SCORE-BASED GENERATIVE MODELING WITH CRITICALLY-DAMPED LANGEVIN DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Score-based generative models (SGMs) have demonstrated remarkable synthesis quality. SGMs rely on a diffusion process that gradually perturbs the data towards a tractable distribution, while the generative model learns to denoise. The complexity of this denoising task is, apart from the data distribution itself, uniquely determined by the diffusion process. We argue that current SGMs employ overly simplistic diffusions, leading to unnecessarily complex denoising processes, which limit generative modeling performance. Based on connections to statistical mechanics, we propose a novel *critically-damped Langevin diffusion* (CLD) and show that CLD-based SGMs achieve superior performance. CLD can be interpreted as running a joint diffusion in an extended space, where the auxiliary variables can be considered "velocities" that are coupled to the data variables as in Hamiltonian dynamics. We derive a novel score matching objective for CLD and show that the model only needs to learn the score function of the conditional distribution of the velocity given data, an easier task than learning scores of the data directly. We also derive a new sampling scheme for efficient synthesis from CLD-based diffusion models. We find that CLD outperforms previous SGMs in synthesis quality for similar network architectures and sampling compute budgets. We show that our novel sampler for CLD significantly outperforms solvers such as Euler–Maruyama. Our framework provides new insights into score-based denoising diffusion models and can be readily used for high-resolution image synthesis.

1 INTRODUCTION

Score-based generative models (SGMs) and denoising diffusion probabilistic models have emerged as a promising class of generative models (Sohl-Dickstein et al., 2015; Song et al., 2021c;b; Vahdat et al., 2021; Kingma et al., 2021). SGMs offer high quality synthesis and sample diversity, do not require adversarial objectives, and have found applications in image (Ho et al., 2020; Nichol & Dhariwal, 2021; Dhariwal & Nichol, 2021; Ho et al., 2021), speech (Chen et al., 2021; Kong et al., 2021; Jeong et al., 2021), and music synthesis (Mittal et al., 2021), image editing (Meng et al., 2021; Sinha et al., 2021; Furusawa et al., 2021), super-resolution (Saharia et al., 2021; Li et al., 2021), image-to-image translation (Sasaki et al., 2021), and 3D shape generation (Luo & Hu, 2021; Zhou et al., 2021). SGMs use a diffusion process to gradually add noise to the data, transforming a complex data distribution to an analytically tractable prior distribution. A neural network is then utilized to learn the score function—the gradient of the log probability density—of the perturbed data. The learnt scores can be used to solve a stochastic differential equation (SDE) to synthesize new samples. This corresponds to an iterative denoising process, inverting the forward diffusion.

In the seminal work by Song et al. (2021c), it has been shown that the score function that needs to be learnt by the neural network is uniquely determined by the forward diffusion process. Consequently, the complexity of the learning problem depends, other than on the data itself, only on the diffusion. Hence, the diffusion process is the key component of SGMs that needs to be revisited to further improve SGMs, for example, in terms of synthesis quality or sampling speed.

Inspired by statistical mechanics (Tuckerman, 2010), we propose a novel forward diffusion process, the *critically-damped Langevin diffusion (CLD)*. In CLD, the data variable, $\mathbf{x}_t$ (time t along the diffusion), is augmented with an additional "velocity" variable $\mathbf{v}_t$ and a diffusion process is run in the joint data-velocity space. Data and velocity are coupled to each other as in Hamiltonian dynamics, and noise is injected only into the velocity variable. Similarly as in Hamiltonian Monte

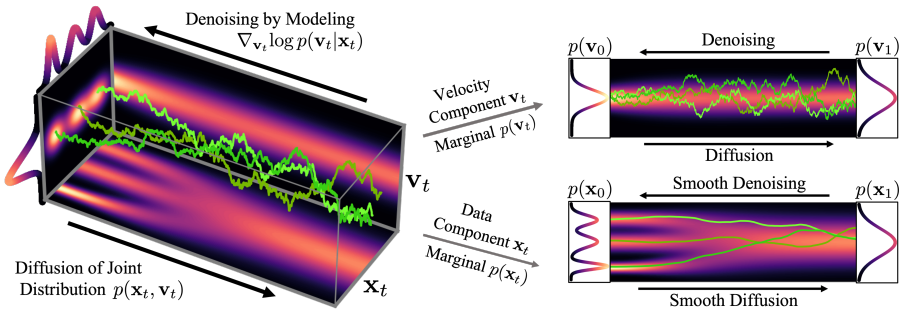


Figure 1: In critically-damped Langevin diffusion, the data $\mathbf{x}_t$ is augmented with a velocity $\mathbf{v}_t$. A diffusion coupling $\mathbf{x}_t$ and $\mathbf{v}_t$ is run in the joint data-velocity space (probabilities in **red**). Noise is injected only into $\mathbf{v}_t$. This leads to smooth diffusion trajectories (**green**) for the data $\mathbf{x}_t$. Denoising only requires $\nabla_{\mathbf{v}_t} \log p(\mathbf{v}_t|\mathbf{x}_t)$.

Carlo (Neal, 2011), the Hamiltonian component helps to efficiently traverse the joint data-velocity space and to transform the data distribution into the prior distribution more smoothly. We derive the corresponding score matching objective and show that for CLD the neural network is tasked with learning only the score of the conditional distribution of velocity given data $\nabla_{\mathbf{v}_t} \log p_t(\mathbf{v}_t|\mathbf{x}_t)$, which is arguably easier than learning the score of diffused data distribution directly. Using techniques from molecular dynamics (Bussi & Parrinello, 2007; Tuckerman, 2010; Leimkuhler & Matthews, 2013), we also derive a novel SDE integrator tailored to CLD's reverse-time synthesis SDE.

We extensively validate CLD and the new SDE solver: **(i)** We show that the neural networks learnt in CLD-based SGMs are smoother than those of previous SGMs. **(ii)** On the CIFAR-10 image modeling benchmark, we demonstrate that CLD-based models outperform previous diffusion models in synthesis quality for similar network architectures and sampling compute budgets. We attribute these positive results to the Hamiltonian component in the diffusion and to CLD's easier score function target, the velocity-data conditional distribution. **(iii)** We show that our novel sampling scheme for CLD significantly outperforms the popular Euler–Maruyama method. **(iv)** We perform ablations on various aspects of CLD and find that CLD does not have difficult-to-tune hyperparameters.

In summary, we make the following technical contributions: **(i)** We propose CLD, a novel diffusion process for SGMs. **(ii)** We derive a score matching objective for CLD, which requires only the conditional distribution of velocity given data. **(iii)** We propose a new type of denoising score matching ideally suited for scalable training of CLD-based SGMs. **(iv)** We derive a tailored SDE integrator that enables efficient sampling from CLD-based models. **(v)** Overall, we provide novel insights into SGMs and point out important new connections to statistical mechanics.

2 Background

Consider a diffusion process $\mathbf{u}_t \in \mathbb{R}^d$ defined by the Itô SDE

$$d\mathbf{u}_t = \boldsymbol{f}(\mathbf{u}_t, t)\, dt + \boldsymbol{G}(\mathbf{u}_t, t)\, d\mathbf{w}_t, \quad t \in [0, T], \tag{1}$$

with continuous time variable $t \in [0, T]$, standard Wiener process $\mathbf{w}_t$, drift coefficient $\boldsymbol{f}\colon \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ and diffusion coefficient $\boldsymbol{G}\colon \mathbb{R}^d \times [0, T] \to \mathbb{R}^{d\times d}$. Defining $\bar{\mathbf{u}}_t := \mathbf{u}_{T-t}$, a corresponding reverse-time diffusion process that inverts the above forward diffusion can be derived (Anderson, 1982; Haussmann & Pardoux, 1986; Song et al., 2021c) (with positive dt and $t \in [0, T]$):

$$d\bar{\mathbf{u}}_t = \Big[-\boldsymbol{f}(\bar{\mathbf{u}}_t, T-t) + \boldsymbol{G}(\bar{\mathbf{u}}_t, T-t)\boldsymbol{G}(\bar{\mathbf{u}}_t, T-t)^\top \nabla_{\bar{\mathbf{u}}_t} \log p_{T-t}(\bar{\mathbf{u}}_t)\Big]\, dt + \boldsymbol{G}(\bar{\mathbf{u}}_t, T-t) d\mathbf{w}_t, \tag{2}$$

where $\nabla_{\bar{\mathbf{u}}_t} \log p_{T-t}(\bar{\mathbf{u}}_t)$ is the score function of the marginal distribution over $\bar{\mathbf{u}}_t$ at time $T - t$.

The reverse-time process can be used as a generative model. In particular, Song et al. (2021c) model data $\mathbf{x}$, setting $p(\mathbf{u}_0){=}p_{\text{data}}(\mathbf{x})$. Currently used SDEs (Song et al., 2021c; Kim et al., 2021) have drift and diffusion coefficients of the simple form $\boldsymbol{f}(\mathbf{x}_t, t){=}f(t)\mathbf{x}_t$ and $\boldsymbol{G}(\mathbf{x}_t, t){=}g(t)\boldsymbol{I}_d$. Generally, $\boldsymbol{f}$ and $\boldsymbol{G}$ are chosen such that the SDE's marginal, equilibrium density is approximately Normal at time T, i.e., $p(\mathbf{u}_T){\approx}\mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$. We can then initialize $\mathbf{x}_0$ based on a sample drawn from a complex data distribution, corresponding to a far-from-equilibrium state. While we let the state $\mathbf{x}_0$ relax towards equilibrium via the forward diffusion, we can learn a model $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ for the score $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$, which can be used for synthesis via the reverse-time SDE in Eq. (2). If $\boldsymbol{f}$ and $\boldsymbol{G}$ take the simple form from above, the denoising score matching (Vincent, 2011) objective for this task is:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim[0,T]} \mathbb{E}_{\mathbf{x}_0\sim p(\mathbf{x}_0)} \mathbb{E}_{\mathbf{x}_t\sim p_t(\mathbf{x}_t|\mathbf{x}_0)} \left[\lambda(t) \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_0)\|_2^2\right] \tag{3}$$

If $\boldsymbol{f}$ and $\boldsymbol{G}$ are affine, the conditional distribution $p_t(\mathbf{x}_t|\mathbf{x}_0)$ is Normal and available analytically (Särkkä & Solin, 2019). Different $\lambda(t)$ result in different trade-offs between synthesis quality and likelihood in the generative model defined by $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ (Song et al., 2021b; Vahdat et al., 2021).

3 Critically-Damped Langevin Diffusion

We propose to augment the data $\mathbf{x}_t \in \mathbb{R}^d$ with auxiliary *velocity*[1] variables $\mathbf{v}_t \in \mathbb{R}^d$ and utilize a diffusion process that is run in the joint $\mathbf{x}_t$-$\mathbf{v}_t$-space. With $\mathbf{u}_t = (\mathbf{x}_t, \mathbf{v}_t)^\top \in \mathbb{R}^{2d}$, we set

$$\boldsymbol{f}(\mathbf{u}_t, t) \coloneqq \begin{pmatrix} 0 & \beta M^{-1} \\ -\beta & -\Gamma\beta M^{-1} \end{pmatrix} \otimes \boldsymbol{I}_d \, \mathbf{u}_t \,, \qquad \boldsymbol{G}(\mathbf{u}_t, t) \coloneqq \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{2\Gamma\beta} \end{pmatrix} \otimes \boldsymbol{I}_d, \tag{4}$$

where $\otimes$ denotes the Kronecker product. The coupled SDE that describes the diffusion process is

$$\begin{pmatrix} d\mathbf{x}_t \\ d\mathbf{v}_t \end{pmatrix} = \underbrace{\begin{pmatrix} M^{-1}\mathbf{v}_t \\ -\mathbf{x}_t \end{pmatrix} \beta dt}_{\text{Hamiltonian component}=:H} + \underbrace{\begin{pmatrix} 0 \\ -\Gamma M^{-1}\mathbf{v}_t \end{pmatrix} \beta dt + \begin{pmatrix} 0 \\ \sqrt{2\Gamma\beta} \end{pmatrix} d\mathbf{w}_t}_{\text{Ornstein-Uhlenbeck process}=:O}, \tag{5}$$

which corresponds to *Langevin dynamics* in each dimension. That is, each x_i is independently coupled to a velocity v_i, which explains the blockwise structure of $\boldsymbol{f}$ and $\boldsymbol{G}$. The *mass* $M \in \mathbb{R}^+$ is a hyperparameter that determines the coupling between the $\mathbf{x}_t$ and $\mathbf{v}_t$ variables; $\beta \in \mathbb{R}^+$ is a constant time rescaling chosen such that the diffusion converges to its equilibrium distribution within $t \in [0, T]$ (in practice, we set T=1) when initialized from a data-defined non-equilibrium state and is analogous to $\beta(t)$ in previous diffusions (we could also use time-dependent $\beta(t)$, but found constant β's to work well, and therefore opted for simplicity); $\Gamma \in \mathbb{R}^+$ is a *friction* coefficient that determines the strength of the noise injection into the velocities. Notice that the SDE in Eq. (5) consists of two components. The H term represents a Hamiltonian component. Hamiltonian dynamics are frequently used in Markov chain Monte Carlo methods to accelerate sampling and efficiently explore complex probability distributions (Neal, 2011). The Hamiltonian component in our diffusion process plays a similar role and helps to quickly and smoothly converge the initial joint data-velocity distribution to the equilibrium, or prior (see Fig. 1). Furthermore, Hamiltonian dynamics on their own are trivially invertible (Tuckerman, 2010), which intuitively is also beneficial in our situation when using this diffusion for training SGMs. The O term corresponds to an Ornstein-Uhlenbeck process (Särkkä & Solin, 2019) in the velocity component, which injects noise such that the diffusion dynamics properly converge to equilibrium for any Γ>0. It can be shown that the equilibrium distribution of this diffusion is $p_{\text{EQ}}(\mathbf{u}) = \mathcal{N}(\mathbf{x}; \mathbf{0}_d, \boldsymbol{I}_d)\,\mathcal{N}(\mathbf{v}; \mathbf{0}_d, M\boldsymbol{I}_d)$ (see App. B.2).

There is a crucial balance between M and Γ (McCall, 2010): For Γ^2<$4M$ (*underdamped* Langevin dynamics) the Hamiltonian component dominates, which implies oscillatory dynamics of $\mathbf{x}_t$ and $\mathbf{v}_t$ that slow down convergence to equilibrium. For Γ^2>$4M$ (*overdamped* Langevin dynamics) the O-term dominates which also slows down convergence, since the accelerating effect by the Hamiltonian component is suppressed due to the strong noise injection. For Γ^2=$4M$ (*critical damping*), an ideal balance is achieved and convergence to $p_{\text{EQ}}(\mathbf{u})$ occurs as fast as possible in a smooth manner without oscillations (also see discussion in App. A.1) (McCall, 2010). Hence, we propose to set Γ^2=$4M$ and call the resulting diffusion *critically-damped Langevin diffusion (CLD)* (see Fig. 1).

Diffusions such as the VPSDE (Song et al., 2021c) correspond to overdamped Langevin dynamics with high friction coefficients Γ (see App. A.2). Furthermore, in previous works noise is injected directly into the data variables (pixels, for images). In CLD, only the velocity variables are subject to direct noise and the data is perturbed only indirectly due to the coupling between $\mathbf{x}_t$ and $\mathbf{v}_t$.

3.1 Score Matching Objective

Considering the appealing convergence properties of CLD, we propose to utilize CLD as forward diffusion process in SGMs. To this end, we initialize the joint $p(\mathbf{u}_0)$=$p(\mathbf{x}_0)\,p(\mathbf{v}_0)$=$p_{\text{data}}(\mathbf{x}_0)\mathcal{N}(\mathbf{v}_0; \mathbf{0}_d, \gamma M\boldsymbol{I}_d)$ with hyperparameter γ<1 and let the distribution diffuse towards the tractable equilibrium—or prior—distribution $p_{\text{EQ}}(\mathbf{u})$. We can then

[1] We call the auxiliary variables *velocities*, as they play a similar role as velocities in physical systems. Formally, our velocity variables would rather correspond to physical momenta, but the term momentum is already widely used in machine learning and our mass M is unitless anyway.

learn the corresponding score functions and define CLD-based SGMs. Following a similar derivation as Song et al. (2021b), we obtain the score matching (SM) objective (see App. B.3):

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\in[0,T]}\mathbb{E}_{\mathbf{u}_t\sim p_t(\mathbf{u}_t)}\left[\lambda(t)\|s_{\boldsymbol{\theta}}(\mathbf{u}_t,t)-\nabla_{\mathbf{v}_t}\log p_t(\mathbf{u}_t)\|_2^2\right] \tag{6}$$

Notice that this objective requires only the velocity gradient of the log-density of the joint distribution, i.e., $\nabla_{\mathbf{v}_t}\log p_t(\mathbf{u}_t)$. This is a direct consequence of injecting noise into the velocity variables *only*. Without loss of generality, $p_t(\mathbf{u}_t)=p_t(\mathbf{x}_t,\mathbf{v}_t)=p_t(\mathbf{v}_t|\mathbf{x}_t)p_t(\mathbf{x}_t)$. Hence,

$$\nabla_{\mathbf{v}_t}\log p_t(\mathbf{u}_t)=\nabla_{\mathbf{v}_t}\left[\log p_t(\mathbf{v}_t|\mathbf{x}_t)+\log p_t(\mathbf{x}_t)\right]=\nabla_{\mathbf{v}_t}\log p_t(\mathbf{v}_t|\mathbf{x}_t) \tag{7}$$

This means that in CLD the neural network-defined score model $s_{\boldsymbol{\theta}}(\mathbf{u}_t,t)$ only needs to learn the score of the conditional distribution $p_t(\mathbf{v}_t|\mathbf{x}_t)$, an arguably easier task than learning the score of $p_t(\mathbf{x}_t)$, as in previous works, or of the joint $p_t(\mathbf{u}_t)$. This is the case, because our velocity distribution is initialized from a simple Normal distribution, such that $p_t(\mathbf{v}_t|\mathbf{x}_t)$ is closer to a Normal distribution for all $t\geq 0$ (and for any $\mathbf{x}_t$) than $p_t(\mathbf{x}_t)$ itself. This is most evident at $t=0$: The data and velocity distributions are independent at $t=0$ and the score of $p_0(\mathbf{v}_0|\mathbf{x}_0)=p_0(\mathbf{v}_0)$ simply corresponds to the score of the Normal distribution $p_0(\mathbf{v}_0)$ from which the velocities are initialized, whereas the score of the data distribution $p_0(\mathbf{x}_0)$ is highly complex and can even be unbounded (Kim et al., 2021). We empirically verify the reduced complexity of the score of $p_t(\mathbf{v}_t|\mathbf{x}_t)$ in Fig. 2. We find that the score that needs to be learnt by the model is more similar to a score corresponding to a Normal distribution for CLD than for the VPSDE. We also measure the complexity of the neural networks that were learnt to model this score via the squared Frobenius norm of their Jacobians. We find that the CLD-based SGMs have significantly simpler and smoother neural networks than VPSDE-based SGMs for most t, in particular when leveraging a mixed score formulation (see next section).

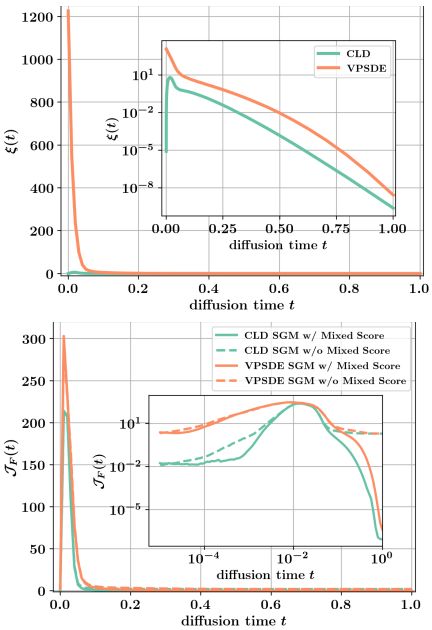


Figure 2: *Top:* Difference $\xi(t)$ (via $L2$ norm) between score of diffused data and score of Normal distribution. *Bottom:* Frobenius norm of Jacobian $\mathcal{J}_F(t)$ of the neural network defining the score function for different t. The underlying data distribution is a mixture of Normals. *Insets:* Different axes (see App. E.1 for detailed definitions of $\xi(t)$ and $\mathcal{J}_F(t)$).

3.2 Scalable Training

A Practical Objective. We cannot train directly with Eq. (6), since we do not have access to the marginal distribution $p_t(\mathbf{u}_t)$. As presented in Sec. 2, we could employ denoising score matching (DSM) and instead sample $\mathbf{u}_0$, and diffuse those samples, which would lead to a tractable objective. However, recall that in CLD the distribution at $t=0$ is the product of a complex data distribution and a Normal distribution over the initial velocity. Therefore, we propose a hybrid version of score matching (Hyvärinen, 2005) and denoising score matching (Vincent, 2011), which we call *hybrid score matching* (HSM). In HSM, we draw samples from $p_0(\mathbf{x}_0)=p_{\text{data}}(\mathbf{x}_0)$ as in DSM, but then diffuse those samples while marginalizing over the full initial velocity distribution $p_0(\mathbf{v}_0)=\mathcal{N}(\mathbf{v};\mathbf{0}_d,\gamma M\boldsymbol{I}_d)$ as in regular SM (HSM is discussed in detail in App. C). Since $p_0(\mathbf{v}_0)$ is Normal (and $\boldsymbol{f}$ and $\boldsymbol{G}$ affine), $p(\mathbf{u}_t|\mathbf{x}_0)$ is also Normal and this remains tractable. We can write this HSM objective as:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\in[0,T]}\mathbb{E}_{\mathbf{x}_0\sim p_0(\mathbf{x}_0)}\mathbb{E}_{\mathbf{u}_t\sim p_t(\mathbf{u}_t|\mathbf{x}_0)}\left[\lambda(t)\|s_{\boldsymbol{\theta}}(\mathbf{u}_t,t)-\nabla_{\mathbf{v}_t}\log p_t(\mathbf{u}_t|\mathbf{x}_0)\|_2^2\right]. \tag{8}$$

In HSM, the expectation over $p_0(\mathbf{v}_0)$ is essentially solved analytically, while DSM would use a sample-based estimate. In particular, when drawing a sample $\mathbf{u}_0$ to diffuse in DSM, we are essentially placing an infinitely sharp Normal with unbounded score (Kim et al., 2021) at $\mathbf{u}_0$, which requires undesirable modifications or truncation tricks for stable training (Song et al., 2021c; Vahdat et al., 2021). Hence, with DSM we could lose some benefits of the CLD framework discussed in Sec. 3.1, whereas HSM is tailored to CLD and fundamentally avoids such unbounded scores. Closed form expressions for the perturbation kernel $p_t(\mathbf{u}_t|\mathbf{x}_0)$ are provided in App. B.1.

Score Model Parametrization. **(i)** Ho et al. (2020) found that it can be beneficial to parameterize the score model to predict the noise that was used in the reparametrized sampling to generate

perturbed samples $\mathbf{u}_t$. For CLD, $\mathbf{u}_t = \boldsymbol{\mu}_t(\mathbf{x}_0) + \boldsymbol{L}_t \boldsymbol{\epsilon}_{2d}$, where $\boldsymbol{\Sigma}_t = \boldsymbol{L}_t \boldsymbol{L}_t^\top$ is the Cholesky decomposition of $p_t(\mathbf{u}_t|\mathbf{x}_0)$'s covariance matrix, $\boldsymbol{\epsilon}_{2d} \sim \mathcal{N}(\boldsymbol{\epsilon}_{2d}; \mathbf{0}_{2d}, \boldsymbol{I}_{2d})$, and $\boldsymbol{\mu}_t(\mathbf{x}_0)$ is $p_t(\mathbf{u}_t|\mathbf{x}_0)$'s mean. Furthermore, $\nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t|\mathbf{x}_0) = -\ell_t \boldsymbol{\epsilon}_{d:2d}$, where $\boldsymbol{\epsilon}_{d:2d}$ denotes those d components of $\boldsymbol{\epsilon}_{2d}$ that actually affect $\nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t|\mathbf{x}_0)$ (since we take velocity gradients only, not all are relevant).

$$\text{With} \quad \boldsymbol{\Sigma}_t = \underbrace{\begin{pmatrix} \Sigma_t^{xx} & \Sigma_t^{xv} \\ \Sigma_t^{xv} & \Sigma_t^{vv} \end{pmatrix}}_{\text{“per-dimension” covariance matrix}} \otimes \boldsymbol{I}_d, \quad \text{we have} \quad \ell_t := \sqrt{\frac{\Sigma_t^{xx}}{\Sigma_t^{xx}\Sigma_t^{vv} - (\Sigma_t^{xv})^2}}. \tag{9}$$

(ii) Vahdat et al. (2021) showed that it can be beneficial to assume that the diffused marginal distribution is Normal at all times and parametrize the model with a Normal score and a residual "correction". For CLD, the score is indeed Normal at $t = 0$ (due to the independently initialized $\mathbf{x}$ and $\mathbf{v}$ at t=0). Similarly, the target score is close to Normal for large t, as we approach the equilibrium.

Based on **(i)** and **(ii)**, we parameterize $s_{\boldsymbol{\theta}}(\mathbf{u}_t, t) = -\ell_t \alpha_{\boldsymbol{\theta}}(\mathbf{u}_t, t)$ with $\alpha_{\boldsymbol{\theta}}(\mathbf{u}_t, t) = \ell_t^{-1} \mathbf{v}_t / \Sigma_t^{vv} + \alpha'_{\boldsymbol{\theta}}(\mathbf{u}_t, t)$, where Σ_t^{vv} corresponds to the v-v component of the "per-dimension" covariance matrix of the Normal distribution $p_t(\mathbf{u}_t|\mathbf{x}_0 = \mathbf{0}_d)$. In other words, we assumed $p_0(\mathbf{x}_0) = \delta(\mathbf{x})$ when defining the analytic term of the score model. Formally, $-\mathbf{v}/\Sigma_t^{vv}$ is the score of a Normal distribution with covariance $\hat{\Sigma}_t^{vv} \boldsymbol{I}_d$. Following Vahdat et al. (2021), we refer to this parameterization as *mixed score parameterization*. Alternative model parameterizations are possible, but we leave their exploration to future work. With this definition, the HSM training objective becomes (details in App. B.3):

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t \in [0,T]} \mathbb{E}_{\mathbf{x}_0 \sim p_0(\mathbf{x}_0)} \mathbb{E}_{\boldsymbol{\epsilon}_{2d} \sim \mathcal{N}(\boldsymbol{\epsilon}_{2d}; \mathbf{0}_{2d}, \boldsymbol{I}_{2d})} \left[\lambda(t) \ell_t^2 \| \boldsymbol{\epsilon}_{d:2d} - \alpha_{\boldsymbol{\theta}}(\mathbf{u}_t = \boldsymbol{\mu}_t(\mathbf{x}_0) + \boldsymbol{L}_t \boldsymbol{\epsilon}_{2d}, t) \|_2^2 \right], \tag{10}$$

which corresponds to training the model to predict the noise only injected into the velocity during reparametrized sampling of $\mathbf{u}_t$, similar to noise prediction in Ho et al. (2020); Song et al. (2021c).

Objective Weightings. For $\lambda(t) = \Gamma\beta$, the objective corresponds to maximum likelihood learning (Song et al., 2021b) (see App. B.3). Analogously to prior work (Ho et al., 2020; Vahdat et al., 2021; Song et al., 2021b), an objective better suited for high quality image synthesis can be obtained by setting $\lambda(t) = \ell_t^{-2}$, which corresponds to "dropping the variance prefactor" ℓ_t^2.

3.3 Sampling from CLD-based SGMs

To sample from the CLD-based SGM we can either directly simulate the reverse-time diffusion process (Eq. (2)) or, alternatively, solve the corresponding probability flow ODE (Song et al., 2021c;b) (see App. B.5). To simulate the SDE of the reverse-time diffusion process, previous works often relied on Euler-Maruyama (EM) (Kloeden & Platen, 1992) and related methods (Ho et al., 2020; Song et al., 2021c; Jolicoeur-Martineau et al., 2021a). We derive a new solver, tailored to CLD-based models. Here, we provide the high-level ideas and derivations (see App. D for details).

Our generative SDE can be written as (with $\bar{\mathbf{u}}_t = \mathbf{u}_{T-t}$, $\bar{\mathbf{x}}_t = \mathbf{x}_{T-t}$, $\bar{\mathbf{v}}_t = \mathbf{v}_{T-t}$):

$$\begin{pmatrix} d\bar{\mathbf{x}}_t \\ d\bar{\mathbf{v}}_t \end{pmatrix} = \underbrace{\begin{pmatrix} -M^{-1}\bar{\mathbf{v}}_t \\ \bar{\mathbf{x}}_t \end{pmatrix} \beta dt}_{A_H} + \underbrace{\begin{pmatrix} 0 \\ -\Gamma M^{-1}\bar{\mathbf{v}}_t \end{pmatrix} \beta dt + \begin{pmatrix} 0 \\ \sqrt{2\Gamma\beta} d\mathbf{w}_t \end{pmatrix}}_{A_O} + \underbrace{\begin{pmatrix} 0 \\ 2\Gamma \left[\mathbf{s}(\bar{\mathbf{u}}_t, T-t) + M^{-1}\bar{\mathbf{v}}_t \right] \end{pmatrix} \beta dt}_{S} \tag{11}$$

It consists of a Hamiltonian component A_H, an Ornstein-Uhlenbeck process A_O, and the score model term S. We could use EM to integrate this SDE; however, standard Euler methods are not well-suited for Hamiltonian dynamics (Leimkuhler & Reich, 2005; Neal, 2011). Furthermore, if S was 0, we could solve the SDE in closed form. This suggests the construction of a novel integrator.

We use the Fokker-Planck[2] operator formalism (Tuckerman, 2010; Leimkuhler & Matthews, 2013; 2015). Using a similar notation as Leimkuhler & Matthews (2013), the Fokker-Planck equation corresponding to the generative SDE is $\partial p_t(\bar{\mathbf{u}}_t)/\partial t = (\hat{\mathcal{L}}_A^* + \hat{\mathcal{L}}_S^*) p_t(\bar{\mathbf{u}}_t)$, where $\hat{\mathcal{L}}_A^*$ and $\hat{\mathcal{L}}_S^*$ are the non-commuting Fokker-Planck operators corresponding to the $A := A_H + A_O$ and S terms, respectively. Expressions for $\hat{\mathcal{L}}_A^*$ and $\hat{\mathcal{L}}_S^*$ are in App. D. We can construct a formal, but intractable solution of the generative SDE as $\bar{\mathbf{u}}_t = e^{t(\hat{\mathcal{L}}_A^* + \hat{\mathcal{L}}_S^*)} \bar{\mathbf{u}}_0$, where the operator $e^{t(\hat{\mathcal{L}}_A^* + \hat{\mathcal{L}}_S^*)}$ (known as the *classical*

[2]The *Fokker-Planck operator* is also known as *Kolmogorov operator*. If the underlying dynamics is fully Hamiltonian, it corresponds to the *Liouville operator* (Leimkuhler & Matthews, 2015; Tuckerman, 2010).

propagator in statistical physics) propagates states $\bar{\mathbf{u}}_0$ for time t according to the dynamics defined by the combined operators $\hat{\mathcal{L}}_A^* + \hat{\mathcal{L}}_S^*$. Although this operation is not analytically tractable, it can serve as starting point to derive a practical integrator. Using the symmetric Trotter theorem or Strang splitting formula as well as the Baker–Campbell–Hausdorff formula (Trotter, 1959; Strang, 1968; Tuckerman, 2010), it can be shown that:

$$e^{t(\hat{\mathcal{L}}_A^* + \hat{\mathcal{L}}_S^*)} = \lim_{N\to\infty} \left[e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*} e^{\delta t \hat{\mathcal{L}}_S^*} e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*} \right]^N \approx \left[e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*} e^{\delta t \hat{\mathcal{L}}_S^*} e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*} \right]^N + \mathcal{O}(N\delta t^3) \tag{12}$$

with time step $\delta t := t/N$, where $N \in \mathbb{N}^+$. The expression suggests that instead of directly evaluating the intractable $e^{t(\hat{\mathcal{L}}_A^* + \hat{\mathcal{L}}_S^*)}$, we can discretize the dynamics over t into N pieces of step size δt, such that we only need to apply the *individual* $e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*}$ and $e^{\delta t \hat{\mathcal{L}}_S^*}$ many times one after another for small time steps δt. A finer discretization results in a smaller error (since $N{=}t/\delta t$ the error effectively scales as $\mathcal{O}(\delta t^2)$ for fixed t). Hence, this implies an integration method. Indeed, $e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*}\bar{\mathbf{u}}_t$ *is* available in closed form, as mentioned before. However, $e^{\delta t \hat{\mathcal{L}}_S^*}\bar{\mathbf{u}}_t$ is not. Therefore, we approximate this latter component of the integrator via a standard Euler step. Thus, the integrator formally has an error of the same order as standard EM methods. Nevertheless, as long as the dynamics is not dominated by the S component, our proposed integration scheme is expected to be more accurate than EM, since we split off the analytically tractable part and only use an Euler approximation for the S term. Recall that the model only needs to learn the score of the conditional distribution $p_t(\mathbf{v}_t|\mathbf{x}_t)$, which is close to Normal for much of the diffusion, in which case the S term will indeed be small. This suggests that the generative SDE dynamics are in fact dominated by A_H and A_O in practice. Note that only the propagator $e^{\delta t \hat{\mathcal{L}}_S^*}$ is computationally expensive, as it involves evaluating the neural network. We coin our novel SDE integrator for CLD-based SGMs *Symmetric Splitting CLD Sampler* (SSCS). A detailed derivation, analyses, and a formal algorithm are presented in App. D.

4 Related Work

Relations to Statistical Mechanics and Molecular Dynamics. Learning a mapping between a simple, tractable and a complex distribution as in SGMs is inspired by annealed importance sampling (Neal, 2001) and the Jarzynski equality from non-equilibrium statistical mechanics (Jarzynski, 1997a;b; 2011). However, after Sohl-Dickstein et al. (2015), little attention has been given to the origins of SGMs in statistical mechanics. Intuitively, in SGMs the diffusion process is initialized in a non-equilibrium state $\mathbf{u}_0$ and we would like to bring the system to equilibrium, i.e., the tractable prior distribution, *as quickly and as smoothly as possible* to enable efficient denoising. This "equilibration problem" is a much-studied problem in statistical mechanics, particularly in molecular dynamics, where a molecular system is often simulated in thermodynamic equilibrium. Algorithms to quickly and smoothly bring a system to and maintain at equilibrium are known as *thermostats*. In fact, CLD is inspired by the Langevin thermostat (Bussi & Parrinello, 2007). In molecular dynamics, advanced thermostats are required in particular for "multiscale" systems that show complex behaviors over multiple time- and length-scales. Similar challenges are also encountered when modeling complex data, such as natural images. Hence, the vast literature on thermostats (Andersen, 1980; Nosé, 1984; Hoover, 1985; Martyna et al., 1992; Hünenberger, 2005; Bussi et al., 2007; Ceriotti et al., 2009; 2010; Tuckerman, 2010) may be valuable for the development of future SGMs. Also the framework for developing SSCS is borrowed from statistical mechanics. The same techniques have been used to derive molecular dynamics algorithms (Tuckerman et al., 1992; Bussi & Parrinello, 2007; Ceriotti et al., 2010; Leimkuhler & Matthews, 2013; 2015; Kreis et al., 2017).

Further Related Work. Generative modeling by learning stochastic processes has a long history (Movellan, 2008; Lyu, 2009; Sohl-Dickstein et al., 2011; Bengio et al., 2014; Alain et al., 2016; Goyal et al., 2017; Bordes et al., 2017; Song & Ermon, 2019; Ho et al., 2020). We build on Song et al. (2021c), which introduced the SDE framework for modern SGMs. Nachmani et al. (2021) recently introduced non-Gaussian diffusion processes with different noise distributions. However, the noise is still injected directly into the data, and no improved sampling schemes or training objectives are introduced. Vahdat et al. (2021) proposed LSGM, which is complementary to CLD: we improve the diffusion process itself, whereas LSGM "simplifies the data" by first embedding it into a smooth latent space. LSGM is an overall more complicated framework, as it is trained in two stages and relies on additional encoder and decoder networks. Recently, techniques to accelerate sampling from pre-trained SGMs have been proposed (San-Roman et al., 2021; Watson et al., 2021;

Kong & Ping, 2021; Song et al., 2021a). Importantly, these methods usually do not permit straightforward likelihood estimation (Song et al., 2021a). Therefore, it is unclear whether they suffer from decreased diversity in the few-step limit. Furthermore, they are usually developed for discrete-step diffusion models and not the continuous time framework, which we use.

A complementary work to CLD is "Gotta Go Fast" (GGF) (Jolicoeur-Martineau et al., 2021a), which introduces an adaptive SDE solver for SGMs, tuned towards image synthesis. GGF uses standard Euler-based methods under the hood (Kloeden & Platen, 1992; Roberts, 2012), in contrast to our SSCS that is derived from first principles. Furthermore, our SDE integrator for CLD does not make any data-specific assumptions and performs extremely well even without adaptive step sizes.

Some works study SGMs for maximum likelihood training (Song et al., 2021b; Kingma et al., 2021; Huang et al., 2021). Note that we did not focus on training our models towards high likelihood. Furthermore, Chen et al. (2020) and Huang et al. (2020) recently trained augmented Normalizing Flows, which have conceptual similarities with our velocity augmentation. Methods leveraging auxiliary variables similar to our velocities are also used in statistics—such as Hamiltonian Monte Carlo (Neal, 2011)—and have found applications, for instance, in Bayesian machine learning (Chen et al., 2014; Ding et al., 2014; Shang et al., 2015). As shown in Ma et al. (2019), our velocity is equivalent to momentum in gradient descent and related methods (Polyak, 1964; Kingma & Ba, 2015). Momentum accelerates optimization; the velocity in CLD accelerates mixing in the diffusion process. Lastly, our CLD method can be considered as a second-order Langevin algorithm, but even higher-order schemes are possible (Mou et al., 2021) and could potentially further improve SGMs.

5 Experiments

Architectures. We focus on image synthesis and implement CLD-based SGMs using NCSN++ and DDPM++ (Song et al., 2021c) with 6 input channels (for velocity and data) instead of 3.

Relevant Hyperparameters. CLD's hyperparameters are chosen as $\beta{=}4$, $\Gamma{=}1$ (or equivalently $M^{-1}{=}4$) in all experiments. We set the variance scaling of the inital velocity distribution to $\gamma{=}0.04$ and use the proposed HSM objective with the weighting $\lambda(t){=}\ell_t^{-2}$, which promotes image quality.

Sampling. We generate model samples via: **(i)** Probability flow using a Runge–Kutta 4(5) method; reverse-time generative SDE sampling using either **(ii)** EM or **(iii)** our SSCS. For methods without adaptive stepsize (EM and SSCS), we use evaluation times chosen according to a quadratic function, like previous work (Song et al., 2021a; Kong & Ping, 2021; Watson et al., 2021) (indicated by QS).

Evaluation. We measure image sample quality for CIFAR-10 via Fréchet inception distance (FID) with 50k samples (Heusel et al., 2017). We also evaluate an upper bound on the negative log-likelihood (NLL): $-\log p(\mathbf{x}_0){\leq}-\mathbb{E}_{\mathbf{v}_0\sim p(\mathbf{v}_0)}\log p_\varepsilon(\mathbf{x}_0,\mathbf{v}_0){-}H$, where H is the entropy of $p(\mathbf{v}_0)$ and $\log p_\varepsilon(\mathbf{x}_0,\mathbf{v}_0)$ is an unbiased estimate of $\log p(\mathbf{x}_0,\mathbf{v}_0)$ from the probability flow ODE (Grathwohl et al., 2019; Song et al., 2021c). As in Vahdat et al. (2021), the stochasticity of $\log p_\varepsilon(\mathbf{x},\mathbf{v})$ prevents us from performing importance weighted NLL estimation over the velocity distribution (Burda et al., 2015). We also record the number of function—neural network—evaluations (NFEs) during synthesis when comparing sampling methods. All implementation details in App. B.5 and E.

5.1 Image Generation

Following Song et al. (2021c), we focus on the widely used CIFAR-10 unconditional image generation benchmark. Our CLD-based SGM achieves an FID of 2.25 based on the probability flow ODE and an FID of 2.23 via simulating the generative SDE (Tab. 1). The only models marginally outperforming CLD are LSGM (Vahdat et al., 2021) and NSCN++/VESDE with 2,000 step predictor-corrector sampling (Song et al., 2021b). However, LSGM uses a model with $\approx 475M$ parameters to achieve its high performance, while we obtain our numbers with a model of $\approx 100M$ parameters. For a fairer comparison, we trained a smaller LSGM also with $\approx 100M$ parameters, which is reported as "LSGM-100M" in Tab. 1 (details in App. E.2.7). Our model has a significantly lower FID score than "LSGM-100M". In contrast to NSCN++/VESDE, we achieve extremely strong results with much fewer NFEs (for example see $n{\in}\{150, 275, 500\}$ in Tab. 3 and also Tab. 2)—the VESDE performs poorly in this regime. We conclude that when comparing models with similar network capacity and under NFE budgets ≤ 500, our CLD-based SGM outperforms all published results in terms of FID. We attribute these positive results to our easier score matching task. Furthermore, our model reaches an NLL bound of 3.31, which is on par with recent works such as Nichol & Dhariwal (2021); Austin

Table 1: Unconditional CIFAR-10 generative performance.

Class	Model	NLL↓	FID↓
Score	CLD-SGM (Prob. Flow) *(ours)*	≤3.31	2.25
	CLD-SGM (SDE) *(ours)*	-	2.23
Score	DDPM++, VPSDE (Prob. Flow) (Song et al., 2021c)	3.13	3.08
	DDPM++, VPSDE (SDE) (Song et al., 2021c)	-	2.41
	DDPM++, sub-VP (Prob. Flow) (Song et al., 2021c)	2.99	2.92
	DDPM++, sub-VP (SDE) (Song et al., 2021c)	-	2.41
	NCSN++, VESDE (SDE) (Song et al., 2021c)	-	2.20
	LSGM (Vahdat et al., 2021)	≤3.43	2.10
	LSGM-100M (Vahdat et al., 2021)	≤2.96	4.60
	DDPM (Ho et al., 2020)	≤3.75	3.17
	NCSN (Song & Ermon, 2019)	-	25.3
	Adversarial DSM (Jolicoeur-Martineau et al., 2021b)	-	6.10
	Likelihood SDE (Song et al., 2021b)	2.84	2.87
	DDIM (100 steps) (Song et al., 2021a)	-	4.16
	FastDDPM (100 steps) (Kong & Ping, 2021)	-	2.86
	Improved DDPM (Nichol & Dhariwal, 2021)	3.37	2.90
	VDM (Kingma et al., 2021)	2.49	7.41 (4.00)
	UDM (Kim et al., 2021)	3.04	2.33
	D3PMs (Austin et al., 2021)	≤3.44	7.34
	Gotta Go Fast (Jolicoeur-Martineau et al., 2021a)	-	2.44
	DDPM Distillation (Luhman & Luhman, 2021)	-	9.36
GANs	SNGAN (Miyato et al., 2018)	-	21.7
	SNGAN+DGflow (Ansari et al., 2021)	-	9.62
	AutoGAN (Gong et al., 2019)	-	12.4
	TransGAN (Jiang et al., 2021)	-	9.26
	StyleGAN2 w/o ADA (Karras et al., 2020)	-	8.32
	StyleGAN2 w/ ADA (Karras et al., 2020)	-	2.92
	StyleGAN2 w/ Diffaug (Zhao et al., 2020)	-	5.79
Aut.-Reg., Flows, VAEs, EBMs	DistAug (Jun et al., 2020)	2.53	42.90
	PixelCNN (Oord et al., 2016)	3.14	65.9
	Glow (Kingma & Dhariwal, 2018)	3.35	48.9
	Residual Flow (Chen et al., 2019)	3.28	46.37
	NVAE (Vahdat & Kautz, 2020)	2.91	23.5
	NCP-VAE (Aneja et al., 2020)	-	24.08
	DC-VAE Parmar et al. (2020)	-	17.90
	IGEBM (Du & Mordatch, 2019)	-	40.6
	VAEBM (Xiao et al., 2021)	-	12.2
	Recovery EBM (Gao et al., 2021)	3.18	9.58

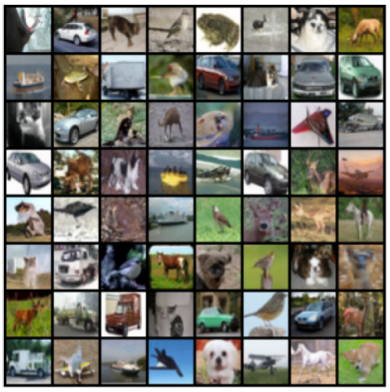


Figure 3: CIFAR-10 samples.

Figure 4: CelebA-HQ-256 samples.

et al. (2021); Vahdat et al. (2021) and indicates that our model is not dropping modes. However, our bound is potentially quite loose (see discussion in App. B.5) and the true NLL might be significantly lower. We did not focus on training our models towards high likelihood.

To demonstrate that CLD is also suitable for high-resolution image synthesis, we additionally trained a CLD-SGM on CelebA-HQ-256, but without careful hyperparameter tuning due to limited compute resources. Model samples in Fig. 4 appear diverse and high-quality (additional samples in App. F).

5.2 Sampling Speed and Synthesis Quality Trade-Offs

We analyze the sampling speed vs. synthesis quality trade-off for CLD-SGM and study SSCS's performance under different NFE budgets in Tabs. 2 and 3. We compare to Song et al. (2021c) and use EM to solve the generative SDE for their VPSDE models and PC (reverse-diffusion + Langevin sampler) for VESDE models, which corresponds to their recommended settings[3]. We also compare to the GGF (Jolicoeur-Martineau et al., 2021a) solver for the generative SDE as well as probability flow sampling with a higher-order adaptive stepsize solver. Further, we compare to LSGM (Vahdat et al., 2021) (using our LSGM-100M model), which also relies on probability flow sampling. We do not compare to few-step sampling methods such as DDIM (Song et al., 2021a), since they do not permit likelihood estimation. This may imply that they trade off synthesis quality for diversity, which makes a fair comparison difficult. With one exception (VESDE with 2,000 NFE) our CLD-SGM outperforms all baselines, both when comparing to adaptive and fixed-step size methods.

Several observations stand out: **(i)** As expected (Sec. 3.3), for CLD, SSCS significantly outperforms EM under limited NFE budgets. When using a fine discretization of the SDE (high NFE), the two perform similarly, which is also expected, as the errors of both methods will become negligible. **(ii)** In the adaptive solver setting, using a simpler ODE solver, we even outperform GGF, which is tuned towards image synthesis. **(iii)** Our CLD-SGM also outperforms the LSGM-100M model in terms of FID. It is worth noting, however, that LSGM was designed primarily for faster synthesis, which it achieves by modeling a smooth distribution in latent space instead of the more complex data distribution directly. This suggests that it would be promising to combine LSGM with CLD and train a CLD-based LSGM, to combine the strengths of the two approaches. It would also be

[3] colab.research.google.com/drive/1dRR_0gNRmfLtPavX2APzUggBuXyjWW55

Table 2: *(right)* Performance using adaptive stepsize solvers (*ODE* is based on probability flow, *GGF* simulates generative SDE). †: taken from Jolicoeur-Martineau et al. (2021a). LSGM corresponds to the small LSGM-100M model for fair comparison (denoted as LSGM-100M in Tab. 1, details in App. E.2.7). Error tolerances were chosen to obtain approximately similar NFEs.

Model	Solver	NFEs↓	FID↓
CLD	ODE	312	**2.25**
VPSDE	GGF	330	2.56†
VESDE	GGF	488	2.99†
CLD	ODE	147	**2.71**
VPSDE	ODE	141	2.76
VPSDE	GGF	151	2.73†
VESDE	ODE	182	7.63
VESDE	GGF	170	10.15†
LSGM	ODE	131	4.60

Table 3: *(bottom)* Performance using non-adaptive SDE solvers (for PC, QS performed poorly). †: 2.23 FID is our evaluation, Song et al. (2021c) reports 2.20 FID. See Tab. 9 in App. F.2 for extended results.

		FID at n function evaluations ↓					
Model	Sampler	n=50	n=150	n=275	n=500	n=1000	n=2000
CLD	EM-QS	52.7	7.00	3.24	2.41	**2.27**	**2.23**
CLD	SSCS-QS	**20.5**	**3.07**	**2.38**	**2.25**	2.30	2.29
VPSDE	EM-QS	28.2	4.06	2.65	2.47	2.66	2.60
VESDE	PC	460	216	11.2	3.75	2.43	**2.23**†

interesting to develop a more advanced, adaptive SDE solver that leverages SSCS as the backbone and, for example, potentially test our method within a framework like GGF. Our current SSCS only allows for fixed step sizes—nevertheless, it achieves excellent performance.

5.3 Ablation Studies

We perform ablation studies to study CLD's new hyperparameters (run with a smaller version of our CLD-SGM used above; App. E for details).

Table 4: Mass hyperparameter.

M^{-1}	NLL↓	FID↓
1	≤3.30	3.23
4	≤3.37	3.14
16	≤3.26	3.16

Table 5: Initial velocity distribution width.

γ	NLL↓	FID↓
0.04	≤3.37	3.14
0.4	≤3.15	3.21
1	≤3.15	3.27

Mass Parameter: Tab. 4 shows results for a CLD-SGM trained with different M^{-1} (also recall that M^{-1} and Γ are tied together via $\Gamma^2 = 4M$; we are always in the critical-damping regime). Different mass values perform mostly similarly. Intuitively, training with smaller M^{-1} means that noise flows from the velocity variables $\mathbf{v}_t$ into the data $\mathbf{x}_t$ more slowly, which necessitates a larger time rescaling β. We found that simply tying M^{-1} and β together via β=$8\sqrt{M}$ works well and did not further fine-tune.

Initial Velocity Distribution: Tab. 5 shows results for a CLD-SGM trained with different initial velocity variance scalings γ. Varying γ similarly has only a small effect, but small γ seems slightly beneficial for FID, while the NLL bound suffers a bit. Due to our focus on synthesis quality as measued by FID, we opted for small γ. Intuitively, this means that the data at t=0 is "at rest", and noise flows from the velocity into the data variables only slowly.

Mixed Score: Similar to previous work (Vahdat et al., 2021), we find training with the mixed score (MS) parametrization (Sec. 3.2) beneficial. With MS, we achieve an FID of 3.14, without only 3.56.

Hybrid Score Matching: We also tried training with regular DSM, instead of HSM. However, training often became unstable. As discussed in Sec. 3.2, this is likely because when using standard DSM our CLD would suffer from unbounded scores close to t=0, similar to previous SDEs (Kim et al., 2021). Consequently, we consider our novel HSM a crucial element for training CLD-SGMs.

We conclude that CLD does not come with difficult-to-tune hyperparameters. We expect our chosen hyperparameters to immediately translate to new tasks and models. In fact, we used the same M^{-1}, γ, MS and HSM settings for CIFAR-10 and CelebA-HQ-256 experiments without fine-tuning.

6 Conclusions

We presented *critically-damped Langevin diffusion*, a novel diffusion process for training SGMs. CLD diffuses the data in a smoother, easier-to-denoise manner compared to previous SGMs, which results in smoother neural network-parametrized score functions, fast synthesis, and improved expressivity. Our experiments show that CLD outperforms previous SGMs on image synthesis for similar-capacity models and sampling compute budgets, while our novel SSCS is superior to EM in CLD-based SGMs. From a technical perspective, in addition to proposing CLD, we derive CLD's score matching objective termed as HSM, a variant of denoising score matching suited for CLD, and we derive a tailored SDE integrator for CLD. Inspired by methods used in statistical mechanics, our work provides new insights into SGMs, and implies promising directions for future research.

We believe that CLD can potentially serve as the backbone diffusion process of next generation SGMs. Future work includes using CLD-based SGMs for generative modeling tasks beyond images, combining CLD with techniques for accelerated sampling from SGMs, adapting CLD-based SGMs towards maximum likelihood, and utilizing other thermostating methods from statistical mechanics.

7 Ethics and Reproducibility

Our paper focuses on fundamental algorithmic advances to improve the generative modeling performance of SGMs. As such, the proposed CLD does not imply immediate ethical concerns. However, we validate CLD on image synthesis benchmarks. Generative modeling of images has promising applications, for example for digital content creation and artistic expression (Bailey, 2020), but can also be used for nefarious purposes (Vaccari & Chadwick, 2020; Mirsky & Lee, 2021; Nguyen et al., 2021). It is worth mentioning that compared to generative adversarial networks (Goodfellow et al., 2014), a very popular class of generative models, SGMs have the promise to model the data more faithfully, without dropping modes and introducing problematic biases. Generally, the ethical impact of our work depends on its application domain and the task at hand.

To aid reproducibility of the results and methods presented in our paper, we will make source code to reproduce all quantitative and qualitative results of the paper publicly available, including detailed instructions. Furthermore, all training details and hyperparameters are already in detail described in the Appendix, in particular in App. E.

References

Guillaume Alain, Yoshua Bengio, Li Yao, Jason Yosinski, Éric Thibodeau-Laufer, Saizheng Zhang, and Pascal Vincent. GSNs: generative stochastic networks. *Information and Inference: A Journal of the IMA*, 5(2):210–249, 03 2016. ISSN 2049-8764. URL https://arxiv.org/abs/1503.05571.

Hans C. Andersen. Molecular dynamics simulations at constant pressure and/or temperature. *The Journal of Chemical Physics*, 72(4):2384–2393, 1980. doi: 10.1063/1.439486. URL https://doi.org/10.1063/1.439486.

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

Jyoti Aneja, Alexander Schwing, Jan Kautz, and Arash Vahdat. NCP-VAE: Variational autoencoders with noise contrastive priors. *arXiv preprint arXiv:2010.02917*, 2020.

Abdul Fatir Ansari, Ming Liang Ang, and Harold Soh. Refining deep generative models via discriminator gradient flow. In *International Conference on Learning Representations*, 2021.

Jacob Austin, Daniel Johnson, Jonathan Ho, Danny Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *arXiv preprint arXiv:2107.03006*, 2021.

J. Bailey. The tools of generative art, from flash to neural networks. *Art in America*, 2020.

Yoshua Bengio, Eric Laufer, Guillaume Alain, and Jason Yosinski. Deep Generative Stochastic Networks Trainable by Backprop. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Florian Bordes, Sina Honari, and Pascal Vincent. Learning to Generate Samples from Noise through Infusion Training. In *5th International Conference on Learning Representations, ICLR*, 2017. URL https://openreview.net/forum?id=BJAFbaolg.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Giovanni Bussi and Michele Parrinello. Accurate sampling using Langevin dynamics. *Phys. Rev. E*, 75:056707, 2007.

Giovanni Bussi, Davide Donadio, and Michele Parrinello. Canonical sampling through velocity rescaling. *The Journal of Chemical Physics*, 126(1):014101, 2007. doi: 10.1063/1.2408420. URL https://doi.org/10.1063/1.2408420.

Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning Gradient Fields for Shape Generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

Michele Ceriotti, Giovanni Bussi, and Michele Parrinello. Langevin Equation with Colored Noise for Constant-Temperature Molecular Dynamics Simulations. *Phys. Rev. Lett.*, 102:020601, 2009. URL https://link.aps.org/doi/10.1103/PhysRevLett.102.020601.

Michele Ceriotti, Michele Parrinello, Thomas E. Markland, and David E. Manolopoulos. Efficient stochastic thermostatting of path integral molecular dynamics. *The Journal of Chemical Physics*, 133(12):124104, 2010. URL https://doi.org/10.1063/1.3489925.

Jianfei Chen, Cheng Lu, Biqi Chenli, Jun Zhu, and Tian Tian. VFlow: More expressive generative flows with variational data augmentation. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1660–1669. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/chen20p.html.

Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. WaveGrad: Estimating Gradients for Waveform Generation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=NsMLjcFaO8O.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 2018.

Ricky T. Q. Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, 2019.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic Gradient Hamiltonian Monte Carlo. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. *arXiv:2105.05233*, 2021.

Nan Ding, Youhan Fang, Ryan Babbush, Changyou Chen, Robert D Skeel, and Hartmut Neven. Bayesian Sampling Using Stochastic Gradient Thermostats. In *Advances in Neural Information Processing Systems*, 2014.

J. R. Dormand and P. J. Prince. A family of embedded Runge–Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems*, pp. 3608–3618, 2019.

Chie Furusawa, Shinya Kitaoka, Michael Li, and Yuri Odagiri. Generative Probabilistic Image Colorization. *arXiv preprint arXiv:2109.14518*, 2021.

Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. In *International Conference on Learning Representations*, 2021.

Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. Autogan: Neural architecture search for generative adversarial networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3224–3234, 2019.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Anirudh Goyal, Nan Rosemary Ke, Surya Ganguli, and Yoshua Bengio. Variational Walkback: Learning a Transition Operator as a Stochastic Recurrent Net. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations*, 2019.

Ulrich G Haussmann and Etienne Pardoux. Time Reversal of Diffusions. *The Annals of Probability*, pp. 1188–1205, 1986.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, 2020.

Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded Diffusion Models for High Fidelity Image Generation. *arXiv:2106.15282*, 2021.

William G. Hoover. Canonical dynamics: Equilibrium phase-space distributions. *Phys. Rev. A*, 31:1695–1697, 1985. URL https://link.aps.org/doi/10.1103/PhysRevA.31.1695.

Chin-Wei Huang, Laurent Dinh, and Aaron Courville. Augmented normalizing flows: Bridging the gap between generative flows and latent variable models. *arXiv preprint arXiv:2002.07101*, 2020.

Chin-Wei Huang, Jae Hyun Lim, and Aaron Courville. A Variational Perspective on Diffusion-Based Generative Models and Score Matching. *arXiv:2106.02808*, 2021.

Philippe H. Hünenberger. *Thermostat Algorithms for Molecular Dynamics Simulations*, volume 173 of *Advanced Computer Simulation. Advances in Polymer Science*. Springer, Berlin, Heidelberg, 2005.

Aapo Hyvärinen. Estimation of Non-Normalized Statistical Models by Score Matching. *J. Mach. Learn. Res.*, 6:695–709, December 2005. ISSN 1532-4435.

Christopher Jarzynski. Equilibrium free-energy differences from nonequilibrium measurements: A master-equation approach. *Phys. Rev. E*, 56:5018–5035, 1997a.

Christopher Jarzynski. Nonequilibrium Equality for Free Energy Differences. *Phys. Rev. Lett.*, 78: 2690–2693, 1997b.

Christopher Jarzynski. Equalities and Inequalities: Irreversibility and the Second Law of Thermodynamics at the Nanoscale. *Annual Review of Condensed Matter Physics*, 2(1):329–351, 2011.

Myeonghun Jeong, Hyeongju Kim, Sung Jun Cheon, Byoung Jin Choi, and Nam Soo Kim. Diff-TTS: A Denoising Diffusion Model for Text-to-Speech. *arXiv preprint arXiv:2104.01409*, 2021.

Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two transformers can make one strong gan. *arXiv preprint arXiv:2102.07074*, 2021.

Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta Go Fast When Generating Data with Score-Based Models. *arXiv preprint arXiv:2105.14080*, 2021a.

Alexia Jolicoeur-Martineau, Rémi Piché-Taillefer, Ioannis Mitliagkas, and Remi Tachet des Combes. Adversarial score matching and improved sampling for image generation. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=eLfqMl3z3lq.

Heewoo Jun, Rewon Child, Mark Chen, John Schulman, Aditya Ramesh, Alec Radford, and Ilya Sutskever. Distribution augmentation for generative modeling. In *International Conference on Machine Learning*, pp. 5006–5019. PMLR, 2020.

Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *arXiv preprint arXiv:2006.06676*, 2020.

Dongjun Kim, Seungjae Shin, Kyungwoo Song, Wanmo Kang, and Il-Chul Moon. Score Matching Model for Unbounded Data Score. *arXiv preprint arXiv:2106.05527*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Diederik P Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational Diffusion Models. *arXiv:2107.00630*, 2021.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pp. 10215–10224, 2018.

Peter E. Kloeden. *Numerical solution of stochastic differential equations / Peter E. Kloeden, Eckhard Platen*. Applications of mathematics; v. 23. Springer, Berlin, 1992. ISBN 0387540628.

Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Springer, Berlin, 1992.

Zhifeng Kong and Wei Ping. On Fast Sampling of Diffusion Probabilistic Models. *arXiv:2106.00132*, 2021.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A Versatile Diffusion Model for Audio Synthesis. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=a-xFK8Ymz5J.

Karsten Kreis, Kurt Kremer, Raffaello Potestio, and Mark E. Tuckerman. From classical to quantum and back: Hamiltonian adaptive resolution path integral, ring polymer, and centroid molecular dynamics. *The Journal of Chemical Physics*, 147(24):244104, 2017. URL https://doi.org/10.1063/1.5000701.

Benedict Leimkuhler and Charles Matthews. Rational construction of stochastic numerical methods for molecular sampling. *Applied Mathematics Research eXpress*, 2013(1):34–56, 2013.

Benedict Leimkuhler and Charles Matthews. *Molecular Dynamics: With Deterministic and Stochastic Numerical Methods*. Interdisciplinary Applied Mathematics. Springer, 2015.

Benedict Leimkuhler and Sebastian Reich. *Simulating Hamiltonian Dynamics*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2005.

Haoying Li, Yifan Yang, Meng Chang, Huajun Feng, Zhihai Xu, Qi Li, and Yueting Chen. SRDiff: Single Image Super-Resolution with Diffusion Probabilistic Models. *arXiv preprint arXiv:2104.14951*, 2021.

Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

Shitong Luo and Wei Hu. Diffusion Probabilistic Models for 3D Point Cloud Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

Siwei Lyu. Interpretation and Generalization of Score Matching. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pp. 359–366, Arlington, Virginia, USA, 2009. AUAI Press.

Yi-An Ma, Niladri Chatterji, Xiang Cheng, Nicolas Flammarion, Peter Bartlett, and Michael I Jordan. Is there an analog of nesterov acceleration for mcmc? *arXiv preprint arXiv:1902.00996*, 2019.

Glenn J. Martyna, Michael L. Klein, and Mark Tuckerman. Nosé–hoover chains: The canonical ensemble via continuous dynamics. *The Journal of Chemical Physics*, 97(4):2635–2643, 1992. doi: 10.1063/1.463940. URL https://doi.org/10.1063/1.463940.

Martin W McCall. *Classical Mechanics: From Newton to Einstein: A Modern Introduction, 2nd Edition*. Wiley, Hoboken, N.J., 2010.

Chenlin Meng, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

Yisroel Mirsky and Wenke Lee. The creation and detection of deepfakes: A survey. *ACM Comput. Surv.*, 54(1), 2021.

Gautam Mittal, Jesse Engel, Curtis Hawthorne, and Ian Simon. Symbolic music generation with diffusion models. *arXiv preprint arXiv:2103.16091*, 2021.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Wenlong Mou, Yi-An Ma, Martin J Wainwright, Peter L Bartlett, and Michael I Jordan. High-order langevin diffusion yields an accelerated mcmc algorithm. *J. Mach. Learn. Res.*, 22:42–1, 2021.

Javier R. Movellan. Contrastive Divergence in Gaussian Diffusions. *Neural Computation*, 20(9): 2238–2252, 2008.

Eliya Nachmani, Robin San Roman, and Lior Wolf. Non Gaussian Denoising Diffusion Models. *arXiv preprint arXiv:2106.07582*, 2021.

Radford M. Neal. Annealed importance sampling. *Statistics and Computing*, 2001.

Radford M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162, 2011.

Thanh Thi Nguyen, Quoc Viet Hung Nguyen, Cuong M. Nguyen, Dung Nguyen, Duc Thanh Nguyen, and Saeid Nahavandi. Deep learning for deepfakes creation and detection: A survey. *arXiv preprint arXiv:1909.11573*, 2021.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models. In *International Conference on Machine Learning*, 2021.

Shuichi Nosé. A unified formulation of the constant temperature molecular dynamics methods. *The Journal of Chemical Physics*, 81(1):511–519, 1984. doi: 10.1063/1.447334. URL https://doi.org/10.1063/1.447334.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *International Conference on Machine Learning*, 2016.

Gaurav Parmar, Dacheng Li, Kwonjoon Lee, and Zhuowen Tu. Dual contradistinctive generative autoencoder. *arXiv preprint arXiv:2011.10063*, 2020.

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964. ISSN 0041-5553. URL https://www.sciencedirect.com/science/article/pii/0041555364901375.

A. J. Roberts. Modify the Improved Euler scheme to integrate stochastic differential equations. *arXiv preprint arXiv:1210.0933*, 2012.

Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *arXiv preprint arXiv:2104.07636*, 2021.

Robin San-Roman, Eliya Nachmani, and Lior Wolf. Noise estimation for generative diffusion models. *arXiv preprint arXiv:2104.02600*, 2021.

Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*, volume 10. Cambridge University Press, 2019.

Hiroshi Sasaki, Chris G. Willcocks, and Toby P. Breckon. UNIT-DDPM: UNpaired Image Translation with Denoising Diffusion Probabilistic Models. *arXiv preprint arXiv:2104.05358*, 2021.

Xiaocheng Shang, Zhanxing Zhu, Benedict Leimkuhler, and Amos J Storkey. Covariance-Controlled Adaptive Langevin Thermostat for Large-Scale Bayesian Sampling. In *Advances in Neural Information Processing Systems*, 2015.

Abhishek Sinha, Jiaming Song, Chenlin Meng, and Stefano Ermon. D2c: Diffusion-denoising models for few-shot conditional generation. *arXiv preprint arXiv:2106.06819*, 2021.

Jascha Sohl-Dickstein, Peter Battaglino, and Michael R. DeWeese. Minimum Probability Flow Learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 2011.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning*, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, 2019.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum Likelihood Training of Score-Based Diffusion Models. *arXiv:2101.09258*, 2021b.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*, 2021c.

Gilbert Strang. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, 5(3):506–517, 1968. doi: 10.1137/0705041. URL https://doi.org/10.1137/0705041.

H. F. Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10:545–551, 1959.

M. Tuckerman, B. J. Berne, and G. J. Martyna. Reversible multiple time scale molecular dynamics. *The Journal of Chemical Physics*, 97(3):1990–2001, 1992. doi: 10.1063/1.463137. URL https://doi.org/10.1063/1.463137.

Mark E. Tuckerman. *Statistical Mechanics: Theory and Molecular Simulation*. Oxford University Press, New York, 2010.

Cristian Vaccari and Andrew Chadwick. Deepfakes and disinformation: Exploring the impact of synthetic political video on deception, uncertainty, and trust in news. *Social Media + Society*, 6(1):2056305120903408, 2020. URL https://doi.org/10.1177/2056305120903408.

Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based Generative Modeling in Latent Space. *arXiv:2106.05931*, 2021.

Pascal Vincent. A Connection Between Score Matching and Denoising Autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

Daniel Watson, Jonathan Ho, Mohammad Norouzi, and William Chan. Learning to Efficiently Sample from Diffusion Probabilistic Models. *arXiv:2106.03802*, 2021.

Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. Vaebm: A symbiosis between variational autoencoders and energy-based models. In *International Conference on Learning Representations*, 2021.

Richard Zhang. Making Convolutional Networks Shift-Invariant Again. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7324–7334. PMLR, 09–15 Jun 2019.

Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *Advances in Neural Information Processing Systems*, 33, 2020.

Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. *arXiv preprint arXiv:2104.03670*, 2021.

A LANGEVIN DYNAMICS

Here, we discuss different aspects of Langevin dynamics. Recall the Langevin dynamics, Eq. (5), from the main paper:

$$\begin{pmatrix} d\mathbf{x}_t \\ d\mathbf{v}_t \end{pmatrix} = \underbrace{\begin{pmatrix} M^{-1}\mathbf{v}_t \\ -\mathbf{x}_t \end{pmatrix} \beta dt}_{\text{Hamiltonian component}=:H} + \underbrace{\begin{pmatrix} 0 \\ -\Gamma M^{-1}\mathbf{v}_t \end{pmatrix} \beta dt + \begin{pmatrix} 0 \\ \sqrt{2\Gamma\beta} \end{pmatrix} d\mathbf{w}_t}_{\text{Ornstein-Uhlenbeck process}=:O}. \tag{13}$$

A.1 DIFFERENT DAMPING RATIOS

As discussed in Sec. 3, Langevin dynamics can be run with different ratios between mass M and squared friction Γ^2. To recap from the main paper:

(i) For $\Gamma^2 < 4M$ (*underdamped* Langevin dynamics), the Hamiltonian component dominates, which implies oscillatory dynamics of $\mathbf{x}_t$ and $\mathbf{v}_t$ that slow down convergence to equilibrium.

(ii) For $\Gamma^2 > 4M$ (*overdamped* Langevin dynamics), the O-term dominates which also slows down convergence, since the accelerating effect by the Hamiltonian component is suppressed due to the strong noise injection.

(iii) For $\Gamma^2 = 4M$ (*critical-damping*), an ideal balance is achieved and convergence to $p_{\text{EQ}}(\mathbf{u})$occurs as fast as possible in a smooth manner without oscillations.

In Fig. 5, we visualize diffusion trajectories according to Langevin dynamics run in the different damping regimes. We observe that underdamped Langevin dynamics show undesired oscillatory behavior, while overdamped Langevin dynamics perform very inefficiently, too. Critical-damping achieves a good balance between the two and mixes and converges quickly. In fact, it can be shown to be optimal in terms of convergence; see, for example, McCall (2010).

Consequently, we propose to set $\Gamma^2 = \Gamma^2_{\text{critical}} := 4M$ in CLD.

A.2 VERY HIGH FRICTION LIMIT AND CONNECTION TO PREVIOUS SDES IN SGMS

Let us re-write the above Langevin dynamics and consider the more general case with time-dependent $\beta(t)$:

$$d\mathbf{x}_t = M^{-1}\mathbf{v}_t\beta(t)dt, \tag{14}$$

$$d\mathbf{v}_t = -\underbrace{\mathbf{x}_t\beta(t)dt}_{\textbf{(ii)}\text{: potential term}} - \underbrace{\Gamma M^{-1}\mathbf{v}_t\beta(t)dt}_{\textbf{(iii)}\text{: friction term}} + \underbrace{\sqrt{2\Gamma\beta(t)}d\mathbf{w}_t}_{\textbf{(iv)}\text{: noise term}}. \tag{15}$$

To solve this SDE, let us assume a simple Euler-based integration scheme, with the update equation for a single step at time t (this integration scheme would not be optimal, as discussed in Sec. 3.3., however, it would be accurate for sufficiently small time steps and we just need this to make the connection to previous works like the VPSDE):

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \beta(t)M^{-1}\mathbf{v}_{n+1}\delta t, \tag{16}$$

$$\mathbf{v}_{n+1} = \underbrace{\mathbf{v}_n}_{\textbf{(i)}\text{: current step velocity}} - \underbrace{\beta(t)\mathbf{x}_n\delta t}_{\textbf{(ii)}\text{: potential term}} - \underbrace{\beta(t)\Gamma M^{-1}\mathbf{v}_n\delta t}_{\textbf{(iii)}\text{: friction term}} + \underbrace{\sqrt{2\beta(t)\Gamma}\mathcal{N}(0,\delta t)}_{\textbf{(iv)}\text{: noise term}}, \tag{17}$$

Now, let's assume a friction coefficient $\Gamma = \Gamma_{\text{max}} := \frac{M}{\beta(t)\delta t}$. Since the time step δt is usually very small, this correspond to a very high friction. In fact, it can be considered the maximum friction limit, at which the friction is so large that the current step velocity **(i)** is completely cancelled out by the friction term **(iii)**. We obtain:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \beta(t)M^{-1}\mathbf{v}_{n+1}\delta t \tag{18}$$

$$\mathbf{v}_{n+1} = -\beta(t)\mathbf{x}_t\delta t + \sqrt{2\frac{M}{\delta t}}\mathcal{N}(0,\delta t). \tag{19}$$

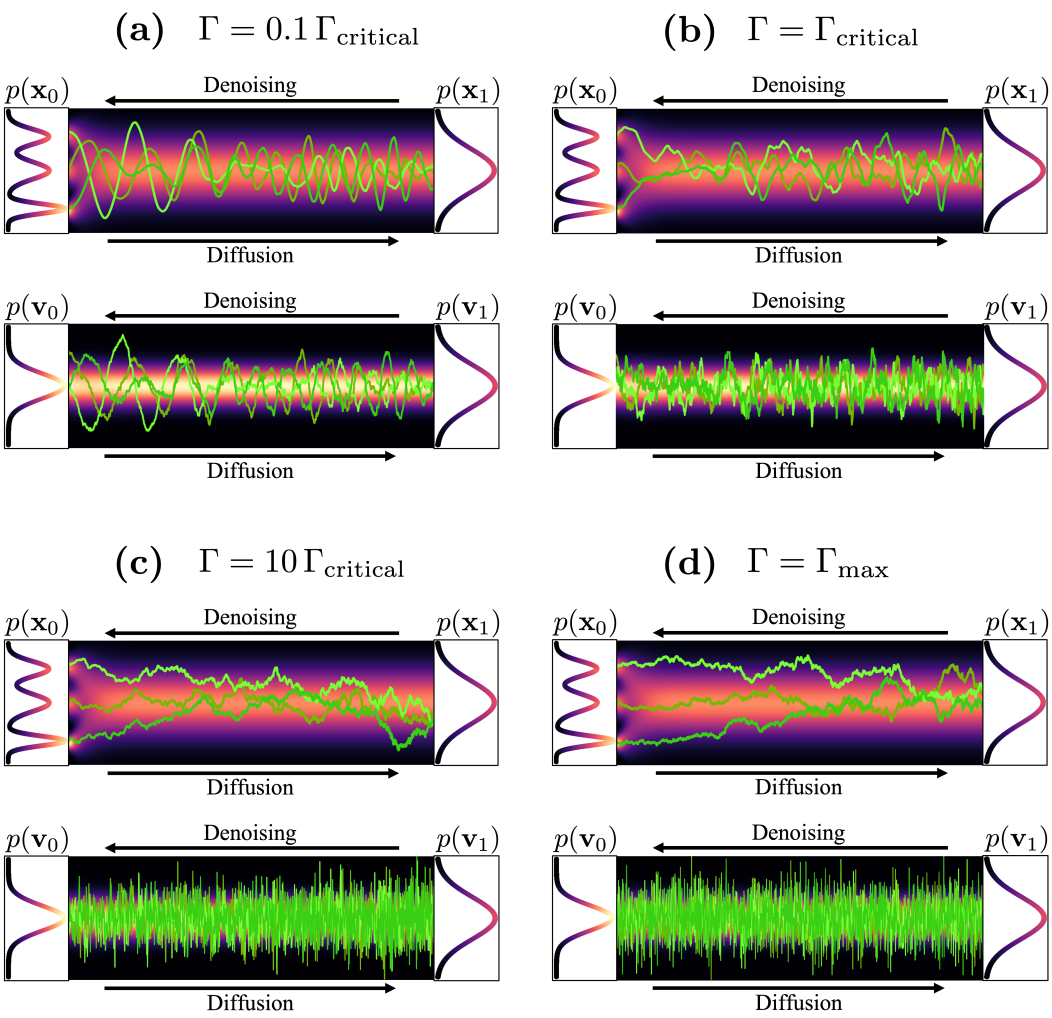


Figure 5: Langevin dynamics in different damping regimes. Each pair of visualizations corresponds to the (coupled) evolution of data $\mathbf{x}_t$ and velocities $\mathbf{v}_t$. We show the marginal **(red) probabilities** and the projections of the **(green) trajectories**. The probabilities always correspond to the same optimal setting $\Gamma = \Gamma_{\text{critical}}$ (recall that $\Gamma_{\text{critical}} = 2\sqrt{M}$ and $\Gamma_{\text{max}} = M/(\beta(t)\delta t)$; see Sec. A.2). The trajectories correspond to different Langevin trajectories run in the different regimes with indicated friction coefficients Γ. We see in **(b)**, that for *critical damping* the $\mathbf{x}_t$ trajectories quickly explore the space and converge according to the distribution indicated by the underlying probability. In the *under-damped regime* **(a)**, even though the trajectories mix quickly we observe undesired oscillatory behavior. For *over-damped Langevin dynamics*, **(c)** and **(d)**, the $\mathbf{x}_t$ trajectories mix and converge only very slowly. Note that the visualized diffusion uses different hyperparameters compared to the diffusion shown in Fig. 1 in the main text: Here, we have chosen a much larger β, such that also the slow overdamped Langevin dynamics trajectories shown here mix a little bit over the visualized diffusion time (while the probability distribution and the trajectories for critical damping converge almost instantly).

Now the velocity update, Eq. (19), does not depend on the current step velocity on the right-hand-side anymore. Hence, we can insert Eq. (19) directly into Eq. (18) and obtain:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \beta(t)^2 M^{-1}\mathbf{x}_n \delta t^2 + \sqrt{2\beta(t)^2 \delta t M^{-1}}\mathcal{N}(0, \delta t) \tag{20}$$

$$= \mathbf{x}_n - \beta(t)^2 M^{-1}\mathbf{x}_n \delta t^2 + \sqrt{2\beta(t)^2 \delta t^2 M^{-1}}\mathcal{N}(0, 1). \tag{21}$$

$$\tag{22}$$

Re-defining $\delta t' := \delta t^2$ and $\beta'(t) := \beta(t)^2$, we obtain

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \beta'(t) M^{-1}\mathbf{x}_n \delta t' + \sqrt{2\beta'(t) \delta t' M^{-1}}\mathcal{N}(0, 1), \tag{23}$$

$$\tag{24}$$

which corresponds to the high-friction overdamped Langevin dynamics that are frequently run, for example, to train energy-based generative models (Du & Mordatch, 2019; Xiao et al., 2021).

Let's further absorb the mass M^{-1} and the time step $\delta t'$ into the time rescaling, defining $\hat{\beta}(t) := 2\beta'(t)M^{-1}\delta t'$. We obtain:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \frac{1}{2}\hat{\beta}(t)\mathbf{x}_n + \sqrt{\hat{\beta}(t)}\mathcal{N}(0,1) \tag{25}$$

$$= (1 - \frac{1}{2}\hat{\beta}(t))\mathbf{x}_n + \sqrt{\hat{\beta}(t)}\mathcal{N}(0,1) \tag{26}$$

$$\approx \sqrt{1-\hat{\beta}(t)}\mathbf{x}_n + \sqrt{\hat{\beta}(t)}\mathcal{N}(0,1), \tag{27}$$

$$\tag{28}$$

where the last approximation is true for sufficiently small $\hat{\beta}(t)$. However, this expression corresponds to

$$\mathbf{x}_{n+1} \sim \mathcal{N}(\mathbf{x}_{n+1}; \sqrt{1-\hat{\beta}(t)}\mathbf{x}_n, \hat{\beta}(t)) \tag{29}$$

which is exactly the transition kernel of the VPSDE's Markov chain (Ho et al., 2020; Song et al., 2021c). We see that the VPSDE corresponds to the high-friction limit of a more general Langevin dynamics-based diffusion process of the form of Eq. (13).

If we assume a diffusion as above but with the potential term **(ii)** set to 0, we can similarly derive the VESDE Song et al. (2021c) as a high-friction limit of the corresponding diffusion. Generally, all previously used diffusions that inject noise directly into the data variables correspond to such high-friction diffusions.

In conclusion, we see that previous high-friction diffusions require an excessive amount of noise to be injected to bring the dynamics to the prior, which intuitively makes denoising harder. For our CLD in the critical damping regime we can run the diffusion for a much shorter time or, equivalently, can inject less noise to converge to the equilibrium, i.e., the prior.

B Critically-Damped Langevin Diffusion

Here, we present further details about our proposed critically-damped Langevin diffusion (CLD). We provide the derivations and formulas that were not presented in the main paper in the interest of brevity.

B.1 Perturbation Kernel

To recap from the main text, in this work we propose to augment the data $\mathbf{x}_t \in \mathbb{R}^d$ with auxiliary *velocity* variables $\mathbf{v}_t \in \mathbb{R}^d$. We then run the following diffusion process in the joint $\mathbf{x}_t$-$\mathbf{v}_t$-space

$$d\mathbf{u}_t := \begin{pmatrix}\mathbf{x}_t \\ \mathbf{v}_t\end{pmatrix} = \boldsymbol{f}(\mathbf{u}_t, t) + \boldsymbol{G}(\mathbf{u}_t, t)d\mathbf{w}_t \tag{30}$$

$$\boldsymbol{f}(\mathbf{u}_t, t) = f \otimes \boldsymbol{I}_d \mathbf{u}_t, \quad f := \begin{pmatrix} 0 & \beta(t)M^{-1} \\ -\beta(t) & -\beta(t)\Gamma M^{-1}\end{pmatrix}, \tag{31}$$

$$\boldsymbol{G}(\mathbf{u}_t, t) = G \otimes \boldsymbol{I}_d, \quad G := \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{2\Gamma\beta(t)}\end{pmatrix}, \tag{32}$$

where $\mathbf{w}_t$ is a standard Wiener process in $\mathbb{R}^{2d}$ and $\beta\colon [0,T] \to \mathbb{R}_0^+$ is a time rescaling [4]. In particular, we consider the *critically-damped Langevin diffusion* which can be obtained by setting $M = \Gamma^2/4$, resulting in the following drift kernel

$$\boldsymbol{f}_{\text{CLD}}(\mathbf{u}_t, t) = f_{\text{CLD}} \otimes \boldsymbol{I}_d \mathbf{u}_t, \quad f_{\text{CLD}} := \begin{pmatrix} 0 & 4\beta(t)\Gamma^{-2} \\ -\beta(t) & -4\beta(t)\Gamma^{-1}\end{pmatrix}. \tag{33}$$

Since we only consider the critically-damped cased in this work, we redefine $\boldsymbol{f} := \boldsymbol{f}_{\text{CLD}}$ and $f := f_{\text{CLD}}$ for simplicity. Since our drift $\boldsymbol{f}$ and diffusion $\boldsymbol{G}$ coefficients are affine, $\mathbf{u}_t$ is Normally distributed for all $t \in [0,T]$ if $\mathbf{u}_{t'}$ is Normally distributed for some $t' \in [0,T]$ (Särkkä &

[4] For our experiments, we only used constant β; however, for generality, we present all derivations for time dependent $\beta(t)$.

Solin, 2019). In particular, given that $\mathbf{u}_{t'} \sim \mathcal{N}(\mathbf{u}_{t'}; \boldsymbol{\mu}_{t'}, \boldsymbol{\Sigma}_{t'} = \Sigma_{t'} \otimes \boldsymbol{I}_d)$, we derive expressions for $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t = \Sigma_t \otimes \boldsymbol{I}_d$, the mean and the covariance matrix of $\mathbf{u}_t$, respectively. Here $\Sigma_{t'}$ is a positive-definite diagonal 2-by-2 matrix and Σ_t is a symmetric positive-definite 2-by-2 matrix.

Following (Section 6.1, Särkkä & Solin, 2019), the mean and covariance matrix of $\mathbf{u}_t$ follow the following respective ordinary differential equations (ODEs)

$$\frac{d\boldsymbol{\mu}_t}{dt} = f \otimes \boldsymbol{I}_d \boldsymbol{\mu}_t, \tag{34}$$

$$\frac{d\boldsymbol{\Sigma}_t}{dt} = (f \otimes \boldsymbol{I}_d)\boldsymbol{\Sigma}_t + [(f \otimes \boldsymbol{I}_d)\boldsymbol{\Sigma}_t]^\top + \left(GG^\top\right) \otimes \boldsymbol{I}_D. \tag{35}$$

Notating $\boldsymbol{\mu}_{t'} = (\mathbf{x}_{t'}, \mathbf{v}_{t'})^\top$, the solutions to the above ODEs are

$$\boldsymbol{\mu}_t = \begin{pmatrix} 2\mathcal{B}(t)\Gamma^{-1}\mathbf{x}_{t'} + 4\mathcal{B}(t)\Gamma^{-2}\mathbf{v}_{t'} + \mathbf{x}_{t'} \\ -\mathcal{B}(t)\mathbf{x}_{t'} - 2\mathcal{B}(t)\Gamma^{-1}\mathbf{v}_{t'} + \mathbf{v}_{t'} \end{pmatrix} e^{-2\mathcal{B}(t)\Gamma^{-1}}, \tag{36}$$

and

$$\boldsymbol{\Sigma}_t = \Sigma(t) \otimes \boldsymbol{I}_d, \tag{37}$$

$$\Sigma(t) = \begin{pmatrix} \Sigma^{xx}(t) & \Sigma^{xv}(t) \\ \Sigma^{xv}(t) & \Sigma^{vv}(t) \end{pmatrix} e^{-4\mathcal{B}(t)\Gamma^{-1}}, \tag{38}$$

$$\Sigma^{xx}(t) = \Sigma^{xx}_{t'} + e^{4\mathcal{B}(t)\Gamma^{-1}} - 1 + 4\mathcal{B}(t)\Gamma^{-1}\left(\Sigma^{xx}_{t'} - 1\right) + 4\mathcal{B}^2(t)\Gamma^{-2}\left(\Sigma^{xx}_{t'} - 2\right) + 16\mathcal{B}(t)^2\Gamma^{-4}\Sigma^{vv}_{t'}, \tag{39}$$

$$\Sigma^{xv}(t) = -\mathcal{B}(t)\Sigma^{xx}_{t'} + 4\mathcal{B}(t)\Gamma^{-2}\Sigma^{vv}_{t'} - 2\mathcal{B}^2(t)\Gamma^{1}\left(\Sigma^{xx}_{t'} - 2\right) - 8\mathcal{B}^2(t)\Gamma^{-3}\Sigma^{vv}_{t'}, \tag{40}$$

$$\Sigma^{vv}(t) = \tfrac{\Gamma^2}{4}\left(e^{4\mathcal{B}(t)\Gamma^{-1}} - 1\right) + \mathcal{B}(t)\Gamma + \Sigma^{vv}_{t'}\left(1 + 4\mathcal{B}(t)^2\Gamma^{-2} - 4\mathcal{B}(t)\Gamma^{-1}\right) + \mathcal{B}(t)^2\left(\Sigma^{xx}_{t'} - 2\right), \tag{41}$$

where $\mathcal{B}(t) = \int_{t'}^{t} \beta(\hat{t})\, d\hat{t}$. For constant $\beta(t) = \beta$ (as is used in all our experiments), we simply have $\mathcal{B}(t) = (t - t')\beta$. The correctness of the proposed mean and covariance matrix can be verified by simply plugging them back in their respective ODEs.

With the above derivations, we can find analytical expression for perturbation kernels $p(\mathbf{u}_t \mid \cdot)$. For example, when conditioning on the the inital data and variance (used in denoising score matching (DSM)), the mean and covariance matrix of the perturbation kernel $p(\mathbf{u}_t \mid \mathbf{u}_0)$ can be obtained by setting $\boldsymbol{\mu}_0 = (\mathbf{x}_0, \mathbf{v}_0)^\top$, $\Sigma^{xx}_0 = 0$, and $\Sigma^{vv}_0 = 0$.

In our experiments, the initial velocity distribution is set to $\mathcal{N}(\mathbf{0}_d, \gamma M \boldsymbol{I}_d)$. Conditioning only on the initial data and marginalizing over the full initial velocity distribution (used in hybrid score matching (HSM), see Sec. C), the mean and covariance matrix of the perturbation kernel $p(\mathbf{u}_t \mid \mathbf{x}_0)$ can be obtained by setting $\boldsymbol{\mu}_0 = (\mathbf{x}_0, \mathbf{0}_d)^\top$, $\Sigma^{xx}_0 = 0$, and $\Sigma^{vv}_0 = \gamma M$.

B.2 Convergence and Equilibrium

Our CLD-based training of SGMs—as well as denoising diffusion models more generally—relies on the fact that the diffusion converges towards an analytically tractable equilibrium distribution for sufficiently large t. In fact, from the above equations we can easily see that,

$$\lim_{t\to\infty} \Sigma^{xx}(t) = 1, \tag{42}$$

$$\lim_{t\to\infty} \Sigma^{xv}(t) = 0, \tag{43}$$

$$\lim_{t\to\infty} \Sigma^{vv}(t) = \frac{\Gamma^2}{4} = M, \tag{44}$$

$$\lim_{t\to\infty} \boldsymbol{\mu}_t = \mathbf{0}_{2d}, \tag{45}$$

which establishes $p_{\text{EQ}}(\mathbf{u}) = \mathcal{N}(\mathbf{x}; \mathbf{0}_d, \boldsymbol{I}_d)\,\mathcal{N}(\mathbf{v}; \mathbf{0}_d, M\boldsymbol{I}_d)$.

Notice that our CLD is an instantiation of the more general Langevin dynamics defined by

$$\begin{pmatrix} d\mathbf{x}_t \\ d\mathbf{v}_t \end{pmatrix} = \begin{pmatrix} M^{-1}\mathbf{v}_t \\ \nabla_{\mathbf{x}_t} \log p_{\text{pot}}(\mathbf{x}_t) \end{pmatrix} \beta dt + \begin{pmatrix} 0 \\ -\Gamma M^{-1}\mathbf{v}_t \end{pmatrix} \beta dt + \begin{pmatrix} 0 \\ \sqrt{2\Gamma\beta} \end{pmatrix} d\mathbf{w}_t. \tag{46}$$

which has the equilibrium distribution $\hat{p}_{\text{EQ}}(\mathbf{u}) = p_{\text{pot}}(\mathbf{x})\,\mathcal{N}(\mathbf{v}; \mathbf{0}_d, M\boldsymbol{I}_d)$ (Leimkuhler & Matthews, 2015; Tuckerman, 2010). However, the perturbation kernel of this Langevin dynamics is not available analytically anymore for arbitrary $p_{\text{pot}}(\mathbf{x})$. In our case, however, we have the analytically tractable $p_{\text{pot}}(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{0}_d, \boldsymbol{I}_d)$. Note that this correspond to the classical "harmonic oscillator" problem from physics.

B.3 CLD Objective

To derive the objective for training CLD-based SGMs, we start with a derivation that targets maximum likelihood training in a similar fashion to Song et al. (2021b). Let p_0 and q_0 be two densities, then

$$D_{\text{KL}}(p_0 \parallel q_0) = D_{\text{KL}}(p_0 \parallel q_0) - D_{\text{KL}}(p_T \parallel q_T) + D_{\text{KL}}(p_T \parallel q_T) \tag{47}$$

$$= -\int_0^T \frac{\partial D_{\text{KL}}(p_t \parallel q_t)}{\partial t} + D_{\text{KL}}(p_T \parallel q_T), \tag{48}$$

where p_t and q_t are the marginal densities of p_0 and q_0, respectively, diffused by our critically-damped Langevin diffusion. As has been shown in Song et al. (2021b), Eq .equation 48 can be written as a mixture of score matching losses. To this end, let us consider the Fokker–Planck equation associated with the the critically-damped Langevin diffusion:

$$\frac{\partial p_t(\mathbf{u}_t)}{\partial t} = \nabla_{\mathbf{u}_t} \cdot \left[\tfrac{1}{2}\left(GG^\top\right) \otimes \boldsymbol{I}_d \nabla_{\mathbf{u}} p_t(\mathbf{u}_t) - p_t(\mathbf{u}_t) f \otimes \boldsymbol{I}_d \mathbf{u}_t\right] \tag{49}$$

$$= \nabla_{\mathbf{u}_t} \cdot \left[\boldsymbol{h}_p(\mathbf{u}_t, t) p_t(\mathbf{u}_t)\right], \quad \boldsymbol{h}_p(\mathbf{u}_t, t) \coloneqq \tfrac{1}{2}\left(GG^\top\right) \otimes \boldsymbol{I}_d \nabla_{\mathbf{u}} \log p_t(\mathbf{u}) - f \otimes \boldsymbol{I}_d \mathbf{u}_t. \tag{50}$$

Similarly, we have $\frac{\partial q_t(\mathbf{u}_t)}{\partial t} = \nabla_{\mathbf{u}_t} \cdot [\boldsymbol{h}_q(\mathbf{u}_t, t) q_t(\mathbf{u}_t)]$. Assuming $\log p_t(\mathbf{u}_t)$ and $\log q_t(\mathbf{u}_t)$ are smooth functions with at most polynomial growth at infinity, we have

$$\lim_{\mathbf{u}_t \to \infty} \boldsymbol{h}_p(\mathbf{u}_t, t) p_t(\mathbf{u}_t) = \lim_{\mathbf{u}_t \to \infty} \boldsymbol{h}_q(\mathbf{u}_t, t) q_t(\mathbf{u}_t) = 0. \tag{51}$$

Using the above fact, we can compute the time-derivative of the Kullback–Leibler divergence between p_t and q_t as

$$\frac{\partial D_{\text{KL}}(p_t \parallel q_t)}{\partial t} = \frac{\partial}{\partial t} \int p_t(\mathbf{u}_t) \log \frac{p_t(\mathbf{u}_t)}{q_t(\mathbf{u}_t)}\, d\mathbf{u}_t \tag{52}$$

$$= -\int p_t(\mathbf{u}_t) \left[\boldsymbol{h}_p(\mathbf{u}_t, t) - \boldsymbol{h}_q(\mathbf{u}_t, t)\right]^\top \left[\nabla_{\mathbf{u}_t} \log p_t(\mathbf{u}_t) - \nabla_{\mathbf{u}_t} \log q_t(\mathbf{u}_t)\right] d\mathbf{u}_t \tag{53}$$

$$= -\frac{1}{2} \int p_t(\mathbf{u}_t) \left[\nabla_{\mathbf{u}_t} \log p_t(\mathbf{u}_t) - \nabla_{\mathbf{u}_t} \log q_t(\mathbf{u}_t)\right]^\top \left(GG^\top\right) \otimes \boldsymbol{I}_d [\nabla_{\mathbf{u}_t} \log p_t(\mathbf{u}_t) - \nabla_{\mathbf{u}_t} \log q_t(\mathbf{u}_t)]\, d\mathbf{u}_t \tag{54}$$

$$= -\beta(t)\Gamma \int p_t(\mathbf{u}_t) \|\nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t) - \nabla_{\mathbf{v}_t} \log q_t(\mathbf{u}_t)\|_2^2\, d\mathbf{u}_t. \tag{55}$$

Notice that due to the form of G, we now have only gradients with respect to the velocity component $\mathbf{v}_t$. Combining the above with Eq. (48), we have

$$D_{\text{KL}}(p_0 \parallel q_0) = \Gamma \mathbb{E}_{t \sim \mathcal{U}[0,1], \mathbf{u}_t \sim p_t(\mathbf{u})} \left[\beta(t) \|\nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t) - \nabla_{\mathbf{v}_t} \log q_t(\mathbf{u}_t)\|_2^2\right] + D_{\text{KL}}(p_T \parallel q_T) \tag{56}$$

$$\approx \mathbb{E}_{t \sim \mathcal{U}[0,1], \mathbf{u}_t \sim p_t(\mathbf{u}_t)} \left[\lambda(t) \|\nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t) - \nabla_{\mathbf{v}_t} \log q_t(\mathbf{u}_t)\|_2^2\right], \tag{57}$$

where $\lambda(t) = \Gamma\beta(t)$. Note that the approximation holds if $D_{\text{KL}}(p_T \parallel q_T)$ is insignificantly small.

As shown in App. C, the above can be rewritten, up to irrelevant constant terms, as either of the following two objectives:

$$\text{HSM}: \mathbb{E}_{t \sim \mathcal{U}[0,1], \mathbf{x}_0 \sim p_0(\mathbf{x}_0), \mathbf{u}_t \sim p_t(\mathbf{u}_t \mid \mathbf{x}_0)} \left[\lambda(t) \|\nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t \mid \mathbf{x}_0) - \nabla_{\mathbf{v}_t} \log q_t(\mathbf{u}_t)\|_2^2\right], \tag{58}$$

$$\text{DSM}: \mathbb{E}_{t \sim \mathcal{U}[0,1], \mathbf{u}_0 \sim p_0(\mathbf{u}_0), \mathbf{u}_t \sim p_t(\mathbf{u}_t \mid \mathbf{u}_0)} \left[\lambda(t) \|\nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t \mid \mathbf{u}_0) - \nabla_{\mathbf{v}_t} \log q_t(\mathbf{u}_t)\|_2^2\right]. \tag{59}$$

For both HSM and DSM, we have shown in App. B.1 that the perturbation kernels $p_t(\mathbf{u}_t \mid \mathbf{x}_0)$ and $p_t(\mathbf{u}_t \mid \mathbf{x}_0)$ are Normal distributions with the following structure of the covariance matrix:

$$\boldsymbol{\Sigma}_t = \Sigma_t \otimes \boldsymbol{I}_d, \quad \Sigma_t = \begin{pmatrix} \Sigma_t^{xx} & \Sigma_t^{xv} \\ \Sigma_t^{xv} & \Sigma_t^{vv} \end{pmatrix}. \tag{60}$$

We can use this fact to compute the gradient $\nabla_{\mathbf{u}_t} p_t(\mathbf{u}_t \mid \cdot)$

$$\begin{aligned} \nabla_{\mathbf{u}_t} p_t(\mathbf{u}_t \mid \cdot) &= -\nabla_{\mathbf{u}_t} \tfrac{1}{2}(\mathbf{u}_t - \boldsymbol{\mu}_t)\boldsymbol{\Sigma}_t^{-1}(\mathbf{u}_t - \boldsymbol{\mu}_t) && (61)\\ &= -\boldsymbol{\Sigma}_t^{-1}(\mathbf{u}_t - \boldsymbol{\mu}_t) && (62)\\ &= -\boldsymbol{L}_t^{-\top}\boldsymbol{L}_t^{-1}(\mathbf{u}_t - \boldsymbol{\mu}_t) && (63)\\ &= -\boldsymbol{L}_t^{-\top}\boldsymbol{\epsilon}_{2d}, && (64) \end{aligned}$$

where $\boldsymbol{\epsilon}_{2d} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_{2d})$ and $\boldsymbol{\Sigma}_t = \boldsymbol{L}_t\boldsymbol{L}_t^\top$ is the Cholesky factorization of the covariance matrix $\boldsymbol{\Sigma}_t$. Note that the structure of $\boldsymbol{\Sigma}_t$ implies that $\boldsymbol{L}_t = L_t \otimes \boldsymbol{I}_d$, where $L_t L_t^\top$ is the Cholesky factorization of Σ_t, i.e,

$$L_t = \begin{pmatrix} L_t^{xx} & L_t^{xv} \\ L_t^{xv} & L_t^{vv} \end{pmatrix} = \begin{pmatrix} \sqrt{\Sigma_t^{xx}} & 0 \\ \frac{\Sigma_t^{xv}}{\sqrt{\Sigma_t^{xx}}} & \sqrt{\frac{\Sigma_t^{xx}\Sigma_t^{vv} - (\Sigma_t^{xv})^2}{\Sigma_t^{xx}}} \end{pmatrix}. \tag{65}$$

Furthermore, we have

$$\begin{aligned} \boldsymbol{L}_t^{-\top} &= L_t^{-\top} \otimes \boldsymbol{I}_d && (66)\\ &= \begin{pmatrix} \sqrt{\Sigma_t^{xx}} & \frac{\Sigma_t^{xv}}{\sqrt{\Sigma_t^{xx}}} \\ 0 & \sqrt{\frac{\Sigma_t^{xx}\Sigma_t^{vv} - (\Sigma_t^{xv})^2}{\Sigma_t^{xx}}} \end{pmatrix}^{-1} \otimes \boldsymbol{I}_d && (67)\\ &= \begin{pmatrix} \frac{1}{\sqrt{\Sigma_t^{xx}}} & \frac{-\Sigma_t^{xz}}{\sqrt{\Sigma_t^{xx}}\sqrt{\Sigma_t^{xx}\Sigma_t^{zz} - (\Sigma_t^{xv})^2}} \\ 0 & \sqrt{\frac{\Sigma_t^{xx}}{\Sigma_t^{xx}\Sigma_t^{vv} - (\Sigma_t^{xv})^2}} \end{pmatrix} \otimes \boldsymbol{I}_d. && (68) \end{aligned}$$

Using the above, we can compute

$$\begin{aligned} \nabla_{\mathbf{v}_t} p_t(\mathbf{u}_t \mid \cdot) &= \left[\nabla_{\mathbf{u}_t} p_t(\mathbf{u}_t \mid \cdot)\right]_{d:2d} && (69)\\ &= \left[-\boldsymbol{L}_t^{-\top}\boldsymbol{\epsilon}_{2d}\right]_{d:2d} && (70)\\ &= -\ell_t \boldsymbol{\epsilon}_{d:2d}, && (71) \end{aligned}$$

where

$$\ell_t := \sqrt{\frac{\Sigma_t^{xx}}{\Sigma_t^{xx}\Sigma_t^{vv} - (\Sigma_t^{xv})^2}}, \tag{72}$$

and $\boldsymbol{\epsilon}_{d:2d}$ denotes those (latter) d components of $\boldsymbol{\epsilon}_{2d}$ that actually affect $\nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t|\cdot)$.

Note that ℓ_t depends on the conditioning in the perturbation kernel, and therefore ℓ_t is different for DSM, which is based on $p(\mathbf{u}_t \mid \mathbf{u}_0)$, and HSM, which is based on $p(\mathbf{u}_t \mid \mathbf{x}_0)$. Therefore, we will henceforth refer to ℓ_t^{HSM} and ℓ_t^{DSM} if distinction of the two cases is necessary (otherwise we will simply refer to ℓ_t for both).

As discussed in Section 3.2, we model $\nabla_{\mathbf{v}_t} \log q_t(\mathbf{u}_t)$ as $s_{\boldsymbol{\theta}}(\mathbf{u}_t, t) = -\ell_t \alpha_{\boldsymbol{\theta}}(\mathbf{u}_t, t)$. Plugging everything back in the maximum likelihood objective function, Eq. (58) and Eq. (59), we obtain

$$\text{HSM}: \mathbb{E}_{t\sim\mathcal{U}[0,1], \mathbf{x}_0\sim p_0(\mathbf{x}_0), \mathbf{u}_t\sim p_t(\mathbf{u}_t|\mathbf{x}_0)}\left[\lambda(t)\left(\ell_t^{\text{HSM}}\right)^2 \|\boldsymbol{\epsilon}_{d:2d} - \alpha_{\boldsymbol{\theta}}(\mathbf{u}_t, t)\|_2^2\right], \tag{73}$$

$$\text{DSM}: \mathbb{E}_{t\sim\mathcal{U}[0,1], \mathbf{u}_0\sim p_0(\mathbf{u}_0), \mathbf{u}_t\sim p_t(\mathbf{u}_t|\mathbf{u}_0)}\left[\lambda(t)\left(\ell_t^{\text{DSM}}\right)^2 \|\boldsymbol{\epsilon}_{d:2d} - \alpha_{\boldsymbol{\theta}}(\mathbf{u}_t, t)\|_2^2\right], \tag{74}$$

where $\mathbf{u}_t$ is sampled via reparameterization:

$$\mathbf{u}_t = \boldsymbol{\mu}_t + \boldsymbol{L}_t\boldsymbol{\epsilon} = \boldsymbol{\mu}_t + \begin{pmatrix} L_t^{xx}\boldsymbol{\epsilon}_{0:d} \\ L_t^{xv}\boldsymbol{\epsilon}_{0:d} + L_t^{vv}\boldsymbol{\epsilon}_{d:2d} \end{pmatrix}. \tag{75}$$

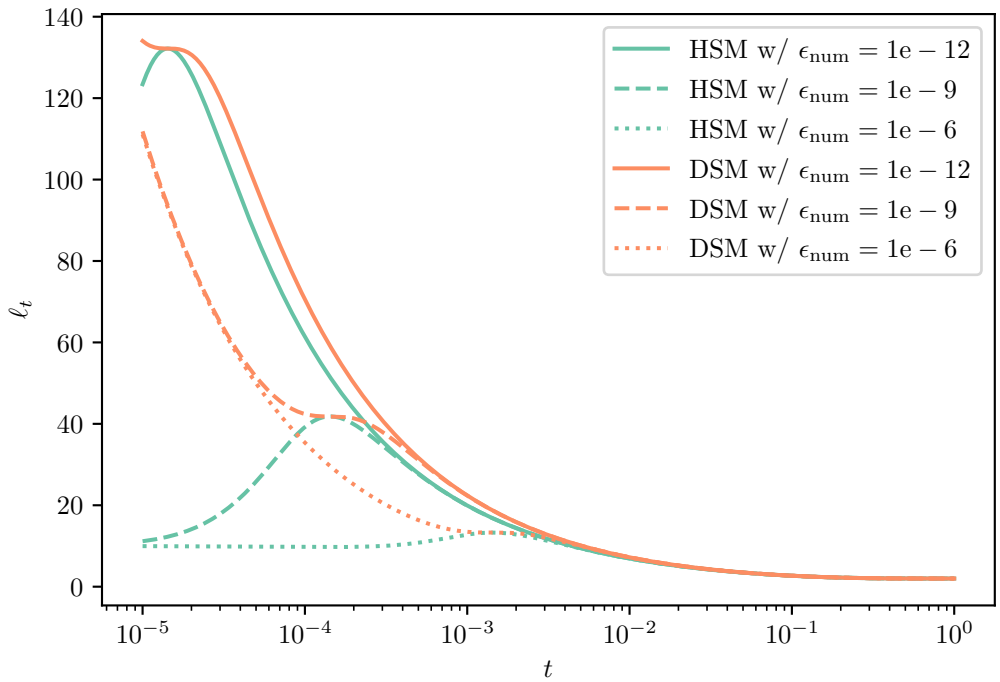


Figure 6: Comparison of ℓ_t^{HSM} (in green) and ℓ_t^{DSM} (in orange) for our main hyperparameter setting: $M = 0.04$ and $\gamma = 0.04$. In contrast to ℓ_t^{DSM}, ℓ_t^{HSM} is analytically bounded. Nevertheless, numerical computation can be unstable (even when using double precision) in which case adding numerical stabilization of $\epsilon_{\text{num}} = 1\text{e}{-9}$ suffices to make HSM work.

Note again that L_t is different for HSM and DSM.

Analogously to prior work (Ho et al., 2020; Vahdat et al., 2021; Song et al., 2021b) an objective better suited for high quality image synthesis can be obtained by "dropping the variance prefactor":

$$\text{HSM}_{\text{reweighted}} : \mathbb{E}_{t\sim\mathcal{U}[0,1],\mathbf{x}_0\sim p_0(\mathbf{x}_0),\mathbf{u}_t\sim p_t(\mathbf{u}_t|\mathbf{x}_0)}\left[\lambda_{\text{reweighted}}^{\text{HSM}}(t)\left(\ell_t^{\text{HSM}}\right)^2\left\|\boldsymbol{\epsilon}_{d:2d}-\alpha_{\boldsymbol{\theta}}(\mathbf{u}_t,t)\right\|_2^2\right] = \tag{76}$$

$$\mathbb{E}_{t\sim\mathcal{U}[0,1],\mathbf{x}_0\sim p_0(\mathbf{x}_0),\mathbf{u}_t\sim p_t(\mathbf{u}_t|\mathbf{x}_0)}\left[\left\|\boldsymbol{\epsilon}_{d:2d}-\alpha_{\boldsymbol{\theta}}(\mathbf{u}_t,t)\right\|_2^2\right], \tag{77}$$

$$\text{DSM}_{\text{reweighted}} : \mathbb{E}_{t\sim\mathcal{U}[0,1],\mathbf{u}_0\sim p_0(\mathbf{u}_0),\mathbf{u}_t\sim p_t(\mathbf{u}_t|\mathbf{u}_0)}\left[\lambda_{\text{reweighted}}^{\text{HSM}}(t)\left(\ell_t^{\text{DSM}}\right)^2\left\|\boldsymbol{\epsilon}_{d:2d}-\alpha_{\boldsymbol{\theta}}(\mathbf{u}_t,t)\right\|_2^2\right] = \tag{78}$$

$$\mathbb{E}_{t\sim\mathcal{U}[0,1],\mathbf{u}_0\sim p_0(\mathbf{u}_0),\mathbf{u}_t\sim p_t(\mathbf{u}_t|\mathbf{u}_0)}\left[\left\|\boldsymbol{\epsilon}_{d:2d}-\alpha_{\boldsymbol{\theta}}(\mathbf{u}_t,t)\right\|_2^2\right], \tag{79}$$

where we set $\lambda_{\text{reweighted}}^{\text{HSM}}(t) = \left(\ell_t^{\text{HSM}}\right)^{-2}$ and $\lambda_{\text{reweighted}}^{\text{DSM}}(t) = \left(\ell_t^{\text{DSM}}\right)^{-2}$. In App C, we show that HSM is equal to DSM (up to addition of a constant).

B.4 CLD-SPECIFIC IMPLEMENTATION DETAILS

Analytically, ℓ_t^{HSM} is bounded (in particular, $\ell_0^{\text{HSM}} = 1/\sqrt{\gamma M}$), whereas ℓ_t^{DSM} is diverging for $t \to 0$. In practice, however, we found that computation of ℓ_t^{HSM} can also be numerically unstable, even when using double precision. As is common practice for computing Cholesky decompositions, we add a numerical stabilization matrix $\epsilon_{\text{num}}\boldsymbol{I}_{2d}$ to $\boldsymbol{\Sigma}_t$ before computing ℓ_t. In Figure 6, we visualize ℓ_t^{HSM} and ℓ_t^{DSM} for different values of ϵ_{num} using our main experimental setup of $M = 0.25$ and $\gamma = 0.04$. Note that very small numerical stabilization of $\epsilon_{\text{num}} = 1\text{e}{-9}$ in combination with the use of double precision makes HSM work in practice.

B.5 LOWER BOUNDS AND PROBABILITY FLOW ODE

Given the score model $s_{\boldsymbol{\theta}}(\mathbf{u}_t, t)$, we can synthesize novel samples via simulating the reverse-time diffusion SDE, Eq. (2) in the main text. This can be achieved, for example, via our novel SSCS, Euler-Maruyama, or methods such as GGF (Jolicoeur-Martineau et al., 2021a). However, Song et al. (2021b;c) have shown that a corresponding ordinary differential equation can be defined that generates samples from the same distribution, in case $s_{\boldsymbol{\theta}}(\mathbf{u}_t, t)$ models the ground truth scores perfectly. This ODE is:

$$d\bar{\mathbf{u}}_t = \left[-\boldsymbol{f}(\bar{\mathbf{u}}_t, T-t) + \frac{1}{2}\boldsymbol{G}(\bar{\mathbf{u}}_t, T-t)\boldsymbol{G}(\bar{\mathbf{u}}_t, T-t)^\top \nabla_{\bar{\mathbf{u}}_t} \log p_{T-t}(\bar{\mathbf{u}}_t)\right] dt \tag{80}$$

This ODE is often referred to as the probability flow ODE. We can use it to generate novel data by sampling the prior and solving this ODE, like previous works (Song et al., 2021c). Note that in practice $s_{\boldsymbol{\theta}}(\mathbf{u}_t, t)$ won't be a perfect model, though, such that the generative models defined by simulating the reverse-time SDE and the probability flow ODE are not exactly equivalent (Song et al., 2021b). Nevertheless, they are very closely connected and it has been shown that their performance is usually very similar or almost the same, when we have learnt a good $s_{\boldsymbol{\theta}}(\mathbf{u}_t, t)$. In addition to sampling the generative SDE in our paper, we also sample from our CLD-based SGMs via this probability flow approach.

With the definition of our CLD, the ODE becomes:

$$\begin{pmatrix} d\bar{\mathbf{x}}_t \\ d\bar{\mathbf{v}}_t \end{pmatrix} = \underbrace{\begin{pmatrix} -M^{-1}\bar{\mathbf{v}}_t \\ \bar{\mathbf{x}}_t \end{pmatrix} \beta dt}_{A'_H} + \underbrace{\begin{pmatrix} 0 \\ \Gamma \mathbf{s}(\bar{\mathbf{u}}_t, T-t) \end{pmatrix} \beta dt}_{S'} \tag{81}$$

Notice the interesting form of this probability ODE for CLD: It corresponds to Hamiltonian dynamics (A'_H) plus the score function term S'. Compared to the generative SDE (Sec. 3.3), the Ornstein-Uhlenbeck term disappears. Generally, symplectic integrators are best suited for integrating Hamiltonian systems (Neal, 2011; Tuckerman, 2010; Leimkuhler & Reich, 2005). However, our ODE is not perfectly Hamiltonian, due to the score term, and modern non-symplectic methods, such as the higher-order adaptive-step size Runge-Kutta 4(5) integrator (Dormand & Prince, 1980), which we use in practice to solve the probability flow ODE, can also accurately simulate Hamiltonian systems over limited time horizons.

Importantly, the ODE formulation also allows us to estimate the log-likelihood of given test data, as it essentially defines a continuous Normalizing flow (Chen et al., 2018; Grathwohl et al., 2019), that we can easily run in either direction. However, in CLD the input into this ODE is not just the data $\mathbf{x}_0$, but also the velocity variable $\mathbf{v}_0$. In this case, we can still calculate a lower bound on the log-likelihood:

$$\log p(\mathbf{x}_0) = \log \left(\int p(\mathbf{x}_0, \mathbf{v}_0) d\mathbf{v}_0 \right) \tag{82}$$

$$= \log \left(\int p(\mathbf{v}_0) \frac{p(\mathbf{x}_0, \mathbf{v}_0)}{p(\mathbf{v}_0)} d\mathbf{v}_0 \right) \tag{83}$$

$$\geq \mathbb{E}_{\mathbf{v}_0 \sim p(\mathbf{v}_0)} \left[\log p(\mathbf{x}_0, \mathbf{v}_0) - \log p(\mathbf{v}_0) \right] \tag{84}$$

$$= \mathbb{E}_{\mathbf{v}_0 \sim p(\mathbf{v}_0)} \left[\log p(\mathbf{x}_0, \mathbf{v}_0) \right] + H(p(\mathbf{v}_0)) \tag{85}$$

where $H(p(\mathbf{v}_0))$ denotes the entropy of $p(\mathbf{v}_0)$ (we have $H(p(\mathbf{v}_0)) = \frac{1}{2}\log(2\pi e \gamma M)$). We can obtain a stochastic, but unbiased estimate of $\log p(\mathbf{x}_0, \mathbf{v}_0) \approx \log p_\varepsilon(\mathbf{x}_0, \mathbf{v}_0)$ via solving the probability flow ODE with initial conditions $(\mathbf{x}_0, \mathbf{v}_0)$ and calculating a stochastic estimate of the log-determinant of the Jacobian via Hutchinson's trace estimator (and also calculating the probability of the output under the prior), as done in Normalizing flows (Chen et al., 2018; Grathwohl et al., 2019) and previous works on SGMs (Song et al., 2021c;b). In the main paper, we report the negative of Eq. (85) as our upper bound on the negative log-likelihood (NLL).

Note that this bound can be potentially quite loose. In principle, it would be desirable to perform an importance-weighted estimation of the log-likelihood, as in importance-weighted autoencoders (Burda et al., 2015), using multiple samples from the velocity distribution. However, this isn't possible, as we only have access to a *stochastic* estimate $\log p_\varepsilon(\mathbf{x}_0, \mathbf{v}_0)$. The problems arising

from this are discussed in detail in Appendix F of Vahdat et al. (2021). We could consider training a velocity encoder network, somewhat similar to Chen et al. (2020), to improve our bound, but we leave this for future research.

B.6 On Introducing a Hamiltonian Component into the Diffusion

Here, we provide additional high-level intuitions and motivations about adding the Hamiltonian component to the diffusion process, as is done in our CLD.

Let us recall how the data distribution evolves in the forward diffusion process of SGMs: The role of the diffusion is to bring the initial non-equilibrium state quickly towards the equilibrium or prior distribution. Suppose for a moment, we could do so with "pure" Hamiltonian dynamics (no noise injection). In that case, we could generate data from the backward model without learning a score or neural network at all, because Hamiltonian dynamics is analytically invertible (flipping the sign of the velocity, we can just integrate backwards in reverse time direction). Now, this is not possible in practice, since Hamiltonian dynamics alone usually cannot convert the non-equilibrium distribution to the prior distribution. Nevertheless, Hamiltonian dynamics essentially achieves a certain amount of mixing on its own; moreover, since it is deterministic and analytically invertible, this mixing comes at no cost in the sense that we do not have to learn a complex score function to invert the Hamiltonian dynamics. Our thought experiment clearly shows that we should strive for a diffusion process that behaves as deterministically (meaning that deterministic implies easily invertible) as possible with as little noise injection as possible. And this is exactly what is achieved by adding the Hamiltonian component in the overall diffusion process. In fact, recall that it is the diffusion coefficient $\boldsymbol{G}$ of the forward SDE that ultimately scales the score function term of the backward generative SDE (and it is the score function that is hard to approximate with complex neural nets). Therefore, in other words, relying more on a deterministic Hamiltonian component for enhanced mixing (mixing just like in MCMC in that it brings us quickly towards the target distribution, in our case the prior) and less on pure noise injection will lead to a nicer generative SDE that relies less on a score function that requires complex and approximate neural network-based modeling, but more on a simple and analytical Hamiltonian component. Such a SDE could then be solved easier with an appropriate integrator (like our SSCS). In the end, we believe that this is the reason why our networks are "smoother" and why given the same network capacity and limited compute budgets we essentially outperform all previous results in the literature (on CIFAR-10).

We would also like to offer a second perspective, inspired by the MCMC literature. In MCMC, "mixing" helps to quickly traverse the high probability parts of the target distribution and, if an MCMC chain is initialized far from the high probability manifold, to quickly converge to this manifold. However, this is precisely the situation we are in with the forward diffusion process of SGMs: The system is initialized in a far-from-equilibrium state (the data distribution) and we need to traverse the space as efficiently as possible to converge to the equilibrium distribution, this is, the prior. Without efficient mixing, it takes longer to converge to the prior, which also implies a longer generation path in the reverse direction—which intuitively corresponds to a harder problem. Therefore, we believe that ideas from the MCMC literature that accelerate mixing and traversal of state space may be beneficial also for the diffusions in SGMs.

C HSM: Hybrid Score Matching

We begin by recalling our maximum likelihood objective function from App. B.3:

$$\mathbb{E}_{t\sim\mathcal{U}[0,1]}\left[\lambda(t)\mathbb{E}_{\mathbf{u}_t\sim p_t(\mathbf{u})}[\|\nabla_{\mathbf{v}_t}\log p_t(\mathbf{u}_t)-s_{\boldsymbol{\theta}}(\mathbf{u},t)\|_2^2]\right], \tag{86}$$

where $s_{\boldsymbol{\theta}}(\mathbf{u},t)$ is our score model. In the following, we dissect the "score matching" part of the above objective:

$$\mathcal{L}_{\mathrm{SM}} \coloneqq \mathbb{E}_{\mathbf{u}_t\sim p_t(\mathbf{u}_t)}\left[\|\nabla_{\mathbf{v}_t}\log p_t(\mathbf{u}_t)-s_{\boldsymbol{\theta}}(\mathbf{u},t)\|_2^2\right] \tag{87}$$

$$= \mathbb{E}_{\mathbf{u}_t\sim p_t(\mathbf{u}_t)}\|s_{\boldsymbol{\theta}}(\mathbf{u}_t,t)\|_2^2 - 2S(\boldsymbol{\theta}) + C_2. \tag{88}$$

where $C_2 \coloneqq \mathbb{E}_{\mathbf{u}_t\sim p_t(\mathbf{u}_t)}\left[\|\nabla_{\mathbf{v}_t}\log p_t(\mathbf{u}_t)\|_2^2\right]$ and $S(\boldsymbol{\theta})$ is the cross term discussed below. Following Vincent (2011), we can rewrite $\mathcal{L}_{\mathrm{SM}}$ as an equivalent (up to addition of a time dependent

constant) denoising score matching objective $\mathcal{L}_{\text{DSM}}$:

$$\mathcal{L}_{\text{DSM}} \coloneqq \mathbb{E}_{\mathbf{u}_0 \sim p(\mathbf{u}_0), \mathbf{u}_t \sim p_t(\mathbf{u}_t \mid \mathbf{u}_0)} \|\nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t \mid \mathbf{u}_0) - s_{\boldsymbol{\theta}}(\mathbf{u}_t, t)\|_2^2 \tag{89}$$

$$= \mathcal{L}_{\text{SM}} + C_3 - C_2, \tag{90}$$

where $C_3 \coloneqq \mathbb{E}_{\mathbf{u}_0 \sim p(\mathbf{u}_0), \mathbf{u}_t \sim p_t(\mathbf{u}_t \mid \mathbf{u}_0)} \left[\|\nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t \mid \mathbf{u}_0)\|_2^2 \right]$. Something that might not necessarily be quite obvious is that there is no fundamental need to "denoise" with the distribution $p(\mathbf{u}_0)$.

"Denoising" only with the data distribution $p(\mathbf{x}_0)$ and marginalizing over the entire initial velocity distribution $p(\mathbf{v}_0)$ results in

$$\mathcal{L}_{\text{HSM}} \coloneqq \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0), \mathbf{u}_t \sim p_t(\mathbf{u}_t \mid \mathbf{x}_0)} \|\nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t \mid \mathbf{x}_0) - s_{\boldsymbol{\theta}}(\mathbf{u}_t, t)\|_2^2 \tag{91}$$

$$= \mathbb{E}_{\mathbf{u}_t \sim p_t(\mathbf{u}_t)} \|s_{\boldsymbol{\theta}}(\mathbf{u}_t, t)\|_2^2 - 2\mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0), \mathbf{u}_t \sim p_t(\mathbf{u}_t \mid \mathbf{x}_0)} [\langle \nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t \mid \mathbf{x}_0), s_{\boldsymbol{\theta}}(\mathbf{u}_t, t) \rangle] + C_4, \tag{92}$$

where $C_4 \coloneqq \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0), \mathbf{u}_t \sim p_t(\mathbf{u}_t \mid \mathbf{x}_0)} \left[\|\nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t \mid \mathbf{x}_0)\|_2^2 \right]$. In our case, this makes sense since $p(\mathbf{v}_0)$ is Normal, and therefore (as shown in App B.1), the perturbation kernel $p_t(\mathbf{u}_t \mid \mathbf{x}_0)$ is still Normal.

In the following, only for completeness, we redo the derivation of Vincent (2011) and show that $\mathcal{L}_{\text{SM}}$ is equivalent to $\mathcal{L}_{\text{HSM}}$ (up to addition of a constant). Starting from $S(\boldsymbol{\theta})$, we have

$$S(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{u}_t \sim p_t(\mathbf{u}_t)} \langle \nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t), s_{\boldsymbol{\theta}}(\mathbf{u}_t, t) \rangle \tag{93}$$

$$= \int_{\mathbf{u}_t} p_t(\mathbf{u}_t) \langle \nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t), s_{\boldsymbol{\theta}}(\mathbf{u}_t, t) \rangle \, d\mathbf{u}_t \tag{94}$$

$$= \int_{\mathbf{u}_t} \langle \nabla_{\mathbf{v}_t} p_t(\mathbf{u}_t), s_{\boldsymbol{\theta}}(\mathbf{u}_t, t) \rangle \, d\mathbf{u}_t \tag{95}$$

$$= \int_{\mathbf{u}_t} \left\langle \nabla_{\mathbf{v}_t} \int_{\mathbf{x}_0} p_t(\mathbf{u}_t \mid \mathbf{x}_0) p_0(\mathbf{x}_0) \, d\mathbf{x}_0, s_{\boldsymbol{\theta}}(\mathbf{u}_t, t) \right\rangle d\mathbf{u}_t \tag{96}$$

$$= \int_{\mathbf{u}_t} \left\langle \int_{\mathbf{x}_0} p_t(\mathbf{u}_t \mid \mathbf{x}_0) p_0(\mathbf{x}_0) \nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t \mid \mathbf{x}_0) \, d\mathbf{x}_0, s_{\boldsymbol{\theta}}(\mathbf{u}_t, t) \right\rangle d\mathbf{u}_t \tag{97}$$

$$= \int_{\mathbf{u}_t} \int_{\mathbf{x}_0} p_t(\mathbf{u}_t \mid \mathbf{x}_0) p_0(\mathbf{x}_0) \langle \nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t \mid \mathbf{x}_0), s_{\boldsymbol{\theta}}(\mathbf{u}_t, t) \rangle \, d\mathbf{x}_0 \, d\mathbf{u}_t \tag{98}$$

$$= \mathbb{E}_{\mathbf{x}_0 \sim p_0(\mathbf{x}_0), \mathbf{u}_t \sim p(\mathbf{u}_t \mid \mathbf{x}_0)} \left[\langle \nabla_{\mathbf{v}_t} \log p_t(\mathbf{u}_t \mid \mathbf{x}_0), s_{\boldsymbol{\theta}}(\mathbf{u}_t, t) \rangle \right]. \tag{99}$$

where $\langle \cdot, \cdot \rangle$ donates the inner product (notation chosen to be consistent with Vincent (2011)). Hence, we have that

$$\mathcal{L}_{\text{HSM}} = \mathcal{L}_{\text{SM}} + C_4 - C_2. \tag{100}$$

This further implies that

$$\mathcal{L}_{\text{HSM}} = \mathcal{L}_{\text{DSM}} + C_4 - C_3. \tag{101}$$

Using the analysis from App B.1, we realize that C_3 and C_4 can be simplified to $d \left(\ell_t^{\text{DSM}}\right)^2$ and $d \left(\ell_t^{\text{HSM}}\right)^2$. Here, we used the fact that the expected squared norm of a multivariate standard Normal random variable is equal to its dimension, i.e., $\mathbb{E}_{\varepsilon \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)} \|\varepsilon\|_2^2 = d$. This analysis then simplifies Equation (101) to

$$\mathcal{L}_{\text{HSM}} = \mathcal{L}_{\text{DSM}} + d \left(\left(\ell_t^{\text{DSM}}\right)^2 - \left(\ell_t^{\text{HSM}}\right)^2 \right). \tag{102}$$

Using this relation, we can also find a connection between the maximum likelihood objective functions. In particular, we have

$$\text{HSM} = \mathbb{E}_{t \sim \mathcal{U}[0,1]} \left[\lambda(t) \mathcal{L}_{\text{HSM}} \right] \tag{103}$$

$$= \mathbb{E}_{t \sim \mathcal{U}[0,1]} \left[\lambda(t) \mathcal{L}_{\text{DSM}} \right] + d \, \mathbb{E}_{t \sim \mathcal{U}[0,1]} \left[\lambda(t) \left(\left(\ell_t^{\text{DSM}}\right)^2 - \left(\ell_t^{\text{HSM}}\right)^2 \right) \right] \tag{104}$$

$$= \text{DSM} + d\Gamma \, \mathbb{E}_{t \in \mathcal{U}[0,1]} \left[\beta(t) \left(\left(\ell_t^{\text{DSM}}\right)^2 - \left(\ell_t^{\text{HSM}}\right)^2 \right) \right], \tag{105}$$

where HSM and DSM are defined in Equation (73) and in Equation (74), respectively. Furthermore, we have the following relation for our reweighted objectives

$$\begin{aligned}
\mathrm{HSM}_{\mathrm{reweighted}} &= \mathbb{E}_{t\sim\mathcal{U}[0,1]}\left[\lambda^{\mathrm{HSM}}_{\mathrm{reweighted}}(t)\mathcal{L}_{\mathrm{HSM}}\right] && (106)\\
&= \mathbb{E}_{t\sim\mathcal{U}[0,1]}\left[\left(\ell_t^{\mathrm{HSM}}\right)^{-2}\mathcal{L}_{\mathrm{HSM}}\right] && (107)\\
&= \mathbb{E}_{t\sim\mathcal{U}[0,1]}\left[\left(\ell_t^{\mathrm{HSM}}\right)^{-2}\mathcal{L}_{\mathrm{DSM}}\right] + d\,\mathbb{E}_{t\sim\mathcal{U}[0,1]}\left[\left(\ell_t^{\mathrm{HSM}}\right)^{-2}\left(\left(\ell_t^{\mathrm{DSM}}\right)^2-\left(\ell_t^{\mathrm{HSM}}\right)^2\right)\right] && (108)\\
&= \mathbb{E}_{t\sim\mathcal{U}[0,1]}\left[\left(\ell_t^{\mathrm{HSM}}\right)^{-2}\mathcal{L}_{\mathrm{DSM}}\right] + d\,\mathbb{E}_{t\sim\mathcal{U}[0,1]}\left[\left(\ell_t^{\mathrm{HSM}}\right)^{-2}\left(\ell_t^{\mathrm{DSM}}\right)^2-1\right]. && (109)
\end{aligned}$$

C.1 Gradient Variance Reduction via HSM

Above, we derived that $\mathcal{L}_{\mathrm{HSM}} = \mathcal{L}_{\mathrm{DSM}} + C$, so one might wonder why we advocate for HSM over DSM. In practice, we never solve expectations analytically but rather approximate them using Monte Carlo estimates. In the remainder of this section, we will show that in practice (Monte Carlo) gradients based on HSM have lower variance than those based on DSM.

From Eq. (103) and Eq. (104), we have

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\mathrm{HSM} &= \mathbb{E}_{t\in\mathcal{U}[0,1]}\left[\lambda(t)\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{HSM}}\right] && (110)\\
&= \mathbb{E}_{t\in\mathcal{U}[0,1]}\left[\lambda(t)\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{DSM}}\right], && (111)
\end{aligned}$$

where $\boldsymbol{\theta}$ are the learnable parameters of the neural network. Instead of comparing the above expectations directly, we instead compare $\lambda(t)\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{HSM}}$ with $\lambda(t)\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{DSM}}$ for $t \in [0,1]$ at discretized time values (as is done in practice). Replacing $\mathcal{L}_{\mathrm{HSM}}$ and $\mathcal{L}_{\mathrm{DSM}}$ with a single Monte Carlo estimate (as is used in practice), we have

$$\begin{aligned}
\lambda(t)\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{HSM}} &\approx \lambda(t)\nabla_{\boldsymbol{\theta}}s_\theta(\mathbf{u}_t,t)\nabla_{s_\theta(\mathbf{u}_t,t)}\|\nabla_{\mathbf{v}_t}\log p_t(\mathbf{u}_t\mid\mathbf{x}_0)-s_{\boldsymbol{\theta}}(\mathbf{u}_t,t)\|_2^2, && (112)\\
&\mathbf{x}_0\sim p(\mathbf{x}_0), \mathbf{u}_t\sim p_t(\mathbf{u}_t\mid\mathbf{x}_0), && (113)\\
\lambda(t)\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{DSM}} &\approx \lambda(t)\nabla_{\boldsymbol{\theta}}s_\theta(\mathbf{u}_t,t)\nabla_{s_\theta(\mathbf{u}_t,t)}\|\nabla_{\mathbf{v}_t}\log p_t(\mathbf{u}_t\mid\mathbf{u}_0)-s_{\boldsymbol{\theta}}(\mathbf{u}_t,t)\|_2^2, && (114)\\
&\mathbf{u}_0\sim p(\mathbf{u}_0), \mathbf{u}_t\sim p_t(\mathbf{u}_t\mid\mathbf{u}_0). && (115)
\end{aligned}$$

where we applied the chain-rule. In both Eq. (112) and Eq. (114), $\mathbf{u}_t$ is sampled from the same distribution. Hence, $\lambda(t)\nabla_{\boldsymbol{\theta}}s_\theta(\mathbf{u}_t,t)$ acts as a common scaling factor, with the variance difference between HSM and DSM originating from the squared norm term. Hence, we ignore $\lambda(t)\nabla_{\boldsymbol{\theta}}s_\theta(\mathbf{u}_t,t)$ and only focus our analysis on the gradient of the norm terms, which we can further simplify:

$$\text{(HSM)}\quad \frac{1}{2}\nabla_{s_\theta(\mathbf{u}_t,t)}\|\nabla_{\mathbf{v}_t}\log p_t(\mathbf{u}_t\mid\mathbf{x}_0)-s_{\boldsymbol{\theta}}(\mathbf{u}_t,t)\|_2^2 = s_{\boldsymbol{\theta}}(\mathbf{u}_t,t)-\log p_t(\mathbf{u}_t\mid\mathbf{x}_0) =: \mathcal{K}_{\mathrm{HSM}}, \tag{116}$$

and

$$\text{(DSM)}\quad \frac{1}{2}\nabla_{s_\theta(\mathbf{u}_t,t)}\|\nabla_{\mathbf{v}_t}\log p_t(\mathbf{u}_t\mid\mathbf{x}_0)-s_{\boldsymbol{\theta}}(\mathbf{u}_t,t)\|_2^2 = s_{\boldsymbol{\theta}}(\mathbf{u}_t,t)-\log p_t(\mathbf{u}_t\mid\mathbf{u}_0) =: \mathcal{K}_{\mathrm{DSM}}. \tag{117}$$

We explore this difference in a realistic setup; in particular, we evaluate $\mathcal{K}_{\mathrm{HSM}}$ and $\mathcal{K}_{\mathrm{DSM}}$ for all data points in the CIFAR-10 training set. We choose $s_{\boldsymbol{\theta}}$ to be our trained ablation CLD model (with the standard setup of $M^{-1} = \beta = 4$, see Sec. E.2.1 for model details).

We then use these samples to estimate the covariance matrices $\mathrm{Cov}_{\mathrm{HSM}}$ and $\mathrm{Cov}_{\mathrm{DSM}}$ of the random variables $\mathcal{K}_{\mathrm{HSM}}$ and $\mathcal{K}_{\mathrm{DSM}}$, respectively.

As is common practice in statistics, we consider only the trace of the estimated covariance matrices [5]. The trace of the covariance matrix (of a random variable) is also commonly referred to as the total variation (of a random variable).

We visualize our results in Fig. 7. For HSM, there is barely any visual difference in $\mathrm{Tr}(\mathrm{Cov})$ for $\gamma = 0.04$ and $\gamma = 1$. For DSM, both $\gamma = 0.04$ and $\gamma = 1$ result in very large $\mathrm{Tr}(\mathrm{Cov})$ values for

[5] Arguably, the most prominent algorithm that follows this practice is principal component analysis (PCA).

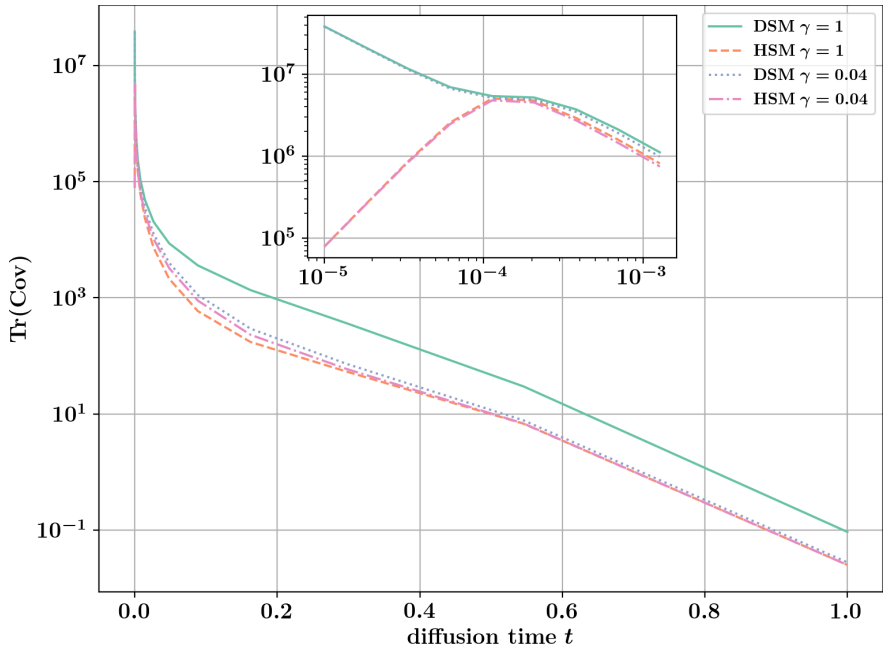


Figure 7: Traces of the estimated covariance matrices.

small t. For large t, $\mathrm{Tr}(\mathrm{Cov})$ is considerably smaller for $\gamma = 0.04$ than for $\gamma = 1$. However, in practice, we found that DSM is even unstable for small γ. Given this analysis, we believe this is due to the large gradient variance for small t. In conclusion, these results demonstrate a clear variance reduction by the HSM objective, in particular for large γ.

Note that re-introducing $\lambda(t)$ weightings would allow us to scale the $\mathrm{Tr}(\mathrm{Cov})$ curves according to the "reweighted" objective or the maximum likelihood objective. However, we believe it is most instructive to directly analyze the gradient of the norm term itself.

D Symmetric Splitting CLD Sampler (SSCS)

In this section, we present a more complete derivation and analysis of our novel Symmetric Splitting CLD Sampler (SSCS).

D.1 Background

Our derivation is inspired by methods from the statistical mechanics and molecular dynamics literature. In particular, we are leveraging symmetric splitting techniques as well as (Fokker–Planck) operator concepts. The high-level idea of symmetric splitting as well as the operator formalism are well-explained in Tuckerman (2010), in particular in Section 3.10, including simple examples. Symmetric splitting methods for stochastic dynamics in particular are discussed in detail in Leimkuhler & Matthews (2015). We also recommend Leimkuhler & Matthews (2013), which discusses splitting methods for Langevin dynamics in a concise but insightful manner.

D.2 Derivation and Analysis

Generative SDE. From Sec. 3.3, recall that our generative SDE can be written as (with $\bar{\mathbf{u}}_t = \mathbf{u}_{T-t}$, $\bar{\mathbf{x}}_t = \mathbf{x}_{T-t}$, $\bar{\mathbf{v}}_t = \mathbf{v}_{T-t}$):

$$\begin{pmatrix} d\bar{\mathbf{x}}_t \\ d\bar{\mathbf{v}}_t \end{pmatrix} = \underbrace{\begin{pmatrix} -M^{-1}\bar{\mathbf{v}}_t \\ \bar{\mathbf{x}}_t \end{pmatrix} \beta dt}_{A_H} + \underbrace{\begin{pmatrix} 0 \\ -\Gamma M^{-1}\bar{\mathbf{v}}_t \end{pmatrix} \beta dt + \begin{pmatrix} 0 \\ \sqrt{2\Gamma\beta} d\mathbf{w}_t \end{pmatrix}}_{A_O} + \underbrace{\begin{pmatrix} 0 \\ 2\Gamma \left[\mathbf{s}(\bar{\mathbf{u}}_t, T-t) + M^{-1}\bar{\mathbf{v}}_t\right] \end{pmatrix} \beta dt}_{S}$$

Fokker–Planck Equation and Fokker–Planck Operators. The evolution of the probability distribution $p_{T-t}(\bar{\mathbf{u}}_t)$ is described by the general Fokker Planck equation (Särkkä & Solin, 2019):

$$\frac{\partial p_{T-t}(\bar{\mathbf{u}}_t)}{\partial t} = -\sum_{i=1}^{2d} \frac{\partial}{\partial \bar{u}_i} \left[\mu_i(\bar{\mathbf{u}}_t, T-t) p_{T-t}(\bar{\mathbf{u}}_t)\right] + \sum_{i=1}^{2d}\sum_{j=1}^{2d} \frac{\partial^2}{\partial \bar{u}_i \partial \bar{u}_j} \left[D_{ij}(\bar{\mathbf{u}}_t, T-t) p_{T-t}(\bar{\mathbf{u}}_t)\right] \tag{118}$$

with

$$\boldsymbol{\mu}(\bar{\mathbf{u}}_t, T-t) = \begin{pmatrix} -M^{-1}\bar{\mathbf{v}}_t \\ \bar{\mathbf{x}}_t \end{pmatrix} \beta + \begin{pmatrix} 0 \\ -\Gamma M^{-1}\bar{\mathbf{v}}_t \end{pmatrix} \beta + \begin{pmatrix} 0 \\ 2\Gamma\left[\mathbf{s}(\bar{\mathbf{u}}_t, T-t) + M^{-1}\bar{\mathbf{v}}_t\right] \end{pmatrix} \beta \tag{119}$$

$$D(\bar{\mathbf{u}}_t, T-t) = \begin{pmatrix} 0 & 0 \\ 0 & 2\Gamma\beta \end{pmatrix} \otimes \boldsymbol{I}_d \tag{120}$$

For our SDE, we can write the Fokker Planck equation in short form as

$$\frac{\partial p_{T-t}(\bar{\mathbf{u}}_t)}{\partial t} = (\hat{\mathcal{L}}_A^* + \hat{\mathcal{L}}_S^*) p_{T-t}(\bar{\mathbf{u}}_t) \tag{121}$$

with the Fokker–Planck operators (defined via their action on functions of the variables $\phi(\bar{\mathbf{u}}_t)$):

$$\hat{\mathcal{L}}_A^* \phi(\bar{\mathbf{u}}_t) := -\beta M^{-1}\bar{\mathbf{v}}_t \nabla_{\bar{\mathbf{x}}_t}\phi(\bar{\mathbf{u}}_t) + \beta \bar{\mathbf{x}}_t \nabla_{\bar{\mathbf{v}}_t}\phi(\bar{\mathbf{u}}_t) - \Gamma M^{-1}\nabla_{\bar{\mathbf{v}}_t}\left[\bar{\mathbf{v}}_t \phi(\bar{\mathbf{u}}_t)\right] + 2\Gamma\beta\Delta_{\bar{\mathbf{v}}_t}\phi(\bar{\mathbf{u}}_t) \tag{122}$$

$$\hat{\mathcal{L}}_S^* \phi(\bar{\mathbf{u}}_t) := 2\Gamma\beta\left[\nabla_{\bar{\mathbf{v}}_t}\left(\left[\mathbf{s}(\bar{\mathbf{u}}_t, T-t) + M^{-1}\bar{\mathbf{v}}_t\right]\phi(\bar{\mathbf{u}}_t)\right)\right. \tag{123}$$

$$\Delta_{\bar{\mathbf{v}}_t} := \sum_{i=1}^{d}\left(\frac{\partial^2}{\partial \bar{x}_i^2} + \frac{\partial^2}{\partial \bar{v}_i^2}\right) \tag{124}$$

We are providing these formulas for transparency and completeness. We do not directly leverage them. However, working with these operators can be convenient. In particular, the operators describe the time evolution of states $\bar{\mathbf{u}}_t$ under the stochastic dynamics defined by the SDE. Given an initial state $\bar{\mathbf{u}}_0$, we can construct a formal solution to the generative SDE via (Tuckerman, 2010; Leimkuhler & Matthews, 2015):

$$\bar{\mathbf{u}}_t = e^{t\left(\hat{\mathcal{L}}_A^* + \hat{\mathcal{L}}_S^*\right)}\bar{\mathbf{u}}_0 \tag{125}$$

where the operator $e^{t\left(\hat{\mathcal{L}}_A^*\phi + \hat{\mathcal{L}}_S^*\phi\right)}$ is known as the classical propagator that propagates states $\bar{\mathbf{u}}_0$ for time t according to the dynamics defined by the combined Fokker–Planck operators $\hat{\mathcal{L}}_A^* + \hat{\mathcal{L}}_S^*$ (to avoid confusion, note that in Eq. (125) the operator $e^{t\left(\hat{\mathcal{L}}_A^*\phi + \hat{\mathcal{L}}_S^*\phi\right)}$ is applied on $\bar{\mathbf{u}}_0$ in an element-wise or "vectorized" fashion on all elements of $\bar{\mathbf{u}}_0$ in parallel). The problem with that expression is that we cannot analytically evaluate it. However, we can leverage it to design an integration method.

Symmetric Splitting Integration. Using the symmetric Trotter theorem or Strang splitting formula as well as the Baker–Campbell–Hausdorff formula (Trotter, 1959; Strang, 1968; Tuckerman, 2010), it can be shown that:

$$e^{t(\hat{\mathcal{L}}_A^* + \hat{\mathcal{L}}_S^*)} = \lim_{N\to\infty}\left[e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*} e^{\delta t \hat{\mathcal{L}}_S^*} e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*}\right]^N \approx \left[e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*} e^{\delta t \hat{\mathcal{L}}_S^*} e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*}\right]^N + \mathcal{O}(N\delta t^3) \tag{126}$$

with time step $\delta t := t/N$, where $N \in \mathbb{N}^+$. The expression suggests that instead of directly evaluating the intractable $e^{t(\hat{\mathcal{L}}_A^* + \hat{\mathcal{L}}_S^*)}$, we can discretize the dynamics over t into N pieces of step size δt, such that we only need to apply the *individual* $e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*}$ and $e^{\delta t \hat{\mathcal{L}}_S^*}$ many times one after another for small time steps δt. A finer discretization implies a smaller error (since $N{=}t/\delta t$ the error effectively scales as $\mathcal{O}(\delta t^2)$ for fixed t). Hence, this implies an integration method. The general idea of such splitting schemes is to split an initially intractable propagator into separate terms, each of which is analytically tractable. In that case, the overall integration error for many steps is only due to the splitting scheme error[6], but not due to the evaluation of the individual updates. Such techniques are,

[6]In principle, the error of the splitting scheme is defined more specifically by the commutator of the non-commuting Fokker–Planck operators. See, for example Leimkuhler & Matthews (2013; 2015); Tuckerman (2010).

for example, popular in molecular dynamics to develop symplectic integrators as well as accurate samplers (Tuckerman et al., 1992; Tuckerman, 2010; Leimkuhler & Matthews, 2013; 2015; Bussi & Parrinello, 2007).

Analyzing the Splitting Terms. Next, we need to analyze the two individual terms:

(i) Let us first analyze $e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*}\bar{\mathbf{u}}_t$: This term describes the stochastic evolution of $\bar{\mathbf{u}}_t$ under the dynamics of a SDE like Eq. (118), but with S set to zero. However, if S is set to zero, the remaining SDE has affine drift and diffusion coefficients. In that case, if the input is Normal (or a discrete state corresponding to a Normal with 0 variance) then the distribution is Normal at all times and we can calculate the evolution analytically. In particular, we can solve the differential equations for the mean $\bar{\boldsymbol{\mu}}_{\frac{\delta t}{2}}$ and covariance $\bar{\boldsymbol{\Sigma}}_{\frac{\delta t_n}{2}}$ of the Normal (these differential equation are, for example, presented in Särkkä & Solin (2019), also see Sec. B.1. The result below can be validated by inserting it into the differential equations—similar to Sec. B.1), and obtain

$$\bar{\boldsymbol{\mu}}_{\frac{\delta t}{2}}(\bar{\mathbf{u}}_t) = \begin{pmatrix} 2\beta\frac{\delta t}{2}\Gamma^{-1}\bar{\mathbf{x}}_t - 4\beta\frac{\delta t}{2}\Gamma^{-2}\bar{\mathbf{v}}_t + \bar{\mathbf{x}}_t \\ \beta\frac{\delta t}{2}\bar{\mathbf{x}}_t - 2\beta\frac{\delta t}{2}\Gamma^{-1}\bar{\mathbf{v}}_t + \bar{\mathbf{v}}_t \end{pmatrix} e^{-2\beta\frac{\delta t}{2}\Gamma^{-1}}, \tag{127}$$

as well as

$$\bar{\boldsymbol{\Sigma}}_{\frac{\delta t}{2}} = \bar{\Sigma}\left(\frac{\delta t}{2}\right) \otimes \boldsymbol{I}_d, \tag{128}$$

$$\bar{\Sigma}\left(\frac{\delta t}{2}\right) = \begin{pmatrix} \bar{\Sigma}^{xx}(\frac{\delta t}{2}) & \bar{\Sigma}^{xv}(\frac{\delta t}{2}) \\ \bar{\Sigma}^{xz}(\frac{\delta t}{2}) & \bar{\Sigma}^{vv}(\frac{\delta t}{2}) \end{pmatrix} e^{-4\beta\frac{\delta t}{2}\Gamma^{-1}}, \tag{129}$$

$$\bar{\Sigma}^{xx}\left(\frac{\delta t}{2}\right) = e^{4\beta\frac{\delta t}{2}\Gamma^{-1}} - 1 - 4\beta\frac{\delta t}{2}\Gamma^{-1} - 8\left(\beta\frac{\delta t}{2}\right)^2\Gamma^{-2}, \tag{130}$$

$$\bar{\Sigma}^{xv}\left(\frac{\delta t}{2}\right) = 4\beta\frac{\delta t}{2}\Gamma^{-2}, \tag{131}$$

$$\bar{\Sigma}^{vv}\left(\frac{\delta t}{2}\right) = \frac{\Gamma^2}{4}\left(e^{4\beta\frac{\delta t}{2}\Gamma^{-1}} - 1\right) + \beta\frac{\delta t}{2}\Gamma - 2\left(\beta\frac{\delta t}{2}\right)^2. \tag{132}$$

Hence, we can write the action of the the propagator $e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*}$ on a state $\bar{\mathbf{u}}_t$ as:

$$e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*}\bar{\mathbf{u}}_t \sim \mathcal{N}(\bar{\mathbf{u}}_{t+\frac{\delta t}{2}}; \bar{\boldsymbol{\mu}}_{\frac{\delta t}{2}}(\bar{\mathbf{u}}_t), \bar{\boldsymbol{\Sigma}}_{\frac{\delta t}{2}}). \tag{133}$$

(ii): Next, we need to analyze $e^{\delta t\hat{\mathcal{L}}_S^*}\bar{\mathbf{u}}_t$. Unfortunately, we cannot calculate the action of the propagator $e^{\delta t\hat{\mathcal{L}}_S^*}$ on $\bar{\mathbf{u}}_t$ analytically and we need to make an approximation. From Eq. (118), we can easily see that the propagator $e^{\delta t\hat{\mathcal{L}}_S^*}$ describes the evolution of the velocity component $\bar{\mathbf{v}}_t$ for time step δt under the ODE:

$$d\bar{\mathbf{v}}_t = 2\beta\Gamma\left[\mathbf{s}(\bar{\mathbf{u}}_t, T-t) + M^{-1}\bar{\mathbf{v}}_t\right]dt \tag{134}$$

We propose to simply solve this ODE for the step δt via a simple step of Euler's method, resulting in:

$$e^{\delta t\hat{\mathcal{L}}_S^*}\bar{\mathbf{u}}_t \approx \bar{\mathbf{u}}_t + \delta t\begin{pmatrix} 0 \\ 2\beta\Gamma\left[\mathbf{s}(\bar{\mathbf{u}}_t, T-t) + M^{-1}\bar{\mathbf{v}}_t\right] \end{pmatrix} + \mathcal{O}(\delta t^2) \tag{135}$$

$$= e^{\delta t\hat{\mathcal{L}}_S^{*\,\text{Euler}}}\bar{\mathbf{u}}_t + \mathcal{O}(\delta t^2), \tag{136}$$

with the informal definition

$$e^{\delta t\hat{\mathcal{L}}_S^{*\,\text{Euler}}}\bar{\mathbf{u}}_t := \bar{\mathbf{u}}_t + \delta t\begin{pmatrix} 0 \\ 2\beta\Gamma\left[\mathbf{s}(\bar{\mathbf{u}}_t, T-t) + M^{-1}\bar{\mathbf{v}}_t\right]. \end{pmatrix} \tag{137}$$

Error Analysis. It is now instructive to study the overall error of our proposed integrator. With the additional Euler integration in one of the splitting terms, we have

$$e^{t(\hat{\mathcal{L}}_A^* + \hat{\mathcal{L}}_S^*)} \approx \left[e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*}\left(e^{\delta t\hat{\mathcal{L}}_S^{*\,\text{Euler}}} + \mathcal{O}(\delta t^2)\right)e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*}\right]^N + \mathcal{O}(N\delta t^3) \tag{138}$$

$$= \left[e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*}\left(e^{\delta t\hat{\mathcal{L}}_S^{*\,\text{Euler}}}\right)e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*}\right]^N + N\mathcal{O}(\delta t^2) \tag{139}$$

$$= \left[e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*}\left(e^{\delta t\hat{\mathcal{L}}_S^{*\,\text{Euler}}}\right)e^{\frac{\delta t}{2}\hat{\mathcal{L}}_A^*}\right]^N + \mathcal{O}(\delta t), \tag{140}$$

where we used $N = \frac{t}{\delta t}$ and only kept the highest error terms. We see that, just like Euler's method, also our SSCS is a first-order integrator with local error $\sim\delta t^2$ and global error $\sim\delta t$. This is of course expected, considering that we used an Euler step for the S term. Nevertheless, as long as the dynamics is not dominated by the S component, our proposed integration scheme is expected to still be more accurate than EM, since we split off the analytically tractable part and only use an Euler approximation for the S term.

To this end, recall that the model only needs to learn the score of the conditional distribution $p_t(\mathbf{v}_t|\mathbf{x}_t)$, which is close to Normal for much of the diffusion, in which case the S term will indeed be small. This suggests that the generative SDE dynamics are in fact dominated by A_H and A_O in practice. From another perspective, note that (recalling that $s_{\boldsymbol{\theta}}(\mathbf{u}_t, t) = -\ell_t \alpha_{\boldsymbol{\theta}}(\mathbf{u}_t, t)$ with $\alpha_{\boldsymbol{\theta}}(\mathbf{u}_t, t) = \ell_t^{-1}\mathbf{v}_t/\Sigma_t^{vv} + \alpha'_{\boldsymbol{\theta}}(\mathbf{u}_t, t)$ from Sec. 3.2):

$$\mathbf{s}(\bar{\mathbf{u}}_t, T-t) + M^{-1}\bar{\mathbf{v}}_t = -\ell_t \alpha'_{\boldsymbol{\theta}}(\mathbf{u}_t, t) - \frac{\mathbf{v}_t}{\Sigma_t^{vv}} + M^{-1}\bar{\mathbf{v}}_t \tag{141}$$

$$= -\ell_t \alpha'_{\boldsymbol{\theta}}(\mathbf{u}_t, t) + \bar{\mathbf{v}}_t \left(\frac{1}{M} - \frac{1}{\Sigma_t^{vv}}\right). \tag{142}$$

For large parts of the diffusion, Σ_t^{vv} is indeed close to $1/M$, such that the $\bar{\mathbf{v}}_t$ term is very small (this cancellation is the reason why we pulled the $M^{-1}\bar{\mathbf{v}}_t$ term into the S component). In Sec. 3, we have also seen that our neural network component can be much smoother than that of previous SGMs. Overall, this suggests that the error of SSCS indeed might be smaller than the error we would obtain when applying a naive Euler–Maruyama integrator to the full generative SDE. Our positive experimental results in Sec. 5.2 validate that. Only in the limit for very small steps, both our SSCS and EM make only very small errors and are expected to perform equally well, which is exactly what we observe in our experiments. Our SSCS turns out to be well-suited for integrating the generative SDE with relatively few synthesis steps.

Note that in practice we do not use SSCS to integrate all the way from $t = 0$ to $t = T$, but only up to $t = T - \epsilon$, and perform a denoising step, similar to previous works (Jolicoeur-Martineau et al., 2021a; Song et al., 2021c). It is worth noting that our SSCS scheme would also be applicable when we used time-dependent $\beta(t)$, as in our more general derivation of the CLD perturbation kernel in App. B. However, since we only used constant β in the main paper, we also presented SSCS in that way.

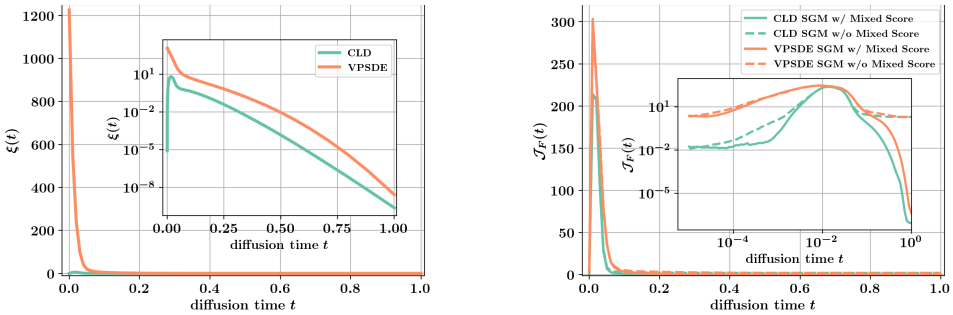
A promising direction for future research would be to extend our SSCS towards adaptive step sizes and to use techniques to facilitate a higher-order integration, while still leveraging the unique advantages of SSCS.

SSCS Algorithm. Finally, we summarize SSCS in terms of a concise algorithm:

Algorithm 1 *Symmetric Splitting CLD Sampler (SSCS)*

Input: Score function $\mathbf{s}_{\boldsymbol{\theta}}(\bar{\mathbf{u}}_t, T-t)$, CLD parameters Γ, β, $M = \Gamma^2/4$, number of sampling steps N, step sizes $\{\delta t_n \geq 0\}_{n=0}^{N-1}$ chosen such that $\epsilon := T - \sum_{n=0}^{N-1} \delta t_n \geq 0$ (stepsizes can vary, for example in QS).
Output: Synthesized model sample $\bar{\mathbf{x}}'_N$, along with a velocity sample $\bar{\mathbf{v}}'_N$

$\bar{\mathbf{x}}_0 \sim \mathcal{N}(\bar{\mathbf{x}}_0; \mathbf{0}_d, \boldsymbol{I}_d)$, $\bar{\mathbf{v}}_0 \sim \mathcal{N}(\bar{\mathbf{v}}_0; \mathbf{0}_d, M\boldsymbol{I}_d)$, $\bar{\mathbf{u}}_0 = (\bar{\mathbf{x}}_0, \bar{\mathbf{v}}_0)$ ▷ Draw initial prior samples from $p_{\text{EQ}}(\mathbf{u})$
$t = 0$ ▷ Initialize time
for $n = 0$ **to** $N-1$ **do**
  $\bar{\mathbf{u}}_{n+\frac{1}{2}} \sim \mathcal{N}(\bar{\mathbf{u}}_{n+\frac{1}{2}}; \bar{\boldsymbol{\mu}}_{\frac{\delta t_n}{2}}(\bar{\mathbf{u}}_n), \bar{\boldsymbol{\Sigma}}_{\frac{\delta t_n}{2}})$ ▷ First half-step: apply $\exp\{\frac{\delta t_n}{2}\hat{\mathcal{L}}_A^*\}$ on $\bar{\mathbf{u}}_n$
  $\bar{\mathbf{u}}'_{n+\frac{1}{2}} \leftarrow \bar{\mathbf{u}}_{n+\frac{1}{2}} + \delta t_n \begin{pmatrix} 0 \\ 2\beta\Gamma\left[\mathbf{s}(\bar{\mathbf{u}}_t, T-t) + M^{-1}\bar{\mathbf{v}}_t\right] \end{pmatrix}$ ▷ Full step: apply $\exp\{\delta t_n \hat{\mathcal{L}}_S^*\}$ on $\bar{\mathbf{u}}_{n+\frac{1}{2}}$
  $\bar{\mathbf{u}}_{n+1} \sim \mathcal{N}(\bar{\mathbf{u}}_{n+1}; \bar{\boldsymbol{\mu}}_{\frac{\delta t_n}{2}}(\bar{\mathbf{u}}'_{n+\frac{1}{2}}), \bar{\boldsymbol{\Sigma}}_{\frac{\delta t_n}{2}})$ ▷ Second half-step: apply $\exp\{\frac{\delta t_n}{2}\hat{\mathcal{L}}_A^*\}$ on $\bar{\mathbf{u}}'_{n+\frac{1}{2}}$
  $t \leftarrow t + \delta t_n$ ▷ Update time
end for
$\bar{\mathbf{u}}'_N \leftarrow \bar{\mathbf{u}}_N - \epsilon \begin{pmatrix} 0 & \beta M^{-1} \\ -\beta & -\Gamma\beta M^{-1} \end{pmatrix} \otimes \boldsymbol{I}_d\, \bar{\mathbf{u}}_N + \epsilon \begin{pmatrix} 0 \\ 2\beta\Gamma\mathbf{s}(\bar{\mathbf{u}}_t, \epsilon) \end{pmatrix}$ ▷ Denoising
$(\bar{\mathbf{x}}'_N, \bar{\mathbf{v}}'_N) = \bar{\mathbf{u}}'_N$ ▷ Extract output data and velocity samples



(a) Difference $\xi(t)$ (via $L2$ norm) between score of diffused data and score of Normal distribution.

(b) Frobenius norm of Jacobian $\mathcal{J}_F(t)$ of the neural network defining the score function for different t.

Figure 8: Toy experiments for mixture of Normals dataset.

Note that the algorithm uses the expressions in Eqs. (127) and (128) for $\bar{\boldsymbol{\mu}}_t$ and $\bar{\boldsymbol{\Sigma}}_t$. Furthermore, in practice in the denoising step at the end, we usually only update the $\bar{\mathbf{x}}'_N$ component of $\bar{\mathbf{u}}'_N$, since we are only interested in the data sample. This saves us the final neural network call during denoising, which only affects the $\bar{\mathbf{v}}'_N$ component. However, we did write the algorithm in the general way, which also allows to correctly generate the velocity sample $\bar{\mathbf{v}}'_N$.

Also note that we could combine the second half-step from one iteration of SSCS with the first half-step from the next iteration of SSCS. This is commonly done in the Leapfrog integrator, which follows a similar structure as our SSCS. However, it is not important in our case, as the only computational costly operation is in the center full step, which involves the neural network evaluation. The first and last half-step come at virtually no computational cost.

E Implementation and Experiment Details

E.1 Score and Jacobian experiment

In this section, we provide details for the experiments presented in Sec. 3.1. For both experiments, we consider a two-dimensional simple mixture of Normals of the form

$$p_{\text{data}}(\mathbf{x}) = \sum_{k=1}^{9} \frac{1}{9} p^{(k)}(\mathbf{x}), \tag{143}$$

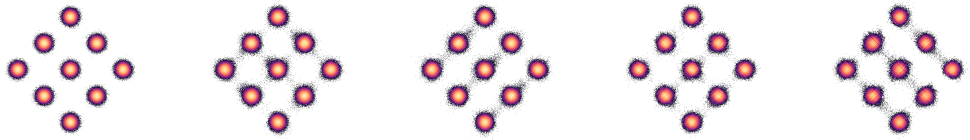
where $p^{(k)}(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k; 0.04^2 \boldsymbol{I}_2)$ and

$$\boldsymbol{\mu}_1 = \begin{pmatrix} -a \\ 0 \end{pmatrix}, \quad \boldsymbol{\mu}_2 = \begin{pmatrix} -a/2 \\ a/2 \end{pmatrix}, \quad \boldsymbol{\mu}_3 = \begin{pmatrix} 0 \\ a \end{pmatrix},$$
$$\boldsymbol{\mu}_4 = \begin{pmatrix} -a/2 \\ -a/2 \end{pmatrix}, \quad \boldsymbol{\mu}_5 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{\mu}_6 = \begin{pmatrix} a/2 \\ a/2 \end{pmatrix},$$
$$\boldsymbol{\mu}_7 = \begin{pmatrix} 0 \\ -a \end{pmatrix}, \quad \boldsymbol{\mu}_8 = \begin{pmatrix} a/2 \\ -a/2 \end{pmatrix}, \quad \boldsymbol{\mu}_9 = \begin{pmatrix} a \\ 0 \end{pmatrix},$$

and $a = 2^{-\frac{1}{2}}$. The choice of this data distribution is not arbitrary. In fact, mixture of Normal distributions are diffused by simply diffusing the components, i.e., setting $p_0(\mathbf{x}_0) = p_{\text{data}}(\mathbf{x})$, we have

$$p_t(\mathbf{x}_t) = \sum_{k=1}^{9} \frac{1}{9} p_t^{(k)}(\mathbf{x}_t), \tag{144}$$

where $p_t^{(k)}$ are the diffused components (analogously for CLD with velocity augmentation). This means that for both CLD as well as VPSDE Song et al. (2021c) we can diffuse $p_{\text{data}}(\mathbf{x})$ analytically. This allows us to perform interesting analyses that would be possible when working, for example, with image data. We visualize the data distribution in Fig. 9.



(a) Data p_{data} (b) CLD w/ MS (c) CLD w/o MS (d) VPSDE w/ MS (e) VPSDE w/o MS

Figure 9: Mixture of Normals: data and trained models.

Score experiment: We empirically verify the reduced complexity of the score of $p_t(\mathbf{v}_t|\mathbf{x}_t)$, which is learned in CLD, compared to the score of $p_t(\mathbf{x}_t)$, which is learned in VPSDE. To avoid scaling issues between VPSDE and CLD, we chose $M = \gamma = 1$ for CLD in this experiment; this results in an equilibrium distribution of $\mathcal{N}(0, 1)$ (for both data and velocity component, which are independent at equilibrium), which is the same as the equilibrium distribution of VPSDE. We then measure the difference of the respective scores at time t and the equilibrium (or prior) scores, i.e. (recall that the score of a Normal distribution $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; 0, 1)$ is simply $\nabla_{\mathbf{x}} \log p(\mathbf{x}) = -\mathbf{x}$),

$$\xi^{\text{VPSDE}}(t) \coloneqq \mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t)} \|\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) + \mathbf{x}_t\|_2^2, \tag{145}$$

$$\xi^{\text{CLD}}(t) \coloneqq \mathbb{E}_{\mathbf{u}_t \sim p(\mathbf{u}_t)} \|\nabla_{\mathbf{v}_t} \log p_t(\mathbf{v}_t \mid \mathbf{x}_t) + \mathbf{v}_t\|_2^2. \tag{146}$$

The expectations are approximated using 10^5 samples from $p(\mathbf{x}_t)$ and $p(\mathbf{u}_t)$ for VPSDE and CLD, respectively. As can be seen in Fig. 8a, $\xi^{\text{CLD}}(t)$ is smaller than $\xi^{\text{VPSDE}}(t)$ for all $t \in [0, T]$. The difference is particularly striking for small time values t. Other previous SDEs, such as the VESDE, sub-VPSDE, etc., are expected to behave similarly. This result implies that the ground truth scores that need to be learnt in CLD are closer to Normal scores than the ground truth scores in previous SDEs like the VPSDE. Since the score of a Normal is very simple—and indeed directly leveraged in our mixed score formulation—we would intuitively expect that the CLD training task is easier.

Complexity experiment: Therefore, to understand the above observations in terms of learning neural networks, we train a small ResNet architecture (less than 100k parameters) for each of the following four setups: both CLD and VPSDE each with and without a mixed score parameterization. The mixed score of the VPSDE simply assumes a standard normal data distribution (which is also the equilibrium distribution of VPSDE) resulting in adding $-\mathbf{x}_t$ to the score function. Formally, $-\mathbf{x}_t$ is the score of a Normal distribution with unit variance.

We train the models for 1M iterations using fresh data synthesized from p_{data} at a batch size of 512. The model and data distributions are visualized in Fig. 9. We measure the complexity of the trained neural networks using the squared Frobenius norm of the networks' Jacobians. For CLD, we have

$$\mathcal{J}_{\text{F}}^{\text{CLD}}(t) \coloneqq \mathbb{E}_{\mathbf{u}_t \sim p(\mathbf{u}_t)} \|\nabla_{\mathbf{u}_t} \alpha'_{\boldsymbol{\theta}}(\mathbf{u}_t)\|_F^2 \tag{147}$$

Similarly, for VPSDE, we compute

$$\mathcal{J}_{\text{F}}^{\text{VPSDE}}(t) \coloneqq \mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t)} \|\nabla_{\mathbf{x}_t} \alpha'_{\boldsymbol{\theta}}(\mathbf{x}_t)\|_F^2 \tag{148}$$

For both CLD and VPSDE, expectations are again approximated using 10^5 samples. As can be seen in Fig. 8b the neural network complexity is lower for CLD compared to VPSDE. A mixed score formulation helps decreasing the neural network complexity for both CLD and VPSDE. This result implies that the arguably simpler training task in CLD indeed also translates to reduced model complexity in that the neural network is smoother as measured by $\mathcal{J}_{\text{F}}(t)$. In large-scale experiments, this would mean that, given similar model capacity, a CLD-based SGM could potentially have a higher expressivity. Or, on the other hand, similar performance could be achieved with a smoother and potentially smaller model. Indeed these findings are in line with our strong results on the CIFAR-10 benchmark.

E.2 Image Modeling Experiments

We perform image modeling experiments on CIFAR-10 as well as CelebA-HQ-256. We report FID scores on CIFAR-10 for our main model in for a sweep of different solvers; see Tab. 3 and

Hyperparameter	CIFAR10 (Main)	CelebA (Qualitative)	CIFAR10 (Ablation)
Model			
EMA rate	0.9999	0.9999	0.9999
# of ResBlocks per Resolution	8	2	2
Normalization	Group Normalization	Group Normalization	Group Normalization
Scaling by σ	✗	✗	✗
Nonlinearity	Swish	Swish	Swish
Attention resolution	16	16	16
Embedding type	Fourier	Positional	Positional
Progressive	None	None	None
Progressive input	Residual	None	None
Progressive combine	Sum	Sum	Sum
Finite Impulse Response (Zhang, 2019)	✓	✗	✗
# of parameters	$\approx 108M$	$\approx 68M$	$\approx 39M$
Training			
# of iterations	800k	320k	1M
# of warmup iterations	100k	100k	100k
Optimizer	Adam	Adam	Adam
Mixed precision	✓	✓	✓
Learning rate	2e−4	1e−4	2e−4
Gradient norm clipping	1.0	1.0	1.0
Dropout	0.1	0.1	0.1
Batch size per GPU	8	4	8
# of GPUs	16	16	16
t-sampling cutoff during training	1e−5	1e−5	1e−5
SDE			
M	0.25	0.25	varies
γ	0.04	0.04	varies
β	4	4	varies
ϵ_{num}	1e−9	1e−6	1e−9

Table 6: Model architectures as well as SDE and training setup for our experiments on CIFAR10 an CelebA-HQ-256.

Tab. 2. We further present generated samples for both models in Sec. 5 using Euler–Maruyama with 2000 quadratic striding steps and Runge–Kutta 4(5) for CIFAR10 and CelebA-HQ-256, respectively. We present additional samples for variety of solver settings in App. F. All (average) NFEs for the Runge–Kutta solver are computed using a batch size of 128.

E.2.1 Training Details and Model Architectures

Our models are based on the NCSN++ and the DDPM++ architectures from Song et al. (2021c). Importantly, we changed the number of input channels from three to six to facilitate the additional velocity variables. Note that the number of additional parameters due to this change is negligible.

For fair comparison, we train our models using the same t-sampling cutoff during training as is used for VESDE and VPSDE in Song et al. (2021c). Note, however, that this is not strictly necessary for CLD as we do not have any "blow-up" of the SDE due to unbounded scores as $t \to 0$.

We summarize our three model architectures as well as our SDE and training setups in Tab. 6.

E.2.2 CIFAR10 Results for VESDE and VPSDE

The results reported for VESDE and VPSDE using the GGF sampler are taken from Jolicoeur-Martineau et al. (2021a). All other results for VESDE and VPSDE are generated by using the provided PyTorch code as well as the provided checkpoints from Song et al. (2021c).

E.2.3 Quadratic Striding

When we simulate our generative SDE numerically, using for example EM or our SSCS, we need to choose a time discretization. Given a certain NFE budget N_{NFE}, how do we choose time step sizes?

The standard approach is to use an equidistant discretization, corresponding to a set of evaluation time steps $\{\delta t_i \geq 0\}_{i=0}^{N_{\text{NFE}}-1}$ with $\delta t_i = \frac{1}{N_{\text{NFE}}} \forall i \in [0, N_{\text{NFE}} - 1]$. However, prior work (Song et al., 2021a; Kong & Ping, 2021; Watson et al., 2021) has shown that it can be beneficial to focus function evaluation on times t close to the "data". This is because the diffusion process distribution is most complex close to the data and almost perfectly Normal close to the prior. Among other techniques, these works used a useful heuristic, denoted as *quadratic striding* (QS), which discretizes the integration interval such that the evaluation times follow a quadratic schedule and the individual time steps follow a linear schedule. We also used this QS approach in our experiments.

We can formally define it as follows (assuming time interval $[0.0, 1.0]$ here for simplicity): Denote the evaluation times as τ_i (including 0.0 and 1.0) and define:

$$\tau_i = c_\tau i^2 \quad \forall i \in [0, N_{\text{NFE}}]. \tag{149}$$

Hence,

$$\delta t_i = \tau_i - \tau_{i-1} = c_\tau (2i - 1) \quad \forall i \in [1, N_{\text{NFE}}], \tag{150}$$

and $c_\tau = \frac{1}{N_{\text{NFE}}^2}$ to ensure that $\tau_{\text{NFE}} = 1.0$.

This describes the time steps as going from $t = 0$ to $t = 1$. During synthesis, however, we are going backwards. Hence, we can define our time steps as

$$\delta t_j = c_\tau \left[2N_{\text{NFE}} - 2j + 1\right] \quad \forall j \in [1, N_{\text{NFE}}], \tag{151}$$

where j now counts time steps in the other direction. Note that this can be easily adapted to general integration intervals $[\epsilon, T]$.

E.2.4 Denoising

As has been pointed out in Jolicoeur-Martineau et al. (2021b), samples that are generated with models similar to ours can contain noise that is hard to detect visually but worsens FID scores significantly.

Denoising Formulas. For fair comparison we use the same denoising setup for all experiments we conducted (including VESDE (PC/ODE) and VPSDE (EM/ODE)) except for LSGM [7]. We simulate the underlying generative ODE/SDE until the time cutoff $\varepsilon = 1\text{e}{-3}$ and then take a single denoising step of the form

$$\mathbf{u}_0 = \mathbf{u}_\varepsilon - \boldsymbol{f}(\mathbf{u}_\varepsilon, \varepsilon)\varepsilon + \boldsymbol{G}(\mathbf{u}_\varepsilon, \varepsilon)\boldsymbol{G}(\mathbf{u}_\varepsilon, \varepsilon)^\top \begin{pmatrix} \mathbf{0}_d \\ \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{u}_\varepsilon, \varepsilon) \end{pmatrix} \varepsilon. \tag{152}$$

This denoising step can be considered as an Euler–Maruyama step without noise injection. For SDEs acting on data directly (VESDE, VPSDE, etc.) the corresponds denoising formula is

$$\mathbf{x}_0 = \mathbf{x}_\varepsilon - \boldsymbol{f}(\mathbf{x}_\varepsilon, \varepsilon)\varepsilon + \boldsymbol{G}(\mathbf{x}_\varepsilon, \varepsilon)\boldsymbol{G}(\mathbf{x}_\varepsilon, \varepsilon)^\top \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_\varepsilon, \varepsilon)\varepsilon. \tag{153}$$

Influence of Denoising on Results. For SDEs acting in the data space directly, it has been reported that this denoising step is crucial to obtain good FID numbers Jolicoeur-Martineau et al. (2021b); Song et al. (2021c). When we simulate the generative probability flow ODE we found that denoising is important in order for the Runge–Kutta solver not to "blow-up" as $t \to 0$. On the other hand, when simulating CLD using our new SSCS solver, we found that denoising only slightly influences FID (see Tab 7). We believe that this might be because the neural network does not have any influence on the denoising step for CLD. Putting the drift and diffusion coefficients of CLD in the denoising formula in Eq. (152), we obtain

$$\mathbf{u}_0 = \mathbf{u}_\varepsilon - \begin{pmatrix} 0 & \beta M^{-1} \\ -\beta & -\beta\Gamma M^{-1} \end{pmatrix} \otimes \boldsymbol{I}_d \mathbf{u}_\varepsilon \varepsilon + \begin{pmatrix} \mathbf{0}_d \\ 2\Gamma\beta \end{pmatrix} s_{\boldsymbol{\theta}}(\mathbf{u}_\varepsilon, \varepsilon)\varepsilon \implies \mathbf{x}_0 = \mathbf{x}_\varepsilon - \beta M^{-1} \mathbf{v}_\varepsilon \varepsilon. \tag{154}$$

This means the denoising for CLD results in $\mathbf{u}_0$ being a linear transformation of $\mathbf{u}_\varepsilon$.

[7] Denosing has not been used in the original LSGM work (Vahdat et al., 2021) and is not needed in their case, since the output of the latent SGM lives in a smooth latent space and is further processed by a decoder.

Table 7: Influence of denoising step on FID scores (using our main CIFAR-10 model).

Sampler	Denoising	FID at n function evaluations ↓ n=50	n=500
SSCS	✓	81.1	2.30
SSCS-QS	✓	20.5	2.25
SSCS	✗	78.9	2.32
SSCS-QS	✗	28.5	2.3

E.2.5 SOLVER TOLERANCES FOR RUNGE–KUTTA 4(5)

In Tab. 2, we report FID numbers for a Runge–Kutta 4(5) solvers (Dormand & Prince, 1980) as well as the "Gotta Go Fast" solver from Jolicoeur-Martineau et al. (2021a) (see their Table 1). For simulating CLD with Runge–Kutta 4(5) we chose the solver tolerances to hit certain regimes of NFEs to facilitate comparison with VPSDE and VESDE. We obtain a mean number of function evaluations of 312 and 137 using Runge–Kutta 4(5) solver tolerances of $1\mathrm{e}{-5}$ and $1\mathrm{e}{-3}$, respectively. For VESDE, VPSDE and LSGM we used $1\mathrm{e}{-5}$ as the ODE solver tolerance.

E.2.6 ABLATIONS EXPERIMENTS

The model architecture used for all ablation experiments can be found in Tab. 6. As pointed out in Sec. 5 we found that the hyperparameters γ and M only have small effect on CIFAR-10 FID numbers. On the other hand, we found that the mixed score parameterization helps significantly in obtaining competitive FIDs.

E.2.7 LSGM-100M MODEL

Our CLD-based SGM has $\approx 108M$ parameters, while the original CIFAR-10 Latent SGM from Vahdat et al. (2021), to which we compare in Tab. 1, uses $\approx 476M$ parameters. To establish a fairer comparison between our CLD-based SGMs and LSGM (Vahdat et al., 2021), we train another smaller LSGM model with $\approx 109M$ parameters. To do this, we followed the exact setup of the "CIFAR10 (balanced)" model from LSGM (see Table 7 in Vahdat et al. (2021)), with a few minor modifications: We used a VAE backbone model with only 10 groups instead of 20, which corresponds to a reduction in parameters by a factor of 2 in the encoder and decoder networks. We also reduced the convolutional channels in the latent space SGM from 512 to 256 and reduced the number of the residual cells per scale from 8 to 4. With these modifications the resulting "LSGM-100M" uses only $\approx 109M$ parameters overall with approximately half of them in the encoder and decoder networks and the other half in the latent SGM. Other than these architecture modifications, our model is trained in exactly the same way as the bigger, original models in Vahdat et al. (2021).

For evaluation, we follow the recommended setting by Vahdat et al. (2021) and use the same Runge-Kutta 4(5) ODE solver with an error tolerance of $1\mathrm{e}{-5}$ to solve the probability flow ODE in LSGM's latent space. LSGM-100M achieves an FID of 4.60, an NLL bound of 2.96 bpd, and requires on average 131 NFE for sampling new images. We report these results in Tabs. 1 and 2 in the main text.

Note that we also tried training a model following the "CIFAR10 (best FID)" setup, but found training to be unstable (however the orignal "CIFAR10 (best FID)" model from Vahdat et al. (2021) only performs marginally better in FID than their "CIFAR10 (balanced)" model anyway). Furthermore, we also tried training another small LSGM with a similar number of parameters but with more parameters in the latent SGM and less in the encoder and decoder networks, compared to the reported LSGM-100M. However, this model performed significantly worse.

Table 8: Performance (measured in negative log-likelihood) using analytical scores for non-adaptive stepsize solvers for varying number of synthesis steps n (function evaluations).

Model	Sampler	$-\log p(x)$ at n function evaluations ↓			
		n=20	n=50	n=100	n=200
CLD	EM	60.6	9.71	0.72	-1.04
CLD	SSCS	**10.5**	**1.55**	**-1.25**	**-1.54**
VPSDE	EM	14.2	4.68	-0.35	-1.11

F Additional Experiments

F.1 Toy Experiments

F.1.1 Analytical Sampling

In order to test combinations of diffusions and numerical samplers in isolation, we consider a dataset for which we know the ground truth score function (for all t) analytically. In particular, we use the mixture of Normals introduced in App. E.1; see Fig. 9a for a visualization of the data distribution. In Fig. 10, we show samples for VPSDE (Euler–Maruyama (EM) sampler) and CLD (EM and SSCS samplers). For quantitative comparison, we also compute negative log-likelihoods for the three combinations (which can be done easily due to our access to the ground truth distribution): as can be seen in Tab. 8, for each number of steps $n \in \{20, 50, 100, 200\}$ CLD with SCSS outperforms both VPSDE and CLD with EM. As discussed in Sec. 3.3, we can see in Tab. 8 that EM is not well-suited for CLD. This is true, in particular, when using a small number of synthesis steps n (function evaluations). In Fig. 10, we see that CLD with EM leads to sampling distributions which are too broad. These results are exactly in line with the "diverging" dynamics that is observed when solving Hamiltonian dynamics with a non-symplectic integrator, such as the standard Euler method (Neal, 2011). This problematic behavior of Euler-based techniques is more pronounced when using fewer steps with larger stepsizes, which is also what we observe in our experiments. These results further motivate the use of our novel SSCS, which addresses these challenges, for sampling from our CLD-based SGMs.

F.1.2 Maximum Likelihood Training

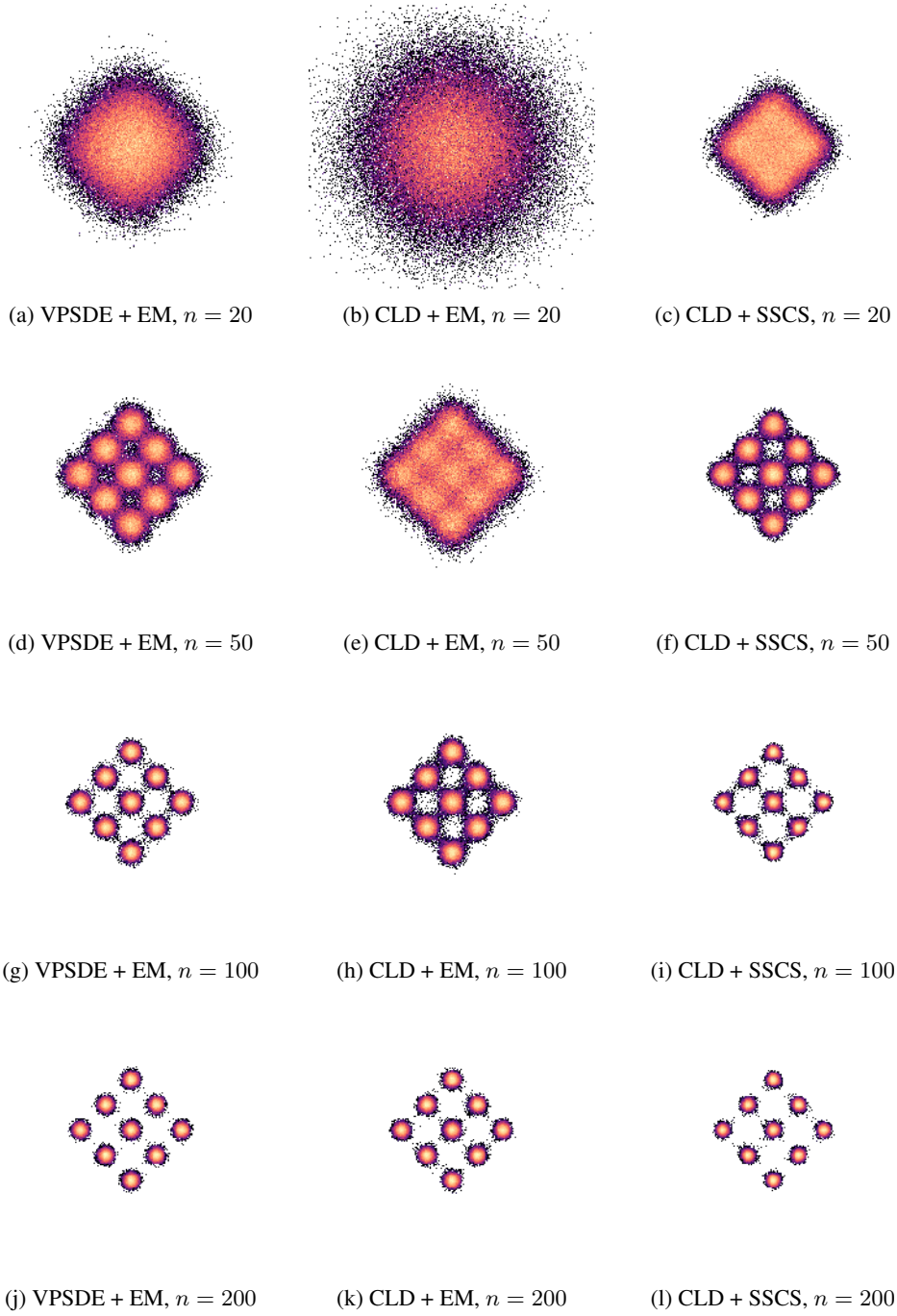
For maximum likelihood training, models based on overdamped Langevin dynamics such as VPSDE need to learn an unbounded score for $t \to 0$. Our model, on the other hand, only ever needs to learn a bounded score even for $t = 0$. For our image data experiments, we use a reweighted objective function to improve visual quality of samples (as is general practice).

Here, we also study training towards maximum likelihood on toy dataset tasks. To explore this, we repeat the neural network complexity experiment from App. E.1 with maximum likelihood training (instead of the reweighted objective). Furthermore, we also train VPSDE-based and CLD-based SGMs on a challenging toy dataset and find that CLD significantly outperforms VPSDE. We leave the study of CLD with maximum likelihood training for high-dimensional (image) datasets to future work.

Complexity Experiment. The setup of this experiment is equivalent to the setup in App. E.1 up to the training objective: in this experiment we do maximum likelihood learning, i.e., we train CLD models with the objective from Eq. (8) with $\lambda(t) = \Gamma\beta$. [8] Furthermore, we test CLD in this setup for three different values of γ. The results of this experiment can be found in Fig. 11. For CLD, we find that larger values of γ generally lead to less complex networks, in particular for smaller times t. However, even for $\gamma = 0.04$ the learned neural network is still significantly smoother than the network learned for VPSDE when a mixed score parameterization is used. [9]

[8] For the ML objective of VPSDE, we refer the reader to Song et al. (2021b).

[9] VPSDE with mixed score parameterization did not converge to the target distribution, and therefore is not included in Fig. 11.



(a) VPSDE + EM, $n = 20$ (b) CLD + EM, $n = 20$ (c) CLD + SSCS, $n = 20$

(d) VPSDE + EM, $n = 50$ (e) CLD + EM, $n = 50$ (f) CLD + SSCS, $n = 50$

(g) VPSDE + EM, $n = 100$ (h) CLD + EM, $n = 100$ (i) CLD + SSCS, $n = 100$

(j) VPSDE + EM, $n = 200$ (k) CLD + EM, $n = 200$ (l) CLD + SSCS, $n = 200$

Figure 10: Mixture of Normals: numerical simulation with analytical score function for different diffusions (VPSDE with EM vs. CLD with EM/SSCS) and number of synthesis steps n. A visualization of the data distribution can be found in Fig. 9a.

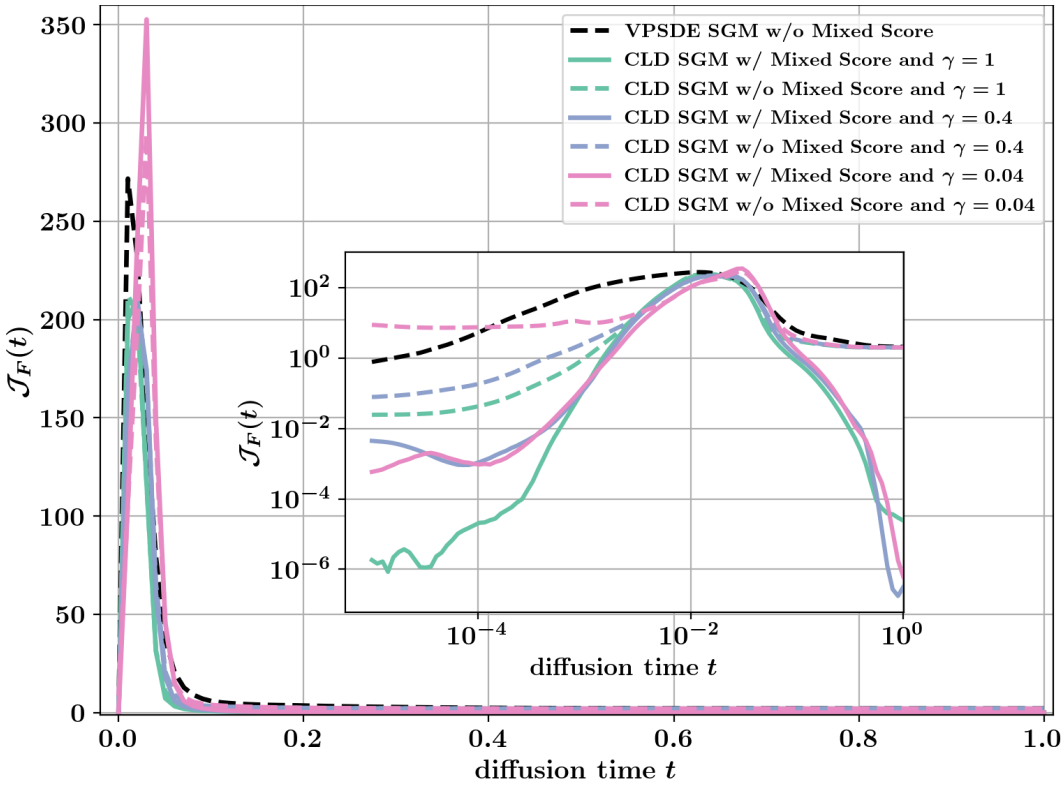
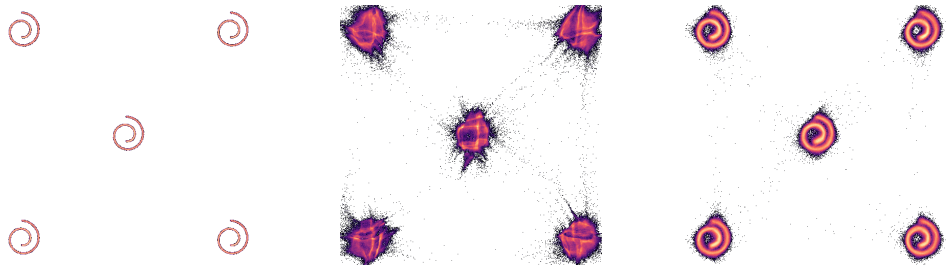


Figure 11: Frobenius norm $\mathcal{J}_F(t)$ of the neural network defining the score function for different t.



(a) Data (b) VPSDE (c) CLD

Figure 12: Data distribution and model samples for multi-scale toy experiment.

Challenging Toy Dataset. Using the same simple ResNet architecture (less than 100k parameters) from the above experiment, we trained a VPSDE-based as well as a CLD-based SGM to maximize the likelihood of a more challenging toy dataset (the dataset is essentially "multi-scale", as it involves both large scale—the placement of the swiss rolls—and fine scale—the swiss rolls themselves—structure). Similar to the other toy datasets, the models are trained for 1M iterations using fresh data synthesized from the data distribution in each batch at a batch size of 512.

In Fig. 12, we compare samples of the models to the data distribution. Even with our simple model architecture, CLD is able to capture the multi-scale structure of the dataset: the five rolls are adequately resembled and only a few samples are in between modes. VPSDE, on the other hand, only captures the modes, but not the fine structure. Furthermore, VPSDE has the undesired behavior of "connecting" the modes.

Overall, we conclude that also in the maximum likelihood training setting CLD is a promising diffusion showing superior behavior compared to the VPSDE in our toy experiments.

Table 9: *(bottom)* Performance using non-adaptive stepsize solvers. Extended version of Tab. 3.

		FID at n function evaluations ↓					
Model	Sampler	n=50	n=150	n=275	n=500	n=1000	n=2000
CLD	EM	143	31.5	10.9	3.96	2.50	2.27
CLD	EM-QS	52.7	7.00	3.24	2.41	**2.27**	**2.23**
CLD	SSCS	81.1	10.5	2.86	2.30	2.32	2.29
CLD	SSCS-QS	**20.5**	**3.07**	**2.38**	**2.25**	2.30	2.29
VPSDE	EM	92.0	30.3	13.1	4.42	2.46	2.43
VPSDE	EM-QS	28.2	4.06	2.65	2.47	2.66	2.60
VESDE	PC	460	216	11.2	3.75	2.43	**2.23**
VESDE	PC-QS	461	388	155	5.47	11.4	11.2

F.2 CIFAR-10 - Extended Results

In this section we provide additional results on CIFAR-10. For an extended version of Tab. 3, please see Tab. 9. Note that quadratic striding improves VPSDE and CLD for all settings but VPSDE with $n = \{1000, 2000\}$. For VESDE (using PC sampling), quadratic striding significantly worsens FID numbers. This is potentially due to variance of VESDE already following an exponential schedule (see Figure 5 in Song et al. (2021c)).

Furthermore, we present additional generated samples from our model: see Fig. 13 and Fig. 14 for samples from EM-QS with 2000 evaluations and SSCS with 150 evaluations, respectively.

F.3 CelebA-HQ-256 - Extended Results

In this section we provide additional qualitative results on CelebA-HQ-256. For high quality samples using our new SSCS solver see Fig. 15.

Samples generated with Runge–Kutta 4(5) at different solver tolerances can be found in Fig 16. We found that our model still generates very good samples even for solver error tolerance of $1\mathrm{e}{-3}$ using an average of 129 function evaluations.

Figure 13: Additional samples using EM-QS with 2000 function evaluations. This setup gave us our best FID number of 2.23.

Figure 14: Additional samples using SSCS-QS. This setup resulted in an FID of 3.07 using only 150 function evaluations.

Figure 15: Samples generated by our model on the CelebA-HQ-256 dataset using our SSCS solver.

(a) ODE solver error tolerance $1e{-5}$; 273 average NFE.

(b) ODE solver error tolerance $1e{-4}$; 190 average NFE.

(c) ODE solver error tolerance $1e{-3}$; 129 average NFE.

(d) ODE solver error tolerance $1e{-2}$; 99.4 average NFE.

Figure 16: Samples generated by our model on the CelebA-HQ-256 dataset using a Runge-Kutta 4(5) adaptive ODE solver to solve the probability flow ODE. We show the effect of the ODE solver error tolerance on the quality of samples. Little visual differences can be seen between $1e{-5}$ and $1e{-4}$. Low frequency artifacts can be observed at $1e{-3}$. Deterioration starts to set in at $1e{-2}$.