Advancing Reasoning in Diffusion Language Models with Denoising Process Rewards

Shaoan Xie[1,2,*] **Lingjing Kong**[1,*] **Xiangchen Song**[1]
Xinshuai Dong[1] **Guangyi Chen**[1,2] **Eric P. Xing**[1,2] **Kun Zhang**[1,2]

[1] Carnegie Mellon University
[2] Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE
[*] Equal contribution

Abstract

Diffusion-based large language models offer a non-autoregressive alternative for text generation, but enabling them to perform complex reasoning remains challenging. Reinforcement learning has recently emerged as an effective post-training strategy for improving their performance; however, existing methods rely primarily on outcome-based rewards, which provide no direct supervision over the denoising process and often result in poorly structured reasoning that is difficult to interpret and inconsistently supports the final prediction. To address this limitation, we introduce *denoising process reward*, a process-level reinforcement signal defined over the denoising trajectory of diffusion language models. This reward is obtained by estimating the contribution of intermediate denoising intervals to the final task outcome, encouraging the model to favor reasoning trajectories that consistently guide generation toward correct predictions. We further propose an efficient stochastic estimator that reuses standard training rollouts, enabling practical process-level supervision at scale. Experiments on challenging reasoning benchmarks demonstrate that our approach yields consistent improvements in reasoning stability, interpretability, and overall task performance.

1 Introduction

Diffusion large language models (dLLMs) have emerged as a compelling alternative to traditional autoregressive models (ARMs), offering the potential to significantly accelerate inference through parallel generation (Nie et al., 2025; Sahoo et al., 2024; Gong et al., 2024; Ye et al., 2025). In particular, mask based dLLMs (MdLLMs) initialize generation from a sequence of `[MASK]` tokens and iteratively refine this sequence into coherent text through a denoising process. While this paradigm has shown promise on a variety of general tasks, enabling MdLLMs to perform complex multi step reasoning remains a significant challenge.

A common approach to improving reasoning in large language models is reinforcement learning. However, existing reinforcement learning methods, such as GRPO (Shao et al., 2024) and its diffusion adaptation diffu-GRPO (Zhao et al., 2025a), typically rely on sparse outcome based rewards derived from final answer correctness. Such rewards provide no direct guidance over the denoising process itself. As a result, MdLLMs are not explicitly encouraged to utilize intermediate denoising steps effectively, often spending them on repetitive loops or vacuous content and forcing the final denoising steps to determine the output. This behavior increases the risk of hallucinations or inconsistent reasoning that coincidentally yields the correct answer, as illustrated in Figure 1.

To overcome the limitations of outcome only supervision, prior work on autoregressive language models has explored the use of process supervision. In particular, Process Reward Models (PRMs) (Uesato et al., 2022; Lightman et al., 2023) are trained using human annotated reasoning steps and applied at inference time to score intermediate reasoning trajectories, selecting responses with higher process level rewards. This approach has been shown to significantly improve final performance.

However, directly extending this strategy to MdLLMs is challenging for three key reasons, as illustrated in Figure 4 and Table 2. First, intermediate generations in MdLLMs consist of partially specified sequences with unresolved mask tokens and parallel token updates, which differ fundamentally from the complete and sequential reasoning traces used to train PRMs. This distribution mismatch leads to unreliable reward signals, unstable training dynamics, and failure cases where the final output reward becomes invalid. Second, it is vulnerable to reward hacking behaviors, where the model learns to increase the process reward score without genuinely improving reasoning quality. Third, incorporating PRMs into training introduces substan-

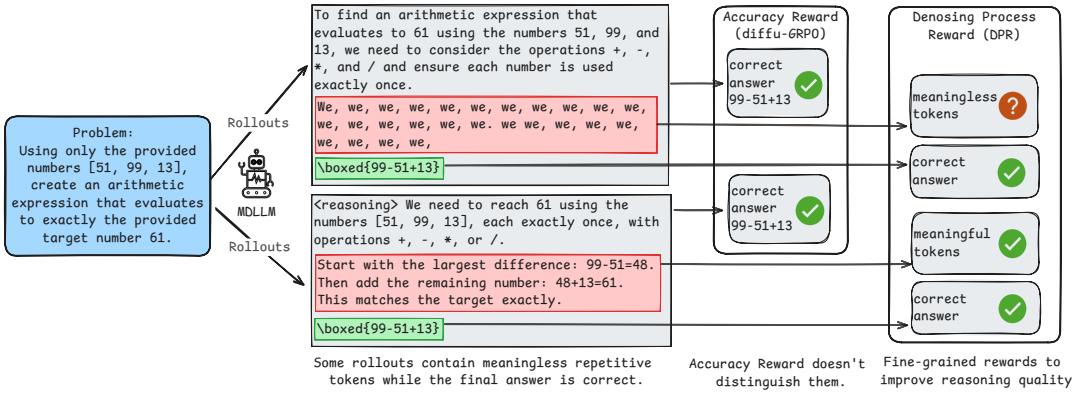

Figure 1: Limitations of outcome-only rewards and motivation for the proposed denoising process reward (DPR). Under outcome-only supervision, intermediate denoising steps are often spent on repetitive or vacuous content and are not effectively utilized to inform the final prediction, resulting in low-quality and uninterpretable reasoning processes. Consequently, the final prediction can be sensitive and unstable, motivating DPR to explicitly supervise intermediate denoising steps to improve reasoning quality and final performance.

tial computational overhead. Scoring intermediate states with a separate reward model substantially increases GPU memory consumption and leads to a pronounced slowdown in training. Together, these issues make PRM based supervision difficult to apply and scale in the diffusion setting.

These limitations motivate the need for an alternative form of supervision that is compatible with the denoising dynamics of MdLLMs. In this work, we propose an automatic reinforcement learning method for MdLLMs based on *Denoising Process Rewards (DPR)*. The key idea is to derive process rewards directly from the denoising process by estimating the contribution of denoising intervals through comparisons of final task outcomes across different timesteps, without relying on external annotations or reward models.

Our contributions are summarized as follows:

1. We identify that MdLLMs exhibit ineffective and unstable reasoning, and propose a denoising process reward that provides process-level supervision without human annotation or external reward models.
2. We develop an efficient realization of DPR that estimates the contribution of denoising intervals by reusing standard rollout outcomes, enabling practical integration into existing reinforcement learning pipelines with minimal additional cost.
3. We conduct extensive experiments on challenging reasoning benchmarks, demonstrating that DPR consistently improves reasoning stability, interpretability, and overall task performance.

2 Related Work

Mask-based diffusion-based large language models. LLaDA (Nie et al., 2025) proposes a mask-based dLLMs. It gradually removes the mask token in each diffusion step. Based on LLaDA, diffu-GRPO (Zhao et al., 2025a) assumes the generated tokens are independent and proposes a randomly masked prompt to estimate the token probability for reinforcement learning with diffusion models. WINO (Hong et al., 2025) proposes a training-free sampling strategy to use a low confidence threshold to generate a draft response and use a high threshold for second verification. TSE (Wang et al., 2025c) observes that the answers generated in intermediate diffusion steps can also be correct and therefore proposes a weighted voting strategy to get the final answer. ReMDM (Wang et al., 2025a) proposes a remasking sampler to address the problem that the generated tokens in dLLMs cannot be revoked. wd1 (Tang et al., 2025) proposes a weighted likelihood estimation for the sequence. Many approaches have been proposed to improve the efficiency of dLLMs, such as KV-cache (Wu et al., 2025; Song et al., 2025; Liu et al., 2025b; Ma et al., 2025). MDLM (Sahoo et al., 2024) derives a continuous-time, Rao-Blackwellized objective for training mask-based dLLM. LongLLaDA (Liu et al., 2025a) proposes an NTK-based RoPE extrapolation to allow long-context text generation. DiffuCoder (Gong et al., 2025) proposes a coupled sampling scheme to estimate the likelihood for GRPO training. MDPO (He et al., 2025) introduces a running confidence remasking strategy to allow low-confidence tokens to be remasked again

during inference time. IGPO (Zhao et al., 2025b) addresses exploration failures in masked diffusion language models by inserting partial ground truth reasoning traces into masked tokens during online sampling, guiding policy updates under sparse rewards. TraceRL (Wang et al., 2025d) incorporates preferred inference trajectories into diffusion model training using a diffusion-based value model, enabling trajectory-level optimization for complex reasoning tasks. In contrast, our method avoids trajectory supervision and auxiliary value models, and instead derives process rewards directly from denoising outcomes.

Process reward model. Verification models have been shown to improve the multi-step reasoning ability of LLMs. Unlike the outcome verifier (Cobbe et al., 2021; Yu et al., 2023) which examines the correctness of the final outcome, the process reward models enhance feedback accuracy by identifying and localizing errors within generated responses. However, collecting step-wise feedback can be costly, especially with human annotators (Uesato et al., 2022; Lightman et al., 2023). Therefore, many efforts have been devoted to the automatic extraction of process rewards. One standard way to assess process correctness is by estimating, via Monte Carlo (MC) methods, the empirical probability of reaching the correct final answers. Given an intermediate step of reasoning, MATH-SHEPHERD (Wang et al., 2023) asks completers to finalize multiple subsequent reasoning processes and estimate the potential of this step based on the correctness of all decoded answers. (Luo et al., 2024) proposes a Monte Carlo Tree Search algorithm to identify the first error in the reasoning process. (Zhang et al., 2025) argues that the MC-based estimation can be noisy and requires an additional LLM-as-judge to filter the process reward data. Inspired by (Wang et al., 2023), (Wang et al., 2025b) constructs process rewards for multimodal LLMs. (Zhang et al., 2024a) proposes a tree search policy with process rewards. Implicit process rewards (Yuan et al., 2024; Cui et al., 2025) trains the outcome reward model and can obtain the token-level process reward as log-likelihood ratios of the policy and reference models.

3 Policy Optimization with Denoising Process Rewards

In this section, we study reinforcement learning for masked diffusion language models and improve both reasoning quality and final task performance by moving beyond outcome-only rewards to a denoising process reward that provides supervision over intermediate denoising steps.

3.1 Preliminary: Outcome-based Group Relative Policy Optimization (GRPO)

GRPO is an on-policy reinforcement learning algorithm that improves language models using relative outcome-based rewards (Shao et al., 2024). We briefly summarize the formulation below.

Response sampling. Given an input question $\mathbf{Q}$, the current policy π_θ generates G candidate responses $\{\mathbf{R}^{(1)}, \ldots, \mathbf{R}^{(G)}\}$. Each response is assigned an outcome reward r_i based on final answer correctness. A mean-normalized advantage is computed as

$$A_i = r_i - \text{mean}(\{r_j\}_{j=1}^{G}), \tag{1}$$

and this advantage is distributed uniformly across all tokens in $\mathbf{R}^{(i)}$.

Learning objective. GRPO follows Proximal Policy Optimization (PPO) (Schulman et al., 2017) and optimizes a clipped surrogate objective with KL regularization toward a reference policy π_{ref}:

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{\mathbf{Q},\{\mathbf{R}^{(i)}\}}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathbf{R}^{(i)}|}\sum_{k=1}^{|\mathbf{R}^{(i)}|} \min\Big(\rho_i^k A_i,\ \text{clip}(\rho_i^k, 1-\varepsilon, 1+\varepsilon)\, A_i\Big) - \beta\, D_{\text{KL}}(\pi_\theta(\cdot \mid \mathbf{Q}) \,\|\, \pi_{\text{ref}}(\cdot \mid \mathbf{Q}))\Bigg], \tag{2}$$

where $\rho_i^k = \frac{\pi_\theta(\mathbf{R}^{(i),k}|\mathbf{Q},\mathbf{R}^{(i),<k})}{\pi_{\theta_{\text{old}}}(\mathbf{R}^{(i),k}|\mathbf{Q},\mathbf{R}^{(i),<k})}$ denotes the token-level likelihood ratio. In the MdLLM setting, GRPO is adapted by approximating sequence likelihoods with a mean-field assumption that factorizes them into independent per-token probabilities, as in diffu-GRPO (Zhao et al., 2025a).

3.2 Policy Optimization with Denoising Process Rewards

A key limitation of outcome-only policy optimization is that it provides no signal about how intermediate denoising steps are used, allowing correct answers to be produced despite uninformative or unstable denoising trajectories (Figure 1). While process-level supervision is a natural remedy, directly applying process reward models (PRMs) to MdLLMs is unreliable, as intermediate diffusion states are partially specified and unordered, leading to noisy and unstable supervision (Figure 4, Table 2).

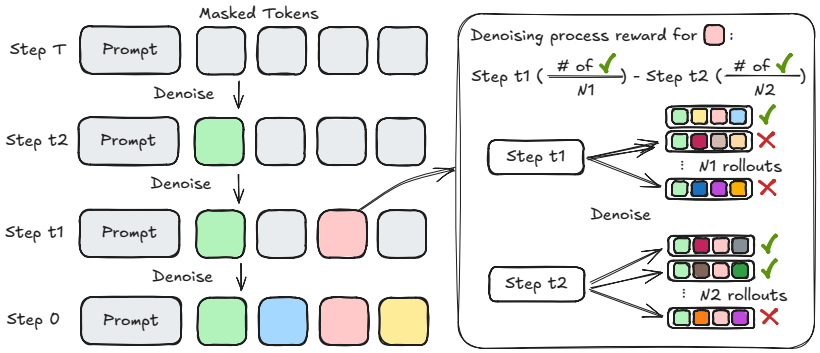

Figure 2: Illustration of the denoising process reward (DPR). We estimate the contribution of intermediate denoising steps by comparing outcome rewards from rollouts initialized at different timesteps. Larger differences indicate greater impact on the final answer.

To address this, we introduce the *denoising process reward (DPR)*, which provides a stable supervision signal over intermediate denoising steps and can be efficiently integrated into the GRPO framework.

Definition. Given a complete reasoning response $\mathbf{R}^{(j)}$, our goal is to evaluate the quality of the reasoning process and assign positive rewards to encourage the model to learn reasoning patterns that contribute to correct final answers. To this end, we decompose the denoising trajectory into segments defined by intermediate denoising steps, where each segment represents the progress made between two timesteps in the denoising process.

To measure the contribution of a denoising segment, we leverage the efficient parallel-decoding property of MdLLMs, which allows the model to complete generation from intermediate denoising states fast. Specifically, we ask the model to complete generation from the two endpoints of a segment and compare the resulting outcome accuracies. If completing from a later intermediate state yields a higher probability of producing a correct final answer than completing from an earlier state, then the denoising steps within that segment contribute positively and should be reinforced during training.

Based on this insight, we define the denoising process reward over the *entire* denoising trajectory, corresponding to the full interval from $t = T$ to $t = 0$. Let $\{x_t\}_{t=0}^{T}$ denote the denoising states along this trajectory. We account for the interval $[0, T]$ by aggregating contributions from all denoising sub-intervals $(t_1, t_2) \subseteq [0, T]$ with $0 \le t_1 < t_2 \le T$.

For each sub-interval, we generate complete response rollouts by continuing denoising from the intermediate states at timesteps t_1 and t_2, yielding rollout sets $\{\mathbf{R}^{(j)}(x_{t_1})\}_{j=1}^{N}$ and $\{\mathbf{R}^{(j)}(x_{t_2})\}_{j=1}^{N}$. The full denoising process reward for the current response is then defined as

$$R_{\text{full}} = \sum_{0 \le t_1 < t_2 \le T} \left(\frac{1}{N_1} \sum_{j=1}^{N} \mathbf{1}\left[\mathbf{R}^{(j)}(x_{t_1})\right] - \frac{1}{N_2} \sum_{j=1}^{N} \mathbf{1}\left[\mathbf{R}^{(j)}(x_{t_2})\right] \right), \tag{3}$$

where $\mathbf{1}[\cdot]$ indicates final answer correctness. Intuitively, this formulation attributes credit to denoising steps according to how much they improve the model's ability to reach a correct final answer. By aggregating contributions across all denoising sub-intervals, R_{full} provides a trajectory-level signal that rewards reasoning processes which consistently guide generation toward correct solutions.

Efficient Estimation of DPR via Reusing Rollouts. Although Eq. 3 defines DPR as an aggregation over all denoising sub-intervals along the trajectory, directly computing this quantity would require rollouts from many intermediate states and is therefore impractical. In practice, we adopt a stochastic and efficient estimator that reuses rollouts already generated during training.

Specifically, we fix the later endpoint of the interval to the fully masked initial state, i.e., $t_2 = T$. Rollouts from x_T correspond to standard generations conditioned only on the input question $\mathbf{Q}$ and are already available when computing outcome-based rewards in GRPO. We then sample a single timestep $t_1 \in \{0, \ldots, T-1\}$ and evaluate the contribution of denoising steps up to t_1 by comparing outcomes from completions starting at x_{t_1} and x_T.

Under this approximation, the denoising process

reward used for learning is given by

$$R_{\text{process}} := R_{\text{full}}(t_1, T), \tag{4}$$

which provides an efficient stochastic estimate of the contribution of intermediate denoising steps while incurring only a single additional rollout from the intermediate state. By sampling different values of t_1 across training updates, this estimator captures supervision over the full denoising trajectory in expectation and achieves competitive performance compared to the full reward, as shown in Table 2.

Integrating DPR via Up-Weighted Advantages. In GRPO, advantages are computed from normalized rewards within a minibatch (Eq. 1). If the denoising process reward were incorporated as a regular reward term, this normalization could over-penalize correct responses that exhibit weaker intermediate denoising progress, assigning them negative advantages despite producing correct final answers. This behavior is undesirable, as correctness must remain the primary learning criterion, with reasoning quality used only to refine learning among correct solutions.

To address this issue, we integrate DPR through an up-weighted advantage formulation that conditions process supervision on outcome correctness. For a rollout $\mathbf{R}^{(i)}$ with GRPO advantage A_i, we define the total advantage as

$$A_i^{\text{total}} = A_i + \mathbf{1}[A_i > 0] \cdot R_{\text{process}}. \tag{5}$$

Under this formulation, DPR selectively increases the gradient contribution of correct rollouts with better intermediate reasoning, while leaving the advantages of other rollouts unchanged. This up-weighting strategy avoids over-penalizing correct responses or rewarding incorrect ones, and leads to improved performance, as shown in Table 2.

4 Experiments

In this section, we present a comprehensive empirical evaluation of the proposed denoising process reward.

4.1 Setup

We build our model on top of diffu-GRPO (Zhao et al., 2025a) and adopt the same experimental setup unless otherwise specified. We provide implementation details in the Appendix.A.

Datasets. We evaluate on four benchmarks: (1) GSM8K (Cobbe et al., 2021), using 7,374 training and 1,319 test problems; (2) MATH (Lightman et al., 2023), with 7,500 training and 500 test problems; (3) COUNTDOWN (Pan et al., 2025), a synthetic dataset of 490K training and 256 test samples requiring arithmetic expression generation; and (4) SUDOKU (Black-Phoenix, 2024), 4×4 puzzles evaluated on a 256-sample split.

Baselines. We compare against recent state-of-the-art MdLLMs: LLaDA (Nie et al., 2025), Diffu-GRPO (Zhao et al., 2025a), TSE (Wang et al., 2025c), and WINO (Hong et al., 2025).

4.2 Results

Superior benchmark performance. Table 1 reports performance on GSM8K, MATH, COUNTDOWN, and SUDOKU. Our method achieves the best results on most benchmarks and sequence lengths, demonstrating that improving reasoning quality leads to more accurate final predictions.

Alignment of reasoning process and final answer. To assess how well MdLLMs produce intermediate reasoning that is consistent with the final answer, we analyze the alignment between the reasoning process and the output. Specifically, we input generations from LLaDA (Nie et al., 2025), diffu-GRPO (Zhao et al., 2025a), and our model into GPT-5, asking it to evaluate "whether a user can reach the final answer by following the reasoning step by step." Results on the COUNTDOWN and GSM8K datasets are shown in Fig. 5. Our method achieves substantially higher alignment ratios across both datasets. This large improvement helps explain the performance gains in Table 1, as our proposed reward explicitly encourages the model to maintain consistency between reasoning steps and final answers through the diffusion-based generation process. We also provide example outputs from the three models in Fig. 1. As shown, LLaDA and diffu-GRPO generate less meaningful reasoning in their responses and ultimately produce incorrect answers.

Comparison with pretrained PRMs. Since our approach provides process-level supervision through denoising process rewards, it is important to compare it with a widely used alternative: employing a pretrained Process Reward Model (PRM) as the training-time reward. To this end, we adopt the pretrained Mistral-7B PRM from Zhang et al. (2024b). Following prior practice, we insert reasoning-step tags every 16 timesteps during masked-token decoding, feed the resulting sequence into the PRM, and compute the process reward as the average PRM score over timestep

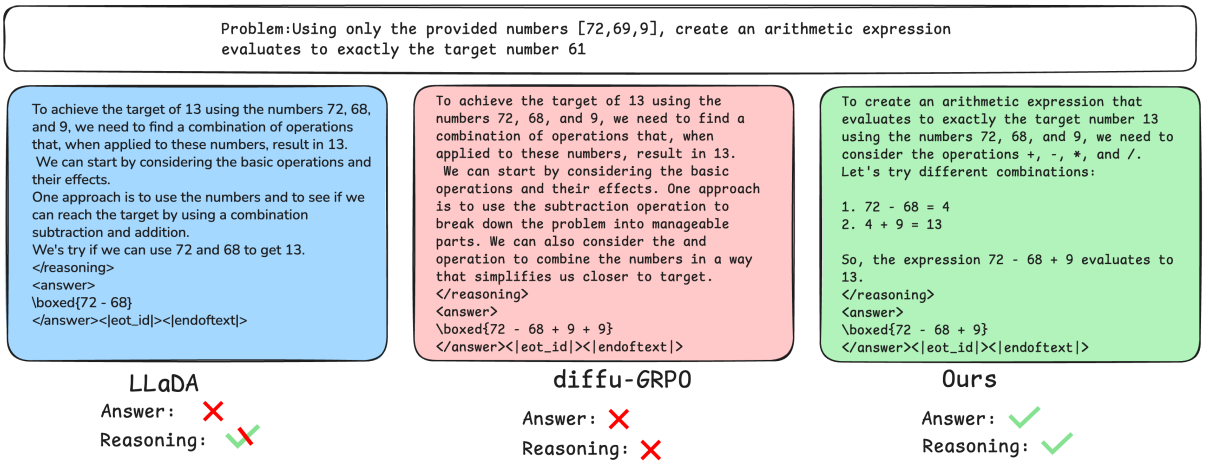

Figure 3: Comparison of generated responses across models. LLaDA (Nie et al., 2025) and diffu-GRPO (Zhao et al., 2025a) both produce incorrect answers to the evaluation question. LLaDA's response includes a brief but partially meaningful reasoning step toward the end, whereas diffu-GRPO continues generating verbose sentences that contribute little to the final prediction. In contrast, our model provides a structured reasoning process and successfully arrives at the correct answer. This highlights that optimizing solely for accuracy-based rewards may lead to sub-optimal outcomes, as such rewards overlook the quality and coherence of reasoning within the response. Additional sample comparisons are provided in Fig. 5 and Fig. 6 in Appendix B.

	COUNTDOWN			GSM8K			SUDOKU			MATH		
Model / Seq Len	**128**	**256**	**512**	**128**	**256**	**512**	**128**	**256**	**512**	**128**	**256**	**512**
LLaDA (Nie et al., 2025)	20.7	19.5	16.0	68.7	76.7	78.2	11.7	6.7	5.5	26.0	32.4	36.2
diffu-GRPO (Zhao et al., 2025a)	33.2	31.3	37.1	72.6	79.8	81.9	18.4	12.9	11.0	**33.2**	37.2	**39.2**
TSE (Wang et al., 2025c)	25.0	23.4	16.4	70.1	78.7	78.9	×	×	×	28.4	35.6	36.2
WINO (Hong et al., 2025)	-	33.2	-	-	75.8	-	-	15.2	-	-	34.2	-
diffu-GRPO+PRM	×	×	×	71.7	80.9	81.5	×	×	×	30.8	36.0	36.0
Ours	**51.6**	**52.0**	**56.3**	**72.9**	**82.2**	**82.4**	**22.4**	**20.3**	**16.1**	32.0	**40.0**	38.4

Table 1: Performance comparison on COUNTDOWN, GSM8K, SUDOKU, and MATH at different sequence lengths. "–" denotes unreported results; "×" denotes unsupported tasks.

Model/Seq Len	COUNTDOWN			sec/it
	128	256	512	
diffu-GRPO	33.2	31.3	37.1	3.19
diffu-GRPO+PRM	-	-	-	7.58
Ours-NoUpweight	41.0	41.4	50.4	3.42
Ours-Cover	55.1	59.4	58.2	6.23
Ours-Random	55.4	54.7	59.8	4.76
Ours	51.6	52.0	56.3	3.42

Table 2: Ablation of design choices and efficiency.

intervals.

Despite PRMs being effective for test-time selection, we observe several significant challenges when using them as training-time rewards for dLLM policy optimization (Figure 4). First, *memory and runtime overhead*: unlike our approach, which reuses the training model to compute rewards, the pretrained PRM introduces substantial GPU and CPU memory consumption, resulting in significantly slower training (7.62 sec/iteration versus 3.42 sec/iteration for our method; Table 2). Second, *training instability*: generated responses frequently cause the PRM to output NaN values, likely due to its reliance on strictly formatted inputs such as explicit step delimiters. Replacing NaN scores with zeros further injects noise into the learning signal. Third, *reward hacking*: although the PRM reward increases steadily during training, task performance does not improve, indicating that the policy learns to exploit weaknesses in the PRM scoring function rather than developing better reasoning.

These issues are reflected in downstream performance. As shown in Table 1, diffu-GRPO+PRM achieves scores of 71.7, 80.9, and 81.5 on GSM8K at sequence lengths 128, 256, and 512, respectively,

Model	Training	SVAMP	ARC
LLaDA	-	83.3	90.2
diffu-GRPO	GSM8K	83.0	89.8
Ours	GSM8K	84.0	90.2
diffu-GRPO	MATH	83.7	91.8
Ours	MATH	85.7	**93.0**
diffu-GRPO	COUNTDOWN	84.0	90.6
Ours	COUNTDOWN	84.0	87.5
diffu-GRPO	SUDOKU	85.0	91.0
Ours	SUDOKU	**86.7**	90.6

Table 3: Generalization ability comparison. The models are evaluated on unseen datasets: the reasoning benchmark SVAMP (Patel et al., 2021) and the commonsense benchmark ARC (Clark et al., 2018).

Step	COUNTDOWN		GSM8K	
	diffu-GRPO	Ours	diffu-GRPO	Ours
1	**1.56**	1.17	12.81	**16.53**
8	**2.73**	1.56	9.48	**16.91**
16	**3.12**	2.34	13.04	**19.33**
24	**4.69**	1.95	17.21	**21.61**
32	6.64	**27.34**	24.26	**30.86**
40	12.50	**33.98**	39.27	**41.17**
48	19.53	**37.11**	49.96	**50.57**
64	33.2	**51.6**	72.6	**72.9**

Table 4: Accuracy of intermediate answers with sequence length 128 and 64 diffusion steps. Intermediate answers are obtained by decoding normally up to a target step and then decoding all remaining tokens in one pass.

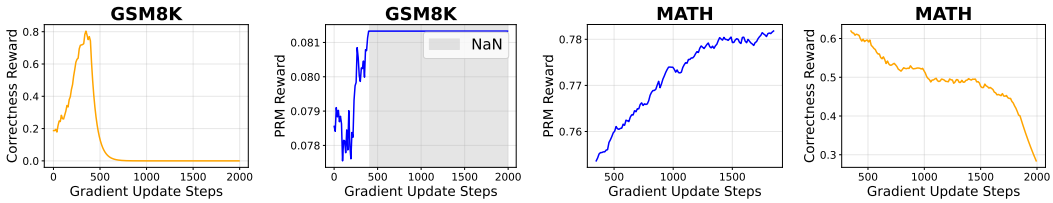

Figure 4: Two failure modes arise when using a pretrained PRM as the training-time reward: instability and potential reward hacking. Unlike our rule-based reward, the PRM must process the entire model-generated response through a large pretrained network. As a result, it frequently encounters unseen or irregular response formats, which can lead to numerical instabilities and NaN outputs. Moreover, as illustrated for the MATH dataset on the right, although the PRM score increases steadily during training, the task accuracy decreases. This divergence indicates that the model learns to exploit weaknesses in the PRM scoring function rather than improving its underlying reasoning quality.

compared to 72.6, 79.8, and 81.9 for diffu-GRPO. In contrast, our method further improves performance to 72.9, 82.2, and 82.4. These results indicate that, even relative to a strong pretrained PRM, our rule-based denoising process reward enables more stable optimization and better downstream performance, supporting our design choices.

Ablation study on model design. We analyze the impact of key design choices in DPR through a set of ablations. Ours-NoUpweight removes the up-weighting strategy and applies the process reward uniformly to all samples. Ours-Cover computes the process reward over all timestep intervals, corresponding to the exact aggregation in Eq. 3. Ours-Random samples t_2 uniformly at random instead of fixing $t_2 = T$. Results on the COUNTDOWN dataset are reported in Table 2.

The results highlight the role of each component. Ours-NoUpweight already substantially outperforms diffu-GRPO, indicating that the proposed denoising process reward is effective even without the up-weighting mechanism. Ours-Cover achieves the highest overall accuracy, but incurs significantly higher training cost due to evaluating all timestep intervals. Ours-Random attains comparable accuracy, but remains slower than our method because it requires additional forward passes and cannot reuse final completions. Our full method (Ours) closely matches the performance of both Ours-Cover and Ours-Random, while maintaining training speed comparable to diffu-GRPO. For reference, diffu-GRPO+PRM is substantially slower due to the overhead introduced by the pretrained PRM.

The proposed reward is effectively learned and facilitates training. We visualize the training rewards of diffu-GRPO (Zhao et al., 2025a) and our model in Fig. 7. Our method consistently achieves higher total rewards—which combine both accuracy and format rewards—explaining the substantial performance gains observed on these datasets. We further present the reward training curves in Fig. 8. The upward trend indicates that the model learns to favor responses yielding correct answers while adhering to reasoning processes that support the final outcome.

Our model demonstrates strong generalization

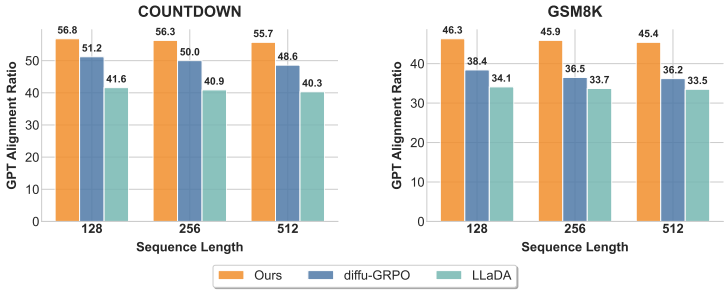

Figure 5: Model reasoning–outcome alignment ratio.

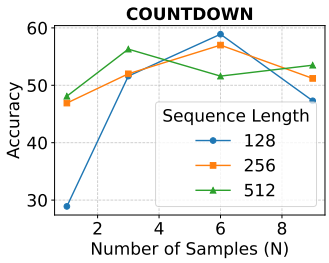

Figure 6: Ablation study.

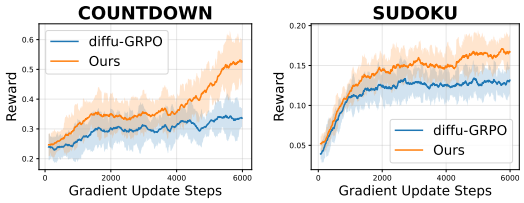

Figure 7: Reward curves during GRPO training.

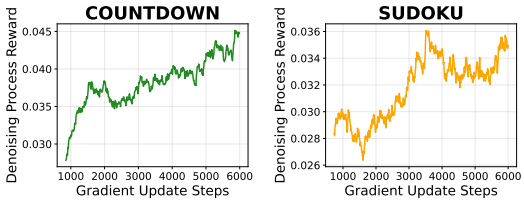

Figure 8: The training curve of our proposed reward.

ability. To thoroughly examine the proposed framework, we further evaluate our models on two unseen datasets: SVAMP (Patel et al., 2021) and ARC (Clark et al., 2018). The SVAMP dataset consists of numerous mathematical reasoning problems, while the ARC dataset focuses on commonsense reasoning tasks (e.g., "When oxygen combines with hydrogen, which substance is formed?"), where the model must select the correct answer from multiple choices. Notably, ARC is fundamentally different from our training datasets (e.g., GSM8K). Our model noticeably improves performance on both SVAMP and ARC.

Our method enables further acceleration through higher intermediate accuracy. Accelerating MDLLMs has been an active area of research (Li et al., 2025; Hong et al., 2025; He et al., 2025). Many approaches rely on the quality of intermediate responses: if these responses are accurate and contribute meaningfully to the final answer, generation can be accelerated. For instance, Prophet (Li et al., 2025) decides whether to decode all remaining tokens in a single step. Motivated by this, we analyze the accuracy of intermediate responses produced by our method. Specifically, during diffusion denoising, at each step, we additionally generate an answer by unmasking all remaining tokens at once. This gives us intermediate answers at every step, in addition to the final output obtained from fully decoding the masked sequence. We present quantitative results in Table 4, and provide additional qualitative examples in Fig. 7 in Appendix B. Across both datasets, our method achieves higher intermediate accuracy, suggesting that it may offer advantages over diffu-GRPO (Zhao et al., 2025a) when combined with MdLLM acceleration techniques.

Effect of the number of samples on the reward. Our DPR is based on an averaged estimation of the accuracy of generated responses. To assess its reliability, we perform an ablation study by varying the number of samples, $N \in \{1, 3, 6, 9\}$. As illustrated in Fig. 6, when $N = 1$, the estimation becomes noisy and leads to suboptimal performance. In contrast, when $N \geq 3$, we observe substantial improvements over both baseline methods, LLaDA (Nie et al., 2025) and diffu-GRPO (Zhao et al., 2025a), across sequence lengths of 128, 256, and 512. These results highlight the robustness of our proposed reward under different parameters.

5 Conclusion

We address the challenge of training diffusion language models for complex reasoning, where the absence of process-level supervision often results in poorly structured use of denoising steps. We show that supervision can be derived directly from the denoising trajectory itself, without external reward models or human annotation. By encouraging denoising steps to contribute meaningfully to the final prediction, our approach leads to more coherent reasoning behavior and more reliable generation. Experiments demonstrate consistent improvements on challenging reasoning benchmarks, stronger alignment between intermediate reasoning and final answers, and improved generation quality.

Limitations

Our method relies on the mean-field assumption adopted in diffu-GRPO (Zhao et al., 2025a) to estimate the log-likelihood of generated responses, which treats token predictions as conditionally independent and therefore neglects token-level dependencies. This approximation is a known limitation of current reinforcement learning approaches for diffusion language models. Unlike autoregressive models, where the chain rule provides a natural and exact factorization of the sequence likelihood, masked diffusion language models do not offer a convenient decomposition that would allow likelihoods to be computed precisely over partially specified intermediate states. As a result, removing the mean-field assumption would require fundamentally new inference or training mechanisms that can model joint token dependencies during the denoising process.

From a practical standpoint, this approximation is also tightly coupled with computational efficiency. More expressive likelihood estimators that capture token-level interactions would significantly increase inference cost and memory consumption, undermining the scalability of reinforcement learning for large diffusion language models. Consequently, our approach, like existing work, must rely on additional approximations to remain tractable.

Developing methods that relax the mean-field assumption while preserving efficient and stable training is an important direction for future work. Potential avenues include structured variational approximations, alternative diffusion parameterizations with tractable likelihoods, or hybrid decoding schemes that combine diffusion-based generation with autoregressive components to better capture token dependencies.

Ethical considerations

Although DPR reduces the need for costly human-labeled process rewards, it does not eliminate the risk of biases inherited from the underlying training data or evaluation benchmarks. The intrinsic rewards derived from the model's denoising dynamics may reinforce spurious correlations or reasoning patterns that perform well on benchmarks but are misaligned with human values or real-world reasoning norms, and care should therefore be taken when deploying DPR-trained models in high-stakes domains such as legal, medical, or policy decision-making. Moreover, DPR is intended as a general optimization framework and does not inherently encode ethical constraints or safety objectives; future work could explore integrating explicit safety-aware or value-aligned objectives into the step-aware reward formulation, as well as evaluating the method across a broader range of tasks and populations to better understand its societal impacts.

LLM usages.

Large Language Models (LLMs) were used solely for polishing the writing and improving the clarity of presentation. All ideas, analyses, results, and conclusions are original contributions of the authors.

References

Black-Phoenix. 2024. 4x4-sudoku-dataset: 1 million 4×4 sudoku boards. https://github.com/Black-Phoenix/4x4-Sudoku-Dataset.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, and 1 others. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, and 1 others. 2024. Scaling diffusion language models via adaptation from autoregressive models. *arXiv preprint arXiv:2410.17891*.

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. 2025. Diffucoder: Understanding and improving masked diffusion models for code generation. *arXiv preprint arXiv:2506.20639*.

Haoyu He, Katrin Renz, Yong Cao, and Andreas Geiger. 2025. Mdpo: Overcoming the training-inference divide of masked diffusion language models. *arXiv preprint arXiv:2508.13148*.

Feng Hong, Geng Yu, Yushi Ye, Haicheng Huang, Huangjie Zheng, Ya Zhang, Yanfeng Wang, and Jiangchao Yao. 2025. Wide-in, narrow-out: Revokable decoding for efficient and effective dllms. *arXiv preprint arXiv:2507.18578*.

Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Yi Liang, Soroush Vosoughi, and Shiwei Liu. 2025. Diffusion language models know the answer before decoding. *arXiv preprint arXiv:2508.19982*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Xiaoran Liu, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. 2025a. Longllada: Unlocking long context capabilities in diffusion llms. *arXiv preprint arXiv:2506.14429*.

Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyuan Wei, Shaobo Wang, and Linfeng Zhang. 2025b. dllm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, and 1 others. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. dkv-cache: The cache for diffusion language models. *arXiv preprint arXiv:2505.15781*.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. Large language diffusion models. *arXiv preprint arXiv:2502.09992*.

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. 2025. Tinyzero. https://github.com/Jiayi-Pan/TinyZero. Accessed: 2025-01-24.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Yuerong Song, Xiaoran Liu, Ruixiao Li, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. 2025. Sparse-dllm: Accelerating diffusion llms with dynamic cache eviction. *arXiv preprint arXiv:2508.02558*.

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. 2025. wd1: Weighted policy optimization for reasoning in diffusion language models. *arXiv preprint arXiv:2507.08838*.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025a. Remasking discrete diffusion models with inference-time scaling. *arXiv preprint arXiv:2503.00307*.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2023. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*.

Weiyun Wang, Zhangwei Gao, Lianjie Chen, Zhe Chen, Jinguo Zhu, Xiangyu Zhao, Yangzhou Liu, Yue Cao, Shenglong Ye, Xizhou Zhu, and 1 others. 2025b. Visualprm: An effective process reward model for multimodal reasoning. *arXiv preprint arXiv:2503.10291*.

Wen Wang, Bozhen Fang, Chenchen Jing, Yongliang Shen, Yangyi Shen, Qiuyu Wang, Hao Ouyang, Hao Chen, and Chunhua Shen. 2025c. Time is a feature: Exploiting temporal dynamics in diffusion language models. *arXiv preprint arXiv:2508.09138*.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. 2025d. Revolutionizing reinforcement learning framework for diffusion large language models. *arXiv preprint arXiv:2509.06949*.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. 2025. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. *arXiv preprint arXiv:2505.22618*.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*.

Fei Yu, Anningzhe Gao, and Benyou Wang. 2023. Ovm, outcome-supervised value models for planning in mathematical reasoning. *arXiv preprint arXiv:2311.09724*.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. 2024. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772.

Hanning Zhang, Pengcheng Wang, Shizhe Diao, Yong Lin, Rui Pan, Hanze Dong, Dylan Zhang, Pavlo Molchanov, and Tong Zhang. 2024b. Entropy-regularized process reward model. *arXiv preprint arXiv:2412.11006*.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. 2025a. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *arXiv preprint arXiv:2504.12216*.

Siyan Zhao, Mengchen Liu, Jing Huang, Miao Liu, Chenyu Wang, Bo Liu, Yuandong Tian, Guan Pang, Sean Bell, Aditya Grover, and 1 others. 2025b. Inpainting-guided policy optimization for diffusion large language models. *arXiv preprint arXiv:2509.10396*.

Appendix

A Implementation

The training code is provided in the supplementary material.

We build our model on top of the diffu-GRPO codebase (Zhao et al., 2025a) and apply GRPO to LLaDA-8B-Instruct (Nie et al., 2025). Following diffu-GRPO, we generate 6 rollouts per problem with a temperature of 0.9 and perform 12 update iterations per step (for SUDOKU, we use a temperature of 0.3 with 8 iterations). The model is trained using LoRA with rank 128 in 4-bit precision and evaluated in float16 precision. The learning rate is set to 3×10^{-6} with 600 warm-up steps. It takes around 72 hours on 8 modern GPU for each dataset. During evaluation, we use zero-shot prompting and greedy decoding with generation lengths of 128, 256, and 512 tokens, consistent with diffu-GRPO. For fair comparison, following the evaluation protocol of diffu-GRPO (Zhao et al., 2025a), we evaluate model checkpoints every 100 training steps and report the best-performing result.

To compute the denoising process reward during training, we sample a single intermediate timestep along the denoising trajectory of each rollout. At this timestep, we extract the partially denoised sequence containing the currently generated text tokens. We then append 64 additional mask tokens to this partial sequence and feed the extended input back into the model, which continues generation through iterative denoising. Empirically, we find that using 64 mask tokens provides a good trade-off between computational efficiency and performance across benchmarks.

After generation, we compute two quantities: (i) the accuracy of the original rollouts generated from the fully masked initial state, which reflects the model's baseline performance on the input question, and (ii) the accuracy of the completions generated from the intermediate denoising state. The denoising process reward is defined as the difference between these two accuracies, capturing how intermediate denoising progress improves the probability of producing a correct final answer.

B Additional Examples

We provide additional qualitative comparisons in this section. In Fig. 5 and Fig. 6, we present sample comparisons on the COUNTDOWN and SUDOKU datasets with mask lengths 128 and 512, respectively. We observe that our method consistently produces more coherent and higher-quality reasoning than **diffu-GRPO**. In contrast, **diffu-GRPO** often generates meaningless or irrelevant reasoning and then abruptly jumps to a final answer. As a consequence, its final predictions can be fragile and highly sensitive to small errors.

For example, in the fourth example of Fig. 5, the reasoning produced by **diffu-GRPO** consists of unrelated words that do not contribute to solving the problem. The model then suddenly outputs an answer; however, it produces the incorrect result $92 - 17 - 17$, which is numerically close to the correct expression $92 - 17 - 77$ but still fails. Such failures highlight the necessity of encouraging prediction-making that meaningfully utilizes intermediate reasoning steps during the generation process.

Furthermore, we present GSM8K examples with intermediate decoding at step 32 in Fig. 7. Notably, even though all mask tokens are decoded at an early stage (step 32, compared to 64 steps required for full decoding), our method already produces a substantial amount of correct, coherent, and well-structured reasoning. In contrast, **diffu-GRPO** often exhibits incomplete or flawed reasoning at this early stage, which leads to incorrect or unstable final predictions.

As a result, our method achieves higher prediction accuracy than **diffu-GRPO** even under early decoding. This behavior suggests that our approach is more robust to reduced decoding steps and can better leverage partial generation. Consequently, our method is more suitable than **diffu-GRPO** for integration with existing diffusion-based LLM acceleration techniques, where early stopping or partial decoding is critical for improving efficiency.

C License of Used Code and Datasets

The GSM8K dataset (Cobbe et al., 2021) and the MATH dataset (Lightman et al., 2023) are released under the MIT License. We obtain the COUNTDOWN and SUDOKU datasets from the repository of diffu-GRPO (Zhao et al., 2025a). The COUNTDOWN dataset is released under the Apache License 2.0, and the codebase of diffu-GRPO is also licensed under Apache License 2.0. These licenses permit use for academic research purposes.

Our use of the GSM8K, MATH, COUNTDOWN, and SUDOKU datasets, as well as the diffu-GRPO codebase, is consistent with their intended use for research, benchmarking, and method development in machine learning.

Example 1: Numbers: [10, 36, 8] Target: 38		
Model	**diffu-GRPO**	**Ours**
Generation	\nTo reach the target of 38 using the numbers 10, 36, and 8, we need to find a combination of operations that, when applied to these numbers, result in 38. We can start by considering the operations of addition, subtraction, multiplication, and division. We can try different combinations of numbers and operations to see if we can reach the target.\n </reasoning>\n <answer>\n \\boxed{36 - 10 + 8}\n </answer><|eot_id|>|>	\nTo create an arithmetic expression that evaluates to exactly the target number 38 using the numbers 10, 36, and 8, we need to consider the operations +, -, *, and /. Let's try different combinations:\n \n 1. 36 + 8 - 10 = 34\n 2. 36 - 8 + 10 = 38\n \nThe combination that works is 36 - 8 + 10. \n \n </reasoning>\n \n \\boxed{36 - 8 + 10}<|eot_id|><|endoftext|>
Evaluation	Answer: ✗ Reasoning: ✗	Answer: ✓ Reasoning: ✓

Example 2: Numbers: [12, 71, 32] Target: 91		
Model	**diffu-GRPO**	**Ours**
Generation	\nTo reach the target of 91 using the numbers 12, 71, and 32, we need to find the correct combination of operations. Let's start by considering the largest number, 71, and see if we can combine it with the other numbers to get 91. We can try adding 71 and 12, which gives us 83. Then, subtracting 32 from 83 gives us 91.\n </reasoning>\n <answer>\n \\boxed{71+12-32}\n </answer><|eot_id|><|endoftext|>	\nTo find an arithmetic expression that evaluates to exactly 91 using the numbers 12, 71, 32, we need to consider the operations +, -, *, and /. Let's try different combinations:\n \n 1. 71 + 32 - 12 = 91\n \n .\n \nThis , we with the expression 71 + 32 - 12, which evaluates to exactly 91.\n </reasoning>\n <answer>\n \\boxed{71 + 32 - 12}\n </answer><|eot_id|><|endoftext|>
Evaluation	Answer: ✗ Reasoning: ✗	Answer: ✓ Reasoning: ✓

Example 3: Numbers: [14, 69, 24] Target: 31		
Model	**diffu-GRPO**	**Ours**
Generation	\nTo find an expression that evaluates to 31 using the numbers 14, 69, and 24, we can to start with the largest number and subtract to get the closest number. We can use 69 - 14 = 55, which is close to 31. Then, we can subtract 24 from 55 to get 31. Therefore, the expression is 69 - 14 - 24.\n </reasoning>\n <answer>\n \\boxed{69-14-24\n </answer><|eot_id|><|endoftext|> }	\nTo create an arithmetic expression that evaluates to exactly the target number 31 using the numbers 14, 69, and 24, we need to consider the operations +, -, *, and /. Let's try different combinations:\n \n 1. 69 - 24 - 14 = 31\n \nThis combination uses each number exactly once and evaluates to the target number 31. Therefore, the expression is meets.\n </reasoning>\n \n <answer>\n \\boxed{69 - 24 - 14}\n </answer><|eot_id|><|endoftext|>
Evaluation	Answer: ✗ Reasoning: ✓	Answer: ✓ Reasoning: ✓

Example 4: Numbers: [15, 78, 62] Target: 1		
Model	**diffu-GRPO**	**Ours**
Generation	\nTo achieve the target of 1 using the numbers 15, 78, and 62, we need to find a combination of operations that, when applied to these numbers, result in 1. We can start by considering the properties of division and multiplication. One way forward is to use the division operation to reduce the numbers to a smaller that can when multiplied by another number result in 1. We can also consider the use of to simplify the numbers further.\n </reasoning>\n <answer>\n \\boxed{15 * 78 / 62}\n </answer><|eot_id|><|endoftext|>	\nTo create an arithmetic expression that evaluates to exactly the target number 1 using the numbers 15, 78, and 62, we need to consider the operations +, -, *, and /. Let's try different combinations:\n \n 1. 78 - 62 = 16\n 2. 16 - 15 = 1\n \nSo , the expression 78 - 62 - 15 works.\n </reasoning>\n <answer>\n \\boxed{78 - 62 - 15}\n </answer><|eot_id|><|endoftext|>
Evaluation	Answer: ✗ Reasoning: ✗	Answer: ✓ Reasoning: ✓

Example 5: Numbers: [17, 92, 77] Target: 32		
Model	**diffu-GRPO**	**Ours**
Generation	\nTo reach the target number 32 using the numbers 17, 92, and 77, we need to find a combination of operations operations that, when applied to these numbers, result in 32. We can start by considering the operations addition and subtraction, as they are straightforward to use. We can try to combine the numbers in a way that allows us to reach the target.\n </reasoning>\n <answer>\n \\boxed{92 - 17 - 17}\n </answer><|eot_id|>	\nTo find an arithmetic expression that evaluates to exactly 32 using the numbers 17, 92, and 77, we need to consider the operations +, -, *, and /. Let's try different combinations:\n \n 1. 92 - 77 + 17 = 32\n \nThis combination works because the expression (92 - 77) + 17 evaluates to 32. \n \n </reasoning>\n \n <answer>\n \\boxed{(92 - 77) + 17}\n </answer><|eot_id|><|endoftext|>
Evaluation	Answer: ✗ Reasoning: ✗	Answer: ✓ Reasoning: ✓

Example 6: Numbers: [20, 28, 19] Target: 67		
Model	**diffu-GRPO**	**Ours**
Generation	\nTo reach the target of 67 using the numbers 20, 28, and 19, we need to find the correct combination of operations. Let's start by considering the largest number, 28, and see if we can use it to get the remaining numbers. We can add 20 to 28 to get 48, and then add 19 to 48 to get 67. This combination uses each number exactly once.\n </reasoning>\n <answer>\n \\boxed{28+2+19}\n </answer><|eot_id|><|endoftext|>	\nTo find an arithmetic expression that evaluates to exactly 67 using the numbers 20, 28, 19, we need to consider the operations +, -, *, and /. Let's try different combinations:\n \n 1. 20 + 28 + 19 = 67\n \n .\n \n 2, 20 + 28 + 19 = 67\n \n ,\n \nEach ,\n \n \n </reasoning>\n <answer>\n \\boxed{20 + 28 + 19}\n </answer><|eot_id|><|endoftext|>
Evaluation	Answer: ✗ Reasoning: ✓	Answer: ✓ Reasoning: ✓

Table 5: Examples on the COUNTDOWN dataset with 128 mask tokens where **diffu-GRPO** prematurely jumps to a final expression without valid intermediate reasoning, or exhibits incorrect reasoning that violates the task constraints, while **Ours** consistently produces correct reasoning processes and valid final answers.

Example 1: Solve the following Sudoku puzzle: 2000104040233010		
Model	**diffu-GRPO**	**Ours**
Generation	\nTo solve the 4x4 Sudoku puzzle, we need to fill in the grid with digits 1-4 such that each row, column, and 2x2 box contains all the digits exactly once. Let's proceed step-by-step:\n \n 1. **Identify the constraints**: The puzzle is a 4x4 grid with digits 1, 2, 3, and 4. The constraints require each row, column, and 2x2 box to contain all the digits exactly once.\n \n 2. **Analyze the initial grid**: The initial grid is \"2000104040233010\". We need to fill in the empty cells while maintaining the constraints.\n \n 3. **Fill in the grid**:\n - Start with the first row: \"2000104040233010\".\n - The the first row is missing the digits 1, 2, 3, and 4.\n - The the first column is missing the digits 1, 2, 3, and 4.\n - The first 2x2 box is missing the digits 1, 2, 3, and 4.\n \n 4. **Complete the grid**:\n - The first row should be filled in as: \"1234\".\n - The first column should be filled in as: \"1234\".\n - The first 2x2 box should be filled in as: \"1234\".\n \n 5. **Verify the solution**:\n - The first row: \"1234\" contains the digits 1, 2, 3, and 4 exactly once.\n - The first column: \"1234\" contains the digits 1, 2, 3, and 4 exactly once.\n - The first 2x2 box: \"1234\" contains the digits 1, 2, 3, and 4 exactly once.\n \n 6. **Construct the final grid**:\n - The final grid is: \"21234104040233010\".\n \n </reasoning>\n \n <answer>\n 21234104040233233010\n </answer><|eot_id|><|endoftext|>	\nTo solve the 4x4 Sudoku puzzle, we need to fill in empty cells with digits 1-4 such that each row, column, and 2x2 box contains all digits exactly once. We can start by analyzing the given digits and using them to in the empty cells while maintaining the Sudoku rules.\n \n 1. **Identify the given digits**: The given digits are 2, 0, 0, 1, 0, 4, 0, 2, 3, 3, 0, 1, 0.\n \n 2. **Fill the grid**: We need to fill the digits 1-4 in the empty cells such that each row, column, and 2x2 box contains all digits exactly once.\n \n 3. **Check the constraints**: After filling the grid, we need to verify that each row, column, and 2x2 box contains all digits 1-4 exactly once.\n \nLet 's start filling the grid:\n \n :\n 2 0 0 0\n 1 0 4 0\n 4 0 2 3\n 3 0 1 0\n :\n \nWe can start by filling the first row and column:\n \n :\n 2 0 0 0\n 1 0 4 0\n 4 0 2 3\n 3 0 1 0\n :\n \nNext , we can fill the second row and column:\n \n :\n 2 0 0 0\n 1 0 4 0\n 4 0 2 3\n 3 0 1 0\n :\n \nFinally , we can fill the third row and column:\n \n :\n 2 0 0 0\n 1 0 4 0\n 4 0 2 3\n 3 0 1 0\n :\n \nNow , we can fill the fourth row and column:\n \n :\n 2 0 0 0\n 1 0 4 0\n 4 0 2 3\n 3 0 1 4\n :\n \nAfter filling the grid, we verify that each row, column, and 2x2 box contains all digits 1-4 exactly once.\n \n </reasoning>\n \n <answer>\n 2000104040233014\n </answer><|eot_id|><|endoftext|>
Evaluation	Correct cells: 2 / 16	Correct cells: 9 / 16

Example 2: Solve the following Sudoku puzzle: 0040341203010004		
Model	**diffu-GRPO**	**Ours**
Generation	\nTo solve the 4x4 Sudoku puzzle, we need to fill in the grid with digits 1-4 such that each row, column, and 2x2 box contains all the digits exactly once. Let's proceed step-by-step:\n \n 1. **Identify the constraints**: The puzzle is a 4x4 grid with digits 1-4. Each row, column, and 2x2 box must contain all digits exactly once.\n \n 2. **Analyze the given grid**: The initial grid is \"0040341203010004\".\n \n 3. **Fill the grid row by row**:\n - The first row: 0, 0, 4, 0, 3, 4, 1, 2, 0, 3, 0, 1, 0, 0, 4\n - The second row: 0, 0, 4, 0, 3, 4, 1, 2, 0, 3, 0, 1, 0, 0, 4\n - The third row: 0, 0, 4, 0, 3, 4, 1, 2, 0, 3, 0, 1, 0, 0, 4\n - The fourth row: 0, 0, 4, 0, 3, 4, 1, 2, 0, 3, 0, 1, 0, 0, 4\n \n 4. **Fill the grid column by column**:\n - The first column: 0, 0, 4, 0, 3, 4, 1, 2, 0, 3, 0, 1, 0, 0, 4\n - The second column: 0, 0, 4, 0, 3, 4, 1, 2, 0, 3, 0, 1, 0, 0, 4\n \n - The third column: 0, 0, 4, 0, 3, 4, 1, 2, 0, 3, 0, 1, 0, 0, 4\n \n <|eot_id|><|endoftext|>	\nTo solve the 4x4 Sudoku puzzle, we need to fill in empty cells with digits 1-4 such that each row, column, and 2x2 box contains all digits exactly once. We can start by examining the given puzzle and filling in the cells step by step.\n \n 1. The first row is \"004034\". We can to fill in the empty cells with digits 1-4 such that each row, column, and 2x2 box contains all digits exactly once.\n \n 2. The second row is \"12030\". We can to fill in the empty cells with digits 1-4 such that each row, column, and 2x2 box contains all digits exactly once.\n \n 3. The third row is \"10004\". We can to fill in the empty cells with digits 1-4 such that each row, column, and 2x2 box contains all digits exactly once.\n \n 4. We first fill in the first row with digits 1-4 such that each row, column, and 2x2 box contains all digits exactly once.\n \n 5. We then fill in the second row with digits 1-4 such that each row, column, and 2x2 box contains all digits exactly once.\n \n 6. We then fill in the third row with digits 1-4 such that each row, column, and 2x2 box contains all digits exactly once.\n \n 7. We then fill in the first column with digits 1-4 such that each row, column, and 2x2 box contains all digits exactly once.\n \n 8. We then fill in the second column with digits 1-4 such that each row, column, and 2x2 box contains all digits exactly once.\n \n 9. We then fill in the third column with digits 1-4 such that each row, column, and 2x2 box contains all digits exactly once.\n \n 10. We then fill in the fourth column with digits 1-4 such that each row, column, and 2x2 box contains all digits exactly once.\n \nBy following these steps, we can solve the given 4x4 Sudoku puzzle.\n </reasoning>\n \n <answer>\n 1234411332212234\n </answer><|eot_id|><|endoftext|>
Evaluation	Correct cells: 0 / 16	Correct cells: 8 / 16

Table 6: Examples on the SUDOKU dataset with 512 mask tokens where **Ours** produces solutions closer to the ground truth than **diffu-GRPO**.

Problem: Carver just turned 45 years old, which makes him 5 years less than twice the age of his son. How old is his son, in years?		
Model	**diffu-GRPO**	**Ours**
Generation	"\nLet \\(x \\) be the age of Carver's son. According to the problem, Carver is 45 years old, which is 5 years less than twice the age of his son. This can be expressed as:\n \\[45 = 2x - 5 \\]\nSolving for \\(x \\):\n \\[55 = 2x = 2 \\ \\]]\\[\\[=2 \\ \\]\n \n „ver's son is 25 \\]\n </</ing>\n «>\n \\boxed{{5}}\n </answer<|eot_id|><|endoftext|>	\nLet \\(x \\) be the age of Carver's son. According to the problem, Carver is 45 years old, which is 5 years less than twice the age of his son. This can be expressed as:\n \\[45 = 2x - 5 \\]\nSolving for \\(x \\):\n \\[45 + 5 = 2x \\]\n \\[50 = 2x \\]\n \n „'s son 2 years \\]\n </reasoning>\n <answer>\n \\boxed{25}\n </answer><|eot_id|><|endoftext|>
Evaluation	Answer: ✗ Reasoning: ✗	Answer: ✓ Reasoning: ✓

Problem: The employees of Google went on a day trip. 4 buses were rented that have the capacity of holding 60 employees. 6 minibusses that can hold 30 employees, and 10 minivans that can hold 15 employees. How many employees can join the day trip?		
Model	**diffu-GRPO**	**Ours**
Generation	The total of the 4 buses is 4 * 60 = 220 employees. \n The total of the 6 minibusses is 6 * 30 = 180 employees. \n The total of the 10 minivans is 10 * 15 = 150 employees. \n The total the of of join day day 220 + 18 + 1550 =5550.\n </</ing>\n <answer>\n \\boxed{{50}\n </</></><|eot_id|><|endoftext|>}	\nThe capacity of the 4 buses is \\(4 \\times 60 = 240\\) employees. The capacity of the 6 minibusses is \\(6 \\times 30 = 180\\) employees. The capacity of the 10 minivans is \\(10 \\times 15 = 150\\) employees. Adding these together, the of employees \\(220 + 18 + + 10 = 577 employees employees\n \n </reasoning>\n <answer>\n \\boxed{570}\n </answer><|eot_id|><|endoftext|> \par
Evaluation	Answer: ✗ Reasoning: ✗	Answer: ✓ Reasoning: ✓

Problem: Jana has 27 puppies. Two thirds of Jana's puppies are Pomeranians. One third of the Pomeranians are girls. How many girl Pomeranians does Jana have?		
Model	**diffu-GRPO**	**Ours**
Generation	\n First, we need to determine how many Pomeranians Jana has. Since two thirds of her puppies are Pomeranians, we calculate:\n \\[\n \\frac{2}{3} \\times 27 = 18\n \\]\n Next, we find to how many of these are Pomeranians are girls.\n \n \\[\\[\n \n \\[\\frac11{}{}{}66666666\\]\n </</ing>\n «>\n \\boxed{66}\n </><|eot_id|><|endoftext|>	\nJana has 27 puppies in total two thirds of them are Pomeranians, so she has 27 * 2/3 = 18 Pomeranians.\nOne third of the Pomeranians are girls, so she has 18 * 1/3 = 6 girl Pomeranians.\n </,ing>\n <»6boxed\n 6»<|eot_id|>
Evaluation	Answer: ✗ Reasoning: ✗	Answer: ✗ Reasoning: ✓

Problem: Lani baked 55 cookies. She ate 5 five cookies and placed the rest equally into five jars. How many cookies were in each jar?		
Model	**diffu-GRPO**	**Ours**
Generation	\nLani baked 555 cookies. She ate 55 cookies, so she has 555 - 55 = 500 cookies left. She placed the remaining cookies equally into five jars, so each jar has 500 / 5 = 100 cookies.\n </</ing>\n <>\n 100\n </></><|endoftext|>	\nLani baked 55 cookies and ate 5 cookies. So, she has 55 - 5 = 50 cookies left. She then placed these 50 cookies equally into 5 jars. Therefore, each jar contains 50 / 5 = 10 cookies.\n </reasoning>\n <answer>\n \\boxed{10}\n </answer><|eot_id|><|endoftext|><|endoftext|>
Evaluation	Answer: ✗ Reasoning: ✗	Answer: ✓ Reasoning: ✓

Problem: James has 6 more candies than Robert. John has twice as many candies as Robert. If John has 54 candies, how many more candies does John have than James?		
Model	**diffu-GRPO**	**Ours**
Generation	\nIf John has 5 many candies as Robert, then Robert has 54 / 2 = 27 candies.\nJames has 6 more candies than Robert, so James has 27 + 6 = 33 candies.\nTo find out how many more candies John has than James, we subtract the number of candies James has from the number of candies candies candies::55 - -33 = = 11.\n \nTherefore „ has{211 more than candies.}	\nIf John has 54 candies, and he has twice as many candies as Robert, then Robert has 54 / 2 = 27 candies.\nIf James has 6 more candies than Robert, then James has 27 + 6 = 33 candies.\nTherefore , John has 54 - 33 = 21 more candies than than.\n </</»>\n \n \n 111\n \n \n \n <|endoftext|>
Evaluation	Answer: ✗ Reasoning: ✗	Answer: ✗ Reasoning: ✓

Table 7: Intermediate responses generated by decoding all mask tokens at step 32 on GSM8K with 128 mask tokens, comparing **diffu-GRPO** and **Ours**. With our proposed reward, **Ours** produces higher-quality reasoning at early decoding steps, which translates into higher prediction accuracy.