Reinforcement Learning from Denoising Feedback

Anonymous ACL submission

Abstract

Policy loss estimation remains a fundamental and long-standing challenge in reinforcement learning (RL) for diffusion language models (DLMs). We introduce **Reinforcement Learning from Denoising Feedback (RLDF)**, a novel training paradigm that leverages feedback obtained from rollout and training processes to facilitate accurate and efficient policy loss estimation. To balance the trade-off between computational efficiency and estimation effectiveness, RLDF optimizes the model toward the clipped clean state from intermediate noisy states, combined with weighted timestep sampling over denoising timesteps. Extensive experiments demonstrate that RLDF achieves consistent and substantial improvements in both performance and generalizability across two representative dLLM architectures, LLaDA and Dream, on multiple reasoning benchmarks. Our work lays a principled foundation for scalable reinforcement learning in diffusion language models.

1 Introduction

Recent breakthroughs in diffusion language models (DLMs) (Austin et al., 2021a; Sahoo et al., 2024), such as Dream (Ye et al., 2025) and LLaDA (Nie et al., 2025), have positioned them as a promising pathway toward highly capable foundation models (Google DeepMind, 2025; Song et al., 2025; Khanna et al., 2025). The appeal of DLMs (Gulrajani and Hashimoto, 2023; Yu et al., 2025b) stems from several intrinsic advantages over autoregressive (AR) architectures: bidirectional attention (Gong et al., 2025), arbitrary decoding order, and the potential for multi-token parallel generation (Wu et al., 2025b,a). Nevertheless, DLMs currently underperform their autoregressive counterparts across many benchmarks, a gap largely attributable to immature training methodologies and pipelines. Among the capabilities most in need of improvement, reasoning (OpenAI, 2024; Anthropic, 2025; Guo et al., 2025; Team et al., 2025) stands out as particularly critical. Reasoning of language models is a frontier that has been extensively explored and substantially advanced within the autoregressive paradigm.

Reinforcement learning (RL) (Kaelbling et al., 1996) has emerged as an effective and highly generalizable approach for cultivating reasoning capabilities in large language models (LLMs). Evolving from PPO (Schulman et al., 2017) through DPO (Rafailov et al., 2023), GRPO (Shao et al., 2024), and subsequent variants (Yu et al., 2025a; Yue et al., 2025), RL has consistently improved LLM reasoning with verifiable reward (Wang et al., 2024; Uesato et al., 2022) across diverse scenarios, including tool use (Jin et al., 2025), mathematics (Snell et al., 2024), and code generation (Le et al., 2022; Shojaee et al., 2023). However, these RL methods cannot be directly transplanted to diffusion language models due to two fundamental discrepancies: the attention mechanism (causal vs. bidirectional) and the generative process (next-token prediction vs. full-sequence denoising).

A central obstacle in adapting RL to DLMs lies in **policy loss estimation**. In autoregressive models, the policy loss is obtained through a single forward pass under causal attention. In contrast, the arbitrary decoding order and bidirectional attention of DLMs render exact log-likelihood computation intractable. Prior approaches (Zhao et al., 2025; Ou et al., 2025) approximate the policy loss via either no-masking or random-masking schemes, both of which introduce a substantial mismatch between training and inference. Alternatively, computing the policy loss at every denoising step (Ni et al., 2026) preserves train–inference alignment but incurs prohibitive computational and time costs due to the iterative nature of the denoising process. **This exposes a core challenge: how to balance the fidelity of policy loss estimation against its**

computational cost.

To address this challenge, we propose **Reinforcement Learning from Denoising Feedback (RLDF)**, a training paradigm that leverages two complementary feedback sources. **Rollout feedback** captures unmasking probabilities and their corresponding timesteps along the denoising trajectory. **Training feedback**, derived from predicted probabilities during denoising training, provides signals from which we retain only the most informative positions for stable and effective policy updates. Building on these signals, we employ a weighted sampling strategy (Rubinstein and Kroese, 2016) that preferentially selects denoising steps with high predictive uncertainty, exerting the greatest influence on the final generation (Choi et al., 2022). The policy loss is computed exclusively over these selected steps, yielding a favorable precision–efficiency trade-off while remaining compatible with diverse masking strategies and readily extensible to emerging training schemes.

Empirical results demonstrate that our method exhibits substantial improvements on math and code tasks across both LLaDA and Dream models, as shown in Table 2. Furthermore, our approach achieves faster convergence rates with more precise policy gradient estimation. We also demonstrate the advantage of leveraging the clean state rather than the next state for loss estimation, which avoids loss spikes and improves the utilization of training signals. In summary, our contributions can be summarized as follows:

- We open-source an extensible and modular RL framework purpose-built for DLMs, to facilitate community research and novel algorithms.
- We propose an improved policy loss estimation method that achieves a better trade-off between efficiency and precision.
- We unlock the reasoning potential of DLMs (LLaDA and Dream), providing empirical insights for DLM community.

2 Related Work

2.1 Reasoning in Diffusion Language Models

Reasoning has long been a central topic in language model research. Diffusion language models have recently demonstrated promising potential across a range of reasoning tasks. The models show superiority in planning tasks (Ye et al., 2024a; Huang et al., 2025) and demonstrate better data learning efficiency (Ni et al., 2025; Prabhudesai et al., 2025; Gao et al., 2025). Complementary lines of work have explored augmenting reasoning capabilities through established techniques such as chain-of-thought prompting (Ye et al., 2024b), leveraging both supervised fine-tuning and reinforcement learning (Ho et al., 2020; Austin et al., 2021a) to elicit more structured and reliable inference. Further efforts have focused on accelerating the reasoning process (Xie et al., 2025), while others have investigated the design of remasking strategies to improve generation quality (Cheng et al., 2026).

2.2 Reinforcement Learning

Reinforcement learning (RL) (Kaelbling et al., 1996; Schulman et al., 2017), especially GRPO (Shao et al., 2024), has proven effective in enhancing the reasoning capabilities of large language models (LLMs) (He et al., 2025; Qian et al., 2025). However, owing to fundamental differences in attention mechanisms and forward computation processes, diffusion language models require distinct strategies for computing RL losses. Existing loss computation methods have undergone successive refinements, evolving from Diffu-GRPO (Zhao et al., 2025; Tang et al., 2025) through Coupled-GRPO (Xie et al., 2025) to ELBO (Ou et al., 2025), etc. Each of these methods attempts to better align the training objective with the denoising nature of diffusion models. Nevertheless, all of these methods suffer from training-inference mismatch to varying degrees, which undermines the reliability of policy gradient estimation and ultimately limits reasoning performance. We are motivated by this limitation and propose a novel loss computation method that considers both accuracy and efficiency, serving as a stronger foundation for reinforcement learning on diffusion language models.

3 Reinforcement Learning from Denoising Feedback

3.1 Denoising Feedback of Diffusion Models

The denoising process of diffusion models contains rich information that can substantially benefit reinforcement learning. We extract feedback from two complementary stages: the **rollout process** and the **training process**. During rollout, we collect reasoning trajectories together with their timestep probabilities, which capture the reasoning context and guide the timestep selection. During training,

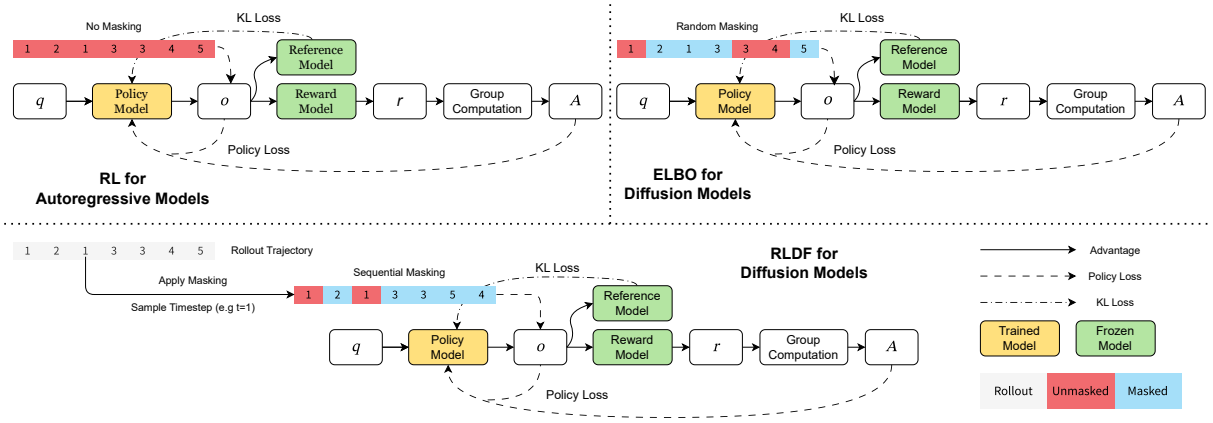

Figure 1: Comparison of training paradigms. **Left:** RL for autoregressive models without masking. **Right:** ELBO-based training for diffusion models with random masking. **Bottom:** Our proposed RLDF, which samples a timestep t and applies sequential masks. Policy loss is estimated based on $\pi_\theta(x_0 \mid x_t)$. This formulation provides a more precise policy loss estimation while using identical trajectory context and objective.

we predict the clean state x_0 from the noisy state x_t and derive the learning signal from the processed $\hat{x}_0$ through token-level clipping. Both sources of feedback are effective in stabilizing training, yet have been largely overlooked in prior paradigms.

3.2 Loss Estimation at Timestep t

To faithfully replicate the context in which each position is unmasked, we adopt a stepwise loss estimation scheme that enables precise gradient estimation without approximation. Formally, given a prompt q and the sequence of tokens decoded across denoising timesteps $o_T, \ldots, o_0$, at each timestep t we compute the forward process $p(o_0 \mid q, o_t)$ and evaluate the loss exclusively on the positions unmasked at timestep t. This procedure exactly mirrors the model's rollout without introducing any approximation error.

However, this exact formulation incurs substantial computational overhead. Specifically, the number of forward passes scales linearly with the number of decoding steps n, as the loss must be computed iteratively for each step. In practice, a standard reasoning trajectory requires approximately 256 decoding steps, resulting in 256 forward passes per sample and rendering naive stepwise training prohibitively expensive in both time and memory.

3.3 Weighted Sampling for Timestep Selection

To mitigate the computational bottleneck, we propose selectively sampling a subset of denoising steps for loss computation rather than processing all steps exhaustively, adopting weighted sampling as the principled mechanism for timestep selection.

Model	Task	Dataset	Pearson r	Spearman ρ
LLaDA	Math	MATH	-0.942	-0.994
	Code	MBPP	-0.950	-0.933
Dream	Math	MATH	-0.930	-0.994
	Code	MBPP	-0.928	-0.997

Table 1: Pearson (r) and Spearman (ρ) correlations between each token's unmasking probability and entropy of the model's predicted distribution at the corresponding position, evaluated on LLaDA and Dream across the MATH and MBPP benchmarks.

Prior work has shown that entropy serves as an effective guidance signal during training (Wang et al., 2025a; Cheng et al., 2026). As shown in Table 1, distribution entropy and token probability at each position exhibit near-perfect correlation across both models and tasks, indicating that decoding probability is a faithful proxy for entropy. We therefore adopt per-step decoding probability as the weight for timestep selection, which not only provides a more intuitive perceptual signal but also avoids explicit entropy computation.

As illustrated in Figure 2, the distribution of per-token decoding probabilities is heavily skewed: the vast majority of tokens are decoded with high confidence (typically above 0.9), while only a small fraction are assigned low probability. Crucially, these low-confidence tokens exert disproportionate influence on the diversity of reasoning trajectories and exhibit strong correlation with final answer accuracy. As shown in Figure 3, such tokens are often connective words in math and instruction verbs in code tasks. We therefore preferentially sample steps with lower average token confidence.

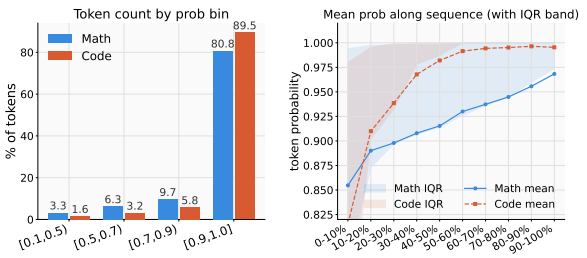

Figure 2: **Token probability distributions across Math and Code tasks.** *(left)* Distribution of tokens by probability bin: over 80% of tokens are unmasked with high confidence ($p \geq 0.9$) in both tasks. *(right)* Mean token probability along denoising timesteps with IQR band: tokens unmasked at later timesteps tend to have higher confidence and lower variance.

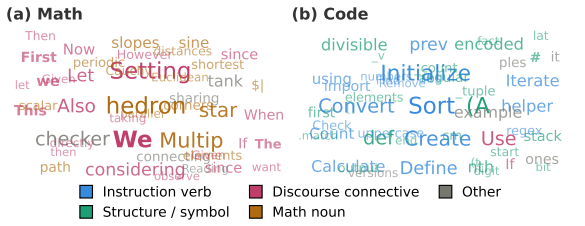

Figure 3: **Word clouds of high-entropy tokens during decoding on Math (left) and Code (right) tasks.** Font size encodes mean entropy H; font weight encodes frequency (bold: $\geq$50, medium: 10–49, regular: $<$10). Colors denote semantic category.

For diffusion language models, this sampling strategy offers a two-fold benefit: it substantially reduces the number of loss computation iterations, while simultaneously concentrating gradient signal on the most informative denoising steps, yielding a lower-variance and more rollout-faithful training objective.

3.4 Loss Estimation of RLDF

The RLDF loss estimation pipeline, illustrated in Figure 1, follows a structured sequence: rollout generation, sequential masking schedule construction, iterative loss estimation on each selected timestep, and a final aggregated backpropagation.

Sampling timesteps with likelihood-based weighting. Rather than selecting timesteps uniformly, we weight each timestep by the average token-level uncertainty of the model. Concretely, for each timestep t we compute

$$u_t = -\frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \log p_i, \tag{1}$$

where $\mathcal{I}_t$ denotes the set of token positions unmasked at timestep t, formally satisfying $\mathcal{I}_t \triangleq \{i : o_{t,i} \neq o_{t-1,i}\}$. p_i is the model's predicted probability for token i, and u_t is the resulting uncertainty estimate for timestep t. Timesteps with lower model confidence (smaller p_i) thus receive larger weight. Here, $o_{t,i}$ denotes the token at position i of the response at timestep t. A temperature-scaled softmax converts these weights into a sampling distribution:

$$w_t = \frac{\exp(u_t/\tau)}{\sum_{t'} \exp(u_{t'}/\tau)}, \tag{2}$$

and k steps are drawn without replacement from w. As $\tau \to 0$, this degenerates to deterministic top-k selection, while $\tau \to \infty$ recovers uniform sampling, allowing a smooth interpolation between exploitation and exploration.

Iterative loss estimation on each timestep. The loss at timestep t of response b is computed by Eq. (3), which reduces to Eq. (4) when only a single iteration is performed. Here, $\hat{A}^{(b)}$ denotes the normalized advantage of response b, and $\hat{o}_0$ denotes its clipped version, which retains only tokens with predicted probability above a threshold. This clipping prevents low-confidence tokens from dominating the training signal. The per-step loss is normalized by the number of retained tokens, aggregated over the selected steps $S_t^{(b)}$, and finally averaged across all responses in the group to form the final objective.

When more than one forward-pass iteration is performed per rollout, the behavior policy $\pi_{\theta_{\text{old}}}$ used to generate samples diverges from the current policy π_θ being optimized. To correct for this off-policy mismatch and stabilize training, we adopt a weighted sampling ratio together with PPO-style clipping:

$$\ell_{\text{policy}}^{(b)}(t) = -\frac{1}{|\hat{o}_0|} \sum_{i \in \hat{o}_0} \min\Big(r_{t,i}^{(b)}(\theta)\,\hat{A}^{(b)}, \; \text{clip}\big(r_{t,i}^{(b)}(\theta),\, 1\pm\varepsilon\big)\hat{A}^{(b)}\Big), \tag{3}$$

where the per-token importance ratio is

$$r_{t,i}(\theta) = \frac{\pi_\theta(\hat{o}_{0,i} \mid q,\, o_t)}{\pi_{\theta_{\text{old}}}(\hat{o}_{0,i} \mid q,\, o_t)}.$$

When only one iteration is performed, $\pi_\theta = \pi_{\theta_{\text{old}}}$, so $r_t^{(i)}(\theta) \equiv 1$ and the clipping is inactive; Eq. (3) then reduces to the REINFORCE form (Williams, 1992):

$$\ell_{\text{policy}}^{(b)}(t) = -\frac{1}{|\hat{o}_0|} \sum_{i \in \hat{o}_0} \log \pi_\theta(\hat{o}_{0,i}^{(b)} \mid q,\, o_t^{(b)})\,\hat{A}^{(b)}. \tag{4}$$

Regularization and aggregation. To further constrain the policy from drifting too far from the reference model π_{ref} and to ensure training stability, we additionally incorporate the unbiased K_3 KL-divergence estimator (Schulman, 2020) between π_θ and π_{ref}:

$$\ell_{\text{KL}}^{(b)}(t) = \mathbb{D}_{\text{KL}}[\pi_\theta(\hat{o}_0^{(b)} \mid q, o_t^{(b)}) \,\|\, \pi_{\text{ref}}(\hat{o}_0^{(b)} \mid q, o_t^{(b)})].$$

After computing Policy and KL loss for every selected step, we aggregate across $S_t^{(b)}$ and then across all B responses in the group to obtain the final loss objective:

$$\mathcal{L}_{\text{RLDF}} = \frac{1}{G}\sum_{b=1}^{G}\frac{1}{|S_t^{(b)}|}\sum_{t\in S_t^{(b)}}\left[\ell_{\text{policy}}^{(b)}(t) + \beta\ell_{KL}^{(b)}\right], \quad (5)$$

where β controls the KL regularization strength. We additionally apply gradient-norm clipping during backpropagation to suppress occasional large updates and stabilize training. The complete training objective is given in Eq. (16).

4 Experiments

4.1 Experiment Setup

Data and Models. Mathematical reasoning and code generation are widely adopted benchmarks for evaluating the reasoning capabilities of large language models. Accordingly, we conduct training and evaluation on both domains. For mathematical reasoning, we use the official training split of MATH (Hendrycks et al., 2021) for training, and evaluate on MATH500 (Lightman et al., 2023), GSM8K (Cobbe et al., 2021), and AMC23 (Art of Problem Solving, 2023). For code generation, we train on the training split of MBPP and evaluate on MBPP (Austin et al., 2021b) and HumanEval (Chen et al., 2021). We adopt two mainstream diffusion language models as our backbones: LLaDA-8B-Instruct(Nie et al., 2025) and Dream-7B-Instruct (Ye et al., 2025).

Training. Our training procedure consists solely of reinforcement learning, initialized from the pretrained diffusion backbones. For policy loss estimation, we apply weighted sampling over 16 denoising timesteps. To prevent excessive deviation from the reference model, we adopt the widely used k_3 estimator for the KL regularization term. During rollout, we follow Wang et al. (2025b) and employ block-wise decoding with KV-cache to accelerate generation. Specifically, we adopt a generation length of 256 and a dynamic remasking threshold of 0.9. Full hyperparameter and implementation details are provided in Appendix D.

Evaluation. To highlight the generalization ability of each RL method, we adopt a strict cross-setting evaluation protocol: each model is trained on a single dataset with a fixed generation length of 256, and then evaluated across all remaining settings. Following prior work, we benchmark all models at generation lengths of 128, 256, and 512 under the dynamic and static sampling strategy. We compare our method against several well-established baselines, including d1 (Zhao et al., 2025), ESPO (Ou et al., 2025), TraceRL (Wang et al., 2025b), and Coupled-GRPO (Gong et al., 2025). In addition, for the LLaDA series, we report the performance of LLaDA-1.5 (Zhu et al., 2025) as an additional reference point.

4.2 Main Results

Strong Performance Across Tasks and Models. Our reinforcement learning from denoising feedback (RLDF) method achieves consistently dominant performance across both LLaDA and Dream models on mathematical reasoning and code generation benchmarks, as reported in Table 2. Nearly all configurations of our approach rank first or second among all baselines, with first-place results in the majority of settings. Notably, the performance gains are consistent across both model families, in contrast to d1 and ESPO, which demonstrate competitive results only on individual models. Relative to the respective base models, our approach yields substantial improvements of up to 10 accuracy points on Dream for MATH500 and MBPP, and approximately 8 points for LLaDA on the same benchmarks. These empirical results provide compelling evidence for the effectiveness of the proposed method.

Unlocking the Reasoning Potential of the Dream Model. A particularly striking finding is that the Dream model converges remarkably rapidly during reinforcement learning, requiring only approximately 100 training steps on both math and code tasks. We hypothesize that this unexpectedly fast emergence of reasoning stems from latent capabilities inherited from the underlying Qwen architecture (Yang et al., 2025), which may have been suppressed during the adaptation with random masking training from autoregressive to masked diffusion modeling. Our reinforcement learning procedure

Model	MATH500			GSM8K			AMC23			HumanEval			MBPP		
	128	256	512	128	256	512	128	256	512	128	256	512	128	256	512
LLaDA Models															
LLaDA-8B-Instruct	32.0	36.4	40.8	75.7	83.4	83.0	7.5	12.5	12.5	39.0	37.2	35.9	37.4	36.6	38.0
LLaDA-1.5	32.2	40.0	39.8	76.1	82.0	84.0	15.0	12.5	7.5	40.8	35.9	38.4	36.2	38.6	38.6
d1	32.6	36.8	38.2	75.9	82.5	82.5	12.5	**17.5**	12.5	38.4	36.5	37.8	38.0	39.2	40.8
ESPO	37.8	38.0	40.6	79.1	83.4	84.2	12.5	15.0	**22.5**	41.4	37.2	**40.2**	**42.2**	43.8	42.6
TraceRL	31.8	33.4	33.6	77.7	80.7	81.5	7.5	5.0	20.0	38.4	35.3	35.9	36.8	39.0	40.2
Coupled-GRPO	33.2	41.2	37.2	77.8	83.5	83.0	15.0	10.0	20.0	37.8	40.9	39.6	38.6	42.4	42.0
RLDF (ours)	**39.6**	**43.8**	**45.0**	**79.2**	**84.7**	**84.9**	**15.0**	**17.5**	20.0	**42.8**	**40.9**	38.5	40.4	**44.6**	**44.8**
Δ vs. Base	+7.6	+7.4	+4.2	+3.5	+1.3	+1.9	+7.5	+5.0	+7.5	+3.8	+3.7	+2.6	+3.0	+8.0	+6.8
Dream Models															
Dream-7B-Instruct	36.8	38.0	40.2	66.9	75.2	74.4	15.0	17.5	17.5	51.2	53.6	55.4	51.8	52.2	52.8
d1	**41.2**	46.0	46.8	73.3	83.4	83.4	**20.0**	15.0	15.0	54.2	53.6	54.2	53.8	54.0	53.6
ESPO	31.0	42.4	43.0	65.8	81.5	78.9	15.0	15.0	17.5	54.2	56.1	55.4	53.2	53.0	52.8
TraceRL	35.2	47.8	47.6	65.9	83.6	84.6	10.0	15.0	17.5	53.6	56.7	54.8	55.2	57.0	55.4
Coupled-GRPO	**41.2**	45.6	45.6	69.7	82.3	82.5	10.0	20.0	25.0	54.9	54.9	57.3	54.2	54.2	54.0
RLDF (ours)	39.8	**48.6**	**49.6**	**74.3**	**86.5**	**86.5**	17.5	**30.0**	**30.0**	**62.8**	**61.0**	**59.2**	**61.2**	**60.4**	**60.6**
Δ vs. Base	+3.0	+10.6	+9.4	+7.4	+11.3	+12.1	+2.5	+12.5	+12.5	+11.6	+7.4	+3.8	+9.4	+8.2	+7.8

Table 2: Performance of different RL methods on LLaDA and Dream models across math and code benchmarks under the **dynamic** unmasking strategy with varying generation steps (128, 256, and 512 steps) with unmasking threshold at 0.9. The best result in each column is highlighted in **bold**, and Δ vs. Base denotes the absolute improvement of RLDF (ours) over the corresponding base model.

effectively reactivates these dormant reasoning capacities, recovering and amplifying the model's problem-solving abilities. We further observe that training on Dream exhibits greater instability compared to LLaDA, manifesting as rapidly escalating KL divergence; we address this by applying a higher KL penalty coefficient ($\beta = 0.04$). This instability may reflect a fundamental discontinuity introduced by the autoregressive-to-diffusion adaptation, pointing to a potential limitation of such model conversion approaches.

Cross-dataset generalization. Our method generalizes consistently across benchmarks, whereas competing methods often improve on one distribution at the cost of others. Specifically, it yields stable gains on out-of-distribution benchmarks including GSM8K, AMC23, and HumanEval. This is particularly notable for the shift from MBPP to HumanEval: MBPP describes function implementations in free-form natural language, while HumanEval requires completing partially defined function stubs under explicit constraints. The strong cross-dataset performance suggests that our models acquire transferable reasoning skills rather than overfitting to surface patterns of the specific training dataset.

Robustness to unmasking strategy. Our main experiments use a dynamic unmasking threshold (set to 0.9). To further test robustness, we evaluate each model under a static unmasking strategy across varying generation lengths and step counts, constraining the model to unmask one token per step and reporting results at 128, 256, and 512 steps in Table 4. The static results largely mirror the trends observed under dynamic unmasking, and our method remains dominant particularly on Dream models. These results indicate that the choice of unmasking strategy is a relatively minor factor compared to the number of denoising steps and the generation length.

5 Ablation Studies and Design Choices

5.1 Effect of Sampling Numbers and Temperature

We investigate the effect of varying the number of sampling timesteps in Figure 4. As expected, increasing the number of sampling timesteps accelerates convergence, analogous to the effect of increasing batch size. Since each additional sampling timestep introduces an extra forward and backward pass, a larger number of timesteps directly increases per-iteration training time. In our implementation, using 16 sampling steps incurs a

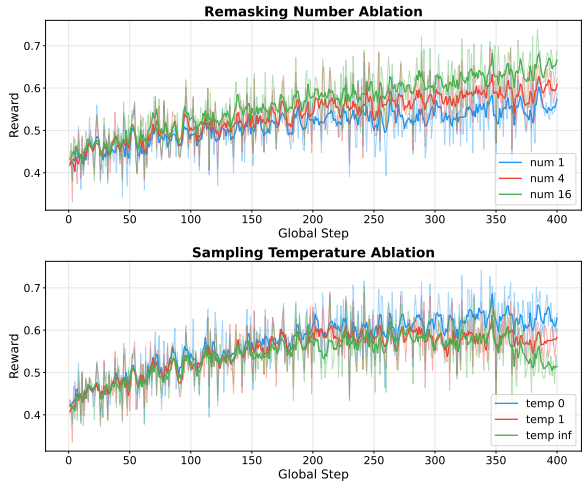

Figure 4: **Ablation on sequential remasking strategy.** Mean rollout reward under varying remasking counts and timestep sampling temperatures. Larger remasking counts and lower sampling temperatures consistently lead to higher rollout rewards.

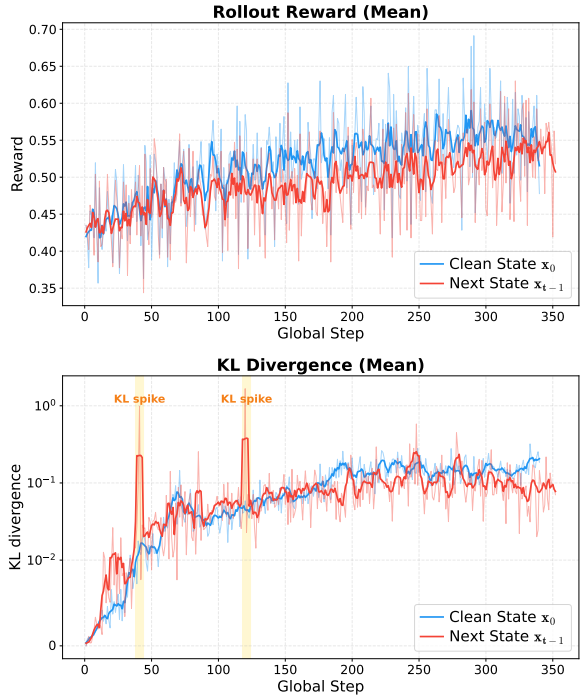

Figure 5: **Clean state estimation yields more stable training than next state estimation.** Training dynamics on LLaDA code tasks under two target choices ($\mathbf{x}_0$ vs. $\mathbf{x}_{t-1}$), measured by rollout reward and KL loss.

computation cost comparable to that of the rollout process alone.

We further observe that the sampling temperature notably affects training stability: lower temperatures yield more stable dynamics. Since lower temperatures bias sampling toward low-entropy timesteps, we attribute this to loss estimation behavior. High-entropy timesteps contain few but informative tokens, making loss estimation inherently unstable due to limited sample size. Low-entropy timesteps, in contrast, provide far more tokens for stable averaging, though probability-based filtering is needed to isolate informative signal from low-value tokens.

This reveals a signal-noise trade-off: sufficient signal is required for stable optimization, yet noise from low-entropy timesteps must not dominate the loss estimation. Consequently, choosing an appropriate probability filtering threshold is crucial for training stability.

5.2 Why $\mathbf{x}_0$ Estimation Outperforms $\mathbf{x}_{t-1}$ Estimation

Theoretically, the one-step posterior can be recovered by marginalizing over the clean signal (Austin et al., 2021a):

$$p_\theta(x_{t-1} \mid x_t) = \sum_{x_0} p_\theta(x_0 \mid x_t)\, q(x_{t-1} \mid x_t, x_0),$$

so any x_{t-1}-prediction model implicitly commits to a particular path through x_0 space, while x_0-prediction retains the full distribution over clean sequences. Predicting x_0 thus exposes the model to a global denoising objective at every timestep, yielding a richer signal and a target distribution whose statistics remain stable across noise levels.

This gap is amplified in practice. Under dynamic unmasking, tokens are revealed only when their probability exceeds a threshold (with a single token unmasked otherwise), so the supervisory signal for x_{t-1}-estimation is severely limited, especially when high-entropy timesteps are preferentially sampled. As shown empirically in Table 3, x_0-estimation leverages $\sim$15% of response tokens per sample, while x_{t-1}-estimation uses only $\sim$2%, discarding most of the available signal.

In Figure 5, $\mathbf{x}_{t-1}$-based estimation also exhibits pronounced KL loss spikes during training, attributable to the instability introduced by computing KL loss over such a limited token set; the policy loss suffers from the same issue. In contrast, $\mathbf{x}_0$-based estimation yields higher rollout rewards, empirically confirming the superiority of clean state estimation for stable and effective training.

5.3 Loss Normalization

Token-level loss normalization has been proposed to mitigate the response-length imbalance com-

monly observed in RL training of autoregressive models. In this section, we investigate its applicability and effectiveness within our setting.

Sample-level normalization first normalizes within each sample and then aggregates within the group, ensuring that every sample contributes equally to the batch. In contrast, token-level normalization operates directly over all tokens in the group, giving every token an equal contribution.

Formally, in contrast to the sample-level loss defined in Eq. (16), the token-level loss is formulated as in Eq. (17). As shown in Figure 6, sample-level loss normalization, even when combined with gradient norm clipping, still slightly underperforms its token-level counterpart. Although token-level normalization introduces a deviation from the standard loss objective, it empirically yields a more stable training process by mitigating spikes in KL divergence.

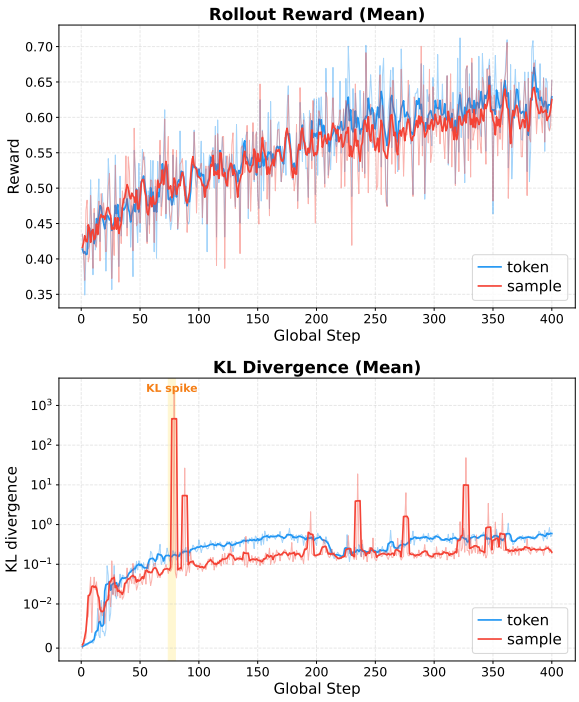

Figure 6: **Comparison of token-level and sample-level loss normalization (both with gradient clipping).** Token-level normalization leads to a more stable training process, with fewer loss spikes and faster convergence.

5.4 Token Clipping is Necessary

At each training iteration, the model performs a forward pass conditioned on the partially observed context at timestep t. Due to the limited context available, the model struggles to make reliable predictions for positions at timesteps $t' \ll t$, often assigning extremely low probabilities to those tokens. Under the loss formulation in Equation (4), these near-zero probabilities can dominate the loss computation, overshadowing the gradient signal from high-confidence tokens — precisely those from which we seek to learn. This imbalance inflates the noise-to-signal ratio and distorts the training dynamics.

Setting		Training Metrics		
Target	**Clip**	**Grad Norm**	**Loss**	**Token Utility**
x_0	w/	8.14	0.0727	13.39%
x_0	w/o	3,232.21	0.6308	62.15%
x_t	N/A	8.46	0.0591	2.07%

Table 3: Comparison of training dynamics under different target states and gradient clipping settings. The first two columns specify the experimental **setting** (target state and whether gradient clipping is applied), while the last three columns report the corresponding **training metrics** averaged over the first 100 steps. Without token clipping, both gradient norm and loss diverge by several orders of magnitude.

To increase the proportion of informative gradient signal in the policy loss, we apply token-level clipping that discards clean-state tokens whose predicted probability falls below a threshold (0.2 in our implementation). The benefit of this design is validated in Table 3: targeting x_0 with token clipping achieves a favorable trade-off, yielding a stable gradient norm while substantially improving token utility over the x_t baseline.

6 Conclusion

In this work, we address the fundamental challenge of log-likelihood estimation in the policy gradient objective of diffusion language models. We propose **RLDF** (Reinforcement Learning from Denoising Feedback), a principled method that achieves a favorable trade-off between estimation precision and computational efficiency by approximating the policy loss on clipped final states $\hat{x}_0$ across a subset of selected timesteps t. Extensive empirical evaluations demonstrate both the effectiveness and the necessity of our approach in enabling stable reinforcement learning for DLMs. Furthermore, we introduce a comprehensive and extensible **RL training framework for DLMs**, encompassing flexible masking strategies, pluggable RLVR reward functions, and diverse training methods, with compatibility for two prominent DLM architectures, LLaDA and Dream.

Limitations

Our experiments were carried out mainly on 8× NVIDIA A100 80GB GPUs using bf16 mixed precision, which may limit reproducibility for researchers without access to comparable computational infrastructure. We did not verify whether our results hold under different hardware (e.g., B200, H100, A6000) or lower-precision regimes (fp16, fp8, or int8/int4 quantization), both of which are increasingly relevant for practical deployment. Numerical differences across precisions could in principle affect training stability and final performance; we leave a systematic study of hardware- and precision-robustness to future work.

References

Anthropic. 2025. Claude 3.7 sonnet system card. Technical report, Anthropic.

Art of Problem Solving. 2023. AMC 2023 problems. https://artofproblemsolving.com/wiki/index.php/AMC_Problems_and_Solutions. Accessed: 2026-05-19.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021a. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021b. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Xin Zhao, Zhenliang Zhang, and Furu Wei. 2026. Reasoning with exploration: An entropy perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 30377–30385.

Jooyoung Choi, Jungbeom Lee, Chaehun Shin, Sungwon Kim, Hyunwoo Kim, and Sungroh Yoon. 2022. Perception prioritized training of diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11472–11481.

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*.

Zitian Gao, Haoming Luo, Lynx Chen, Jason Klein Liu, Ran Tao, Joey Zhou, and Bryan Dai. 2025. What makes diffusion language models super data learners? *arXiv preprint arXiv:2510.04071*.

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*.

Google DeepMind. 2025. Gemini diffusion. https://deepmind.google/models/gemini-diffusion/. Accessed: 2026-04-27.

Ishaan Gulrajani and Tatsunori B Hashimoto. 2023. Likelihood-based diffusion language models. *Advances in Neural Information Processing Systems*, 36:16693–16715.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Qi He, Cheng Qian, Xiusi Chen, Bingxiang He, Yi R Fung, and Heng Ji. 2025. Veri-r1: Toward precise and faithful claim verification via online reinforcement learning. *arXiv preprint arXiv:2510.01932*.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Zemin Huang, Zhiyang Chen, Zijun Wang, Tiancheng Li, and Guo-Jun Qi. 2025. Reinforcing the diffusion chain of lateral thought with diffusion language models. *arXiv preprint arXiv:2505.10446*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, and 1 others. 2025. Mercury: Ultra-fast language models based on diffusion. *arXiv e-prints*, pages arXiv–2506.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. 2022. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328.

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*.

Jinjie Ni, Qian Liu, Longxu Dou, Chao Du, Zili Wang, Hang Yan, Tianyu Pang, and Michael Qizhe Shieh. 2025. Diffusion language models are super data learners. *arXiv preprint arXiv:2511.03276*.

Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, and 1 others. 2026. The flexibility trap: Why arbitrary order limits reasoning potential in diffusion language models. *arXiv preprint arXiv:2601.15165*.

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*.

OpenAI. 2024. Learning to reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/.

Jingyang Ou, Jiaqi Han, Minkai Xu, Shaoxuan Xu, Jianwen Xie, Stefano Ermon, Yi Wu, and Chongxuan Li. 2025. Principled rl for diffusion llms emerges from a sequence-level perspective. *arXiv preprint arXiv:2512.03759*.

Mihir Prabhudesai, Mengning Wu, Amir Zadeh, Katerina Fragkiadaki, and Deepak Pathak. 2025. Diffusion beats autoregressive in data-constrained settings. *arXiv preprint arXiv:2507.15857*.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Reuven Y Rubinstein and Dirk P Kroese. 2016. *Simulation and the Monte Carlo method*. John Wiley & Sons.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T 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. 2020. Approximating KL divergence. http://joschu.net/blog/kl-approx.html. Blog post.

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*.

Parshin Shojaee, Aneesh Jain, Sindhu Tipirneni, and Chandan K Reddy. 2023. Execution-based code generation using deep reinforcement learning. *arXiv preprint arXiv:2301.13816*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, and 1 others. 2025. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*.

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*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

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*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, and 1 others. 2025a. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.

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

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. 2025a. Fast-dllm v2: Efficient block-diffusion llm. *arXiv preprint arXiv:2509.26328*.

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

Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, and 1 others. 2025. Dreamcoder 7b: An open diffusion language model for code. *arXiv preprint arXiv:2509.01142*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2024a. Beyond autoregression: Discrete diffusion for complex reasoning and planning. *arXiv preprint arXiv:2410.14157*.

Jiacheng Ye, Shansan Gong, Liheng Chen, Lin Zheng, Jiahui Gao, Han Shi, Chuan Wu, Xin Jiang, Zhenguo Li, Wei Bi, and 1 others. 2024b. Diffusion of thought: Chain-of-thought reasoning in diffusion language models. *Advances in Neural Information Processing Systems*, 37:105345–105374.

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*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025a. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Runpeng Yu, Qi Li, and Xinchao Wang. 2025b. Discrete diffusion in large language and multimodal models: A survey. *arXiv preprint arXiv:2506.13759*.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, and 1 others. 2025. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*.

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

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and 1 others. 2025. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*.

A Preliminaries

A.1 Forward and Denoising Process

The forward process independently masks each token $x_{0,i}$ with probability t or leaves it unchanged with probability $1-t$, yielding the partially masked sequence x_t:

$$q(x_t \mid x_0) = \prod_{i=1}^{L} \mathrm{Cat}\big(x_{t,i};\ (1-t)\,\mathbf{e}_{x_{0,i}} + t\,\mathbf{e}_{\texttt{[MASK]}}\big)\,. \quad (6)$$

During denoising, dLLMs support both static and dynamic strategies. In static decoding, the model unmasks the top-k most confident token positions at each step, where k is a fixed hyperparameter. In dynamic decoding, tokens are unmasked based solely on predicted logits, with positions whose confidence exceeds a predefined threshold being unmasked at each step.

To precisely describe the denoising process, we formalize the denoising process with the following definition. Given a prompt q, let $\{o_T, o_{T-1}, \ldots, o_0\}$ describe the denoising process, where T is the number of denoising steps at inference time. Each o_t represents the model's response state at denoising timestep t. Specifically, o_T corresponds to the fully masked initial state, o_0 denotes the final clean response, and intermediate states o_t $(0 < t < T)$ contain a mixture of decoded tokens and remaining mask tokens.

A.2 Training Techniques

Following the masked language modeling paradigm of BERT, MDMs are trained via random masking. Specifically, a parametric model $p_\theta(\cdot \mid x_t)$ takes the partially masked sequence x_t as input and simultaneously predicts all masked tokens (denoted as `M`). It is optimized via a cross-entropy loss computed exclusively over the masked positions:

$$\mathcal{L}(\theta) \triangleq -\mathbb{E}_{t,\,x_0,\,x_t}\left[\frac{1}{t}\sum_{i=1}^{L} \mathbb{1}[x_{t,i} = \mathbf{M}] \log p_\theta(x_{0,i} \mid x_t)\right]. \quad (7)$$

However, a fundamental discrepancy exists between the training and denoising processes. During training, masking is applied randomly and independently across token positions, whereas during denoising, the set of masked positions evolves deterministically across steps based on model confidence. This mismatch results in substantially different masking patterns and contextual inputs between the two phases, leading to inconsistent attention distributions and key-value representations. Such train-inference discrepancy poses a significant challenge for reasoning-oriented fine-tuning, as the model is optimized under a distribution of masked contexts that poorly reflects those encountered at inference time.

A.3 Challenges in Log Probability Estimation

Unlike autoregressive models, whose loss can be computed within a single forward pass, diffusion language models do not enjoy this benefit due to their bidirectional attention mechanism. Specifically, the key-value (KV) representations change across denoising steps, making the forward computation distinct at each step. Early methods such as d1 (Zhao et al., 2025) and wd1 (Tang et al., 2025) adopt a full-sequence loss strategy, which computes the average loss over all token positions conditioned on the prompt only in a single forward pass:

$$\mathcal{L}_{\text{full-seq}}(o \mid q) = \log p_\theta(o \mid q, o_T). \tag{8}$$

Subsequently, the Evidence Lower Bound (ELBO) was introduced, which approximates the log-likelihood via random masking over denoising timesteps:

$$\mathcal{L}_{\text{ELBO}}(o \mid q) = \mathbb{E}_{t,\,m}\left[\log p_\theta\big(o \mid q,\, o^m\big)\right], \tag{9}$$

where o^m denotes the response sequence randomly masked by $\mathcal{M}$. However, all of these methods compute the loss under a context that is inconsistent with the actual rollout conditions at inference time, giving rise to a fundamental training-inference mismatch that undermines the reliability of policy gradient estimation in RL training.

B Classic Variants of Policy Loss

We introduce three prevalent methods for policy loss estimation, derived from the no masking, random masking, and sequential masking perspectives, respectively.

For brevity, we define the per-token importance ratio as

$$r_{t,i}(\theta) \;=\; \frac{\phi_{\pi_\theta}(o_{0,i} \mid c)}{\phi_{\pi_{\theta_{\text{old}}}}(o_{0,i} \mid c)}, \tag{10}$$

where the conditioning context c varies across the three variants: $c = q$ for no masking, $c = (q,\, o^{\mathcal{M}_i})$ for random masking, and $c = (q,\, o_t)$ for sequential masking. For all three objectives, A_i^k denotes the token-level advantage, ε is the PPO clipping threshold, and β controls the strength of the KL regularization toward the reference policy π_{ref}. By convention, quantities without an explicit subscript t correspond to the final step $t = T$ (e.g., $r_i \triangleq r_{0,i}$, $o_i \triangleq o_{0,i}$).

All Masking. Diffu-GRPO (Zhao et al., 2025) computes the policy-gradient loss over *all* tokens of every sampled response. For each query q, a randomly masked variant q' is constructed and used as the conditioning context, providing a one-step likelihood estimation at every position $k \in \{1, \ldots, |o_i|\}$:

$$\mathcal{L}_{\text{diffu}}(\theta) = \mathbb{E}\Bigg[\frac{1}{G}\sum_{b=1}^{G}\frac{1}{|o^b|}\sum_{i=1}^{|o^b|}\min\Big(r_i^{(b)}A_i^{(b)}, \text{clip}(r_i^{(b)}, 1-\varepsilon, 1+\varepsilon)A_i^{(b)}\Big) - \beta\,\mathbb{D}_{\text{KL}}\big[\phi_{\pi_\theta} \,\|\, \phi_{\pi_{\text{ref}}}\big]\Bigg]. \tag{11}$$

The expectation is taken over $q \sim \mathcal{D}$, and $\{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$. Because the loss is averaged uniformly over $|o_i|$ tokens, every output position contributes equally regardless of whether it was actually denoised at that step.

Random Masking. Instead of scoring all tokens, random masking restricts the loss to a randomly sampled mask set $\mathcal{M}_b \subseteq \{1, \ldots, |o^{(b)}|\}$ for each response, while the unmasked tokens $o^{\overline{\mathcal{M}_b}}$ and prompt q serve as the context. This corresponds to a Monte-Carlo estimator of the masked-diffusion ELBO at the token level:

$$\mathcal{L}_{\text{ELBO}}(\theta) = \mathbb{E}\Bigg[\frac{1}{G}\sum_{b=1}^{G}\frac{1}{|\mathcal{M}_b|}\sum_{i\in\mathcal{M}_b}\min\Big(r_i^{(b)}A_i^{(b)}, \text{clip}(r_i^{(b)}, 1-\varepsilon, 1+\varepsilon)A_i^{(b)}\Big) - \beta\,\mathbb{D}_{\text{KL}}\big[\phi_{\pi_\theta} \,\|\, \phi_{\pi_{\text{ref}}}\big]\Bigg]. \tag{12}$$

The expectation is taken over $q \sim \mathcal{D}$, $\{o^{(b)}\}_{b=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$, and $\mathcal{M}_b \sim \text{masking}(o^{(b)})$. By conditioning on the visible tokens and supervising only the masked positions, ELBO yields a lower-variance, theoretically grounded surrogate that is consistent with the training objective of masked diffusion language models. ESPO (Ou et al., 2025) is modified from the ELBO estimation.

Sequential Masking. The sequential masking method further aligns the RL signal with the actual generation process by following the diffusion denoising trajectory $\tau = (o_T, o_{T-1}, \ldots, o_0)$, where o_T denotes the fully masked initial state and $o_0 = o$ is the final clean response. At each denoising timestep t, the policy unmasks a subset of positions $\mathcal{I}_t$ conditioned on the partially denoised state o_t, producing the next state o_{t-1}. The loss is computed at exactly those positions:

$$\mathcal{L}_\tau(\theta) = \mathbb{E}\Bigg[\frac{1}{G}\sum_{b=1}^{G}\frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{I}_t^{(b)}|}\sum_{i\in\mathcal{I}_t^{(b)}}\min\Big(r_{t,i}^{(b)}A_i^{(b)}, \text{clip}\,(r_{t,i}^{(b)}, 1-\varepsilon, 1+\varepsilon)A_{t,i}^{(b)}\Big) - \beta\,\mathbb{D}_{\text{KL}}\big[\phi_{\pi_\theta}\,\|\,\phi_{\pi_{\text{ref}}}\big]\Bigg]. \tag{13}$$

The expectation is taken over $q \sim \mathcal{D}$ and $o_{t-1} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q, o_t)$ for $t = T, T-1, \ldots, 1$ and $b = 1, \ldots, G$. Unlike the previous two variants, every token is supervised under the exact context in which it was generated, so the optimization landscape matches the inference-time denoising procedure step by step. This faithfulness comes at the cost of T-fold more forward passes per response. Our method adapts the sequential masking variant by predicting the clean state $\mathbf{x}_0$ instead of $\mathbf{x}_{t-1}$, retaining its trajectory-level precision at a fraction of the computational cost, thus striking a favorable balance between fidelity and efficiency.

C Reward Function

For **math tasks**, we extract the content of the last `\boxed{...}` from the model output and compare it against the ground truth via a math-equivalence check (combining string normalization, `sympy`-based symbolic comparison, and the `math_verify` library). The reward is binary:

$$r_{\text{math}} = \begin{cases} 1, & \text{if } \hat{a} \equiv a^* \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

where $\hat{a}$ is the extracted answer and a^* the ground truth.

For **code tasks**, we extract the Python code block enclosed by ```` ```python ``` ```` and execute it against the provided unit tests in a sandboxed runner. The reward is the test-case pass rate:

$$r_{\text{code}} = \frac{\#\,\text{passed test cases}}{\#\,\text{total test cases}} \in [0, 1], \tag{15}$$

yielding a dense signal that rewards partial correctness and provides smoother gradients than the binary math reward.

D Implementation Details

D.1 Data Processing

Out-of-memory (OOM) errors frequently occur during both rollout and training. To mitigate this, we restrict training data to subsets of the MATH and MBPP training splits with a maximum sequence length of 320 tokens for LLaDA and 256 tokens for Dream. Additionally, since binary rewards on MATH yield limited advantage diversity within each group, we retain only samples for which LLaDA produces at least one correct response across 16 generations, ensuring sufficient richness in the reward signal.

D.2 Remasking Strategy

We adopt a sequential remasking strategy with weighted timestep sampling. The sampling temperature defined in Equation 2 is set at one. To balance training and rollout efficiency, we set k=16, meaning 16 timesteps are sampled per training example.

D.3 Training

We use separate batch sizes for rollout and training. Prior to each training update, rollout responses are filtered to retain only samples exhibiting non-trivial policy gradients, i.e., those with varying reward values within a group. We apply a REINFORCE loss with a single gradient update per sample, rather than PPO. The KL penalty coefficient β is set to 0.01 for LLaDA and 0.04 for Dream. The learning rate is fixed at $1e^{-6}$ throughout training.

D.4 Rollout

A dynamic unmasking strategy is used for both rollout and evaluation. Tokens are unmasked greedily by highest probability when no token exceeds the confidence threshold of 0.9. To accelerate generation, we employ block-wise denoising with a block size of 32. LLaDA samples via the Gumbel-Max trick, while Dream uses explicit categorical sampling. The rollout batch size is set to 8, with 4 generations per sample.

E Additional Details

This appendix provides supplementary materials that complement the main text, including the full RLDF algorithm (Algorithms 1–3), the complete sample-level and token-level training objectives (Eq. 16 and Eq. 17), and additional experimental results under the static unmasking strategy (Table 4).

Algorithm 1 Reinforcement Learning from Denoising Feedback

Require: Policy π_θ, reference π_{ref}, dataset $\mathcal{D}$; group size G, inner iterations N, sampled steps k; clip range $(\epsilon_{\text{lo}}, \epsilon_{\text{hi}})$, KL weight β, learning rate η, std floor δ, sampling temperature τ

for each sample $(q, y) \sim \mathcal{D}$ **do**
  $\theta_{\text{old}} \leftarrow \theta$
  for $b = 1, \ldots, G$ **do**
    Sample response $o^{(b)} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$; record step probs u and denoising trajectories $\mathcal{T}$
  end for
  $r^{(b)} \leftarrow R(o^{(b)}, y)$; $\quad \mu \leftarrow \frac{1}{G}\sum_b r^{(b)}$; $\quad \sigma \leftarrow \sqrt{\frac{1}{G}\sum_b (r^{(b)} - \mu)^2}$
  $A^{(b)} \leftarrow \left[(r^{(b)} - \mu)/\max(\sigma, \delta)\right]$
  for $b = 1, \ldots, G$ **do**
    $\mathcal{S}^{(b)} \leftarrow \textsc{SampleSteps}(u, \tau)$ ▷ See Algorithm 2
  end for
  for $n = 1, \ldots, N$ **do**
    $\nabla_\theta \mathcal{L} \leftarrow 0$
    for $b = 1, \ldots, G$ **do**
      for $t \in \mathcal{S}^{(b)}$ **do**
        Apply masking to $o^{(b)}$ at step t to obtain denoising state $o_t^{(b)}$
        $\nabla_\theta \mathcal{L} \mathrel{+}= \frac{1}{Gk} \nabla_\theta \textsc{StepLoss}(o_0^{(b)}, o_t^{(b)})$ ▷ See Algorithm 3
      end for
    end for
    $\theta \leftarrow \theta - \eta \cdot \nabla_\theta \mathcal{L}$
  end for
end for

Algorithm 2 Weighted Timestep Sampling

Require: Recorded step probabilities u from rollout; sampling temperature τ; number of steps to sample k; trajectories $\mathcal{T}$

function $\textsc{SampleSteps}(u, \tau, k)$
  Compute per-step confidence $u_t \triangleq \dfrac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \phi_{\pi_{\theta_{\text{old}}}}(\hat{o}_{t-1,i} \mid q, o_t)$ for each $t \in \mathcal{T}$
  Sample k steps $\mathcal{S} \subset \mathcal{T}$ without replacement from $\mathrm{softmax}(-u_t/\tau)$
  return sampled step set $\mathcal{S}$
end function

Algorithm 3 Loss Estimation of Each Step

Require: Query q; original response o_0 and intermediate state o_t; current policy π_θ, old policy $\pi_{\theta_{\text{old}}}$, reference policy π_{ref}; KL weight β; inner iterations N

function $\textsc{StepLoss}(o_0, o_t)$
  Clip o_0 to obtain $\hat{o}_0$ by thresholding on $\pi_\theta(o_0 \mid q, o_t)$
  if $N = 1$ **then** ▷ REINFORCE
    $\ell_{\text{policy}} = -\dfrac{1}{|\hat{o}_0|} \log \pi_\theta(\hat{o}_0 \mid q,\, o_t)\, \hat{A}$
  else ▷ PPO
    $r_t(\theta) = \phi_{\pi_\theta}(\hat{o}_0 \mid q, o_t) \,/\, \phi_{\pi_{\theta_{\text{old}}}}(\hat{o}_0 \mid q, o_t)$
    $\ell_{\text{policy}} = \min\Big(r_t \hat{A},\ \mathrm{clip}(r_t,\ 1{-}\varepsilon,\ 1{+}\varepsilon)\, \hat{A}\Big)$
  end if
  $\ell_{\text{KL}} = D_{\text{KL}}\big[\phi_{\pi_\theta}(\cdot \mid q, o_t) \,\big\|\, \phi_{\pi_{\text{ref}}}(\cdot \mid q, o_t)\big]$
  $\ell(t) = \ell_{\text{policy}} + \beta\, \ell_{\text{KL}}$
  return $\ell(t)$
end function

$$\mathcal{L}_{\mathrm{RLDF}}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\, o\sim\pi_{\theta_{\mathrm{old}}}(\cdot|q)}\Bigg[-\frac{1}{G}\sum_{b=1}^{G} \mathrm{softmax}_t\Big(-\frac{1}{\tau\,|\mathcal{I}_t|}\sum_{i\in\mathcal{I}_t} \log \phi_{\pi_{\theta_{\mathrm{old}}}}(\hat{o}_{t-1,i} \mid q, o_t)\Big) \frac{1}{|\hat{o}_0^{(b)}|}\sum_{i\in\hat{o}_0^{(b)}} \Bigg[\min\Big(\frac{\phi_{\pi_\theta}(\hat{o}_{0,i}^{(b)} \mid q, o_t^{(b)})}{\phi_{\pi_{\theta_{\mathrm{old}}}}(\hat{o}_{0,i}^{(b)} \mid q, o_t^{(b)})}\hat{A}^{(b)}, \mathrm{clip}\Big(\frac{\phi_{\pi_\theta}(\hat{o}_{0,i}^{(b)} \mid q, o_t^{(b)})}{\phi_{\pi_{\theta_{\mathrm{old}}}}(\hat{o}_{0,i}^{(b)} \mid q, o_t^{(b)})}, 1-\varepsilon,\, 1+\varepsilon\Big)\hat{A}^{(b)}\Big) + \beta\, D_{\mathrm{KL}}\Big[\phi_{\pi_\theta}(\cdot \mid q, o_t^{(b)}) \,\|\, \phi_{\pi_{\mathrm{ref}}}(\cdot \mid q, o_t^{(b)})\Big]\Bigg]\Bigg] \tag{16}$$

$$\mathcal{L}_{\mathrm{RLDF}}^{\mathrm{token}}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\, o\sim\pi_{\theta_{\mathrm{old}}}(\cdot|q)}\Bigg[-\frac{1}{\sum_{b=1}^{G}|\hat{o}_0^{(b)}|}\sum_{b=1}^{G}\sum_{i\in\hat{o}_0^{(b)}} \mathrm{softmax}_t\Big(-\frac{1}{\tau\,|\mathcal{I}_t|}\sum_{j\in\mathcal{I}_t} \log \phi_{\pi_{\theta_{\mathrm{old}}}}(\hat{o}_{t-1,j} \mid q, o_t)\Big) \Bigg[\min\Big(\frac{\phi_{\pi_\theta}(\hat{o}_{0,i}^{(b)} \mid q, o_t^{(b)})}{\phi_{\pi_{\theta_{\mathrm{old}}}}(\hat{o}_{0,i}^{(b)} \mid q, o_t^{(b)})}\hat{A}^{(b)}, \mathrm{clip}\Big(\frac{\phi_{\pi_\theta}(\hat{o}_{0,i}^{(b)} \mid q, o_t^{(b)})}{\phi_{\pi_{\theta_{\mathrm{old}}}}(\hat{o}_{0,i}^{(b)} \mid q, o_t^{(b)})}, 1-\varepsilon,\, 1+\varepsilon\Big)\hat{A}^{(b)}\Big) + \beta\, D_{\mathrm{KL}}\Big[\phi_{\pi_\theta}(\cdot \mid q, o_{t,i}^{(b)}) \,\|\, \phi_{\pi_{\mathrm{ref}}}(\cdot \mid q, o_{t,i}^{(b)})\Big]\Bigg]\Bigg] \tag{17}$$

Model	**MATH500**			**GSM8K**			**AMC23**			**HumanEval**			**MBPP**		
	128	256	512	128	256	512	128	256	512	128	256	512	128	256	512
LLaDA Models															
LLaDA-8B-Instruct	32.4	36.4	39.0	76.0	83.3	81.5	7.5	15.0	12.5	40.9	37.2	37.2	37.0	36.8	38.4
LLaDA-1.5	31.8	40.8	39.6	76.2	82.3	83.5	15.0	12.5	10.0	43.3	36.0	39.0	36.8	38.4	38.6
d1	32.2	36.8	38.8	76.0	83.0	81.7	12.5	**17.5**	15.0	40.2	37.2	37.8	37.8	39.6	41.2
ESPO	38.6	38.2	41.8	78.8	83.7	**86.0**	10.0	**17.5**	20.0	40.9	37.8	41.5	**42.2**	**45.0**	42.8
TraceRL	32.0	33.8	33.0	77.6	80.7	82.4	7.5	5.0	20.0	40.9	34.8	37.8	36.8	39.2	40.4
Coupled-GRPO	33.0	40.6	37.0	77.9	83.5	83.6	15.0	10.0	**22.5**	39.6	39.6	39.6	38.6	41.8	41.4
RLDF (ours)	**40.8**	**43.6**	**43.0**	**79.3**	**84.6**	84.5	**17.5**	15.0	17.5	**43.9**	**43.3**	**42.7**	40.0	44.8	**45.8**
Δ vs. Base	+8.4	+7.2	+4.0	+3.3	+1.3	+3.0	+10.0	+0.0	+5.0	+3.0	+6.1	+5.5	+3.0	+8.0	+7.4
Dream Models															
Dream-7B-Instruct	36.2	38.4	39.2	66.2	74.9	73.9	15.0	17.5	17.5	48.2	50.0	51.2	50.8	51.0	51.8
d1	**42.2**	46.2	47.0	72.7	84.7	83.5	**20.0**	15.0	15.0	53.7	51.8	51.8	52.4	53.4	53.2
ESPO	30.8	42.4	43.4	66.4	80.7	79.8	15.0	15.0	17.5	52.4	55.5	53.0	50.8	51.8	51.4
TraceRL	35.0	46.0	44.4	65.4	83.4	84.8	10.0	15.0	17.5	52.4	51.2	53.7	53.6	54.6	53.8
Coupled-GRPO	40.2	43.6	45.4	70.0	82.3	82.9	12.5	17.5	**30.0**	53.0	53.0	56.1	51.8	52.2	52.2
RLDF (ours)	39.4	**48.4**	**49.0**	**74.0**	**86.7**	**86.6**	17.5	**30.0**	**30.0**	**58.5**	**57.9**	**57.9**	**58.2**	**57.2**	**56.8**
Δ vs. Base	+3.2	+10.0	+9.8	+7.8	+11.8	+12.7	+2.5	+12.5	+12.5	+10.4	+7.9	+6.7	+7.4	+6.2	+5.0

Table 4: Performance of different RL methods on LLaDA and Dream models across math and code benchmarks under the **static** unmasking strategy with varying generation steps (128, 256, and 512) with one token unmasked each step. The best result in each column is highlighted in **bold**, and Δ vs. Base denotes the absolute improvement of RLDF (ours) over the corresponding base model.