Boundary-Guided Policy Optimization for Memory-efficient RL of Diffusion Large Language Models

Anonymous ACL submission

Abstract

A key challenge in applying reinforcement learning (RL) to diffusion large language models (dLLMs) lies in the intractability of their likelihood functions, which are essential for the RL objective, necessitating corresponding approximation in each training step. While existing methods approximate the log-likelihoods by their evidence lower bounds (ELBOs) via customized Monte Carlo (MC) sampling, the forward computational graphs of all MC samples need to be retained for the gradient computation of non-linear terms in the RL objective, resulting in significant memory overhead. This constraint restricts feasible sample sizes, leading to imprecise likelihood approximations and ultimately distorting the RL objective. To overcome this limitation, we propose *Boundary-Guided Policy Optimization* (BGPO), a memory-efficient RL algorithm that maximizes a specially constructed lower bound of the ELBO-based objective. This lower bound is carefully designed to satisfy two key properties: (1) Linearity: it is formulated in a linear sum where each term depends only on a single MC sample, thereby enabling gradient accumulation across samples and ensuring constant memory usage; (2) Equivalence: Both the value and gradient of this lower bound are equal to those of the ELBO-based objective in on-policy training, making it also an effective approximation for the original RL objective. These properties allow BGPO to adopt a large MC sample size, resulting in more accurate likelihood approximations and improved RL objective estimation, which in turn leads to enhanced performance. Experiments show that BGPO significantly outperforms previous RL algorithms for dLLMs in math problem solving, code generation, and planning tasks.

1 Introduction

Recently, diffusion large language models (dLLMs) have emerged as promising alternatives to conventional autoregressive models (ARMs), demonstrating competitive performance across various language modeling tasks (Nie et al., 2025b; Ye et al., 2025; Gong et al., 2025b; Cheng et al., 2025). Unlike ARMs, which generate sequences in a left-to-right, token-by-token manner, dLLMs iteratively unmask tokens in parallel, offering the potential for significant inference acceleration (DeepMind, 2025; Inception Labs et al., 2025; Song et al., 2025; Wu et al., 2025). Despite these advancements, existing works primarily focus on pre-training and supervised fine-tuning of dLLMs, while leveraging reinforcement learning (RL) to further enhance dLLMs remains a challenging problem, even though RL has demonstrated significant efficacy in improving various capabilities of LLMs (OpenAI, 2024; DeepSeek-AI et al., 2025).

A key challenge in applying RL to dLLMs lies in the intractability of their likelihood functions (Zhu et al., 2025; Zhao et al., 2025a; Tang et al., 2025). Specifically, the iterative, non-sequential generation process precludes exact calculation of the log-likelihoods for generated responses (Zhu et al., 2025; Zhao et al., 2025a; Tang et al., 2025), which are essential for different RL algorithms (Schulman et al., 2017; Shao et al., 2024). In light of this, recent works have explored approximating the log-likelihoods by their evidence lower bounds (ELBOs) via customized Monte Carlo (MC) sampling (Zhu et al., 2025). While increasing the MC sample size can yield highly accurate approximations (Ho et al., 2020; Song et al., 2021), this approach incurs substantial memory overhead during RL training. In particular, computing the gradient of the non-linear functions in the RL objective necessitates storing the forward computational graphs for all MC samples, dramatically increasing memory consumption. As a result, practical implementations can only adopt relatively small sample sizes (e.g., $n_t = 4$ as illustrated in the left of Figure 1) due to hardware constraints, which directly amplifies errors in log-likelihood approximation

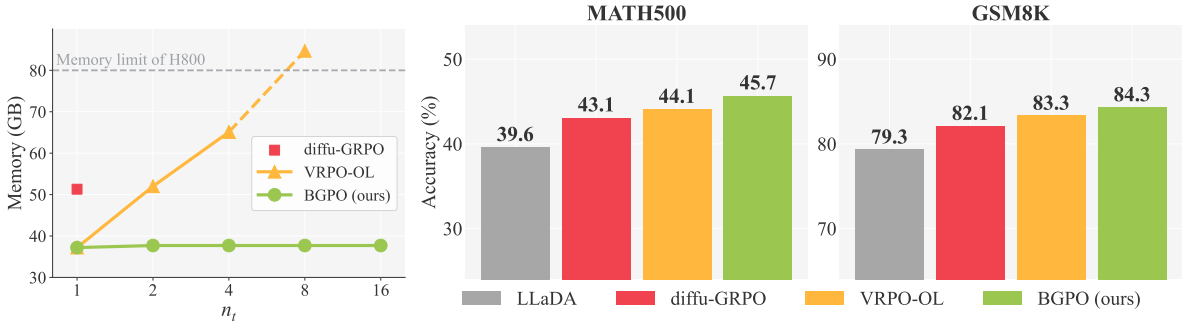

Figure 1: Left: Comparison of memory usage of previous ELBO-based RL method (VRPO-OL) and our BGPO using different Monte Carlo sample size n_t for the RL objective approximation. The max response length is set to 512. Middle and right: Performance of LLaDAs with different RL algorithms on mathematical tasks.

and introduces substantial bias and variance for the estimated objective and its gradients, ultimately degrading performance.

To address this limitation, we propose *Boundary-Guided Policy Optimization* (BGPO), a memory-efficient RL algorithm for dLLMs that supports large MC sample sizes for log-likelihood and RL objective approximation. Specifically, BGPO maximizes a constructed lower bound of the ELBO-based objective. This lower bound is carefully designed to satisfy two critical properties: (1) Linearity: it is formulated in a linear sum where each term associates with a single MC sample, thereby enabling gradient accumulation across samples and ensuring constant memory usage irrespective of sample size; (2) Equivalence: Both the value and gradient of the lower bound are equal to those of the ELBO-based objective in on-policy training, ensuring that the lower bound can also effectively approximate the original RL objective. These properties allow BGPO to adopt a large MC sample size to obtain a more accurate approximation for the RL objective, thereby achieving better performance.

To validate the effectiveness of BGPO, we conduct RL experiments with LLaDA-8B-Instruct on math problem solving, code generation, and planning tasks. The results show that BGPO significantly improves the performance of LLaDA-8B-Instruct across all tasks, and also outperforms previous RL algorithms for dLLMs. Further analysis demonstrates that increasing the MC sample size effectively reduces the bias and variance of gradients and improves model performance. Notably, BGPO achieves these improvements with only marginal increases in average training step time, despite its larger sample size.

In summary, our main contributions include: (1) We propose BGPO, a memory-efficient RL algorithm for dLLMs that supports large MC sample sizes in the approximation of log-likelihoods and the RL objective; (2) We theoretically prove the equivalence of the BGPO objective and the ELBO-based objective in on-policy training, demonstrating that BGPO also provides an effective approximation of the original RL objective; (3) Through comprehensive experiments, we validate the efficacy of BGPO and demonstrate the value of larger MC sample sizes in boosting model performance. We hope our work establishes a firm foundation for future research on RL for dLLMs.

2 Preliminary

2.1 Masked Diffusion Language Models

Masked dLLMs employ a non-autoregressive generation paradigm, generating text through progressive denoising. At their core lies a mask predictor p_θ (Austin et al., 2021a; Ou et al., 2025), which learns the data distribution through a forward-reverse framework. Starting from the original text at $t = 0$, the forward process gradually masks the input tokens until the sequence is fully masked at $t = 1$. Following LLaDA (Nie et al., 2025b), at time $t \in (0, 1)$, each token is replaced by the mask token $\mathbf{M}$ with probability t and remaining unmasked with probability $1 - t$. Conversely, the reverse process employs the mask predictor to recover this sequence by iteratively predicting the masked tokens as time reverses from 1 to 0. In conditional generation scenarios, the prompt x always remains unmasked, and the forward-reverse process is only applied to the response y.

2.2 Challenges of Applying RL to dLLMs.

Reinforcement learning (RL) has proved effective for improving language models, and the basic ob-

jective is to maximize:

$$\begin{aligned}\mathcal{J}(\theta) =&\mathbb{E}_{x\sim\mathcal{D},y\sim\pi_\theta(\cdot|x)}A(x,y)\\ =&\mathbb{E}_{x\sim\mathcal{D},y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\frac{\pi_\theta(y|x)}{\pi_{\theta_{\text{old}}}(y|x)}A(x,y),\\ =&\mathbb{E}_{x\sim\mathcal{D},y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\mathcal{R}(x,y),\end{aligned} \tag{1}$$

where π_θ, $\pi_{\theta_{\text{old}}}$, and $A(x,y)$ denote the current policy, old policy, and sequence-level advantage, respectively, and

$$\mathcal{R}(x,y) = e^{\log\pi_\theta(y|x)-\log\pi_{\theta_{\text{old}}}(y|x)}A(x,y). \tag{2}$$

However, applying RL to dLLMs is nontrivial, since the iterative denoising generation makes the exact computation of $\log\pi_\theta(y|x)$ intractable.

To address this, recent works have developed several methods to approximate $\log\pi_\theta(y|x)$. diffu-GRPO(Zhao et al., 2025a) adopts single-pass estimation, simply making $\log\pi_\theta(y|x) = \sum_{i=1}^{|y|}\log p_\theta(y^i|x')$, where y^i is the i-th token of y and x' is randomly masked prompt. Though efficient, it introduces notable bias relative to the exact policy. Alternatively, VRPO (Zhu et al., 2025) proposes to approximate $\log\pi_\theta(y|x)$ by using its evidence lower bound (ELBO):

$$B_{\pi_\theta}(y|x) \triangleq \mathbb{E}_{t\sim\mathcal{U}[0,1],y_t\sim q(\cdot|t,y,x)}\ell_{\pi_\theta}(y_t,t,y|x), \tag{3}$$

where $q(\cdot|t,y,x)$ denotes the forward masking process for the response y at time t, and

$$\ell_{\pi_\theta}(y_t,t,y|x) \triangleq \frac{1}{t}\sum_{i=1}^{|y_t|}\mathbf{1}[y_t^i=\mathbf{M}]\log p_\theta(y^i|y_t,x). \tag{4}$$

Specifically, they estimate $B_\pi(y|x)$ via customized Monte Carlo sampling:

$$\hat{B}_{\pi_\theta}(y|x)=\frac{1}{n_t}\sum_{j=1}^{n_t}\ell_{\pi_\theta}(y_{t^{(j)}},t^{(j)},y|x), \tag{5}$$

where $t^{(j)} \overset{\text{i.i.d.}}{\sim} \mathcal{U}[0,1]$ and $y_{t^{(j)}} \overset{\text{i.i.d.}}{\sim} q(\cdot|t^{(j)},y,x)$ are sampled timestamp and corresponding partially masked responses. Substituting $\hat{B}_{\pi_\theta}(y|x)$ into Eq. 1 yields an approximated RL objective:

$$\hat{\mathcal{J}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\hat{\mathcal{R}}(x,y), \tag{6}$$

where

$$\hat{\mathcal{R}}(x,y) = e^{\hat{B}_{\pi_\theta}(y|x)-\hat{B}_{\pi_{\text{old}}}(y|x)}A(x,y). \tag{7}$$

Notably, previous works have shown that when the sample size n_t is large enough, the bias of $\hat{B}_\pi(y|x)$ for a well-trained model relative to $\log\pi_\theta(y|x)$ will become negligible (Ho et al., 2020; Song et al., 2021). However, using a large n_t in training requires a huge amount of GPU memory: Each time $\hat{\mathcal{R}}(x,y)$ is computed, n_t forward passes of p_θ need to be executed (i.e., Eq. 4 and 5), and all the n_t computational graphs must be retained in the memory for calculating the gradient of the exponential function in Eq. 7. Therefore, in practice, the sample size can only remain small (e.g., $n_t = 4$), which results in inaccurate approximations for the likelihoods as well as the final objective, seriously affecting the final performance.

To break through this limitation, we propose *Boundary-Guided Policy Optimization* (BGPO), a memory-efficient RL algorithm for dLLMs that supports a large Monte Carlo sample size, thereby enabling more accurate approximations and achieving better performance. A detailed introduction is provided in the following sections.

3 BGPO

Following Zhu et al. (2025), our BGPO algorithm also uses the estimated ELBO $\hat{B}_{\pi_\theta}(y|x)$ to approximate $\log\pi_\theta(y|x)$. The main difference is that instead of directly maximizing the approximated objective $\hat{\mathcal{J}}(\theta)$, BGPO turns to maximize a constructed tight lower bound of $\hat{\mathcal{J}}(\theta)$:

$$\hat{\mathcal{J}}_{\text{lb}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\hat{\mathcal{R}}_{\text{lb}}(x,y), \tag{8}$$

where $\hat{\mathcal{R}}_{\text{lb}}(x,y)\leq\hat{\mathcal{R}}(x,y)$. Specifically, $\hat{\mathcal{R}}_{\text{lb}}(x,y)$ is carefully designed so that it satisfies the following two properties[1]:

- Linearity: $\hat{\mathcal{R}}_{\text{lb}}(x,y)$ is formulated as $\sum_{j=1}^{n_t}g_j$, where g_j is a function about the partially masked sample $y_{t^{(j)}}$ at time $t^{(j)}$. Therefore, we can backpropagate the gradient of g_j separately for each $y_{t^{(j)}}$ and update the policy after all backpropagations, so that the memory usage becomes irrelevant to the MC sample size n_t.
- Equivalence: In on-policy training (i.e., $\pi_{\theta_{\text{old}}} = \pi_\theta$), the value and gradient of $\hat{\mathcal{R}}_{\text{lb}}(x,y)$ are always equal to those of $\hat{\mathcal{R}}(x,y)$, making $\hat{\mathcal{J}}_{\text{lb}}(\theta)$ be equivalent to $\hat{\mathcal{J}}(\theta)$ and also an effective approximation for the original RL objective $\mathcal{J}(\theta)$.

These two properties allow BGPO to use a larger MC sample size in the likelihood approximation,

[1]For simplicity, we mainly discuss $\hat{\mathcal{R}}_{\text{lb}}$ and $\hat{\mathcal{R}}$ in this section, while all their properties can be directly applied to $\hat{\mathcal{J}}_{\text{lb}}$ and $\hat{\mathcal{J}}$, without influence by the expectation function.

which effectively reduces the bias and variance of $\hat{\mathcal{J}}_{\text{lb}}(\theta)$ and its gradient, leading to better performance. In the following, we will introduce the construction of $\hat{\mathcal{R}}_{\text{lb}}(x, y)$ in detail.

3.1 Linear Lower Bound Construction

The construction of $\hat{\mathcal{R}}_{\text{lb}}(x, y)$ is different based on the sign of the advantage $A(x, y)$:

- For $A(x, y) \geq 0$, we construct $\hat{\mathcal{R}}_{\text{lb}}(x, y)$ using Taylor expansion;
- For $A(x, y) < 0$, we construct $\hat{\mathcal{R}}_{\text{lb}}(x, y)$ using Jenson's inequality.

Lemma 1. ***[First-order Taylor Expansion]*** *For any $\delta \in \mathbb{R}$, the exponential function satisfies*

$$e^{\delta} \geq 1 + \delta.$$

When $A(x, y) \geq 0$, we apply the first-order Taylor expansion in Eq. 7, which yields:

$$\begin{aligned}
\hat{\mathcal{R}}(x, y) &= e^{\hat{B}_{\pi_\theta}(y|x) - \hat{B}_{\pi_{\text{old}}}(y|x)} A(x, y) \\
&= e^{\left(\frac{1}{n_t}\sum_{j=1}^{n_t} d_j\right)} A(x, y) \\
&\geq \left(1 + \frac{1}{n_t}\sum_{j=1}^{n_t} d_j\right) A(x, y) \\
&= \sum_{j=1}^{n_t} \frac{(1 + d_j)A(x, y)}{n_t}, \qquad (9)
\end{aligned}$$

where

$$d_j = \ell_{\pi_\theta}(y_{t^{(j)}}, t^{(j)}, y|x) - \ell_{\pi_{\theta_{\text{old}}}}(y_{t^{(j)}}, t^{(j)}, y|x). \qquad (10)$$

Lemma 2. ***[Jensen's Inequality]*** *For a convex function f and a finite set $\{x_i\}_{i=1}^n$, we have*

$$f\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) \leq \frac{1}{n}\sum_{i=1}^{n} f(x_i). \qquad (11)$$

When $A(x, y) < 0$, by applying Jensen's Inequality in Eq. 7, we have:

$$\begin{aligned}
\hat{\mathcal{R}}(x, y) &= e^{\left(\frac{1}{n_t}\sum_{j=1}^{n_t} d_j\right)} A(x, y) \\
&\geq \left(\frac{1}{n_t}\sum_{j=1}^{n_t} e^{d_j}\right) A(x, y) \\
&= \sum_{j=1}^{n_t} \frac{e^{d_j} A(x, y)}{n_t}. \qquad (12)
\end{aligned}$$

Putting everything together and letting:

$$g_j = \begin{cases} \frac{(1+d_j)A(x,y)}{n_t}, & \text{if } A(x, y) \geq 0; \\ \frac{e^{d_j} A(x,y)}{n_t}, & \text{if } A(x, y) < 0, \end{cases} \qquad (13)$$

$\hat{\mathcal{R}}_{\text{lb}}(x, y)$ is constructed as a linear sum of g_j:

$$\hat{\mathcal{R}}_{\text{lb}}(x, y) = \sum_{j=1}^{n_t} g_j. \qquad (14)$$

As shown in Algorithm 1, the linearity of $\hat{\mathcal{R}}_{\text{lb}}(x, y)$ (as well as $\hat{\mathcal{J}}_{\text{lb}}(\theta)$) enables us to separate the gradient backpropagations for each $y_{t^{(j)}}$, thus keeping the memory usage constant and allowing a larger sample size n_t.

3.2 Proof of Equivalence

In on-policy training where $\pi_\theta = \pi_{\theta_{\text{old}}}$, the value of ℓ_{π_θ} is equal to $\ell_{\pi_{\theta_{\text{old}}}}$, which means the value of d_j is always 0. By applying this in Eq. 7 and 14, we can find the values of $\hat{\mathcal{R}}_{\text{lb}}(x, y)$ and $\hat{\mathcal{R}}(x, y)$ are both equal to $A(x, y)$. Moreover, the gradient of $\hat{\mathcal{R}}_{\text{lb}}(x, y)$ is also the same as that of $\hat{\mathcal{R}}(x, y)$ when $d_j = 0$. Specifically, by applying the chain rule of the derivative, we have:

$$\begin{aligned}
\nabla_\theta \hat{\mathcal{R}}(x, y) &= \nabla_\theta \left(e^{\left(\frac{1}{n_t}\sum_{j=1}^{n_t} d_j\right)} A(x, y)\right) \\
&= e^{\left(\sum_{j=1}^{n_t} \frac{d_j}{n_t}\right)} \frac{A(x, y)\nabla_\theta \left(\sum_{j=1}^{n_t} d_j\right)}{n_t} \\
&\overset{d_j=0}{=} \sum_{j=1}^{n_t} \frac{A(x, y)\nabla_\theta d_j}{n_t}. \qquad (15)
\end{aligned}$$

Similarly, when $A(x, y) \geq 0$, we have:

$$\begin{aligned}
\nabla_\theta \hat{\mathcal{R}}_{\text{lb}}(x, y) &= \nabla_\theta \left(\sum_{i=1}^{n_t} \frac{(1 + d_j)A(x, y)}{n_t}\right) \\
&= \sum_{j=1}^{n_t} \frac{A(x, y)\nabla_\theta d_j}{n_t}, \qquad (16)
\end{aligned}$$

and when $A(x, y) < 0$, we have:

$$\begin{aligned}
\nabla_\theta \hat{\mathcal{R}}_{\text{lb}}(x, y) &= \nabla_\theta \left(\sum_{i=1}^{n_t} \frac{e^{d_j} A(x, y)}{n_t}\right) \\
&= \sum_{i=1}^{n_t} \frac{A(x, y)e^{d_j}\nabla_\theta d_j}{n_t} \\
&\overset{d_j=0}{=} \sum_{j=1}^{n_t} \frac{A(x, y)\nabla_\theta d_j}{n_t}. \qquad (17)
\end{aligned}$$

Algorithm 1 *BGPO*

Input: dataset $\mathcal{D}$; initial policy model π_θ; hyperparameters: G, n_t, η.

for iteration $= 1, 2, \ldots, M$ **do**
  Update the old policy $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$ and sample a batch $\mathcal{D}_b$ from $\mathcal{D}$
  for each prompt $x \in \mathcal{D}_b$ **do**
    Sample G response $\{y^{(i)}\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot|x^{(b)})$ and compute advantages $\{A(x, y^{(i)})\}_{i=1}^{G}$ using Eq. 18
    for $i = 1$ **to** G **do**
      Sample n_t timestamp $\{t^{(j)}\}_{j=1}^{n_t} \sim U[0, 1]$
      for $j = 1$ **to** n_t **do**
        Sample partially masked response $y^{(i)}_{t^{(j)}} \sim q(\cdot|t^{(j)}, y^{(i)}, x)$
        Compute g_j using Eq. 13 and let $\mathcal{L}_j \leftarrow -\frac{g_j}{G}$
        Backpropagate the gradient of $\mathcal{L}_j$ (▷ graident accumulation)
      end for
    end for
  end for
  Update the policy $\theta \leftarrow \theta - \eta \nabla_\theta$
end for

Output: π_θ

Therefore, $\hat{\mathcal{R}}_{\text{lb}}(x, y)$ and $\hat{\mathcal{R}}(x, y)$ (as well as $\hat{\mathcal{J}}_{\text{lb}}(\theta)$ and $\hat{\mathcal{J}}(\theta)$) are equivalent in terms of both value and gradient in on-policy training. This means like $\hat{\mathcal{J}}(\theta)$, $\hat{\mathcal{J}}_{\text{lb}}(\theta)$ is also an effective approximation of $\mathcal{J}(\theta)$, and using a large sample size n_t can reduce the bias and various of $\hat{\mathcal{J}}_{\text{lb}}(\theta)$ and its gradient, leading to better model performance.

3.3 Final Loss of BGPO

In practice, we adopt group-based advantage estimation. Specifically, for each prompt x, we sample G responses $y^{(1)}, \ldots, y^{(G)}$ from $\pi_{\theta_{\text{old}}}(\cdot|x)$. Let $r(x, y^{(i)})$ denotes the reward of $y^{(i)}$. The advantage of $y^{(i)}$ is defined as:

$$A(x, y^{(i)}) = \frac{r(x,y^{(i)}) - \text{mean}(\{r(x,y^{(j)})\}_{j=1}^{G})}{\text{std}(\{r(x,y^{(j)})\}_{j=1}^{G})}. \quad (18)$$

Accordingly, the loss for BGPO is formulated as:

$$\mathcal{L}_{\text{BGPO}} = -\mathbb{E}_{\substack{x\sim\mathcal{D}, \\ \{y^{(i)}\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)}} \left[\frac{1}{G} \sum_{i=1}^{G} \hat{\mathcal{R}}_{\text{lb}}(x, y^{(i)}) \right]. \quad (19)$$

Finally, we summarize our BGPO algorithm in Algorithm 1.

4 Experiment

In this section, we empirically validate the efficacy of BGPO through extensive RL experiments.

4.1 Setup

Models. We employ LLaDA-8B-Instruct (Nie et al., 2025b), a state-of-art dLLM that has undergone pre-training and supervised fine-tuning, as our initial policy model.

Datasets. We conduct RL experiments in three domains: math problem solving, code generation, and planning tasks (Ye et al., 2025). For math problem solving, we train the model on a mix of the training splits of MATH (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021), and evaluate on the respective test sets. For code generation, we use 16K medium-difficulty problems filtered from DeepCoder (Luo et al., 2025) as the training set, and adopt MBPP (Austin et al., 2021b) and HumanEval (Chen et al., 2021) as the test sets. For planning tasks, we train and evaluate on Countdown (Pan et al., 2025) and Sudoku (Arel, 2025), adopting the same training and test splits as d1 (Zhao et al., 2025a).

Implementation Details. We implement BGPO based on the VeRL (Sheng et al., 2025) framework. The maximum response lengths for math problem solving, coding generation, and planning tasks are set to be 512, 512, and 256, respectively. The batch size, rollout group size G, and learning rate are set to 16, 8, and 5×10^{-7}, respectively. The MC sample size n_t is set to 32 for Sudoku and 16 for other tasks. See Table 4 for more detailed hyperparameters. Following Zhao et al. (2025a), we evaluate the trained models (including baselines) every 20 steps and report results from the best-performing checkpoint. All experiments are conducted on 8 ×H800 GPUs.

Baselines. We mainly compare BGPO with two representative RL algorithms for dLLMs that are introduced in Section 2: (1) diffu-GRPO (Zhao et al., 2025a), which approximates the log-likelihoods with single-pass mean-field estimation; (2) VRPO-OL, which is the online version of VRPO (Zhu

Model	Mathematics		Coding		Planning	
	MATH500	GSM8K	HumanEval	MBPP	Sudoku	Countdown
Prior works with LLaDA						
d1-LLaDA (Zhao et al., 2025a)	40.2	82.1	-	-	16.7	32.0
wd1 (Tang et al., 2025)	39.0	82.3	-	-	25.2	46.1
LLaDA-IGPO (Zhao et al., 2025b)	42.8	83.6	-	-	-	-
LLaDA-1.5 (Zhu et al., 2025)	42.6	83.3	45.0*	40.0*	-	-
RL from LLaDA-8B-Instruct						
LLaDA-8B-Instruct (Nie et al., 2025b)	39.6	79.3	45.1	39.1	12.0	19.5
+ diffu-GRPO (Zhao et al., 2025a)	43.1	82.1	47.0	40.3	26.7	53.1
+ VRPO-OL (Zhu et al., 2025)	44.1	83.3	44.8	41.5	26.1	84.8
+ BGPO (ours)	**45.7**	**84.3**	**47.6**	**41.7**	**26.9**	**87.5**

Table 1: Performance comparison between BGPO and different baselines on mathematics, coding, and planning tasks. "*" indicates we re-evaluate the model using the same code environment.

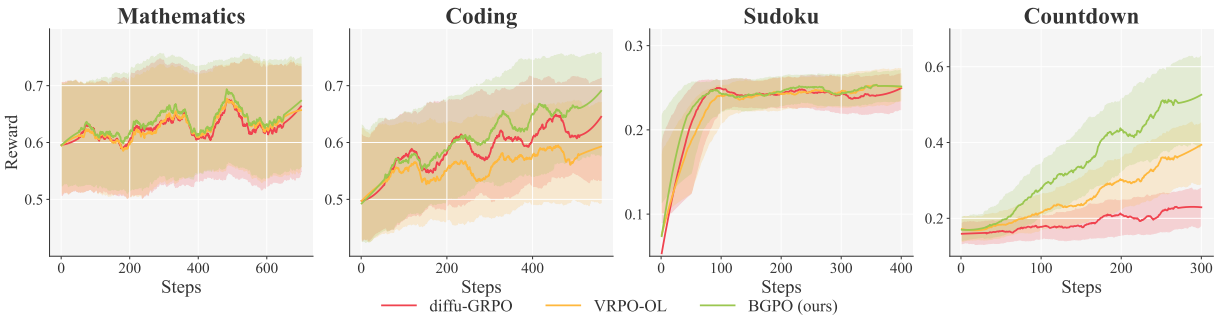

Figure 2: Training reward dynamics of BGPO, diffu-GRPO, and VRPO-OL across different tasks.

et al., 2025) that adopts ELBO-based likelihood approximation and uses the objective in Eq 6. We set the MC sampling sizes of VRPO-OL to the maximum that H800 can support, i.e., $n_t = 4$ for math and planning tasks and $n_t = 2$ for code generation, since the prompts of coding tasks are longer. Besides, we also present the results from several prior works as references, including d1 (Zhao et al., 2025a), wd1 (Tang et al., 2025), LLaDA-IGPO (Zhao et al., 2025b), and LLaDA 1.5 (Zhu et al., 2025), though their training setting are partially different from ours.

4.2 Main Results

Table 1 presents the performance of BGPO and different baselines on math problem solving, code generation, and planning tasks. As shown in the table, our BGPO algorithm achieves significant improvement over LLaDA-8B-Instruct, and also outperforms previous RL algorithms (i.e., diffu-GRPO and VRPO-OL) on all tasks, indicating that BGPO can produce a more accurate approximation of the RL objective compared to these baselines. Specifically, BGPO improves the performance of LLaDA-8B-Instruct by about 5.5% and 2.5% on mathematical and coding tasks, respectively, and dramatically improves the performance on Sudoku and Countdown by 14.9% and 68.0%. Moreover, the model trained with BGPO also outperforms all previous LLaDA-based models, such as d1 and LLaDA-1.5, achieving state-of-the-art results.

In addition, Figure 2 shows the reward dynamics of BGPO, diffu-GRPO, and VRPO-OL during the training on different tasks. The reward of BGPO is higher than the other two baselines in most steps. Particularly, BGPO exhibits a notably faster reward increase and significantly higher reward on the Countdown task, where the exploration space is relatively simple. These phenomena demonstrate that the larger MC sample size of BGPO brings a more accurate optimization direction, which aims to maximize the expectation of rewards.

4.3 Effect of Increasing MC Sample Sizes

To demonstrate the effect of increasing the MC sample size n_t in approximating the RL objective, we train LLaDA-8B-Instruct on math problem solving using BGPO with different n_t. As shown in

Model	MATH500	GSM8K
LLaDA-8B-Instruct	39.6	79.3
+ BGPO ($n_t = 1$)	43.5	83.5
+ BGPO ($n_t = 2$)	44.1	82.5
+ BGPO ($n_t = 4$)	43.7	82.7
+ BGPO ($n_t = 8$)	45.3	83.9
+ BGPO ($n_t = 16$)	**45.7**	**84.3**

Table 2: Performance of BGPO with different Monte Carlo sampling size n_t on mathematics benchmarks.

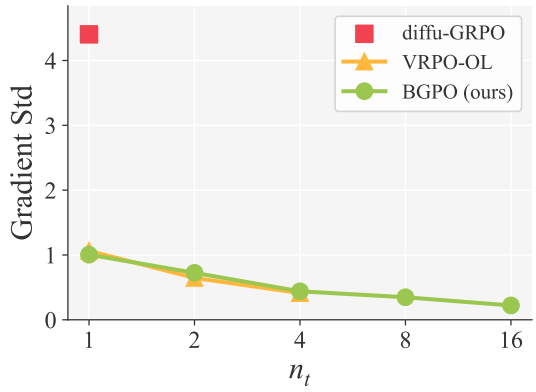

Figure 3: Standard deviation (std) of gradients of different RL algorithms with different MC sampling size n_t. The std is normalized by the absolute value of each parameter to unify the scale.

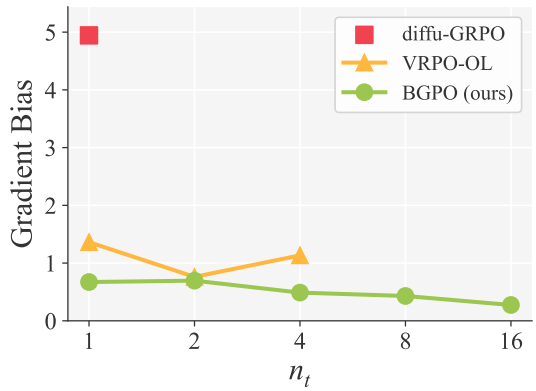

Figure 4: Gradient bias (normalized by the absolute value of parameter) with different MC sampling size n_t.

Table 2, the model performance consistently improves as n_t increases from 1 to 16, implying that larger MC sample sizes can produce more approximations of the RL objective.

To further illustrate this, we compare the standard deviation (i.e., root of variance) and bias of the loss gradients of different RL algorithms with different n_t [2]. Specifically, we compute the gradient of a batch 8 times with different MC sampling sizes, and then calculate the standard deviation for each parameter, normalized by the absolute value of the parameter to unify the scale. For the bias calculation, we use the gradient of BGPO with $n_t = 256$ to simulate the golden gradient, and also normalize the bias of parameters by their absolute value. As shown in Figure 3 and 4, the gradient variance and bias of diffu-GRPO are quite large, since it adopts single-pass estimation and also partially masks the prompt. In contrast, the gradient variance and bias of VRPO-OL and our BGPO gradually decrease as the MC sampling size n_t increases, and BGPO can even obtain a smaller variance and bias by using a larger n_t, since the memory overhead of BGPO remains constant, regardless of n_t. This enables BGPO to have a more accurate optimization direction and more stable training, resulting in better model performance.

4.4 Out-of-domain Performance

To evaluate the out-of-domain generalization capability of BGPO, we train the models on math and coding tasks, respectively, and evaluate them on other tasks. As presented in Table 3, the model trained on math tasks improve the performance on the planning tasks, and the model trained on coding tasks achieves improvement on both math and planning tasks, demonstrating the good generalizability of BGPO.

4.5 Training Speed Comparison

A potential concern for BGPO is that the large MC sample size may substantially increase the running time of each RL step, affecting the training efficiency. To allay this concern, we compare the averaged training step time of BGPO and baseline algorithms on math problem solving, with the maximum response length to be 512. As shown in Figure 5, even BGPO adopts a much larger MC sample size (i.e., $4\times$ of VRPO-OL), its averaged training step time only slightly increased. This is because the dominant time cost of each step lies in the response rollout phase (i.e., sampling G responses for each prompt) rather than the objective computation and policy updating phases.

5 Related Work

5.1 Diffusion Large Language Models

Diffusion large language models (dLLMs), which generate text through masked diffusion (Austin

[2] We do not directly compare the variance and bias of the loss since the value of loss is always 0 in on-policy training.

Model	Mathematics		Coding		Planning	
	MATH500	GSM8K	HumanEval	MBPP	Sudoku	Countdown
LLaDA-8B-Instruct	39.6	79.3	45.1	39.1	6.3	14.5
+BGPO (train on math tasks)	45.7	84.3	44.2	38.6	8.6	21.1
+BGPO (train on coding tasks)	40.8	80.4	47.6	41.7	9.2	21.5

Table 3: Out-of-domain performance of BGPO. The in-domain results are in gray.

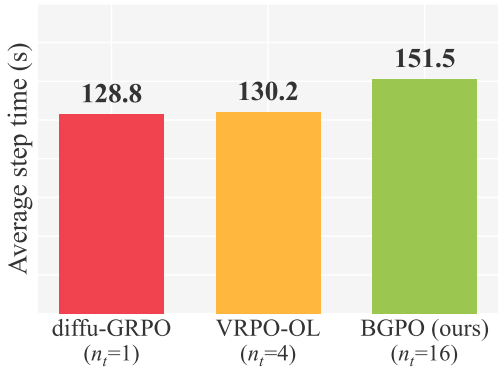

Figure 5: Training speed comparison between baselines.

et al., 2021a; Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2025; Nie et al., 2025a), have recently achieved significant advances, demonstrating performance comparable to similarly-sized autoregressive models. Among existing open-source dLLMs, DiffuLLaMA (Gong et al., 2025a), Dream (Ye et al., 2025), and SDAR (Cheng et al., 2025) are adapted from pre-trained autoregressive LLMs, while LLaDA (Nie et al., 2025b) is trained from scratch using bidirectional attention by maximizing the ELBOs of log-likelihoods, presenting a complete process of pre-training and supervised fine-tuning of dLLMs. Moreover, several commercial dLLMs like Mercury (Inception Labs et al., 2025), Gemini Diffusion (DeepMind, 2025), and Seed Diffusion (Song et al., 2025) not only achieve leading performance in code generation but also offer significantly faster inference, demonstrating the practical viability of dLLMs and their promising alternative to autoregressive LLMs.

5.2 Reinforcement Learning for dLLMs

Applying RL to dLLMs presents unique challenges compared to autoregressive models. The iterative, non-sequential generation process of dLLMs makes their likelihood functions intractable, necessitating the approximation of log-likelihoods for policy optimization. For instance, d1 (Zhao et al., 2025a) proposed diffu-GRPO, which approximates the log-likelihoods of dLLMs through single-pass mean-field estimation. Following wd1 (Tang et al., 2025; Zhao et al., 2025b) and IGPO (Zhao et al., 2025b) also adopt this approximation approach. Though efficient, the single-pass estimation introduces notable bias relative to the exact likelihoods. Alternatively, VRPO (Zhu et al., 2025) in LLaDA 1.5 approximates the log-likelihoods by their ELBOs, which is estimated via Monte Carlo (MC) sampling. Theoretically, this method can produce highly accurate approximations by using a large MC sample size. However, the practical sample size used in training is severely constrained by the GPU memory limit, since the computational graphs of all samples need to be retained for the gradient calculation of the non-linear function in the RL objective. While our BGPO algorithm addresses this memory-inefficiency limitation and supports large MC sample sizes, thereby effectively reducing the bias and variance of approximations and achieving better performance.

6 Conclusion

In this work, we propose BGPO, a memory-efficient RL algorithm for dLLMs that supports a large Monte Carlo sample size for approximating the sequence-level log-likelihoods and the final objective, thereby effectively reducing the bias and variance of approximations and leading to better model performance. We theoretically prove the equivalence of our BGPO objective and the previous ELBO-based objective, and conduct extensive experiments to validate the efficacy of BGPO. We hope that our work lays a solid foundation for future research on RL of dLLMs.

7 Limitations

In this work, we only conduct experiments on 8B-level models, since there are no larger open-source dLLMs, and our computational resources are also limited. Nonetheless, we believe our BGPO algorithm can be well applied to larger dLLMs due to

its solid theoretical foundation.

8 Ethical Considerations

All the models and datasets used in this work are publicly published with permissible licenses.

References

Arel. 2025. Arel's sudoku generator. https://www.ocf.berkeley.edu/~arel/sudoku/main.html. Accessed: 2025-09-23.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021a. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 17981–17993.

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. 2021b. Program synthesis with large language models. *CoRR*, abs/2108.07732.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé 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, and 39 others. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Wenghai Wang, Qipeng Guo, Kai Chen, Biqing Qi*, and Bowen Zhou. 2025. Sdar: A synergistic diffusion–autoregression paradigm for scalable sequence generation.

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. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

DeepMind. 2025. Gemini diffusion.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 81 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. 2025a. Scaling diffusion language models via adaptation from autoregressive models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

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. 2025. Mercury: Ultra-fast language models based on diffusion.

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. 2025. Deepcoder: A fully open-source 14b coder at o3-mini level. https://pretty-radio-b75.notion.site/DeepCoder-A-Fully-Open-Source-14B-Coder-at-O3-mini-Leve Notion Blog.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2025a. Scaling up masked diffusion models on text. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

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

OpenAI. 2024. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/. Accessed: 2025-05-07.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. 2025. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

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

Subham S. Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems, EuroSys 2025, Rotterdam, The Netherlands, 30 March 2025 - 3 April 2025*, pages 1279–1297. ACM.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. 2024. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

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, and 3 others. 2025. Seed diffusion: A large-scale diffusion language model with high-speed inference. *CoRR*, abs/2508.02193.

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

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

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

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

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

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

Task	Response length	Diffusion step	Block size	MC sample size n_t		
				diffu-GRPO	VRPO-OL	BGPO
Mathematics	512 / 512*	256 / 512*	32 / 32*	1	4	16
Coding	512 / 512*	512 / 512*	32 / 32*	1	2	16
Sudoku	256 / 256*	128 / 256*	32 / 32*	1	4	32
Countdown	256 / 256*	128 / 256*	32 / 32*	1	4	16

Table 4: Detailed hyperparameters for different tasks. "*" denotes the different hyperparameters used in evaluation.

A Detailed hyperparameters

We present detailed hyperparameters of BGPO on different tasks in Table 4. Following previous works, we adopt a block-wise decoding strategy in both training and evaluation. The choices of response length, diffusion step, and block size also follow Zhu et al. (2025) and Zhao et al. (2025a) for obtaining the best performance.