Reinforcement Learning for Diffusion LLMs with Entropy-Guided Step Selection and Stepwise Advantages

Vishnu Teja Kunde* **Fatemeh Doudi*** **Mahdi Farahbakhsh**
Dileep Kalathil **Krishna Narayanan** **Jean-Francois Chamberland**

Department of Electrical and Computer Engineering
Texas A&M University

{vishnukunde,fatemehdoudi,mahdi.farahbakhsh, dileep.kalathil,krn,chmbrlnd}@tamu.edu

Abstract

Reinforcement learning (RL) has been effective for post-training autoregressive (AR) language models, but extending these methods to diffusion language models (DLMs) is challenging due to intractable sequence-level likelihoods. Existing approaches therefore rely on surrogate likelihoods or heuristic approximations, which can introduce bias and obscure the sequential structure of denoising. We formulate diffusion-based sequence generation as a finite-horizon Markov decision process over the denoising trajectory and derive the policy gradient that decomposes over denoising steps in terms of stepwise advantages, without requiring explicit evaluation of the sequence likelihood. Grounded in this result, we develop tractable approximations for large-scale training: (i) denoising steps are selected for policy updates via an entropy-guided approximation bound, and (ii) stepwise advantages are estimated using a one-step denoising completion naturally provided by the diffusion model, avoiding costly multi-step rollouts or auxiliary value networks. Experiments demonstrate state-of-the-art results on nearly all benchmarks spanning coding, logical reasoning, and mathematical reasoning, outperforming existing RL post-training approaches for DLMs. Code is available at this repo.

1 Introduction

Diffusion Language Models (DLMs) [Sahoo et al., 2024, Nie et al., 2025, Ye et al., 2025] have recently emerged as a compelling alternative to autoregressive language models (ARLMs). Instead of generating tokens sequentially from left to right, DLMs produce text through an iterative denoising process, typically via masked discrete diffusion, enabling bidirectional context and multi-token parallelism for higher token throughput while maintaining competitive output quality. These advantages have spurred rapid progress in DLM architectures and algorithms, including multimodal generation [Yang et al., 2025, Li et al., 2025], long-context modeling [Liu et al., 2025], accelerated inference [Wu et al., 2025], and code generation [Song et al., 2025]. Motivated by the transformative impact of reinforcement learning (RL) post-training on ARLMs [Guo et al., 2025], there is now growing interest in developing effective and scalable RL algorithms tailored to DLMs.

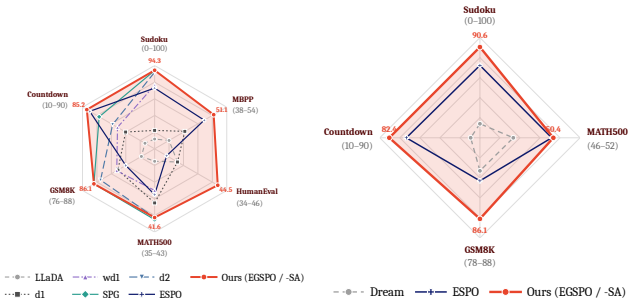
Despite the success of RL post-training for ARLMs, extending these methods to DLMs is *not* a direct translation. RL for ARLMs relies on causal token-wise factorization, which yields a natural token-space Markov decision process (MDP) and enables efficient computation of log-likelihoods and importance ratios. DLMs fundamentally break this structure: generation proceeds through a denoising trajectory in masked space, and the likelihood of the final output does not admit a readily usable

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

token-wise decomposition. As a result, naïvely porting standard policy-gradient objectives leads to intractable or prohibitively expensive likelihood evaluations. At the same time, diffusion generation offers opportunities largely absent in ARLMs. Model uncertainty evolves non-uniformly along the denoising trajectory, motivating *stepwise credit/advantage assignment* and *stepwise compute allocation* across diffusion steps. Moreover, masked DLMs output token distributions for all positions at each step, enabling a lightweight "full unmasking" that can serve as an increasingly accurate proxy of the eventual output later in the trajectory, and thus provide partial learning signals for intermediate steps without an explicit pretrained value function. Together, these challenges and opportunities motivate a principled approach to RL for DLMs that exploits diffusion structure.

A growing body of work has begun exploring RL post-training for DLMs via surrogate objectives and tractable likelihood approximations [Zhao et al., 2025b, Tang et al., 2025, Wang et al., 2025a, Ou et al., 2026, Wang et al., 2025c,d, Zhao et al., 2025a]. While practically effective, these methods typically begin with a *chosen approximation* to the likelihood, objective, or advantage and optimize the resulting surrogate directly, leaving the connection to the true RL objective less explicit and step-level optimization across the denoising trajectory largely implicit (see Section 2 for detailed comparison).



(a) (b)

Figure 1: Benchmark accuracy for (a) LLaDA-8B-Instruct and (b) Dream-7B-Instruct. Ours reports the better of EGSPO and EGSPO-SA per task, using the best accuracy across generation lengths. Axis ranges are shown beside each benchmark. See Tables 1, 2, and 3 for results and evaluation protocols.

We take a complementary, first-principles approach that makes the diffusion structure explicit, rather than treating a DLM as a black-box sampler. Instead of beginning with a particular surrogate likelihood approximation, we ask the fundamental questions that should underlie any principled RL method: What is the right MDP formalism for DLMs? What is the policy gradient for the true RL objective? How can it be approximated tractably at scale? And how can diffusion-time structure enable *stepwise* advantage (credit) estimation and *compute allocation* across denoising steps? We answer these questions in the affirmative, leading to the following main contributions.

- **Policy gradient with stepwise advantages:** Using a finite-horizon MDP formulation of masked diffusion generation, we derive a policy gradient that decomposes over denoising steps, yielding a principled notion of *stepwise advantages* without requiring explicit evaluation of the final sequence likelihood.
- **Tractable estimators exploiting diffusion structure:** We turn this theory into a practical algorithm by exploiting two DLM-native capabilities absent in ARLMs. First, we allocate training compute *across denoising steps* using the model's intrinsic uncertainty: higher-entropy steps are prioritized under a fixed budget. We call this method **E**ntropy-**G**uided **S**tepwise **P**olicy **O**ptimization (**EGSPO**). Second, we estimate stepwise advantages via a lightweight full-sequence "one-shot" completion from intermediate states, yielding intermediate learning signals without an extra value network or costly multi-step rollouts. We call this method **E**ntropy-**G**uided **S**tepwise **P**olicy **O**ptimization with **S**tepwise **A**dvantages (**EGSPO-SA**). Together, these ideas make RL training scalable for DLMs while preserving the diffusion-time structure.
- **State-of-the-art results:** We achieve state-of-the-art results on nearly all benchmarks spanning *coding*, *logical reasoning*, and *mathematical reasoning*, outperforming existing RL post-training approaches for DLMs on LLaDA [Nie et al., 2025] and Dream [Ye et al., 2025] models.

2 Related Work

Diffusion Language Models: DLMs [Sahoo et al., 2024, Nie et al., 2025, Ye et al., 2025] extend discrete diffusion frameworks [Austin et al., 2021a, Lou et al., 2023, Song et al., 2025] to natural language, enabling parallel multi-token generation. Prior work has explored architectural simplifications [Shi et al., 2024], theoretical connections between discrete and continuous diffusion [Zhao et al., 2024], and multimodal extensions [Yang et al., 2025, Li et al., 2025]. Inference efficiency has been

improved via remasking schemes [Wang et al., 2025b], token reordering [Arriola et al., 2025, Sahoo et al., 2025], and entropy- or confidence-based sampling [Ben-Hamu et al., 2025, Wu et al., 2025].

RL for Diffusion Image Models: RL is used for preference alignment in image diffusion models via policy-gradient methods [Black et al., 2024, Fan et al., 2023, Uehara et al., 2024] or preference-based optimization [Wallace et al., 2024, Yang et al., 2024]. DDPO [Black et al., 2024] similarly casts denoising as an MDP, but derives the policy gradient only in reward form without stepwise value functions or advantages. Furthermore, image diffusion models are orders of magnitude smaller than 7–8B DLMs, so running all T denoising steps per update is affordable there; our entropy-guided step selection addresses the budget constraint that arises specifically at LLM scale.

RL for Diffusion Language Models: RL post-training for DLMs has grown rapidly. *d1* [Zhao et al., 2025b] adapts GRPO via a mean-field likelihood approximation; *wd1* [Tang et al., 2025] stabilizes training with weighted policy updates while retaining similar likelihood proxies; *SPG* [Wang et al., 2025a] optimizes pessimistic/optimistic surrogate objectives derived from likelihood bounds; *ESPO* [Ou et al., 2026] treats entire sequence generation as a single action and uses the ELBO as a tractable sequence-level likelihood proxy with per-token importance ratio normalization; *TraceRL* [Wang et al., 2025d] trains a trajectory-aware diffusion value model; and *DiFFPO* [Zhao et al., 2025a] uses surrogate policies with importance sampling corrections. Despite their empirical success, these methods begin with a chosen approximation to the likelihood, objective, or advantage and optimize it directly, leaving step-level credit assignment largely implicit and the connection to the true RL objective opaque. The most closely related work is *d2* [Wang et al., 2025c], which derives a reward-weighted policy gradient without an explicit MDP, using step-merging estimators that collapse multiple denoising steps into a single update and discard per-step structure; our entropy-guided step selection instead retains the most informative individual steps via Lemma 4. *TraceRL* [Wang et al., 2025d] is conceptually similar to *d2* but evaluates on non-standard model architectures, so we treat *d2* as the canonical trajectory-level baseline. Hong et al. [2026] train an unmasking-head policy, focusing on the unmasking mechanism rather than finetuning of the model for a specific task. Xie et al. [2025] augment Diffu-GRPO with process rewards that measure the contribution of denoising intervals to final correctness. Our method instead estimates stepwise advantages by comparing the final reward with rewards of greedy one-step completions from intermediate states.

3 Preliminaries and Problem Formulation

3.1 Preliminaries

Notations. Let $\mathbf{x} = \mathbf{x}_{0:T,1:L}$ denote a sequence of $T+1$ steps of L-length sentences. Each of $\mathbf{x}_t$ can take values from a finite set of vocabulary $\mathcal{V} \cup \{\mathbf{m}\}$ where $\mathbf{m}$ is a special token denoting mask. The state $\mathbf{x}_0$ denotes the clean sentence and can take values only in $\mathcal{V}$.

Masked Diffusion Language Model (MDLM): MDLMs [Sahoo et al., 2024, Nie et al., 2025, Song et al., 2025, Ye et al., 2025] generate text via iterative unmasking. A clean sequence $\mathbf{x}_0$ is masked as $\mathbf{x}_{t,\ell} \sim \mathrm{Cat}(\cdot \mid \alpha_t \mathbf{x}_{0,\ell} + (1-\alpha_t)\mathbf{m})$ with decreasing α_t. The reverse process is parameterized by $f_{\boldsymbol{\theta}}$: given $\mathbf{x}_{t+1}$, denoised samples follow $\mathbf{x}_{s,\ell} \sim \pi_{\boldsymbol{\theta}}^{s|t}(\cdot \mid \mathbf{x}_t) = q^{s|t,0}(\cdot \mid \mathbf{x}_t, \mathbf{x}_0 = f_{\boldsymbol{\theta}}(\cdot \mid \mathbf{x}_t))$, and iterative generation factorizes as $\pi_{\boldsymbol{\theta}}{}^{s:t-1|t}(\mathbf{x}_{s:t-1} \mid \mathbf{x}_t) \triangleq \prod_{u=t-1}^{s} \pi_{\boldsymbol{\theta}}{}^{u|u+1}(\mathbf{x}_u \mid \mathbf{x}_{u+1})$. Let $M_t(\mathbf{x}_{t+1}) \triangleq \{\ell : \mathbf{x}_{t+1,\ell} = \mathbf{m}\}$ denote masked positions at step t. Since $f_{\boldsymbol{\theta}}$ models a product distribution over masked positions, it induces a one-step distribution $\pi_{\boldsymbol{\theta}}{}^{0|t}(\mathbf{x}_0 \mid \mathbf{x}_t) = \prod_{\ell:\mathbf{x}_{t,\ell}=\mathbf{m}} f_{\boldsymbol{\theta},\ell}(\mathbf{x}_{0,\ell} \mid \mathbf{x}_t)$. We drop time-step superscripts on $\pi_{\boldsymbol{\theta}}$ where clear from context. The denoising network $f_{\boldsymbol{\theta}}$ is trained to predict clean tokens $\mathbf{x}_0$ from a partially masked sequence, rather than directly predicting the next intermediate state. A masked-token reconstruction objective takes the form

$$\mathcal{L}_{\mathrm{denoise}}(\boldsymbol{\theta}) = -\mathbb{E}_{\mathbf{x}_0,u,\mathbf{x}_u}\left[w_u \sum_{\ell:\mathbf{x}_{u,\ell}=\mathbf{m}} \log f_{\boldsymbol{\theta},\ell}(\mathbf{x}_{0,\ell} \mid \mathbf{x}_u) \right]. \tag{1}$$

For LLaDA [Nie et al., 2025], $u \sim \mathrm{Unif}(0,1)$ is the masking probability, $w_u = 1/u$, and $\mathbf{x}_0$ is drawn from the training data. During generation, the sampler uses these clean-token predictions to construct the next partially unmasked state.

RL for LLMs: Given a query $\mathbf{q} \sim \mathcal{D}$, an LM generates an output $\mathbf{x}_0$ and receives a reward $r(\mathbf{x}_0, \mathbf{q})$. RL fine-tuning seeks to maximize the expected reward:

$$J(\boldsymbol{\theta}) \triangleq \mathbb{E}_{\mathbf{q},\mathbf{x}_0\sim\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{q})}[r(\mathbf{x}_0, \mathbf{q})]. \tag{2}$$

Policy-gradient methods are standard tools for optimizing (2). Among them, the Group Relative Policy Optimization (GRPO) [Shao et al., 2024] is widely popular for RL with Verifiable Reward (RLVR) tasks, as it eliminates the need for a trained value model used in PPO-style algorithms [Schulman et al., 2017]. The GRPO loss function for an ARLM is given by

$$L_{\mathrm{AR}}(\boldsymbol{\theta}) = -\mathbb{E}_{\mathbf{q},\mathbf{x}_0\sim\pi_{\boldsymbol{\theta}_{\mathrm{old}}}(\cdot|\mathbf{q})}\left[\frac{1}{L}\sum_{\ell=1}^{L}\min\big(\rho_\ell A^{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}},\ \mathrm{clip}_{1-\epsilon}^{1+\epsilon}(\rho_\ell)A^{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}}\big)\right] + \beta D_{\mathrm{KL}}(\pi_{\boldsymbol{\theta}}\|\pi_{\mathrm{ref}}),$$

where $\rho_\ell = \frac{\pi_{\boldsymbol{\theta}}(\mathbf{x}_{0,\ell}|\mathbf{x}_{0,<\ell},\mathbf{q})}{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}(\mathbf{x}_{0,\ell}|\mathbf{x}_{0,<\ell},\mathbf{q})}$ is the importance ratio, and the group-relative advantage $A^{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}}(\mathbf{x}_0^j, \mathbf{q}) = r(\mathbf{x}_0^j, \mathbf{q}) - \frac{1}{G}\sum_{i=1}^{G} r(\mathbf{x}_0^i, \mathbf{q})$ is estimated from G completions $\mathbf{x}_0^j \sim \pi_{\boldsymbol{\theta}_{\mathrm{old}}}(\cdot \mid \mathbf{q})$.

3.2 Problem Formulation: RL for Diffusion LLMs

RL finetuning of an AR-LLM exploits its auto-regressive structure by modeling the token generation as a token-space MDP, and the LLM as a policy. Since $\mathbf{x}_0 = \mathbf{x}_{0,1:L}$, we can define the initial state as $\mathbf{q}$, state at step ℓ as $(\mathbf{q}, \mathbf{x}_{0,<\ell})$ and action at step ℓ as the next token $\mathbf{x}_{0,\ell} \sim \pi_{\boldsymbol{\theta}}(\cdot \mid \mathbf{x}_{0,<\ell}, \mathbf{q})$. Any policy gradient algorithm for solving (2) has to calculate the term $\pi_{\boldsymbol{\theta}}(\mathbf{x}_0 \mid \mathbf{q})$, and here it decomposes as $\pi_{\boldsymbol{\theta}}(\mathbf{x}_0 \mid \mathbf{q}) = \prod_{\ell=1}^{L}\pi_{\boldsymbol{\theta}}(\mathbf{x}_{0,\ell} \mid \mathbf{x}_{0,<\ell}, \mathbf{q})$. Due to causal attention of AR-LLM, all these terms can be efficiently computed using only a single model forward pass. **Unfortunately, DLMs break these conveniences** because there is no token-wise causal factorization of $\pi_{\boldsymbol{\theta}}(\mathbf{x}_0 \mid \mathbf{q})$ available. Naively porting the RL training methods for AR-LLM would require evaluating (explicitly or implicitly) sequence-level likelihood terms that are intractable as there is no direct decomposition over tokens. Prior work addresses this with surrogate objectives and likelihood approximations [Zhao et al., 2025b, Tang et al., 2025, Wang et al., 2025a,c, Ou et al., 2026], but these approaches necessarily make compromises: the optimized objective may deviate from Eq. (2), gradients may be biased, and the role of *individual denoising steps* in producing the final reward is often ignored.

In this work, we take a complementary approach. Rather than beginning with a particular surrogate likelihood approximation, we ask the foundational questions that should underlie any principled RL method for diffusion LMs. (i) *MDP formalism:* What are the states, actions, and transitions that faithfully represent diffusion-based generation while remaining amenable to RL analysis? (ii) *Policy-gradient for Eq.* (2)*:* Can we derive an *unbiased* gradient expression that respects the sequential structure of denoising? How can we estimate that gradient tractably? (iii) *Stepwise advantages:* In AR decoding, actions are tied to token positions; in diffusion decoding, actions are tied to denoising step. Can diffusion structure yield improved stepwise advantages that do not have direct AR analogues? We provide positive answers to these questions in Section 4.

4 Methodology

4.1 Diffusion MDP and Policy Gradient

We formalize the unmasking process as a T-step MDP indexed by diffusion time t running from T down to 0. The state at diffusion step t is $(\mathbf{x}_t, \mathbf{q})$, the action is the next denoised state $\mathbf{x}_{t-1}$, and transitions are deterministic given the policy. The reward is zero at all intermediate steps and equals $r(\mathbf{x}_0, \mathbf{q})$ at the terminal step $t = 0$. The value of a policy π at time step t is given by

$$V_t^{\pi}(\mathbf{x}_t, \mathbf{q}) = \mathbb{E}_{\mathbf{x}_{<t}\sim\pi^{<t|t}(\cdot|\mathbf{x}_t,\mathbf{q})}[r(\mathbf{x}_0, \mathbf{q})], \tag{3}$$

where $\mathbf{x}_{<t} \sim \pi^{<t|t}(\cdot \mid \mathbf{x}_t, \mathbf{q})$ denotes that the expectation is with respect to the trajectories that start at $\mathbf{x}_t$ and generate $\mathbf{x}_0$ in t steps. The value of a policy V^{π}, averaged over the initial state, is then defined as $V^{\pi} = \mathbb{E}_{\mathbf{q}}[V_T^{\pi}(\mathbf{m}, \mathbf{q})]$. By comparing Eq. (2) and Eq. (3), it is straightforward to see that $J(\boldsymbol{\theta}) = V^{\pi_{\boldsymbol{\theta}}}$, showing that our MDP formalism will indeed solve the finetuning problem. We now derive the policy gradient for the diffusion MDP.

Proposition 1 (Policy Gradient). *The gradient of the objective $J(\boldsymbol{\theta})$ (c.f. Eq. (2)) is given by*

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{q},\mathbf{x}\sim\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{q})}[r(\mathbf{x}_0,\mathbf{q})\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{q})] \tag{4}$$

$$= \sum_{t=0}^{T-1} \mathbb{E}_{\mathbf{q},\mathbf{x}\sim\pi_{\boldsymbol{\theta}}}[A_t^{\pi_{\boldsymbol{\theta}}}(\mathbf{x}_{t+1},\mathbf{x}_0,\mathbf{q})\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t+1})], \tag{5}$$

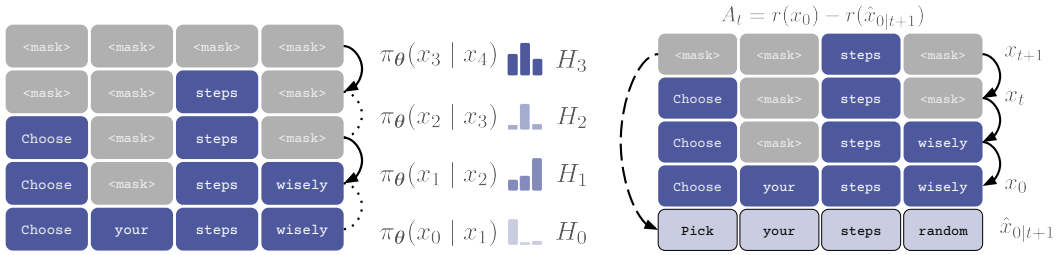
where the stepwise advantage is given by

$$A_t^{\pi_{\boldsymbol{\theta}}}(\mathbf{x}_{t+1},\mathbf{x}_0,\mathbf{q}) \triangleq r(\mathbf{x}_0,\mathbf{q}) - V_{t+1}^{\pi_{\boldsymbol{\theta}}}(\mathbf{x}_{t+1},\mathbf{q}). \tag{6}$$

Here $\mathbf{x} \triangleq \mathbf{x}_{0:T}$ denotes the full denoising trajectory and $\log \pi_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{q}) = \sum_{t=0}^{T-1} \log \pi_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t+1})$. The proof is given in Appendix A.

Remark 2 (Variance reduction via per-step baseline). Both forms are exact. Eq. (4) broadcasts raw reward uniformly; Eq. (5) subtracts the per-step baseline $V_{t+1}^{\pi_{\boldsymbol{\theta}}}$, concentrating the signal on steps where the policy's expected return deviates from the final outcome and reducing variance. Estimating $V_{t+1}^{\pi_{\boldsymbol{\theta}}}$ tractably is addressed in Section 4.3.

Remark 3. Let $U_t \triangleq \{\ell : \mathbf{x}_{t,\ell} \neq \mathbf{x}_{t+1,\ell}\}$ denote the positions unmasked at step t. Since the unmasked tokens are conditionally independent given $\mathbf{x}_{t+1}$, the log-policy decomposes as $\log \pi_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t+1}) = \log \pi_{\boldsymbol{\theta}}(U_t \mid \mathbf{x}_{t+1}) + \sum_{\ell\in U_t} \log f_{\boldsymbol{\theta},\ell}(\mathbf{x}_{t,\ell} \mid \mathbf{x}_{t+1})$. In practice, U_t is chosen by confidence-based decoding: unmasking the positions with the highest $\max_v f_{\boldsymbol{\theta},\ell}(v \mid \mathbf{x}_{t+1})$. The position-selection term $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(U_t \mid \mathbf{x}_{t+1})$ is not differentiable in the standard sense, since U_t is determined by an argmax (top-k) over confidence scores. We therefore treat U_t as fixed when differentiating, following Wang et al. [2025c], yielding $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t+1}) = \sum_{\ell\in U_t} \nabla_{\boldsymbol{\theta}} \log f_{\boldsymbol{\theta},\ell}(\mathbf{x}_{t,\ell} \mid \mathbf{x}_{t+1})$ and $\pi(\mathbf{x}_t \mid \mathbf{x}_{t+1}) = \prod_{\ell\in U_t} f_{\boldsymbol{\theta},\ell}(\mathbf{x}_{t,\ell} \mid \mathbf{x}_{t+1})$. This is an empirical design choice. However, the resulting gradient still aligns with the correct objective in the following sense: improving the token log-likelihood at a selected position ℓ increases $\max_v f_{\boldsymbol{\theta},\ell}(v \mid \mathbf{x}_{t+1})$, which reinforces that position being selected by confidence-based decoding in future rollouts, so the two components of the gradient are implicitly coupled through training.



(a) **Entropy-Guided Step Selection** (b) **Stepwise Advantage Estimation**

Figure 2: (a) **Entropy-guided step selection**: K steps with highest entropy H_t are selected (solid lines); with $H_3 > H_1 > H_2 > H_0$ and K=2, steps 3 and 1 are chosen. (b) **Stepwise advantage estimation**: Reward of greedy one-step completion $\hat{\mathbf{x}}_{0|t+1}$ from $\mathbf{x}_{t+1}$ approximates V_t; A_t measures the additional reward from continuing the full denoising trajectory to $\mathbf{x}_0$.

4.2 Entropy-Guided Step Selection

In ARLMs, given a sequence $\mathbf{x}_0$, the network output at position ℓ models the likelihood $\pi_{\boldsymbol{\theta}}(\mathbf{x}_{0,\ell} \mid \mathbf{x}_{0,<\ell})$ for all $\ell \in [1:L]$ due to the causal attention mask. Thus, a single forward pass gives all the required likelihoods. This convenience is broken in diffusion LM due to its bidirectional attention, and it can compute only the term $\pi_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t+1})$ in the forward pass of $f_{\boldsymbol{\theta}}(\cdot \mid \mathbf{x}_{t+1})$. The policy gradient in Eq. (5) has T such terms, each requiring a separate forward pass through the network. This is computationally expensive as T is typically of the order of $10^2 - 10^3$.

To overcome this challenge, we take the gradient only at a subset $S \subseteq [0:T-1]$ with $|S| \leq K$. Writing $\nabla_{\boldsymbol{\theta}} J_t(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{q},\mathbf{x}\sim\pi_{\boldsymbol{\theta}}}[A_t^{\pi_{\boldsymbol{\theta}}} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t+1})]$, so that $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_t \nabla_{\boldsymbol{\theta}} J_t(\boldsymbol{\theta})$, the subset approximation is $\nabla_{\boldsymbol{\theta}} J_S(\boldsymbol{\theta}) = \sum_{t\in S} \nabla_{\boldsymbol{\theta}} J_t(\boldsymbol{\theta})$. Two intuitive heuristics are: *(i) Random*: sample K steps uniformly at random; *(ii) Uniform*: take $S = \{kT/K : k \in [0{:}K{-}1]\}$. However, these approaches overlook the structure of the policy at each denoising step along the trajectory. A more principled approach would be to choose S with $|S| \leq K$ such that the gradient approximation

error $\Delta_S = \|\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} J_S(\boldsymbol{\theta})\|$ is minimized. Since evaluating this objective Δ_S directly is expensive, we instead minimize a surrogate objective.

Lemma 4. *Let ${\pi_{\boldsymbol{\theta}}}^{t|t+1}$ be the softmax policy with logits $g_{\boldsymbol{\theta}}^{t,i}$ over $i \in [|\mathcal{V}|]$ for each $t \in [0 : T-1]$. Let $\|\nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta}}^t\| \triangleq \max_{i,j} \|\nabla_{\boldsymbol{\theta}}(g_{\boldsymbol{\theta}}^{t,i} - g_{\boldsymbol{\theta}}^{t,j})\|$ and assume that $\max_t \|\nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta}}^t\| \leq B$. Let $H({\pi_{\boldsymbol{\theta}}}^{t|t+1})$ be the entropy of $\pi_{\boldsymbol{\theta}}(\cdot \mid \mathbf{x}_{t+1})$. Defining $\Delta_S = \|\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} J_S(\boldsymbol{\theta})\|$, we have*

$$\Delta_S \leq B(\max_{t \in [T]} |A_t^{\pi_{\boldsymbol{\theta}}}|) \sum_{t \notin S} H({\pi_{\boldsymbol{\theta}}}^{t|t+1}). \tag{7}$$

Proof is given in Section A.

Remark 5. The factor $\max_t |A_t^{\pi_{\boldsymbol{\theta}}}|$ in Eq. (7) is bounded by $r_{\max} - r_{\min}$, the range of the reward function. In practice, rewards are bounded in a known interval $[-M, M]$.

Using the upper-bound in Eq. (7) as the surrogate objective for Δ_S, we solve the following optimization problem to select time steps S such that

$$\min_{S \subseteq [0:T-1]} \sum_{t \notin S} H({\pi_{\boldsymbol{\theta}}}^{t|t+1}), \quad |S| \leq K. \tag{8}$$

The solution of Eq. (8) is given by the greedy choice:

$$S^* = \text{top}[K](t : H({\pi_{\boldsymbol{\theta}}}^{t|t+1})) \tag{9}$$

Intuitively, at these steps the policy has the highest uncertainty, so allocating gradient computation to them minimizes the upper bound on the gradient approximation error Δ_S. This bound may be loose and does not guarantee that the selected subset minimizes Δ_S itself. A diagnostic comparison with full-step updates in Appendix B further supports the empirical effectiveness of entropy-guided selection.

4.3 Estimating Stepwise Advantages

To compute the stepwise advantage $A_t^{\pi_{\boldsymbol{\theta}}}$, we need to evaluate the value function $V_t^{\pi_{\boldsymbol{\theta}}}$ for each t (c.f. Eq. (3)). Evaluating it using Monte Carlo sampling will require generating multiple trajectories that start at $\mathbf{x}_t$ and reach $\mathbf{x}_0$ in t steps, which requires a large number of additional forward passes through the model. To overcome this difficulty, we leverage a structure unique to DLMs: $f_{\boldsymbol{\theta}}$ directly models a distribution over clean tokens $\mathbf{x}_0$ from any masked state $\mathbf{x}_t$ via the one-step denoising distribution ${\pi_{\boldsymbol{\theta}}}^{0|t}$ (c.f. Section 3.1). *This is a capability ARLMs do not possess without a separate value network.* We use it to propose:

$$\hat{V}_t^{\pi_{\boldsymbol{\theta}}}(\mathbf{x}_t, \mathbf{q}) = \mathbb{E}_{\mathbf{x}_0 \sim {\pi_{\boldsymbol{\theta}}}^{0|t}(\cdot|\mathbf{x}_t, \mathbf{q})}[r(\mathbf{x}_0, \mathbf{q})]. \tag{10}$$

This expectation is further approximated by the reward evaluated for a single greedy completion $\hat{\mathbf{x}}_{0|t}$, **available at no extra cost** from the forward pass already computed during rollout:

$$\hat{\mathbf{x}}_{0,\ell}(\mathbf{x}_t) = \begin{cases} \arg\max_v f_{\boldsymbol{\theta},\ell}(v \mid \mathbf{x}_t), & \mathbf{x}_{t,\ell} = \mathbf{m} \\ \mathbf{x}_{t,\ell}, & \text{otherwise.} \end{cases} \tag{11}$$

The approximation $\hat{V}_{t+1}^{\pi_{\boldsymbol{\theta}}}$ is biased: the one-step surrogate ${\pi_{\boldsymbol{\theta}}}^{0|t+1}$ diverges from the full t-step rollout when many tokens remain masked (large t), with the bias diminishing as $t \to 0$. Using $\hat{V}_{t+1}^{\pi_{\boldsymbol{\theta}}}$, we define the approximate stepwise advantage:

$$\hat{A}_t^{\pi_{\boldsymbol{\theta}}}(\mathbf{x}_{t+1}, \mathbf{x}_0, \mathbf{q}) \triangleq r(\mathbf{x}_0, \mathbf{q}) - \hat{V}_{t+1}^{\pi_{\boldsymbol{\theta}}}(\mathbf{x}_{t+1}, \mathbf{q}) \approx r(\mathbf{x}_0, \mathbf{q}) - r(\hat{\mathbf{x}}_{0|t+1}, \mathbf{q}). \tag{12}$$

This estimate uses the reward of a greedy one-step completion as a state-dependent baseline. Although this baseline generally differs from the true value function, that discrepancy alone does not bias the full on-policy score-function gradient: any action-independent baseline, treated as fixed when differentiating, has zero expected contribution. Its quality instead affects the variance of the gradient estimate, and variance reduction is not guaranteed. We combine the terminal reward and the estimated stepwise advantage through the effective per-step coefficient

$$r(\mathbf{x}_0, \mathbf{q}) + \lambda\, \hat{A}_t^{\pi_{\boldsymbol{\theta}}}(\mathbf{x}_{t+1}, \mathbf{x}_0, \mathbf{q}), \tag{13}$$

where $\lambda \geq 0$ controls the contribution of the stepwise term. Writing this coefficient as $(1+\lambda)r - \lambda \hat{V}_{t+1}^{\pi_{\boldsymbol{\theta}}}$, the full on-policy update, before step selection and surrogate optimization, equals $(1+\lambda)\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ in expectation under the conditions of Proposition 1. Thus, up to an overall scale, λ controls the baseline weight. This identity does not establish unbiasedness of the practical update, which additionally uses entropy-guided step selection, group centering, clipping, and KL regularization.

Intuitively, $\hat{A}_t^{\pi_{\boldsymbol{\theta}}}$ measures how much better the full trajectory rollout from $\mathbf{x}_{t+1}$ performed relative to a one-step greedy completion: positive means iterative denoising from $\mathbf{x}_{t+1}$ was crucial; near-zero means the one-step prediction already captures the outcome. This per-step signal is obtained at no extra cost from the rollout forward pass, with no auxiliary value network required. *To the best of our knowledge, we are the first to derive stepwise advantages from this DLM-specific structure in a principled way for RL fine-tuning.*

When is the stepwise signal useful? We expect the stepwise term to help when the reward of a greedy one-step completion from an intermediate state reliably predicts the terminal reward. In tasks such as Sudoku and Countdown, intermediate states progressively fix parts of the solution, making this signal useful. In mathematical reasoning, the final answer may depend on reasoning tokens that remain unresolved, making intermediate greedy completions less reliable. This provides an intuition for the consistent gains of EGSPO-SA on logical reasoning and its mixed results relative to EGSPO on mathematical reasoning.

4.4 From Policy Gradient to GRPO Loss

For G sampled completions $\{\mathbf{x}^j\} \sim \pi_{\boldsymbol{\theta}_{\text{old}}}(\cdot \mid \mathbf{q})$ and each selected step $t \in S$, the effective per-step coefficient from Eq. (13) is instantiated via Eqs. (11) and (12) as $\tilde{A}_t^j = r(\mathbf{x}_0^j, \mathbf{q}) + \lambda \, \hat{A}_t^{\pi_{\boldsymbol{\theta}_{\text{old}}}}(\mathbf{x}^j, \mathbf{q})$. We apply GRPO-style centering across completions for each selected step, $\hat{\tilde{A}}_t^j = \tilde{A}_t^j - \frac{1}{G}\sum_{i=1}^{G} \tilde{A}_t^i$. The training loss is a sum over selected steps, with each term applying PPO-style clipping to its own importance ratio $\rho_t = \pi_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t+1}) / \pi_{\boldsymbol{\theta}_{\text{old}}}(\mathbf{x}_t \mid \mathbf{x}_{t+1})$ plus a KL penalty, consistent with the sum-over-t structure of the policy gradient in Eq. (5).

We refer to this method as *Entropy-Guided Stepwise Policy Optimization with Stepwise Advantages* (**EGSPO-SA**). Setting $\lambda = 0$ recovers entropy-guided step selection with the sequence-level reward alone, denoted **EGSPO**.

Remark 6 (λ across training). In all experiments, λ is fixed within each diffusion episode and does not depend on the denoising step t. Separately, λ may be annealed across training steps for some tasks, analogous to standard hyperparameter scheduling; per-task values are in Appendix F.

Exact gradient and practical approximations. Proposition 1 establishes the exact policy gradient for the diffusion MDP. To obtain a tractable algorithm, we introduce three approximations: (i) we treat the confidence-based position-selection set U_t as fixed when differentiating, as discussed in Remark 3; (ii) we replace the full sum over denoising steps with a subset of K steps selected using the entropy-based upper bound in Lemma 4; and (iii) we estimate the stepwise value functions using rewards of greedy one-step completions. These estimates are then incorporated into a GRPO-style surrogate with group centering, PPO-style clipping, and KL regularization. Thus, Proposition 1 provides the theoretical foundation, while the practical training update incorporates the above approximations and surrogate optimization choices.

5 Experimental Results

5.1 Experimental Setup

Base Models. All experiments use two DLMs with distinct designs: LLaDA-8B-Instruct [Nie et al., 2025], trained from scratch with block-wise inference, and Dream-7B-Instruct [Ye et al., 2025], adapted from a Qwen2.5 ARLM via continued pretraining with non-block-wise sampling. Evaluating on both ensures our method generalizes across diverse diffusion LM architectures. Fine-tuning is performed without SFT on either backbone. Details are in Appendix F.

Datasets. We evaluate EGSPO and EGSPO-SA on a range of benchmark tasks, including mathematical reasoning (GSM8K [Cobbe et al., 2021], MATH500 [Lightman et al., 2023]), logical reasoning

Table 1: Results on reasoning benchmarks for **LLaDA-8B-Instruct.** We report accuracy at different generation lengths and the best performance across them. d2 reports only the best performance; wd1 does not report evaluation at length 128. For Sudoku, † denotes 3-shot evaluation, whereas all other entries are 0-shot. ESPO results are re-evaluated from the publicly released ESPO checkpoints under the uniform protocol shared by all other entries in the table, since ESPO's original protocol uses 1-shot evaluation on Sudoku and decodes 1 token/step on GSM8K and MATH500; their originally published numbers under that protocol are shown in gray for completeness.

	Sudoku				**Countdown**				**GSM8K**				**MATH500**			
Model / Seq. Len.	128	256	512	Best	128	256	512	Best	128	256	512	Best	128	256	512	Best
LLaDA-8B-Instruct	11.7	6.7	5.5	11.7	20.7	19.5	16.0	20.7	68.7	76.7	78.2	78.2	26.0	32.4	36.2	36.2
d1	22.1	16.7	9.5	22.1	34.8	32.0	42.2	42.2	73.2	81.1	82.1	82.1	33.8	38.6	40.2	40.2
wd1	-	76.4	62.8	76.4	-	51.2	46.1	51.2	-	80.8	82.3	82.3	-	34.4	39.0	39.0
SPG	82.9†	94.0†	93.1†	94.0†	68.8	71.5	70.3	71.5	78.5	**86.1**	84.5	**86.1**	33.4	40.0	**41.8**	**41.8**
ESPO	72.9	58.6	54.5	72.9	81.6	82.0	79.3	82.0	76.7	79.9	80.8	80.8	32.0	34.6	39.4	39.4
ESPO	92.7	84.7	80.5	92.7	81.6	82.0	79.3	82.0	80.0	82.3	83.7	83.7	36.0	39.0	43.4	43.3
d2				91.9				56.6				85.0				41.6
EGSPO (Ours)	93.3	93.6	89.1	93.6	82.0	81.6	82.4	82.4	**80.7**	86.0	**86.1**	**86.1**	32.2	37.8	39.0	39.0
EGSPO-SA (Ours)	**93.7**	**94.3**	**93.4**	**94.3**	**84.0**	**85.2**	**83.2**	**85.2**	77.0	84.6	85.0	85.0	**35.0**	**40.4**	41.6	41.6

Table 2: Results on reasoning benchmarks for **Dream-7B-Instruct**. We report accuracy at different generation lengths and the best performance across them. For GSM8K and MATH500, all models are evaluated using 1 token/step decoding to match ESPO's protocol; ESPO is also trained with 1 token/step decoding, whereas our method is trained with 2 token/step decoding. For Sudoku and Countdown, both training and evaluation use 2 token/step decoding for all methods.

Model / Seq Len	**Sudoku**				**Countdown**				**GSM8K**				**MATH500**			
	128	256	512	Best	128	256	512	Best	128	256	512	Best	128	256	512	Best
Dream	9.3	2.1	14.0	14.0	8.5	7.8	17.4	17.4	75.8	81.3	80.7	81.3	38.2	45.7	48.0	48.0
ESPO	71.7	72.3	71.3	72.3	68.8	66.8	64.8	68.8	**79.6**	82.3	82.0	82.3	40.3	47.4	50.3	50.3
EGSPO (ours)	86.4	**90.3**	87.0	90.3	76.6	73.4	72.3	76.6	77.4	82.1	82.3	82.3	**42.4**	**49.2**	**50.4**	**50.4**
EGSPO-SA (ours)	**90.6**	**90.3**	**89.2**	**90.6**	**79.7**	**82.4**	**78.5**	**82.4**	77.6	**84.9**	**86.1**	**86.1**	41.2	47.2	47.8	47.8

(Sudoku and Countdown [Ye et al., 2024]), and coding (MBPP [Austin et al., 2021b], HumanEval [Chen et al., 2021]). Across all benchmarks, we adopt the same terminal reward functions as used in prior work; detailed reward definitions are provided in Appendix C.

Baselines. We evaluate our algorithms against existing RL post-training methods for DLMs on both base models. The training and evaluation protocols of these baselines differ from ours in several aspects, which we detail below.

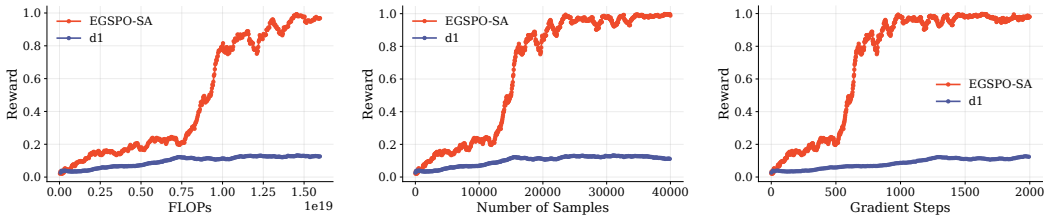
LLaDA experiments: For mathematical and logical reasoning tasks, we compare against d1 [Zhao et al., 2025b], d2 [Wang et al., 2025c], wd1 [Tang et al., 2025], SPG [Wang et al., 2025a], and ESPO [Ou et al., 2026]. For coding tasks, only d1 and ESPO report these results. In the tables, we report the numbers as published in the d1, d2, wd1, and SPG papers, all of which share our training protocol: a generation length of 256, two-token decoding per denoising step, and LoRA with the same rank. This is the standard training setup in the RL for DLM literature. For ESPO, we evaluate results by ourselves using their publicly released checkpoint under this unified protocol, since their reported results use 1 token/step decoding for GSM8K, MATH500, and coding, and 1-shot evaluation for Sudoku. 1 token/step decoding bypasses the joint probability learning that makes multi-token diffusion generation promising and non-trivial, and thus less appealing for pure diffusion-style language modeling. Full details on training and evaluation protocol differences across all baselines are in Appendix E.

Dream experiments: We only compare against ESPO, as no other RL post-training baselines for DLMs report results on this model. Since ESPO was trained and evaluated with 1-token/step decoding for GSM8K and MATH500, we also report our results with 1-token/step evaluation, even though our training used 2-token/step decoding.

5.2 Main Results

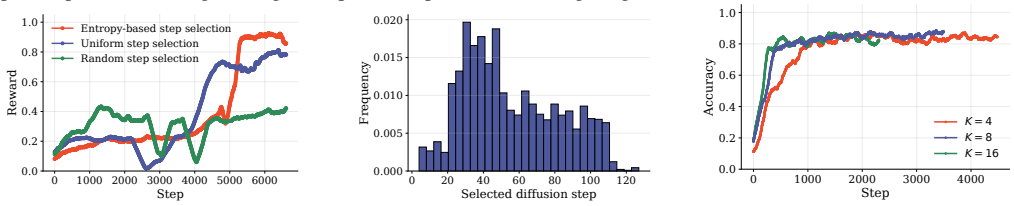
Tables 1, 2, and 3 report performance across all tasks and generation lengths on both models. Training curves are provided in Appendix G, and qualitative examples for each dataset are shown in Appendix J.

Logical reasoning: EGSPO and EGSPO-SA achieve the largest gains on Sudoku and Countdown, consistently across both backbones. EGSPO-SA outperforms EGSPO across all generation lengths on both tasks, with the gap most pronounced on Sudoku. This pattern is consistent with the intuition in Section 4.3: intermediate states progressively fix parts of the solution in logical reasoning tasks,



(a) Reward vs. FLOPs (b) Reward vs. Samples (c) Reward vs. Gradient Steps

Figure 3: Compute efficiency comparison between EGSPO-SA and d1 on Sudoku. **(a)** Total estimated FLOPs across 8 GPUs, including rollout generation, reference-policy evaluation, and policy forward/backward passes. **(b)** Cumulative prompt–completion pairs seen during training. **(c)** Optimizer updates (accounting for gradient accumulation).



(a) (b) (c)

Figure 4: Ablation studies analyzing (a) entropy-guided step selection (EGSPO) versus uniform and random step selection, (b) the distribution of selected denoising steps, and (c) performance as a function of the step budget K.

making greedy one-step completions a useful reference for evaluating the final rollout reward. On LLaDA, EGSPO-SA achieves 94.3% on Sudoku (vs. 94.0% for SPG under the easier 3-shot setting, and 72.9% for ESPO) and 85.2% on Countdown (vs. 82.0% for ESPO and 71.5% for SPG), both under 0-shot evaluation. On Dream, EGSPO-SA reaches 90.6% on Sudoku and 82.4% on Countdown, surpassing ESPO by 18 and 14 percentage points respectively.

Mathematical reasoning: On GSM8K and MATH500, our algorithms match or exceed existing approaches. On LLaDA, EGSPO achieves 86.1% on GSM8K, matching SPG (86.1%), and EGSPO-SA reaches 41.6% on MATH500, matching d2 and within 0.2% of SPG (41.8%). The gap between EGSPO and EGSPO-SA is small on both tasks, which is consistent with the observation from Section 4.3 that the sequence-level reward already provides a sufficient learning signal when task structure does not tightly couple denoising steps. On a few GSM8K configurations, EGSPO outperforms EGSPO-SA. One possible explanation is that greedy completions from intermediate reasoning states provide a less reliable reference for the final reward, so incorporating the stepwise term does not consistently improve the practical training update. On Dream, EGSPO-SA achieves 86.1% on GSM8K, outperforming ESPO (82.3%).

Coding: Table 3 show results on HumanEval and MBPP for LLaDA-8B-Instruct. Both EGSPO and EGSPO-SA substantially outperform d1 and ESPO under the matched 2 token/step decoding protocol. EGSPO-SA achieves 51.1% on MBPP (vs. 44.7% for d1 and 49.0% for ESPO) and 44.5% on HumanEval (vs. 37.8% for d1 and 36.0% for ESPO), demonstrating that entropy-guided step selection is effective for program synthesis, where token-level uncertainty varies substantially across the denoising trajectory.

Table 3: Results on coding benchmarks for **LLaDA-8B-Instruct.** We report accuracy at different generation lengths and the best performance across them. ESPO uses full fine-tuning, whereas d1, EGSPO, and EGSPO-SA use LoRA-based fine-tuning. ESPO results are re-evaluated from the publicly released ESPO checkpoints under the uniform protocol shared by all other entries in the table, since ESPO's original protocol decodes 1 token/step on coding tasks; their originally published numbers under that protocol are shown in gray for completeness.

	HumanEval				MBPP			
Model	128	256	512	Best	128	256	512	Best
LLaDA	27.4	35.3	37.8	37.8	36.2	41.2	40.4	41.2
d1	31.1	32.9	37.8	37.8	40.5	44.7	42.8	44.7
ESPO	25.0	31.7	36.0	36.0	49.0	48.6	49.0	49.0
ESPO	28.1	42.1	50.0	50.0	47.4	44.6	44.2	47.4
EGSPO	**32.3**	40.2	39.6	40.2	49.9	**50.6**	**50.3**	50.6
EGSPO-SA	**32.3**	**41.5**	**44.5**	**44.5**	**51.1**	48.7	49.2	**51.1**

5.3 Ablations and Additional Experiments

Compute Efficiency: Figure 3 compares EGSPO-SA and d1 on Sudoku along three compute axes (FLOPs, prompt–completion pairs, and optimizer updates) using LLaDA-8B-Instruct as the base model. Across all three, EGSPO-SA converges to near-perfect reward while d1 plateaus below 0.2, showing that EGSPO-SA is more efficient in compute, data, and optimization. FLOPs are estimated using $2NL$ per forward pass and $6NL$ per forward-and-backward pass, where N is the model parameter count and L is the input sequence length. Old-policy log-probabilities are

cached during generation and incur no additional forward passes. An analytical cost comparison with additional baselines is provided in Appendix D.

Effect of Entropy-Based Step Selection: Under a fixed budget of K denoising steps per trajectory, we compare *uniform* (evenly spaced), *random* (randomly sampled), and *entropy-based* (top-K by average unmasking token entropy) selection. For a fixed K, all three strategies use the same number of rollout, reference-policy, and policy forward/backward passes; they differ only in which denoising steps are selected. Entropy scores are obtained from the rollout predictions without additional model forward passes. As shown in Figure 4a, EGSPO achieves higher final performance and converges more rapidly under matched compute, as it concentrates updates on high-uncertainty timesteps, consistent with the analysis in Section 4.2. Uniform consistently outperforms random as training progresses, indicating that covering the trajectory provides more stable updates than stochastic sampling.

Distribution of Selected Denoising Steps: Figure 4b shows the distribution of timesteps chosen by entropy-based selection. Contrary to the naive expectation that early, heavily masked steps would dominate, selected steps span the diffusion horizon with a modest concentration on intermediate timesteps. In early steps, the model is confident across the unmasked-token choices; intermediate steps are where partial structure has formed but multiple completions remain plausible, yielding the highest token-level uncertainty; late steps are largely deterministic and rarely selected. Entropy-based selection thus targets timesteps where the model is genuinely uncertain about token assignments, not simply those that are more masked.

Effect of Step Budget K: We compare step budgets under a fixed wall-clock budget of 48 hours on 8 NVIDIA H200 GPUs (384 GPU-hours per run), including rollout generation, reference-policy evaluation, and policy optimization. Increasing K requires gradient computation at more denoising steps per batch, reducing the number of optimizer updates completed within this budget. This gives approximately 4400, 3600, and 2400 total gradient steps for $K = 4, 8, 16$ respectively. Table 4 shows that $K = 8$ achieves the best GSM8K accuracy across all generation lengths: too few gradient terms ($K = 4$) gives insufficient coverage of the denoising trajectory, while too few total steps ($K = 16$) under-optimizes the policy. Entropy-guided selection concentrates the K budget on the most informative steps, making this tradeoff favorable at moderate K; training curves are shown in Figure 4c.

Table 4: GSM8K accuracy (EGSPO, LLaDA) for different step budgets K and generation length

K	Approx. Grad. Steps	128	256	512
4	4400	76.5	83.4	83.6
8	3600	**80.7**	**86.0**	**86.1**
16	2400	78.2	83.4	83.6

Advantage weight λ. We additionally ablate the choice of constant vs. step-varying λ in Appendix H, finding that a λ fixed within each episode (no denoising-step dependence) matches a diffusion time-step dependent weight λ_t, which justifies this simpler within-episode choice. We select K, the initial value of λ, and whether to anneal λ across training using training reward curves: candidate configurations are trained for 1000–2000 steps, and the selected configuration is then trained to completion. Per-task settings are listed in Appendix F. We retain the training and test splits used by d1, without introducing a validation split. Test-set performance is not used for hyperparameter selection.

6 Conclusion

We introduced a principled RL framework for diffusion language models that makes denoising structure explicit. By formulating masked diffusion generation as a finite-horizon MDP, we derived the policy gradient that decomposes along the diffusion trajectory and yields a well-defined notion of stepwise advantages, without evaluating intractable sequence-level likelihoods. We proposed entropy-guided step selection for adaptive compute allocation and lightweight one-step completions for intermediate advantage estimation, enabling efficient stepwise policy optimization with theoretical guarantees. Empirical results demonstrate consistent improvements over existing RL post-training methods for DLMs across coding, logical, and mathematical reasoning benchmarks.

An interesting future work is to extend our algorithms to self-correction and remasking-capable DLMs, where tokens may be revised after being decoded. This requires revisiting both the action space and the policy gradient derivation under non-monotonic trajectories.

Acknowledgements and Disclosures

Portions of this research were conducted using advanced computing resources provided by Texas A&M High Performance Research Computing. This work was supported in part by the National Science Foundation (NSF) under grants CNS-2148354 and FuSe2-2425399, including support under CNS-2148354 from NSF, federal agencies, and industry partners through the Resilient & Intelligent NextG Systems (RINGS) program; and by the U.S. Army Combat Capabilities Development Command (DEVCOM) under grant W911NF2520046. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsoring agencies or industry partners.

References

Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 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.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL `https://arxiv.org/abs/2107.03374`.

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.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. DPOK: Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

Chunsan Hong, Seonho An, Min-Soo Kim, and Jong Chul Ye. Improving discrete diffusion unmasking policies beyond explicit reference policies. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=on6cb46OhD`.

Shufan Li, Konstantinos Kallidromitis, Hritik Bansal, Akash Gokul, Yusuke Kato, Kazuki Kozuka, Jason Kuen, Zhe Lin, Kai-Wei Chang, and Aditya Grover. LaViDa: A Large Diffusion Model for Vision-Language Understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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 *International Conference on Learning Representations (ICLR)*, 2023.

Xiaoran Liu, Yuerong Song, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. LongLLaDA: Unlocking Long Context Capabilities in Diffusion LLMs. *arXiv preprint arXiv:2506.14429*, 2025.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=S5YeC9llIL`.

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

Subham Sekhar Sahoo, Zhihan Yang, Yash Akhauri, Johnna Liu, Deepansha Singh, Zhoujun Cheng, Zhengzhong Liu, Eric Xing, John Thickstun, and Arash Vahdat. Esoteric language models. *arXiv preprint arXiv:2506.01928*, 2025.

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.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in Neural Information Processing Systems (NeurIPS)*, 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.

Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M Tseng, Tommaso Biancalani, and Sergey Levine. Fine-tuning of continuous-time diffusion models as entropy-regularized control. *arXiv preprint arXiv:2402.15194*, 2024.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Chenyu Wang, Paria Rashidinejad, DiJia Su, Song Jiang, Sid Wang, Siyan Zhao, Cai Zhou, Shannon Zejiang Shen, Feiyu Chen, Tommi Jaakkola, et al. SPG: Sandwiched policy gradient for masked diffusion language models. *arXiv preprint arXiv:2510.09541*, 2025a.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

Guanghan Wang, Yair Schiff, Gilad Turok, and Volodymyr Kuleshov. d2: Improved techniques for training reasoning diffusion language models. *arXiv preprint arXiv:2509.21474*, 2025c.

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*, 2025d.

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*, 2025.

Shaoan Xie, Lingjing Kong, Xiangchen Song, Xinshuai Dong, Guangyi Chen, Eric P. Xing, and Kun Zhang. Advancing reasoning in diffusion language models with denoising process rewards. *arXiv preprint arXiv:2510.01544*, 2025.

Kai Yang, Jian Tao, Jiafei Lyu, Chunjiang Ge, Jiaxin Chen, Weihan Shen, Xiaolong Zhu, and Xiu Li. Using human feedback to fine-tune diffusion models without any reward model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*, 2025.

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

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.

Hanyang Zhao, Dawen Liang, Wenpin Tang, David Yao, and Nathan Kallus. DiFFPO: Training diffusion llms to reason fast and furious via reinforcement learning. *arXiv preprint arXiv:2510.02212*, 2025a.

Lingxiao Zhao, Xueying Ding, Lijun Yu, and Leman Akoglu. Improving and unifying discrete & continuous-time discrete denoising diffusion. *CoRR*, 2024.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction describe three main contributions: (i) a policy gradient for the diffusion MDP that decomposes over denoising steps with stepwise advantages; (ii) tractable estimators using entropy-guided step selection and one-step advantage estimation; and (iii) empirical improvements on coding, logical reasoning, and mathematical reasoning benchmarks. The policy gradient is established in Proposition 1 and proved in Appendix A. The practical algorithms are described in Sections 4.2–4.4 and Algorithm 1, with the distinction between the exact theoretical gradient and practical approximations clarified at the end of Section 4. Empirical results are reported in Tables 1 and 3.

2. **Limitations**

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

   Answer: [Yes]

   Justification: The limitation is included in Appendix K.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: All theoretical results (Proposition 1, Lemma 4, and Proposition 7) are proved in full in Appendix A, with assumptions explicitly stated in each result and all results numbered and cross-referenced in the main text.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The algorithm is fully described in Algorithm 1. Hyperparameters, optimizer settings, LoRA configuration, hardware, batch size, generation length, denoising steps, and task-specific reward functions are all detailed in Appendices C, E, F. The base model (LLaDA-8B-Instruct) and all datasets used are publicly available. Code is available at the anonymous repo linked in the abstract.

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

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

   Answer: [Yes]

   Justification: Code is available at the anonymous repo linked in the abstract. All datasets used are publicly available.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: All training and evaluation details are provided in Appendix F, including optimizer (Adam), learning rate, LoRA configuration, batch size, hardware, generation lengths, denoising steps, step budget K, temperature, and checkpoint selection criteria. Reward functions and dataset splits are detailed in Appendix C.

7. **Experiment statistical significance**

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

Answer: [No]

Justification: The paper does not report error bars or confidence intervals. Results are reported as single-run accuracies across multiple generation lengths, following the evaluation protocol of prior work in this area (d1, wd1, SPG, d2), which also do not report error bars.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendix F specifies that training is conducted on 8 NVIDIA H200 GPUs. Training durations are provided per task (4k steps for coding, 6k for mathematical reasoning, 10k for logical reasoning).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics. The work involves no human subjects or sensitive personal data and evaluates RL post-training methods on reasoning and coding benchmarks.

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: This work presents a foundational RL training algorithm for diffusion language models, evaluated on reasoning and coding benchmarks. It is not tied to any specific application or deployment, and there is no direct path to negative societal impacts beyond those inherent to language model research in general.

11. **Safeguards**

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

    Answer: [Yes]

    Justification: The paper fine-tunes an existing publicly available model (LLaDA-8B-Instruct) on reasoning tasks and poses no significant misuse risks beyond those of the base model itself.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All datasets and the base model (LLaDA-8B-Instruct) are properly cited with references to their original papers. All assets used are publicly available.

13. **New assets**

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

    Answer: [Yes]

    Justification: The code released in the github repo documented alongside the paper. Training details, hyperparameters, and implementation specifics are provided in Appendices C, E, F, and the algorithm is fully described in Algorithm 1.

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLM usage is described in Appendix L.

A Theoretical Results

Proposition 1 (Recalled). For a policy $\pi_{\boldsymbol{\theta}}$ and the objective $J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{q},\mathbf{x}\sim\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{q})}[r(\mathbf{x}_0,\mathbf{q})]$, the policy gradient is given by (5).

Proof. Writing $\mathbf{x} = \mathbf{x}_{0:T-1}$, we can evaluate directly

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{q}} \left[\sum_{\mathbf{x}} r(\mathbf{x}_0,\mathbf{q}) \pi_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{q}) \right] = \mathbb{E}_{\mathbf{q}} \left[\sum_{\mathbf{x}} r(\mathbf{x}_0,\mathbf{q}) \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{q}) \right]$$

$$= \mathbb{E}_{\mathbf{q}} \left[\sum_{\mathbf{x}} r(\mathbf{x}_0,\mathbf{q}) \pi_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{q}) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{q}) \right] = \mathbb{E}_{\mathbf{q},\mathbf{x}\sim\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{q})} \left[r(\mathbf{x}_0,\mathbf{q}) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{q}) \right].$$

By definition, since $\pi_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{q}) = \pi_{\boldsymbol{\theta}}(\mathbf{x}_{0:T-1} \mid \mathbf{q}) = \prod_{t=0}^{T-1} \pi_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t+1}, \mathbf{q})$, we have $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} J_t(\boldsymbol{\theta})$, where the per step policy gradient is defined as $\nabla_{\boldsymbol{\theta}} J_t(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{q},\mathbf{x}\sim\pi_{\boldsymbol{\theta}}}[r(\mathbf{x}_0,\mathbf{q}) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t+1}, \mathbf{q})]$. The term inside the expectation is a random variable that depends on $\mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{x}_0, \mathbf{q}$. For any action $\mathbf{x}_t$ independent baseline, in particular, the random variable $V_{t+1}^{\pi_{\boldsymbol{\theta}}}(\mathbf{x}_{t+1})$, we have

$$\mathbb{E}_{\mathbf{x}_t\sim\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x}_{t+1})}[V_{t+1}^{\pi_{\boldsymbol{\theta}}}(\mathbf{x}_{t+1}) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t+1})] = V_{t+1}^{\pi_{\boldsymbol{\theta}}}(\mathbf{x}_{t+1}) \mathbb{E}_{\mathbf{x}_t\sim\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x}_{t+1})}[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t+1})]$$

$$= V_{t+1}^{\pi_{\boldsymbol{\theta}}}(\mathbf{x}_{t+1}) \sum_{\mathbf{x}_t} \pi_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t+1}) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t+1}) = V_{t+1}^{\pi_{\boldsymbol{\theta}}}(\mathbf{x}_{t+1}) \sum_{\mathbf{x}_t} \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t+1})$$

$$= V_{t+1}^{\pi_{\boldsymbol{\theta}}}(\mathbf{x}_{t+1}) \nabla_{\boldsymbol{\theta}} \left(\sum_{\mathbf{x}_t} \pi_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t+1}) \right) = V_{t+1}^{\pi_{\boldsymbol{\theta}}}(\mathbf{x}_{t+1}) \nabla_{\boldsymbol{\theta}}(1) = 0.$$

Therefore, $A_t^{\pi_{\boldsymbol{\theta}}} = r(\mathbf{x}_0,\mathbf{q}) - V_{t+1}^{\pi_{\boldsymbol{\theta}}}(\mathbf{x}_{t+1},\mathbf{q})$ satisfies $\nabla_{\boldsymbol{\theta}} J_t(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{q},\mathbf{x}\sim\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{q})}[A_t^{\pi_{\boldsymbol{\theta}}} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t+1})]$.

□

Proposition 7 (Entropy Bound)**.** *Let $\pi_{\boldsymbol{\theta}}$ be a softmax policy with logits $g_{\boldsymbol{\theta}}^i$ for i over a finite set, i.e., $\pi_{\boldsymbol{\theta},i} \triangleq \frac{\exp(g_{\boldsymbol{\theta},i})}{\sum_{i'} \exp(g_{\boldsymbol{\theta}}^{i'})}$. Let A_i be constants such that $\|A\|_\infty \triangleq \max_i |A_i|$ is finite. Let $J(\boldsymbol{\theta}) \triangleq \mathbb{E}_{i\sim\pi_{\boldsymbol{\theta}}}[A_i]$ and suppose $\|\nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta}}\| \triangleq \max_{i,j} \|\nabla_{\boldsymbol{\theta}}(g_{\boldsymbol{\theta},i} - g_{\boldsymbol{\theta},j})\|$ is bounded. Then, $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{i\sim\pi_{\boldsymbol{\theta}}}[A_i \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta},i}]$ satisfies*

$$\|\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})\| \leq H(\pi_{\boldsymbol{\theta}}) \|A\|_\infty \|\nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta}}\|, \tag{14}$$

where $H(\pi_{\boldsymbol{\theta}}) \triangleq \mathbb{E}_{i\sim\pi_{\boldsymbol{\theta}}}[-\log \pi_{\boldsymbol{\theta},i}]$ is the entropy of $\pi_{\boldsymbol{\theta}}$.

Proof. Starting with the gradient of the objective J, we have $\|\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})\| \leq \mathbb{E}_{i\sim\pi_{\boldsymbol{\theta}}}[|A_i| \|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta},i}\|] \leq \|A\|_\infty \mathbb{E}_{i\sim\pi_{\boldsymbol{\theta}}}[\|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta},i}\|]$. Now, for any i, using chain-rule and the derivative of softmax policy, we have $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta},i} = (1 - \pi_{\boldsymbol{\theta},i}) \nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta},i} - \sum_{j\neq i} \pi_{\boldsymbol{\theta},j} \nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta},j} = \sum_{j\neq i} \pi_{\boldsymbol{\theta},j} (\nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta},i} - \nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta},j})$. Therefore, by triangle inequality, we obtain

$$\|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta},i}\| \leq \sum_{j\neq i} \pi_{\boldsymbol{\theta},j} \|\nabla_{\boldsymbol{\theta}}(g_{\boldsymbol{\theta},i} - g_{\boldsymbol{\theta},j})\| \leq \left(\sum_{j\neq i} \pi_{\boldsymbol{\theta}^j} \right) \left(\max_{i,j} \|\nabla_{\boldsymbol{\theta}}(g_{\boldsymbol{\theta},i} - g_{\boldsymbol{\theta},j}))\| \right) \tag{15}$$

$$= (1 - \pi_{\boldsymbol{\theta},i}) \|\nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta}}\| \leq (-\log \pi_{\boldsymbol{\theta},i}) \|\nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta}}\|, \tag{16}$$

where the last step follows from $1 - x \leq -\log x$ for $x \in (0,1)$. Using the above bound in the first inequality gives the result. □

Proof of Lemma 4. By Proposition 7, we obtain the following bound on the error in terms of the per-step entropy:

$$\Delta_S = \|\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} J_S(\boldsymbol{\theta})\| = \|\sum_{t\notin S} \nabla_{\boldsymbol{\theta}} J_t(\boldsymbol{\theta})\| \leq \sum_{t\notin S} \|\nabla_{\boldsymbol{\theta}} J_t(\boldsymbol{\theta})\|$$

$$\leq \sum_{t\notin S} H(\pi_{\boldsymbol{\theta}}{}^{t|t+1}) \|A_t^{\pi_{\boldsymbol{\theta}}}\|_\infty \|\nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta}}^{t|t+1}\|,$$

where $g_{\boldsymbol{\theta}}$ are the logits from which the policy $\pi_{\boldsymbol{\theta}}$ is constructed by taking softmax. Bounding $\|A_t^{\pi_{\boldsymbol{\theta}}}\|_\infty \leq \max_{t'} |A_{t'}^{\pi_{\boldsymbol{\theta}}}|$ and $\|\nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta}}^{t|t+1}\| \leq B$ uniformly over t, we obtain $\Delta_S \leq B(\max_t |A_t^{\pi_{\boldsymbol{\theta}}}|)\sum_{t\notin S} H({\pi_{\boldsymbol{\theta}}}^{t|t+1})$. □

B Diagnostic Evaluation of Entropy-Guided Step Selection

To assess whether entropy-guided selection preserves the direction of the full denoising-step update, we compute gradients at all $T = 128$ steps on a frozen LLaDA-8B-Instruct model, using 128 completions per task. We compare the cosine similarity between the update formed from K selected steps and the full T-step update. Computing all step gradients is used only for this diagnostic and is not required during training.

Table 5: Cosine similarity between selected-step and full-step updates. Entropy-guided selection gives higher alignment than uniform and random selection across all evaluated tasks and budgets.

Task	K	Entropy	Uniform	Random
Sudoku	4	0.466	0.182	0.194
	8	0.603	0.258	0.276
Countdown	4	0.447	0.187	0.202
	8	0.583	0.288	0.298
GSM8K	4	0.576	0.120	0.150
	8	0.720	0.253	0.237
MATH500	4	0.510	0.181	0.169
	8	0.654	0.252	0.251

Table 5 provides empirical support for entropy-guided selection: its selected-step updates align more closely with the full-step updates than those obtained by uniform or random selection. These results complement the upper-bound analysis in Lemma 4.

C Datasets and Reward Functions

We largely follow the experimental protocol of d1 [Zhao et al., 2025b], adopting the same reward formulations and train–test splits to ensure comparability.

C.1 GSM8K.

We use the training split of the GSM8K dataset for reinforcement learning and evaluate on the official test split. Rewards follow the Unsloth-style formulation and consist of five additive components:

- an XML structure reward that assigns $+0.125$ for each correctly placed formatting tag, with small penalties for extraneous content appearing after closing tags;
- a soft format reward of $+0.5$ for outputs matching the pattern `<reasoning>...</reasoning><answer>...</answer>`
- a strict format reward of $+0.5$ for exact adherence to the expected structure, including correct line breaks;
- an integer-answer reward of $+0.5$ if the predicted answer is a valid integer;
- a correctness reward of $+2.0$ when the predicted answer matches the ground-truth solution.

C.2 MATH500.

For MATH500, we train on the training split and evaluate on the test split. The reward function consists of formatting and correctness components:

- a format reward of 1.0 if `<answer></answer>` tags are present and the answer is enclosed in a `\boxed` expression;

Algorithm 1 Entropy-Guided Stepwise Policy Optimization with Stepwise Advantages (EGSPO-SA)

Require: Policy $\pi_{\boldsymbol{\theta}}$, reference policy π_{ref}, reward r, prompt distribution $\mathcal{D}$, denoising steps T, step budget K, completions G, advantage weight λ, KL weight β

Initialize $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{\text{ref}}$
while not converged **do**
  Sample prompt $\mathbf{q} \sim \mathcal{D}$
  Sample G diffusion trajectories $\mathbf{x}^j_{0:T} \sim \pi_{\boldsymbol{\theta}_{\text{old}}}(\cdot \mid \mathbf{q})$ for $j = 1, \ldots, G$
  Compute per-step entropies H^j_t of $\pi_{\boldsymbol{\theta}_{\text{old}}}(\cdot \mid \mathbf{x}^j_{t+1})$
  Select step subset $S^j \leftarrow \text{top}[K](t : H^j_t)$
  Compute reference-policy probabilities $\pi_{\text{ref}}(\mathbf{x}^j_t \mid \mathbf{x}^j_{t+1})$ for $j = 1, \ldots, G$ and $t \in S^j$.
  Greedily complete trajectory from $\mathbf{x}^j_{t+1}$: $\hat{\mathbf{x}}^j_{0|t+1} \leftarrow \text{GreedyComplete}(\mathbf{x}^j_{t+1})$ (as per Eq. (11))
  Estimate per-step coefficients per completion: $A^j_t \leftarrow r(\mathbf{x}^{(j)}_0, \mathbf{q}) + \lambda(r(\mathbf{x}^{(j)}_0, \mathbf{q}) - r(\hat{\mathbf{x}}^j_{0|t+1}, \mathbf{q}))$.
  Compute centered stepwise advantages: $\bar{A}^j_t \leftarrow A^j_t - \frac{1}{G}\sum_{i=1}^{G} A^i_t$
  Compute importance ratios $\rho^j_t \leftarrow \pi_{\boldsymbol{\theta}}(\mathbf{x}^j_t \mid \mathbf{x}^j_{t+1}) / \pi_{\boldsymbol{\theta}_{\text{old}}}(\mathbf{x}^j_t \mid x^j_{t+1})$
  Estimate KL term using $\pi_{\boldsymbol{\theta}}$ and π_{ref} likelihoods using trajectories from $\pi_{\boldsymbol{\theta}_{\text{old}}}$
  Update $\pi_{\boldsymbol{\theta}}$ using GRPO with stepwise advantages $\bar{A}^j_t$ and KL regularization weighted by β
end while
Return $\pi_{\boldsymbol{\theta}}$

- a format reward of 0.75 if `<answer></answer>` tags are present without a boxed expression;
- a format reward of 0.5 if a boxed expression is present without answer tags;
- a format reward of 0.25 if neither answer tags nor a boxed expression are present;
- a correctness reward of $+2.0$ when the boxed answer exactly matches the ground-truth solution.

C.3 Sudoku.

For the 4×4 Sudoku task, we use a publicly available dataset of one million synthetically generated puzzles.[2] The dataset was generated using the Arel solver. For evaluation, we randomly sample 256 puzzles using the same generator. The reward is computed as the fraction of correctly filled cells among positions that were initially empty in the input puzzle, focusing evaluation on reasoning performance rather than copying pre-filled values.

C.4 Coding.

For coding experiments, we train on the KodCode-Light-RL-10k dataset. The reward function consists of three components. First, an XML structure reward identical to GSM8K is used, with an additional $+0.5$ bonus when the generated program is enclosed within answer tags; outputs not wrapped in `python` code blocks receive zero structural reward. Second, a correctness score is computed using unit tests, where the reward corresponds to the fraction of tests passed rather than a binary success signal. Finally, a safety constraint assigns a reward of 0 if the generated code imports restricted modules, including `os`, `sys`, `shutil`, `subprocess`, `socket`, `psutil`, `ctypes`, `pathlib`, `builtins`, or `import`.

D Analytical Compute Comparison

We complement the measured comparison in Figure 3 with an analytical accounting of training cost. We express costs in forward-pass equivalents: one model forward pass costs one unit, and a forward-and-backward pass costs approximately three units. At a response length of 256 with two tokens decoded per step, generation requires 128 forward passes. Table 6 reports the additional

[2] `https://github.com/Black-Phoenix/4x4-Sudoku-Dataset`

update costs under the listed configurations, including reference- and old-policy evaluations where required.

Table 6: Analytical cost per generated answer in forward-pass equivalents. Totals include 128 generation passes. Here μ is the number of update iterations, m and M are Monte Carlo sample counts, and K is our selected-step budget.

Method	Settings	Grad. passes	Update cost	Total
SPG	$m = 2,\ \mu = 4$	8	$8 \times 3 = 24$	152
Ours	$K = 8,\ \mu = 1$	8	$8 \times 3 + 8 = 32$	160
wd1	$\mu = 12$	12	$12 \times 3 = 36$	164
d1	$\mu = 12$	12	$12 \times 3 + 2 = 38$	166
ESPO	$M = 2,\ \mu = 8$	32	$32 \times 3 = 96$	224

Generation accounts for most of the total cost for SPG, our method, wd1, and d1. Their total costs are therefore similar despite differences in update allocation. Our method uses eight selected trajectory steps and eight reference-policy forward passes, while old-policy log-probabilities are cached during generation. ESPO's published single-token decoding configuration for mathematical reasoning and coding instead requires 256 generation passes, increasing its total to 352 forward-pass equivalents. These estimates describe computational cost under the listed configurations, rather than measured wall-clock performance.

E Details on Baselines

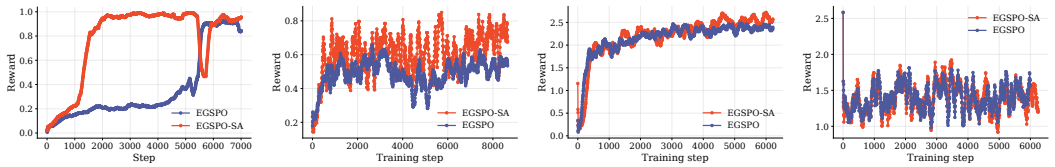
For LLaDA-8B-Instruct experiments, our method along with d1, wd1, and SPG fine-tune LLaDA-8B-Instruct at a fixed generation length of 256 with two tokens decoded in parallel per denoising step, using LoRA with the same rank. The remaining baselines deviate from this protocol as detailed below.

- **d2** trains and evaluates with task-specific generation lengths (e.g., 128 for Sudoku and Countdown, 512 for MATH500), so its reported numbers are not produced under a uniform decoding configuration across tasks.
- **ESPO** uses single-token decoding on GSM8K, MATH500, and coding tasks, and two-token decoding on Sudoku and Countdown, for both training and evaluation. It additionally uses full fine-tuning rather than LoRA on coding tasks, and reports a 1-shot result on Sudoku. To place ESPO on equal footing with the rest of our table, we take the publicly released ESPO checkpoints and re-evaluate them under our uniform protocol (two tokens decoded per step, and 0-shot evaluation) on all reasoning and coding benchmarks.
- **SPG** reports a 3-shot result on Sudoku, while all other entries in our table, including ours, are 0-shot. SPG's 3-shot number is retained as published, since the authors did not release their checkpoints.

F Hyperparameter Settings and Implementation Details

We follow prior diffusion-RL work for most hyperparameter choices. Low-Rank Adaptation (LoRA) is applied with rank $r = 128$ and scaling factor $\alpha = 64$. Training is conducted on 8 NVIDIA H200 GPUs. Optimization is performed using Adam with $\beta_1 = 0.9$, $\beta_2 = 0.99$, weight decay of 0.1, and a learning rate of 3×10^{-5}. Task-specific settings for batch size, number of completions per prompt, step budget, gradient accumulation, and the advantage weight λ are listed in Tables 7 and 8.

For RL rollout, we use a generation length of 256 tokens with 128 diffusion steps during training. We adopt block-wise generation with a block size of 32, and two tokens are denoised in parallel at each step for all tasks. For each prompt, we generate G completions, and for each completion we select the top K denoising steps with the highest entropy for policy updates. We use $K = 8$ for all tasks except coding, where $K = 16$: coding tasks have longer effective horizons (more tokens must be placed correctly in syntactically valid positions), so broader trajectory coverage from a larger K outweighs the reduction in total gradient steps under the fixed compute budget. We use a sampling temperature of 0.9 during training for all tasks except Countdown, where a temperature of 0.2 is used. At inference time, we apply greedy decoding for all tasks.



(a) Sudoku (b) Countdown (c) GSM8K (d) MATH500

Figure 5: Training curves for EGSPO and EGSPO-SA on Sudoku, Countdown, GSM8K, and MATH500 using the training settings described in Section F.

At evaluation time, we report results using generation lengths of 128, 256, and 512 tokens. Models are evaluated once the reward curves stabilize: coding tasks are trained for 4k steps, mathematical reasoning tasks for around 6k steps, and logical reasoning tasks for 10k steps. For all tasks, we select the checkpoint with the highest average performance across the evaluated generation lengths.

The advantage weight λ varies by task and backbone; the settings are listed in Tables 7 and 8.

For Sudoku and GSM8K on LLaDA, we use a constant λ. For Countdown, MATH500, and most Dream tasks, we find that initializing λ at the values listed in Tables 7–8 and halving it every 500 training steps further improves performance, analogous to standard hyperparameter scheduling in deep RL (e.g., KL coefficient annealing).

Table 7: Task-specific hyperparameters for **LLaDA-8B-Instruct**. B: batch size (prompt-completion pairs per update), G: completions per prompt, K: step budget.

Task	B	G	K	Grad. Accum.	λ
Sudoku	48	8	8	2	1 (constant)
Countdown	96	8	8	2	1 (halved every 500 steps)
GSM8K	48	8	8	8	0.2 (constant)
MATH500	96	8	8	8	1 (halved every 500 steps)
Coding	96	8	16	4	1 (halved every 500 steps)

Table 8: Task-specific hyperparameters for **Dream-7B-Instruct**. B: batch size (prompt-completion pairs per update), G: completions per prompt, K: step budget.

Task	B	G	K	Grad. Accum.	λ
Sudoku	192	16	8	2	1 (constant)
Countdown	192	16	8	2	0.2 (halved every 500 steps)
GSM8K	192	16	8	2	0.5 (halved every 500 steps)
MATH500	192	16	8	2	0.5 (halved every 500 steps)

G Training Dynamics

Figure 5 shows the training reward curves for EGSPO and EGSPO-SA across four LLaDA tasks. Convergence behavior is similar on the mathematical reasoning benchmarks (GSM8K and MATH500), where both methods reach comparable final reward at similar rates. On the logical reasoning benchmarks (Sudoku and Countdown), EGSPO-SA exhibits earlier learning onset and reaches a higher asymptotic reward, with the gap most pronounced on Sudoku. This pattern is consistent with the main results in Section 5.2.

H Ablation on the Advantage Weight λ

The main paper uses a constant λ within a gradient step while training. One might wonder whether a *diffusion time-step* dependent weight λ_t would help, motivated by the intuition that the one-step value approximation $\hat{V}_{t+1}^{\pi_\theta}$ is a better proxy for $V_{t+1}^{\pi_\theta}$ when t is small (most tokens already unmasked) and a coarser one when t is large (many tokens still masked). A natural schedule reflecting this is $\lambda_t = 1 - t/T$, which equals 0 at the start of generation ($t = T$) and 1 at the end ($t = 0$).

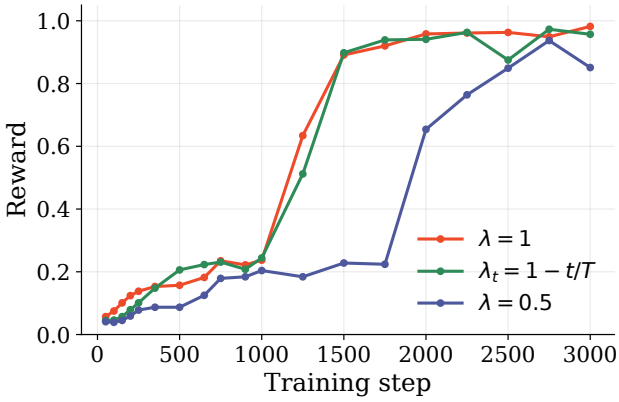


Figure 6: Ablation on the advantage weight λ on Sudoku. Constant $\lambda = 1$ and the step-varying schedule $\lambda_t = 1 - t/T$ converge to similar accuracy; constant $\lambda = 0.5$ is noticeably slower. The magnitude of λ matters more than its functional form across denoising steps.

We ran an ablation on Sudoku comparing three settings: constant $\lambda = 1$, constant $\lambda = 0.5$, and the step-varying schedule $\lambda_t = 1 - t/T$. As shown in Figure 6, the constant $\lambda = 1$ and the step-varying schedule perform similarly throughout training and converge to comparable final accuracy. By contrast, $\lambda = 0.5$ converges more slowly and to a lower asymptote, indicating that the magnitude of λ matters more than its step-dependence. Fixing λ within each episode (no denoising-step dependence) is therefore the practical choice, and we did not find evidence that step-varying schedules λ_t provide a consistent benefit. The initial value of λ and whether to apply an across-training annealing schedule are treated as task-level hyperparameters, as detailed in Table 7.

I Algorithms

Algorithm 1 summarizes the proposed entropy-guided reinforcement learning procedure for diffusion language models. EGSPO-SA operates directly on the denoising process by treating each denoising step as an action and optimizing the policy using a GRPO-style objective.

The method applies entropy-based step selection to identify a subset of informative denoising timesteps for policy updates, avoiding unnecessary optimization at low-entropy, near-deterministic steps. In addition, EGSPO-SA incorporates stepwise advantages. For each selected denoising step, the partial trajectory is greedily completed to estimate the remaining reward, which is used as a baseline for credit assignment. This yields denser and more precise training signals along the denoising trajectory.

EGSPO corresponds to a special case of EGSPO-SA obtained by setting the advantage weight $\lambda = 0$, in which a single sequence-level reward is broadcast uniformly across the selected denoising steps.

J Qualitative Examples

We provide qualitative examples comparing EGSPO and EGSPO-SA with d1 on representative tasks to illustrate differences in reasoning behavior and outputs.

Math Example

Problem. Suppose that I have 6 different books, 2 of which are math books. In how many ways can I stack my 6 books on a shelf if I do not want the math books to be next to each other?

Ground Truth.

480

Model Generations.

- **EGSPO-SA (Correct)**

  1. The total number of ways to arrange 6 books on a shelf is $6! = 720$.
  2. Treat the 2 math books as a single unit. This reduces the problem to arranging 5 units (the math book unit and the other 4 individual books).
  3. The number of ways to arrange 5 units is $5! = 120$.
  4. Within the math book unit, the 2 math books can be arranged in $2! = 2$ ways.
  5. Therefore, the number of valid arrangements where the math books are next to each other is $5! \times 2! = 120 \times 2 = 240$.
  6. Subtract the number of arrangements where the math books are not next to each other from the total number of arrangements: $720 - 240 = 480$.

  So, the number of ways to stack the 6 books on a shelf such that the math books are not next to each other is $\boxed{480}$.

- **EGSPO (Correct)**

  First, we calculate the total number of ways to arrange the 6 books without any restrictions. This is given by the factorial of 6, which is $6! = 720$.

  Next, we calculate the number of arrangements in which the two math books are next to each other. We treat the two math books as a single unit. This reduces the problem to arranging 5 units (the block of two math books and the other 4 books). The number of ways to arrange these 5 units is $5! = 120$.

  Within the block, the two math books can be arranged in $2! = 2$ ways. Therefore, the total number of arrangements where the two math books are next to each other is $120 \times 2 = 240$.

  To find the number of arrangements where the two math books are not next to each other, we subtract the number of arrangements where they are next to each other from the total number of arrangements:

  $$720 - 240 = 480$$

- **d1 (Incorrect)**

  To solve this problem, we need to consider the number of ways to arrange the 6 books such that the two math books are not next to each other. We can treat the pair of math books as a single unitblock" to simplify the problem. This means we are now arranging 5 "blocks" (the pair of math books and the other 4 books).

  The number of ways to arrange these5 blocks is $5!$ (5 factorial). However, within the pair of math books, there are 2 ways to arrange the two math books (either math first and science second, or science first and math second).

  Therefore, the total number of ways to arrange the 6 books such that the math books are not next to each other is $5! \times 2$.

  Calculating this, we get:

  $$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

  the total number of arrangements is:

  $$120 \times 2 = 240$$

Sudoku Example

Problem. Solve the following 4×4 Sudoku puzzle:

0000304014022304

Ground Truth.

4123324114322314

Model Generations.

- **EGSPO-SA (Correct)**

  To solve this Sudoku puzzle, we need to fill the empty cells with the digits 1-4 such that each row, each, and each 2x2 box contains all the digits exactly once. We can start by filling the first row with the digits 3 and 4. Then, we can fill the second row with the digits 1 and 4. Next, we can fill the remaining rows and columns with the remaining digits. </reasoning> <answer> 4123324114322314 </answer>

- **EGSPO (Correct)**

  To solve the 4x4 Sudoku puzzle, we need to fill in the empty cells with the digits 1-4 such that each row, column, and 2x2 box contains the digits 1-4 exactly once. We can start by analyzing the given puzzle and filling in the empty cells according to the rules. We need to ensure that each row, column, and 2x2 box contains the digits 1-4 exactly once. </reasoning> <answer> 4123324114322314 </answer>

- **d1 (Incorrect)**

  To solve this 4x4 Sudoku puzzle, we need to fill in the empty cells with digits 1-4 such that each row, column, and 2x2 box contains all digits exactly once. We can start by placing the known digits and then filling in the empty cells guided by the rules of Sudoku. We will use a systematic approach to fill in the grid, ensuring that each row, column, and box conforms to the Sudoku rules. </reasoning> <answer> 1234314214322314 </answer>

GSM8k Example

Problem. Imma has 3 cats. She feeds her cats twice a day with 60 grams of cat food. How many days will 720 grams of cat food last?
Ground Truth.

2.0

Model Generations.

- **EGSPO-SA (Correct)**

  First, we need to determine the total amount of cat food consumed by all three cats in one day. Each cat eats 60 grams of cat food per feed, and and she feeds each cat twice a day. Therefore, each cat eats 60 grams * 2 feeds = 120 grams of cat food per day.
  Since there are 3 cats, the total amount of cat food consumed by all cats per day is: 120 grams perat * 3 cats = 360 grams of cat food per day.
  Now, we need to find out how many days 720 grams of cat food will last. Therefore, we divide the total amount of cat food (720 grams) by the amount consumed in one day (360 grams).
  Therefore, the number of days is: 720 grams / 360 grams/day = 2 days.
  Therefore, 720 grams of cat food will last for 2 days.
  <answer> $\boxed{2}$ </answer>

- **EGSPO (Correct)**

  Imma has 3 cats, and she feeds each cat twice a day with 60 grams of cat food. Therefore, each cat consumes 2 * 60 grams = 120 grams of cat food per day. For 3 cats, the total food consumed per day is 3 * 120 grams = 360 grams. To find out how long a supply of 720 grams of cat food will last, we divide the total amount of food by the daily consumption: 720 grams / 360 grams = 2 days.
  <answer> $\boxed{2}$ </answer>

- **d1 (Incorrect)**

  To determine how many days 720 grams of cat food will last, we need to first calculate the total amount of cat food consumed per day by allma's cats. Each cat is fed twice a day with 60 grams of food food, so, each cat consumes 60 grams per 2 meals = 30 grams per day. Since Imma has 3 cats, the total daily consumption is 30 grams per cat * 3 cats = 90 grams per day. Now, we need to find out how many days 720 grams will last by dividing the total amount of cat food by the daily consumption: 720 grams / 90 grams per day = 8 days.
  <answer> $\boxed{8}$ </answer>

Countdown Example

Problem. Numbers: [45, 72, 25]
Target: 52

Model Generations.

- **EGSPO-SA (Correct)**

  To create an arithmetic expression that evaluates to exactly 52 using the numbers 45, 72, and 25, we need to consider the operations +, -, *, and /. Let's start by checking the following:
  1. The sum of 45 and 72 is 117, which is greater than 52.
  2. The sum of 45 and 25 is 67, which is less than 52.
  3. The sum of 72 and 25 is 97, which is greater than 52.
  4. The product of 45 and 72 is greater than 52.
  5. The product of 72 and 25 is 1800, which is greater than 52.
  6. The difference of 72 and 45 is 27, which is less than 52.
  7. The difference of 72 and 25 is 47.

  <answer> $\boxed{72 - 45 + 25}$ </answer>

- **EGSPO (Correct)**

  To create an arithmetic expression that evaluates to 52 using the numbers 45, 72, and 25, we need to consider the properties of the operations +, -, *, and /. Let's break down the possible combinations:
  1. **Addition**: 45 + 72 = 117 2. **Subtraction**: 117 - 25 = 92 3. **Multiplication**: 72 * 25 = 1800 4. **Division**: 1800 / 45 = 40
  We need to find a combination that results in 52. First, let's try subtraction and subtraction:
  - 72 - 45 = 27
  - 27 - 25 = 2

  None of these combinations us 52. Let's try subtraction and addition:
  - 72 - 45 = 27
  - 27 + 25 = 52

  </reasoning>

  <answer> $\boxed{72 - 45 + 25}$ </answer>

- **d1 (Incorrect)**

  To find an arithmetic expression that evaluates to 52 using the numbers 45, 72, and 25, we need to consider the operations +, -, *, and / and ensure each number is used exactly once. Let's start by trying different combinations:
  1. **, we can try adding:**
  - 45 + 72 = 117
  - 117 - 25 = 92
  - 92 - 45 = 47
  - 47 + 25 = 72
  - 72 - 45 = 27
  - 27 + 25 = 52

  2. **, we can try subtracting:**
  - 45 - 72 = -27
  - -27 - 25 = -52

  3. **, we can try multiplying:**
  - 45 * 72 = 3240
  - 3240 / 25 = 52

  <answer> $\boxed{3240/25}$

K Limitation

While our framework provides a principled approach to RL for diffusion language models, it has several limitations worth noting.

Verifiable rewards only. All experiments use tasks with verifiable rewards (correctness on math, logical reasoning, and code), where the reward signal is well-defined and bounded by a known constant. Extending to settings with learned, unbounded rewards, preference-based feedback, or non-verifiable rewards such as formal proofs would require additional care, as the boundedness assumption underlying our entropy bound would no longer hold tightly.

Scope of diffusion paradigm. Our MDP formulation and policy gradient theorem are derived for masked diffusion language models with monotonic unmasking, where tokens are not revised once decoded. Extending to remasking-capable masked diffusion [Wang et al., 2025b] or to non-masked

discrete diffusion paradigms [Lou et al., 2023] would require revisiting the action space and transition dynamics of the MDP. We leave a general formulation that unifies these settings to future work.

L LLM Usage

Large language models were used to assist with editing, formatting, and revising the manuscript. The authors reviewed the resulting text and take responsibility for the accuracy of all statements, mathematical arguments, and reported results. LLMs were not used as a component of the proposed training method or experimental evaluation.