REINFORCING DIFFUSION LANGUAGE MODELS WITH CLEAN STATE OBJECTIVES ALONG DENOISING TRAJECTORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Policy loss estimation remains a fundamental and long-standing challenge in reinforcement learning (RL) for diffusion language models (DLMs). We introduce **Clean State Policy Optimization (CSPO)**, a novel training paradigm that selectively samples denoising timesteps and directly targets the final clean state for policy optimization. To reduce computational cost and stabilize training dynamics, CSPO optimizes the model toward the clipped clean state from intermediate noisy states, combined with weighted timestep sampling over denoising timesteps. Extensive experiments demonstrate that CSPO achieves consistent and substantial improvements in both performance and generalizability across two representative DLM architectures, LLaDA and Dream, on multiple reasoning benchmarks. Our work provides a practical approach to efficient and stable 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 & 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 AR 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 all-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

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

due to the iterative nature of the denoising process. Moreover, estimating the loss from only a few tokens decoded in subsequent steps leads to high variance and unstable estimates. **These limitations highlight a core challenge: how to preserve rollout-derived denoising contexts while keeping optimization computationally tractable and ensuring stable loss estimation.**

To address this challenge, we propose **Clean State Policy Optimization (CSPO)**, a training method that samples a subset of denoising timesteps and optimizes a clean state objective using the corresponding rollout contexts. Specifically, we employ a weighted sampling strategy (Rubinstein & Kroese, 2016) that prioritizes timesteps with high predictive uncertainty, motivated by their potential influence on the final generation (Choi et al., 2022). Restricting loss computation to these selected timesteps reduces the computational cost relative to evaluating the full denoising trajectory. At each selected timestep, we construct a clipped clean state target by retaining only tokens whose predicted probabilities exceed a confidence threshold.

Experiments on LLaDA and Dream demonstrate consistent improvements over the base models and competitive performance against existing RL methods on mathematical reasoning and code generation benchmarks, as shown in Table 2. Further evaluations show that these gains extend across datasets, generation lengths, and unmasking strategies. Ablation studies support the effectiveness of our design choices: uncertainty-guided timestep selection improves training dynamics, while clipped clean state supervision uses more token-level training signals and yields more stable training than next-state supervision. Our main contributions are as follows:

- We propose **Clean State Policy Optimization (CSPO)**, which selectively samples rollout timesteps and optimizes for clean states from the intermediate states, reducing computational cost while improving training stability.
- We open-source **Drift**, an extensible and modular RL framework purpose-built for DLMs, to facilitate community research and novel algorithms.
- We unlock the reasoning potential of DLMs (LLaDA and Dream) on extensive tasks, providing empirical insights for DLM community.

2 Related Work

2.1 Diffusion Language Model

Diffusion language models (DLMs) have evolved significantly, spanning approaches from continuous (Li et al., 2022; Dieleman et al., 2022) to discrete domains (Hoogeboom et al., 2021). Recently, masked diffusion language models (MDLMs) define a text generation paradigm in which the output is produced through an iterative denoising process over discrete token sequences (Austin et al., 2021a). In contrast to autoregressive models, which generate tokens sequentially from left to right, DLMs employ bidirectional attention mechanisms and perform predictions across all token positions at each forward pass. Recent advances in training methodology have yielded a number of capable DLM foundation models, including LLaDA (Nie et al., 2025), which is trained from scratch via full pre-training, and Dream (Ye et al., 2025), which is obtained by adapting the pretrained Qwen2.5-7B (Yang et al., 2025) to the diffusion framework.

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 (Gong 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, these methods introduce varying degrees of mismatch between the masking contexts used for optimization and those encountered during denoising. Motivated by this gap, we develop a trajectory-conditioned surrogate objective that empirically balances optimization effectiveness and computational cost for RL on DLMs.

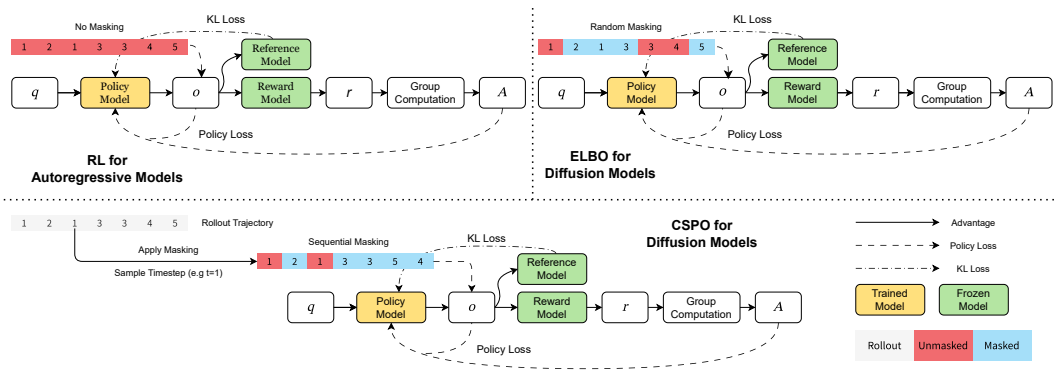


Figure 1: Comparison of training paradigms. **Left:** RL for autoregressive models without masking. **Right:** ELBO-based training for diffusion models with random masking. **Bottom:** Our proposed CSPO, which samples a timestep t and applies sequential masks. Policy loss is estimated based on $\pi_\theta(x_0 \mid x_t)$. In practice, x_0 will be clipped into $\hat{x}_0$ to mitigate excessive loss estimates and improve training stability.

3 Clean State Policy Optimization

3.1 Loss Estimation along Trajectories

As a context-aligned reference objective, one may evaluate token-prediction losses under every intermediate state encountered during rollout. 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.

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.

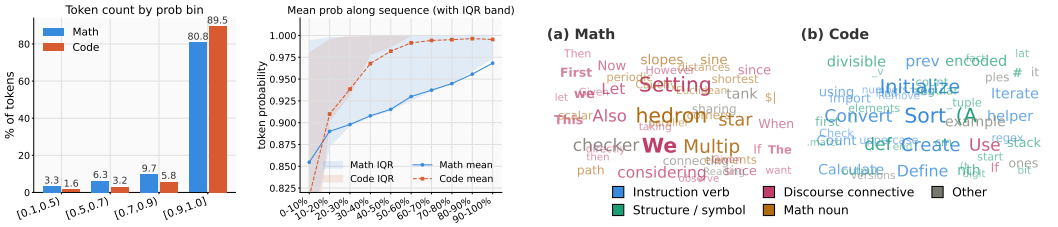
3.2 Clean State as Objective

Predicting x_0 is a common training objective for diffusion language models and directly targets the final clean output. In policy optimization for language models, focusing only on the few tokens denoised at each timestep can lead to noisy loss estimates and unstable training dynamics. We therefore use the final clean state as the optimization target. To prevent low-probability token predictions from degrading loss estimation, we retain only positions where the predicted probability of the corresponding clean-state token exceeds a predefined threshold. We denote the resulting clipped clean-state target by $\hat{x}_0$. The ablation results in Section 5.2 provide empirical support for this design.

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 mechanism for timestep selection.

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.



(a) Token probability distributions on two tasks. *(left)* Over 80% of tokens are unmasked with high conf. ($p \geq 0.9$) in both tasks. *(right)* Mean token probability along denoising timesteps (IQR shaded); later-unmasked tokens show higher conf. and lower var.

(b) 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.

Figure 2: **Token-level analysis during denoising process on Math and Code tasks.** (a) Per-token entropy distributions across Math and Code tasks; (b) word clouds of high-entropy tokens.

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

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

Table 1: Pearson (r) and Spearman (ρ) correlations between each token's unmasking probability and the entropy of its predicted distribution, computed on LLaDA and Dream across MATH and MBPP.

For diffusion language models, this sampling strategy reduces the number of loss-computation iterations while concentrating the optimization signal on uncertain denoising states. It therefore yields a computationally efficient, trajectory-conditioned objective that retains intermediate contexts observed during rollout.

3.4 CSPO Training Objective

The loss estimation procedure for clean state policy optimization (CSPO), 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 probability-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

$$u_t \triangleq -\frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \log \phi_{\pi_{\theta_{\text{old}}}} (\hat{o}_{t-1,i} \mid q, o_t), \qquad t \in \mathcal{T}. \tag{1}$$

where $\mathcal{I}_t$ denotes the set of token positions unmasked at timestep t, formally satisfying $\mathcal{I}_t \triangleq \{i : o_{t,i} \neq o_{t-1,i}\}$. $\phi_{\pi_{\theta_{\text{old}}}} (\hat{o}_{t-1,i} \mid q, o_t)$ 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. Formally, the selected timestep set is sampled as $S \sim \mathcal{P}(\cdot \mid w, k)$, where $w = (w_t)_{t\in\mathcal{I}}$ and $\mathcal{P}$ denotes the distribution over subsets of size k induced by weighted sampling without replacement from the candidate timestep set $\mathcal{I}$.

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 with PPO-style clipping:

$$\ell_{\text{policy}}^{(b)}(t) = -\frac{1}{|\hat{o}_0|}\sum_{i\in\hat{o}_0}\min\Big(r_{t,i}^{(b)}(\theta)\,\hat{A}^{(b)},\ \text{clip}\big(r_{t,i}^{(b)}(\theta),\,1\pm\varepsilon\big)\hat{A}^{(b)}\Big),\ \text{where } r_{t,i}(\theta) = \frac{\pi_\theta(\hat{o}_{0,i}\mid q,\,o_t)}{\pi_{\theta_{\text{old}}}(\hat{o}_{0,i}\mid q,\,o_t)}. \tag{3}$$

When only one iteration is performed, $\pi_\theta = \pi_{\theta_{\text{old}}}$, the clipping should be 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 k_3 KL-divergence estimator (Schulman, 2020) between π_θ and π_{ref}:

$$\ell_{\text{KL}}^{(b)}(t) = \mathbb{D}_{\text{KL}}[\pi_\theta(\hat{o}_0^{(b)}\mid q, o_t^{(b)})\,\|\,\pi_{\text{ref}}(\hat{o}_0^{(b)}\mid q, o_t^{(b)})]. \tag{5}$$

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

$$\mathcal{L}_{\text{CSPO}} = \frac{1}{G}\sum_{b=1}^{G}\frac{1}{|S_t^{(b)}|}\sum_{t\in S_t^{(b)}}\left[\ell_{\text{policy}}^{(b)}(t) + \beta\ell_{KL}^{(b)}(t)\right], \tag{6}$$

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. (7).

$$\mathcal{L}_{\text{CSPO}}(\theta) = \mathbb{E}_{\substack{q\sim\mathcal{D},\,\mathcal{T}^{(1:G)}\sim\pi_{\theta_{\text{old}}}(\cdot|q)\\ S^{(1:G)}\sim\mathcal{P}(\cdot|w^{(1:G)},k)}}\left[\frac{1}{G}\sum_{b=1}^{G}\frac{1}{|S^{(b)}|}\sum_{t\in S^{(b)}}\left\{-\frac{1}{|\hat{o}_0^{(b)}|}\sum_{i\in\hat{o}_0^{(b)}}\min\left(\frac{\phi_{\pi_\theta}(\hat{o}_{0,i}^{(b)}\mid q,o_t^{(b)})}{\phi_{\pi_{\theta_{\text{old}}}}(\hat{o}_{0,i}^{(b)}\mid q,o_t^{(b)})}\hat{A}^{(b)},\right.\right.\right.$$
$$\left.\left.\left.\text{clip}\left(\frac{\phi_{\pi_\theta}(\hat{o}_{0,i}^{(b)}\mid q,o_t^{(b)})}{\phi_{\pi_{\theta_{\text{old}}}}(\hat{o}_{0,i}^{(b)}\mid q,o_t^{(b)})},1-\varepsilon,\,1+\varepsilon\right)\hat{A}^{(b)}\right) + \beta\,\ell_{\text{KL}}^{(b)}(t;\theta)\right\}\right]. \tag{7}$$

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 (Austin et al., 2021b) and evaluate on MBPP 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. Full hyperparameter and implementation details are provided in Appendix E.

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

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
CSPO (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
CSPO (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) 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 CSPO (ours) over the corresponding base model.

Strong Performance Across Tasks and Models. Our clean state policy optimization (CSPO) 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. An unexpected 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 from autoregressive to masked diffusion modeling. Our reinforcement learning procedure 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 3. 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.

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
CSPO (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
CSPO (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 3: 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 CSPO (ours) over the corresponding base model.

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 3. 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 computation cost comparable to that of the rollout process alone.

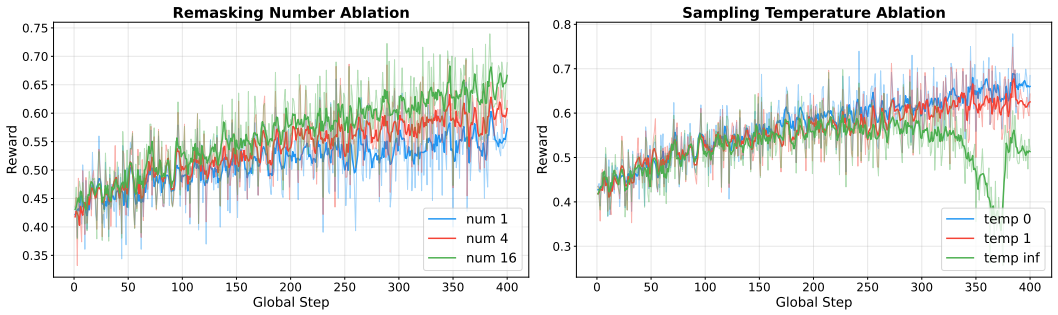


Figure 3: **Ablation on sequential remasking strategy.** Mean rollout reward under varying remasking counts and timestep sampling temperatures. Larger remasking counts and lower sampling temperatures consistently lead to higher rollout rewards.

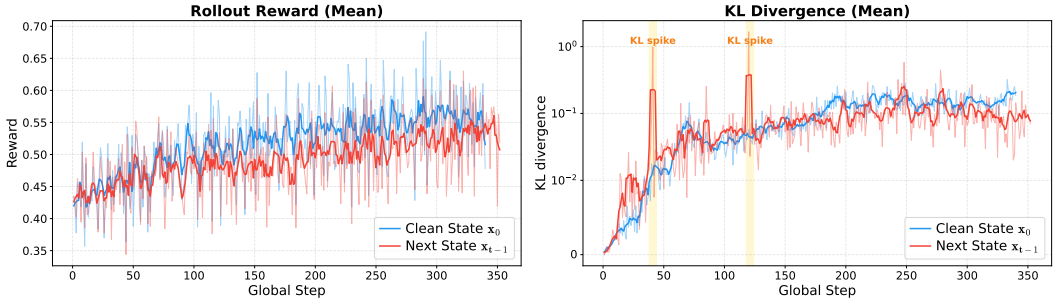


Figure 4: **Clean state estimation yields more stable training than next state estimation.** Training dynamics on LLaDA code tasks under two target choices ($\mathbf{x}_0$ vs. $\mathbf{x}_{t-1}$), measured by rollout reward and KL loss.

We further observe that the sampling temperature notably affects training stability: lower temperatures yield more stable dynamics. Since lower temperatures bias sampling toward high-entropy timesteps, we attribute this to the loss estimation behavior. High-entropy timesteps are dominated by the few informative tokens that most influence the final generation.

In contrast, high temperatures spread sampling toward low-entropy timesteps, whose tokens the model already predicts with high confidence. Repeatedly reinforcing these already-confident predictions drives their probabilities toward saturation and further collapses the entropy, ultimately destabilizing training. This is reflected in Figure 3, where the high-temperature run collapses sharply after roughly 300 steps.

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:

$$p_\theta(x_{t-1} \mid x_t) = \sum_{x_0} p_\theta(x_0 \mid x_t)\, q(x_{t-1} \mid x_t, x_0), \tag{8}$$

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 4, x_0-estimation leverages $\sim$13% of response tokens per sample, while x_{t-1}-estimation uses only $\sim$2%, discarding most of the available signal.

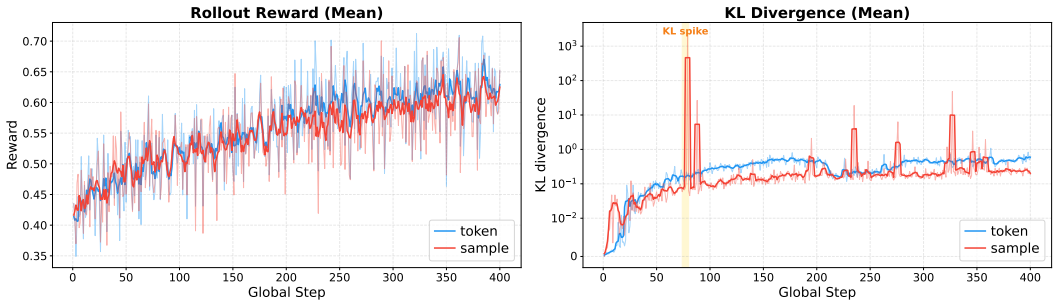


Figure 5: **Comparison of token-level and sample-level loss normalization (both with gradient clipping).** Token-level normalization leads to a more stable training process, with fewer loss spikes and faster convergence.

As shown in Figure 4, $\mathbf{x}_{t-1}$-based estimation also exhibits pronounced KL loss spikes during training, attributable to the instability introduced by computing KL divergence over such a limited token set; the policy loss suffers from the same issue. In contrast, $\mathbf{x}_0$-based estimation yields consistently higher rollout rewards, empirically confirming the superiority of clean state estimation for stable and effective training.

5.3 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-1}	N/A	8.46	0.0591	2.07%

Table 4: Comparison of training dynamics under different target states and gradient clipping settings. The first two columns specify the experimental setting, while the last three columns report the corresponding averaged training metrics. Grad Norm and Loss diverge largely without clipping.

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 4: 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. The choice of token normalization also influences training dynamics, as illustrated in Figure 2 and discussed further in Section D.

6 Conclusion

In this work, we address the fundamental challenge of policy loss estimation for reinforcement learning in diffusion language models. We propose **Clean State Policy Optimization (CSPO)**, which selectively samples denoising timesteps to improve training efficiency and employs clipped clean-state targets to ensure stable optimization.

Extensive empirical evaluations demonstrate both the effectiveness and practical value of our approach in enabling stable and performant reinforcement learning for DLMs. Furthermore, we introduce a comprehensive and extensible RL training framework **Drift** for DLMs, encompassing flexible masking strategies and diverse training methodologies, with native compatibility for the two most prominent DLM architectures, LLaDA and Dream.

AI USE STATEMENT

In this work, generative AI tools were used to improve the clarity, grammar, and presentation of the manuscript and to assist with debugging research code. All AI-assisted textual edits and suggested code changes were reviewed by the authors, and the modified code was tested and verified for correctness. Generative AI tools were not used to formulate research hypotheses, design the methodology or experiments, derive mathematical claims or proofs, or interpret experimental results. The authors take full responsibility for the final content of this work.

REFERENCES

Anthropic. Claude 3.7 sonnet system card. Technical report, Anthropic, 2025. URL https://www.anthropic.com/claude-3-7-sonnet-system-card.

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

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

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

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

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

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

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

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

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

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

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

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

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

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

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems*, 34:12454–12465, 2021.

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

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

Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, et al. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 2025.

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

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35: 4328–4343, 2022.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2023.

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

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

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

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

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

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

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

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

John Schulman. Approximating KL divergence. `http://joschu.net/blog/kl-approx.html`, 2020. Blog post.

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

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

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

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

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

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

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

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

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 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)*, pp. 9426–9439, 2024.

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

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

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

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

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

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

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

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

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

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

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

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

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:

$$q(x_t \mid x_0) = \prod_{i=1}^{L} \mathrm{Cat}\big(x_{t,i};\ (1-t)\,\mathbf{e}_{x_{0,i}} + t\,\mathbf{e}_{\texttt{[MASK]}}\big)\,. \tag{9}$$

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

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, MDLMs 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:

$$\mathcal{L}(\theta) \triangleq -\mathbb{E}_{t,\,x_0,\,x_t}\left[\frac{1}{t}\sum_{i=1}^{L}\mathbb{1}[x_{t,i} = \mathbf{M}]\log p_\theta(x_{0,i} \mid x_t)\right]. \tag{10}$$

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:

$$\mathcal{L}_{\text{full-seq}}(o \mid q) = \log p_\theta(o \mid q, o_T). \tag{11}$$

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

$$\mathcal{L}_{\text{ELBO}}(o \mid q) = \mathbb{E}_{t,\,m}\left[\log p_\theta\big(o \mid q,\ o^m\big)\right], \tag{12}$$

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

$$r_{t,i}(\theta) \;=\; \frac{\phi_{\pi_\theta}(o_{0,i} \mid c)}{\phi_{\pi_{\theta_{\text{old}}}}(o_{0,i} \mid c)}, \tag{13}$$

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 $i \in \{1, \ldots, |o^{(b)}|\}$:

$$\mathcal{L}_{\text{diffu}}(\theta) = \mathbb{E}\Bigg[-\frac{1}{G}\sum_{b=1}^{G} \frac{1}{|o^{(b)}|} \sum_{i=1}^{|o^{(b)}|} \min\Big(r_i^{(b)} A^{(b)},\; \text{clip}(r_i^{(b)}, 1-\varepsilon, 1+\varepsilon)\, A^{(b)} \Big) + \beta\, \mathbb{D}_{\text{KL}}\big[\phi_{\pi_\theta} \,\|\, \phi_{\pi_{\text{ref}}}\big] \Bigg] \tag{14}$$

The expectation is taken over $q \sim \mathcal{D}$, and $\{o^{(b)}\}_{b=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$. Because the loss is averaged uniformly over $|o^{(b)}|$ 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:

$$\mathcal{L}_{\text{ELBO}}(\theta) = \mathbb{E}\Bigg[-\frac{1}{G}\sum_{b=1}^{G} \frac{1}{|\mathcal{M}_b|} \sum_{i \in \mathcal{M}_b} \min\Big(r_i^{(b)} A^{(b)},\; \text{clip}(r_i^{(b)}, 1-\varepsilon, 1+\varepsilon)\, A^{(b)} \Big) + \beta\, \mathbb{D}_{\text{KL}}\big[\phi_{\pi_\theta} \,\|\, \phi_{\pi_{\text{ref}}}\big] \Bigg] \tag{15}$$

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:

$$\mathcal{L}_{\tau}(\theta) = \mathbb{E}\Bigg[-\frac{1}{G}\sum_{b=1}^{G} \frac{1}{T}\sum_{t=1}^{T} \frac{1}{|\mathcal{I}_t^{(b)}|} \sum_{i \in \mathcal{I}_t^{(b)}} \min\Big(r_{t,i}^{(b)} A^{(b)},\; \text{clip}(r_{t,i}^{(b)}, 1-\varepsilon, 1+\varepsilon)\, A^{(b)} \Big) + \beta\, \mathbb{D}_{\text{KL}}\big[\phi_{\pi_\theta} \,\|\, \phi_{\pi_{\text{ref}}}\big] \Bigg] \tag{16}$$

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:

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

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:

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

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

D Loss Normalization

Token-level loss normalization has been proposed to mitigate the response-length imbalance commonly 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. (7), the token-level loss is formulated as in Eq. (17). As shown in Figure 5, 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.

$$\mathcal{L}_{\text{CSPO}}^{\text{token}}(\theta) = \mathbb{E}_{\substack{q\sim\mathcal{D},\, \mathcal{T}^{(1:G)}\sim\pi_{\theta_{\text{old}}}(\cdot|q) \\ S^{(1:G)}\sim\mathcal{P}(\cdot|w^{(1:G)},k)}} \left[\frac{1}{\sum_{b=1}^{G} |\hat{o}_0^{(b)}|} \sum_{b=1}^{G} \frac{1}{|S^{(b)}|} \sum_{t\in S^{(b)}} \sum_{i\in\hat{o}_0^{(b)}} \left\{ -\min\left(\frac{\phi_{\pi_\theta}(\hat{o}_{0,i}^{(b)} \mid q, o_t^{(b)})}{\phi_{\pi_{\theta_{\text{old}}}}(\hat{o}_{0,i}^{(b)} \mid q, o_t^{(b)})} \hat{A}^{(b)}, \right.\right.\right.$$
$$\left.\left.\left. \text{clip}\left(\frac{\phi_{\pi_\theta}(\hat{o}_{0,i}^{(b)} \mid q, o_t^{(b)})}{\phi_{\pi_{\theta_{\text{old}}}}(\hat{o}_{0,i}^{(b)} \mid q, o_t^{(b)})}, 1-\varepsilon, 1+\varepsilon \right) \hat{A}^{(b)} \right) + \beta\, \ell_{\text{KL},i}^{(b)}(t;\theta) \right\} \right]. \tag{17}$$

E Implementation Details

E.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.

E.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.

E.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.

E.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, adding Gumbel noise to logits and taking the argmax to implicitly approximate softmax sampling, with temperature controlling noise magnitude. Dream instead uses explicit categorical sampling, applying temperature scaling with optional top-p/top-k truncation before sampling from the softmax distribution. The rollout batch size is set to 8, with 4 generations per sample.

F PLANNING

To broaden our evaluation, we apply each method to planning tasks, adopting the data-processing pipeline of Zhao et al. (2025). Table 5 reports results on the Countdown and Sudoku planning tasks across two diffusion-LM backbones (LLaDA and Dream) and both unmasking schedules. CSPO improves over the corresponding base model in all settings, with gains ranging from 34 to 71 points. These improvements are consistent across both backbones, whereas prior RL methods are markedly backbone-dependent. ESPO and Coupled-GRPO are competitive on LLaDA but degrade substantially on Dream, while d1 exhibits the opposite pattern. CSPO is the only method that remains strong on both backbones. We note that Countdown and Sudoku are highly structured and closely tied to their training distributions, which leaves room for solutions that exploit task-specific regularities rather than reflecting general reasoning ability. We therefore analyze mathematics and code in detail in our main experiments (Section 4.2).

Method	**LLaDA**								**Dream**							
	Dynamic				**Static**				**Dynamic**				**Static**			
	Countdown		**Sudoku**		**Countdown**		**Sudoku**		**Countdown**		**Sudoku**		**Countdown**		**Sudoku**	
	256	512	256	512	256	512	256	512	256	512	256	512	256	512	256	512
Base (Instruct)	26.2	21.5	9.2	2.2	26.6	22.7	8.6	2.2	27.7	25.8	41.8	31.4	30.1	27.0	26.6	22.0
d1	38.3	30.1	8.2	2.2	37.1	30.5	8.4	2.2	68.8	58.6	63.4	48.2	**69.1**	60.9	51.4	58.6
ESPO	70.3	64.1	81.8	55.0	71.1	64.1	83.0	61.4	29.7	19.9	59.2	65.8	26.6	19.1	55.2	61.8
Coupled-GRPO	58.6	52.3	**86.6**	**72.0**	57.0	51.2	**86.6**	**71.6**	32.8	29.7	35.6	41.6	33.6	27.7	42.4	37.0
CSPO (ours)	**75.0**	**74.6**	80.4	**72.0**	**75.0**	**75.0**	80.2	71.2	**69.5**	**67.2**	**81.2**	**82.6**	64.8	**64.1**	**78.0**	**82.6**
Δ vs. Base	+48.8	+53.1	+71.2	+69.8	+48.4	+52.3	+71.6	+69.0	+41.8	+41.4	+39.4	+51.2	+34.7	+37.1	+51.4	+60.6

Table 5: Performance of RL methods on **LLaDA** and **Dream** under dynamic and static unmasking, on Countdown (Countdown) and Sudoku with 256/512 generation steps. "Base (Instruct)" is LLaDA-8B-Instruct / Dream-7B-Instruct respectively. Best per column in **bold**; Δ vs. Base is CSPO over the corresponding base.

G ALGORITHM

We present the full procedure of CSPO in the algorithms below, organized as a main loop (Algorithm 1) together with two subroutines: a weighted timestep sampling function (Algorithm 2) and a per-timestep loss estimation (Algorithm 3). The main loop follows a GRPO-style group-relative scheme: for each sample, it draws a group of G responses from the old policy, normalizes their rewards into advantages, and updates the policy over N inner iterations. Since a full denoising trajectory contains far more timesteps than is practical to optimize over, the main loop calls SAMPLESTEPS to select an informative subset of k timesteps for each response, and TIMESTEPLOSS to compute the objective at each sampled timestep. We describe each subroutine in turn below.

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**
    $\mathcal{S}^{(b)} \leftarrow$ SAMPLESTEPS(u, τ) ▷ 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 $o_t^{(b)}$
        $\nabla_\theta \mathcal{L} \mathrel{+}= \frac{1}{Gk} \nabla_\theta$ TIMESTEPLOSS$(o_0^{(b)}, o_t^{(b)})$ ▷ 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 SAMPLESTEPS(u, τ, k)
  Compute per-step confidence $u_t \triangleq -\dfrac{1}{|\mathcal{I}_t|}\sum_{i \in \mathcal{I}_t} \log \phi_{\pi_{\theta_{\text{old}}}}(\hat{o}_{t-1,i} \mid q, o_t)$ 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 Timestep

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 TIMESTEPLOSS(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
  $\ell_{\text{KL}} = D_{\text{KL}}\left[\phi_{\pi_\theta}(\cdot \mid q, o_t) \,\middle\|\, \phi_{\pi_{\text{ref}}}(\cdot \mid q, o_t)\right]$
  $\ell(t) = \ell_{\text{policy}} + \beta\, \ell_{\text{KL}}$
  return $\ell(t)$
end function

H PROMPT

Code Prompt

```
{{problem}}
Place your code within a single Python code block
```python ```.
Do not include more than one code block.
```

Math Prompt

```
You need to put your final answer in \boxed{}.  This is the
problem:
{{problem}}
```

Sudoku Prompt

```
Please solve the following 4 × 4 Sudoku puzzle.  The puzzle is
provided as a 4×4 grid where '0' represents empty cells.

Rules:
- Fill empty cells with digits 1--4.
- Each row must contain digits 1--4 exactly once.
- Each column must contain digits 1--4 exactly once.
- Each 2×2 box must contain digits 1--4 exactly once.

Example:
Puzzle:
0 4 0 1
0 0 2 0
1 0 0 3
0 3 1 0

Solution:
2 4 3 1
3 1 2 4
1 2 4 3
4 3 1 2

Important:  Your solution must be a complete 4 × 4 grid using
only digits 1--4, with each row on a new line and digits
separated by single spaces.

Respond in this exact format:
<reasoning>
Your step-by-step solving process
</reasoning>
<answer>
row1
row2
row3
row4
</answer>

Now solve this puzzle:
Puzzle:
{{puzzle}}
```
```

Countdown Prompt

```
Please solve the following Countdown puzzle.

You are given 3 numbers: {{numbers}}. Your goal is to use
these numbers with basic arithmetic operations (+, -, *, /) to
reach the target number {{target}}.

Rules:
- You must use each number at most once.
- You can use +, -, *, / and parentheses.
- Intermediate results must be positive integers (no
fractions, no negative numbers).
- You do not need to use all the numbers.

Show your step-by-step reasoning, then output the full
expression that equals the target number inside \boxed{}.

For example, if the numbers are 5, 3, 2 and the target is 11,
your answer would be \boxed{5 + 3 * 2}.

Now solve this puzzle:
Numbers: {{numbers}}
Target: {{target}}
```

I EFFICIENCY OF CSPO

Let T denote the total number of denoising timesteps in a rollout trajectory and k the number of timesteps selected for policy-loss estimation, where $k \ll T$. Exhaustive trajectory-wise optimization evaluates the loss at every timestep, requiring T forward and backward passes per response. In contrast, CSPO evaluates the loss only at the k selected timesteps, reducing the loss-estimation complexity from $\mathcal{O}(T)$ to $\mathcal{O}(k)$. Under our default configuration of $T = 256$ and $k = 16$, this corresponds to a $16\times$ reduction in the number of loss- estimation passes. Moreover, k provides a flexible compute–performance trade-off and can be adjusted according to the available computational budget and hardware resources.

Exhaustive trajectory-wise optimization can also introduce substantial memory and communication overhead. Retaining computation graphs for multiple trajectory states increases activation memory, while processing the states sequentially requires repeated forward and backward passes. In memory-constrained implementations, activation checkpointing, gradient accumulation, or CPU offloading may be required. In particular, offloading intermediate activations or per-sample gradients to CPU memory and subsequently transferring them back to the GPU introduces additional host–device communication and increases training time. By evaluating only k trajectory states, CSPO reduces the number of such operations as well as the associated computation and communication overhead.

The resulting end-to-end acceleration is generally smaller than the theoretical T/k reduction because rollout generation still requires executing the denoising trajectory. More precisely, the costs of exhaustive trajectory-wise optimization and CSPO can be expressed as

$$C_{\text{full}} = C_{\text{rollout}} + TC_{\text{loss}} + C_{\text{sys}}(T), \tag{18}$$

$$C_{\text{CSPO}} = C_{\text{rollout}} + kC_{\text{loss}} + C_{\text{sys}}(k) + C_{\text{sample}}, \tag{19}$$

where C_{loss} denotes the cost of one loss-estimation forward/backward pass, C_{sys} captures implementation-dependent overhead such as activation recomputation, gradient accumulation, and CPU–GPU communication, and C_{sample} is the cost of timestep selection. Since CSPO computes the sampling weights from confidence values already recorded during rollout, C_{sample} is negligible relative to a model forward/backward pass. Therefore, the primary efficiency gain of CSPO lies in reducing the optimization cost relative to exhaustive trajectory-wise training, rather than reducing the cost of rollout generation or outperforming single-step objectives in computational efficiency.