 for Static), but the threshold criterion alone no longer separates the QAT selectors. In Infer-NVFP4, Dynamic seeds 31 and 32 peak at 0.223 and recover, whereas seed 33 reaches 3.30 and ends at a sample-limit event after step 140. None and Static reach larger display-window peaks of 15.26 and 7.97, respectively. Thus Dynamic improves the aggregate stability profile, but the third seed rules out a claim of universal collapse prevention.

4.3 Dynamic allocation analysis

Saved artifacts confirm active reassignment. Infer-NVFP4 selects 40–60 modules per boundary and reaches a minimum adjacent-set Jaccard overlap of 0.088; QAT selects 51–63 modules and reaches 0.393. Every allocation uses the full budget. Figures 3 and 4 show mismatch and validation accuracy per seed. Full-support means stop with the first endpoint in each group, so the interval after $H = 120$ is trajectory context rather than a common-horizon comparison.

4.4 Aligned quality and runtime cost

At the shared step-170 validation snapshot, QAT-Dynamic leads the QAT variants at 0.460 ± 0.027, followed by QAT-None (0.433 ± 0.025, $n = 2$) and QAT-Static (0.421 ± 0.136); the BF16 baseline reaches 0.500 ± 0.010. We do not form an Infer-NVFP4 step-170 aggregate because no selector retains all three seeds at that checkpoint; their later quality is shown only as per-seed trajectories.

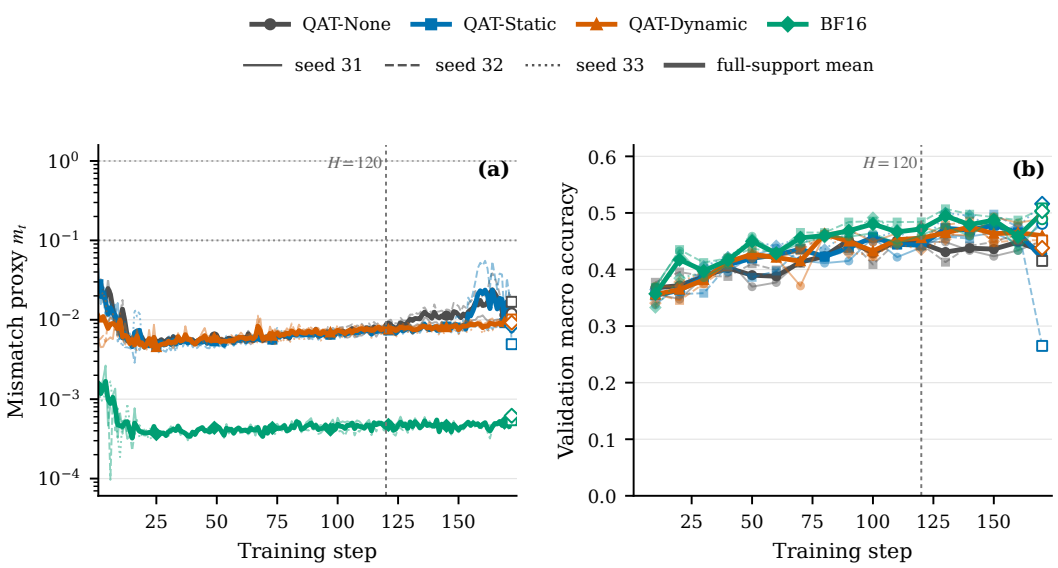


Figure 4: QAT selector ablation with BF16 reference. QAT-None has two observed seeds; the other curves have three. Panels and line styles follow Figure 3.

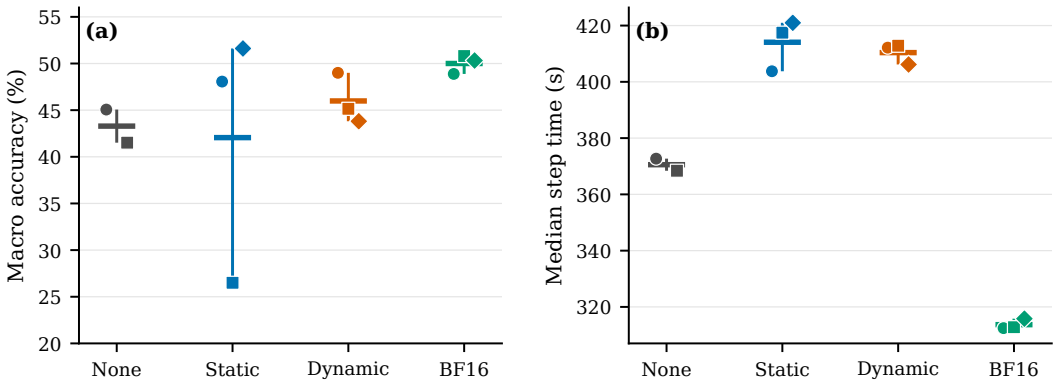


Figure 5: Validation accuracy at step 170 and median logged step time over steps 1–120. Markers denote seeds; horizontal ticks are means and vertical bars are observed ranges, not confidence intervals.

For QAT-Dynamic, the median step time over steps 1–120 is 410.4 ± 3.7 s, compared with 370.5 ± 3.0 s for QAT-None (10.8% overhead). Infer-NVFP4 Dynamic records 339.0 ± 19.7 s, comparable to None (336.2 ± 18.3) and Static (333.2 ± 16.0).

Low precision does not shorten a logged step here. On H100, the Marlin weight-only path computes the full projection before protected slices are recomputed in BF16; requantization and repacking add fixed rollout-phase cost. Native NVFP4 hardware could avoid this duplicate work if protection is aligned to fused-operator boundaries, but that speedup remains unmeasured. The supplementary appendix gives the full operator-path rationale.

4.5 Active-set incremental reselection

In a single-seed W4A16-QAT follow-up, SAIR accepts local reselection at steps 120 and 160. Against Fixed-Full Dynamic at the same boundaries, it shortens the artifact-derived boundary window by 34.9% and 39.9%, or 37.6% on average. This matched comparison measures selector orchestration from saved timestamps, not kernel time or whole-run speedup; the protocol and per-boundary values appear in the supplementary appendix.

5 Analysis and Limitations

In the recorded Qwen3-8B/DAPO setting, Dynamic lowers mean Infer-NVFP4 m_{120} to 0.217, from 2.871 for None and 3.114 for Static, with no severe crossing by $H = 120$. Seed 33 becomes severe later, so reallocation reduces rather than removes seed-dependent failure. Within QAT, Dynamic roughly halves the mean peak gap over $H = 120$ and has the highest step-170 QAT accuracy (0.460 ± 0.027). These results support checkpoint-dependent reassignment as a complement to weight adaptation in this configuration.

The first methodological limitation is the independent-module sensitivity proxy. It measures the marginal forward benefit of protecting one module relative to a BF16 teacher, then ranks modules as if these benefits were approximately separable. Quantization errors can instead amplify or cancel across residual, attention, and MLP paths, so the ranking need not recover the optimal protected subset. Exhaustive joint probing is combinatorial and must be repeated at every boundary, which can erase the savings from low-precision rollout. A practical extension would screen modules individually and rescore only small, structurally local combinations; the supplementary appendix gives the complexity argument.

The second limitation is that forward-output perturbation does not directly measure the training update. Rewards, advantages, and importance weights can amplify examples with small logit changes, while large forward differences need not rotate the policy gradient. Gradient-based sensitivity would be closer to the optimization target, but a finite-rollout BF16 gradient is itself noisy. Separating quantization bias from sampling noise would require paired rollouts, repeated BF16 references, and additional backward passes. A variance-aware selector could first screen modules with the forward proxy and then compare gradient displacement for a small candidate set. The current results support the proxy's utility, not its equivalence to true gradient sensitivity; the supplementary appendix reports the motivating gradient-variance evidence.

The evidence remains scoped to one model and RL recipe. QAT changes fake-quant hooks, fused-slice, and MBridge flags together, and selector schedules are not a matched restart intervention. Six configurations have three observed seeds; QAT-None has two. All 20 scalar trajectories cover $H = 120$, but four Infer runs end before step 172, so no Infer step-170 aggregate is formed. Threshold crossings are post-hoc proxy events, and logged step time excludes selector probes and restarts.

6 Conclusion

Low precision in LLM reinforcement learning is not a numerical choice fixed before training: it changes rollout trajectories, updates, and hence which modules are sensitive next. Dynamic closes this feedback loop by reassigning a fixed high-precision budget as the policy evolves. Across three Infer-NVFP4 seeds, it improves group-level stability and quality over static or no protection, while the late failure of seed 33 shows that allocation does not remove stochastic instability. With QAT, Dynamic roughly halves the mean peak mismatch over $H = 120$ and attains the highest step-170 accuracy (0.460 ± 0.027), supporting complementary adaptation in allocation and weight space. The implementation adds 10.8% logged step time relative to QAT-None before selector and restart costs; at two accepted boundaries, SAIR shortens the artifact-derived reselection window by 37.6% on average.

AI use statement

In this work, generative AI tools were used only to assist with drafting and editing the manuscript. No AI tools were used for the research methodology, experimental design, implementation, experiments, data analysis, figure or table generation, or citation selection and verification. The authors reviewed all AI-assisted writing and take responsibility for the final manuscript.

References

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30318–30332, 2022. doi: 10.52202/068431-2198.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Q1u25ahSuy.

Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ-V2: Hessian aware trace-weighted quantization of neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 18518–18529, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/d77c703536718b95308130ff2e5cf9ee-Abstract.html.

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

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

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. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.26. URL https://aclanthology.org/2024.findings-acl.26/.

NVIDIA et al. Pretraining large language models with nvfp4. *arXiv preprint arXiv:2509.25149*, 2025. doi: 10.48550/arXiv.2509.25149.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022. doi: 10.52202/068431-2011.

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. *arXiv preprint arXiv:2601.18150*, 2026. doi: 10.48550/arXiv.2601.18150.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8Wuvhh0LYW.

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. *arXiv preprint arXiv:2601.14243*, 2026. doi: 10.48550/arXiv.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 of *Proceedings of Machine Learning Research*, pp. 38087–38099. PMLR, 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. ZeroQuant: Efficient and affordable post-training quantization for large-scale transformers. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27168–27183, 2022. doi: 10.52202/068431-1970.

Qiying Yu et al. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.