Implicit Diffusion: Efficient Optimization through Stochastic Sampling

Pierre Marion [1] [*] Anna Korba [2] Peter Bartlett [3] Mathieu Blondel [3] Valentin De Bortoli [3] Arnaud Doucet [3]
Felipe Llinares-López [†] Courtney Paquette [3] Quentin Berthet [3]

Abstract

We present a new algorithm to optimize distributions defined implicitly by parameterized stochastic diffusions. Doing so allows us to modify the outcome distribution of sampling processes by optimizing over their parameters. We introduce a general framework for first-order optimization of these processes, that performs jointly, in a single loop, optimization and sampling steps. We showcase it in training and finetuning applications.

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

In [2], 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

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

We present in this short version the main ideas of this work, and refer to it for further details and theoretical results.

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).

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 $\mathcal{P}$ [3]. 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 [4].

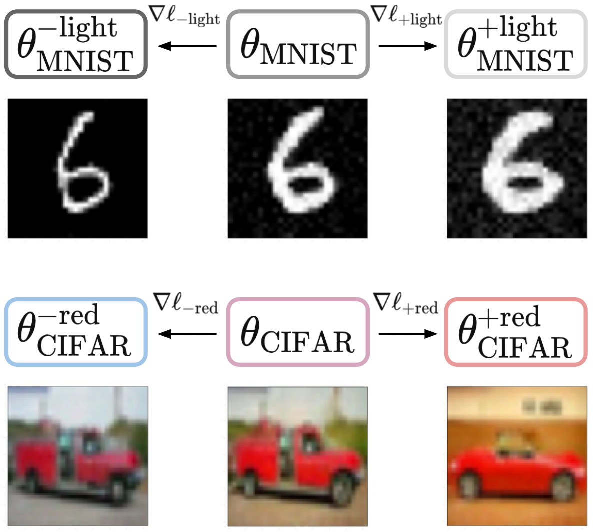
$\theta^{-\text{light}}_{\text{MNIST}}$
$\nabla\ell_{-\text{light}}$
θ_{MNIST}
$\nabla\ell_{+\text{light}}$
$\theta^{+\text{light}}_{\text{MNIST}}$
$\theta^{-\text{red}}_{\text{CIFAR}}$
$\nabla\ell_{-\text{red}}$
θ_{CIFAR}
$\nabla\ell_{+\text{red}}$
$\theta^{+\text{red}}_{\text{CIFAR}}$

Figure 1. Optimizing through sampling with **Implicit Diffusion** to finetune denoising diffusion models. The reward is the average brightness for MNIST and the red channel average for CIFAR-10.

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.

Main Contributions. In this work, we introduce the algorithm of **Implicit Diffusion**, an effective and principled technique for optimizing through a sampling operation. It allows us to train or finetune models that are used to generate samples. Our main contributions are the following:

- 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 [2].
- We show in Section 5 its performance and efficiency in **experimental settings**. Applications include finetuning denoising diffusions and training energy-based models.

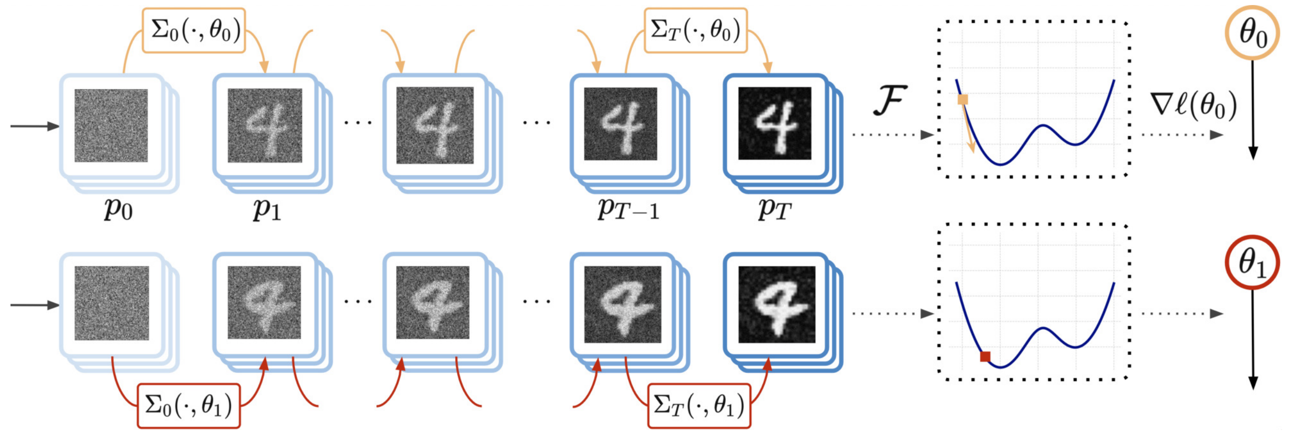
$\Sigma_0(\cdot,\theta_0)$
$\Sigma_T(\cdot,\theta_0)$
p_0
p_1
p_{T-1}
p_T
$\mathcal{F}$
$\nabla\ell(\theta_0)$
θ_0
θ_1
$\Sigma_0(\cdot,\theta_1)$
$\Sigma_T(\cdot,\theta_1)$

Figure 2. Illustration of one 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.

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 are mappings 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 since there is not always an explicit way to write down its dependency on θ. More formally, iterative sampling operators are 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*, that starts from $p_0 \in \mathcal{P}$ and iterates

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

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

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

$$\min_{\theta \in \mathbb{R}^p} \ell(\theta) := \min_{\theta \in \mathbb{R}^p} \mathcal{F}(\pi^\star(\theta))\,. \tag{2}$$

This formulation allows us to transform a problem over distributions in $\mathcal{P}$ into a finite-dimensional problem over $\theta \in \mathbb{R}^p$. Optimizing a loss over θ allows for convenient post-optimization sampling: for some $\theta_{\text{opt}} \in \mathbb{R}^d$ obtained by solving problem (2) one can sample from $\pi^\star(\theta_{\text{opt}})$. This is the common paradigm in model finetuning.

2.2. Examples

Langevin dynamics. Langevin dynamics [5] are defined by the stochastic differential equation (SDE)

$$\mathrm{d}X_t = -\nabla_1 V(X_t, \theta)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t\,, \tag{3}$$

where V and $\theta \in \mathbb{R}^p$ are such that this SDE has a solution for $t > 0$ that converges in distribution. We consider in this case $\Sigma : \theta \mapsto \pi^\star(\theta)$ the limiting distribution of X_t when $t \to \infty$, given by the Gibbs distributions

$$\pi^\star(\theta)[x] = \exp(-V(x, \theta))/Z_\theta\,. \tag{4}$$

Denoising diffusion. Denoising diffusion [1], [6], [7] consists in running the SDE, for $Y_0 \sim \mathcal{N}(0, I)$,

$$\mathrm{d}Y_t = \{Y_t + 2s_\theta(Y_t, T-t)\}\mathrm{d}t + \sqrt{2}\mathrm{d}B_t\,, \tag{5}$$

where $s_\theta : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ is a parameterized *score function*. Its aim is to reverse a forward Ornstein–Uhlenbeck 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 [8], which allows approximate sampling from p_{data}.

Two optimization objectives $\mathcal{F}$ are of particular interest in this case: for some reward $R : \mathbb{R}^d \to \mathbb{R}$ over our samples, we consider $\mathcal{F}(p) := -\mathbb{E}_{x\sim p}[R(x)]$. Second, to approximate a reference distribution p_{ref} with sample access, it is possible to take $\mathcal{F}(p) := \mathrm{KL}(p_{\text{ref}} \,||\, p)$. We also consider linear combination of these objectives.

3. Methods

Solving the optimization problem (2) with first-order methods presents several challenges, that we review here. We then introduce an overview of our approach, before getting in the details of our proposed algorithms.

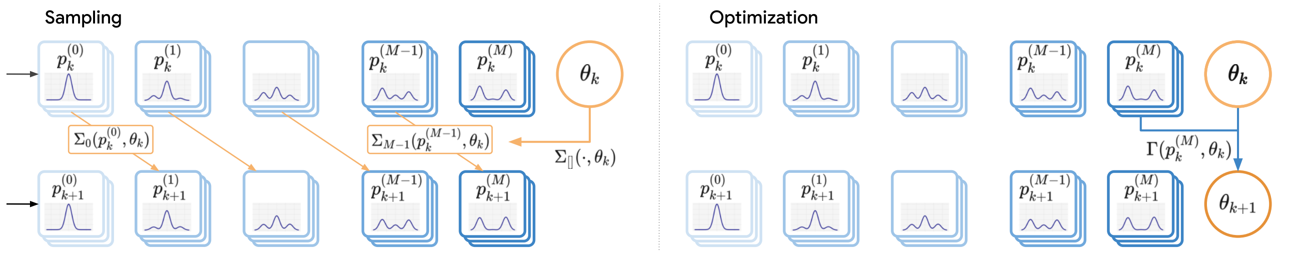

Figure 3. Illustration of the Implicit Diffusion optimization algorithm, in the finite time setting. Left: Sampling - one step of the parameterized sampling scheme is applied in parallel to all distributions in the queue. Right: Optimization - the last element of the queue is used to compute a gradient for the parameter.

3.1. Overview

Estimation of gradients through sampling. Applying a first-order method to (2) requires computing and evaluating gradients of $\ell := \mathcal{F} \circ \pi^\star$. There is no closed form for ℓ, the gradient must be evaluated in another fashion, and we consider the following setting.

Definition 3.1 (Implicit gradient estimation)**.** We consider settings where $\Sigma_s, \mathcal{F}$ are such that the gradient of ℓ can be *implicitly estimated*: there is a function $\Gamma : \mathcal{P} \times \mathbb{R}^p \to \mathbb{R}^p$ such that $\nabla \ell(\theta) = \Gamma(\pi^\star(\theta), \theta)$. □

Beyond nested-loop approaches. Sampling from $\pi^\star(\theta)$ requires iterations of the sampling process Σ_s, suggesting a nested loop: at each optimization step k, running an inner loop for a large amount T of steps of Σ_s as in (1) to evaluate a gradient. It can be inefficient: it requires solving the inner sampling problem *at each optimization step*. Further, nested loops are typically impractical with modern accelerator-oriented computing hardware. We step away from the nested-loop paradigm and *jointly* iterate on both the sampling problem (inner problem), and the optimization problem over $\theta \in \mathbb{R}^p$ (outer $\mathcal{F}$).

3.2. Methods for gradient estimation through sampling

Several methods are used in our work to perform implicit gradient estimation as in Definition 3.1. We describe in our long version [2] several methods based on direct analytical derivation, implicit differentiation, and the adjoint method.

3.3. Implicit Diffusion optimization algorithm

Our **proposed approach** is to circumvent solving the inner problem (i.e. sample exactly or approximately from $\pi^\star(\theta_k)$ at each update of θ_k). We propose 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 depending on the current parameter θ_k, as detailed in Algorithm 1.

Algorithm 1 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)
  end for
output $\theta_K$
```

This point of view is well-suited for stationary processes with infinite-time horizon. We also adapt our approach to a finite time setting.

Finite time-horizon: queuing trick. When $\pi^\star(\theta)$ is obtained or approximated by a large, but *finite* number T of iterations of the operator Σ_s, we propose to leverage hardware parallelism to evaluate in parallel several, say M, dynamics of the distribution p_k, through a queue of length M. We present for simplicity in Figure 3 the case where $M = T$. At each step, the M-th element of the queue $p_k^{(M)}$ provides a distribution to update θ through evaluation of Γ.

Algorithm 2 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)
  end for
output $\theta_K$
```

Updating a single dynamic of probabilities $(p_k)_{k\geq 0}$ would only provide a single gradient estimate (after T sampling steps). Moreover, leveraging parallelism, the running time of our algorithm is $\mathcal{O}(K)$, gaining a factor of T compared to the nested-loop approach.

4. Theoretical analysis

We include in [2] a theoretical analysis of our algorithms.

5. Experiments

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

5.1. Reward training of Langevin processes

We consider the case of an explicit $V(\cdot, \theta)$ and reward $R(x)$. We run six sampling algorithms, including the infinite time-horizon version of **Implicit Diffusion** (Algorithm 1), all starting from $p_0 = \mathcal{N}(0, I_d)$ and for $K = 5,000$ steps.

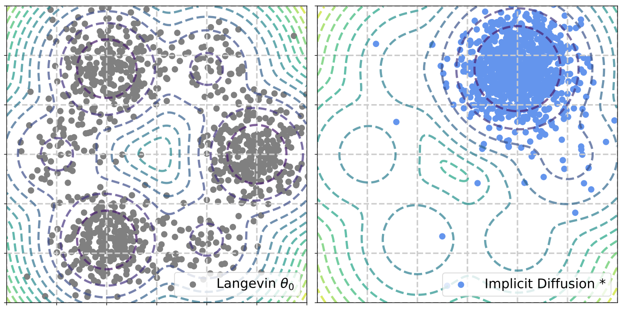

Figure 4. Contours and samples for $\pi^\star(\theta_{\text{opt}})$ and Implicit Diffusion.

Both qualitatively (Figure 4) and quantitatively (Figure 5), we observe that our approach efficiently optimizes through sampling. We analyze their performance both in terms of *steps* (number of optimization steps–updates in θ) and *gradient evaluations* (number of sampling steps).

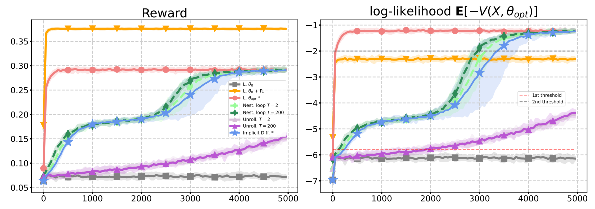

Figure 5. Metrics for reward training of Langevin processes (see Section 5.1), 10 runs. **Left:** Reward on the sample distribution, at each outer objective step, averaged on a batch. **Right:** Log-likelihood of $\pi^\star(\theta_{\text{opt}})$ on the sample distribution, at each outer step, averaged on a batch–higher is better.

After K optimization steps, our algorithm yields both $\theta_{\text{opt}} := \theta_K$ and a sample $\hat{p}_K$ approximately from $\pi^\star(\theta_{\text{opt}})$. We compare our approach with several baselines (see Appendix A).

5.2. Reward training of denoising diffusion models

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 model pretrained on these datasets, 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. [9] and references therein, for positive and negative values of λ. We run **Implicit Diffusion** using the finite time-horizon variant described in Algorithm 2, applying the adjoint method on SDEs for gradient estimation. We report selected samples generated by $\pi^\star(\theta_t)$, as well as reward and KL divergence estimates (see Figures 1, 6 and 7).

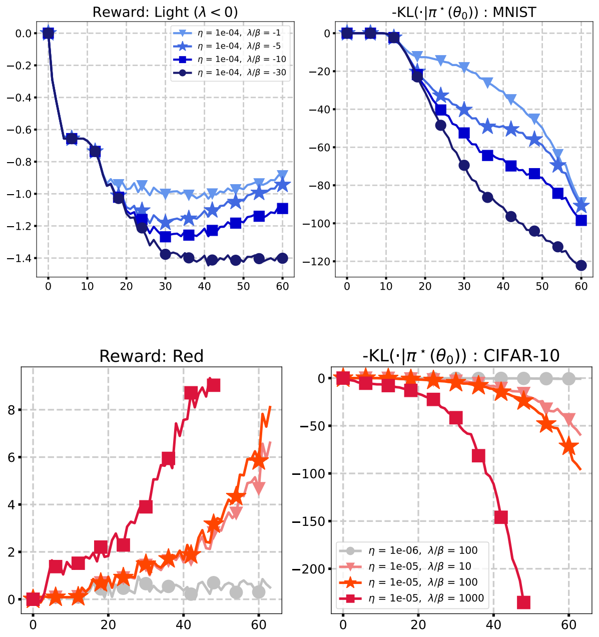

Figure 6. Reward training with **Implicit Diffusion** for various λ, η. For each dataset, we plot the evolution of the reward (left) and of the divergence w.r.t. the distribution after pretraining (right).

We report results on models pretrained on the image datasets MNIST [10], CIFAR-10 [11], and LSUN (bedrooms) [12] (see Appendix A). 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 in the case of MNIST).

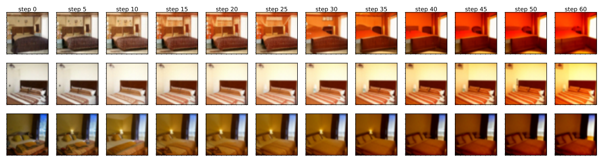

Figure 7. Samples of reward training after pretraining on LSUN ($\lambda/\beta = 10$). The reward incentives for redder images. Images are re-sampled with the same seed every five steps (see Appendix A.2).

References

[1] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 6840–6851.

[2] P. Marion, A. Korba, P. Bartlett, *et al.*, "Implicit diffusion: Efficient optimization through stochastic sampling," *arXiv preprint arXiv:2402.05468*, 2024.

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

[4] R. Jordan, D. Kinderlehrer, and F. Otto, "The variational formulation of the fokker–planck equation," *SIAM journal on mathematical analysis*, vol. 29, no. 1, pp. 1–17, 1998.

[5] G. O. Roberts and R. L. Tweedie, "Exponential convergence of langevin distributions and their discrete approximations," *Bernoulli*, pp. 341–363, 1996.

[6] A. Hyvärinen, "Estimation of non-normalized statistical models by score matching," *Journal of Machine Learning Research*, vol. 6, no. 24, pp. 695–709, 2005.

[7] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural Computation*, vol. 23, no. 7, pp. 1661–1674, 2011.

[8] B. D. O. Anderson, "Reverse-time diffusion equation models," *Stochastic Processes and their Applications*, vol. 12, no. 3, pp. 313–326, 1982.

[9] D. M. Ziegler, N. Stiennon, J. Wu, *et al.*, "Fine-tuning language models from human preferences," *arXiv preprint arXiv:1909.08593*, 2019.

[10] Y. LeCun and C. Cortes, *MNIST handwritten digit database*, 1998. [Online]. Available: `http://yann.lecun.com/exdb/mnist/`.

[11] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[12] 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.

[13] J. Bolte, E. Pauwels, and S. Vaiter, "One-step differentiation of iterative algorithms," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36, Curran Associates, Inc., 2023, pp. 77 089–77 103.

[14] 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.

[15] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 8780–8794.

[16] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[17] J. Bradbury, R. Frostig, P. Hawkins, *et al.*, *JAX: Composable transformations of Python+NumPy programs*, version 0.3.13, 2018. [Online]. Available: `http://github.com/google/jax`.

[18] 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*, Springer, 2015, pp. 234–241.

[19] 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 *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[20] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 18–24 Jul 2021, pp. 8162–8171.

[21] 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)*, Oct. 2023, pp. 2096–2105.

[22] K. D. Dvijotham, S. Omidshafiei, K. Lee, *et al.*, "Algorithms for optimal adaptation of diffusion models to reward functions," in *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, 2023.

[23] Y. Fan, O. Watkins, Y. Du, *et al.*, "Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models," *arXiv preprint arXiv:2305.16381*, 2023.

[24] 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.

[25] 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.

[26] H. Dong, W. Xiong, D. Goyal, *et al.*, "RAFT: Reward ranked finetuning for generative foundation model alignment," *Transactions on Machine Learning Research*, 2023, ISSN: 2835-8856.

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

[28] K. Lee, H. Liu, M. Ryu, *et al.*, "Aligning text-to-image models using human feedback," *arXiv preprint arXiv:2302.12192*, 2023.

[29] A. Graikos, N. Malkin, N. Jojic, and D. Samaras, "Diffusion models as plug-and-play priors," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 14 715–14 728.

[30] 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. [Online]. Available: `https://openreview.net/forum?id=_CDixzkzeyb`.

[31] M. Kwon, J. Jeong, and Y. Uh, “Diffusion models already have a semantic latent space,” in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: `https://openreview.net/forum?id=pd1P2eUBVfq`.

[32] Q. Wu, Y. Liu, H. Zhao, *et al.*, “Uncovering the disentanglement capability in text-to-image diffusion models,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 1900–1910.

[33] 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.

[34] 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.

[35] J. Yang, K. Ji, and Y. Liang, “Provably faster algorithms for bilevel optimization,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 13 670–13 682, 2021.

[36] 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*, PMLR, 2022, pp. 2466–2488.

[37] M. Dagréou, P. Ablin, S. Vaiter, and T. Moreau, “A framework for bilevel optimization that enables stochastic and global variance reduction algorithms,” in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 26 698–26 710.

[38] 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*, vol. 33, no. 1, pp. 147–180, 2023.

[39] 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*, PMLR, 2023, pp. 5134–5180.

[40] L. Sharrock, D. Dodd, and C. Nemeth, “Tuning-free maximum likelihood training of latent variable models via coin betting,” in *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, S. Dasgupta, S. Mandt, and Y. Li, Eds., ser. Proceedings of Machine Learning Research, vol. 238, PMLR, Feb. 2024, pp. 1810–1818.

[41] Y. F. Atchadé, G. Fort, and E. Moulines, “On perturbed proximal gradient algorithms,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 310–342, 2017.

[42] 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*, vol. 31, pp. 1–18, 2021.

[43] L. Xiao and T. Zhang, “A proximal stochastic gradient method with progressive variance reduction,” *SIAM Journal on Optimization*, vol. 24, no. 4, pp. 2057–2075, 2014.

[44] L. Rosasco, S. Villa, and B. C. Vũ, “Convergence of stochastic proximal gradient algorithm,” *Applied Mathematics & Optimization*, vol. 82, pp. 891–917, 2020.

[45] A. Nitanda, “Stochastic proximal gradient descent with acceleration techniques,” *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[46] V. B. Tadić and A. Doucet, “Asymptotic bias of stochastic gradient search,” *Annals of Applied Probability*, vol. 27, no. 6, pp. 3255–3304, 2017.

[47] A. Eberle, “Reflection couplings and contraction rates for diffusions,” *Probability theory and related fields*, vol. 166, pp. 851–886, 2016.

[48] Z. Wang and J. Sirignano, “Continuous-time stochastic gradient descent for optimizing over the stationary distribution of stochastic differential equations,” *Mathematical Finance*, vol. 34, no. 2, pp. 348–424, 2024.

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

Appendix

A. Experimental details

We provide here details about our experiments in Section 5.

A.1. Langevin processes

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 $\theta \in \mathbb{R}^6$,

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

where Z is an absolute renormalization constant that is independent of θ. This 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 4, and used to compute the values of reward and likelihood reported in Figure 5. We also display in Figure 10 the dynamics of the probabilities throughout these algorithms.

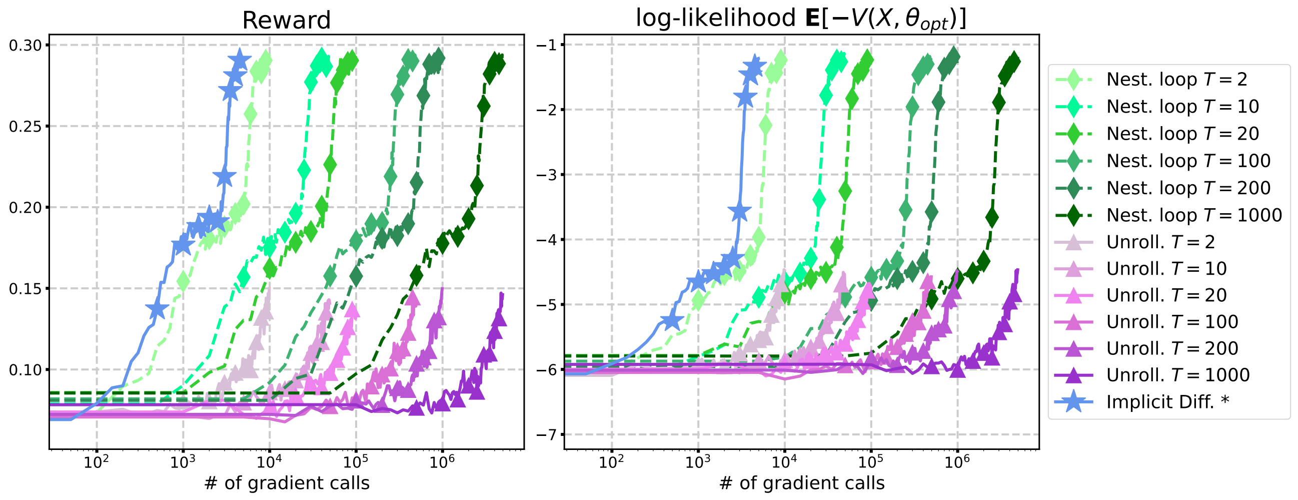

Figure 8. Comparison between Implicit Diffusion, nested loop algorithm and unrolling algorithm, for various number of inner steps T. The x-axis is the total number of gradient evaluations (roughly equal to the number of optimization steps multiplied by the number of inner loop steps T). **Left**: Evolution 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

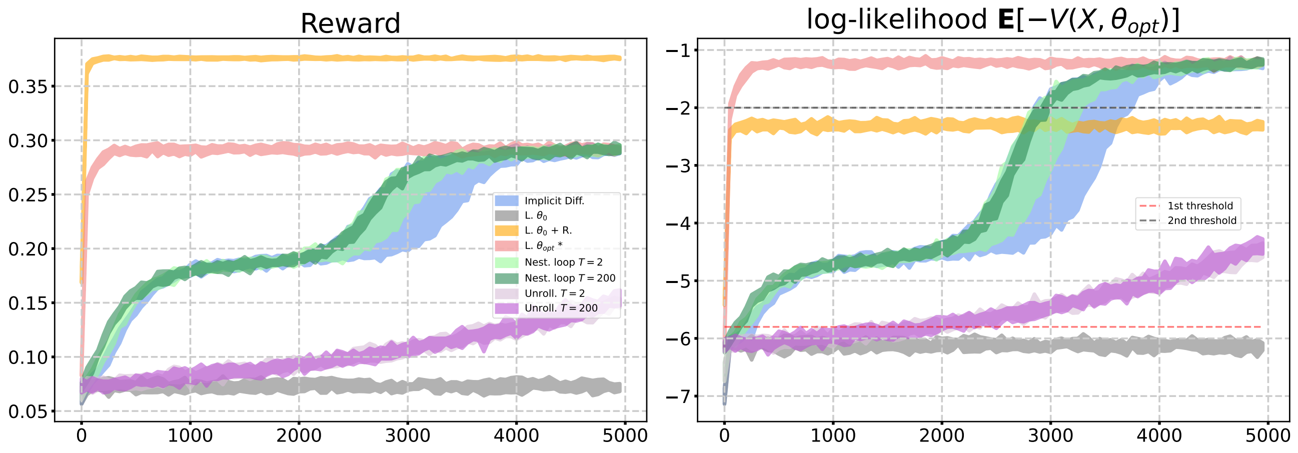

Figure 9. Confidence intervals for metrics for reward training of Langevin processes. **Left**: Evolution of the reward. **Right**: Evolution of the log-likelihood.

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 1), 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 a nested-loop algorithm 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 [13]. It has been applied successfully to denoising diffusions [14].

- Langevin θ_0 + R (▼): This is a discrete-time process approximating 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 [15]. Note that this approach 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 10 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 5. 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 8 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 5 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 9 shows the same plot with confidence intervals only (without plotting the average value).

Training from scratch. We present in Figure A.1 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 11. 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 $T = 40,000$ steps with a batch of size $1,000$.

A.2. Denoising diffusion models

We start by giving 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. We use the Adam optimizer [16], with various values for the learning rate (see e.g. Figure 6). The code was implemented in JAX [17]. As mentioned in the main text, we use a U-Net model [18].

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 [19] 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 [20]
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 [20]
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 15 metrics on the rewards and KL divergence with respect to the original distribution, in the case where $\lambda > 0$. As in Figure 6, we observe the competition between the reward and the divergence with respect to the distribution after pretraining. We also display in Figures 13 and 14 some 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 present samples for negative and positive λ, rewarding respectively darker and brighter images. We emphasize that these samples are realized at different steps of our algorithm 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.

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 16, akin to Figures 13 and 14, 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.

Additional figures for LSUN. As for other datasets, we report in Figure 17 metrics on the rewards and KL divergence with respect to the original distribution.

B. 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. [21] 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 [22]–[24]. Closer to our approach are works that propose finetuning of denoising diffusion models by backpropagating through sampling [14], [25]–[27]. However, they sample only once [28], 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., [15], [29]–[33]). 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 [34]–[38]. Closest to our setting is [37], 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 on of our theoretical results. 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 can be seen as a McKean-Vlasov process. [39] and [40] 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 [41]–[46]. In that setting, one can leverage convergence results of the Langevin algorithm under mild assumption such as [47] 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 [42], Appendix B for instance. Finally, [48] and [49] 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.

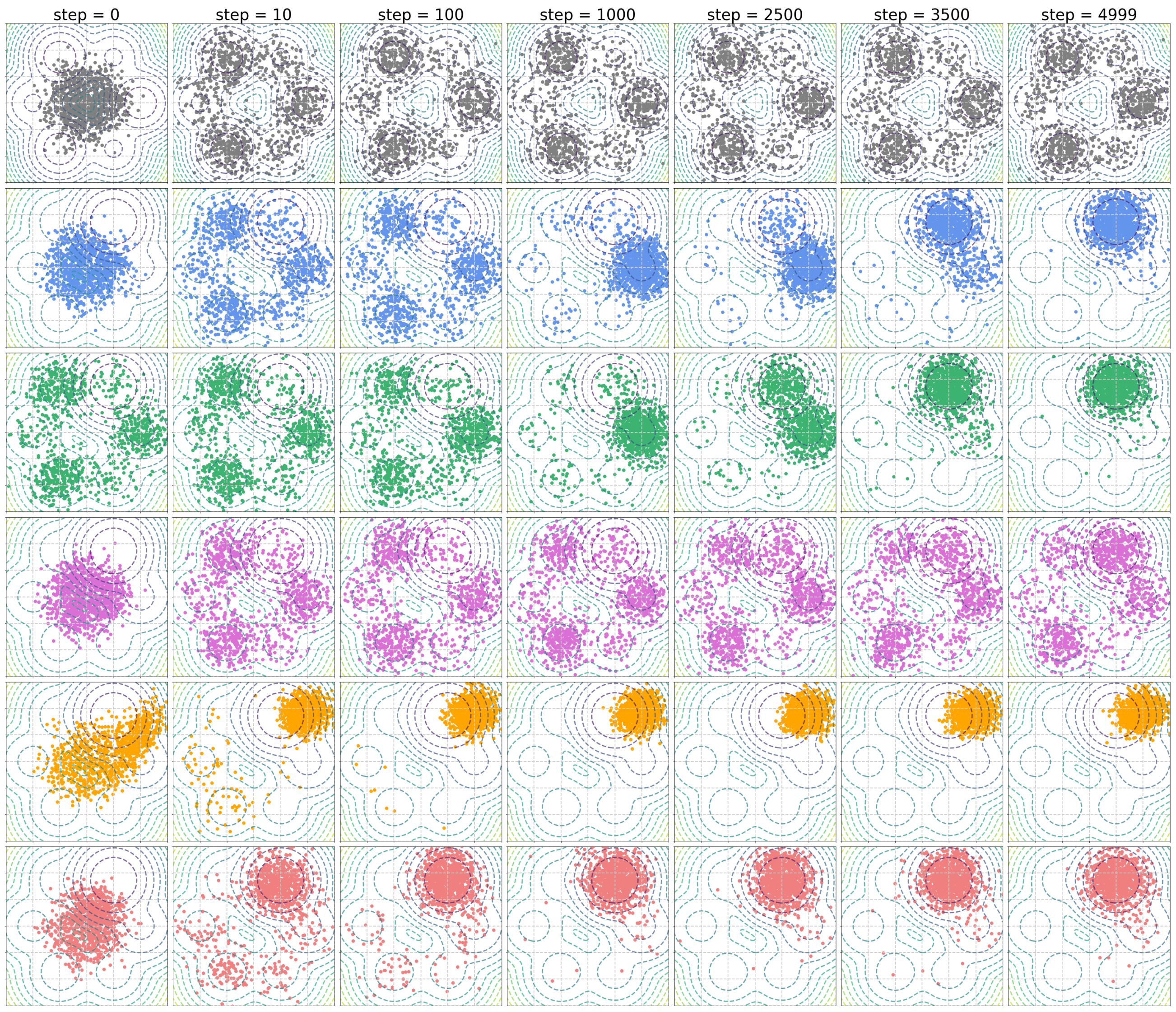

Figure 10. Dynamics of samples for four sampling algorithms after different time steps (for instance, the first column is after one step). **First row:** Langevin θ_0 (●) with $\pi^\star(\theta_0)$ contour lines. **Second:** Implicit Diffusion (●) with $\pi^\star(\theta_{\text{opt}})$ contour lines. **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} (●).

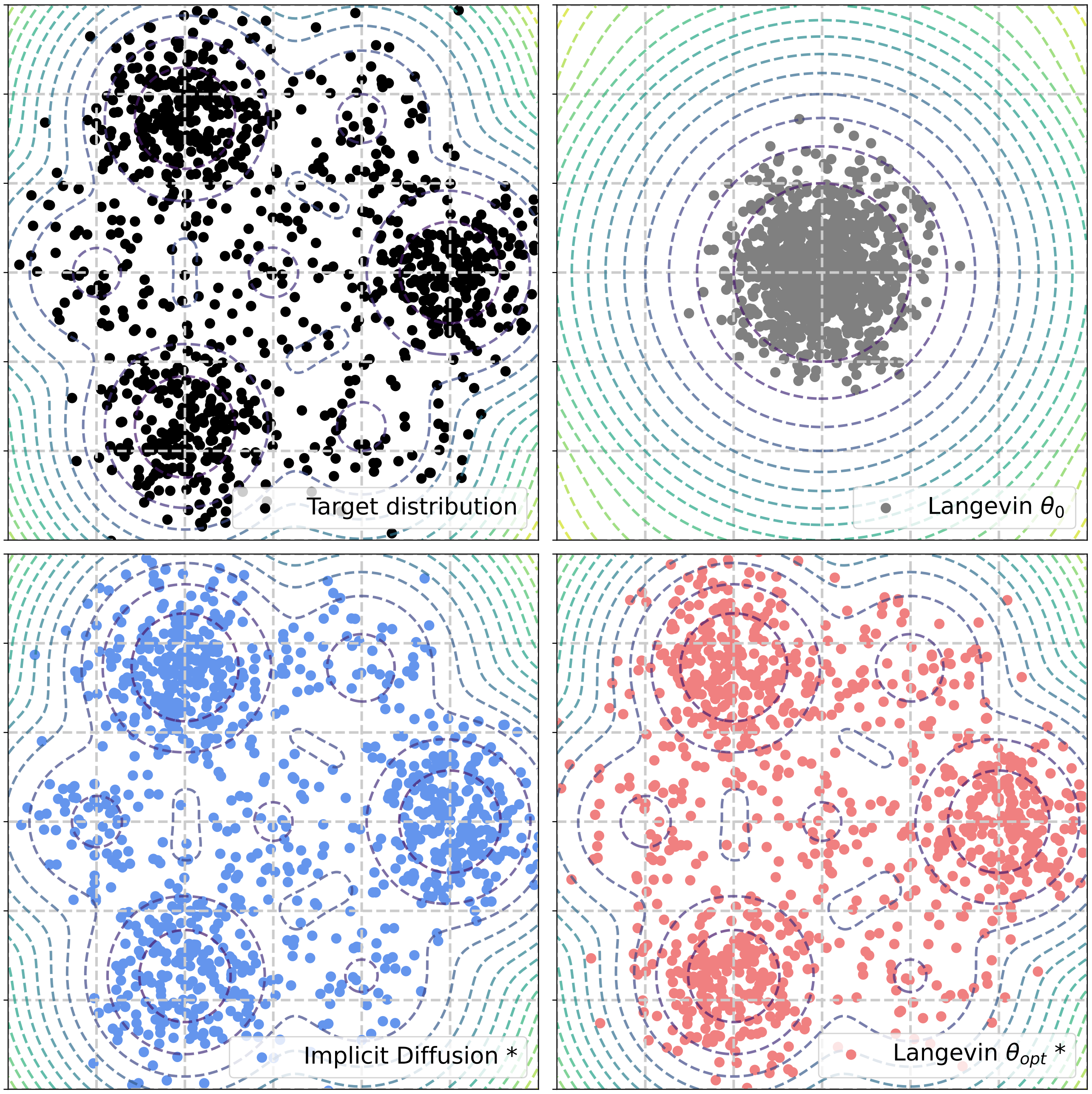

Figure 11. Contour lines and samples from sampling algorithms. We start from a model generating a standard Gaussian (top-right figure), and our goal is to learn to generate a mixture of several Gaussians (top-left figure). We use Implicit Diffusion where the reward is the KL between the target distribution 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).

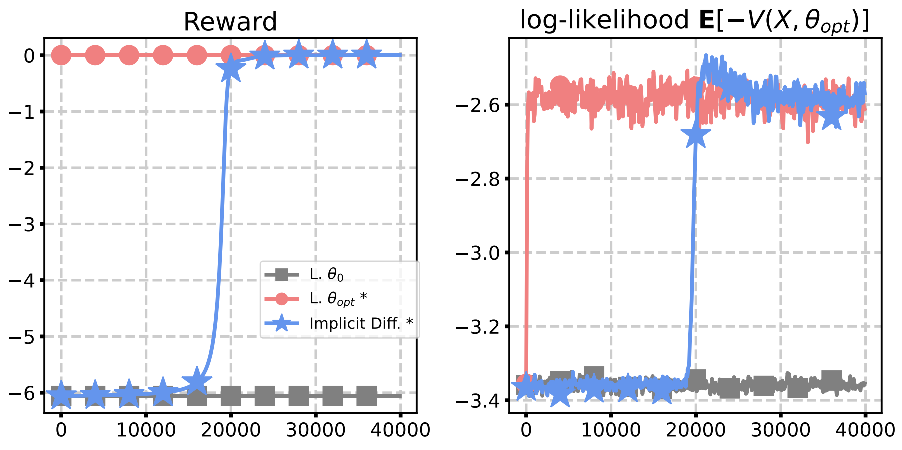

Figure 12. Evolution of the reward and of the log-likelihood of the samples for the initial model, for the Implicit Diffusion algorithm, and for the trained model after Implicit Diffusion. The reward is the (opposite of the) KL between the target distribution and the current one, so a reward equal to zero means we learnt to reproduce the target distribution.

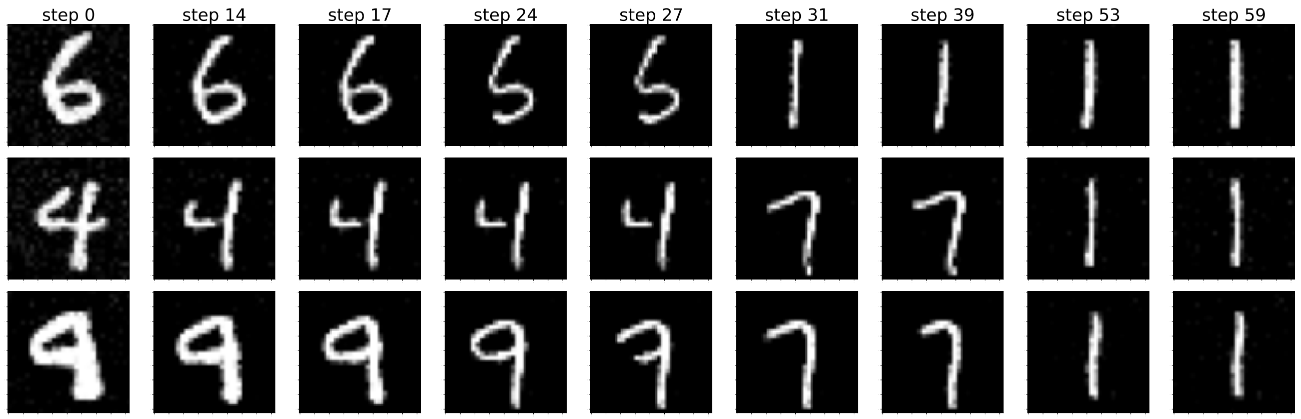

Figure 13. Reward training for a model pretrained on MNIST. The reward favors **darker** images ($\lambda < 0, \beta > 0$). Selected examples are shown coming from a single experiment with $\lambda/\beta = -30$. All digits are re-sampled at the same selected steps of the Implicit Diffusion algorithm.

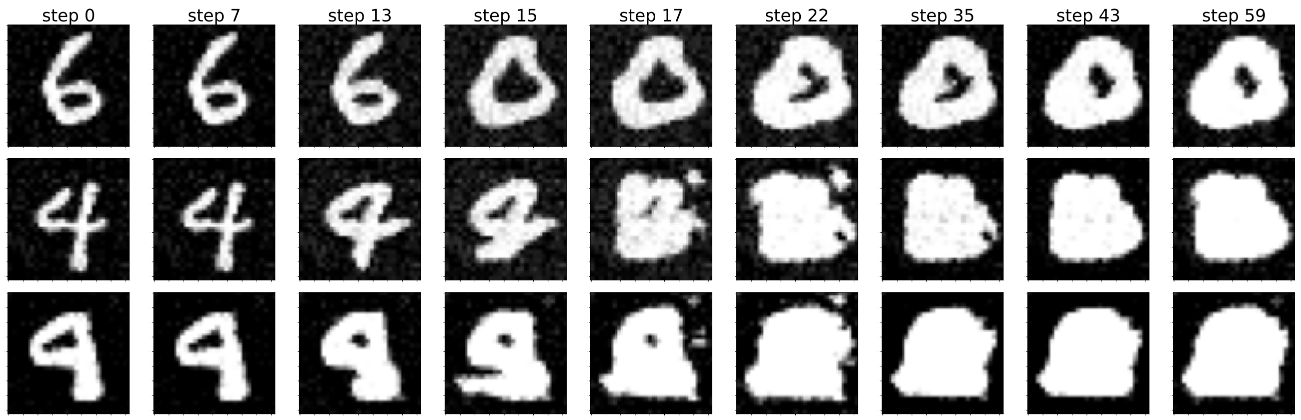

Figure 14. Reward training for a model pretrained on MNIST. The reward favors **brighter** images ($\lambda > 0, \beta > 0$). Selected examples are shown coming from a single experiment with $\lambda/\beta = 30$. All digits are re-sampled at the same selected steps of the Implicit Diffusion algorithm.

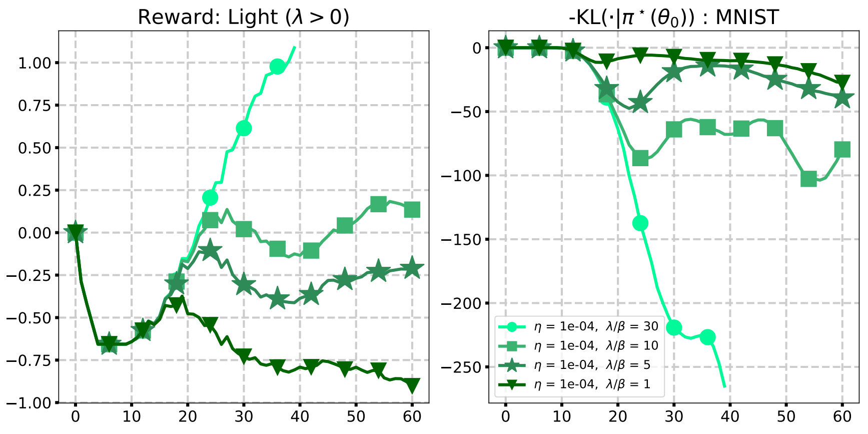

Figure 15. Score function reward training with **Implicit Diffusion** pretrained on MNIST for various $\lambda > 0$ (brighter). **Left:** Reward, average brightness of image. **Right:** Divergence w.r.t. the original pretrained distribution.

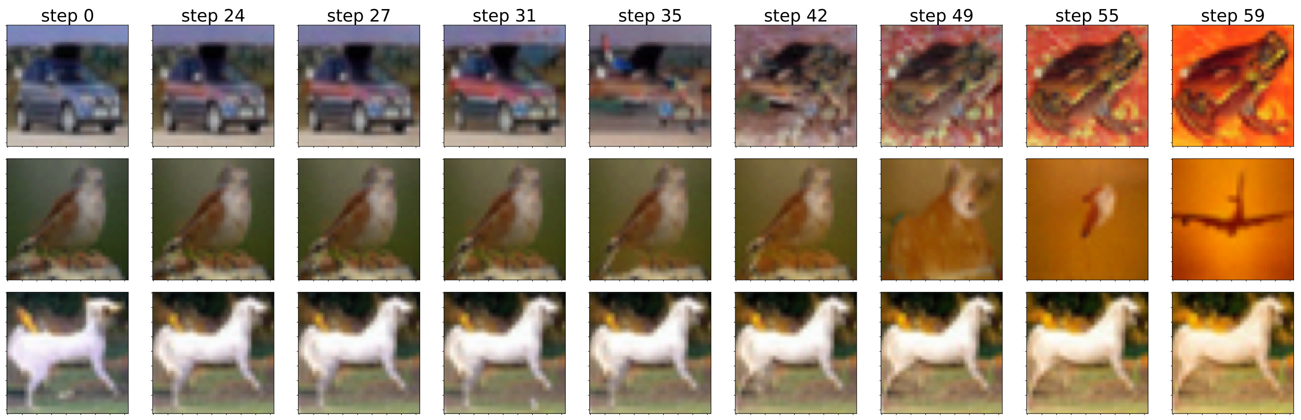

Figure 16. Reward training for a model pretrained on CIFAR-10. The reward favors **redder** images ($\lambda > 0$, $\beta > 0$). Selected examples are shown coming from a single experiment with $\lambda/\beta = 1,000$. All images are re-sampled at the same selected steps of the Implicit Diffusion algorithm, as explained in Appendix A.2.

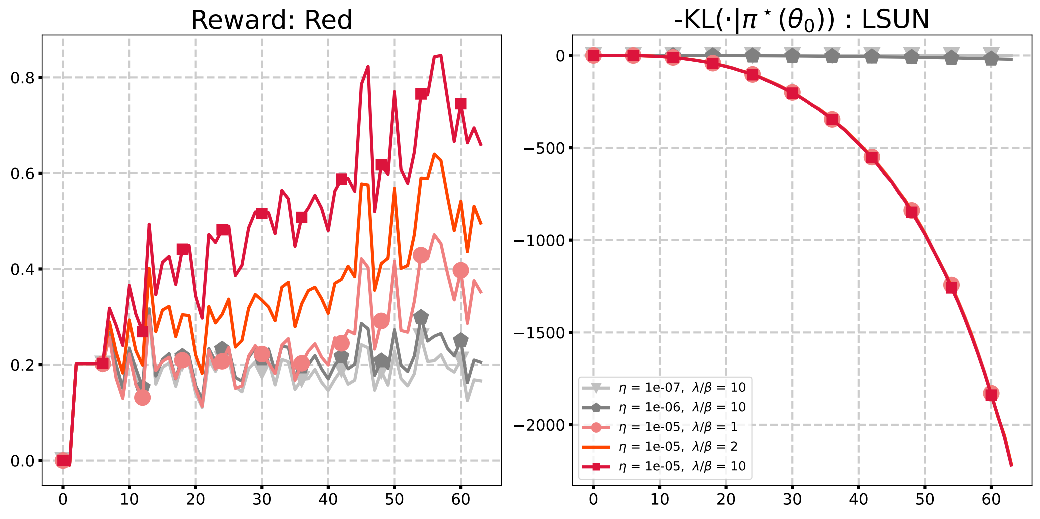

Figure 17. Score function reward training with **Implicit Diffusion** pretrained on LSUN for various $\lambda > 0$ (brighter). **Left:** Reward, average brightness of image. **Right:** Divergence w.r.t. the original pretrained distribution.