Amortizing intractable inference in diffusion models for vision, language, and control

Siddarth Venkatraman* Mila, Université de Montréal

Moksh Jain* Mila, Université de Montréal

Luca Scimeca* Mila, Université de Montréal

Minsu Kim* Mila, Université de Montréal KAIST

Marcin Sendera* Mila, Université de Montréal Jagiellonian University

Mohsin Hasan Mila, Université de Montréal

Luke Rowe Mila, Université de Montréal

Sarthak Mittal Mila, Université de Montréal

Pablo Lemos Mila, Université de Montréal Ciela Institute Dreamfold

Emmanuel Bengio Recursion

Alexandre Adam Mila, Université de Montréal Ciela Institute

Jarrid Rector-Brooks Mila, Université de Montréal Dreamfold

Yoshua Bengio Mila, Université de Montréal CIFAR

Glen Berseth Mila, Université de Montréal CIFAR

Nikolay Malkin Mila, Université de Montréal University of Edinburgh

{ siddarth.venkatraman,moksh.jain,luca.scimeca minsu.kim,marcin.sendera,...,nikolay.malkin }@mila.quebec

Abstract

Diffusion models have emerged as effective distribution estimators in vision, language, and reinforcement learning, but their use as priors in downstream tasks poses an intractable posterior inference problem. This paper studies *amortized* sampling of the posterior over data, $\mathbf{x} \sim p^{\text{post}}(\mathbf{x}) \propto p(\mathbf{x})r(\mathbf{x})$, in a model that consists of a diffusion generative model prior $p(\mathbf{x})$ and a black-box constraint or likelihood function $r(\mathbf{x})$. We state and prove the asymptotic correctness of a data-free learning objective, *relative trajectory balance*, for training a diffusion model that samples from this posterior, a problem that existing methods solve only approximately or in restricted cases. Relative trajectory balance arises from the generative flow network perspective on diffusion models, which allows the use of deep reinforcement learning techniques to improve mode coverage. We illustrate the broad potential of unbiased inference of arbitrary posteriors under diffusion priors across a collection of experiments: in vision (classifier guidance), language (infilling under a discrete diffusion LLM), and multimodal data (text-to-image generation). Beyond generative modeling, we apply relative trajectory balance to the problem of continuous control with a score-based behavior prior, achieving state-of-the-art results on benchmarks in offline reinforcement learning. Code is available at this link.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Table 1: Sources of diffusion priors and constraints.

Domain	Prior $p(\mathbf{x})$	Constraint $r(\mathbf{x})$	Posterior
Conditional image generation (§3.1)	Image diffusion model $p(\mathbf{x})$	Classifier likelihood $p(c \mid \mathbf{x})$	Class-conditional distribution $p(\mathbf{x} \mid c)$
Text-to-image generation (§3.2)	Text-to-image foundation model	RLHF reward model	Aligned text-to-image model
Language infilling (§3.3)	Discrete diffusion model	Autoregressive completion likelihood	Infilling distribution
Offline RL policy extraction (§3.4)	Diffusion model as behavior policy	Boltzmann dist. of Q-function	Optimal KL-constrained policy

1 Introduction

Diffusion models [68, 27, 72] are a powerful class of hierarchical generative models, used to model complex distributions over images [51, 12, 63], text [4, 13, 40, 24, 23, 43], and actions in reinforcement learning [30, 83, 32] to name a few. In each of these domains, downstream problems require sampling product distributions, where a pretrained diffusion model serves as a prior $p(\mathbf{x})$ that is multiplied by an auxiliary constraint $r(\mathbf{x})$. For example, if $p(\mathbf{x})$ is a prior over images defined by a diffusion model, and $r(\mathbf{x}) = p(c \mid \mathbf{x})$ is the likelihood that an image $\mathbf{x}$ belongs to class c, then class-conditional image generation requires sampling from the Bayesian posterior $p(\mathbf{x} \mid c) \propto p(\mathbf{x})p(c \mid \mathbf{x})$. In offline reinforcement learning, if $\mu(a \mid s)$ is a conditional diffusion model over actions serving as a behavior policy, KL-constrained policy improvement [55, 44] requires sampling from the normalized product of $\mu(a \mid s)$ with a Boltzmann distribution defined by a Q-function, $\pi^*(a \mid s) \propto \mu(a \mid s)\exp(\beta Q(s, a))$. In language modeling, various conditional generation problems [43, 23, 29] amount to posterior sampling under a discrete diffusion model prior. Table 1 summarizes four such problems that the proposed method improves upon prior work.

The hierarchical nature of the generative process in diffusion models, which generate samples from $p(\mathbf{x})$ by a deep chain of stochastic transformations, makes exact sampling from posteriors $p(\mathbf{x})r(\mathbf{x})$ under a black-box function $r(\mathbf{x})$ intractable. Common solutions to this problem involve inference techniques based on linear approximations [73, 33, 31, 11] or stochastic optimization [22, 48]. Others estimate the 'guidance' term – the difference in drift functions between the diffusion models sampling the prior and posterior – by training a classifier on noised data [12], but when such data is not available, one must resort to approximations or Monte Carlo estimates [70, 14, 10], which are challenging to scale to high-dimensional problems. Reinforcement learning methods that have recently been proposed for this problem [8, 16] are biased and prone to mode collapse (Fig. 1).

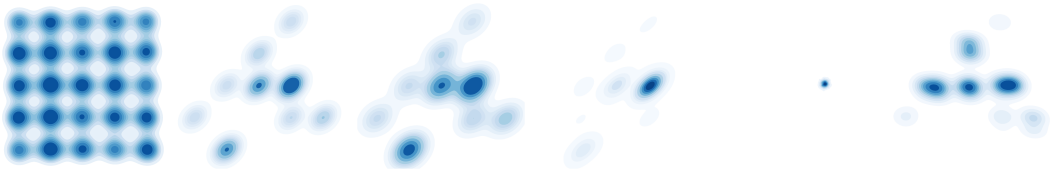
Contributions. Inspired by recent techniques in training diffusion models to sample distributions defined by unnormalized densities [89, 62, 78, 65], we propose an asymptotically unbiased training objective, called relative trajectory balance (RTB), for training diffusion models that sample from posterior distributions under a diffusion model prior (§2.2). RTB is derived from the perspective of diffusion models as continuous generative flow networks [38]. This perspective also allows us to freely leverage off-policy training, when data with high density under the posterior is available (§2.3). RTB can be applied to iterative generative processes beyond standard diffusion models: our methods generalize to discrete diffusion models and extend existing methods for autoregressive language models (§2.4).

Our experiments demonstrate the versatility of our approach in a variety of domains:

- In **vision**, we show that RTB achieves competitive classifier-guided image generation for unconditional diffusion vision priors (§3.1) and can be used to improve caption-conditioned generation under text-to-image foundation model priors (§3.2).
- In **language modeling**, we report strong results for infilling tasks with discrete diffusion language models (§3.3).
- Finally, we show that RTB achieves state-of-the-art results on **continuous control** benchmarks that leverage score-based behavior priors (§3.4).

2 Learning posterior samplers with diffusion priors

We consider the problem of posterior inference under a prior given by a hierarchical generative model. In this section, we present the mathematical setting (§2.1), our proposed RTB objective (§2.2), and training methods for RTB (§2.3). We will first discuss the case of a diffusion prior over $\mathbb{R}^d$, and later discuss how the methods generalize to arbitrary hierarchical priors (§2.4).



(a) Prior (b) Posterior (c) RTB (ours) (d) RL (KL reg.) (e) RL (no reg.) (f) CG

Figure 1: Sampling densities learned by various posterior inference methods. The prior is a diffusion model sampling a mixture of 25 Gaussians (a) and the posterior is the product of the prior with a constraint that masks all but 9 of the modes (b). Our method (RTB) samples close to the true posterior (c). RL methods with tuned KL regularization yield inaccurate inference (d), while without KL regularization, they mode-collapse (e). A classifier guidance (CG) approximation (f) results in biased outcomes. For details, see §C.

2.1 Background and setting: Diffusion models as hierarchical generative models

A denoising diffusion model generates data $\mathbf{x}_1$ by a Markovian generative process:

$$\textit{(noise)} \quad \mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \mathbf{x}_{2\Delta t} \to \ldots \to \mathbf{x}_1 = \mathbf{x} \quad \textit{(data)}, \tag{1}$$

where $\Delta t = \frac{1}{T}$ and T is the number of discretization steps.[1] The initial distribution $p(\mathbf{x}_0)$ is fixed (typically to $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$) and the transition from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$ is modeled as a Gaussian perturbation with time-dependent variance:

$$p(\mathbf{x}_{t+\Delta t} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t+\Delta t} \mid \mathbf{x}_t + u_t(\mathbf{x}_t)\Delta t, \sigma_t^2 \Delta t \boldsymbol{I}). \tag{2}$$

The scaling of the mean and variance by Δt is insubstantial for fixed T, but ensures that the diffusion process is well-defined in the limit $T \to \infty$ assuming regularity conditions on u_t [53, 64]. The process given by (1, 2) is then identical to Euler-Maruyama integration of the stochastic differential equation (SDE) $d\mathbf{x}_t = u_t(\mathbf{x}_t)\, dt + \sigma_t\, d\mathbf{w}_t$.

The likelihood of a denoising trajectory $\mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \cdots \to \mathbf{x}_1$ factors as

$$p(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1) = p(\mathbf{x}_0) \prod_{i=1}^{T} p(\mathbf{x}_{i\Delta t} \mid \mathbf{x}_{(i-1)\Delta t}) \tag{3}$$

and defines a marginal density over the data space:

$$p(\mathbf{x}_1) = \int p(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1)\, d\mathbf{x}_0\, d\mathbf{x}_{\Delta t} \ldots d\mathbf{x}_{1-\Delta t}. \tag{4}$$

A reverse-time process, $\mathbf{x}_1 \to \mathbf{x}_{1-\Delta t} \to \cdots \to \mathbf{x}_0$, with densities q, can be defined analogously, and similarly defines a conditional density over trajectories:

$$q(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_{1-\Delta t} \mid \mathbf{x}_1) = \prod_{i=1}^{T} q(\mathbf{x}_{(i-1)\Delta t} \mid \mathbf{x}_{i\Delta t}). \tag{5}$$

In the training of diffusion models, as discussed below, the process q is typically fixed to a simple distribution (usually a discretized Ornstein-Uhlenbeck process), and the result of training is that p and q are close as distributions over trajectories.

Diffusion model training as divergence minimization. Diffusion models parametrize the drift $u_t(\mathbf{x}_t)$ in (Equation 2) as a neural network $u(\mathbf{x}_t, t; \theta)$ with parameters θ and taking $\mathbf{x}_t$ and t as input. We denote the distributions over trajectories induced by (Equation 3, Equation 4) by p_θ to show their dependence on the parameter.

In the most common setting, diffusion models are trained to maximize the likelihood of a dataset. In the notation above, this corresponds to assuming $q(\mathbf{x}_1)$ is fixed to an empirical measure (with the

[1] The time indexing suggestive of an SDE discretization is used for consistency with the diffusion samplers literature [89, 65]. The indexing $\mathbf{x}_T \to \mathbf{x}_{T-1} \to \cdots \to \mathbf{x}_0$ is often used for diffusion models trained from data.

points of a training dataset $\mathcal{D}$ assumed to be i.i.d. samples from $q(\mathbf{x}_1)$). Training minimizes with respect to θ the divergence between the processes q and p_θ:

$$\begin{aligned} &D_{\mathrm{KL}}(q(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1) \,\|\, p_\theta(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1)) \\ &= D_{\mathrm{KL}}(q(\mathbf{x}_1) \,\|\, p_\theta(\mathbf{x}_1)) + \mathbb{E}_{\mathbf{x}_1 \sim q(\mathbf{x}_1)} D_{\mathrm{KL}}(q(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_{1-\Delta t} \mid \mathbf{x}_1) \,\|\, p_\theta(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_{1-\Delta t} \mid \mathbf{x}_1)) \\ &\geq D_{\mathrm{KL}}(q(\mathbf{x}_1) \,\|\, p_\theta(\mathbf{x}_1)) = \mathbb{E}_{\mathbf{x}_1 \sim q(\mathbf{x}_1)}[-\log p_\theta(\mathbf{x}_1)] + \text{const.} \end{aligned} \tag{6}$$

where the inequality – an instance of the data processing inequality for the KL divergence – shows that minimizing the divergence between distributions over trajectories is equivalent to maximizing a lower bound on the data log-likelihood under the model p_θ.

As shown in [71], minimization of the KL in (Equation 6) is essentially equivalent to the traditional approach to training diffusion models via denoising score matching [80, 68, 27]. Such training exploits that for typical choices of the noising process q, the optimal $u_t(\mathbf{x}_t)$ can be expressed in terms of the Stein score of $q(\mathbf{x}_1)$ convolved with a Gaussian, allowing an efficient stochastic regression objective for u_t. For full generality of our exposition for arbitrary iterative generative processes, we prefer to think of (Equation 6) as the primal objective and denoising score matching as an efficient means of minimizing it.

Trajectory balance and distribution-matching training. From (Equation 6) we also see that the bound is tight if the conditionals of p_θ and q on $\mathbf{x}_1$ coincide, *i.e.*, q is equal to the posterior distribution of p conditioned on $\mathbf{x}_1$. Indeed, the model p_θ minimizes (Equation 6) for a distribution with continuous density $q(\mathbf{x}_1)$ if and only if, for all denoising trajectories,

$$p_\theta(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1) = q(\mathbf{x}_1) q(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_{1-\Delta t} \mid \mathbf{x}_1). \tag{7}$$

This was named the *trajectory balance (TB) constraint* by [38] – by analogy with a constraint for discrete-space iterative sampling [46] – and is a time-discretized version of a constraint used for enforcing equality of continuous-*time* path space measures in [52] (see [7] for asymptotic analysis).

In [61, 38], the constraint (7) was used for the training of diffusion models in a *data-free* setting, where instead of i.i.d. samples from $q(\mathbf{x}_1)$ one has access to a (possibly unnormalized) density $q(\mathbf{x}_1) = e^{-\mathcal{E}(\mathbf{x}_1)}/Z$ from which one wishes to sample. These objectives minimize the squared log-ratio between the two sides of (7), which allows the trajectories $\mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \cdots \to \mathbf{x}_1$ used for training to be sampled from any training distribution, such as 'exploratory' modifications of p_θ or trajectories found by local search (MCMC) in the target space. The flexibility of off-policy exploration that this allows was studied by [65]. Such objectives contrast with on-policy, simulation-based approaches that require differentiating through the sampling process [*e.g.*, 89, 78, 6, 79].

2.2 Intractable inference under diffusion priors

Consider a diffusion model p_θ, defining a marginal density $p_\theta(\mathbf{x}_1)$, and a positive constraint function $r : \mathbb{R}^d \to \mathbb{R}_{>0}$. We are interested in training a diffusion model p_ϕ^{post}, with drift function u_ϕ^{post}, that would sample the product distribution $p^{\mathrm{post}}(\mathbf{x}_1) \propto p_\theta(\mathbf{x}_1) r(\mathbf{x}_1)$. If $r(\mathbf{x}_1) = p(\mathbf{y} \mid \mathbf{x}_1)$ is a conditional distribution over another variable $\mathbf{y}$, then p^{post} is the Bayesian posterior $p_\theta(\mathbf{x}_1 \mid \mathbf{y})$.

Because samples from $p^{\mathrm{post}}(\mathbf{x}_1)$ are not assumed to be available, one cannot directly train p using the objective (6). Nor can one directly apply objectives for distribution-matching training, such as those that enforce (7), since the marginal $p_\theta(\mathbf{x}_1)$ is not available. However, we make the following observation (proof in §A).

Proposition 1 (Relative TB constraint)**.** *If p_θ, p_ϕ^{post}, and the scalar Z_ϕ jointly satisfy the* relative trajectory balance (RTB) constraint

$$Z_\phi \cdot p_\phi^{\mathrm{post}}(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1) = r(\mathbf{x}_1) p_\theta(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1) \tag{8}$$

for every denoising trajectory $\mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \cdots \to \mathbf{x}_1$, then $p_\phi^{\mathrm{post}}(\mathbf{x}_1) \propto p_\theta(\mathbf{x}_1) r(\mathbf{x}_1)$, i.e., the diffusion model p_ϕ^{post} samples the posterior distribution. Furthermore, if p_θ also satisfies the TB constraint (7) with respect to the noising process q and some target density $q(\mathbf{x}_1)$, then p_ϕ^{post} satisfies the TB constraint with respect to the target density $q^{\mathrm{post}}(\mathbf{x}_1) \propto q(\mathbf{x}_1) r(\mathbf{x}_1)$, and $Z = \int q(\mathbf{x}_1) r(\mathbf{x}_1)\, d\mathbf{x}_1$.

Note that the two joints appearing in (8) are defined as products over transitions, via (3).

Relative trajectory balance as a loss. Analogously to the conversion of the TB constraint (7) into a trajectory-dependent training objective in [46, 38], we define the *relative trajectory balance loss* as the discrepancy between the two sides of (8), seen as a function of the vector ϕ that parametrizes the posterior diffusion model and the scalar Z_ϕ (parametrized via $\log Z_\phi$ for numerical stability):

$$\mathcal{L}_{\text{RTB}}(\mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \cdots \to \mathbf{x}_1; \phi) := \left(\log \frac{Z_\phi \cdot p_\phi^{\text{post}}(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1)}{r(\mathbf{x}_1) p_\theta(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1)} \right)^2 . \tag{9}$$

Optimizing this objective to 0 for all trajectories ensures that (8) is satisfied. While the RTB constraint (8) has a similar form to TB (7), RTB involves the ratio of two denoising processes, while TB involves the ratio of a forward and a backward process. However, the name 'relative TB' is justified by interpreting the densities in a TB constraint relative to a measure defined by the prior model; see §2.4.

If we assume $p_\theta(\mathbf{x}_0) = p_\phi^{\text{post}}(\mathbf{x}_0)$ are fixed (*e.g.*, to a standard normal), then (9) reduces to

$$\left(\log \frac{Z_\phi}{r(\mathbf{x}_1)} + \sum_{i=1}^{T} \log \frac{p_\phi^{\text{post}}(\mathbf{x}_{i\Delta t} \mid \mathbf{x}_{(i-1)\Delta t})}{p_\theta(\mathbf{x}_{i\Delta t} \mid \mathbf{x}_{(i-1)\Delta t})} \right)^2 . \tag{10}$$

Notably, the gradient of this objective with respect to ϕ does not require differentiation (backpropagation) into the sampling process that produced a trajectory $\mathbf{x}_0 \to \cdots \to \mathbf{x}_1$. This offers two advantages over on-policy simulation-based methods: (1) the ability to optimize $\mathcal{L}_{\text{RTB}}$ as an off-policy objective, *i.e.*, sampling trajectories for training from a distribution different from p_ϕ^{post} itself, as discussed further in §2.3; (2) backpropagating only to a subset of the summands in (10), when computing and storing gradients for all steps in the trajectory is prohibitive for large diffusion models (see §H.1).

Comparison with classifier guidance. It is interesting to contrast the RTB training objective with the technique of *classifier guidance* [12] used for some problems of the same form. If $r(\mathbf{x}_1) = p(\mathbf{y} \mid \mathbf{x}_1)$ is a conditional likelihood, classifier guidance relies upon writing $u_t(\mathbf{x}_t) - u_t^{\text{post}}(\mathbf{x}_t)$ explicitly in terms of $\nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_t)$, by combining the expression of the optimal drift u_t in terms of the score of the target distribution convolved with a Gaussian (cf. §2.1), with the 'Bayes' rule' for the Stein score: $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t \mid \mathbf{y}) = \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_t)$.

Classifier guidance gives the *exact* solution for the posterior drift when a differentiable classifier on noisy data, $p(\mathbf{y} \mid \mathbf{x}_t) = \int p(\mathbf{y} \mid \mathbf{x}_1) p(\mathbf{x}_1 \mid \mathbf{x}_t)\, d\mathbf{x}_1$, is available. Unfortunately, such a classifier is not, in general, tractable to derive from the classifier on noiseless data, $p(\mathbf{y} \mid \mathbf{x}_1)$, and cannot be learned without access to unbiased data samples. RTB is an asymptotically unbiased objective that recovers the difference in drifts (and thus the gradient of the log-convolved likelihood) in a data-free manner.

2.3 Training, parametrization, and conditioning

Training and exploration. The choice of which trajectories we use to take gradient steps with the RTB loss can have a large impact on sample efficiency. In *on-policy* training, we use the current policy p_ϕ^{post} to generate trajectories $\tau = (\mathbf{x}_0 \to \ldots \to \mathbf{x}_1)$, evaluate the reward $\log r(\mathbf{x}_1)$ and the likelihood of τ under p_θ, and a gradient updates on ϕ to minimize $\mathcal{L}_{\text{RTB}}(\tau; \phi)$.

However, on-policy training may be insufficient to discover the modes of the posterior distribution. In this case, we can perform *off-policy* exploration to ensure mode coverage. For instance, given samples $\mathbf{x}_1$ that have high density under the target distribution, we can sample *noising* trajectories $\mathbf{x}_1 \leftarrow \mathbf{x}_{1-\Delta t} \leftarrow \ldots \leftarrow \mathbf{x}_0$ starting from these samples and use such trajectories for training. Another effective off-policy training technique uses replay buffers. We expect the flexibility of mixing on-policy training with off-policy exploration to be a strength of RTB over on-policy RL methods, as was shown for distribution-matching training of diffusion models in [65].

Conditional constraints and amortization. Above we derived and proved the correctness of the RTB objective for an arbitrary positive constraint $r(\mathbf{x}_1)$. If the constraints depend on other variables $\mathbf{y}$ – for example, $r(\mathbf{x}_1; \mathbf{y}) = p(\mathbf{y} \mid \mathbf{x}_1)$ – then the posterior drift u_ϕ^{post} can be conditioned on $\mathbf{y}$ and the learned scalar $\log Z_\phi$ replaced by a model taking $\mathbf{y}$ as input. Such conditioning achieves amortized inference and allows generalization to new $\mathbf{y}$ not seen in training. Similarly, all of the preceding discussion easily generalizes to *priors* that are conditioned on some context variable.

Efficient parametrization and Langevin inductive bias. Because the deep features learned by the prior model u_θ are expected to be useful in expressing the posterior drift u_ϕ^{post}, we can choose to initialize u_ϕ^{post} as a copy of u_θ and to fine-tune it, possibly in a parameter-efficient way (as described in each section of §3). This choice is inspired by the method of amortizing inference in large language models by fine-tuning a prior model to sample an intractable posterior [29].

Furthermore, if the constraint $r(\mathbf{x}_1)$ is differentiable, we can impose an inductive bias on the posterior drift similar to the one introduced for diffusion samplers of unnormalized target densities in [89] and shown to be useful for off-policy methods in [65]. namely, we write

$$u_\phi^{\text{post}}(\mathbf{x}_t, t) = \text{NN}_1(\mathbf{x}_t, t; \phi) + \text{NN}_2(\mathbf{x}_t, t, \phi)\nabla_{\mathbf{x}_t} \log r(\mathbf{x}_t), \tag{11}$$

where NN_1 and NN_2 are neural networks outputting a vector and a scalar, respectively. This parametrization allows the constraint to provide a signal to guide the sampler at intermediate steps.

Stabilizing the loss. We propose two simple design choices for stabilizing RTB training. First, the loss in (9) can be replaced by the empirical *variance* over a minibatch of the quantity inside the square, which removes dependence on $\log Z_\phi$ and is especially useful in conditional settings, consistent with the findings of [65]. This amounts to a relative variant of the VarGrad objective [61] (see (23) in §G). Second, we employ loss clipping: to reduce sensitivity to an imperfectly fit prior model, we do not perform updates on trajectories where the loss is close to 0 (see §E,§F).

2.4 Generative flow networks and extension to other hierarchical processes

RTB as TB under the prior measure. The theoretical foundations for continuous generative flow networks [38] establish the correctness of enforcing constraints such as trajectory balance (7) for training sequential samplers, such as diffusion models, to match unnormalized target densities. While we have considered Gaussian transitions and identified transition kernels with their densities with respect to the Lebesgue measure over $\mathbb{R}^d$, these foundations generalize to more general *reference measures*. In §B, we show how the RTB constraint can be recovered as a special case of the TB constraint for a certain choice of reference measure derived from the prior.

Extension to arbitrary sequential generation. While our discussion was focused on diffusion models for continuous spaces, the RTB objective can be applied to any Markovian sequential generative process, in particular, one that can be formulated as a generative flow network in the sense of [5, 38]. This includes, in particular, generative models that generate objects by a sequence of discrete steps, including autoregressive models and discrete diffusion models. In the case of discrete diffusion, where the intermediate latent variables $\mathbf{x}_t$ lie not in $\mathbb{R}^d$ but in the space of sequences, one simply replaces the Gaussian transition densities by transition probability *masses* in the RTB constraint (8) and objective (9). In the case of autoregressive models, where only one sequence of steps can generate any given object, the backward process q becomes trivial, and the RTB constraint for a model p_ϕ^{post} to sample a sequence $\mathbf{x}$ from a distribution with density $r(\mathbf{x})p_\theta(\mathbf{x})$ is simply $Z_\phi p_\phi^{\text{post}}(\mathbf{x}) = r(\mathbf{x})p_\theta(\mathbf{x})$ for all sequences $\mathbf{x}$. We note that a sub-trajectory generalization of this objective was used in [29] to amortize intractable inference in autoregressive language models.

3 Experiments

In this section, we present empirical results to validate the efficacy of relative trajectory balance. Our experiments are designed to demonstrate the wide applicability of RTB to sample from posteriors for diffusion priors with arbitrary rewards on vision, language, and continuous control tasks.

3.1 Class-conditional posterior sampling from unconditional diffusion priors

We evaluate RTB in a classifier-guided visual task where we wish to learn a diffusion posterior $p_\phi^{\text{post}}(\mathbf{x} \mid c) \propto p_\theta(\mathbf{x})p(c \mid \mathbf{x})$ given a pretrained diffusion prior $p_\theta(\mathbf{x})$ and a classifier $r(\mathbf{x}) = p(c \mid \mathbf{x})$.

Setup. We consider two 10-class image datasets, MNIST and CIFAR-10, using off-the-shelf unconditional diffusion priors from [27] and standard classifiers $p(c \mid \mathbf{x})$ for both datasets. We perform parameter-efficient fine-tuning of p_ϕ^{post}, initialized as a copy of the prior p_θ, using the RTB objective (see §E.1 for details). The RTB objective is optimized on trajectories sampled

Table 2: Classifier-guided posterior sampling with pretrained unconditional diffusion priors. We report the mean$_{\pm \text{std}}$ of each metric computed across all relevant classes for each experiment set, and highlight ±5% from highest/lower experimental value. The FID is computed between learned posterior samples and the true samples from the class in question. DP and LGD-MC fail to appropriately model the posterior distribution (high average $\log r(\mathbf{x})$) while DDPO mode-collapses. RTB achieves comparable or superior performance to all other baselines, optimally balancing high reward and diversity as measured by FID. See Table E.1 for conditional variants.

Dataset →	MNIST			MNIST even/odd			CIFAR-10		
Algorithm ↓ Metric →	$\mathbb{E}[\log r(\mathbf{x})]$ (↑)	FID (↓)	Diversity (↑)	$\mathbb{E}[\log r(\mathbf{x})]$ (↑)	FID (↓)	Diversity (↑)	$\mathbb{E}[\log r(\mathbf{x})]$ (↑)	FID (↓)	Diversity (↑)
DPS	$-2.1597_{\pm 0.423}$	$1.2913_{\pm 0.410}$	$0.1609_{\pm 0.000}$	$-1.2270_{\pm 0.202}$	$1.1498_{\pm 0.182}$	$0.1713_{\pm 0.000}$	$-3.6025_{\pm 0.503}$	$0.7371_{\pm 0.216}$	$0.2738_{\pm 0.000}$
LGD−MC	$-2.1389_{\pm 0.480}$	$1.2873_{\pm 0.412}$	$0.1600_{\pm 0.000}$	$-1.1720_{\pm 0.199}$	$1.1445_{\pm 0.184}$	$0.1600_{\pm 0.000}$	$-3.0988_{\pm 0.359}$	$0.7402_{\pm 0.214}$	$0.2743_{\pm 0.000}$
DDPO	$-1.5_{\pm 4.7} \times 10^{-3}$	$1.5822_{\pm 0.583}$	$0.1350_{\pm 0.005}$	$-8.6_{\pm 12.3} \times 10^{-11}$	$1.8024_{\pm 0.423}$	$0.1314_{\pm 0.002}$	$-2.7_{\pm 8.5} \times 10^{-4}$	$1.7686_{\pm 0.589}$	$0.1575_{\pm 0.015}$
DPOK	$-0.1379_{\pm 0.225}$	$1.2063_{\pm 0.316}$	$0.1442_{\pm 0.004}$	$-0.0783_{\pm 0.082}$	$1.2536_{\pm 0.206}$	$0.1631_{\pm 0.007}$	$-2.4414_{\pm 3.266}$	$0.5316_{\pm 0.157}$	$0.2415_{\pm 0.024}$
RTB (ours)	$-0.1734_{\pm 0.194}$	$1.1823_{\pm 0.288}$	$0.1474_{\pm 0.003}$	$-0.1816_{\pm 0.175}$	$1.1794_{\pm 0.171}$	$0.1679_{\pm 0.004}$	$-2.1625_{\pm 0.879}$	$0.4717_{\pm 0.138}$	$0.2440_{\pm 0.011}$

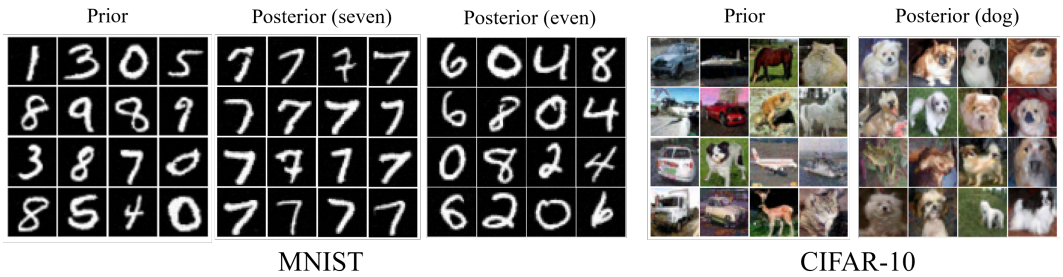


Figure 2: Samples from RTB fine-tuned diffusion posteriors.

on-policy from the current posterior model. We compare RTB with two RL-based fine-tuning techniques derived from DPOK [16] and DDPO [8] and with two classifier guidance baselines, namely DPS [11], and LGD-MC [70]. We consider three experimental settings: MNIST single-digit posterior (learning to sample images of each digit class c), CIFAR-10 single-class posterior (analogous to the previous), and MNIST multi-digit posterior. The latter is a multimodal posterior, for which we set $r(\mathbf{x}) = \max_{i \in \{0,2,4,6,8\}} p(c = i \mid \mathbf{x})$ to generate even digits, and similarly for odd digits.

Results. Samples from the RTB-fine-tuned posterior models are shown in Fig. 2. In Table 2 we report mean$_{\pm \text{std}}$ of various metrics across all trained posteriors. We observe that models fine-tuned with RTB generate class samples with both the highest diversity (highest mean pairwise cosine distance in Inceptionv3 feature space) and closeness to true samples of the target classes (FID), while achieving high expected $\log r(\mathbf{x})$. Pure RL fine-tuning (no KL regularization) displays mode collapse characteristics, achieving high rewards in exchange for significantly poorer diversity and FID scores (see also Fig. E.1). Classifier-guidance-based methods, like DP and LGD-MC, exhibit high diversity, but fail to appropriately model the posterior distribution (lowest $\log r(\mathbf{x})$). Additional results can be found in §E.2.

3.2 Fine-tuning a text-to-image diffusion model

Diffusion models for text-conditional image generation [*e.g.* 63] can struggle to consistently generate images $\mathbf{x}$ that adhere to complex prompts $\mathbf{z}$, for example, those that involve composing multiple objects (*e.g.*, "A cat and a dog") or specify "unnatural" appearances (*e.g.*, "A green-colored rabbit"). Fine-tuning pretrained text-to-image diffusion models $p_\theta(\mathbf{x}_1 \mid \mathbf{z})$ as RL policies to maximize some reward $r(\mathbf{x}_1, \mathbf{z})$ based on human preferences has become the standard approach to tackle this issue [8, 16, 77]. Simply maximizing the reward function can result in mode collapse as well as over-optimization of the reward. This is typically handled by constraining the fine-tuned model $\tilde{p}$ to be close to the prior p:

$$\operatorname*{argmax}_{\tilde{p}} \mathbb{E}_{\tilde{p}(\mathbf{x}_1 \mid \mathbf{z})}[r(\mathbf{x}_1, \mathbf{z})], \quad D_{\mathrm{KL}}[\tilde{p}(\mathbf{x}_1 \mid \mathbf{z}) \parallel p(\mathbf{x}_1 \mid \mathbf{z})] \leq \epsilon. \tag{12}$$

The optimal $\tilde{p}$ for (12) is $\tilde{p}(\mathbf{x}_1 \mid \mathbf{z}) \propto p(\mathbf{x}_1 \mid \mathbf{z}) \exp(\beta r(\mathbf{x}_1, \mathbf{z}))$ for some inverse temperature β. The marginal KL is intractable for diffusion models, so methods like DPOK [16] optimize an upper bound on the marginal KL in the form of a per-step KL penalty $-\gamma \sum_{i=1}^{T} D_{\mathrm{KL}}[\tilde{p}(\mathbf{x}_{i\Delta t} \mid \mathbf{x}_{(i-1)\Delta t}, \mathbf{z}) || p(\mathbf{x}_{i\Delta t} \mid \mathbf{x}_{(i-1)\Delta t}, \mathbf{z})]$ added to the reward. By contrast, RTB can avoid the bias in such an approximation and directly learn to generate unbiased samples from the posterior $\tilde{p}(\mathbf{x}_1 \mid \mathbf{z})$.

Setup. We demonstrate how RTB can be used to fine-tune pretrained text-to-image diffusion models. We use the latent diffusion model Stable Diffusion v1-5 [63] as a prior over 512×512

(a) Four roses. (b) A green rabbit. (c) A cat and a dog. (d) Cybernetic man.

Figure 4: Images generated from prior (top row), DPOK (middle row) and RTB (bottom row) for 4 different prompts. Images in the same column share the random DDIM seed. More images in §H.2.

images. Following DPOK [16], we use ImageReward [87], which has been trained to match human preferences as well as prompt accuracy to attributes such as the number of objects, color, and compositionality, as the reward $\log r(\mathbf{x}_1, \mathbf{z})$. As reference, we present comparisons against DPOK with the default KL regularization $\gamma = 0.01$ and DPOK with $\gamma = 0.0$, which is equivalent to DDPO [8]. We measure the final average reward and the diversity of the generated image, as measured by the average pairwise cosine distance between CLIP embeddings [58] of a batch of generated images. Further details about the experimental setup and ablations are discussed in §H.

Results. Fig. 3 plots the diversity versus log reward on a set of prompts from [16, 87]. In terms of average $\log r(\mathbf{x}_1, \mathbf{z})$, RTB either matches or outperforms DPOK, while generally achieving lower reward than DDPO. The CLIP diversity score for RTB and DPOK are on average higher than DDPO, which is expected since it does not use KL regularization. For qualitative image assessments, refer to Fig. 4 and §H.2. Through this experiment, we show that RTB scales well to high dimensional, multimodal data, matching state-of-the-art methods for fine-tuning text-to-image diffusion models.

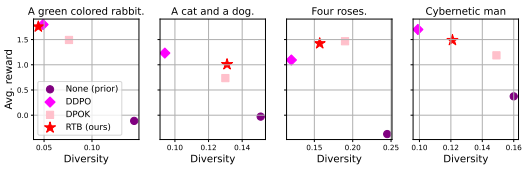


Figure 3: Fine-tuning Stable Diffusion with ImageReward. We report mean $\log r(\mathbf{x}_1, \mathbf{z})$ and diversity, measured as the mean cosine distance between CLIP embeddings for a batch of 100 generated images.[2]

3.3 Text infilling with discrete diffusion language models

To evaluate our approach on discrete diffusion models, we consider the problem of text infilling [91], which involves filling in missing tokens given some context tokens. While discrete diffusion models – unlike their continuous counterparts – can be challenging to train [4, 9, 49, 74], *score entropy discrete diffusion* [SEDD; 43] matches the language modeling performance of autoregressive language models of similar scale. Non-autoregressive generation in diffusion language models can provide useful inductive biases for infilling, such as the ability to attend to context on both sides of a target token.

Setup. We use the ROCStories corpus [50], a dataset of short stories containing 5 sentences each. We adopt the task setup from [29], where the first 3 sentences of a story $\mathbf{x}$ and the last sentence $\mathbf{y}$ are given, and the goal is to generate the fourth sentence $\mathbf{z}$ such that the overall story is coherent and consistent. The fourth sentence can involve a turning point in the story and is thus challenging to fill in. We aim to model the posterior $p^{\text{post}}(\mathbf{z} \mid \mathbf{x}, \mathbf{y}) \propto p(\mathbf{z} \mid \mathbf{x}) p_{\text{reward}}(\mathbf{y} \mid \mathbf{x}, \mathbf{z})$ where p is a SEDD language model prior (a conditional model over $\mathbf{z}$ given $\mathbf{x}$) and p_{reward} is an autoregressive language model fine-tuned with a maximum likelihood objective on a held-out subset of the dataset. As baselines, we consider simply prompting the diffusion language model with $\mathbf{x}$ (Prompt ($\mathbf{x}$)) and $\mathbf{x}, \mathbf{y}$ (Prompt ($\mathbf{x}, \mathbf{y}$)). Additionally, to contextualize the performance, we also consider autoregressive language model baselines from [29], which studied this problem under an autoregressive prior $p(\mathbf{z} \mid \mathbf{x})$. SFT is trained on $50,000$ examples compared to 1000 for RTB, and serves as an upper bound on the performance in this task. See §F for further details about the experimental setup.

[2]Full prompt for "Cybernetic man": "A half - masked rugged laboratory engineer man with cybernetic enhancements as seen from a distance, scifi character portrait by greg rutkowski, esuthio, craig mullins."

Results. Following [29], we use three standard metrics to measure the similarity of the generated infills with the reference infills from the dataset: BERTScore [90] (with DeBERTa [26]), BLEU-4 [54], and GLEU-4 [86]. Table 3 summarizes the results. We observe that the diffusion language model performs significantly better than the autoregressive language model without any fine-tuning. RTB further improves the performance over prompting, and even outperforms the strongest autoregressive baseline of GFlowNet fine-tuning. We provide some examples of generated text in §F.

Table 3: Results on the story infilling task with autoregressive and discrete diffusion language models. Metrics are computed with respect to reference infills from the dataset. All metrics are mean$_{\pm \text{std}}$ over 5 samples for each of the 100 test examples. RTB with discrete diffusion prior performs better than best baseline with autoregressive prior.

Model	Algorithm ↓ Metric →	BLEU-4	GLEU-4	BERTScore
Autoreg.	Prompting	$0.010_{\pm 0.002}$	$0.022_{\pm 0.001}$	$0.005_{\pm 0.001}$
	Supervised fine-tuning	$0.012_{\pm 0.001}$	$0.023_{\pm 0.001}$	$0.013_{\pm 0.002}$
	GFN fine-tuning [29]	$0.019_{\pm 0.001}$	$0.031_{\pm 0.002}$	$0.102_{\pm 0.005}$
Discrete diffusion	Prompt (**x**)	$0.011_{\pm 0.002}$	$0.023_{\pm 0.002}$	$0.014_{\pm 0.003}$
	Prompt (**x**, **y**)	$0.014_{\pm 0.003}$	$0.027_{\pm 0.003}$	$0.092_{\pm 0.004}$
	RTB (ours)	$0.025_{\pm 0.002}$	$0.045_{\pm 0.002}$	$0.156_{\pm 0.003}$
	SFT (upper bound)	$0.031_{\pm 0.002}$	$0.057_{\pm 0.004}$	$0.182_{\pm 0.005}$

3.4 KL-constrained policy search in offline reinforcement learning

The goal of RL algorithms is to learn a policy $\pi(a \mid s)$, *i.e.*, a mapping from states s to actions a in an environment, that maximizes the expected cumulative discounted reward [75]. In the offline RL setting [39], the agent has access to a dataset $\mathcal{D} = \{(s_t^i, a_t^i, s_{t+1}^i, r_t^i)\}_{i=1}^N$ of transitions (where each sample (s_t, a_t, s_{t+1}, r_t) indicates that an agent taking action a_t at state s_t transitioned to the next state s_{t+1} and received reward r_t). This dataset is assumed to be generated by a *behavior policy* $\mu(a \mid s)$, which may be a diffusion model trained on $\mathcal{D}$. Offline RL algorithms must learn a new policy π which achieves high return using only this dataset without interacting with the environment.

An important problem in offline RL is policy extraction from trained Q-functions [55, 25, 44]. For reliable extrapolation, one wants the policy to predict actions that have high Q-values, but also have high density under the behavior policy μ, as naive maximization can result in choosing actions with low probability under μ and thus unreliable predictions from the Q-function. This is formulated as a KL-constrained policy search problem:

$$\operatorname*{argmax}_{\pi} \mathbb{E}_{s\sim d_\mu, a\sim\pi(a|s)}[Q(s,a)], \quad \mathbb{E}_{s\sim d_\mu}[D_{\mathrm{KL}}(\pi(a \mid s) \parallel \mu(a \mid s))] \leq \epsilon, \tag{13}$$

where d_μ is the distribution over states induced by following the policy μ. The optimal policy π in (13) is the product distribution $\pi^*(a \mid s) \propto \mu(a \mid s)\exp(\beta Q(s,a))$ for some inverse temperature β. If $\mu(a \mid s)$ is a conditional diffusion model over continuous actions a conditioned on state s, we use RTB to fine-tune a diffusion behavior policy to sample from π^*, using μ as the prior and $\exp(\beta Q(s,a))$ as the target constraint. We use a Q-function trained using IQL [36].

Setup. We test on continuous control tasks in the D4RL suite [18], which consists of offline datasets collected using a mixture of SAC policies of varying performance. We evaluate on the halfcheetah, hopper and walker2d MuJoCo [76] locomotion tasks, each of which contains three datasets of transitions: "medium" (collected from an early-stopped policy), "medium-expert" (collected from both an expert and an early-stopped policy) and "medium-replay" (transitions stored in the replay buffer prior to early stopping). We compare against standard offline RL baselines (Behavior Cloning (BC), CQL [37], and IQL [36]) and diffusion-based offline RL methods which are currently state-of-the-art: Diffuser [D; 30], Decision Diffuser [DD; 2], D-QL [83], IDQL [25], and QGPO [44]. For algorithm implementation details, hyperparameters, and a report of baselines, see §G.

Results. Table 4 shows that RTB matches state-of-the-art results across the D4RL tasks. In particular, RTB performs strongly in the medium-replay tasks, which contain the most suboptimal data and consequently the poorest behavior prior. We highlight that our performance is similar to QGPO [44], which learns intermediate energy densities for diffusion posterior sampling.

4 Other related work

Composing iterative generative processes. Beyond the approximate posterior sampling algorithms and application-specific techniques discussed in §1 and §3, several recent works have explored the use of hierarchical models, such as diffusion models, as modular components in generative processes. Diffusion models can be used to sample product distributions to induce compositional structure in images [41, 15]. Amortized Bayesian inference [35, 60, 59, 20] is another domain of sampling from product distributions where diffusion models are now being used [21]. Beyond product models, [19]

Table 4: Average rewards of trained policies on D4RL locomotion tasks (mean$_{\pm \text{std}}$ over 5 random seeds). Following past work, numbers within 5% of maximum in every row are highlighted.

Task ↓ Algorithm →	BC	CQL	IQL	D	DD	D-QL	IDQL	QGPO	**RTB (ours)**
halfcheetah-medium-expert	55.2	91.6	86.7	79.8	90.6$_{\pm 1.3}$	96.1$_{\pm 0.3}$	95.9	93.5$_{\pm 0.3}$	74.93$_{\pm 1.72}$
hopper-medium-expert	52.5	105.4	91.5	107.2	111.8$_{\pm 1.8}$	110.7$_{\pm 1.3}$	108.6	108.0$_{\pm 2.5}$	96.71$_{\pm 3.53}$
walker2d-medium-expert	107.5	108.8	109.6	108.4	108.8$_{\pm 1.7}$	109.7$_{\pm 0.3}$	112.7	110.7$_{\pm 0.6}$	109.52$_{\pm 0.11}$
halfcheetah-medium	42.6	44.0	47.4	44.2	49.1$_{\pm 1.0}$	50.6$_{\pm 0.5}$	51.0	54.1$_{\pm 0.4}$	53.70$_{\pm 0.33}$
hopper-medium	52.9	58.5	66.3	58.5	79.3$_{\pm 3.6}$	82.4$_{\pm 4.6}$	65.4	98.0$_{\pm 2.6}$	82.76$_{\pm 7.07}$
walker2d-medium	75.3	72.5	78.3	79.7	82.5$_{\pm 1.4}$	85.1$_{\pm 0.9}$	82.5	86.0$_{\pm 0.7}$	87.29$_{\pm 3.15}$
halfcheetah-medium-replay	36.6	45.5	44.2	42.2	39.3$_{\pm 4.1}$	47.5$_{\pm 0.3}$	45.8	47.6$_{\pm 1.4}$	48.11$_{\pm 0.56}$
hopper-medium-replay	18.1	95.0	94.7	96.8	100.0$_{\pm 0.7}$	100.7$_{\pm 0.6}$	92.1	96.9$_{\pm 2.6}$	100.40$_{\pm 0.21}$
walker2d-medium-replay	26.0	77.2	73.9	61.2	75.0$_{\pm 4.3}$	94.3$_{\pm 1.5}$	85.1	84.4$_{\pm 4.1}$	93.57$_{\pm 2.63}$

studies ways to amortize other kinds of compositions of hierarchical processes, including diffusion models, while [67] proposes methods to sample the product of many iterative processes in application to federated learning. Finally, models without hierarchical structure, such as normalizing flows, have been used to amortize intractable inference in pretrained diffusion models [*e.g.*, 17]. In contrast, our method performs posterior inference by *fine-tuning* a prior model, developing a direction on flexible extraction of information from large pretrained models [29].

Diffusion samplers. Several prior works seek to amortize MCMC sampling from unnormalized densities by training diffusion models for efficient mode-mixing [6, 89, 78, 62, 79, 3]. Our work is most closely related to continuous GFlowNets [38], which offer an alternative perspective on training diffusion samplers using off-policy flow consistency objectives [38, 88, 65]. Recently, Berner et al. [7] have shown connections among existing families of diffusion sampling algorithms and analyzed their continuous-time limits.

5 Conclusions and future work

Relative trajectory balance provides a new approach to training diffusion models to generate unbiased posterior samples given a diffusion prior and an arbitrary reward function. Through experiments on a variety of domains – vision, language, continuous control – we demonstrated the flexibility and general applicability of RTB. RTB can be optimized with off-policy trajectories, and future work can explore ways to leverage off-policy training, using techniques such as local search [34, 65] to improve sample efficiency and mode coverage. Simulation-based objectives in the style of [89] are also applicable to the amortized sampling problems we consider and should be explored, as should simulation-free extensions, *e.g.*, through objectives that are local in time [45]. The ability to handle arbitrary black-box likelihoods also makes RTB a useful candidate for inverse problems in domains such as 3D object synthesis with likelihood computed via a renderer [*e.g.*, 56, 82], imaging problems in astronomy [*e.g.*, 1], medical imaging [*e.g.*, 73], and molecular structure prediction [*e.g.*, 84].

Moreover, RTB could facilitate a breakthrough in modeling molecular dynamics—a notoriously challenging task due to the need to sample rare-event trajectories in chemical simulations—by converting these problems into posterior inference over amplified distributions of rare-event samples. Notably, Seong et al. [66] have already explored a preliminary version of this concept by employing TB with a reward multiplied by the prior likelihood, which is effectively equivalent to RTB.

Limitations. RTB learns the posterior through simulation-based training, which can be slow and memory-intensive. Additionally, the RTB objective is computed on complete trajectories without any local credit-assignment signal, which can result in high variance in the gradients. Guarantees on the error incurred by imperfect fit of the prior model, amortization, and time discretization (analogous to [7]'s analysis for diffusion samplers) have not been obtained and should be considered in future work.

Broader impact. While our contributions focus on an algorithmic approach for learning posterior samplers with diffusion priors, we acknowledge that like other advances in generative modelling, our approach can potentially be used by nefarious actors to train generative models to produce harmful content and misinformation. At the same time, our approach can be also be used to mitigate biases captured in pretrained models and applied to various scientific problems.

Acknowledgments and Disclosure of Funding

The authors thank Adam Coogan, Yashar Hezaveh, Guillaume Lajoie, and Laurence Perreault Levasseur for helpful suggestions in the course of this project and Mandana Samiei for comments on a draft of the paper.

The authors acknowledge funding from CIFAR, NSERC, IVADO, UNIQUE, FACS Acuité, NRC AI4Discovery, Samsung, and Recursion.

The research was enabled in part by computational resources provided by the Digital Research Alliance of Canada (https://alliancecan.ca), Mila (https://mila.quebec), and NVIDIA.

References

[1] Alexandre Adam, Adam Coogan, Nikolay Malkin, Ronan Legin, Laurence Perreault-Levasseur, Yashar Hezaveh, and Yoshua Bengio. Posterior samples of source galaxies in strong gravitational lenses with score-based priors. *arXiv preprint arXiv:2211.03812*, 2022.

[2] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? *International Conference on Learning Representations (ICLR)*, 2023.

[3] Tara Akhound-Sadegh, Jarrid Rector-Brooks, Avishek Joey Bose, Sarthak Mittal, Pablo Lemos, Cheng-Hao Liu, Marcin Sendera, Siamak Ravanbakhsh, Gauthier Gidel, Yoshua Bengio, Nikolay Malkin, and Alexander Tong. Iterated denoising energy matching for sampling from Boltzmann densities. *International Conference on Machine Learning (ICML)*, 2024.

[4] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Neural Information Processing Systems (NeurIPS)*, 2021.

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

[6] Julius Berner, Lorenz Richter, and Karen Ullrich. An optimal control perspective on diffusion-based generative modeling. *Transactions on Machine Learning Research (TMLR)*, 2024.

[7] Julius Berner, Lorenz Richter, Marcin Sendera, Jarrid Rector-Brooks, and Nikolay Malkin. From discrete-time policies to continuous-time diffusion samplers: Asymptotic equivalences and faster training. *arXiv preprint arXiv:2501.06148*, 2025.

[8] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2024.

[9] Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Neural Information Processing Systems (NeurIPS)*, 2022.

[10] Gabriel Cardoso, Yazid Janati el idrissi, Sylvain Le Corff, and Eric Moulines. Monte carlo guided denoising diffusion models for bayesian linear inverse problems. *International Conference on Learning Representations (ICLR)*, 2024.

[11] Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *International Conference on Learning Representations (ICLR)*, 2023.

[12] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. *Neural Information Processing Systems (NeurIPS)*, 2021.

[13] Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

[14] Zehao Dou and Yang Song. Diffusion posterior sampling for linear inverse problem solving: A filtering perspective. *International Conference on Learning Representations (ICLR)*, 2024.

[15] 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. *International Conference on Machine Learning (ICML)*, 2023.

[16] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Reinforcement learning for fine-tuning text-to-image diffusion models. *Neural Information Processing Systems (NeurIPS)*, 2023.

[17] Berthy T. Feng, Jamie Smith, Michael Rubinstein, Huiwen Chang, Katherine L. Bouman, and William T. Freeman. Score-based diffusion models as principled priors for inverse imaging. *International Conference on Computer Vision (ICCV)*, 2023.

[18] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[19] Timur Garipov, Sebastiaan De Peuter, Ge Yang, Vikas Garg, Samuel Kaski, and Tommi Jaakkola. Compositional sculpting of iterative generative processes. *Neural Information Processing Systems (NeurIPS)*, 2023.

[20] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

[21] Tomas Geffner, George Papamakarios, and Andriy Mnih. Compositional score modeling for simulation-based inference. *International Conference on Machine Learning (ICML)*, 2023.

[22] Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. Diffusion models as plug-and-play priors. *Neural Information Processing Systems (NeurIPS)*, 2022.

[23] Ishaan Gulrajani and Tatsunori B Hashimoto. Likelihood-based diffusion language models. *Neural Information Processing Systems (NeurIPS)*, 2023.

[24] Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. SSD-LM: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *Association for Computational Linguistics (ACL)*, 2023.

[25] 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.

[26] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *International Conference on Learning Representations (ICLR)*, 2021.

[27] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Neural Information Processing Systems (NeurIPS)*, 2020.

[28] Edward J. Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *International Conference on Learning Representations (ICLR)*, 2022.

[29] Edward J. Hu, Moksh Jain, Eric Elmoznino, Younesse Kaddar, Guillaume Lajoie, Yoshua Bengio, and Nikolay Malkin. Amortizing intractable inference in large language models. *International Conference on Learning Representations (ICLR)*, 2024.

[30] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *International Conference on Machine Learning (ICML)*, 2022.

[31] Zahra Kadkhodaie and Eero P. Simoncelli. Solving linear inverse problems using the prior implicit in a denoiser. *Neural Information Processing Systems (NeurIPS)*, 2021.

[32] Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Neural Information Processing Systems (NeurIPS)*, 2024.

[33] Bahjat Kawar, Gregory Vaksman, and Michael Elad. SNIPS: Solving noisy inverse problems stochastically. *Neural Information Processing Systems (NeurIPS)*, 2021.

[34] Minsu Kim, Taeyoung Yun, Emmanuel Bengio, Dinghuai Zhang, Yoshua Bengio, Sungsoo Ahn, and Jinkyoo Park. Local search GFlowNets. *International Conference on Learning Representations (ICLR)*, 2024.

[35] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. *International Conference on Learning Representations (ICLR)*, 2014.

[36] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. *International Conference on Learning Representations (ICLR)*, 2022.

[37] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. *Neural Information Processing Systems (NeurIPS)*, 2020.

[38] Salem Lahlou, Tristan Deleu, Pablo Lemos, Dinghuai Zhang, Alexandra Volokhova, Alex Hernández-Garcıa, Léna Néhale Ezzine, Yoshua Bengio, and Nikolay Malkin. A theory of continuous generative flow networks. *International Conference on Machine Learning (ICML)*, 2023.

[39] Sergey Levine, Aviral Kumar, G. Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[40] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-LM improves controllable text generation. *Neural Information Processing Systems (NeurIPS)*, 2022.

[41] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. *European Conference on Computer Vision (ECCV)*, 2022.

[42] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. *arXiv preprint arXiv:2303.16634*, 2023.

[43] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

[44] Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. *International Conference on Machine Learning (ICML)*, 2023.

[45] Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning GFlowNets from partial episodes for improved convergence and stability. *International Conference on Machine Learning (ICML)*, 2022.

[46] Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in GFlowNets. *Neural Information Processing Systems (NeurIPS)*, 2022.

[47] Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward Hu, Katie Everett, Dinghuai Zhang, and Yoshua Bengio. GFlowNets and variational inference. *International Conference on Learning Representations (ICLR)*, 2023.

[48] Morteza Mardani, Jiaming Song, Jan Kautz, and Arash Vahdat. A variational perspective on solving inverse problems with diffusion models. *International Conference on Learning Representations (ICLR)*, 2024.

[49] Chenlin Meng, Kristy Choi, Jiaming Song, and Stefano Ermon. Concrete score matching: Generalized score matching for discrete data. *Neural Information Processing Systems (NeurIPS)*, 2022.

[50] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2016.

[51] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabili1stic models. *International Conference on Machine Learning (ICML)*, 2021.

[52] Nikolas Nüsken and Lorenz Richter. Solving high-dimensional Hamilton–Jacobi–Bellman PDEs using neural networks: perspectives from the theory of controlled diffusions and measures on path space. *Partial Differential Equations and Applications*, 2(4):48, 2021.

[53] Bernt Øksendal. *Stochastic Differential Equations: An Introduction with Applications*. Springer, 2003.

[54] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. *Association for Computational Linguistics (ACL)*, 2002.

[55] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv prepring arXiv:1910.00177*, 2019.

[56] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. *International Conference on Learning Representations (ICLR)*, 2023.

[57] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[58] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. 2021.

[59] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. *International Conference on Machine Learning (ICML)*, 2015.

[60] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *International Conference on Machine Learning (ICML)*, 2014.

[61] Lorenz Richter, Ayman Boustati, Nikolas Nüsken, Francisco J. R. Ruiz, and Ömer Deniz Akyildiz. VarGrad: A low-variance gradient estimator for variational inference. *Neural Information Processing Systems (NeurIPS)*, 2020.

[62] Lorenz Richter, Julius Berner, and Guan-Horng Liu. Improved sampling via learned diffusions. *International Conference on Learning Representations (ICLR)*, 2023.

[63] Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[64] Simo Särkkä and Arno Solin. *Applied stochastic differential equations*. Cambridge University Press, 2019.

[65] Marcin Sendera, Minsu Kim, Sarthak Mittal, Pablo Lemos, Luca Scimeca, Jarrid Rector-Brooks, Alexandre Adam, Yoshua Bengio, and Nikolay Malkin. On diffusion models for amortized inference: Benchmarking and improving stochastic control and sampling. *arXiv preprint arXiv:2402.05098*, 2024.

[66] Kiyoung Seong, Seonghyun Park, Seonghwan Kim, Woo Youn Kim, and Sungsoo Ahn. Collective variable free transition path sampling with generative flow network. *arXiv preprint arXiv:2405.19961*, 2024.

[67] Tiago Silva, Amauri H Souza, Luiz Max Carvalho, Samuel Kaski, and Diego Mesquita. Federated contrastive GFlowNets, 2024. URL https://openreview.net/forum?id=VJDFhkwQg6.

[68] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *International Conference on Machine Learning (ICML)*, 2015.

[69] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *International Conference on Learning Representations (ICLR)*, 2021.

[70] Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. Loss-guided diffusion models for plug-and-play controllable generation. *International Conference on Maching Learning (ICML)*, 2023.

[71] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Neural Information Processing Systems (NeurIPS)*, 2021.

[72] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations (ICLR)*, 2021.

[73] Yang Song, Liyue Shen, Lei Xing, and Stefano Ermon. Solving inverse problems in medical imaging with score-based generative models. *International Conference on Learning Representations (ICLR)*, 2022.

[74] Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. *International Conference on Learning Representations (ICLR)*, 2023.

[75] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

[76] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

[77] Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M Tseng, Tommaso Biancalani, and Sergey Levine. Fine-tuning of continuous-time diffusion models as entropy-regularized control. *arXiv preprint arXiv:2402.15194*, 2024.

[78] Francisco Vargas, Will Grathwohl, and Arnaud Doucet. Denoising diffusion samplers. *International Conference on Learning Representations (ICLR)*, 2023.

[79] Francisco Vargas, Shreyas Padhy, Denis Blessing, and Nikolas Nüsken. Transport meets variational inference: Controlled Monte Carlo diffusions. *International Conference on Learning Representations (ICLR)*, 2024.

[80] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

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

[82] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score Jacobian chaining: Lifting pretrained 2D diffusion models for 3D generation. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[83] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2023.

[84] Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

[85] Sean Welleck, Ilia Kulikov, Jaedeok Kim, Richard Yuanzhe Pang, and Kyunghyun Cho. Consistency of a recurrent language model with respect to incomplete decoding. *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[86] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[87] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Neural Information Processing Systems (NeurIPS)*, 2023.

[88] Dinghuai Zhang, Ricky Tian Qi Chen, Cheng-Hao Liu, Aaron Courville, and Yoshua Bengio. Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. *International Conference on Learning Representations (ICLR)*, 2024.

[89] Qinsheng Zhang and Yongxin Chen. Path integral sampler: a stochastic control approach for sampling. *International Conference on Learning Representations (ICLR)*, 2022.

[90] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating text generation with bert. *International Conference on Learning Representations*, 2020.

[91] Wanrong Zhu, Zhiting Hu, and Eric Xing. Text infilling. *arXiv preprint arXiv:1901.00158*, 2019.

A Proofs

Proposition 1 (Relative TB constraint)**.** *If p_θ, p_ϕ^{post}, and the scalar Z_ϕ jointly satisfy the* relative trajectory balance (RTB) constraint

$$Z_\phi \cdot p_\phi^{\text{post}}(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1) = r(\mathbf{x}_1) p_\theta(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1) \tag{8}$$

for every denoising trajectory $\mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \cdots \to \mathbf{x}_1$, then $p_\phi^{\text{post}}(\mathbf{x}_1) \propto p_\theta(\mathbf{x}_1) r(\mathbf{x}_1)$, i.e., the diffusion model p_ϕ^{post} samples the posterior distribution. Furthermore, if p_θ also satisfies the TB constraint (7) with respect to the noising process q and some target density $q(\mathbf{x}_1)$, then p_ϕ^{post} satisfies the TB constraint with respect to the target density $q^{\text{post}}(\mathbf{x}_1) \propto q(\mathbf{x}_1) r(\mathbf{x}_1)$, and $Z = \int q(\mathbf{x}_1) r(\mathbf{x}_1)\, d\mathbf{x}_1$.

Proof of Prop. 1. Suppose that p_θ, p_ϕ^{post}, and Z jointly satisfy (8). Then necessarily $Z \neq 0$, since the quantities on the right side are positive. We then have, using (4),

$$\begin{aligned}
p_\phi^{\text{post}}(\mathbf{x}_1) &= \int p_\phi^{\text{post}}(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1)\, d\mathbf{x}_0\, d\mathbf{x}_{\Delta t} \ldots d\mathbf{x}_{1-\Delta t} \\
&= \frac{1}{Z} r(\mathbf{x}_1) \int p_\theta(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1)\, d\mathbf{x}_0\, d\mathbf{x}_{\Delta t} \ldots d\mathbf{x}_{1-\Delta t} \\
&= \frac{1}{Z} r(\mathbf{x}_1) p_\theta(\mathbf{x}_1) && \propto p_\theta(\mathbf{x}_1) r(\mathbf{x}_1),
\end{aligned}$$

as desired.

Now suppose that p_θ also satisfies the TB constraint (7) with respect to $q(\mathbf{x}_1)$. Then, for any denoising trajectory,

$$q(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_{1-\Delta t} \mid \mathbf{x}_1) = \frac{p_\theta(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1)}{q(\mathbf{x}_1)} = \frac{p_\phi^{\text{post}}(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1)}{q(\mathbf{x}_1) r(\mathbf{x}_1)/Z}. \tag{14}$$

showing that p_ϕ^{post} satisfies the TB constraint with respect to the noising process q and the (not yet shown to be normalized) density $\frac{1}{Z} q(\mathbf{x}_1) r(\mathbf{x}_1)$. We integrate out the variables $\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_{1-\Delta t}$ in (14), giving

$$\begin{aligned}
1 &= \frac{p_\phi^{\text{post}}(\mathbf{x}_1)}{q(\mathbf{x}_1) r(\mathbf{x}_1)/Z} \\
q(\mathbf{x}_1) r(\mathbf{x}_1) &= Z p_\phi^{\text{post}}(\mathbf{x}_1).
\end{aligned}$$

Integrating over $\mathbf{x}_1$ shows $\int q(\mathbf{x}_1) r(\mathbf{x}_1)\, d\mathbf{x}_1 = Z$. □

B Relative TB as TB under the prior measure

The theoretical foundations for continuous generative flow networks [38] establish the correctness of enforcing constraints such as trajectory balance (7) for training sequential samplers, such as diffusion models, to match unnormalized target densities. While we have considered Gaussian transitions and identified transition kernels with their densities with respect to the Lebesgue measure over $\mathbb{R}^d$, these foundations generalize to more general *reference measures*. In application to diffusion samplers, suppose that $\pi_{\text{ref}}(\mathbf{x}_t)$ is a collection of Lebesgue-absolutely continuous densities over $\mathbb{R}^d$ for $t = 0, \Delta t, \ldots, 1$ and that $\overrightarrow{\pi}_{\text{ref}}(\mathbf{x}_t \mid \mathbf{x}_{t-\Delta t})$, $\overleftarrow{\pi}_{\text{ref}}(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t)$ are collections of Lebesgue-absolutely continuous transition kernels. If these densities jointly satisfy the detailed balance condition $\pi_{\text{ref}}(\mathbf{x}_t) \overleftarrow{\pi}_{\text{ref}}(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t) = \pi_{\text{ref}}(\mathbf{x}_{t-\Delta t}) \overrightarrow{\pi}_{\text{ref}}(\mathbf{x}_t \mid \mathbf{x}_{t-\Delta t})$, then they satisfy the conditions to be reference measures. A main result of [38] is that if a pair of forward and backward processes satisfies the trajectory balance constraint (7) jointly with a reward density r, then the forward process p samples from the distribution with density r, with all densities interpreted as *relative to the reference measures* $\pi_{\text{ref}}, \overleftarrow{\pi}_{\text{ref}}, \overrightarrow{\pi}_{\text{ref}}$.[3]

[3] Recall that the relative density (or Radon-Nikodym derivative) of a distribution with density p under the Lebesgue measure relative to one with density π is simply the ratio of densities p/π.

If p_θ is a diffusion model that satisfies the TB constraint jointly with some reverse process q and target density $q(\mathbf{x}_1)$, then one can take the reference transition kernels $\overrightarrow{\pi}_{\text{ref}}, \overleftarrow{\pi}_{\text{ref}}$ to be p and q, respectively. In this case, the TB constraint for a target density $\frac{1}{Z}r(\mathbf{x}_1)$ and forward transition p_ϕ^{post} is

$$\frac{p_\phi^{\text{post}}(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1)}{\overrightarrow{\pi}_{\text{ref}}(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1)} = \frac{\frac{1}{Z}r(\mathbf{x}_1)\cancel{q(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_{1-\Delta t} \mid \mathbf{x}_1)}}{\cancel{\overleftarrow{\pi}_{\text{ref}}(\mathbf{x}_0, \mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_{1-\Delta t} \mid \mathbf{x}_1)}}, \tag{15}$$

which is identical to the RTB constraint (8). If (15) holds, then p_ϕ^{post} samples from the distribution with density $\frac{1}{Z}r(\mathbf{x}_1)$ relative to $\pi_{\text{ref}}(\mathbf{x}_1)$, which is exactly $\frac{1}{Z}p_\theta(\mathbf{x}_1)r(\mathbf{x}_1)$. We have thus recovered RTB as a case of TB for non-Lebesgue reference measures.

C Posterior inference on two-dimensional Gaussian mixture model

Setup We conduct toy experiments in low-dimensional spaces using samples from a Gaussian mixture model with multiple modes to visually demonstrate its validity. The prior distribution $p(\mathbf{x}_1)$ is trained on a Gaussian mixture model with 25 evenly weighted modes, while the target posterior $p^{\text{post}}(\mathbf{x}_1) = r(\mathbf{x}_1)p(\mathbf{x}_1)$ uses a reward $r(\mathbf{x}_1)$ to select and re-weight 9 modes from $p(\mathbf{x}_1)$. More specifically, the resulting posterior is:

$$p^{\text{post}}(\mathbf{x}_1) = \frac{1}{\sum_j \tilde{\pi}_j} \sum_i \tilde{\pi}_i \mathcal{N}(\mathbf{x}_1 \mid \mu_i, \mathbf{I}) \tag{16}$$

$$\{\mu_i\} = \{(-10,-5), (-5,-10), (-5,0), (10,-5), (0,0), (0,5), (5,-5), (5,0), (5,10)\} \tag{17}$$

$$\{\tilde{\pi}_i\} = \{4, 10, 4, 5, 10, 5, 4, 15, 4\} \tag{18}$$

Our objective is to sample from the posterior $p^{\text{post}}(\mathbf{x}_1)$. We compare our method with several baselines, including policy gradient reinforcement learning (RL) with KL constraint and classifier-guided diffusion models. For RL, we implemented the REINFORCE method with a mean baseline and a KL constraint, following recent work training diffusion models to optimize a reward function [8]. Sampling according to the RL policy leads to a distribution $q_\theta(\mathbf{x}_1)$, which is trained with the objective:

$$J(\theta) = \mathbb{E}_{q_\theta(\mathbf{x}_1)}\left[r(\mathbf{x}_1)\right] + \alpha D_{\text{KL}}\left(q_\theta(\mathbf{x}_1) \| p(\mathbf{x}_1)\right) \tag{19}$$

While the exact computation of $KL\left(q_\theta(\mathbf{x}_1) \| p(\mathbf{x}_1)\right)$ is intractable, we follow the approximation method introduced by Fan et al. [16], which sums the divergence at every diffusion step. This approximation optimizes an upper bound of the marginal KL.

The other baseline is classifier (energy) guidance, which given a diffusion prior, samples using a posterior score function estimate:

$$\nabla_{\mathbf{x}_t} \log p^{\text{post}}(\mathbf{x}_t) \approx \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log r(\mathbf{x}_t) \tag{20}$$

Note that this is a biased approximation of the true intractable score:

$$\nabla_{\mathbf{x}_t} \log p^{\text{post}}(\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log \mathbb{E}_{p(\mathbf{x}_1|\mathbf{x}_t)}\left[r(\mathbf{x}_1)\right] \text{ [44]} \tag{21}$$

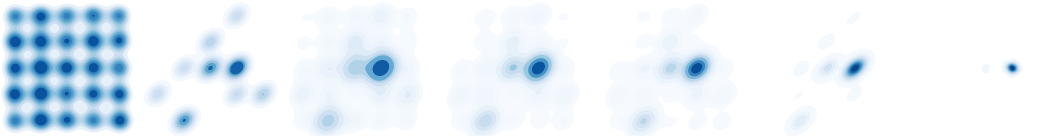
For our experiments, we follow the source code[4] provided in recent diffusion sampler benchmarks [65]. We utilize a batch size of 500, with finetuning at 5,000 training iterations, a learning rate of 0.0001, a diffusion time scale of 5.0, 100 steps, and a log variance range of 4.0. The neural architecture employed is identical to that used in [65]. For pretraining the prior model, we use the same hyperparameters as above, but with 10,000 training iterations using maximum likelihood estimation with true samples.

Results. As we reported in the main text, in Fig. 1, we present illustrative results. The classifier-guided diffusion model shows biased posterior sampling (Fig. 1f), failing to provide accurate inference. RL with a per step KL constraint cannot exactly optimize for the posterior distribution, making the tuning of the KL weight α crucial to achieving desirable output Fig. C.1. RTB asymptotically achieves the true posterior without introducing a balancing hyperparameter α. Another advantage of our approach is off-policy exploration for efficient mode coverage. RL methods for fine-tuning diffusion models (e.g., DPOK [16], DDPO [8]) typically use policy gradient style methods that are on-policy. By using a simple off-policy trick introduced by [47, 38] and demonstrated by Sendera et al. [65], we can introduce randomness into the exploration process in diffusion by adding $\frac{\epsilon^2}{T}$, where ϵ is a noise hyperparameter and T is the diffusion timestep, into the variances and annealing it to zero over training iterations. We set $\epsilon = 0.5$ for off-policy exploration. As shown in Fig. C.2, RTB with off-policy exploration gives very close posterior inferences, whereas off-policy exploration in RL with $\alpha = 0.5$ (which is a carefully selected hyperparameter) does not improve performance due to its on-policy nature.

D Code

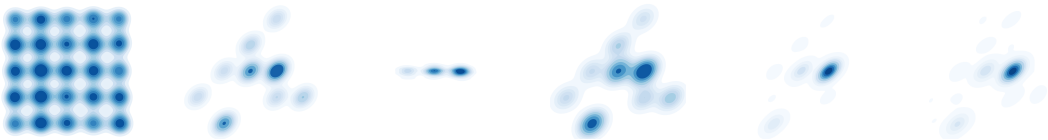
Code for all experiments is available at `https://github.com/GFNOrg/diffusion-finetuning` and will continue to be maintained and extended.

[4] `https://github.com/GFNOrg/gfn-diffusion`



(a) Prior (b) Posterior (c) $\alpha = 1.0$ (d) $\alpha = 0.8$ (e) $\alpha = 0.7$ (f) $\alpha = 0.5$ (g) $\alpha = 0.3$

Figure C.1: Tuning the KL weight α in reinforcement learning: influences the balance between sticking to the prior distribution and moving towards the modes of the reward density. A higher α value maintains closer adherence to the prior, while a lower α allows a gradual shift towards high values of $r(\mathbf{x})$. Setting α below 0.3 tends to cause mode collapse, moving too far from the prior and focusing on maximizing rewards for single modes. $\alpha = 0.5$ gives us samples that closest resembles the posterior.



(a) Prior (b) Posterior (c) RTB (on) (d) RTB (off) (e) RL (on) (f) RL (off)

Figure C.2: Off-policy exploration benefits for RTB training. RTB, with simple off-policy exploration techniques that increase randomness in the diffusion process, significantly improves mode coverage. On the other hand, policy gradient RL methods which are typically used to finetune diffusion models are on-policy, and hence prone to mode collapse.

E On classifier guidance and RTB posterior sampling

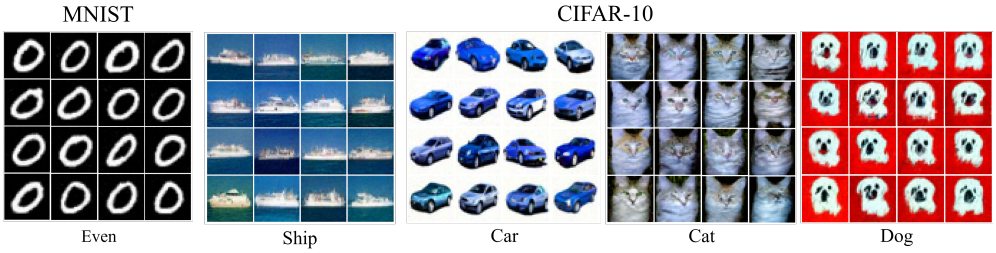


Figure E.1: Samples from a posterior model fine-tuned with RL (no KL). We observe early mode collapse, showcasing high-reward samples with minimal diversity.

E.1 Experimental Details

In our experiments, we fine-tune pretrained unconditional diffusion models with our RTB objective, to sample from a posterior distribution in the form $p^{\text{post}}(x \mid y) = p(y \mid x)p(x)$. In this section, we detail the experimental settings for RTB as well as the compared baselines.

Experiments setting. For MNIST, we pretrain a noise-predicting diffusion model on 28×28 (upscaled to 32×32) single channel images of digits from the MNIST datasets. We discretize the forward and backward processes into 200 steps and train our model until convergence. For CIFAR-10, we use a pretrained model from [27], trained to generate 32×32 3-channel images from the CIFAR-10 dataset, while discretizing the noising/denoising processes into 1000 steps. For fine-tuning the prior, we parametrize the posterior with LoRA weights [28], with the number of parameters equal to about 3% of the prior model's parameter count. We train our models on a single NVIDA V100 GPU.

We compute FID as a similarity score estimate of the *true* posterior distribution from the data. As such, the computation is limited to the total number of per-class-samples present in the data, (between 5k and 6k for CIFAR-10 and MNIST digits, and 30k for the even/odd task).

RTB. For RTB fine-tuning, we finetune a diffusion model following the objective in Equation 9. We impose the objective while sampling denoising paths following a DDPM sampling scheme, with only 20% to 50% of the original trained steps. We employ loss clipping at 0.1, to account for imperfect constraints in the pretrained prior, and train each of our models for 1500 training iterations, well into convergence trends.

RL [16]. We implement two RL-based fine-tuning techniques derived from DPOK [16] and DDPO [8], respectively with and without KL regularization. These implementations use a reinforcement learning baseline similar to the one in our experiments described in §3.2. By following the same sampling scheme as in our RTB experiments, we enable a direct comparison with RTB. To fine-tune the KL weight, we perform a search over $\alpha \in \{0.01, 0.1, 1.0\}$.

DP [11]. We implement and adapt the Gaussian version of the posterior sampling scheme in [11], originally devised for noisy inverse problems. This method relaxes some of our experimental constraints, as it requires a differentiable reward $r(\mathbf{x})$. We perform a sweep over ten values of the suggest parameter range for the step size $\zeta \in [.1, 1.]$ on MNIST single-digit sampling, and choose $\zeta = 0.1$ for our experiments.

LGD-MC [70]. We adapt the implementation of the algorithm in [70] to sample from the classifier-based posteriors in CIFAR-10 and MNIST. Similarly to the DP baseline, we use our pretrained classifier to perform measurements at each sampling step, and use a Monte Carlo estimate of the gradient correction to guide the denoising process. We choose $\zeta = 0.1$ following the DP experiments and default the number of particles to 10 as per the authors' guidelines.

E.2 Additional findings.

Classifier-guidance baselines. We find that the DP and LGD-MC classifier-guidance based baselines struggle to sample from the true posterior distribution in our experimental settings. The baselines achieve the lowers classifier average rewards in all tested settings. Despite choosing $\zeta = 0.1$ as the validate best performing hyperparameter, we also also observe the posterior samples from DP and LGD-MC to be close to the prior. As such, DP and LGD-MC score high in diversity, and low in FID for the Even/Odd experimental scenario, as expected from prior sampling benchmarks, but failing to appropriately model the posterior distribution.

RL and mode collapse. In the pure Reinforcement Learning objective imposed for the experiments in §3.1 (no KL), we observe a significantly higher reward than other baseline methods, while showcasing increased FID and lower diversity. In Fig. E.1 we show a random set of 16 samples for posterior models trained on 4 different classes of the CIFAR-10 datasets, as well as the *Even* objective from the MNIST dataset, after 500 training iterations. In the figure, we observe early mode collapse and reward exploitation, visually evident from the little to no variation amongst samples for each class class, and single-digit collapse in the multi-modal *even* digits objective (see samples in Fig. 2 for comparison with our RTB-finetuned models).

Conditional architectures. We repeat the even/odd posterior sampling experiment of §3.1 in a conditional setting, where the condition is an input to the posterior model. For the posterior architecture, we use a naive modification of the prior with an extra input channel, which is populated a full mask of 0 or 1 for conditioning on the even and odd classes, respectively. The results are shown in Table E.1. We look forward to future work which develops more specialized architectures for handling conditional constraints.

Table E.1: Conditional experiment for MNIST even/odd posterior. Note that the posterior model in the conditional experiment is different from that in the baselines because it uses a different architecture that includes an additional input channel.

Dataset →	MNIST even/odd	
Algorithm ↓ Metric →	$\mathbb{E}[\log r(\mathbf{x})]$ (↑)	FID (↓)
DPS	$-1.2270_{\pm 0.202}$	$1.1498_{\pm 0.182}$
LGD−MC	$-1.1720_{\pm 0.199}$	$1.1445_{\pm 0.184}$
DDPO	$-8.6_{\pm 12.3} \times 10^{-11}$	$1.8024_{\pm 0.423}$
DPOK	$-0.0783_{\pm 0.082}$	$1.2536_{\pm 0.206}$
RTB (unconditional)	$-0.1816_{\pm 0.175}$	$1.1794_{\pm 0.171}$
RTB (conditional)	-0.1236	0.9112

F Infilling with discrete diffusion

Additional details. We illustrate some examples from the ROC Stories dataset used for training in Table F.1. For the prior we use the `sedd-small`[5] model, which uses an absorbing noising process [4] with a log linear noise schedule, as the diffusion prior $p(\mathbf{z} \mid \mathbf{x})$. The posterior model is

[5] https://huggingface.co/louaaron/sedd-small

Table F.1: Examples of training samples for the language infilling task.

Beginning ($\mathbf{x}$)	**Middle** ($\mathbf{z}$)	**End** ($\mathbf{y}$)
I was going to a Halloween party. I looked through my clothes but could not find a costume. I cut up my old clothes and constructed a costume.	I put my costume on and went to the party.	My friends loved my costume.
Allen thought he was a very talented poet. He attended college to study creative writing. In college, he met a boy named Carl.	Carl told him that he wasn't very good.	Because of this, Allen swore off poetry forever.

parameterized as a copy of the prior. To condition the diffusion model on the beginning $\mathbf{x}$ we set the tokens at the appropriate location in the state in the initial time step, *i.e.* $t = 0$. Our implementation is based on the original SEDD codebase [6]. Training this model is computationally expensive (in terms of memory and speed) so we utilize the stochastic TB trick, only propagating the gradients through a subset of the steps of the trajectory. We also use the loss clipping trick as discussed in §2.3. Specifically, we clip the loss below a certain threshold to 0, resulting in updates only when the loss is larger. This threshold – referred to as the loss clipping coefficient – is a hyperparameter. As this is a conditional problem we also use the relative VarGrad objective. We also use some tempering on the reward likelihood which helps in learning (*i.e.*, $p_{\text{reward}}(\mathbf{y} \mid \mathbf{x}, \mathbf{z})^{\beta}$) where β is the inverse temperature parameter. We perform all experiments on an NVIDIA A100-Large GPU. Note that we also tried a baseline of simply fine-tuning the diffusion model on the data but encountered some training instabilities that we could not fix. The hyperparameters used for training RTB in our experiments are detailed in Table F.2.

Reward. For training p_{reward} we follow the training procedure and implementation from [29][7]. Specifically, we fine-tune a `GPT-2 Large` model [57] on the stories dataset with full parameter fine-tuning using the `trl` library [81]. We trained for 20 epochs with a batch size of 64 and 32 gradient accumulation steps and a learning rate of 0.0005.

Baselines. For the baselines, we adopt the implementations from [29]. A critical difference in our experiments compared to [29] is that the posterior model is not initialized with a base model that is fine-tuned on the stories dataset. To condition the model on X and Y, as well as for the prompting baseline, we use the following prompt:

```
"Beginning: {X}\n End: {Y}\n Middle: "
```

During training for the autoregressive GFlowNet fine-tuning, a $(\mathbf{x}, \mathbf{y})$ pair is sampled from the dataset and then sample (batch size) $\mathbf{x}$s for every (X, Y), and $p_{\text{reward}}(XZY)$ is used as the reward. Both the GFlowNet fine-tuning and supervised fine-tuning baseline use LoRA fine-tuning. We use the default hyperparamteres from [29]. At test time, we sample 100 infills for each example in the test set from all the models at temperature 0.9, and average over 5 such draws.

Additional results. Table F.3, Table F.4 and Table F.5 illustrates some examples of the infills generated by the diffusion models. We note that the general quality of the samples is poor, due to a relatively weak prior. At the same time we can observe that the prompting baselines often generate infills that are unrelated to the current story. We also note that the RTB fine-tuned model can sometimes generate repitions as the reward model tends to assign high likelihood to repititions [85]. We also attempted a LLMEval [42] for evaluating the coherence of the stories but did not obtain statistically significant results.

[6] https://github.com/louaaron/Score-Entropy-Discrete-Diffusion
[7] https://github.com/GFNOrg/gfn-lm-tuning

Table F.2: Hyperparameters for the story infilling task.

Batch size	16
Gradient accumulation steps	8
Learning rate	1e-5
Warmup Step	20
Optimizer	AdamW
Reward temperature start	1.2
Reward inverse temperature end	0.9
Reward inverse temperature horizon	5000
Number of training steps	1500
Loss clipping coefficient	0.1
Discretization steps T	15

Table F.3: Examples of infills generated by the posterior trained with RTB along with **reference infills** for the stories infilling task.

Beginning (**x**)	**Middle** (**z**)	**End** (**y**)
David noticed he had put on a lot of weight recently. He examined his habits to try and figure out the reason. He realized he'd been eating too much fast food lately.	**He stopped going to burger places and started a vegetarian diet.** He reviewed his habits to try to figure out how to change He asked he thought try to cut down on the amount amount. He examined his habits to try and figure out the reason. He realized he had been eating too much fast food recently.	After a few weeks, he started to feel much better.
Robbie was competing in a cross country meet. He was halfway through when his leg cramped up. Robbie wasn't sure he could go on.	**He stopped for a minute and stretched his bad leg.** Robbie was sure he could go on. Robbie was sure. He was floating and twisting his leg in half then. His body just caught up with his legs. Robbie was. He held his leg forward as he went through and his	Robbie began to run again and finished the race in second place.

G Offline RL

G.1 Training details

Our method requires first training a diffusion-based behavior policy π_θ and a Q-function Q_ψ. Once π_θ and Q_ψ are trained, The posterior policy π_γ is trained using RTB, with its weights initialized to the trained behavior policy weights θ.

The behavior policy π_θ is parametrized as a state-conditioned noise-predicting denoising diffusion probabilistic model (DDPM) [27] with a linear schedule, and 75 denoising steps. The diffusion model takes as input a state s, a noised action a_t and a noise level t and predicts the source noise ϵ. The state s and noised action a_t are concatenated with Fourier features computed on the noise level t, which are then fed through a 3-layer MLP of hidden dimensionality 256, with layer normalization and a GeLU activation after each hidden layer. The behavior policy is trained using the Adam optimizer with batch size 512 and learning rate 5e-4 for 10000 epochs. The Q-function Q_ψ is trained using IQL. We use the same IQL experimental configurations and training hyperparameters as in [36]. That is, we set $\tau = 0.7$. The architecture for Q_ψ is a 3-layer MLP with hidden dimensionality 256 and ReLU activations, which is trained using the Adam optimizer with a learning rate 3e-4 and batch size 256 for 750000 gradient steps. The task rewards are normalized as in [36] and the target network is updated

Table F.4: Examples of infills generated by Prompt $(\mathbf{x}, \mathbf{y})$ along with **reference infills** for the stories infilling task.

Beginning ($\mathbf{x}$)	**Middle** ($\mathbf{z}$)	**End** ($\mathbf{y}$)
David noticed he had put on a lot of weight recently. He examined his habits to try and figure out the reason. He realized he'd been eating too much fast food lately.	**He stopped going to burger places and started a vegetarian diet.** He'd had less opportunities to eat properly all of last. Doctors made the note of the situation. He was treated. He told him the guy for a mic replacement.\n\n He felt empty for one reason and new fresh, too.	After a few weeks, he started to feel much better.
Robbie was competing in a cross country meet. He was halfway through when his leg cramped up. Robbie wasn't sure he could go on.	**He stopped for a minute and stretched his bad leg.** Robbie wasn't sure Robbie's fuel tank was full. Robbie took a photograph with a close friend.\n\n Only Stacey Ebers and Rand were out there. Robbie got bigger as the position got better.\n\n	Robbie began to run again and finished the race in second place.

Table F.5: Examples of infills generated by Prompt $(\mathbf{x})$ along with **reference infills** for the stories infilling task.

Beginning ($\mathbf{x}$)	**Middle** ($\mathbf{z}$)	**End** ($\mathbf{y}$)
David noticed he had put on a lot of weight recently. He examined his habits to try and figure out the reason. He realized he'd been eating too much fast food lately.	**He stopped going to burger places and started a vegetarian diet.** David, "All I had told eat was a problem. He got the backside what about that and he made the, He made just good of fast food and spliced it down. He explained everything to them, reached them out, the problem.	After a few weeks, he started to feel much better.
Robbie was competing in a cross country meet. He was halfway through when his leg cramped up. Robbie wasn't sure he could go on.	**He stopped for a minute and stretched his bad leg.** Robbie and Robbie was piling. Robbie and Robbie fistfight. I said goodbye. Robbie at dinner. Robbie agreed with. He cut away a little to Robbie's pace fleetingly. He held off all the police and place. Robbie.	Robbie began to run again and finished the race in second place.

Table G.1: Mixed vs. online training on DR4L Tasks. We report mean$_{\pm \text{std}}$ over 5 random seeds.

Task	RTB (Online)	RTB (Mixed)
halfcheetah-medium-replay	46.88$_{\pm 0.51}$	48.11$_{\pm 0.56}$
hopper-medium-replay	99.23$_{\pm 3.22}$	100.40$_{\pm 0.21}$
walker2d-medium-replay	94.01$_{\pm 0.28}$	93.57$_{\pm 2.63}$

Table G.2: Temperature $\alpha = \frac{1}{\beta}$ for D4RL tasks

Task	α
halfcheetah-medium-expert	0.1
hopper-medium-expert	0.5
walker2d-medium-expert	0.1
halfcheetah-medium	0.05
hopper-medium	0.1
walker2d-medium	0.05
halfcheetah-medium-replay	0.05
hopper-medium-replay	0.05
walker2d-medium-replay	0.1

with soft updates of $m = 0.005$. The posterior policy π_γ is trained using the relative trajectory balance objective. π_γ is also parametrized as a state-conditioned noise-predicting DDPM, initialized as a copy of the prior. We additionally use the Langevin dynamics inductive bias (11), and learn an additional MLP for the energy scaling network. The posterior noise prediction network also outputs an additive correction to the output of the prior noise prediction network. That is, the predicted noise of the posterior diffusion model is defined as $\epsilon(s, a_t, t) := \epsilon(s, a_t, t; \theta) + \epsilon(s, a_t, t; \gamma)$, where $\epsilon(\cdot; \theta)$ is the output of the prior noise prediction network and $\epsilon(\cdot; \gamma)$ is the output of the posterior noise prediction network. We train all models on a single NVIDIA A100-Large GPU. The only hyperparameter tuned per task is the temperature α which we show in Table G.2.

Note that in these experiments both the prior and constraint are conditioned on the state $\mathbf{s}$. To prevent having to learn a neural network for $\log Z_\phi(\mathbf{s})$, we employ a variant of VarGrad objective [61]. For each state $\mathbf{s}$ sampled in the minibatch, we further generate $k = 64$ on-policy trajectories ${\tau^{(i)}}_{i=1}^{k}$ with π_γ. Each of these trajectories can be used to implicitly estimate $\log Z(\mathbf{s})$:

$$\log \hat{Z}(\mathbf{s})^{(i)} = \log \pi_\theta(\tau^{(i)} \mid \mathbf{s}) + Q_\psi(\mathbf{s}, \mathbf{a}_1^{(i)}) - \log \pi_\gamma(\tau^{(i)} \mid \mathbf{s}) \tag{22}$$

We then minimize the sample variance across the batch:

$$\mathcal{L}_{\text{RTB}}^{\text{VarGrad}}(\gamma) = \frac{1}{k} \sum_{i=1}^{k} \left(\log \hat{Z}(\mathbf{s})^{(i)} - \frac{1}{k} \sum_{j=1}^{k} \log \hat{Z}(\mathbf{s})^{(j)} \right)^2 \tag{23}$$

RTB allows off-policy training so we are not restricted to train with samples generated on-policy. We thus also leverage the offline dataset, which are samples from the prior and noise them with the DDPM noising process to generate off-policy trajectories with high density under the prior. Since there are actions in the replay buffer from high reward episodes in the tasks, this can help training efficiency compared to purely online training. We ran 5 seeds of training each with mixed training (off-policy and on-policy) and pure on-policy training on the medium-replay tasks, with results shown in Table G.1, where mixed training outperforms pure online training on two of the three tasks.

G.2 Baseline details

As is standard in offline RL, we use the reported performance numbers from the previous papers. CQL, IQL are reported from the IQL paper. Diffuser (D), DD, D-QL and QGPO are reported from the QGPO paper. Their implementation improved the performance of D and D-QL compared to their original papers. IDQL results are reported from the IDQL paper. We follow the evaluation protocol of previous work, and report the mean performance over 10 episodes, averaged across 5 random seeds at the end of training (150k training steps).

H Fine-tuning text-to-image diffusion models

We build off the DPOK implementation[8], which fine-tunes stable-diffusion-v1-5 with ImageReward function. The posterior model to be fine-tuned is initialized as a copy of the prior model. We use LoRA [28] since it is significantly more efficient than fine-tuning the entire model. Sampling of images is done with 50 steps of DDIM [69]. Even with LoRA, it is still difficult to fit gradients of all steps in the diffusion trajectory in memory. To help with this, we use a "stochastic subsampling" trick (§H.1).

We train all models on a single NVIDIA A100-Large GPU. For the main experiments, we use the default parameters for DPOK of reward weight $\beta = 10$ and KL weight = 0.01. For RTB we fix $\beta = 1.0$ for all prompts. We next perform an ablation over different values of β.

We plot in Table H.1 the final average reward and diversity score for models trained with different values of reward weight β for the prompt "A green colored rabbit.". As expected, we find that increasing β increases reward at the cost of diversity for RTB and DPOK. The exception is $\beta = 10$ for RTB which has slightly lower final reward than $\beta = 1$, which we could attribute to more difficult optimization due to the peaky distribution associated with higher reward weight.

Table H.1: Ablation of reward weights β for "A green colored rabbit.".

Model ↓	β ↓ Metric →	Reward (↑)	diversity (↑)
Prior	-	-0.113	0.597
DPOK (KL weight=0.01)	$\beta = 0.01$	-0.27	0.1488
	$\beta = 0.1$	-0.06	0.1486
	$\beta = 1.0$	0.638	0.1362
	$\beta = 10.0$	1.492	0.076
DDPO (KL weight=0.0)	$\beta = 10.0$	1.795	0.0493
RTB	$\beta = 0.01$	0.485	0.1431
	$\beta = 0.1$	1.525	0.0721
	$\beta = 1.0$	1.756	0.0436
	$\beta = 10.0$	1.568	0.0689

H.1 Memory-efficient learning

We propose two methods to reduce the memory requirement of RTB fine-tuning.

Stochastic subsampling. The expected gradient of the RTB objective (9) is unaffected by propagating gradient to a randomly sampled subset of the timesteps in a trajectory and rescaling by the inverse proportion of timesteps sampled. Stochastically subsampling timesteps for gradient propagation in this way can significantly decrease memory consumption because computation graphs for the remaining timesteps do not need to be maintained; however, such subsampling increases gradient variance, so it is preferable to keep gradients for as many timesteps as possible to fit in memory. For our text-to-image experiments, we found sampling 8 timesteps out of 50 to keep gradients was sufficient.

Batched gradient computation. An important property of the RTB objective is that computing its gradient does not require storing the computation graph of all timesteps. The gradient of the RTB objective for a single trajectory is just the sum of per-step log-likelihood gradients scaled by the RTB residual:

$$\nabla_\phi \mathcal{L}_{\text{RTB}}(\tau;\phi) = 2\left(\log \frac{Z_\phi}{r(\mathbf{x}_1)} + \sum_{i=1}^{T} \log \frac{p_\phi^{post}(\mathbf{x}_{i\Delta t} \mid \mathbf{x}_{(i-1)\Delta t})}{p_\theta(\mathbf{x}_{i\Delta t} \mid \mathbf{x}_{(i-1)\Delta t})}\right) \cdot \nabla_\phi \sum_{i=1}^{T} \log p_\phi^{\text{post}}(\mathbf{x}_{i\Delta t} \mid \mathbf{x}_{(i-1)\Delta t}).$$

Because the likelihood gradients can be accumulated during the forward pass, this allows for a batched gradient accumulation version of the update. For trajectory length (number of diffusion steps) T and accumulation batch size (number of time steps receiving a gradient signal in each backward pass) B, the number of batched forward passes required scales as $\frac{T}{B}$.

[8] https://github.com/google-research/google-research/tree/master/dpok

Only the accumulation batch size B, not the trajectory length T, is constrained by the memory budget. This means we can easily scale training with large number of diffusion steps without increasing the variance of the gradient through stochastic subsampling, with training time growing linearly with number of time steps under a fixed memory budget. Although this method is not used in the main experiments presented here, preliminary experiments confirm these observations.

We highlight that both methods are not applicable to diffusion samplers based on differentiable simulation (*e.g.*, PIS and DDS), which need to store the entire computation graph of SDE integration. For these methods, the memory requirement scales linearly with the trajectory length.

H.2 Generated images

H.2.1 A green-colored rabbit

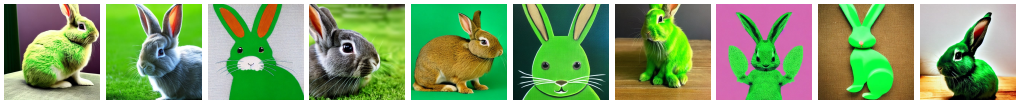


Figure H.1: Prior

Figure H.2: DDPO

Figure H.3: DPOK

Figure H.4: RTB

H.2.2 Four roses

Figure H.5: Prior

Figure H.6: DDPO

Figure H.7: DPOK

Figure H.8: RTB

H.2.3 A cat and a dog

Figure H.9: Prior

Figure H.10: DDPO

Figure H.11: DPOK

Figure H.12: RTB

H.2.4 A half - masked rugged laboratory engineer man with cybernetic enhancements as seen from a distance, scifi character portrait by greg rutkowski, esuthio, craig mullins.

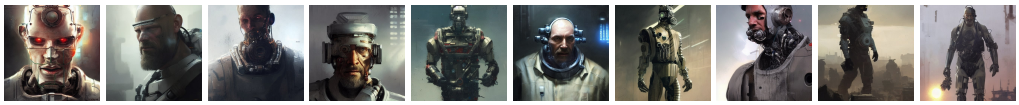


Figure H.13: Prior

Figure H.14: DDPO

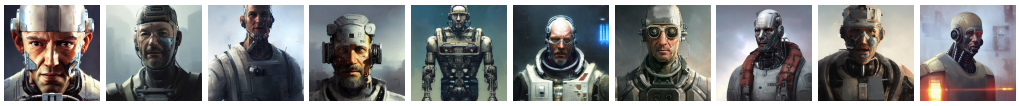


Figure H.15: DPOK

Figure H.16: RTB

I Compute resources

For classifier guidance experiments §3.1 we use train on a single NVIDIA V100 GPU. For text-conditional image generation §3.2, text infilling §F and offline §G, we use a single NVIDIA A100 large GPU. The total estimated compute time for all our experiments is 3000 hours.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We discuss our theoretical claims about the RTB ojective in §2, and report experimental results in §3.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our proposed method in §5.

   Guidelines:

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: See §A.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have provided code to reproduce our experiments in §D, and described training details in §C, §E, §F, §G and §H.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.

- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is provided in §D and can be used to reproduce our main results. We will be releasing our code publicly..

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Training details are outlined in §C, §E, §F, and §G.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: All experimental results have error bars, except the Stable Diffusion finetuning experiment §3.2 which was only trained on one seed per prompt due to compute constraints.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Compute resources used in experiments is outlined in §E, §F, §G, §H and summarised in §I.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The paper conforms to the stated ethics guidelines.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss broader impact of the work in §5.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not release models that have risk of misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Creaters of code we use are properly credited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not have any crowdsourcing experiments or research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We do not have any experiments with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.