DiPOD: Diffusion Policy Optimization without Drifting Apart

Anonymous authors
Paper under double-blind review

Abstract

RL post-training has become increasingly pivotal for improving diffusion policies, but existing diffusion policy-gradient methods are often unstable and cannot achieve reliable policy improvement. We identify the cause as the double-drift phenomenon: optimizing a variational surrogate can let the ELBO separate from the true log-likelihood, which then makes the resulting proxy policy gradient misaligned with the true policy gradient of expected return. We propose **DiPOD**, a diffusion policy optimization framework that maintains tight-bound behavior throughout training by interleaving self-distillation with policy-improving gradient updates. This leads to a simple and practical algorithm: augmenting each diffusion policy-gradient update with an on-policy ELBO regularizer. Across diffusion language model post-training and continuous-control diffusion policies, DiPOD substantially stabilizes training and reaches higher rewards than previous methods.

1 Introduction

Diffusion models are emerging as a powerful paradigm for discrete generation in language, code, and mathematical reasoning, while diffusion and flow models more broadly support strong performance in continuous domains such as images (Ho et al., 2020; Song et al., 2020; Song & Ermon, 2019), videos (Ho et al., 2022b;a), and robotic control (Black et al., 2024; Bjorck et al., 2025). This promise is especially compelling for reasoning with diffusion LLMs (dLLMs) (Nie et al., 2025; Xie et al., 2025; Song et al., 2025), which offer fast parallel sampling and flexible non-autoregressive decoding (Jiang et al., 2025; Ermon, 2026; Chen et al., 2026a). Yet post-training diffusion policies with reinforcement learning remains fundamentally difficult: standard policy-gradient methods rely on the log-likelihood $\log \pi_\theta(a|o)$, while for diffusion models this quantity is generally not tractable. Recent methods (Zhao et al., 2025; McAllister et al., 2025; Wang et al., 2025) have therefore introduced a range of more tractable proxies for the likelihood, but these approaches still do not provide a systematic understanding of when proxy-based policy-gradient updates remain reliable. This gap is particularly consequential for dLLMs, where unstable RL post-training causes reasoning gains to lag far behind those of autoregressive counterparts.

In this work, we propose **DiPOD**—**Di**ffusion **P**olicy optimization with**O**ut **D**rifting apart—a principled framework for **reliable policy-gradient updates** that is **native to diffusion and flow policies**. When applied to dLLM post-training, DiPOD substantially stabilizes learning and improves reasoning performance on benchmarks such as GSM8K, MATH500, Countdown, and Sudoku, including being the first method to saturate Sudoku in the zero-shot setting (Section 4.2 and Table 1).

DiPOD is designed to satisfy two desiderata: (A) *improve expected return* and (B) *preserve the diffusion-model structure by keeping the proxy objective tight*. Our analysis will show how the second supports the first by linking bound tightness to policy-gradient accuracy.

Variational-Inference Approaches to RL Post-training. Why is updating diffusion with policy gradient hard in the first place? Treating the denoising chain as an MDP makes likelihoods tractable (Black et al., 2023; Ren et al., 2024), but it ties training to a sampler and is a poor fit for dLLMs, whose post-training should preserve flexible decoding orders and inference budgets. Recent *non-MDP* dLLM methods therefore approximate likelihoods directly, using mean-field or one-step

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

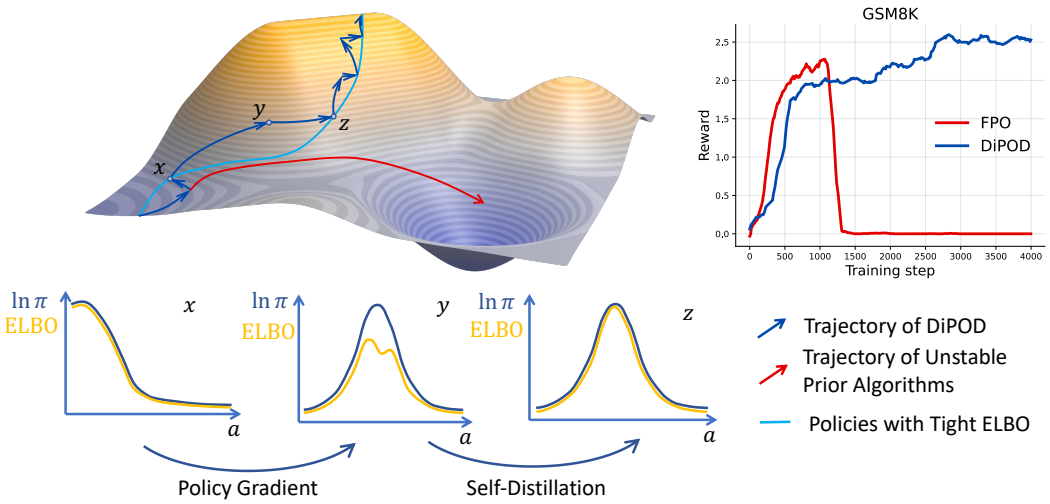


Figure 1: **Illustration of DiPOD** using an expected-return landscape over the parameter space of the policy. We want the policy to both guarantee policy improvement and have a tight ELBO, as illustrated by the colored curves in the figure. Prior proxy-based algorithms (the plotted reward curve comes from a real FPO (McAllister et al., 2025) experiment on GSM8K (Cobbe et al., 2021); see Figure 3) can initially achieve policy improvements, but later become unstable as the ELBOs are no longer tight. In contrast, DiPOD interleaves self-distillation steps and adequate gradient updates, as illustrated by the blue arrows. Gradient updates guarantee policy improvements locally, and self-distillation brings the parameter to where the ELBO is tight while keeping the expected reward unchanged.

estimators, partial dependency restorations, or imposed autoregressive orders (Zhao et al., 2025; Xie et al., 2025); these can alter the optimized policy from the executed diffusion policy. Variational-inference (VI) approaches instead replace $\log \pi_\theta$ with evidence bounds such as ELBO (McAllister et al., 2025; Ou et al., 2026; Gong et al., 2026) or EUBO (Wang et al., 2025), preserving diffusion's native sampling flexibility while connecting naturally to pretraining objectives.

The key caveat is that this faithfulness is only local. Variational diffusion-RL methods can work well *initially*: near a well-pretrained initialization where the evidence bound is tight, ELBO remains a good local proxy for the true log-likelihood. However, RL updates do not preserve this tight-bound regime. As optimization proceeds, the discrepancy between ELBO and log-likelihood can grow, and our analysis pinpoints the Achilles heel: once ELBO drifts from log-likelihood, policy-gradient fidelity drifts with it. We refer to this coupled effect as the **double drift** phenomenon.

DiPOD exploits the fact that many existing VI-based gradient estimators are *adequate* (Definition 2.1): when the evidence bound is tight, they recover the true policy-gradient integrand. We interleave policy-gradient updates with self-distillation on reference-policy rollouts (Figure 1). In the idealized setting, self-distillation tightens the bound while preserving the policy's output distribution, restoring the condition under which subsequent updates improve expected return.

Practically, we implement a simple approximation of this interleaving: for each batch of rollouts, we update parameters using the usual diffusion policy gradient estimator *plus* an ELBO maximization term on the same rollout data. This additional ELBO term acts as a regularizer that reduces variational discrepancy on-policy, improving alignment between the proxy gradient and the true policy gradient. Empirically, this simple DiPOD implementation is surprisingly effective: it substantially stabilizes training and improves reward optimization across both discrete dLLM post-training tasks (e.g., GSM8K Cobbe et al. (2021), MATH500 Lightman et al. (2024), Sudoku Arel (n.d.), Countdown Pan et al. (2025)) and continuous-control flow policies (e.g., motion tracking for humanoid robots; details in Section F).

In summary, we make three contributions:

- We identify a fundamental failure mode in variational diffusion RL: **double drift**. As RL updates loosen the evidence bound, ELBO drifts from log-likelihood, which in turn causes proxy policy-gradient updates to drift from the intended policy-gradient direction (Section 3.1).
- We propose **DiPOD**, a principled framework for reliable diffusion policy optimization that makes staying in the tight-bound regime an explicit design objective, achieved by interleaving policy-gradient updates with policy-preserving self-distillation (Section 3.2).
- We derive a **simple, drop-in practical algorithm** that instantiates this principle by adding a per-update ELBO regularizer, substantially improving the stability and reward optimization of diffusion RL in practice (Algorithm 2 and Section 4).

1.1 Related Work

Diffusion models. Diffusion models are a powerful class of generative models, with applications in image generation (Ho et al., 2020; Song et al., 2020; Song & Ermon, 2019), video generation (Ho et al., 2022b;a), and robotics (Black et al., 2024; Bjorck et al., 2025). More recently, diffusion language models have emerged as a promising alternative to autoregressive models for fast and flexible decoding (Nie et al., 2025; Xie et al., 2025; Song et al., 2025). We study diffusion policies through the variational-inference perspective (Kingma & Gao, 2023; Sohl-Dickstein et al., 2015), which is the lens used by DiPOD to reason about likelihood proxies and evidence bounds.

Policy gradients. Policy-gradient methods directly optimize a parameterized policy for expected return (Williams, 1992; Schulman et al., 2017), and have been central to high-dimensional control (Schulman et al., 2016), locomotion (Rudin et al., 2022), manipulation (Schwarke et al., 2023), and modern language-model post-training (Shao et al., 2024). DiPOD targets this online policy-optimization setting for diffusion policies.

RL with diffusion policies. The main obstacle to applying policy gradients to diffusion policies is that the exact log-likelihood is intractable. Prior work either treats the denoising process itself as an MDP (Black et al., 2023; Ren et al., 2024), replaces likelihoods with variational evidence bounds (McAllister et al., 2025; Wang et al., 2025; Ou et al., 2026; Gong et al., 2026), or simplifies dependencies in diffusion language models to obtain tractable likelihood surrogates (Zhao et al., 2025; Yang et al., 2025; Xie et al., 2025; Tang et al., 2025). These approaches trade off tractability, sampler flexibility, and alignment between the optimized proxy and the executed diffusion policy. DiPOD focuses on the variational line and stabilizes it by keeping the evidence bound aligned with the true likelihood during policy optimization. A fuller discussion of related work appears in Section A.

2 A Variational Inference View to Policy Gradient

Policy Gradient Method. The goal of reinforcement learning (RL) is to learn a policy π_θ that maximizes the expected return in an environment. Here θ denotes the parameters of the policy. At timestep t, the policy takes an observation o_t from the environment, then executes action $a_t \sim \pi_\theta(\cdot|o_t)$, and the environment provides reward r_t as feedback. The return is defined as the cumulative reward until the environment reaches a terminal state. A policy gradient algorithm updates the policy parameters θ using an estimator of the gradient of the expected return. Most policy gradient methods can be viewed as using a variant of the following gradient estimator

$$\mathbb{E}_\theta \left[\nabla_\theta \log \pi_\theta(a_t|o_t) \hat{A}_t(o_t, a_t) \right], \tag{1}$$

where $\hat{A}_t$ is the estimated advantage function at timestep t. The expectation is taken over the randomness of the environment and the policy. Here $\mathbb{E}_\theta$ means that a_t is generated according to $\pi_\theta(\cdot|o_t)$. Well-known practical algorithms such as PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024) can be understood as introducing modifications to this update in order to improve stability and sample efficiency.

Estimating Equation (1) becomes intractable when π_θ is parameterized by a diffusion or flow model, motivating a range of proxy objectives, including those based on variational bounds.

Variational Bounds. The evidence lower bound (ELBO) satisfies

$$\text{ELBO}_\theta(a_t|o_t) = \log \pi_\theta(a_t|o_t) - \mathcal{D}^{\text{L}}_\theta(o_t, a_t) \tag{2}$$

where $\mathcal{D}^{\text{L}}_\theta$ is a non-negative discrepancy term that measures the gap between the ELBO and true log-likelihood. Furthermore, with appropriate and well-trained θ, $\mathcal{D}^{\text{L}}_\theta(o_t, a_t) = 0$ for all o_t and a_t.[1] This means that ELBO is a tight lower bound of the true log-likelihood. Outside the reinforcement learning context, generative models are usually trained by maximizing

$$\mathbb{E}_{o_t, a_t \sim p_{\text{data}}} \left[\text{ELBO}_\theta(a_t|o_t)\right] \tag{3}$$

where p_{data} is the distribution that the model tries to learn. Such surrogate objectives are useful because they avoid direct likelihood computation. For diffusion models, under idealized assumptions, perfect training makes the ELBO tight, so $\text{ELBO}_\theta(a_t|o_t) = \log \pi_\theta(a_t|o_t)$. For flow models trained by conditional flow matching (CFM), the analogous idealized optimum has zero CFM loss, so the learned vector field matches the target conditional vector field.

Sometimes it is also useful to consider the corresponding upper bound on log likelihood. The evidence upper bound (EUBO) satisfies

$$\text{EUBO}_\theta(a_t|o_t) = \log \pi_\theta(a_t|o_t) + \mathcal{D}^{\text{U}}_\theta(o_t, a_t)$$

where $\mathcal{D}^{\text{U}}_\theta$ is a non-negative discrepancy term that can reach zero with appropriate θ as well. While $\mathcal{D}^{\text{L}}_\theta$ can be interpreted as a KL discrepancy between posterior distributions, $\mathcal{D}^{\text{U}}_\theta$ is characterized by a Renyi-type posterior discrepancy. Furthermore, if $\text{ELBO}_\theta(a_t|o_t) = \log \pi_\theta(a_t|o_t)$ for pair (o_t, a_t), EUBO also satisfies $\text{EUBO}_\theta(a_t|o_t) = \log \pi_\theta(a_t|o_t)$. Hence, if a diffusion model is perfectly trained, we have $\text{ELBO}_\theta(a_t|o_t) = \text{EUBO}_\theta(a_t|o_t) = \log \pi_\theta(a_t|o_t)$ for all o_t and a_t. As opposed to the ELBO, however, EUBO itself is not tractable for diffusion models and must be approximated in practice.

Adequate Gradient Estimators. Variational bounds provide attractive surrogates for the intractable log-likelihood because they can be tight. As we will see in the following sections, some variational-inference-based (VI-based) gradient estimators satisfy a useful property in this tight regime: once the surrogate coincides with the true log-likelihood, the estimator also coincides with the policy-gradient integrand in Equation (1). We call such estimators adequate.

Definition 2.1 (Adequate estimator)**.** A gradient estimator $g_\theta(o_t, a_t)$ is called *adequate* if, for any (o_t, a_t), whenever the evidence bound is tight so that $\text{ELBO}_\theta(a_t|o_t) = \log \pi_\theta(a_t|o_t)$, it satisfies

$$g_\theta(o_t, a_t) = \hat{A}_t(o_t, a_t) \nabla_\theta \log \pi_\theta(a_t|o_t).$$

3 METHOD

As discussed in the previous section, diffusion policies allow efficient sampling but make the exact log-likelihood $\log \pi_\theta(a_t|o_t)$ (and its gradient) intractable, forcing RL for diffusion models to rely on proxy objectives/estimators. Our introduction diagnosed a **double drift** mechanism that is particularly acute for *variational-inference (VI)* based diffusion RL: (i) the ELBO can drift from the true log-likelihood as the variational discrepancy grows, and consequently (ii) the proxy policy gradient will drift apart from the true policy gradient of expected return. In this section, we first use this double-drift lens to clarify limitations of representative diffusion-RL algorithms (Section 3.1). We then develop a principled framework that *prevents drifting* by keeping the evidence bound tight on-policy, which in turn keeps policy-gradient updates aligned with the true policy gradient. Finally, we derive a simple practical algorithm that implements this principle as a drop-in regularization term.

3.1 THE DOUBLE-DRIFT PHENOMENON

Many diffusion-RL methods differ in formulation, but the key question under our lens is: *does the proxy remain faithful to* $\log \pi_\theta$*, and does its induced gradient remain faithful to the true policy gradient?* For VI-based approaches, the core issue is that an evidence bound update can change both the likelihood term and the variational gap, and once the gap grows, the proxy gradient inevitably deviates from $\nabla_\theta \log \pi_\theta$.

[1] Here we assume that θ has enough representation power to cover the zero discrepancy case.

FPO (McAllister et al., 2025). FPO replaces the intractable likelihood score in the policy-gradient update with an ELBO score. Formally, FPO uses the gradient estimator

$$g_\theta^{\text{FPO}}(o_t, a_t) = \nabla_\theta \text{ELBO}_\theta(a_t|o_t)\hat{A}_t(o_t, a_t), \tag{4}$$

up to PPO-style clipping for stability. The intuition is that the ELBO acts as a tractable proxy for the log-likelihood score: in particular, increasing the ELBO on positive-advantage samples should both increase the likelihood of desirable actions and push the diffusion model closer to a tight-bound regime. However, since ELBO decomposes as the true log-likelihood minus a nonnegative discrepancy term (cf. Equation (2)), ELBO changes generally *do not uniquely determine* how $\log \pi_\theta(a_t|o_t)$ changes. This creates the first drift: ELBO–likelihood inconsistency. Concretely:

- The ELBO may increase even if $\log \pi_\theta(a_t|o_t)$ decreases, provided the discrepancy decreases more. This ambiguity is especially problematic for negative-advantage updates: the ELBO can be reduced by *increasing* the discrepancy, i.e., "cheating" by destabilizing the diffusion model rather than reliably decreasing the true likelihood of undesirable actions.
- Once the discrepancy is non-negligible, the second drift follows automatically: the proxy gradient used in FPO is $\nabla_\theta \text{ELBO}_\theta(a_t|o_t)$, but $\nabla_\theta \text{ELBO}_\theta = \nabla_\theta \log \pi_\theta - \nabla_\theta \mathcal{D}_\theta^{\text{L}}$. Thus, the update direction can become partially spent on changing the variational gap rather than following $\nabla_\theta \log \pi_\theta$, and the induced policy-gradient step may not align with the true policy gradient in Equation (1).

These issues help explain why ELBO-based diffusion RL can be unstable in practice: once ELBO drifts from likelihood, the proxy gradient can drift from the true policy gradient as well.

SPG (Wang et al., 2025). SPG explicitly targets an objective that mirrors the policy-gradient structure by using ELBO for positive advantages and EUBO for negative advantages. Formally, SPG uses the gradient estimator

$$g_\theta^{\text{SPG}}(o_t, a_t) = \mathbb{1}_{\hat{A}>0} \nabla_\theta \text{ELBO}_\theta(a_t|o_t)\hat{A} + \mathbb{1}_{\hat{A}<0} \nabla_\theta \text{EUBO}_\theta(a_t|o_t)\hat{A}, \tag{5}$$

where $\hat{A}$ abbreviates $\hat{A}_t(o_t, a_t)$. The objective corresponding to the SPG update (Equation (5)) is a lower bound on the original objective:

$$\mathbb{E}_\theta \left[\mathbb{1}_{\hat{A}>0} \text{ELBO}_\theta(a_t|o_t)\hat{A} + \mathbb{1}_{\hat{A}<0} \text{EUBO}_\theta(a_t|o_t)\hat{A} \right] \tag{6}$$

because ELBO is a lower bound and EUBO is an upper bound on the true log-likelihood. Moreover, SPG has a nice property: when the objective is maximized, ELBO and EUBO equal the log-likelihood, and the expected return is maximized as well. However, under the double-drift lens it still has drawbacks:

- Similar to the exact likelihood, EUBO is not tractable. The SPG implementation adopts an approximation to EUBO such that Equation (6) remains a true lower bound. This lower-bound property often stabilizes training, but the approximation can break the exact link to the idealized objective-level argument and is currently specialized to certain settings.
- Most importantly for our purposes: an objective-level guarantee does **not** imply gradient consistency. Even if an objective becomes exact at convergence, there is no guarantee that intermediate gradient ascent steps computed from variational bounds align with the true policy gradient in Equation (1) when the bound is not tight (i.e., when the discrepancy is non-negligible).

3.2 Preventing Double Drift through Self-Distillation

We now make the core principle precise: to prevent the second drift (proxy-gradient drift), we must prevent the first drift (ELBO–likelihood drift) from accumulating along the training trajectory. Concretely, when the diffusion model is perfectly trained (or close to it), the evidence bound is tight and gradients of variational bounds coincide with gradients of the true log-likelihood.

Let us take ELBO as an example. Recall from Equation (2) that ELBO is a tight lower bound of the true likelihood and the discrepancy $\mathcal{D}_\theta^{\text{L}}(o_t, a_t)$ is minimized at zero. Assuming $\mathcal{D}_\theta^{\text{L}}(o_t, a_t)$ is smooth in θ, at a tight-bound point we have $\nabla_\theta \mathcal{D}_\theta^{\text{L}}(o_t, a_t) = 0$, and thus

$$\text{ELBO}_\theta(a_t|o_t) = \log \pi_\theta(a_t|o_t), \Rightarrow \nabla_\theta \text{ELBO}_\theta(a_t|o_t) = \nabla_\theta \log \pi_\theta(a_t|o_t).$$

Hence, g_θ^{FPO} is adequate, and if we start from a well-trained diffusion model (e.g., optimized under Equation (3)), the *initial* ELBO-based update direction in FPO aligns with the true policy gradient in Equation (1), and it remains a good approximation as long as training stays near the tight-bound regime.

A similar logic applies to SPG. When the diffusion model is perfectly trained, $\mathcal{D}_\theta^{\text{L}}(o_t, a_t) = \mathcal{D}_\theta^{\text{U}}(o_t, a_t) = 0$, and hence

$$\nabla_\theta \text{ELBO}_\theta(a_t|o_t) = \nabla_\theta \text{EUBO}_\theta(a_t|o_t) = \nabla_\theta \log \pi_\theta(a_t|o_t),$$

meaning that g_θ^{SPG} is adequate, and this justifies the strong empirical performance of VI-based RL methods for diffusion models when initialized from a pretrained model: their gradients are *adequate* at initialization. The core issue is that RL updates can move the model away from this tight-bound regime, allowing the discrepancy to grow; once that happens, ELBO drifts from likelihood and the proxy gradient drifts from the true policy gradient.

Key idea: repeatedly tighten the bound on-policy. We propose to *interleave* policy-gradient updates (using an adequate estimator) with a policy-preserving self-distillation step that tightens ELBO under a *reference rollout distribution*. Intuitively, self-distillation directly counters the diffusion-side drift by reducing the discrepancy on-policy; this in turn restores the local tightness that makes the proxy gradient a faithful substitute for $\nabla_\theta \log \pi_\theta$, preventing the RL-side drift.

Algorithm 1 DiPOD (Interleave)

Input: An adequate gradient estimator g, a policy parameterized by diffusion model π_θ, $n \in \mathbb{N}^+$, learning rate $\eta \in \mathbb{R}^+$
Output: Policy π_θ

```
Initialize policy π_θ
Set π_ref ← π_θ
while π_θ has not converged do
    Maximize E_ref [ELBO_θ(a_t|o_t)].                ▷ Self-distillation
    for i = 1, ..., n do
        θ ← θ + ηE_θ [g_θ(o_t, a_t)].                ▷ Policy update
    end for
    Set π_ref ← π_θ
end while
return π_θ
```

Algorithm 1 alternates between (i) tightening the evidence bound *under rollouts from the latest reference policy* ($\mathbb{E}_{\text{ref}}$ means $a_t \sim \pi_{\text{ref}}(\cdot|o_t)$), and (ii) taking policy updates using an adequate estimator. In the idealized limit where the self-distillation step is optimized to convergence, it produces a diffusion model with (approximately) zero discrepancy on the reference rollout distribution while preserving the reference policy's output distribution; the subsequent policy update is then well-aligned with the true policy-gradient direction. Crucially, we refresh π_{ref} to be the *most recent* policy, so the self-distillation step continually patches on-policy drift rather than distilling toward a stale reference.

In particular, under standard smoothness, realizability and optimization assumptions, we can theoretically prove the following theorem.

Theorem 3.1 (Informal)**.** *Assume every self-distillation step returns a model that is within ε of the optimal ELBO value on the current reference distribution. Choosing a sufficiently small learning rate η ensures that each DiPOD policy update improves the expected return, unless the current policy gradient is already $O(\sqrt{\varepsilon})$. If the procedure stops after an ε-optimal on-policy self-distillation step at $\theta_\star$, then both $\|\nabla_\theta \mathcal{J}(\theta_\star)\| = O(\sqrt{\varepsilon})$ and the ELBO discrepancy under the final reference distribution is small: $\mathbb{E}_{(o,a)\sim\rho_{\theta_\star}}[\mathcal{D}_{\theta_\star}^{\text{L}}(o, a)] \le \varepsilon$.*

The theorem explains how our two desiderata fit together: keeping the ELBO tight supports reliable improvements in expected return. Under its assumptions, DiPOD can keep improving while the true policy gradient is sufficiently large. At a stable endpoint, both the true gradient and the ELBO gap are small, so the policy is approximately stationary and the likelihood proxy remains accurate. The formal statement and proof appear in Theorem C.8.

Self-Distillation beyond Acceleration. A prominent use of self-distillation in diffusion language models is to accelerate inference while retaining generation quality: methods such as SDTT (Deschenaux & Gulcehre, 2025), dParallel (Chen et al., 2026b), and T3D (Zhang et al., 2026) train models to generate with fewer decoding steps. DiPOD identifies another role for self-distillation: supporting reliable policy optimization. In our idealized formulation, maximizing the ELBO on

reference-policy samples tightens the variational bound while preserving the reference policy's output distribution. This restores the condition under which adequate estimators recover the true policy gradient, enabling subsequent updates to improve expected return. Our practical on-policy ELBO regularizer approximates this mechanism during training. Self-distillation thus serves a purpose beyond reducing inference cost: keeping likelihood proxies accurate as the policy changes, so that learning can continue to improve the policy.

3.3 A Simple and Practical Implementation

Although Algorithm 1 directly addresses double drift, a literal implementation can be inefficient if self-distillation is run to convergence before every policy update. We therefore introduce a simple approximation that is efficient, easy to implement, and effective in practice.

In a typical diffusion RL algorithm (such as FPO and SPG), each gradient step uses a batch of rollouts $\mathcal{B} = \{(o^i, a^i)\}_{i=1,\dots,m}$ and updates parameters as

$$\theta \leftarrow \theta + \eta \cdot \frac{1}{m} \sum_{i=1}^{m} g_\theta(o^i, a^i).$$

We absorb self-distillation into each update by augmenting the policy-update direction with an ELBO maximization term computed on the same rollout batch:

$$\theta \leftarrow \theta + \eta \cdot \frac{1}{m} \sum_{i=1}^{m} \left[g_\theta(o^i, a^i) + \beta \nabla_\theta \mathrm{ELBO}_\theta(a^i | o^i) \right], \tag{7}$$

where $\beta \in \mathbb{R}^+$ controls the strength of the on-policy ELBO tightening.

Equation (7) can be viewed as a first-order approximation to performing (i) a small amount of self-distillation and (ii) a policy-gradient update per rollout batch, sharing the same samples for efficiency. Equivalently, the added ELBO term acts as a regularizer that reduces the variational discrepancy $\mathcal{D}_\theta^{\mathrm{L}}$ over on-policy rollouts, tightening the likelihood approximation and thereby mitigating the *first drift* (ELBO–likelihood drift). As a consequence, the proxy gradient used in g_θ remains better aligned with $\nabla_\theta \log \pi_\theta$, mitigating the *second drift* (proxy-gradient drift).

We summarize the resulting procedure in Algorithm 2. Relative to a standard VI-based diffusion RL pipeline, the only modification is the extra ELBO term in the θ update, making the method a simple drop-in enhancement for a broad class of variational-bound-based diffusion RL algorithms.

Algorithm 2 DiPOD (Practical Implementation)

Input: An adequate gradient estimator g, a policy parameterized by diffusion model π_θ, $\beta \in \mathbb{R}^+$, learning rate $\eta \in \mathbb{R}^+$, rollout size $m \in \mathbb{N}^+$
Output: Policy π_θ
1: Initialize policy π_θ
2: **while** π_θ has not converged **do**
3: Sample rollouts $\mathcal{B} = \{(o^i, a^i)\}_{i=1,\dots,m}$
▷ Rollouts may be sampled from the current policy.
4: Update θ using Equation (7) with $\mathcal{B}$.
5: **end while**
6: **return** π_θ

Although we present our method in the context of diffusion policies, the proposed principle is not specific to diffusion models. At a high level, our algorithm only requires (i) a tractable variational lower bound of the form $\mathrm{ELBO}_\theta(x|c)$ and (ii) an RL objective that depends on an intractable likelihood only through policy-gradient-style updates. Consequently, the same "tighten-the-bound to prevent gradient drift" strategy applies to a broad class of generative policies trained with ELBO objectives, including latent-variable models such as VAEs and other variational generative models.

Relation to KL Regularization. Many RL methods stabilize learning by regularizing policy changes relative to a reference policy π_{ref}, using mechanisms such as KL penalties, trust-region

constraints, or probability-ratio clipping (Schulman et al., 2015; 2017). DiPOD's ELBO regularization additionally tightens the likelihood bound. In the idealized self-distillation step, this tightening preserves the reference policy's output distribution while restoring accurate policy-gradient estimation. Policy proximity alone does not ensure a tight bound. The mechanisms are therefore distinct and compatible: regularization toward a reference policy controls policy changes, while ELBO regularization supports the accuracy of the gradients used to make them.

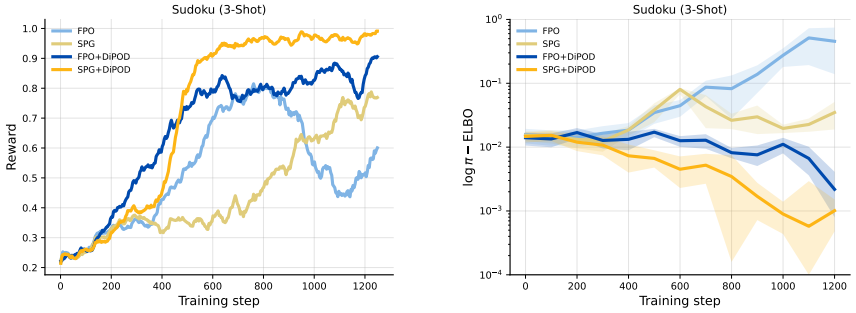
4 Experiments

We first examine double drift in a three-shot Sudoku case study, then evaluate DiPOD on language reasoning benchmarks. We also include continuous-control experiments in Appendix F and an illustrative two-token experiment in Appendix D.

4.1 Case Study: Double Drift on Three-Shot Sudoku

Setup. We compare FPO (McAllister et al., 2025) and SPG (Wang et al., 2025), each with and without DiPOD, on three-shot Sudoku using LLaDA-8B-Instruct (Nie et al., 2025). All four runs use training seed 42, 1,250 optimizer steps, and a generation limit of 256 tokens, with $\beta = 0.05$ for DiPOD. At 13 checkpoints, we use 24 on-policy completions to obtain direct Monte Carlo estimates of the log-likelihood–ELBO gap per token. Estimating this gap is computationally expensive (about 16 minutes per completion on one H200 GPU). These finite-sample estimates do not make exact log-likelihood evaluation tractable. Evaluating on-policy samples tracks bound tightness on the changing rollout distribution that drives policy updates, making this a diagnostic of the double drift that affects policy-gradient accuracy (Section 3.1). The evaluation model, sampling procedure, and uncertainty estimates are detailed in Section E.3.

Results. In Figure 2, FPO's reward initially improves and then declines as the estimated ELBO gap grows, while FPO+DiPOD maintains a smaller gap and higher late-training reward. SPG+DiPOD likewise shows a smaller late-training gap and higher reward than SPG. These joint trends are consistent with the double-drift mechanism; the estimated gap serves as a diagnostic of bound tightness.



(a) On-policy training reward.

(b) Estimated ELBO gap per token.

Figure 2: Three-shot Sudoku case study. Left: on-policy reward with a 10-observation moving average. Right: estimated log-likelihood–ELBO gap per token under the block diffusion evaluation model, on a logarithmic axis with 95% resampling intervals. Both panels show the same four training runs; measurement details are in Section E.3.

4.2 Reasoning Tasks with Diffusion Language Models

Setup. We follow the basic experimental setups in d1 (Zhao et al., 2025) and SPG (Wang et al., 2025). We start from LLaDA-8B-Instruct (Nie et al., 2025), an open-source pretrained diffusion large language model, and conduct RL experiments on it. We include four tasks: GSM8K (Cobbe et al., 2021), MATH500 (Lightman et al., 2024), Countdown (Pan et al., 2025), and Sudoku (Arel, n.d.).

Table 1: Blue numbers denote improvements over the corresponding baseline (e.g., FPO+DiPOD vs. FPO, SPG+DiPOD vs. SPG). Evaluation follows the protocol of SPG (Wang et al., 2025). We adopt the reported results for d1 from the SPG paper, except for Sudoku, where we report our reproduced d1 result.

Method	GSM8K	MATH500	Countdown	Sudoku
d1	80.60	36.00	30.90	22.10
FPO	81.50	36.60	12.50	6.88
FPO+DiPOD	83.24 (+1.74)	37.00 (+0.40)	75.39 (+62.89)	25.78 (+18.90)
SPG	84.23	37.80	51.95	25.12
SPG+DiPOD	**84.91** (+0.68)	**40.00** (+2.20)	**80.08** (+28.13)	**97.56** (+72.44)

Baselines and Hyperparameters. We evaluate the performance of Algorithm 2 with the FPO gradient estimator and with the SPG gradient estimator separately. For FPO experiments, we compare d1, FPO, and DiPOD (Algorithm 2) with the FPO gradient estimator. We implement FPO by replacing the log likelihoods in d1 with ELBOs, resulting in a GRPO version of FPO, while keeping the hyperparameters unchanged. We also keep the hyperparameters in DiPOD the same as d1. Similarly, for SPG experiments, we compare d1, SPG, and Algorithm 2 with the SPG gradient estimator. We use SPG with Mixture, which performs the best among all variants in the SPG paper. For DiPOD, we keep the hyperparameters the same as SPG. **For the main language experiments at sequence length** 256**, we fix** $\beta = 0.05$ **in Equation (7), highlighting the consistent improvements of DiPOD over baseline algorithms.** We include an ablation study on β in the appendix. Additional results at sequence lengths 128 and 512, including their task-specific β settings, are reported in Section E.1. For the main experiments, we fix the sequence length of the diffusion language model to be 256 and the number of decoding steps to be 128. For all benchmarks, we evaluate the performance in the zero-shot setting.

Results. We summarize results in Table 1, and show reward dynamics in Figure 3. The table and figures show that FPO+DiPOD consistently improves the performance over FPO. Compared to SPG, the state-of-the-art algorithm in diffusion language model RL, SPG+DiPOD, shows competitive performance in math reasoning tasks, including GSM8K and MATH500, and achieves a significant leap in logical reasoning tasks, including Countdown and Sudoku. For mathematical reasoning, we conjecture that the fixed context window of diffusion language models is the primary bottleneck for solving harder problems, which fundamentally require longer chains of thought. Consequently, we observe only marginal improvements on GSM8K and MATH500. To ensure the fairness of the comparison, we keep the random seeds the same as in the SPG codebase.

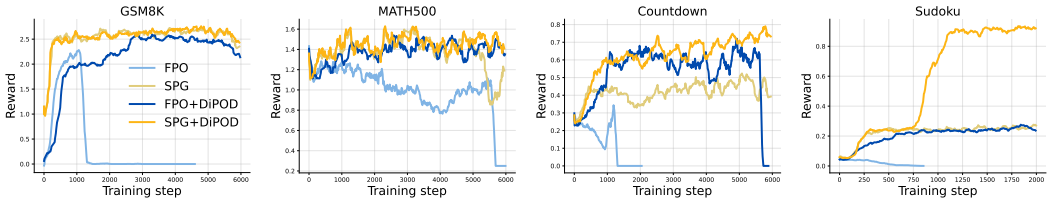


Figure 3: Reward dynamics on reasoning tasks. DiPOD stabilizes FPO and SPG training and achieves competitive or better final performance.

5 Conclusion

We identify a double drift issue in diffusion reinforcement learning: loose variational bounds induce proxy gradients that drift from the true policy gradient, undermining policy improvement. DiPOD preserves gradient consistency by maintaining on-policy bound tightness. Its most notable empirical outcome is substantially improved reasoning capability and training stability for diffusion language models, with additional gains in continuous control. Future work should optimize the interleaving schedule and study alternatives to ELBO regularization.

AI USE STATEMENT

We used generative AI tools to improve the clarity of the writing and generate visualizations. The DiPOD code was written by the human authors. We take responsibility for all content in this paper, including material produced with AI assistance.

REFERENCES

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement-residual rl for precise assembly. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 01–08. IEEE, 2025.

Arel. Arel's sudoku generator, n.d. URL `https://www.ocf.berkeley.edu/~arel/sudoku/main.html`. Accessed January 26, 2026.

Marianne Arriola, Aaron Gokaslan, Justin Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, volume 2025, pp. 50726–50753, 2025.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Jian Chen, Yesheng Liang, and Zhijian Liu. Dflash: Block diffusion for flash speculative decoding. *arXiv preprint arXiv:2602.06036*, 2026a.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dParallel: Learnable parallel decoding for dLLMs. In *International Conference on Learning Representations*, 2026b.

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

Justin Deschenaux and Caglar Gulcehre. Beyond autoregression: Fast LLMs via self-distillation through time. In *International Conference on Learning Representations*, 2025.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *Advances in Neural Information Processing Systems*, 37:53945–53968, 2024.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky TQ Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *The Thirteenth International Conference on Learning Representations*, 2025.

Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. Expo: Stable reinforcement learning with expressive policies. *arXiv preprint arXiv:2507.07986*, 2025.

Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In *International conference on machine learning*, pp. 8489–8510. PMLR, 2023.

Stefano Ermon. Introducing mercury 2. `https://www.inceptionlabs.ai/blog/introducing-mercury-2`, 2026. Inception blog, accessed May 7, 2026.

Linjiajie Fang, Ruoxue Liu, Jing Zhang, Wenjia Wang, and Bingyi Jing. Diffusion actor-critic: Formulating constrained policy iteration as diffusion noise regression for offline reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

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. In *International Conference on Learning Representations*, volume 2026, pp. 15723–15750, 2026.

Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

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.

Félix G Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. Robust motion in-betweening. *ACM Transactions on Graphics (TOG)*, 39(4):60:1–60:12, 2020.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022a.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022b.

Haozhe Jiang, Nika Haghtalab, and Lijie Chen. Diffusion language models are provably optimal parallel samplers. *arXiv preprint arXiv:2512.25014*, 2025.

Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. *Advances in Neural Information Processing Systems*, 37:52996–53021, 2024.

Diederik Kingma and Ruiqi Gao. Understanding diffusion objectives as the elbo with simple data augmentation. *Advances in Neural Information Processing Systems*, 36:65484–65516, 2023.

Qiyang Li and Sergey Levine. Q-learning with adjoint matching. *arXiv preprint arXiv:2601.14234*, 2026.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. *arXiv preprint arXiv:2507.07969*, 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy agnostic rl: Offline rl and online rl fine-tuning of any class and backbone. *arXiv preprint arXiv:2412.06685*, 2024.

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.

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.

Jingyang Ou, Jiaqi Han, Minkai Xu, Shaoxuan Xu, Jianwen Xie, Stefano Ermon, Yi Wu, and Chongxuan Li. Principled rl for diffusion llms emerges from a sequence-level perspective. In *International Conference on Learning Representations*, volume 2026, pp. 4941–4966, 2026.

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. Tinyzero, 2025.

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

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.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on robot learning*, pp. 91–100. PMLR, 2022.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 1889–1897. PMLR, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

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

Clemens Schwarke, Victor Klemm, Matthijs Van der Boon, Marko Bjelonic, and Marco Hutter. Curiosity-driven learning of joint locomotion and manipulation tasks. In *Proceedings of the 7th Conference on Robot Learning*, volume 229, pp. 2594–2610. PMLR, 2023.

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

Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *arXiv preprint arXiv:2011.10024*, 2020.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

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

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

Andrew Wagenmaker, Yunchu Zhang, Mitsuhiko Nakamoto, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *Conference on Robot Learning*, pp. 258–282. PMLR, 2025.

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

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

Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, et al. Dream-coder 7b: An open diffusion language model for code. *arXiv preprint arXiv:2509.01142*, 2025.

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.

Brent Yi, Hongsuk Choi, Himanshu Gaurav Singh, Xiaoyu Huang, Takara E Truong, Carmelo Sferrazza, Yi Ma, Rocky Duan, Pieter Abbeel, Guanya Shi, et al. Flow policy gradients for robot control. *arXiv preprint arXiv:2602.02481*, 2026.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. In *The Thirteenth International Conference on Learning Representations*, 2025.

Tunyu Zhang, Xinxi Zhang, Ligong Han, Haizhou Shi, Xiaoxiao He, Zhuowei Li, Hao Wang, Kai Xu, Akash Srivastava, Chengzhi Mao, Hao Wang, Vladimir Pavlovic, and Dimitris N. Metaxas. Few-step diffusion language models via trajectory self-distillation. *arXiv preprint arXiv:2602.12262*, 2026.

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.

A Additional Related Work

Diffusion models. Diffusion models are a powerful class of generative models, with wide applications including image generation (Ho et al., 2020; Song et al., 2020; Song & Ermon, 2019), video generation (Ho et al., 2022b;a), and robotics (Black et al., 2024; Bjorck et al., 2025). More recently, diffusion language models have emerged as a strong alternative to autoregressive models for fast inference and flexible decoding (Nie et al., 2025; Xie et al., 2025; Song et al., 2025). Diffusion models can be derived from several viewpoints; in this paper, we use the variational-inference perspective (Kingma & Gao, 2023; Sohl-Dickstein et al., 2015), which connects the intractable likelihood to tractable evidence bounds.

Policy gradients. In reinforcement learning, policy-gradient methods optimize a parameterized policy by directly differentiating its expected cumulative return (Williams, 1992; Schulman et al., 2017). They have led to strong results in high-dimensional control (Schulman et al., 2016), locomotion (Rudin et al., 2022), manipulation (Schwarke et al., 2023), and related sequential decision-making problems. Policy-gradient-style methods also play a central role in the post-training of large language models (Shao et al., 2024; Rafailov et al., 2023), making their extension to diffusion language models an important algorithmic question.

RL with diffusion policies. Classical policy-gradient methods cannot be directly applied to diffusion models because the exact likelihood is intractable. One line of work tackles this problem by treating the denoising process as a Markov Decision Process (Black et al., 2023; Ren et al., 2024). This makes training tractable, but ties the optimization procedure to a particular denoising sampler. Another line of work uses variational objectives, such as diffusion or flow-matching evidence bounds, to bypass exact likelihood computation while preserving the native diffusion sampling process (McAllister et al., 2025; Wang et al., 2025). In language domains, several methods instead approximate likelihoods by simplifying or partially restoring inter-token dependencies (Zhao et al., 2025; Yang et al., 2025; Xie et al., 2025; Tang et al., 2025). DiPOD is closest to the variational line: it identifies how evidence-bound looseness can corrupt policy-gradient updates, and introduces self-distillation as a mechanism for keeping the proxy aligned with the true policy.

Other diffusion-policy RL and control methods. Beyond online policy-gradient methods, there is a rich literature on critic-based or residual-control methods for diffusion policies. Some works leave a pretrained diffusion policy unchanged and post-process generated actions using a learned critic (Hansen-Estruch et al., 2023; Mark et al., 2024; Li et al., 2025; Dong et al., 2025). Others learn a residual policy that modifies the generated action in noise space (Wagenmaker et al., 2025; Singh et al., 2020) or action space (Yuan et al., 2025; Ankile et al., 2025). A related set of off-policy methods uses a critic to supervise the score or denoising function directly (Fang et al., 2025; Ding et al., 2024; Li & Levine, 2026). These methods address complementary settings, while DiPOD focuses on stabilizing online policy-gradient updates when the diffusion-policy likelihood must be replaced by a variational proxy.

Guidance and stochastic-control perspectives. Guidance is also related to RL with diffusion policies, since the guided sampler can be interpreted as tilting a base distribution according to reward or energy. Early guidance approaches can suffer from bias in the resulting sampling distribution (Du et al., 2023; Karras et al., 2024). Adjoint Matching (Domingo-Enrich et al., 2025) takes a stochastic optimal-control perspective and derives an efficient fine-tuning objective that converges to the desired tilted distribution under its assumptions. DiPOD instead studies the policy-gradient setting where rewards may be non-differentiable and where the central challenge is maintaining alignment between the evidence-bound proxy and the true diffusion-policy likelihood.

B Background on Reinforcement Learning and Diffusion Models

B.1 Reinforcement Learning

This section provides a concise introduction to the RL algorithms relevant to this paper.

Markov Decision Process. A Markov Decision Process (MDP) consists of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a state transition kernel P, and a reward function r. A policy is defined as a function $\pi(\cdot|s)$ that outputs a distribution on $\mathcal{A}$ given a state s from $\mathcal{S}$. In the MDP, an agent with policy π starts from a designated state $s_0 \in \mathcal{S}$ at timestep $t = 0$. At each timestep t, the policy receives the current state s_t and takes an action $a_t \sim \pi(\cdot|s_t)$. The environment transitions to state $s_{t+1} \sim P(\cdot|s_t, a_t)$, and gives the agent a reward $r_t = r(s_t, a_t)$. The episode terminates when the agent reaches a terminal state $s^* \in \mathcal{S}$. An RL algorithm aims to optimize the cumulative reward, defined as

$$\mathcal{J}(\theta) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(\cdot|s_t, a_t)} \left[\sum_{t \geq 0} \gamma^t r_t \right]$$

where $\gamma \in (0, 1)$ is a discount factor. The value function is defined as the expected cumulative reward starting from a state s. Formally:

$$V_t^\pi(s) = \mathbb{E}_\pi \left[\sum_{\tau \geq t} \gamma^{\tau - t} r_\tau | s_t = s \right]$$

where $\mathbb{E}_\pi$ means that the actions are sampled from policy π. The Q-function is defined as the expected cumulative reward starting from state s and action a. Formally:

$$Q_t^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{\tau \geq t} \gamma^{\tau - t} r_\tau | s_t = s, a_t = a \right]$$

Policy Gradient Algorithms. The policy gradient theorem states that

$$\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_{a_t \sim \pi_\theta(\cdot|s_t), s_t \sim d^{\pi_\theta}} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) Q_t^{\pi_\theta}(s_t, a_t) \right]$$

where d^π is the state distribution under policy π. GAE (Schulman et al., 2016; Greensmith et al., 2004) introduces an alternative form of the policy gradient that admits lower estimation variance:

$$\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_\theta \left[\nabla_\theta \log \pi_\theta(a_t|s_t) A_t^{\pi_\theta}(s_t, a_t) \right]$$

where the advantage function is defined as $A_t^\pi(s, a) = Q_t^\pi(s, a) - V_t^\pi(s)$ and we abbreviate the expectation subscript for simplicity.

In the off-policy setting, where a behavior policy π_{ref} is used to collect data, the algorithm aims to optimize a slightly different objective defined as

$$\mathcal{J}'(\theta) = \mathbb{E}_{s_t \sim d^{\pi_{\text{ref}}}} \left[V_t^{\pi_\theta}(s_t) \right].$$

In this setting, the following policy gradient update:

$$\mathbb{E}_{\text{ref}} \left[\frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} A_t^{\pi_\theta}(s_t, a_t) \right]$$

guarantees policy improvement on $\mathcal{J}'(\theta)$. In practice, $A_t^{\pi_\theta}(s_t, a_t)$ is often estimated through a combination of rewards from the samples and a learned value function.

Proximal Policy Optimization (PPO). PPO is an instantiation of the off-policy gradient update introduced above. It performs gradient updates through

$$\nabla_\theta \mathbb{E}_{\text{ref}} \left[\min \left\{ r(\theta) \hat{A}_t, \text{clip}(r(\theta), 1 \pm \varepsilon) \hat{A}_t \right\} \right]$$

where we abbreviate

$$r(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}$$

and $\hat{A}_t$ is an estimation of $A_t^{\pi_\theta}(s_t, a_t)$. ε is a hyperparameter between 0 and 1 that controls the strength of regularization to the policy update. The training data is sampled from π_{ref}, and π_{ref} is synced with π_θ every few gradient steps. FPO (McAllister et al., 2025) builds on the PPO framework by substituting the log likelihoods of π_θ and π_{ref} as their corresponding ELBOs.

Group Relative Policy Optimization. GRPO is a variant of PPO for autoregressive language models. In language models, a prompt q, consisting of a sequence of tokens, serves as the input. The language model then generates another sequence of tokens o as the output. This process can be framed as an MDP. At each timestep t, the state is the token sequence $q|o_{\leq t}$ that has been generated so far. Here $o_{\leq t}$ means the first t tokens of o. The action o_{t+1} is the next token to be output. In many scenarios the reward is given only when o is fully generated, and the reward is often based only on the last few tokens, while the intermediate tokens can be thought of as reasoning traces. In GRPO, for each prompt q, we sample g outputs $o^1, \ldots, o^g$ using π_{ref}, and receive rewards $r^1, \ldots, r^g$. We estimate the advantage as

$$\hat{A}_t^i = \frac{r^i - \text{mean}(r^{1:g})}{\text{std}(r^{1:g})}.$$

and estimate the policy-gradient objective for q as

$$\frac{1}{g}\sum_{i=1}^{g}\frac{1}{|o^i|}\sum_{t=1}^{|o^i|}\min\left\{r(\theta)_t^i\hat{A}_t^i, \text{clip}(r(\theta)_t^i, 1\pm\varepsilon)\hat{A}_t^i\right\}$$

where

$$r(\theta)_t^i = \frac{\pi_\theta(o_t^i|q, o_{<t}^i)}{\pi_{\text{ref}}(o_t^i|q, o_{<t}^i)}.$$

Compared to PPO, GRPO uses relative advantages inside a group of rollouts instead of the true advantage, eliminating the need to train a value function or calculate different value functions for different timesteps. Dream-Coder (Xie et al., 2025) uses exactly the same algorithm for diffusion language models, ignoring that likelihoods cannot be decomposed in the same way as autoregressive models. *Diffu*-GRPO (Zhao et al., 2025), wd1 (Tang et al., 2025), and UniGRPO (Yang et al., 2025) account for this factor, but still partially ignore inter-token dependencies for tractability.

FPO/FPO++. FPO++ (Yi et al., 2026) is an improved flow-policy-gradient algorithm that replaces the single action-level FPO ratio with per-sample ratios and uses an asymmetric trust region. In our notation, if $\widehat{\text{ELBO}}_\theta^{(i)}(a_t|o_t)$ denotes the negative conditional-flow-matching loss for the i-th Monte Carlo sample, its per-sample ratio is

$$\hat{\rho}_{\text{FPO++}}^{(i)}(\theta) = \exp\left(\widehat{\text{ELBO}}_\theta^{(i)}(a_t|o_t) - \widehat{\text{ELBO}}_{\theta_{\text{old}}}^{(i)}(a_t|o_t)\right).$$

FPO++ then applies PPO clipping for $\hat{A}_t \geq 0$ and an SPO-style quadratic trust region for $\hat{A}_t < 0$. This gives an asymmetric interpretation. For positive-advantage samples, FPO++ keeps the original FPO intuition: it treats an ELBO lift as if it were a log-likelihood lift, so decreasing the conditional-flow-matching loss is taken to increase the likelihood of desirable actions. This is exact in the tight-bound regime, and otherwise is the same local approximation underlying FPO.

For negative-advantage samples, the SPO branch has a more explicit adaptive-regularization form. Recall that

$$\psi_{\text{SPO}}(\rho, \hat{A}_t) = \rho\hat{A}_t - \frac{|\hat{A}_t|}{2\varepsilon^{\text{clip}}}(\rho-1)^2, \qquad \nabla_\theta\hat{\rho}_{\text{FPO++}}^{(i)} = \hat{\rho}_{\text{FPO++}}^{(i)}\nabla_\theta\widehat{\text{ELBO}}_\theta^{(i)}(a_t|o_t).$$

Therefore, for $\hat{A}_t < 0$,

$$\nabla_\theta\psi_{\text{SPO}}\left(\hat{\rho}_{\text{FPO++}}^{(i)}, \hat{A}_t\right) = \left[\hat{\rho}_{\text{FPO++}}^{(i)}\hat{A}_t - \frac{|\hat{A}_t|}{\varepsilon^{\text{clip}}}\hat{\rho}_{\text{FPO++}}^{(i)}\left(\hat{\rho}_{\text{FPO++}}^{(i)} - 1\right)\right]\nabla_\theta\widehat{\text{ELBO}}_\theta^{(i)}(a_t|o_t),$$

which is the usual ratio-weighted policy-gradient term plus an adaptive DiPOD-like ELBO term $\beta_-^{(i)}\nabla_\theta\widehat{\text{ELBO}}_\theta^{(i)}(a_t|o_t)$ with signed coefficient

$$\beta_-^{(i)}(\theta, \hat{A}_t) = -\frac{|\hat{A}_t|}{\varepsilon^{\text{clip}}}\hat{\rho}_{\text{FPO++}}^{(i)}\left(\hat{\rho}_{\text{FPO++}}^{(i)} - 1\right),$$

where the sign is for gradient ascent on ψ_{SPO}; equivalently, the sign flips if one writes the implementation as minimizing $-\psi_{\mathrm{SPO}}$. Up to this sign convention and the multiplicative ratio factor, this is exactly a coefficient proportional to $|\hat{A}_t|(\hat{\rho}^{(i)}_{\mathrm{FPO++}} - 1)/\varepsilon^{\mathrm{clip}}$. The term vanishes on-policy and grows with the ratio deviation, pulling negative-advantage ratios back toward 1. This estimator is adequate in the sense of Definition 2.1: when the bound is tight and the reference policy is synchronized with the current policy, $\hat{\rho}^{(i)}_{\mathrm{FPO++}} = 1$, the adaptive term vanishes, and $\mathbb{E}_i[\nabla_\theta \widehat{\mathrm{ELBO}}^{(i)}_\theta(a_t|o_t)] = \nabla_\theta \mathrm{ELBO}_\theta(a_t|o_t) = \nabla_\theta \log \pi_\theta(a_t|o_t)$. Averaging over Monte Carlo samples therefore recovers the policy-gradient integrand $\hat{A}_t \nabla_\theta \log \pi_\theta(a_t|o_t)$.

B.2 Diffusion Models

This section provides a brief introduction to diffusion models from a variational inference perspective. We refer interested readers to Wang et al. (2025) for more details regarding diffusion language models.

Variational Inference. A central goal in generative modeling is to learn a parameterized distribution p_θ that assigns high probability to observed data. Given a dataset drawn from an unknown data distribution p_{data}, the standard training objective is maximum likelihood estimation (MLE),

$$\max_\theta \mathbb{E}_{x_0 \sim p_{\text{data}}}[\log p_\theta(x_0)].$$

This objective applies broadly, independent of the specific model family. In many modern generative models, including diffusion models, it is natural to introduce latent variables to describe a multi-step sampling procedure: one may first sample a latent variable z, then sample the data x_0 conditional on z. This leads to a latent-variable model of the form

$$p_\theta(x_0) = \int p_\theta(x_0, z) \mathrm{d}z,$$

where z can be high-dimensional (and in diffusion, typically corresponds to an entire trajectory of intermediate variables). While MLE still aims to maximize $\log p_\theta(x_0)$, the marginalization over z often makes $\log p_\theta(x_0)$ intractable to evaluate and differentiate.

Variational inference addresses this intractability by introducing an auxiliary distribution $q(z|x_0)$, often called a variational posterior, that approximates the true posterior $p_\theta(z|x_0)$. For any choice of $q(z|x_0)$, Jensen's inequality gives

$$\begin{aligned}
&\log p_\theta(x_0) \\
=&\log \int q(z|x_0) \frac{p_\theta(x_0, z)}{q(z|x_0)} \mathrm{d}z && (8)\\
\geq&\mathbb{E}_{q(z|x_0)} \left[\log p_\theta(x_0, z) - \log q(z|x_0)\right] && (9)\\
=:&\mathrm{ELBO}_\theta(x_0). && (10)
\end{aligned}$$

The quantity $\mathrm{ELBO}_\theta(x_0)$ is the evidence lower bound. Maximizing the ELBO provides a tractable surrogate for maximizing $\log p_\theta(x_0)$, because it replaces the log of an integral with an expectation under $q(z|x_0)$.

Equivalently, using $p_\theta(z|x_0) = p_\theta(x_0, z)/p_\theta(x_0)$,

$$\log p_\theta(x_0) = \mathrm{ELBO}_\theta(x_0) + \mathrm{KL}\left(q(z|x_0) \| p_\theta(z|x_0)\right), \tag{11}$$

so the bound is tight if and only if $q(z|x_0) = p_\theta(z|x_0)$. Here the KL divergence is defined as $\mathrm{KL}(q\|p) := \mathbb{E}_q\left[\log \frac{q}{p}\right]$, corresponding to the $\mathcal{D}^{\mathrm{L}}_\theta$ in Section 2 which is nonnegative and equals 0 iff $q = p$. In diffusion models, a carefully chosen q makes the ELBO decompose into simple local terms, yielding a practical learning objective while still targeting maximum likelihood.

Diffusion as structured variational inference. Diffusion models instantiate the generic latent variable as an entire noising trajectory,

$$z \equiv x_{1:T} := (x_1, \ldots, x_T), \tag{12}$$

and define a *fixed* forward (variational) process

$$q(z|x_0) = q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}), \tag{13}$$

where each $q(x_t|x_{t-1})$ is a simple corruption kernel. The generative model uses a simple prior $p(x_T)$ and learned reverse transitions:

$$\begin{aligned} p_\theta(x_0, z) =& p_\theta(x_{0:T}) \\ =& p_\theta(x_0|x_1) \prod_{t=2}^{T} p_\theta(x_{t-1}|x_t) p(x_T). \end{aligned} \tag{14}$$

Substituting Equation (13)–Equation (14) into Equation (10) yields a sum of tractable terms:

$$\begin{aligned} &\mathrm{ELBO}_\theta(x_0) \\ =& \mathbb{E}_{q(x_1|x_0)}[\log p_\theta(x_0|x_1)] - \mathrm{KL}(q(x_T|x_0)\|p(x_T)) \\ &- \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)}[\mathrm{KL}(q(x_{t-1}|x_t, x_0)\|p_\theta(x_{t-1}|x_t))], \end{aligned} \tag{15}$$

where $q(x_{t-1}|x_t, x_0)$ is the exact posterior of the fixed forward process (obtained from Bayes' rule). This expression highlights that maximizing $\mathrm{ELBO}_\theta(x_0)$ amounts to fitting each reverse kernel $p_\theta(x_{t-1}|x_t)$ to the corresponding diffusion posterior under q.

EUBO (evidence upper bound). Using the same latent-variable setup, define the (nonnegative) importance weight

$$w_\theta(z; x_0) := \frac{p_\theta(x_0, z)}{q(z|x_0)}. \tag{16}$$

Then the evidence satisfies $p_\theta(x_0) = \mathbb{E}_{q(z|x_0)}[w_\theta(z; x_0)]$. For any $\beta \geq 1$, Jensen (applied to the convex map $u \mapsto u^\beta$) gives

$$\begin{aligned} &\log p_\theta(x_0) = \log \mathbb{E}_q[w_\theta] \\ &\leq \frac{1}{\beta} \log \mathbb{E}_{q(z|x_0)} \left[w_\theta(z; x_0)^\beta\right] =: \mathrm{EUBO}_{\theta,\beta}(x_0), \end{aligned} \tag{17}$$

so $\mathrm{EUBO}_{\theta,\beta}(x_0)$ is an upper bound on the log-evidence (tight when $q(z|x_0) = p_\theta(z|x_0)$).

The looseness of this upper bound is naturally characterized by a Renyi divergence. In particular,

$$\begin{aligned} &\mathrm{EUBO}_{\theta,\beta}(x_0) - \log p_\theta(x_0) \\ =& \frac{1}{\beta} \log \mathbb{E}_{q(z|x_0)} \left[\left(\frac{p_\theta(z|x_0)}{q(z|x_0)}\right)^\beta\right] \\ =& \frac{\beta - 1}{\beta} D_\beta \left(p_\theta(z|x_0)\|q(z|x_0)\right). \end{aligned}$$

Here $D_\beta(p\|q) := \frac{1}{\beta - 1} \log \mathbb{E}_q \left[\left(\frac{p}{q}\right)^\beta\right]$ is the Renyi divergence of order β, corresponding to the $\mathcal{D}_\theta^{\mathrm{U}}$ in Section 2. This quantity is nonnegative and equals 0 iff $p_\theta(z|x_0) = q(z|x_0)$, so the EUBO is tight exactly when the variational posterior matches the true posterior.

In diffusion, the latent variable is the noising trajectory $z \equiv x_{1:T}$ with fixed forward process $q(x_{1:T}|x_0)$, and $p_\theta(x_0, z) = p_\theta(x_{0:T})$ is defined by the learned reverse transitions. Thus,

$$\mathrm{EUBO}_{\theta,\beta}(x_0) = \frac{1}{\beta} \log \mathbb{E}_{q(x_{1:T}|x_0)} \left[\left(\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right)^\beta\right], \tag{18}$$

which is typically much harder to optimize than the ELBO because it involves a log-moment over entire forward paths and does not decompose into a simple sum of per-timestep KL terms.

Directly maximizing the likelihood $\log p_\theta(x_0) = \log \int p_\theta(x_0, z)\mathrm{d}z$ is generally intractable because it requires marginalizing over high-dimensional latents z (in diffusion, entire trajectories). Likewise, $\mathrm{EUBO}_{\theta,\beta}(x_0)$ is typically intractable since it involves a log-moment $\log \mathbb{E}_q[w_\theta^\beta]$ over the same latent space, which does not admit a simple additive decomposition. In contrast, the diffusion ELBO is tractable because it moves the logarithm inside an expectation under the fixed forward process q, yielding terms that can be estimated by sampling $z \sim q(\cdot|x_0)$ (often reducing to per-timestep losses).

SPG (Wang et al., 2025) obtains a practical EUBO surrogate for masked dLLMs by exploiting the special absorbing-mask forward process. Let $x_{1:n}$ be a token sequence, where x_i is the token at position i, and let m denote the mask token. In this paragraph, t indexes the diffusion noising step, while i indexes the sequence position. We write $z_t = (z_{t,1}, \ldots, z_{t,n})$ for the corrupted sequence at diffusion time t, so $z_{t,i}$ is the i-th token of the corrupted sequence at that time. The absorbing-mask forward process uses a monotone schedule α_t with, in the discrete-time convention, $\alpha_1 = 1$ and $\alpha_T = 0$. At time t, each coordinate remains clean with probability α_t and is masked with probability $1 - \alpha_t$:

$$q(z_{t,i} = x_i|x_i) = \alpha_t, \qquad q(z_{t,i} = m|x_i) = 1 - \alpha_t.$$

Thus each coordinate is either unchanged or masked, and once a coordinate becomes m it stays masked under the forward process. For this process, the only nontrivial reverse-model term for coordinate i occurs when $z_{t+1,i} = m$ and the model predicts the clean token x_i from the corrupted sequence z_{t+1}. The factor $(\alpha_t - \alpha_{t+1})/(1 - \alpha_{t+1})$ below is the posterior probability that coordinate i was still clean at time t given that it is masked at time $t + 1$.

If we directly instantiate the Renyi EUBO in Equation (17) for this masked diffusion model, we obtain a path-level log-moment:

$$\mathrm{EUBO}^{\mathrm{path}}_{\theta,\beta}(x_{1:n}) = \frac{1}{\beta} \log \mathbb{E}_{z_{1:T} \sim q(\cdot|x_{1:n})} \left[\prod_{t=1}^{T-1} \prod_{i=1}^{n} \left(\frac{p_\theta(z_{t,i}|z_{t+1})}{q(z_{t,i}|z_{t+1}, x_{1:n})} \right)^\beta \right]. \tag{19}$$

This is the literal, unapproximated EUBO for the masked diffusion model, but it is not in the same form as the usual masked-diffusion ELBO training loss: it contains a product over all diffusion times and token positions inside a single outer logarithm. By contrast, the masked-diffusion ELBO moves the logarithm inside the expectation and reduces to weighted masked-token prediction terms,

$$\mathcal{L}_{\mathrm{ELBO}}(x_{1:n}; \theta) = \sum_{i=1}^{n} \mathbb{E}_{t, z_t} \left[w(t) \mathbb{1}\{z_{t,i} = m\} \log \pi_\theta(x_i|z_t) \right],$$

up to schedule-dependent constants. SPG's approximation sits between these two objects: it keeps an upper-bound structure derived from the path-level EUBO, but uses the absorbing-mask structure to obtain a per-coordinate log-moment that can be estimated with masked-token prediction probabilities. Applying the Renyi variational bound and decomposing over coordinates yields the discrete-time upper-bound surrogate

$$\mathcal{L}^{\mathrm{SPG}}_{\mathrm{EUBO}}(x_{1:n}; \theta) = \frac{1}{\beta} \sum_{i=1}^{n} \log \left(\sum_{t=1}^{T-1} \mathbb{E}_{z_{t+1} \sim q(\cdot|x_{1:n})} \left[\frac{\alpha_t - \alpha_{t+1}}{1 - \alpha_{t+1}} \mathbb{1}\{z_{t+1,i} = m\} \right. \right.$$
$$\left. \left. \pi_\theta(x_i|z_{t+1})^\beta \right] \right) + C(T), \tag{20}$$

where $C(T)$ is independent of θ and therefore does not affect policy-gradient updates. In the continuous-time implementation, this is written in the equivalent form

$$\widetilde{\mathcal{L}}^{\mathrm{SPG}}_{\mathrm{EUBO}}(x_{1:n}; \theta) = \frac{1}{\beta} \sum_{i=1}^{n} \log \mathbb{E}_{t, z_t} \left[w(t) \mathbb{1}\{z_{t,i} = m\} \pi_\theta(x_i|z_t)^\beta \right], \tag{21}$$

with $w(t)$ collecting the continuous-time analogue of the schedule-dependent masking weight above, including any density correction for how t is sampled. The logarithm remains outside the expectation, so single-sample Monte Carlo estimates of this quantity are generally biased; nevertheless, its gradient can be estimated by differentiating the sampled log-moment. This construction is specific to the categorical absorbing-mask structure of dLLMs, which is why the SPG EUBO approximation does not directly provide a general-purpose EUBO estimator for arbitrary diffusion or flow policies.

Block diffusion. Block diffusion generates a sequence one block at a time, using diffusion within each block while keeping earlier blocks fixed (Arriola et al., 2025). For a prompt x and a fixed completion length L, split the completion y into B consecutive blocks $y_1, \dots, y_B$ and write $c_b = (x, y_{<b}, L)$ for the context of block b. The model factorizes as

$$p_\theta^{\text{block}}(y \mid x, L) = \prod_{b=1}^{B} p_\theta(y_b \mid c_b), \tag{22}$$

where each conditional distribution is defined by a diffusion process within that block. Let z_b denote its diffusion trajectory and $q_b(z_b \mid y_b)$ its forward noising process. Applying the variational bound in Equation (10) to each block gives

$$\mathcal{L}_b(\theta) := \mathbb{E}_{q_b(z_b|y_b)}\left[\log \frac{p_\theta(y_b, z_b \mid c_b)}{q_b(z_b \mid y_b)}\right] \le \log p_\theta(y_b \mid c_b). \tag{23}$$

Summing these bounds yields an ELBO for the whole completion:

$$\mathcal{L}_{\text{block}}(\theta) := \sum_{b=1}^{B} \mathcal{L}_b(\theta) \le \sum_{b=1}^{B} \log p_\theta(y_b \mid c_b) = \log p_\theta^{\text{block}}(y \mid x, L). \tag{24}$$

Thus both the sequence ELBO and its gap from log-likelihood add across blocks. The bound holds for any model weights, whether or not they were trained with this blockwise objective.

C Theoretical Analysis of DiPOD

In this section, we formally analyze the convergence of DiPOD. In particular, we formalize and prove Theorem 3.1.

C.1 Setup and Assumptions

Throughout this section, we work in the fully observable MDP setting described in Section B, so observations are states. To keep the notation aligned with the main body, we write o for this observation/state variable rather than switching to s. Let d_{π_θ} be the on-policy observation distribution induced by π_θ in the same sense used by the policy-gradient theorem (for example, the discounted occupancy measure in the episodic case or the stationary distribution in the continuing case), and define

$$\rho_\theta(o, a) := d_{\pi_\theta}(o)\pi_\theta(a|o).$$

For any reference distribution μ over observation-action pairs, define the self-distillation objective

$$F_\mu(\theta) := \mathbb{E}_\mu\left[\text{ELBO}_\theta(a|o)\right]$$

and the expected ELBO and EUBO discrepancies

$$D_\mu^{\text{L}}(\theta) := \mathbb{E}_{(o,a)\sim\mu}\left[\mathcal{D}_\theta^{\text{L}}(o, a)\right] = \mathbb{E}_\mu\left[\log \pi_\theta(a|o) - \text{ELBO}_\theta(a|o)\right],$$
$$D_\mu^{\text{U}}(\theta) := \mathbb{E}_{(o,a)\sim\mu}\left[\mathcal{D}_\theta^{\text{U}}(o, a)\right] = \mathbb{E}_\mu\left[\text{EUBO}_\theta(a|o) - \log \pi_\theta(a|o)\right].$$

For a gradient estimator $g_\theta(o, a)$, write its on-policy expected update as

$$G_g(\theta) := \mathbb{E}_{(o,a)\sim\rho_\theta}\left[g_\theta(o, a)\right].$$

We are going to build our analysis on the following standard assumptions.

Assumption C.1 (Smooth Policy-Gradient Dynamics)**.** The policy-gradient theorem holds for $\mathcal{J}$ on the region of parameter space visited by the algorithm, and $\mathcal{J}$ is $L_{\mathcal{J}}$-smooth:

$$\mathcal{J}(\theta') \ge \mathcal{J}(\theta) + \langle \nabla\mathcal{J}(\theta), \theta' - \theta\rangle - \frac{L_{\mathcal{J}}}{2}\|\theta' - \theta\|^2.$$

Assumption C.2 (Lipschitzness of Gradient Estimator)**.** The estimator g_θ is adequate in the sense of Definition 2.1. Moreover, there exists an adequacy coefficient $c_g < \infty$ such that whenever the current on-policy ELBO discrepancy is at most ε,

$$D_{\rho_\theta}^{\text{L}}(\theta) \le \varepsilon \quad \Longrightarrow \quad \|G_g(\theta) - \nabla\mathcal{J}(\theta)\| \le c_g\sqrt{\varepsilon}.$$

This is the quantitative form of adequacy used in the theorem: near a tight on-policy ELBO, the expected adequate-gradient update is Lipschitz-close to the true policy gradient.

Assumption C.3 (Realizability)**.** For every reference policy π_{ref} considered by the algorithm, with reference distribution $\mu(o,a) = d_{\mathrm{ref}}(o)\pi_{\mathrm{ref}}(a|o)$, there exists a parameter $\theta^\dagger$ such that $\pi_{\theta^\dagger}(\cdot|o) = \pi_{\mathrm{ref}}(\cdot|o)$ for d_{ref}-almost every o and $\mathcal{D}^{\mathrm{L}}_{\theta^\dagger}(o,a) = 0$ for μ-almost every (o,a). Equivalently, the policy class can represent the reference policy with a tight ELBO on the reference distribution.

Assumption C.4 (On-policy Self-Distillation Oracle)**.** At each self-distillation step, DiPOD chooses the current on-policy reference distribution $\mu = \rho_{\theta_{\mathrm{ref}}}$. The self-distillation oracle returns a parameter $\bar{\theta}$, and hence a policy $\pi_{\bar{\theta}}$, whose on-reference ELBO value is within ε of the best achievable value:

$$F_\mu(\bar{\theta}) \geq \sup_\theta F_\mu(\theta) - \varepsilon.$$

Equivalently, the returned policy/model is ε-suboptimal for the on-reference ELBO maximization problem. This suboptimality may come from either policy mismatch with the reference distribution or from a nonzero ELBO discrepancy.

Remark C.5 (Reference versus returned-policy distributions). The oracle assumption controls ELBO optimality under the reference distribution used for self-distillation. An approximate oracle can still return a policy whose own occupancy distribution differs from the reference occupancy distribution. For this reason, the performance theorem below keeps track of both the reference-distribution discrepancy guaranteed by self-distillation and the returned policy's own on-policy discrepancy, which is the quantity needed to control the next policy-gradient update.

Before proceeding to the performance guarantee, we first show that both FPO and SPG satisfy Assumption C.2 under standard bounded-advantage and smoothness conditions. The same self-bounding argument can also be extended to other adequate-gradient estimators whose bias is controlled by smooth nonnegative variational discrepancies.

Proposition C.6 (FPO and SPG instantiate Lipschitzness)**.** *Assume* $|A^{\pi_\theta}(o,a)| \leq A_{\max}$.

1. ***FPO.*** *Let*

$$g^{\mathrm{FPO}}_\theta(o,a) = A^{\pi_\theta}(o,a)\nabla_\theta \mathit{ELBO}_\theta(a|o).$$

If $\mathcal{D}^{\mathrm{L}}_\theta(o,a)$ *is nonnegative and* L_{L}*-smooth in* θ *for each* (o,a)*, then*

$$\left\|G_{g^{\mathrm{FPO}}}(\theta) - \nabla\mathcal{J}(\theta)\right\| \leq A_{\max}\sqrt{2L_{\mathrm{L}}D^{\mathrm{L}}_{\rho_\theta}(\theta)}.$$

Thus FPO satisfies Assumption C.2 with $c_g = A_{\max}\sqrt{2L_{\mathrm{L}}}$.

2. ***SPG.*** *Let*

$$g^{\mathrm{SPG}}_\theta(o,a) = \mathbb{1}_{A>0}A\nabla_\theta \mathit{ELBO}_\theta(a|o) + \mathbb{1}_{A<0}A\nabla_\theta \mathit{EUBO}_\theta(a|o),$$

where A *abbreviates* $A^{\pi_\theta}(o,a)$*. If* $\mathcal{D}^{\mathrm{L}}_\theta$ *and* $\mathcal{D}^{\mathrm{U}}_\theta$ *are nonnegative and respectively* L_{L}*- and* L_{U}*-smooth, then*

$$\left\|G_{g^{\mathrm{SPG}}}(\theta) - \nabla\mathcal{J}(\theta)\right\| \leq A_{\max}\sqrt{2L_{\mathrm{L}}D^{\mathrm{L}}_{\rho_\theta}(\theta)} + A_{\max}\sqrt{2L_{\mathrm{U}}D^{\mathrm{U}}_{\rho_\theta}(\theta)}.$$

Consequently, if the EUBO discrepancy is also tight in the self-distilled region, e.g. $D^{\mathrm{U}}_{\rho_\theta}(\theta) \leq C_{\mathrm{U}}D^{\mathrm{L}}_{\rho_\theta}(\theta)$*, then SPG satisfies Assumption C.2 with* $c_g = A_{\max}(\sqrt{2L_{\mathrm{L}}} + \sqrt{2L_{\mathrm{U}}C_{\mathrm{U}}})$.

Proof. We use the self-bounding property of nonnegative smooth functions. If $f(\theta) \geq 0$ and f is L-smooth, then

$$\|\nabla f(\theta)\|^2 \leq 2Lf(\theta). \tag{25}$$

Indeed, smoothness at $\theta' = \theta - \frac{1}{L}\nabla f(\theta)$ gives

$$0 \leq f(\theta') \leq f(\theta) - \frac{1}{2L}\|\nabla f(\theta)\|^2.$$

For FPO, the policy-gradient theorem gives

$$\nabla\mathcal{J}(\theta) = \mathbb{E}_{\rho_\theta}\left[A^{\pi_\theta}(o,a)\nabla_\theta \log \pi_\theta(a|o)\right].$$

Using $\mathrm{ELBO}_\theta = \log \pi_\theta - \mathcal{D}_\theta^{\mathrm{L}}$,

$$G_{g^{\mathrm{FPO}}}(\theta) - \nabla \mathcal{J}(\theta) = -\mathbb{E}_{\rho_\theta}\left[A^{\pi_\theta}(o,a)\nabla_\theta \mathcal{D}_\theta^{\mathrm{L}}(o,a)\right].$$

Therefore,

$$\begin{aligned}\left\|G_{g^{\mathrm{FPO}}}(\theta) - \nabla \mathcal{J}(\theta)\right\| &\le A_{\max}\mathbb{E}_{\rho_\theta}\left[\left\|\nabla_\theta \mathcal{D}_\theta^{\mathrm{L}}(o,a)\right\|\right] \\ &\le A_{\max}\sqrt{\mathbb{E}_{\rho_\theta}\left[\left\|\nabla_\theta \mathcal{D}_\theta^{\mathrm{L}}(o,a)\right\|^2\right]} \\ &\le A_{\max}\sqrt{2L_{\mathrm{L}} D_{\rho_\theta}^{\mathrm{L}}(\theta)},\end{aligned}$$

where the last step uses Equation (25).

For SPG, use $\mathrm{EUBO}_\theta = \log \pi_\theta + \mathcal{D}_\theta^{\mathrm{U}}$ in addition to the ELBO decomposition:

$$G_{g^{\mathrm{SPG}}}(\theta) - \nabla \mathcal{J}(\theta) = -\mathbb{E}_{\rho_\theta}\left[\mathbb{1}_{A>0} A \nabla_\theta \mathcal{D}_\theta^{\mathrm{L}}(o,a)\right] + \mathbb{E}_{\rho_\theta}\left[\mathbb{1}_{A<0} A \nabla_\theta \mathcal{D}_\theta^{\mathrm{U}}(o,a)\right].$$

The triangle inequality, bounded advantages, Cauchy-Schwarz, and Equation (25) give

$$\left\|G_{g^{\mathrm{SPG}}}(\theta) - \nabla \mathcal{J}(\theta)\right\| \le A_{\max}\sqrt{2L_{\mathrm{L}} D_{\rho_\theta}^{\mathrm{L}}(\theta)} + A_{\max}\sqrt{2L_{\mathrm{U}} D_{\rho_\theta}^{\mathrm{U}}(\theta)}.$$

The final statement follows by substituting $D_{\rho_\theta}^{\mathrm{U}}(\theta) \le C_{\mathrm{U}} D_{\rho_\theta}^{\mathrm{L}}(\theta)$. □

C.2 Performance Guarantee

Lemma C.7 (Guarantee for Self-Distillation Steps)**.** *Let $\mu(o,a) = d_{\mathrm{ref}}(o)\pi_{\mathrm{ref}}(a|o)$ be a reference state-action distribution satisfying Assumption C.3. If an ELBO oracle returns $\bar{\theta}$ satisfying*

$$F_\mu(\bar{\theta}) \ge \sup_\theta F_\mu(\theta) - \varepsilon,$$

then

$$\mathbb{E}_{o\sim d_{\mathrm{ref}}}\left[\mathrm{KL}(\pi_{\mathrm{ref}}(\cdot|o) \,\|\, \pi_{\bar{\theta}}(\cdot|o))\right] + D_\mu^{\mathrm{L}}(\bar{\theta}) \le \varepsilon. \tag{26}$$

Proof. We proceed in three steps.

Step 1: upper bound the best possible self-distillation objective. For any candidate parameter θ, the ELBO decomposition in Equation (2) gives

$$F_\mu(\theta) = \mathbb{E}_\mu\left[\log \pi_\theta(a|o)\right] - D_\mu^{\mathrm{L}}(\theta).$$

Since $D_\mu^{\mathrm{L}}(\theta) \ge 0$,

$$F_\mu(\theta) \le \mathbb{E}_{o\sim d_{\mathrm{ref}},\, a\sim\pi_{\mathrm{ref}}(\cdot|o)}\left[\log \pi_\theta(a|o)\right].$$

Rewriting the right-hand side relative to π_{ref},

$$\begin{aligned}&\mathbb{E}_{o\sim d_{\mathrm{ref}},\, a\sim\pi_{\mathrm{ref}}(\cdot|o)}\left[\log \pi_\theta(a|o)\right] \\ &= \mathbb{E}_\mu\left[\log \pi_{\mathrm{ref}}(a|o)\right] - \mathbb{E}_{o\sim d_{\mathrm{ref}}}\left[\mathrm{KL}(\pi_{\mathrm{ref}}(\cdot|o) \,\|\, \pi_\theta(\cdot|o))\right] \\ &\le \mathbb{E}_\mu\left[\log \pi_{\mathrm{ref}}(a|o)\right].\end{aligned}$$

Thus,

$$F_\mu(\theta) \le \mathbb{E}_\mu\left[\log \pi_{\mathrm{ref}}(a|o)\right]. \tag{27}$$

Step 2: show that the upper bound is attainable. By realizability, $F_\mu(\theta^\dagger) = \mathbb{E}_\mu[\log \pi_{\mathrm{ref}}(a|o)]$. Together with Equation (27), this implies

$$\sup_\theta F_\mu(\theta) = \mathbb{E}_\mu\left[\log \pi_{\mathrm{ref}}(a|o)\right].$$

Step 3: use oracle optimality. The ε-optimality of $\bar{\theta}$ gives

$$F_\mu(\bar{\theta}) \geq \mathbb{E}_\mu \left[\log \pi_{\mathrm{ref}}(a|o)\right] - \varepsilon.$$

Expanding $F_\mu(\bar{\theta})$ as in Step 1 gives

$$F_\mu(\bar{\theta}) = \mathbb{E}_\mu \left[\log \pi_{\mathrm{ref}}(a|o)\right] - \mathbb{E}_{o\sim d_{\mathrm{ref}}} \left[\mathrm{KL}(\pi_{\mathrm{ref}}(\cdot|o) \,\|\, \pi_{\bar{\theta}}(\cdot|o))\right] - D^{\mathrm{L}}_\mu(\bar{\theta}).$$

Combining the previous two displays yields Equation (26). □

Theorem C.8 (DiPOD improves return under on-policy tightness)**.** *Suppose Assumptions C.1 to C.4 hold. Consider one idealized DiPOD cycle with current parameter θ_k. The self-distillation oracle uses ρ_{θ_k} as reference and returns $\bar{\theta}_k$. Then Lemma C.7 gives the reference-distribution decomposition*

$$\mathbb{E}_{o\sim d_{\pi_{\theta_k}}} \left[\mathrm{KL}\big(\pi_{\theta_k}(\cdot|o) \,\|\, \pi_{\bar{\theta}_k}(\cdot|o)\big)\right] + D^{\mathrm{L}}_{\rho_{\theta_k}}(\bar{\theta}_k) \leq \varepsilon. \tag{28}$$

As emphasized in Remark C.5, this is a guarantee on the reference distribution ρ_{θ_k}. Now let $\bar{\varepsilon}_k := D^{\mathrm{L}}_{\rho_{\bar{\theta}_k}}(\bar{\theta}_k)$ be the discrepancy on the returned policy's own on-policy distribution, and perform the policy update

$$\theta_{k+1} = \bar{\theta}_k + \eta G_g(\bar{\theta}_k).$$

If $0 < \eta \leq 1/L_{\mathcal{J}}$, then

$$\mathcal{J}(\theta_{k+1}) \geq \mathcal{J}(\bar{\theta}_k) + \frac{\eta}{2}\left[\left\|\nabla\mathcal{J}(\bar{\theta}_k)\right\|^2 - c_g^2\bar{\varepsilon}_k\right]. \tag{29}$$

Therefore, whenever $\|\nabla\mathcal{J}(\bar{\theta}_k)\| > c_g\sqrt{\bar{\varepsilon}_k}$, the policy update strictly improves the true return relative to the returned policy $\bar{\theta}_k$. If the oracle is exact ($\varepsilon = 0$), or more generally returns a policy-preserving model so that $\rho_{\bar{\theta}_k} = \rho_{\theta_k}$, then $\bar{\varepsilon}_k \leq \varepsilon$ and $\mathcal{J}(\bar{\theta}_k) = \mathcal{J}(\theta_k)$; in this case the full DiPOD cycle strictly improves $\mathcal{J}$ whenever $\|\nabla\mathcal{J}(\bar{\theta}_k)\| > c_g\sqrt{\varepsilon}$.

Moreover, suppose the iterates converge to a stable post-distillation policy $\theta_\star$ for which the final on-policy self-distillation step is ε-optimal and returns $\theta_\star$. Here stable means that the subsequent idealized update

$$\theta_\star^+ = \theta_\star + \eta G_g(\theta_\star)$$

with $0 < \eta \leq 1/L_{\mathcal{J}}$ does not strictly improve $\mathcal{J}$. Then

$$\|\nabla\mathcal{J}(\theta_\star)\| \leq c_g\sqrt{\varepsilon}, \qquad D^{\mathrm{L}}_{\rho_{\theta_\star}}(\theta_\star) \leq \varepsilon.$$

Thus the final policy is approximately stationary for the true return and has a tight ELBO on its own final on-policy distribution.

Proof. The oracle decomposition Equation (28) follows directly by applying Lemma C.7 with $\mu = \rho_{\theta_k}$.

By definition of $\bar{\varepsilon}_k$ and the Lipschitzness of the estimator,

$$\left\|G_g(\bar{\theta}_k) - \nabla\mathcal{J}(\bar{\theta}_k)\right\| \leq c_g\sqrt{\bar{\varepsilon}_k}.$$

Let

$$v := \nabla\mathcal{J}(\bar{\theta}_k), \qquad e := G_g(\bar{\theta}_k) - v.$$

Then $\|e\| \leq c_g\sqrt{\bar{\varepsilon}_k}$. Using $L_{\mathcal{J}}$-smoothness with $\theta_{k+1} = \bar{\theta}_k + \eta(v + e)$,

$$\mathcal{J}(\theta_{k+1}) \geq \mathcal{J}(\bar{\theta}_k) + \eta\langle v, v + e\rangle - \frac{L_{\mathcal{J}}\eta^2}{2}\|v + e\|^2.$$

When $\eta \leq 1/L_{\mathcal{J}}$,

$$\begin{aligned}
\mathcal{J}(\theta_{k+1}) &\geq \mathcal{J}(\bar{\theta}_k) + \eta\langle v, v + e\rangle - \frac{\eta}{2}\|v + e\|^2 \\
&= \mathcal{J}(\bar{\theta}_k) + \frac{\eta}{2}\left[\|v\|^2 - \|e\|^2\right] \\
&\geq \mathcal{J}(\bar{\theta}_k) + \frac{\eta}{2}\left[\left\|\nabla\mathcal{J}(\bar{\theta}_k)\right\|^2 - c_g^2\bar{\varepsilon}_k\right].
\end{aligned}$$

This proves Equation (29). Strict improvement follows whenever $\|\nabla \mathcal{J}(\bar{\theta}_k)\| > c_g\sqrt{\bar{\varepsilon}_k}$. If the oracle is policy preserving, then $\rho_{\bar{\theta}_k} = \rho_{\theta_k}$, so Equation (28) implies $\bar{\varepsilon}_k = D^{\mathrm{L}}_{\rho_{\theta_k}}(\bar{\theta}_k) \le \varepsilon$, and policy preservation also gives $\mathcal{J}(\bar{\theta}_k) = \mathcal{J}(\theta_k)$. When $\varepsilon = 0$, the oracle decomposition gives zero expected KL from π_{θ_k} to $\pi_{\bar{\theta}_k}$ on $d_{\pi_{\theta_k}}$ and zero reference-distribution ELBO discrepancy; under the same MDP dynamics this is the exact policy-preserving case.

For the final claim, apply the oracle decomposition to the final self-distillation step with reference $\rho_{\theta_\star}$ and returned parameter $\theta_\star$. The KL term is zero, so $D^{\mathrm{L}}_{\rho_{\theta_\star}}(\theta_\star) \le \varepsilon$. If $\|\nabla \mathcal{J}(\theta_\star)\| > c_g\sqrt{\varepsilon}$, the improvement bound applied to $\theta^+_\star$ would give a strict increase in $\mathcal{J}$, contradicting stability. Thus $\|\nabla \mathcal{J}(\theta_\star)\| \le c_g\sqrt{\varepsilon}$. □

D Illustrative Two-Token Post-Training Experiment

We use the same toy experiment as the one in Appendix C.3 of SPG (Wang et al., 2025). In the toy experiment, the discrete diffusion model generates a distribution on two discrete tokens $x = (x_1, x_2)$. Both x_1 and x_2 take values from $\mathcal{V} = \{\mathrm{A}, \mathrm{B}\}$. In the generation process, x_1 and x_2 can also be a special mask token M. The generation starts from $x = \mathrm{MM}$, and tokens are decoded according to a uniformly random order. The model is parameterized by six real numbers:

$$a = \mathrm{logit}(\pi(x_1 = \mathrm{A}|x = \mathrm{MA})), b = \mathrm{logit}(\pi(x_1 = \mathrm{A}|x = \mathrm{MM})), c = \mathrm{logit}(\pi(x_2 = \mathrm{A}|x = \mathrm{AM})),$$
$$d = \mathrm{logit}(\pi(x_2 = \mathrm{A}|x = \mathrm{MM})), e = \mathrm{logit}(\pi(x_1 = \mathrm{A}|x = \mathrm{MB})), f = \mathrm{logit}(\pi(x_2 = \mathrm{A}|x = \mathrm{BM})),$$

where the logit function is defined as $\mathrm{logit}(p) = \ln \dfrac{p}{1-p}$. One can explicitly calculate $\log \pi$, ELBO and $\mathcal{D}^{\mathrm{L}}$ using these parameters to monitor the drift. As an explicit example, the log-likelihood and ELBO for AA can be calculated as

$$\log \pi(\mathrm{AA}) = \log\left(\frac{1}{2}\mathrm{S}(b)\mathrm{S}(c) + \frac{1}{2}\mathrm{S}(a)\mathrm{S}(d)\right),$$
$$\mathrm{ELBO}(\mathrm{AA}) = \frac{1}{2}(\log \mathrm{S}(a) + \log \mathrm{S}(b) + \log \mathrm{S}(c) + \log \mathrm{S}(d)).$$

Here $\mathrm{S}(x) = \dfrac{1}{1+e^{-x}}$ is the sigmoid function, the inverse function of the logit function, which recovers probabilities from logits. The parameters are all initialized to be 0.5 so that ELBO equals log-likelihood for all outputs, resulting in a pretrained diffusion model. We set the reward function to be

$$r(\mathrm{AA}) = 0.8, r(\mathrm{AB}) = 1, r(\mathrm{BA}) = 0.7, r(\mathrm{BB}) = 1.$$

The reward function is chosen for clarity of exposition and is without loss of generality.

Algorithm implementation. We implement FPO by updating the model parameters according to Equation (4), and SPG by updating the model parameters according to Equation (5). In the SPG implementation, a surrogate for EUBO (the $\mathcal{L}_{\mathrm{EUBO}}$ in Wang et al. (2025)) is used to tackle the intractability issue, and we follow their implementation in our experiment. In this toy setting, we directly calculate the policy gradient without invoking Monte Carlo samples for gradient estimation. We implement Algorithm 2 with FPO and SPG gradient estimators for comparison as well. We set the learning rate to be 0.1, the beta parameter in EUBO to be 1.5, β in DiPOD to be 0.2, and run these algorithms for 1500 policy-gradient steps.

Results. We summarize the results in Figure 4. For FPO, the variational gap grows substantially as training proceeds. For SPG, while staying more controlled, the gap is still substantial. Applying DiPOD to both algorithms effectively controls the gap.

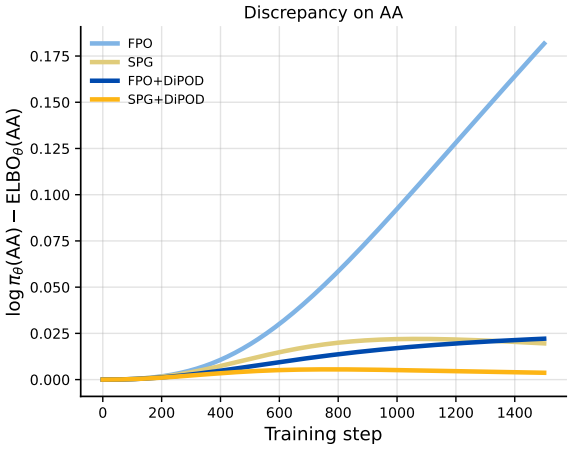


Figure 4: Variational gap $\mathcal{D}_\theta^{\mathrm{L}}$ during FPO, SPG, and DiPOD training in the two-token experiment.

E Additional Experiments for Diffusion Language Models

E.1 Experiments with More Sequence Lengths

We additionally report results at generation sequence lengths 128, 256, and 512 for SPG and SPG+DiPOD. We use the same setup as in Section 4.2. For GSM8K and MATH500 at all three lengths, and for Countdown and Sudoku at length 256, we use $\beta = 0.05$. For Countdown, we use $\beta = 0.04$ at length 128 and $\beta = 0.01$ at length 512. For Sudoku at lengths 128 and 512, we use $\beta = 0.02$. We summarize all tasks and sequence lengths in a single table. For GSM8K, MATH500, and Countdown, we copy the LLaDA-8B-Instruct and d1 baselines from SPG (Wang et al., 2025). For Sudoku, we do *not* use the 3-shot numbers reported in SPG; instead, we use the zero-shot LLaDA-8B-Instruct baselines and the d1 results at lengths 128 and 512 from d1 (Zhao et al., 2025), our reproduced d1 result at length 256, and the wd1 baselines from wd1 (Tang et al., 2025). The wd1 paper reports zero-shot results only at sequence lengths 256 and 512; for GSM8K, MATH500, and Countdown at sequence length 128, we therefore use the reproduced wd1 numbers in SPG (Wang et al., 2025), while the Sudoku 128 entry remains unavailable because the SPG Sudoku setting is 3-shot.

Compared with SPG, SPG+DiPOD improves GSM8K performance at all three sequence lengths. On MATH500, it improves performance at lengths 128 and 256, while performing slightly below SPG at length 512. On Countdown, SPG+DiPOD outperforms SPG at all three lengths, achieving 83.59%, 80.08%, and 71.09% at lengths 128, 256, and 512, respectively. On Sudoku, SPG+DiPOD improves accuracy from 44.40% to 44.74% at length 128, from 25.12% to 97.56% at length 256, and from 34.64% to 66.94% at length 512. The strongest Sudoku performance remains at length 256, where SPG+DiPOD reaches near-100% accuracy in the zero-shot setting.

E.2 Ablation on the DiPOD Coefficient β

We ablate the DiPOD coefficient β in Equation (7). Due to computational constraints, we conduct this study only on Countdown. This choice is also motivated by task characteristics: on GSM8K and MATH500, the gains are relatively small in our main results, while on Sudoku, single-run outcomes can exhibit discrete phase-transition-like behavior, which may lead to misleading conclusions in a limited ablation. Therefore, Countdown provides the cleanest setting for isolating the effect of β. Consistent with the fairness protocol used in the main experiments, we keep the random seed fixed across all ablation runs.

Table 3 shows that the choice of β has a clear effect on performance. A moderate regularization strength works best, with $\beta = 0.05$ achieving the strongest result. When β is too small, the additional ELBO tightening is not strong enough to sufficiently control the drift during policy optimization; when β is too large, the optimization appears to overemphasize the regularization term, which can

	GSM8K			MATH500			Countdown			Sudoku		
Method	128	256	512	128	256	512	128	256	512	128	256	512
LLaDA-8B-Instruct	69.5	77.2	79.8	28.2	32.4	34.6	18.8	16.8	16.8	11.7	6.7	5.5
d1	72.2	80.6	81.3	31.4	36.0	39.4	30.9	30.9	34.4	22.1	22.1	9.5
wd1	74.6	80.8	82.3	31.0	34.4	39.0	48.8	51.2	46.1	N/A	76.4	62.8
SPG	81.05	84.23	83.40	33.40	37.80	41.80	75.00	51.95	69.53	44.40	25.12	34.64
SPG+DiPOD	81.65	84.91	85.29	36.00	40.00	41.40	83.59	80.08	71.09	44.74	97.56	66.94

Table 2: Results across generation sequence lengths. For GSM8K, MATH500, and Countdown, the LLaDA-8B-Instruct and d1 baselines are copied from SPG. For Sudoku, we use zero-shot baselines: our reproduced d1 result at length 256, and published results from d1 and wd1 for the remaining entries. For wd1, sequence length 128 is taken from SPG's reproduced wd1 row for GSM8K, MATH500, and Countdown, since the original wd1 paper only reports zero-shot results at lengths 256 and 512; the Sudoku 128 entry is therefore unavailable.

Table 3: Ablation of the DiPOD coefficient β on Countdown. We report final accuracy under the same setup as Section 4.2.

β	Performance
0	51.95
0.01	53.91
0.03	55.47
0.05	80.08
0.055	76.56
0.06	71.09
0.07	52.73
0.10	55.86

weaken the policy-improving update. Overall, these results support the use of a moderate on-policy ELBO regularization strength and validate our default choice of $\beta = 0.05$ in the main language-model experiments.

E.3 Details of the Three-Shot Sudoku Case Study

We provide the setup and measurement procedure for the case study in Section 4.1 and Figure 2.

Setup. We compare FPO (McAllister et al., 2025), SPG (Wang et al., 2025), and their DiPOD variants on three-shot Sudoku using LLaDA-8B-Instruct. All four runs use training seed 42 and 1,250 optimizer steps, with $\beta = 0.05$ for both DiPOD variants. We examine training reward together with the estimated ELBO discrepancy to connect the double-drift mechanism in Section 3.1 to training behavior.

Reward curve. Each logged reward averages 96 fresh completions, comprising 16 prompt groups of six completions. Rollouts reused for later optimizer updates are not counted as new samples. Rewards are logged at steps $1, 5, \ldots, 1249$ and displayed with a trailing 10-observation moving average, using expanding windows initially. The recorded reward step is one greater than the generating model version used for the gap measurements.

Gap measurement. At model checkpoints $t \in \{0, 100, \ldots, 1200\}$, we evaluate 24 on-policy completions, comprising six completions from each of four prompt groups. Let θ_t denote the model parameters after iteration t. For prompt x_i and completion y_i, let L_i be the active completion length through the first EOS token, or 256 if no EOS occurs. We plot the mean of the individually length-normalized gap estimates,

$$\widehat{G}_t = \frac{1}{24} \sum_{i=1}^{24} \frac{\widehat{\ell}_{t,i} - \widehat{\mathcal{L}}_{t,i}}{L_i}, \qquad \ell_{t,i} = \log \pi_{\theta_t}^{\text{eval}}(y_i \mid x_i, L_i), \tag{30}$$

where $\widehat{\ell}_{t,i}$ estimates the evaluation model's log-likelihood and $\widehat{\mathcal{L}}_{t,i}$ estimates the ELBO for that same model, defined below. All logarithms are natural. The axis label $\log \pi - \text{ELBO}$ is shorthand for this estimated per-token quantity.

Evaluation model. Blockwise decoding is a standard choice in recent diffusion language model RL pipelines: both d1 (Zhao et al., 2025) and SPG (Wang et al., 2025) generate from left to right in 32-token blocks with confidence-based unmasking. Confidence-based unmasking is a decoding heuristic used to choose which positions to reveal during generation. LLaDA pretraining fits the model distribution using a diffusion likelihood bound on randomly masked text, without this confidence-based selection rule (Nie et al., 2025). For a fixed checkpoint–completion pair (t, i), write $\theta = \theta_t$ and $(x, y, L) = (x_i, y_i, L_i)$, and use the block notation of Section B.2: $B = \lceil L/32 \rceil$, $n_b = |y_b| \le 32$, and $c_b = (x, y_{<b}, L)$. Let $[n_b] = \{1, \dots, n_b\}$ and let y_b^H be block y_b with positions $H \subseteq [n_b]$ replaced by the mask token m. Write $p_{\theta,j}(\cdot \mid y_b^H, c_b)$ for the model's token distribution at position j, renormalized to exclude m. In every model call, the prompt and earlier blocks are visible, later blocks and positions beyond L remain masked, and the saved prompt attention mask is retained. For each block, sample a uniform permutation σ_b of its positions and reveal one token at a time:

$$\begin{aligned} \sigma_b &\sim \mathrm{Unif}(\mathfrak{S}_{n_b}), & H_{b,s} &= \{\sigma_{b,s}, \dots, \sigma_{b,n_b}\}, \\ y_{b,\sigma_{b,s}} &\sim p_{\theta,\sigma_{b,s}}(\cdot \mid y_b^{H_{b,s}}, c_b), & s &= 1, \dots, n_b. \end{aligned} \tag{31}$$

Here $\mathfrak{S}_{n_b}$ is the set of all $n_b!$ permutations. Reversing the reveal order gives a forward noising trajectory z_b with $q_b(z_b \mid y_b) = 1/n_b!$. Using the importance-weight notation of Section B.2, define

$$w_{\theta,b}(z_b) := \frac{p_\theta(y_b, z_b \mid c_b)}{q_b(z_b \mid y_b)} = \prod_{s=1}^{n_b} p_{\theta,\sigma_{b,s}}(y_{b,\sigma_{b,s}} \mid y_b^{H_{b,s}}, c_b). \tag{32}$$

The uniform position-selection factors cancel in this ratio. Thus $p_\theta(y_b \mid c_b) = \mathbb{E}_{q_b}[w_{\theta,b}]$ and $\mathcal{L}_b(\theta) = \mathbb{E}_{q_b}[\log w_{\theta,b}]$, giving the sequence quantities

$$\begin{aligned} \ell_{t,i} &= \log \pi_{\theta_t}^{\mathrm{eval}}(y_i \mid x_i, L_i) = \sum_{b=1}^{B} \log \mathbb{E}_{q_b}[w_{\theta,b}], \\ \mathcal{L}_{t,i} &= \sum_{b=1}^{B} \mathcal{L}_b(\theta) = \sum_{b=1}^{B} \mathbb{E}_{q_b}[\log w_{\theta,b}] \le \ell_{t,i}. \end{aligned} \tag{33}$$

This is the block ELBO in Equation (24), applied to the same learned weights for all four methods. Uniform reveals replace the confidence-based decoding heuristic and can change the output distribution; the RL updates also use method-specific masking and scoring objectives. The gap therefore diagnoses bound tightness under this fixed evaluation model, rather than directly measuring the confidence-based decoder's likelihood gap or the error in the training gradient.

Estimating likelihood and ELBO. All estimates below score the recorded tokens in y_i; only reveal orders or masked subsets are sampled. For likelihood, independently draw $K = 4 \times 16 = 64$ paths per block, corresponding to four replicates of 16 uniform reveal orders:

$$z_b^{(k)} \overset{\text{iid}}{\sim} q_b(\cdot \mid y_b), \quad k = 1, \dots, K, \qquad \widehat{\ell}_{t,i} = \sum_{b=1}^{B} \log\left(\frac{1}{K} \sum_{k=1}^{K} w_{\theta,b}(z_b^{(k)}) \right). \tag{34}$$

Thus all 64 weights are averaged within each block before taking the logarithm. For the ELBO, a uniform reveal order has a uniform hidden subset whenever h positions remain, and its next revealed position is uniform within that subset. Consequently, the same path ELBO can be written in terms of the mean masked-token log-probability:

$$\begin{aligned} a_{\theta,b}(H) &:= \frac{1}{|H|} \sum_{j \in H} \log p_{\theta,j}(y_{b,j} \mid y_b^H, c_b), \\ \mathcal{L}_b(\theta) &= \sum_{h=1}^{n_b} \mathbb{E}_{H \sim \mathrm{Unif}\{H \subseteq [n_b] : |H| = h\}}[a_{\theta,b}(H)]. \end{aligned} \tag{35}$$

We evaluate every count $h = 1, \ldots, n_b$ and draw $R = 4$ independent subsets at each count:

$$H_{b,h}^{(r)} \overset{\text{iid}}{\sim} \text{Unif}\{H \subseteq [n_b] : |H| = h\}, \qquad r = 1, \ldots, R,$$
$$\widehat{\mathcal{L}}_{t,i} = \sum_{b=1}^{B} \sum_{h=1}^{n_b} \frac{1}{R} \sum_{r=1}^{R} a_{\theta,b}(H_{b,h}^{(r)}). \tag{36}$$

The factor $1/h$ is already included in $a_{\theta,b}$; the contributions are summed over all counts without a further $1/n_b$ factor. Both estimates use the same recorded completion and are divided by L_i only when forming Equation (30).

Measurement cost. At the sampling budgets above, joint likelihood/ELBO evaluation averaged 16.1 minutes per completion on one NVIDIA H200 GPU. Each method's full curve (13 checkpoints with 24 completions each) was evaluated using eight GPUs and required 10.9–11.4 hours of wall-clock time. Across all four methods, the $1{,}248$ completion evaluations used 355.5 allocated GPU-hours, including model loading and job overhead, according to the recorded job accounting. These timings cover the likelihood/ELBO measurement jobs and exclude RL training, separate pilot convergence checks, and CPU calculations for the shaded intervals.

Confidence bands. At each checkpoint, we resample the saved measurements within the blocks and masked-token counts defined above, and the four prompt groups with each group's six completions kept together, allowing repeats. We recompute $\widehat{G}_t$ $2{,}000$ times using Equations (34) and (36), and use the 2.5th and 97.5th percentiles for the shaded interval. These intervals reflect variation within one training run; they exclude differences across training seeds and systematic likelihood-estimation error, which remains unresolved for the late-training baselines.

Joint reward and discrepancy dynamics. In Figure 2, FPO initially improves reward, then exhibits a pronounced decline as its estimated ELBO discrepancy grows. FPO+DiPOD keeps the discrepancy small and maintains a higher reward late in training. SPG+DiPOD likewise achieves a smaller late-training discrepancy and higher reward than SPG. These measurements use completions collected during training and evaluate the likelihood and ELBO. They therefore provide a diagnostic of bound tightness on those completions, rather than a direct measurement of the training policy's gap or gradient error.

F Continuous-Control Motion Tracking

F.1 Motion Tracking Experiment

We evaluate DiPOD on continuous-control diffusion policies using the motion-tracking setup of FPO++ (Yi et al., 2026), a variant of FPO with an adequate gradient estimator. The policy controls the Unitree G1 humanoid to track reference motions from LAFAN (Harvey et al., 2020) and is trained from scratch. All downstream FPO++ hyperparameters follow the original motion-tracking setup. The only added component is an initial self-distillation stage before normal FPO++ policy-gradient training. We use this single-stage version because policy-preserving self-distillation during high-dimensional motion-control training is itself a nontrivial algorithmic component; designing a fully interleaved self-distillation schedule for this setting is left to future work.

Self-distillation procedure. The self-distillation stage instantiates the self-distillation step in the original interleaved DiPOD procedure Algorithm 1. At the beginning of training, we clone the randomly initialized actor into a frozen teacher. We then collect rollouts using teacher actions: the frozen teacher samples actions, and we store the resulting observation and teacher action pairs. The student actor is trained on this growing dataset by maximizing the empirical average of the evidence lower bound. This phase is not reward learning: the critic, advantages, and surrogate policy-gradient loss are used only after self-distillation is complete. Thus the initial distillation step preserves the initial policy distribution while tightening the gap between ELBO and log-likelihood before the subsequent policy-gradient stage.

Algorithm 3 Initial self-distillation for motion tracking

Input: initial actor π_θ, vectorized environments, distillation iterations K

1: Clone the actor into a frozen teacher $\bar{\pi}$
2: Initialize replay buffer $\mathcal{D} \leftarrow \emptyset$
3: **for** $k = 1, \ldots, K$ **do**
4: Collect teacher rollouts in the live environments and append sampled observation-action pairs (o, a) to $\mathcal{D}$
5: Sample a minibatch $\{(o_i, a_i)\}_{i=1}^{B}$ from $\mathcal{D}$
6: Update only the student actor by maximizing $\frac{1}{B}\sum_{i=1}^{B} \mathrm{ELBO}_\theta(a_i|o_i)$
7: **end for**
8: Reset the optimizer state and continue with the FPO++ policy-gradient stage

In our runs, self-distillation uses 100 iterations, 8 rollout steps per iteration, minibatch size 16384, AdamW with learning rate $3 \cdot 10^{-4}$ and weight decay 10^{-4}, and actor gradient clipping at 1.0. The frozen teacher uses 64 sampling steps. The tracking tasks use 4096 parallel environments.

Computational cost. We run the experiments on a single NVIDIA L40S GPU. For *dance_1_subject_1*, the self-distillation stage takes 95.5s in total, or about 0.96s per self-distillation iteration. A policy-gradient iteration takes about 15s, so the 1000-iteration training runs shown in the figures take roughly 4.2 hours. Thus, the initial self-distillation stage adds only about 1.5 minutes of computation, which is negligible relative to the full training process.

Results. We report the full set of motion-tracking curves in Figures 5 and 6. Across all six LAFAN tracking tasks, DiPOD improves sample efficiency over the reproduced FPO++ baseline: the reward curves rise faster and the episode-length curves indicate more sustained successful tracking. The consistency across *dance_1_subject_1*, *dance_1_subject_2*, *fight_1_subject_2*, *jumps_1_subject_1*, *run_1_subject_2*, and *walk_1_subject_1* suggests that the initial self-distillation step provides a general benefit rather than a task-specific gain.

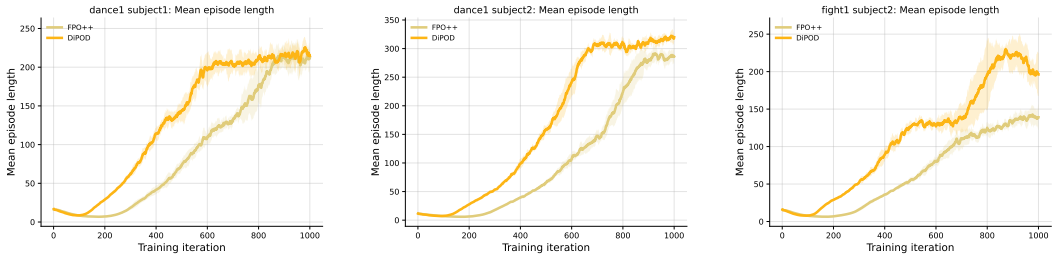


(a) *dance_1_subject_1* (b) *dance_1_subject_2* (c) *fight_1_subject_2*

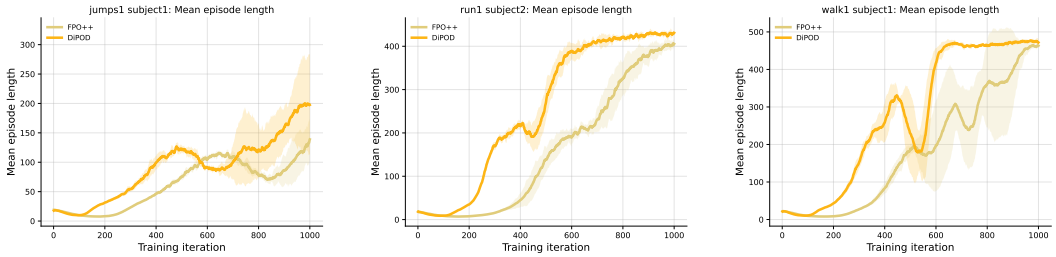


(d) *jumps_1_subject_1* (e) *run_1_subject_2* (f) *walk_1_subject_1*

Figure 5: Mean reward curves for all six LAFAN motion-tracking tasks on the G1 humanoid.



(a) *dance_1_subject_1* (b) *dance_1_subject_2* (c) *fight_1_subject_2*



(d) *jumps_1_subject_1* (e) *run_1_subject_2* (f) *walk_1_subject_1*

Figure 6: Mean episode length curves for all six LAFAN motion-tracking tasks on the G1 humanoid.