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 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 against previous methods.

1 INTRODUCTION

Diffusion models have emerged as a unified paradigm for generative modeling, spanning continuous domains such as images, video, and robotic control, as well as discrete domains such as language, code, and math reasoning with the rise of diffusion LLMs (dLLMs). As post-training via reinforcement learning becomes a primary driver of capability and alignment, a natural question follows: *Can we post-train diffusion models reliably with policy gradients?*

In this work we answer this question by introducing **DiPOD**—**Di**ffusion **P**olicy optimization with**O**ut **D**rifting apart—a principled framework for diffusion policy optimization that enforces *both ELBO–likelihood consistency* and *proxy–true policy gradient consistency*. Our key diagnosis is a *double drift* phenomenon: many diffusion-RL formulations optimize a variational proxy objective whose *ELBO can drift apart from the true log-likelihood* from the diffusion perspective; once this happens, the resulting *proxy policy gradient drifts apart from the true policy gradient of expected return* from the RL perspective. DiPOD prevents this double drift by *interleaving* (i) a self-distillation step that tightens the evidence bound *without changing the policy's output distribution*, with (ii) policy gradient updates that improve expected return. From this insight, we derive a surprisingly simple drop-in implementation: add a per-update ELBO regularization term to the diffusion policy gradient update. Despite its simplicity, this modification significantly stabilizes training and improves reward optimization in practice.

Why is updating diffusion with policy gradient hard in the first place? Standard policy gradient relies on advantage-weighted gradients of $\log \pi_\theta(a|o)$, but for diffusion policies the exact likelihood (and its gradient) is often intractable even though sampling is feasible. Early work sidestepped this by treating the denoising chain as an MDP (Black et al., 2023; Ren et al., 2024), which restores tractability at the cost of binding training to a particular sampler that effectively multiplies the decision horizon by the number of integration steps, making credit assignment cumbersome. This has motivated *non-MDP* alternatives that preserve flexible decoding while approximating the likelihood term: some Zhao et al. (2025); Xie et al. (2025) use direct likelihood surrogates (e.g., mean-field or one-step estimators in dLLM post-training, partial dependency restorations, or imposing an autoregressive order) at the cost of optimizing a proxy policy that may differ from the executed diffusion policy, while others McAllister et al. (2025); Wang et al. (2025) adopt a more principled variational-inference (VI) view, replacing $\log \pi_\theta$ with tractable evidence bounds such as ELBO (and sometimes EUBO) that

become exact when the bound is tight. The latter appears particularly appealing: evidence bounds preserve diffusion's native sampling flexibility and connect naturally to pretraining objectives.

A natural expectation is that an ELBO perspective should yield a tightly bounded and flexible route to diffusion RL. However, practice shows that training dynamics can be unstable, and improvements in the proxy objective do not reliably translate into monotonic policy improvement. Our analysis pinpoints the achilles heel: **Once ELBO drifts apart from likelihood, the policy gradient side is inevitably affected.**

In diffusion RL, an ELBO-weighted update entangles two effects: changing the true log-likelihood and changing the discrepancy. Critically, the same ELBO change can correspond to very different changes in $\log \pi_\theta$. For negative-advantage updates, the ELBO can even "cheat" by decreasing through an *increase* in discrepancy rather than reliably reducing the true likelihood of undesirable actions. This creates the first drift: *ELBO–likelihood inconsistency*. Once this happens, the second drift follows: *proxy–true policy gradient inconsistency*. The policy gradient we ultimately want is proportional to $\nabla_\theta \log \pi_\theta(a_t \mid o_t)$, but ELBO-based methods use $\nabla_\theta \mathrm{ELBO}_\theta = \nabla_\theta \log \pi_\theta - \nabla_\theta D_\theta$. Whenever the discrepancy is non-negligible, a gradient step on the proxy objective is not guaranteed to align with the true policy gradient direction, and this misalignment can accumulate over training—the **double drift** phenomenon. This also explains why variational diffusion-RL methods can work well *initially*: near a well-pretrained initialization where the bound is locally tight, the discrepancy is small and smooth, and the proxy gradient can be *adequate locally*. The problem is that RL updates can move the model out of this tight-bound regime, after which both drifts appear and compound.

DiPOD addresses this by making tight-bound behavior an explicit design constraint. Our key observation is that many ELBO-based estimators are *adequate locally*: if the diffusion model were perfectly trained so that the evidence bound is tight, the proxy gradient matches the true log-likelihood gradient, and a policy gradient step based on the proxy coincides with the true policy gradient. The problem is that policy optimization moves the model away from this tight-bound regime, causing the proxy gradient to progressively misalign. A way to prevent this misalignment is by repeatedly *pulling the model back* toward a tight-bound regime via self-distillation (which tightens the evidence bound under a reference rollout distribution without changing the policy distribution), and only then taking policy gradient steps. This interleaving ensures that throughout training, policy gradient updates remain aligned with the intended policy-improvement direction.

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 baseline is surprisingly effective: it substantially stabilizes training, reaches higher rewards, and improves diverse task performance across both discrete dLLM post-training tasks (e.g., GSM8K Cobbe et al. (2021), MATH500 Lightman et al. (2023), Sudoku Arel, Countdown Pan et al. (2025)) and continuous-control diffusion policies (e.g., MuJoCo-style manipulation/control tasks). (details in Section 4).

In summary, we make three contributions:

- We provide an ELBO-lens analysis of diffusion policy gradient methods and identify a fundamental failure mode: **double drift**, driven by *ELBO–likelihood inconsistency* and the resulting *proxy–true policy gradient inconsistency*.
- We propose **DiPOD**, an interleaved self-distillation / policy-update framework that enforces ELBO–likelihood consistency and thereby restores policy gradient consistency throughout training.
- We derive a **simple, drop-in practical algorithm** that approximates the interleaving by adding a per-update ELBO regularizer, yielding substantially more stable diffusion RL training and improved performance in practice.

1.1 Related Works

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), robotics (Black et al.; Bjorck et al., 2025). More recently, diffusion

language models also emerge as a strong alternative to autoregressive models for fast inference (Nie et al., 2025; Xie et al., 2025; Song et al., 2025). Historically, there are different ways to derive diffusion models, and in this paper we adopt the variational inference perspective (Kingma & Gao, 2023; Sohl-Dickstein et al., 2015).

Policy Gradients. In reinforcement learning, Policy gradient optimizes a parameterized policy by directly performing gradient descent on its expected cumulative return (Williams, 1992; Schulman et al., 2017), and has led to great success in high-dimensional control tasks (Schulman et al., 2015), locomotion tasks (Rudin et al., 2022), manipulation tasks (Schwarke et al., 2023), and so on. It also plays a central role in the post-training of large language models (Shao et al., 2024; Rafailov et al., 2023).

Diffusion RL. Classical policy gradient methods cannot be directly applied to diffusion models due to the intractable likelihood. A line of work tackles this problem by treating the denoising process as a Markov Decision Process (Black et al., 2023; Ren et al., 2024). Another line of work builds upon variational inferences (McAllister et al., 2025; Wang et al., 2025). In language domains, a line of work approximates the likelihoods by simplifying inter-token dependencies (Zhao et al., 2025; Yang et al., 2025; Xie et al., 2025).

2 Preliminaries

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 the 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 model, since the exact likelihood $\pi_\theta(a_t \mid o_t)$ and its gradient are unavailable—even though sampling from $\pi_\theta(\cdot \mid o_t)$ remains feasible. This challenge has motivated a variety of solutions that introduce proxy formulations to make policy optimization tractable. A line of works in language models addresses this challenge by approximating the likelihood, typically simplifying dependencies across tokens from a tractability perspective. For examples, *diffu*-GRPO (Zhao et al., 2025), UniGRPO Yang et al. (2025) and Dream-Coder (Xie et al., 2025) build upon the GRPO (Shao et al., 2024) framework, and achieve substantial improvements on language tasks. In contrast, A more principled line of work replaces the likelihood with 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. In diffusion models, $\mathcal{D}^{\text{L}}_\theta$ measures the mismatch between the learned denoising process and the true reverse diffusion process. Furthermore, with appropriate θ, $\mathcal{D}^{\text{L}}_\theta(o_t, a_t) = 0$ for all o_t and a_t.

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

where p_{data} is the distribution that the model tries to learn. Note that when a diffusion model is perfectly trained, ELBO equals the log-likelihood.

Similar to ELBO, another variational bound, EUBO, satisfies

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

where $\mathcal{D}_\theta^{\text{U}}$ is a non-negative discrepancy term that can reach zero with appropriate θ as well. 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. However, unlike ELBO, EUBO is fundamentally intractable and must be approximated in practice.

Prior Algorithms. In FPO (McAllister et al., 2025), $\log \pi_\theta(a_t|o_t)$ is replaced by $\text{ELBO}(a_t|o_t)$ to make Equation (1) tractable and obtain strong performances. As a result, the gradient update becomes a variant of

$$\mathbb{E}_\theta \left[\nabla_\theta \text{ELBO}(a_t|o_t) \hat{A}_t(o_t, a_t) \right]. \tag{4}$$

A follow-up work, SPG (Wang et al., 2025), points out a key limitation of FPO: the resulting optimization objective $\mathbb{E}_\theta \left[\text{ELBO}(a_t|o_t) \hat{A}_t(o_t, a_t) \right]$ is not a lower bound of the original objective due to the negative advantages. To address this issue, SPG incorporates EUBO, and updates θ by

$$\mathbb{E}_\theta \left[\mathbb{1}_{\hat{A}>0} \nabla_\theta \text{ELBO}_\theta(a_t|o_t) \hat{A} + \mathbb{1}_{\hat{A}<0} \nabla_\theta \text{EUBO}_\theta(a_t|o_t) \hat{A} \right] \tag{5}$$

where $\mathbb{1}$ is an indicator function, and we shorten $\hat{A}_t(o_t, a_t)$ to $\hat{A}$ for simplicity. In this way, the corresponding objective becomes a true lower bound of the original objective.

Finally, we note that a closely related challenge arises in reinforcement learning from preference data, such as reinforcement learning from human feedback (RLHF), where supervision is provided via preferences rather than explicit rewards. Typical RLHF pipelines involve objectives that requires likelihood computation. Similar to our setting, these objectives can be handled using variational bounds. For instance, VRPO (Zhu et al., 2025) replaces the log-likelihood terms in DPO (Rafailov et al., 2023) with ELBO, enabling efficient supervised fine-tuning for diffusion language models.

3 Method

As discussed in the previous section, diffusion policies allow efficient sampling but make the exact log-likelihood $\log \pi_\theta(a_t \mid o_t)$ (and its gradient) difficult to compute, forcing diffusion RL 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 apart 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 drift* 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 $\nabla_\theta \log \pi_\theta$.

FPO (McAllister et al., 2025). FPO uses an ELBO surrogate in place of $\log \pi_\theta$ and motivates its update by interpreting ELBO increases on (o_t, a_t) pairs with positive advantage as increasing the likelihood of desirable actions while improving the diffusion model by tightening the bound. 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 need 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. 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 is **objective-consistent**: *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 destroy objective-consistency in practice and is currently specialized to certain settings.
- Most importantly for our purposes: **objective consistency 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 in One Way

The discussion above suggests a concrete target: to prevent the second drift (proxy-gradient drift), we must prevent the first drift (ELBO–likelihood drift) from accumulating along the training trajectory. At the same time, it is important to note why VI-based methods can work well *initially*. 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

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

Hence, 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

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

This justifies the strong empirical performance of VI-based diffusion RL methods when initialized from a pretrained model: their gradients are *locally adequate* near 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.

We capture this "good when tight" behavior via the following definition.

Definition 3.1 (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

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

For example, the FPO gradient estimator can be written as

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

and by the analysis above it is adequate (and similarly for SPG). The remaining question is how to turn an estimator that is only *locally* adequate into an algorithm that *stays* in the adequate regime throughout training. That is, we need to prevent the first drift so that the second drift does not happen.

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 + η E_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 $a_t \sim \pi_{\text{ref}}(\cdot \mid o_t)$), 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.

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 $\mathcal{B} = \left\{(o^i, a^i)\right\}_{i=1,\ldots,m}$ 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

```
Initialize policy π_θ
while π_θ has not converged do
    Sample rollouts B = {(o^i, a^i)}_{i=1,...,m}
    ▷ Rollouts may be sampled from the current policy.
    Update θ using Equation (7) with B.
end while
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.

4 Experiment

In this section, we illustrate the effectiveness and versatility of Algorithm 2 through extensive experiments. We start from a Two-Token post-training experiment that allows us to explicitly show the drift of ELBO, and then show results for diffusion language models and continuous control tasks with MuJoCo Playground.

4.1 Two-Token Post-training

We show the drift of ELBO using the FPO algorithm in a toy post-training setting on discrete diffusion models.

Setup. 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 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 $\text{logit}(x) = \ln \frac{p}{1-p}$. One can explicitly calculate $\mathcal{D}_\theta^{\text{L}}$ using these parameters. The parameters are initialized to be all 0.5 so that ELBO equals log-likelihood for any output, simulating a pretrained diffusion model. We set the reward function to be $r(\text{AA}) = 0.8, r(\text{AB}) = 1, r(\text{BA}) = 0.7, r(\text{BB}) = 1$.

Algorithm Implementation. We implement FPO as updating the model parameters by Equation (4), and SPG as updating the model parameters by Equation (5). In this toy setting we can directly calculate the policy gradient without invoking Monte-Carlo samples for estimation. We implement Algorithm 2 with FPO and SPG gradient estimators for comparison as well. We set 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 1. We can see that for FPO, the variation gap becomes really large as the training goes on. For SPG, while staying more controlled, the gap is still substantial. By applying DiPOD to both algorithms, we can see that the gap can be effectively controlled.

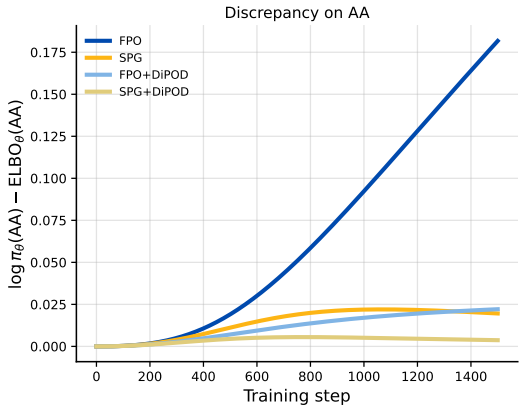

Figure 1: We use different colors to show the change of variational gap $\mathcal{D}_\theta^{\text{L}}$ throughout FPO, SPG, and DiPOD training.

4.2 Reasoning Tasks with dLLMs

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., 2023), Countdown (Pan et al., 2025), and Sudoku (Arel).

Baselines and Hyperparameter. 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 by ELBOs, while keeping the hyperparameters unchanged. Since the original FPO implementation contains design choices that do not transfer directly to the language setting, we denote the algorithm in this part as FPO*. We also keep the hyperparameters in DiPOD the same as d1. Similarly, for SPG experiments, we compare d1, SPG and Algorithm 2 with 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 all language experiments, we fix $\beta = 0.05$ in Equation (7), highlighting the consistent improvements of DiPOD over baseline algorithms.** 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 performances in the zero-shot setting.

Results. We summarize results in Table 1. From the tables we can see that FPO+DiPOD consistently improve the performances over FPO. Compared to SPG, the state-of-the-art algorithm in diffusion language model RL, SPG+DiPOD shows competitive performances in math reasoning tasks including

Table 1: Performance on reasoning benchmarks. Gains in blue indicate improvement over the corresponding baseline method (e.g., FPO*+DiPOD over FPO*, SPG+DiPOD over SPG).

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)	57.03 (+44.53)	25.78 (+18.90)
SPG	**86.10**	**40.00**	70.70	25.12
SPG+DiPOD	82.94	**40.00**	**80.08** (+9.38)	**97.56** (+72.44)

GSM8k and MATH500, and achieves significant leap in logical reasoning tasks including Countdown and Sudoku.

4.3 MuJoCo Playground

Setup We follow the setup of the MuJoCo experiments in FPO. We test the algorithms on BallInCup, FingerSpin, FingerTurnEasy and FingerTurnHard. The original FPO algorithm already does well on FingerTurnEasy and FingerTurnHard, while there is still space for improvements for BallInCup and FingerSpin. For each environment, we run experiments for 8 random seeds and report the mean performance.

Baseline and Hyperparameter. We keep the hyperparameters from the FPO paper as a baseline, and implement DiPOD on top of it. While keeping the existing hyperparameters in FPO, we show the performance of Algorithm 2 with $\beta \in \{0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.004, 0.008\}$.

Results. We summarize the results in Figure 2. For BallInCup and FingerSpin, we can see that with appropriate β the performance can be further enhanced. For FingerTurnEasy and FingerTurnHard, we can see that DiPOD reliably maintains the strong performance of FPO.

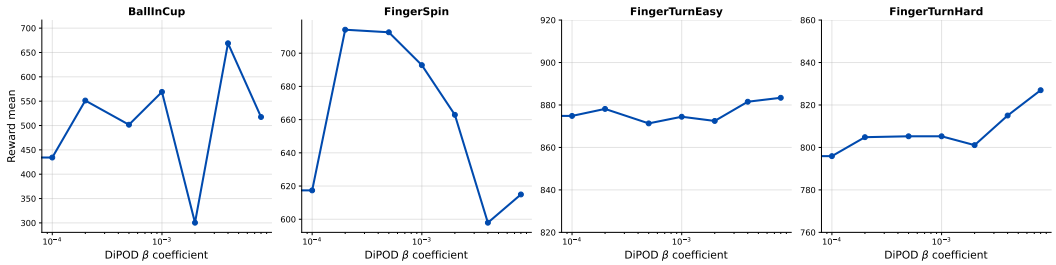

Figure 2: Mean rewards of FPO+DiPOD on MuJoCo control tasks with different β

5 Conclusion

We identify a fundamental double drift issue in diffusion reinforcement learning: when variational bounds drift away from the true likelihood, the resulting proxy policy gradients can drift away from the true policy gradient, breaking policy improvement guarantees. We propose DiPOD, an interleaved framework that maintains on-policy bound tightness to preserve gradient consistency, and showed that a simple per-update ELBO regularization is sufficient to substantially stabilize training and improve performance across tasks. An important direction for future work is to understand the optimal way to interleave bound tightening and policy updates, as well as to explore alternative mechanisms for enforcing gradient consistency beyond ELBO regularization.

References

Arel. Arel's sudoku generator. URL https://www.ocf.berkeley.edu/~arel/sudoku/main.html.

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, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. $\pi 0$: A vision-language-action flow model for general robot control. corr, abs/2410.24164, 2024. doi: 10.48550. *arXiv preprint ARXIV.2410.24164*.

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

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.

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.

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.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

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.

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, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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.

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.

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.

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.

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

A Backgrounds on Reinforcement Learning and Diffusion Models

A.1 Reinforcement Learning

In this part we provide a concise introduction to relevant RL algorithms in this paper.

Markovian Decision Process. A Markovian 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 an reward $r_t \sim R(\cdot|s_t, a_t)$. The episode terminates when the agent reaches a terminal state $s^* \in \mathcal{S}$. An RL algorithm aims at optimizing the culumative 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:

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

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

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

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 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

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

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 perfoms gradient update 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 for PPO on 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. Such 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 a thinking 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 with for q as

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

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 advantage inside a group of rollouts instead of the true advantage, eliminating the need to train a value function and calculating a different value function for different timesteps. Dream-Coder (Xie et al., 2025) use 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) and UniGRPO (Yang et al., 2025) account for this factor, but still partially ignore the inter-token dependencies for tractability.

A.2 Diffusion Models

In this part we provide a brief introduction to diffusion models. 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

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

The quantity $\text{ELBO}_\theta(x_0)$ 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 $p_\theta(z|x_0) = p_\theta(x_0, z)/p_\theta(x_0)$,

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

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 $\text{KL}(q\|p) := \mathbb{E}_q\left[\log\frac{q}{p}\right]$, corresponding to the $\mathcal{D}_\theta^{\text{L}}$ in Section 2 which is nonnegative and equals 0 iff $q = p$. This decomposition motivates variational inference: ELBO maximization simultaneously (i) increases the data likelihood through the joint term $\log p_\theta(x_0, z)$ and (ii) encourages the variational posterior $q(z|x_0)$ to match the true posterior $p_\theta(z|x_0)$, thereby tightening the bound. 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 diffusion 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).