Single-Step Consistent Diffusion Samplers

Anonymous Author(s)
Affiliation
Address
email

Abstract

Sampling from unnormalized target distributions is a fundamental yet challenging task in machine learning and statistics. Existing sampling algorithms typically require many iterative steps to produce high-quality samples, leading to high computational costs that limit their practicality in time-sensitive or resource-constrained settings. In this work, we introduce *consistent diffusion samplers*, a new class of samplers designed to generate high-fidelity samples in a single step. We first propose Consistency-Distilled Diffusion Samplers (CDDS), which demonstrates that consistency distillation can be accomplished within sampling contexts in the absence of pre-collected training datasets. To eliminate the need for a pre-trained sampler, we further propose Self-Consistent Diffusion Samplers (SCDS), which performs self-distillation during training. SCDS learns to perform diffusion sampling and to skip intermediate steps via a self-consistency loss. Through extensive experiments on a variety of synthetic and real-world unnormalized distributions, we show that our approaches yield high-fidelity samples using less than 1% of the network evaluations required by traditional diffusion samplers.

1 Introduction

Sampling from densities of the form

$$p_{\text{target}} = \frac{\rho}{Z}, \quad \text{with} \quad Z = \int_{\mathbb{R}^D} \rho(X) \mathrm{d}X \tag{1}$$

with ρ evaluable pointwise but Z intractable, is a central problem in machine learning [21, 30] and statistics [3, 31], and has applications in scientific fields like physics [1, 33, 53], chemistry [17, 23, 24], and many other fields involving probabilistic models.

Many established sampling algorithms are inherently iterative, with the accuracy of the final samples depending heavily on the number of steps. Classical Markov chain Monte Carlo (MCMC) methods asymptotically converge to the target distribution with an infinite number of steps [27, 39]. Nonetheless, the finite number of feasible steps in real-world applications means that MCMC can only provide an approximation and cannot guarantee an exact solution. As an improvement, more recent diffusion-samplers [7, 50, 55] guarantee convergence in a finite number of steps. However, they often necessitate hundreds of iterations to yield high-quality samples. Such iterative samplers tend to suffer from slow mixing, making them still impractical for use in large models and resource-limited scenarios.

From another perspective, recent work on diffusion generative models [22, 44, 47, 48] has demonstrated the feasibility of fewer-step sampling. This can be achieved using knowledge distillation [41, 46] or consistency training [46], potentially enabling even single-step generation. However, directly applying the distillation techniques to distillate existing diffusion samplers is challenging, as it often requires large datasets of samples that are expensive to collect in practice. Moreover, the

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

learning targets of diffusion samplers and consistency training are conflicted in some aspects. These make reducing the sampling steps of diffusion samplers a challenging problem.

In this paper, we propose *consistent diffusion samplers* to produce high-quality samples in a single step. We first show that diffusion-based samplers can be *consistently distilled* into single-step diffusion samplers and propose the Consistency-Distilled Diffusion Samplers (CDDS) approach. Instead of storing a large dataset of fully diffused samples, CDDS exploits incomplete trajectories and noisy samples encountered during the diffusion process, hence reducing the unnecessary costs. We further introduce the Self-Consistent Diffusion Sampler (SCDS) method that does not require a pretrained diffusion sampler. Instead, it fully amortizes exploration by jointly learning both diffusion sampling and large cut off steps that match the outcome of paths of small steps. This enables single-step sampling yet retains the option to refine samples through multiple iterations if desired, subsuming existing diffusion-based approaches. Our contributions can be summarized as follows:

- We show that diffusion-based samplers for unnormalized distributions can be effectively distilled into single-step consistent samplers without pre-collecting large datasets of samples.
- We introduce a self-consistent diffusion sampler that learns to perform single-step sampling by jointly training diffusion-based transitions and large shortcut steps via a self-consistency criterion. This method only trains one neural network and does not require pretrained samplers or high-quality datasets.
- Through extensive evaluations on synthetic and real unnormalized distributions, we demonstrate that our method delivers competitive sample quality while drastically reducing sampling steps.

2 Related Work

Monte Carlo-based Samplers. Monte Carlo-based Samplers, such as Markov chain Monte Carlo (MCMC), are a classical approach for sampling from unnormalized target densities. The key idea is to construct a Markov chain whose stationary distribution matches the target distribution [10]. Prominent examples include the Metropolis-Hastings algorithm [20, 29], Gibbs sampling [19], and Langevin dynamics [35, 40]. By exploiting geometric structure in the target distribution, Hamiltonian Monte Carlo [10, 12, 14, 27] often leads to more efficient exploration. To address scalability challenges in high-dimensional or large-dataset scenarios, stochastic gradient MCMC variants [12, 52, 56, 57] have been introduced. Although these MCMC methods reduce per-step computational costs or improve mixing, they remain inherently iterative, requiring many transitions to yield high-quality samples.

Diffusion-based Samplers. An alternative viewpoint frames sampling as an optimal control task [7, 37, 38, 55], where controlled stochastic differential equations transport an initial distribution to the target via a Schrödinger bridge [42, 43]. This approach has recently motivated numerous diffusion-based sampling methods [11, 13, 18, 36, 50]. Further improvements have been explored through Hamiltonian dynamics [8], intermediate resampling strategies [11], and physics-informed neural networks to evolve densities [49]. Recent methods such as CMCD [51] jointly optimize forward and backward diffusion dynamics. Additionally, theoretical connections between GFlowNets [4, 5] and diffusion-based sampling have been investigated [6, 54]. For an extensive review of relevant metrics and baseline samplers, see [9]. Current diffusion methods partially amortize sampling costs in training but still require iterative inference-time generation. In this work, we fully amortize exploration during training, enabling efficient single-step sampling at inference.

Consistent Generative Models. Recent work in generative modeling has introduced the notion of consistency. Consistency models [26, 45, 46] learn direct mapping from any intermediate state to the terminal state. Progressive distillation [28, 41] incrementally distills a trained diffusion model into a more efficient version that takes half as many steps. Shortcut models [16] apply this distillation principle during training, enabling direct learning of efficient transitions without relying on a pretrained teacher. We extend this line of work to the setting of sampling from unnormalized densities, assuming only pointwise access to the target density ρ, without requiring data samples.

3 Preliminaries: Sampling via Controlled Stochastic Processes

Diffusion samplers aim to draw samples from a complex target density $p_{\text{target}} = \rho/Z$ by transporting them from a simpler prior density p_{prior}. We consider forward and reverse-time stochastic processes on $\mathbb{R}^d$ over a time interval $[0, T]$, each described by the following SDEs:

$$\mathrm{d}X_s = (\mu + \sigma u)(X_s, s)\mathrm{d}s + \sigma(s)\mathrm{d}W_s, \quad X_0 \sim \pi, \tag{2}$$

$$\mathrm{d}X_s = (-\mu + \sigma v)(X_s, T-s)\mathrm{d}s + \sigma(T-s)\mathrm{d}W_s, \quad X_0 \sim \tau, \tag{3}$$

where $u, v \in \mathcal{U} \subset C(\mathbb{R}^d \times [0,T], \mathbb{R}^d)$ are control functions, $\mu \in C(\mathbb{R}^d \times [0,T], \mathbb{R}^d)$ is a linear drift, $\sigma \in C([0,T], \mathbb{R})$ is the diffusion coefficient, and $\mathrm{d}W_s$ denote forward Brownian increments. We seek u and v such that (2) and (3) become time-reversal counterparts.

Let $\mathbb{P}^{u,\pi}$ and $\overline{\mathbb{P}}^{v,\tau}$ be the path measures induced by (2) and (3), respectively. Consider a divergence $D : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_{\geq 0}$. We aim to solve the optimization problem:

$$u^*, v^* \in \operatorname*{arg\,min}_{u,v \in \mathcal{U}} D(\mathbb{P}^{u,\pi} \mid \overline{\mathbb{P}}^{v,\tau}). \tag{4}$$

When the divergence D reaches its minimum of zero in (4), the marginal distribution at terminal time matches exactly, $\mathbb{P}_T^{u,\pi} = \tau$. Consequently, by selecting $\pi = p_{\text{prior}}$ and $\tau = p_{\text{target}}$, one can generate samples from the target distribution p_{target} through simulating (2) with the optimal control u^*.

Nelson's identity provides a local characterization of optimality conditions [2, 15, 32]. It states that the forward path measure $\mathbb{P}^{u,\pi}$ coincides with the reverse-time measure $\overline{\mathbb{P}}^{v,\tau}$ if and only if the forward drift can be expressed as the backward drift adjusted by the scale score $u(\cdot, s) = v(\cdot, s) + \sigma(s)\nabla \log \mathbb{P}_s^{u,\pi}(\cdot)$. In practice, the marginal distributions $\mathbb{P}^{u,\pi}s$ are generally intractable. Therefore, we typically approximate solutions to (4) by parameterizing the control u with a neural network u_θ. Training these models typically proceeds through the following iterative steps:

1. Simulate a batch of M trajectories $\{(X_s^{(i)})_{0 \leq s \leq T}\}$, $i = 1, \ldots, M$, using the generative process (2).
2. Compute the divergence measure and its gradient with respect to the parameters θ.
3. Update the parameters θ accordingly, and repeat the process until convergence.

The Kullback-Leibler (KL) [7, 50, 55] and the log-variance (LV) divergences are common choices [6, 34, 38]:

$$D_{\text{KL}}(\mathbb{P} \mid \mathbb{Q})(X) = \mathbb{E}\left[\log \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(X)\right] + \log Z, \quad D_{\text{LV}}(\mathbb{P} \mid \mathbb{Q})(X) = \mathbb{V}\left[\log \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(X)\right]. \tag{5}$$

The likelihood ratio appearing in (5) is given explicitly by the Radon-Nikodym derivative:

$$\log \frac{\mathrm{d}\mathbb{P}^{u,\pi}}{\mathrm{d}\overline{\mathbb{P}}^{v,\tau}} = \int_0^T (u+v) \cdot \left(u_\theta + \frac{v-u}{2} + \nabla \cdot (\sigma v - \mu)\right)\mathrm{d}s + \int_0^T (u+v)\mathrm{d}W_s + \log \frac{p_{\text{prior}}(X_0^\theta)}{p_{\text{target}}(X_T^\theta)} \tag{6}$$

where X^θ is the trajectory obtained by simulating the forward SDE (2) using the parameterized control u_θ. The log normalization constant from the target density disappears upon taking gradients, making this a practical objective for training. See [7] and Appendix A.2 of [38] for detailed derivations.

Once trained, the optimized control u_θ allows generation of samples from p_{target} through forward simulations of (2). In practice, this continuous-time process must be discretized into finite steps $0 = t_1 < t_2 < \cdots < t_N = T$, introducing a trade-off between computational cost and accuracy.

4 Consistency Distilled Diffusion Samplers

In this section, we present the *consistency distilled diffusion sampler* (CDDS) method to solve the problem of single-step sampling from unnormalized densities by distillating from a pretrained sampler. Concretely, our goal is to learn a consistency function $f : (X_t, t) \mapsto X_T$, which maps any intermediate state X_t directly to a sample X_T from the target distribution.

A straightforward method is to first appoximate a dataset by simulating the generative SDE in (2) and producing samples $\{\hat{X}_T^i\}_{i=1}^M$, and then applying existing consistency distillation or consistency

Algorithm 1 CDDS Training

Input: Model parameters θ, a pre-trained control u, learning rate η
$\theta' \leftarrow \theta$
repeat
  Sample $X_0 \sim p_{\text{prior}}$ and $n \sim \mathcal{U}\{1, N-1\}$
  Simulate PF ODE of (2) to get $\hat{X}_{t_n}$ and $\hat{X}_{t_{n+1}}$
  $\mathcal{L}(\theta, \theta') \leftarrow \|f_{\theta'}(\hat{X}_{t_{n+1}}, t_{n+1}), f_\theta(\hat{X}_{t_n}, t_n)\|_2$
  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta, \theta'; u)$
  $\theta' \leftarrow \text{stopgrad}(\theta)$
until convergence

Algorithm 2 SCDS Training

Input: Model parameters θ, weights $\lambda_{\text{S}}, \lambda_{\text{SC}}$
$\theta' \leftarrow \theta$
repeat
  Sample $X_0 \sim p_{\text{prior}}$, d, and t
  Simulate (2) to get $X_{0:T}$
  Compute target X_{t+2d} from (10)
  Compute shortcut $\hat{X}_{t+2d}$ from (9)
  Compute sampling loss $\mathcal{L}_{\text{S}}$ via (5)
  Compute consistency loss $\mathcal{L}_{\text{SC}}$ via (11)
  $\theta \leftarrow \nabla_\theta \left(\lambda_{\text{S}} \mathcal{L}_{\text{S}} + \lambda_{\text{SC}} \mathcal{L}_{\text{SC}}\right)$
  $\theta' \leftarrow \text{stopgrad}\, \theta$
until convergence

training methods [46] to learn the function f. However, this approach is expensive as it necessitates pre-collecting and storing a large dataset. Moreover, the accumulation of numerical errors arises from the numerical solver , resulting in significant global error.

To solve the problem, we propose to leverage intermediate states of the pretrained model during each training iteration. Using these multiple and short intervals among intermediate helps keep the overall global error small.

During model training, we minimize the discrepancy between the outputs of the consistency function at the consecutive intermediate states of the probability flow ODE [48] associated with (2):

$$\mathcal{L}_{\text{CD}}(\theta, \theta'; u)(X) := \mathbb{E}\Big[\|f_{\theta'}(\hat{X}_{t_{n+1}}, t_{n+1}), f_\theta(\hat{X}_{t_n}, t_n)\|_2\Big], \tag{7}$$

where the expectation is over discrete time indices n and $\theta' = \text{stopgrad}(\theta)$ indicates gradient stopping on the target term. Notably, unlike standard consistency generative models, the states $\hat{X}_{t_{n+1}}$ and $\hat{X}_{t_n}$ are obtained from partial integrations of the probability flow ODE rather than from real data samples. Consequently, training CDDS incurs computational costs similar to training traditional diffusion samplers while substantially accelerating inference. The training procedure is summarized in Algorithm 1.

If the loss (7) is driven to zero, the learned consistency function recovers the true mapping of the probability flow ODE, implying that CDDS can achieve arbitrarily accurate single-step sampling in the limit of sufficiently small integration steps. We formally state this in Theorem 1.

Theorem 1. *Let $f_\theta(X_t, t)$ be a consistency function parameterized by θ, and let $f(X_t, t; u)$ denote the consistency function of the PF ODE defined by the control u. Assume that f_θ is $L-$Lipschitz continuous. Additionally, assume that for each step $n \in \{1, 2, \ldots, N-1\}$, the ODE solver called at t_n has a local error bounded by $O((t_{n+1} - t_n)^{p+1})$ for some $p \geq 1$. If $\mathcal{L}_{CD}(\theta, \theta'; u) = 0$, then:*

$$\sup_{n, X_{t_n}} \|f_\theta(X_{t_n}, t_n) - f(X_{t_n}, t_n; u)\|_2 = O((\Delta t)^p), \tag{8}$$

where $\Delta t := \max_{n \in \{1,2,\ldots,N-1\}} |t_{n+1} - t_n|$.

A complete proof is provided in the Appendix. This theoretical result shows that consistency functions can be distilled from diffusion samplers when only an unnormalized density oracle is available, enabling principled single-step sampling without requiring access to data from the target distribution.

Remark. While our distillation approach builds upon the core principles of consistency generative models, it differs in setting and requirements. Instead of relying on having access to a dataset from p_{target}, our method extends consistency distillation to sampling from unnormalized distributions, making it applicable beyond generative modeling tasks.

5 Self–Consistent Diffusion Samplers

5.1 Overivew

Even though CDDS demonstrates that single-step sampling is feasible, it requires a pretrained diffusion sampler for distillation. In this section, we further introduce *self-consistent diffusion sampler* (SCDS) that achieves single-step sampling without requiring the pre-trained diffusion sampler. The key idea is to adapt consistency training into the diffusion-based sampler training. However, the challenge arises from merging two perspectives: diffusion-based samplers learn a time-dependent control function that steers an SDE from a simple prior distribution to the target distribution. Typically, the control is trained on a fixed schedule (e.g., N small increments of length T/N along a discretized time axis), requiring multiple steps. In contrast, consistency models learn a direct mapping from any intermediate state on an ODE to the terminal state. In other words, at time t the model is implicitly taught to jump a large step of length $T - t$. Crucially, consistency training in the original formulation [45, 46] assumes the availability of intermediate states generated by perturbing real data; in the context of sampling from unnormalized densities, however, we cannot generate these states directly because we lack data and thus must learn both sampling and self-distillation simultaneously. Furthermore, as discussed in Section 3, diffusion-based samplers typically parameterize the control function, whose purpose differs fundamentally from that of the consistency function used in standard consistency training.

To reconcile these perspectives and overcome this challenge, we propose conditioning the control function $u_\theta(X_t, t, d)$ on both the current time t and the desired step size d. By adjusting d, the model can adapt between short incremental steps (as in standard diffusion samplers) and large jumps (as in consistency models). This design amortizes the learning of both small and large transitions into one network and recovers consistency models' single-step sampling by setting $d = T - t$ and diffusion sampling by setting $d = T/N$. In doing so, we avoid training with two contrasting learning targets, hence making it feasible to train a single-step diffusion sampler from scratch.

5.2 Training

Learning the base case $\mathbf{d = T/N}$. In standard generative modeling scenarios (where a dataset is available), the base case $d = T/N$ can be learned directly from data using deterministic trajectories [25, 16]. These trajectories provide explicit guidance toward high-density regions of the target distribution.

However, when working with an unnormalized density, a key challenge is discovering high-probability regions [11]. Thus, the process (2) is particularly well-suited for learning the base case as the Brownian motion helps probe different parts of the space, allowing the model to learn and adapt itself to the target distribution. By optimizing $u_\theta(X_t, t, d = T/N)$ under (5), we ensure that the model can generate meaningful transitions from the prior to these regions of interest, forming a strong foundation for self-consistent learning at larger step sizes.

Enforcing self-consistency. To ensure that the step-size-conditioned control function $u_\theta(X_t, t, d)$ remains accurate across varying step sizes, we introduce a self-consistency loss. The key idea is that taking a large step should yield the same result as taking multiple smaller steps. To do so, we impose a consistency condition on the Euler discretization of the probability flow ODE associated with the forward process (5). Specifically, we require that a single large step of size $2d$,

$$\hat{X}_{t+2d} = X_t + 2d\left(\mu + \tfrac{1}{2}\sigma u_\theta\right)(X_t, t, 2d), \tag{9}$$

yields the same result as two smaller steps of size d. The intermediate state after the first small step is computed as

$$X_{t+d} = X_t + d\left(\mu + \tfrac{1}{2}\sigma u_{\theta'}\right)(X_t, t, d),$$

and the final state after the second small step is

$$X_{t+2d} = X_{t+d} + d\left(\mu + \tfrac{1}{2}\sigma u_{\theta'}\right)(X_{t+d}, t+d, d), \tag{10}$$

where $\theta' = \mathrm{stopgrad}(\theta)$. The self-consistency objective is a simple least square minimization problem:

$$\mathcal{L}_{\text{SC}} = \mathbb{E}\left[\|X_{t+2d} - \hat{X}_{t+2d}\|^2\right], \tag{11}$$

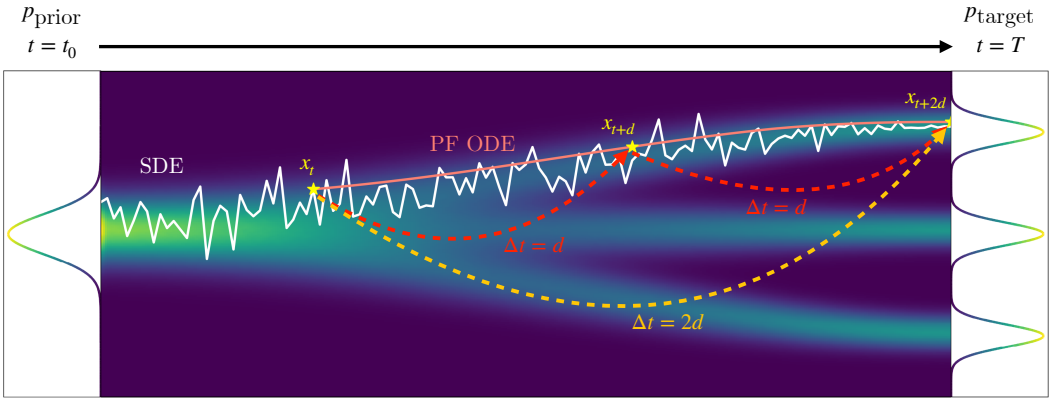

Figure 1: Graphical illustration of the training procedure for SCDS over the path space. First, the SDE (2) is simulated (white) to optimize (5). Next, a timestep t and a step size d are randomly sampled. From X_t on the simulated SDE trajectory, we execute two consecutive steps of size d (red) along the probability flow ODE trajectory (pink) of (2), obtaining the target X_{t+2d}. Finally, the large step of size $2d$ (orange) predicts $\hat{X}_{t+2d}$ directly from X_t, and the self-consistency loss (11) minimizes the squared difference between $\hat{X}_{t+2d}$ and the two-step target X_{t+2d}, ensuring multi-scale consistency.

where the expectation is taken over time indices and step sizes drawn from the simulated trajectories.

This loss encourages the model to correct for numerical errors when taking large steps, allowing it to "skip" multiple smaller steps while remaining consistent with the dynamics of the probability flow ODE.

End-to-end training algorithm. As abovementioned, our training procedure jointly optimizes two objectives: (i) a sampling loss (5) for the base case $d = T/N$, which ensures exploration and score approximation by simulating the SDE (2), and (ii) the self-consistency loss (11) enforced on the probability flow ODE of (2) for larger steps, which enforces consistency across multiple time scales. This end-to-end formulation enables the model to fully amortize the cost of sampling into a single forward pass at inference time.

To enable the recursive halving of steps, we discretize the time interval $[0, T]$ into $N + 1$ points, where N is chosen as a power of two. The sampling loss is computed by simulating the forward SDE along this time grid.

For self-consistency training, we sample step sizes d and times t such that d are powers of two (multiplied by T/N) dividing the remaining time $T - t$. This ensures that from any time t, we can take exactly k steps of size d to reach the terminal state for some integer k. This way, training focuses on time sequences that are applicable during inference.

To compute the self-consistency loss, we extract X_t from the simulated forward SDE. Using X_t and the sampled step size d, we compute the shortcut step $\hat{X}_{t+2d}$ using (9) and the two-step target trajectory X_{t+2d} using (10). We then optimize their squared difference via the objective (11), ensuring that larger steps remain consistent with fine-grained trajectories. The training procedure is summarized in Algorithm 2 and illustrated in Figure 1. Compared to standard diffusion-based samplers, which typically require hundreds of evaluations per iteration during training, our method introduces only 3 additional function evaluations. Thus, the overhead from SCDS training is marginal, typically amounting to less than a few percent of the total computational cost per iteration.

5.3 Inference.

Few-step sampling. With a well-trained control u_θ, sampling can be performed in a single step by drawing from the prior and applying a single Euler update with step size $d = T$, as shown in Algorithm 3. This accelerates generation compared to traditional diffusion-based samplers. Alternatively, our method provides a flexible tradeoff between computational efficiency and sample

Algorithm 3 Single-Step Sampling with SCDS

Input: Trained model u_θ
Sample $X_0 \sim p_{\text{prior}}$
$X_T \leftarrow X_0 + T\left(\mu + \frac{1}{2}\sigma u_\theta\right)(X_0, 0, T)$
Return X_T

Algorithm 4 Multi-Step Sampling with SCDS

Input: Trained model u_θ, number of sampling steps N
Sample $X_0 \sim p_{\text{prior}}$
Initialize $d \leftarrow T/N$ and $t \leftarrow 0$
for $k = 1, \ldots, N$ **do**
  $X_{t+d} \leftarrow X_t + d\left(\mu + \frac{1}{2}\sigma u_\theta\right)(X_t, t, d)$
  $t \leftarrow t + d$
end for
Return X_T

quality, allowing for multi-step refinement when needed, thus recovering standard diffusion-based sampling. This iterative procedure is detailed in Algorithm 4.

Approximating the normalization constant. Beyond sample generation, a common goal in probabilistic inference is to estimate the normalization constant Z of the target density. Because SCDS is formulated within the optimal control framework of stochastic sampling [7], it inherits a natural estimator of Z through the Radon–Nikodym derivative between forward and backward path measures. By discretizing the likelihood ratio in (6) along simulated trajectories, SCDS enables efficient estimation of $\log Z$. In contrast, consistency-based generative models [16, 46] are trained using fully supervised losses on labeled data. As a result, they lack a built-in mechanism to estimate Z or compute likelihood ratios.

6 Experiments

We empirically evaluate the sampling efficiency and effectiveness of the proposed CDDS and SCDS across diverse standard benchmarks in Bayesian inference and sampling [9].

Specifically, we consider Bayesian posterior inference tasks, such as Ionosphere (35-D) and the high-dimensional Log-Gaussian Cox Process (LGCP) (1600-D), alongside representative synthetic targets: a Gaussian Mixture Model (GMM) of nine components in 2-D, a 2-D Many-Well with 32 well-separated modes (MW54), the widely-studied Funnel distribution in ten dimensions, and a 50-D Many-Well distribution with 32 modes (MW52). Details are provided in Appendix

We report the Sinkhorn distance $\mathcal{W}_\gamma^2$, the Effective Sample Size (ESS), and the absolute estimation error of the log normalization constant $\Delta \log Z$ for tasks with accessible ground-truth samples. For the real-world Ionosphere and LGCP tasks, we report the evidence lower bound (ELBO). ELBO, ESS and $\Delta \log Z$ rely on importance weights. With $N > 1$ we use the discretized RND of (6); for $N = 1$ the running-cost term vanishes and the weight reduces to the boundary likelihood, which can potentially inflate ELBO scores. Hence we regard $\mathcal{W}_\gamma^2$ and $\Delta \log Z$ as primary quality indicators, and report ELBO only when ground truth samples are not available.

We benchmark our methods against established diffusion-based samplers: the Path Integral Sampler (PIS) [55], the Denoising Diffusion Sampler (DDS) [50], both trained with the KL divergence, and the Time-Reversed Diffusion Sampler (DIS) [7] trained with log-variance divergence.

Throughout our experiments, we use a pre-trained DIS as the teacher model for CDDS. Furthermore, since the optimization problem (4) may admit infinitely many solutions, we fix the noising process (3) in SCDS to ensure uniqueness of the learned solution, following the same approach as in DIS. Detailed implementation and hyperparameters are provided in Appendix A.1. The code is available at this repository.

6.1 Results

Single-step sampling. Rows shaded in in Tables 1-2 compare one–step CDDS and SCDS with fully-discretized (128–256 step) baselines. Several consistent patterns emerge. On the low–dimensional GMM and MW54 targets, CDDS attains Sinkhorn values with a factor less then double the DIS baseline. ESS is also two or three orders of magnitude larger than DIS with a single step, illustrating

that distillation corrects most of the degeneracy of naive single-step DIS. The ELBO of CDDS on Ionosphere and LGCP even exceeds that of every 256-step baseline; we attribute this to the discretized estimator of the Radon–Nikodym derivative (RND) (6), which for single-step sampling collapses to the boundary likelihood and can therefore *over-estimate* evidence.

Without access to a teacher, SCDS learns its own control and offers a trade-off: In one step SCDS delivers usable samples (e.g. GMM, Funnel) but is generally less accurate than CDDS because it has to discover the score field from scratch.

Table 1: Synthetic-benchmark results. Each task block reports the Sinkhorn distance ($\mathcal{W}_\gamma^2 \downarrow$), the effective sample size (ESS $\uparrow$), and absolute log-normalization error ($|\Delta \log Z| \downarrow$).

	PIS	DDS	DIS	DIS	CDDS (ours)	SCDS (ours)
NFE $\downarrow$	128	128	128	1	1	1
			GMM (2D)			
$\mathcal{W}_\gamma^2$	1.7946	0.0898	0.0203	0.0559	0.0313	0.0478
ESS	1×10^{-5}	0.0065	0.8054	0.00057	0.3395	0.0504
$\lvert\Delta \log Z\rvert$	2.1806	1.6819	0.0899	14.7669	1.5717	1.0743
			MW54 (5D)			
$\mathcal{W}_\gamma^2$	0.1377	0.1366	0.1230	6.2804	0.2815	0.3913
ESS	0.0664	0.0051	0.2677	1.7×10^{-5}	0.0442	2.1×10^{-5}
$\lvert\Delta \log Z\rvert$	1.9974	2.4154	1.2056	6766.2891	1.0966	11.2524
			Funnel (10D)			
$\mathcal{W}_\gamma^2$	6.0731	5.8600	5.1755	10.5224	8.8849	5.4922
ESS	1×10^{-5}	0.0582	0.1305	4.9×10^{-5}	5.1×10^{-5}	0.00014
$\lvert\Delta \log Z\rvert$	0.4381	0.5641	0.6407	∞	1.3316	10.3557
			MW52 (50D)			
$\mathcal{W}_\gamma^2$	6.8035	6.7830	6.8808	31.2532	6.1764	7.1110
ESS	1×10^{-5}	0.4412	0.0028	1×10^{-5}	5.7×10^{-5}	1×10^{-5}
$\lvert\Delta \log Z\rvert$	42.4502	42.4245	39.7814	9116.0713	63.7244	87.7095

Table 2: ELBO ($\uparrow$) on two real-world benchmarks. Shaded columns denote single–step inference.

	PIS	DDS	DIS	SCDS	DIS	CDDS (ours)	SCDS (ours)
NFE $\downarrow$	256	256	256	256	1	1	1
Ionosphere (35D)	-39.3	-1510.3	-77.4	-73.7	-3252.7	-27.5	-567.3
LGCP (1600D)	397.5	314.8	365.6	103.9	-3.09×10^{6}	1118.0	-4579.7

Cost-benefit analysis of single-step inference. Figure 2 reports the total number (training plus inference) of network-function evaluations (NFEs) for SCDS versus PIS, DDS, and DIS, that use the same N-step discretization during training. SCDS adds only three NFEs per training iteration but replaces the entire N-step Euler integration with a single forward pass at test time. With batch size B and I training iterations, the extra training cost is $3IB$ NFEs, while every sample produced at inference saves $N-1$ NFEs; hence the break-even point is $S_{\text{break}} = 3IB/(N-1)$. For the 2-D GMM experiment ($N = 128$, $B = 512$, $I = 10{,}000$) this yields $S_{\text{break}} \approx 1.2 \times 10^4$. We generated five million test samples to obtain Table 1, thereby saving about 620 million NFEs relative to the baselines. In the 1600-D LGCP task the model is trained longer but with smaller batches ($N = 256$, $B = 64$, $I = 50{,}000$), giving $S_{\text{break}} \approx 3.7 \times 10^4$, and generating one million samples saves over half a billion NFEs. Training cost is thus a fixed, modest overhead that is quickly amortised in realistic simulations.

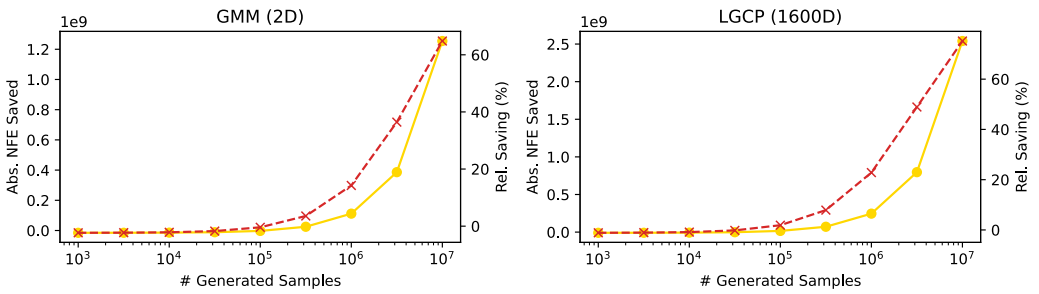

Figure 2: Absolute (yellow) and relative (red, dashed) network-function evaluations (NFE) savings of SCDS over the baseline diffusion samplers PIS, DDS, and DIS as a function of the number of generated samples (log scale). Left: 2-D GMM training regime; Right: 1600-D LGCP regime.

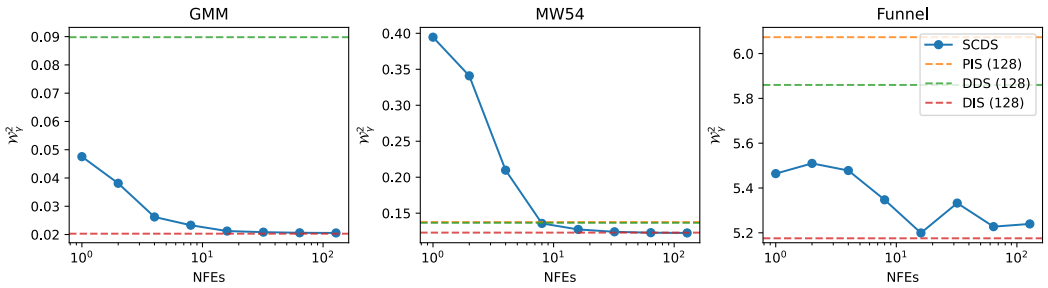

Figure 3: Sinkhorn distance as a function of the number of diffusion steps for SCDS, illustrating the flexible trade-off between computational cost and sample quality. Horizontal dashed lines represent baseline diffusion samplers with 128 steps.

Practically, we recommend using CDDS when a reliable pretrained sampler is available and the goal is single-step generation, as CDDS training is cheaper than SCDS yet could achieve comparable single-step quality. If no pretrained sampler exists, SCDS provides a more economical and flexible solution, as it learns directly from the unnormalized target and allows adjusting the number of inference steps to trade computational resources for improved sample quality.

Trade compute for sample quality. Figure 3 shows how the Sinkhorn distance decays as we allocate more network-function evaluations (NFEs) to SCDS at inference time. On the 2-D GMM and the 5-D Many-Well tasks the curve is strictly monotone: with only 4–8 Euler updates SCDS already matches the accuracy of DDS, and at 32–64 steps it recovers the DIS reference obtained with 128 steps. The same trend is visible on the 10-D Funnel, but with a mild "bump" at 32 steps. SCDS allows practitioners to flexibly adjust the computational budget, progressively improving sample quality until it matches the accuracy of traditional multi-step diffusion samplers.

Additional results, comparisons with other baselines, and ablations are provided in Appendix A.3.

7 Conclusion

We introduced two novel approaches for efficient sampling from unnormalized target distributions: *consistency-distilled diffusion samplers* (CDDS) and the *self-consistent diffusion sampler* (SCDS). CDDS uses consistency distillation without generating a large dataset of samples. SCDS requires no pre-trained samplers and simultaneously learns to sample high-density regions and to take large steps across the path space. Our empirical results across a range of benchmarks demonstrated that both methods achieve competitive accuracy with as few as one or two steps. These findings highlighted the potential of consistency-based methods for sampling from unnormalized densities.

References

[1] M. S. Albergo, G. Kanwar, and P. E. Shanahan. Flow-based Generative Models for Markov chain Monte Carlo in Lattice Field Theory. *Physical Review D*, 100(3):034515, 2019.

[2] B. D. O. Anderson. Reverse-time Diffusion Equation Models. *Stochastic Processes and their Applications*, 12:313–326, 1982.

[3] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1-2):5–43, 2003.

[4] E. Bengio, M. Jain, M. Korablyov, D. Precup, and Y. Bengio. Flow Network based Generative Models for Non-Iterative Diverse Candidate Generation. In *Advances in Neural Information Processing Systems*, 2021.

[5] Y. Bengio, S. Lahlou, T. Deleu, E. J. Hu, M. Tiwari, and E. Bengio. GFlowNet Foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

[6] J. Berner, L. Richter, M. Sendera, J. Rector-Brooks, and N. Malkin. From Discrete-Time Policies to Continuous-Time Diffusion Samplers: Asymptotic Equivalences and Faster Training. *arXiv preprint arXiv:2501.06148*, 2025.

[7] J. Berner, L. Richter, and K. Ullrich. An Optimal Control Perspective on Diffusion-based Generative Modeling. *Transactions on Machine Learning Research*, 2024.

[8] D. Blessing, J. Berner, L. Richter, and G. Neumann. Underdamped Diffusion Bridges with Applications to Sampling. In *International Conference on Learning Representations*, 2025.

[9] D. Blessing, X. Jia, J. Esslinger, F. Vargas, and G. Neumann. Beyond ELBOs: A Large-Scale Evaluation of Variational Methods for Sampling. In *International Conference on Machine Learning*, 2024.

[10] S. P. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng. Handbook of Markov Chain Monte Carlo: Hardcover. *CHANCE*, 25:53–55, 2012.

[11] J. Chen, L. Richter, J. Berner, D. Blessing, G. Neumann, and A. Anandkumar. Sequential Controlled Langevin Diffusions. In *International Conference on Learning Representations*, 2025.

[12] T. Chen, E. Fox, and C. Guestrin. Stochastic Gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning*, 2014.

[13] A. Doucet, W. Grathwohl, A. G. Matthews, and H. Strathmann. Score-based Diffusion Meets Annealed Importance Sampling. In *Advances in Neural Information Processing Systems*, 2022.

[14] S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

[15] H. Föllmer. Time Reversal on Wiener Space. In *Stochastic Processes—Mathematics and Physics*, pages 119–129. Springer, 1986.

[16] K. Frans, D. Hafner, S. Levine, and P. Abbeel. One Step Diffusion via Shortcut Models. In *International Conference on Learning Representations*. ICLR, 2025.

[17] D. Frenkel and B. Smit. *Understanding Molecular Simulation: From Algorithms to Applications*. Academic Press, Amsterdam, The Netherlands, 2002.

[18] T. Geffner and J. Domke. Langevin Diffusion Variational Inference. In *International Conference on Artificial Intelligence and Statistics*, 2023.

[19] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984.

[20] W. K. Hastings. Monte Carlo Sampling Methods using Markov Chains and their Applications. *Biometrika*, 57(1):97–109, 1970.

[21] J. M. Hernández-Lobato and R. Adams. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In *International Conference on Machine Learning*, 2015.

[22] J. Ho, A. Jain, and P. Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, 2020.

[23] L. Holdijk, Y. Du, F. Hooft, P. Jaini, B. Ensing, and M. Welling. Stochastic Optimal Control for Collective Variable Free Sampling of Molecular Transition Paths. *Advances in Neural Information Processing Systems*, 2024.

[24] S. A. Hollingsworth and R. O. Dror. Molecular Dynamics Simulation for All. *Neuron*, 99(6):1129–1143, 2018.

[25] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations*, 2023.

[26] C. Lu and Y. Song. Simplifying, Stabilizing and Scaling Continuous-time Consistency Models. In *International Conference on Learning Representations*, 2025.

[27] D. J. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[28] C. Meng, R. Rombach, R. Gao, D. Kingma, S. Ermon, J. Ho, and T. Salimans. On Distillation of Guided Diffusion Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[29] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[30] R. M. Neal. Bayesian Learning for Neural Networks. 1995.

[31] R. M. Neal. Annealed Importance Sampling. *Statistics and Computing*, 11:125–139, 2001.

[32] E. Nelson. *Dynamical Theories of Brownian Motion*. Princeton University Press, Princeton, NJ, 1967.

[33] F. Noé, J. Köhler, and H. Wu. Boltzmann Generators – Sampling Equilibrium States of Many-Body Systems with Deep Learning. *Science*, 365, 2019.

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

[35] G. Parisi. Correlation Functions and Computer Simulations. *Nuclear Physics B*, 180(3):378–384, 1981.

[36] A. Phillips, H.-D. Dau, M. J. Hutchinson, V. D. Bortoli, G. Deligiannidis, and A. Doucet. Particle Denoising Diffusion Sampler. In *International Conference on Machine Learning*, 2024.

[37] L. Richter. *Solving high-dimensional PDEs, approximation of path space measures and importance sampling of diffusions*. Doctoral dissertation, BTU Cottbus-Senftenberg, 2021.

[38] L. Richter and J. Berner. Improved Sampling via Learned Diffusions. In *International Conference on Learning Representations*, 2024.

[39] C. P. Robert. Convergence Control Methods for Markov Chain Monte Carlo Algorithms. *Statistical Science*, 10(3):231–253, 1995.

[40] P. J. Rossky, J. D. Doll, and H. L. Friedman. Brownian Dynamics as Smart Monte Carlo Simulation. *The Journal of Chemical Physics*, 69(10):4628–4633, 11 1978.

[41] T. Salimans and J. Ho. Progressive Distillation for Fast Sampling of Diffusion Models. In *International Conference on Learning Representations*, 2022.

[42] E. Schrödinger. Über die Umkehrung der Naturgesetze. *Sitzungsberichte der Preussischen Akademie der Wissenschaften Berlin, Physikalisch-Mathematische Klasse*, pages 144–153, 1931.

[43] E. Schrödinger. Sur la Théorie Relativiste de l'Électron et l'Interprétation de la Mécanique Quantique. *Annales de l'Institut Henri Poincaré*, 2:269–310, 1932.

[44] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep Unsupervised Learning Using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning*, 2015.

[45] Y. Song and P. Dhariwal. Improved Techniques for Training Consistency Models. In *International Conference on Learning Representations*, 2024.

[46] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. Consistency Models. In *International Conference on Machine Learning*, 2023.

[47] Y. Song and S. Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. *Advances in Neural Information Processing Systems*, 2019.

[48] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*, 2021.

[49] J. Sun, J. Berner, L. Richter, M. Zeinhofer, J. Müller, K. Azizzadenesheli, and A. Anandkumar. Dynamical Measure Transport and Neural PDE Solvers for Sampling. *arXiv preprint arXiv:2407.07873*, 2024.

[50] F. Vargas, W. S. Grathwohl, and A. Doucet. Denoising Diffusion Samplers. In *International Conference on Learning Representations*, 2023.

[51] F. Vargas, S. Padhy, D. Blessing, and N. Nüsken. Transport Meets Variational Inference: Controlled Monte Carlo Diffusions. In *International Conference on Learning Representations*, 2024.

[52] M. Welling and Y. W. Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *International Conference on Machine Learning*, 2011.

[53] D. Wu, L. Wang, and P. Zhang. Solving Statistical Mechanics using Variational Autoregressive Networks. *Physical Review Letters*, 122(8):080602, 2019.

[54] D. Zhang, R. T. Q. Chen, C.-H. Liu, A. Courville, and Y. Bengio. Diffusion Generative Flow Samplers: Improving Learning Signals through Partial Trajectory Optimization. In *International Conference on Learning Representations*, 2024.

[55] Q. Zhang and Y. Chen. Path Integral Sampler: A Stochastic Control Approach For Sampling. In *International Conference on Learning Representations*, 2022.

[56] R. Zhang, A. F. Cooper, and C. De Sa. AMAGOLD: Amortized Metropolis Adjustment for Efficient Stochastic Gradient MCMC. In *International Conference on Artificial Intelligence and Statistics*, 2020.

[57] R. Zhang, C. Li, J. Zhang, C. Chen, and A. G. Wilson. Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning. In *International Conference on Learning Representations*, 2020.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: Contributions are detailed in Sec. 4 and 5, supported by experiment results in Sec. 6.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

Justification: Sec. 4 explicitly discusses the dependency of CDDS on a pre-trained control, and Sec. 5 clearly acknowledges that our approach introduces a modest increase in training time relative to baseline methods. We also included a Limitation section in Appendix A.5.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: See Appendix A.4 for proofs.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: The paper clearly specifies model architectures, training procedures, hyperparameters, and evaluation metrics required to reproduce the main experimental results (Sections 4, 5, and Appendix A.1, A.2). The code is available at this repository.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: The code is available at this repository.

   Guidelines:

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

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Experiments and hyper-parameters are detailed in Appendix A.2.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We conducted multiple independent simulation runs for all experiments and found standard deviations consistently negligible relative to the reported means. To maintain readability, these small standard deviations are omitted from the main tables but are reported in Appendix A.3.

   Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

Justification: We specify the neural network architecture and explicitly report the number of network function evaluations for each task, a standard measure to estimate computational cost. Additional details regarding the compute infrastructure and hardware used for experiments are provided in Appendix A.2.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

Answer: [Yes]

Justification: This paper focuses on accelerating diffusion samplers without any practices that directly violate NeurIPS Code of Ethics.

Guidelines:

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

10. **Broader impacts**

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

Answer: [Yes]

Justification: See Appendix A.6.

Guidelines:

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

11. **Safeguards**

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

Answer: [NA]

Justification: No new dataset or pretrained model is released.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: This paper reuses some datasets and codes from previous work, and they have been cited in the paper. The license and terms of use are also properly respected.

Guidelines:

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

13. **New assets**

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

Answer: [NA]

Justification: This paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.