Noisy-GRPO: Trajectory-Consistent Exploration for LLM Reinforcement Learning

Anonymous ACL submission

Abstract

Effective exploration is crucial for reinforcement learning in large language models (LLMs), yet common approaches such as temperature scaling and entropy regularization inject temporally uncorrelated noise at the token level, a dithering-style strategy ill-suited for long-horizon reasoning. We propose Noisy-GRPO, a trajectory-consistent exploration method that injects stochastic perturbations through a lightweight noisy head at the logits level. The noise realization is fixed within each trajectory but varies across rollouts, inducing coherent exploration patterns while preserving inter-trajectory diversity. Perturbations are modulated by hidden representations for context-aware exploration and confined to the Top-K logits to maintain generation stability. Experiments on mathematical reasoning benchmarks show that Noisy-GRPO outperforms strong GRPO-style and RLVR exploration baselines on five of six benchmarks, with a +2.9% average accuracy gain over DAPO on Qwen2.5-7B-Instruct. Analyses further show that Noisy-GRPO improves exploration efficiency by increasing rewarded prompt coverage, maintaining useful reward variance, and learning increasingly selective perturbations.

1 Introduction

Reinforcement learning has substantially enhanced the reasoning capabilities of large language models (LLMs) (OpenAI et al., 2024; Yang et al., 2025), particularly on mathematical and code generation tasks (Guo et al., 2025; Yu et al., 2025; Shao et al., 2024). Reinforcement learning with verifiable rewards (RLVR) (Lambert et al., 2025) is especially attractive because correctness can often be automatically checked. Fundamentally, however, RL success depends on effective exploration (Sutton and Barto, 2018; Kearns and Singh, 2002; Jaksch et al., 2010): policies must sample diverse trajectories that yield different rewards (Osband et al., 2016; Hong et al., 2018), thereby providing informative gradient signals for policy optimization.

In current LLM reinforcement learning pipelines, exploration is primarily encouraged through increased sampling temperature (Zheng et al., 2025) or entropy regularization (Jiang et al., 2025). These mechanisms increase token-level stochasticity, but the induced randomness is largely independent across decoding steps—a form of dithering-style exploration that produces temporally fragmented perturbations ill-suited for long-horizon reasoning (Holtzman et al., 2020; Fortunato et al., 2018). Increasing local uncertainty can therefore broaden outputs without reliably producing coherent alternative reasoning trajectories.

Classical reinforcement learning suggests a different direction: temporally correlated perturbations can induce coherent exploratory behaviors across an episode (Osband et al., 2013; Plappert et al., 2018; Fortunato et al., 2018). Translating this principle to LLM RL is nontrivial. Directly perturbing pretrained language-model parameters or full-vocabulary logits risks destabilizing well-calibrated token distributions, while the extremely large action space of LLMs (Brown et al., 2020) and the parallel rollout regime of modern RLVR (Shao et al., 2024; Sheng et al., 2025) make naive structured exploration inefficient.

These limitations raise a natural question: can we design an exploration mechanism that maintains behavioral coherence within individual trajectories, while still preserving sufficient diversity across different rollouts?

Therefore, we propose Noisy-GRPO, a trajectory-consistent exploration method for reinforcement learning in large language models. Noisy-GRPO injects stochastic perturbations through a lightweight noisy head at the logits level. The noise realization is fixed within each trajectory but varies across rollouts, inducing temporally coherent exploration patterns while preserving

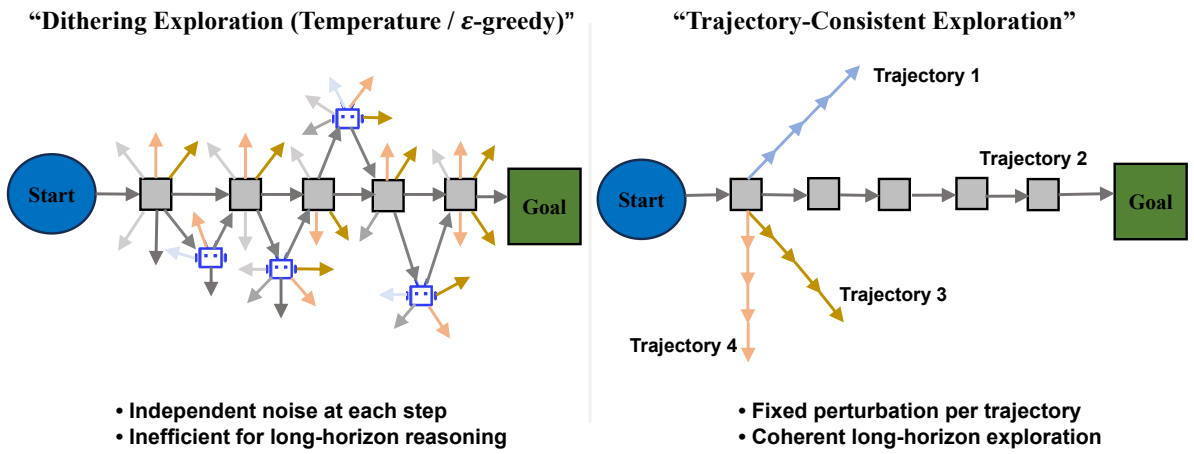


Figure 1: Dithering-style exploration versus trajectory-consistent exploration. Dithering methods inject independent noise at each step, leading to fragmented local deviations, while trajectory-consistent exploration applies a fixed perturbation within each rollout, producing coherent long-horizon behaviors.

diversity across trajectories. Perturbations are modulated by hidden representations for context-aware exploration and confined to the Top-K logits to maintain generation stability. This design introduces a learnable exploration mechanism while remaining compatible with parallel rollout training.

In summary, our contributions are as follows:

- We provide a trajectory-level perspective on exploration in reinforcement learning for large language models, framing widely used techniques such as temperature scaling and entropy regularization as forms of dithering-style exploration that inject temporally uncorrelated randomness at the token level, which is inefficient for long-horizon reasoning.
- We propose **Noisy-GRPO**, a trajectory-consistent exploration framework that injects stochastic perturbations at the logits level using a lightweight noisy head. By fixing noise realizations within each trajectory while varying them across rollouts, modulating perturbations through hidden representations, and confining exploration to a Top-K action subspace, Noisy-GRPO achieves coherent, context-aware exploration while remaining parameter-efficient and compatible with parallel training.
- We conduct extensive experiments on mathematical reasoning benchmarks, showing that Noisy-GRPO outperforms strong GRPO-style and RLVR exploration baselines on five of six benchmarks, with a +2.9% average accuracy gain over DAPO on Qwen2.5-7B-Instruct. Through analyses of training dynamics, exploration effectiveness, and learned perturbations, we show that Noisy-GRPO improves rewarded prompt coverage, maintains useful reward variance, and learns increasingly selective perturbations.

2 Preliminaries

2.1 Group Relative Policy Optimization

In reinforcement learning with verifiable rewards (RLVR), an autoregressive policy π_θ generates a trajectory $y = (a_1, \ldots, a_T)$ for a prompt x and receives a sparse terminal reward $r(x, y) \in \{0, 1\}$ from a verifier. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) estimates advantages through within-group comparisons rather than a learned value function. For a prompt x, the old policy samples a group of N trajectories $\{y^{(i)}\}_{i=1}^{N}$, where $y^{(i)} = (a_1^{(i)}, \ldots, a_{T_i}^{(i)})$, and obtains rewards $r^{(i)} = r(x, y^{(i)})$. A common group-relative advantage centers rewards within the group:

$$A^{(i)} = r^{(i)} - \mu_r, \qquad \mu_r = \frac{1}{N}\sum_{j=1}^{N} r^{(j)}, \tag{1}$$

with optional normalization by the group standard deviation in some implementations. The token-level importance ratio is

$$\rho_t^{(i)}(\theta) = \frac{\pi_\theta(a_t^{(i)} \mid x, a_{<t}^{(i)})}{\pi_{\theta_{\text{old}}}(a_t^{(i)} \mid x, a_{<t}^{(i)})}. \tag{2}$$

The clipped GRPO objective can then be written as

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\Bigg[\frac{1}{N}\sum_{i=1}^{N}\frac{1}{T_i}\sum_{t=1}^{T_i}\min\Big(\rho_t^{(i)}(\theta)A^{(i)}, \text{clip}(\rho_t^{(i)}(\theta), 1-\epsilon, 1+\epsilon)A^{(i)}\Big)\Bigg]. \quad (3)$$

Thus, GRPO-style RLVR relies on exploration that can produce reward variation within rollout groups.

2.2 Token-Level Dithering in LLM Rollouts

A standard way to encourage exploration in LLM rollouts is to sample from a temperature-scaled policy,

$$a_t \sim \text{Cat}(\text{softmax}(z_t/T)), \quad (4)$$

where z_t denotes the logits at step t and $T > 0$ is the sampling temperature. By the Gumbel-Max trick, this sampling process is equivalent to

$$a_t = \arg\max_j \{z_{t,j} + T g_{t,j}\}, \quad g_{t,j} \overset{\text{i.i.d.}}{\sim} \text{Gumbel}(0,1). \quad (5)$$

Since a fresh Gumbel noise vector is sampled at each decoding step, the induced perturbations are temporally uncorrelated within a trajectory. Temperature sampling therefore increases token-level randomness, but does not create a persistent trajectory-level exploratory mode. Entropy regularization similarly shapes the marginal token distribution at each step, but by itself does not impose temporal correlation across stochastic choices in a rollout. Appendix A gives a toy MDP illustrating why trajectory-consistent perturbations can require exponentially fewer rollouts than token-level dithering in coherence-dependent tasks.

3 Related Work

RLVR training and token-level exploration. Reinforcement learning with verifiable rewards (RLVR) has become a central paradigm for improving reasoning-oriented language models (Lambert et al., 2025; Guo et al., 2025). GRPO estimates advantages from groups of sampled responses without a learned value model (Shao et al., 2024), and DAPO further strengthens GRPO-style training with decoupled clipping, dynamic sampling, and token-level policy-gradient losses (Yu et al., 2025). Most rollout exploration mechanisms still operate by modifying token-level sampling distributions. Temperature sampling increases stochasticity by flattening the next-token distribution (Holtzman et al., 2020; Zheng et al., 2025), while entropy-based methods regularize or control token uncertainty during policy optimization (Wang et al., 2025; Shen, 2025). Recent RLVR exploration methods further refine this idea through selective entropy regularization (SIREN) (Jiang et al., 2025), Pass@k-derived signals (Chen et al., 2025), or high-temperature initiation followed by regular decoding (FIRE) (Chen et al., 2024). These methods improve token-level uncertainty, sampling, or reward design; Noisy-GRPO instead focuses on the temporal structure of rollout perturbations, making exploration consistent within a trajectory while preserving diversity across rollouts.

Structured exploration in reinforcement learning. Classical reinforcement learning has long shown that temporally correlated exploration can be more effective than independent per-step dithering. Posterior-sampling and randomized value-function methods induce consistent exploratory policies over an episode (Osband et al., 2013, 2016), while parameter-space perturbation methods such as parameter noise and NoisyNet produce coherent behavioral changes across decisions (Plappert et al., 2018; Fortunato et al., 2018). These ideas motivate trajectory-consistent exploration, but direct parameter or full-vocabulary perturbations are poorly suited to LLM RL due to calibrated language-model heads, extremely large vocabularies, and parallel rollout requirements.

4 Method

Effective exploration in LLM reinforcement learning requires two key properties: (1) *intra-trajectory coherence*—exploration behavior should remain consistent throughout a generation sequence, and (2) *inter-trajectory diversity*—different rollouts should explore distinct regions of the policy space. Our method, **Noisy-GRPO**, achieves both properties by injecting structured perturbations through a lightweight Noisy Residual Head operating at the logits level.

The core design principles are as follows:

- **Trajectory-consistent noise**: The noise realization is fixed within each trajectory but varies across rollouts, ensuring coherent exploration within sequences while maintaining diversity across samples.

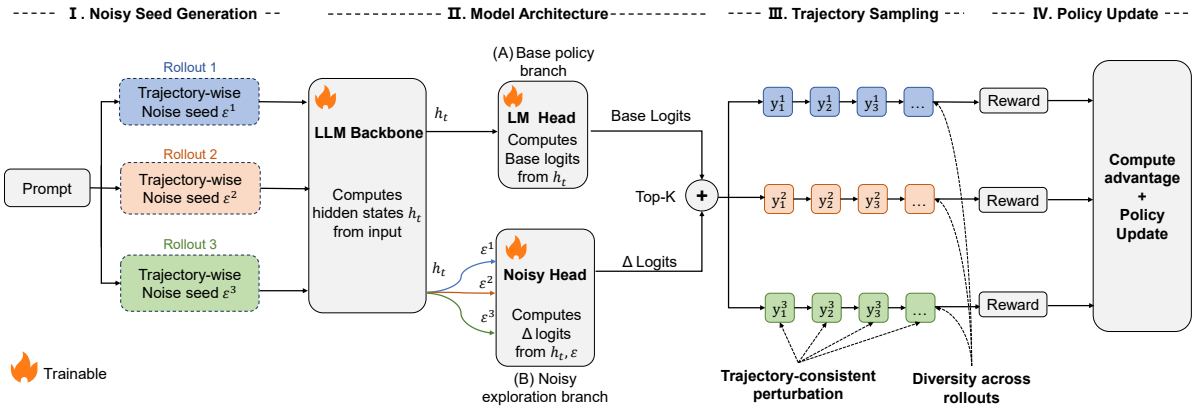


Figure 2: Overview of Noisy-GRPO. **(I) Noise Seed Generation**: Each rollout receives a unique noise seed $\varepsilon^{(i)}$, fixed within the trajectory but varying across rollouts. **(II) Model Architecture**: The LLM backbone computes hidden states h_t, passed to both the base LM Head and the Noisy Head. Perturbations are applied to Top-K logits only. **(III) Trajectory Sampling**: Tokens are sampled from the perturbed distribution, producing trajectories that are internally coherent yet diverse across rollouts. **(IV) Policy Update**: Policy gradients are computed using group-relative advantages.

- **Context-gated perturbation**: Perturbations are modulated by hidden representations, allowing exploration patterns to adapt to different semantic contexts.

- **Localized action space**: Noise is restricted to the Top-K logits, focusing exploration on plausible candidates.

We detail each component in the following subsections.

4.1 Trajectory-Consistent Logit Perturbations

We begin by formalizing trajectory-consistent exploration at the level of policy logits. Consider an autoregressive language model parameterized by θ. Given a prompt x and previously generated tokens $a_{<t}$, the model produces a hidden representation $h_t \in \mathbb{R}^H$ and the corresponding base logits

$$z_t = f_\theta(h_t), \tag{6}$$

which define the policy

$$\pi_\theta(a_t \mid x, a_{<t}) = \mathrm{Softmax}(z_t)_{a_t}. \tag{7}$$

As discussed in Section 2.2, common rollout exploration mechanisms induce token-level stochasticity that is resampled at each decoding step. To enable structured exploration at the trajectory level, we instead associate each rollout with a single noise realization that is fixed throughout the entire generation process.

Formally, for each rollout i, we sample a trajectory-level seed

$$\varepsilon^{(i)} \sim \mathcal{N}(0, I), \tag{8}$$

which is reused at every generation step within the same trajectory:

$$\varepsilon_t^{(i)} = \varepsilon^{(i)}, \quad \forall t = 1, \ldots, T. \tag{9}$$

Different rollouts are assigned independent seeds. This design ensures that stochastic perturbations are temporally correlated within a trajectory, while preserving diversity across rollouts.

We inject this trajectory-wise noise at the logits level by defining perturbed logits

$$\tilde{z}_t^{(i)} = z_t + \bar{\Delta}(h_t; \varepsilon^{(i)}), \tag{10}$$

where $\bar{\Delta}(\cdot) \in \mathbb{R}^{|\mathcal{V}|}$ denotes the vocabulary-level perturbation induced by the trajectory seed. The resulting policy is

$$\pi_\theta^{(i)}(a_t \mid x, a_{<t}) = \mathrm{Softmax}(\tilde{z}_t^{(i)})_{a_t}. \tag{11}$$

Under this formulation, each rollout can be interpreted as sampling from a distinct but internally consistent stochastic policy. Rather than inducing independent randomness at each token, the fixed noise realization biases the entire generation toward a coherent alternative behavior, enabling exploration over complete reasoning trajectories.

This trajectory-consistent formulation contrasts with dithering-style exploration, where noise is independently resampled at each time step. By correlating stochasticity across the temporal dimension,

our approach produces structured exploratory behaviors that are better suited for long-horizon reasoning tasks. In the next section, we show how the perturbation is implemented using a lightweight noisy head whose effects are modulated by the model's hidden representations, enabling context-aware exploration.

4.2 Context-Gated Noisy Head

Having established the trajectory-consistent noise injection mechanism, we now describe how the Top-K perturbation vector $\Delta(h_t; \varepsilon) \in \mathbb{R}^K$ is implemented.

Design Motivation. The hidden representation h_t encodes rich contextual information accumulated over the generation history. Rather than applying a fixed perturbation pattern, we allow h_t to act as a gate that selects which noise patterns are activated for the current context, enabling context-aware exploration without explicitly directing the policy toward specific actions.

Architecture. We implement the perturbation through a low-rank bilinear transformation. Specifically, we construct two noise-embedded matrices:

$$U^{(i)} = \sigma_{\text{in}} \odot \varphi(\varepsilon_{\text{in}}^{(i)}) \in \mathbb{R}^{R \times H}, \tag{12}$$

$$B^{(i)} = \sigma_{\text{out}} \odot \varphi(\varepsilon_{\text{out}}^{(i)}) \in \mathbb{R}^{R \times K}, \tag{13}$$

where R is the rank, H is the hidden dimension, and K is the number of top logits to perturb. The noise tensors $\varepsilon_{\text{in}}^{(i)} \in \mathbb{R}^{R \times H}$ and $\varepsilon_{\text{out}}^{(i)} \in \mathbb{R}^{R \times K}$ are deterministically generated from the trajectory seed $\varepsilon^{(i)}$, ensuring consistency within each rollout. The learnable parameters σ_{in} and σ_{out} control the scale of noise along each dimension and are optimized end-to-end through policy gradients. The k-th output dimension corresponds to the k-th ranked base logit under z_t, rather than to a fixed vocabulary item.

Given the hidden representation h_t, the perturbation is computed as:

$$\Delta_t^{(i)} = \Delta(h_t; \varepsilon^{(i)}) = \frac{1}{\sqrt{HR}} (h_t {U^{(i)}}^\top) B^{(i)} \in \mathbb{R}^K, \tag{14}$$

where the scaling factor $1/\sqrt{HR}$ stabilizes the magnitude.

Interpretation. The design factorizes into three components: (1) *noise direction* (ε) provides random directions fixed within each trajectory but varying across rollouts; (2) *noise magnitude* (σ) controls perturbation strength along each direction, learned automatically through RL gradients; and (3) *context gating* ($h_t {U^{(i)}}^\top$) determines which noise patterns are activated based on the current hidden state, producing context-dependent perturbations.

4.3 Localized Exploration in the Top-K Action Subspace

LLMs operate over extremely large action spaces (often exceeding 100,000 tokens), where low-probability tokens typically correspond to implausible continuations. Injecting perturbations uniformly over the entire vocabulary is both inefficient and destabilizing.

Noisy-GRPO restricts perturbations to the Top-K logits at each step. Let $z_t \in \mathbb{R}^{|\mathcal{V}|}$ denote the base logits and $\mathcal{K}_t$ the index set of the Top-K entries. We define $\text{rank}_t(j) \in \{1, \ldots, K\}$ as the rank position of token j within $\mathcal{K}_t$, sorted by the base logits. The perturbed logits are

$$\tilde{z}_{t,j}^{(i)} = \begin{cases} z_{t,j} + \Delta_{t,\text{rank}_t(j)}^{(i)}, & j \in \mathcal{K}_t, \\ z_{t,j}, & j \notin \mathcal{K}_t. \end{cases} \tag{15}$$

Equivalently, the K-dimensional perturbation vector is scattered back to the vocabulary positions selected by $\mathcal{K}_t$. This design localizes exploration to plausible tokens, ensuring stochasticity is applied only where it can meaningfully influence decision-making. The Top-K set is computed from the base logits z_t, independent of the perturbation, preventing noise from promoting implausible tokens into the candidate set.

Noisy-GRPO uses the same GRPO objective, but evaluates token probabilities under the perturbed policy induced by the trajectory seed. The noisy-head parameters $\phi = \{\sigma_{\text{in}}, \sigma_{\text{out}}\}$ are optimized jointly with the backbone: positive group-relative advantages reinforce the sampled actions and the perturbations that increased their probabilities, while negative advantages suppress them. For multi-epoch updates over the same rollout batch, we cache and replay each trajectory seed when recomputing log probabilities to keep importance ratios well defined; details are given in Appendix B. Appendix C reports the parameter, compute, and latency overhead.

5 Experiments

5.1 Experimental Setup

Training setup. We implement Noisy-GRPO with verl and train Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct (Yang et al., 2024) on MATH (7,500 problems) (Hendrycks et al., 2021a), using the DAPO recipe (Yu et al., 2025). Unless otherwise specified, Noisy-GRPO uses rank $R = 32$, Top-$K = 300$, and noisy-head initialization $\theta_{\text{in}} = \theta_{\text{out}} = 0.25$. Full hyperparameters are provided in Appendix D.

Baselines and Evaluation. We compare against GRPO temperature controls, DAPO, token-level or RLVR exploration baselines (higher-temperature sampling, entropy regularization, FIRE, SIREN, and Pass@k), and a token-noise mechanism control. Evaluation covers GSM8K (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021b), AMC 2023, OlympiadBench (He et al., 2024), Minerva Math (Lewkowycz et al., 2022), and AIME 2024. We use greedy decoding for all except AIME 2024, which reports avg@32.

5.2 Main Results and Baseline Comparisons

Table 1 reports the main comparison on Qwen2.5-7B-Instruct. The comparison is organized by the role of each baseline: base GRPO temperature controls, a strong DAPO recipe, explicit exploration baselines, and a token-level noise variant included as a mechanism control. Section 5.4 analyzes this control in detail, and Appendix F reports the corresponding 3B comparison.

Takeaway. Temperature tuning alone provides limited gains: GRPO peaks at $T = 1.0$, and DAPO with lower or higher rollout temperature underperforms DAPO at $T = 1.0$. Stronger RLVR exploration baselines such as SIREN and Pass@k improve substantially, but Noisy-GRPO remains best on five of six benchmarks and is within 0.1 points of the best result on AIME 2024. The token-level i.i.d. noisy-head control also improves over DAPO, but remains well below trajectory-consistent Noisy-GRPO, indicating that the gains are not due to generic noise injection alone.

5.3 Exploration Dynamics and Learned Perturbations

To understand why Noisy-GRPO improves over DAPO, we analyze the training process from three complementary perspectives: global training dynamics, group-level exploration signals, and the evolution of learned noise scales. These analyses show that Noisy-GRPO does not merely increase randomness; it sustains useful exploration, induces longer reasoning trajectories, and learns increasingly selective perturbations over training.

5.3.1 Training Dynamics

Figure 3 compares training dynamics between Noisy-GRPO and DAPO on Qwen2.5-7B-Instruct.

Entropy. DAPO exhibits rapid entropy collapse, dropping from 0.8 to 0.15 within 50 steps. Noisy-GRPO starts higher (1.2) and decays more gradually, stabilizing around 0.3. This slower entropy decay indicates that trajectory-consistent perturbations help preserve exploration capacity during the phase where DAPO quickly becomes exploitative.

Response Length. Noisy-GRPO generates longer responses throughout training (stabilizing at 4,200 vs. DAPO's 3,000 tokens). We view this length increase as part of the learned exploration behavior: the model samples and reinforces more extended reasoning trajectories. Importantly, this occurs together with slower entropy decay and higher group-level diversity (Section 5.3.2), suggesting a broader change in exploration dynamics rather than a pure verbosity effect. Appendix G further reports a token-budget matched evaluation showing that Noisy-GRPO remains above DAPO under the same generation budget, so the gains are not solely explained by longer outputs.

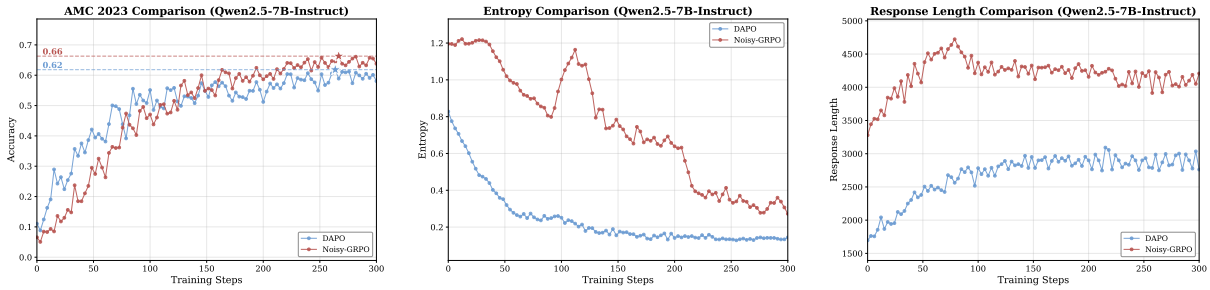
Accuracy. The AMC 2023 curve illustrates a characteristic exploration-exploitation trade-off. DAPO achieves higher early accuracy, which is consistent with its faster entropy decay: the policy quickly concentrates on high-confidence trajectories already favored by the pretrained model. This early exploitation improves short-term accuracy, but it also reduces answer diversity and group reward variance, limiting later learning signals. Noisy-GRPO explores more broadly at the beginning, which slows early gains but preserves diverse trajectories and useful reward variation. As a result, it continues improving after DAPO plateaus, surpassing DAPO around step 150 and ultimately reaching 66% vs. 62%.

5.3.2 Exploration Effectiveness

Prior work shows that policy-level noise induces larger state space coverage than per-action

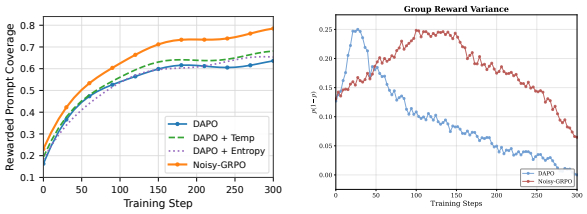
Method	GSM8K	Minerva	MATH500	AMC 2023	Olympiad	AIME 2024
Base GRPO with temperature sampling						
GRPO ($T = 0.1$)	89.8	34.5	76.1	58.6	37.6	20.9
GRPO ($T = 1.0$)	90.4	35.1	77.0	59.3	38.2	21.7
GRPO ($T = 1.5$)	90.1	34.8	76.5	58.8	37.9	21.3
GRPO-style recipe and token-level exploration baselines						
DAPO ($T = 0.1$)	91.2	35.8	77.4	60.2	39.0	22.4
DAPO ($T = 1.0$)	91.6	36.1	78.2	60.8	39.2	23.0
DAPO ($T = 1.5$)	91.4	36.0	77.8	60.3	39.1	22.8
DAPO + Entropy	91.0	35.9	78.0	60.7	38.9	22.7
FIRE	91.8	36.7	78.3	61.2	39.7	23.8
SIREN	<u>92.5</u>	<u>37.6</u>	<u>80.1</u>	<u>64.5</u>	<u>42.5</u>	**25.5**
Pass@k	92.3	37.1	79.7	64.3	41.4	24.6
Ours and mechanism control						
Noisy-GRPO (token noise)	92.0	36.8	78.7	62.0	40.3	23.7
Noisy-GRPO ($T = 0.1$)	**92.8**	**38.7**	**80.7**	**65.4**	**43.1**	<u>25.4</u>

Table 1: Main comparison on Qwen2.5-7B-Instruct. All values report accuracy (%); AIME 2024 uses avg@32. Best results are bolded and second-best results are underlined. DAPO + Entropy uses $\lambda = 2\times10^{-3}$.



(a) AMC 2023 Accuracy (b) Entropy (c) Response Length

Figure 3: Training dynamics on Qwen2.5-7B-Instruct. Noisy-GRPO maintains higher entropy, generates longer responses, and achieves higher final accuracy despite slower initial progress.



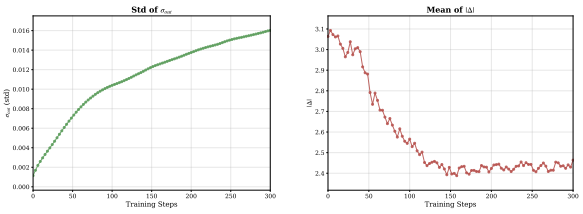
(a) Rewarded prompt coverage (b) Group reward variance $p(1-p)$

Figure 4: Exploration effectiveness on Qwen2.5-7B-Instruct. Noisy-GRPO covers more prompts with at least one rewarded rollout under the same rollout budget and maintains higher reward variance throughout training.

noise (Plappert et al., 2018; Eberhard et al., 2023). In GRPO-style RLVR, exploration quality is tied directly to whether a fixed rollout budget reaches reward-bearing trajectories and produces useful relative learning signals. We therefore evaluate exploration effectiveness using rewarded prompt coverage and group reward variance (Figure 4).

Rewarded Prompt Coverage. Coverage alone does not imply useful exploration. We therefore measure the fraction of prompts for which at least one rollout receives a positive reward under the same rollout budget. Noisy-GRPO maintains higher rewarded prompt coverage than DAPO and token-level exploration controls, indicating that trajectory-consistent perturbations more effectively reach reward-bearing reasoning paths rather than merely increasing local sampling randomness.

Group Reward Variance. Both methods reach the theoretical maximum $p(1-p) = 0.25$, but with different dynamics. DAPO peaks early (step 25) and collapses to near-zero by step 300, indicating degenerate groups with no learning signal. Noisy-GRPO maintains $p(1-p) \approx 0.25$ through steps 100–150 and retains meaningful variance ($\sim$0.06) at step 300, providing sustained learning signals that explain continued late-stage improvement.



(a) Std of σ_{out} (b) Mean of $|\Delta|$

Figure 5: Evolution of learned perturbations during training on Qwen2.5-7B-Instruct. The standard deviation of σ_{out} increases continuously, while the mean perturbation magnitude decreases, reflecting differentiated and increasingly focused exploration.

5.3.3 Noise-Scale Evolution

The noise scales σ_{in} and σ_{out} are learnable parameters updated through the same policy-gradient objective as the backbone. Figure 5 shows that the learned perturbations become differentiated and more focused over training.

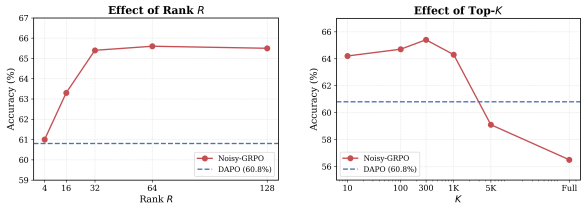
Learned Perturbation Dynamics. The standard deviation of σ_{out} increases 16-fold throughout training (0.001 to 0.016), indicating that individual components learn distinct values based on reward feedback rather than converging to uniform scales. Meanwhile, the mean $|\Delta|$ decreases from 3.1 to 2.4, suggesting an adaptive exploration schedule: larger perturbations are useful early for discovering reward-bearing trajectories, while later training requires less noise as the policy stabilizes and excessive perturbations become less beneficial. Together, these trends show that Noisy-GRPO does not merely inject broad homogeneous noise; it learns differentiated and increasingly focused perturbations.

Token-Level Probe. We record Top-K candidate tokens whose logits receive positive perturbations from the noisy head during training and group them into coarse token types. Table 2 shows a shift from surface-level artifacts to reasoning and transition markers: logic tokens increase from 2.7% to 27.6%, while punctuation decreases from 47.3% to 10.7%. Since the noisy head is optimized by trajectory-level rewards, this pattern suggests that reward feedback selectively reinforces perturbations around tokens that can steer reasoning paths, consistent with prior observations on high-entropy reasoning forks (Wang et al., 2025).

Appendix H provides rank-wise concentration results, full token lists, qualitative word-cloud visualizations, and relative perturbation strength analyses; Appendix I reports per-problem training heatmaps.

Progress	Logic	Glue	Names	Punct.	Other
20%	2.7	4.5	24.4	47.3	21.1
60%	15.4	13.2	19.5	20.5	31.4
100%	27.6	24.3	12.3	10.7	25.1

Table 2: Token-type distribution (%) of Top-K candidates positively upweighted by the noisy head during training. Logic includes reasoning markers such as "Therefore", "Since", and "First".



(a) Effect of Rank R (b) Effect of Top-K

Figure 6: Ablation studies on rank R and Top-K.

5.4 Ablation Studies

We conduct ablation studies on Qwen2.5-7B-Instruct using AMC 2023 accuracy.

Effect of Rank. Performance improves as R increases from 4 to 32 (61.0% to 65.4%), but yields no benefit beyond $R = 32$. This suggests $R = 32$ provides sufficient expressiveness while maintaining efficiency.

Effect of Top-K. Performance peaks at $K = 300$ (65.4%) and degrades substantially for larger values: $K = 5000$ drops to 59.1%, and full vocabulary (152K) to 56.5%—below the DAPO baseline (60.8%). This confirms that perturbations must be restricted to plausible tokens.

Additional mechanism ablations, including the context-gating control and seed sensitivity, are reported in Appendix J.

6 Conclusion

We introduced Noisy-GRPO, a trajectory-consistent exploration method for RLVR training of large language models. Instead of increasing token-level stochasticity through temperature sampling or entropy regularization, Noisy-GRPO samples rollout-level perturbations that remain fixed within a trajectory and vary across rollouts, enabling coherent reasoning exploration while preserving inter-trajectory diversity.

7 Limitations

Our evaluation focuses on mathematical reasoning; effectiveness on other domains remains to be validated. While our method provides trajectory-consistent exploration, it does not explicitly direct exploration toward high-uncertainty regions. The Top-K threshold may require task-specific tuning, and scaling behavior beyond 7B parameters deserves further study.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Weizhe Chen, Zhicheng Zhang, Guanlin Liu, Renjie Zheng, Wenlei Shi, Chen Dun, Zheng Wu, Xing Jin, and Lin Yan. 2024. Flaming-hot initiation with regular execution sampling for large language models. *arXiv preprint arXiv:2410.21236*.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. 2025. Pass@k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*.

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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Onno Eberhard, Jakob Hollenstein, Cristina Pinneri, and Georg Martius. 2023. Pink noise is all you need: Colored noise exploration in deep reinforcement learning. In *International Conference on Learning Representations*.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, and 1 others. 2018. Noisy networks for exploration. In *International Conference on Learning Representations*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021a. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Zi-Wen Hong, Tyler Shann, Shih-Yang Su, Yi-Hsiang Chang, and Chun-Yi Lee. 2018. Diversity-driven exploration strategy for deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Thomas Jaksch, Ronald Ortner, and Peter Auer. 2010. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600.

Yuxian Jiang, Yafu Li, Guanxu Chen, Dongrui Liu, Yu Cheng, and Jing Shao. 2025. Rethinking entropy regularization in large reasoning models. *Preprint*, arXiv:2509.25133.

Michael Kearns and Satinder Singh. 2002. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2):209–232.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and 4 others. 2025. Tulu 3: Pushing frontiers in open language model post-training. *Preprint*, arXiv:2411.15124.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, and 244 others. 2024. Openai o1 system card. *Preprint*, arXiv:2412.16720.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Ian Osband, Daniel Russo, and Benjamin Van Roy. 2013. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, volume 26.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. 2018. Parameter space noise for exploration. In *International Conference on Learning Representations*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Han Shen. 2025. On entropy control in llm-rl algorithms. *Preprint*, arXiv:2509.03493.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, page 1279–1297. ACM.

Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*, 2 edition. The MIT Press.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. 2025. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2024. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

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

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. 2025. Enhancing efficiency and exploration in reinforcement learning for llms. *Preprint*, arXiv:2505.18573.

A A Toy MDP for Trajectory-Consistent Exploration

We provide a simple toy setting to illustrate why temporally correlated perturbations can be more efficient than token-level dithering in tasks where reward depends on coherent multi-step behavior. Consider a horizon-T MDP with binary actions $a_t \in \{0, 1\}$ and a hidden target bit $b \in \{0, 1\}$. The agent receives terminal reward

$$r = 1 \quad \text{iff} \quad a_1 = a_2 = \cdots = a_T = b, \tag{16}$$

and $r = 0$ otherwise. This setting abstracts long-horizon reasoning problems in which only a globally coherent sequence of decisions yields a verifiable reward.

Under token-level dithering, each action is perturbed independently. If the policy has no persistent trajectory-level bias toward either coherent sequence, the probability of sampling a rewarded trajectory is

$$\Pr(r = 1) = 2^{-T}, \tag{17}$$

so the expected number of rollouts required to observe a non-zero reward signal is 2^T. In contrast, a trajectory-consistent perturbation samples a single rollout-level mode and reuses it across the episode. When the sampled mode corresponds to either all-zero or all-one behavior, the probability of producing a coherent sequence is constant with respect to T, up to the probability of selecting the correct hidden bit. Thus, the expected number of rollouts is $O(1)$ in this stylized setting.

This toy example does not claim that all reasoning tasks reduce to binary coherence decisions. Rather, it formalizes the intuition behind our

method: when rewards depend on coordinated decisions across many steps, independent per-token randomness can make useful trajectories exponentially rare, whereas trajectory-consistent exploration preserves coherent behavioral alternatives.

B Seed Replay for Policy-Gradient Updates

Noisy-GRPO treats the trajectory-level seed as part of the stochastic policy used to generate a rollout. For a sampled trajectory $y^{(i)}$, the rollout policy at step t is

$$\pi^{(i)}_{\theta,\phi}(a_t \mid x, a_{<t}) = \mathrm{Softmax}\Big(z_t + \bar{\Delta}(h_t; \varepsilon^{(i)})\Big)_{a_t}. \tag{18}$$

GRPO-style objectives compute token-level importance ratios between the current policy and the old rollout policy. With Noisy-GRPO, this ratio should compare the probability of the sampled action under the same trajectory-level perturbation:

$$\rho^{(i)}_t(\theta, \phi) = \frac{\pi_{\theta,\phi}(a^{(i)}_t \mid x, a^{(i)}_{<t}; \varepsilon^{(i)})}{\pi_{\theta_{\mathrm{old}},\phi_{\mathrm{old}}}(a^{(i)}_t \mid x, a^{(i)}_{<t}; \varepsilon^{(i)})}. \tag{19}$$

If a fresh seed were sampled when recomputing log probabilities during update epochs, the numerator would correspond to a different perturbed policy than the one that produced the trajectory. This would mix policy changes with seed changes and make the importance ratio inconsistent. Therefore, we cache the seed used for each rollout and replay it whenever log probabilities are recomputed for that trajectory.

C Noisy Head Parameter and Compute Overhead

The noisy head introduces two learnable scale tensors, $\sigma_{\mathrm{in}} \in \mathbb{R}^{R\times H}$ and $\sigma_{\mathrm{out}} \in \mathbb{R}^{R\times K}$, where H is the hidden dimension, R is the low-rank dimension, and K is the Top-K perturbation size. Thus, the added parameter count is

$$R(H + K). \tag{20}$$

For Qwen2.5-7B-Instruct, where $H = 4096$, $R = 32$, and $K = 300$, this is

$$32 \times (4096 + 300) = 140{,}672 \tag{21}$$

parameters, about 2.0×10^{-5} of a 7B-parameter backbone.

Per token, the additional computation consists of the low-rank projection $h_t U^{(i)\top}$ and multiplication by $B^{(i)}$, scaling as $\mathcal{O}(HR + RK)$. With the same configuration, this is approximately

$$4096 \times 32 + 32 \times 300 \approx 1.41 \times 10^5 \tag{22}$$

multiply-add operations per token. This is small compared with the standard vocabulary projection cost $\mathcal{O}(H|\mathcal{V}|)$; for $|\mathcal{V}| = 151{,}936$, the projection cost is approximately 6.22×10^8 multiply-add operations per token.

We also measure generation latency under the same decoding setting with and without the noisy head enabled. The average latency is 31.94 ms/-token without the noisy head and 32.06 ms/token with the noisy head, corresponding to a relative overhead of approximately 0.38%. This empirical measurement is consistent with the small additional computation estimated above.

D Training Configuration Details

We adopt the DAPO training configuration with the following hyperparameters:

DAPO Configuration.

- Clip-Higher: $\varepsilon_{\mathrm{high}} = 0.28$, $\varepsilon_{\mathrm{low}} = 0.20$
- Dynamic sampling enabled
- Token-level policy gradient loss
- Overlong reward shaping: maximum response length of 8192 tokens, cache length of 1536 tokens
- No KL divergence penalty or entropy regularization

Noisy Head Configuration. We initialize θ_{in} and θ_{out} to 0.25, use rank $R = 32$, Top-$K = 300$, rollout temperature $T = 0.1$, and a noisy-head learning rate of 1×10^{-3}.

Batch Sizes. Training batch size is 128, rollout batch size is 12, and mini-batch size is 32, resulting in 48 gradient steps per training batch. All experiments use 8 NVIDIA A100 80GB GPUs.

Optimization. We use a constant learning rate of 1×10^{-6} without warmup or scheduling for the backbone model. Unless otherwise specified, rollout sampling uses the temperature reported for each method.

Baselines and Evaluation. The main comparison includes GRPO with rollout temperatures $T \in \{0.1, 1.0, 1.5\}$, DAPO with rollout temperatures $T \in \{0.1, 1.0, 1.5\}$, DAPO with entropy regularization, FIRE, SIREN, Pass@k training, and a token-noise variant that resamples noisy-head perturbations independently at each token. AIME 2024 reports avg@32; all other benchmarks use greedy decoding.

E Prompt Template

We use a simple instruction-following template for mathematical reasoning rollouts and evaluation:

> **User:** Solve the following math problem. Show your reasoning, and put your final answer in `\boxed{}`.
> `{problem}`
> **Assistant:**

For datasets with short-answer verification, we extract the final answer from the last `\boxed{}` expression when present; otherwise, we fall back to the last parseable numerical or symbolic answer.

F Qwen2.5-3B Baseline Comparison

Table 3 reports the Qwen2.5-3B-Instruct comparison between DAPO and Noisy-GRPO. Noisy-GRPO improves the average accuracy from 49.1% to 50.20%, with gains on five of six benchmarks.

Benchmark	DAPO	Noisy-GRPO
GSM8K	87.1	**88.0**
MATH500	69.2	**71.4**
AMC 2023	54.4	**56.5**
Olympiad	31.4	**32.9**
Minerva	**32.7**	32.3
AIME 2024	19.6	**20.1**
Average	49.1	**50.20**

Table 3: Qwen2.5-3B-Instruct comparison across mathematical reasoning benchmarks. All values report accuracy (%); AIME 2024 uses avg@32.

G Length-Controlled Evaluation

Table 4 reports a token-budget matched evaluation on Qwen2.5-7B-Instruct. We set `max_new_tokens=1984` for Noisy-GRPO, matching DAPO's average response length. Noisy-GRPO remains above DAPO on all six benchmarks under this matched generation budget, while the full setting retains additional gains from longer learned reasoning trajectories.

Method	GSM	Min.	M500	AMC	Oly.	AIME
DAPO	91.6	36.1	78.2	60.8	39.2	23.0
Noisy (matched)	92.4	37.7	79.4	63.6	42.3	24.2
Noisy	**92.8**	**38.7**	**80.7**	**65.4**	**43.1**	**25.4**

Table 4: Length-controlled evaluation. "Noisy" denotes Noisy-GRPO. The matched setting uses `max_new_tokens=1984` to match DAPO's average response length. All values report accuracy (%); AIME 2024 uses avg@32.

H Additional Learned Perturbation Analyses

We provide additional analyses of the learned perturbations. These probes are intended as interpretability diagnostics rather than causal evidence.

Direction Concentration. Because the noisy head is implemented as a low-rank context-gated perturbation, each rank component can be viewed as a reusable perturbation direction. We measure whether perturbation energy concentrates into a smaller set of influential directions. For each rank component r, we compute its average squared perturbation energy E_r and define

$$S(8) = \frac{\sum_{r \in \text{Top-8}} E_r}{\sum_{r=1}^{R} E_r}. \tag{23}$$

Table 5 shows that $S(8)$ steadily increases over training, suggesting that Noisy-GRPO learns increasingly selective exploration directions.

Training Progress	20%	40%	60%	80%	100%
$S(8)$	0.28	0.33	0.41	0.47	0.52

Table 5: Rank-wise concentration of perturbation energy. Higher values indicate that perturbations are increasingly dominated by a small subset of rank components.

Upweighted Tokens. Table 6 lists the top tokens that receive positive perturbations from the noisy head at different training checkpoints. Early in training, the upweighted candidates include many surface tokens such as names and punctuation. Later checkpoints increasingly emphasize reasoning and transition markers.

Token-Type Distribution. Table 7 reports the full checkpoint-wise token-type distribution corresponding to the compressed version shown in Table 2. The distribution shifts away from names and punctuation and toward logic and connective tokens over training.

Progress	Top Upweighted Tokens
20%	James, Mrs, #, 1, base
40%	track, based, we, that, makes
60%	Calcul, The, If, Since, to
80%	Now, that, Therefore, According
100%	Therefore, Since, According, After, First

Table 6: Representative tokens whose logits receive the largest positive perturbations from the noisy head during training.

Progress	Logic	Glue	Names	Punct.	Other
20%	2.7	4.5	24.4	47.3	21.1
40%	8.4	12.6	22.5	23.5	33.0
60%	15.4	13.2	19.5	20.5	31.4
80%	20.3	19.4	16.5	14.4	29.4
100%	27.6	24.3	12.3	10.7	25.1

Table 7: Token-type distribution (%) of Top-K candidates positively upweighted by the noisy head during training.

Word-Cloud Visualization. Figure 7 provides a qualitative visualization of the same token-level probe. The color categories follow Table 7, and the word sizes reflect positive perturbation strength. The visualization makes the same trend visible: early perturbations are concentrated on surface tokens, while late perturbations more frequently emphasize logic and connective tokens.

Additional Training Sets. We will include noise-scale dynamics across additional training sets in this appendix.

Relative Perturbation Strength. Absolute perturbation magnitudes are difficult to interpret without considering the scale of the base logits. We therefore define the relative perturbation strength

$$\rho_t = \frac{\mathbb{E}[|\Delta_t|]}{\mathrm{std}(z_{t,\text{Top-}K})}, \tag{24}$$

where the denominator is the standard deviation of the base logits within the Top-K candidate set at step t. Intuitively, $\rho_t \approx 1$ means that the learned perturbation is comparable to the natural logit variation among plausible candidates: it can affect close ranking decisions without fully overriding high-confidence base-policy preferences.

Figure 8 shows that ρ_t remains within the diagnostic band $[0.8, 1.2]$ throughout training. This suggests that the learned noise scale stays in a moderate regime: too small a value would be largely absorbed by the softmax, while too large a value could overwhelm the base ranking and approach high-temperature behavior within the Top-K set. The numerator and denominator remain on the same order of magnitude, providing a scale-based explanation for the initialization $\theta_{\text{in}} = \theta_{\text{out}} = 0.25$ and for the adaptive perturbation dynamics observed in Section 5.3.3.

I Per-Problem Learning Stability

J Additional Ablation Studies

Gating Mechanism. Table 8 reports the effect of replacing hidden-state gating with a fixed random vector. Random-gated noise achieves only 59.7%, below DAPO (60.8%), while context-gated perturbations reach 65.4%. This indicates that trajectory consistency alone is insufficient; perturbations must be modulated by hidden representations.

Method	Accuracy (%)
DAPO	60.8
Noisy-GRPO w. Random-Gated	59.7
Noisy-GRPO (Context-Gated)	**65.4**

Table 8: Effect of gating mechanism on AMC 2023.

Noisy Head Hyperparameter Sensitivity. We provide sensitivity analysis of key hyperparameters in Section 5.4. Default values are: initialization $\theta_{\text{in}} = \theta_{\text{out}} = 0.25$, rank $R = 32$, and Top-K $= 300$.

Seed Sensitivity. We additionally evaluate Noisy-GRPO across five random seeds on AMC 2023. Figure 10 shows that performance remains stable, with an average accuracy of 65.36% and a cross-seed standard deviation of 0.36, while consistently outperforming the DAPO baseline.

K Core Implementation

We summarize the core changes used to add Noisy-GRPO to the verl/vLLM training stack. Unrelated boilerplate, logging, and error handling are omitted for clarity.

Trajectory Seed Propagation. Each rollout is assigned a persistent `sample_id`, which is passed to vLLM through `SamplingParams.extra_args`. The same identifier is later used to regenerate trajectory-level noise.

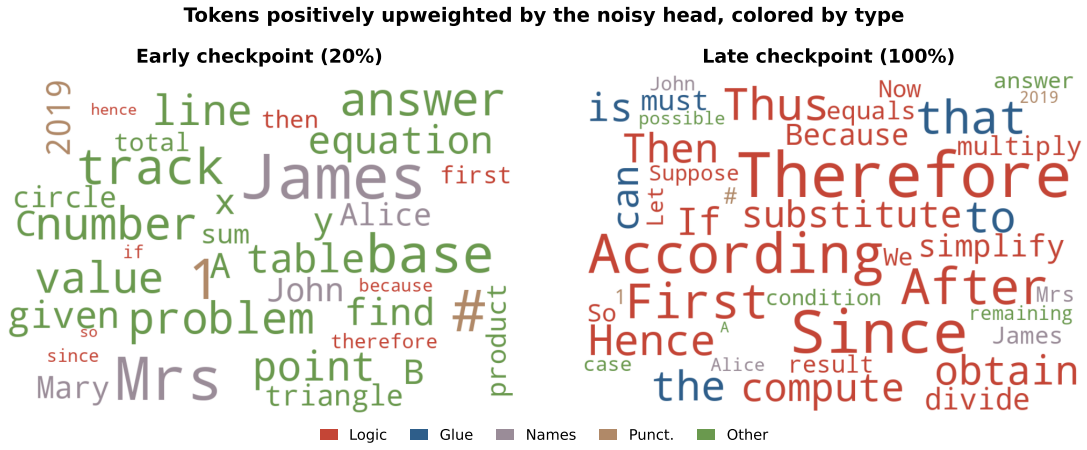


Figure 7: Qualitative word-cloud visualization of tokens positively upweighted by the noisy head at early and late training checkpoints. Colors indicate token categories, and quantitative category distributions are reported in Table 7.

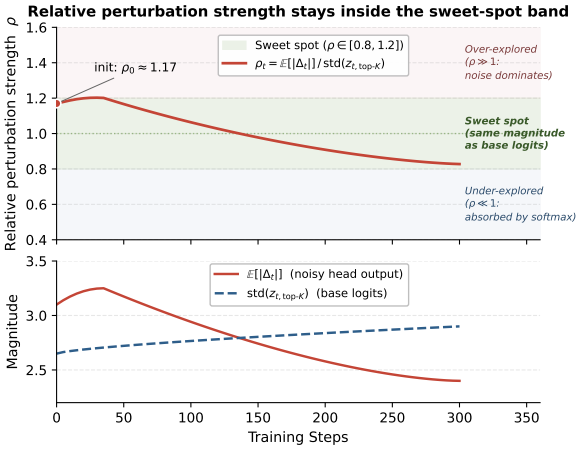


Figure 8: Relative perturbation strength during training. The ratio $\rho_t = \mathbb{E}[|\Delta_t|]/\mathrm{std}(z_{t,\text{Top-}K})$ stays in a moderate diagnostic band, indicating that the learned perturbations remain comparable to the natural scale of Top-K base logits.

Noisy Residual Head. The noisy head stores two learnable scale tensors and deterministically regenerates input- and output-side noise from the trajectory seed. The hidden state gates the low-rank perturbation directions.

Top-K Logit Perturbation in vLLM. During rollout generation, vLLM computes base logits as usual. Noisy-GRPO then adds the learned residual only to the Top-K candidates selected from the base logits.

Training and Rollout Integration. The noisy head is inserted into the FSDP actor before wrapping, optimized with the actor parameters, and mirrored into the vLLM rollout engine. Because vLLM does not natively load this extra module, its weights are synchronized manually.

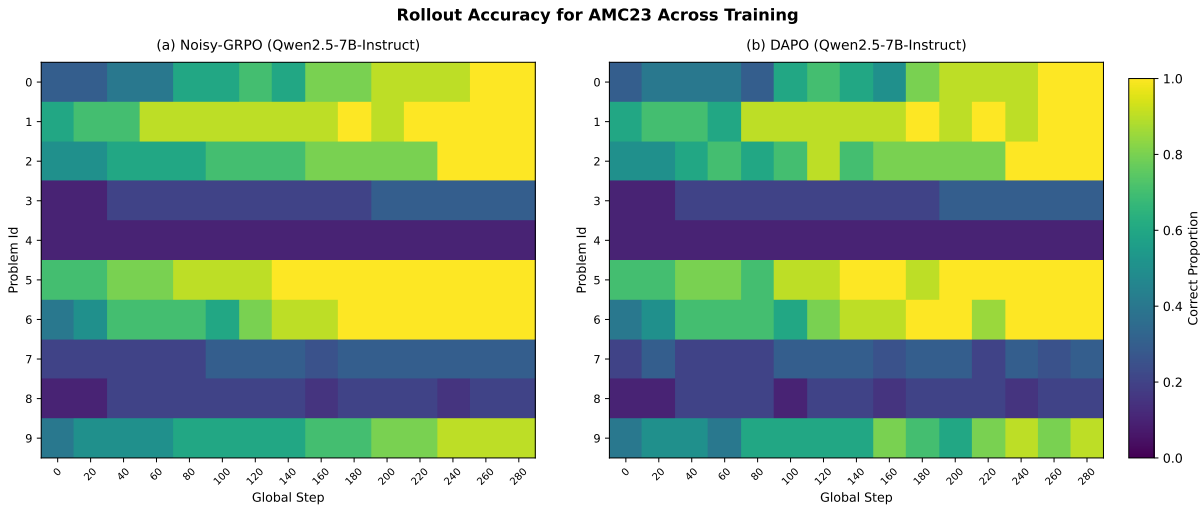


Figure 9: Per-problem rollout accuracy on AMC 2023 across training steps. Noisy-GRPO shows a more consistent transition from low to high accuracy, while DAPO exhibits more frequent regressions and fluctuations.

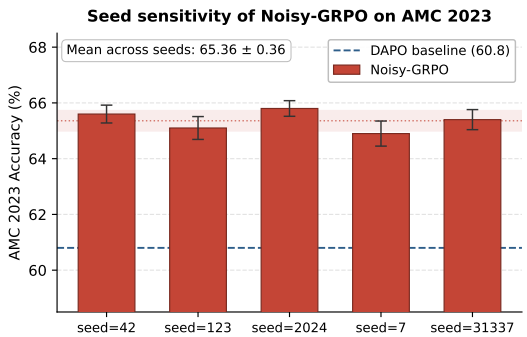


Figure 10: Seed sensitivity of Noisy-GRPO on AMC 2023.

```
+ # ray_trainer.py: assign rollout-level identifiers.
+ gen_batch.non_tensor_batch["sample_id"] = np.arange(B, dtype=np.int64)
+
+ # vllm_rollout_spmd.py: attach one id to each request.
+ sample_ids = prompts.non_tensor_batch.get("sample_id", None)
+ for request, sample_id in zip(vllm_inputs, sample_ids):
+ sp = copy.deepcopy(self.sampling_params)
+ sp.extra_args = int(sample_id)
+ request["sampling_params"] = sp
```

Figure 11: Trajectory-level seed propagation.

```
+ class NoisyResidualHead(nn.Module):
+ self.theta_in = nn.Parameter(full((R, H), init_sigma))
+ self.theta_out = nn.Parameter(full((R, K), init_sigma))
+
+ def _generate_noise(self, shape, seed, device, dtype):
+ g = torch.Generator(device=device).manual_seed(int(seed))
+ return torch.randn(*shape, generator=g, device=device, dtype=dtype)
+
+ def compute_delta(self, hidden_states, sample_id):
+ eps_in = stack([_generate_noise((R,H), sid) for sid in sample_id])
+ eps_out = stack([_generate_noise((R,K), sid + 1) for sid in sample_id])
+ u = softplus(self.theta_in)[None] * phi(eps_in)
+ b = softplus(self.theta_out)[None] * phi(eps_out)
+ a = einsum("blh,brh->blr", hidden_states, u) / sqrt(H)
+ return einsum("blr,brk->blk", a, b) / sqrt(R)
```

Figure 12: Low-rank noisy residual head.

```
+ # gpu_model_runner.py: after base logits are computed.
+ if hasattr(self.model, "noisy_head") and self.model.noisy_head is not None:
+ sample_ids = []
+ for req_idx in req_idx_for_rows.tolist():
+ rid = self.input_batch.req_ids[req_idx]
+ sample_ids.append(self.requests[rid].sampling_params.extra_args)
+
+ delta = self.model.noisy_head.compute_delta(sample_hidden_states, sample_ids)
+ _, top_k_indices = logits.topk(delta.shape[-1], dim=-1)
+ logits.scatter_add_(dim=-1, index=top_k_indices, src=delta)
```

Figure 13: Top-K logit perturbation in vLLM.

```
+ # fsdp_workers.py: add noisy_head before FSDP wrapping.
+ if role == "actor" and config.model.get("enable_noisy_head", False):
+ actor_module.add_module("noisy_head", NoisyResidualHead(H,V,R,init))
+ actor_module.noisy_head.float()
+
+ # Include noisy-head parameters in optimization.
+ noisy_params = list(underlying.noisy_head.parameters())
+ param_groups.append({"params": noisy_params, "lr": optim_config.lr})
+
+ # vllm_rollout_spmd.py: mirror the module into rollout engine.
+ model.noisy_head = NoisyResidualHead(H,V,R,init)
+
+ # During weight sync, copy noisy_head.theta_in/out separately.
+ model.noisy_head.theta_in.copy_(noisy_sd["noisy_head.theta_in"])
+ model.noisy_head.theta_out.copy_(noisy_sd["noisy_head.theta_out"])
```

Figure 14: Training and rollout integration.