ReDit: Reward Dithering for Improved LLM Policy Optimization

Chenxing Wei [*1,2] **Jiarui Yu** [3] **Ying Tiffany He** [1] **Hande Dong** [3] **Yao Shu** [4] **Fei Yu** [1,2]

Abstract

DeepSeek-R1 has successfully enhanced Large Language Model (LLM) 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 Model (LLM) development(AI@Meta, 2025; Anthropic, 2024; OpenAI et al., 2024). Initially, RL from human feedback (RLHF) (Christiano et al., 2017; Ziegler et al., 2019) was employed to align pre-trained LLMs with human preferences (Lang et al., 2024; Ouyang et al., 2022). This typically involves training a separate reward model (RM) on human preference data (Kaufmann et al., 2024), which then guides the LLM policy optimization (Lambert, 2025). While effective, this approach introduces considerable training overhead (Cao et al., 2024b). Subsequently, methods like Direct Preference Optimization (DPO) (Rafailov et al., 2023) 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 (DeepSeek-AI et al., 2025) with Group Relative Policy Optimization (Shao et al., 2024)(GRPO) proposes an alternative: optimizing the LLM policy directly using a rule-based reward system (Kong & Yang, 2022; Wang et al., 2025), 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 (DeepSeek-AI et al., 2025). The simplicity and unbiased nature of these rule-based rewards prevent LLMs from hacking them, potentially fostering enhanced reasoning capabilities (Chan et al., 2023).

However, such reward functions are often discrete, posing significant optimization challenges (Rengarajan et al., 2022; Vasan et al., 2024; Goyal et al., 2019). Consider an RL scenario with a binary reward (Chatterji et al., 2021): 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 (Cao et al., 2024a). Although the model may engage in exploratory behavior on difficult examples, the corresponding gradients remain minimal due to small advantage magnitudes (Chan et al., 2024). Thus, these hard examples and potentially beneficial explorations (Chan et al., 2024) are largely unexploited during the early stages. Conversely, the model may repeatedly reinforce easy examples (Xie et al., 2024), thus reducing incentives to explore alternative strategies for more difficult problems (Weaver & Tao, 2001). This phenomenon can lead to training stagnation in intermediate and advanced

[*]Work done during an internship at Tencent. [1]College of Computer Science and Software Engineering, Shenzhen University, China [2]Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), China [3]Tencent, Shenzhen, China [4]Hong Kong University of Science and Technology (Guangzhou), China. Correspondence to: Yao Shu <yaoshu@hkust-gz.edu.cn>.

Proceedings of the 2^{nd} Workshop on Models of Human Feedback for AI Alignment at the International Conference on Machine Learning (ICML), Vancouver, Canada. PMLR 267, 2025.

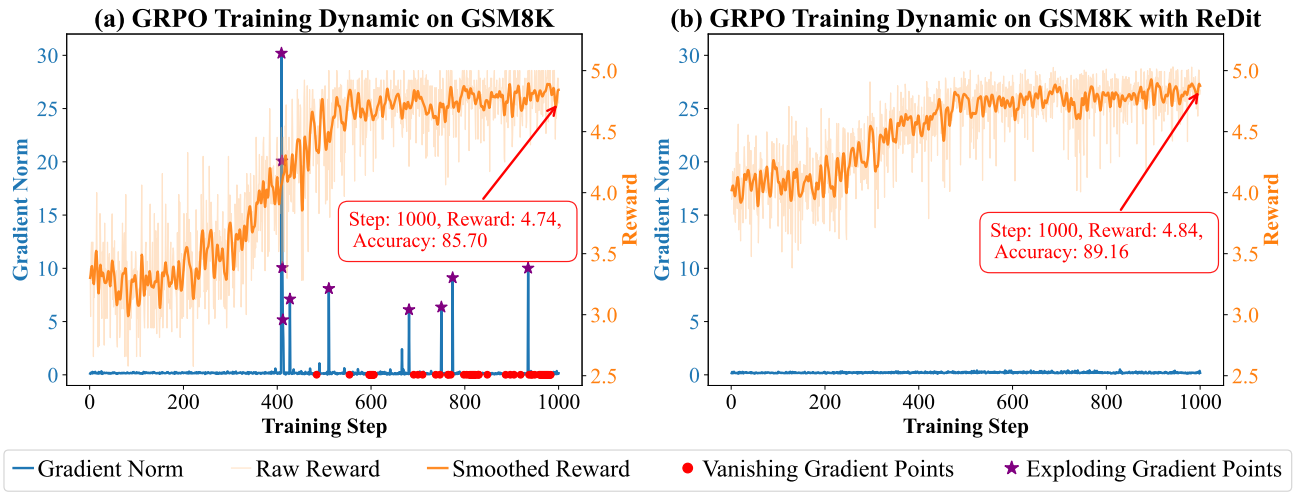

Figure 1. Training Dynamics of Gradient Norm and Reward for Qwen2.5-7B (Qwen et al., 2025) on GSM8K Dataset. Fig.s (a) and (b) compare gradient distributions and reward trends across training steps. The original GRPO method (Fig. (a)) suffers from significant gradient instability—both vanishing (red dots, norms ¡ 0.01) and exploding (purple asterisks, norms ¿ 5). In contrast, ReDit with Gaussian reward smoothing (Fig. (b)) effectively stabilizes optimization throughout training.

stages. Consistent with this, as shown in Fig. 1(a), we observe that the policy model frequently suffers from gradient vanishing (Razin et al., 2024; Abdul Hameed et al., 2023) or explosion (Zhang et al., 2025) during these phases. This combination of insufficient exploration and gradient instability substantially impedes model convergence, representing a critical obstacle to efficient RL in LLM.

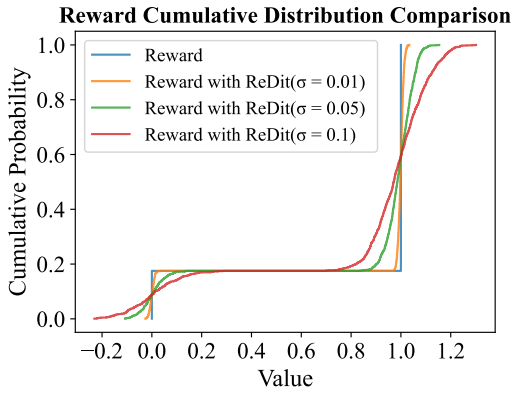

Figure 2. The figure illustrates how ReDit of different variances gradually smooth the reward distribution, showing the smoothing effect of perturbations of different variances.

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 (Ivison et al., 2024; Chen et al., 2024; Wen et al., 2025) have established that a singular focus on increasing reward model accuracy does not necessarily translate to enhanced language model performance. In particular, Wen et al. (2025) 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, an optimal reward system must balance accuracy with appropriate levels of variance.

Inspired by these observations and theoretical insights, we propose ReDit, a simple yet effective technique that applies zero-mean random perturbations to discrete reward signals during training. By introducing controlled noise to the reward function (Fig. 2), ReDit transforms hard reward boundaries into smoother gradients. This softened approach generates greater reward variance within mini-batches which, as established in previous research, enhances model performance and accelerates convergence.

Fig.1 demonstrates the impact of ReDit on LLM policy optimization for GSM8K. The orange lines reveal that GRPO with ReDit achieves substantially higher early-phase rewards than the baseline, indicating the effectiveness of ReDit . We hypothesize that ReDit encourages broader exploration by assigning varied rewards to outputs that only partially meet strict evaluation criteria, thereby accelerating convergence. Although both approaches eventually attain high rewards after 1000 training steps, our method exhibits superior test set performance, suggesting better generalization. As evidenced by Fig.1(a), ReDit maintains stable gradients throughout training while the baseline suffers from gradient vanishing (red point) and explosion (purple star). These results confirm the advantages of ReDit: more stable optimization, faster convergence, and enhanced overall performance.

Moreover, theoretical analysis indicates that a greater reward variance can enhance performance and accelerate con-

vergence in RL (Wen et al., 2025). 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, RL algorithms, perturbation distributions and different LLMs 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 6.1), introduces beneficial gradient variance that mitigates vanishing and exploding gradients (Proposition 6.2), and significantly improves convergence speed (Proposition 6.3).

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) (Hallak et al., 2015) 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 (Shao et al., 2024) 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 $\hat{A}_{i,t}^{\text{GRPO}}$ 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:

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

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 (Yu et al., 2025), Dr.GRPO (Liu et al., 2025), and REINFORCE++ (Hu et al., 2025) generally adopt this discrete reward paradigm (see Appendix A for more related 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).

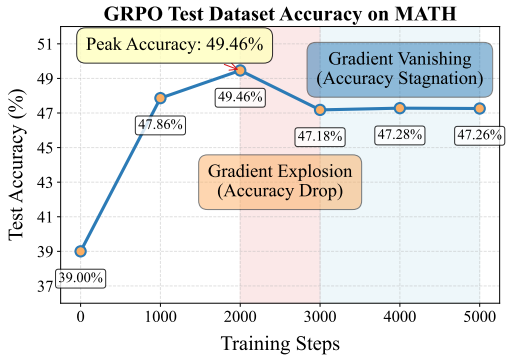

Figure 3. GRPO has unstable performance on the MATH test set. The figure plots the test accuracy achieved for the checkpoints saved during the training run shown in Fig. 4(b).

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. 4 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:

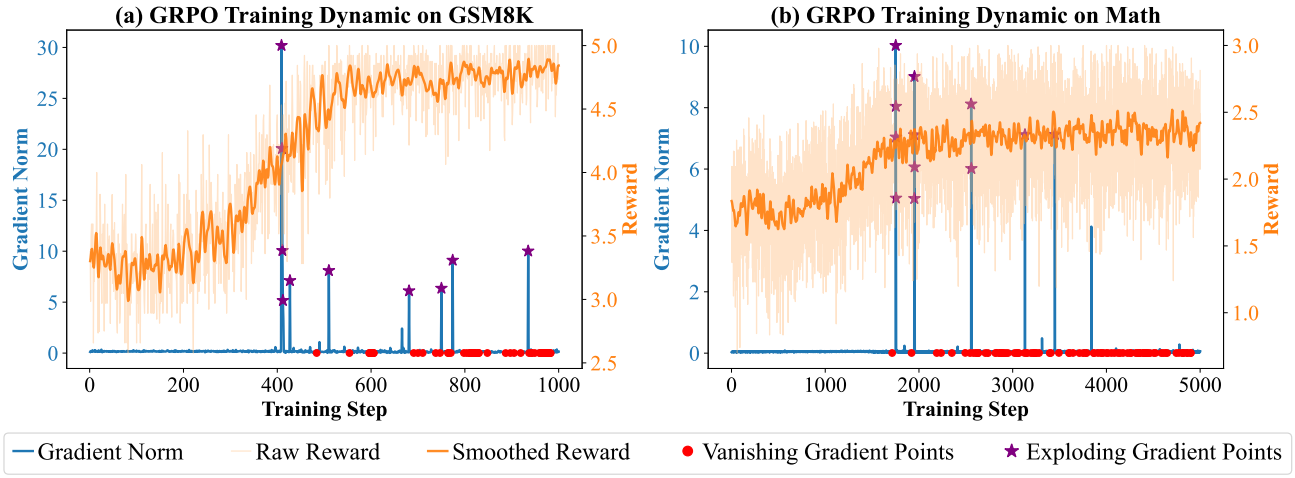

Figure 4. Qwen2.5-7B Gradient norm and reward training dynamics of standard GRPO on GSM8k and MATH datasets. During the whole optimization process, the gradient of standard GRPO is unstable, and there are a lot of gradient vanishing or gradient exploding cases.

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. 4(b) post-step 2000.

Gradient Eplosion. 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. 4(a). Such spikes induce reward fluctuations in subsequent steps, hindering smooth convergence and learning efficiency.

The discrete, sparse rewards induce unstable oscillations between vanishing and exploding gradients. Fig. 3 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 approach to improve the quality of the reward signal. Our solution derives from Theorem 3.3 and Theorem 3.4, which reveal fundamental relationships between reward variance, accuracy, and learning efficiency. From Definition 1, 2 in Razin et al. (2025), The accuracy and variance of the reward function is as follows:

Definition 3.1. 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:

$$\text{acc}_{x,\mathcal{D}}(r_{RM}) := \mathbb{E}_{\{y,y'\}\sim\mathcal{D}}\Big[\mathbb{1}\Big[\text{sign}\big(r_{RM}(x,y) - r_{RM}(x,y')\big) = \text{sign}\big(r_G(x,y) - r_G(x,y')\big)\Big]\Big], \tag{3}$$

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

Definition 3.2. 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:

$$\text{Var}_{y\sim\pi_\theta(\cdot|x)}[r_{RM}(x,y)] := \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\Big[\Big(r_{RM}(x,y) - \mathbb{E}_{y'\sim\pi_\theta(\cdot|x)}\big[r_{RM}(x,y')\big]\Big)^2\Big]. \tag{4}$$

Theorem 3.3 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.

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

Theorem 3.3 (Policy network optimization time lower bound)**.** *From Theorem 1 in Razin et al. (2025). 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:*

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

Complementarily, Theorem 3.4 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.

Theorem 3.4 (Policy network optimization time upper bound)**.** *From Theorem 2 in Razin et al. (2025). Assume* π_θ *is a policy of the form* $\pi_\theta(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}[\theta_{:,\mathbf{x}}]_{\mathbf{y}}$*. Given a hint* $\mathbf{x} \in S$*, 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* $\mathcal{O}(\pi_{\theta(0)}(y^\gamma|\mathbf{x})^{-1})$*.*

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.

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 π_θ.
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_{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 π_θ.

The crucial modification introduced by ReDit lies in *how* the advantage term $\hat{A}_{i,t}^{\text{GRPO}}$ 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 6 in Algorithm 1):

$$\tilde{r}_i = r_i + \epsilon_i \tag{5}$$

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:

$$\begin{aligned} \hat{A}_{i,t}^{\text{GRPO}} &\propto \frac{r_i - \text{mean}(\{r_k\}_{k=1}^G)}{\text{std}(\{r_k\}_{k=1}^G)} \\ &\xrightarrow{\text{ReDit}} \quad \hat{A}_{i,t}^{\text{Dithering}} \propto \frac{\tilde{r}_i - \text{mean}(\{\tilde{r}_k\}_{k=1}^G)}{\text{std}(\{\tilde{r}_k\}_{k=1}^G)} \end{aligned} \tag{6}$$

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 8 in Algorithm 1).

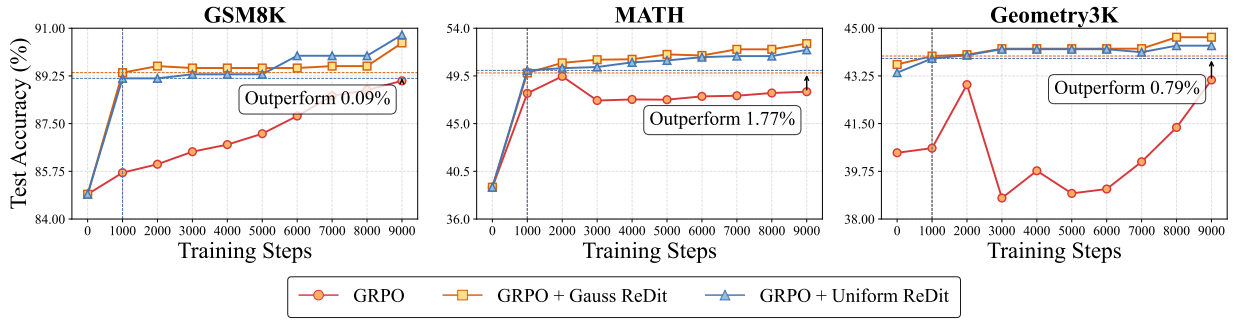

Figure 5. Test accuracy across datasets on Qwen2.5-7B-Instruct and Qwen2.5-VL-7B-Instruct. The horizontal dashed line marks ReDit's performance at 1000 steps, which GRPO fails to match even after 9000 steps.

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.

Datasets. Our dataset selection and setup largely follow the methodology of (Shao et al., 2024), primarily to assess the mathematical reasoning capabilities of the models. This encompasses mathematical problem-solving datasets such as GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), as well as the multimodal geometric reasoning dataset Geometry3K (Lu et al., 2021). 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. 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 Shao et al. (2025) 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, Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct, Ministral-8B-Instruct-2410, and Mistral-7B-Instruct-v0.3, to establish the generalizability of ReDit.

Other Training Settings. For parameter-efficient fine-tuning, we employed Low-Rank Adaptation (LoRA) (Hu et al., 2022). Our implementation leverages the official GRPO implementation within the TRL library (von Werra et al., 2020). Specific configurations for LoRA and GRPO parameters are detailed in the Appendix D.3. Model evaluation was conducted using the OpenCompass (Contributors, 2023). All experiments were executed on single 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 E.

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. 5) 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 superior final accuracy. The Gaussian variant, in particular, consistently yields the strongest results and promotes more stable training trajecto-

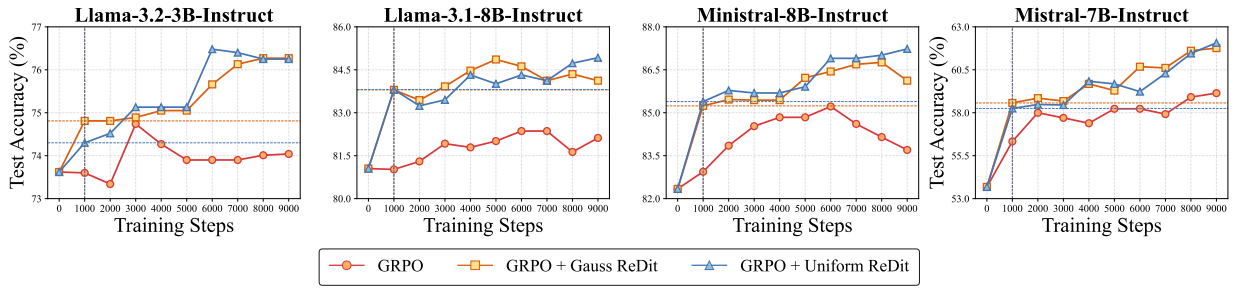

Figure 6. Accuracy of different LLMs on GSM8K. ReDit improves training efficiency and final performance in various LLMs.

ries with lower volatility compared to the baseline.

Table 1. Test accuracy comparison across datasets for original Backbone, GRPO, and ReDit represents percentage point improvement of the superior ReDit variant over the GRPO.

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

Generalization to Diverse Baselines. Fig. 8 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 2. These accuracy gains (Table 1) complement the qualitative evidence in Fig. 8, confirming that ReDit enables faster and more stable learning across diverse algorithms.

Table 2. Comparison of the 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**	**87.69**	**87.96**
w/ **ours(Uniform)**	88.57	87.34	87.59
Δ	+1.82	+1.56	+1.71

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 E.7 illustrates the different perturbation schedules, while Fig. 7 presents 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 E.7.

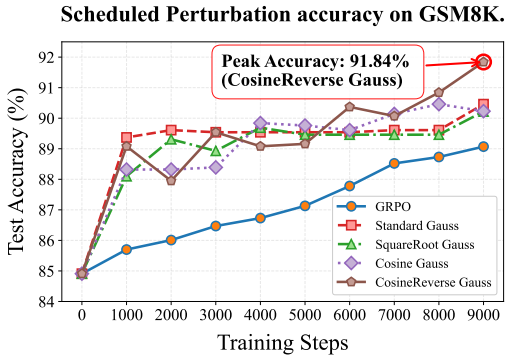

Figure 7. CosineReverse achieves the best performance.

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 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.

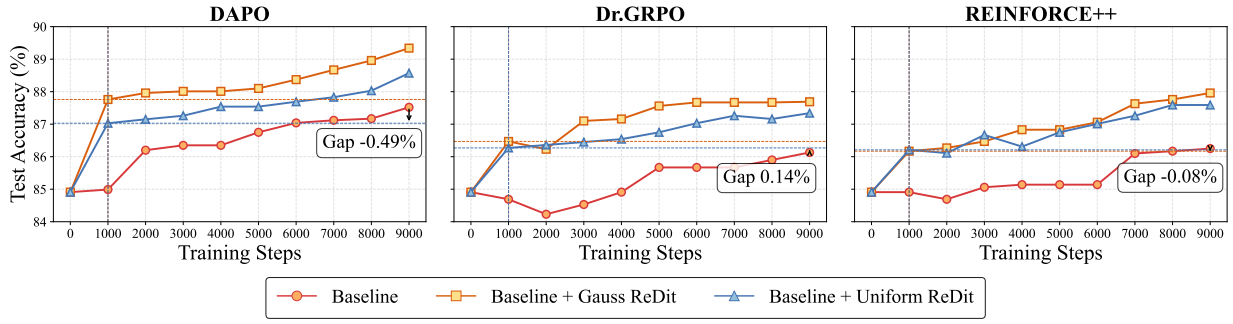

Figure 8. Accuracy of different GRPO variants tested on the GSM8K dataset. The horizontal dashed line highlights the performance of using ReDit at about 1000 training steps, and even after 9000 steps, its accuracy is comparable to the baseline.

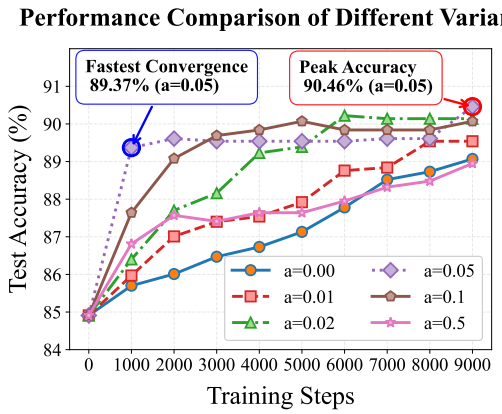

Figure 9. Appropriate perturbation achieves the best performance.

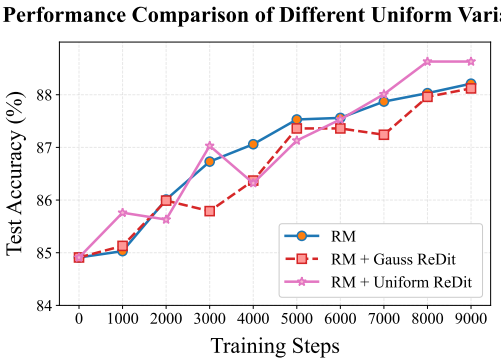

Figure 10. ReDit has little effect on improving the performance of GRPO based RM.

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.

Comparison with Direct Gradient Manipulation Baselines. We benchmark ReDit against established techniques that directly address gradient instability: Gradient Clipping (Zhang et al., 2020), which mitigates exploding gradients, and Dynamic Sampling (Yu et al., 2025), 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.

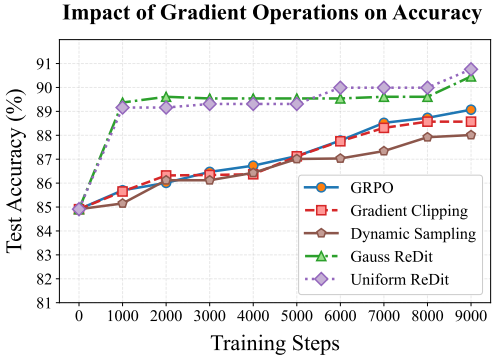

Figure 11. Appropriate perturbation achieves the best performance.

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 (Shao et al., 2024)), not token-level rewards. We investigate how Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ injection improves convergence. The perturbed objective is:

$$\tilde{J}(\pi_\theta) = \mathbb{E}_{q\sim p_Q}\left[\mathbb{E}_{o\sim\pi_\theta(\cdot|q)}\tilde{R}(q, o)\right], \tag{7}$$

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

Proposition 6.1 (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 6.1 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.1 for a detailed proof.

Proposition 6.2 (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:*

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

Remark. In Proposition 6.2, 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.2.

Proposition 6.3 (Optimization time upper and lower bounds)**.** *When we introduce Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ to the reward, it not only increases the variance of the reward, but also reduces the accuracy of the originally perfect reward function. According to (Razin et al., 2025), for the target $J(\pi_\theta)$ (Eq. (7)), the time t_γ required for the real reward r_G to increase γ satisfies:*

$$\Omega\left(\left(\mathbb{E}_{q\sim p_Q}\left[\mathrm{var}_{o\sim\pi_\theta(\cdot|q)}R(q, o)\right] + \sigma^2\right)^{-\frac{1}{3}}\right) \leq t_\gamma.$$

$$t_\gamma \leq \mathcal{O}\left(\pi_{\theta(0)}(\mathbf{y}^\gamma|\mathbf{x})^{-1}\right).$$

Remark. Proposition 6.3 analyzes the time required for a policy to reach or exceed a certain performance threshold starting from $\pi_{\theta(0)}$. Compared with optimization using the unperturbed reward, the perturbed reward introduces a larger reward variance, which can reduce the lower bound of the convergence time. Meanwhile, the perturbation introduces inaccuracies relative to the perfect reward. This inaccuracy limits the upper bound of the convergence time. This advantage arises because the random noise in the perturbation can effectively improve the perturbed rewards of non-zero probability outputs under the initial policy $\pi_{\theta(0)}$, thereby effectively encouraging broader exploration, see Fig. 1. This makes these outputs easier to perform gradient ascent, which helps to discover outputs with higher true rewards. For detailed proof, see Appendix B.3.

7. Limitations and Conclusions

ReDit enhances RL by introducing zero-mean noise to discrete rewards, effectively smoothing gradients, preventing gradient pathologies, and accelerating convergence through increased reward variance. Our empirical evaluation across multiple benchmarks confirms these benefits, demonstrating improvements in both speed and performance. Though effective, the approach requires perturbation variance tuning—currently done through experimentation. Future work will focus on automating this parameter selection.

References

Abdul Hameed, M. S., Chadha, G. S., Schwung, A., and Ding, S. X. Gradient monitored reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):4106–4119, 2023. doi: 10.1109/TNNLS.2021.3119853.

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

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

Cao, M., Shu, L., Yu, L., Zhu, Y., Wichers, N., Liu, Y., and Meng, L. Enhancing reinforcement learning with dense rewards from language model critic. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9119–9138, Miami, Florida, USA, November 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.515. URL https://aclanthology.org/2024.emnlp-main.515/.

Cao, Y., Zhao, H., Cheng, Y., Shu, T., Chen, Y., Liu, G., Liang, G., Zhao, J., Yan, J., and Li, Y. Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2024b. ISSN 2162-2388. doi: 10.1109/tnnls.2024.3497992. URL http://dx.doi.org/10.1109/TNNLS.2024.3497992.

Chan, A. J., Sun, H., Holt, S., and van der Schaar, M. Dense reward for free in reinforcement learning from human feedback. In *International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=eyxVRMrZ4m.

Chan, H., Mnih, V., Behbahani, F., Laskin, M., Wang, L., Pardo, F., Gazeau, M., Sahni, H., Horgan, D., Baumli, K., Schroecker, Y., Spencer, S., Steigerwald, R., Quan, J., Comanici, G., Flennerhag, S., Neitz, A., Zhang, L. M., Schaul, T., Singh, S., Lyle, C., Rocktäschel, T., Parker-Holder, J., and Holsheimer, K. Vision-language models as a source of rewards. In *Second Agent Learning in Open-Endedness Workshop*, 2023. URL https://openreview.net/forum?id=Xw1hVTWxxQ.

Chatterji, N. S., Pacchiano, A., Bartlett, P. L., and Jordan, M. I. 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. ISBN 9781713845393.

Chen, Y., Zhu, D., Sun, Y., Chen, X., Zhang, W., and Shen, X. The accuracy paradox in RLHF: When better reward models don't yield better language models. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2980–2989, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.174. URL https://aclanthology.org/2024.emnlp-main.174/.

Christiano, P. F., Leike, J., Brown, T. B., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 4302–4310, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Chu, X., Huang, H., Zhang, X., Wei, F., and Wang, Y. Gpg: A simple and strong reinforcement learning baseline for model reasoning, 2025. URL https://arxiv.org/abs/2504.02546.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Contributors, O. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass, 2023.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang,

X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Goyal, P., Niekum, S., and Mooney, R. J. Using natural language for reward shaping in reinforcement learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, pp. 2385–2391. AAAI Press, 2019. ISBN 9780999241141.

Hallak, A., Castro, D. D., and Mannor, S. Contextual markov decision processes, 2015. URL https://arxiv.org/abs/1502.02259.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Hu, J., Liu, J. K., and Shen, W. Reinforce++: An efficient rlhf algorithm with robustness to both prompt and reward models, 2025. URL https://arxiv.org/abs/2501.03262.

Ivison, H., Wang, Y., Liu, J., Wu, Z., Pyatkin, V., Lambert, N., Smith, N. A., Choi, Y., and Hajishirzi, H. Unpacking DPO and PPO: Disentangling best practices for learning from preference feedback. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=JMBWTlazjW.

Kaufmann, T., Weng, P., Bengs, V., and Hüllermeier, E. A survey of reinforcement learning from human feedback, 2024. URL https://arxiv.org/abs/2312.14925.

Kong, D. and Yang, L. F. 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. ISBN 9781713871088.

Lambert, N. Reinforcement learning from human feedback, 2025. URL https://arxiv.org/abs/2504.12501.

Lang, H., Huang, F., and Li, Y. Fine-tuning language models with reward learning on policy. In Duh, K., Gomez, H., and Bethard, S. (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1382–1392, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.75. URL https://aclanthology.org/2024.naacl-long.75/.

Liu, Z., Chen, C., Li, W., Qi, P., Pang, T., Du, C., Lee, W. S., and Lin, M. Understanding r1-zero-like training: A critical perspective, 2025. URL https://arxiv.org/abs/2503.20783.

Lu, P., Gong, R., Jiang, S., Qiu, L., Huang, S., Liang, X., and Zhu, S.-C. Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning, 2021. URL https://arxiv.org/abs/2105.04165.

Ma, H., Fu, G., Luo, Z., Wu, J., and Leong, T.-Y. Exploration by random reward perturbation, 2025. URL https://arxiv.org/abs/2506.08737.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han,

J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, J. H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Michael, Pokorny, Pokrass, M., Pong, V. H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M. B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Gray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=TG8KACxEON.

Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=HPuSIXJaa9.

Razin, N., Zhou, H., Saremi, O., Thilak, V., Bradley, A., Nakkiran, P., Susskind, J. M., and Littwin, E. Vanishing gradients in reinforcement finetuning of language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IcVNBR7qZi.

Razin, N., Wang, Z., Strauss, H., Wei, S., Lee, J. D., and Arora, S. What makes a reward model a good teacher? an optimization perspective, 2025. URL https://arxiv.org/abs/2503.15477.

Rengarajan, D., Vaidya, G., Sarvesh, A., Kalathil, D., and Shakkottai, S. Reinforcement learning with sparse rewards using guidance from offline demonstration. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=YJ1WzgMVsMt.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Shao, R., Li, S. S., Xin, R., Geng, S., Wang, Y., Oh, S., Du, S. S., Lambert, N., Min, S., Krishna, R., Tsvetkov, Y., Hajishirzi, H., Koh, P. W., and Zettlemoyer, L. Spurious rewards: Rethinking training signals in rlvr, 2025. URL https://arxiv.org/abs/2506.10947.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Tariq, S., Chhetri, M. B., Nepal, S., and Paris, C. A2c: A modular multi-stage collaborative decision framework for human-ai teams, 2024. URL https://arxiv.org/abs/2401.14432.

Vasan, G., Wang, Y., Shahriar, F., Bergstra, J., Jagersand, M., and Mahmood, A. R. Revisiting sparse rewards for goal-reaching reinforcement learning, 2024. URL https://arxiv.org/abs/2407.00324.

von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., Huang, S., Rasul, K., and Gallouédec, Q. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

Wang, Z., Feng, P., Lin, Y., Cai, S., Bian, Z., Yan, J., and Zhu, X. Crowdvlm-r1: Expanding r1 ability to vision language model for crowd counting using fuzzy group relative policy reward, 2025. URL https://arxiv.org/abs/2504.03724.

Weaver, L. and Tao, N. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, pp. 538–545, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558608001.

Wen, X., Lou, J., Lu, Y., Lin, H., XingYu, Lu, X., He, B., Han, X., Zhang, D., and Sun, L. Rethinking reward model evaluation: Are we barking up the wrong tree? In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Cnwz9jONi5.

Xie, T., Zhao, S., Wu, C. H., Liu, Y., Luo, Q., Zhong, V., Yang, Y., and Yu, T. Text2reward: Reward shaping with language models for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tUM39YTRxH.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Fan, T., Liu, G., Liu, L., Liu, X., Lin, H., Lin, Z., Ma, B., Sheng, G., Tong, Y., Zhang, C., Zhang, M., Zhang, W., Zhu, H., Zhu, J., Chen, J., Chen, J., Wang, C., Yu, H., Dai, W., Song, Y., Wei, X., Zhou, H., Liu, J., Ma, W.-Y., Zhang, Y.-Q., Yan, L., Qiao, M., Wu, Y., and Wang, M. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Zhang, H., Lei, Y., Gui, L., Yang, M., He, Y., Wang, H., and Xu, R. CPPO: Continual learning for reinforcement learning with human feedback. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=86zAUE80pP.

Zhang, J., He, T., Sra, S., and Jadbabaie, A. Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJgnXpVYwS.

Zhang, K., Hong, Y., Bao, J., Jiang, H., Song, Y., Hong, D., and Xiong, H. Gvpo: Group variance policy optimization for large language model post-training, 2025. URL https://arxiv.org/abs/2504.19599.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019. URL https://arxiv.org/abs/1909.08593.

A. Related Work

Reinforcement Learning with Discrete Rewards. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) 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 (Yu et al., 2025) introduced a dynamic sampling strategy to improve gradient effectiveness by dynamically filtering invalid samples, thereby increasing sample efficiency, although this reduced training speed. CPPO (Zhang et al., 2024) 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 (Chu et al., 2025) 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 (Liu et al., 2025) 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.

The Reward Design Challenge in LLM Reinforcement Learning.

The predominant approach to aligning Large Language Models (LLMs), Reinforcement Learning from Human Preferences (RLHF), relies on a learned reward model to score model outputs (Christiano et al., 2017). However, this paradigm introduces a fundamental trade-off between reward accuracy and variance. On one hand, low-fidelity reward models are prone to reward hacking, where the policy model exploits inaccuracies in the reward signal rather than achieving the intended goal (Ivison et al., 2024; Chen et al., 2024; Wen et al., 2025). On the other hand, increasing the reward model's accuracy often reduces reward variance, which can lead to vanishing gradients and subsequently slow down policy optimization (Razin et al., 2024). This accuracy-variance dilemma is formalized in recent theoretical work, which posits that an effective reward function must strike a balance between its bias and variance (Razin et al., 2025). Several lines of work attempt to navigate this challenge. For instance, GRPO (Shao et al., 2024; Liu et al., 2025) sidesteps reward model inaccuracies by using a deterministic, high-accuracy reward function. However, by providing sparse, low-variance rewards, it can exacerbate gradient instability and hinder optimization. Other approaches rely on heuristics. For instance, the addition of spurious rewards has shown empirical gains but lacks a theoretical foundation and has only been validated on a narrow range of models (Shao et al., 2025). In stark contrast, our method, ReDit, is not only underpinned by a rigorous theoretical framework but has also demonstrated broad effectiveness and applicability across various LLMs. Similarly, while methods like Random Reward Perturbation (RRP) (Ma et al., 2025) also use perturbations, their focus is on improving sample efficiency in classic RL algorithms like PPO (Schulman et al., 2017), DQN (Mnih et al., 2015) and A2C (Tariq et al., 2024), not on accelerating the convergence of modern LLM policy optimization frameworks like GRPO. Our work, therefore, is specifically designed to address the aforementioned accuracy-variance trade-off by introducing structured perturbations to a high-accuracy reward signal, preserving low bias while injecting sufficient variance for stable and efficient optimization.

B. Theorems and proofs

B.1. Proof of Proposition 6.1

The proof of Proposition 6.1 is expressed as follows:

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

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{q\sim p_Q}\mathbb{E}_{o\sim\pi_\theta(\cdot|q)}\left[R(q,o)\nabla_\theta \log \pi_\theta(o|q)\right]. \tag{8}$$

For the noise-injected objective, its gradient becomes:

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

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

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

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

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

$$\mathbb{E}[\nabla_\theta \tilde{J}] = \mathbb{E}[\nabla_\theta J] + 0 = \mathbb{E}[\nabla_\theta J]. \tag{12}$$

□

B.2. Proof of Proposition 6.2

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:

$$\nabla \hat{\tilde{J}}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left[\nabla \log \pi_\theta(o_i|q_i)\cdot(R(q_i,o_i)+\epsilon_i)\right], \tag{13}$$

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

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

Under this condition, the Eq. (13) simplifies to:

$$\nabla \hat{\tilde{J}}(\theta) = \underbrace{\nabla \hat{J}(\theta)}_{\text{origin gradient}} + \underbrace{\frac{1}{n}\sum_{i=1}^{n}\left[\nabla \log \pi_\theta(o_i|q_i)\cdot\epsilon_i\right]}_{\text{noise gradient}}. \tag{15}$$

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:

$$\text{Var}\left(\epsilon \cdot \nabla_\theta \log \pi_\theta(o|q)\right) = \mathbb{E}\left[\epsilon^2 \cdot \|\nabla_\theta \log \pi_\theta(o|q)\|^2\right] - \left(\mathbb{E}\left[\epsilon \cdot \nabla_\theta \log \pi_\theta(o|q)\right]\right)^2. \tag{16}$$

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

$$\mathbb{E}[\epsilon^2] = \text{Var}(\epsilon) + (\mathbb{E}[\epsilon])^2 = \sigma^2 + 0 = \sigma^2. \tag{17}$$

This allows us to simplify the variance expression to:

$$\text{Var(noise gradient)} = \text{Var}\left(\epsilon \cdot \nabla_\theta \log \pi_\theta\right) = \sigma^2 \cdot \mathbb{E}\left[\|\nabla_\theta \log \pi_\theta(o|q)\|^2\right] > 0, \tag{18}$$

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

□

B.3. Proof of Proposition 6.3

The proof of Proposition 6.3 is expressed as follows:

Proof. Suppose the original reward model is $r_{RM}(\mathbf{x},\mathbf{y})$. We add a zero-mean Gaussian random variable ϵ with variance σ^2 to it, resulting in the new reward signal $\tilde{r}_{RM}(\mathbf{x},\mathbf{y})$:

$$\tilde{r}_{RM}(\mathbf{x},\mathbf{y}) = r_{RM}(\mathbf{x},\mathbf{y}) + \epsilon, \tag{19}$$

where $\epsilon \sim \mathcal{N}(0,\sigma^2)$, and we assume that ϵ is independent of $\mathbf{y}$ (conditioned on $\mathbf{x}$).

1. Change in Reward Variance (Based on Definition 2). Definition 2 defines the reward variance induced by the reward model r_{RM} under policy π_θ and prompt $\mathbf{x}$ as:

$$\text{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r_{RM}(\mathbf{x},\mathbf{y})] := \mathbb{E}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}\left[\left(r_{RM}(\mathbf{x},\mathbf{y}) - \mathbb{E}_{\mathbf{y}'\sim\pi_\theta(\cdot|\mathbf{x})}[r_{RM}(\mathbf{x},\mathbf{y}')]\right)^2\right]. \tag{20}$$

This is the standard definition of variance: the expected squared deviation from the mean.

Now, we compute the variance of the perturbed reward signal $\tilde{r}_{RM}$:

$$\text{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[\tilde{r}_{RM}(\mathbf{x},\mathbf{y})] = \text{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r_{RM}(\mathbf{x},\mathbf{y}) + \epsilon]. \tag{21}$$

Using the property of variance for two random variables A and B:

$$\text{Var}(A+B) = \text{Var}(A) + \text{Var}(B) + 2\text{Cov}(A,B). \tag{22}$$

Here, $A = r_{RM}(\mathbf{x},\mathbf{y})$ (a random variable depending on $\mathbf{y}$), and $B = \epsilon$ (independent of $\mathbf{y}$). Since ϵ is independent of $\mathbf{y}$, A and B are independent, so $\text{Cov}(A,B) = 0$. Therefore:

$$\text{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r_{RM}(\mathbf{x},\mathbf{y}) + \epsilon] = \text{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r_{RM}(\mathbf{x},\mathbf{y})] + \text{Var}(\epsilon). \tag{23}$$

Since $\text{Var}(\epsilon) = \sigma^2$, the perturbed reward variance becomes:

$$\text{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[\tilde{r}_{RM}(\mathbf{x},\mathbf{y})] = \text{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r_{RM}(\mathbf{x},\mathbf{y})] + \sigma^2. \tag{24}$$

Conclusion: If $\sigma^2 > 0$, adding zero-mean Gaussian noise to the reward model increases the reward variance by exactly σ^2. This aligns with our intuitive understanding based on Theorem 3.3.

2. Change in Reward Model Accuracy (Based on Definition 1). Definition 1 defines the accuracy of a reward model r_{RM} at a given prompt $\mathbf{x}$ and distribution $\mathcal{D}$ over unordered output pairs as:

$$\text{acc}_{\mathbf{x},\mathcal{D}}(r_{RM}) := \mathbb{E}_{(\mathbf{y},\mathbf{y}')\sim\mathcal{D}}\left[\mathbb{1}\left[\text{sign}(r_{RM}(\mathbf{x},\mathbf{y}) - r_{RM}(\mathbf{x},\mathbf{y}')) = \text{sign}(r_G(\mathbf{x},\mathbf{y}) - r_G(\mathbf{x},\mathbf{y}'))\right]\right], \tag{25}$$

where r_G is the ground truth reward function. This measures the probability that the reward model correctly ranks a pair $(\mathbf{y},\mathbf{y}')$ relative to the ground truth.

Now consider the accuracy of the perturbed reward model $\tilde{r}_{RM}$:

$$\text{acc}_{\mathbf{x},\mathcal{D}}(\tilde{r}_{RM}) = \mathbb{E}_{(\mathbf{y},\mathbf{y}')\sim\mathcal{D}}\left[\mathbb{1}\left[\text{sign}(\tilde{r}_{RM}(\mathbf{x},\mathbf{y}) - \tilde{r}_{RM}(\mathbf{x},\mathbf{y}')) = \text{sign}(r_G(\mathbf{x},\mathbf{y}) - r_G(\mathbf{x},\mathbf{y}'))\right]\right]. \tag{26}$$

We analyze the difference:

$$\tilde{r}_{RM}(\mathbf{x},\mathbf{y}) - \tilde{r}_{RM}(\mathbf{x},\mathbf{y}') = (r_{RM}(\mathbf{x},\mathbf{y}) + \epsilon_1) - (r_{RM}(\mathbf{x},\mathbf{y}') + \epsilon_2), \tag{27}$$

where $\epsilon_1, \epsilon_2 \sim \mathcal{N}(0,\sigma^2)$ are independently sampled. Let:

$$\Delta r_{RM} = r_{RM}(\mathbf{x},\mathbf{y}) - r_{RM}(\mathbf{x},\mathbf{y}'), \quad \Delta r_G = r_G(\mathbf{x},\mathbf{y}) - r_G(\mathbf{x},\mathbf{y}'), \tag{28}$$

and define $\eta = \epsilon_1 - \epsilon_2 \sim \mathcal{N}(0, 2\sigma^2)$. Then:

$$\tilde{r}_{RM}(\mathbf{x},\mathbf{y}) - \tilde{r}_{RM}(\mathbf{x},\mathbf{y}') = \Delta r_{RM} + \eta. \tag{29}$$

The condition for accuracy becomes:

$$\text{sign}(\Delta r_{RM} + \eta) = \text{sign}(\Delta r_G), \tag{30}$$

compared to the original condition:

$$\text{sign}(\Delta r_{RM}) = \text{sign}(\Delta r_G). \tag{31}$$

We now analyze how the addition of noise affects this condition:

- If the original model is accurate, i.e., $\text{sign}(\Delta r_{RM}) = \text{sign}(\Delta r_G)$, then adding noise η may cause $\text{sign}(\Delta r_{RM} + \eta) \neq \text{sign}(\Delta r_G)$, especially when $|\eta|$ is large enough to flip the sign of Δr_{RM}.

- If the original model is inaccurate, i.e., $\text{sign}(\Delta r_{RM}) \neq \text{sign}(\Delta r_G)$, there is a small chance that η flips the sign back to match $\text{sign}(\Delta r_G)$, but this is not systematic and depends on the values of Δr_{RM} and Δr_G.

In general, since η is independent of both Δr_{RM} and Δr_G, it is more likely to disrupt the correct sign relationship rather than correct it. Thus, adding random noise tends to make the ranking decisions more random.

Conclusion: Adding zero-mean Gaussian noise to the reward model makes its relative ranking of output pairs more random, thereby **reducing** the reward model's accuracy (acc).

3. Upper and lower bounds of training time). According to Theorem 3.3, the lower bound on the training time t_γ is influenced by the variance of the reward function. Specifically, using Equation (24), we derive that the lower bound on t_γ satisfies:

$$\Omega\left(\left(\mathbb{E}_{q\sim p_Q}\left[\text{Var}_{o\sim\pi_\theta(\cdot|q)}R(q,o)\right]+\sigma^2\right)^{-\frac{1}{3}}\right)\leq t_\gamma. \tag{32}$$

According to Theorem 3.4, in some cases (particularly when a perfectly accurate model would lead to arbitrarily slow training), an inaccurate model could instead lead to faster increases in the true reward. We derive that the lower bound on t_γ satisfies:

$$t_\gamma \leq \mathcal{O}(\pi_{\theta(0)}(\mathbf{y}^\gamma|\mathbf{x})^{-1}) \tag{33}$$

□

C. Training Dynamic

In this section, we show more Training Dynamic information.

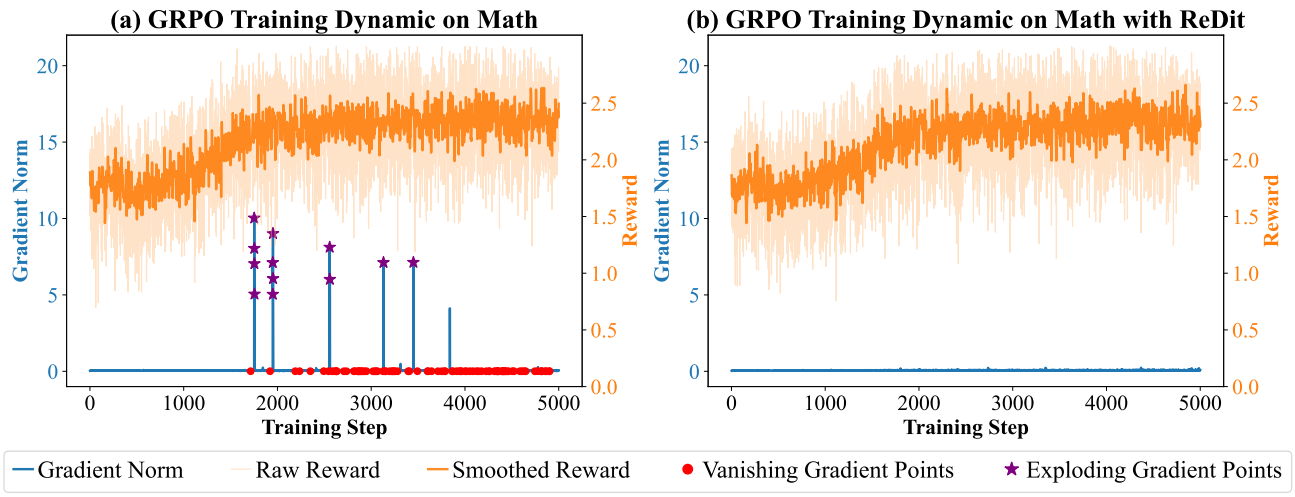

Figure 12. Training Dynamics of Gradient Norm and Reward on Math Dataset.

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

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.

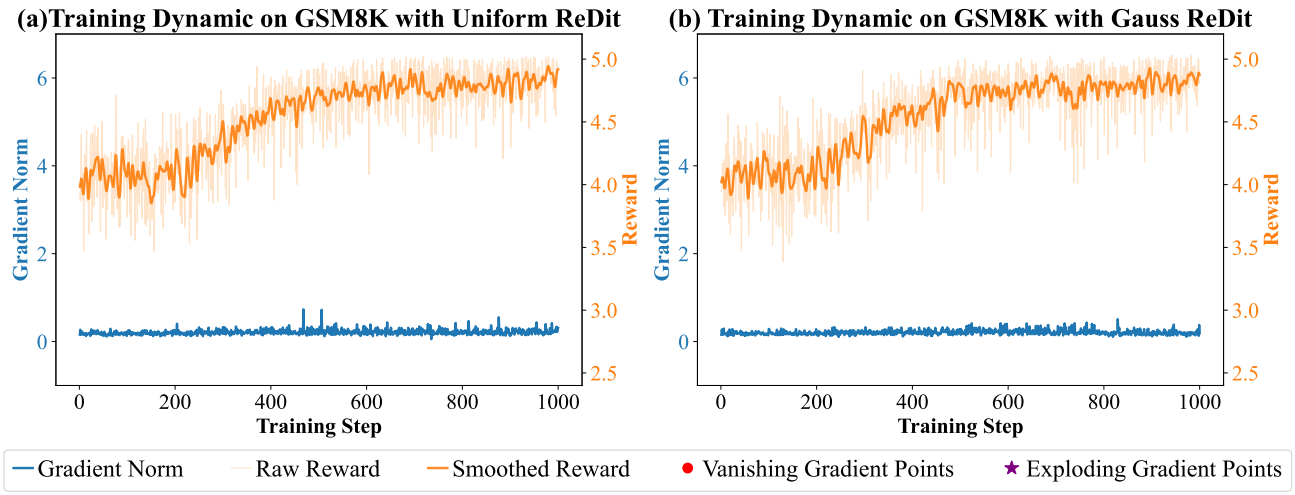

Figure 13. Training dynamics of gradient norm and reward on the GSM8K dataset, showing the impact of perturbations of different distributions.

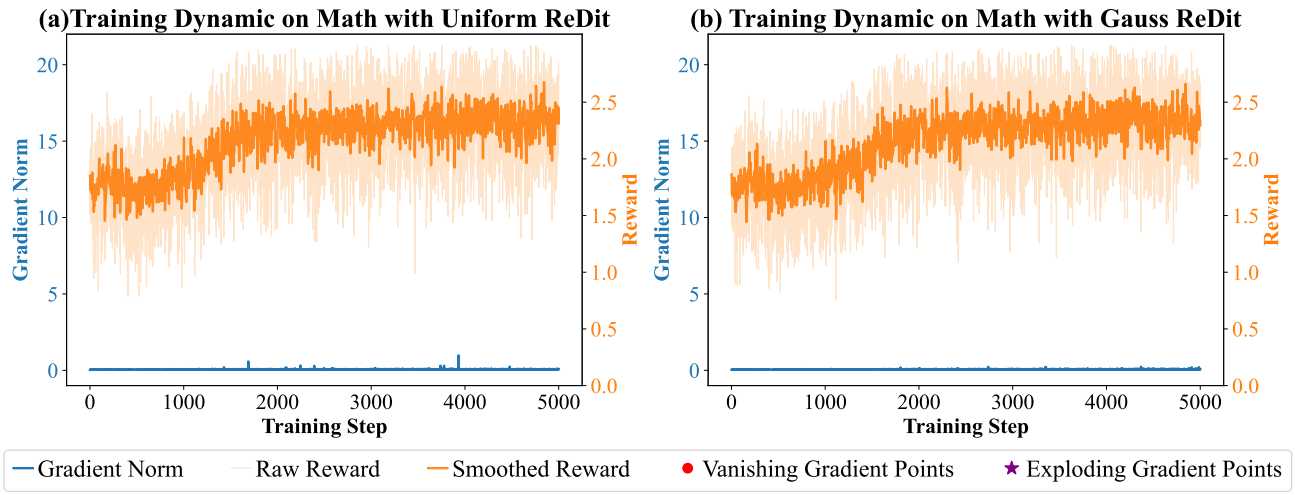

Figure 14. Training dynamics of gradient norm and reward on the Math dataset, showing the impact of perturbations of different distributions.

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}
```

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

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

```
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

```
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

```
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

```
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

```
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

E. More result

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

E.1. Main Result

In this section, we show the results in Figure 5, the performance of GRPO and GRPO+ReDit on different datasets.

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

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.

E.2. Baseline Result

In this section, we present all numerical results in Fig. 8. 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.

E.3. Different LLMs Result

In this section, we present all numerical results in Fig. 6.

E.4. Reward Model Result

In this section, we present all numerical results in Fig. 10.

E.5. Direct Gradient Manipulation Result

In this section, we present all numerical results in Fig. 11.

E.6. Variance Result

In this section, we show more results on the performance of ReDit as the perturbation changes. As shown in Figure 15, 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 17 and 18.

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

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

E.7. Scheduled Perturbation Result

In this section, we show the changing trends of different scheduled perturbation strategies, as shown in Figure 16. We took the perturbation of Gauss distribution as an example and conducted experiments. The experimental results are shown in Table 19. The CosineReverse strategy shows the best performance.

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. Llama-3.2-3B-Instruct performance comparison of different training steps on the GSM8K dataset

Method \Step	0	1000	2000	3000	4000	5000	6000	7000	8000	9000
Instruct model	73.62	-	-	-	-	-	-	-	-	-
GRPO	-	73.60	73.34	74.74	74.27	73.90	73.90	73.90	74.01	74.04
Uniform ReDit	-	74.81	74.81	74.89	75.05	75.05	75.66	76.13	76.27	76.27
Gauss ReDit	-	74.30	74.52	75.13	75.13	75.13	76.48	76.40	76.25	76.25

Table 12. Llama-3.1-8B-Instruct performance Comparison of Different Training Steps on the GSM8K Dataset

Method \Step	0	1000	2000	3000	4000	5000	6000	7000	8000	9000
Instruct model	81.05	-	-	-	-	-	-	-	-	-
GRPO	-	81.02	81.30	81.92	81.79	82.01	82.36	82.36	81.63	82.12
Uniform ReDit	-	83.81	83.24	83.45	84.32	84.01	84.32	84.12	84.73	84.92
Gauss ReDit	-	83.80	83.44	83.92	84.47	84.86	84.62	84.12	84.35	84.12

Table 13. Mistral-7B-Instruct-v0.3 performance Comparison of Different Training Steps on the GSM8K Dataset

Method \Step	0	1000	2000	3000	4000	5000	6000	7000	8000	9000
Instruct model	53.68	-	-	-	-	-	-	-	-	-
GRPO	-	56.33	58.00	57.70	57.39	58.23	58.23	57.92	58.91	59.14
Uniform ReDit	-	58.25	58.48	58.45	59.83	59.68	59.23	60.29	61.45	62.07
Gauss ReDit	-	58.57	58.85	58.68	59.68	59.29	60.68	60.61	61.61	61.76

Table 14. Ministral-8B-Instruct-2410 performance Comparison of Different Training Steps on the Geometry3K Dataset

Method \Step	0	1000	2000	3000	4000	5000	6000	7000	8000	9000
Instruct model	82.34	-	-	-	-	-	-	-	-	-
GRPO	-	82.94	83.85	84.53	84.84	84.84	85.22	84.61	84.15	83.70
Uniform ReDit	-	85.39	85.78	85.69	85.69	85.91	86.90	86.90	87.01	87.23
Gauss ReDit	-	85.24	85.46	85.44	85.44	86.22	86.44	86.69	86.76	86.12

Table 15. Performance Comparison at Different Training Steps on Reward Model

Method \Step	0	1000	2000	3000	4000	5000	6000	7000	8000	9000
Instruct model	84.91	-	-	-	-	-	-	-	-	-
RM	-	85.03	86.01	86.73	87.06	87.53	87.56	87.87	88.03	88.21
RM + Gauss ReDit	-	85.13	85.99	85.79	86.37	87.36	87.36	87.24	87.96	88.12
RM + Uniform ReDit	-	85.76	85.63	87.03	86.32	87.13	87.53	88.01	88.63	88.63

Table 16. Performance Comparison at Different Training Steps on Different Gradient Manipulation

Method \Step	0	1000	2000	3000	4000	5000	6000	7000	8000	9000
Instruct model	84.91	-	-	-	-	-	-	-	-	-
GRPO	-	85.03	86.01	86.73	87.06	87.53	87.56	87.87	88.03	88.21
Gradient Clipping	-	85.65	86.32	86.34	86.37	87.12	87.75	88.31	88.57	88.57
Dynamic Sampling	-	85.15	86.12	86.12	86.43	87.01	87.03	87.34	87.92	88.01
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

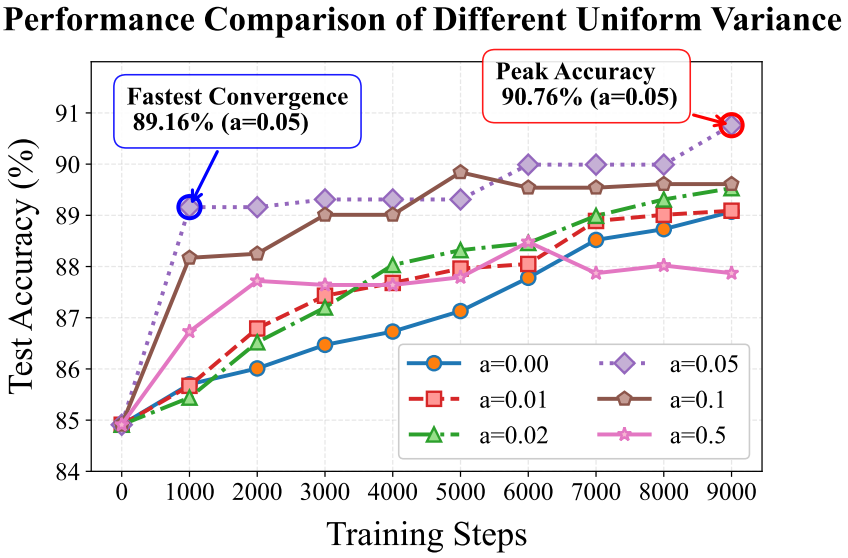

Figure 15. ReDit uniform perturbation performance changes with variance.

Table 17. 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 18. 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

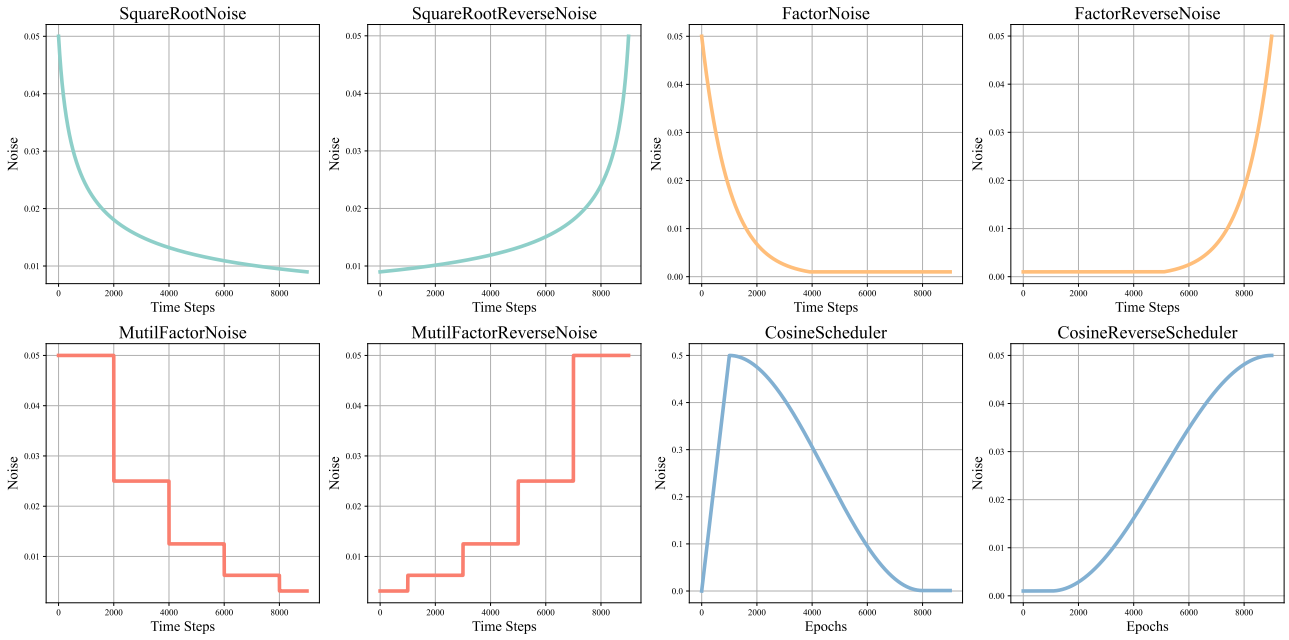

Figure 16. ReDit scheduled perturbation Variance trend with training step (taking the original variance as 0.05 as an example)

Table 19. 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