GDSD: Reinforcement Learning as Guided Denoiser Self-Distillation for Diffusion Language Models

Xiaohang Tang[*]
UCL Dept. of Statistical Science
UCL Centre for AI

Keyue Jiang[*]
Alibaba Group
UCL Centre for AI & Dept. of EEE

Che Liu
Imperial College London

Qifang Zhao
Alibaba Group

Xiaoxiao Xu
Alibaba Group

Sangwoong Yoon[†]
UNIST

Ilija Bogunovic[†]
UCL Centre for AI
University of Basel

Abstract

Reinforcement learning (RL) can be used to improve the policy (denoiser) of diffusion large language models (dLLMs), while being hindered by the intractability of the policy likelihood. A dominant and efficient family of methods replaces the likelihood in standard RL with its evidence lower bound (ELBO), estimated from randomly masked sequences. Despite being well aligned with pre-training, these approaches introduce bias through training–inference mismatch by using the ELBO as a likelihood surrogate, which can degrade performance. In this work, we propose **Guided Denoiser Self-Distillation (GDSD)** to directly distill the denoiser of dLLMs from an advantage-guided self-teacher, derived from the closed-form optimum of reverse-KL regularized RL. GDSD matches the dLLM's denoiser logits to the teacher's via a normalization-free objective, which reduces RL to likelihood-free self-distillation and thus bypasses the TIM biases. Recent ELBO-based methods emerge as instances of applying different distillation divergences, but with diagnosable pathologies that GDSD avoids. On planning, math, and coding benchmarks with LLaDA-8B and Dream-7B, GDSD consistently outperforms prior state-of-the-art ELBO-based methods with a more stable training reward dynamics, achieving test-accuracy improvements of up to $+19.6\%$. These results suggest that direct denoiser self-distillation, without relying on an ELBO likelihood surrogate, can provide a more stable and effective RL procedure for dLLMs. Code is available at https://github.com/GaryBall/GDSD.

1 Introduction

Diffusion Large Language Models (dLLMs) have emerged as efficient alternatives to autoregressive models (ARMs). dLLMs generate multiple tokens in a single decoding step and do not follow a strictly left-to-right generation order, thereby improving generation efficiency and unlocking token dependencies beyond the left-to-right paradigm in ARMs. This promise is also reflected in several recent releases, which have scaled model size while substantially improving inference efficiency. Open-weight reasoning dLLMs have grown from 8B-parameter models [34, 72] to 100B parameters in LLaDA 2.0 [6], with inference reported to be more than $3\times$ faster than that of even smaller ARMs

[*]Equal Contribution, [†]Corresponding Authors.

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

[7]. Closed models such as Mercury similarly report speedups of up to $10\times$ relative to ARMs [25]. Despite these efficiency gains, dLLMs still lag behind the state-of-the-art ARMs in generation quality, highlighting the need for effective fine-tuning methods, such as reinforcement learning (RL).

The central obstacle to RL with dLLMs is that the policy likelihood *is intractable*. Two families of methods have emerged as strong-performing solutions. RL based on **trajectory likelihood** estimates the exact likelihood of the trajectory of a sequence in the reverse process by accumulating the transition probabilities [21, 57, 10, 54]. While unbiased in principle, these methods incur substantial computational cost per gradient step, being misaligned with the pre-training objectives. RL based on **sequence likelihood** applies likelihood evidence lower bound (ELBO) as a surrogate, estimated from randomly masked sequences [67, 63, 72, 51, 35, 55]. This approach is computationally efficient, and naturally aligned with the objective of current dLLMs pre-training, which is itself also defined on a randomly masked sequence. Thus, we focus on the sequence likelihood family in this work.

RL based on sequence likelihood with ELBO surrogate is inherently off-policy, yet existing methods attempt to correct it with importance sampling ratio based on ELBO. These methods naively plug the ELBO surrogate into policy gradient or PPO-style objectives in the place of true likelihood, introducing a well-known bias called training-inference mismatch (TIM) [40, 27]. First, the non-negligible gap between the ELBO estimate and the true likelihood [24] biases the importance ratio, which has been shown to degrade performance [24], and can even cause training collapse [70]. Additionally, dLLM rollouts are produced by iterative re-masking and block-wise decoders [64, 34, 2], whose induced sampling distribution differs from the training policy (i.e., ELBO). The importance-sampling formulation requires a tractable training-sampling relationship that ELBO does not provide, which motivates a different formulation.

To address the TIM bias, we shift from importance-sampling RL with the ELBO surrogate to direct, off-policy self-distillation on the denoiser. To our knowledge, this is the first principled method recasting online RL for dLLMs as denoiser self-distillation to avoid the likelihood-ratio TIM bias by design. Our contributions are:

- We reduce RL with dLLMs to denoiser self-distillation: under reverse-KL regularized RL, the closed-form optimal policy induces a guided denoising distribution, an advantage guided denoiser as self-teacher, which we aim to distill at each RL step. (Section 3.1)
- We propose **Guided Denoiser Self-Distillation (GDSD)**, a squared-logit distillation loss with a normalization-free reformulation that eliminates the partition function via logit centralization. Distillation allows for off-policy updates without likelihood ratios, thus bypassing the likelihood-ratio TIM bias of ELBO-based methods (Sections 3.2, 3.3)
- We provide insights on how existing ELBO-based methods are connected to alternative divergence choices under our distillation framework. We demonstrate that our squared-logit instance avoids both the data inefficiency of weighted ELBO instances (wd1 [51], DMPO [73]) and the TIM bias of policy gradient instances (SPG [55], UniGRPO [63], ESPO [35]). (Section 4)
- In experiments compared to prior state-of-the-art ELBO-based methods, GDSD shows more stable training reward dynamics. On planning tasks, GDSD with Dream-7B achieves test-accuracy gains of up to $+19.6\%$. With LLaDA-8B, GDSD also consistently improves performance across planning, math, and coding benchmarks, with gains ranging from $+0.6\%$ to $+5\%$. (Section 5)

2 Preliminaries

In this section, we introduce the formulation of current diffusion large language models (dLLMs), namely masked diffusion models, and their reinforcement learning formulation.

2.1 Masked Diffusion Models

Masked Diffusion Models (MDMs) [43] represent a prominent class of non-autoregressive generative models that operate over a discrete state space. Let $\mathcal{V}$ denote the categorical vocabulary and $[\mathrm{M}]$ a dedicated mask token, such that the augmented space is $\mathcal{V}' = \mathcal{V} \cup \{[\mathrm{M}]\}$.

For a *clean sequence* with length N, denoted by $x_0 \in \mathcal{V}^N$, the forward process $q(x_t|x_0)$ defines a continuous-time transition for $t \in (0, 1]$, where *masked sequence* $x_t \in {\mathcal{V}'}^N$. Forward transition from state x_s to x_t $(0 \le s < t)$ is formulated as: $q(x_t|x_s) = \mathrm{Cat}(x_t; Q(s,t)^\top x_s)$ where $Q(s,t)$ is the

transition kernel. The MDM, denoted by p_θ, is then trained to model the reverse process. In particular, we term the reverse model for clean sequence prediction (i.e. $p(x_0|x_t)$) as *denoiser* in this work.

Let $x_0^{(n)}$ denote the n-th token of the clean sequence, open large-scale MDMs, such as LLaDA-8B [34] and Dream-7B [64], directly model the token-level denoising distribution $p_\theta(x_0^{(n)}|x_t)$, and are pre-trained by *maximizing* the following negative cross-entropy objective [36, 48]:

$$L(x_0^{(n)}; p_\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\ x_t\sim q(x_t|x_0)}\left[w(t)\cdot \mathbf{1}(x_t^{(n)} = [\mathrm{M}])\log p_\theta(x_0^{(n)}|x_t)\right]. \tag{1}$$

The sequence-level objective is obtained by accumulating the token-level objective, specifically $L(x_0; p_\theta) = \sum_{n=1}^{N} L(x_0^{(n)}; p_\theta)$, which is equivalent to the likelihood evidence lower bound (ELBO). Equation (1), the token-level contribution to the ELBO [35], has also been used to approximate the token-level likelihood in reinforcement learning (RL) [63, 17].

2.2 RL for Masked Diffusion Models

The goal of RL is to maximize a reward function r, which may be either a verifiable reward [47] or the output of a pre-trained reward model [37]. Let π denote policy, being the conditional probability of generating clean completion x_0 conditioned on prompt c [2]. Let $A(x_0)$ denote the advantage function, abbreviated as A. Advantage can be estimated with rewards: $A(x_0) = r(x_0) - \mathbb{E}_{\pi_{\text{old}}}[r(x_0)]$ [47, 29]. The policy gradient (PG) and Proximal Policy Optimization (PPO) objectives are defined as

$$\mathcal{L}_{\text{PG}} = \mathbb{E}_{x_0\sim\pi_{\text{old}}}\Big[\frac{\pi_\theta(x_0)}{\pi_{\text{old}}(x_0)}A\Big],\ \mathcal{L}_{\text{PPO}} = \mathbb{E}_{x_0\sim\pi_{\text{old}}}\left[\sum_{n=1}^{N}\min\Big(\frac{\pi_\theta(x_0^{(n)})}{\pi_{\text{old}}(x_0^{(n)})}A, \text{clip}(\frac{\pi_\theta(x_0^{(n)})}{\pi_{\text{old}}(x_0^{(n)})}, 1\pm\epsilon)A\Big)\right].$$

However, in dLLMs, both the token-level likelihood $\pi_\theta(x_0^{(n)})$ and the sequence-level likelihood $\pi_\theta(x_0)$ are intractable, making the direct application of RL difficult. Zhao et al. [67] propose a simplified likelihood approximation by fully masking the completion meanwhile randomly masking the prompt, but this has been shown to underperform ELBO-based methods [35].

2.3 RL with Likelihood Surrogate ELBO

A widely adopted family of methods estimates ELBO (Equation (1)) and uses it as a surrogate data likelihood for RL [17, 63, 35, 7]. In each RL step, after sampling a clean sequence x_0, t proportion of tokens are randomly selected to mask for computing an ELBO estimate. There are various advantages of these methods.

- First, ELBO estimation is computationally *efficient*. The masking operation itself is extremely cheap, requiring only random token selection. The main computational cost comes from model inference on a fixed number (Monte-Carlo sample size) of masked sequences. In practice, however, RL remains effective even with a small sample size (e.g. 4) [63, 35].
- Moreover, ELBO objective is well *aligned with the pre-training*, since both objectives rely on randomly masked sequence (Section 2.1). This makes RL using a randomly masked sequence a principled and widely adopted choice.

Despite these strengths, RL with ELBO surrogates introduces non-negligible bias that can potentially cause training collapse, which we diagnose in the next section.

2.4 Bias in ELBO-based Objectives

In RL based on sequence likelihood, ELBO surrogate and the special decoding process of dLLMs jointly introduce a mismatch between the policy optimized during training and the policy used at inference, termed *Training-Inference Mismatch (TIM)* [40]. First, the ELBO is a lower bound of the true likelihood with a non-negligible gap [24] and estimation error, which leads to potentially large bias and variance in the likelihood ratio [51, 70], further degrade performance [24] and even cause training collapse [70]. Second, prevailing dLLM decoders are iterative re-masking and block-wise, producing a sampling distribution differs from the training policy.

[2] In the rest of this paper, we omit the prompt notation c in policy to abbreviate $\pi(x_0|c)$ as $\pi(x_0)$ for simplicity.

Formally, RL methods such as policy gradient (PG) leverage the importance-sampling ratio $\pi_\theta(x_0)/\pi_{\text{old}}(x_0)$ for importance sampling following distribution $\pi_{\text{old}}(x_0)$. But current methods replace marginal likelihood with ELBO: $\hat{\pi}_{\text{old}}(x_0) = L(x_0; p_{\text{old}})$. The sampling process discretizes the continuous reverse process into K steps $\{t_1, \cdots, t_K\} \in [0, 1]$, inducing a sampling distribution:

$$\pi^{\text{rm}}_{\text{old}}(x_0) = \prod_{k=1}^{K} \underbrace{m(\sigma_k \mid x_{t_k}; p_{\text{old}})}_{\text{remasked token selection}} \cdot \prod_{n \in \sigma_k^c} \underbrace{p_{\text{old}}(x_0^{(n)} \mid x_{t_k})}_{\text{token prediction}}, \tag{2}$$

where m is the selection policy to decide which tokens to remask, such as low-confidence selection, and σ_k^c denotes the set of unmasked tokens, the complement of the predetermined remasking set σ_k. This leads to a clear mismatch $\pi^{\text{rm}}_{\text{old}}(x_0) \neq \hat{\pi}_{\text{old}}(x_0)$, further causing the likelihood-ratio correction ineffective and objective bias in ELBO-based methods that are based on PG or PPO [55, 35]:

$$\mathcal{L}_{\text{PG}} = \mathbb{E}_{x_0 \sim \pi^{\text{rm}}_{\text{old}}}\Big[\frac{\hat{\pi}_\theta(x_0)}{\hat{\pi}_{\text{old}}(x_0)} A\Big], \; \mathcal{L}_{\text{PPO}} = \mathbb{E}_{x_0 \sim \pi^{\text{rm}}_{\text{old}}}\Bigg[\sum_{n=1}^{N} \min\Big(\frac{\hat{\pi}_\theta(x_0^{(n)})}{\hat{\pi}_{\text{old}}(x_0^{(n)})} A, \text{clip}\big(\frac{\hat{\pi}_\theta(x_0^{(n)})}{\hat{\pi}_{\text{old}}(x_0^{(n)})}, 1 \pm \epsilon\big) A\Big)\Bigg].$$

Such TIM bias, including even those arising from numerical rounding errors, has been shown to lead to catastrophic training collapse [70, 40, 27]. To address this issue, we propose reducing the RL to an equivalent off-policy self-distillation problem without relying on likelihood.

3 Method

3.1 RL as Guided Denoising

Prominent RL methods such as PPO and GRPO [45, 50, 47] restrict the policy update through a clipping operator on the importance ratio, which is well-suited to autoregressive policies whose likelihoods are tractable. For dLLMs, since likelihood is intractable, we instead adopt a reverse-KL penalty, which likewise preserves the monotonic improvement property [51]:

$$\max_{\pi} \; \mathbb{E}_{x_0 \sim \pi}[\psi A(x_0) - D_{\text{KL}}(\pi \| \pi_{\text{old}}) - \beta D_{\text{KL}}(\pi \,\|\, \pi_{\text{ref}})], \tag{3}$$

where ψ is the guidance coefficient, β is the regularization coefficient, and π_{ref} is the (frozen) reference model. Using the method of Lagrange multipliers, the objective has a closed-form solution π^*, satisfying for any output clean sequence x_0:

$$\pi^*(x_0) \propto \pi_{\text{old}}(x_0)^{(1-\beta)} \cdot \pi_{\text{ref}}(x_0)^{\beta} \cdot \exp\big(\psi A(x_0)\big). \tag{4}$$

Here π^* denotes the optimal policy of the reverse-KL regularized RL problem, a marginal distribution over clean sequences. In this work, we aim to learn a masked diffusion model (MDM) whose induced sampling distribution matches π^*. To achieve this, we first restrict the *training-time* forward process:

Assumption 3.1 (Consistent Forward Process)**.** We apply the same forward process masking on all the diffusion models (policies including reference model, old model, and current parametrized model). For simplicity, we let $q(x_t|x_0)$ denote the forward process.

Assumption 3.1 genuinely reflects the practice of likelihood approximation with randomly masked sequences, i.e., obtaining the same batch of masked sequences to compute the ELBO for all models (p_θ, p_{old}, p_{ref}) [63, 35]. Based on a consistent forward process, the RL policy update can then be converted into guided sampling:

Lemma 3.2 (**Energy-Guided Denoising Distribution** [30, 51])**.** *Based on consistent masking in Assumption 3.1, the closed-form solution to reverse-KL regularized RL in Equation (4) induces an energy-guided MDM p^*, which we denote as a **teacher** denoiser, satisfying $\forall x_0, t, x_t$:*

$$p^*(x_0|x_t) = p^{ref}_{old}(x_0|x_t) \cdot \exp(\psi A(x_0) - A_t(x_t)), \tag{5}$$

where $p^{ref}_{old}(x_0|x_t) \propto p_{old}(x_0|x_t)^{1-\beta} p_{ref}(x_0|x_t)^{\beta}$, the negative energy guidance $A(\cdot)$ is the advantage function, and the log-normalization-constant $A_t(x_t) = \log \mathbb{E}_{x_0 \sim p^{ref}_{old}(\cdot|x_t)}[\exp\big(\psi A(x_0)\big)]$.

In particular, Equation (5) shows that the teacher denoiser p^* is a MDM with base model $p^{\text{ref}}_{\text{old}}$, guided by the energy function $\mathcal{E}(x_0) = -\psi A(x_0)$. Consequently, we can directly distill from the guided self-teacher p^*, and thus reduce the RL policy optimization problem to self-distillation to bypass the bias in ELBO-based methods.[3]

[3]We use *self-distillation* in the sense that the teacher is built entirely from the model's own frozen (old and reference) logits, shifted by the advantages of its own rollouts, without any external teacher model, and that the student is trained by

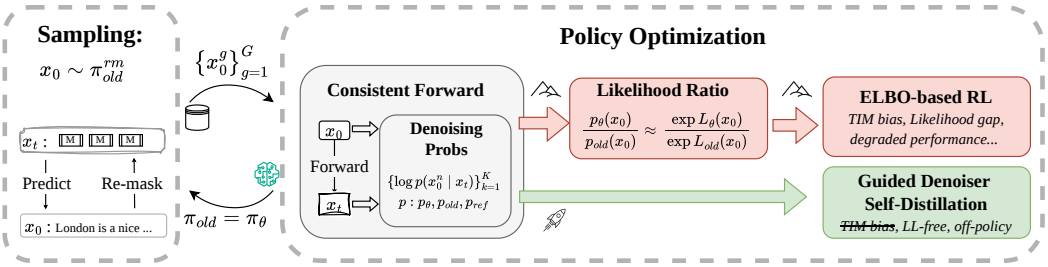


Figure 1: Overview of reinforcement learning for diffusion Large Language Models (dLLMs). Prior methods are based on ELBO $L_\theta(x_0)$ to approximate the likelihood ratio, which introduces training-inference-mismatch (TIM) bias. Our method Guided Denoiser Self-Distillation conducts denoiser self-distillation directly, which bypasses likelihood computation and TIM bias.

3.2 Guided Denoiser Self-Distillation

To directly approximate the guided denoising distribution p^* for masked diffusion models (MDMs), we propose **Guided Denoiser Self-Distillation (GDSD)**, a framework that trains p_θ by distilling from the teacher energy-guided denoiser p^*. Concretely, GDSD minimizes the distance between the parametrized denoiser $p_\theta(x_0|x_t)$ and the teacher distribution $p^*(x_0|x_t)$ via logit matching [20]:

Guided Denoiser Self-Distillation (GDSD)

$$\mathbb{E}_{t\sim U(0,1),\ x_0\sim\pi_{\text{old}},\ x_t\sim q(\cdot|x_0)}\Big[\log p_\theta(x_0|x_t) - \underbrace{\big(\log p_{\text{old}}^{\text{ref}}(x_0|x_t) + \psi A(x_0) - A_t(x_t)\big)}_{\log p^*(x_0|x_t)}\Big]^2. \quad (6)$$

The sequence-level denoising distribution factorizes into token-level denoising models [49, 67, 63, 51]: $\log p_\theta(x_0|x_t) = \sum_{n=1}^N \log p_\theta(x_0^{(n)}|x_t)$, where $p_\theta(x_0^{(n)}|x_t)$ is the token-level denoiser of the current dLLMs. The advantage function A is estimated from rewards. The main challenge in directly optimizing Equation (6) is the intractable normalization constant A_t, which we eliminate in Section C.2 with normalization-free reformulation. In the rest of this section, we highlight a few advantages of GDSD.

RL via Denoiser Self-Distillation. GDSD retains the main advantages of ELBO-based methods that leverage randomly masked sequences: computational efficiency and natural alignment with the pre-training objective. Unlike prior approaches, however, GDSD eliminates the need for ELBO due to its likelihood-free formulation. The objective leverages off-policy samples, sampled through iterative self-refinement, and distills the dLLM denoiser from a teacher, obtained from dLLM's old logits shifted by the advantage. Such guided self-distillation can increase the denoising probability of clean samples with a positive advantage, and decrease it for those with a negative advantage (i.e., negative samples). As a result, GDSD naturally handles negative samples without requiring additional designs, in contrast to prior works [51, 55].

Bypassing TIM Bias. GDSD does not form a likelihood ratio in its objective, so the two structural biases of importance-sampling (Section 2.3), including the ELBO–likelihood gap and the re-masking sampler–policy gap, never enter the loss objective by design. Distillation originally allows for training on off-policy samples, which absorbs the potential harms by the special process of dLLMs decoding.

Off-Policy Sampling and Coverage. Equation (6) is a regression on samples from π_{old}. Under realizability and sufficient coverage, its population minimizer is $p_\theta = p^*$ irrespective of the sampling distribution, and low-probability samples receive small weights rather than exploding inverse-probability weights $1/\hat{\pi}_{\text{old}}(x_0)$. With finite samples, however, poorly covered regions are fitted poorly and the loss estimate can have high variance when p^* deviates strongly from π_{old}. We control this gap via ψ, by refreshing π_{old} every μ steps, and by variance reduction (Section C.2); empirically, $\mu = 1$ gives significantly lower Sudoku accuracy than $\mu = 8$ with LLaDA-8B.

logit matching [20]. This differs from recent self-distillation methods for autoregressive LLMs, whose teacher is the model conditioned on privileged information [68, 22, 62]; our goal is instead to bypass the likelihood ratio in dLLMs. Incorporating privileged information into the teacher of GDSD is a natural extension.

Minor Changes to RL Pipeline. GDSD can be implemented with only minor modifications to existing RL pipelines. We summarize the standard RL pipeline for dLLMs in Figure 1. In the sampling stage, we follow existing dLLM RL workflows (e.g., Diffu-GRPO [67]), using π_{old} to generate a group of sequences through iterative prediction and re-masking. We then follow the standard sequence-likelihood computing procedure: we randomly sample several time steps t and apply a consistent forward process by drawing $x_t \sim q(\cdot|x_0)$. At each sampled step, we compute the denoising log-probabilities $\{\log p(x_0^k|x_t)\}_{k=1}^K$ under all relevant policies, namely π_{old}, π_{ref}, and π_θ. ELBO-based methods use these denoising probabilities to estimate the ELBO and then substitute it for the likelihood in the RL objective. By contrast, GDSD is likelihood-free and computes its loss directly from the collected denoising probabilities.

3.3 Normalization-Free Optimization

The main challenge in implementing GDSD in Equation (6) is computing the normalization constant in the teacher denoiser p^*. We address this by firstly decomposing the geometric mixture model $p_{\text{old}}^{\text{ref}}$ into p_{old} and p_{ref} , and gather the normalization term of mixture policy $p_{\text{old}}^{\text{ref}}$ and A_t into a single constant Z_t:

$$\log p^*(x_0|x_t) = (1-\beta)\log p_{\text{old}}(x_0|x_t) + \beta \log p_{\text{ref}}(x_0|x_t) + \psi A(x_0) - \log Z_t(x_t), \tag{7}$$

where Z_t is the partition function, which requires a summation over the exponential completion space $\mathcal{X}$:

$$Z_t(x_t) = \sum_{x_0 \in \mathcal{X}} p_{\text{old}}(x_0|x_t)^{(1-\beta)} p_{\text{ref}}(x_0|x_t)^{\beta} \exp\big(\psi A(x_0)\big).$$

Estimating Z_t requires drawing additional samples of x_0 (e.g., sampling from $p_{\text{old}}(x_0|x_t)$) and performing model inference on them, resulting in extra computational overhead. To make GDSD efficient for online RL, we therefore propose two normalization-free logit-matching methods to bypass the bottleneck of computing Z_t by exploiting the translation invariance of the Softmax operator.

Naive Solution: Direct Matching. A naive method is to directly match the student logits to the unnormalized teacher denoiser in Equation (7) without the partition term $\log Z_t(x_t)$. This is motivated by the translation invariance of the Softmax operator: $\text{Softmax}(\mathbf{y}) = \text{Softmax}(\mathbf{y} + c)$ for any constant c. Therefore, given masked sequence x_t, if the logits of the student denoiser match the teacher's up to an additive term independent of x_0, the resulting denoising distribution is identical to p^* after Softmax.

Token-level Logit Centralization (TLC). Direct matching is simple and effective. However, during iterative matching to the unnormalized target, the scale of the logits can shift uncontrollably. Inspired by zero-meaned logit matching by Hinton et al. [20], we propose an alternative method: centralizing the logits of denoisers. For any MDM p, we define the token-level logit-centralized model $\bar{p}$ as:

$$\log \bar{p}(x_0|x_t) \overset{\text{def}}{=} \sum_n^N \log p(x_0^{(n)}|x_t) - \sum_n^N \frac{1}{|\mathcal{V}|} \sum_{x_0'^{(n)} \in \mathcal{V}} \log p(x_0'^{(n)}|x_t) \tag{8}$$

Due to the factorization of the sequence-level denoising distribution, TLC is equivalent to centralizing the sequence-level log-probability: $\log \bar{p}(x_0|x_t) = \log p(x_0|x_t) - \sum_{x_0} \log p(x_0|x_t)$. By the translation invariance of the softmax operator, the denoising probability distribution induced by these centralized logits is identical to the one induced by the raw logits. Since the partition function Z_t is independent of x_0, centralization eliminates Z_t entirely from the objective:

Proposition 3.3. *Define the advantage baseline $b := \sum_{x_0 \in \mathcal{X}} A(x_0)/|\mathcal{X}|$. For any clean and masked sequence x_0 and x_t, the centralized logits of p^* becomes normalization-free (Proof in A.2):*

$$\log \bar{p}^*(x_0|x_t) = (1-\beta)\log \bar{p}_{old}(x_0|x_t) + \beta \log \bar{p}_{ref}(x_0|x_t) + \psi A(x_0) - b. \tag{9}$$

Since the advantage baseline b is independent of x_0, diffusion time t, and even masked sequence x_t, being a globally constant offset to all the logits at all time t, in practice we can further omit the baseline b directly, meanwhile preserving the resulting denoising distribution. Additionally, since the logits are centralized and the advantage itself has zero mean, i.e., $\mathbb{E}_{\pi_{\text{old}}}[A(x_0)] = 0$, the scale of the logits is prevented from drifting away during iterative updates. Combined with normalization-free implementation, we derive the practical objective of GDSD (additional details in C.2):

Guided Denoiser Self-Distillation (GDSD) (practical)

$$\mathbb{E}_{t\sim U(0,1),\ x_0\sim\pi_{\text{old}}^{\text{rm}},\ x_t\sim q(\cdot|x_0)}\left[\left(\log\frac{\bar{p}_\theta(x_0|x_t)}{\bar{p}_{\text{old}}(x_0|x_t)}-\psi A(x_0)\right)^2+\beta\left(\log\frac{\bar{p}_\theta(x_0|x_t)}{\bar{p}_{\text{ref}}(x_0|x_t)}\right)^2\right]. \quad (10)$$

All reported results are obtained with Equation (10) (direct matching replaces $\bar{p}$ by p). Proposition C.1 in Section C.2 shows that its gradient equals that of Equation (6) with the centralized teacher of Equation (9) under an explicit reparameterization of (ψ, β), and states the scope of this equivalence.

4 Connection and Comparison to Related Work

In this section, we draw the important connections between our method and closely related work.

GDSD applies logit matching to approximate the teacher denoising distribution p^* with p_θ. Specifically, the objective is the squared l_2 distance between the logit vectors of the student denoiser and the teacher with samples from a mixture of the old and reference policy. Apart from l^2, it is also eligible to apply other functions for self-distillation, such as forward-KL and reverse-KL divergence.

Advantage-Weighted ELBO. Adopting forward-KL divergence for distillation derives advantage-weighted ELBO objectives. This formulation recovers the objectives employed in wd1 (AW-DCE) [51] and DMPO [73]. Define the forward-KL objective as:

$$\mathcal{L}_{\text{fwd}}(\theta)\overset{\text{def}}{=}\mathbb{E}_{t\sim U(0,1),\ x_t\sim p_t^*}[D_{\text{KL}}(p^*(\cdot|x_t)\parallel p_\theta(\cdot|x_t))].$$

Since the partition function $A_t(x_t)$ and $p_{\text{old}}^{\text{ref}}$ terms in p^* are independent of θ, then importance sampling reveals this is equivalent to the ELBO re-weighted by the exponential of the advantage:

Proposition 4.1 (Forward-KL Distillation Induces Advantage-Weighted ELBO)**.** *Up to terms independent of θ, optimizing $\mathcal{L}_{\text{fwd}}$ is equivalent to an ELBO-style denoising loss reweighted by the exponential advantage (proof in A.3):*

$$\nabla_\theta\mathcal{L}_{fwd}(\theta)\propto\nabla_\theta\mathbb{E}_{x_0\sim p_{old}^{ref}}\Big[\exp(\psi A(x_0))\underbrace{\mathbb{E}_{t\sim U(0,1),\ x_t\sim q(\cdot|x_0)}\big[-\log p_\theta(x_0\mid x_t)\big]}_{ELBO}\Big].$$

A primary limitation of such exponential-weighting schemes is data inefficiency [39]: samples x_0 with negative advantages are assigned negligible weights, contributing to the gradient mildly. This makes the exponentially weighted method weaker than policy gradient methods. While Tang et al. [51] attempts to mitigate this by introducing an auxiliary loss to penalize negative samples, such explicit penalization can trigger training instability [35]. In contrast, GDSD naturally incorporates the sign of the advantage function, allowing for the stable and efficient utilization of the entire sample batch.

Regularized Policy Gradient. Applying reverse-KL distillation to the energy-guided teacher recovers an policy-gradient-based regularized RL objective (proof in A.4):

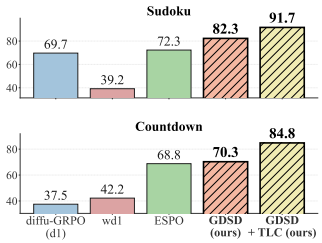
$$\begin{aligned}&\mathcal{L}_{\text{rev}}(\theta)\overset{\text{def}}{=}\mathbb{E}_{t\sim U(0,1),x_t\sim p_{\theta,t}}\big[D_{\text{KL}}(p_\theta(\cdot|x_t))\parallel p^*(\cdot|x_t)\big]\\&=\underbrace{\mathbb{E}_{x_0\sim\pi_\theta}[-\psi A(x_0)]}_{\text{reward maximization}}+\underbrace{\mathbb{E}_{t\sim U(0,1),x_0\sim\pi_\theta,x_t\sim q(\cdot|x_0)}[A_t(x_t)]}_{\text{reward baseline: diffusion state value}}+\underbrace{\mathbb{E}_{x_0\sim\pi_\theta}[L(x_0;p_\theta)-L(x_0;p_{\text{old}}^{\text{ref}})]}_{\text{regularization}},\end{aligned}$$

where L denotes ELBO. This decomposition shows that reverse-KL distillation recovers a regularized policy gradient (PG) form. The leading term corresponds to maximizing the expected advantage under the current policy π_θ, while $A_t(x_t)$ acts as a state-dependent baseline. The remaining term regularizes the current denoiser against the old/reference mixture $p_{\text{old}}^{\text{ref}}$. Therefore, PG-based methods can be viewed as optimizing a related objective, but with the likelihood term approximated by ELBO, including SPG [55], UniGRPO [63], ESPO [35]. However, as detailed in Section 2.3, all of these methods that require importance sampling, such as PG for dLLMs, have inevitable TIM bias.

Reinforcement Learning via Logit-Matching. Logit-matching loss has been used in *auto-regressive* models post-training [71, 61, 52], including the large-scale one in Kimi [53]. Concurrent work EMBR

Table 1: Zero-shot test accuracy of different methods using diffusion language model Dream-7B-Instruct. We summarize the best performance across generation length in the right figure. GDSD demonstrates significant improvements over ELBO-based methods, achieving up to $+19.6\%$.

	Sudoku				Countdown			
Model / Seq Len	**128**	**256**	**512**	**Avg.**	**128**	**256**	**512**	**Avg.**
Dream-7B	9.3	2.1	14.0	8.5	8.5	7.8	17.4	11.2
+ diffu-GRPO (d1)	64.4	69.7	51.1	61.7	27.3	27.7	37.5	30.8
+ wd1	29.5	39.2	30.3	33.0	28.9	37.9	42.2	36.3
+ ESPO	71.7	72.3	71.3	71.8	68.8	66.8	64.8	66.8
+ **GDSD** (ours)	82.3	82.2	79.3	81.3	70.3	68.4	63.7	67.5
+ **GDSD w/ TLC** (ours)	**91.1**	**91.3**	**91.7**	**91.4**	**83.2**	**82.4**	**84.8**	**83.5**



[46] applies an energy-matching objective to dLLMs for preference learning, in both online (replay-buffer) and offline settings; unlike GDSD, it still relies on the ELBO and matches energies of clean sequences rather than denoiser logits. Squared log-posterior regression toward a reward-tilted target also appears in DDPP [41] for steering masked diffusion models, and squared log-ratio losses appear in GFlowNets [5]; we compare with both in B. Recently, logit-matching loss has also been shown to be exceptionally stable in continuous diffusion RL [13]. In contrast, GDSD is a normalization-free method in general RL settings for (discrete) dLLMs. (More related work in B)

5 Experiment

We perform RL based on two open-source diffusion LLMs, LLaDA-8B-Instruct and Dream-v0-Instruct-7B, evaluated across six benchmarks spanning three domains: mathematical reasoning (GSM8K, MATH500), planning (Countdown, Sudoku), and coding (HumanEval, MBPP). All trainings except coding tasks are based on Low-Rank Adaptation (LoRA). A separate model is trained for each task, except for coding, where a single model is trained on AceCoder-87K and evaluated on both HumanEval and MBPP. Training for planning and coding tasks follows the reward configuration (verifiable reward) following ESPO [35], while on the mathematical reasoning we incorporate format-reward following d1 [67]. (See more details in D.3)

Evaluation. Given the incompatible evaluation protocols across prior literature, we follow the ESPO settings and directly adopt their reported results for LLaDA, d1, wd1, and UniGRPO (detailed in Appendix D.4); math results are reproduced from the official repositories and evaluated via lm-eval with diffusion steps equal to the generation length. To unify evaluation, we use zero-shot for Sudoku, Countdown, GSM8K, MATH500, and HumanEval, and 3-shot for MBPP, with logical planning tested at generation lengths of 128, 256, and 512.

Implementation. We implement mainly two versions of GDSD, both of which remove the intractable normalization constant (i.e. Z_t in Equation (7)). One is default GDSD direct matching without token logit centralization (TLC) and the normalization constant is removed (i.e. Equation (10) but without TLC), which we name GDSD directly. We also implement GDSD with TLC following the same equation. For baseline implementation and their results, we either reproduce results by evaluating the released checkpoints or re-run training using their official implementation.

5.1 Main Results

We evaluate GDSD on the pure diffusion model Dream-7B in Table 9. Compared to prior state-of-the-art ELBO-based methods, GDSD significantly improves the average test accuracy $+9.5\%$ on average and $+10\%$ with the best-performing generation length. Adding token logit centralization yields an additional gain of roughly, achieving up to $+19.6\%$ over the baselines. These results confirm the effectiveness of both GDSD and the centralization.

For the block diffusion model [2], the most widely tested base model LLaDA-8B, we conduct a comprehensive evaluation on planning, math, and coding benchmarks. We provide the training reward dynamics in Figure 2, and testing accuracy in Table 2 and Table 3. As for training reward, on the planning and coding tasks, GDSD demonstrates stable convergence relative to prior state-of-the-art ELBO-based methods. As for testing accuracy, GDSD outperforms prior methods on almost all the

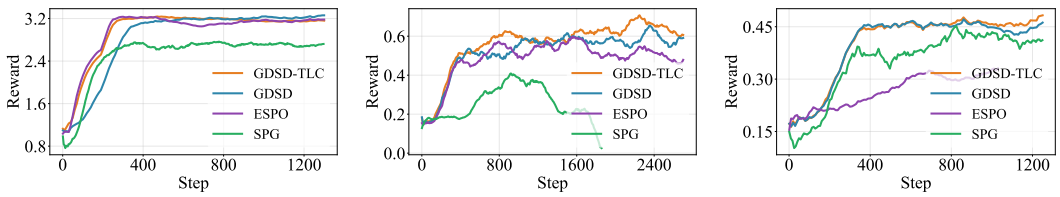


Figure 2: Training reward dynamics trained on LLaDA-8B-Instruct with different methods on different training datasets: **Left:** GSM8K; **Mid:** Countdown; **Right:** Coding. Our method demonstrates more stable training reward dynamics across datasets.

benchmarks, achieving average accuracy gains up to $+5\%$ on Sudoku, $+1.9\%$ on Countdown, $+3\%$ on GSM8K and $+1.1\%$ on MATH, $+4.6\%$ on HumanEval-Plus.

These results, particularly the accuracy improvement of up to $+20\%$, demonstrate the effectiveness of our denoiser self-distillation method reformulated from RL. Even when GDSD performs comparably to ELBO-based baselines on some benchmarks, the results still support our central claim: ELBO is not necessary for RL with dLLMs and can be removed to avoid the bias analyzed in Section 2.3. Overall, the empirical results validate GDSD as an effective alternative to ELBO-based importance-sampling RL, providing a cleaner and more principled optimization route.

5.2 Ablation Study

In this section, we perform ablation study on different elements of GDSD to confirm the effectiveness of our design choices.

Token Logit Centralization (TLC). Across all benchmarks in Table 2, GDSD by direct matching without TLC achieves clear improvements over prior ELBO-based methods. This confirms the effect of the normalization constant is negligible in practice. Notably, although GDSD with TLC is more faithful to the theory, it sometimes leads to degraded test performance. However, the lack of consistent improvement in test accuracy suggests a potential generalization gap. We hypothesize that, by self-centralizing, TLC makes the update focus more strongly on relative logit differences, which improves fitting to the training reward but may also amplify overfitting to training-specific signals, which explains the stable training dynamics of applying TLC in Figure 2 and Figure 3.

Guidance Coefficient ψ. We further conduct an ablation study on the guidance strength ψ in Figure 3 to examine the role of energy guidance in GDSD. The results show that increasing ψ generally leads to higher training rewards, indicating that stronger energy guidance produces a teacher denoiser more biased toward high-advantage samples. This trend supports the effectiveness of the proposed guided-denoiser distillation formulation: GDSD directly distills the energy-guided target distribution, and therefore can translate stronger guidance signals into improved optimization performance.

Regularization Coefficient β. Training rewards are insensitive to β for $\beta \leq 10^{-3}$ and degrade at $\beta = 10^{-2}$ (Section D.7). We use $\psi = 10$ for all tasks and a small β (Table 4).

Re-masking Policy. To probe the role of the sampler–likelihood mismatch directly, we also train LLaDA-8B with random re-masking during rollouts (Section D.6). GDSD w/ TLC reaches a late-stage reward of 0.94 on Sudoku versus 0.76 for ESPO and 0.25 for SPG, while on GSM8K it converges faster but ends comparable to ESPO.

6 Limitations

Our experiments are limited to 7B–8B open dLLMs (LLaDA-8B, Dream-7B), the standard testbed of our baselines, and mostly to verifiable rewards; GDSD only consumes a scalar advantage, so larger dLLMs and reward-model-based RLHF need no modification, but we have not evaluated them due to compute. GDSD removes the likelihood ratio, and hence the TIM bias of ELBO-based methods, but remains an off-policy regression whose optimum relies on realizability and coverage by π_{old} (Section 3.2). Its normalization-free practical objective preserves the teacher distribution

Table 2: Test accuracy of different methods using *block* diffusion [2] model LLaDA-8B-Instruct. The results with † are reproduced by us; others are extracted from the corresponding papers. Numbers following benchmark (k) represent k-shot evaluation. Our methods demonstrate consistent improvement on average accuracy.

	Sudoku(0)				Countdown				GSM8K			
Model / Seq Len	**128**	**256**	**512**	**Avg.**	**128**	**256**	**512**	**Avg.**	**128**	**256**	**512**	**Avg.**
LLaDA-8B	24.8	16.2	6.0	15.7	20.7	19.5	16.0	18.7	71.3	76.2	80.2	75.9
+ diffu-GRPO (d1)	26.7	24.1	15.9	22.2	33.2	31.3	37.1	33.9	74.6	78.1	81.2	78.0
+ wd1	22.6	76.4	62.8	53.9	47.7	51.2	46.1	48.3	77.2	80.8	82.3	80.1
+ UniGRPO	/	/	/	/	44.5	43.0	57.0	48.2	74.9	82.5	82.7	80.0
+ DMPO	32.8	24.6	20.0	25.6	67.2	80.9	**82.8**	77.0	74.8	82.4	85.2	80.8
+ SPG	87.3†	88.2†	71.7†	82.4	68.8	70.7	70.3	69.9	82.8†	85.1†	84.9†	84.2
+ ESPO	**92.7**	84.7	80.5	86.0	81.6	82.0	79.3	81.0	81.6†	82.5†	83.0†	82.4
+ **GDSD** (ours)	89.5	89.8	88.8	89.4	82.4	**84.0**	**82.8**	**83.1**	**83.9**	**86.4**	**86.0**	**85.4**
+ **GDSD w/ TLC** (ours)	90.4	**92.0**	**90.6**	**91.0**	**85.6**	80.5	75.0	80.4	81.3	85.6	85.0	84.0

	MATH500				HumanEval-Plus(0)				MBPP(3)			
Model / Seq Len	**128**	**256**	**512**	**Avg.**	**128**	**256**	**512**	**Avg.**	**128**	**256**	**512**	**Avg.**
LLaDA-8B	34.4	35.2	41.4	37.0	23.2	30.5	41.5	31.7	36.2	42.0	38.1	38.8
+ diffu-GRPO (d1)	34.9	36.6	41.7	37.7	22.0	29.9	37.2	29.7	34.8	36.6	38.0	36.5
+ wd1	33.3	37.7	39.8	36.9	29.9	29.9	32.9	30.9	38.0	37.2	34.4	36.5
+ UniGRPO	32.4	37.4	39.4	36.4	/	/	/	/	/	/	/	/
+ DMPO	30.0	38.2	42.8	37.0	/	/	/	/	/	/	/	/
+ SPG	33.4	40.0	41.8	38.4	32.9†	34.2†	37.8†	35.0	40.4†	40.8†	40.4†	40.5
+ ESPO	36.0	39.0	43.4	39.5	24.4	36.6	**42.7**	34.6	**43.6**†	43.2†	41.2†	42.7
+ **GDSD** (ours)	**37.0**	40.4	**44.6**	**40.6**	36.0	38.4	41.5	38.6	40.6	41.8	**43.6**	42.0
+ **GDSD w/ TLC** (ours)	34.6	**41.6**	44.0	40.1	**38.4**	**39.6**	39.6	**39.2**	43.0	**43.6**	43.2	**43.3**

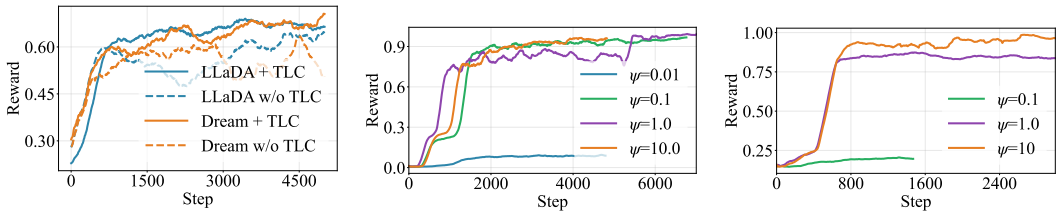


Figure 3: Ablation study on Token-level Logit Centralization (TLC) and the energy-guidance coefficient ψ. **Left**:Training dynamics on Countdown with different models with and without TLC; **Mid:** Training dynamics on Sudoku with Dream-7B.; **Right:** Training dynamics on Sudoku with LLaDA-8B. TLC demonstrate a more stable training behavior. The improved training reward by increasing the temperature ψ implies the effectiveness of GDSD.

but approximates Equation (6) rather than estimating it without bias (Section C.2). Finally, TLC improves training stability but not consistently test accuracy, so we select it on validation.

7 Conclusion

In this work, we argue that existing RL methods for dLLMs that use the likelihood evidence lower bound (ELBO) as a likelihood surrogate introduce bias, which can degrade performance and potentially cause training collapse. We propose Guided Denoiser Self-Distillation (GDSD), an off-policy self-distillation framework that distills toward the closed-form optimum of reverse-KL-regularized RL without requiring likelihood ratios, thereby bypassing this bias. Empirically, GDSD consistently improves over state-of-the-art ELBO-based RL methods on Dream-7B and LLaDA-8B across planning, math, and coding benchmarks. The gains are especially pronounced on Dream-7B, reaching up to $+20\%$ absolute improvement. These results demonstrate that GDSD provides an effective and stable alternative for RL with dLLMs.

8 Acknowledgements

We thank the anonymous reviewers and area chair for their helpful comments. Ilija Bogunovic was supported by the EPSRC New Investigator Award EP/X03917X/1; the Engineering and Physical Sciences Research Council EP/S021566/1. Sangwoong Yoon was supported in part by the InnoCORE program of the Ministry of Science and ICT (No. 1.260017.01), the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (RS-2025-25442824, AI Star Fellowship Program (Ulsan National Institute of Science and Technology)) and (No. RS-2020-II201336, Artificial Intelligence Graduate School Program (UNIST)), and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2026-25506577). Xiaohang Tang was supported by the UKRI Engineering and Physical Sciences Research Council [grant number EP/T517793/1, EP/W524335/1]. Keyue Jiang was supported by UKRI EPSRC [grant number EP/R513143/1]. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

[1] Arel. Arel's sudoku generator. https://www.ocf.berkeley.edu/~arel/sudoku/main.html, 2025. Accessed: 2025-04-08.

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

[3] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

[4] Brian Bartoldson, Siddarth Venkatraman, James Diffenderfer, Moksh Jain, Tal Ben-Nun, Seanie Lee, Minsu Kim, Johan Obando Ceron, Yoshua Bengio, and Bhavya Kailkhura. Trajectory balance with asynchrony: Decoupling exploration and learning for fast, scalable llm post-training. *Advances in Neural Information Processing Systems*, 38:113901–113931, 2026.

[5] Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

[6] Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, et al. Llada2. 0: Scaling up diffusion language models to 100b. *arXiv preprint arXiv:2512.15745*, 2025.

[7] Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, et al. Llada2. 1: Speeding up text diffusion via token editing. *arXiv preprint arXiv:2602.08676*, 2026.

[8] Huayu Chen, Cheng Lu, Zhengyi Wang, Hang Su, and Jun Zhu. Score regularized policy optimization through diffusion behavior. *arXiv preprint arXiv:2310.07297*, 2023.

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

[10] Shirui Chen, Jiantao Jiao, Lillian J Ratliff, and Banghua Zhu. dultra: Ultra-fast diffusion language models via reinforcement learning. *arXiv preprint arXiv:2512.21446*, 2025.

[11] Xiaoyuan Cheng, Xiaohang Tang, and Yiming Yang. Safe and stable control via lyapunov-guided diffusion models. *arXiv preprint arXiv:2509.25375*, 2025.

[12] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[13] Jaemoo Choi, Yuchen Zhu, Wei Guo, Petr Molodyk, Bo Yuan, Jinbin Bai, Yi Xin, Molei Tao, and Yongxin Chen. Rethinking the design space of reinforcement learning for diffusion models: On the importance of likelihood estimation beyond loss design. *arXiv preprint arXiv:2602.04663*, 2026.

[14] 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. *CoRR*, abs/2110.14168, 2021.

[15] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[16] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

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

[18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[19] Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

[20] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[21] 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 arXiv:2505.10446*, 2025.

[22] Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

[23] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[24] Haozhe Jiang, Haiwen Feng, Jiantao Jiao, Angjoo Kanazawa, and Nika Haghtalab. Diffusion policy optimization without drifting apart. In *ICLR 2026 2nd Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*.

[25] 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. URL https://arxiv.org/abs/2506.17298.

[26] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*. OpenReview.net, 2024.

[27] Jiacai Liu, Yingru Li, Yuqian Fu, Jiawei Wang, Qian Liu, and Zhuo Jiang. When speed kills stability: Demystifying RL collapse from the training-inference mismatch, September 2025. URL https://richardli.xyz/rl-collapse.

[28] Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025.

[29] 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. *arXiv preprint arXiv:2503.20783*, 2025.

[30] Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive Energy Prediction for Exact Energy-Guided Diffusion Sampling in Offline Reinforcement Learning. In *International Conference on Machine Learning*, pages 22825–22855. PMLR, 2023.

[31] Haofei Lu, Dongqi Han, Yifei Shen, and Dongsheng Li. What makes a good diffusion planner for decision making? *arXiv preprint arXiv:2503.00535*, 2025.

[32] David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. *arXiv preprint arXiv:2507.21053*, 2025.

[33] Chenlin Meng, Kristy Choi, Jiaming Song, and Stefano Ermon. Concrete Score Matching: Generalized Score Matching for Discrete Data. *Advances in Neural Information Processing Systems*, 35:34532–34545, 2022.

[34] 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.

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

[36] 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, 2025. URL https://arxiv.org/abs/2406.03736.

[37] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training Language Models to Follow Instructions with Human Feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

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

[39] Simon Park, Abhishek Panigrahi, Yun Cheng, Dingli Yu, Anirudh Goyal, and Sanjeev Arora. Generalizing from simple to hard visual reasoning: Can we mitigate modality imbalance in vlms? *arXiv preprint arXiv:2501.02669*, 2025.

[40] Penghui Qi, Zichen Liu, Xiangxin Zhou, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Defeating the training-inference mismatch via fp16. *arXiv preprint arXiv:2510.26788*, 2025.

[41] Jarrid Rector-Brooks, Mohsin Hasan, Zhangzhi Peng, Chenghao Liu, Sarthak Mittal, Nouha Dziri, Michael Bronstein, Pranam Chatterjee, Alexander Tong, and Joey Bose. Steering masked discrete diffusion models via discrete denoising posterior prediction. In *International Conference on Learning Representations*, volume 2025, pages 25383–25414, 2025.

[42] Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.

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

[44] John Schulman. Approximating kl divergence, 2020. *URL http://joschu. net/blog/kl-approx. html*, 2020.

[45] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust Region Policy Optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[46] Shiv Shankar. Energy matching based preference learning for diffusion language models. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 776–786, 2026.

[47] 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 arXiv:2402.03300*, 2024.

[48] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167, 2024.

[49] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and Generalized Masked Diffusion for Discrete Data, 2025. URL https://arxiv.org/abs/2406.04329.

[50] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

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

[52] Xiaohang Tang, Sangwoong Yoon, Seongho Son, Huizhuo Yuan, Quanquan Gu, and Ilija Bogunovic. Rspo: Regularized self-play alignment of large language models. *arXiv preprint arXiv:2503.00030*, 2025.

[53] Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

[54] Gilad Turok, Chris De Sa, and Volodymyr Kuleshov. Duel: Exact likelihood for masked diffusion via deterministic unmasking. *arXiv preprint arXiv:2603.01367*, 2026.

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

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

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

[58] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

[59] Chenxing Wei, Jiazhen Kang, Hong Wang, Jianqing Zhang, Hao Jiang, Xiaolong Xu, Ningyuan Sun, Ying He, F Richard Yu, Yao Shu, et al. Lfpo: Likelihood-free policy optimization for masked diffusion models. *arXiv preprint arXiv:2603.01563*, 2026.

[60] Esmeralda S Whitammer, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in neural information processing systems*, 35:5955–5967, 2022.

[61] Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. Self-play preference optimization for language model alignment. *arXiv preprint arXiv:2405.00675*, 2024.

[62] Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr. *arXiv preprint arXiv:2604.03128*, 2026.

[63] Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal Large Diffusion Language Models. *arXiv preprint arXiv:2505.15809*, 2025.

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

[65] Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. Acecoder: Acing coder rl via automated test-case synthesis, 2025. URL https://arxiv.org/abs/2502.01718.

[66] Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-Weighted Flow Matching for Offline Reinforcement Learning. *arXiv preprint arXiv:2503.04975*, 2025.

[67] Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling Reasoning in Diffusion Large Language Models via Reinforcement Learning. *arXiv preprint arXiv:2504.12216*, 2025.

[68] Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

[69] Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. Diffusionnft: Online diffusion reinforcement with forward process. *arXiv preprint arXiv:2509.16117*, 2025.

[70] Jianyuan Zhong, Kaibo Wang, Ding Ding, Zijin Feng, Haoli Bai, Yang Xiang, Jiacheng Sun, and Qiang Xu. Stabilizing reinforcement learning for diffusion language models. *arXiv preprint arXiv:2603.06743*, 2026.

[71] Banghua Zhu, Hiteshi Sharma, Felipe Vieira Frujeri, Shi Dong, Chenguang Zhu, Michael I Jordan, and Jiantao Jiao. Fine-Tuning Language Models with Advantage-Induced Policy Alignment. *arXiv preprint arXiv:2306.02231*, 2023.

[72] Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, et al. LLaDA 1.5: Variance-Reduced Preference Optimization for Large Language Diffusion Models. *arXiv preprint arXiv:2505.19223*, 2025.

[73] Yuchen Zhu, Wei Guo, Jaemoo Choi, Petr Molodyk, Bo Yuan, Molei Tao, and Yongxin Chen. Enhancing reasoning for diffusion llms via distribution matching policy optimization. *arXiv preprint arXiv:2510.08233*, 2025.

A Proof

A.1 Lemma: Token-level Logit Centralization

Lemma A.1. *Denote the n-th token of clean completion x_0 as $x_0^{(n)}$, for any MDM p, we have the token-level logit centralization defined in Equation (8) equivalent to sequence-level centralized logit $\log p(x_0|x_t) - \sum_{x_0} \log p(x_0|x_t)$:*

$$\log p(x_0|x_t) - \sum_{x_0} \log p(x_0|x_t) = \sum_{n}^{N} \log p(x_0^{(n)}|x_t) - \sum_{n}^{N} \frac{1}{|\mathcal{V}|} \sum_{x_0'^{(n)} \in \mathcal{V}} \log p(x_0'^{(n)}|x_t) \tag{11}$$

Proof. The Left-Hand Side (LHS) of Equation Equation (11) represents the centralized log-probability of the sequence x_0:

$$\text{LHS} = \log p(x_0|x_t) - \frac{1}{|\mathcal{X}|} \sum_{x_0 \in \mathcal{X}} \log p(x_0|x_t)$$

Using the factorization property $\log p(x_0|x_t) = \sum_n \log p(x_0^{(n)}|x_t)$, we substitute this into both terms:

$$\text{LHS} = \sum_{n=1}^{N} \log p(x_0^{(n)}|x_t) - \frac{1}{|\mathcal{X}|} \sum_{x_0 \in \mathcal{X}} \left(\sum_{n=1}^{N} \log p(x_0^{(n)}|x_t) \right).$$

The space of all possible sequences $\mathcal{X}$ is the Cartesian product of the vocabulary $\mathcal{V}$ for each position n: $\mathcal{X} = \mathcal{V} \times \mathcal{V} \times \cdots \times \mathcal{V}$. The total number of sequences is $|\mathcal{X}| = |\mathcal{V}|^N$. We can rewrite the sum over $\mathcal{X}$ as a nested sum over each token position:

$$\sum_{x_0 \in \mathcal{X}} \sum_{n=1}^{N} \log p(x_0^{(n)}|x_t) = \sum_{n=1}^{N} \left(\sum_{x_0 \in \mathcal{X}} \log p(x_0^{(n)}|x_t) \right).$$

Then for a fixed position n, the term $\log p(x_0^{(n)}|x_t)$ only depends on the token at that specific position. In the sum over all $|\mathcal{V}|^N$ possible sequences, each specific token $v \in \mathcal{V}$ appears at position n exactly $|\mathcal{V}|^{N-1}$ times. Therefore:

$$\sum_{x_0 \in \mathcal{X}} \log p(x_0^{(n)}|x_t) = |\mathcal{V}|^{N-1} \sum_{v \in \mathcal{V}} \log p(v|x_t).$$

Substitute this back into the LHS equation:

$$\text{LHS} = \sum_{n=1}^{N} \log p(x_0^{(n)}|x_t) - \frac{1}{|\mathcal{V}|^N} \sum_{n=1}^{N} \left(|\mathcal{V}|^{N-1} \sum_{v \in \mathcal{V}} \log p(v|x_t) \right)$$

Simplify the fraction $\frac{|\mathcal{V}|^{N-1}}{|\mathcal{V}|^N} = \frac{1}{|\mathcal{V}|}$:

$$\text{LHS} = \sum_{n=1}^{N} \log p(x_0^{(n)}|x_t) - \sum_{n=1}^{N} \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \log p(v|x_t).$$

By swapping the index v for $x_0'^{(n)}$ to match the notation in the image:

$$\text{LHS} = \sum_{n=1}^{N} \log p(x_0^{(n)}|x_t) - \sum_{n=1}^{N} \frac{1}{|\mathcal{V}|} \sum_{x_0'^{(n)} \in \mathcal{V}} \log p(x_0'^{(n)}|x_t)$$

This matches the RHS of Equation Equation (11) exactly.

By the factorization of the sequence-level denoising distribution, the target model satisfies $p^*(x_0|x_t) = \prod_{n=1}^{N} p^*(x_0^{(n)}|x_t)$, where $p^*(x_0^{(n)}|x_t) = \text{SoftMax}\big(\log \bar{p}^*(x_0^{(n)}|x_t)\big)$. It follows that GDSD with token-level logit centralization has the same optimal probability distribution as the original GDSD formulation. □

A.2 Proof to Proposition 3.3

Proof. According to Lemma A.1, token-level logit centralization (TLC) is equivalent to sequence-level centralization. Therefore, the centralized logit of the teacher denoiser becomes

$$\log \bar{p}^*(x_0|x_t) \tag{12}$$

$$= \log p^*(x_0|x_t) - \frac{1}{|\mathcal{X}|}\sum_{x_0} \log p^*(x_0|x_t) \tag{13}$$

$$= \beta \log p_{\text{old}}(x_0|x_t) + (1-\beta)\log p_{\text{ref}}(x_0|x_t) + \log \sum_{x_0} p_{\text{old}}(x_0|x_t)^{1-\beta} \cancel{p_{\text{ref}}(x_0|x_t)^{\beta}}$$

$$+ \psi A(x_0) - \cancel{A_t(x_t)} \tag{14}$$

$$- \Big(\frac{1}{|\mathcal{X}|}\sum_{x_0} \beta \log p_{\text{old}}(x_0|x_t) + (1-\beta)\log p_{\text{ref}}(x_0|x_t) + \log \sum_{x_0} p_{\text{old}}(x_0|x_t)^{1-\beta} \cancel{p_{\text{ref}}(x_0|x_t)^{\beta}}$$

$$+ \psi A(x_0) - \cancel{A_t(x_t)} \Big) \tag{15}$$

$$= \beta \log p_{\text{old}}(x_0|x_t) + (1-\beta)\log p_{\text{ref}}(x_0|x_t) - \frac{1}{|\mathcal{X}|}\sum_{x_0} \big(\beta \log p_{\text{old}}(x_0|x_t) + (1-\beta)\log p_{\text{ref}}(x_0|x_t)\big)$$

$$+ \psi A(x_0) - \sum_{x_0} A(x_0) \tag{16}$$

$$= \beta \log \bar{p}_{\text{old}}(x_0|x_t) + (1-\beta)\log \bar{p}_{\text{ref}}(x_0|x_t) + \psi A(x_0) - \sum_{x_0} A(x_0). \tag{17}$$

The first equality follows from the definition of centralized logits, which subtracts the average logit over $\mathcal{X}$. We then substitute the log-form of $p^*(x_0 \mid x_t)$ into both the original logit and its average. The geometric-mixture normalizer and $A_t(x_t)$ depend only on x_t, so they are constant with respect to x_0 and cancel after centralization. The remaining old-policy and reference-policy terms can then be regrouped into $\log \bar{p}_{\text{old}}(x_0 \mid x_t)$ and $\log \bar{p}_{\text{ref}}(x_0 \mid x_t)$, while the advantage term is centralized by subtracting its average over $\mathcal{X}$. □

A.3 Proof of Proposition 4.1

Proof. Starting with the definition of the forward KL objective for a diffusion large language model (dLLM):

$$\mathcal{L}_{\text{fwd}}(\theta) = \mathbb{E}_{t\sim U(0,1), x_t\sim p_t^*}[D_{\text{KL}}(p^*(\cdot|x_t)||p_\theta(\cdot|x_t))].$$

Expanding the Kullback–Leibler divergence term into its expectation form:

$$\mathcal{L}_{\text{fwd}}(\theta) = \mathbb{E}_{t\sim U(0,1), x_t\sim p_t^*}\mathbb{E}_{x_0\sim p^*(\cdot|x_t)}[\log p^*(x_0|x_t) - \log p_\theta(x_0|x_t)].$$

By substituting the definition of the energy-guided target denoising distribution

$$p^*(x_0|x_t) = p_{\text{old}}^{\text{ref}}(x_0|x_t)\cdot \exp(\psi A(x_0) - A_t(x_t)),$$

the objective becomes:

$$\mathbb{E}_{t,x_t}\mathbb{E}_{x_0\sim p^*}[\log p_{\text{old}}^{\text{ref}}(x_0|x_t) + \psi A(x_0) - A_t(x_t) - \log p_\theta(x_0|x_t)].$$

To find the gradient with respect to the model parameters θ, we identify the terms independent of θ:

- $\log p_{\text{old}}^{\text{ref}}(x_0|x_t)$ depends only on the old and reference policies.
- $\psi A(x_0)$ is the advantage function based on the reward.
- $A_t(x_t)$ is the log-normalization constant (partition function).

Since these terms have zero gradients w.r.t. θ, the gradient of the objective simplifies to:

$$\nabla_\theta \mathcal{L}_{\text{fwd}}(\theta) = \nabla_\theta \mathbb{E}_{t\sim U(0,1), x_t\sim p_t^*}\mathbb{E}_{x_0\sim p^*(\cdot|x_t)}[-\log p_\theta(x_0|x_t)]$$

Finally, by applying importance sampling to change the expectation from the target distribution p^* to the reference distribution $\pi_{\text{old}}^{\text{ref}}$, we incorporate the advantage as a weight:

$$\nabla_\theta \mathcal{L}_{\text{fwd}}(\theta) \propto \nabla_\theta \mathbb{E}_{x_0 \sim \pi_{\text{old}}^{\text{ref}}}[\exp(A(x_0)) \mathbb{E}_{t \sim U(0,1), x_t \sim p_t^*}[-\log p_\theta(x_0|x_t)]].$$

This result is precisely the objective for wd1 with ELBO as likelihood approximation and positive weight, namely Advantage-Weighted Discrete Cross-Entropy (AW-DCE) [51] and Distribution Matching Policy Optimization (DMPO) [73]. □

A.4 Proof of Reverse-KL Distillation to Policy Gradient

Proof. By the definition of the energy-guided teacher,

$$\log p^*(x_0 \mid x_t) = \log p_{\text{old}}^{\text{ref}}(x_0 \mid x_t) + \psi A(x_0) - A_t(x_t). \tag{18}$$

Therefore,

$$\begin{aligned}
\mathcal{L}_{\text{rev}}(\theta) &= \mathbb{E}_{t, x_t \sim p_{\theta,t}} \left[D_{\text{KL}}\big(p_\theta(\cdot \mid x_t) \,\|\, p^*(\cdot \mid x_t)\big) \right] && (19)\\
&= \mathbb{E}_{t, x_t \sim p_{\theta,t},\, x_0 \sim p_\theta(\cdot|x_t)} \left[\log p_\theta(x_0 \mid x_t) - \log p^*(x_0 \mid x_t) \right] && (20)\\
&= \mathbb{E}_{t, x_t, x_0} \left[-\psi A(x_0) + A_t(x_t) + \log p_\theta(x_0 \mid x_t) - \log p_{\text{old}}^{\text{ref}}(x_0 \mid x_t) \right] && (21)\\
&= \mathbb{E}_{x_0 \sim \pi_\theta} \left[-\psi A(x_0) \right] + \mathbb{E}_{t \sim U(0,1),\, x_0 \sim \pi_\theta,\, x_t \sim q(\cdot|x_0)} \left[A_t(x_t) \right] && (22)\\
&\quad + \mathbb{E}_{x_0 \sim \pi_\theta} \left[L(x_0; p_\theta) - L(x_0; p_{\text{old}}^{\text{ref}}) \right], && (23)
\end{aligned}$$

where the last equality uses the sampling equivalence $x_0 \sim \pi_\theta,\ x_t \sim q(\cdot \mid x_0)$ and the definition

$$L(x_0; p) := \mathbb{E}_{t \sim U(0,1),\, x_t \sim q(\cdot|x_0)} \left[-\log p(x_0 \mid x_t) \right].$$

□

B Additional Related Work

Reinforcement Learning for Diffusion Models. Reinforcement learning for diffusion models has been widely studied in the context of diffusion policies for robotics, with applications including planning, offline RL, imitation learning, and safe control [58, 23, 19, 30, 8, 42, 66, 12, 31, 32, 11]. However, these works are primarily built upon continuous diffusion models and are typically evaluated on relatively small-scale policies. More recently, several works have explored RL fine-tuning for large-scale vision diffusion models [28, 69, 13]. These methods operate under continuous diffusion or flow-matching formulations, where the score function or velocity vector field is explicitly modeled for image generation. In contrast, discrete diffusion language models define denoising distributions or concrete score [33] over tokens and do not expose analogous continuous score or velocity fields. This structural difference makes existing RL methods for continuous diffusion models difficult to apply directly to dLLMs, motivating RL objectives designed specifically for discrete denoising distributions.

Reinforcement Learning via Logit Matching. Various post-training methods for autoregressive models can be interpreted through the lens of logit matching. For example, Kimi performs RL by matching the tractable policy to an exponential-reward-weighted old policy [53]. Self-play policy optimization methods adopt a similar form in preference optimization [61, 52]. These methods can be viewed as exponential-weight updates over the policy [15], where the weighting signal is the reward or advantage of a given response. They are also closely related to soft policy iteration methods, such as SAC [18]. However, these approaches assume access to a tractable response likelihood, which is unavailable in diffusion language models. More recently, diffusion RL methods have adopted logit-matching-style objectives. For continuous diffusion models, PAR [13] shows that logit matching is stable and effective for RL post-training. EMBR [46] proposes an energy-matching objective for preference learning with diffusion language models, formulated for max-entropy RL in preference learning and supporting both online replay-buffer training and offline preference learning. GDSD differs in three ways: EMBR still relies on the ELBO, whereas GDSD is likelihood-free; EMBR matches energies of clean sequences, whereas GDSD matches denoiser logits (x_0-posteriors), aligned with MDM pre-training; and EMBR estimates the normalization constant by Monte-Carlo, whereas GDSD removes it by TLC at no additional cost. More broadly, we derive GDSD from online objective PPO with reverse-KL regularization, driven by scalar verifiable or reward-model advantages, analyze challenges of online RL, including TIM and normalization constant, and address them accordingly.

Reward Steering of Masked Diffusion Models. DDPP [41] formulates steering a pretrained masked diffusion model toward a reward-tilted distribution as matching a reward-induced denoising posterior, and develops squared log-posterior regression objectives involving the pretrained denoiser, the reward, and an unknown log-partition function, which is estimated by importance sampling (DDPP-IS), learned with an auxiliary head (DDPP-LB), or avoided through a reverse-KL objective that requires a differentiable reward (DDPP-KL). GDSD shares this posterior-matching construction, but differs in: (i) *problem setting*: GDSD is online policy iteration derived from the closed-form optimum of KL-regularized RL (Equation (4)), whose teacher is the advantage tilt of the current π_{old}–π_{ref} mixture, refreshed every μ steps with group-relative advantages, rather than a fixed reward tilt of a pretrained model; (ii) *objective*: GDSD has an additional reference-regularization term with an independently tunable coefficient (Proposition C.1), which we found necessary for stable training on reasoning benchmarks; (iii) *normalization*: TLC removes the partition function without estimators, auxiliary samples, or extra heads; and (iv) *analysis*: we study the TIM bias of ELBO-based online RL for reasoning dLLMs and recover existing ELBO-based methods as alternative distillation divergences (Section 4).

GFlowNets and Trajectory Balance. GFlowNet objectives such as trajectory balance [60], recently applied to LLM post-training [4], also minimize squared log-ratio losses toward a reward-tilted target and can be trained off-policy, so their loss has a similar form to that of GDSD. The two differ in what is matched: trajectory balance is defined over generation trajectories, amortizes sampling from a fixed reward-tilted target, and learns a partition function (or state flows); GDSD is defined at the denoiser level (x_0-posterior given a randomly masked x_t, aligned with MDM pre-training), performs policy iteration toward a closed-form target that changes at every outer iteration, and removes the normalization constant by TLC instead of learning it.

Exact-Likelihood Diffusion RL. A few RL methods for dLLMs are based on trajectory likelihood. Unlike ELBO-based objectives, trajectory likelihood follows the actual decoding process and therefore provides an exact likelihood under the sampling distribution. A key ingredient in computing this likelihood is the token generation order. LLaDOU introduces an additional head to predict which tokens should be unmasked and trains this mechanism end-to-end with RL [21]. Similar ideas have also been explored for accelerating generation [10]. It is also possible to use trajectory likelihood without modifying the model architecture. For example, TraceRL applies reverse-process transition probabilities for policy optimization [57]. However, trajectory-likelihood methods are computationally expensive, since the RL objective must be accumulated over diffusion steps, and current dLLMs predict the full sequence at each step. To reduce this cost, TraceRL and d2 [56] introduce step-merging strategies that trade off likelihood accuracy for efficiency. Therefore, although trajectory-likelihood methods avoid the bias of ELBO-based likelihood surrogates, they often require architectural changes or incur high computational cost as the number of diffusion steps increases. Moreover, their objective differs from the ELBO used in pre-training. In contrast, we aim to develop an RL method that retains the efficiency and pre-training compatibility of random-masking ELBO objectives, while avoiding the bias introduced by using ELBO as a likelihood surrogate.

Likelihood-free Diffusion RL. DiffusionNFT [69] proposes likelihood-free RL for continuous diffusion post-training by reformulating RL as reinforcement guidance and optimizing a flow-matching objective with positive and negative generations. While this avoids trajectory-likelihood estimation, it relies on the continuous diffusion/flow formulation and its velocity-field training target. In contrast, dLLMs define discrete token-denoising distributions, where such velocity fields are unavailable. LFPO [59] extends the idea of DiffusionNFT to RL post-training for dLLMs. However, LFPO is essentially a reward-weighted cross-entropy/ELBO method [59], closely related to the forward-KL distillation instance discussed in Theorem 4.1. To mitigate the data inefficiency caused by down-weighting negative samples, LFPO follows DiffusionNFT by constructing a negative dataset for explicit negative-response penalization. In contrast, GDSD performs logit matching on reward-guided denoisers, allowing both positive and negative advantage signals to be incorporated directly without relying on ELBO weighting or auxiliary negative-sample penalties.

Algorithm 1 Guided Denoiser Self-Distillation (GDSD)

Require: Initial dLLM π_θ, with reference policy π_{ref}, prompt dataset $\mathcal{D}$, reward function r, hyperparameters β, ψ, time-step samples K

Initialize $\pi_{\text{old}} \leftarrow \pi_\theta$
for each training iteration **do**
  Sample a batch of prompts $\{c^{(i)}\}_{i=1}^{B}$ from $\mathcal{D}$
  // Sampling stage
  for each prompt $c^{(i)}$ **do**
    Generate G completions $\{x_0^{(i,g)}\}_{g=1}^{G} \sim \pi_{\text{old}}(\cdot|c^{(i)})$ via iterative re-masking
    Compute rewards $r(x_0^{(i,g)})$ and advantages $A(x_0^{(i,g)}) = r(x_0^{(i,g)}) - \text{mean}(r(\cdot))$
  end for
  // Training stage
  for each completion x_0 with advantage $A(x_0)$ **do**
    Sample K time steps $\{t_k\}_{k=1}^{K} \sim U(0,1)$
    for $k = 1, \ldots, K$ **do**
      Apply consistent forward process: $x_{t_k} \sim q(\cdot|x_0)$
    end for
    Compute denoising log-probabilities $\{\log p(x_0|x_{t_k})\}_{k=1}^{K}$, for all models $p = p_\theta, p_{\text{old}}, p_{\text{ref}}$.
    Apply token-level logit centralization (Equation (8)) on denoising log-probabilities (GDSD w/ TLC; skipped for direct matching).
    Compute GDSD loss $\mathcal{L}_{\text{GDSD}}$ following Equation (10), averaged over $\{x_{t_k}\}_{k=1}^{K}$
  end for
  Update θ via gradient descent on $\mathcal{L}_{\text{GDSD}}$
  Update old policy: $\pi_{\text{old}} \leftarrow \pi_\theta$ every μ iterations
end for

C Additional Details

C.1 Algorithm

Algorithm 1 summarizes Guided Denoiser Self-Distillation (GDSD). At each training iteration, GDSD first samples a batch of prompts and uses the old policy to generate multiple completions through iterative re-masking decoding. These completions are scored by the reward function, and their advantages are computed within each prompt group by subtracting the group mean reward.GDSD then converts these reward signals into denoising-level supervision. For each sampled completion, it randomly samples diffusion times and applies the same forward masking process to construct masked states. The current, old, and reference models are evaluated on these masked inputs to obtain their denoising logits. Token-level logit centralization is applied to remove irrelevant logit offsets and eliminate the need to estimate the intractable normalization constant of the energy-guided teacher. Finally, GDSD updates the current model with the practical loss in Equation (10), whose gradient equals that of matching its centralized logits to the guided teacher built from the centralized old and reference logits and the advantage signal (Proposition C.1). The old policy is periodically refreshed from the current model. Overall, GDSD turns RL into a denoiser self-distillation: rewards guide the teacher construction, while training remains a stable logit-matching problem instead of relying on ELBO-based likelihood estimation.

C.2 Additional Practical Design

In this section, we present the final practical objective for Guided Denoiser Self-Distillation (GDSD), incorporating key design choices that optimize computational efficiency. We introduce the practical objective and provide detailed explanations as follows:

Guided Denoiser Self-Distillation (GDSD) (practical)

$$\mathbb{E}_{t\sim U(0,1),\ x_0\sim\pi_{\text{old}}^{\text{rm}},\ x_t\sim q(\cdot|x_0)}\left[\left(\log\frac{\bar{p}_\theta(x_0|x_t)}{\bar{p}_{\text{old}}(x_0|x_t)}-\psi A(x_0)\right)^2+\beta\left(\log\frac{\bar{p}_\theta(x_0|x_t)}{\bar{p}_{\text{ref}}(x_0|x_t)}\right)^2\right]. \quad (24)$$

External Regularization and Relation to Equations (6) and (9). All reported results are obtained with Equation (10) (equivalently Equation (24)); the direct-matching variant replaces $\bar{p}$ by p. Equation (9) is not used as a separate loss: it shows that centralization turns the normalization constant into an x_0-independent offset b, which is then dropped. Equation (10) decouples the reference model's log-probability from the advantage-matching term as an external regularizer, a k2 KL approximation [44] as in standard RLHF pipelines, which separates the advantage-guided update from the reference constraint and eases tuning. The following proposition shows that this decoupling is an exact reparameterization of the squared logit-matching loss.

Proposition C.1 (Gradient Equivalence of the Decoupled Loss)**.** *Let $\tilde{\beta}\in[0,1)$, $\tilde{\psi}>0$, and let $l^*(x_0|x_t)=(1-\tilde{\beta})\log\bar{p}_{old}(x_0|x_t)+\tilde{\beta}\log\bar{p}_{ref}(x_0|x_t)+\tilde{\psi}A(x_0)$ be the centralized teacher logit of Equation (9) with b dropped. Then for every (x_0,x_t),*

$$\nabla_\theta\left(\log\bar{p}_\theta(x_0|x_t)-l^*(x_0|x_t)\right)^2=(1-\tilde{\beta})\,\nabla_\theta\left[\left(\log\frac{\bar{p}_\theta(x_0|x_t)}{\bar{p}_{old}(x_0|x_t)}-\psi A(x_0)\right)^2+\beta\left(\log\frac{\bar{p}_\theta(x_0|x_t)}{\bar{p}_{ref}(x_0|x_t)}\right)^2\right], \quad (25)$$

with $\psi=\tilde{\psi}/(1-\tilde{\beta})$ and $\beta=\tilde{\beta}/(1-\tilde{\beta})$. Equivalently, Equation (10) with coefficients (ψ,β) has $(1+\beta)$ times the gradient of Equation (6) with the centralized teacher and coefficients $\tilde{\psi}=\psi/(1+\beta)$, $\tilde{\beta}=\beta/(1+\beta)$. The identity holds pointwise, hence also for Monte-Carlo estimates, and holds identically when $\bar{p}$ is replaced by p (direct matching).

Proof. Let $r=\log\bar{p}_\theta-l^*$. Since $\bar{p}_{\text{old}}$, $\bar{p}_{\text{ref}}$ and A do not depend on θ, $\nabla_\theta r^2=2r\,\nabla_\theta\log\bar{p}_\theta$. Splitting $\log\bar{p}_\theta=(1-\tilde{\beta})\log\bar{p}_\theta+\tilde{\beta}\log\bar{p}_\theta$ gives $r=(1-\tilde{\beta})\left[\log\frac{\bar{p}_\theta}{\bar{p}_{\text{old}}}-\psi A+\beta\log\frac{\bar{p}_\theta}{\bar{p}_{\text{ref}}}\right]$. Since $\nabla_\theta\left[(\log\frac{\bar{p}_\theta}{\bar{p}_{\text{old}}}-\psi A)^2+\beta(\log\frac{\bar{p}_\theta}{\bar{p}_{\text{ref}}})^2\right]=2\left[\log\frac{\bar{p}_\theta}{\bar{p}_{\text{old}}}-\psi A+\beta\log\frac{\bar{p}_\theta}{\bar{p}_{\text{ref}}}\right]\nabla_\theta\log\bar{p}_\theta$, substituting gives Equation (25). □

In our experiments $\beta\le 5\times10^{-2}$ (Table 4), so $\tilde{\beta}\approx\beta$ and $\tilde{\psi}\approx\psi$.

Scope of the Guarantees. Proposition C.1 is exact. Dropping the x_0-independent offset ($\log Z_t$ for direct matching, b for TLC) does not change the denoising distribution represented by the teacher logits, by translation invariance of the Softmax (Section 3.3). This, however, is a statement about the target rather than about the loss: the student's log-probabilities cannot absorb an arbitrary offset, so the minimizer of the practical loss is not guaranteed to coincide exactly with that of Equation (6). Likewise, TLC changes the parameterization of the regression target, so its gradient is not identical to that of Equation (6) with uncentralized logits. We therefore regard Equation (10) as a normalization-free practical approximation of Equation (6) with an exact teacher distribution, rather than an unbiased estimator of it; what GDSD removes by construction is the likelihood-ratio TIM bias of ELBO-based methods.

Single Inference in Training. Obtaining denoising probabilities (e.g. $\log p_\theta(x_0|x_t)$) at different steps t requires model inference for parametrized policies. This computation leads to significant overhead if denoising probabilities at different time step t are computed via *separate* model inferences. For efficiency, we batch x_t at all sampled steps t, and compute the denoising probabilities with *single* model inference. For other models (old and reference model), we apply the same design.

Variance-Reduced and Re-weighted Logits. We follow prior work [72, 35, 17] and apply shared coupled sampling for masked sequences. Specifically, we merge the denoising logits obtained from a masked sequence with the one obtained from its complementary mask to reduce variance. We further reweight the logits using a $1/t$ schedule, which emphasizes updates near the clean sequence. Both designs lead to empirical improvements in our experiments.

D Additional Experimental Details

D.1 Histogram of Accuracy

We summarize the test accuracy of our methods compared to existing ELBO-based methods in Figure 4. Our methods GDSD consistently outperform baselines, particularly significant on Dream-7B base models.

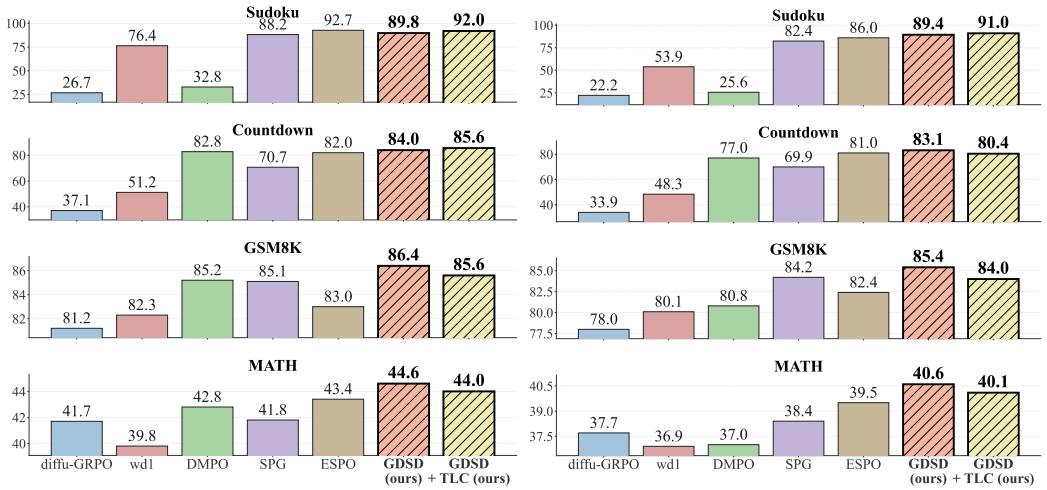


Figure 4: **Left:** Test accuracy of different methods (best across generation length 128, 256, and 512) on base model LLaDA-8B. **Right:** Average test accuracy of different methods across generation lengths, on base model LLaDA-8B.

D.2 Coding Benchmarks

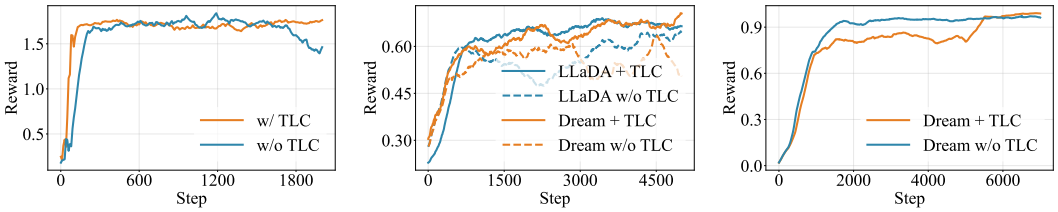
In this section, we provide additional details and additional results on coding benchmarks.

Table 3: **Performance on coding benchmarks.** Results with † denotes the results re-evaluated by pass@1 with lm-eval [16] on the checkpoints from Ou et al. [35] or trained from scratch; other baseline results are extracted from Wang et al. [55]. GDSD generally improves the average performance over ELBO-based methods.

	HumanEval(0)								MBPP(3)			
	-				Plus				-			
Model / Seq Len	**128**	**256**	**512**	**Avg.**	**128**	**256**	**512**	**Avg.**	**128**	**256**	**512**	**Avg.**
LLaDA	28.1	35.4	34.8	32.8	23.2	30.5	41.5	31.7	36.2	42.0	38.1	38.8
+ diffu-GRPO (d1)	29.3	37.8	37.2	34.8	22.0	29.9	37.2	29.7	34.8	36.6	38.0	36.5
+ wd1	25.6	39.0	38.4	34.3	29.9	29.9	32.9	30.9	38.0	37.2	34.4	36.5
+ SPG	37.2†	41.5†	44.5†	41.1	32.9†	34.2†	37.8†	35.0	40.4†	40.8†	40.4†	40.5
+ ESPO	42.1†	**48.2**†	41.4†	43.9	24.4	36.6	**42.7**	34.6	**43.6**†	43.2†	41.2†	42.7
+ **GDSD** (ours)	42.1	45.1	**45.7**	**44.3**	36.0	38.4	41.5	38.6	40.6	41.8	**43.6**	42.0
+ **GDSD w/ TLC** (ours)	**43.3**	43.9	43.3	43.5	**38.4**	**39.6**	39.6	**39.2**	43.0	**43.6**	43.2	**43.3**

D.3 Experimental Setups

We conduct RL fine-tuning based on open-sourced dLLMs, LLaDA-8B-Instruct [34] and Dream-v0-Instruct-7B [64]. We experiment on six benchmarks: GSM8K [14] and MATH500 [26] for mathematical reasoning, Countdown [38] and Sudoku [1] for logical reasoning, HumanEval [9] and MBPP [3] for coding. The training configurations of logical planning (Countdown and Sudoku) and coding (HumanEval and MBPP) tasks follow the zero-shot setting as in ESPO [35]. While for mathematical reasoning tasks, we found the format reward as d1 [67], wd1 [51], and SPG [55] assists



(a) LLaDA-8B trained on GSM8k (b) Countdown (c) Sudoku

Figure 5: Reward dynamics of GDSD with and without token logits centralization (TLC). TLC consistently improves the stability of training process and leads to a higher reward level.

in RL performance, thus we follow the reward design in SPG [55]. For coding tasks, we train the base model on AceCoder-87K [65].

Evaluation Setup. We found that the evaluation protocol varies significantly across previous literature, so we aim to unify the evaluation as much as possible for a fair comparison. For logical planning tasks (Sudoku and Countdown), we follow the zero-shot evaluation in [35] and test on generation lengths of 128, 256, and 512. The denoising steps are set to be half of the sequence length. For other tasks, the number of denoising steps is set equal to the sequence length to improve performance.

Remark. We wish to clarify different evaluation settings in various papers for the future benefit of the community. In terms of reasoning tasks and coding tasks, ESPO respectively evaluates LLaDA with OpenCompass and Dream with lmeval, and they set the diffusion steps as $1\times$ generation length. Meanwhile, other papers like d1 [67], wd1 [51], SPG [55] follow a similar evaluation with the codes provided in d1 [67] and set the diffusion steps as $0.5\times$ generation length. As for the planning tasks, both pipelines set the diffusion steps as $0.5\times$ generation length, but d1, wd1, and ESPO follow a 0-shot evaluation for Sudoku, and SPG shuffles the dataset and tests in a 3-shot setting. For fair comparison, we fix zero-shot settings for Sudoku, Countdown, GSM8k, Math500, and humenval and 3-shot settings for MBPP. We fixed the diffusion steps as $1\times$ generation length and evaluated using lmeval. For results previously reported as incompatible with such a setting, we reproduce the results either from the checkpoints given (ESPO) or re-implement the training (SPG). We summarize the evaluation settings as in section D.3

Reward Setups. We follow SPG [55] for the reward function design, which encourages both correct answers and proper formatting. We provide details as follows.

GSM8K. We follow the Unsloth reward setup[4]:

- XML Structure: +0.125 per correct formatting tag; small penalties for overlong contents after the closing tag.
- Formatting:
  (Soft) +0.5 for outputs that have the following content:
  `<reasoning>...</reasoning><answer>...</answer>`.
  (Strict) +0.5 for exact formatting.
- Validity: +0.5 if the answer is valid (an integer).
- Correctness: +2.0 if the answer is correct.

MATH500. A format reward and a correctness reward are used:

- Formatting: 1.00 if <answer></answer> tags are present with \boxed inside; 0.75 if answer tags are present without \boxed; 0.50 if answer tags are not present but \boxed is present; 0.25 if neither the answer tags nor \boxed is present.
- Correctness: 2.00 if the answer in `\boxed{}` is correct.

[4] https://unsloth.ai/blog/r1-reasoning

Table 4: Training Hyperparameters across Different Tasks.

Hyperparameter	Sudoku	Countdown	GSM8k	Math500	HumanEval	MBPP
Task Category	Planning	Planning	Reasoning	Reasoning	Coding	Coding
Num Iterations (μ)	8	8	8	8	8	8
Train Batch Size	10	10	10	8	3	3
Generation Batch Size	8	8	6	8	10	10
Num Generations	$1\times$ Number of GPUs					
Num MC (K)	2	2	2	2	4	4
Gradient Accumulation	4	4	4	4	20	20
Training Completion Length	256					
Max Prompt Length	400					
Training Diffusion Steps	128					
Training Temperature	1.0					
Learning Rate	1e-5	1e-5	3e-6	1e-5	3e-6	3e-6
Training Steps	5k	10k	3k	3k	2k	2k
Learning Rate Scheduler	constant_with_warmup					
Warmup Ratio	0.001					
Weight Decay	0.01					
Max Grad Norm	0.2	0.2	0.2	0.2	0.8	0.8
Beta (KL Coef)	1e-3	5e-4	1e-3	1e-4	5e-2	5e-2
Psi (ψ)	10.0					
Epsilon (Clip)	0.2					
Scale Rewards	false					
LoRA Rank (r)	128				Full parameter tuning	
LoRA Alpha (α)	64				Full parameter tuning	
LoRA Dropout	0.05				Full parameter tuning	
Use PEFT	true				false	
Gradient Checkpointing	false				true	

Table 5: Unified evaluation settings adopted for fair comparison across tasks and benchmarks.

Task	Sudoku	Countdown	GSM8k	Math500	HumanEval	MBPP
Task Category	Planning	Planning	Reasoning	Reasoning	Coding	Coding
Evaluation Source Code	`d1`	`d1`	`lmeval`	`lmeval`	`lmeval`	`lmeval`
Num of Shots	0-shot	0-shot	0-shot	0-shot	0-shot	3-shot
Generation Length	{128,256,512}					
Diffusion Steps	{64,128,256}		{128,256,512}			

Countdown. The reward covers three cases: +1.0 if the expression reaches the target using the exact numbers; +0.1 if the numbers are correct but does not reach the target; +0.0 otherwise.

Sudoku. The reward is based on cell-level matching against the ground-truth:

- Answer Extraction: The model is expected to output the solution with `<answer>...</answer>` tags, and the digits inside the last answer tag are extracted.
- Length Normalization: The extracted answer is normalized to length 16 for the 4×4 Sudoku puzzle. If it is shorter than 16 digits, it is padded with zeros; if it is longer, it is truncated.
- Correctness: The reward is the fraction of cells compared to originally empty ones that match the ground-truth solution:

$$r = \frac{\#\{\text{correctly filled empty cells}\}}{\#\{\text{empty cells}\}}.$$

- Missing Answer: If no valid answer is extracted, the reward is 0.0.

Coding. The reward consists of a format reward and an execution reward:

- Formatting: The model output must contain a Markdown code block in the target language, e.g., ‘‘‘`python ...` ’’’. The format reward is 1.0 if the code block is formatted correctly and the syntax is valid; 0.5 if correct code block formatting but with a syntax error; 0.0 otherwise.

- Execution: Completions with format reward 1.0 are executed. Each generated program is concatenated with the provided test cases and run in a sandboxed execution environment.
- Correctness: The execution reward is the test pass rate:

$$r = \frac{\#\{\text{passed test cases}\}}{\#\{\text{total test cases}\}}.$$

- Invalid Code: If the completion fails the format check or has invalid syntax, it is not executed and receives execution reward 0.0.

D.4 Hyperparameter Settings and Implementation Details

We follow ESPO [35] for most hyperparameter settings. LoRA (Low-Rank Adaptation) (LoRA) with rank $r = 128$ and scaling factor $\alpha = 64$ is adopted for training.

For RL rollout, we use a sequence length of 256 tokens, and 128 diffusion steps. We employ confidence-based semi-autoregressive generation with block size 32, and set the temperature as 0.9. We set the number of completions per prompt g as 6, and the number of Monte Carlo estimation samples m as 2 due to computational constraints. Since the rollout stage dominates the training time, the average time per gradient update step for is similar to that of the other baselines (Table 8).

We train 3K steps (i.e., number of gradient updates) for GSM8K and MATH500, 2K steps for coding, 5K steps for Sudoku and 10K for countdown. For all RL models, we run evaluation every 100 steps and report the result of the checkpoint with the highest average accuracy.

D.5 Additional Results Analysis

Ablation study on Token Logits Centralization. We plot the reward curve with (w/) and without (w/o) Token Logits Centralization (TLC) on several datasets in Figure 5. Results are obtained with all other hyperparameters fixed. Specifically, the curve shows that TLC consistently improves both the final attainable reward and the stability of the optimization process. Across GSM8K, Countdown, and Sudoku, adding TLC leads to a higher and more stable reward plateau, indicating that TLC helps the model obtain reward more effectively during likelihood-free optimization. We summarize the behaviour analysis below:

- **TLC reduces unstable reward optimization.** In the experiment of GSM8k (Figure 5a), the model w/o TLC reaches a competitive reward peak at an early stage, but it fails to remain in this high-reward region and later exhibits a noticeable degradation. In contrast, the TLC curve quickly rises and then stays around a high reward level, suggesting that TLC stabilizes optimization after the model discovers rewarding behaviors. A similar pattern was found in Countdown experiments (Figure 5b): without TLC, the reward curves show clear oscillations and repeated upward/downward fluctuations, while TLC substantially stabilizes the training dynamics and improves the final reward for both LLaDA and Dream.
- **TLC potentially escapes the model from local minima.** We found an interesting pattern in the reward curve for Sudoku experiments (Figure 5c): At the beginning, the model with TLC does not immediately achieve the highest reward and stays around a moderate reward level, roughly around 0.8. However, after approximately 4k training steps, the TLC undergoes a sharp reward improvement and eventually surpasses the non-TLC baseline. Our checkpoint analysis confirms that this transition corresponds to a substantial improvement in evaluation accuracy: before the jump, the model accuracy is only around 50%, while after the jump it increases to around 90%, compared with roughly 80% for the model trained without TLC. This suggests that the non-TLC run may converge to a suboptimal local basin, whereas TLC helps the model escape this basin and continue improving.

We believe these empirical results are consistent with our analysis: TLC keeps the scale of the logits from drifting while avoiding the likelihood-ratio bias of ELBO-based methods, enabling more stable optimization than ELBO-based and likelihood-based methods.

Training Reward Dynamics. According to the reward dynamics in Figure 2 and Figure 3, GDSD exhibits more stable and effective training than previous ELBO-based methods. TLC further shows slightly better convergence than direct matching in GDSD. In contrast, SPG on Countdown and ESPO on coding exhibit poor training performance.

D.6 Robustness to the Re-masking Policy

To test whether the benefit of GDSD is tied to the confidence-based re-masking used in our main experiments, we re-run RL post-training of LLaDA-8B with *random* re-masking during rollouts, keeping all other settings fixed. Table 6 summarizes the training rewards. On Sudoku, GDSD w/ TLC reaches a substantially higher and more stable reward than ESPO and SPG. On GSM8K, GDSD w/ TLC converges faster than both baselines but reaches a final reward comparable to ESPO. Together with the main results on Dream-7B (confidence-based re-masking over the full sequence) and LLaDA-8B (block-wise semi-autoregressive decoding), this indicates that GDSD remains effective across rollout policies.

Table 6: Training rewards of LLaDA-8B with *random* re-masking during rollouts.

Task	Method	Init (step 1)	Mid	Final	Late-stage mean $\pm$ std	Steps to threshold
Sudoku	GDSD w/ TLC (ours)	0.15	0.96	0.94 @ step 1120	0.94 ± 0.01	$\sim$740
(mid: step 1000;	ESPO	0.18	0.74	0.76 @ step 1060	0.76 ± 0.05	$\sim$780
threshold: 0.5)	SPG	0.12	0.24	0.39 @ step 1993	$\sim 0.25 \pm 0.01$	Never
GSM8K	GDSD w/ TLC (ours)	0.72	3.11	2.99 @ step 1000	2.88 ± 0.10	$\sim$100
(mid: step 500;	ESPO	0.72	3.12	2.98 @ step 1000	2.95 ± 0.12	$\sim$170
threshold: 2.0)	SPG	0.70	2.35	2.73 @ step 1000	2.38 ± 0.14	$\sim$165 (unstable)

D.7 Sensitivity to the Regularization Coefficient

Table 7 reports training rewards of GDSD w/ TLC on LLaDA-8B for different regularization coefficients β in Equation (10). Performance is insensitive to β for $\beta \leq 10^{-3}$, but degrades markedly at $\beta = 10^{-2}$, where the reference constraint dominates (on Sudoku, the reward never exceeds 0.27). For the guidance coefficient, Figure 3 shows that the reward improves roughly monotonically over $\psi \in [0.01, 10]$, with only overly small values being unstable. In practice, we use a single large guidance coefficient ($\psi = 10$) for all tasks and a small β (Table 4).

Table 7: Training rewards of GDSD w/ TLC on LLaDA-8B for different β. Late-stage statistics are over steps 800–1000 (GSM8K) and 900–1100 (Sudoku).

Task	β	Init (step 1)	Step 500	Final	Late-stage mean $\pm$ std	Steps to threshold
GSM8K	10^{-4}	1.08	3.32	3.25	3.17 ± 0.10	$\sim$51
(final: step 1000;	10^{-3}	1.08	3.38	3.27	3.19 ± 0.11	$\sim$50
threshold: 2.0)	10^{-2}	1.08	3.03	2.32	2.08 ± 0.23	$\sim$55
Sudoku	10^{-4}	0.17	0.27	0.65	0.48 ± 0.13	$\sim$1030
(final: step 1100;	10^{-3}	0.16	0.26	0.88	0.89 ± 0.03	$\sim$735
threshold: 0.5)	10^{-2}	0.17	0.19	0.22	0.20 ± 0.02	Never

E Others

E.1 Compute Resources

Experiments are conducted on 8 GPUs, including AMD Instinct MI308X and A800.

Computational Overhead. GDSD requires exactly the same forward passes as ELBO-based baselines: π_θ, π_{old}, and π_{ref} are evaluated on the same K masked copies, which ESPO and UniGRPO also compute to form their ELBO estimates, and GDSD differs only in the scalar loss assembled from these log-probabilities. It therefore requires no additional forward passes or model copies. Table 8 reports the measured wall-clock time per training step on 16 AMD Instinct MI308X GPUs with 160 completions per step.

E.2 Broader Impact

This work studies efficient reinforcement learning for diffusion large language models. By avoiding likelihood estimation and reverse-chain policy gradients, the proposed method can reduce the computational cost and instability of post-training, making alignment and reasoning-oriented training more accessible.

Table 8: Wall-clock time per training step (16 AMD Instinct MI308X GPUs, 160 completions per step).

Method	**Time per step (min)**
GDSD w/ TLC (ours)	0.29
ESPO	0.32
SPG	0.22

Table 9: Zero-shot test accuracy of different methods using diffusion language model Dream-7B-Instruct. We summarize the best performance across generation length in the right figure. GDSD demonstrates significant improvements over ELBO-based methods, achieving up to $+19.6\%$.

	Sudoku				**Countdown**			
Model / Seq Len	**128**	**256**	**512**	**Avg.**	**128**	**256**	**512**	**Avg.**
Dream-7B	9.3	2.1	14.0	8.5	8.5	7.8	17.4	11.2
+ diffu-GRPO (d1)	64.4	69.7	51.1	61.7	27.3	27.7	37.5	30.8
+ wd1	29.5	39.2	30.3	33.0	28.9	37.9	42.2	36.3
+ ESPO	71.7	72.3	71.3	71.8	68.8	66.8	64.8	66.8
+ **GDSD-fixed** (ours)	91.1	91.3	91.7	91.4	**83.2**	**82.4**	**84.8**	**83.5**
+ **GDSD w/ TLC** (ours)	**93.1**	**93.3**	**92.8**	**93.1**				

However, stronger and cheaper post-training may also amplify risks associated with language models, including misinformation, spam, academic dishonesty, automated persuasion, and harmful content generation. Since our method relies on reward signals, misspecified rewards may reinforce biases, unsafe behaviors, or undesirable shortcuts.

Models trained with our method should therefore be evaluated beyond reasoning and coding benchmarks, including safety, robustness, bias, hallucination, and misuse-related evaluations. Deployment should be paired with safeguards such as data curation, red-teaming, monitoring, and access controls when appropriate.