PRINCIPLED RL FOR DIFFUSION LLMS EMERGES FROM A SEQUENCE-LEVEL PERSPECTIVE

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning (RL) has proven highly effective for autoregressive language models, but adapting these methods to diffusion large language models (dLLMs) presents fundamental challenges. The core difficulty lies in likelihood approximation: while autoregressive models naturally provide token-level conditional probabilities essential for token-level RL objectives (e.g., GRPO), dLLMs generate sequences through iterative non-autoregressive denoising steps that lack this factorization. To address this fundamental mismatch, we propose ELBO-based Sequence-level Policy Optimization (ESPO), a principled RL framework that treats entire sequence generation as a single action and uses the ELBO as a tractable sequence-level likelihood proxy. Our method incorporates per-token normalization of importance ratios and robust KL-divergence estimation to ensure stable large-scale training. Extensive experiments on mathematical reasoning, coding, and planning tasks demonstrate that ESPO significantly outperforms token-level baselines, achieving dramatic improvements of 20-40 points on the Countdown task, while maintaining consistent gains on math and coding benchmarks. Our approach establishes sequence-level optimization as a principled and empirically effective paradigm for RL in dLLMs.

1 INTRODUCTION

Large language models (LLMs) (OpenAI, 2023) have become a cornerstone of modern natural language processing, achieving remarkable progress across math (Guo et al., 2025), coding (Hui et al., 2024), and planning tasks (Yao et al., 2023). While autoregressive (AR) modeling has long dominated this field, recent advances in *diffusion large language models* (dLLMs) have demonstrated strong potential as an alternative formulation (Ou et al., 2024; Shi et al., 2024; Sahoo et al., 2024; Nie et al., 2025; Ye et al., 2025). By modeling language generation as an iterative denoising process, dLLMs bypass the left-to-right dependency of AR models and offer advantages in long context (Liu et al., 2025a), multimodal (Yang et al., 2025; You et al., 2025a; Li et al., 2025; Yu et al., 2025) and fast inference (Inception Labs et al., 2025; DeepMind, 2025; Song et al., 2025).

With the advent of powerful pretrained dLLMs, the next frontier lies in post-training (Ouyang et al., 2022) to further enhance their capabilities. Among various post-training paradigms, reinforcement learning (RL) has emerged as a powerful approach that enables test-time scaling (Snell et al., 2025) through verifiable rewards (Guo et al., 2025). It has yielded substantial gains on reasoning tasks in recent AR models (OpenAI, 2024), such as math (Cobbe et al., 2021b), coding (Chen et al., 2021), and reasoning (Liu et al., 2023b). Motivated by this success, a natural question arises: how can we extend reinforcement learning to dLLMs?

Applying RL to dLLMs, however, is nontrivial. Mainstream RL algorithms in language modeling (e.g., GRPO (Shao et al., 2024)) assume a left-to-right factorization of the sequence likelihood and rely on token-level importance ratios $\frac{\pi_\theta(y^k|x,y^{<k})}{\pi_{\theta_{\text{old}}}(y^k|x,y^{<k})}$. In contrast, dLLMs generate sequences non-autoregressively, making such token-level conditionals either ill-defined or computationally expensive. Prior attempts to address this discrepancy have resorted to heuristic approximations—such as mean-field surrogates (Zhao et al., 2025) or token-level ELBO contributions (Yang et al., 2025; Gong et al., 2025)—or else computationally heavy trajectory-level formulations (Huang et al., 2025).

None of these approaches fully resolves the mismatch between autoregressive RL objectives and the holistic generation process of dLLMs.

In this work, we address this fundamental conflict by introducing *ELBO-based Sequence-level Policy Optimization (ESPO)*, a sequence-level reinforcement learning framework tailored for dLLMs. Our key insight is that dLLMs should not be forced into an autoregressive token-level action space. Instead, we treat the generation of an entire sequence as a single action, leveraging the ELBO as a tractable proxy for the intractable sequence log-likelihood. We further incorporate stabilization techniques essential for large-scale training, including per-token normalization of the ELBO ratio and robust KL-regularization. Our method eliminates the inconsistencies introduced by heuristic token-level approximations and enables stable, computationally efficient training.

Empirically, we validate the effectiveness of our design through extensive ablation studies, which confirm that combining sequence-level optimization with the ELBO objective provides a stable and principled foundation for reinforcement learning in dLLMs. Beyond ablations, we further evaluate our method on mainstream tasks spanning mathematics, coding, and planning. Across both LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025), our approach consistently outperforms prior dLLM-RL baselines such as d1 (Zhao et al., 2025) and wd1 (Tang et al., 2025), with particularly strong gains on planning tasks that require global consistency.

In summary, we make the following contributions:

- We provide a systematic analysis of why standard autoregressive RL objectives are incompatible with the non-autoregressive dLLMs, clarifying the limitations of existing heuristic approaches.
- We propose ESPO, a principled sequence-level RL framework that leverages the ELBO as a tractable proxy for sequence likelihood and introduces stabilized ratio and KL estimators for robust large-scale training.
- We demonstrate through comprehensive experiments and ablation studies that ESPO yields stable optimization and consistent improvements across math, coding, and planning benchmarks, surpassing prior dLLM-RL methods.

2 Background

2.1 Diffusion Large Language Models

Diffusion Large Language Models (dLLMs), or more specifically Masked Diffusion Models (MDMs), define a forward process that gradually corrupts the clean input by replacing tokens with the mask token M. Given the prompt x and the clean completions y, the forward process $q_t(y_t \mid y, x)$ at time t is defined as follows:

$$q_t(y_t|y,x) = \prod_{i=1}^{L} q_t(y_t^i|y^i,x) \quad \text{and} \quad q_t(y_t^i|y^i,x) = \begin{cases} 1-t, & y_t^i = y^i, \\ t, & y_t^i = \mathrm{M}. \end{cases} \tag{1}$$

Unlike autoregressive models, the exact log-likelihood $\log \pi_\theta(y|x)$ in dLLMs is typically approximated via the evidence lower bound (ELBO) (Ou et al., 2024; Shi et al., 2024; Sahoo et al., 2024):

$$\mathcal{L}_\theta(y|x) \triangleq \mathbb{E}_{t\sim\mathcal{U}[0,1]}\mathbb{E}_{y_t\sim q_t(y_t|y,x)}\left[\frac{1}{t}\sum_{i=1}^{L}\mathbf{1}[y_t^i=\mathrm{M}]\log p_\theta(y^i|y_t,x)\right] \le \log \pi_\theta(y|x). \tag{2}$$

As noted in Ou et al. (2024); Nie et al. (2025); Zhu et al. (2025), Eq. (2) has an equivalent, lower-variance variant that replaces the continuous masking ratio t with a discrete number of masked tokens l:

$$\mathcal{L}'_\theta(y|x) \triangleq \mathbb{E}_{l\sim\mathcal{U}(\{1,2,\ldots,L\})}\mathbb{E}_{y_l\sim q_l(y_l|y,x)}\left[\frac{L}{l}\sum_{i=1}^{L}\mathbf{1}[y_l^i=\mathrm{M}]\log p_\theta(y^i|y_l,x)\right], \tag{3}$$

where l denotes the number of tokens masked (sampled uniformly), and y_l represents the corrupted sequence obtained by masking l tokens without replacement. While Eq. (2) and Eq. (3) only provide a bound, the ELBO has been empirically shown to be a tight and practically effective surrogate for the intractable exact likelihood. This holds true for both likelihood-based evaluation (e.g., perplexity (Lou et al., 2024)) and training (e.g., DPO variants (Zhu et al., 2025)).

2.2 REINFORCEMENT LEARNING

Policy gradient methods have been shown to be highly effective for post-training LLMs (Ouyang et al., 2022). Among them, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is widely adopted as it eliminates this need for a value model (Schulman et al., 2017) by replacing it with a simpler Monte Carlo estimation: Given a prompt x, GRPO samples a group of G candidate completions $\{y^{(i)}\}_{i=1}^{G}$ from the old policy $\pi_{\theta_{\text{old}}}$. Instead of estimating the baseline with a learned value function, it computes the relative advantage of each sample as its reward minus the group mean reward (Liu et al., 2025c). Incorporating a KL penalty, the resulting optimization objective is:

$$\mathcal{J}_{\text{GRPO}}(\pi_\theta) = \mathbb{E}_{x\sim\mathcal{D},y^{(1)},\ldots,y^{(G)}\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y^{(i)}|}\sum_{k=1}^{|y^{(i)}|}\min(\rho^{k,(i)}\hat{A}^{(i)}, \text{clip}(\rho^{k,(i)}, 1-\epsilon, 1+\epsilon)\hat{A}^{(i)}) - \beta D_{\text{KL}}(\pi_\theta, \pi_{\text{ref}})\right], \quad (4)$$

where $\rho^{k,(i)} = \frac{\pi_\theta(y^{k,(i)}|x, y^{<k,(i)})}{\pi_{\theta_{\text{old}}}(y^{k,(i)}|x, y^{<k,(i)})}$ is the token-level importance ratio between policies, and $\hat{A}^{(i)} = R(x, y^{(i)}) - \frac{1}{G}\sum_{j=1}^{G} R(x, y^{(j)})$ denotes the group-relative advantage. A more comprehensive introduction to reinforcement learning is provided in Appendix B.1.

3 THE CHALLENGE OF THE TOKEN-LEVEL PERSPECTIVE IN DLLMS

The core challenge in applying reinforcement learning to dLLMs stems from a fundamental mismatch between the dLLMs' probabilistic modeling and the assumptions inherent in standard RL algorithms. Mainstream policy gradient algorithms, including GRPO as formulated in Eq. (4), are intrinsically designed for autoregressive models that factorize the sequence likelihood into a product of conditional probabilities as $\pi_\theta(y|x) = \prod_{k=1}^{L}\pi_\theta(y^k|x, y^{<k})$. This factorization naturally defines a sequence of actions, allowing the objective to assign rewards at the token level via the importance ratio $\frac{\pi_\theta(y^k|x,y^{<k})}{\pi_{\theta_{\text{old}}}(y^k|x,y^{<k})}$. However, dLLMs generate text non-autoregressively, refining a complete sequence over iterative denoising steps. Consequently, the autoregressive conditional probability $\pi_\theta(y^k|x, y^{<k})$ is ill-defined or hard to compute (see Appendix B.2 for detailed discussions), forcing existing methods to rely on heuristic proxies to bridge this gap.

Early attempts to resolve this incompatibility focused on finding a suitable token-level substitute for the AR conditional probability $\pi_\theta(y^k|x, y^{<k})$. For instance, d1 (Zhao et al., 2025) employed a mean-field approximation $\log p_\theta(y^k|x)$ as a proxy for $\log\pi_\theta(y^k|x, y^{<k})$. This approach is inaccurate as it ignores the context provided by other tokens in the sequence y. Recognizing this limitation, subsequent work such as UniGRPO (Yang et al., 2025) and Coupled-GRPO (Gong et al., 2025) proposed a more sophisticated proxy: the token's contribution to the ELBO, $\mathcal{L}_\theta^k(y|x)$[1]:

$$\mathcal{L}_\theta^k(y|x) \triangleq \mathbb{E}_{t\sim\mathcal{U}[0,1]}\mathbb{E}_{y_t\sim q_t(y_t|y,x)}\left[\frac{1}{t}\mathbf{1}[y_t^k = \text{M}]\log p_\theta(y^k|y_t, x)\right]. \quad (5)$$

The $\mathcal{L}_\theta^k(y|x)$ is better aligned with the nature of dLLM generation, since its computation involves predicting a masked token given both x and the surrounding unmasked context from y. However, the ELBO, $\mathcal{L}_\theta(y|x) = \sum_{k=1}^{L}\mathcal{L}_\theta^k(y|x)$, is only valid on sequence level as lowerbound of $\log\pi_\theta(y|x)$. An individual component $\mathcal{L}_\theta^k(y|x)$ has no formal probabilistic interpretation as a conditional likelihood. Therefore, the decomposition of ELBO at the token level and its heuristic substitution into the GRPO objective (Eq. (4)) introduces an unknown inconsistency.

As analyzed above, the core issue is not merely about finding a better token-level proxy, but that the token-level decomposition itself fundamentally does not fit for dLLMs. Forcing a dLLM into a token-level AR framework rests on an improper assumption. This motivates our approach: instead of adapting the dLLM model to fit the algorithm, we must adapt the algorithm to respect the holistic, sequence-level nature of the dLLM model.

[1]Implementations may vary; for instance, UniGRPO's ELBO-like term omits the $\frac{1}{t}$ coefficient from Eq. (5), akin to the simplified objective in DDPM (Ho et al., 2020).

4 A PRINCIPLED SEQUENCE-LEVEL RL FRAMEWORK FOR dLLMS

Motivated by analysis in Section 3, we propose *ELBO-based Sequence-level Policy Optimization (ESPO)* algorithm, which is tailored for dLLMs. Our approach is built on a sequence-level action space, uses the ELBO as a tractable proxy for the sequence log-likelihood, and incorporates crucial stabilization techniques for both the policy gradient objective and the KL-divergence regularizer.

4.1 THE SEQUENCE-LEVEL POLICY OBJECTIVE WITH ELBO APPROXIMATION

We begin by reformulating the RL objective to align with the nature of dLLM generation.

Sequence-Level Objective. Instead of viewing each token as an independent action, we treat the generation of the entire sequence y as a single, atomic action. This naturally leads to a sequence-level adaptation of the GRPO framework (Eq. (4)), where the token-level summation is removed. The objective now depends on a sequence-level importance ratio $\frac{\pi_\theta(y^{(i)}|x)}{\pi_{\theta_{\text{old}}}(y^{(i)}|x)}$. By substituting the intractable log-likelihood $\log \pi(y^{(i)}|x)$ with its ELBO approximation $\mathcal{L}(y^{(i)}|x)$, we obtain the sequence-level ratio for dLLM:

$$\rho_{\text{seq}}^{(i)} = \frac{\exp \mathcal{L}_\theta(y^{(i)}|x)}{\exp \mathcal{L}_{\theta_{\text{old}}}(y^{(i)}|x)} = \exp(\mathcal{L}_\theta(y^{(i)}|x) - \mathcal{L}_{\theta_{\text{old}}}(y^{(i)}|x)) \tag{6}$$

Plugging this into the GRPO objective gives us a vanilla sequence-level objective[2] :

$$\mathcal{J}_{\text{seq}}(\pi_\theta) = \mathbb{E}_{x\sim\mathcal{D}, y^{(1)},\ldots,y^{(G)}\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}\min(\rho_{\text{seq}}^{(i)}\hat{A}^{(i)}, \text{clip}(\rho_{\text{seq}}^{(i)}, 1-\epsilon, 1+\epsilon)\hat{A}^{(i)})\right], \tag{7}$$

While this sequence-level formulation correctly avoids the pitfall of splitting the ELBO at the token level, we found this vanilla formulation to be practically unusable. The magnitude of the raw ELBO difference, $\mathcal{L}_\theta(y^{(i)}|x) - \mathcal{L}_{\theta_{\text{old}}}(y^{(i)}|x)$, typically scales linearly with the sequence length L. The subsequent exponentiation results in astronomically large or infinitesimally small ratios, causing unstable optimization.

To address this instability, we draw inspiration from GSPO (Zheng et al., 2025) and normalize the log-ratio by the sequence length L. This transforms the unstable, raw log-likelihood difference into a stable, per-token scale. Our final, stabilized importance ratio is:

$$\rho_{\text{seq}}^{(i)} = \exp\left(\frac{1}{L}(\mathcal{L}_\theta(y^{(i)}|x) - \mathcal{L}_{\theta_{\text{old}}}(y^{(i)}|x))\right) = \exp\left(\frac{1}{L}\sum_{k=1}^{L}(\mathcal{L}_\theta^k(y^{(i)}|x) - \mathcal{L}_{\theta_{\text{old}}}^k(y^{(i)}|x))\right). \tag{8}$$

Plugging this stabilized ratio into the sequence-level objective (Eq. (7)) enables effective training.

Empirical Validation. To validate our design choices of using a sequence-level action space with an ELBO approximation, we conduct a complete ablation study on the Sudoku benchmark. We compare four variants: token-level actions with a mean-field likelihood, token-level with ELBO, sequence-level with mean-field, and our proposed sequence-level with ELBO. The precise mathematical formulations are detailed in Appendix C.1. As shown in Fig. 1, the results provide strong empirical support for our analysis:

- ***Mean-field is a poor proxy.*** Both mean-field variants (orange and green curve) fail to learn effectively, confirming that this approximation is fundamentally misaligned with the conditional denoising process.
- ***Token-level ELBO is unstable.*** The Token-level + ELBO approach (red curve), while initially promising, suffers from high instability and eventual collapse. This highlights the inconsistency of breaking the ELBO's integrity across tokens.
- ***Sequence-level ELBO is superior.*** Our proposed method (blue curve) is the only one to achieve fast, stable learning that converges to the highest reward, validating that the sequence-level action space paired with the holistic ELBO proxy is the correct and most effective approach.

[2]The KL term is omitted here for simplicity; a detailed discussion is provided later in Section 4.2.

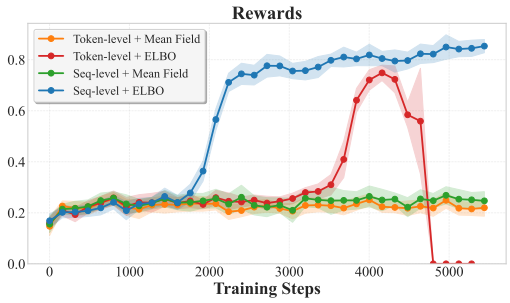

Figure 1: **Training performance on the Sudoku task under different action space (*Token-level* vs. *Sequence-level*) and likelihood approximations (*Mean-field* vs. *ELBO*).** Our method (blue) combines a sequence-level action space with an ELBO approximation, yielding the most stable and highest performance.

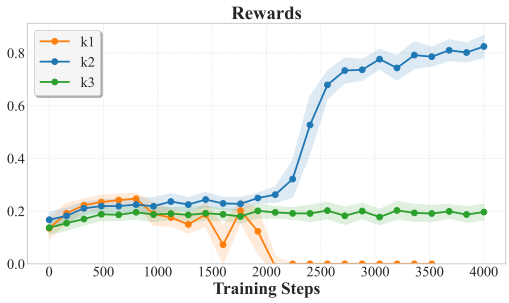

Figure 2: **Training performance on the Sudoku task with different KL-divergence estimators.** The k_2 estimator (blue) achieves stable and superior performance. The k_1 estimator (orange) is highly unstable and collapses, while the k_3 estimator (green) stagnates.

4.2 Stable KL-Divergence Estimation

The complete GRPO objective includes a KL-divergence term to regularize policy updates against a reference policy. A common choice for token-level autoregressive models is the k_3 estimator (Schulman, 2017). However, a direct application of the k_3 estimator to our sequence-level objective is highly problematic. Its formulation, when approximating log-likelihoods with the ELBO, is:

$$\widehat{\mathbb{KL}}_{\text{k3}} = \exp\Big(\mathcal{L}_{\text{ref}}(y^{(i)}|x) - \mathcal{L}_\theta(y^{(i)}|x)\Big) - 1 - \Big(\mathcal{L}_{\text{ref}}(y^{(i)}|x) - \mathcal{L}_\theta(y^{(i)}|x)\Big). \tag{9}$$

As the formula shows, the k_3 estimator contains an exponential term, which reintroduces the unstable problem similar to Eq. (6) at the sequence level. To circumvent this exponential instability, we adopt the more robust k_2 estimator (Schulman, 2017), which is known to yield a correct gradient for KL optimization (Tang & Munos, 2025b). Our practical and stable KL estimate becomes:

$$\widehat{\mathbb{KL}}_{\text{k2}} = \tfrac{1}{2}\Big(\mathcal{L}_\theta(y^{(i)}|x) - \mathcal{L}_{\text{ref}}(y^{(i)}|x)\Big)^2. \tag{10}$$

Unlike the k_3 estimator, the k_2 estimator is a simple quadratic function of the ELBO difference. This polynomial form avoids the exponential term entirely, ensuring that the gradient signal from the KL regularizer remains stable and well-behaved, even for long sequences.

Empirical Validation. To demonstrate the critical impact of the KL estimator, we conduct an ablation study on the Sudoku task comparing the performance of the k_1, k_2, and k_3 estimators. More details can be referenced in Appendices C.2 and G.3. As shown in Fig. 2, the choice of estimator is crucial for stable learning. The k_3 estimator (green) fails to learn, with rewards stagnating at a low level, which is consistent with our analysis of its unstable property. The k_1 estimator (orange) is also highly unstable; while it shows some initial progress, its performance violently fluctuates before collapsing to zero midway through training. In stark contrast, the k_2 estimator (blue) enables stable and efficient learning, consistently improving and ultimately converging to the highest reward. This result empirically validates that the k_2 estimator is the most robust and effective choice for our sequence-level framework.

5 Experiment

5.1 Experimental Setup

Models & Tasks We apply our ESPO algorithm to two open-source dLLMs: LLaDA-8B-Instruct (Nie et al., 2025) and Dream-7B-Instruct (Ye et al., 2025). For reference, we also report evaluation results on LLaDA-1.5 for comparison. Following prior work (Zhao et al., 2025; Tang

Table 1: **Model performance on mathematics and planning benchmarks.** For each task, we train a separate model. Countdown results with † for LLaDA, diffu-GRPO, and wd1 are from Zhao et al. (2025); Tang et al. (2025), while other results are reproduced as detailed in Section 5.1. Δ denotes the improvement of ESPO over LLaDA or Dream model without reinforcement post-training.

	GSM8K(0)				MATH(0)				Countdown				Sudoku			
Model / Seq Len	**128**	**256**	**512**	**Avg.**	**128**	**256**	**512**	**Avg.**	**128**	**256**	**512**	**Avg.**	**128**	**256**	**512**	**Avg.**
LLaDA	71.3	76.2	80.2	75.9	34.4	35.2	41.4	37.0	20.7†	19.5†	16.0†	18.7†	24.8	16.2	6.0	15.7
+ diffu-GRPO(d1)	74.6	78.1	81.2	78.0	34.9	36.6	41.7	37.7	33.2†	31.3†	37.1†	33.9†	26.7	24.1	15.9	22.2
+ wd1	77.2	80.8	82.3	80.1	33.3	37.7	39.8	36.9	47.7	51.2†	46.1†	48.3	22.6	22.0	24.6	23.1
+ ESPO (ours)	**80.0**	**82.3**	**83.7**	**82.0**	**36.0**	**39.0**	**43.4**	**39.5**	**81.6**	**82.0**	**79.3**	**81.0**	**92.7**	**84.7**	**80.5**	**86.0**
Δ	**+8.7**	**+6.1**	**+3.5**	**+6.1**	**+1.6**	**+3.8**	**+2.0**	**+2.5**	**+60.9**	**+62.5**	**+63.3**	**+62.3**	**+67.9**	**+68.5**	**+74.5**	**+70.3**
Dream	75.8	81.3	80.7	79.3	38.2	45.7	48.0	44.0	8.5	7.8	17.4	11.2	9.3	2.1	14.0	8.5
+ diffu-GRPO(d1)	77.0	81.9	81.7	80.2	39.4	46.9	48.9	45.1	27.3	27.7	37.5	30.8	64.4	69.7	51.1	61.7
+ wd1	76.3	**82.4**	**82.9**	80.5	39.5	**47.4**	**50.5**	45.8	28.9	37.9	42.2	36.3	29.5	39.2	30.3	33.0
+ ESPO (ours)	**79.6**	82.3	82.0	**81.3**	**40.3**	**47.4**	50.3	**46.0**	**68.8**	**66.8**	**64.8**	**66.8**	**71.7**	**72.3**	**71.3**	**71.8**
Δ	**+3.8**	**+1.0**	**+1.3**	**+2.0**	**+2.1**	**+1.7**	**+2.3**	**+2.0**	**+60.3**	**+59.0**	**+47.4**	**+55.6**	**+62.4**	**+70.2**	**+57.3**	**+63.3**

Table 2: **Model performance on coding benchmarks.** We train a single model and evaluate it across multiple coding benchmarks (HumanEval and MBPP) at different sequence lengths. Δ denotes the improvement of ESPO over LLaDA model without reinforcement post-training. ESPO consistently enhances the performance while even achieving competitive results compared with LLaDA-1.5, which was trained on a privately collected dataset at a significantly larger scale.

	HumanEval(0)								MBPP(3)							
	-				Plus				-				Plus			
Model / Seq Len	**128**	**256**	**512**	**Avg.**	**128**	**256**	**512**	**Avg.**	**128**	**256**	**512**	**Avg.**	**128**	**256**	**512**	**Avg.**
LLaDA	26.8	37.8	48.8	37.8	23.2	30.5	41.5	31.7	38.2	37.0	38.2	37.8	36.8	36.8	37.8	37.1
+ diffu-GRPO(d1)	25.6	36.0	**50.0**	37.2	22.0	29.9	37.2	29.7	34.8	36.6	38.0	36.5	31.8	34.8	39.4	35.3
+ wd1	**34.7**	38.4	38.4	37.2	**29.9**	29.9	32.9	30.9	38.0	37.2	34.4	36.5	31.8	32.8	35.6	33.4
+ ESPO (ours)	28.1	**42.1**	**50.0**	**40.1**	24.4	**36.6**	**42.7**	**34.6**	**47.4**	**44.6**	**44.2**	**45.4**	**38.9**	**41.6**	**42.6**	**41.0**
Δ	**+1.3**	**+4.3**	**+1.2**	**+2.3**	**+1.2**	**+6.1**	**+1.2**	**+2.9**	**+9.2**	**+7.6**	**+6.0**	**+7.6**	**+2.1**	**+4.8**	**+4.8**	**+3.9**
LLaDA-1.5	29.3	39.6	51.9	40.3	23.2	32.3	45.1	33.5	39.6	39.9	38.8	39.4	38.8	40.4	37.3	38.8

et al., 2025), we focus on three categories of reasoning tasks: (i) Mathematics: GSM8K (Cobbe et al., 2021a) and MATH (Hendrycks et al., 2021), (ii) Coding: HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021b), EvalPlus (HumanEval+ and MBPP+) (Liu et al., 2023a) and (iii) Planning: Countdown and Sudoku (Ye et al., 2024). For mathematical tasks, we train on the official training split of each dataset. For coding tasks, we follow Gong et al. (2025) and train on a subset of AceCoder-87K (Zeng et al., 2025). For planning tasks (Countdown and Sudoku), we train on synthetic training data following Zhao et al. (2025).

Training To evaluate ESPO's effectiveness, we apply it directly to the models without additional task-specific SFT. For stable training, we adopt two standard variance reduction techniques: following Zhu et al. (2025), we use antithetic sampling, which shares the same noise level and mask positions when estimating ELBO differences; and, inspired by Gong et al. (2025), we employ a coupled-sampling scheme. All experiments use 2 Monte Carlo samples and a policy update value of $\mu = 8$. Additional details on variance-reduction strategies are provided in Appendix D, while further training configurations and extended results can be found in Appendices E and G.

Evaluation Following d1, we evaluate all benchmarks at generation lengths of 128, 256, and 512. For mathematics and coding tasks, we use the official evaluation scripts provided by the LLaDA and Dream repositories, respectively. For planning tasks, we use the evaluation code based on d1. To ensure fair comparison, we re-run those baselines whose reported settings differ from ours or whose results are missing for specific lengths (e.g., when length-128 results were not originally reported).

5.2 Benchmark Results

As shown in Table 1 and Table 2, our sequence-level RL algorithm, ESPO, consistently and significantly outperforms both the original models and prior token-level RL baselines like diffu-GRPO (Zhao et al., 2025) and wd1 (Tang et al., 2025), while achieving performance on coding tasks

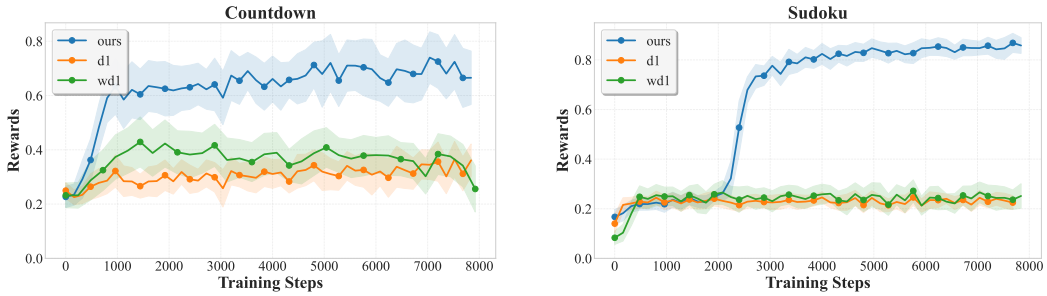

Figure 3: **Training dynamics with different methods for Countdown and Sudoku tasks on LLaDA-8b-Instruct.**

that is comparable to LLaDA-1.5. Notably, although our models are trained only on the sequence length of 256, the improvements generalize effectively to other lengths.

Dominant Performance on Planning Tasks. The most dramatic improvements remain in planning tasks. On Countdown, ESPO consistently outperforms the strongest baselines by 20–40 absolute points, while on the Sudoku tasks, the gains are up to over 60 points depending on the sequence length. This strongly validates our core hypothesis: a sequence-level objective is superior for tasks requiring holistic consistency, a property that token-level optimization fails to capture effectively.

Consistent Gains on Mathematics and Coding. On established mathematics and coding benchmarks, the gains are more modest but still consistently positive. Note that the pre-existing knowledge of the base models acts as a performance ceiling, limiting the maximum achievable gains through RL fine-tuning alone. Despite this, our method reliably enhances performance, surpassing all previous token-level dLLM-RL methods and comparable with LLaDA-1.5. This demonstrates the broad effectiveness of our approach even on knowledge-intensive tasks.

5.3 Training Dynamics

Fig. 3 shows the training reward dynamics on Countdown and Sudoku, highlighting our method's superior performance. The difference is most pronounced on Sudoku, where our sequence-level method rapidly converges to a near-optimal policy, while the token-level d1 and wd1 baselines completely stagnate at low rewards. Similar trends can be observed on the Countdown task.

5.4 Ablation Experiments

Number of Monte Carlo Samples We investigate the impact of the number of Monte Carlo (MC) samples used to estimate the ELBO. For this ablation, all other hyperparameters match those in our main experiments. As shown in Fig. 4, increasing the number of MC samples consistently improves training performance, though the magnitude of the effect is task-dependent. On the Sudoku task, the impact is dramatic: MC=1 results in slow convergence, whereas increasing to MC=2 and MC=4 makes it significantly faster. This suggests that a more stable and lower-variance ELBO estimation is crucial for navigating Sudoku's reward landscape. For the Countdown task, the trend is similar but less pronounced.

Number of different policy update values μ In these experiments, we varied μ while keeping all other settings identical to our main setup. Our method demonstrates remarkable robustness to the number of policy updates (μ) per data collection phase, as shown in Fig. 5. On both Countdown and Sudoku, our approach successfully scales to large μ values, converging to a similarly high reward across all settings, though smaller values (e.g., 8, 12) exhibit faster initial convergence on Sudoku. We hypothesize that this robustness is linked to task complexity. While the policy for relatively simple planning tasks can be updated aggressively, more complex domains like mathematics and coding may have more brittle reward landscapes. In such cases, a large μ could lead to instability, making smaller, more conservative update frequencies a safer and more effective choice.

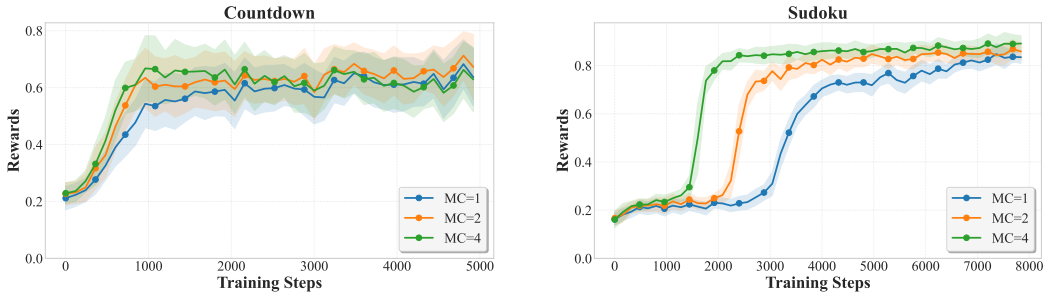

Figure 4: **Ablation study on the number of Monte Carlo samples for Countdown and Sudoku.** We evaluate training performance with different MC sample counts (1, 2, 4), showing the effect of increased sampling on reward optimization.

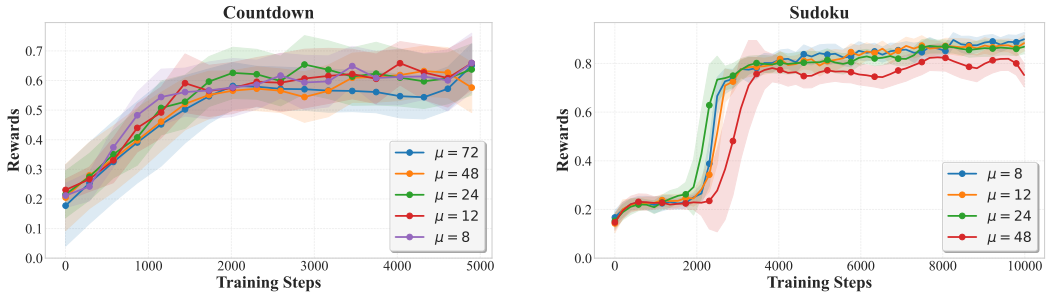

Figure 5: **Ablation study on the policy update values (μ) for Countdown and Sudoku.** The reward curves illustrate performance across a range of μ values. While smaller values (e.g., 8, 12) lead to faster initial convergence on Sudoku, the method is robust and achieves similarly high rewards across all settings for Countdown and Sudoku tasks.

5.5 Discussion of Training Efficiency

In reinforcement learning for dLLMs, the total training time can be roughly decomposed into two components: data generation (sampling) and policy updates (training). For dLLMs, the generation phase typically dominates: each step requires non-autoregressive denoising over the full sequence, preventing extensive reuse of KV cache. In contrast, the number of Monte Carlo samples M used for ELBO estimation only affects the policy update phase, which is less computationally intensive.

To quantify the computational cost, we use FLOPs as a hardware-agnostic proxy. For a model with N parameters and sequence length D, the forward and backward passes approximately require $C_{\text{forward}} = 2ND$ and $C_{\text{backward}} = 4ND$ FLOPs (Kaplan et al., 2020), respectively. We obtain the total FLOPS per sample with coupled sampling as $2ND(K + 6\mu M)$ (see Appendix F for details), where K is the sampling step, μ is the policy update values, and M is the number of MC samples.

We further validate this analysis on coding tasks using the same training configuration as in Section 5.1, except that the number of MC samples M is varied from 1, 2, and 4. The resulting theoretical FLOPs and empirical wall-clock training time are summarized in Table 3.

First, we observe that the theoretical FLOPs capture the overall trend of wall-clock time, consistent with the fact that dLLM inference is a compute-bound process. The small discrepancy may be attributed to factors such as GPU utilization, memory bandwidth, and communication overhead.

Second, focusing on the relative growth as M increases, our approach shows a much more moderate increase in training cost. Since generation costs are fixed (dominated by K), increasing M only adds to the policy update term, which is minor when M is within a proper range. For example, increasing M from 1 to 4 raises the total FLOPs by only about 47% in our case, consistent with the observed wall-clock time. In contrast, for ELBO-based DPO algorithms such as VRPO (Zhu et al., 2025), their training time scales almost linearly with M (e.g., a 4× increase from $M = 1$ to $M = 4$).

Table 3: **Training cost under different numbers of Monte Carlo (MC) samples for 100 steps** on the coding task. Following the setting in Section 5.1, the training parameters are set as denoising steps $K = 256$ and policy updates value $\mu = 8$. Experiments are conducted under H200 GPUs.

MC Samples (M)	**1**	**2**	**4**
Wall-clock Time (hrs)	5.61 (100%)	6.78 (121%)	9.06 (161%)
Theoretical FLOPs	$608ND$ (100%)	$704ND$ (116%)	$896ND$ (147%)

6 Related work

Diffusion language models. dLLMs have recently emerged as a powerful approach for sequence modeling, leveraging discrete diffusion (Austin et al., 2021a; Campbell et al., 2022; Meng et al., 2023; Lou et al., 2024; Zhao et al., 2024), or in particular masked diffusion models (Ou et al., 2024; Sahoo et al., 2024; Shi et al., 2024). Distinct from autoregressive models (OpenAI, 2023) that heavily rely on the left-to-right causal factorization of the sequence, dLLMs instead operate directly on the sequence-level for each diffusion step and enable flexible order sampling (Kim et al., 2025) and parallel decoding (Arriola et al., 2025). While they have shown great promise in various domains (Kwon et al., 2025; Ma et al., 2025; Liu et al., 2025b; Wu et al., 2025; Hu et al., 2025; You et al., 2025b) when scaled up (Nie et al., 2024; Prabhudesai et al., 2025; Ni & the team, 2025), and their adaptation to downstream tasks with specific reward signals still has room for exploration.

RL for language models. Reinforcement learning (Schulman et al., 2017) has proven effective in enhancing language model performance, particularly when rewards can be obtained via automated verifiers, a paradigm known as reinforcement learning with verifiable rewards (RLVR) (OpenAI, 2024). Methods such as GRPO (Shao et al., 2024; Guo et al., 2025) and related works (Luo et al., 2025) have improved reasoning capabilities and achieved strong performance across diverse tasks. However, directly applying these techniques to dLLMs is challenging, primarily due to the difficulty of likelihood computation.

RL for diffusion language models. Several works have explored leveraging RL to enhance the performance of dLLMs. For general discrete diffusion models, Zhang et al. (2025) proposes a framework by target concrete score matching,Venkatraman et al. (2024) proposes Relative Trajectory Balance to achieve unbiased sampling of posterior distribution, and Zekri & Boullé (2025) proposes score entropy policy optimization. For masked diffusion models, LLadou (Huang et al., 2025) uses RL and trains an additional module to predict the position for decoding. It models each denoising step as an action, stores the entire trajectory, and optimizes the conditional probability of each intermediate state. However, in offline settings, optimizing over multi-step trajectories requires repeated forward and backward passes growing linearly with the number of denoising steps, which is prohibitive for large-scale models. Various works (Zhao et al., 2025; Tang et al., 2025) have proposed mean-field surrogates of the likelihood that are efficient to compute to enhance efficiency. Yet, the approximations significantly trade off with quality and hinder the multi-step generation capability of the model. Our work instead develops a discrete diffusion RL objective from a principled sequence-level likelihood evaluation via efficient MC estimate, different from Yang et al. (2025); Gong et al. (2025) that evaluates the likelihood on the token-level heuristics.

7 Conclusion

In this work, we identified the fundamental incompatibility between autoregressive RL objectives and the non-autoregressive nature of dLLMs, and proposed ESPO, a principled sequence-level reinforcement learning framework that leverages the ELBO as a tractable proxy for sequence likelihood. By treating sequence generation as a single action and introducing stabilized importance ratios and KL regularization, ESPO eliminates inconsistencies in prior token-level approaches and enables robust and efficient large-scale training. Extensive experiments on math, coding, and planning benchmarks demonstrate that our method consistently outperforms existing dLLM-RL baselines. Our results establish sequence-level optimization as a principled and empirically effective paradigm for RL in diffusion language models.

ETHICS STATEMENT

This work is purely methodological and does not involve human subjects or sensitive data. Experiments are conducted on publicly available benchmarks, and results are reported in aggregate. We highlight that while our techniques improve efficiency and accuracy, they should be applied responsibly to avoid potential misuse.

REPRODUCIBILITY STATEMENT

We will open-source our code and checkpoints after the blind review. Comprehensive explanation and details of our theory and experiments can be found in Appendices C to E.

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. In *The Thirteenth International Conference on Learning Representations*, 2025.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 17981–17993, 2021a.

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

Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and A. Doucet. A continuous time framework for discrete denoising models. In *Advances in Neural Information Processing Systems*, 2022.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 11305–11315. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01103.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *ArXiv preprint*, abs/2110.14168, 2021a.

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*, abs/2110.14168, 2021b.

DeepMind. Gemini diffusion, 2025. URL `https://deepmind.google/models/gemini-diffusion/`.

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

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*, abs/2501.12948, 2025.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Zhanqiu Hu, Jian Meng, Yash Akhauri, Mohamed S. Abdelfattah, Jae-sun Seo, Zhiru Zhang, and Udit Gupta. Accelerating diffusion language model inference via efficient kv caching and guided diffusion. *ArXiv preprint*, abs/2505.21467, 2025.

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

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report, 2024.

Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. Mercury: Ultra-fast language models based on diffusion, 2025.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. *arXiv preprint arXiv:2502.06768*, 2025.

Taeyoun Kwon, Junhyuk Ahn, Taegeun Yun, Heeju Jwa, Yoonchae Choi, Siwon Park, Nam-Joon Kim, Jangchan Kim, Hyun Gon Ryu, and Hyuk-Jae Lee. Whisfusion: Parallel asr decoding via a diffusion transformer, 2025.

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 language model for multimodal understanding. *ArXiv preprint*, abs/2505.16839, 2025.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation, 2023a.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents, 2023b.

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

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

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *Conference on Language Modeling (COLM)*, 2025c.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning, ICML*, 2024.

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepcoder: A fully open-source 14b coder at o3-mini level, 2025. Notion Blog.

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

Chenlin Meng, Kristy Choi, Jiaming Song, and Stefano Ermon. Concrete score matching: Generalized score matching for discrete data, 2023.

Jinjie Ni and the team. Diffusion language models are super data learners, 2025. Notion Blog.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. *ArXiv preprint*, abs/2410.18514, 2024.

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

OpenAI. Gpt-4 technical report. *ArXiv preprint*, abs/2303.08774, 2023.

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

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *ArXiv preprint*, abs/2406.03736, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

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

Subham S. Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

John Schulman. kl-approx blogpost. `http://joschu.net/blog/kl-approx.html`, 2017.

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

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

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, Yuwei Fu, Jing Su, Ge Zhang, Wenhao Huang, Mingxuan Wang, Lin Yan, Xiaoying Jia, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Yonghui Wu, and Hao Zhou. Seed diffusion: A large-scale diffusion language model with high-speed inference, 2025.

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

Yunhao Tang and Rémi Munos. On a few pitfalls in kl divergence gradient estimation for rl. *arXiv preprint arXiv:2506.09477*, 2025a.

Yunhao Tang and Rémi Munos. On a few pitfalls in kl divergence gradient estimation for rl, 2025b.

Siddarth Venkatraman, Moksh Jain, Luca Scimeca, Minsu Kim, Marcin Sendera, Mohsin Hasan, Luke Rowe, Sarthak Mittal, Pablo Lemos, Emmanuel Bengio, Alexandre Adam, Jarrid Rector-Brooks, Yoshua Bengio, Glen Berseth, and Nikolay Malkin. Amortizing intractable inference in diffusion models for vision, language, and control. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

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

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*, abs/2505.22618, 2025.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

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

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b, 2025.

Zebin You, Shen Nie, Xiaolu Zhang, Jun Hu, Jun Zhou, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. Llada-v: Large language diffusion models with visual instruction tuning. *arXiv preprint arXiv:2505.16933*, 2025a.

Zebin You, Jingyang Ou, Xiaolu Zhang, Jun Hu, Jun Zhou, and Chongxuan Li. Effective and efficient masked image generation models, 2025b.

Runpeng Yu, Xinyin Ma, and Xinchao Wang. Dimple: Discrete diffusion multimodal large language model with parallel decoding. *ArXiv preprint*, abs/2505.16990, 2025.

Oussama Zekri and Nicolas Boullé. Fine-tuning discrete diffusion models with policy gradient methods. *ArXiv preprint*, abs/2502.01384, 2025.

Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. Acecoder: Acing coder rl via automated test-case synthesis. *ArXiv preprint*, abs/2502.01718, 2025.

Ruixiang Zhang, Shuangfei Zhai, Yizhe Zhang, James Thornton, Zijing Ou, Joshua Susskind, and Navdeep Jaitly. Target concrete score matching: A holistic framework for discrete diffusion. *ArXiv preprint*, abs/2504.16431, 2025.

Lingxiao Zhao, Xueying Ding, Lijun Yu, and Leman Akoglu. Improving and unifying discrete&continuous-time discrete denoising diffusion. *ArXiv preprint*, abs/2402.03701, 2024.

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

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025.

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

A The Use of Large Language Models

During the writing of this work, Large Language Models (LLMs) were used as auxiliary tools to assist with language polishing, grammar checking, and improving the readability of tables and figure captions. We will take full responsibility for all parts of the paper.

B From Generic RL Principles to dLLM RL Formulations

Scope of Derivation: We focus on the output reward setting, where the reward $R(x, y)$ is evaluated only upon the completion of the entire sequence y. This is the mainstream setup for reasoning tasks (e.g., Math, Coding) in post-training.

B.1 Generic RL Objective and Gradient Derivations

The standard RL objective is defined to maximize the expected accumulated reward :

$$J(\pi_\theta) = \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[R(x, y)\right]. \tag{11}$$

Using the log-derivative trick and introducing Importance Sampling (IS) to evaluate updates under the behavior policy π_{old}, the gradient can be derived as below (Williams, 1992):

$$\nabla_\theta J(\pi_\theta) = \nabla_\theta \mathbb{E}_{y\sim\pi_{\text{old}}(\cdot|x)}\left[\frac{\pi_\theta(y|x)}{\pi_{\text{old}}(y|x)} R(x, y)\right], \tag{12}$$

$$= \mathbb{E}_{y\sim\pi_{\text{old}}(\cdot|x)}\left[\frac{\pi_\theta(y|x)}{\pi_{\text{old}}(y|x)} \nabla_\theta \log \pi_\theta(y|x) R(x, y)\right], \tag{13}$$

$$= \mathbb{E}_{y\sim\pi_{\text{old}}(\cdot|x)}\left[\underbrace{\frac{\pi_\theta(y|x)}{\pi_{\text{old}}(y|x)}}_{\text{IS Ratio}} \nabla_\theta \log \pi_\theta(y|x) A(x, y)\right], \tag{14}$$

where $A(x, y)$ denotes the advantage function (typically $R(x, y) - b(x)$) introduced for variance reduction.

For Autoregressive models, the sequence likelihood factorizes as $\pi(y|x) = \prod_{i=1}^{d} \pi(y^i|x, y^{<i})$. Substituting this into Eq. (14) yields the full sequence-level gradient:

$$\mathbb{E}_{y\sim\pi_{\text{old}}}\left[\frac{\prod_{i=1}^{d}\pi_\theta(y^i|x,y^{<i})}{\prod_{i=1}^{d}\pi_{\text{old}}(y^i|x,y^{<i})}\sum_{i=1}^{d}\nabla_\theta\log\pi_\theta(y^i|x,y^{<i})A(x,y)\right]. \tag{15}$$

The cumulative product term (in green) introduces severe variance and numerical instability (exploding or vanishing gradients) as d increases, rendering Eq. (15) impractical for long sequences.

To address this, Token-level RL methods (e.g., PPO (Schulman et al., 2017), GRPO (Shao et al., 2024)) approximate the full ratio by decomposing it into per-token ratios. While practical implementations utilize clipping mechanisms or KL penalties to constrain policy updates and ensure the validity of this approximation, we omit these regularizers here to focus on the core gradient structure:

$$\nabla J_{\text{PPO/GRPO}} = \mathbb{E}_{y\sim\pi_{\text{old}}}\left[\frac{1}{d}\sum_{i=1}^{d}\frac{\pi_\theta(y^i|x,y^{<i})}{\pi_{\text{old}}(y^i|x,y^{<i})}\nabla_\theta\log\pi_\theta(y^i|x,y^{<i})A(x,y)\right]. \tag{16}$$

Conversely, Sequence-level RL methods (e.g., GSPO (Zheng et al., 2025)) argue that applying off-policy correction at the token level for a sequence-level reward is heuristic. They retain the sequence-level perspective but stabilize the ratio via length normalization. Similarly, omitting clipping and KL terms for simplicity, the gradient is formulated as:

$$\nabla J_{\text{GSPO}} = \mathbb{E}_{y\sim\pi_{\text{old}}}\left[\frac{1}{d}\left(\frac{\prod_{i=1}^{d}\pi_\theta(y^i|x,y^{<i})}{\prod_{i=1}^{d}\pi_{\text{old}}(y^i|x,y^{<i})}\right)^{\frac{1}{d}}\sum_{i=1}^{d}\nabla_\theta\log\pi_\theta(y^i|x,y^{<i})A(x,y)\right]. \tag{17}$$

B.2 Adapting RL Objectives to Diffusion Language Models

Applying the derived principles to dLLMs involves a critical choice regarding the action space and likelihood approximation.

The most direct approach is to formulate the RL problem at the *trajectory level* as a multi-step Markov decision process, where each denoising step t constitutes an action (Huang et al., 2025). Here, a "trajectory" $(y_T, y_{T-1}, \ldots, y_0)$ represents the evolution from pure noise y_T to clean data y_0. In this case, we factorizes over diffusion timesteps t as $\prod_{t=T}^{1}\pi_\theta(y_{t-1}|x, y_t)$ rather than token index i as $\prod_{i=1}^{d}\pi(y^i|x, y^{<i})$.

In this view, the RL gradient involves summing over all T timesteps, where T equals the number of sampling steps (typically $T \in [50, 1000]$). Taking the simplest online policy gradient as an example:

$$\nabla J_{\text{Traj}} = \mathbb{E}_{\pi_\theta}[\underbrace{\sum_{t=1}^{T}\nabla_\theta\log\pi_\theta(y_{t-1}|y_t,x)}_{\text{Requires } T \text{ gradient evaluations}}\cdot R(x,y_0)]. \tag{18}$$

While theoretically sound, this approach suffers from a prohibitive computational bottleneck due to the nature of the generative process. **Crucially, unlike Autoregressive models where all conditional probabilities** $\log\pi(y^i|y^{<i})$ **can be computed in parallel via a single network forward pass (enabled by causal masking), dLLMs fundamentally require** T **sequential network evaluations to compute the full trajectory probability.** Consequently, calculating gradients for ∇J_{Traj} requires backpropagating through T separate forward passes for every sample. This scales the compute cost **linearly with** T (e.g., $50\times \sim 1000\times$ more expensive than a single step), rendering it infeasible for large-scale post-training.

Given this constraint, we can optimize the final generated sequence directly using the ELBO as a proxy for the intractable log-likelihood. This is computationally feasible because the ELBO can be efficiently estimated via Monte Carlo (MC) sampling, requiring only K forward passes. In practice, the MC sample size is typically small ($K \approx 2 \sim 4$), offering a direct trade-off between the quality of the likelihood proxy and computational efficiency. This leads to the fundamental divergence between token-level and sequence-level formulations:

- **Sequence-Level Validity:** The ELBO $\mathcal{L}_\theta(y|x)$ is a mathematically rigorous variational lower bound for the joint sequence log-likelihood: $\mathcal{L}_\theta(y|x) \leq \log \pi_\theta(y|x)$. Therefore, substituting $\mathcal{L}_\theta(y|x)$ into the Sequence-level gradient formulation (Eq. (17)) preserves the mathematical integrity of the objective. The sequence-level ratio acts as a valid proxy for the global importance weight.
- **Token-Level Invalidity:** In contrast, the token-decomposed ELBO term $\mathcal{L}^i_\theta(y|x)$ is not a valid approximation for the autoregressive conditional probability $\log \pi_\theta(y^i|x, y^{<i})$ required by the Token-level PPO/GRPO formulation (Eq. (16)). The AR conditional $\pi_\theta(y^i|x, y^{<i})$ strictly depends on past tokens $y^{<i}$ (unidirectional), whereas the ELBO term $\mathcal{L}^i_\theta(y|x)$ relies on the bidirectional context of the denoising process. Forcing this mismatched proxy into Eq. (16) introduces an unknown inconsistency, explaining the instability observed in token-level baselines.

In summary, ESPO is a principled algorithm where the intractable likelihood is replaced by a reasonable variational bound (ELBO), constituting the only valid RL formulation for dLLMs under computational constraints.

C Details for Ablation Study in Section 4

C.1 Ablation for Action Space and Likelihood Approximation

To better analyze the objectives used in the ablation study (Fig. 1), we start by revisiting the GRPO formulation in Eq. (4). For clarity, we focus on its form for a single data sample $(x, y^{(i)})$ of length L. This yields the single-sample objective:

$$\mathcal{J}(x, y^{(i)}|\theta) = \frac{\hat{A}^{(i)}}{L} \sum_{k=1}^{L} \min\Big(\rho^{k,(i)},\ \text{clip}(\rho^{k,(i)}, 1-\epsilon, 1+\epsilon)\Big), \tag{19}$$

where $\rho^{k,(i)} = \frac{\pi_\theta(y^{k,(i)}|x, y^{<k,(i)})}{\pi_{\theta_{\text{old}}}(y^{k,(i)}|x, y^{<k,(i)})}$.

To highlight the core mechanism, we omit the clipping operator and obtain the simplified form:

$$\mathcal{J}(x, y^{(i)}|\theta) = \frac{\hat{A}^{(i)}}{L} \sum_{k=1}^{L} \rho^{k,(i)}. \tag{20}$$

Building on the simplified form in Eq. (20), we now extend the formulation to dLLMs. Since dLLMs do not provide autoregressive conditional probabilities directly, prior work replaces or approximates $\rho^{k,(i)}$ with diffusion-based likelihood surrogates. This gives rise to four variants used in our ablation:

- **Token-level + Mean-field (Orange Curve):** This approach, introduced by d1 (Zhao et al., 2025), applies a token-wise importance ratio using the mean-field approximation.

$$\mathcal{J}(x, y^{(i)}|\theta) = \frac{\hat{A}^{(i)}}{L} \sum_{k=1}^{L} \frac{p_\theta(y^{k,(i)}|x)}{p_{\theta_{\text{old}}}(y^{k,(i)}|x)} = \frac{\hat{A}^{(i)}}{L} \sum_{k=1}^{L} \exp\Big(\log p_\theta(y^{k,(i)}|x) - \log p_{\theta_{\text{old}}}(y^{k,(i)}|x)\Big). \tag{21}$$

- **Sequence-level + Mean-field (Green Curve):** This baseline extends the mean-field approach to the sequence level.

$$\mathcal{J}(x, y^{(i)}|\theta) = \hat{A}^{(i)} \cdot \exp\left(\frac{1}{L} \sum_{k=1}^{L} \Big[\log p_\theta(y^{k,(i)}|x) - \log p_{\theta_{\text{old}}}(y^{k,(i)}|x)\Big]\right). \tag{22}$$

- **Token-level + ELBO (Red Curve):** This method replaces the mean-field term with the token's contribution to the ELBO, but problematically computes the ratio for each token individually. Gong et al. (2025); Yang et al. (2025) follows this approach.

$$\mathcal{J}(x, y^{(i)}|\theta) = \frac{\hat{A}^{(i)}}{L} \sum_{k=1}^{L} \exp\Big(\mathcal{L}^k_\theta(y^{(i)}|x) - \mathcal{L}^k_{\theta_{\text{old}}}(y^{(i)}|x)\Big). \tag{23}$$

- **Sequence-level + ELBO (Ours, Blue Curve):** Our proposed method. It uses the ELBO as a proxy for the entire sequence log-likelihood and normalizes the log-ratio for stability, as described in Eq. (7) and Eq. (8).

$$\begin{aligned}\mathcal{J}(x, y^{(i)}|\theta) &= \hat{A}^{(i)} \cdot \exp\left(\frac{1}{L}\left[\mathcal{L}_\theta(y^{(i)}|x) - \mathcal{L}_{\theta_{\text{old}}}(y^{(i)}|x)\right]\right) \\ &= \hat{A}^{(i)} \cdot \exp\left(\frac{1}{L}\sum_{k=1}^{L}\left[\mathcal{L}^k_\theta(y^{(i)}|x) - \mathcal{L}^k_{\theta_{\text{old}}}(y^{(i)}|x)\right]\right). \end{aligned} \tag{24}$$

C.2 Ablation for KL Divergence Estimator

The choice of KL-divergence estimator is critical for stable sequence-level optimization. We analyze the three common kl estimators. As demonstrated in Fig. 2 of the main paper, only the k_2 estimator enables stable and effective learning. We provide a detailed analysis of their pitfalls here, supported by new visualizations of their training dynamics in Fig. 6.

Pitfalls in the k_3 estimator. The k_3 estimator commonly adopted in GRPO is given by

$$\widehat{\mathbb{KL}}_{\text{k3}}(\pi_\theta, \pi_{\text{ref}}) = \mathbb{E}_{y\sim\pi_\theta}\left[\frac{\pi_\theta(y)}{\pi_{\text{ref}}(y)} - 1 + \log\frac{\pi_\theta(y)}{\pi_{\text{ref}}(y)}\right], \tag{25}$$

where π_θ and π_{ref} are the target and reference policies. However, we argue that the k_3 estimator has several pitfalls.

First, the k_3 estimator introduces training instability. For a model with the output instead directly parameterizes the $\log$ policy, or logits, the k_3 estimator requires to take the exponent:

$$\widehat{\mathbb{KL}}_{\text{k3}} = \mathbb{E}_y\left[\exp\left(\mathcal{L}_\theta(y) - \mathcal{L}_{\text{ref}}(y)\right) - 1 + \mathcal{L}_\theta(y) - \mathcal{L}_{\text{ref}}(y)\right], \tag{26}$$

where $\mathcal{L}(y)$ approximates $\log \pi(y)$. We empirically find that during RL training $\mathcal{L}(y)$ may incur instability which will be magnified by the exponent, *i.e.*, the first term in Eq. 26. As shown in Fig. 6(right column), the k_3 estimate and its gradient norm are characterized by extreme, infrequent spikes, which destabilize and derail the entire training process.

Besides, while the k_3 estimator itself is an unbiased estimate of KL-divergence, its gradient is not (Tang & Munos, 2025a). In fact, it is shown that its gradient is instead an unbiased estimate of reverse-KL (Tang & Munos, 2025a):

$$\mathbb{E}\left[\widehat{\nabla\mathbb{KL}}_{\text{k3}}(\pi_\theta, \pi_{\text{ref}})\right] = \mathbb{E}\left[\widehat{\nabla\mathbb{KL}}(\pi_{\text{ref}}, \pi_\theta)\right] \neq \mathbb{E}\left[\widehat{\nabla\mathbb{KL}}(\pi_\theta, \pi_{\text{ref}})\right]. \tag{27}$$

Therefore, the optimization using k_3 estimator will exploit gradients that indeed optimize towards the reverse-KL, which will empirically lead to a different converged policy with bounded model capacity.

Pitfalls in the k_1 estimator. Another alternative is the original k_1 estimator for KL-divergence, which is given by

$$\widehat{\mathbb{KL}}_{\text{k1}}(\pi_\theta, \pi_{\text{ref}}) = \mathbb{E}_{y\sim\pi_\theta}\left[\log\frac{\pi_\theta(y)}{\pi_{\text{ref}}(y)}\right]. \tag{28}$$

However, by taking the gradient *w.r.t.* θ on Eq. 28, it is proved(Tang & Munos, 2025a) that the gradient equals zero, which means the $k_1$estimator has no signal of the KL constraint. This is empirically confirmed in Fig. 2 (right) of the main paper, where the k_1 policy (orange) initially learns but then catastrophically collapses. Fig. 6 (left column) further illustrates this: the KL estimate plummets into negative values, and the gradient norm grows uncontrollably, reflecting a complete loss of stability as the policy drifts unconstrained.

The k_2 estimator. We propose to employ the k_2 estimator

$$\widehat{\mathbb{KL}}_{\text{k2}}(\pi_\theta, \pi_{\text{ref}}) = \mathbb{E}_{y\sim\pi_\theta}\left[\frac{1}{2}\left(\log\pi_\theta(y) - \log\pi_{\text{ref}}(y)\right)^2\right], \tag{29}$$

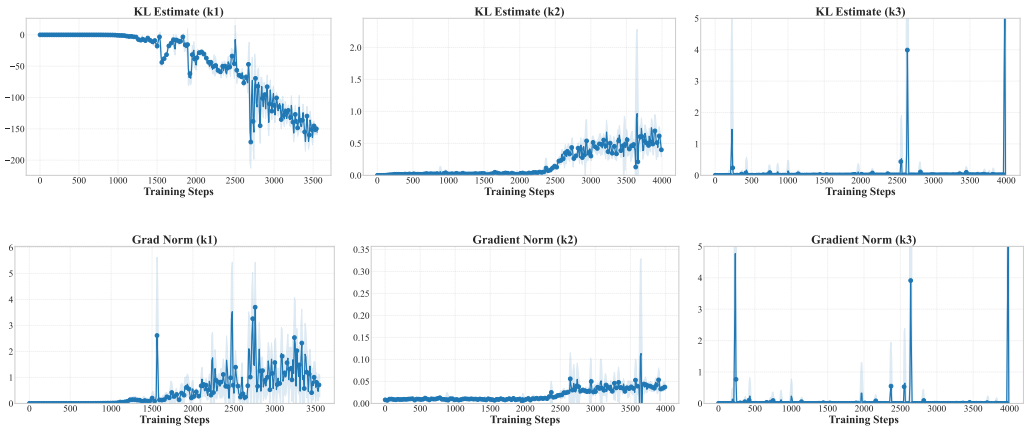

Figure 6: **Comparison of training dynamics with different KL divergence estimators.** The top row illustrates the KL estimates, while the bottom row displays the gradient norms for the k_1, k_2, and k_3 estimators. The k_1 estimator (left) lacks effective constraints, leading to model collapse. The k_3 estimator (right) suffers from severe instability and gradient spikes. In contrast, the k_2 estimator (middle) demonstrates superior stability and consistent gradient norms throughout training.

which leads to

$$\widehat{\mathbb{KL}}_{\text{k2}} = \mathbb{E}_{y\sim\pi_\theta}\left[\frac{1}{2}\left(\mathcal{L}_\theta(y) - \mathcal{L}_{\text{ref}}(y)\right)^2\right], \tag{30}$$

under the $\log$ parameterization. The k_2 estimator takes the form of an MSE loss that does not involve exponential term. As shown in Fig. 6 (middle column), both the KL estimate and its gradient norm are exceptionally stable, smooth, and well-behaved throughout training. Furthermore, the gradient of the k_2 estimator is indeed unbiased (Tang & Munos, 2025a):

$$\mathbb{E}\left[\widehat{\nabla\mathbb{KL}}_{\text{k2}}(\pi_\theta, \pi_{\text{ref}})\right] = \mathbb{E}\left[\widehat{\nabla\mathbb{KL}}(\pi_\theta, \pi_{\text{ref}})\right] \tag{31}$$

as opposed to the biased estimate in k_3 estimator.

This analysis, confirms that k_2 is the only robust and theoretically sound choice for our high-variance, sequence-level optimization framework.

D Variance Reduction

As discussed in Section 2.1, there are two equivalent formulations of the ELBO. Empirically, Eq. (3) exhibits lower variance than Eq. (2) (Ou et al., 2024; Nie et al., 2025). Therefore, we adopt Eq. (3) as our estimator for the ELBO. To further stabilize training, we incorporate two variance reduction techniques: antithetic sampling through mask sharing (Zhu et al., 2025) and coupled sampling (Gong et al., 2025), which we describe below.

Antithetic Sampling through Mask Sharing. Antithetic sampling through mask sharing is used when computing the difference between two ELBO terms, such as $\mathcal{L}_\theta(y|x) - \mathcal{L}_{\theta_{\text{old}}}(y|x)$ in Eq. (8) and $\mathcal{L}_\theta(y|x) - \mathcal{L}_{\theta_{\text{ref}}}(y|x)$ in Eq. (10). Concretely, we share the sampled timesteps and masked positions between the Monte Carlo estimators of the two policies, thereby reducing variance through negative correlation.

Coupled Sampling. Coupled sampling was originally proposed for the ELBO in the form of Eq. (2) (Gong et al., 2025), but we adapt it to Eq. (3). Note that Eq. (3) is equivalent to

$$\mathbb{E}_{l\sim\mathcal{U}(\{0,1,2,\ldots,L\})}\mathbb{E}_{y_l\sim q(y_l|l,y,x)}[\ell_\theta(y_l, l, y|x)], \tag{32}$$

where

$$\ell_\theta(y_l, l, y|x) \triangleq \begin{cases} \dfrac{L+1}{l} \displaystyle\sum_{i=1}^{L} \mathbf{1}[y_l^i = \mathbf{M}] \log p_\theta(y^i \mid y_l, x), & l > 0, \\ 0, & l = 0. \end{cases} \tag{33}$$

Based on this formulation, we introduce a complementary masking strategy. For each sampled mask y_l, we construct a complementary mask $\tilde{y}_l$ such that the two masks partition the token set: every token masked in y_l is unmasked in $\tilde{y}_l$, and vice versa. We then average the two losses, which is also equivalent:

$$\mathbb{E}_{l\sim\mathcal{U}(\{0,1,2,\ldots,L\})}\mathbb{E}_{y_l\sim q(y_l|l,y,x)}\left[\frac{\ell_\theta(y_l, l, y|x) + \ell_\theta(\tilde{y}_l, L-l, y|x)}{2}\right]. \tag{34}$$

This construction guarantees that every token contributes at least once to the learning signal. Further, the estimator achieves lower variance and yields a more stable optimization objective.

E Experiment Details

As described in Section 5.1, we train separate models for each of the GSM8K, Math, Sudoku, and Countdown tasks using their respective training datasets. For code tasks, we train a unified model on AceCoder-87K (Zeng et al., 2025) and evaluate it on four benchmarks: HumanEval, HumanEval-Plus, MBPP, and MBPP-Plus.

All models are trained with a maximum sequence length of 256, while evaluation is performed at sequence lengths of 128, 256, and 512 to better assess length generalization.

E.1 Inference Setting

For simpler planning tasks (Sudoku and Countdown), we unmask 2 tokens per denoising step, resulting in $L/2$ total denoising steps (where $L = 256$ is the sequence length). For other tasks, the number of denoising steps is set equal to the sequence length to improve performance.

Sampling strategies are model-specific:

- **LLaDA**: We employ low-confidence sampling (Chang et al., 2022) combined with a semi-autoregressive decoding strategy (Arriola et al., 2025; Nie et al., 2025), with block length set to 32 for all tasks. Training and evaluation share most sampling settings, except that the temperature is 0.9 during training and 0 (greedy decoding) during evaluation, following the evaluation codebase (Nie et al., 2025).
- **Dream**: We use top negative entropy remasking and pure diffusion sampling, without semi-autoregressive decoding (Ye et al., 2025). Temperature is set to 0.9 during training and 0.1 during evaluation, while other sampling settings remain consistent.

E.2 Training Details

All reinforcement learning training is conducted using the TRL library (von Werra et al., 2020).

Parameter-Efficient Fine-Tuning For GSM8K, Math, Countdown, and Sudoku, we apply LoRA (Hu et al., 2022) with rank $r = 128$ and scaling factor $\alpha = 64$. For code tasks, full parameter fine-tuning is used to maximize performance.

Optimization Policy update value is set to $\mu = 8$, and the number of Monte Carlo (MC) samples is $M = 2$ for computational efficiency. Models are optimized using AdamW (Loshchilov & Hutter, 2019) with $\beta_1 = 0.9$, $\beta_2 = 0.99$. A constant learning rate schedule is used. For LoRA-based tasks, the learning rate is $3 \cdot 10^{-6}$ with weight decay 0.01 and gradient clipping 0.2. For code tasks, the learning rate is $1 \cdot 10^{-6}$, weight decay 0.1, and gradient clipping 0.8.

Table 4: **Model performance on Sudoku under the Dream setting (generation length = 24).** Unlike the d1 setting, the Dream setting restricts generation to 24 tokens, encouraging models to output answers directly without intermediate reasoning.

Model	Accuracy (%)
Dream-7B-Instruct	98.1
+ diffu-GRPO (d1)	96.0
+ ESPO (ours)	98.0

Batching We set the group size G in Eq. (7) and total batch size according to the difficulty of the tasks: GSM8K, Countdown, Sudoku use $G = 6$ and total batch size 96, Math uses $G = 16$ and batch size 256, and code tasks use $G = 10$ and batch size 160. Gradient accumulation is applied to enlarge the effective batch size.

Training Steps Models are evaluated when the reward curve converges. Code tasks are trained for 2k steps, Math for 3k steps, LLaDA planning tasks for 10k steps, and Dream planning tasks for 8k steps.

Special Notes on Sudoku The Sudoku setup in Dream (Ye et al., 2025) differs from d1 (Zhao et al., 2025). The prompt and data format in Dream are inconsistent with d1 (see Appendix H.1). Additionally, Dream restricts the generation length to 24 tokens, encouraging direct answers without intermediate reasoning. In contrast, d1 uses longer generation lengths (128/256/512), allowing the model to produce both intermediate reasoning steps and final answers.

In the main text, we report results under the d1 setting, as it better reflects the ability of models to perform multi-step reasoning when guided by our reinforcement learning algorithm. For completeness, we also present model performance under the Dream setting in Table 4.

Special Notes on Coding It is important to note that for coding tasks, we employed full fine-tuning, unlike the LoRA setting used for Math and Planning. In this regime, we observed significant instability with the baseline methods. Specifically, wd1 shows to be highly sensitive to hyperparameters; using default settings resulted in exploding gradient norms (reaching $10^3 \sim 10^4$) and immediate training collapse. While we were able to stabilize wd1 by carefully tuning the hyperparameters, it still suffered from performance degradation on certain metrics compared to the base model, as shown in Table 2. In contrast, ESPO remained stable and effective.

F Details of Training FLOPs

Recent advances in dLLMs (Wu et al., 2025; Ma et al., 2025; Liu et al., 2025b) enable partial usage of KV cache, but it will lead to the loss of downstream performance. Therefore, our analysis focuses on the vanilla case without KV cache.

As we mentioned in Section 5.5, for a model with N parameters and sequence length D, we use $C_{\text{forward}} = 2ND$ and $C_{\text{backward}} = 4ND$ to approximate the forward and backward passes FLOPS according to Kaplan et al. (2020). Combining the costs of generation and policy updates, we obtain the naive formula of total FLOPs per sample:

$$F_{\text{total}} = K \cdot C_{\text{forward}} + \mu M(C_{\text{forward}} + C_{\text{backward}}) = 2ND(K + 3\mu M), \tag{35}$$

where K is the sampling step, μ is the policy update values, and M is the number of MC samples. When using the coupled-sampling technique ,we will double the FLOPS of policy updates step, which leads to:

$$F_{\text{total}} = K \cdot C_{\text{forward}} + 2\mu M(C_{\text{forward}} + C_{\text{backward}}) = 2ND(K + 6\mu M), \tag{36}$$

Which indicates increasing M only adds to the policy update term ($6\mu M$ in Eq. (36)). When $K \gg 6\mu M$, the overall training cost grows mildly with M.

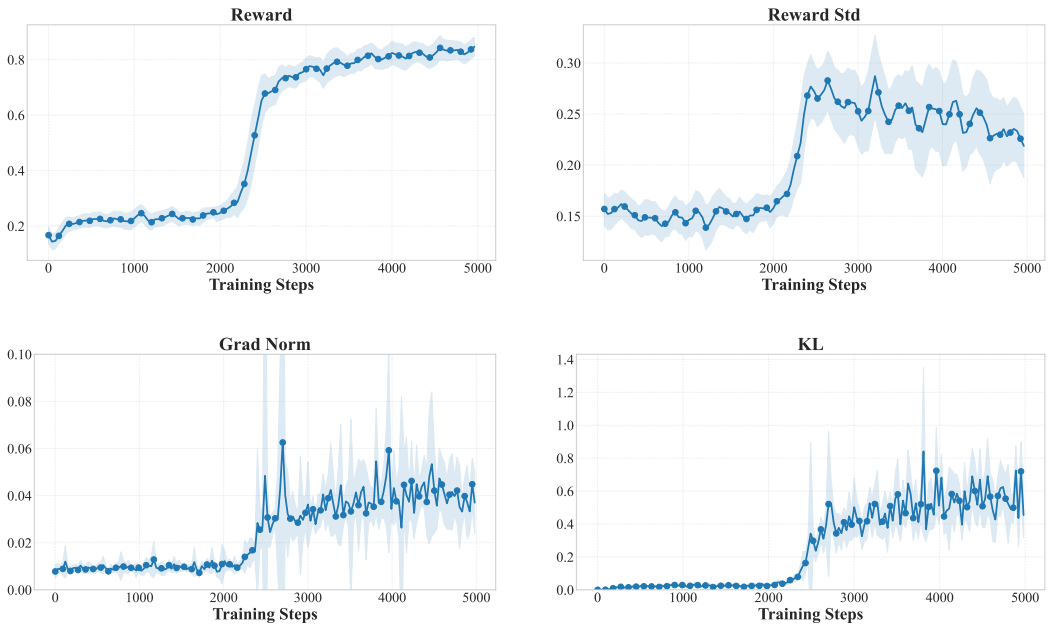

Figure 7: **Detailed training dynamics of the Sudoku Task on LLaDA-8B-Instruct.** We track reward, reward standard deviation, gradient norm, and KL divergence to analyze the learning process of our ESPO method.

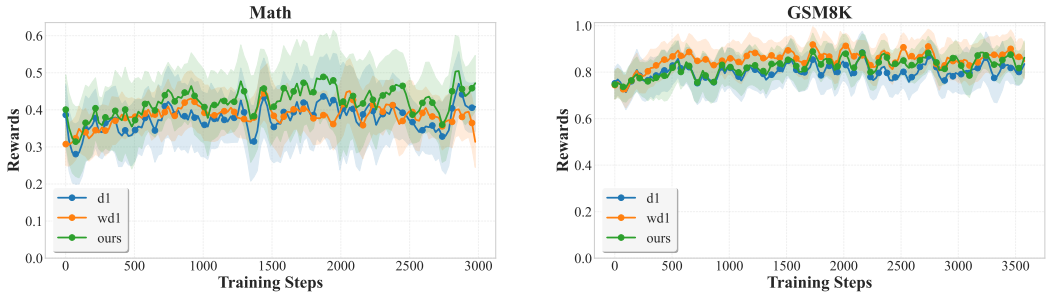

Figure 8: **Training dynamics with different methods for MATH and GSM8K benchmarks on LLaDA-8B-Instruct.**

G Additional Experimental Results

G.1 Detailed Training Dynamics of Sudoku

As shown in Fig. 7, the training dynamics exhibit a clear phase transition characterized by a sharp rise in reward. During the initial stage, the model explores with low rewards and stable gradients. However, coinciding exactly with the sudden performance jump (around step 2500), we observe a sharp spike in the gradient norm and a rapid surge in KL divergence. These signals indicate that the model has successfully discovered the strict logical rules of Sudoku, triggering a rapid restructuring of its policy and a decisive shift away from the reference model to exploit the newly discovered solution mechanism.

G.2 Training Curves on Mathematical Benchmarks

As illustrated in Fig. 8, the training reward trends for ESPO and the baselines are relatively stable and similar, indicating that all evaluated methods can robustly fit the training distribution given the reward signals. However, as observed in prior work (Liu et al., 2025c), the final reward on the

Table 5: **Performance comparison on Sudoku (LLaDA-8B-Instruct).** We compare the original d1 (using k_3), d1 adapted with the k_2 estimator, and our ESPO method.

Method / Seq Len	**128**	**256**	**512**
diffu-GRPO (d1) + k_3 (Original)	26.7	24.1	15.9
diffu-GRPO (d1) + k_2 (Ablation)	26.3	23.1	18.8
ESPO (Ours) + k_2	**92.7**	**84.7**	**80.5**

training set often exhibits a weak correlation with the accuracy on the held-out validation set due to the potential for overfitting. While the training rewards are similar, ESPO's superior performance on the test set (as reported in Table 1) demonstrates that our sequence-level optimization leads to better generalization rather than mere memorization of the training data.

G.3 Ablation Study on KL Estimator with Token-Level Baselines

To rigorously investigate whether the superior performance of ESPO is primarily driven by the robust k_2 KL estimator rather than our sequence-level framework, we conducted an additional ablation study on the token-level baseline. The wd1 method (Tang et al., 2025) reformulates the original RL objective as a weighted likelihood objective (similar to AWR) and does not involve an explicit KL estimation. Therefore, we only conducted ablations for diffu-GRPO (d1)Zhao et al. (2025) in the sudoku task, replacing its original unstable k_3 estimator with the robust k_2 estimator in Table 5.

It shows that simply applying the k_2 estimator to the token-level baseline (d1 + k_2) only yields performance comparable to the original baseline (d1 + k_3). Both variants stagnate at a low success rate ($\sim 20 - 26\%$). In stark contrast, ESPO achieves a dramatic improvement ($> 80\%$).

H TEXT DEMOS

H.1 SUDOKU PROMPTS

One-shot prompt for Sudoku task (d1-style settings):

```
Please solve the following 4x4 Sudoku puzzle. The puzzle is provided as a
     16-character string reading left-to-right, top-to-bottom, where '0'
    represents empty cells.

**Rules:**
- Fill empty cells with digits 1-4.
- Each row must contain digits 1-4 exactly once.
- Each column must contain digits 1-4 exactly once.
- Each 2x2 box must contain digits 1-4 exactly once.

**Example:**
Puzzle: 0401002010030310
This puzzle grid looks like this:
0 4 | 0 1
0 0 | 2 0
----+----
1 0 | 0 3
0 3 | 1 0

Solution: 2431312412434312
The solved grid looks like this:
2 4 | 3 1
3 1 | 2 4
----+----
1 2 | 4 3
4 3 | 1 2

**Important:** Your solution must be a COMPLETE 16-character string with
    only the digits 1-4, representing your final solved grid.

Respond in this exact format:
<reasoning>
Your step-by-step solving process
</reasoning>
<answer>
[16-character solution string with no spaces or separators]
</answer>


Now, solve the following Sudoku puzzle: 1240300020140100
```

One-shot prompt for Sudoku task (dream-style settings):

```
<|im_start|>system
You are an AI assistant skilled at solving Sudoku puzzles.<|im_end|>
<|im_start|>user
Fill the positions where the values are 0 in a 4x4 grid with digits 1-4
    so that each column, each row, and each of the four 2x2 subgrids that
     compose the grid contains all of the digits from 1 to 4.

Input:
0321
0030
0400
2100
Output:
4321
1234
3412
2143

Input:
0001
2100
1430
3000
Output:
<|im_end|>
<|im_start|>assistant
```

H.2 SUDOKU ANSWERS

Below, we show demos for answers for LLaDA+d1 and LLaDA+ ESPO:

Answer from LLaDA+d1:

```
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 examining the given digits and
    trying to fill them in according to the Sudoku rules.

1. The first row has the 0' and the 4's in the first and fourth columns.
2. The second row has the 4's in the first and fourth columns.
3. The third row has the 3 in the second column.
4. The fourth row has the 2 in the fourth column.
5. The fifth row has the 1 in the first column.
6. The fourth row has the 3 in the first column.
7. The first row has the 4 in the fourth column.

Using this information, we can start filling in the grid. We need to
    ensure that each row, column, and 2x2 box contains the digits 1-4
    exactly once.
</reasoning>
<answer>
12340
40221
30114
</answer>
```

Answer from LLaDA+ ESPO (ours):

```
<reasoning>
To solve the 4x4 Sudoku puzzle, we need to fill in the grid with digits
    1-4 such that each row, column, and 2x2 box contains all the digits
    exactly once. We can start by filling in the cells with the given
    digits and check for conflicts.

First, let's fill in the grid with the given digits:
3 2 0 4
0 0 3 0
1 4 0 0
2 0 0 1

Next, let's try filling fill in the remaining with digits 1-4, ensuring
    each row, column, and 2x2 box contains all the digits once.

After trying different combinations, we find a valid solution:
3 2 1 4
4 1 3 2
1 4 2 3
2 3 4 1

This solution satisfies all the rules: each row, column, and box contains
     all digits 1-4 exactly once.

</reasoning>

<answer>
3214413214232341
</answer>
```

H.3 COUNTDOWN

Countdown (LLaDA+d1): Target: 94 — Numbers: [77, 73, 98]

```
<reasoning>
To create an arithmetic expression that evaluates to 94 using the numbers
     [77, 73, 98], we need to find the correct combination of operations
    (+, -, *, /) that will inbalance the the larger number 98 and the
    smaller numbers to 94.
</reasoning>
<answer>
77*73-98
</answer>
```

Countdown (LLaDA+ESPO): Target: 67 — Numbers:[72, 92, 47]

```
<reasoning>
To create an arithmetic expression that evaluates to exactly 67 using the
     numbers [72, 92, 47], we need to consider the operations +, -, *,
    and / Let's start by examining the largest numbers and see if they
    can combine to give us target 67.

1. If we add 92 and 72, we get 164, which is too than 67.
2. If we subtract 92 and 72, we get 20, which is less than 67.
3. If we add 92 and 47, we get 139, which is more than 67.
4. If we subtract 72 and 47, we get 25, which is less than 67.

. If we combine 92 and 72, we get 164, which is more than 67.

Combining 92 and 47 will the closest answer to 67.
</reasoning>
<answer>
92 -72 + 47
</answer>
```