TRIAGE: DIRECTION-AWARE MISMATCH STABILIZATION OF NATIVE NVFP4 REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-precision execution can substantially accelerate reinforcement learning (RL) for large language models, but discrepancies between learner and sampler execution can destabilize policy optimization. In this paper, we characterize the interaction between mismatch and the policy-gradient direction, distinguishing locally amplifying from contracting update contributions that mismatch magnitude alone cannot identify. In native NVFP4 runs, we observe an early imbalance between the two amplifying regions, favoring negative-advantage, negative-gap updates. Their tail tokens become concentrated in a small fraction of response segments before mismatch spreads globally. Motivated by these findings, we introduce **TRIAGE**, a direction-aware stabilization method that uses segment-level diagnosis to selectively rebalance policy-gradient updates and applies bounded repair to residual severe mismatch. TRIAGE modifies the optimization objective while retaining native NVFP4 weight-and-activation 4-bit (W4A4) forward execution on both the sampler and learner. Experiments on Qwen3-4B and Qwen3-30B-A3B show stable optimization throughout the evaluated training horizon and achieve full precision level performance across five mathematical reasoning benchmarks, while native NVFP4 with TRIAGE provides up to 2.3× higher rollout throughput than BF16. Our code is available at **https://anonymous.4open.science/r/TRIAGE**.

1 INTRODUCTION

Reinforcement learning (RL) has become central to improving the reasoning capabilities of large language models (LLMs) (DeepSeek-AI, 2025). As RL expands toward long-horizon reasoning, coding, and agentic tasks, increasingly long generation trajectories make rollout a dominant wall-clock bottleneck of the training loop (Qiu et al., 2026; Zhuge et al., 2026; Xie et al., 2025), making efficient rollout a primary systems challenge.

NVIDIA Blackwell GPUs provide native weight-and-activation 4-bit (W4A4) Tensor Core operations, offering up to $4\times$ the peak matrix-compute throughput of BF16 and making NVFP4 an attractive precision setting for long-sequence rollout (Alvarez et al., 2025; Li et al., 2025). However, trajectories are sampled by one execution policy and optimized using probabilities recomputed by another. Differences between the rollout and training execution paths already induce learner–sampler probability discrepancies, and low-precision execution can further enlarge this mismatch (Zhong et al., 2026; Gu et al., 2026). This creates a trade-off between hardware efficiency and stable policy optimization.

Existing approaches mitigate learner–sampler mismatch through two main strategies. Quantization-aware training (QAT) attempts to align the learner with the low-precision sampler (Chen et al., 2025), but standard QAT retains high-precision arithmetic and therefore does not preserve native W4A4 execution on the learner forward path. Importance-based corrections such as truncated importance sampling (TIS), clipping, and masking intervene according to the magnitude of learner–sampler disagreement (Qiu et al., 2026; Li et al., 2026). More fundamentally, magnitude alone describes how far the learner and sampler disagree, not whether the subsequent policy-gradient update will reduce or further amplify that disagreement. This raises a more essential question: *what determines whether learner–sampler mismatch contracts or amplifies under policy optimization?*

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.

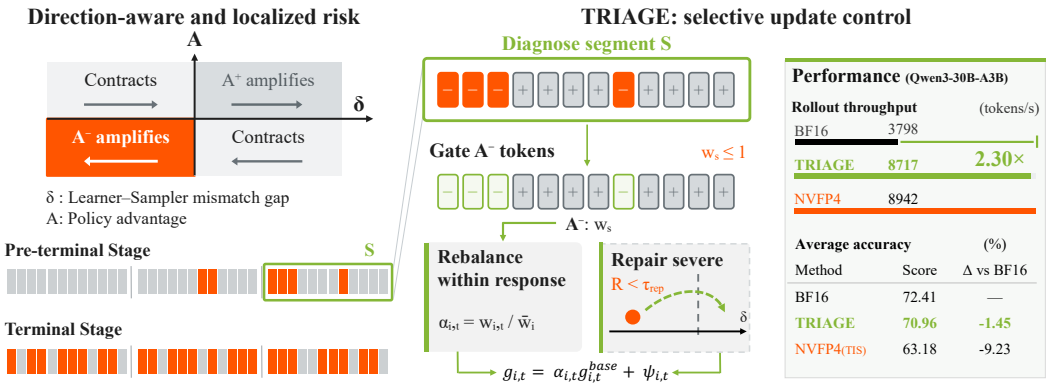

Figure 1: **Overview of TRIAGE for native NVFP4 RL.**

In this work, we show that learner–sampler mismatch must be considered together with the direction of the policy update (Figure 1, left). If the learner already assigns a token a lower probability than the sampler, further reducing that probability can widen the mismatch rather than correct it. In native NVFP4 runs, updates of this kind become increasingly prominent relative to those that increase probabilities already above the sampler's. This imbalance appears before mismatch becomes broadly severe. Tokens with larger discrepancies in this direction also become concentrated within short portions of responses, where they can be hidden by response-level averages. These observations motivate a different stabilization principle: *identify and control updates that tend to amplify mismatch, rather than intervening on mismatch magnitude alone*.

Motivated by these findings, we propose **TRIAGE**, a direction-aware stabilization framework for native low-precision RL. As illustrated in Figure 1, TRIAGE uses segment-level statistics to identify locally persistent risk, selectively attenuates only mismatch-amplifying $\mathcal{A}^-$ (Advantage < 0, mismatch gap < 0) tokens within risky segments, rebalances update mass, and applies a bounded repair term to residual severe mismatch. By acting on the optimization objective rather than replacing low-precision execution with fake-quantized computation, TRIAGE retains native NVFP4 W4A4 forward execution on both the sampler and learner. Across Qwen3-4B and training-sensitive Qwen3-30B-A3B MoE models, NVFP4 with TRIAGE achieves up to 2.3× the rollout throughput of BF16, while maintaining stable optimization throughout the training horizon and recovering comparable performance on mathematical reasoning benchmarks. Our contributions are summarized as follows:

❶ **A directional characterization of learner–sampler mismatch.** We characterize mismatch risk through its interaction with the policy-gradient direction and identify a directional, locally concentrated precursor to native NVFP4 instability.

❷ **TRIAGE, a selective stabilization mechanism for low-precision RL.** TRIAGE combines segment-level risk diagnosis, direction-selective token gating, response-level update rebalancing, and targeted repair to suppress self-amplifying mismatch while preserving contracting updates.

❸ **An end-to-end native NVFP4 RL recipe at scale.** Across experiments totaling more than 34,500 GPU-hours on B300 GPUs, we validate native W4A4 execution on both sampler and learner forward paths across dense and MoE models, achieving up to 1.3× end-to-end RL speedup while maintaining the BF16-level performance.

2 Preliminaries

2.1 Group-Relative Policy Optimization

Group-Relative Policy Optimization (GRPO) (Shao et al., 2024) is widely used in RL for reasoning models. Given a prompt x, GRPO samples a group of G responses $\{y_i\}_{i=1}^{G}$ from an old policy π_{old} and evaluates their rewards $\{r_i\}_{i=1}^{G}$. For outcome-based RL, the group-relative advantage is typically $A_i = (r_i - \mu_r)/(\sigma_r + \epsilon)$, where μ_r and σ_r are the mean and standard deviation of group rewards, and is assigned to all valid tokens of response i. Let $\rho_{i,t} = \pi_\theta(y_{i,t} \mid x, y_{i,<t})/\pi_{\text{old}}(y_{i,t} \mid x, y_{i,<t})$ denote the token-level importance ratio. Ignoring optional regularization terms, the clipped

policy objective can be written as

$$\mathcal{L}_{\mathrm{GRPO}} = -\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min(\rho_{i,t}A_i,\ \mathrm{clip}(\rho_{i,t}, 1-\varepsilon, 1+\varepsilon)A_i)\,, \tag{1}$$

Locally, a positive advantage increases the probability of sampled tokens, whereas a negative advantage decreases it; this directional property will be central to our analysis in Section 3 (Qiu et al., 2026; Zhuge et al., 2026).

2.2 Decoupled Rollout and Training

Modern LLM RL systems typically decouple rollout generation from gradient computation: a high-throughput inference engine such as SGLang or vLLM (Zheng et al., 2024; Kwon et al., 2023) acts as the *sampler*, while a training framework such as Megatron-LM acts as the *learner* (Qiu et al., 2026; Shoeybi et al., 2019). Let π_{s} and π_{l} denote the sampler and learner policies, respectively. For a sampled token y_t under the same prefix $y_{<t}$, we define their log-probability gap as

$$\delta_t = \log \pi_{\mathrm{l}}(y_t \mid x, y_{<t}) - \log \pi_{\mathrm{s}}(y_t \mid x, y_{<t}). \tag{2}$$

Ideally, synchronized policies should yield $\delta_t = 0$. In practice, the two execution paths can disagree even when they nominally represent the same model (Zhong et al., 2026).

This gap admits a useful interpretation in the language of off-policy correction. Because sampled tokens are drawn from the sampler, $y_{i,t} \sim \pi_{\mathrm{s}}(\cdot \mid x, y_{i,<t})$, the importance weight that exact policy evaluation would require is not $\rho_{i,t} = \pi_\theta/\pi_{\mathrm{old}}$ but $\pi_\theta/\pi_{\mathrm{s}}$, which factors as

$$\frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_{\mathrm{s}}(y_{i,t} \mid x, y_{i,<t})} = \underbrace{\frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_{\mathrm{l}}(y_{i,t} \mid x, y_{i,<t})}}_{\text{optimization drift}} \cdot \underbrace{\exp\big(\delta_{i,t}\big)}_{\text{learner–sampler ratio}}. \tag{3}$$

Under synchronous weight transfer with one update per rollout, the first factor is one at the start of each iteration, and the entire off-policy error of the update is carried by $\exp(\delta_{i,t})$. Truncated importance sampling, clipping, and masking (Qiu et al., 2026; Li et al., 2026) all intervene on exactly this factor. Crucially, $\exp(\delta_{i,t})$ is a *magnitude-only* statistic: it records how far the two policies disagree, but not how the ensuing gradient will change that disagreement. This gap between what existing corrections measure and what optimization stability requires motivates our directional analysis in Section 3.

2.3 Low-Precision Rollout with NVFP4

Reduced-precision rollout lowers both matrix-computation cost and model-memory traffic, making it particularly attractive for long-sequence RL. However, quantization introduces a forward perturbation that is qualitatively different from ordinary policy staleness (Li et al., 2025). Even under an identical prefix, quantized weights and activations perturb hidden states and output logits, producing a local probability discrepancy between the sampler and learner. During decoding, such local perturbations can additionally alter token sampling and hence the subsequent prefix, allowing quantization error to accumulate into trajectory-level divergence (Zhuge et al., 2026; Gu et al., 2026).

NVFP4 is a Blackwell-supported format combining E2M1 FP4 values, an E4M3 FP8 microscale per 16 values, and an FP32 second-level scale (NVIDIA et al., 2025). For an element x_j in a 16-element block b, its quantized representation can be abstracted as

$$\hat{x}_j = S\, s_b\, Q_{\mathrm{E2M1}}\left(\frac{x_j}{S\, s_b}\right), \qquad j \in b, \tag{4}$$

where s_b is an E4M3 FP8 microscale shared by 16 consecutive values, S is an FP32 second-level scale shared across the tensor, and $Q_{\mathrm{E2M1}}(\cdot)$ denotes rounding to the FP4 E2M1 grid. The fine-grained E4M3 microscale improves local range utilization, while the tensor-level FP32 scale extends the range representable by the collection of FP8 block scales.

3 Understanding Learner–Sampler Policy Mismatch

We now study how the learner–sampler mismatch introduced in Section 2 evolves under policy optimization. Our analysis reveals a staged failure process. Section 3.1 derives which mismatches are locally amplified by the policy-gradient update. Section 3.2 then identifies when this directional

risk becomes visible along training. Finally, Section 3.3 shows where the risk accumulates within long responses and why response-level averages fail to expose it.

3.1 Directional Dynamics of Policy Mismatch

Which mismatches are self-amplifying under an RL policy update? Let $z_{i,t}(\theta) = \log \pi_\theta(y_{i,t} \mid x, y_{i,<t})$ denote the learner log-probability of rollout token (i,t), $\delta_{i,t}$ is log-probability gap from equation 2, and $g_{i,t}$ the signed coefficient applied to its policy-gradient term. A generic learner update with step size $\eta > 0$ is

$$\theta^+ = \theta + \eta \sum_{i,t} g_{i,t} \nabla_\theta z_{i,t}(\theta), \tag{5}$$

where for vanilla GRPO at the start of a single-update iteration $g_{i,t} = A_i$; importance correction and stabilization mechanisms modify this coefficient. To track whether the mismatch grows or contracts, define the mismatch energy $D = \frac{1}{2} \sum_{i,t} \delta_{i,t}^2$. Holding the rollout policy fixed during one update, a first-order expansion gives

$$\Delta D = \eta\, \boldsymbol{\delta}^\top K \mathbf{g} + \mathcal{O}(\eta^2) = \eta \sum_{i,t} \kappa_{i,t} \delta_{i,t} g_{i,t} + \eta R_{\text{cross}} + \mathcal{O}(\eta^2), \tag{6}$$

where K is the Gram matrix of token log-probability gradients, $\kappa_{i,t} = \|\nabla_\theta z_{i,t}\|_2^2$, and R_{cross} collects cross-token interactions. Thus $\Delta D > 0$ means the current update batch inflates the total mismatch energy. Appendix B derives this expansion for a general preconditioned update with explicit optimizer residuals. Equation 6 is the identity-preconditioner special case.

Writing the amplifying and contracting diagonal masses as

$$B_{\text{amp}} = \sum_{i,t} \kappa_{i,t} [\delta_{i,t} g_{i,t}]_+, \qquad B_{\text{con}} = \sum_{i,t} \kappa_{i,t} [-\delta_{i,t} g_{i,t}]_+ \tag{7}$$

where $[x]_+ = \max(x, 0)$, it becomes

$$\Delta D = \eta\, (B_{\text{amp}} - B_{\text{con}} + R_{\text{cross}}) + \mathcal{O}(\eta^2). \tag{8}$$

Thus, within the diagonal current-gradient contribution, $\delta_{i,t} g_{i,t} > 0$ amplifies the existing mismatch, whereas $\delta_{i,t} g_{i,t} < 0$ contracts it. In particular, for vanilla GRPO, the criterion reduces to the sign interaction $A_i \delta_{i,t}$. The two mismatch-amplifying regions are

$$\mathcal{A}^- = \{(i,t) : A_i < 0,\ \delta_{i,t} < 0\}, \qquad \mathcal{A}^+ = \{(i,t) : A_i > 0,\ \delta_{i,t} > 0\}. \tag{9}$$

The remaining two sign combinations are locally mismatch-contracting. Figure 1, left summarizes this directional geometry.

This distinction explains why mismatch magnitude alone is insufficient: **a large mismatch may already be self-correcting, while a moderate one grows systematically when the gradient acts along it.** Cross-token coupling, optimizer-history terms, and higher-order effects can still alter the realized motion of an individual token, so we use $\delta_{i,t} g_{i,t}$ as a local directional risk indicator rather than an unconditional predictor of the realized sign of $\delta_{i,t} \Delta \delta_{i,t}$. Appendix C.3 audits the scope of this interpretation. Our main use of the decomposition is therefore population-level: it predicts that instability may first appear as an imbalance between the two amplifying regions, even before the marginal gap distribution becomes broadly severe.

3.2 Directional Asymmetry Precedes Negative-Tail Explosion

When does the imbalance become visible? We characterize native-NVFP4 GRPO trajectories of Qwen3-4B-Base and Qwen3-30B-A3B-Base using four 25-update trailing windows placed at 25%, 50%, 75%, and 100% of each run's characterization horizon. We refer to these windows descriptively as *healthy*, *drift*, *pre-terminal*, and *terminal*, respectively.

To measure the distribution of tokens between the two amplifying regions in equation 9, define the advantage-weighted token occupancy

$$w_A(\mathcal{R}) = \frac{\sum_{(i,t) \in \mathcal{R}} |A_i|}{\sum_{i,t} |A_i|}, \quad \rho_{\text{asym}} = \frac{w_A(\mathcal{A}^-)}{w_A(\mathcal{A}^+)}. \tag{10}$$

Figure 2 reveals a separation between directional drift and magnitude collapse. Across the drift-to-pre-terminal interval, the marginal gap distribution still appears largely benign: the fraction of tokens satisfying $|\delta| < 0.05$ remains at least 83.0% on 4B and 91.9% on 30B. Nevertheless, ρ_{asym} rises from 1.02 to 2.32 by pre-terminal on 4B, and from 1.34 to 1.85 on 30B. The two theoretically symmetric amplifying regions therefore become empirically imbalanced well before a broad magnitude increase is visible.

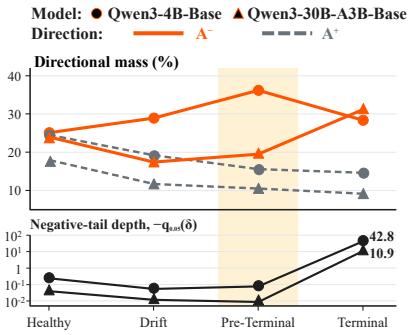

Figure 2: Directional asymmetry precedes terminal negative-tail expansion

In the terminal stage, the lower tail expands abruptly. Within the respective mismatch estimands, the 5% gap quantile $q_{0.05}(\delta)$ reaches -42.75 on 4B and -10.94 on 30B. **Directional asymmetry therefore identifies an earlier warning interval, whereas the negative-tail explosion characterizes the terminal regime.**

3.3 Localized Accumulation before Global Failure

Where does the amplifying mismatch accumulate? For each negative-advantage response, we partition it into non-overlapping segments S of $W = 64$ tokens. We call tokens satisfying $A_i < 0$ and $\delta_{i,t} < \tau_{\text{ana}}$ *amplifying-tail tokens*, where the model-specific threshold $\tau_{\text{ana}}^{(m)}$ is fixed as the fifth percentile of δ in the corresponding healthy window. Their local concentration is measured by

$$f_S^{\text{amp}}(\tau_{\text{ana}}) = \frac{1}{|S|} \sum_{t \in S} \mathbf{1}[A_i < 0,\ \delta_{i,t} < \tau_{\text{ana}}], \quad (11)$$

we call a segment *tail-heavy* when $f_S^{\text{amp}} > 0.1$.

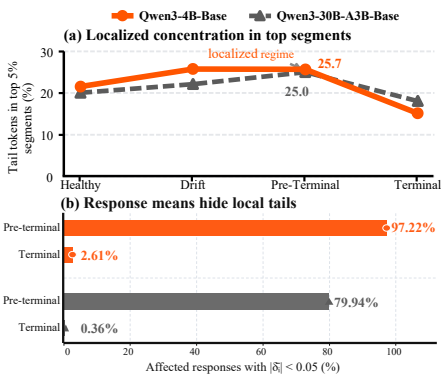

Figure 3: **Localized mismatch is hidden by response means before terminal.** (a) Share of amplifying-tail tokens contained in the top 5% of segments ranked by f_S^{amp}. (b) Among negative-advantage responses containing at least one tail-heavy segment, the fraction with $|\bar{\delta}_i| < 0.05$

From the healthy to the pre-terminal window, their global frequency decreases from 4.8–4.9% to 2.8–3.6% across the two models. Nevertheless, the top 5% of segments contain approximately one quarter of all amplifying-tail tokens, about five times the mass implied by uniform placement, while adjacent tail-heavy persistence increases from approximately 2.6× to 3.8–4.0× the chance rate. The precursor is therefore not an expanding global tail, but a reorganization of a small tail population into a few persistent local regions.

Response averages obscure this structure: at pre-terminal, 97.2% (4B) and 79.9% (30B) of responses containing tail-heavy segments still satisfy $|\bar{\delta}_i| < 0.05$, where $\bar{\delta}_i = n_i^{-1} \sum_{t \in \mathcal{T}_i} \delta_{i,t}$. By terminal, roughly half of the eligible segments are tail-heavy and top-segment concentration declines.

These observations motivate segment-level diagnosis with intervention restricted to $\mathcal{A}^-$ tokens, rather than suppressing every token in an affected response or segment.

4 Method

Section 3 shows that the risk of learner–sampler mismatch depends on its interaction with the policy-update direction: updates with $\delta_{i,t} g_{i,t} > 0$ amplify the existing gap, whereas updates with $\delta_{i,t} g_{i,t} < 0$ contract it. TRIAGE acts on this distinction through two complementary modifications of the effective update coefficient. Let $g_{i,t}^{\text{base}}$ denote the per-token coefficient of the underlying importance-corrected objective, including its reduction factors. We then write

$$g_{i,t} = \alpha_{i,t}\, g_{i,t}^{\text{base}} + \psi_{i,t}, \quad (12)$$

The two terms partition by advantage sign: $\alpha_{i,t}$ reallocates policy-gradient mass within negative-advantage responses (Section 4.1), while $\psi_{i,t}$ adds bounded corrective pressure on positive-advantage responses (Section 4.2).

4.1 Segment Diagnosis with Direction-Selective Gating

Following Section 3.3, TRIAGE therefore uses short contiguous segments as an intermediate diagnostic scale, a granularity that has also been found useful for avoiding the extremes of token- and response-level importance estimation (Yang et al., 2026).

For a diagnostic segment S of response i, we compute

$$\bar{\delta}_S = \frac{\sum_{t\in S} m_{i,t}\delta_{i,t}}{\sum_{t\in S} m_{i,t}}, \qquad f_S^{\text{tail}} = \frac{\sum_{t\in S} m_{i,t}\mathbf{1}[\delta_{i,t} < \tau_{\text{tail}}]}{\sum_{t\in S} m_{i,t}}, \tag{13}$$

which respectively capture persistent displacement and sparse deep-tail mismatch. We map these statistics to a segment gate weight

$$w_S = \mathcal{G}\left(\bar{\delta}_S, f_S^{\text{tail}}\right) \in [w_{\min}, 1], \tag{14}$$

where a larger negative displacement or a heavier negative tail yields stronger attenuation. The exact operating rule, including eligibility conditions and the fixed operating thresholds, is deferred to Appendix D.

Crucially, a risky segment does not imply that all of its tokens should be attenuated. For a negative-advantage response, tokens with $A_i < 0,\ \delta_{i,t} < 0$ contribute to mismatch amplification, whereas tokens with $A_i < 0,\ \delta_{i,t} > 0$ naturally contract the discrepancy. TRIAGE therefore applies

$$w_{i,t} = \begin{cases} w_S, & A_i < 0,\ \delta_{i,t} < 0,\ t \in S, \\ 1, & \text{otherwise.} \end{cases} \tag{15}$$

Thus, segment statistics determine *whether* local intervention is warranted, while the directional criterion from Section 3 determines *which* token updates are modified. In particular, naturally contracting tokens that coexist in the same risky segment are not directly gated. We do not gate the positive-advantage amplifying region $\{A_i > 0,\ \delta_{i,t} > 0\}$: the directional imbalance identified in Section 3 concentrates in the negative-advantage region, and attenuating probability increases on reward-positive responses would directly suppress the learning signal.

The weighted policy loss is

$$\mathcal{L}_{\text{PG},i}^{\text{TRIAGE}} = \frac{\sum_t m_{i,t} w_{i,t} \ell_{i,t}^{\text{base}}}{\sum_t m_{i,t} w_{i,t}}, \tag{16}$$

where $\ell_{i,t}^{\text{base}}$ denotes the existing importance-corrected token loss. Relative to the original response mean, this induces

$$\alpha_{i,t} = \frac{w_{i,t}}{\bar{w}_i}, \qquad \bar{w}_i = \frac{\sum_t m_{i,t} w_{i,t}}{\sum_t m_{i,t}}. \tag{17}$$

The normalization therefore reallocates relative update mass away from the targeted amplifying subset rather than uniformly weakening the entire negative-advantage response, directly modifying the balance between B_{amp} and B_{con} identified in Section 3.

4.2 Repair on Positive-Advantage Responses

Direction-selective gating controls the allocation of existing policy gradients, but it is multiplicative: it can attenuate an amplifying update toward zero, yet cannot reverse its sign or directly reduce the remaining learner–sampler gap. We therefore add a repair mechanism on positive-advantage responses. On these responses, the policy gradient on negative-gap tokens is already mismatch-contracting, so raising sampled-token log-probabilities reinforces a motion that agrees with reward-driven learning; applying the same correction to negative-advantage responses would instead oppose their policy gradients, which is why that side is handled by gating alone.

Let R denote a repair segment of response $i(R)$, with $n_R = \sum_{t\in R} m_{i(R),t}$ valid tokens and mean gap $\bar{\delta}_R$. We define

$$q_R = [\tau_{\text{rep}} - \bar{\delta}_R]_+, \qquad \phi_\beta(q) = \beta^2\left[\sqrt{1 + (q/\beta)^2} - 1\right], \tag{18}$$

where q_R is the one-sided violation of the repair boundary and ϕ_β is a pseudo-Huber penalty.

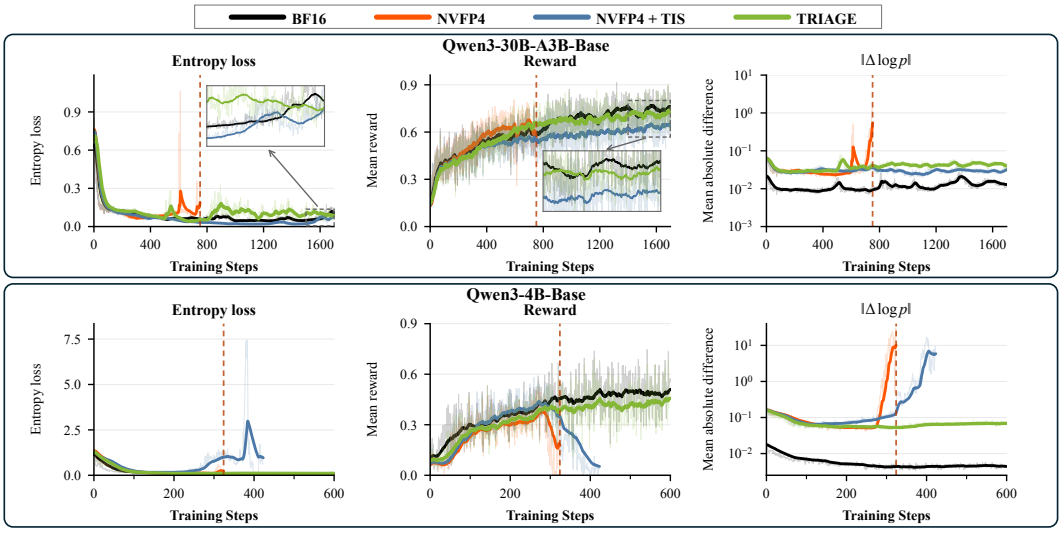

Figure 4: Training dynamics on Qwen3-30B-A3B-Base and Qwen3-4B-Base. From left to right: entropy loss, training reward, and mean absolute train–rollout log-probability gap against optimizer steps.

Gating is active throughout training, whereas repair is enabled from a fixed update k_{rep} onward, with $\lambda_k = \lambda_{\text{rep}} \mathbf{1}[k \geq k_{\text{rep}}]$. All operating thresholds and the repair schedule are fixed before each run. At update k, the objective is

$$\mathcal{L}_k = \mathcal{L}_{\text{PG}}^{\text{TRIAGE}} + \frac{\lambda_k}{N_{\text{rep}}} \sum_{R \in \mathcal{S}_{\text{rep}}^{\text{valid}}} \mathbf{1}[A_{i(R)} > 0]\, \phi_\beta(q_R), \tag{19}$$

Under the ascent-form notation of Equation 12, one gradient-descent step on the repair objective contributes the additive coefficient

$$\psi_{i,t} = \frac{\lambda_k}{N_{\text{rep}}} \frac{m_{i,t}}{n_R} \phi'_\beta(q_R)\, \mathbf{1}[A_i > 0,\ q_R > 0]. \tag{20}$$

All valid tokens within an active segment share the same corrective coefficient. For negative-gap tokens, $\delta_{i,t}\psi_{i,t} < 0$, so the repair term contributes to contraction in the diagonal decomposition of Section 3. Positive-gap tokens may also receive the coefficient: the objective repairs negative segment-level displacement rather than enforcing contraction for every token. Appendix D.3 derives the positive-advantage repair coefficient and its bound.

5 Experiments

5.1 Experimental Setup

Models and task. We use Qwen3-30B-A3B-Base as the primary model and Qwen3-4B-Base for replication and ablation studies. We perform RL with GRPO on the DAPO dataset. To better understand the quantization error, we use 20,480 as the max response length. Detailed training settings are listed in Appendix E.

Settings. We compare four training configurations: (i) **BF16**, the baseline; (ii) **NVFP4**, native W4A4 execution with Transformer Engine; (iii) **NVFP4+TIS**, adding token-level importance correction (Qiu et al., 2026); and (iv) **NVFP4+TRIAGE**, our full method. All NVFP4 configurations share the same native execution path: W4A4 kernels with per-token second-level activation scaling (Research et al., 2026) on both the sampler and the learner forward pass. TIS uses a truncation threshold of $C = 2.0$. TRIAGE operating rules and segment definitions are given in Appendix D; quantization and system settings are described in Appendices E and F.

Evaluation Benchmarks and Metrics. We evaluate mathematical reasoning on several widely used benchmarks, including GSM8K (Cobbe et al., 2021), MATH-500 (Lightman et al., 2023; Hendrycks et al., 2021), AIME 2024/2025, and AMC 2023 (MAA). We also track training reward, policy entropy, and the mean absolute train–rollout log-probability gap throughout training.

Table 1: Performance across five benchmarks. Step denotes the evaluated training checkpoint. Stable runs are evaluated at the final training step, while runs marked with † collapse before the planned training budget and are evaluated at the last saved checkpoint before collapse. Best result among NVFP4 settings for each model is shown in **bold**.

Model	Setting	Step	AIME24	AIME25	MATH500	AMC23	GSM8K	**Avg**
30B	BF16	1700	53.89%	36.67%	92.65%	88.04%	90.81%	**72.41%**
	NVFP4†	700	44.44%	30.00%	88.80%	77.17%	88.95%	**65.87%**
	NVFP4+TIS	1700	36.11%	27.22%	88.30%	69.02%	95.26%	**63.18%**
	NVFP4+TRIAGE	1700	51.67%	**38.33%**	91.60%	77.72%	**95.47%**	**70.96%**
4B	BF16	600	24.44%	23.33%	84.1%	67.40%	92.04%	**58.26%**
	NVFP4†	300	8.33%	8.33%	78.4%	50.00%	90.45%	**47.10%**
	NVFP4+TIS†	300	16.11%	18.33%	80.90%	53.75%	84.72%	**50.76%**
	NVFP4+TRIAGE	300	19.44%	17.22%	82.00%	56.52%	89.99%	**53.03%**
	NVFP4+TRIAGE	600	**27.22%**	20.00%	**85.80%**	67.39%	**92.04%**	**58.49%**

Table 2: Natural-EOS generation from the base checkpoint. Values are means over ten post-warm-up iterations at two global batch sizes. Initialization, evaluation, and checkpoint saving are excluded. Parentheses report speedups over BF16.

	End-to-end cost (s/iter) ↓		Rollout throughput (token/s) ↑	
Setting	Batch 256	Batch 512	Batch 256	Batch 512
BF16	208.2 (1.00×)	254.5 (1.00×)	2977 (1.00×)	3798 (1.00×)
NVFP4	169.4 (1.23×)	191.0 (1.33×)	6201 (2.08×)	8942 (2.35×)
NVFP4+TRIAGE	172.1 (1.21×)	195.8 (1.30×)	6745 (2.27×)	8717 (2.30×)

5.2 Experimental Results

Training stability. Figure 4 shows that naive NVFP4 becomes unstable after approximately 300 steps on Qwen3-4B-Base, with a sharp reward drop and rapid growth in the mean absolute train–rollout log-probability gap. TIS delays this failure but does not prevent it. On Qwen3-30B-A3B-Base, naive NVFP4 also exhibits rapid mismatch growth after roughly 700 steps. TIS does not exhibit a comparable divergence over 1,700 steps, but its reward trajectory remains below BF16 and TRIAGE in later training. In contrast, TRIAGE remains stable throughout the evaluated 600- and 1,700-step schedules, approximately doubling the observed stable training horizon on 4B relative to naive NVFP4.

Benchmark performance. Table 1 shows that TRIAGE achieves near-BF16 average performance: 58.49% versus 58.26% on 4B, and 70.96% versus 72.41% on 30B. At the shared step-300 checkpoint on 4B, TRIAGE already outperforms naive NVFP4 and NVFP4+TIS by 5.93 and 2.27 percentage points, respectively, showing that its gains are not solely due to longer training. Although NVFP4+TIS does not collapse over the full 1700-step training horizon, it still scores below the pre-collapse naive NVFP4 checkpoint at step 700 (63.18% versus 65.87%).

These observations are consistent with the directional and localized mismatch analysis in Section 3: the global mean does not capture how mismatch interacts with policy updates or where it concentrates within responses, supporting the importance of selective control of mismatch amplification.

5.3 Rollout Throughput and Learner Overhead

Rollout throughput. With TRIAGE, native NVFP4 achieves 6,745 and 8,717 output tokens/s at global batch sizes 256 and 512, respectively, corresponding to 2.27× and 2.30× the BF16 throughput. TRIAGE thus retains the more-than-twofold rollout throughput advantage observed for native NVFP4 in these runs.

End-to-end RL efficiency. Iteration timing includes rollout, learner updates, weight conversion and synchronization, and orchestration. TRIAGE reduces average iteration time from 208.2 to 172.1 s at batch size 256 and from 254.5 to 195.8 s at batch size 512, yielding 1.21× and 1.30× end-to-end speedups, respectively.

TRIAGE overhead. We separately isolate TRIAGE's incremental learner cost on eight B300 GPUs using five paired repetitions of the same frozen batch. Enabling gating and repair yields an observed median learner-update overhead of 0.84% and a median absolute increase of 0.225 s.

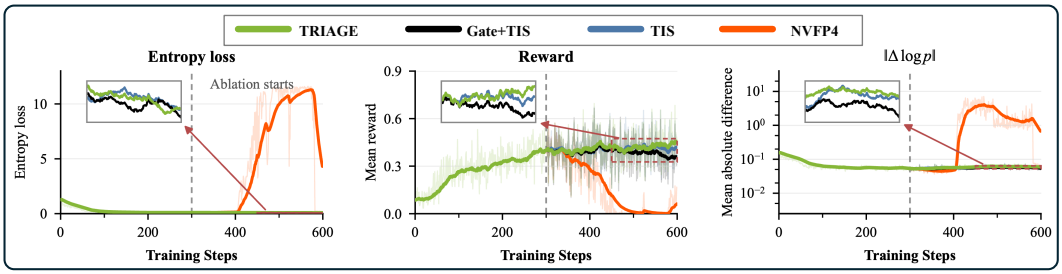

Figure 5: Ablation on Qwen3-4B-Base. **Full TRIAGE**; **Gate+TIS** removes repair; **TIS** additionally removes gating; **NVFP4** removes all three components. The dashed line marks the branch point.

Detailed throughput settings, environment settings, independent configuration-selection results, stress-tests results, and complete timing distributions are provided in Appendix F.

5.4 Ablation Studies

We perform component ablations on Qwen3-4B-Base. All variants branch from the same full-TRIAGE checkpoint at step 300. Full TRIAGE achieves the highest reward. Removing repair keeps the mean log-probability gap low but leaves the advantage-driven update imbalance uncorrected, reducing the reward. TIS alone remains stable when continued from the TRIAGE checkpoint, unlike its failure when used from the beginning in Figure 4. This contrast reflects the stage-dependent mismatch dynamics identified in Section 3. Naive NVFP4 also allows mismatch to grow sharply, leading to a late-stage collapse.

6 Related Work

Low-precision training. Native low-precision training requires careful control of quantization error. NVIDIA's NVFP4 pretraining recipe (NVIDIA et al., 2025; Xie et al., 2025) and Quartet II (Panferov et al., 2026) address this through quantization design and improved gradient estimation. For RL, QeRL (Huang et al., 2025) combines NVFP4 with LoRA and adaptive quantization noise. FP8-RL (Qiu et al., 2026) stabilizes quantized rollout with importance-based corrections. We focus on the optimization dynamics of the residual learner–sampler mismatch under native NVFP4 W4A4 execution.

Training–inference alignment. Implementation differences can destabilize RL even with synchronized model weights (Zhong et al., 2026). QaRL (Gu et al., 2026) aligns the learner with quantized rollout through low-bit forward computation and introduces sequence-level dual clipping for negative samples. QUADS (Zhuge et al., 2026) combines weight-only fake quantization on the learner with rollout-side activation compensation. TRIAGE instead controls how residual mismatch evolves under policy updates, modifying the optimization objective while retaining native W4A4 forward execution on both the learner and sampler.

Directional control. ACRL (Fan et al., 2026) uses advantage-guided, four-quadrant probability adjustment, with response-level discrepancy relative to a reference controlling token-wise reweighting. TRIAGE derives a local mismatch-energy decomposition and identifies directional asymmetry and localized negative tails in native NVFP4 runs.

7 Conclusion

We presented TRIAGE, a direction-aware stabilization method for native NVFP4 RL. Our analysis distinguishes locally amplifying and contracting policy-gradient contributions and identifies directional imbalance and segment-localized negative tails before global mismatch growth. TRIAGE translates these observations into selective gating, within-response rebalancing, and bounded repair, while preserving native W4A4 forward execution on both learner and sampler. On Qwen3-4B-Base and Qwen3-30B-A3B-Base, TRIAGE sustains stable optimization over the evaluated horizons and achieves near-BF16 average performance across five mathematical reasoning benchmarks. In our measured configurations, native NVFP4 with TRIAGE achieves up to 2.30× rollout throughput and 1.30× end-to-end speedups over BF16. These results support controlling how mismatch evolves under policy updates, rather than uniformly minimizing its magnitude.

AI USE STATEMENT

We used large language models to assist with manuscript drafting and editing, discussion of experimental design, interpretation of results, and figure design. The authors reviewed and revised the AI-assisted content and checked the interpretations against the experimental results. The authors take full responsibility for the final content, including all methods, results, and conclusions.

REPRODUCIBILITY STATEMENT

We describe TRIAGE in Section 4. Model configurations, training hyperparameters, evaluation protocols, and hardware and software settings are detailed in Appendix C.1. Checkpoint selection and efficiency measurement procedures are specified in Tables 1 and 2, respectively. We release the complete implementation together with reproduction scripts. An anonymized implementation and reproduction scripts are provided with the submission.

REFERENCES

Eduardo Alvarez, Omri Almog, Eric Chung, Simon Layton, Dusan Stosic, Ronny Krashinsky, and Kyle Aubrey. Introducing NVFP4 for efficient and accurate low-precision inference. NVIDIA Technical Blog, June 2025. Published June 24, 2025.

Mengzhao Chen, Chaoyi Zhang, Jing Liu, Yutao Zeng, Zeyue Xue, Zhiheng Liu, Yunshui Li, Jin Ma, Jie Huang, Xun Zhou, et al. Scaling law for quantization-aware training. *arXiv preprint arXiv:2505.14302*, 2025.

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.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Wenwu Fan, Qihong Lin, Zhijie Xia, Zhuo Zheng, Sihao Wang, Qiang Chen, and Liangsheng Zhu. ACRL: Adaptive Control of Training-Inference Discrepancy for Stable Reinforcement Learning. *arXiv preprint arXiv:2607.24062*, 2026. doi: 10.48550/arXiv.2607.24062. URL https://arxiv.org/abs/2607.24062.

Hao Gu, Hao Wang, Jiacheng Liu, Lujun Li, Qiyuan Zhu, Bei Liu, Binxing Xu, Lei Wang, Xintong Yang, Sida Lin, Sirui Han, and Yike Guo. QaRL: Rollout-aligned quantization-aware rl for fast and stable training under training–inference mismatch. *arXiv preprint arXiv:2604.07853*, 2026.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

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. *arXiv preprint arXiv:2510.11696*, 2025. doi: 10.48550/arXiv.2510.11696. URL https://arxiv.org/abs/2510.11696.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Yuhang Li, Reena Elangovan, Xin Dong, Priyadarshini Panda, and Brucek Khailany. QuRL: Efficient reinforcement learning with quantized rollout. *arXiv preprint arXiv:2602.13953*, 2026.

Zhen Li, Yupeng Su, Runming Yang, Congkai Xie, Zheng Wang, Zhongwei Xie, Ngai Wong, and Hongxia Yang. Quantization meets reasoning: Exploring llm low-bit quantization degradation for mathematical reasoning. *arXiv preprint arXiv:2501.03035*, 2025.

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.

MAA. American mathematics competitions. `https://maa.org/student-programs/amc/`. Accessed: 2026-09-21.

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, 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. *arXiv preprint arXiv:2509.25149*, 2025. doi: 10.48550/arXiv.2509.25149. URL `https://arxiv.org/abs/2509.25149v1`.

Andrei Panferov, Erik Schultheis, Soroush Tabesh, and Dan Alistarh. Quartet II: Accurate LLM Pre-Training in NVFP4 by Improved Unbiased Gradient Estimation. *arXiv preprint arXiv:2601.22813*, 2026. doi: 10.48550/arXiv.2601.22813. URL `https://arxiv.org/abs/2601.22813`.

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.

Cursor Research, Aaron Chan, Ahmed Shalaby, Alexander Wettig, Aman Sanger, Andrew Zhai, Anurag Ajay, Ashvin Nair, Charlie Snell, Chen Lu, et al. Composer 2 technical report. *arXiv preprint arXiv:2603.24477*, 2026.

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.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Congkai Xie, Shuo Cai, Wenjun Wang, Pengxiang Li, Zhijie Sang, Kejing Yang, Yiming Zhang, Zhen Li, Guanghao Zhu, Zeyu Liu, et al. Infir: Crafting effective small language models and multimodal small language models in reasoning. *arXiv preprint arXiv:2502.11573*, 2025.

Kun Yang, Zikang Chen, Yanmeng Wang, Zhigen Li, Ning Cheng, Shaojun Wang, and Jing Xiao. SSPO: Subsentence-level policy optimization. *arXiv preprint arXiv:2511.04256*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

Tianle Zhong, Neiwen Ling, Yifan Pi, Zijun Wei, Tianshu Yu, Geoffrey Fox, Peng Wu, and Xiao Yu. Diagnosing Training Inference Mismatch in LLM Reinforcement Learning. *arXiv preprint arXiv:2605.14220*, 2026. doi: 10.48550/arXiv.2605.14220. URL https://arxiv.org/abs/2605.14220.

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. *arXiv preprint arXiv:2607.15810*, 2026.

A Limitations and Future Work

Our study focuses on establishing the importance of *direction-aware* stabilization for low-precision RL. While truncated importance sampling (TIS) can stabilize the absolute mismatch gap, our 30B experiments show that completing the training trajectory does not necessarily translate into strong downstream performance. In contrast, TRIAGE explicitly accounts for whether learner–sampler mismatch is locally corrected or amplified by the policy update. Nevertheless, our empirical study remains limited in scale. Due to computational constraints, we do not evaluate models at the 100B+ scale or substantially longer training horizons. Our main runs are terminated after the optimization trajectories exhibit clear convergence or stable late-stage behavior. Evaluating whether the same directional dynamics persist over substantially longer training and on larger sparse models is therefore an important direction for future work.

Our current experiments also focus on synchronous RL. The directional criterion underlying TRIAGE is not inherently tied to synchronous sampling, and may be particularly relevant to asynchronous RL, where policy staleness introduces an additional source of learner–sampler mismatch. However, we have not experimentally validated TRIAGE in this setting. Extending the analysis to jointly characterize quantization mismatch and asynchronous policy staleness is an interesting direction for future study.

Finally, we believe that the end-to-end efficiency of native NVFP4 RL remains below the acceleration potential of Blackwell and Vera Rubin Tensor Core computation. Although NVFP4 substantially accelerates rollout generation in our experiments, the overall RL iteration speedup is approximately $1.3\times$. A growing fraction of runtime is instead spent on operations outside the accelerated GEMMs, including parameter synchronization between the trainer and sampler and quantization-related reductions and data transformations such as Amax computation. Future systems could therefore obtain larger gains by co-designing TRIAGE with more tightly integrated low-precision kernels and RL frameworks, including more efficient parameter synchronization and fused quantization paths. On the algorithmic side, our directional analysis also suggests a broader design space beyond the current repair rule: rather than treating all mismatch-amplifying tokens uniformly, future work could develop finer-grained repair mechanisms that adapt to the magnitude, locality, and persistence of directional mismatch.

B Derivation of Learner–Sampler Mismatch Dynamics

This appendix derives the local mismatch dynamics in Section 3.1, including a positive-semidefinite preconditioner and explicit optimizer residuals.

B.1 Fixed-Sampler Setup and Update Coefficients

Let $p \equiv (i,t)$ index a valid rollout token, with sampled token a_p and prefix s_p corresponding to $y_{i,t}$ and $(x, y_{i,<t})$ in the main text. Define

$$z_p(\theta) = \log \pi_\theta(a_p \mid s_p), \qquad b_p = \log \pi_{\mathrm{s}}(a_p \mid s_p), \qquad \delta_p(\theta) = z_p(\theta) - b_p. \tag{21}$$

During one learner update, the trajectories, prefixes, masks, advantages, and recorded sampler log-probabilities are fixed. Hence $\nabla_\theta \delta_p = \nabla_\theta z_p$. The Taylor expansion below assumes locally bounded first and second derivatives of the log-probability map. All gradients are evaluated at the pre-update parameters θ_k.

Write the current policy-gradient signal as

$$u_k = \sum_p g_p \nabla_\theta z_p. \tag{22}$$

Here g_p includes the loss-reduction factors. The main-text directional discussion suppresses these positive factors when writing $g_{i,t} = A_i$; we retain them here to specify coefficient magnitudes. For the vanilla GRPO response mean at the start of a single-update iteration, where the learner/old-learner ratio is one and clipping is inactive,

$$g_{i,t}^{\mathrm{eff}} = \gamma_k \frac{A_i}{n_i}, \qquad n_i = \sum_t m_{i,t} > 0, \quad m_{i,t} \in \{0,1\}, \tag{23}$$

on valid tokens, with zero coefficient on masked tokens. Here $\gamma_k > 0$ is the common outer averaging factor for the update; for the single-prompt group objective in Section 2, $\gamma_k = 1/G$. Thus $\mathrm{sign}(g_{i,t}^{\mathrm{eff}}) = \mathrm{sign}(A_i)$ for valid tokens. A positive batch-wide reduction constant preserves this sign, whereas the response-dependent factor $1/n_i$ must be retained when comparing coefficient magnitudes. Importance weighting, clipping, masking, and TRIAGE are represented by their actual effective coefficients g_p; a token whose coefficient is zero has no diagonal update contribution.

B.2 Preconditioned Gap and Energy Dynamics

Express the local optimizer update as

$$\Delta\theta_k = \theta_{k+1} - \theta_k = \eta \mathcal{P}_k u_k + r_k^{\mathrm{opt}}, \qquad \eta > 0, \quad \mathcal{P}_k \succeq 0, \tag{24}$$

where $\mathcal{P}_k$ is symmetric. The residual r_k^{opt} contains components outside the preconditioned current-gradient signal, such as history-dependent momentum and decoupled weight decay. This decomposition accommodates the local current-gradient component of AdamW-style updates while keeping those residual contributions explicit.

Let J have rows $\nabla_\theta z_p^\top$. Define

$$K^{(\mathcal{P})} = J\mathcal{P}_k J^\top, \qquad K_{p,q}^{(\mathcal{P})} = \nabla_\theta z_p^\top \mathcal{P}_k \nabla_\theta z_q, \qquad \mathbf{r}_k^z = J r_k^{\mathrm{opt}}. \tag{25}$$

The matrix $K^{(\mathcal{P})}$ is positive semidefinite, although its off-diagonal entries can have either sign. Applying Taylor expansion with the sampler fixed gives

$$\Delta\boldsymbol{\delta} = J\Delta\theta_k + \mathcal{O}(\|\Delta\theta_k\|_2^2) = \eta K^{(\mathcal{P})}\mathbf{g} + \mathbf{r}_k^z + \mathcal{O}(\|\Delta\theta_k\|_2^2). \tag{26}$$

For $D(\theta) = \frac{1}{2}\|\boldsymbol{\delta}(\theta)\|_2^2$, the exact energy difference is

$$\Delta D = \boldsymbol{\delta}^\top \Delta\boldsymbol{\delta} + \frac{1}{2}\|\Delta\boldsymbol{\delta}\|_2^2. \tag{27}$$

Since $\Delta\boldsymbol{\delta} = \mathcal{O}(\|\Delta\theta_k\|_2)$, substitution yields

$$\Delta D = \eta \boldsymbol{\delta}^\top K^{(\mathcal{P})}\mathbf{g} + R_{\mathrm{opt}} + \mathcal{O}(\|\Delta\theta_k\|_2^2), \qquad R_{\mathrm{opt}} = \boldsymbol{\delta}^\top \mathbf{r}_k^z. \tag{28}$$

Setting $\mathcal{P}_k = I$ and $r_k^{\mathrm{opt}} = 0$ recovers $K = JJ^\top$ and the $\mathcal{O}(\eta^2)$ expansion in Equation 6 of the main text.

B.3 Diagonal Contributions and Their Scope

Separate the diagonal and cross-token terms by defining

$$\kappa_p = K_{p,p}^{(\mathcal{P})} = \nabla_\theta z_p^\top \mathcal{P}_k \nabla_\theta z_p \geq 0, \qquad R_{\mathrm{cross}} = \sum_p \sum_{q \neq p} \delta_p K_{p,q}^{(\mathcal{P})} g_q. \tag{29}$$

Then

$$\Delta D = \eta \left(\sum_p \kappa_p \delta_p g_p + R_{\mathrm{cross}} \right) + R_{\mathrm{opt}} + \mathcal{O}(\|\Delta\theta_k\|_2^2). \tag{30}$$

For the token energy $D_p = \frac{1}{2}\delta_p^2$, the diagonal contribution of the current gradient is

$$\Delta D_p^{\mathrm{diag}} = \eta \kappa_p \delta_p g_p. \tag{31}$$

Whenever $\kappa_p > 0$, its sign equals $\mathrm{sign}(\delta_p g_p)$: positive values amplify the existing gap and negative values contract it within this diagonal contribution. For the vanilla coefficients in Equation 23, this is the sign of $A_i \delta_{i,t}$, giving the two amplifying regions in Equation 9.

The realized motion of the same token also includes the remaining terms:

$$\delta_p \Delta\delta_p = \eta \kappa_p \delta_p g_p + \eta \delta_p \sum_{q \neq p} K_{p,q}^{(\mathcal{P})} g_q + \delta_p r_{k,p}^z + \mathcal{O}(\|\Delta\theta_k\|_2^2). \tag{32}$$

Cross-token coupling, optimizer residuals, and higher-order effects can therefore change the realized sign of $\delta_p \Delta\delta_p$. The criterion $\delta_p g_p$ isolates the token's own current-gradient contribution; Appendix C.3 measures its agreement with realized motion in the characterization runs.

Table 3: Model-specific configuration of the learner–sampler mismatch characterization runs. The characterization horizon denotes the updates used in the analysis rather than the configured maximum run length.

Configuration	Qwen3-4B-Base	Qwen3-30B-A3B-Base
Architecture	Dense	MoE
NVFP4 scope	All MLP Layers	All Expert Layers
Max. completion	16,384	20,480
Routing replay	N/A	Enabled
Expert parallelism	N/A	8
Characterization horizon	324 updates	724 updates

B.4 Amplifying and Contracting Masses

Using $[x]_+ = \max(x, 0)$, define

$$B_{\mathrm{amp}} = \sum_p \kappa_p [\delta_p g_p]_+, \qquad B_{\mathrm{con}} = \sum_p \kappa_p [-\delta_p g_p]_+. \tag{33}$$

The identity $x = [x]_+ - [-x]_+$ gives

$$\Delta D = \eta \left(B_{\mathrm{amp}} - B_{\mathrm{con}} + R_{\mathrm{cross}} \right) + R_{\mathrm{opt}} + \mathcal{O}(\|\Delta\theta_k\|_2^2). \tag{34}$$

Thus B_{amp} and B_{con} quantify the two diagonal contributions, with the optimizer and cross-token effects retained. The advantage-weighted token masses used in Section 3.2 are tractable empirical summaries of the amplifying regions; they do not estimate B_{amp} or B_{con}, which also depend on gradient geometry and gap magnitude.

C Characterization Protocol and Additional Analyses

This appendix specifies the characterization runs, mismatch estimands, and analysis windows behind Section 3. It then evaluates when the diagonal indicator agrees with realized token motion and reports the complete localization statistics and segment-length sensitivity.

C.1 Models, Training Setup, and Evaluation

Models and training setup. We characterize learner–sampler mismatch on Qwen3-4B-Base and Qwen3-30B-A3B-Base. Both runs use DAPO-Math-17K, containing 17,398 training prompts, with learner seed 1234 and rollout/data seed 42. Each rollout samples 16 prompts with 16 responses per prompt, giving a global response batch size of 256, followed by one optimizer update. Sampler weights are refreshed after every learner update.

Training rollouts use temperature 1.0, top-p 1.0, and top-$k = -1$. The maximum completion length is 16,384 tokens for Qwen3-4B-Base and 20,480 tokens for Qwen3-30B-A3B-Base. We use Adam with a constant learning rate of 10^{-6} and no warm-up, $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-8}$, weight decay 0.1, and gradient clipping at 1.0. Master parameters, accumulated gradients, and optimizer states are maintained in FP32, while the sampler KV cache remains BF16.

NVFP4 configuration. We use native NVFP4 (W4A4) with one-dimensional block scaling and disable 16×16 square weight scaling. Randomized Hadamard transforms (RHT) are enabled, and stochastic rounding is used for gradient quantization. For Qwen3-4B-Base, NVFP4 is applied to the gate, up, and down projections of every MLP layer, while attention, normalization, and the tied embedding/output weights remain in BF16. For Qwen3-30B-A3B-Base, all expert MLP layers are quantized to NVFP4, while attention, normalization, embeddings, routing gates, and the independent LM head remain in BF16. Routing replay is enabled for the MoE model.

Table 3 summarizes the model-specific settings and the portions of each run used for the characterization analysis.

Characterization-time evaluation. During training, we evaluate AIME-2024 every ten learner updates using four independently sampled responses per problem. Decoding uses temperature 0.7 and top-p 1.0. These evaluations are used to track the training trajectories and determine the analysis windows in Section 3; they are distinct from the final checkpoint evaluation described below.

Table 4: Progress-normalized analysis windows used for learner–sampler mismatch characterization. For each model, we place analysis anchors at 25%, 50%, 75%, and 100% of the characterization trajectory and use the 25-update trailing window ending at each anchor. Stage names are descriptive shorthand for these progress-normalized anchors rather than independently selected training phases.

	Healthy	Drift	Pre-terminal	Terminal
Normalized anchor p	25%	50%	75%	100%
Qwen3-4B-Base ($S = 324$)	57–81	138–162	219–243	300–324
Qwen3-30B-A3B-Base ($S = 724$)	157–181	338–362	519–543	700–724

Independent checkpoint evaluation. Final benchmark evaluation is performed independently from the training sampler using a SGLang service. We evaluate AIME-2024, AIME-2025, MATH500, AMC23, and GSM8K using six sampled responses per problem. All benchmarks are decoded with temperature 0.7, top-p 1.0, top-$k = -1$, and min-$p = 0$, with thinking enabled and a maximum output length of 20,480 tokens.

The primary metric is the mean sample accuracy (`accuracy:mean`), which corresponds to sample-averaged pass@1 under repeated stochastic decoding. When additionally reported, pass@k is computed per problem as

$$1 - \frac{\binom{n-c}{k}}{\binom{n}{k}},$$

where n is the number of sampled responses and c is the number scored correct.

Hardware and software. All characterization runs use NVIDIA Blackwell B300 GPUs. The training environment uses PyTorch 2.9.1 with CUDA 13.0, Transformer Engine 2.10.0, miniTransformer 1.0.0, SGLang 0.5.10, and Megatron-LM and Slime at the revisions released with our artifact. Exact package revisions, container specifications, launch commands, and resolved configurations are included in the released reproducibility artifact.

C.2 Mismatch Estimands and Analysis Windows

We characterize native-NVFP4 GRPO trajectories of Qwen3-4B-Base and Qwen3-30B-A3B-Base. The 4B run measures the ordinary learner–sampler gap. For 30B, learner-side teacher-forced log probabilities are evaluated while replaying the sampler MoE routing decisions; its measured gap is therefore a residual mismatch after conditioning on routing:

$$\delta_{i,t}^{\mathrm{30B,res}} = z_{i,t}^{\text{learner,route replay}} - b_{i,t}, \tag{35}$$

where $b_{i,t} = \log \pi_{\mathrm{s}}(y_{i,t} \mid x, y_{i,<t})$ is the frozen generation-time log-probability and is reused unchanged in post-update evaluation. Absolute mismatch magnitudes from the two runs are consequently not treated as directly comparable.

Across successful updates, the logged trajectories contain 232,089,183 valid completion tokens for 4B and 230,713,088 for 30B. Zero-advantage responses account for 41.27% and 33.16% of these tokens, respectively. Marginal gap statistics use all valid completion tokens, whereas the gradient-bearing occupancy reported in the main text excludes $A_i = 0$. The $|A|$-weighted statistics automatically assign zero mass to such tokens.

For model m with characterization horizon S_m, we place analysis anchors at normalized progress $p \in \{0.25, 0.50, 0.75, 1.00\}$. The anchor update and its inclusive 25-update trailing window are

$$a_m(p) = \mathrm{round}(pS_m), \qquad \mathcal{W}_m(p) = \{a_m(p) - 24, \ldots, a_m(p)\}. \tag{36}$$

The four stage-wise analysis windows used in Section 3.2 are listed in Table 4.

The collapse trigger is run-specific. For the 4B run, let $c_s = \mathbf{1}[\mathrm{raw_reward}_s \le 0.10 \wedge \mathrm{pass@16}_s \le 0.3125]$ from the logged training metrics at update s, the latter computed over the rollout group of 16 samples per prompt; the collapse onset is the first update completing three consecutive violations, $\min\{s : c_{s-2} = c_{s-1} = c_s = 1\} = 304$. For the 30B run, the onset is the first update with a non-finite logged gradient norm, update 724, which terminates the trajectory. The two criteria are used only to align stages within each run and do not assert a shared failure mechanism.

For the 30B terminal window, update 724 contributes its pre-update snapshot to distributional and segment statistics, but the failed optimizer step is not included in successful-update token counts. For 4B, the terminal window extends beyond the first collapse trigger and is used to characterize the resulting global-mismatch regime rather than to identify the trigger time itself.

C.3 Empirical Scope of the Directional Risk Indicator

The decomposition in Appendix B retains cross-token interactions and optimizer residuals. We measure directly when its diagonal sign indicator agrees with the realized one-update motion.

C.3.1 One-Update Directional Audit

For a saved anchor k, let θ_k^- and θ_k^+ denote the learner parameters before and after one successful optimizer update. We reuse the identical rollout batch, sampled prefixes, advantages, masks, and logged sampler log probabilities, and perform teacher-forced learner replay before and after the update:

$$\delta_{i,t}^- = z_{i,t}^- - b_{i,t}, \qquad \delta_{i,t}^+ = z_{i,t}^+ - b_{i,t}. \tag{37}$$

Hence the realized gap motion is

$$\Delta\delta_{i,t} = \delta_{i,t}^+ - \delta_{i,t}^-. \tag{38}$$

The fixed-sampler identity

$$\Delta\delta_{i,t} = z_{i,t}^+ - z_{i,t}^- \tag{39}$$

is checked numerically with absolute tolerance 2×10^{-6} and zero relative tolerance. Replay is executed under `eval()` and both attention and hidden-state dropout are disabled. For the 30B model, the MoE routing cache is reused exactly. We do not snapshot and restore the kernel-level stochastic-rounding RNG state; the audit therefore measures the realized implementation-level one-update motion rather than asserting a bitwise-identical quantization-noise realization before and after the parameter update. The reference sampler remains fixed in this audit; it does not measure the change in execution mismatch after sampler weights are refreshed.

Each probe contains exactly one optimizer step per rollout. An anchor is retained only when `optimizer.step()` reports a successful update and the optimizer scheduler advances. The late anchors are updates 248–252 for 4B and 598–602 for 30B.

For the response-mean reducer, the vanilla token coefficient is $g_{i,t}^{\mathrm{eff}} = \gamma_k A_i / n_i$ (Eq. 23). The audit uses $\widetilde{g}_{i,t} = A_i / n_i$, retaining the response-length factor and removing the positive update-common factor γ_k. This preserves the diagonal sign and normalized agreement within each update. Pooled scores across anchors use these normalized coefficients, without restoring anchor-specific outer factors. On an eligible token $j \equiv (i,t)$, define

$$p_j = \mathrm{sign}(\delta_j^- \widetilde{g}_j), \qquad r_j = \mathrm{sign}(\delta_j^- \Delta\delta_j). \tag{40}$$

The main audit requires $|\delta_j^-| > 10^{-12}$, $\widetilde{g}_j \neq 0$, and $\Delta\delta_j \neq 0$; exact zeros are treated as ties and excluded.

For a selected token population $\mathcal{S}$, the unweighted agreement is

$$\mathrm{Agr}_{\mathrm{unw}}(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{j \in \mathcal{S}} \mathbf{1}[p_j = r_j]. \tag{41}$$

We additionally weight tokens by the magnitude of their diagonal directional signal,

$$u_j = |\delta_j^- \widetilde{g}_j|. \tag{42}$$

The weighted agreement is

$$\mathrm{Agr}_{\mathrm{w}}(\mathcal{S}) = \frac{\sum_{j \in \mathcal{S}} u_j \, \mathbf{1}[p_j = r_j]}{\sum_{j \in \mathcal{S}} u_j}. \tag{43}$$

As an outlier-robustness check, we also cap u_j at its 99.9th percentile within each reported population before applying Eq. 43.

Table 5: Agreement (%) between the diagonal directional indicator $\mathrm{sign}(\delta g^{\mathrm{eff}})$ and realized mismatch motion $\mathrm{sign}(\delta\Delta\delta)$ on late-stage $\mathcal{A}^-$ tokens. "Weighted" uses $|\delta g^{\mathrm{eff}}|$; "Capped" additionally caps this weight at the population-specific 99.9th percentile. † denotes low-sample group.

Model	Stratum	Tokens	Unw.	Weighted	Capped
Qwen3-4B-Base	All	1,640,752	38.88	47.73	46.81
Qwen3-4B-Base	$\lvert\delta\rvert > 0.5$	47,436	47.99	50.09	49.70
Qwen3-4B-Base	$\lvert\delta\rvert > 1$	18,822	50.43	52.34	51.99
Qwen3-4B-Base	$\lvert\delta\rvert > 3$	1,419	59.62	63.05	63.01
Qwen3-4B-Base	$\lvert\delta\rvert > 6$	392†	60.20	65.75	65.73
Qwen3-30B-A3B-Base	All	231,854	33.69	25.27	25.44
Qwen3-30B-A3B-Base	$\lvert\delta\rvert > 0.5$	9,228	25.88	24.07	24.07
Qwen3-30B-A3B-Base	$\lvert\delta\rvert > 1$	5,887	24.27	23.27	23.35
Qwen3-30B-A3B-Base	$\lvert\delta\rvert > 3$	2,026	21.08	21.86	21.95
Qwen3-30B-A3B-Base	$\lvert\delta\rvert > 6$	723†	19.78	22.18	22.20

C.3.2 Agreement as Mismatch Becomes Severe

Table 5 reports the late-stage audit in the negative-gap amplifying region $\mathcal{A}^- = \{A_i < 0, \delta_{i,t} < 0\}$.

For the Qwen3-4B-Base vanilla-mismatch run, the diagonal indicator becomes substantially more informative as mismatch severity increases. Weighted agreement rises from 47.73% over all eligible late-stage $\mathcal{A}^-$ tokens to 63.05% for $|\delta| > 3$. The latter population still contains $1{,}419$ tokens across 316 responses. More importantly, the weighted agreement for this stratum remains above the 50% reference at each of the five saved anchors, ranging from 59.03% to 66.49%. Capping the largest 0.1% of weights changes the pooled result only from 63.05% to 63.01%, showing that the trend is not driven by a handful of extreme-weight tokens. We do not base the main claim on the $|\delta| > 6$ row because of its lower support.

The same token-wise trend does not reproduce for the Qwen3-30B-A3B-Base routing-replay residual estimand: weighted agreement remains below the 50% reference and decreases rather than increases with severity. In this conditioned residual setting, the diagonal term in Equation 34 alone is not a reliable token-wise predictor. The current measurements do not isolate whether this behavior is primarily caused by cross-token coupling, routing-conditioned residual structure, or their interaction. We therefore restrict token-wise predictiveness claims to the Qwen3-4B-Base vanilla characterization and use the two models jointly only for the population-level directional and localization patterns reported in the main text.

C.4 Additional Analysis of Localized Mismatch

This section provides the construction, complete stage-wise statistics, and sensitivity checks for the segment-level localization analysis in Section 3.3.

C.4.1 Segment Construction and Threshold Calibration

Each response is partitioned independently by its original zero-based completion-token position t: segment k contains valid positions with $\lfloor t/W \rfloor = k$. Masked positions are excluded from segment statistics without reindexing the remaining tokens. Empty segments are omitted, and the final nonempty segment is retained even if it contains fewer than W valid tokens. No segment crosses a response boundary. The main analysis uses $W = 64$.

Because the localized failure mode of interest is the negative-gap amplifying region, segment statistics are computed over responses with $A_i < 0$. For a segment S in response i, define

$$f_S^{\mathrm{amp}}(\tau) = \frac{1}{|S|} \sum_{t \in S} \mathbf{1}[A_i < 0,\ \delta_{i,t} < \tau]. \tag{44}$$

The analysis threshold is calibrated once per model from the healthy reference window:

$$\tau_{\mathrm{ana}}^{(m)} = Q_{0.05}\left(\{\delta_{i,t} : (i,t) \in \mathcal{T}_{m,\mathrm{healthy}}\}\right), \tag{45}$$

where the quantile is computed over *all valid response tokens* in the healthy window, rather than only the $A_i < 0$ subset. This gives

$$\tau_{\text{ana}}^{(4\text{B})} = -0.24208, \qquad \tau_{\text{ana}}^{(30\text{B})} = -0.03949. \tag{46}$$

A segment is called *tail-heavy* when

$$h_S = \mathbf{1}\left[f_S^{\text{amp}}(\tau_{\text{ana}}) > 0.1\right]. \tag{47}$$

This calibration separates the localization question from later global tail growth: τ_{ana} is fixed before drift rather than re-estimated at every stage.

C.4.2 Localization Metrics

Let $\mathcal{S}$ denote all eligible segments pooled within an analysis window. We rank them by f_S^{amp} and retain the top $\lceil 0.05|\mathcal{S}| \rceil$ as $\mathcal{S}_{5\%}$. Ties are resolved by decreasing tail count, followed by canonical response and segment identifiers.

The amplifying-tail base rate is

$$r_{\text{amp}} = \frac{\sum_{S\in\mathcal{S}} \sum_{t\in S} \mathbf{1}[\delta_{i,t} < \tau_{\text{ana}}]}{\sum_{S\in\mathcal{S}} |S|}. \tag{48}$$

The fraction of amplifying-tail tokens contained in the top-5% segments is

$$C_{\text{tok}} = \frac{\sum_{S\in\mathcal{S}_{5\%}} \sum_{t\in S} \mathbf{1}[\delta_{i,t} < \tau_{\text{ana}}]}{\sum_{S\in\mathcal{S}} \sum_{t\in S} \mathbf{1}[\delta_{i,t} < \tau_{\text{ana}}]}. \tag{49}$$

The 5% uniform-allocation reference describes equal tail mass per segment. It is not a random-position null for C_{tok}: ranking segments by their observed tail fractions raises expected top-5% containment even under random placement. We account for this selection in Section C.4.5.

Tail-heavy segment coverage is

$$H_{\text{seg}} = \frac{1}{|\mathcal{S}|} \sum_{S\in\mathcal{S}} h_S. \tag{50}$$

For adjacency, let $\mathcal{E}$ contain directed pairs $(S_{i,k}, S_{i,k+1})$ for which both segments are nonempty and belong to the same response. Probabilities over $\mathcal{E}$ weight every eligible pair equally. We report

$$P_{\text{adj}} = \frac{\Pr_{\mathcal{E}}(h_{S_{i,k+1}} = 1 \mid h_{S_{i,k}} = 1)}{\Pr_{\mathcal{E}}(h_{S_{i,k+1}} = 1)}. \tag{51}$$

The denominator is the successor-heavy marginal over eligible pairs, not H_{seg}. Thus $P_{\text{adj}} > 1$ indicates association relative to this pooled marginal. Response-level heterogeneity can also produce such an association; the response-conditioned controls below separate this reference from randomized token positions and segment order.

C.4.3 Stage-Wise Localization with $W = 64$

Table 6 reports the full stage-wise evolution of the core localization statistics.

The tail's frequency and its distribution across segments evolve differently. From the healthy to pre-terminal window, r_{amp} decreases from 4.90% to 2.75% on 4B and from 4.79% to 3.59% on 30B, while C_{tok} increases to 25.68% and 25.01%. At the terminal stage, tail-heavy coverage reaches 48.64% and 57.02%, and top-5% containment decreases. These statistics distinguish concentrated pre-terminal mismatch from the broader coverage of the terminal regime.

Response means can remain small despite these local tails. Figure 6 shows the full empirical distribution of $|\bar{\delta}_i|$ among negative-advantage responses containing at least one tail-heavy segment, with $\bar{\delta}_i = n_i^{-1} \sum_{t\,\text{valid}} \delta_{i,t}$. At pre-terminal, 97.22% of affected 4B responses and 79.94% of affected 30B responses satisfy $|\bar{\delta}_i| < 0.05$; the terminal fractions are 2.61% and 0.36%. Replacing the magnitude of the signed mean with the mean absolute token gap gives pre-terminal fractions of 52.37% and 50.21% at the same cutoff. Signed cancellation therefore contributes to dilution, while a local tail can also coexist with a small mean absolute gap.

Table 6: Stage-wise localization of negative-gap amplifying tokens with $W = 64$. r_{amp}, H_{seg}, and C_{tok} are percentages. P_{adj} uses the successor-heavy marginal over eligible adjacent pairs. The table reports descriptive statistics; matched random-position and segment-order controls are shown in Figure 7.

Model	Stage	r_{amp}	H_{seg}	C_{tok}	P_{adj}
Qwen3-4B-Base	Healthy	4.90	15.08	21.56	2.65
Qwen3-4B-Base	Drift	2.60	5.43	25.74	4.22
Qwen3-4B-Base	Pre-terminal	2.75	6.24	25.68	3.98
Qwen3-4B-Base	Terminal	31.53	48.64	15.13	1.84
Qwen3-30B-A3B-Base	Healthy	4.79	14.46	20.10	2.62
Qwen3-30B-A3B-Base	Drift	3.82	10.43	22.15	3.28
Qwen3-30B-A3B-Base	Pre-terminal	3.59	10.25	25.01	3.78
Qwen3-30B-A3B-Base	Terminal	22.91	57.02	18.09	1.45

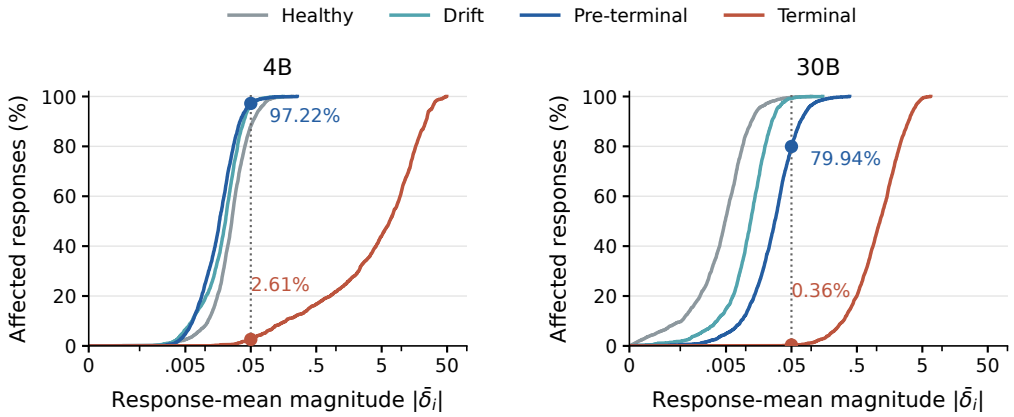

Figure 6: **Local tails within responses with small mean mismatch.** Empirical CDFs of $|\bar{\delta}_i|$ among $A_i < 0$ responses with at least one tail-heavy $W = 64$ segment. All four stage windows are shown for each model, with affected responses selected separately in each window and weighted equally. The vertical line marks 0.05; labels give the pre-terminal and terminal fractions below this cutoff.

C.4.4 Sensitivity to Segment Length

We repeat the analysis with $W = 128$, retaining the final shorter segment and using the same model-specific τ_{ana}. Table 7 summarizes the healthy-to-pre-terminal transition and the terminal regime.

The sequence-level dilution result is also preserved: 96.11% and 75.99% of affected 4B and 30B responses satisfy $|\bar{\delta}_i| < 0.05$ at the pre-terminal stage, whereas the corresponding terminal values fall to 1.86% and 0.23%. At terminal failure, top-5% token containment also decreases to 14.83% and 17.63%, and persistence approaches $1.84\times$ and $1.40\times$, respectively. These descriptive trends are consistent across the two segment lengths. The conditional randomization analyses below use the default $W = 64$ partition.

The fixed severe-gap criterion $\delta < -6$ provides a complementary severity check. With $W = 64$, it yields only 11, 7, and 49 tail-heavy segments in the healthy, drift, and pre-terminal 4B windows, and 0, 4, and 20 on 30B. This sparsity makes early adjacency ratios unstable. At terminal, the criterion includes 24.88% and 10.07% of eligible tokens, respectively. We therefore retain the healthy-calibrated threshold for localization and use $\delta < -6$ to describe the extreme terminal tail.

C.4.5 Response-Conditioned Localization Controls

To test whether concentration exceeds the effect of response-level tail rates and top-segment selection, we randomize tail positions within each response. The analysis retains the same four windows, thresholds, and $W = 64$ segmentation. Figure 7 reports the original metrics under this control and a separate exploratory test of segment order.

Table 7: Segment-length sensitivity with $W = 128$. Entries before and after the arrow correspond to the healthy and pre-terminal stages. r_{amp}, C_{tok}, and H_{seg} are percentages; P_{adj} uses the same eligible-pair definition as the $W = 64$ analysis.

Model	r_{amp}	C_{tok}	P_{adj}	Terminal H_{seg}
Qwen3-4B-Base	$4.90 \rightarrow 2.75$	$18.39 \rightarrow 21.81$	$2.82 \rightarrow 5.57$	48.98
Qwen3-30B-A3B-Base	$4.79 \rightarrow 3.59$	$17.23 \rightarrow 21.76$	$2.85 \rightarrow 4.37$	60.25

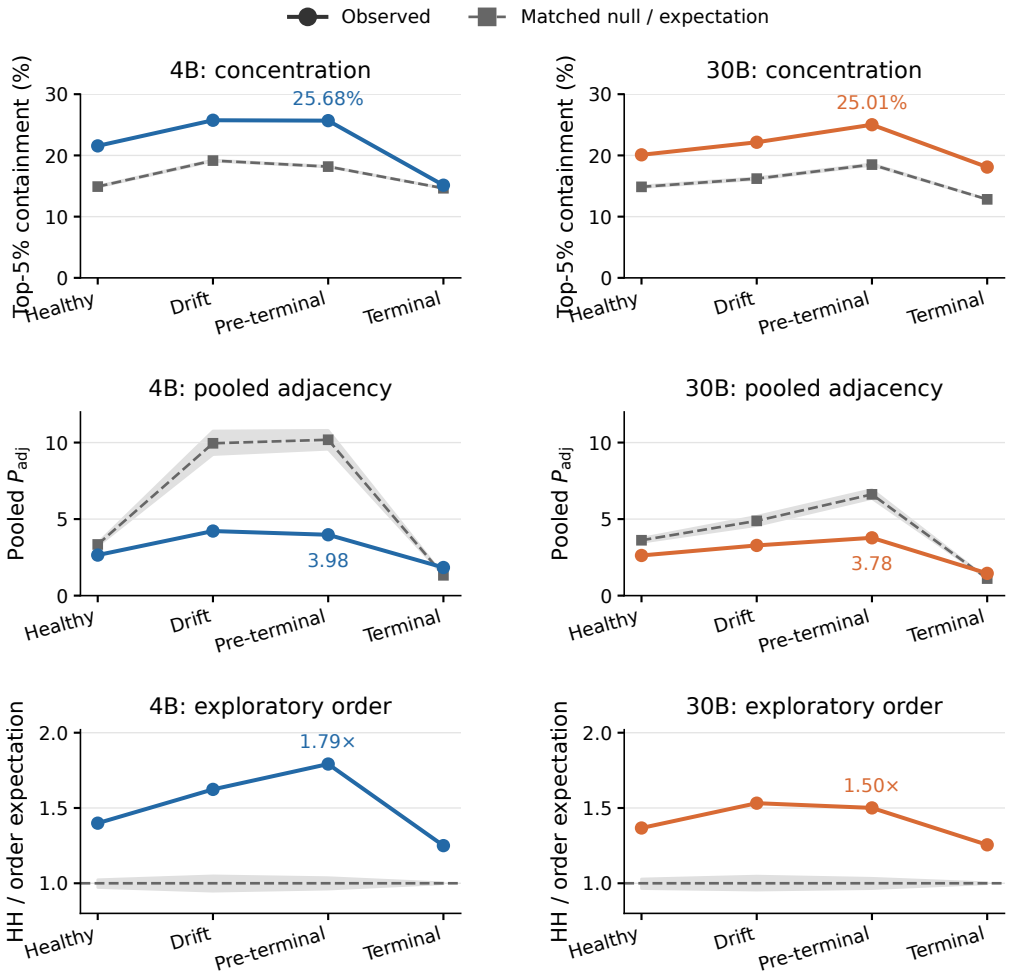

Figure 7: **Response-conditioned controls for local structure.** Top and middle: observed C_{tok} and P_{adj} compared with within-response tail-label shuffling, retaining all eligible segments, including short segments. Dashed curves show null medians. Bottom: the exploratory segment-order test, restricted to full segments and fixing each response's heavy-segment count; the dashed line is the analytic random-order expectation normalized to one. Shading denotes central 95% conditional randomization intervals from 499 draws per window, not across-run confidence intervals. Both models and all four windows are retained.

For every $A_i < 0$ response, the primary control fixes valid-token positions, response length, and total tail-token count, then uniformly permutes binary tail labels. Each of 499 randomizations recomputes segment tail fractions, the global top-5% ranking, and P_{adj}. Multivariate hypergeometric allocation implements this permutation exactly while preserving segment capacities, including short segments. We use randomization seed 20260924 and report central 95% conditional randomization intervals. These intervals describe random placement within the observed responses, not variability across independent training runs.

At pre-terminal, observed C_{tok} is 25.68% on 4B and 25.01% on 30B, compared with null medians of 18.17% and 18.50%. The corresponding null intervals are $[18.11\%, 18.23\%]$ and $[18.38\%, 18.63\%]$. The excesses of 7.51 and 6.51 percentage points show concentration beyond response-level tail-rate heterogeneity and rank selection. Such excess concentration is also present in the healthy windows; its magnitude does not increase monotonically across both trajectories.

The original P_{adj} has a different comparison: its pre-terminal value is 3.98 versus a token-shuffle median of 10.18 on 4B, and 3.78 versus 6.62 on 30B. Thus its elevation above one does not establish persistence beyond this response-conditioned null. Tail-label shuffling changes which segments are heavy, as well as their arrangement.

After examining this primary control, we conducted an exploratory segment-order test that conditions on the observed heavy-segment counts. For response i, we retain the original positions of its m_i full segments and uniformly choose h_i of these positions to carry heavy labels, preserving the observed heavy-segment count. Partial segments are excluded, and gaps do not create new adjacency edges. Let E_i be the set of genuinely adjacent full-segment pairs and T the total number of heavy–heavy pairs. Its random-order expectation is

$$\mathbb{E}_0[T] = \sum_{i:m_i \geq 2} |E_i| \frac{h_i(h_i - 1)}{m_i(m_i - 1)}.$$

At pre-terminal, observed counts are $2{,}892$ versus an expectation of $1{,}613.85$ on 4B, and $1{,}584$ versus $1{,}055.58$ on 30B, giving enrichments of $1.79\times$ and $1.50\times$. This comparison identifies non-random ordering conditional on the observed heavy-segment counts. It is a different statistic from P_{adj} and does not exclude within-response position trends or establish causal propagation along the response.

C.5 Full-Resolution Characterization Trajectories

Figure 8 places the four stage-wise summaries of Section 3.2 in the full saved trajectory: updates 1–324 for 4B and 1–724 for 30B. For each update, we merge the data-parallel shards and compute the fraction of valid completion tokens with $|\delta| < 0.05$ and the exact 0.05 and 0.01 gap quantiles. These marginal statistics include zero-advantage tokens. The ordinary 4B gap and routing-conditioned 30B residual retain their distinct estimands.

The directional panels compare raw $|A_i|$ weights with response-length-normalized $|A_i|/n_i$ weights over the strict regions $\mathcal{A}^- = \{A_i < 0, \delta_{i,t} < 0\}$ and $\mathcal{A}^+ = \{A_i > 0, \delta_{i,t} > 0\}$. Faint curves retain per-update ratios; bold curves pool numerator and denominator weights over trailing 25-update windows before taking their ratio. They are not moving averages of per-update ratios. Tokens with $\delta = 0$ belong to neither strict region, and tokens with $A_i = 0$ carry zero weight.

The complete trajectories show temporal variation that is lost when updates are pooled into broad consecutive windows. The directional ratio is non-monotonic, and length normalization reduces the raw-weighted imbalance, particularly on 30B. The bulk and tail panels locate these changes relative to the recorded terminal regime; the four shaded windows remain descriptive summaries of one characterization run per model. Section C.6 gives the corresponding stage-wise sensitivity results.

C.6 Sensitivity to Aggregation Choices

We examine two choices in the directional summaries: the trailing update-window width and the weighting assigned to each valid token. For a region $\mathcal{R}$, define

$$w_q(\mathcal{R}) = \frac{\sum_{i,t} q_{i,t}\, \mathbf{1}[(i,t) \in \mathcal{R}]}{\sum_{i,t} q_{i,t}}, \qquad \rho_q = \frac{w_q(\mathcal{A}^-)}{w_q(\mathcal{A}^+)}.$$

The stage-based summaries in Table 8 retain the 25-update windows of Section C.2, shown as gray spans in Figure 8. The primary summary uses $q_{i,t} = |A_i|$. The alternative uses $q_{i,t} = |A_i|/n_i$, where n_i is the valid completion length of response i, to include the response-mean reduction factor.

Extending the pre-terminal window from 25 to 50 updates changes the raw-weighted ratio from 2.319 to 2.454 on 4B and from 1.851 to 1.735 on 30B. Both windows retain the raw-weighted pre-

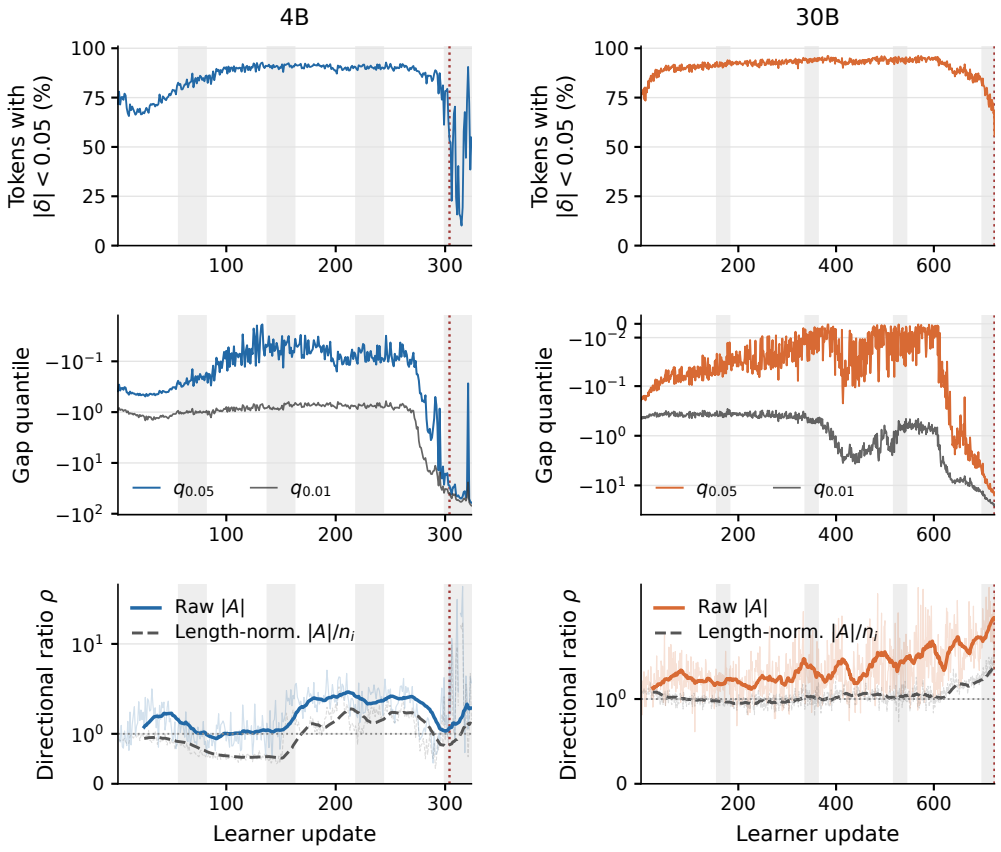

Figure 8: **Bulk, tail, and directional summaries along the full characterization trajectories.** Top: fraction of all valid completion tokens with $|\delta| < 0.05$ at each update. Middle: exact per-update signed gap quantiles $q_{0.05}$ and $q_{0.01}$. Bottom: raw and length-normalized $\mathcal{A}^-/\mathcal{A}^+$ weight ratios, with per-update values shown faintly and trailing 25-update pooled values in bold. The middle and bottom rows use symmetric-logarithmic vertical axes. Gray spans mark the original analysis windows; red lines mark the run-specific collapse triggers (4B: 304; 30B: 724). The 30B endpoint uses the pre-update snapshot of the failed update.

terminal imbalance. The intervals are 219–243 versus 194–243 for 4B and 519–543 versus 494–543 for 30B.

Length normalization changes the relative influence of long responses. It reduces the pre-terminal ratio to 1.434 on 4B and 1.049 on 30B, and changes whether the ratio exceeds one in some earlier stages. Raw advantage-weighted token mass and length-normalized coefficient mass therefore answer different questions. We retain the former for Section 3.2 and report the latter explicitly here. Both $\mathcal{A}^-$ and $\mathcal{A}^+$ are amplifying regions under the diagonal sign criterion, so their ratio describes a directional allocation within that part of the decomposition, not the balance between all amplifying and contracting contributions.

D TRIAGE SPECIFICATION AND DERIVATIONS

This appendix specifies the segment gate, derives the response-level reweighting in Section 4.1, and derives the repair coefficient in Section 4.2. Throughout, advantages, sampler log probabilities, masks, and gate weights are held fixed during each learner update. The learner log probabilities used by repair retain their gradients.

Table 8: **Sensitivity of directional summaries.** The window comparison holds raw advantage weighting fixed. The weighting comparison holds each 25-update stage window fixed. Masses are percentages of the corresponding total weight.

(a) Window width: 25 versus 50 trailing updates

Model	Stage	ρ_{25}	ρ_{50}	$w_{25}(\mathcal{A}^-)$	$w_{50}(\mathcal{A}^-)$
4B	Pre-terminal	2.319	2.454	36.18	36.90
30B	Pre-terminal	1.851	1.735	19.50	18.80

(b) Weight definition: raw versus length-normalized

Model	Stage	ρ_{raw}	ρ_{eff}	$w_{\mathrm{raw}}(\mathcal{A}^-)$	$w_{\mathrm{eff}}(\mathcal{A}^-)$
4B	Healthy	1.023	0.654	25.11	19.98
4B	Drift	1.504	0.796	28.92	20.83
4B	Pre-terminal	2.319	1.434	36.18	28.51
4B	Terminal	1.936	1.297	28.38	25.30
30B	Healthy	1.337	0.963	23.98	20.30
30B	Drift	1.488	1.019	17.43	14.03
30B	Pre-terminal	1.851	1.049	19.50	14.17
30B	Terminal	3.391	1.652	31.13	22.00

D.1 Segment Construction and the Gate Operating Rule

Let $m_{i,t} \in \{0,1\}$ be the valid completion-token mask and $n_i = \sum_t m_{i,t}$. Diagnostic segments contain consecutive response positions and do not cross response boundaries. We retain the final shorter segment and omit segments with no valid tokens. Masked positions contribute neither to segment statistics nor to the loss. For a segment S, let $n_S = \sum_{t \in S} m_{i,t} > 0$. The two diagnostic statistics are those in Eq. 13.

For outcome-based GRPO, the advantage is constant within a response. Consequently, a segment is eligible for gating exactly when $A_i < 0$. Two sources of risk determine its weight: a negative mean gap and a concentration of deep negative-gap tokens. The mean-gap component uses thresholds $\tau_{\mathrm{sev}} < \tau_{\mathrm{neg}} < 0$ and weights $0 < w_{\mathrm{sev}} \le w_{\mathrm{neg}} \le 1$:

$$w_S^{\mathrm{mean}} = \begin{cases} w_{\mathrm{sev}}, & \bar{\delta}_S < \tau_{\mathrm{sev}}, \\ w_{\mathrm{neg}}, & \tau_{\mathrm{sev}} \le \bar{\delta}_S < \tau_{\mathrm{neg}}, \\ 1, & \bar{\delta}_S \ge \tau_{\mathrm{neg}}. \end{cases} \tag{52}$$

The number of deep-tail tokens and the tail component are

$$k_S^{\mathrm{tail}} = \sum_{t \in S} m_{i,t} \mathbf{1}[\delta_{i,t} < \tau_{\mathrm{tail}}] = n_S f_S^{\mathrm{tail}}, \qquad w_S^{\mathrm{tail}} = 1 - (1 - w_{\mathrm{sev}}) \min\left(\frac{k_S^{\mathrm{tail}}}{K_{\mathrm{tail}}}, 1 \right), \tag{53}$$

where $K_{\mathrm{tail}} > 0$ is the count at which tail attenuation reaches w_{sev}. Combining the two components gives

$$w_S = \begin{cases} \min(w_S^{\mathrm{mean}}, w_S^{\mathrm{tail}}), & A_i < 0, \\ 1, & A_i \ge 0. \end{cases} \tag{54}$$

Thus $w_{\min} = w_{\mathrm{sev}}$. In the compact notation of Eq. 14, $\mathcal{G}$ uses the segment's valid count when converting f_S^{tail} to k_S^{tail}. This accounts for a short final segment or masked positions without introducing a separate gate symbol. Finally, Eq. 15 applies w_S only to negative-gap tokens in an eligible segment. The remaining tokens retain raw weight one.

D.2 From the Weighted Loss to the Effective Update

Write $H_i = \sum_t m_{i,t} w_{i,t}$ and $\bar{w}_i = H_i / n_i$. Because the weights are positive on valid tokens, $H_i > 0$ for every nonempty response. Treating the gate as a constant during backpropagation gives

$$\nabla_\theta \mathcal{L}_{\mathrm{PG},i}^{\mathrm{TRIAGE}} = \frac{1}{H_i} \sum_t m_{i,t} w_{i,t} \nabla_\theta \ell_{i,t}^{\mathrm{base}} = \frac{1}{n_i} \sum_t m_{i,t} \frac{w_{i,t}}{\bar{w}_i} \nabla_\theta \ell_{i,t}^{\mathrm{base}}. \tag{55}$$

This recovers $\alpha_{i,t} = w_{i,t}/\bar{w}_i$ in Eq. 17, including the original response and batch reduction factors in $g^{\text{base}}_{i,t}$. In particular,

$$\frac{1}{n_i}\sum_t m_{i,t}\alpha_{i,t} = 1, \qquad \frac{\alpha_{i,t}}{\alpha_{i,u}} = \frac{w_{i,t}}{w_{i,u}} \quad \text{for valid tokens } t, u. \tag{56}$$

The gate therefore changes relative allocation within a response. An ungated token has $\alpha_{i,t} = 1/\bar{w}_i \geq 1$; an attenuated raw weight need not yield $\alpha_{i,t} < 1$. If every valid token receives the same raw weight, all effective multipliers are one. The unit average of α does not impose a conservation law on the parameter-gradient norm or on the mismatch-energy contributions.

For comparison, a fixed-denominator variant uses

$$\mathcal{L}^{\text{fixed}}_{\text{PG},i} = \frac{1}{n_i}\sum_t m_{i,t}w_{i,t}\ell^{\text{base}}_{i,t}, \qquad \alpha^{\text{fixed}}_{i,t} = w_{i,t}. \tag{57}$$

This variant changes the response's overall multiplier as well as its within-response allocation.

D.3 Repair Objective and Additive Coefficient

Repair uses non-overlapping segments R, with response index $i(R)$, valid count $n_R = \sum_{t\in R} m_{i(R),t} > 0$, and mean gap

$$\bar{\delta}_R = \frac{1}{n_R}\sum_{t\in R} m_{i(R),t}\delta_{i(R),t}, \qquad q_R = [\tau_{\text{rep}} - \bar{\delta}_R]_+. \tag{58}$$

Here $\delta_{i,t} = z_{i,t}(\theta) - \log \pi_{\text{s}}(y_{i,t} \mid x, y_{i,<t})$, using the same sampler notation as the main text. Let $\mathcal{S}^{\text{valid}}_{\text{rep}}$ contain all nonempty repair segments in the optimizer-update batch and $N_{\text{rep}} = |\mathcal{S}^{\text{valid}}_{\text{rep}}|$. The denominator counts every such segment, including segments with $q_R = 0$ and segments from non-positive-advantage responses. For $N_{\text{rep}} > 0$, define

$$\mathcal{L}_{\text{rep},k} = \frac{\lambda_k}{N_{\text{rep}}}\sum_{R\in\mathcal{S}^{\text{valid}}_{\text{rep}}} \mathbf{1}[A_{i(R)} > 0]\phi_\beta(q_R), \qquad \lambda_k = \lambda_{\text{rep}}\mathbf{1}[k \geq k_{\text{rep}}]. \tag{59}$$

When the valid segment set is empty, the repair term and its gradient are zero. Counting all valid segments keeps the denominator independent of which segments violate the repair boundary.

For a token $t \in R$ with $i = i(R)$ in an active segment, the sampler term is fixed and

$$\frac{\partial q_R}{\partial z_{i,t}} = -\frac{m_{i,t}}{n_R}, \tag{60}$$

$$\phi'_\beta(q) = \frac{q}{\sqrt{1 + (q/\beta)^2}}. \tag{61}$$

At $q_R = 0$ the composed penalty has zero derivative. Let $R(i,t)$ be the unique repair segment containing a valid token. Gradient descent on Eq. 59 contributes

$$\Delta\theta_{\text{rep}} = -\eta\nabla_\theta\mathcal{L}_{\text{rep},k} = \eta\sum_{i,t}\psi_{i,t}\nabla_\theta z_{i,t}, \tag{62}$$

$$\psi_{i,t} = \frac{\lambda_k}{N_{\text{rep}}}\frac{m_{i,t}}{n_{R(i,t)}}\phi'_\beta(q_{R(i,t)})\mathbf{1}[A_i > 0,\ q_{R(i,t)} > 0]. \tag{63}$$

The coefficient is zero for tokens outside the valid repair set. Equation 63 recovers Eq. 20, including the positive-advantage condition and the activation schedule.

Bounded corrective coefficients. For $\beta > 0$ and finite $q \geq 0$, the pseudo-Huber penalty obeys

$$\begin{aligned} \phi_\beta(q) &\sim \tfrac{1}{2}q^2 && (q/\beta \to 0), \\ \phi_\beta(q) &= \beta q - \beta^2 + \mathcal{O}(\beta^3/q) && (q/\beta \to \infty), \\ 0 &\leq \phi'_\beta(q) < \beta. \end{aligned} \tag{64}$$

Consequently, throughout training,

$$0 \leq \psi_{i,t} \leq \frac{\lambda_k\beta}{N_{\text{rep}}n_{R(i,t)}} \tag{65}$$

Table 9: Gate and repair configuration. The diagnostic segment length matches the 64-token length used in characterization; τ_{ana} remains a separate characterization threshold. Repair starts inclusively at the persisted, zero-based rollout-update index k_{rep}.

Parameter	Meaning	Value
W_{gate}	Diagnostic segment length	64
τ_{neg}	Negative mean-gap threshold	-0.5
τ_{sev}	Severe mean-gap threshold	-1.5
τ_{tail}	Deep-tail token threshold	-6.0
w_{neg}	Negative mean-gap weight	0.3
w_{sev}	Severe/minimum raw weight	0.1
K_{tail}	Tail-count saturation	2
W_{rep}	Repair segment length	128
τ_{rep}	Mean-gap repair boundary	-0.03
β	Pseudo-Huber scale	0.022
λ_{rep}	Repair coefficient after activation	0.015
k_{rep}	Inclusive repair start index	0

for valid tokens and nonempty repair sets. The upper inequality is strict when $\lambda_k > 0$. The non-strict form also covers the period before repair is enabled. This bound controls the coefficient multiplying each log-probability gradient; a bound on the parameter-gradient norm would additionally depend on the norms of those gradients.

Repair on positive-advantage segments. When $\lambda_k > 0$, an active positive-advantage segment with $q_R > 0$ assigns the same positive coefficient to all valid tokens. Negative-gap tokens therefore have $\delta_{i,t}\psi_{i,t} < 0$ in the diagonal decomposition of Section 3.1. Positive-gap tokens in the same segment also receive this coefficient. Thus the objective corrects negative segment-level displacement without imposing contraction on every token.

D.4 Implementation and Configuration

Gate diagnostics and weights are detached. The positive-advantage repair branch retains live learner gaps for valid tokens in a segment with $A_i > 0$. For the remaining tokens, detached copies preserve the segment mean used by the diagnostic computation. The loss numerator then includes only selected positive-advantage segments, while the denominator counts all valid repair segments. Both the resulting repair value and its learner gradient therefore match Eq. 59.

The activation boundary is inclusive: repair is enabled when the persisted rollout-update index reaches k_{rep}. With one learner update per rollout, this index supplies the schedule in Eq. 59. The schedule changes the repair coefficient; gate diagnostics, segment counts, and the policy objective are unchanged before activation.

To implement the response-mean policy objective in Eq. 16, first form each response's weighted mean and then apply the outer response average. The repair denominator is computed over the full optimizer-update batch. If the loss interface subsequently divides accumulated numerators by one denominator for the complete update, scale each local selected repair sum by that denominator divided by N_{rep} before the outer division. The result is the same global segment mean regardless of the data-parallel or microbatch partition.

One learner update. The following procedure implements the objective in Eq. 19:

1. Obtain learner log probabilities on the sampled prefixes and subtract the fixed generation-time sampler log probabilities. Retain a live gap for repair and a detached copy for gate diagnostics.
2. Partition each response into diagnostic segments, compute the masked statistics, and evaluate Eqs. 52–54. Apply the resulting weights only to the negative-gap tokens of negative-advantage responses.
3. Compute each response's weighted policy mean and average these response losses using the baseline's batch reduction.
4. Partition responses into repair segments and count every nonempty segment across the optimizer-update batch. Set λ_k from the fixed schedule and evaluate the positive-advantage repair objective.

5. Add the two losses and take the learner optimizer step. The effective current-gradient coefficient is $g_{i,t} = \alpha_{i,t} g_{i,t}^{\text{base}} + \psi_{i,t}$; optimizer-history effects are represented by the residual in Appendix B.

E Training and Evaluation Protocols

E.1 Main Training Comparisons

The main comparisons in Section 5 evaluate training stability and downstream reasoning performance with Qwen3-30B-A3B-Base as the primary model and Qwen3-4B-Base for replication and ablation. We use the DAPO (Yu et al., 2026) dataset for training and GRPO for policy optimization. The characterization runs in Section 3 instead provide the fixed trajectories used for directional and localization analysis. Their run-specific horizons, collapse criteria, and mismatch estimands are specified in Appendix C.1 and Appendix C.2.

The main training configurations are BF16, native NVFP4, NVFP4 with token-level truncated importance sampling (TIS), and NVFP4 with TRIAGE. The TIS truncation threshold is $C = 2.0$. TRIAGE applies the gating and repair operations in Section 4 to the importance-corrected objective. The native NVFP4 configurations use W4A4 forward execution on the sampler and learner, with the activation-scaling recipe described in Section 5.1. The maximum response length in the main training setup is $20{,}480$ tokens.

E.2 Downstream Evaluation

We evaluate mathematical reasoning on AIME 2024, AIME 2025, MATH500, AMC23, and GSM8K. Table 1 identifies the training step of each evaluated checkpoint. Its † marker denotes a run that collapses before its planned budget; the reported result then uses the last saved checkpoint preceding collapse. The Qwen3-4B-Base TRIAGE results include the intermediate step-300 checkpoint and the final step-600 checkpoint.

To summarize the five benchmark scores, we use the unweighted mean

$$\text{Avg} = \frac{1}{5} \sum_{b=1}^{5} \text{Score}_b, \tag{66}$$

where each Score_b is expressed as a percentage. This gives each benchmark equal weight, independently of its number of problems. Training reward, entropy loss, and the mean absolute learner–sampler log-probability gap are tracked separately as training diagnostics in Figure 4.

F Efficiency and Profiling

F.1 Natural-Generation RL Measurements

The rollout-throughput measurements in Table 2 use Qwen3-30B-A3B-Base on one node with eight NVIDIA Blackwell B300 GPUs. Slime coordinates colocated Megatron-LM learner and SGLang sampler execution. The learner parallel configuration is tensor, pipeline, and context parallel size 1, expert parallel size 8, and expert tensor parallel size 1. Each prompt has 16 sampled responses. Global batch sizes 256 and 512 therefore correspond to 16 and 32 prompts, respectively.

Each run starts from the base checkpoint and trains on DAPO-Math-17K with seed 42 and the same data order. Responses terminate naturally at EOS or at the $20{,}480$-token limit. A run contains 12 rollout iterations: iterations 0–1 are warm-up and iterations 2–11 supply the ten measured observations. Initialization, evaluation, and checkpoint saving are excluded.

For measured iteration j, let M_j be the number of generated completion tokens and T_j^{rollout} the rollout duration. The reported throughput is the arithmetic mean of per-iteration throughputs,

$$\overline{\text{TPS}} = \frac{1}{10} \sum_{j=2}^{11} \frac{M_j}{T_j^{\text{rollout}}}. \tag{67}$$

This counts the actual output tokens produced by each precision configuration. Iteration wall time includes rollout, learner updates, weight conversion and synchronization, and orchestration. Learner timing includes log-probability recomputation and the backward and optimizer update. In colocated execution, overlapping wait intervals are not added again to the wall-time total.

F.2 Paired Learner-Overhead Measurement

The incremental learner cost of TRIAGE is measured separately on eight B300 GPUs with five paired repetitions of the same frozen batch. Gating and repair are enabled for one member of each pair. The observed median learner-update overhead is 0.84%, with a median absolute increase of 0.225 s. This comparison holds the training workload fixed when measuring the additional objective computation; natural-generation iteration times also reflect changes in output lengths.

F.3 Fixed-output sampler evaluation

F.3.1 Independent Fixed-Output Configuration Selection

We additionally measure standalone sampler throughput with a fixed output workload. For Qwen3-30B-A3B, each batch contains 512 responses of 4,096 output tokens with a client concurrency limit of 512 on one B300 GPU. Throughput divides the total output tokens by the complete client batch time, including queuing and final request completion. The KV cache uses BF16. This sampler experiment uses PyTorch 2.9.1+cu130, SGLang 0.5.10, FlashInfer 0.6.12, and `sglang-kernel` 0.4.1+cu130.

Table 10: **Standalone Qwen3-30B-A3B throughput after configuration selection.** Ranges are min–max over five repetitions. Speedup is the NVFP4/BF16 median-throughput ratio.

Precision	Mem. frac.	Median	Range (output tokens/s)	Speedup
BF16	0.95	15,470.01	15,447.56–15,509.83	1.000×
NVFP4	0.90	18,175.67	18,126.37–18,195.69	1.175×

Both precisions receive a 12-candidate configuration search. The two leading candidates are remeasured, and the selected configuration for each precision is measured five times on the same physical GPU. Both selected configurations use the FlashInfer TRT-LLM MoE backend. The selected static memory fractions are 0.95 for BF16 and 0.90 for NVFP4. Table 10 reports the resulting medians and observed ranges. Under these independently selected configurations, NVFP4 throughput is $1.175\times$ the BF16 throughput.

F.3.2 Concurrency and output length

Figure 9 varies the serving operating point with fixed output workloads. The concurrency sweep uses a common starting configuration with memory fraction 0.8. Length sweeps retain the precision-specific selected configurations in Table 10. Throughput increases with concurrency over the measured 4K sweep. At fixed concurrency, it decreases as output length grows, and the relative gain varies with the operating point. These measurements complement Section 5.3 by holding the output workload fixed.

F.4 Controlled Long-Tail Workload

We isolate response-length heterogeneity with a fixed request-to-length mapping on eight independent single-GPU Qwen3-30B-A3B sampler replicas. At batch size 256, the mixed workload contains 224 responses of 2,048 tokens, 24 of 8,192 tokens, and eight of 20,480 tokens. Batch size 512 doubles each count. A uniform control assigns 3,200 tokens per response, preserving the total output-token count. The mapping uses seed 20260912, with identical requests and target lengths for both precisions.

For mixed lengths, median batch time decreases from 156.13 to 120.83 s at batch size 256 and from 165.26 to 129.28 s at batch size 512. The time ratios are $1.292\times$ and $1.278\times$. The equal-token uniform controls yield larger ratios of $1.538\times$ and $1.513\times$.

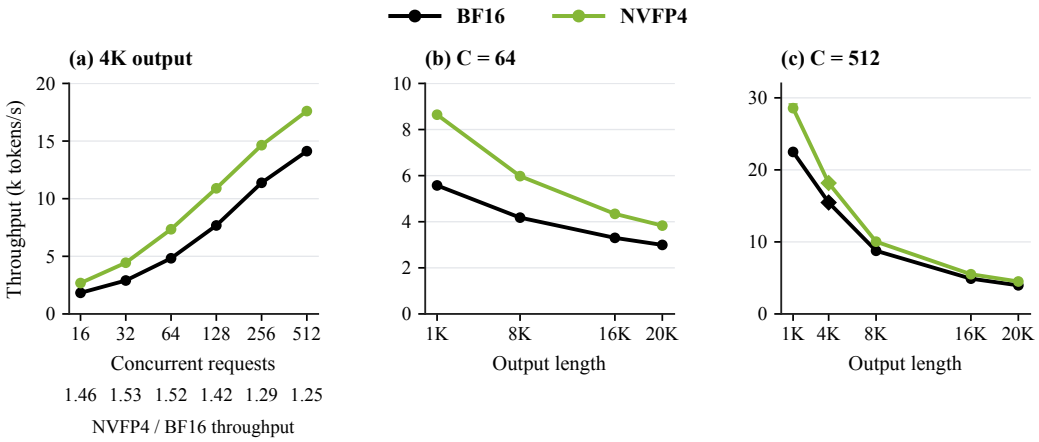

Figure 9: **Fixed-output serving sweeps on Qwen3-30B-A3B.** Each batch contains 512 responses on one B300 GPU. (a) Concurrency at 4K output tokens. The row below gives NVFP4/BF16 throughput ratios. (b,c) Length at concurrency 64 and 512. K denotes 1,024 tokens. Points show medians and whiskers min–max over three runs, except the separately confirmed C512/4K point (diamond), which uses five. The common-configuration C64/4K point is omitted from the selected-configuration length curve.

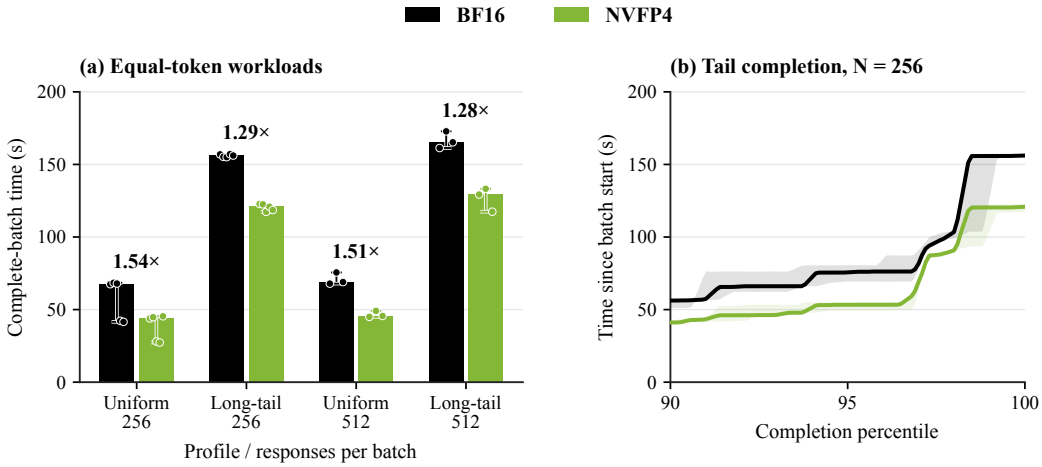

Figure 10: **Controlled long-tail and equal-token uniform workloads.** (a) Complete-batch wall time. Bars show medians, points individual runs, and whiskers min–max. Labels are BF16/NVFP4 median-time ratios. (b) Completion-time quantiles for the final 10% of requests in the 256-response mixed workload, with pointwise median and min–max across five repetitions. Batch size 256 uses five repeats per condition and batch size 512 uses three. Timing includes request handling, returned log-probabilities, and final draining, excluding initialization and warm-up.

NVFP4 therefore reduces absolute waiting time in these long-tail batches by approximately 35–36 s, while the relative gain is smaller than for uniform outputs. The completion curves expose the late portion of the batch that average throughput alone does not describe. These controlled sampler workloads complement Section 5.3 by fixing the output-length assignment.