ReDit: Reward Dithering for Improved LLM Policy Optimization

Chenxing Wei$^{\dagger\S\natural}$**, Jiarui Yu**$^{\circ}$**, Ying He**†**, Hande Dong**$^{\circ}$**, Yao Shu**$^{\wr *}$**, F. Richard Yu**‡
§Guangdong Lab of AI and Digital Economy (SZ), China
†College of Computer Science and Software Engineering, Shenzhen University, China
$^{\wr}$Hong Kong University of Science and Technology (Guangzhou), China
$^{\circ}$ Tencent, Shenzhen, China
‡School of Information Technology, Carleton University, Canada
weichenxing2023@email.szu.edu.cn, yaoshu@hkust-gz.edu.cn

Abstract

DeepSeek-R1 has successfully enhanced Large Language Models (LLMs) reasoning capabilities through its rule-based reward system. While it's a "perfect" reward system that effectively mitigates reward hacking, such reward functions are often discrete. Our experimental observations suggest that discrete rewards can lead to gradient anomaly, unstable optimization, and slow convergence. To address this issue, we propose ReDit (Reward Dithering), a method that dithers the discrete reward signal by adding simple random noise. With this perturbed reward, exploratory gradients are continuously provided throughout the learning process, enabling smoother gradient updates and accelerating convergence. The injected noise also introduces stochasticity into flat reward regions, encouraging the model to explore novel policies and escape local optima. Experiments across diverse tasks and different LLMs demonstrate the effectiveness and efficiency of ReDit. On average, ReDit achieves performance comparable to vanilla GRPO with only approximately 10% the training steps, and furthermore, still exhibits a 4% performance improvement over vanilla GRPO when trained for a similar duration. Visualizations confirm significant mitigation of gradient issues with ReDit. Moreover, theoretical analyses are provided to further validate these advantages.

1 Introduction

Reinforcement learning (RL) is pivotal in Large Language Models (LLMs) development [1, 2, 3, 4]. Initially, RL from human feedback (RLHF) [5, 6] was employed to align pre-trained LLMs with human preferences [7, 8]. This typically involves training a separate reward model (RM) on human preference data [9], which then guides the LLM policy optimization [10]. While effective, this approach introduces considerable training overhead [11]. Subsequently, methods like Direct Preference Optimization (DPO) [12, 13] were developed, enabling LLMs to learn directly from preference data and thus bypassing explicit RM training. However, these methods still require extensive collection of high-quality preference data. For reasoning tasks such as mathematics and coding, DeepSeek-R1 [14] with Group Relative Policy Optimization [15](GRPO) proposes an alternative: optimizing the LLM policy directly using a rule-based reward system [16, 17], thereby avoiding the need for external RMs or large preference datasets. For instance, such a system might assign a reward of 1 for outputs meeting predefined criteria (e.g., correctness, format compliance) and 0 otherwise [14]. The simplicity and unbiased nature of these rule-based rewards prevent LLMs from hacking them, potentially fostering enhanced reasoning capabilities [18].

*corresponding author. $^\natural$ Work done during an internship at Tencent.

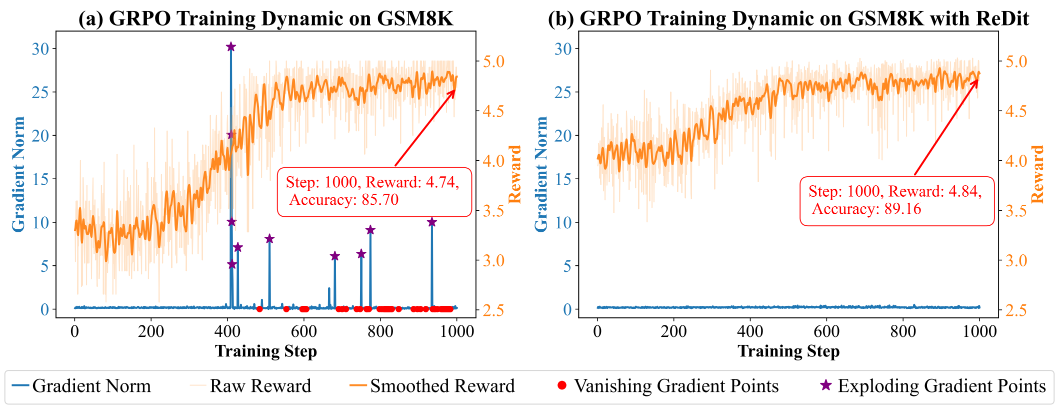

Figure 1: Training Dynamics of Gradient Norm and Reward for Qwen2.5-7B [31] on GSM8K [32] Dataset. The left and right figures compare original gradient norm (before gradient clipping [33]) and reward trends across training steps. The original GRPO method (the left figure) suffers from significant gradient instability—both vanishing (red dots, norms < 0.01) and exploding (purple asterisks, norms > 5). In contrast, ReDit with Gaussian reward smoothing (the right figure) effectively stabilizes optimization throughout training.

However, such reward functions are often discrete, posing significant optimization challenges [19, 20, 21]. Consider an RL scenario with a binary reward [22]: a policy model receives 1 for a correct answer and 0 otherwise. During early training phases, a policy LLM rarely generates completely correct answers, resulting in predominantly zero rewards across mini-batches [23]. Although the model may engage in exploratory behavior on difficult examples, the corresponding gradients remain minimal due to small advantage magnitudes [24]. Thus, these hard examples and potentially beneficial explorations [24] are largely unexploited during the early stages. Conversely, the model may repeatedly reinforce easy examples [25], thus reducing incentives to explore alternative strategies for more difficult problems [26]. This phenomenon can lead to training stagnation in intermediate and advanced stages. Consistent with this, as shown in Fig. 1 (the left figure), we observe that the policy model frequently suffers from gradient vanishing [27, 28] or explosion [29, 30] during these phases. This combination of insufficient exploration and gradient instability substantially impedes model convergence, representing a critical obstacle to efficient RL in LLMs.

This observed phenomenon highlights that even perfectly accurate discrete reward functions face significant limitations within gradient-based optimization frameworks. Lending theoretical support to this, recent studies [34, 35, 36] have established that a singular focus on increasing reward model accuracy does not necessarily translate to enhanced language model performance. In particular, [36] theoretically substantiates the necessity for effective reward models to integrate adequate variance and uncertainty to enable efficient optimization. The theoretical details are given in Sec. 3.2. Consequently, we believe that an excellent reward system should achieve a careful equilibrium between correctness and sufficient variance.

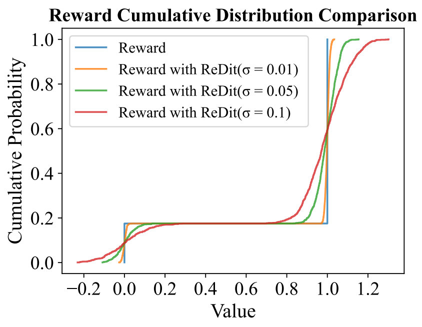

Figure 2: The figure illustrates how ReDit of different variances gradually smooth the reward distribution, showing the smoothing effect of perturbations of different variances on the reward distribution.

Inspired by these observations and theoretical insights, we propose ReDit, a simple but effective technique that introduces zero-mean random perturbations to discrete reward signals during training. By adding small random noise to the reward function (Fig. 2), ReDit effectively softens the original hard reward function. Compared to a hard reward function, a softened one can provide greater reward variance within mini-batches, which, as indicated by previous research, can lead to enhanced model performance and accelerated convergence.

Fig. 1 (the right figure) illustrates the impact of our proposed ReDit on LLM policy optimization for the GSM8K [32] task. The orange lines indicate that during early training phases, GRPO with ReDit achieves significantly higher average rewards compared to the baseline (GRPO without ReDit), demonstrating the efficacy of our method. We hypothesize that ReDit encourages broader exploration by assigning varied rewards to outputs that only partially meet strict evaluation criteria, thereby accelerating convergence. Towards the end of training (1000 steps), while both policy models attain high rewards, our approach with ReDit exhibits superior performance on the test set, indicating enhanced generalization. Additionally, ReDit demonstrates more robust gradient updating. As shown in Fig. 1 (the left figure), phenomena such as gradient vanishing (red point) and explosion (purple star) emerge during training with the baseline. In contrast, GRPO with ReDit maintains stable gradients throughout the training process. These findings highlight the advantages of ReDit: more stable policy optimization, faster convergence, and improved overall performance.

Moreover, theoretical analysis indicates that a greater reward variance can enhance performance and accelerate convergence in reinforcement learning [36]. We increase reward variance within mini-batches while preserving the expected gradient through reward dithering. By carefully injecting noise into the reward function, ReDit achieves a balance between reward signal fidelity and reward variance, leading to enhanced policy optimization.

In summary, our main contributions are:

- We observe that policy optimization under discrete reward functions suffer from unstable gradients and slow convergence(Section 3.1).
- We propose Reward Dithering (ReDit), a simple yet effective technique that introduces perturbations to discrete rewards. This method is shown to accelerate convergence speed and enhance final model performance (Algorithm 1 and Section 4).
- Extensive experiments across diverse downstream tasks, reinforcement learning algorithms, and perturbation distributions demonstrate that ReDit achieves superior performance and enhanced convergence (Section 5).
- Theoretical analysis proves that ReDit produces an unbiased estimate of the original gradient (Proposition 1) and introduces beneficial gradient variance that mitigates vanishing and exploding gradients (Proposition 2).

2 Preliminaries

We frame LLM generation as a sequential decision-making problem solvable via RL. The process is modeled as a Markov Decision Process (MDP) [37] where the state $s_t = q; o_{<t}$ includes the prompt q and generated tokens $o_{<t}$, the action o_t is the next token selected from the vocabulary, and the policy $\pi_\theta(o_t|s_t)$ is parameterized by θ. The goal is to optimize the policy to maximize the expected sequence-level reward $R(q, o) = \sum_{t=1}^{|o|} r(s_t, o_t)$ over the prompt distribution p_Q:

$$J(\pi_\theta) = \mathbb{E}_{q\sim p_Q}\left[\mathbb{E}_{o\sim\pi_\theta(\cdot|q)}[R(q, o)]\right]. \tag{1}$$

Recently, GRPO [15] was proposed as a PPO alternative that eliminates the need for independent RMs and value functions. GRPO typically processes sparse, discrete rewards directly, rather than continuous RM scores. For tasks like mathematical reasoning, this discrete reward $R(q, o) \in \{0, 1\}$ is often determined by a simple function checking correctness or format. GRPO estimates the advantage $\hat{A}^{\text{GRPO}}_{i,t}$ by sampling G responses $\{o_i\}_{i=1}^{G}$ and normalizing their discrete rewards within the set. Its objective function, which includes a KL divergence term $D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})$ for stability, is given by:

$$\begin{aligned} J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim p_Q}&\left[\frac{1}{G}\sum_{i=1}^{G}\sum_{t=1}^{|o_i|}\min\left(r_{i,t}(\theta)\hat{A}^{\text{GRPO}}_{i,t}, \text{clip}\left(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}^{\text{GRPO}}_{i,t}\right)\right] \\ &-\beta\mathbb{E}_{q\sim p_Q}\left[D_{\text{KL}}(\pi_\theta(\cdot|q)||\pi_{\text{ref}}(\cdot|q))\right], \end{aligned} \tag{2}$$

where $r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|s_{i,t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|s_{i,t})}$. Subsequent methods such as DAPO [38], Dr.GRPO [39], and REINFORCE++ [40] generally adopt this discrete reward paradigm (see Appendix A for more related

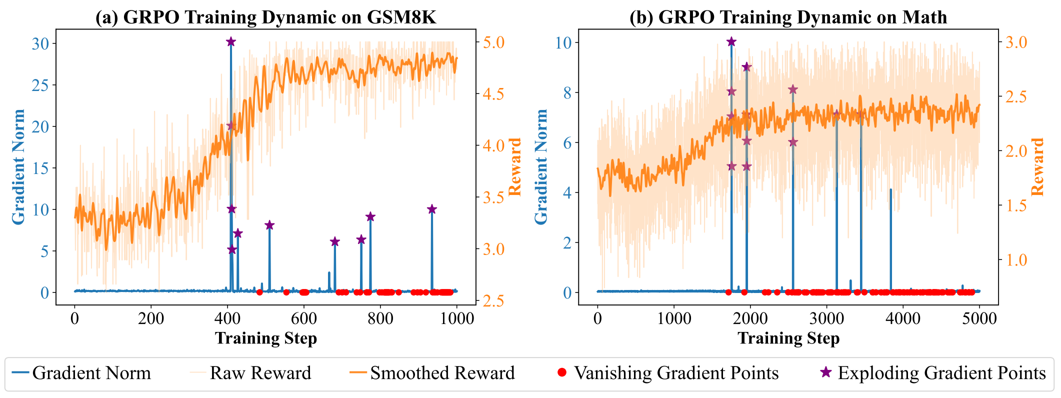

Figure 3: Qwen2.5-7B [31] Gradient norm and reward training dynamics of standard GRPO on GSM8k and MATH datasets. During the whole optimization process, the gradient of standard GRPO is unstable, and there are a lot of gradient vanishing or gradient exploding cases.

work). While simplifying the overall RL process by avoiding complex RMs, this shift to discrete, sequence-level rewards introduces significant optimization challenges. The inherent sparsity and abrupt value changes (e.g., 0 to 1) hinder policy gradient estimation and lead to training instability (Section 3.1).

3 Motivation

This section articulates the fundamental motivations driving our research and establishes the critical challenges that our work aims to address. In Section 3.1, we examine the optimization challenges inherent in discrete reward structures, followed by an exposition of the theoretical principles informing our methodological framework in Section 3.2.

3.1 Difficulties in Optimization Caused by Discrete Rewards

Optimizing LLM policies using algorithms like GRPO in conjunction with discrete sequence-level rewards (e.g., binary correctness metrics) presents significant optimization challenges. Fig. 3 plots the policy gradient norm (blue line) and average reward (orange line) during standard GRPO training on the GSM8K and MATH datasets, respectively. Two main issues are immediately apparent: **Gradient Vanishing.** The figure illustrates instances where the gradient norm approaches zero (red dot), occurring when most examples in a GRPO batch yield identical binary rewards. Consequently, the population relative advantage estimate $\hat{A}_{i,t}^{\text{GRPO}}$ becomes negligible across examples, providing insufficient learning signals and causing training stagnation. This phenomenon is evident in Fig. 3(the right figure) post-step 2000.

Gradient Explosion. Conversely, training dynamics exhibit sporadic sharp spikes in gradient norm (purple asterisks) when small policy changes cause sequences to transition from incorrect (reward 0) to correct (reward 1). These transitions create disproportionately large advantage estimates for newly successful sequences, triggering sudden, destabilizing gradient updates as shown in Fig. 3(the left figure). Such spikes induce reward fluctuations in subsequent steps, hindering smooth convergence and learning efficiency.

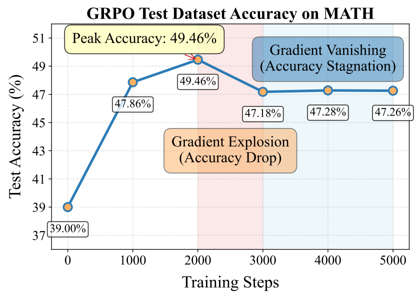

Figure 4: GRPO has unstable performance on the MATH test set. The figure plots the test accuracy achieved for the checkpoints saved during the training run shown in Fig. 3(the right figure).

The discrete, sparse rewards induce unstable oscillations between vanishing and exploding gradients. Fig. 4 demonstrates that model performance fluctuates correspondingly with these oscillations. This inherent instability not only compromises optimization efficiency but also serves as a key motivation for our research.

3.2 Theoretical Principles to Address the Limitations of Discrete Rewards

To overcome the critical challenges with discrete rewards outlined in Section 3.1, we propose a direct approach to enhance reward signal quality. Our solution derives from key theoretical insights in policy optimization, particularly Theorem 1 and Theorem 2, which reveal fundamental relationships between reward variance, model accuracy, and learning efficiency. See Appendix B.1 for definitions.

Theorem 1 (Policy network optimization time lower bound)**.** *From Theorem 1 in [41]. Suppose that we maximize the objective (Eq.* (1)*), using a general autoregressive policy* $\pi_\theta(\mathbf{y}|\mathbf{x}) = \prod_{l=1}^{\mathbf{y}} \mathrm{softmax}(f_\theta(\mathbf{x}, \mathbf{y}_{<l})_{\mathbf{y}_l})$*. For any* $\gamma > 0$*, prompt* $\mathbf{x} \in \mathcal{X}$*, and reward function* r*, the time it takes until* $\mathbb{E}_{\mathbf{y}\sim\pi_{\theta(t)}(\cdot|\mathbf{x})}[r(\mathbf{x},\mathbf{y})] \geq \mathbb{E}_{\mathbf{y}\sim\pi_{\theta(0)}(\cdot|\mathbf{x})}[r(\mathbf{x},\mathbf{y})] + \gamma$ *is:*

$$\Omega\left(\mathbb{E}_{\mathbf{x}'\sim S}\left[\mathrm{var}_{\mathbf{y}\sim\pi_{\theta(0)}(\cdot|\mathbf{x}')}(r(\mathbf{x}',\mathbf{y}))\right]^{-\frac{1}{3}}\right) \tag{3}$$

The reward variance is: $\mathrm{var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r(\mathbf{x},\mathbf{y})] := \mathbb{E}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[[r(\mathbf{x},\mathbf{y}) - \mathbb{E}_{\mathbf{y}'\sim\pi_\theta(\cdot|\mathbf{x})}[r(\mathbf{x},\mathbf{y}')]]^2]$.

Theorem 1 establishes that the time $t_\gamma$required for policy improvement is inversely proportional to reward variance. When rewards exhibit insufficient variance—failing to adequately differentiate between high-quality and low-quality outputs under policy π_θ, convergence slows significantly. This finding suggests that strategically increasing reward variance can accelerate policy convergence.

Theorem 2 (Policy network optimization time upper bound)**.** *From Theorem 2 in [41]. Assume* π_θ *is a policy of the form* $\pi_\theta(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}[\theta_{:,\mathbf{x}}]_{\mathbf{y}}$*. Given a prompt* $\mathbf{x} \in S$*, let* $\gamma > 0$ *and denote by* $t_\gamma > 0$ *the initial time of* $\mathbb{E}_{\mathbf{y}\sim\pi_{\theta(t)}(\cdot|\mathbf{x})}[r_{\mathrm{G}}(\mathbf{x},\mathbf{y})] \geq \mathbb{E}_{\mathbf{y}\sim\pi_{\theta(0)}(\cdot|\mathbf{x})}[r_{\mathrm{G}}(\mathbf{x},\mathbf{y})] + \gamma$*. For any initial policy* $\pi_{\theta(0)}$*, a perfect RM converges to* t_γ *that can be arbitrarily large, while a relatively inaccurate RM has an upper bound of* $\mathcal{O}(\pi_{\theta(0)}(y^\gamma|\mathbf{x})^{-1})$*.*

Complementarily, Theorem 2 demonstrates that effective reward models must incorporate a calibrated degree of uncertainty. This controlled uncertainty creates essential exploration space during early training stages, preventing premature convergence and facilitating more efficient optimization.

While perfectly accurate reward functions resist reward hacking, they paradoxically impede optimization by producing discrete rewards with minimal variance and insufficient randomness. This limitation severely constrains the growth rates of both training reward r_{RM} and true reward r_G during policy gradient updates. To address this fundamental tension, we introduce ReDit—a method that injects zero-mean perturbations into discrete rewards. This approach preserves the expected reward value while introducing beneficial variance and controlled uncertainty in each update step, dramatically improving both model performance and convergence speed.

4 Reward Dithering (ReDit)

As discussed previously, the discrete nature of rewards commonly used in GRPO can lead to unstable gradient dynamics. To address this, we propose **ReDit** . The core idea, detailed in Algorithm 1, is to inject calibrated, zero-mean perturbations into the discrete rewards obtained from sampled outputs before using them to compute the GRPO objective for policy updates. Importantly, our ReDit method preserves the overall optimization structure of the GRPO objective function as defined in Eq. (2), the optimization still aims to maximize this objective.

The crucial modification introduced by ReDit lies in *how* the advantage term $\hat{A}_{i,t}^{\mathrm{GRPO}}$ within Eq. (2) is computed. Instead of directly using the raw discrete rewards $r_i = r(o_i)$ obtained for each sampled output o_i in the batch $\{o_i\}_{i=1}^G$ (line 3 in Algorithm 1), we first compute **smoothed rewards** $\tilde{r}_i$. This is done by adding independently sampled zero-mean perturbation ϵ_i (e.g., from $\mathcal{N}(0,\sigma^2)$ or $\mathcal{U}[-a,a]$) to each discrete reward (line 4 in Algorithm 1):

$$\tilde{r}_i = r_i + \epsilon_i \tag{4}$$

These smoothed rewards $\{\tilde{r}_k\}_{k=1}^G$ are then used as the basis for calculating the advantage. GRPO often computes advantage based on the relative performance within the batch, typically involving normalization. With ReDit, the core component of the advantage calculation, which relies on these rewards, is effectively modified as follows:

$$\hat{A}_{i,t}^{\text{GRPO}} \propto \frac{r_i - \text{mean}(\{r_k\}_{k=1}^G)}{\text{std}(\{r_k\}_{k=1}^G)} \quad \xrightarrow{\text{ReDit}} \quad \hat{A}_{i,t}^{\text{Dithering}} \propto \frac{\tilde{r}_i - \text{mean}(\{\tilde{r}_k\}_{k=1}^G)}{\text{std}(\{\tilde{r}_k\}_{k=1}^G)} \tag{5}$$

Thus, the relative standing of each output o_i within the batch, which informs its advantage $\hat{A}_{i,t}^{\text{GRPO}}$ used in Eq. (2), is determined by the continuous smoothed reward $\tilde{r}_i$ rather than the discrete r_i. This substitution transforms the optimization landscape. By introducing continuous variations via $\tilde{r}_i$, the added noise provides informative, non-zero gradients even when discrete rewards r_i are sparse or identical within a batch, mitigating gradient vanishing. It also dampens the sharp changes in expected advantage resulting from small policy shifts affecting discrete outcomes, thus reducing the likelihood of gradient explosion. This overall smoothing effect facilitates a more stable gradient flow, enabling more robust and efficient optimization of the policy π_θ using the GRPO objective (line 5 in Algorithm 1).

Algorithm 1 ReDit within one optimization step

```
1: Input: Base policy $\pi_{\theta_{\text{old}}}$; Discrete reward function $r : \mathcal{O} \to \{0, 1, 2, 3, ...\}$; Prompt $q$; Number
   of samples $G$. Noise parameters: Gaussian std dev $\sigma > 0$ or Uniform radius $a > 0$.
2: Output: Updated policy $\pi_\theta$.
3: Sample $G$ outputs $\{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$ and compute $r_i \leftarrow r(o_i)$ for $i = 1, \ldots, G$.
4: Sample $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ or $\mathcal{U}[-a, a]$ and compute $\tilde{r}_i \leftarrow r_i + \epsilon_i$ for $i = 1, \ldots, G$.// Generate
   noise and smooth rewards.
5: Compute $J_{\text{GRPO}}$ using $\{\tilde{r}_i\}_{i=1}^G$ and $\theta \leftarrow \text{Optimize}(\theta_{\text{old}}, J_{\text{GRPO}}, \tilde{r}_i)$.// Optimization
6: return Updated policy $\pi_\theta$.
```

5 Empirical Results

This section presents a thorough evaluation of our ReDit framework, assessing its effectiveness and efficiency. We begin by detailing the datasets and experimental configurations in Section 5.1. Subsequently, Section 5.2 provides a comprehensive analysis of the primary findings. To isolate the contributions of key components, we also conduct ablation studies, the results of which are presented in Section 5.3.

5.1 Datasets and Setup

To rigorously evaluate the effectiveness of our proposed ReDit framework, we conducted extensive experiments. The specific experimental settings are detailed below.

Table 1: Comparison of the mean and variance of accuracy for different baselines under 9000 steps on GSM8K.

Name	DAPO	DR.GRPO	REINFORCE++
Baseline	87.52	86.13	86.25
w/ **ours(Gauss)**	**89.34** ($\pm$ 0.04)	**87.69** ($\pm$ 0.06)	**87.96** ($\pm$ 0.03)
w/ **ours(Uniform)**	88.57 ($\pm$ 0.01)	87.34 ($\pm$ 0.07)	87.59 ($\pm$ 0.09)
Δ	+1.82	+1.56	+1.71

Datasets. Our dataset selection and setup largely follow the methodology of [15], primarily to assess the mathematical reasoning capabilities of the models. This encompasses mathematical problem-solving datasets such as GSM8K [32] and MATH [42], as well as the multimodal geometric reasoning dataset Geometry3K [43]. Each dataset provides distinct training and test splits, which we utilize accordingly for model training and subsequent evaluation. See the Appendix D.1 for details of the dataset.

Reward Functions. We designed dataset-specific reward functions. For the GSM8K dataset, which involves simpler problem structures, we implemented several reward types: accuracy-based, strict format adherence, sort format adherence, integer value correctness, and inference step adherence.

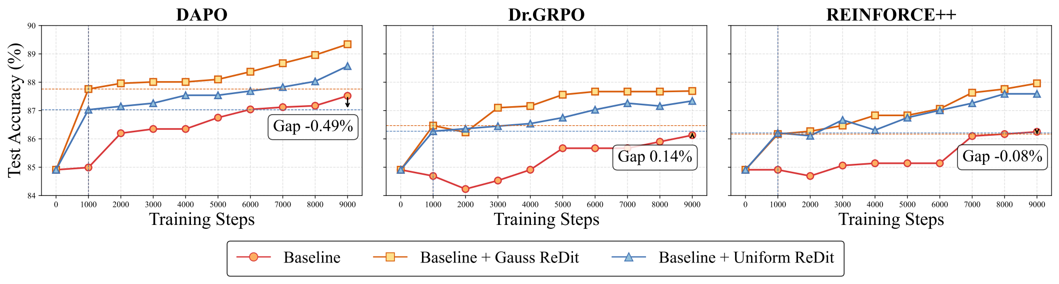

Figure 5: Accuracy of different GRPO variants (DAPO, DR.GRPO, REINFORCE++) tested on the GSM8K dataset. The horizontal dashed line highlights the performance of using ReDit at about 1000 training steps, and even after 9000 steps, its accuracy is comparable to the baseline.

For the more complex MATH and Geometry3K datasets, our supervision relied solely on accuracy-based and inference-based reward functions. Detailed implementations of these reward functions are provided in the Appendix D.2.

Initial Policy. To rigorously assess the effectiveness of ReDit without confounding factors introduced by supervised fine-tuning (SFT), we initialized our experiments directly with instruct models without any additional SFT training. Previous research by [44] demonstrated that even random rewards can enhance performance for Qwen models. Therefore, we conducted comprehensive evaluations across a diverse set of instruction-tuned models, including Qwen2.5-7B-Instruct, Qwen2.5-VL-7B-Instruct [31], Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct [1], Ministral-8B-Instruct-2410, and Mistral-7B-Instruct-v0.3 [45], to establish the generalizability of ReDit.

Random Seeds. Our method incorporates random sampling, which can introduce variance to the experimental outcomes. To thoroughly assess the impact of this stochasticity and ensure the robustness of our results, we executed all main experiments using multiple distinct random seeds. Specifically, we selected five seeds: None (no seed), 42, 123, 888, 2025, and 9999. All metrics reported in our final results represent the mean and variance computed across these five independent runs.

Other Training Settings. For parameter-efficient fine-tuning, we employed Low-Rank Adaptation (LoRA) [46]. Our implementation leverages the official GRPO implementation within the TRL library [47]. Specific configurations for LoRA and GRPO parameters are detailed in the Appendix D.3. Model evaluation was conducted using the OpenCompass [48]. All experiments were executed on one NVIDIA H20 GPU.

5.2 Main Results

In our main experiments, we validate the effectiveness of our proposed ReDit. For these experiments, we primarily use either a uniform smoothing kernel with radius $a = 0.05$ or a Gaussian smoothing kernel with standard deviation $\sigma = a/\sqrt{3}$. More experimental results can be found in the Appendix F.

Table 2: Test mean and variance of accuracy comparison across datasets for original Backbone, GRPO, and ReDit.

Name	**GSM8K**	**MATH**	**Geometry3K**
Backbone	84.91	39	40.43
GRPO(Baseline)	89.07	48.01	43.10
w/ **ours(Gauss)**	**90.76** (± 0.06)	**52.55** (± 0.03)	**44.67** (± 0.03)
w/ **ours(Uniform)**	90.46 (± 0.07)	51.96 (± 0.06)	44.36 (± 0.04)
Δ	+1.69	+4.54	+1.57

Accelerated Convergence Across Datasets and LLMs. We demonstrate that integrating our proposed method, ReDit, with GRPO substantially accelerates convergence and improves final performance across a wide range of datasets (Fig. 7) and LLMs, including Llama-3.2-3B, Llama-3.1-8B, Ministral-8B and Mistral-7B (Fig. 6). On all tested models, both Gaussian and uniform variants of ReDit enable GRPO to reach a competitive performance level within merely 1000 training steps. Notably, this performance already surpasses that of the baseline GRPO trained for the full 9000 steps. Consequently, ReDit not only enhances training efficiency but also leads to

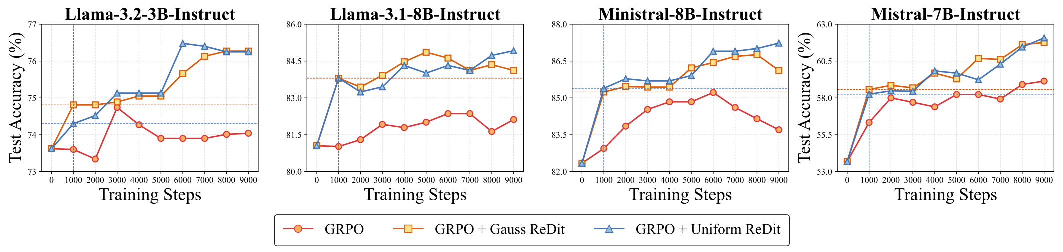

Figure 6: Accuracy of different LLMs on GSM8K. ReDit improves training efficiency and final performance in various LLMs.

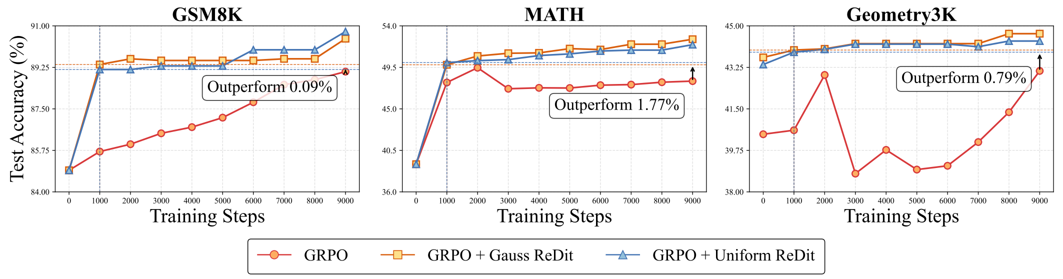

Figure 7: Test accuracy across datasets. The horizontal dashed line marks ReDit's performance at 1000 steps, which GRPO fails to match even after 9000 steps.

superior final accuracy. The Gaussian variant, in particular, consistently yields the strongest results and promotes more stable training trajectories with lower volatility compared to the baseline. We also present more experimental results in the appendix. The results on the code dataset are detailed in Appendix E.1, the results on the full parameter fine-tuning method are detailed in Appendix E.2, and the results on the Deepseek-R1 distillation model are detailed in Appendix E.3.

Generalization to Diverse Baselines. Fig. 5 presents results from applying ReDit to additional reinforcement learning baselines (DAPO, Dr.GRPO, and REINFORCE++) on the GSM8K dataset. Across all algorithms, ReDit (both Gaussian and uniform variants) consistently enhances performance and accelerates learning. Beyond these early-stage improvements, ReDit also substantially boosts the final accuracy of these baselines, as quantitatively demonstrated in Table 1. These accuracy gains (Table 2) complement the qualitative evidence in Fig. 5, confirming that ReDit enables faster and more stable learning across diverse algorithms.

Optimal Performance with Scheduled Perturbation. We further investigate convergence behavior under various scheduled perturbation schemes: SquareRoot, Cosine, and CosineReverse perturbations. These schedules dynamically adjust perturbation variance throughout training, potentially benefiting model learning. Fig. in the Appendix F.4 illustrates the different perturbation schedules, while Fig. 8presents their performance. Compared to standard GRPO, ReDit achieves both faster convergence and superior final performance, with the CosineReverse perturbation schedule yielding particularly strong results. Additional details are provided in the Appendix F.4.

5.3 Ablation Studies

Perturbation variance affects performance. To study the sensitivity of ReDit to the perturbation amplitude, we performed an ablation study by varying the parameter a in the Gaussian smoothing kernel with standard deviation $\sigma = a/\sqrt{3}$. This effectively changes the variance of the applied perturbation. As shown in Fig. 9, applying reward smoothing (i.e., for any $a > 0.00$) consistently leads to faster convergence compared to the baseline without smoothing ($a = 0.00$). Moreover, in most cases, increasing the perturbation amplitude (larger a) tends to improve the final performance of the model. Notably, the configuration with $a = 0.05$ shows superior performance, achieving not only

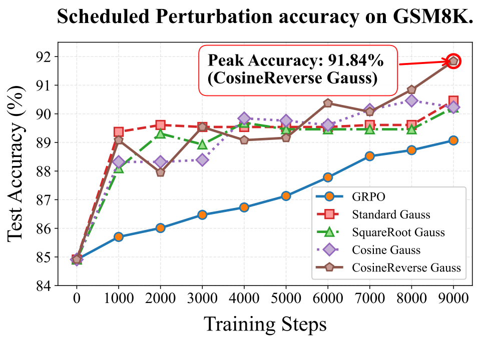

Figure 8: CosineReverse perturbation achieves the best performance.

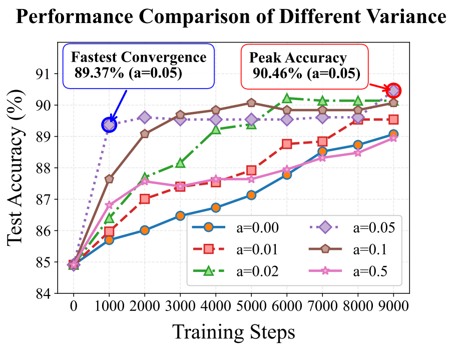

Figure 9: Appropriate perturbation achieves the best performance.

the fastest convergence but also the best peak model performance, see the Fig. 9 annotation. However, these results highlight a key trade-off. While moderate perturbations are beneficial, excessive perturbations (e.g., $a = 0.5$) may over-smooth the reward landscape. This may mask the original reward signal and lead to performance degradation. Conversely, if the perturbation variance is too small (e.g., $a = 0.01$), the smoothing effect is small and the improvement over the baseline is limited. This suggests that there is an optimal perturbation variance. We recommend conducting preliminary experiments on a smaller dataset to effectively determine this optimal variance before applying it to larger-scale training scenarios. For a detailed theoretical introduction to σ, please refer to Section 6.

Isolating the Effect on Discrete Rewards. To verify that the performance gains of ReDit stem specifically from smoothing discrete rewards, we conducted a crucial ablation study. In this experiment, we replaced the discrete reward signal with a continuous one generated by a reward model pre-trained on human preference data. This model provides a continuous quality score within the range [0,1]. We then applied the ReDit perturbation mechanism directly to these continuous rewards. The results, presented in Fig. 10, show that applying ReDit in this setting yields no discernible impact on either the convergence speed of model or its final performance. This outcome strongly indicates that the benefits of ReDit are nullified when the reward landscape is already smooth. We therefore conclude that the efficacy of ReDit lies specifically in addressing the optimization challenges inherent to sparse and discrete reward signals.

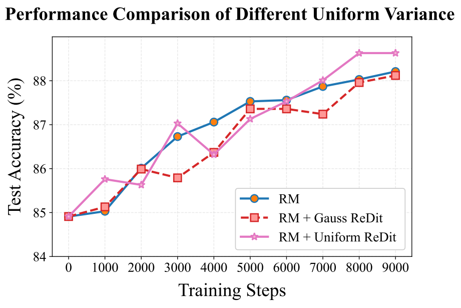

Figure 10: ReDit has little effect on improving the performance of GRPO based RM.

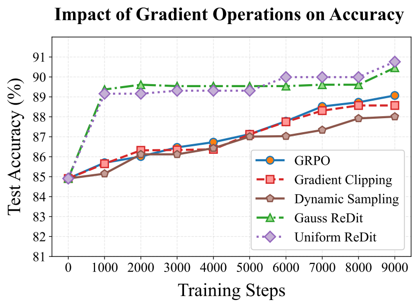

Figure 11: Appropriate perturbation achieves the best performance.

Comparison with Direct Gradient Manipulation Baselines. We benchmark ReDit against established techniques that directly address gradient instability: Gradient Clipping [49], which mitigates exploding gradients, and Dynamic Sampling [38], which alleviates vanishing gradients. The objective is to compare our ReDit approach with methods that operate directly on the gradient signal. As illustrated in Figure 11, ReDit substantially outperforms both baseline methods. We attribute this performance gap to the inherent limitations of these heuristics. Gradient Clipping, for instance, crudely truncates gradient magnitudes, a non-principled operation that can introduce significant estimation bias. Conversely, while Dynamic Sampling can be effective for vanishing gradients, it offers no mechanism to prevent gradients from exploding. In contrast, ReDit stabilizes the training process by smoothing the reward, which provides a more principled solution to prevent both gradient vanishing and explosion, thereby leading to more efficient and effective training.

6 Theoretical Insights

We provides a theoretical analyzing how perturbing discrete reward signals with, e.g., Gaussian noise, accelerates RL convergence, offering a principled explanation for observed empirical benefits.

Problem Setup. Our analysis uses a simplified RL framework (from Eq. (1)) focusing on binary rewards $R(q,o) \in \{0,1\}$ for complete outputs (e.g., GRPO [15]), not token-level rewards. We investigate how Gaussian noise $\epsilon \sim \mathcal{N}(0,\sigma^2)$ injection improves convergence. The perturbed objective is:

$$\tilde{J}(\pi_\theta) = \mathbb{E}_{q\sim p_Q}\left[\mathbb{E}_{o\sim\pi_\theta(\cdot|q)}\tilde{R}(q,o)\right], \tag{6}$$

where the perturbed reward is $\tilde{R}(q,o) = R(q,o) + \epsilon$.

Proposition 1 (Unbiased estimate of gradient)**.** *Introducing noise will still ensure the unbiased estimate of the gradient of the original optimization target Eq.* (1)*, that is:*

$$\mathbb{E}\left[\nabla_\theta \tilde{J}(\pi_\theta)\right] = \mathbb{E}\left[\nabla_\theta J(\pi_\theta)\right].$$

Remark. Proposition 1 provides theoretical proof that introducing Gaussian noise perturbations into the discrete reward signal preserves the unbiased nature of the policy gradient estimate. This means that, under the perturbed reward, the expected direction of the policy update is consistent with the original objective being optimized. Maintaining this unbiased nature ensures that the injected noise does not introduce systematic biases into the learning dynamics, thus providing a theoretical basis for the empirical observation that our approach helps to consistently improve performance. See Appendix B.2 for a detailed proof.

Proposition 2 (Introducing the variance of gradient estimation)**.** *Suppose we are optimizing a non-degenerate strategy, that is, its gradient $\nabla_\theta \log \pi_\theta$ is not completely zero. Introducing noise will introduce gradient noise on the originally calculated gradient, and its variance is:*

$$\mathit{Var}(\mathit{Gradient\ Noise}) = \sigma^2 \cdot \mathbb{E}\left[\|\nabla_\theta \log \pi_\theta(o|q)\|^2\right] > 0.$$

Remark. In Proposition 2, we analyze how Gaussian reward perturbations affect the variance of policy gradient estimates. Adding Gaussian noise $\epsilon \sim \mathcal{N}(0,\sigma^2)$ to the reward introduces a "gradient noise" component proportional to $\epsilon \cdot \nabla_\theta \log \pi_\theta(o|q)$ in the gradient estimate. The increased variance has significant optimization benefits: **Mitigate vanishing gradients:** Gradient noise provides consistent stochastic updates even when the original gradient terms are small or vanishing, thus helping to avoid flat regions. **Avoid exploding gradients:** The randomness induced by the noise enables the optimization trajectory to probabilistically bypass unstable regions of high curvature. Furthermore, the noise variance σ can be adjusted to control the magnitude of the gradient noise for optimal results. This mechanism enhances the robustness of policy optimization and explains the empirical improvements observed in training stability and convergence speed from reward perturbations. For detailed derivation, see Appendix B.3.

7 Limitations and Conclusions

ReDit Improve the stability of reinforcement learning with zero-mean reward noise - theoretically smoothing gradients, preventing gradient instability, and accelerating convergence by increasing reward variance. Benchmarks verify significant improvements in convergence speed and performance. While our approach works, it requires careful tuning of the perturbation variance (although we adopt a small dataset search strategy, extensive experiments are still needed), and future work will aim to achieve automatic variance.

References

[1] AI@Meta. The llama 4 herd. `https://ai.meta.com/blog/llama-4-multimodal-intelligence/`, 2025. Accessed: 2025.

[2] Anthropic. Claude 3.5 sonnet. `https://www.anthropic.com/news/claude-3-5-sonnet`, 2024. Accessed: 2024.

[3] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024.

[4] Chenxing Wei, Yao Shu, Ying Tiffany He, and Fei Yu. Flexora: Flexible low-rank adaptation for large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pages 14643–14682, Vienna, Austria, July 2025. Association for Computational Linguistics.

[5] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4302–4310, Red Hook, NY, USA, 2017. Curran Associates Inc.

[6] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

[7] Hao Lang, Fei Huang, and Yongbin Li. Fine-tuning language models with reward learning on policy. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1382–1392, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[8] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[9] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A survey of reinforcement learning from human feedback, 2024.

[10] Nathan Lambert. Reinforcement learning from human feedback, 2025.

[11] Yuji Cao, Huan Zhao, Yuheng Cheng, Ting Shu, Yue Chen, Guolong Liu, Gaoqi Liang, Junhua Zhao, Jinyue Yan, and Yun Li. Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods. *IEEE Transactions on Neural Networks and Learning Systems*, page 1–21, 2024.

[12] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[13] Chenxing Wei, Hong Wang, Ying He, Fei Yu, and Yao Shu. Test-time policy adaptation for enhanced multi-turn interactions with llms, 2025.

[14] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X.

Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

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

[16] Dingwen Kong and Lin F. Yang. Provably feedback-efficient reinforcement learning via active reward learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.

[17] Zhiqiang Wang, Pengbin Feng, Yanbin Lin, Shuzhang Cai, Zongao Bian, Jinghua Yan, and Xingquan Zhu. Crowdvlm-r1: Expanding r1 ability to vision language model for crowd counting using fuzzy group relative policy reward, 2025.

[18] Harris Chan, Volodymyr Mnih, Feryal Behbahani, Michael Laskin, Luyu Wang, Fabio Pardo, Maxime Gazeau, Himanshu Sahni, Dan Horgan, Kate Baumli, Yannick Schroecker, Stephen Spencer, Richie Steigerwald, John Quan, Gheorghe Comanici, Sebastian Flennerhag, Alexander Neitz, Lei M Zhang, Tom Schaul, Satinder Singh, Clare Lyle, Tim Rocktäschel, Jack Parker-Holder, and Kristian Holsheimer. Vision-language models as a source of rewards. In *Second Agent Learning in Open-Endedness Workshop*, 2023.

[19] Desik Rengarajan, Gargi Vaidya, Akshay Sarvesh, Dileep Kalathil, and Srinivas Shakkottai. Reinforcement learning with sparse rewards using guidance from offline demonstration. In *International Conference on Learning Representations*, 2022.

[20] Gautham Vasan, Yan Wang, Fahim Shahriar, James Bergstra, Martin Jagersand, and A. Rupam Mahmood. Revisiting sparse rewards for goal-reaching reinforcement learning, 2024.

[21] Prasoon Goyal, Scott Niekum, and Raymond J. Mooney. Using natural language for reward shaping in reinforcement learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, page 2385–2391. AAAI Press, 2019.

[22] Niladri S. Chatterji, Aldo Pacchiano, Peter L. Bartlett, and Michael I. Jordan. On the theory of reinforcement learning with once-per-episode feedback. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc.

[23] Meng Cao, Lei Shu, Lei Yu, Yun Zhu, Nevan Wichers, Yinxiao Liu, and Lei Meng. Enhancing reinforcement learning with dense rewards from language model critic. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9119–9138, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[24] Alex James Chan, Hao Sun, Samuel Holt, and Mihaela van der Schaar. Dense reward for free in reinforcement learning from human feedback. In *International Conference on Machine Learning*, 2024.

[25] Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2reward: Reward shaping with language models for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[26] Lex Weaver and Nigel Tao. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, page 538–545, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[27] Noam Razin, Hattie Zhou, Omid Saremi, Vimal Thilak, Arwen Bradley, Preetum Nakkiran, Joshua M. Susskind, and Etai Littwin. Vanishing gradients in reinforcement finetuning of language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[28] Mohammed Sharafath Abdul Hameed, Gavneet Singh Chadha, Andreas Schwung, and Steven X. Ding. Gradient monitored reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):4106–4119, 2023.

[29] Shauharda Khadka and Kagan Tumer. Evolution-guided policy gradient in reinforcement learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 1196–1208, Red Hook, NY, USA, 2018. Curran Associates Inc.

[30] Kaichen Zhang, Yuzhong Hong, Junwei Bao, Hongfei Jiang, Yang Song, Dingqian Hong, and Hui Xiong. Gvpo: Group variance policy optimization for large language model post-training, 2025.

[31] Qwen, :, 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, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

[32] 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, 2021.

[33] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*, 2020.

[34] Hamish Ivison, Yizhong Wang, Jiacheng Liu, Zeqiu Wu, Valentina Pyatkin, Nathan Lambert, Noah A. Smith, Yejin Choi, and Hannaneh Hajishirzi. Unpacking DPO and PPO: Disentangling best practices for learning from preference feedback. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[35] Yanjun Chen, Dawei Zhu, Yirong Sun, Xinghao Chen, Wei Zhang, and Xiaoyu Shen. The accuracy paradox in RLHF: When better reward models don't yield better language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2980–2989, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[36] Xueru Wen, Jie Lou, Yaojie Lu, Hongyu Lin, XingYu, Xinyu Lu, Ben He, Xianpei Han, Debing Zhang, and Le Sun. Rethinking reward model evaluation: Are we barking up the wrong tree? In *The Thirteenth International Conference on Learning Representations*, 2025.

[37] Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes, 2015.

[38] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025.

[39] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025.

[40] Jian Hu, Jason Klein Liu, and Wei Shen. Reinforce++: An efficient rlhf algorithm with robustness to both prompt and reward models, 2025.

[41] Noam Razin, Zixuan Wang, Hubert Strauss, Stanley Wei, Jason D. Lee, and Sanjeev Arora. What makes a reward model a good teacher? an optimization perspective, 2025.

[42] 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, 2021.

[43] Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning, 2021.

[44] Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, and Luke Zettlemoyer. Spurious rewards: Rethinking training signals in rlvr, 2025.

[45] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[46] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[47] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`, 2020.

[48] OpenCompass Contributors. Opencompass: A universal evaluation platform for foundation models. `https://github.com/open-compass/opencompass`, 2023.

[49] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*, 2020.

[50] Han Zhang, Yu Lei, Lin Gui, Min Yang, Yulan He, Hui Wang, and Ruifeng Xu. CPPO: Continual learning for reinforcement learning with human feedback. In *The Twelfth International Conference on Learning Representations*, 2024.

[51] Xiangxiang Chu, Hailang Huang, Xiao Zhang, Fei Wei, and Yong Wang. Gpg: A simple and strong reinforcement learning baseline for model reasoning, 2025.

[52] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, page 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[53] Marek Grześ. Reward shaping in episodic reinforcement learning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, page 565–573, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.

[54] Abhishek Gupta, Aldo Pacchiano, Yuexiang Zhai, Sham M. Kakade, and Sergey Levine. Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[55] Maryam Fazel, Rong Ge, Sham M. Kakade, and Mehran Mesbahi. Global convergence of policy gradient methods for linearized control problems, 2018.

[56] Chenxing Wei, Yao Shu, Mingwen Ou, Ying Tiffany He, and Fei Richard Yu. Paft: Prompt-agnostic fine-tuning, 2025.

Acknowledgement

This work was supported by the Shenzhen Science and Technology Program (Grant No. ZDSYS20220527171400002), the National Natural Science Foundation of China (Grant Nos. 62271324, 62231020, and 62371309), and the Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ) (Grant No. GML-KF-24-32).

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims are put in the abstract and Section 1

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.

- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: For further details, please refer to the "Limitations and Conclusions" section, found in Section 7.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The claims are put in the abstract and Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: For detailed information, please refer to the "Method" section in Section 4. Complete pseudocode can be found in Algorithm 1. Specific experimental settings are discussed in Appendix D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the instruction and code in supplemental material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Specific experimental settings are discussed in Appendix D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For a detailed comparison, please refer to the experimental sections in Section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [No]

Justification: Specific experimental environments can be found in Sec 5.1.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have checked the NeurIPS code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This paper mainly studies the impact of discrete rewards on reinforcement learning. It does not have any direct positive or negative social impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We have checked this and confirmed the paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The original papers or URLs of the codes, models, and data sets used in this article have been cited in the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide the instruction in supplemental material about our code and data.

    Guidelines:

    - The answer NA means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: This paper only utilizes LLMs for writing and editing.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Related Work

Reinforcement Learning with Discrete Rewards. Group Relative Policy Optimization (GRPO) [15] utilizes discrete rewards generated by a rule-based reward function to guide the policy model update. This reward function, known for its simplicity and unbiasedness, effectively mitigates reward hacking and has demonstrated strong performance. However, GRPO faces challenges related to slow training speed and unstable gradients during training. To address these issues, various methods have been proposed. DAPO [38] introduced a dynamic sampling strategy to improve gradient effectiveness by dynamically filtering invalid samples, thereby increasing sample efficiency, although this reduced training speed. CPPO [50] prunes completions with low absolute advantages, significantly reducing the number of gradient calculations and updates required, which enhances training efficiency but can lead to gradient estimation errors. GPG [51] directly optimizes the original reinforcement learning objective, eliminating the need for a proxy loss function and improving training efficiency. However, this simplification may result in a significant divergence between the actor and policy models. Dr.GRPO [39] improves token efficiency while maintaining inference performance. Despite these efforts, a critical challenge remains: these algorithms largely neglect the inherent difficulties introduced by discrete rewards during the optimization process. The oscillations caused by gradient vanishing and exploding are major contributors to the slow optimization speed. Our work specifically aims to overcome the challenges in gradient optimization that arise from using discrete rewards.

Addressing Reward Design Challenges in LLM Reinforcement Learning. Designing effective reward functions for identifying optimal strategies is a well-established area of research outside the context of Large Language Models (LLMs) [52, 53, 54]. However, a consensus on the optimal approach for LLM reinforcement learning has not yet been reached [55]. The RLHF framework proposed training a reward model to score LLM outputs [5]. A recurring challenge, as noted by numerous studies, is that low reward model accuracy can induce reward hacking [34, 35, 36]. Conversely, improving accuracy often reduces reward variance, which can slow down policy model convergence due to vanishing gradients [27]. Although the reward function presented in GRPO provides perfectly correct rewards, thereby avoiding reward hacking, it exacerbates gradient instability and hinders optimization speed [15, 39]. Recent theoretical findings indicate that a successful reward function requires a trade-off between variance and inaccuracy [41]. Motivated by this, our work seeks to design a reward function that effectively addresses the problem of reward hacking while simultaneously facilitating efficient optimization.

B Theorems and proofs

B.1 Definitions

From Definition 1, 2 in [41], The accuracy and variance of the reward function is as follows:

Definition 1. *Given a prompt $x \in \mathcal{X}$, the accuracy of a reward model $r_{RM} : \mathcal{X} \times \mathcal{Y} \to [-1, 1]$ with respect to a distribution $\mathcal{D}$ over unordered output pairs is defined by:*

$$Acc_{x,\mathcal{D}}(r_{RM}) := \mathbb{E}_{\{y,y'\}\sim\mathcal{D}}\left[\mathbb{1}\left[sign\big(r_{RM}(x,y) - r_{RM}(x,y')\big) = sign\big(r_G(x,y) - r_G(x,y')\big)\right]\right], \tag{7}$$

where r_G is the ground truth reward, $\mathbb{1}[\cdot]$ is an indicator function, and sign $: \mathbb{R} \to \{-1, 0, 1\}$ is the sign function.[2]

Definition 2. *Given a policy π_θ, prompt $x \in \mathcal{X}$, and reward model $r_{RM} : \mathcal{X} \times \mathcal{Y} \to [-1, 1]$, the reward variance induced by r_{RM} for π_θ and x is defined by:*

$$Var_{y\sim\pi_\theta(\cdot|x)}[r_{RM}(x,y)] := \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[\left(r_{RM}(x,y) - \mathbb{E}_{y'\sim\pi_\theta(\cdot|x)}\big[r_{RM}(x,y')\big]\right)^2\right]. \tag{8}$$

[2]For a set of prompts, accuracy refers to the mean accuracy over the set.

B.2 Proof of Proposition 1

The proof of Proposition 1 is expressed as follows:

Proof. By the policy gradient theorem, the gradient of the original objective (1) expands to:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{q\sim p_Q}\mathbb{E}_{o\sim\pi_\theta(\cdot|q)}\left[R(q,o)\nabla_\theta \log \pi_\theta(o|q)\right]. \tag{9}$$

For the noise-injected objective, its gradient becomes:

$$\nabla_\theta \tilde{J}(\pi_\theta) = \mathbb{E}_{q\sim p_Q}\mathbb{E}_{o\sim\pi_\theta(\cdot|q)}\left[\tilde{R}(q,o)\nabla_\theta \log \pi_\theta(o|q)\right]. \tag{10}$$

Substituting $\tilde{R}(q,o) = R(q,o) + \epsilon$ and leveraging linearity of expectation:

$$\begin{aligned}\mathbb{E}\left[\nabla_\theta \tilde{J}\right] &= \mathbb{E}_{q,o,\epsilon}\left[(R(q,o)+\epsilon)\nabla_\theta \log \pi_\theta(o|q)\right] &(11)\\ &= \underbrace{\mathbb{E}_{q,o}\left[R(q,o)\nabla_\theta \log \pi_\theta(o|q)\right]}_{\mathbb{E}[\nabla_\theta J]} + \mathbb{E}_\epsilon[\epsilon]\cdot\mathbb{E}_{q,o}\left[\nabla_\theta \log \pi_\theta(o|q)\right]. &(12)\end{aligned}$$

Zero-mean noise: $\mathbb{E}_\epsilon[\epsilon] = 0$ by definition of $\mathcal{N}(0,\sigma^2)$. Thus, the cross-term vanishes:

$$\mathbb{E}[\nabla_\theta \tilde{J}] = \mathbb{E}[\nabla_\theta J] + 0 = \mathbb{E}[\nabla_\theta J]. \tag{13}$$

□

B.3 Proof of Proposition 2

Proof. Consider the perturbed objective function with noise-augmented reward $R(q,o)+\epsilon$. The estimated value of the gradient of the noise enhancement objective function using n samples is:

$$\nabla \hat{\tilde{J}}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left[\nabla \log \pi_\theta(o_i|q_i)\cdot(R(q_i,o_i)+\epsilon_i)\right], \tag{14}$$

where $\epsilon_i \sim \mathcal{N}(0,\sigma^2)$ is the Gaussian noise. The original reward gradient is:

$$\nabla \hat{J}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left[\nabla \log \pi_\theta(o_i|q_i)\cdot(R(q_i,o_i))\right]. \tag{15}$$

Under this condition, the Eq. (14) simplifies to:

$$\nabla \hat{\tilde{J}}(\theta) = \underbrace{\nabla \hat{J}(\theta)}_{\text{origin gradient}} + \underbrace{\frac{1}{n}\sum_{i=1}^{n}\left[\nabla \log \pi_\theta(o_i|q_i)\cdot\epsilon_i\right]}_{\text{noise gradient}}. \tag{16}$$

While the expectation $\mathbb{E}_\epsilon[\epsilon]=0$ implies the noise contribution's mean is zero, the variance of the gradient term persists. To compute this variance, we use the definition: $\text{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$. Applying this to the noise-induced component $\epsilon\cdot\nabla_\theta \log \pi_\theta(o|q)$, we get:

$$\text{Var}\left(\epsilon\cdot\nabla_\theta \log \pi_\theta(o|q)\right) = \mathbb{E}\left[\epsilon^2\cdot\|\nabla_\theta \log \pi_\theta(o|q)\|^2\right] - \left(\mathbb{E}\left[\epsilon\cdot\nabla_\theta \log \pi_\theta(o|q)\right]\right)^2. \tag{17}$$

Since $\mathbb{E}[\epsilon]=0$, the second term vanishes. For the first term, note that:

$$\mathbb{E}[\epsilon^2] = \text{Var}(\epsilon) + (\mathbb{E}[\epsilon])^2 = \sigma^2 + 0 = \sigma^2. \tag{18}$$

This allows us to simplify the variance expression to:

$$\text{Var(noise gradient)} = \text{Var}\left(\epsilon\cdot\nabla_\theta \log \pi_\theta\right) = \sigma^2\cdot\mathbb{E}\left[\|\nabla_\theta \log \pi_\theta(o|q)\|^2\right] > 0, \tag{19}$$

provided $\nabla_\theta \log \pi_\theta$ is not identically zero (a reasonable assumption for non-degenerate policies).

□

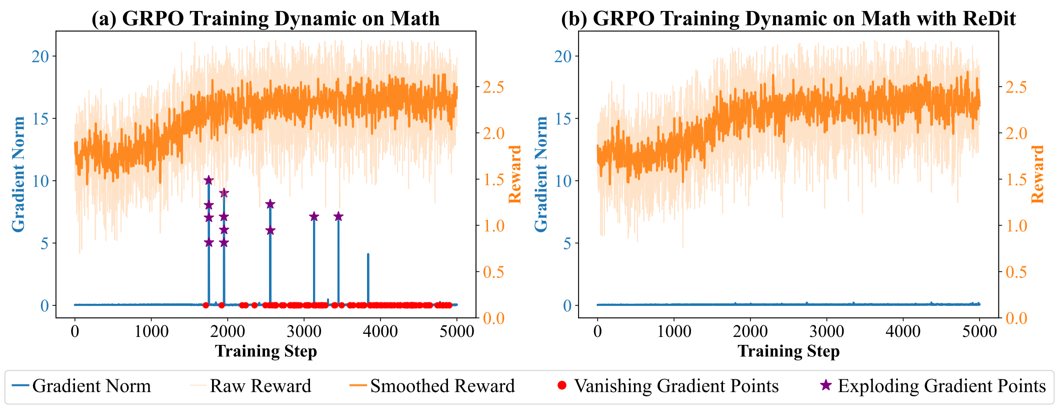

Figure 12: Training Dynamics of Gradient Norm and Reward on Math Dataset.

C Training Dynamic

In this section, we show more Training Dynamic information.

Figure 12 shows the training dynamics of using and not using ReDit on the Math dataset, indicating that using ReDit can solve the problems of gradient oscillation and gradient vanishing, and improve training stability

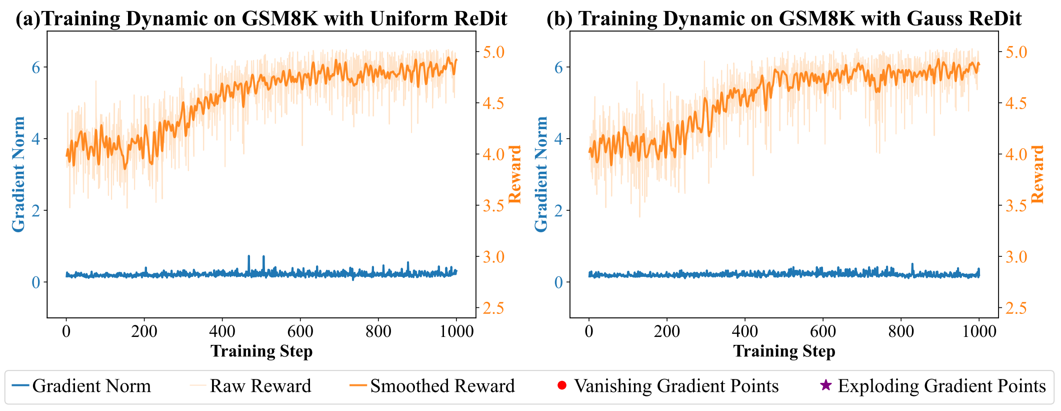

Figure 13: Training dynamics of gradient norm and reward on the GSM8K dataset, showing the impact of perturbations of different distributions.

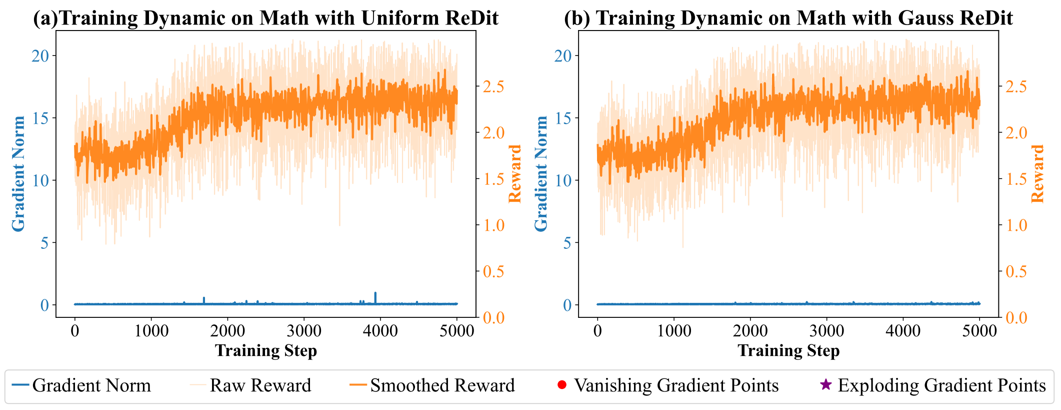

Figure 14: Training dynamics of gradient norm and reward on the Math dataset, showing the impact of perturbations of different distributions.

Fig 13 and Fig 14 Training dynamics using uniform and Gaussian perturbations. For both uniform and Gaussian perturbations, ReDit shows amazing gradient stability and training stability.

D Experimental setting

D.1 Dataset

In this section, we introduce the statistics of the dataset and the additional processing performed on the dataset. The statistics of the dataset are shown in Table 3.

Table 3: Number of samples in the train, validation, and test datasets for various dateset.

Number of samples	train dataset	validation dataset	test dataset
GSM8K	7473	-	1319
MATH	7506	-	5003
Geometry3K	2100	300	601

In addition, We added new templates to the original dataset to ensure the model could complete the required tasks and output formats. It is important to note that the added templates did not alter the original dataset, and special processing was performed for different LLMs. The specific examples are as follows:

Dataset Format of GSM8K

```
dataset: GSM8K
    "prompt": [
        {"role": "system", "content": "Respond in the following format:
        <reasoning> ... </reasoning> <answer> ...</answer>"},
        {"role": "user", "content": "What is the largest single-digit prime number?"},
        {"role": "assistant", "content": "<reasoning> 9 is divisble by 3 and 8
        is divisible by 2, but 7 is prime. </reasoning>
        <answer>7</answer>",
        {"role": "user", "content": {question}}
        ],
    "answer": {answer}
```

Dataset Format of MATH

```
dataset: MATH
    "prompt": [
        {"role": "system", "content": "Respond in the following format:
        <reasoning> ... </reasoning> <answer> ...</answer>"},
        {"role": "user", "content": "{question}
        Let"s think step by step and output the final answer within \\boxed{}."
        ],
    "answer": {answer}
```

Dataset Format of Geometry3K

```
dataset: Geometry3K
    "prompt": [
        {"role": "user", "content": [{
            "type": "image",
            "image": {image},
        },
        {
            "type": "text",
            "text": {question} + ".
            You FIRST think about the reasoning process as an internal monologue and
            then provide the final answer. The reasoning process MUST BE enclosed
            within <think> </think> tags. The final answer MUST BE put in \\boxed{}."
            },],
        }
    ]
    "answer": {answer}
```

D.2 Reward function

We design five reward functions for the GSM8K dataset and show how to implement ReDit:

GSM8K Accuracy Reward Function

```python
def correctness_reward_func_with_noise(prompts, completions, answer, **kwargs) -> list[float]:
    def extract_number(s: str) -> str:
        match = re.search(r'\d+', s)
        return match.group(0) if match else ''
    responses = [completion[0]['content'] for completion in completions]
    q = prompts[0][-1]['content']
    extracted_responses = [extract_xml_answer(r) for r in responses]
    original_rewards = [2.0 if extract_number(r) == extract_number(a) else 0.0 for r, a in zip(extracted_responses, answer)]

    # ReDit add
    noisy_rewards = [r + random.uniform(-m * 2.0, m * 2.0) for r in original_rewards]
    #noisy_rewards = [r + random.gauss(0, 2.0 * m / (3 ** 0.5)) for r in original_rewards]
    return noisy_rewards
```

GSM8K Int Reward Function

```python
def int_reward_func_with_noise(completions, **kwargs) -> list[float
    ]:
    responses = [completion[0]['content'] for completion in
        completions]
    extracted_responses = [extract_xml_answer(r) for r in responses
        ]
    original_rewards = [0.5 if r.isdigit() else 0.0 for r in
        extracted_responses]

    # ReDit add
    noisy_rewards = [r + random.uniform(-m * 0.5, m * 0.5) for r in
         original_rewards]
    #noisy_rewards = [r + random.gauss(0, 0.5 * m / (3 ** 0.5)) for
         r in original_rewards]
    return noisy_rewards
```

GSM8K Strict Format Reward Function

```python
def strict_format_reward_func_with_noise(completions, **kwargs) ->
    list[float]:
    pattern = r"^<reasoning>\n[\s\S]*?\n</reasoning>\n<answer>\n[\s
        \S]*?</answer>$"
    completion_contents = [completion[0]["content"].strip() for
        completion in completions]
    matches = [re.match(pattern, content, re.DOTALL | re.MULTILINE)
         for content in completion_contents]
    original_rewards = [1.0 if match else 0.0 for match in matches]

    # ReDit add
    noisy_rewards = [r + random.uniform(-m * 1.0, m * 1.0) for r in
         original_rewards]
    #noisy_rewards = [r + random.gauss(0, 1.0 * m / (3 ** 0.5)) for
         r in original_rewards]
    return noisy_rewards
```

GSM8K Sort Format Reward Function

```python
def soft_format_reward_func_with_noise(completions, **kwargs) ->
    list[float]:
    pattern = r"^<reasoning>[\s\S]*?</reasoning>[\s\S]*?<answer>[\s
        \S]*?</answer>$"
    completion_contents = [completion[0]["content"].strip() for
        completion in completions]
    matches = [re.match(pattern, content, re.DOTALL | re.MULTILINE)
         for content in completion_contents]
    original_rewards = [1.0 if match else 0.0 for match in matches]

    #  ReDit add
    noisy_rewards = [r + random.uniform(-m * 1.0, m * 1.0) for r in
         original_rewards]
    #noisy_rewards = [r + random.gauss(0, 1.0 * m / (3 ** 0.5)) for
         r in original_rewards]
    return noisy_rewards
```

GSM8K Reasoning Format Reward Function

```python
def xmlcount_reward_func_with_noise(completions, **kwargs) -> list[
    float]:
    def count_xml(text) -> float:
        count = 0.0
        if text.count("<reasoning>\n") == 1:
            count += 0.125
        if text.count("\n</reasoning>\n") == 1:
            count += 0.125
        if text.count("\n<answer>\n") == 1:
            count += 0.125
            #count -= len(text.split("\n</answer>\n")[-1])*0.001
        if text.count("\n</answer>") == 1:
            count += 0.125
            count -= (len(text.split("\n</answer>")[-1]) - 1)*0.001
        return count
    contents = [completion[0]["content"] for completion in
        completions]
    original_rewards = [count_xml(c) for c in contents]

    # ReDit add
    noisy_rewards = [r + random.uniform(-m * 0.5, m * 0.5) for r in
         original_rewards]
    #noisy_rewards = [r + random.gauss(0, 0.5 * m / (3 ** 0.5))
        for r in original_rewards]
    return noisy_rewards
```

As shown in the above code block, ReDit does not need to be modified in a complex way, only the reward function needs to be modified, and any method can be easily integrated. The reward functions of other datasets can be found in the code.

D.3 Specific experimental parameters

In this section, we present the experimental parameters, including LoRA parameters, GRPO and other baseline experimental parameters.

Table 4: LoRA Parameters

LoRA Target	**LoRA Rank**	**LoRA Alpha**	**LoRA Dropout**
q & v Proj	8	64	0.05

Table 5: GRPO Parameters

Learning Rate	**Num Generations**	**Epochs**
5e-6	4	10

Table 6: DAPO Parameters

Clip Ratio Low	**Clip Ratio Low**	**Clip Ratio C**	**Num Generations Max**
0.2	0.28	10.0	10

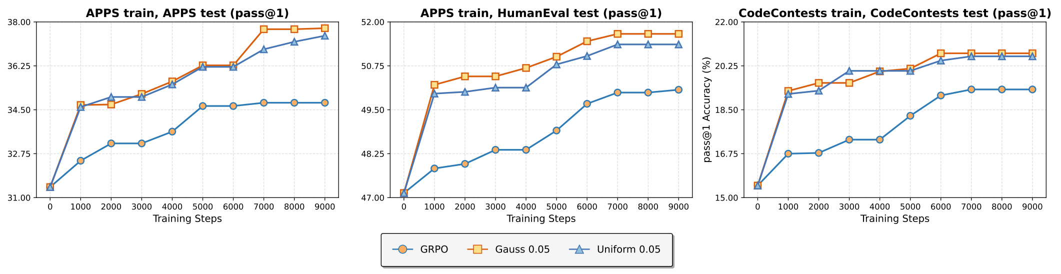

Figure 15: Performance comparison on three code generation benchmarks: (left) APPS test, (center) HumanEval test, and (right) CodeContests test. The $pass@1$ accuracy is reported across training steps. Both ReDit variants (Gauss 0.05 and Uniform 0.05) consistently and significantly outperform the GRPO baseline, confirming the general applicability of our method to the coding domain.

E Additional experiments

E.1 Results on the Code Generation Datasets

To validate the general applicability of our method (ReDit) beyond mathematical reasoning, we conducted a comprehensive evaluation on the domain of code generation. We performed experiments on three widely-used coding benchmarks: APPS, HumanEval, and CodeContests. This evaluation was designed to test the hypothesis that the core benefit of ReDit—stabilizing the learning signal to improve optimization—is a general principle that is not limited to a single domain.

The results are presented in 15. The plots consistently demonstrate that both the Gaussian (Gauss 0.05) and Uniform (Uniform 0.05) variants of ReDit significantly and consistently outperform the GRPO baseline across all three coding benchmarks. On all datasets, our method not only achieves a higher final $pass@1$ accuracy but also exhibits a faster convergence rate. This strongly suggests that ReDit provides a more robust optimization pathway, and its benefits generalize effectively to complex tasks such as code generation.

E.2 Results on Full Parameter Fine-Tuning

To confirm that the benefits of ReDit are not limited to parameter-efficient fine-tuning (PEFT) methods like LoRA, we conducted additional experiments using a full parameter fine-tuning approach. This evaluation addresses whether ReDit's effectiveness is a general property of the optimization process itself, rather than an artifact of a specific tuning method [56].

For these experiments, we utilized a setup that differs from our primary PEFT experiments; specifically, we employed the VERL framework for training with 8 GPUs. We evaluated this full fine-tuning setup on our three mathematical reasoning benchmarks: GSM8K, MATH, and Geo3k.

The results are presented in 16. The plots clearly demonstrate that ReDit (both Gauss 0.05 and Uniform 0.05 variants) consistently outperforms the GRPO baseline across all three benchmarks in this demanding full-tuning setting. The performance gap is particularly notable on the MATH and Geo3k datasets, where the GRPO baseline shows signs of stagnation, while our method continues to improve. These findings confirm that ReDit is a robust and general-purpose technique, delivering consistent performance gains in both parameter-efficient and full fine-tuning paradigms.

E.3 Results on DeepSeek Distillation Models

A potential concern regarding our primary results is that they are predominantly focused on the Qwen2.5 model family. To further demonstrate the robustness and architectural generalizability of ReDit, we conducted additional experiments to validate its effectiveness on specialized reasoning models.

Specifically, we evaluated our method on two expert distillation models: **DeepSeek-R1-Distill-Llama-8B** and **DeepSeek-R1-Distill-Qwen-7B**. We used the GSM8K benchmark to compare their

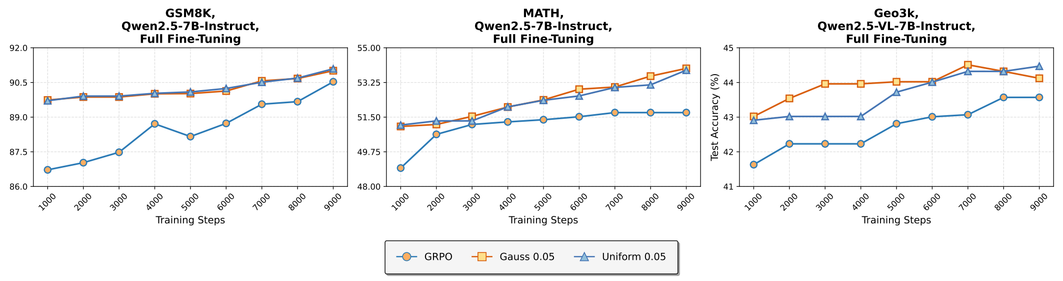

Figure 16: Performance comparison on mathematical reasoning benchmarks using **full parameter fine-tuning**. The plots show test accuracy across training steps for (left) GSM8K, (center) MATH, and (right) Geo3k. In this full-tuning setting, both ReDit variants (Gauss 0.05 and Uniform 0.05) consistently achieve higher test accuracy than the GRPO baseline, confirming that our method's benefits generalize beyond parameter-efficient tuning (PEFT).

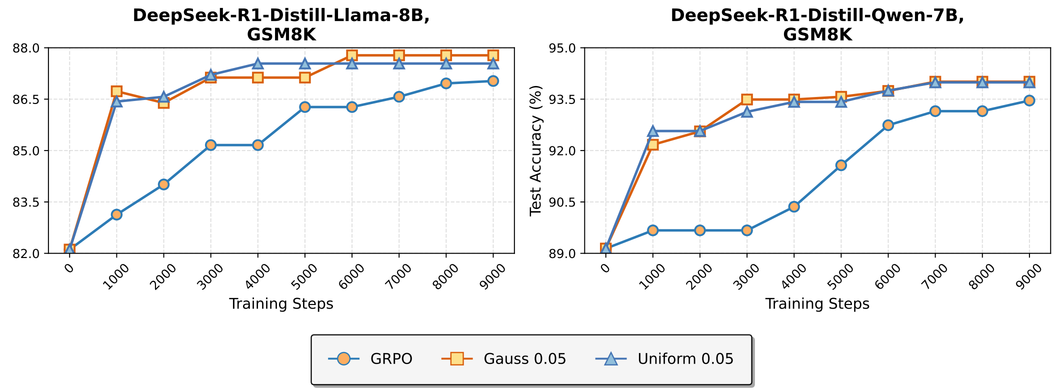

Figure 17: Evaluating ReDit's generalizability on specialized reasoning models. The plots show Test Accuracy (%) on the GSM8K benchmark for (left) **DeepSeek-R1-Distill-Llama-8B** and (right) **DeepSeek-R1-Distill-Qwen-7B**. These results confirm that ReDit's performance advantage over the GRPO baseline holds across different model architectures, not just the Qwen models used in the main experiments.

mathematical reasoning performance against the GRPO baseline. The results are presented in 17. The plots clearly show that both the Gaussian (Gauss 0.05) and Uniform (Uniform 0.05) variants of ReDit consistently outperform the GRPO baseline on both DeepSeek models.

This finding is significant as it confirms that ReDit's ability to stabilize the training signal and improve performance is a general principle. It is not limited to a single model family but holds true across various model architectures, including those specifically optimized for reasoning tasks.

F More result

In this section, we present detailed numerical results for all experiments.

F.1 Main Result

In this section, we show the results in Figure 7, the performance of GRPO and GRPO+ReDit on different datasets.

Tables 7, 8, 9 show the comparison of ReDit on different datasets. ReDit significantly improves the convergence speed of GRPO. At any same step, ReDit achieves better performance.

Table 7: Performance Comparison of Different Training Steps on the Math Dataset

Method \Step	0	1000	2000	3000	4000	5000	6000	7000	8000	9000
Instruct model	39	-	-	-	-	-	-	-	-	-
GRPO	-	47.86	49.46	47.18	47.28	47.26	47.57	47.63	47.89	48.01
Uniform ReDit	-	50.02	50.23	50.34	50.78	50.96	51.27	51.37	51.37	51.96
Gauss ReDit	-	49.78	50.73	51.03	51.07	51.53	51.43	52.01	52.01	52.55

Table 8: Performance Comparison of Different Training Steps on the GSM8K Dataset

Method \Step	0	1000	2000	3000	4000	5000	6000	7000	8000	9000
Instruct model	84.91	-	-	-	-	-	-	-	-	-
GRPO	-	85.70	86.01	86.47	86.73	87.13	87.78	88.52	88.73	89.07
Uniform ReDit	-	89.16	89.16	89.31	89.31	89.31	89.99	89.99	89.99	90.76
Gauss ReDit	-	89.02	89.37	89.61	89.54	89.54	89.54	89.61	89.61	90.46

F.2 Baseline Result

In this section, we present all numerical results in Fig. 5. As shown in Table 10, we demonstrate the effect of using ReDit on GSM8K based on the GRPO improvement method. The experimental results show that ReDit can also improve the convergence speed and performance on these algorithms.

F.3 Variance Result

In this section, we show more results on the performance of ReDit as the perturbation changes. As shown in Figure 18, the variance of uniform perturbation is similar to the variance of Gaussian perturbation, and the appropriate variance can achieve the best performance. The specific numerical results are shown in Tables 11 and 12.

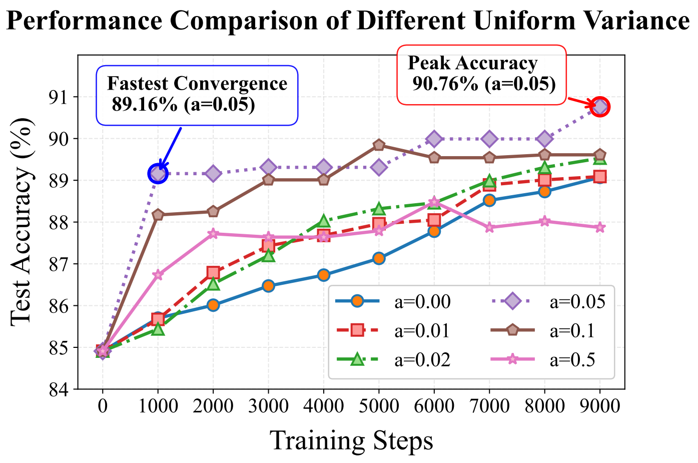

Figure 18: ReDit uniform perturbation performance changes with variance.

F.4 Scheduled Perturbation Result

In this section, we show the changing trends of different scheduled perturbation strategies, as shown in Figure 19. We took the perturbation of Gauss distribution as an example and conducted experiments. The experimental results are shown in Table 13. The CosineReverse strategy shows the best performance.

Table 9: Performance Comparison of Different Training Steps on the Geometry3K Dataset

Method \Step	0	1000	2000	3000	4000	5000	6000	7000	8000	9000
Instruct model	40.43	-	-	-	-	-	-	-	-	-
GRPO	-	40.60	42.93	38.77	39.77	38.94	39.10	40.10	41.36	43.10
Uniform ReDit	-	43.37	43.89	44.01	44.23	44.23	44.23	44.12	44.36	44.36
Gauss ReDit	-	43.67	43.98	44.03	44.25	44.25	44.25	44.25	44.67	44.67

Table 10: Performance Comparison at Different Training Steps on Different Baseline

Method \Step	1000	2000	3000	4000	5000	6000	7000	8000	9000
DAPO	84.99	86.20	86.35	86.35	86.75	87.04	87.12	87.17	87.52
Uniform ReDit	87.03	87.15	87.26	87.54	87.54	87.69	87.83	88.03	88.57
Gauss ReDit	87.76	87.96	88.01	88.01	88.10	88.37	88.67	88.96	89.34
DR.GRPO	84.69	84.23	84.53	84.91	85.67	85.67	85.67	85.90	86.13
Uniform ReDit	86.27	86.36	86.45	86.54	86.75	87.03	87.26	87.16	87.34
Gauss ReDit	86.47	86.23	87.10	87.16	87.56	87.67	87.67	87.67	87.69
REINFORCE++	84.91	84.69	85.06	85.14	85.14	85.14	86.10	86.17	86.25
Uniform ReDit	86.21	86.11	86.67	86.31	86.75	87.01	87.26	87.59	87.59
Gauss ReDit	86.17	86.27	86.47	86.83	86.83	87.06	87.63	87.76	87.96

Table 11: Performance Comparison of Different variance on the Gauss Perturbation

Variance \Step	1000	2000	3000	4000	5000	6000	7000	8000	9000
0.01	85.97	87.01	87.40	87.54	87.92	88.76	88.84	89.54	89.54
0.02	86.40	87.70	88.16	89.23	89.39	90.22	90.14	90.14	90.14
0.05	89.02	89.37	89.61	89.54	89.54	89.54	89.61	89.61	90.46
0.1	87.64	89.08	89.69	89.84	90.07	89.84	89.84	89.84	90.07
0.3	87.87	88.48	88.78	88.93	89.39	89.39	89.39	89.46	89.46
0.5	86.81	87.57	87.41	87.64	87.64	87.95	88.32	88.48	88.95

Table 12: Performance Comparison of Different variance on the Uniform Perturbation

Variance \Step	1000	2000	3000	4000	5000	6000	7000	8000	9000
0.01	85.67	86.79	87.43	87.68	87.96	88.05	88.89	89.01	89.09
0.02	85.44	86.52	87.20	88.03	88.32	88.46	88.99	89.31	89.53
0.05	89.16	89.16	89.31	89.31	89.31	89.99	89.99	89.99	90.76
0.1	88.17	88.25	89.01	89.01	89.84	89.54	89.54	89.61	89.61
0.3	87.49	88.25	88.25	88.02	88.17	87.95	88.93	88.70	88.78
0.5	86.73	87.72	87.64	87.64	87.79	88.48	87.87	88.02	87.87

Table 13: Performance Comparison of Different Scheduled Perturbation Methods

Method \Step	1000	2000	3000	4000	5000	6000	7000	8000	9000
SquareRoot	88.10	89.31	88.93	89.69	89.46	89.46	89.46	89.46	90.22
SquareRootReverse	88.55	89.54	89.46	90.07	90.07	89.31	89.61	89.54	89.69
Factor	88.25	88.63	89.69	89.46	89.23	89.54	89.46	89.31	89.69
FactorReverse	88.48	88.32	89.39	88.78	88.93	89.54	89.61	89.76	89.46
MutilFactor	87.87	89.31	89.01	89.01	89.01	89.61	89.16	89.61	89.46
MutilFactorReverse	88.17	88.78	88.86	89.01	88.93	88.93	89.39	89.16	89.54
Cosine	88.32	88.32	89.39	89.84	89.76	89.61	90.14	90.46	90.23
CosineReverse	89.08	87.95	89.54	89.08	89.16	90.37	90.07	90.84	91.84

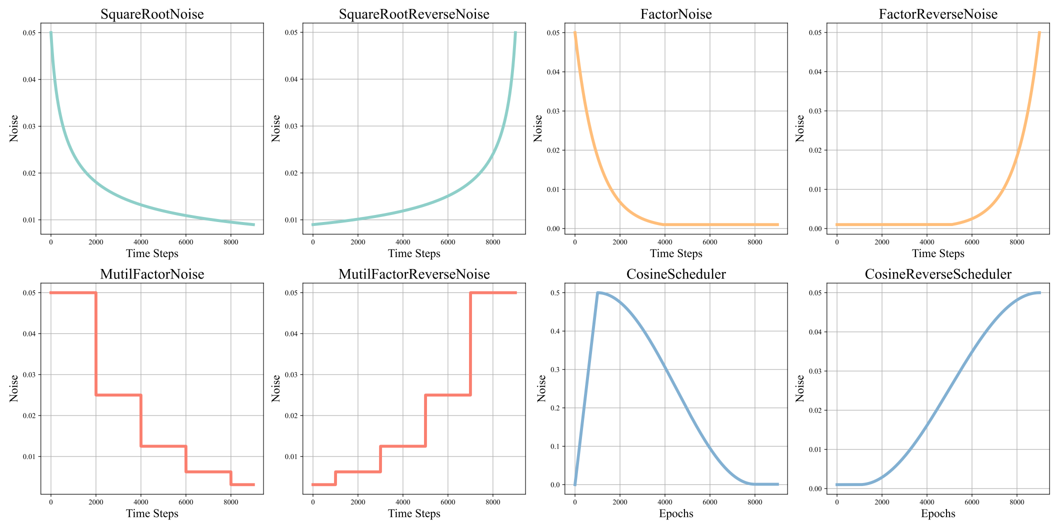

Figure 19: ReDit scheduled perturbation Variance trend with training step (taking the original variance as 0.05 as an example)