Implicit Diffusion: Efficient optimization through stochastic sampling

Pierre Marion [12]
Institute of Mathematics, EPFL
Lausanne, Switzerland

Anna Korba
ENSAE CREST
IP Paris, France

Peter Bartlett, Mathieu Blondel, Valentin De Bortoli, Arnaud Doucet, Felipe Llinares-Lopez , Courtney Paquette, Quentin Berthet[1]
Google DeepMind

Abstract

Sampling and automatic differentiation are both ubiquitous in modern machine learning. At its intersection, differentiating through a sampling operation, with respect to the parameters of the sampling process, is a problem that is both challenging and broadly applicable. We introduce a general framework and a new algorithm for first-order optimization of parameterized stochastic diffusions, performing jointly, in a single loop, optimization and sampling steps. This approach is inspired by recent advances in bilevel optimization and automatic implicit differentiation, leveraging the point of view of sampling as optimization over the space of probability distributions. We provide theoretical and experimental results showcasing the performance of our method.

1 INTRODUCTION

Sampling from a target distribution is a ubiquitous task at the heart of various methods in machine learning, optimization, and statistics. Increasingly, sampling algorithms rely on iteratively applying large-scale parameterized functions (e.g. neural networks), as in denoising diffusion models (Ho et al., 2020).

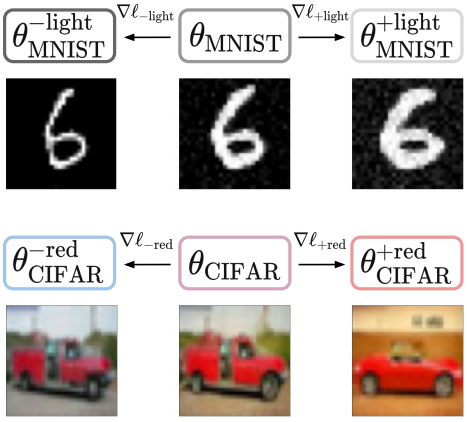


Figure 1: Optimizing through sampling with **Implicit Diffusion** to finetune denoising diffusion models. Reward is brightness for MNIST and red for CIFAR-10.

This iterative *sampling operation* implicitly maps a parameter $\theta \in \mathbb{R}^p$ to some distribution $\pi^\star(\theta)$ in $\mathcal{P}$.

In this work, we focus on optimization problems over these implicitly parameterized distributions. For a space of distributions $\mathcal{P}$ (e.g. over $\mathbb{R}^d$), and a function $\mathcal{F} : \mathcal{P} \to \mathbb{R}$, our main problem is

$$\min_{\theta \in \mathbb{R}^p} \ell(\theta) := \min_{\theta \in \mathbb{R}^p} \mathcal{F}(\pi^\star(\theta))$$

This setting encompasses for instance learning parameterized Langevin diffusions, contrastive learning of energy-based models (Gutmann and Hyvärinen, 2012) or finetuning denoising diffusion models (e.g., Dvijotham et al., 2023; Clark et al., 2024), as illustrated by Figure 1. Applying first-order optimizers to this problem raises the challenge of computing gradients of functions of the target distribution with respect to the parameter: we have to *differentiate through a sampling operation*, where the link between θ and $\pi^\star(\theta)$ can be *implicit* (see, e.g., Figure 2).

[1]Corresponding authors. Address correspondance at `pierre.marion@epfl.ch` and `qberthet@google.com`.

[2]Work mostly done while a student researcher at Google DeepMind.

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

To this aim, we propose to exploit the perspective of *sampling as optimization*, where the task of sampling is seen as an optimization problem over the space of probability distributions $\mathcal{P}$ (see Korba and Salim, 2022, and references therein). Typically, approximating a target probability distribution π can be cast as the minimization of a dissimilarity functional between probability distributions w.r.t. π, that only vanishes at the target. For instance, Langevin diffusion dynamics follow a gradient flow of a Kullback-Leibler (KL) objective w.r.t. the Wasserstein-2 distance (Jordan et al., 1998).

This allows us to draw a link between optimization through stochastic sampling and *bilevel optimization*, which often involves computing derivatives of the solution of a parameterized optimization problem obtained after iterative steps of an algorithm. Bilevel optimization is an active area of research with many relevant applications in machine learning, such as hyperparameter optimization (Franceschi et al., 2018) or meta-learning (Liu et al., 2019). In particular, there is a significant effort in the literature for developing tractable and provably efficient algorithms in a large-scale setting (Pedregosa, 2016; Chen et al., 2021b; Arbel and Mairal, 2022; Blondel et al., 2022; Dagréou et al., 2022)–see Appendix D for additional related work. This literature focuses mostly on problems with finite-dimensional variables, in contrast with our work where the solution of the inner problem is a distribution in $\mathcal{P}$.

These motivating similarities, while useful, are not limiting. We also consider settings where the sampling iterations are not readily interpretable as an optimization algorithm. Denoising diffusion cannot directly be formalized as descent dynamics of an objective functional over $\mathcal{P}$, but its output is determined by a parameter θ (i.e. weights of the score matching neural networks).

Main Contributions. In this work, we introduce the algorithm of **Implicit Diffusion**, an effective and principled technique for optimizing through a sampling operation. More precisely,

- We present a general framework describing parameterized sampling algorithms, and introduce Implicit Diffusion optimization, a **single-loop** optimization algorithm to optimize through sampling.
- We provide **theoretical guarantees** in the continuous and discrete time settings in Section 4.
- We showcase in Section 5 its performance and efficiency in **experimental settings**. Applications include finetuning denoising diffusions and training energy-based models.

To allow for reproducibility, we provide an implementation[3] of our algorithm in JAX (Bradbury et al., 2018).

Notation. For a set $\mathcal{X}$ (such as $\mathbb{R}^d$), we write $\mathcal{P}$ for the set of probability distributions on $\mathcal{X}$, omitting reference to $\mathcal{X}$. For f a differentiable function on $\mathbb{R}^d$, we denote by ∇f its gradient function. If f is a differentiable function of k variables, we let $\nabla_i f$ denote its gradient w.r.t. its i-th variable.

2 PROBLEM PRESENTATION

2.1 Sampling and optimization perspectives

The core operation that we consider is sampling by running a stochastic diffusion process that depends on a parameter $\theta \in \mathbb{R}^p$. We consider *iterative sampling operators* that map from a *parameter space* to a *space of probabilities*. We denote by $\pi^\star(\theta) \in \mathcal{P}$ the outcome distribution of this sampling operator. This parameterized distribution is defined in an *implicit* manner: there is not always an explicit way to write down its dependency on θ. More formally, it is defined as follows.

Definition 2.1 (Iterative sampling operators)**.** For a parameter $\theta \in \mathbb{R}^p$, a sequence of parameterized functions $\Sigma_s(\cdot, \theta)$ from $\mathcal{P}$ to $\mathcal{P}$ defines a diffusion *sampling process*, from $p_0 \in \mathcal{P}$ iterating

$$p_{s+1} = \Sigma_s(p_s, \theta)\,. \tag{1}$$

The outcome $\pi^\star(\theta) \in \mathcal{P}$, when $s \to \infty$, or for some $s = T$ defines a *sampling operator* $\pi^\star : \mathbb{R}^p \to \mathcal{P}$. □

We embrace the formalism of stochastic processes as acting on probability distributions. This perspective focuses on the *dynamics* of the distribution $(p_s)_{s\geq 0}$, and allows us to more clearly present our optimization problem and algorithms. In practice, however, in all the examples that we consider, this is realized by an iterative process on some random variable X_s such that $X_s \sim p_s$.

Example 2.2. Consider the process defined by $X_{s+1} = X_s - 2\delta(X_s - \theta) + \sqrt{2\delta}B_s$, where the B_s are i.i.d. standard Gaussian, $X_0 \sim p_0 := \mathcal{N}(\mu_0, \sigma_0^2)$, and $\delta \in (0, 1)$. This is the discrete-time version of Langevin dynamics for $V(x, \theta) = 0.5(x - \theta)^2$ (see Section 2.2). The dynamics induced on probabilities $p_s = \mathcal{N}(\mu_s, \sigma_s^2)$ are $\mu_s = \theta + (1-2\delta)^s(\mu_0 - \theta)$ and $\sigma_s^2 = 1 + (1-2\delta)^{2s}(\sigma_0^2 - 1)$. The sampling operator for $s \to \infty$ is therefore defined by $\pi^\star : \theta \to \mathcal{N}(\theta, 1)$. □

More generally, we may consider the iterates X_s of the process defined for noise variables $(B_s)_{s\geq 0}$

$$X_{s+1} = f_s(X_s, \theta) + B_s\,. \tag{2}$$

[3] `https://github.com/google-deepmind/implicit_diffusion`

Applying $f_s(\cdot, \theta)$ to $X_s \sim p_s$ implicitly defines a dynamic $\Sigma_s(\cdot, \theta)$ on the distribution. The dynamics on the variables in (2) induce dynamics on the distributions described in (1). Note that, in the special case of normalizing flows (Kobyzev et al., 2019; Papamakarios et al., 2021), explicit formulas for p_s can be derived and evaluated.

Remark 2.3. *i)* We consider settings with discrete time steps, fitting our focus on algorithms to sample and optimize through sampling. This encompasses in particular the discretization of many continuous-time stochastic processes of interest. Most of our motivations are of this latter type, and we describe these distinctions in our examples (see Section 2.2).

ii) As noted above, these dynamics are often realized by an iterative process on variables X_s, or even on an i.i.d. batch of samples $(X_s^1, \dots, X_s^n)$. When the iterates $\Sigma_s(p_s, \theta)$ are written in our presentation (e.g. in optimization algorithms in Section 3), it is often a shorthand to mean that we have access to samples from p_s, or equivalently to an empirical version $\hat{p}_{s,(n)}$ of the population distribution p_s. Sample versions of our algorithms are described in Appendix A.

iii) One of the **special cases** considered in our analysis are stationary processes with infinite time horizon, where the sampling operation can be interpreted as optimizing over the set of distributions

$$\pi^\star(\theta) = \operatorname*{argmin}_{p \in \mathcal{P}} \mathcal{G}(p, \theta)\,, \quad \text{for some } \mathcal{G} : \mathcal{P} \times \mathbb{R}^p \to \mathbb{R}\,. \tag{3}$$

In this case, the iterative operations in (1) can often be directly interpreted as descent steps for the objective $\mathcal{G}(\cdot, \theta)$. However, our methodology is **not limited** to this setting: we also consider general sampling schemes with no stationarity and no inner $\mathcal{G}$, but only a sampling process defined by Σ_s.

Optimization objective. We aim to optimize with respect to θ the output of the sampling operator, for a function $\mathcal{F} : \mathcal{P} \to \mathbb{R}$. In other words, we consider the optimization problem

$$\min_{\theta \in \mathbb{R}^p} \ell(\theta) := \min_{\theta \in \mathbb{R}^p} \mathcal{F}(\pi^\star(\theta))\,. \tag{4}$$

This formulation transforms a problem over distributions in $\mathcal{P}$ to a finite-dimensional problem over $\theta \in \mathbb{R}^p$. Further, this allows for convenient post-optimization sampling: for some $\theta_{\text{opt}} \in \mathbb{R}^p$ obtained by solving (4), one can sample from $\pi^\star(\theta_{\text{opt}})$. This is the common paradigm in model finetuning.

2.2 Examples

Langevin dynamics. They are defined by the stochastic differential equation (SDE) (Roberts and Tweedie, 1996)

$$\mathrm{d}X_t = -\nabla_1 V(X_t, \theta)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t\,, \tag{5}$$

where V and $\theta \in \mathbb{R}^p$ are such that this SDE has a solution for $t > 0$ that converges in distribution. Here $\pi^\star(\theta)$ is the limiting distribution of X_t when $t \to \infty$, which is the Gibbs distribution

$$\pi^\star(\theta)[x] = \exp(-V(x, \theta))/Z_\theta\,, \tag{6}$$

where $Z_\theta = \int \exp(-V(x, \theta))\mathrm{d}x$ is the normalization factor. To fit our setting of iterative sampling algorithms (2), one can consider the discretization for $\gamma > 0$

$$X_{k+1} = X_k - \gamma \nabla_1 V(X_k, \theta) + \sqrt{2\gamma}\Delta B_{k+1}\,.$$

For $\mathcal{G}(p, \theta) = \mathrm{KL}(p || \pi^\star(\theta))$, the outcome of the sampling operator $\pi^\star(\theta)$ is a minimum of $\mathcal{G}(\cdot, \theta)$, and the SDE (5) implements a gradient flow for $\mathcal{G}$ in the space of measures, with respect to the Wasserstein-2 distance (Jordan et al., 1998). Two optimization objectives $\mathcal{F}$ are of particular interest in this case. First, we may want to maximize some reward $R : \mathbb{R}^d \to \mathbb{R}$ over our samples, in which case the objective writes $\mathcal{F}(p) := -\mathbb{E}_{x \sim p}[R(x)]$. Second, to approximate a reference distribution p_{ref} with sample access, it is possible to take $\mathcal{F}(p) := \mathrm{KL}(p_{\text{ref}} \,||\, p)$. This case corresponds to training energy-based models (Gutmann and Hyvärinen, 2012). It is also possible to consider a linear combination of these two objectives.

Denoising diffusion. It consists in running the SDE for $Y_0 \sim \mathcal{N}(0, I)$,

$$\mathrm{d}Y_t = \{Y_t + 2s_\theta(Y_t, T - t)\}\mathrm{d}t + \sqrt{2}\mathrm{d}B_t\,, \tag{7}$$

where $s_\theta : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ is a parameterized *score function* (Hyvärinen, 2005; Vincent, 2011; Ho et al., 2020). Its aim is to reverse a forward process $\mathrm{d}X_t = -X_t\mathrm{d}t + \sqrt{2}\mathrm{d}B_t$, where we have sample access to $X_0 \sim p_{\text{data}} \in \mathcal{P}$. More precisely, denoting by p_t the distribution of X_t, if $s_\theta \approx \nabla \log p_t$, then the distribution of Y_T is close to p_{data} for large T (Anderson, 1982), which allows approximate sampling from p_{data}. Implementations of s_θ include U-Nets (Ronneberger et al., 2015) or Vision Transformers that split the image into patches (Peebles and Xie, 2023). We present for simplicity an unconditioned model, but conditioning (on class, prompt, etc.) also falls in our framework.

We are interested in optimizing through diffusion sampling and consider $\pi^\star(\theta)$ as the distribution of Y_T. A key example is when θ_0 represents the weights of a model s_{θ_0} that has been pretrained by score matching and one wants to finetune the target distribution $\pi^\star(\theta)$, e.g. in order to increase a reward $R : \mathbb{R}^d \to \mathbb{R}$. Figure 3 situates our contribution within the broader

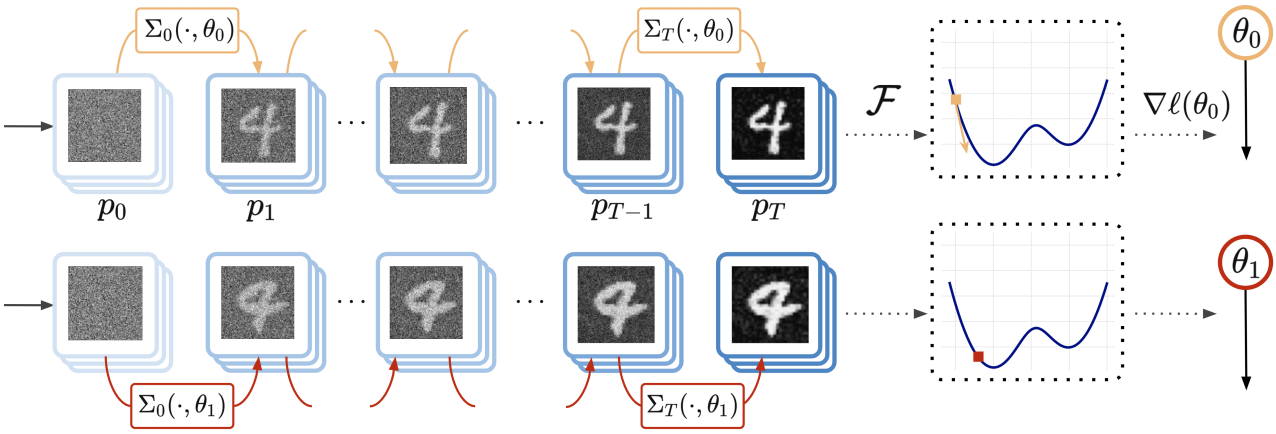


Figure 2: A step of optimization through sampling. For a given parameter θ_0, the sampling process is defined by applying Σ_s for $s \in [T]$, producing $\pi^\star(\theta_0)$. The goal of optimization through sampling is to update θ to minimize $\ell = \mathcal{F} \circ \pi^\star$. Here the objective $\mathcal{F}$ corresponds to having lighter images (on average), which produces thicker digits.

literature on this problem (see details in Appendix D). Note that this finetuning step does not require access to p_{data}. As for Langevin dynamics, we consider in our algorithms discrete approximations of the process (7). However in this case, there exists no natural functional $\mathcal{G}$ minimized by the sampling process. An alternative to (7) producing the same marginal distributions is the ordinary differential equation (ODE)

$$Y_0 \sim \mathcal{N}(0, I)\,, \quad \mathrm{d}Y_t = \{Y_t + s_\theta(Y_t, T-t)\}\mathrm{d}t\,. \tag{8}$$

3 METHODS

The objective (4) presents several challenges, that we review here. We then introduce an overview of our approach, before detailing our algorithms.

3.1 Overview

Estimation of gradients through sampling. Even with samples from $\pi^\star(\theta)$, applying a first-order method to (4) requires evaluating gradients of $\ell = \mathcal{F} \circ \pi^\star$. Since there is no closed form for ℓ and no explicit computational graph, we consider the following alternative setting to evaluate gradients.

Definition 3.1 (Implicit gradient estimation)**.** The gradient of ℓ can be *implicitly estimated* if $\Sigma_s, \mathcal{F}$ are such that there exists $\Gamma : \mathcal{P} \times \mathbb{R}^p \to \mathbb{R}^p$ such that $\nabla \ell(\theta) = \Gamma(\pi^\star(\theta), \theta)$. □

In practice we rarely reach exactly the distribution $\pi^\star(\theta)$, e.g. because a finite number of iterations of sampling is performed. Then, if $\hat{\pi} \approx \pi^\star(\theta)$, the gradient can be approximated by $\hat{g} = \Gamma(\hat{\pi}, \theta)$. In particular, given access to approximate samples of $\pi^\star(\theta)$, it is possible to compute an estimate of $\nabla \ell(\theta)$, and this is at the heart of our methods–see Appendix A.1 for more details. Note that when Γ is linear in its first argument, sample access to $\pi^\star(\theta)$ yields unbiased estimates of the gradient. This case has been studied with various approaches (see Sutton et al., 1999; Fu and Hu, 2012; Pflug, 2012; De Bortoli et al., 2021, and Appendix D).

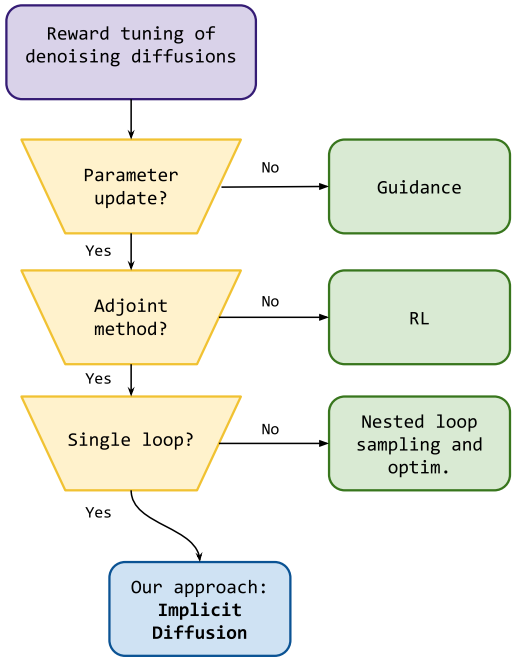


Figure 3: Main approaches for reward tuning of denoising diffusions. References are given in Appendix D.

Remark 3.2. Definition 3.1 is in fact always satisfied by the tautological definition $\Gamma(p, \theta) := \nabla \ell(\theta)$. Rather than the existence of Γ, key questions are whether Γ is easily computable, can be stochastically approximated as explained above, or enjoys theoretical guarantees. We present several sampling problems in Section 3.2

where this is the case. This point of view should also extend beyond sampling to any setting that involves optimizing over a parameterized distribution, as for instance in Wasserstein Distributionally Robust Optimization (Mohajerin Esfahani and Kuhn, 2018), a method for robust learning. We leave these extensions to future investigations.

Beyond nested-loop approaches. Sampling from $\pi^\star(\theta)$ is usually only feasible via iterations of the sampling process Σ_s. The most straightforward method is then a nested loop: at each optimization step k, run an inner loop for a large number T of steps of Σ_s to produce $\hat{\pi}_k \approx \pi^\star(\theta_k)$, and use it to evaluate a gradient. Algorithm 1 formalizes this baseline. This approach can be inefficient for two reasons: first, it requires solving the inner sampling problem *at each optimization step*. Further, nested loops are typically impractical with accelerator-oriented hardware. These issues can be partially alleviated by techniques like gradient checkpointing (see references in Appendix D).

Algorithm 1 Vanilla nested-loop approach (Baseline)

input $\theta_0 \in \mathbb{R}^p$, $p_0 \in \mathcal{P}$
  for $k \in \{0, \dots, K-1\}$ (outer optimization loop) **do**
    $p_k^{(0)} \leftarrow p_0$
    for $s \in \{0, \dots, T-1\}$ (inner sampling loop) **do**
      $p_k^{(s+1)} \leftarrow \Sigma_s(p_k^{(s)}, \theta_k)$
    $\hat{\pi}_k \leftarrow p_k^{(T)}$
    $\theta_{k+1} \leftarrow \theta_k - \eta \Gamma(\hat{\pi}_k, \theta_k)$ (or another optimizer)
output θ_K

We rather follow an approach inspired by methods in bilevel optimization, aiming to *jointly* iterate on both the sampling problem (evaluation of $\pi^\star$–the inner problem), and the optimization problem over $\theta \in \mathbb{R}^p$ (the outer objective $\mathcal{F}$). We describe these methods in Section 3.3 and Algorithms 2 and 3. The connection with bilevel optimization is especially seamless when sampling can indeed be cast as an optimization problem over distributions in $\mathcal{P}$, as in (3). However, as noted above, our approach generalizes beyond.

3.2 Gradient estimation through sampling

We explain how to perform implicit gradient estimation as in Definition 3.1, that is, how to derive expressions for the function Γ, in several cases of interest.

Direct analytical derivation. For Langevin dynamics, it is possible to derive analytical expressions for Γ depending on the outer objective $\mathcal{F}$. We illustrate this idea for the two objectives introduced in Section 2.2. First, in the case where $\mathcal{F}_{\mathrm{rew}}(p) = -\mathbb{E}_{x\sim p}[R(x)]$, a computation detailed in Appendix A.2 and the use of Definition 3.1 show that, for $\ell_{\mathrm{rew}}(\theta) = \mathcal{F}_{\mathrm{rew}}(\pi^\star(\theta))$,

$$\begin{aligned} \nabla \ell_{\mathrm{rew}}(\theta) &= \mathrm{Cov}_{X\sim\pi^\star(\theta)}[R(X), \nabla_2 V(X,\theta)]\,, \\ \Gamma_{\mathrm{rew}}(p,\theta) &= \mathrm{Cov}_{X\sim p}[R(X), \nabla_2 V(X,\theta)]\,. \end{aligned} \tag{9}$$

Note that this formula does not involve gradients of R, hence our approach handles any non-differentiable reward. Second, in the case where $\mathcal{F}_{\mathrm{ref}}(p) = \mathrm{KL}(p_{\mathrm{ref}} \,||\, p)$, we then have (Gutmann and Hyvärinen, 2012), for $\ell_{\mathrm{ref}}(\theta) = \mathcal{F}_{\mathrm{ref}}(\pi^\star(\theta))$,

$$\nabla \ell_{\mathrm{ref}}(\theta) = \mathbb{E}_{X\sim p_{\mathrm{ref}}}[\nabla_2 V(X,\theta)] - \mathbb{E}_{X\sim\pi^\star(\theta)}[\nabla_2 V(X,\theta)].$$

This is known as contrastive learning when p_{ref} is given by data, and suggests taking Γ as

$$\Gamma_{\mathrm{ref}}(p,\theta) := \mathbb{E}_{X\sim p_{\mathrm{ref}}}[\nabla_2 V(X,\theta)] - \mathbb{E}_{X\sim p}[\nabla_2 V(X,\theta)]\,. \tag{10}$$

This extends to linear combinations of Γ_{rew} and Γ_{ref}.

Implicit differentiation. When, as in (3), $\pi^\star(\theta) = \operatorname{argmin} \mathcal{G}(\cdot,\theta)$, under generic assumptions on $\mathcal{G}$ the implicit function theorem (see Krantz and Parks, 2002; Blondel et al., 2022 and Appendix A.4) shows that $\nabla \ell(\theta) = \Gamma(\pi^\star(\theta),\theta)$ with

$$\Gamma(p,\theta) = \int \mathcal{F}'(p)[x]\gamma(p,\theta)[x]\mathrm{d}x\,.$$

Here $\mathcal{F}'(p) : \mathcal{X} \to \mathbb{R}$ denotes the first variation of $\mathcal{F}$ at $p \in \mathcal{P}$ (see Definition B.1) and $\gamma(p,\theta)$ is the solution of the linear system $\int \nabla_{1,1}\mathcal{G}(p,\theta)[x,x']\gamma(p,\theta)[x']\mathrm{d}x' = -\nabla_{1,2}\mathcal{G}(p,\theta)[x]$. Although this gives us a general way to define gradients of $\pi^\star(\theta)$ with respect to θ, solving this linear system is generally not feasible. One exception is when sampling over a finite state space $\mathcal{X}$, in which case $\mathcal{P}$ is finite-dimensional, and the integrals boil down to matrix-vector products.

Differential adjoint method. The adjoint method allows computing gradients through differential equation solvers (Pontryagin, 1987; Li et al., 2020), applying in particular for denoising diffusion. It can be connected to implicit differentiation, by defining $\mathcal{G}$ over a measure path instead of a single measure p (see, e.g., Kidger, 2022). To introduce this method, consider the ODE $\mathrm{d}Y_t = \mu(t, Y_t, \theta)\mathrm{d}t$ integrated between 0 and some $T > 0$. This setting encompasses the denoising diffusion ODE (8). Assume that the outer objective $\mathcal{F}$ writes as the expectation of some differentiable reward R, namely $\mathcal{F}(p) = \mathbb{E}_{x\sim p}[R(x)]$. Let $Z_0 \sim p$, $A_0 = \nabla R(Z_0)$, $G_0 = 0$, and consider the ODE system

$$\mathrm{d}Z_t = -\mu(t, Z_t, \theta)\mathrm{d}t\,, \quad \mathrm{d}A_t = A_t^\intercal \nabla_2 \mu(T-t, Z_t, \theta)\mathrm{d}t\,,$$
$$\mathrm{d}G_t = A_t^\intercal \nabla_3 \mu(T-t, Z_t, \theta)\mathrm{d}t\,.$$

Defining $\Gamma(p, \theta) := G_T$, the adjoint method shows that $\Gamma(\pi^\star(\theta), \theta)$ is an unbiased estimate of $\nabla \ell(\theta)$. We refer to Appendix A.3 for details and explanations on how to differentiate through the SDE sampler (7) and to incorporate a KL term in the reward by using Girsanov's theorem.

3.3 Implicit Diffusion optimization algorithm

To circumvent solving the inner sampling problem in Algorithm 1, we propose in Algorithm 2 a **joint single-loop approach** that keeps track of a single dynamic of probabilities $(p_k)_{k\geq 0}$. At each optimization step, the probability p_k is updated with one sampling step that depends on the current parameter θ_k. As noted in Section 3.1, there are parallels with approaches in the literature when Γ is linear, but we go beyond in making no such assumption.

Algorithm 2 Implicit Diff. optimization, infinite time

input $\theta_0 \in \mathbb{R}^p$, $p_0 \in \mathcal{P}$
  for $k \in \{0, \dots, K-1\}$ (joint single loop) **do**
    $p_{k+1} \leftarrow \Sigma_k(p_k, \theta_k)$
    $\theta_{k+1} \leftarrow \theta_k - \eta \Gamma(p_k, \theta_k)$ (or another optimizer)
output θ_K

This point of view is well-suited for stationary processes with infinite-time horizon, but does not directly adapt to differentiation through diffusions with a finite-time horizon (and no stationary property). Indeed, it does not make sense in this case to run sampling for an arbitrary number of steps. Our approach can then be adapted as follows.

Algorithm 3 Implicit Diff. optimization, finite time

input $\theta_0 \in \mathbb{R}^p$, $p_0 \in \mathcal{P}$
input $P_M = [p_0^{(0)}, \dots, p_0^{(M)}]$
  for $k \in \{0, \dots, K-1\}$ (joint single loop) **do**
    $p_{k+1}^{(0)} \leftarrow p_0$
    parallel $p_{k+1}^{(m+1)} \leftarrow \Sigma_m(p_k^{(m)}, \theta_k)$ for $m \in [M-1]$
    $\theta_{k+1} \leftarrow \theta_k - \eta \Gamma(p_k^{(M)}, \theta_k)$ (or another optimizer)
output θ_K

Finite time-horizon: queuing trick. When $\pi^\star(\theta)$ is obtained by a large, but *finite* number T of iterations of the operator Σ_s, we leverage hardware parallelism to evaluate in parallel several, say M, steps of the dynamics of the distribution p_k, through a queue of length M. We present for simplicity in Figure 4 and in Algorithm 3 the case where $M = T$ and discuss extensions in Appendix A.3. At each step, the last element of the queue $p_k^{(M)}$ provides a distribution to update θ through evaluation of Γ. Note that its dynamics in the previous M steps, from $p_{k-M}^{(0)}$ to $p_k^{(M)}$, used the M previous values of the parameter $\theta_{k-M}, \dots, \theta_{k-1}$. In particular, it provides a biased estimate of $\nabla \ell(\theta_k)$. Importantly, leveraging parallelism, the runtime of our algorithm is $\mathcal{O}(K)$, gaining a factor of T compared to the nested-loop approach, but at a higher memory cost.

4 THEORETICAL ANALYSIS

Our theoretical guarantees cover Langevin diffusions in the continuous and discrete settings, and a simple case of denoising diffusion. Proofs are given in Appendix B.

4.1 Langevin with continuous flow

The continuous-time equivalent of Algorithm 2 in the case of Langevin dynamics is

$$\begin{aligned} \mathrm{d}X_t &= -\nabla_1 V(X_t, \theta_t)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t\,, \\ \mathrm{d}\theta_t &= -\varepsilon_t \Gamma(p_t, \theta_t)\mathrm{d}t\,, \end{aligned} \tag{11}$$

where p_t denotes the distribution of X_t and $\varepsilon_t > 0$ corresponds to the ratio of learning rates between the inner and the outer problems. In practice $\Gamma(p_t, \theta_t)$ is approximated on a finite sample, making the dynamics in θ_t stochastic. We leave the analysis of these stochastic dynamics for future work. A possible tool to do so is the *propagation of chaos* (Chaintron and Diez, 2022; Suzuki et al., 2023), a theory which aims at quantifying the deviation between the dynamics of a system of finitely many interacting particles and the limiting behavior described by a mean-field density.

Recalling the definition (6) of $\pi^\star(\theta)$, we require the following assumptions.

Assumption 4.1. $\pi^\star(\theta_t)$ verifies the Log-Sobolev inequality with constant $\mu > 0$ for all $t \geq 0$, i.e., for all $p \in \mathcal{P}$, $\mathrm{KL}(p \,||\, \pi^\star(\theta_t)) \leq \frac{1}{2\mu} \|\nabla \log\Big(\frac{p}{\pi^\star(\theta_t)}\Big)\|^2_{L^2(p)}$.

Assumption 4.2. V is continuously differentiable and for $\theta \in \mathbb{R}^p, x \in \mathbb{R}^d$, $\|\nabla_2 V(x, \theta)\| \leq C$, for some $C > 0$.

Assumption 4.1 generalizes μ-strong convexity of the potentials $(V(\cdot, \theta_t))_{t\geq 0}$, including for instance distributions π whose potentials are bounded perturbations of a strongly convex potential (Bakry et al., 2014; Vempala and Wibisono, 2019). Assumptions 4.1 and 4.2 hold for example when the potential defines a mixture of Gaussians and the parameters θ determine the weights of the mixture (see Appendix B.1.2 for details). We also assume that the outer updates are bounded and Lipschitz continuous for the KL divergence.

Assumption 4.3. For $p, q \in \mathcal{P}$, $\theta \in \mathbb{R}^p$, $\|\Gamma(p, \theta)\| \leq C$ and $\|\Gamma(p, \theta) - \Gamma(q, \theta)\| \leq K_\Gamma \sqrt{\mathrm{KL}(p||q)}$, for some $C, K_\Gamma > 0$.

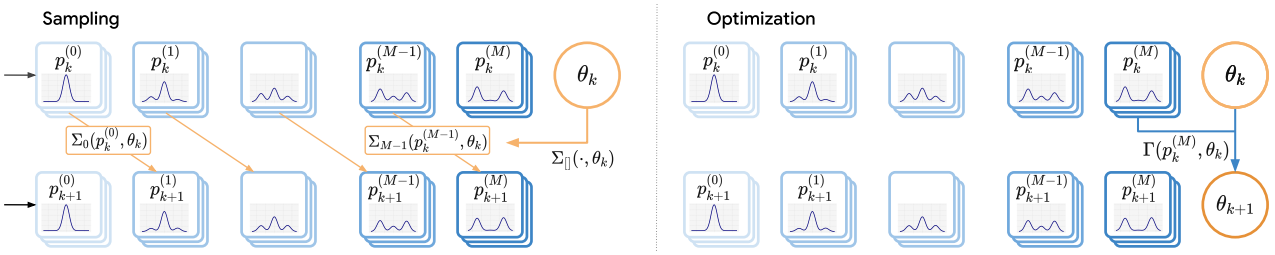


Figure 4: Illustration of the Implicit Diffusion algorithm, in the finite time setting. Left: Sampling - one step of the parameterized sampling scheme is applied in parallel to all distributions in the queue. Right: Optimization - the last element of the queue is used to compute a gradient for the parameter.

The next proposition shows that this assumption holds for the examples of interest given in Section 2.

Proposition 4.4. *Consider a bounded function* $R : \mathbb{R}^d \to \mathbb{R}$. *Then, under Assumption 4.2, functions* Γ_{rew} *and* Γ_{ref} *defined by* (9)–(10) *satisfy Assumption 4.3.*

Since we make no strong convexity assumption, we cannot hope to prove convergence to a global minimizer, rather convergence of the average objective gradients. Note that, with strong convexity, it is a rather standard extension to prove convergence to the global minimizer (see, e.g., in a similar context, Dagréou et al., 2022).

Theorem 4.5. *Take* $\varepsilon_t = \min(1, 1/\sqrt{t})$ *in* (11). *Then, under Assumptions 4.1, 4.2, and 4.3,*

$$\frac{1}{T}\int_0^T \|\nabla \ell(\theta_t)\|^2 \mathrm{d}t \leq \frac{c(\ln T)^2}{T^{1/2}}\,, \qquad \text{for some } c > 0\,.$$

The proof starts by noting that updates in θ would follow the gradient flow for ℓ if $p_t = \pi^\star(\theta_t)$. The deviation to these ideal dynamics can be controlled by the KL divergence of p_t from $\pi^\star(\theta_t)$, which can itself be bounded since updates in X_t are gradient steps for the KL (see Section 2.2). Finally, the decay of the ratio of learning rates ε_t ensures that $\pi^\star(\theta_t)$ is not moving away from p_t too fast due to updates in θ_t. Taking ε_t small amounts to a *two-timescale approach*, a tool commonly used to tackle non-convex optimization problems in machine learning (Heusel et al., 2017; Arbel and Mairal, 2022; Hong et al., 2023; Marion and Berthier, 2023).

4.2 Langevin with discrete flow

We now consider the discrete version of (11), namely

$$\begin{aligned} X_{k+1} &= X_k - \gamma_k \nabla_1 V(X_k, \theta_k) + \sqrt{2\gamma_k}\Delta B_{k+1}\,, \\ \theta_{k+1} &= \theta_k - \gamma_k \varepsilon_k \Gamma(p_k, \theta_k)\,, \end{aligned} \tag{12}$$

where p_k denotes the distribution of X_k. This setting is more challenging due to the discretization bias (Dalalyan, 2017). We make classical smoothness assumptions to analyze discrete gradient descent (e.g., Cheng and Bartlett, 2018):

Assumption 4.6. The functions $\nabla_1 V(\cdot, \theta)$, $\nabla_1 V(x, \cdot)$ and $\nabla \ell$ are respectively L_X-Lipschitz for all $\theta \in \mathbb{R}^p$, L_Θ-Lipschitz for all $x \in \mathbb{R}^d$, and L-Lipschitz.

We can then show the following convergence result.

Theorem 4.7. *Take* $\gamma_k = c_1/\sqrt{k}$ *and* $\varepsilon_k = 1/\sqrt{k}$ *in* (12). *Under Assumptions 4.1, 4.2, 4.3, and 4.6,*

$$\frac{1}{K}\sum_{k=1}^K \|\nabla \ell(\theta_k)\|^2 \leq \frac{c_2 \ln K}{K^{1/3}}\,, \qquad \text{for some } c_1, c_2 > 0\,.$$

The proof technique to bound the KL in discrete iterations is inspired by Cheng and Bartlett (2018). Comparing to the continuous case, this step incurs a discretization error proportional to $\gamma_k \to 0$, inducing a slower convergence rate. The result is similar to Dagréou et al. (2022) for finite-dimensional bilevel optimization, albeit our final convergence rate is slower.

4.3 Denoising diffusion

The case of denoising diffusion is more challenging since $\pi^\star(\theta)$ can **not** be readily characterized as the stationary point of an iterative process. We study a 1D Gaussian case and leave more general analysis for future work. Considering $p_{\text{data}} = \mathcal{N}(\theta_{\text{data}}, 1)$ and the forward process of Section 2.2, a straightforward computation shows that the score is given by $\nabla \log p_t(x) = -(x - \theta_{\text{data}} e^{-t})$. A natural parameterization of the score function is therefore $s_\theta(x, t) := -(x - \theta e^{-t})$. With this parameterization, the output of the sampling process (7) is $\pi^\star(\theta) = \mathcal{N}(\theta(1 - e^{-2T}), 1)$. Remarkably, $\pi^\star(\theta)$ is Gaussian for all $\theta \in \mathbb{R}$, making the analytical study tractable.

Assume that pretraining with samples of p_{data} yields $\theta = \theta_0$, and we want to finetune the model towards some other $\theta_{\text{target}} \in \mathbb{R}$ by optimizing the reward $R(x) = -(x - \theta_{\text{target}})^2$ over samples of $\pi^\star(\theta)$. A short computation shows that $\nabla \ell(\theta) = -\mathbb{E}_{x \sim \pi^\star(\theta)} R'(x)(1 - e^{-2T})$, hence one can take $\Gamma(p, \theta) = -\mathbb{E}_{x \sim p} R'(x)(1 - e^{-2T})$. In this setting, we study a continuous-time version of

Algorithm 3, where Σ is the denoising diffusion (7) and Γ is given above, and show convergence of θ to θ_{target}. This shows that Algorithm 3 successfully finetunes the parameter to optimize the reward. We refer to Appendix B.3 for details.

Proposition 4.8. *(informal) Let $(\theta_t)_{t\geq 0}$ be given by the continuous-time equivalent of Algorithm 3. Then $\|\theta_{2T} - \theta_{\text{target}}\| = \mathcal{O}(e^{-T})$, and $\pi^\star(\theta_{2T}) = \mathcal{N}(\mu_{2T}, 1)$ with $\mu_{2T} = \theta_{\text{target}} + \mathcal{O}(e^{-T})$.*

5 EXPERIMENTS

We empirically illustrate the performance of Implicit Diffusion. Details are given in Appendix C.

5.1 Reward training of Langevin processes

We consider the case where the potential $V(\cdot, \theta)$ is a logsumexp of quadratics—so that the outcome distributions are mixtures of Gaussians. We optimize the reward $R(x) = \mathbf{1}(x_1 > 0)\exp\big(-\|x-\mu\|^2\big)$, for $\mu \in \mathbb{R}^d$, thereby illustrating the ability of our method to optimize rewards that are not differentiable. We run six sampling algorithms, including the infinite time-horizon version of **Implicit Diffusion** (Algorithm 2), all starting from $p_0 = \mathcal{N}(0, I_d)$ and for $K = 5,000$ steps.

- ■ Langevin diffusion (5) with potential $V(\cdot, \theta_0)$ for some fixed $\theta_0 \in \mathbb{R}^p$, no reward.
- ★ Implicit Diffusion with $\mathcal{F}(p) = -\mathbb{E}_{X\sim p}[R(X)]$, yields both a sample $\hat{p}_K$ and parameters θ_{opt}.
- ▼ Langevin (5) with potential $V(\cdot, \theta_0) - \lambda R_{\text{smooth}}$, where R_{smooth} is a smoothed version of R.
- ● Langevin (5) with potential $V(\cdot, \theta_{\text{opt}})$. This is inference post-training with Implicit Diffusion.
- ♦ Nested loop (Algorithm 1) with T inner sampling steps for each gradient step.
- ▲ Unrolling through the last step of sampling with T inner sampling steps for each outer step.

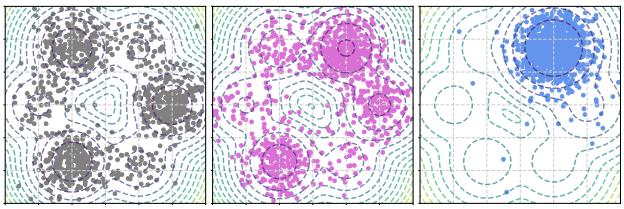


Figure 5: Contour lines and samples for (●): Langevin θ_0 - (●) Unrolling with $T = 100$ inner sampling steps - (●) Implicit Diffusion.

Both qualitatively (Figure 5) and quantitatively (Figure 6), we observe that our approach efficiently optimizes through sampling. We analyze performance both in terms of *steps* (number of optimization steps–updates in θ) and *gradient evaluations* (number of sampling steps). After K optimization steps, our algorithm yields both $\theta_{\text{opt}} := \theta_K$ and a sample $\hat{p}_K$ approximately from $\pi^\star(\theta_{\text{opt}})$. Then, it is convenient and fast to sample post hoc, with a Langevin process using θ_{opt}–as observed in Figure 6. This is similar in spirit to inference with a finetuned model, post-reward training. We compare our approach with several baselines. First, directly adding a reward term (▼) is less efficient: it tends to overfit on the reward, as the target distribution of this process is out of the family of $\pi^\star(\theta)$'s. Second, unrolling through the last step of sampling (▲, ▲) leads to much slower optimization. Finally, using the nested loop approach (♦, ♦) makes each optimization step T times more costly, while barely improving the performance after a given number of steps (due to less biased gradients). When comparing in terms of number of gradient evaluations, Implicit Diffusion strongly outperforms the nested loop approach. In other words, for a given computational budget, it is optimal to take $T = 1$ and perform more optimization steps, which corresponds to the Implicit Diffusion single-loop approach. Further comparison plots, as well as a variant of this experiment where we learn a reference distribution (i.e., train from scratch an energy-based model) are included in Appendix C. We also include an experiment where the potential V parameterizes the means and the covariances of the Gaussians in addition to the weights of the mixture.

5.2 Reward training of denoising diffusion

We also apply **Implicit Diffusion** for reward finetuning of denoising diffusion models pretrained on image datasets. We denote by θ_0 the weights of a pretrained model, such that $\pi^\star(\theta_0) \approx p_{\text{data}}$. For various reward functions on the samples $R : \mathbb{R}^d \to \mathbb{R}$, we consider

$$\mathcal{F}(p) := -\lambda \mathbb{E}_{x\sim p}[R(x)] + \beta\, \mathrm{KL}(p \,||\, \pi^\star(\theta_0))\,,$$

common in reward finetuning (see, e.g., Ziegler et al., 2019, and references therein), for positive and negative values of λ. We run **Implicit Diffusion** using the finite time-horizon variant (Algorithm 3), applying the adjoint method on SDEs for gradient estimation. We report selected samples of $\pi^\star(\theta_t)$, as well as reward and KL divergence estimates (see Figures 1 and 7–9).

We report results on models pretrained on the image datasets MNIST (LeCun and Cortes, 1998), CIFAR-10 (Krizhevsky, 2009), and LSUN (bedrooms) (Yu et al., 2016). For MNIST, we use a 2.5M parameters model (no label conditioning). Our reward is the average brightness (i.e. average of all pixel values). For CIFAR-10 and LSUN, we pretrain a 53.2M parameters model,

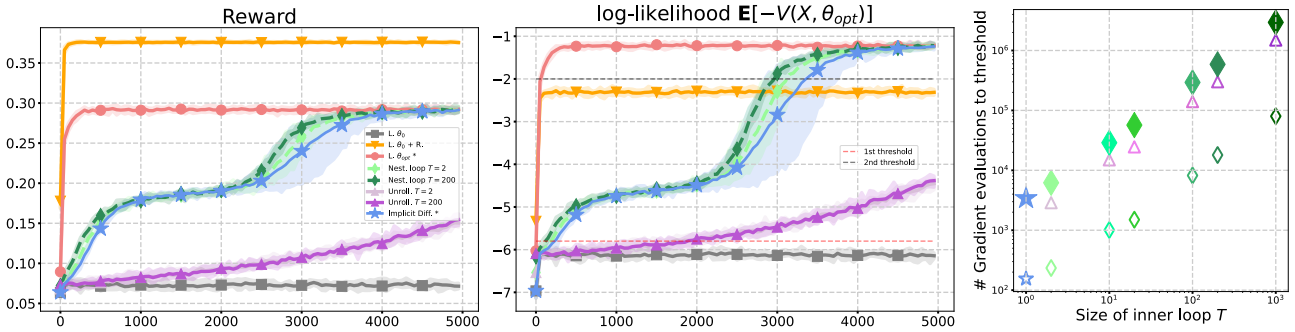


Figure 6: Metrics for reward training of Langevin processes, 10 runs. The setting and color code are detailed in Section 5.1. **Left:** Reward on the sample distribution, at each outer objective step, averaged on a batch. **Middle:** Log-likelihood of $\pi^\star(\theta_{\text{opt}})$ on the sample distribution, at each outer step, averaged on a batch–higher is better. **Right:** Number of gradient evaluations needed to reach a given log-likelihood threshold (y-axis)–lower is better, for various sizes of inner loop T (x-axis) and various methods (same symbols and colors as in left plots). Implicit Diffusion is $T = 1$. White symbols with colored edges correspond to a log-likelihood threshold of -5.8 (red dashed line in the middle plot) and fully-colored symbols to a threshold of -2.0 (black dashed line in the middle plot).

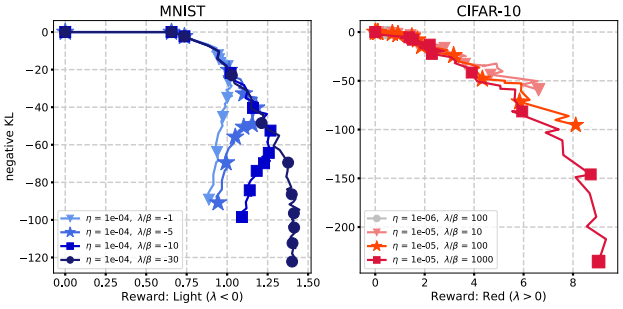


Figure 7: Reward training with **Implicit Diffusion** for various learning rates η and reward strengths λ/β. For each dataset, we plot together the reward and the negative KL divergence w.r.t. $\pi^\star(\theta_0)$.

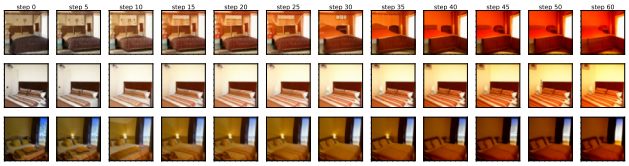


Figure 8: Samples of reward training after pretraining on LSUN ($\lambda/\beta = 10$). The reward incentives for redder images. Images are re-sampled with the same seed every five steps (see Appendix C.2).

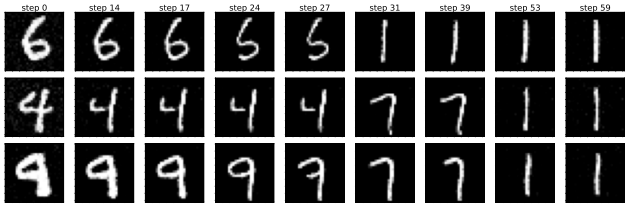


Figure 9: Samples of reward training after pretraining on MNIST. The reward favors **darker** images ($\lambda/\beta = -30$). Images are re-sampled with the same seed every five steps (see Appendix C.2).

with label conditioning for CIFAR-10. Our reward is the average brightness of the red channel minus the average on the other channels. For pretraining, we follow the simple diffusion method (Hoogeboom et al., 2023) and use U-Net models (Ronneberger et al., 2015). We display visual examples in Figures 1, 8, 9 and in Appendix C, where we also report additional metrics. While the finetuned models diverge from the original distribution, they retain overall semantic information (e.g. brighter digits are thicker, rather than on a gray background). We observe in Figure 7 the competition between reward and divergence to the pretrained distribution.

Possible limitations of our approach include sensitivity to the choice of hyperparameters (learning rate η, reward strength λ/β, size of the queue M), bias in the gradient estimation, practical applicability to larger scale problems or more complex rewards. We plan to investigate these questions in future research.

Acknowledgments

The authors would like to thank Fabian Pedregosa for very fruitful discussions on implicit differentiation and bilevel optimization that led to this project, Vincent Roulet for very insightful notes and comments about early drafts of this work as well as help with experiment implementation, Emiel Hoogeboom for extensive help

on pretraining diffusion models, and Clément Crépy for help with open-sourcing. PM and AK thank Google for their academic support in the form respectively of a Google PhD Fellowship and a gift in support of her academic research.

References

L. Ambrosio, N. Gigli, and G. Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2005.

B. D. O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

M. Arbel and J. Mairal. Amortized implicit differentiation for stochastic bilevel optimization. In *International Conference on Learning Representations*, 2022.

Y. F. Atchadé, G. Fort, and E. Moulines. On perturbed proximal gradient algorithms. *The Journal of Machine Learning Research*, 18(1):310–342, 2017.

D. Bakry, I. Gentil, M. Ledoux, et al. *Analysis and geometry of Markov diffusion operators*, volume 103. Springer, 2014.

K. Black, M. Janner, Y. Du, I. Kostrikov, and S. Levine. Training diffusion models with reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

M. Blondel, Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-López, F. Pedregosa, and J.-P. Vert. Efficient and modular implicit differentiation. *Advances in neural information processing systems*, 35:5230–5242, 2022.

J. Bolte, E. Pauwels, and S. Vaiter. One-step differentiation of iterative algorithms. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 77089–77103. Curran Associates, Inc., 2023.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

L.-P. Chaintron and A. Diez. Propagation of chaos: A review of models, methods and applications. i. models and methods. *Kinetic and Related Models*, 15(6):895–1015, 2022.

H.-B. Chen, S. Chewi, and J. Niles-Weed. Dimension-free log-sobolev inequalities for mixture distributions. *Journal of Functional Analysis*, 281(11): 109236, 2021a.

T. Chen, Y. Sun, and W. Yin. Closing the gap: Tighter analysis of alternating stochastic gradient methods for bilevel problems. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 25294–25307. Curran Associates, Inc., 2021b.

T. Chen, Y. Sun, Q. Xiao, and W. Yin. A single-timescale method for stochastic bilevel optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 2466–2488. PMLR, 2022.

X. Cheng and P. L. Bartlett. Convergence of Langevin MCMC in KL-divergence. In F. Janoos, M. Mohri, and K. Sridharan, editors, *Proceedings of ALT2018*, volume 83 of *Proceedings of Machine Learning Research*, pages 186–211. PMLR, 2018.

K. Clark, P. Vicol, K. Swersky, and D. Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *The Twelfth International Conference on Learning Representations*, 2024.

M. Dagréou, P. Ablin, S. Vaiter, and T. Moreau. A framework for bilevel optimization that enables stochastic and global variance reduction algorithms. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 26698–26710. Curran Associates, Inc., 2022.

A. S. Dalalyan. Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 79(3):651–676, 2017.

V. De Bortoli, A. Durmus, M. Pereyra, and A. F. Vidal. Efficient stochastic optimisation by unadjusted langevin monte carlo: Application to maximum marginal likelihood and empirical bayesian estimation. *Statistics and Computing*, 31:1–18, 2021.

P. Dhariwal and A. Nichol. Diffusion models beat GANs on image synthesis. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc., 2021.

H. Dong, W. Xiong, D. Goyal, Y. Zhang, W. Chow, R. Pan, S. Diao, J. Zhang, K. SHUM, and T. Zhang. RAFT: Reward ranked finetuning for generative foundation model alignment. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

K. D. Dvijotham, S. Omidshafiei, K. Lee, K. M. Collins, D. Ramachandran, A. Weller, M. Ghavamzadeh, M. Nasr, Y. Fan, and J. Z. Liu. Algorithms for optimal adaptation of diffusion models to reward functions. In *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, 2023.

A. Eberle. Reflection couplings and contraction rates for diffusions. *Probability theory and related fields*, 166:851–886, 2016.

Y. Fan, O. Watkins, Y. Du, H. Liu, M. Ryu, C. Boutilier, P. Abbeel, M. Ghavamzadeh, K. Lee, and K. Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *arXiv preprint arXiv:2305.16381*, 2023.

L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1568–1577. PMLR, 10–15 Jul 2018.

M. C. Fu and J.-Q. Hu. *Conditional Monte Carlo: Gradient estimation and Optimization Applications*, volume 392. Springer Science & Business Media, 2012.

A. Graikos, N. Malkin, N. Jojic, and D. Samaras. Diffusion models as plug-and-play priors. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 14715–14728. Curran Associates, Inc., 2022.

A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

Z. Guo, Y. Xu, W. Yin, R. Jin, and T. Yang. A novel convergence analysis for algorithms of the adam family and beyond. *arXiv preprint arXiv:2104.14840*, 2021.

M. U. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of machine learning research*, 13(2), 2012.

A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-or. Prompt-to-prompt image editing with cross-attention control. In *The Eleventh International Conference on Learning Representations*, 2023.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

M. Hong, H.-T. Wai, Z. Wang, and Z. Yang. A two-timescale stochastic algorithm framework for bilevel optimization: Complexity analysis and application to actor-critic. *SIAM Journal on Optimization*, 33 (1):147–180, 2023.

E. Hoogeboom, J. Heek, and T. Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *Proceedings of The 40th International Conference on Machine Learning*, 2023.

A. Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.

R. Jordan, D. Kinderlehrer, and F. Otto. The variational formulation of the fokker–planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

P. Kidger. On neural differential equations. *arXiv preprint arXiv:2202.02435*, 2022.

P. Kidger, J. Foster, X. C. Li, and T. Lyons. Efficient and accurate gradients for neural SDEs. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18747–18761. Curran Associates, Inc., 2021.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

I. Kobyzev, S. Prince, and M. A. Brubaker. Normalizing flows: Introduction and ideas. *stat*, 1050:25, 2019.

A. Korba and A. Salim. Sampling as first-order optimization over a space of probability measures, 2022. Tutorial at ICML 2022. Accessible at `https://akorba.github.io/resources/Baltimore_July2022_ICMLtutorial.pdf`, consulted on 01/30/2024.

S. G. Krantz and H. R. Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

J. Kuntz, J. N. Lim, and A. M. Johansen. Particle algorithms for maximum likelihood training of latent variable models. In *International Conference on Artificial Intelligence and Statistics*, pages 5134–5180. PMLR, 2023.

M. Kwon, J. Jeong, and Y. Uh. Diffusion models already have a semantic latent space. In *The Eleventh International Conference on Learning Representations*, 2023.

Y. LeCun and C. Cortes. MNIST handwritten digit database, 1998. URL `http://yann.lecun.com/exdb/mnist/`.

K. Lee, H. Liu, M. Ryu, O. Watkins, Y. Du, C. Boutilier, P. Abbeel, M. Ghavamzadeh, and S. S. Gu. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023.

X. Li, T.-K. L. Wong, R. T. Q. Chen, and D. K. Duvenaud. Scalable gradients and variational inference for stochastic differential equations. In C. Zhang, F. Ruiz, T. Bui, A. B. Dieng, and D. Liang, editors, *Proceedings of The 2nd Symposium on Advances in Approximate Bayesian Inference*, volume 118 of *Proceedings of Machine Learning Research*, pages 1–28. PMLR, 08 Dec 2020.

H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

P. Marion and R. Berthier. Leveraging the two timescale regime to demonstrate convergence of neural networks. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

P. Mohajerin Esfahani and D. Kuhn. Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming*, 171(1): 115–166, 2018.

A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 18–24 Jul 2021.

A. Nitanda. Stochastic proximal gradient descent with acceleration techniques. *Advances in Neural Information Processing Systems*, 27, 2014.

B. G. Pachpatte and W. Ames. *Inequalities for Differential and Integral Equations*. Elsevier, 1997.

G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22(1):2617–2680, 2021.

G. A. Pavliotis. *Stochastic processes and applications*. Springer, 2016.

F. Pedregosa. Hyperparameter optimization with approximate gradient. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 737–746, New York, New York, USA, 20–22 Jun 2016. PMLR.

W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4195–4205, October 2023.

G. C. Pflug. *Optimization of Stochastic Models: the Interface between Simulation and Optimization*, volume 373. Springer Science & Business Media, 2012.

L. S. Pontryagin. *Mathematical Theory of Optimal Processes*. Routledge, 1987.

P. Protter. *Stochastic integration and differential equations. A new approach*, volume 21 of *Stochastic Modelling and Applied Probability*. Springer Berlin, Heidelberg, 2005.

G. O. Roberts and R. L. Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, pages 341–363, 1996.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

L. Rosasco, S. Villa, and B. C. Vũ. Convergence of stochastic proximal gradient algorithm. *Applied Mathematics & Optimization*, 82:891–917, 2020.

L. Sharrock, D. Dodd, and C. Nemeth. Tuning-free maximum likelihood training of latent variable models via coin betting. In S. Dasgupta, S. Mandt, and Y. Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 1810–1818. PMLR, 02–04 May 2024.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, 1999.

T. Suzuki, A. Nitanda, and D. Wu. Uniform-in-time propagation of chaos for the mean-field gradient langevin dynamics. In *The Eleventh International Conference on Learning Representations*, 2023.

V. B. Tadić and A. Doucet. Asymptotic bias of stochastic gradient search. *Annals of Applied Probability*, 27 (6):3255–3304, 2017.

A. Tsybakov. *Introduction to nonparametric estimation*. Springer Series in Statistics. Springer, New York, 2009.

B. Tzen and M. Raginsky. Theoretical guarantees for sampling and inference in generative models with latent diffusions. In A. Beygelzimer and D. Hsu, editors, *Proceedings of the Thirty-Second Conference*

on Learning Theory, volume 99 of *Proceedings of Machine Learning Research*, pages 3084–3114. PMLR, 25–28 Jun 2019.

S. Vempala and A. Wibisono. Rapid convergence of the unadjusted Langevin algorithm: Isoperimetry suffices. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

P. Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7): 1661–1674, 2011.

B. Wallace, A. Gokul, S. Ermon, and N. Naik. End-to-end diffusion latent optimization improves classifier guidance. *arXiv preprint arXiv:2303.13703*, 2023.

Z. Wang and J. Sirignano. A forward propagation algorithm for online optimization of nonlinear stochastic differential equations. *arXiv preprint arXiv:2207.04496*, 2022.

Z. Wang and J. Sirignano. Continuous-time stochastic gradient descent for optimizing over the stationary distribution of stochastic differential equations. *Mathematical Finance*, 34(2):348–424, 2024.

D. Watson, W. Chan, J. Ho, and M. Norouzi. Learning fast samplers for diffusion models by differentiating through sample quality. In *International Conference on Learning Representations*, 2022.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Q. Wu, Y. Liu, H. Zhao, A. Kale, T. Bui, T. Yu, Z. Lin, Y. Zhang, and S. Chang. Uncovering the disentanglement capability in text-to-image diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1900–1910, June 2023a.

X. Wu, K. Sun, F. Zhu, R. Zhao, and H. Li. Human preference score: Better aligning text-to-image models with human preference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2096–2105, October 2023b.

L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

J. Yang, K. Ji, and Y. Liang. Provably faster algorithms for bilevel optimization. *Advances in Neural Information Processing Systems*, 34:13670–13682, 2021.

F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2016.

Z. Zhang, L. Liu, Z. Lin, Y. Zhu, and Z. Zhao. Unsupervised discovery of interpretable directions in h-space of pre-trained diffusion models. *arXiv preprint arXiv:2310.09912*, 2023.

D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. **Yes**
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. **Yes**
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. **No**

   The setting and algorithms are described in Sections 1–3, and further details are given in Section A. We open-sourced the source code related to the experiments on reward training of Langevin processes.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. **Yes**
   (b) Complete proofs of all theoretical results. **Yes**
   (c) Clear explanations of any assumptions. **Yes**

   See Section 4 for precise mathematical statements and assumptions, and Appendix B for proofs.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). **No**
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). **Yes**
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). **Yes**
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). **Yes**

We open-sourced the source code related to the experiments on reward training of Langevin processes. Error bars are provided over independent repetitions for Langevin experiments, as described in Section 5 and Appendix C, while they are too costly to compute for the denoising diffusion experiments. The computing infrastructure is described in Section 5 and Appendix C.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. **Yes**
   (b) The license information of the assets, if applicable. **No**
   (c) New assets either in the supplemental material or as a URL, if applicable. **Not Applicable**
   (d) Information about consent from data providers/curators. **Not Applicable**
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. **Not Applicable**

   See Section 5 and Appendix C for citations of the datasets and main code packages used in this project.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. **Not Applicable**
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. **Not Applicable**
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. **Not Applicable**

APPENDIX

Organization of the Appendix. Section A is devoted to explanations of our methodology. Section A.1 explains the gradient estimation setting we consider. Then, in the case of Langevin dynamics (Section A.2), and denoising diffusions (Section A.3), we explain how Definition 3.1 and our Implicit Differentiation algorithms (Algorithms 2 and 3) can be instantiated. Section A.4 gives more details about the implicit differentiation approaches sketched in Section 3.2. Section B contains the proofs of our theoretical results, while Section C gives details for the experiments of Section 5 as well as additional explanations and plots. Finally, Section D is dedicated to additional related work.

A IMPLICIT DIFFUSION ALGORITHMS

A.1 Gradient estimation abstraction: Γ

As discussed in Section 3.1, we focus on settings where the gradient of the loss $\ell : \mathbb{R}^p \to \mathbb{R}$, defined by

$$\ell(\theta) := \mathcal{F}(\pi^\star(\theta))\,,$$

can be estimated by using a function Γ. More precisely, following Definition 3.1, we assume that there exists a function $\Gamma : \mathcal{P} \times \mathbb{R}^p \to \mathbb{R}^p$ such that $\nabla\ell(\theta) = \Gamma(\pi^\star(\theta), \theta)$. In practice, for almost every setting there is no closed form for $\pi^\star(\theta)$, and even sampling from it can be challenging (e.g. here, if it is the outcome of infinitely many sampling steps). When we run our algorithms, the dynamic is in practice applied to variables, as discussed in Section 2.1. Using a batch of variables of size n, initialized independent with $X_0^i \sim p_0$, we have at each step k of joint sampling and optimization a batch of variables forming an empirical measure $\hat{p}_k^{(n)}$. We consider cases where the operator Γ is well-behaved: if $\hat{p}^{(n)} \approx p$, then $\Gamma(\hat{p}^{(n)}, \theta) \approx \Gamma(p, \theta)$, and therefore where this finite sample approximation can be used to produce an accurate estimate of $\nabla\ell(\theta)$.

A.2 Langevin dynamics

We explain how to derive the formulas (9)–(10) for Γ, and give the sample version of Algorithm 2 in these cases.

Recall that the stationary distribution of the dynamics (5) is the Gibbs distribution (6), with the normalization factor $Z_\theta = \int \exp(-V(x,\theta))\mathrm{d}x$. Assume that the outer objective can be written as the expectation of some (potentially non-differentiable) reward R, namely $\mathcal{F}(p) := -\mathbb{E}_{x\sim p}[R(x)]$. Then our objective is

$$\ell_{\mathrm{rew}}(\theta) = -\int R(x)\frac{\exp(-V(x,\theta))}{Z_\theta}\mathrm{d}x\,.$$

As a consequence,

$$\begin{aligned}
\nabla\ell_{\mathrm{rew}}(\theta) &= \int R(x)\nabla_2 V(x,\theta)\frac{\exp(-V(x,\theta))}{Z_\theta}\mathrm{d}x \\
&\quad - \int R(x)\frac{\exp(-V(x,\theta))\int \nabla_2 V(x',\theta)\exp(-V(x',\theta))\mathrm{d}x'}{Z_\theta^2}\mathrm{d}x \\
&= \mathbb{E}_{X\sim\pi^\star(\theta)}[R(X)\nabla_2 V(X,\theta)] - \mathbb{E}_{X\sim\pi^\star(\theta)}[R(X)]\mathbb{E}_{X\sim\pi^\star(\theta)}[\nabla_2 V(X,\theta)] \\
&= \mathrm{Cov}_{X\sim\pi^\star(\theta)}[R(X), \nabla_2 V(X,\theta)]\,.
\end{aligned}$$

This computation is sometimes referred to as the REINFORCE trick (Williams, 1992). This suggests taking

$$\Gamma_{\mathrm{rew}}(p,\theta) = \mathrm{Cov}_{X\sim p}[R(X), \nabla_2 V(X,\theta)]\,.$$

In this case, the sample version of Algorithm 2 is

$$\begin{aligned}
X_{k+1}^{(i)} &= X_k^{(i)} - \gamma_X \nabla_1 V(X_k^{(i)}, \theta_k) + \sqrt{2\gamma_X}\Delta B_k^{(i)}\,, \quad \text{for all } i \in \{1,\dots,n\} \\
\theta_{k+1} &= \theta_k - \gamma_\theta \hat{\mathrm{Cov}}[R(X_k^{(i)}), \nabla_2 V(X_k^{(i)}, \theta_k)]\,,
\end{aligned}$$

where $(\Delta B_k)_{k\geq 0}$ are i.i.d. standard Gaussian random variables and $\hat{\text{Cov}}$ is the empirical covariance over the sample.

When $\mathcal{F}(p) := \text{KL}(p_{\text{ref}} \,|\, p)$, e.g., when we want to regularize towards a reference distribution p_{ref} with sample access, for $\ell_{\text{ref}}(\theta) = \mathcal{F}(\pi^\star(\theta))$, we have

$$\ell_{\text{ref}}(\theta) = \int \log\Big(\frac{p_{\text{ref}}[x]}{\pi^\star(\theta)[x]}\Big) p_{\text{ref}}[x]\mathrm{d}x\,,$$

thus

$$\nabla \ell_{\text{ref}}(\theta) = -\int \frac{\partial \pi^\star(\theta)}{\partial \theta}[x] \cdot \frac{1}{\pi^\star(\theta)[x]} p_{\text{ref}}[x]\mathrm{d}x\,.$$

Leveraging the explicit formula (6) for $\pi^\star(\theta)$, we obtain

$$\begin{aligned}
\nabla \ell_{\text{ref}}(\theta) &= \int \frac{\nabla_2 V(x,\theta)\exp(-V(x,\theta))}{\pi^\star(\theta)[x] Z_\theta} p_{\text{ref}}[x]\mathrm{d}x + \int \frac{\exp(-V(x,\theta))\nabla_\theta Z_\theta}{\pi^\star(\theta)[x] Z_\theta^2} p_{\text{ref}}[x]\mathrm{d}x \\
&= \int \nabla_2 V(x,\theta) p_{\text{ref}}[x]\mathrm{d}x + \int \frac{\nabla_\theta Z_\theta}{Z_\theta} p_{\text{ref}}[x]\mathrm{d}x \\
&= \mathbb{E}_{X\sim p_{\text{ref}}}[\nabla_2 V(X,\theta)] - \int \nabla_2 V(x,\theta)\frac{\exp(-V(x,\theta))}{Z_\theta}\mathrm{d}x \\
&= \mathbb{E}_{X\sim p_{\text{ref}}}[\nabla_2 V(X,\theta)] - \mathbb{E}_{X\sim \pi^\star(\theta)}[\nabla_2 V(X,\theta)]\,,
\end{aligned}$$

where the third equality uses that $\int p_{\text{ref}}[x]\mathrm{d}x = 1$. This suggests taking

$$\Gamma_{\text{ref}}(p,\theta) = \mathbb{E}_{X\sim p}[\nabla_2 V(X,\theta)] - \mathbb{E}_{X\sim \pi^\star(\theta)}[\nabla_2 V(X,\theta)]\,.$$

The terms in this gradient can be estimated: for a model $V(\cdot,\theta)$, the gradient function w.r.t θ can be obtained by automatic differentiation. Samples from p_{ref} are available by assumption, and samples from $\pi^\star(\theta)$ can be replaced by the X_k in joint optimization as above. We recover the formula for contrastive learning of energy-based model to data from p_{ref} (Gutmann and Hyvärinen, 2012).

This can also be used for finetuning, combining a reward R and a KL term, with $\mathcal{F}(p) = -\lambda \mathbb{E}_{X\sim p}[R(x)] + \beta \text{KL}(p \,||\, p_{\text{ref}})$. The sample version of Algorithm 2 then can be written

$$\begin{aligned}
X_{k+1}^{(i)} &= X_k^{(i)} - \gamma_X \nabla_1 V(X_k^{(i)},\theta_k) + \sqrt{2\gamma_X}\Delta B_k^{(i)} \quad \text{for all } i \in \{1,\dots,n\} \\
\theta_{k+1} &= \theta_k - \gamma_\theta\Big[\hat{\lambda}\,\hat{\text{Cov}}[R(X_k), \nabla_2 V(X_k,\theta_k)] + \beta\Big(\sum_{j=1}^m \nabla_2 V(\tilde{X}_k^{(j)},\theta) - \frac{1}{n}\sum_{i=1}^n \nabla_2 V(X_k^{(i)},\theta)\Big)\Big],
\end{aligned}$$

where $(\Delta B_k)_{k\geq 0}$ are i.i.d. standard Gaussian random variables, $\hat{\text{Cov}}$ is the empirical covariance over the sample, and $\tilde{X}_k^{(j)} \sim p_{\text{ref}}$.

A.3 Adjoint method and denoising diffusions

We explain how to use the adjoint method to backpropagate through differential equations, and apply this to derive instantiations of Algorithm 3 for denoising diffusions.

ODE sampling. We begin by recalling the adjoint method in the ODE case (Pontryagin, 1987). Consider the ODE $\mathrm{d}Y_t = \mu(t, Y_t, \theta)\mathrm{d}t$ integrated between 0 and some $T > 0$. For some differentiable function $R : \mathbb{R}^d \to \mathbb{R}$, the derivative of $R(Y_T)$ with respect to θ can be computed by the adjoint method. More precisely, it is equal to G_T defined by

$$\begin{aligned}
Z_0 &= Y_T\,, & \mathrm{d}Z_t &= -\mu(t, Z_t, \theta)\mathrm{d}t\,, \\
A_0 &= \nabla R(Y_T)\,, & \mathrm{d}A_t &= A_t^\intercal \nabla_2 \mu(T-t, Z_t, \theta)\mathrm{d}t\,, \\
G_0 &= 0\,, & \mathrm{d}G_t &= A_t^\intercal \nabla_3 \mu(T-t, Z_t, \theta)\mathrm{d}t\,.
\end{aligned}$$

Note that sometimes the adjoint equations are written with a reversed time index ($t' = T - t$), which is not the formalism we adopt here.

In the setting of denoising diffusion presented in Section 2.2, we are not interested in computing the derivative of a function of a single realization of the ODE, but of the expectation over $Y_T \sim \pi^\star(\theta)$ of the derivative of $R(Y_T)$ with respect to θ. In other words, we want to compute $\nabla \ell(\theta) = \nabla(\mathcal{F} \circ \pi^\star)(\theta)$, where $\mathcal{F}(p) = \mathbb{E}_{x\sim p}[R(x)]$. Rewriting the equations above in this case, we obtain that G_T defined by

$$
\begin{aligned}
&Z_0 \sim \pi^\star(\theta)\,, && \mathrm{d}Z_t = -\mu(t, Z_t, \theta)\mathrm{d}t\,,\\
&A_0 = \nabla R(Z_0)\,, && \mathrm{d}A_t = A_t^\intercal \nabla_2 \mu(T-t, Z_t, \theta)\mathrm{d}t\,,\\
&G_0 = 0\,, && \mathrm{d}G_t = A_t^\intercal \nabla_3 \mu(T-t, Z_t, \theta)\mathrm{d}t
\end{aligned}
$$

is an unbiased estimator of $\nabla \ell(\theta)$. Recalling Definition 3.1, this means that we can take $\Gamma(p, \theta) := G_T$ defined by

$$
\begin{aligned}
&Z_0 \sim p, && \mathrm{d}Z_t = -\mu(t, Z_t, \theta)\mathrm{d}t\\
&A_0 = \nabla R(Z_0), && \mathrm{d}A_t = A_t^\intercal \nabla_2 \mu(T-t, Z_t, \theta)\mathrm{d}t\\
&G_0 = 0, && \mathrm{d}G_t = A_t^\intercal \nabla_3 \mu(T-t, Z_t, \theta)\mathrm{d}t\,.
\end{aligned}
$$

This is exactly the definition of Γ given in Section 3.2. We apply this to the case of denoising diffusions, where is μ given by (8). To avoid a notation clash between the number of iterations $T = M$ of the sampling algorithm, and the maximum time T of the ODE in (8), we rename the latter to T_{horizon}. A direct instantiation of Algorithm 3 with an Euler solver is the following algorithm.

Algorithm 4 Implicit Diff. optimization, denoising diffusions with ODE sampling

input $\theta_0 \in \mathbb{R}^p$, $p_0 \in \mathcal{P}$
input $P_M = [Y_0^{(0)}, \dots, Y_0^{(M)}] \sim \mathcal{N}(0,1)^{\otimes(m\times d)}$
  for $k \in \{0, \dots, K-1\}$ **do**
    $Y_{k+1}^{(0)} \sim \mathcal{N}(0,1)$
    parallel $Y_{k+1}^{(m+1)} \leftarrow Y_k^{(m)} + \frac{1}{M}\mu(\frac{mT_{\text{horizon}}}{M}, Y_k^{(m)}, \theta_k)$ for $m \in [M-1]$
    $Z_k^{(0)} \leftarrow Y_k^{(M)}$
    $A_k^{(0)} \leftarrow \nabla R(Z_k^{(0)})$
    $G_k^{(0)} \leftarrow 0$
    for $t \in \{0, \dots, T-1\}$ **do**
      $Z_k^{(t+1)} \leftarrow Z_k^{(t)} - \frac{1}{T}\mu(\frac{tT_{\text{horizon}}}{T}, Z_k^{(t)}, \theta_k)$
      $A_k^{(t+1)} \leftarrow A_k^{(t)} + \frac{1}{T}(A_k^{(t)})^\intercal \nabla_2 \mu(\frac{tT_{\text{horizon}}}{T}, Z_k^{(t)}, \theta_k)$
      $G_k^{(t+1)} \leftarrow G_k^{(t)} + \frac{1}{T}(G_k^{(t)})^\intercal \nabla_2 \mu(\frac{tT_{\text{horizon}}}{T}, Z_k^{(t)}, \theta_k)$
    $\theta_{k+1} \leftarrow \theta_k - \eta G_k^{(T)}$
output θ_K

Several comments are in order. First, the dynamics of $Y_k^{(M)}$ in the previous M steps, from $Y_{k-M}^{(0)}$ to $Y_{k-1}^{(M-1)}$, uses the M previous values of the parameter $\theta_{k-M}, \dots, \theta_{k-1}$. This means that $Y_k^{(M)}$ does not correspond to the result of sampling with any given parameter θ, since we are at the same time performing the sampling process and updating θ.

Besides, the computation of $\Gamma(p, \theta)$ is the outcome of an iterative process, namely calling an ODE solver. Therefore, it is also possible to use the same queuing trick as for sampling iterations to decrease the cost of this step by leveraging parallelization. For completeness, the variant is given below.

Algorithm 5 Implicit Diff. optimization, denoising diffusions with ODE sampling, variant with a double queue

input $\theta_0 \in \mathbb{R}^p$, $p_0 \in \mathcal{P}$
input $P_M = [Y_0^{(0)}, \dots, Y_0^{(M)}] \sim \mathcal{N}(0,1)^{\otimes(m\times d)}$
  for $k \in \{0, \dots, K-1\}$ (joint single loop) **do**
    $Y_{k+1}^{(0)} \sim \mathcal{N}(0,1)$
    $Z_{k+1}^{(0)} \leftarrow Y_k^{(M)}$
    $A_{k+1}^{(0)} \leftarrow \nabla R(Z_{k+1}^{(0)})$
    $G_{k+1}^{(0)} \leftarrow 0$
    parallel $Y_{k+1}^{(m+1)} \leftarrow Y_k^{(m)} + \frac{1}{M}\mu(\frac{mT_{\text{horizon}}}{M}, Y_k^{(m)}, \theta_k)$ for $m \in [M-1]$
    parallel $Z_{k+1}^{(m+1)} \leftarrow Z_k^{(m)} - \frac{1}{M}\mu(\frac{mT_{\text{horizon}}}{M}, Z_k^{(m)}, \theta_k)$ for $m \in [M-1]$
    parallel $A_{k+1}^{(m+1)} \leftarrow A_k^{(m)} + \frac{1}{M}(A_k^{(m)})^\intercal \nabla_2\mu(\frac{mT_{\text{horizon}}}{M}, Z_k^{(m)}, \theta_k)$ for $m \in [M-1]$
    parallel $G_{k+1}^{(m+1)} \leftarrow G_k^{(m)} + \frac{1}{M}(G_k^{(m)})^\intercal \nabla_2\mu(\frac{mT_{\text{horizon}}}{M}, Z_k^{(m)}, \theta_k)$ for $m \in [M-1]$
    $\theta_{k+1} \leftarrow \theta_k - \eta G_k^{(M)}$
output θ_K

Second, each variable $Y_k^{(m)}$ consists of a single sample of $\mathbb{R}^d$. The algorithm straightforwardly extends when each variable $Y_k^{(m)}$ is a batch of samples. Finally, we consider so far the case where the size of the queue M is equal to the number of sampling steps T. We give below the variant of Algorithm 4 when $M \neq T$ but M divides T. Taking M from 1 to T balances between a single-loop and a nested-loop algorithm.

Algorithm 6 Implicit Diff. optimization, denoising diffusions with ODE sampling, $M \neq T$, M divides T

input $\theta_0 \in \mathbb{R}^p$, $p_0 \in \mathcal{P}$
input $P_M = [Y_0^{(0)}, \dots, Y_0^{(M)}] \sim \mathcal{N}(0,1)^{\otimes(m\times d)}$
  for $k \in \{0, \dots, K-1\}$ **do**
    $Y_{k+1}^{(0)} \sim \mathcal{N}(0,1)$
    parallel $Y_{k+1/2}^{(m+1)} \leftarrow Y_k^{(m)}$ for $m \in [M-1]$
    for $t \in \{0, \dots, T/M-1\}$ **in parallel for** $m \in [M-1]$ **do**
      $Y_{k+1/2}^{(m+1)} \leftarrow Y_{k+1/2}^{(m)} + \frac{1}{T}\mu((\frac{m}{M} + \frac{t}{T})T_{\text{horizon}}, Y_{k+1/2}^{(m)}, \theta_k)$
    parallel $Y_{k+1}^{(m+1)} \leftarrow Y_{k+1/2}^{(m+1)}$ for $m \in [M-1]$
    $Z_k^{(0)} \leftarrow Y_k^{(M)}$
    $A_k^{(0)} \leftarrow \nabla R(Z_k^{(0)})$
    $G_k^{(0)} \leftarrow 0$
    for $t \in \{0, \dots, T-1\}$ **do**
      $Z_k^{(t+1)} \leftarrow Z_k^{(t)} - \frac{1}{T}\mu(\frac{tT_{\text{horizon}}}{T}, Z_k^{(t)}, \theta_k)$
      $A_k^{(t+1)} \leftarrow A_k^{(t)} + \frac{1}{T}(A_k^{(t)})^\intercal \nabla_2\mu(\frac{tT_{\text{horizon}}}{T}, Z_k^{(t)}, \theta_k)$
      $G_k^{(t+1)} \leftarrow G_k^{(t)} + \frac{1}{T}(G_k^{(t)})^\intercal \nabla_2\mu(\frac{tT_{\text{horizon}}}{T}, Z_k^{(t)}, \theta_k)$
    $\theta_{k+1} \leftarrow \theta_k - \eta G_k^{(T)}$
output θ_K

Algorithm 5 extends to this case similarly.

SDE sampling. The adjoint method is also defined in the SDE case (Li et al., 2020). Consider the SDE

$$\mathrm{d}Y_t = \mu(t, Y_t, \theta)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t\,, \tag{13}$$

integrated between 0 and some $T > 0$. This setting encompasses the denoising diffusion SDE (7) with the appropriate choice of μ. For some differentiable function $R : \mathbb{R}^d \to \mathbb{R}$ and for a *given realization* $(Y_t)_{0\le t\le T}$ of the

SDE, the derivative of $R(Y_T)$ with respect to θ is equal to G_T defined by

$$\begin{aligned} A_0 &= \nabla R(Y_T)\,, & \mathrm{d}A_t &= A_t^\intercal \nabla_2 \mu(T-t, Y_{T-t}, \theta)\mathrm{d}t\,, \\ G_0 &= 0\,, & \mathrm{d}G_t &= A_t^\intercal \nabla_3 \mu(T-t, Y_{T-t}, \theta)\mathrm{d}t\,. \end{aligned} \tag{14}$$

This is a similar equation as in the ODE case. The main difference is that it is not possible to recover Y_{T-t} only from the terminal value of the path Y_T, but that we need to keep track of the randomness from the Brownian motion B_t. Efficient ways to do so are presented in Li et al. (2020); Kidger et al. (2021). In a nutshell, they consist in only keeping in memory the seed used to generate the Brownian motion, and recomputing the path from the seed.

Using the SDE sampler allows us to incorporate a KL term in the reward. Indeed, consider the SDE (13) for two different parameters θ_1 and θ_2, with associated variables Y_t^1 and Y_t^2. Then, by Girsanov's theorem (see Protter, 2005, Chapter III.8, and Tzen and Raginsky, 2019 for use in a similar context), the KL divergence between the paths Y_t^1 and Y_t^2 is

$$\mathrm{KL}((Y_t^1)_{t\geq 0} \,||\, (Y_t^2)_{t\geq 0}) = \int_0^T \mathbb{E}_{y\sim q_t^1} \|\mu(t,y,\theta_1) - \mu(t,y,\theta_2)\|^2 \mathrm{d}t\,, \tag{15}$$

where q_t^1 denotes the distribution of Y_t^1. This term can be (stochastically) estimated at the same time as the SDE (13) is simulated, by appending a new coordinate to Y_t (and to μ) that integrates (15) over time. Then, adding the KL in the reward is as simple as adding a linear term in $\tilde{R} : \mathbb{R}^{d+1} \to \mathbb{R}$, that is, $\tilde{R}(x) = R(x[:-1]) + x[-1]$ (using Numpy notation, where '-1' denotes the last index). The same idea is used in Dvijotham et al. (2023) to incorporate a KL term in reward finetuning of denoising diffusion models. Finally, note that, if we had at our disposal a reward R that indicates if an image is "close" to p_{data} (for instance implemented by a neural network), we could use our algorithm to train from scratch a denoising diffusion.

A.4 Implicit differentiation

Finite dimension. Take $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ a continuously differentiable function. Then $x^\star(\theta) = \operatorname{argmin} g(\cdot, \theta)$ implies a stationary point condition $\nabla_1 g(x^\star(\theta_0), \theta_0) = 0$. In this case, it is possible to define and analyze the function $x^\star : \mathbb{R}^p \to \mathbb{R}^m$ and its variations. Note that this generalizes to the case where $x^\star(\theta)$ can be written as the root of a parameterized system.

More precisely, the **implicit function theorem** (see, e.g., Griewank and Walther, 2008; Krantz and Parks, 2002, and references therein) can be applied. Under differentiability assumptions on g, for (x_0, θ_0) such that $\nabla_1 g(x_0, \theta_0) = 0$ with a continuously differentiable $\nabla_1 g$, and if the Hessian $\nabla_{1,1} g$ evaluated at (x_0, θ_0) is a square invertible matrix, then there exists a function $x^\star(\cdot)$ over a neighborhood of θ_0 satisfying $x^\star(\theta_0) = x_0$. Furthermore, for all θ in this neighborhood, we have that $\nabla_1 g(x^\star(\theta), \theta) = 0$ and its Jacobian $\partial x^\star(\theta)$ exists. It is then possible to differentiate with respect to θ both sides of the equation $\nabla_1 g(x^\star(\theta_0), \theta_0) = 0$, which yields a linear equation satisfied by this Jacobian

$$\nabla_{1,1} g(x^\star(\theta_0), \theta_0) \partial x^\star(\theta) + \nabla_{1,2} g(x^\star(\theta_0), \theta_0) = 0\,.$$

This formula can be used for automatic implicit differentiation, when both the evaluation of the derivatives in this equation and the inversion of the linear system can be done automatically Blondel et al. (2022).

Extension to space of probabilities. When $\mathcal{G} : \mathcal{P} \times \mathbb{R}^p \to \mathbb{R}$ and $\pi^\star(\theta) = \operatorname{argmin} \mathcal{G}(\cdot, \theta)$ as in (3), under assumptions on differentiability and uniqueness of the solution on $\mathcal{G}$, this can also be extended to a distribution setting. We write here the infinite-dimensional equivalent of the above equations, involving derivatives or variations over the space of probabilities, and refer to Ambrosio et al. (2005) for more details.

First, we have that

$$\nabla \ell(\theta) = \nabla_\theta\big(\mathcal{F}(\pi^\star(\theta))\big) = \int \mathcal{F}'(p)[x] \nabla_\theta \pi^\star(\theta)[x] \mathrm{d}x\,,$$

where $\mathcal{F}'(p) : \mathcal{X} \to \mathbb{R}$ denotes the first variation of $\mathcal{F}$ at $p \in \mathcal{P}$ (see Definition B.1). This yields $\nabla \ell(\theta) = \Gamma(\pi^\star(\theta), \theta)$ with

$$\Gamma(p, \theta) = \int \mathcal{F}'(p)[x] \gamma(p, \theta)[x] \mathrm{d}x\,.$$

where $\gamma(p, \theta)$ is the solution of the linear system

$$\int \nabla_{1,1}\mathcal{G}(p,\theta)[x,x']\gamma(p,\theta)[x']\mathrm{d}x' = -\nabla_{1,2}\mathcal{G}(p,\theta)[x]\,,$$

Although this gives us a general way to define gradients of $\pi^\star(\theta)$ with respect to θ, solving this linear system is generally not feasible. One exception is when sampling over a finite state space $\mathcal{X}$, in which case $\mathcal{P}$ is finite-dimensional, and the integrals boil down to matrix-vector products.

B THEORETICAL ANALYSIS

B.1 Langevin with continuous flow

B.1.1 Additional definitions

Notations. We denote by $\mathcal{P}_2(\mathbb{R}^d)$ the set of probability measures on $\mathbb{R}^d$ with bounded second moments. Given a Lebesgue measurable map $T : X \to X$ and $\mu \in \mathcal{P}_2(X)$, $T_\#\mu$ is the pushforward measure of μ by T. For any $\mu \in \mathcal{P}_2(\mathbb{R}^d)$, $L^2(\mu)$ is the space of functions $f : \mathbb{R}^d \to \mathbb{R}$ such that $\int \|f\|^2 d\mu < \infty$. We denote by $\|\cdot\|_{L^2(\mu)}$ and $\langle\cdot,\cdot\rangle_{L^2(\mu)}$ respectively the norm and the inner product of the Hilbert space $L^2(\mu)$. We consider, for $\mu, \nu \in \mathcal{P}_2(\mathbb{R}^d)$, the 2-Wasserstein distance $W_2(\mu,\nu) = \inf_{s\in\mathcal{S}(\mu,\nu)} \int \|x-y\|^2 ds(x,y)$, where $\mathcal{S}(\mu,\nu)$ is the set of couplings between μ and ν. The metric space $(\mathcal{P}_2(\mathbb{R}^d), W_2)$ is called the Wasserstein space.

Let $\mathcal{F} : \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}^+$ a functional.

Definition B.1. Fix $\nu \in \mathcal{P}(\mathbb{R}^d)$. If it exists, the *first variation of* $\mathcal{F}$ *at* ν is the function $\mathcal{F}'(\nu) : \mathbb{R}^d \to \mathbb{R}$ s. t. for any $\mu \in \mathcal{P}(\mathbb{R}^d)$, with $\xi = \mu - \nu$:

$$\lim_{\epsilon\to 0} \frac{1}{\epsilon}(\mathcal{F}(\nu + \epsilon\xi) - \mathcal{F}(\nu)) = \int_{\mathbb{R}^d} \mathcal{F}'(\nu)(x) d\xi(x),$$

and is defined uniquely up to an additive constant.

We will extensively apply the following formula:

$$\frac{\mathrm{d}\mathcal{F}(p_t)}{\mathrm{d}t} = \int \mathcal{F}'(p_t)\frac{\partial p_t}{\partial t} = \int \mathcal{F}'(p_t)[x]\frac{\partial p_t[x]}{\partial t}\mathrm{d}x. \tag{16}$$

We will also rely regularly on the definition of a Wasserstein gradient flow, since Langevin dynamics correspond to a Wasserstein gradient flow of the Kullback-Leibler (KL) divergence Jordan et al. (1998). A Wasserstein gradient flow of $\mathcal{F}$ Ambrosio et al. (2005) can be described by the following continuity equation:

$$\frac{\partial \mu_t}{\partial t} = \boldsymbol{\nabla} \cdot (\mu_t \nabla_{W_2}\mathcal{F}(\mu_t)), \quad \nabla_{W_2}\mathcal{F}(\mu_t) = \nabla\mathcal{F}'(\mu_t), \tag{17}$$

where $\mathcal{F}'$ denotes the first variation. Equation (17) holds in the sense of distributions (i.e. the equation above holds when integrated against a smooth function with compact support), see (Ambrosio et al., 2005, Chapter 8). In particular, if $\mathcal{F} = \mathrm{KL}(\cdot|\pi)$ for $\pi \in \mathcal{P}_2(\mathbb{R}^d)$, then $\nabla_{W_2}\mathcal{F}(\mu) = \nabla\log(\mu/\pi)$. In this case, the corresponding continuity equation is known as the Fokker-Planck equation, and in particular it is known that the law p_t of Langevin dynamics:

$$\mathrm{d}X_t = \nabla\log(\pi(X_t))\mathrm{d}t + \sqrt{2}\mathrm{d}B_t$$

satisfies the Fokker-Planck equation (Pavliotis, 2016, Chapter 3).

B.1.2 Gaussian mixtures satisfy the Assumptions

We begin by a more formal statement of the result alluded to in Section 4.1.

Proposition B.2. *Let*

$$V(x,\theta) := -\log\Big(\sum_{i=1}^{p} H(\theta_i)\exp\big(-\|x-z_i\|^2\big)\Big),$$

for some fixed $z_1, \dots, z_p \in \mathbb{R}^d$ and where

$$H(x) := \eta + (1-\eta) \cdot \frac{1}{1+e^{-x}}$$

is a shifted version of the logistic function for some $\eta \in (0,1)$. Then Assumptions 4.1 and 4.2 hold.

Proof. Assumption 4.1 holds since a mixture of Gaussians is Log-Sobolev with a bounded constant (Chen et al., 2021a, Corollary 1). Note that the constant deteriorates as the modes of the mixture get further apart.

Furthermore, Assumption 4.2 holds since, for all $\theta \in \mathbb{R}^p$ and $x \in \mathbb{R}^d$,

$$\|\nabla_2 V(x,\theta)\|_1 = \frac{\sum_{i=1}^p H'(\theta_i) \exp\big(-\|x-z_i\|^2\big)}{\sum_{i=1}^p H(\theta_i) \exp(-\|x-z_i\|^2)} \leq \frac{\sum_{i=1}^p \exp\big(-\|x-z_i\|^2\big)}{\sum_{i=1}^p \eta \exp(-\|x-z_i\|^2)} = \frac{1}{\eta}\,.$$

□

B.1.3 Proof of Proposition 4.4

In the case of the functions Γ defined by (9)–(10), we see that Γ is bounded under Assumption 4.2 and when the reward R is bounded. The Lipschitz continuity can be obtained as follows. Consider for instance the case of (10) where $\Gamma(p,\theta)$ is given by

$$\Gamma_{\mathrm{ref}}(p,\theta) = \mathbb{E}_{X\sim p_{\mathrm{ref}}}[\nabla_2 V(X,\theta)] - \mathbb{E}_{X\sim p}[\nabla_2 V(X,\theta)]\,.$$

Then

$$\begin{aligned}\|\Gamma_{\mathrm{ref}}(p,\theta) - \Gamma_{\mathrm{ref}}(q,\theta)\| &= \|\mathbb{E}_{X\sim q}[\nabla_2 V(X,\theta)] - \mathbb{E}_{X\sim p}[\nabla_2 V(X,\theta)]\| \\ &\leq C\,\mathrm{TV}(p,q) \\ &\leq \frac{C}{\sqrt{2}}\sqrt{\mathrm{KL}(p||q)}\,,\end{aligned}$$

where the first inequality comes from the fact that the total variation distance is an integral probability metric generated by the set of bounded functions, and the second inequality is Pinsker's inequality (Tsybakov, 2009, Lemma 2.5). The first case of Section A.2 unfolds similarly.

B.1.4 Proof of Theorem 4.5

The dynamics (11) can be rewritten equivalently on $\mathcal{P}_2(\mathbb{R}^d)$ and $\mathbb{R}^p$ as

$$\frac{\partial p_t}{\partial t} = \boldsymbol{\nabla} \cdot (p_t \nabla_{W_2} \mathcal{G}(p_t, \theta_t)) \tag{18}$$

$$\mathrm{d}\theta_t = -\varepsilon_t \Gamma(p_t, \theta_t)\mathrm{d}t\,, \tag{19}$$

where $\mathcal{G}(p,\theta) = \mathrm{KL}(p\,||\,\pi^*_\theta)$, see Appendix B.1.1. The Wasserstein gradient in (18) is taken with respect to the first variable of $\mathcal{G}$.

Evolution of the loss. Recall that Γ satisfies by Definition 3.1 that $\nabla\ell(\theta) = \Gamma(\pi^\star(\theta),\theta)$. Thus we have, by (19),

$$\begin{aligned}\frac{\mathrm{d}\ell}{\mathrm{d}t}(t) &= \Big\langle \nabla\ell(\theta_t), \frac{\mathrm{d}\theta_t}{\mathrm{d}t}\Big\rangle \\ &= -\varepsilon_t \langle \nabla\ell(\theta_t), \Gamma(p_t,\theta_t)\rangle \\ &= -\varepsilon_t \langle \nabla\ell(\theta_t), \Gamma(\pi^\star(\theta_t),\theta_t)\rangle + \varepsilon_t \langle \nabla\ell(\theta(t)), \Gamma(\pi^\star(\theta_t),\theta_t) - \Gamma(p_t,\theta_t)\rangle \\ &\leq -\varepsilon_t \|\nabla\ell(\theta_t)\|^2 + \varepsilon_t \|\nabla\ell(\theta_t)\| \|\Gamma(\pi^\star(\theta_t),\theta_t) - \Gamma(p_t,\theta_t)\|\,.\end{aligned}$$

Then, by Assumption 4.3,

$$\frac{\mathrm{d}\ell}{\mathrm{d}t}(t) \leq -\varepsilon_t \|\nabla\ell(\theta_t)\|^2 + \varepsilon_t K_\Gamma \|\nabla\ell(\theta_t)\| \sqrt{\mathrm{KL}(p_t\,||\,\pi^\star(\theta_t))}\,.$$

Using $ab \leq \frac{1}{2}(a^2+b^2)$, we get

$$\frac{\mathrm{d}\ell}{\mathrm{d}t}(t) \leq -\frac{1}{2}\varepsilon_t \|\nabla\ell(\theta_t)\|^2 + \frac{1}{2}\varepsilon_t K_\Gamma^2\, \mathrm{KL}(p_t\,||\,\pi^\star(\theta_t))\,, \tag{20}$$

Bounding the KL divergence of p_t from $\pi^*(\theta_t)$. Recall that

$$\mathrm{KL}(p_t \,||\, \pi^\star(\theta_t)) = \int \log\left(\frac{p_t}{\pi^\star(\theta_t)}\right) p_t \,.$$

Thus, by the chain rule formula (16),

$$\frac{\mathrm{d}\,\mathrm{KL}(p_t \,||\, \pi^\star(\theta_t))}{\mathrm{d}t} = \int \log\left(\frac{p_t}{\pi^\star(\theta_t)}\right) \frac{\partial p_t}{\partial t} - \int \frac{p_t}{\pi^\star(\theta_t)} \frac{\partial \pi^\star(\theta_t)}{\partial t} := a - b \,.$$

From an integration by parts, using (18) and by Assumption 4.1, we have

$$\begin{aligned} a = \int \log\left(\frac{p_t}{\pi^\star(\theta_t)}\right) \frac{\partial p_t}{\partial t} &= \int \log\left(\frac{p_t}{\pi^\star(\theta_t)}\right) \boldsymbol{\nabla} \cdot \left(p_t \nabla \log\left(\frac{p_t}{\pi^\star(\theta_t)}\right)\right) \\ &= \left\langle \nabla \log\left(\frac{p_t}{\pi^\star(\theta_t)}\right), -\nabla \log\left(\frac{p_t}{\pi^\star(\theta_t)}\right) \right\rangle_{L^2(p_t)} = -\left\| \nabla \log\left(\frac{p_t}{\pi^\star(\theta_t)}\right) \right\|^2_{L^2(p_t)} \\ &\leq -2\mu \,\mathrm{KL}(p_t \,||\, \pi^\star(\theta_t)) \,. \end{aligned}$$

Moving on to b, we have

$$b = \int \frac{p_t}{\pi^\star(\theta_t)} \frac{\partial \pi^\star(\theta_t)}{\partial t} = \int p_t \frac{\partial \log(\pi^\star(\theta_t))}{\partial t} \,.$$

By the chain rule and (19), we have for $x \in \mathcal{X}$

$$\frac{\partial \pi^\star(\theta_t)}{\partial t}[x] = \left\langle \frac{\partial \pi^\star(\theta_t)}{\partial \theta}[x], \frac{d\theta_t}{\mathrm{d}t} \right\rangle = \left\langle \frac{\partial \pi^\star(\theta_t)}{\partial \theta}[x], -\varepsilon_t \Gamma(p_t, \theta_t) \right\rangle \,.$$

Using $\pi^\star(\theta) \propto e^{-V(\theta,\cdot)}$ (with similar computations as for $\nabla \ell_{\mathrm{ref}}$ in Section A.2), we have

$$\frac{\partial \pi^\star(\theta_t)}{\partial t}[x] = \left\langle -\nabla_2 V(x, \theta_t) \pi^\star(\theta_t)[x] + \mathbb{E}_{X \sim \pi^\star(\theta_t)}(\nabla_2 V(X, \theta_t)) \pi^\star(\theta_t)[x], -\varepsilon_t \Gamma(p_t, \theta_t) \right\rangle ,$$

and

$$\frac{\partial \log(\pi^\star(\theta_t))}{\partial t} = \varepsilon_t \langle \nabla_2 V(\cdot, \theta_t) - \mathbb{E}_{X \sim \pi^\star(\theta_t)}(\nabla_2 V(X, \theta_t)), \Gamma(p_t, \theta_t) \rangle .$$

This yields

$$\begin{aligned} |b| &= \left| \varepsilon_t \int \langle \nabla_2 V(x, \theta_t) - \mathbb{E}_{X \sim \pi^\star(\theta_t)}(\nabla_2 V(X, \theta_t)), \Gamma(p_t, \theta_t) \rangle p_t[x] \mathrm{d}x \right| \\ &= \varepsilon_t \left| \left\langle \int \nabla_2 V(x, \theta_t) dp_t[x] - \mathbb{E}_{X \sim \pi^\star(\theta_t)}(\nabla_2 V(X, \theta_t)), \Gamma(p_t, \theta_t) \right\rangle \right| \\ &\leq \varepsilon_t \|\Gamma(p_t, \theta_t)\|_{\mathbb{R}^p} \left(\|\nabla_2 V(\cdot, \theta_t)\|_{L^2(p_t)} + \|\nabla_2 V(\cdot, \theta_t)\|_{L^2(\pi^\star(\theta_t))} \right) \\ &\leq 2C^2 \varepsilon_t \,, \end{aligned}$$

where the last step uses Assumptions 4.2 and 4.3. Putting everything together, we obtain

$$\frac{d\,\mathrm{KL}(p_t \,||\, \pi^\star(\theta_t))}{\mathrm{d}t} \leq -2\mu \,\mathrm{KL}(p_t \,||\, \pi^\star(\theta_t)) + 2C^2 \varepsilon_t .$$

Using Grönwall's inequality (Pachpatte and Ames, 1997) to integrate the inequality, the KL divergence can be bounded by

$$\mathrm{KL}(p_t \,||\, \pi^\star(\theta_t)) \leq \mathrm{KL}(p_0 \,||\, \pi^\star(\theta_0)) e^{-2\mu t} + 2C^2 \int_0^t \varepsilon_s e^{2\mu(s-t)} \mathrm{d}s \,.$$

Coming back to (20), we get

$$\frac{\mathrm{d}\ell}{\mathrm{d}t}(t) \leq -\frac{1}{2}\varepsilon_t \|\nabla \ell(\theta_t)\|^2 + \frac{1}{2}\varepsilon_t K_\Gamma^2 \Big(\mathrm{KL}(p_0 \,||\, \pi^\star(\theta_0)) e^{-2\mu t} + 2C^2 \int_0^t \varepsilon_s e^{2\mu(s-t)} \mathrm{d}s \Big) .$$

Integrating between 0 and T, we have

$$\ell(T) - \ell(0) \leq -\frac{1}{2}\int_0^T \varepsilon_t \|\nabla \ell(\theta_t)\|^2 \mathrm{d}t + \frac{K_\Gamma^2 \,\mathrm{KL}(p_0 \,||\, \pi^\star(\theta_0))}{2} \int_0^T \varepsilon_t e^{-2\mu t} \mathrm{d}t \\ + K_\Gamma^2 C^2 \int_0^T \int_0^t \varepsilon_t \varepsilon_s e^{2\mu(s-t)} \mathrm{d}s\mathrm{d}t \,.$$

Since ε_t is decreasing, we can bound ε_t by ε_T in the first integral and rearrange terms to obtain

$$\frac{1}{T}\int_0^T \|\nabla \ell(\theta_t)\|^2 \mathrm{d}t \leq \frac{2}{T\varepsilon_T}(\ell(0) - \inf \ell) + \frac{K_\Gamma^2 \,\mathrm{KL}(p_0 \,||\, \pi^\star(\theta_0))}{T\varepsilon_T} \int_0^T \varepsilon_t e^{-2\mu t} \mathrm{d}t \\ + \frac{2K_\Gamma^2 C^2}{T\varepsilon_T} \int_0^T \int_0^t \varepsilon_t \varepsilon_s e^{2\mu(s-t)} \mathrm{d}s\mathrm{d}t \,. \tag{21}$$

Recall that, by assumption of the Theorem, $\varepsilon_t = \min(1, \frac{1}{\sqrt{t}})$. Thus $T\varepsilon_T = \sqrt{T}$, and the first term is bounded by a constant times $T^{-1/2}$. It is also the case of the second term since $\int_0^T \varepsilon_t e^{-2\mu t} \mathrm{d}t$ is converging. Let us now estimate the magnitude of the last term. Let $T_0 \geq 2$ (depending only on μ) such that $\frac{\ln(T_0)}{2\mu} \leq \frac{T_0}{2}$. For $t \geq T_0$, let $\alpha(t) := t - \frac{\ln t}{2\mu}$. We have, for $t \geq T_0$,

$$\begin{aligned}
\int_0^t \varepsilon_s e^{2\mu(s-t)} ds &= \int_0^{\alpha(t)} \varepsilon_s e^{2\mu(s-t)} ds + \int_{\alpha(t)}^t \varepsilon_s e^{2\mu(s-t)} ds \\
&\leq \varepsilon_0 e^{-2\mu t} \int_0^{\alpha(t)} e^{2\mu s} ds + (t - \alpha(t)) \varepsilon_{\alpha(t)} \\
&\leq \frac{\varepsilon_0}{2\mu} e^{2\mu(\alpha(t)-t)} + \frac{\varepsilon_{\alpha(t)} \ln t}{2\mu} \\
&\leq \frac{\varepsilon_0}{2\mu t} + \frac{\varepsilon_{t/2} \ln t}{2\mu} \,,
\end{aligned}$$

where in the last inequality we used that $\alpha(t) \geq t/2$ and ε_t is decreasing. For $t < T_0$, we can simply bound the integral $\int_0^t \varepsilon_s e^{\mu(s-t)} ds$ by $\varepsilon_0 T_0$. We obtain

$$\int_0^T \int_0^t \varepsilon_t \varepsilon_s e^{2\mu(s-t)} \mathrm{d}s \leq \int_0^{T_0} \varepsilon_t \varepsilon_0 T_0 \mathrm{d}t + \int_{T_0}^T \frac{\varepsilon_t \varepsilon_0}{2\mu t} + \frac{\varepsilon_t \varepsilon_{t/2} \ln t}{2\mu} \mathrm{d}t \,.$$

Recall that $\varepsilon_t = \min(1, \frac{1}{\sqrt{t}})$, and that $T_0 \geq 2$. Thus

$$\int_0^T \int_0^t \varepsilon_t \varepsilon_s e^{2\mu(s-t)} \mathrm{d}s \leq \int_0^{T_0} \varepsilon_0 T_0 \mathrm{d}t + \frac{\varepsilon_0}{2\mu} \int_{T_0}^T \frac{\varepsilon_t}{t} \mathrm{d}t + \frac{\ln T}{2\mu} \int_2^T \varepsilon_t \varepsilon_{t/2} \mathrm{d}t \,.$$

The first two integrals are converging when $T \to \infty$ and the last integral is $\mathcal{O}(\ln T)$. Plugging this into (21), we finally obtain the existence of a constant $c > 0$ such that

$$\frac{1}{T}\int_0^T \|\nabla \ell(\theta_t)\|^2 \mathrm{d}t \leq \frac{c(\ln T)^2}{T^{1/2}} \,.$$

B.2 Langevin with discrete flow–proof of Theorem 4.7

We take

$$\gamma_k = \frac{1}{k^{1/3}} \min\Big(\frac{1}{L_X}, \frac{1}{\sqrt{L_\Theta}}, \frac{1}{\mu}, \frac{\mu}{4L_X^2}\Big) \quad \text{and} \quad \varepsilon_k = \frac{1}{k^{1/3}}. \tag{22}$$

Bounding the KL divergence of p_{k+1} from $\pi^\star(\theta_{k+1})$. Recall that p_k is the law of X_k. We leverage similar ideas to the proof of (Cheng and Bartlett, 2018, Theorem 3), exploiting the Log Sobolev inequality to bound the KL along one Langevin Monte Carlo iteration (an approach that was further streamlined in (Vempala and

Wibisono, 2019, Lemma 3)). The starting point is to notice that one Langevin Monte Carlo iteration can be equivalently written as a continuous-time process over a small time interval $[0, \gamma_k]$. More precisely, let

$$\rho_0 := p_k\,, \quad x_0 \sim \rho_0\,,$$

and x_t satisfying the SDE

$$\mathrm{d}x_t = -\nabla_1 V(x_0, \theta_k)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t\,.$$

Then, following the proof of (Cheng and Bartlett, 2018, Theorem 3), p_{k+1} has the same distribution as the output at time $\gamma := \gamma_k$ of the continuity equation

$$\frac{\partial \rho_t}{\partial t}[x] = \boldsymbol{\nabla} \cdot \big(\rho_t[x](\mathbb{E}_{\rho_{0|t}}[\nabla_1 V(x_k, \theta_k)|x_t = x] + \nabla \log \rho_t)\big)$$

where $\rho_{0|t}$ is the conditional distribution of x_0 given x_t. Similarly, θ_{k+1} is equal to the output at time γ of

$$\vartheta_t := \theta_k - t\varepsilon_k \Gamma(\mu_k, \theta_k). \tag{23}$$

We have:

$$\frac{d\,\mathrm{KL}(\rho_t \,||\, \pi^\star(\vartheta_t))}{\mathrm{d}t} = \int \log\left(\frac{\rho_t}{\pi^\star(\vartheta_t)}\right)\frac{\partial \rho_t}{\partial t} + \int \frac{\rho_t}{\pi^\star(\vartheta_t)}\frac{\partial \pi^\star(\vartheta_t)}{\partial t} := a + b.$$

We first bound b similarly to the proof of Theorem 4.5, under Assumptions 4.2 and 4.3:

$$b = \varepsilon_k \int \langle \nabla_2 V(x, \vartheta_t) - \mathbb{E}_{X\sim\pi^\star(\vartheta_t)}(\nabla_2 V(X, \vartheta_t)), \Gamma(\mu_k, \theta_k)\rangle p_t[x]\mathrm{d}x \le 2\varepsilon_k C^2.$$

Then we write a as

$$\begin{aligned}
a &= \int \log\Big(\frac{\rho_t[x]}{\pi^\star(\vartheta_t)[x]}\Big)\boldsymbol{\nabla} \cdot \big(\rho_t[x](\mathbb{E}_{\rho_{0|t}}[\nabla_1 V(x_0, \theta_k)|x_t = x] + \nabla \log \rho_t[x])\big)\mathrm{d}x \\
&= -\int \rho_t(x)\Big\langle \nabla \log\Big(\frac{\rho_t[x]}{\pi^\star(\vartheta_t)[x]}\Big), \mathbb{E}_{\rho_{0|t}}[\nabla_1 V(x_0, \theta_k)|x_t = x] + \nabla \log \rho_t[x]\Big\rangle \mathrm{d}x \\
&= -\int \rho_t(x)\Big\langle \nabla \log\Big(\frac{\rho_t[x]}{\pi^\star(\vartheta_t)[x]}\Big), \mathbb{E}_{\rho_{0|t}}[\nabla_1 V(x_0, \theta_k)|x_t = x] - \nabla_1 V(x, \theta_k) + \nabla_1 V(x, \theta_k) \\
&\qquad\qquad - \nabla_1 V(x, \vartheta_t) + \nabla \log\left(\frac{\rho_t[x]}{\pi^\star(\vartheta_t)[x]}\right)\Big\rangle \mathrm{d}x \\
&= -\int \rho_t \Big\|\nabla \log\Big(\frac{\rho_t}{\pi^\star(\vartheta_t)}\Big)\Big\|^2 \\
&\quad + \int \rho_t[x]\Big\langle \nabla \log\left(\frac{\rho_t[x]}{\pi^\star(\vartheta_t)[x]}\right), \nabla_1 V(x, \theta_k) - \mathbb{E}_{\rho_{0|t}}[\nabla_1 V(x_0, \theta_k)|x_t = x]\Big\rangle \mathrm{d}x \\
&\quad + \int \rho_t[x]\Big\langle \nabla \log\left(\frac{\rho_t[x]}{\pi^\star(\vartheta_t)[x]}\right), \nabla_1 V(x, \vartheta_t) - \nabla_1 V(x, \theta_k)\Big\rangle \mathrm{d}x \\
&=: a_1 + a_2 + a_3.
\end{aligned}$$

Denote the Fisher divergence by

$$\mathrm{FD}(\rho_t \,||\, \pi^\star(\vartheta_t)) := \int \rho_t \Big\|\nabla \log\Big(\frac{\rho_t}{\pi^\star(\vartheta_t)}\Big)\Big\|^2.$$

The first term a_1 is equal to $-\mathrm{FD}(\rho_t \,||\, \pi^\star(\vartheta_t))$. To bound the second term a_2, denote ρ_{0t} the joint distribution of (x_0, x_t). Then

$$a_2 = \int \rho_{0t}[x_0, x_t]\langle \nabla \log\left(\frac{\rho_t[x_t]}{\pi^\star(\vartheta_t)[x_t]}\right), \nabla_1 V(x_t, \theta_k) - \nabla_1 V(x_0, \theta_k)\rangle \mathrm{d}x_0 \mathrm{d}x_t$$

Using $\langle a, b\rangle \le \|a\|^2 + \frac{1}{4}\|b\|^2$ and recalling that $x \mapsto \nabla_1 V(x, \theta)$ is L_X-Lipschitz for all $\theta \in \mathbb{R}^p$ by Assumption 4.6,

$$\begin{aligned}
a_2 &\le \mathbb{E}_{(x_0,x_t)\sim\rho_{0t}}\|\nabla_1 V(x_t, \theta_k) - \nabla_1 V(x_0, \theta_k)\|^2 + \frac{1}{4}\mathbb{E}_{(x_0,x_t)\sim\rho_{0t}}\Big\|\nabla \log\left(\frac{\rho_t[x_t]}{\pi^\star(\vartheta_t)[x_t]}\right)\Big\|^2 \\
&\le L_X^2 \mathbb{E}_{(x_0,x_t)\sim\rho_{0t}}\|x_t - x_0\|^2 + \frac{1}{4}\,\mathrm{FD}(\rho_t \,||\, \pi^\star(\vartheta_t))\,.
\end{aligned}$$

Proceeding similarly for a_3, we obtain

$$a_3 \leq \mathbb{E}_{x\sim\rho_t}\|\nabla_1 V(x,\vartheta_t) - \nabla_1 V(x,\theta_k)\|^2 + \frac{1}{4}\,\mathrm{FD}(\rho_t||\pi^\star(\vartheta_t))\,.$$

Since $\theta \mapsto \nabla_1 V(x,\theta)$ is L_Θ-Lipschitz for all $x \in \mathbb{R}^d$ by Assumption 4.6, we get

$$a_3 \leq L_\Theta\|\vartheta_t - \theta_k\|^2 + \frac{1}{4}\,\mathrm{FD}(\rho_t\,||\,\pi^\star(\theta_t))\,.$$

Moreover, by (23) and under Assumption 4.3, we have $\|\vartheta_t - \theta_k\|^2 = t^2\varepsilon_k^2\|\Gamma(\mu_k,\theta_k)\|^2 \leq t^2\varepsilon_k^2 C^2$, which yields

$$a_3 \leq L_\Theta t^2\varepsilon_k^2 C^2 + \frac{1}{4}\,\mathrm{FD}(\rho_t\,||\,\pi^\star(\vartheta_t))\,.$$

Putting everything together,

$$\begin{aligned}\frac{d\,\mathrm{KL}(\rho_t\,||\,\pi^\star(\vartheta_t))}{\mathrm{d}t} &= a + b \\ &\leq -\frac{1}{2}\,\mathrm{FD}(\rho_t\,||\,\pi^\star(\vartheta_k)) + L_X^2\mathbb{E}_{(x_0,x_t)\sim\rho_{0t}}\|x_t - x_0\|^2 + L_\Theta t^2\varepsilon_k^2 C^2 + 2\varepsilon_k C^2 \\ &\leq -\mu\,\mathrm{KL}(\rho_t\,||\,\pi^\star(\vartheta_t)) + L_X^2\mathbb{E}_{(x_0,x_t)\sim\rho_{0t}}\|x_t - x_0\|^2 + L_\Theta t^2\varepsilon_k^2 C^2 + 2\varepsilon_k C^2\end{aligned}$$

where the last inequality uses Assumption 4.1 and where the two last terms in the r.h.s. can be seen as additional bias terms compared to the analysis of (Cheng and Bartlett, 2018, Theorem 3) and (Vempala and Wibisono, 2019, Lemma 3). Let us now bound $\mathbb{E}_{(x_0,x_t)\sim\rho_{0t}}\|x_t - x_0\|^2$. Recall that $x_t \overset{d}{=} x_0 - t\nabla_1 V(x_0,\theta_k) + \sqrt{2t}z_0$, where $z_0 \sim \mathcal{N}(0,I)$ is independent of x_0. Then

$$\begin{aligned}\mathbb{E}_{(x_0,x_t)\sim\rho_{0t}}\|x_t - x_0\|^2 &= \mathbb{E}_{(x_0,x_t)\sim\rho_{0t}}\| - t\nabla_1 V(x_0,\theta_k) + \sqrt{2t}z_0\|^2 \\ &= t^2\mathbb{E}_{x_0\sim\rho_0}\|\nabla_1 V(x_0,\theta_k)\|^2 + 2td.\end{aligned}$$

Finally, since $x \mapsto \nabla_1 V(x,\theta)$ is L_X-Lipschitz for all $x \in \mathbb{R}^d$ by Assumption 4.6, and under Assumption 4.1, we get by (Vempala and Wibisono, 2019, Lemma 12) that

$$\mathbb{E}_{x_0\sim\rho_0}\|\nabla_1 V(x_0,\theta_k)\|^2 \leq \frac{4L_X^2}{\mu}\,\mathrm{KL}(\rho_0\,||\,\pi^\star(\theta_k)) + 2dL_X\,.$$

All in all,

$$\begin{aligned}\frac{d\,\mathrm{KL}(\rho_t\,||\,\pi^\star(\vartheta_t))}{\mathrm{d}t} \leq -\mu\,\mathrm{KL}(\rho_t\,||\,\pi^\star(\vartheta_t)) &+ \frac{4L_X^4 t^2}{\mu}\,\mathrm{KL}(\rho_0\,||\,\pi^\star(\theta_k)) \\ &+ 2dL_X^3 t^2 + 2L_X^2 td + L_\Theta t^2\varepsilon_k^2 C^2 + 2\varepsilon_k C^2\,.\end{aligned}$$

Recall that we want to integrate t between 0 and γ. For $t \leq \gamma$, we have

$$\begin{aligned}\frac{d\,\mathrm{KL}(\rho_t\,||\,\pi^\star(\vartheta_t))}{\mathrm{d}t} &\leq -\mu\,\mathrm{KL}(\rho_t\,||\,\pi^\star(\vartheta_t)) + \frac{4L_X^4\gamma^2}{\mu}\,\mathrm{KL}(\rho_0\,||\,\pi^\star(\theta_k)) \\ &\quad + 2dL_X^3\gamma^2 + 2L_X^2\gamma d + L_\Theta\gamma^2\varepsilon_k^2 C^2 + 2\varepsilon_k C^2 \\ &\leq -\mu\,\mathrm{KL}(\rho_t\,||\,\pi^\star(\vartheta_t)) + \frac{4L_X^4\gamma^2}{\mu}\,\mathrm{KL}(\rho_0\,||\,\pi^\star(\theta_k)) + 4L_X^2\gamma d + 3\varepsilon_k C^2\end{aligned}$$

since $L_X\gamma \leq 1$, $L_\Theta\gamma^2 \leq 1$ and $\varepsilon_k \leq 1$ by (22). Denote by C_1 the second term and C_2 the sum of the last two terms. Then, by Grönwall's inequality (Pachpatte and Ames, 1997),

$$\mathrm{KL}(\rho_t\,||\,\pi^\star(\vartheta_t)) \leq \frac{(C_1 + C_2)e^{-\mu t}(e^{\mu t} - 1)}{\mu} + \mathrm{KL}(\rho_0\,||\,\pi^\star(\theta_k))e^{-\mu t}\,.$$

Since $\mu t \leq \mu\gamma \leq 1$ by (22), we have $e^{\mu t} \leq 1 + 2\mu\gamma$, and

$$\begin{aligned}\mathrm{KL}(\rho_t\,||\,\pi^\star(\vartheta_t)) &\leq 2(C_1 + C_2)\gamma e^{-\mu t} + \mathrm{KL}(\rho_0\,||\,\pi^\star(\theta_k))e^{-\mu t} \\ &= 2C_2\gamma e^{-\mu t} + \Big(1 + \frac{8L_X^4\gamma^3}{\mu}\Big)\,\mathrm{KL}(\rho_0\,||\,\pi^\star(\theta_k))e^{-\mu t}\,.\end{aligned}$$

Since $\gamma \leq \frac{\mu}{4L_X^2}$ by (22), we have $\frac{8L_X^4\gamma^3}{\mu} \leq \frac{\mu\gamma}{2}$, and

$$\mathrm{KL}(\rho_t \,||\, \pi^\star(\vartheta_t)) \leq 2C_2\gamma e^{-\mu t} + \left(1 + \frac{\mu\gamma}{2}\right) \mathrm{KL}(\rho_0 \,||\, \pi^\star(\theta_k)) e^{-\mu t} \,.$$

We therefore obtain, by evaluating at $t = \gamma$ and renaming $p_{k+1} = \rho_\gamma$, $\pi^\star(\theta_{k+1}) = \pi^\star(\vartheta_\gamma)$, $p_k = \rho_0$, and $\gamma_k = \gamma$,

$$\mathrm{KL}(p_{k+1} \,||\, \pi^\star(\theta_{k+1})) \leq 2C_2\gamma_k e^{-\mu\gamma_k} + \left(1 + \frac{\mu\gamma_k}{2}\right) \mathrm{KL}(p_k \,||\, \pi^\star(\theta_k)) e^{-\mu\gamma_k} \,.$$

Bounding the KL over the whole dynamics. Let $C_3 := (1 + \frac{\mu\gamma_k}{2})e^{-\mu\gamma_k}$. We have $C_3 < 1$, and by summing and telescoping,

$$\mathrm{KL}(p_k \,||\, \pi^\star(\theta_k)) \leq \mathrm{KL}(p_0 \,||\, \pi^\star(\theta_0)) C_3^k + \frac{2C_2C_3\gamma_k e^{-\mu\gamma_k}}{1 - C_3} \,.$$

We have

$$\frac{C_3 e^{-\mu\gamma_k}}{1 - C_3} = \frac{C_3}{e^{\mu\gamma_k} - (1 + \frac{\mu\gamma_k}{2})} \leq \frac{2}{\mu\gamma_k} \,,$$

by using $e^x \geq 1 + x$ and $C_3 \leq 1$. Thus

$$\mathrm{KL}(p_k \,||\, \pi^\star(\theta_k)) \leq \mathrm{KL}(p_0 \,||\, \pi^\star(\theta_0)) C_3^k + \frac{4C_2}{\mu} \,.$$

Replacing C_2 by its value,

$$\mathrm{KL}(p_k \,||\, \pi^\star(\theta_k)) \leq \mathrm{KL}(p_0 \,||\, \pi^\star(\theta_0)) C_3^k + \frac{16L_X^2 d\gamma_k}{\mu} + \frac{12C^2\varepsilon_k}{\mu} \,.$$

Since $e^x \geq 1 + x$, $C_3 \leq e^{-\mu\gamma_k/2}$, and

$$\mathrm{KL}(p_k \,||\, \pi^\star(\theta_k)) \leq \mathrm{KL}(p_0 \,||\, \pi^\star(\theta_0)) e^{-\frac{\mu\gamma_k k}{2}} + \frac{16L_X^2 d\gamma_k}{\mu} + \frac{12C^2\varepsilon_k}{\mu} \,. \tag{24}$$

We obtain three terms in our bound that have different origins. The first term corresponds to an exponential decay of the KL divergence at initialization. The second term is linked to the discretization error, and is proportional to γ_k. The third term is due to the fact that $\pi^\star(\theta_k)$ is moving due to the outer problem updates. It is proportional to the ratio of learning rates ε_k.

Evolution of the loss. By Assumption 4.6, the loss ℓ is L-smooth, and recall that $\theta_{k+1} = \theta_k - \gamma_k\varepsilon_k\Gamma(p_k, \theta_k)$. We have

$$\begin{aligned}
\ell(\theta_{k+1}) &= \ell(\theta_k - \gamma_k\varepsilon_k\Gamma(p_k, \theta_k)) \\
&\leq \ell(\theta_k) - \gamma_k\varepsilon_k\langle\nabla\ell(\theta_k), \Gamma(p_k, \theta_k)\rangle + \frac{L\gamma_k^2\varepsilon_k^2}{2}\|\Gamma(p_k, \theta_k)\|^2 \\
&\leq \ell(\theta_k) - \gamma_k\varepsilon_k\langle\nabla\ell(\theta_k), \Gamma(p_k, \theta_k)\rangle + \frac{LC^2\gamma_k^2\varepsilon_k^2}{2}
\end{aligned}$$

by Assumption 4.3. Furthermore,

$$\Gamma(\mu_k, \theta_k) = \Gamma(\pi^\star(\theta_k), \theta_k) + \Gamma(p_k, \theta_k) - \Gamma(\pi^\star(\theta_k), \theta_k) = \nabla\ell(\theta_k) + \Gamma(p_k, \theta_k) - \Gamma(\pi^\star(\theta_k), \theta_k) \,.$$

Thus

$$\begin{aligned}
\ell(\theta_{k+1}) &\leq \ell(\theta_k) - \gamma_k\varepsilon_k\|\nabla\ell(\theta_k)\|^2 + \gamma_k\varepsilon_k\|\nabla\ell(\theta_k)\|\|\Gamma(p_k, \theta_k) - \Gamma(\pi^\star(\theta_k), \theta_k)\| + \frac{LC^2\gamma_k^2\varepsilon_k^2}{2} \\
&\leq \ell(\theta_k) - \gamma_k\varepsilon_k\|\nabla\ell(\theta_k)\|^2 + \gamma_k\varepsilon_k\|\nabla\ell(\theta_k)\|\sqrt{\mathrm{KL}(p_k||\pi^\star(\theta_k))} + \frac{LC^2\gamma_k^2\varepsilon_k^2}{2}
\end{aligned}$$

by Assumption 4.3. Using $ab \leq \frac{1}{2}(a^2 + b^2)$, we obtain

$$\ell(\theta_{k+1}) \leq \ell(\theta_k) - \frac{\gamma_k\varepsilon_k}{2}\|\nabla\ell(\theta_k)\|^2 + \frac{\gamma_k\varepsilon_k}{2}\mathrm{KL}(p_k||\pi^\star(\theta_k)) + \frac{LC^2\gamma_k^2\varepsilon_k^2}{2} \,.$$

Conclusion. Summing and telescoping,

$$\ell(\theta_{K+1}) - \ell(\theta_1) \leq -\frac{1}{2}\sum_{k=1}^{K} \gamma_k \varepsilon_k \|\nabla \ell(\theta_k)\|^2 + \frac{1}{2}\sum_{k=1}^{K} \gamma_k \varepsilon_k \,\mathrm{KL}(p_k || \pi^\star(\theta_k)) + \frac{LC^2}{2}\sum_{k=1}^{K} \gamma_k^2 \varepsilon_k^2 \,.$$

Lower bounding γ_k by γ_K and ε_k by ε_K in the first sum, then reorganizing terms, we obtain

$$\frac{1}{K}\sum_{k=1}^{K} \|\nabla \ell(\theta_k)\|^2 \leq \frac{2(\ell(\theta_1) - \inf \ell)}{K\gamma_K \varepsilon_K} + \frac{1}{K\gamma_K \varepsilon_K}\sum_{k=1}^{K} \gamma_k \varepsilon_k \,\mathrm{KL}(p_k || \pi^\star(\theta_k)) + \frac{LC^2}{K\gamma_K \varepsilon_K}\sum_{k=1}^{K} \gamma_k^2 \varepsilon_k^2 \,.$$

Bounding the KL divergence by (24), we get

$$\begin{aligned}\frac{1}{K}\sum_{k=1}^{K} \|\nabla \ell(\theta_k)\|^2 \leq \frac{2(\ell(\theta_1) - \inf \ell)}{K\gamma_K \varepsilon_K} + \frac{1}{K\gamma_K \varepsilon_K}\sum_{k=1}^{K} \mathrm{KL}(p_0 || \pi^\star(\theta_0)) \gamma_k \varepsilon_k e^{-\frac{\mu \gamma_k k}{2}} \\ + \frac{1}{K\gamma_K \varepsilon_K}\sum_{k=1}^{K} \frac{16 L_X^2 d \gamma_k^2 \varepsilon_k}{\mu} + \frac{1}{K\gamma_K \varepsilon_K}\sum_{k=1}^{K} \frac{12 C^2 \gamma_k \varepsilon_k^2}{\mu} + \frac{LC^2 \gamma}{K\gamma_K \varepsilon_K}\sum_{k=1}^{K} \gamma_k^2 \varepsilon_k^2 \,.\end{aligned}$$

By definition (22) of γ_k and ε_k, we see that the first and last sums are converging, and the middle sums are $\mathcal{O}(\ln K)$. Therefore, we obtain

$$\frac{1}{K}\sum_{k=1}^{K} \|\nabla \ell(\theta_k)\|^2 \leq \frac{c \ln K}{K^{1/3}}$$

for some $c > 0$ depending only on the constants of the problem.

B.3 Denoising diffusion

Our goal is first to find a continuous-time equivalent of Algorithm 3. To this aim, we first introduce a slightly different version of the algorithm where the queue of M versions of p_k is not present at initialization, but constructed during the first M steps of the algorithm. As a consequence, θ changes for the first time after M steps of the algorithm, when the queue is completed and the M-th element of the queue has been processed through all the sampling steps.

Algorithm 7 Implicit Diff. optimization, finite time (no warm start)

input $\theta_0 \in \mathbb{R}^p$, $p_0 \in \mathcal{P}$
  $p_0^{(0)} \leftarrow p_0$
  for $k \in \{0, \dots, K-1\}$ (joint single loop) **do**
    $p_{k+1}^{(0)} \leftarrow p_0$
    parallel $p_{k+1}^{(m+1)} \leftarrow \Sigma_m(p_k^{(m)}, \theta_k)$ for $m \in [\min(k, M-1)]$
    if $k \geq M$ **then**
      $\theta_{k+1} \leftarrow \theta_k - \eta_k \Gamma(p_k^{(M)}, \theta_k)$ (or another optimizer)
output θ_K

In order to obtain the continuous-time equivalent of Algorithm 7, it is convenient to change indices defining $p_k^{(m)}$. Note that in the update of $p_k^{(m)}$ in the algorithm, the quantity $m - k$ is constant. Therefore, denoting $j = m - k$, the algorithm above is exactly equivalent to

$$\begin{aligned} p_j^{(0)} &= p_0 \\ p_j^{(m+1)} &= \Sigma_m(p_j^{(m)}, \theta_{j+m}), \quad m \in [M-1] \\ \theta_{k+1} &= \theta_k \quad \text{if } k < M \\ \theta_{k+1} &= \theta_k - \eta_k \Gamma(p_{k-M}^{(M)}, \theta_k) \quad \text{else.} \end{aligned}$$

It is then possible to translate this algorithm in a continuous setting in the case of denoising diffusions. We obtain

$$\begin{aligned} Y_t^0 &\sim \mathcal{N}(0,1) \\ \mathrm{d}Y_t^\tau &= \{Y_t^\tau + 2s_{\theta_{t+\tau}}(Y_t^\tau, T-\tau)\}\mathrm{d}\tau + \sqrt{2}\mathrm{d}B_\tau \\ \mathrm{d}\theta_t &= 0 \quad \text{for } t < T \\ \mathrm{d}\theta_t &= -\eta_k \Gamma(Y_{t-T}^T, \theta_t)\mathrm{d}t \quad \text{for } t \geq T, \end{aligned} \tag{25}$$

Let us choose the score function s_θ as in Section 4.3. The backward equation (7) then writes

$$\mathrm{d}Y_t = (Y_t + 2s_\theta(Y_t, T-t))\mathrm{d}t + \sqrt{2}\mathrm{d}B_t = -(Y_t - 2\theta e^{-(T-t)})\mathrm{d}t + \sqrt{2}\mathrm{d}B_t\,. \tag{26}$$

We now turn our attention to the computation of Γ. Recall that Γ should satisfy $\Gamma(\pi^\star(\theta), \theta) = \nabla\ell(\theta)$, where $\pi^\star(\theta)$ is the distribution of Y_T and $\ell(\theta) = \mathbb{E}_{Y\sim\pi^\star(\theta)}(L(Y))$ for some loss function $L : \mathbb{R} \to \mathbb{R}$. To this aim, for a given realization of the SDE (26), let us compute the derivative of $L(Y_T)$ with respect to θ using the adjoint method (14). We have $\nabla_2\mu(T-t, Y_{T-t}, \theta) = -1$, hence $\frac{\mathrm{d}A_t}{\mathrm{d}t} = -1$, and $A_t = L'(Y_T)e^{-t}$. Furthermore, $\nabla_3\mu(T-t, Y_{T-t}, \theta) = 2e^{-t}$. Thus

$$\frac{\mathrm{d}L(Y_T)}{\mathrm{d}\theta} = \int_0^T A_t \nabla_3\mu(T-t, Y_{T-t}, \theta)\mathrm{d}t = \int_0^T 2L'(Y_T)e^{-2t}\mathrm{d}t = L'(Y_T)(1-e^{-2T})\,.$$

As a consequence,

$$\nabla\ell(\theta) = \mathbb{E}_{Y\sim q_T^*(\theta)}\Big(\frac{\mathrm{d}L(Y)}{\mathrm{d}\theta}\Big) = \mathbb{E}_{Y\sim q_T^*(\theta)}(L'(Y)(1-e^{-2T}))\,.$$

This prompts us to define

$$\Gamma(p,\theta) = \mathbb{E}_{X\sim p}(L'(X)(1-e^{-2T})) = \mathbb{E}_{X\sim p}((X-\theta_{\text{target}})(1-e^{-2T})) = (\mathbb{E}_{X\sim p}X - \theta_{\text{target}})(1-e^{-2T})$$

for L defined by $L(x) = (x-\theta_{\text{target}})^2$ as in Section 4.3. Note that, in this case, Γ depends only on p and not on θ. Replacing s_θ and Γ by their values in (25), we obtain the coupled differential equations in Y and θ

$$\begin{aligned} Y_t^0 &\sim \mathcal{N}(0,1) \\ \mathrm{d}Y_t^\tau &= \{-Y_t^\tau + 2\theta_{t+\tau}e^{-(T-\tau)}\}\mathrm{d}\tau + \sqrt{2}\mathrm{d}B_\tau \\ \mathrm{d}\theta_t &= 0 \quad \text{for } t < T \\ \mathrm{d}\theta_t &= -\eta(\mathbb{E}Y_{t-T}^T - \theta_{\text{target}})(1-e^{-2T})\mathrm{d}t \quad \text{for } t \geq T, \end{aligned} \tag{27}$$

We can now formalize Proposition 4.8, with the following statement.

Proposition B.3. *Consider the dynamics* (27)*. Then*

$$\|\theta_{2T} - \theta_{\text{target}}\| = \mathcal{O}(e^{-T})\,, \quad \text{and} \quad \pi^\star(\theta_{2T}) = \mathcal{N}(\mu, 1) \text{ with } \mu = \theta_{\text{target}} + \mathcal{O}(e^{-T})\,.$$

Proof. Let us first compute the expectation of Y_T^τ. To this aim, denote $Z_t^\tau = e^\tau Y_t^\tau$. Then we have

$$\mathrm{d}Z_t^\tau = e^\tau(\mathrm{d}Y_t^\tau + Y_t^\tau\mathrm{d}t) = 2\theta_{t+\tau}e^{2\tau-T}\mathrm{d}\tau + \sqrt{2}\mathrm{d}B_\tau\,.$$

Since $\mathbb{E}(Z_t^0) = \mathbb{E}(Y_t^0) = 0$, we obtain that $\mathbb{E}(Z_t^T) = 2\int_0^T \theta_{t+\tau}e^{2(\tau-T)}\mathrm{d}\tau$, and

$$\mathbb{E}(Y_t^T) = e^{-T}\mathbb{E}(Z_t^T) = 2\int_0^T \theta_{t+\tau}e^{2(\tau-T)}\mathrm{d}\tau\,.$$

Therefore, we obtain the following evolution equation for θ when $t \geq T$:

$$\dot\theta_t = -\eta\Big(2\int_0^T \theta_{t-T+\tau}e^{2(\tau-T)}\mathrm{d}\tau - \theta_{\text{target}}\Big)(1-e^{-2T})\,.$$

By the change of variable $\tau \leftarrow T-\tau$ in the integral, we have, for $t \geq T$,

$$\dot\theta_t = -\eta\Big(2\int_0^T \theta_{t-\tau}e^{-2\tau}\mathrm{d}\tau - \theta_{\text{target}}\Big)(1-e^{-2T})\,.$$

Let us introduce the auxiliary variable $\psi_t = \int_0^T \theta_{t-\tau} e^{-2\tau} \mathrm{d}\tau$. We have $\dot{\theta}_t = -\eta(2\psi_t - \theta_{\text{target}})$, and

$$\dot{\psi}_t = \int_0^T \dot{\theta}_{t-\tau} e^{-2\tau} \mathrm{d}\tau = -[\theta_{t-\tau} e^{-2\tau}]_0^T - 2\int_0^T \theta_{t-\tau} e^{-2\tau} d\tau = \theta_t - \theta_{t-T} e^{-2T} - 2\psi_t \,.$$

Recall that θ_t is constant equal to θ_0 for $t \in [0, T]$. Therefore, for $t \in [T, 2T]$, $\xi := (\theta, \psi)$ satisfies the first order linear ODE with constant coefficients

$$\dot{\xi} = A\xi + b, \quad A = \begin{pmatrix} 0 & -2\eta \\ 1 & -2 \end{pmatrix}, \quad b = \begin{pmatrix} \eta\theta_{\text{target}}(1 - e^{-2T}) \\ -\theta_0 e^{-2T} \end{pmatrix} .$$

For $\eta > 0$, A is invertible, and

$$A^{-1} = \frac{1}{2\eta} \begin{pmatrix} -2 & 2\eta \\ -1 & 0 \end{pmatrix} .$$

Hence the linear ODE has solution, for $t \in [T, 2T]$,

$$\xi_t = -(I - e^{(t-T)A})A^{-1}b + e^{(t-T)A}\xi_T = -A^{-1}b + e^{(t-T)A}(A^{-1}b + \xi_T) \,.$$

Thus

$$\xi_{2T} = -A^{-1}b + e^{TA}(A^{-1}b + \xi_T) = -A^{-1}b + e^{TA}\left(A^{-1}b + \begin{pmatrix} \theta_0 \\ \frac{\theta_0(1-e^{-2T})}{2} \end{pmatrix}\right) .$$

A straightforward computation shows that

$$[A^{-1}b]_0 = -\theta_{\text{target}}(1 - e^{-2T}) + \theta_0 e^{-2T} \,,$$

and that A has eigenvalues with negative real part. Putting everything together, we obtain

$$\theta_{2T} = [\xi_{2T}]_0 = \theta_{\text{target}} + \mathcal{O}(e^{-T}) \,.$$

Finally, recall that we have $\pi^\star(\theta) = \mathcal{N}(\theta(1-e^{-2T}), 1)$. Thus $\pi^\star(\theta_{2T}) = \mathcal{N}(\mu_{2T}, 1)$ with $\mu_{2T} = \theta_{\text{target}} + \mathcal{O}(e^{-T})$. □

C EXPERIMENTAL DETAILS

We provide here details about the experiments that we have presented in Section 5.

C.1 Langevin processes

We provide in this section details about our experiments on Langevin processes. In Section C.1.1, we do so for the experiment described in Section 5.1, where the reward is an explicit function $R(\cdot)$. In Section C.1.2, we show additional experiments showing that Implicit Diffusion can also be used to "train from scratch" a model: in this case the reward is a negative KL term that is evaluated from samples of a target distribution. We present results in several parametrization settings.

C.1.1 Reward training

We consider a parameterized family of potentials for $x \in \mathbb{R}^2$ and $\theta \in \mathbb{R}^6$ defined by

$$V(x, \theta) = -\log\Big(\sum_{i=1}^6 \sigma(\theta)_i \exp\big(-\|x - \mu_i\|^2\big)\Big) \,,$$

where the $\mu_i \in \mathbb{R}^2$ are the six vertices of a regular hexagon and σ is the softmax function mapping $\mathbb{R}^6$ to the unit simplex. In this setting, for any $\theta \in \mathbb{R}^6$,

$$\pi^\star(\theta) = \frac{1}{Z}\sum_{i=1}^6 \sigma(\theta)_i \exp\big(-\|x - \mu_i\|^2\big) \,,$$

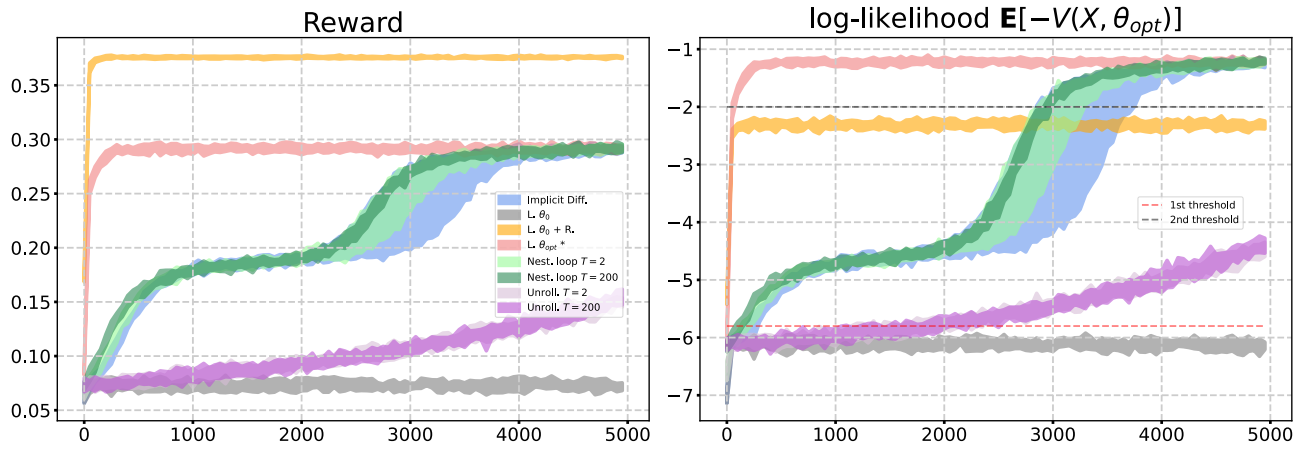


Figure 10: Confidence intervals for metrics for reward training of Langevin processes. **Left**: Evolution of the reward. **Right**: Evolution of the log-likelihood.

where Z is an absolute renormalization constant that is independent of θ. This fact simplifies drawing contour lines, but we do not use this prior knowledge in our algorithms, and only use calls to functions $\nabla_1 V(\cdot, \theta)$ and $\nabla_2 V(\cdot, \theta)$ for various $\theta \in \mathbb{R}^6$.

We run six sampling algorithms, all initialized with $p_0 = \mathcal{N}(0, I_2)$. For all of them we generate a batch of variables $X^{(i)}$ of size $1,000$, all initialized independently with $X_0^{(i)} \sim \mathcal{N}(0, I_2)$. The sampling and optimization steps are realized in parallel over the batch. The samples are represented after $K = 5,000$ steps of each algorithm in Figure 5, and used to compute the values of reward and likelihood reported in Figure 6. We also display in Figure 12 the dynamics of the probabilities throughout these algorithms.

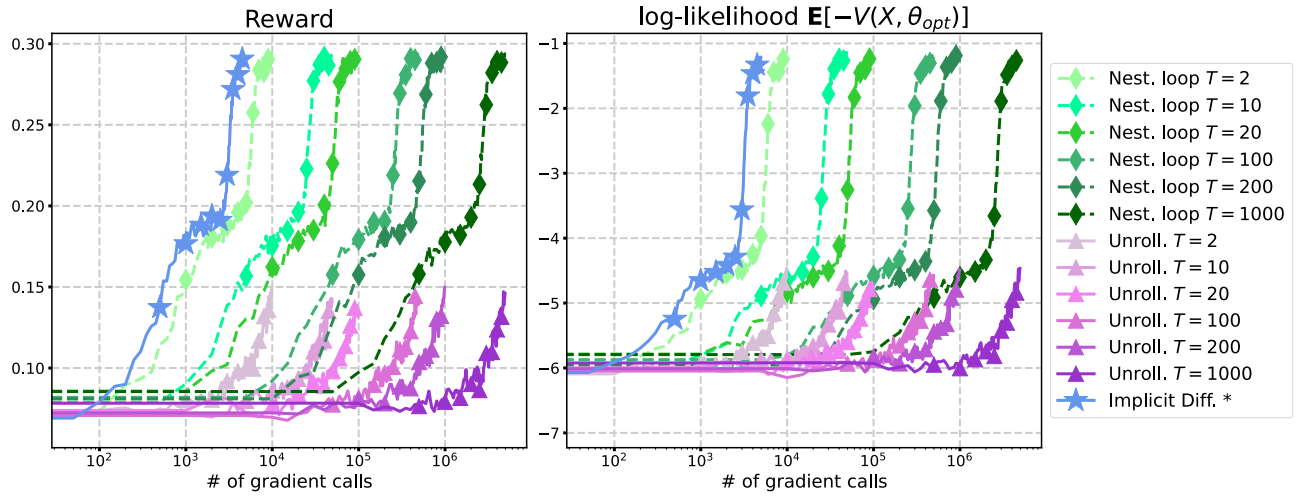


Figure 11: Comparison between Implicit Diffusion, nested loop algorithm and unrolling algorithm, for various number of inner steps T. The x-axis is the total number of gradient evaluations (roughly equal to the number of optimization steps multiplied by the number of inner loop steps T). **Left**: Evolution of the reward. **Right**: Evolution of the log-likelihood.

We provide here additional details of and motivation for these algorithms, denoted by the colored markers that represent them in these figures.

- Langevin θ_0 (■): This is the discrete-time process (a Langevin Monte Carlo process) approximating a Langevin diffusion with potential $V(\cdot, \theta_0)$ for fixed $\theta_0 := (1, 0, 1, 0, 1, 0)$. There is no reward here; the time-continuous Langevin process converges to $\pi^\star(\theta_0)$, which has some symmetries. It can be thought of as a pretrained model, and the Langevin sampling algorithm as an inference-time generative algorithm.
- **Implicit Diffusion** (★): We run the infinite-time horizon version of our method (Algorithm 2), aiming to minimize $\ell(\theta) := \mathcal{F}(\pi^\star(\theta))$ for $\mathcal{F}(p) = -\mathbb{E}_{X\sim p}[R(X)]$ with $R(x) = \mathbf{1}(x_1 > 0)\exp\big(-\|x - \mu\|^2\big)$ where $\mu = (1, 0.95)$. This algorithm yields both a sample $\hat{p}_K$ and parameters θ_{opt} after K steps, and can be thought of as jointly sampling and reward finetuning.
- Nested loop (♦): We run Algorithm 1 with T inner sampling steps for each gradient step. For $T = 1$, this is exactly Implicit Diffusion. For $T \gg 1$, it means we compute nearly perfectly $\pi^\star(\theta_t)$ at each optimization step.
- Unrolling through the last step of sampling (▲): For each optimization step, we perform T sampling steps, then differentiate through the last step of sampling by automatic differentiation. This is akin to a stop gradient method. The learning rate here is chosen as $2\gamma_\theta/\gamma_X$ to improve its performance, for a fair comparison. Recent studies show that differentiating through the last sampling step is an efficient and theoretically-grounded method in bilevel optimization (Bolte et al., 2023). It has been applied successfully to denoising diffusions (Clark et al., 2024).
- Langevin θ_0 + R (▼): This is the discretization of a Langevin diffusion with reward-guided potential $V(\cdot, \theta_0) - \lambda R_{\text{smooth}}$, where R_{smooth} is a smoothed version of R (replacing the indicator by a sigmoid). Using this approach is different from finetuning: it proposes to modify the sampling algorithm, and does not yield new parameters θ. This is akin to guidance of generative models (Dhariwal and Nichol, 2021). Note that this requires a differentiable reward R_{smooth}, contrarily to our approach that handles non-differentiable rewards.
- Langevin θ_{opt} - post **Implicit Diffusion** (●): This is a discrete-time process approximating a Langevin diffusion with potential $V(\cdot, \theta_{\text{opt}})$, where θ_{opt} is the outcome of reward training by our algorithm. This can be thought of as doing inference with the new model parameters, post reward training with Implicit Diffusion.

As mentioned in Section 5.1, this setting illustrates the advantage of our method, which allows the efficient optimization of a function over a constrained set of distribution, without overfitting outside this class. We display in Figure 12 snapshots throughout some selected steps of these six algorithms (in the same order and with the same colors as indicated above). We observe that the dynamics of Implicit Diffusion are slower than those of Langevin processes (sampling), which can be observed also in the metrics reported in Figure 6. The reward and log-likelihood change slowly, plateauing several times: when θ_k in this algorithm is initially close to θ_0, the distribution gets closer to $\pi^\star(\theta_0)$ (steps 0-100). It then evolves towards another distribution (steps 1000-2500), after θ has been affected by accurate gradient updates, before converging to $\pi^\star(\theta_{\text{opt}})$. The two-timescale dynamics is by design: the sampling dynamics are much faster, aiming to quickly lead to an accurate evaluation of gradients with respect to θ_k. This corresponds to our theoretical setting where $\varepsilon_k \ll 1$. To complement the comparisons between our algorithm and other baselines included in Section 5.1, we also provide in Figure 11 a comparison between Implicit Diffusion, the nested loop and unrolling approaches, in terms of reward and log-likelihood optimization, **as a function of the number of gradient evaluations** (i.e., number of sampling steps), rather than number of optimization steps. Again, it is apparent that the algorithmic cost of doing several steps ($T > 1$) of inner loop is much higher than the small improvement obtained by a better estimate of the gradients. Finally, the confidence intervals in Figure 6 are computed by performing 10 independent repetitions of the experiment, and reporting the largest and lowest metrics across the 10 repetitions, at each time step. For readability, Figure 10 shows the same plot with confidence intervals only (without plotting the average value).

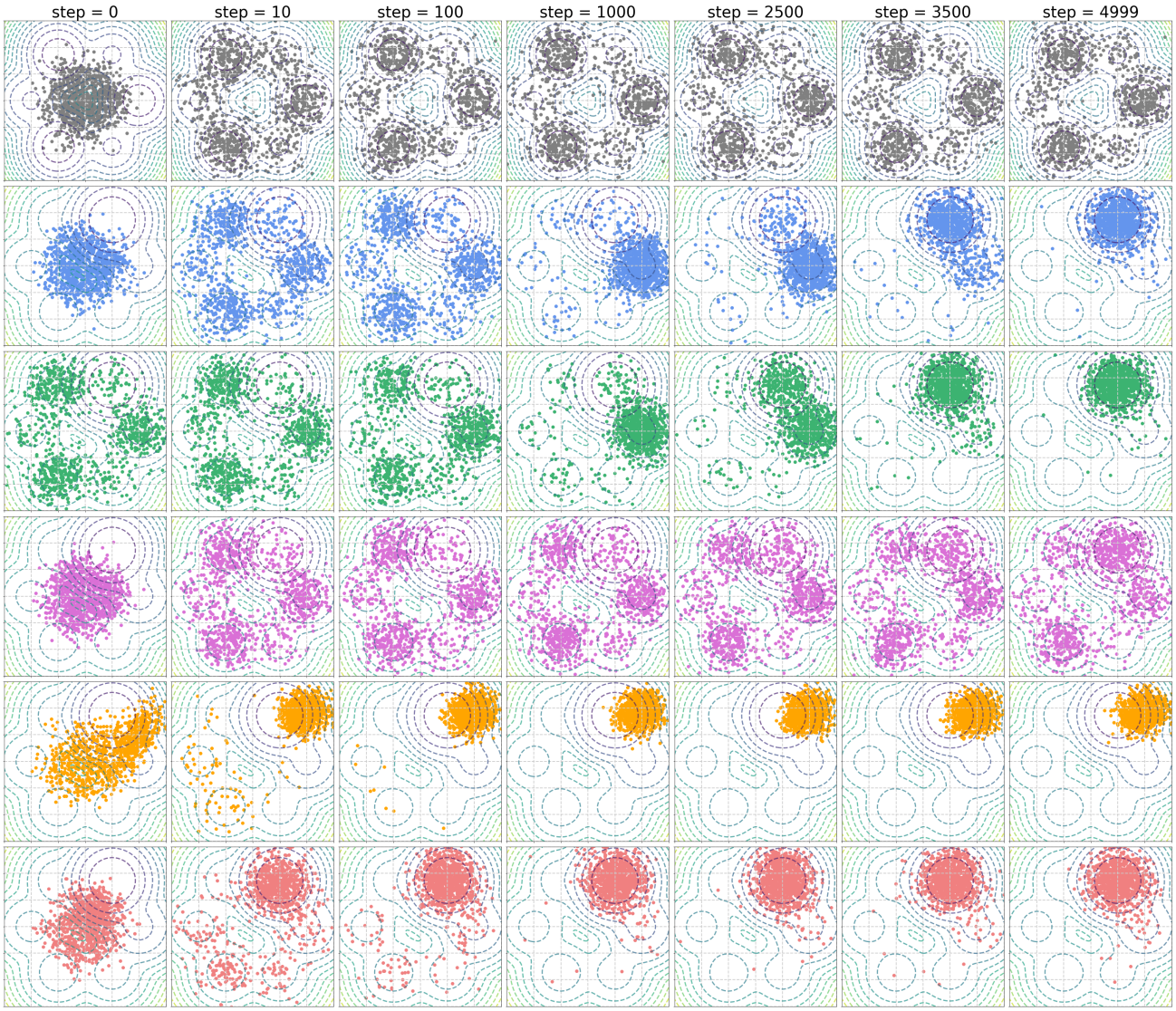


Figure 12: Dynamics of samples for four sampling algorithms after different time steps (for instance, the first column is after one step). **First row:** Langevin θ_0 (●) with $\pi^\star(\theta_0)$ contour lines. **Second:** Implicit Diffusion (●) with $\pi^\star(\theta_{\text{opt}})$ contour lines. **Third:** Nested loop algorithm with $T = 100$ (●). **Fourth:** Unrolling through the last step of sampling with $T = 100$ (●). **Fifth:** Langevin θ_0 + smoothed Reward (●). **Sixth:** Langevin θ_{opt} (●).

C.1.2 Training from scratch

We present in Figure 13 a variant of this experiment where we start from a model generating a standard Gaussian, and our goal is to learn to generate a mixture of several Gaussians. For this, comparing with the setup presented above, we add a 7th potential well at the origin, and choose at initialization $\theta_0 = (-7, -7, -7, -7, -7, -7, 11)$. This means that the distribution at initialization is extremely close to being a standard Gaussian, as can be seen in the top-right plot of Figure 13a. The target is $\theta^* = (1.5, 0, 1.5, 0, 1.5, 0, 0)$. We use Implicit Diffusion where the reward is the KL between the target distribution and the current one. This KL admits explicit gradients (see Section 3.2) which can be evaluated with samples of the target distribution. We train for $K = 40,000$ steps with a batch of size $1,000$. Learning rates are $\gamma_X = 2.5 \cdot 10^{-2}$ and $\gamma_\theta = 7 \cdot 10^{-3}$.

Training from scratch with a full Gaussian mixture model parameterization. We present in Figure 14 a variant of this experiment where we now parameterize the means and covariances of the Gaussians in addition to the weights of the mixture. More precisely, we consider the potential

$$V(x, \theta) = -\log\Big(\sum_{i=1}^{2} \frac{\sigma(w)_i}{2\pi\sqrt{\det(\Sigma_i)}} \exp\big(-(x-\mu_i)^\intercal \Sigma_i^{-1}(x-\mu_i)\big)\Big)\,,$$

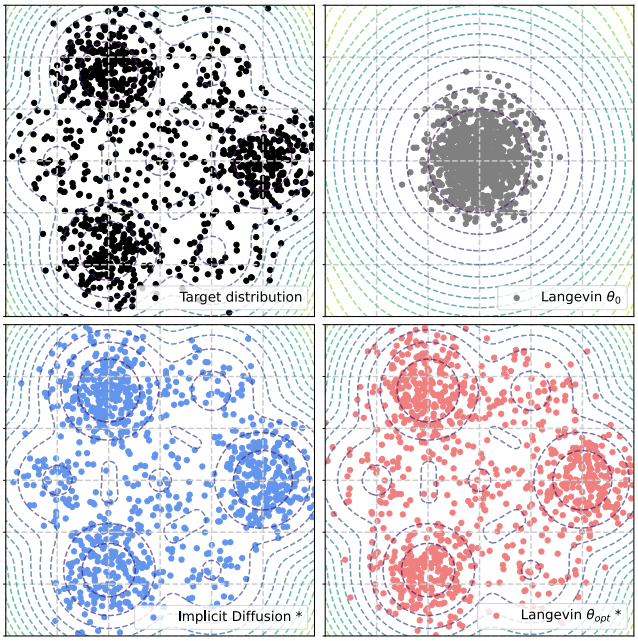
where σ is still the softmax function mapping $\mathbb{R}^2$ to the unit simplex, and $\theta = \{w, (\mu_i, \Sigma_i)_{1\leq i\leq 2}\} \in \mathbb{R}^2 \times (\mathbb{R}^2, \mathbb{R}^{2\times 2})^2$. At initialization, the parameters are

$$\theta_0 = \big\{(0,0), ((4,0), I_2), ((-2, 2\sqrt{3}), I_2)\big\}\,,$$

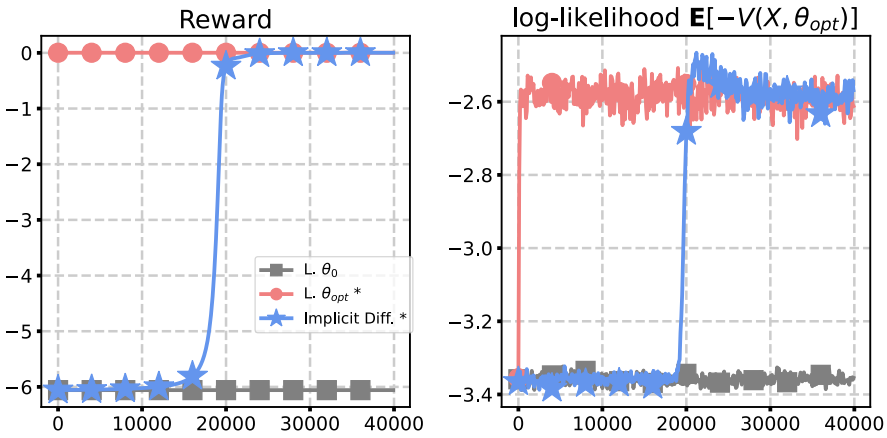
while our target is

$$\theta^* = \left\{(1.5, 0), \left((-4, 0), \begin{pmatrix} 0.75 & -0.5 \\ -0.5 & 1.5 \end{pmatrix}\right), \left((2, -2\sqrt{3}), \begin{pmatrix} 0.75 & 0.5 \\ 0.5 & 1.25 \end{pmatrix}\right)\right\}.$$

As in the previous experiment, we use Implicit Diffusion where the reward is the KL between the target distribution and the current one. This KL admits explicit gradients (see Section 3.2) which can be evaluated with samples of the target distribution. We train for $K = 40,000$ steps with a batch of size $1,000$. Learning rates are $\gamma_X = 5 \cdot 10^{-2}$ and $\gamma_\theta = 5 \cdot 10^{-4}$. We observe on the Figure that Implicit Diffusion is able to learn the target distribution.

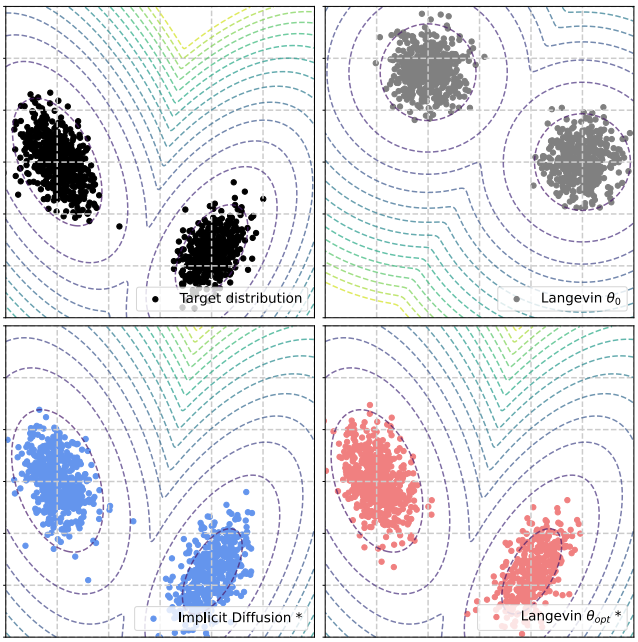


(a) Contour lines and samples from sampling algorithms. We start from a model generating a standard Gaussian (top-right figure), and our goal is to learn to generate a mixture of several Gaussians (top-left figure). We use Implicit Diffusion where the reward is the KL between the target distribution and the current one. We observe that Implicit Diffusion is able to learn the target distribution (bottom-left figure). After running Implicit Diffusion, it is easy to generate new samples that are close to the target distribution (bottom-right figure).

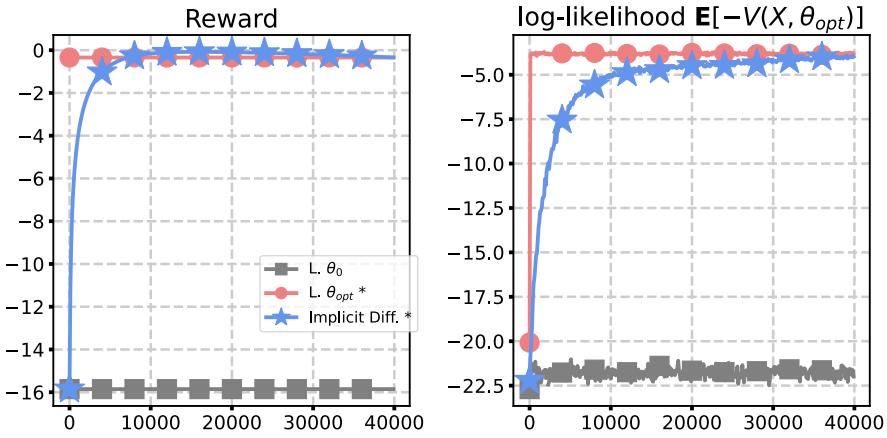


(b) Evolution of the reward and of the log-likelihood of the samples for the initial model, for the Implicit Diffusion algorithm, and for the trained model after Implicit Diffusion. The reward is the (opposite of the) KL between the target distribution and the current one, so a reward equal to zero means we learnt to reproduce the target distribution.

Figure 13: Optimizing through sampling with **Implicit Diffusion** to train from scratch an energy-based model (Langevin diffusion).



(a) Contour lines and samples from sampling algorithms. We start from a model generating a mixture of two Gaussians (top-right figure), and our goal is to learn to generate another mixture with different means, covariances and weights (top-left figure). We use Implicit Diffusion where the reward is the KL between the target distribution and the current one. We observe that Implicit Diffusion is able to learn the target distribution (bottom-left figure). After running Implicit Diffusion, it is easy to generate new samples that are close to the target distribution (bottom-right figure).



(b) Evolution of the reward and of the log-likelihood of the samples for the initial model, for the Implicit Diffusion algorithm, and for the trained model after Implicit Diffusion. The reward is the (opposite of the) KL between the target distribution and the current one, so a reward equal to zero means we learnt to reproduce the target distribution.

Figure 14: Optimizing through sampling with **Implicit Diffusion** to train from scratch an energy-based model (Langevin diffusion). The potential V is a logsumexp of 2 quadratic forms, where the weights, centers, and the matrix of the forms are learnable parameters, so that the outcome distribution can be any mixture of 2 Gaussians.

C.2 Denoising diffusion models

We first provide additional experimental configurations that are common between both datasets before explaining details specific to each one.

Common details. The KL term in the reward is computed using Girsanov's theorem as explained in Appendix A.3. We use the Adam optimizer (Kingma and Ba, 2015), with various values for the learning rate (see e.g. Figure 7). The code was implemented in JAX (Bradbury et al., 2018). As mentioned in the main text, we use a U-Net model (Ronneberger et al., 2015).

MNIST. We use an Ornstein-Uhlenbeck noise schedule, meaning that the forward diffusion is $\mathrm{d}X_t = -X_t \mathrm{d}t + \sqrt{2}\mathrm{d}B_t$ (as presented in Section 2.2). We pretrain for 18k steps in 7 minutes on 4 TPUv2. For reward training, we train on a TPUv2 for 4 hours with a queue of size $M = 4$, $T = 64$ steps, and a batch size of 32. Further hyperparameters for pretraining and reward training are given respectively in Tables 1 and 2.

Name	Value
Noise schedule	Ornstein-Uhlenbeck
Optimizer	Adam with standard hyperparameters
EMA decay	0.995
Learning rate	10^{-3}
Batch size	32

Table 1: Hyperparameters for pretraining of denoising diffusion models on MNIST.

Name	Value
Number of sampling steps	256
Sampler	Euler
Noise schedule	Ornstein-Uhlenbeck
Optimizer	Adam with standard hyperparameters

Table 2: Hyperparameters for reward training of denoising diffusion models pretrained on MNIST.

CIFAR-10 and LSUN. For CIFAR-10, we pretrain for 500k steps in 30 hours on 16 TPUv2, reaching an FID score (Heusel et al., 2017) of 2.5. For reward training, we train on a TPUv3 for 9 hours with a queue of size $M = 4$ and $T = 64$ steps, and a batch size of 32. For LSUN, we pretrain for 13 hours, reaching a FID score of 2.26. For reward training, we train on a TPUv3 for 9 hours with a queue of size $M = 2$, $T = 64$ steps, and a batch size of 16. Further hyperparameters for pretraining and reward training are given respectively in Tables 3 and 4.

Name	Value
Number of sampling steps	$1,024$
Sampler	DDPM
Noise schedule	Cosine (Nichol and Dhariwal, 2021)
Optimizer	Adam with $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\varepsilon = 10^{-12}$
EMA decay	0.9999
Learning rate	$2 \cdot 10^{-4}$
Batch size	2048
Number of samples for FID evaluation	50k

Table 3: Hyperparameters for pretraining of denoising diffusion models on CIFAR-10 and LSUN.

Name	Value
Number of sampling steps	$1,024$
Sampler	Euler
Noise schedule	Cosine (Nichol and Dhariwal, 2021)
Optimizer	Adam with standard hyperparameters

Table 4: Hyperparameters for reward training of denoising diffusion models pretrained on CIFAR-10 and LSUN.

Additional figures for MNIST. We report in Figure 16 metrics on the rewards and KL divergence with respect to the original distribution, in the case where $\lambda > 0$. As in Figure 7, we observe the competition between the reward and the divergence with respect to the distribution after pretraining. The results are sensitive to the choice of hyperparameters. In particular, when the ratio λ/β is too small, we do not observe an improvement of the reward. Note that we did not perform extensive hyperparameter finetuning for these plots, so it is likely that better results could be obtained with more hyperparameter finetuning. We also display in Figures 15 some additional selected examples of samples generated by our denoising diffusion model with parameters θ_k, at several steps k of our algorithm. Note that the random number generator system of JAX allows us, for illustration purposes, to sample for different parameters from the same seed. We take advantage of this feature to visualize the evolution of a given realization of the stochastic denoising process depending on θ.

Recall that we consider

$$\mathcal{F}(p) := -\lambda \mathbb{E}_{x\sim p}[R(x)] + \beta \, \mathrm{KL}(p \,||\, \pi^\star(\theta_0)) \,,$$

where $R(x)$ is the average value of all the pixels in x. The figures in the main text and in the appendix present samples for negative and positive λ, rewarding respectively darker and brighter images. We emphasize that these samples are generated for evaluation purposes. To generate the samples, at various steps of the optimization procedure, we run the full denoising process for the current value of the parameters. In particular, these samples are different from the ones used to perform the joint sampling and parameter updates in Implicit Diffusion.

We have purposefully chosen, for illustration purposes, samples for experiments with the highest magnitude of λ/β, i.e. those that favor reward optimization over proximity to the original distribution. As noted in Section 5, we observe qualitatively that reward training, while shifting some aspects of the distribution (here the average brightness), and necessarily diverging from the original pretrained model, manages to do so while retaining some important global characteristics of the dataset–even though the pretraining dataset is never observed during reward training. Since we chose to display samples from experiments with the most extreme incentives towards the reward, we observe that the similarity with the pretraining dataset can be forced to break down after a certain number of reward training steps. We also observe some mode collapse; we comment further on this point below.

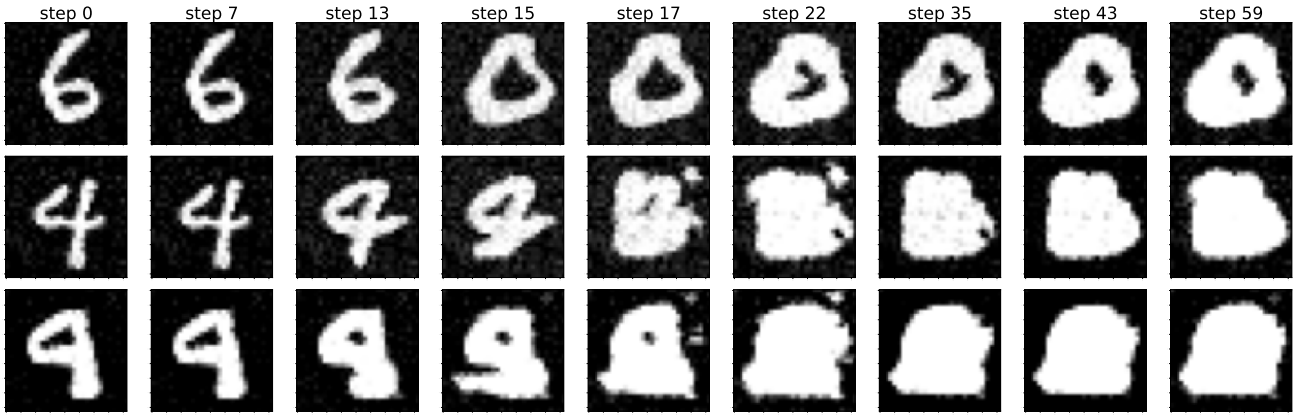


Figure 15: Reward training for a model pretrained on MNIST. The reward favors **brighter** images ($\lambda > 0$, $\beta > 0$). Selected examples are shown coming from a single experiment with $\lambda/\beta = 30$. All digits are re-sampled at the same selected steps of the Implicit Diffusion algorithm.

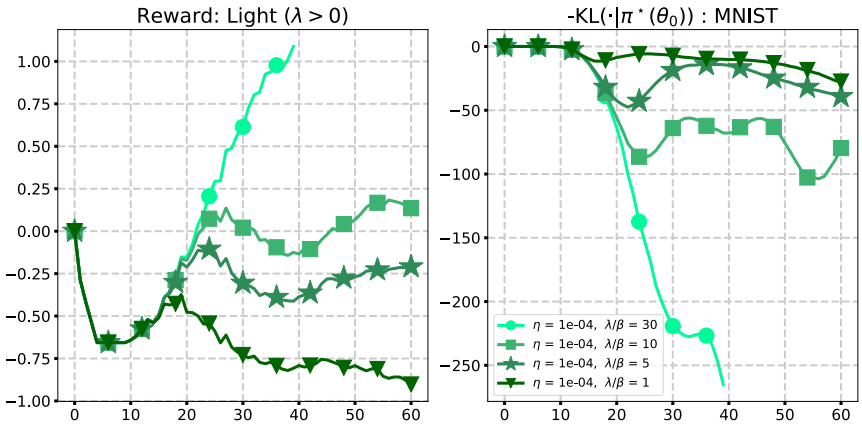


Figure 16: Score function reward training with **Implicit Diffusion** pretrained on MNIST for various $\lambda > 0$ (brighter). **Left:** Reward, average brightness of image. **Right:** Divergence w.r.t. the original pretrained distribution.

Additional figures for CIFAR-10. We recall that we consider, for a model with weights θ_0 pretrained on CIFAR-10, the objective function

$$\mathcal{F}(p) := -\lambda \mathbb{E}_{x\sim p}[R(x)] + \beta\, \mathrm{KL}(p \,||\, \pi^\star(\theta_0))\,,$$

where $R(x)$ is the average over the red channel, minus the average of the other channels. We show in Figure 17 the result of the denoising process for some fixed samples and various steps of the reward training, for the experiment with the most extreme incentive towards the reward.

We observe as for MNIST some mode collapse, although less pronounced here. Since the pretrained model has been trained with label conditioning for CIFAR-10, it is possible that this phenomenon could be a byproduct of this pretraining feature.

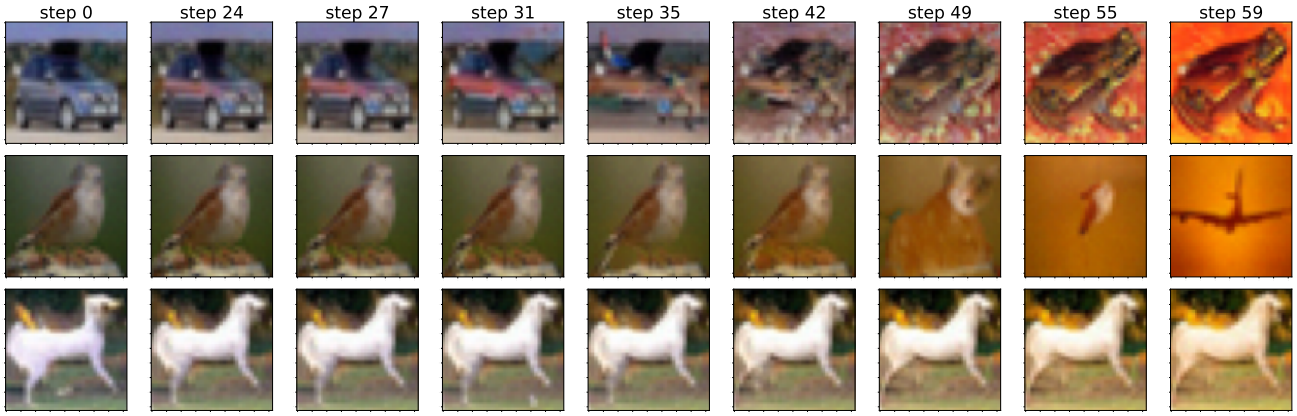


Figure 17: Reward training for a model pretrained on CIFAR-10. The reward favors **redder** images ($\lambda > 0$, $\beta > 0$). Selected examples are shown coming from a single experiment with $\lambda/\beta = 1,000$. All images are re-sampled at the same selected steps of the Implicit Diffusion algorithm, as explained in Appendix C.2.

Additional figures for LSUN. As for other datasets, we report in Figure 18 metrics on the rewards and KL divergence with respect to the original distribution.

Extensions. We emphasize that our methodology covers a wide range of sampling processes and rewards. For instance, it could be applied to diffusions in discrete spaces for language modeling, or for more complex rewards such as aesthetic scores.

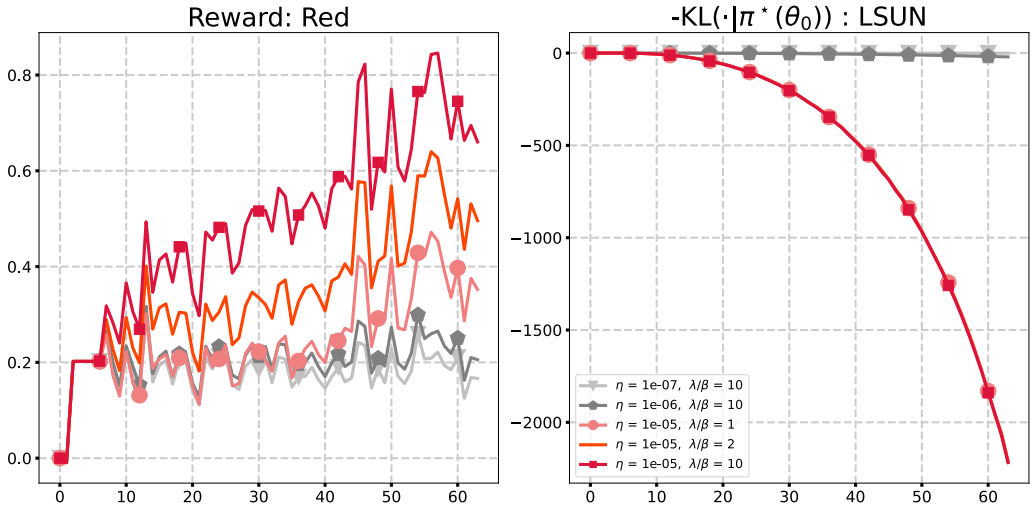


Figure 18: Score function reward training with **Implicit Diffusion** pretrained on LSUN for various $\lambda > 0$ (brighter). **Left:** Reward, average brightness of image. **Right:** Divergence w.r.t. the original pretrained distribution.

D ADDITIONAL RELATED WORK

Reward finetuning of denoising diffusion models. A large body of work has recently tackled the task of finetuning denoising diffusion models, with various point of views. Wu et al. (2023b) update weight parameters in a supervised fashion by building a high-reward dataset, then using score matching. Other papers use reinforcement learning approaches to finetune the parameters of the model (Dvijotham et al., 2023; Fan et al., 2023; Black et al., 2024). Closer to our approach are works that propose finetuning of denoising diffusion models by backpropagating through sampling (Watson et al., 2022; Dong et al., 2023; Wallace et al., 2023; Clark et al., 2024). However, they sample only once (Lee et al., 2023), or use a nested loop approach (described in Section 3.1) and resort to implementation techniques such as gradient checkpointing or gradient rematerialization to limit the memory burden. We instead depart from this point of view and propose a **single-loop** approach. Furthermore, our approach is much more general than denoising diffusion models and includes any iterative sampling algorithm such as Langevin sampling.

We emphasize that the finetuning approach differs from guidance of diffusion models (see, e.g., Dhariwal and Nichol, 2021; Graikos et al., 2022; Hertz et al., 2023; Kwon et al., 2023; Wu et al., 2023a; Zhang et al., 2023). In the latter case, the sampling scheme is modified to bias sampling towards maximizing the reward. On the contrary, finetuning directly modifies the weights of the model without changing the sampling scheme. Both approaches are complementary, and it can happen that in practice people prefer to modify the weights of the model rather than the sampling scheme: e.g., to distribute weights that take into account the reward and that can be used with any standard sampling scheme, without asking downstream users to modify their sampling method or requiring them to share the reward mechanism.

Single-loop approaches for bilevel optimization. Our single-loop approach for differentiating through sampling processes is inspired by recently-proposed single-loop approaches for bilevel optimization problems (Guo et al., 2021; Yang et al., 2021; Chen et al., 2022; Dagréou et al., 2022; Hong et al., 2023). Closest to our setting is Dagréou et al. (2022), where strong convexity assumptions are made on the inner problem while gradients for the outer problem are assumed to be Lipschitz and bounded. They also show convergence of the average of the objective gradients, akin to our Theorem 4.7. However, contrarily to their analysis, we study the case where the inner problem is a sampling problem (or infinite-dimensional optimization problem). Our methodology also extends to the non-stationary case, e.g. encompassing denoising diffusion models.

Study of optimization through Langevin dynamics in the linear case. In the case where the operator Γ can be written as an expectation w.r.t. p_t then the dynamics of θ in (11) can be seen as a McKean-Vlasov process. Kuntz et al. (2023) and Sharrock et al. (2024) propose efficient algorithms to approximate this process

using the convergence of interacting particle systems to McKean-Vlasov process when the number of particles is large. In the same setting, where Γ can be written as an expectation w.r.t. p_t, discretization of such dynamics have been extensively studied (Atchadé et al., 2017; De Bortoli et al., 2021; Xiao and Zhang, 2014; Rosasco et al., 2020; Nitanda, 2014; Tadić and Doucet, 2017). In that setting, one can leverage convergence results of the Langevin algorithm under mild assumption such as Eberle (2016) to prove the convergence of a sequence $(\theta_k)_{k\in\mathbb{N}}$ to a local minimizer such that $\nabla\ell(\theta^\star) = 0$, see (De Bortoli et al., 2021, Appendix B) for instance. Finally, Wang and Sirignano (2024) and Wang and Sirignano (2022) propose and analyze a single-loop algorithm to differentiate through solutions of SDEs. Their algorithm uses forward-mode differentiation, which does not scale well to large-scale machine learning problems.

AUTHOR CONTRIBUTION STATEMENT

PM worked on designing the methodology, implemented the codebase for experiments, proved theoretical guarantees for proposed method, contributed importantly to writing the paper. AK contributed to designing the methodology, worked on proving theoretical guarantees, made some contributions to the paper. PB, MB, VDB, AD, FL, CP (by alphabetical order) contributed to discussions in designing the methodology, provided references, made remarks and suggestions on the manuscript and provided some help with the codebase implementation. QB proposed the initial idea, proposed the general methodology and worked on designing it, contributed to the codebase implementation, ran experiments, and contributed importantly to writing the paper.