Score-based Generative Modeling in Latent Space

Anonymous Author(s)
Affiliation
Address
email

Abstract

Score-based generative models (SGMs) have recently demonstrated impressive results in terms of both sample quality and distribution coverage. However, they are usually applied directly in data space and often require thousands of network evaluations for sampling. Here, we propose the *Latent Score-based Generative Model* (LSGM), a novel approach that trains SGMs in a latent space, relying on the variational autoencoder framework. Moving from data to latent space allows us to train more expressive generative models, apply SGMs to non-continuous data, and learn smoother SGMs in a smaller space, resulting in fewer network evaluations and faster sampling. To enable training LSGMs end-to-end in a scalable and stable manner, we (i) introduce a new score-matching objective suitable to the LSGM setting, (ii) propose a novel parameterization of the score function that allows SGM to focus on the mismatch of the target distribution with respect to a simple Normal one, and (iii) analytically derive multiple techniques for variance reduction of the training objective. LSGM obtains a state-of-the-art FID score of 2.10 on CIFAR-10, outperforming all existing generative results on this dataset. On CelebA-HQ-256, LSGM is on a par with previous SGMs in sample quality while outperforming them in sampling time by two orders of magnitude. In modeling binary images, LSGM achieves state-of-the-art likelihood on the binarized OMNIGLOT dataset.

1 Introduction

The long-standing goal of likelihood-based generative learning is to learn a data distribution faithfully, while also generating high-quality samples. Achieving these two goals simultaneously is a tremendous challenge, which has led to the development of a plethora of different generative models. Recently, score-based generative models (SGMs) demonstrated astonishing results in terms of both high sample quality and likelihood [1, 2]. These models define a forward diffusion process that maps data to noise by gradually perturbing the input data. Generation corresponds to a reverse process that synthesizes novel data via iterative denoising, starting from random noise. The problem then reduces to learning *the score function*—the gradient of the log-density—of the perturbed data [3]. In a seminal work, Song et al. [2] show how this modeling approach is described with a stochastic differential equation (SDE) framework which can be converted to maximum likelihood training [4]. Variants of SGMs have been applied to images [1, 2, 5, 6], audio [7, 8, 9, 10], graphs [11] and point clouds [12, 13].

Albeit high quality, sampling from SGMs is computationally expensive. This is because generation amounts to solving a complex SDE, or equivalently ordinary differential equation (ODE) (denoted as the *probability flow ODE* in [2]), that maps a simple base distribution to the complex data distribution. The resulting differential equations are typically complex and solving them accurately requires numerical integration with very small step sizes which results in thousands of neural network evaluations [1, 2, 6]. Furthermore, generation complexity is uniquely defined by the underlying data distribution and the forward SDE for data perturbation, implying that synthesis speed cannot be increased easily without sacrifices. Moreover, SDE-based generative models are currently defined for continuous data and cannot be applied effortlessly to binary, categorical, or graph-structured data.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

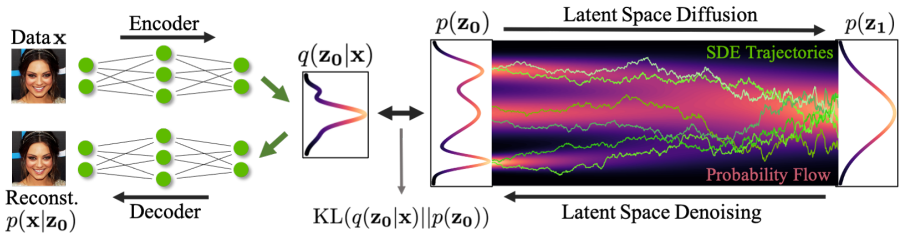

Figure 1: In our latent score-based generative model, data is mapped to latent space via encoder $q(\mathbf{z}_0|\mathbf{x})$ and a diffusion process is applied in the latent space ($\mathbf{z}_0 \rightarrow \mathbf{z}_1$). Synthesis starts from the base distribution $p(\mathbf{z}_1)$ and generates samples in latent space via denoising ($\mathbf{z}_0 \leftarrow \mathbf{z}_1$). Then, the samples are mapped from latent to data space using decoder $p(\mathbf{x}|\mathbf{z}_0)$. The model is trained end-to-end.

Here, we propose the *Latent Score-based Generative Model* (LSGM), a new approach for learning SGMs in latent space, leveraging a variational autoencoder (VAE) framework [14, 15]. We map the input data to latent space and apply the score-based generative model there. The score-based model is then tasked with modeling the distribution over the embeddings of the data set. Novel data synthesis is achieved by first generating embeddings via drawing from a simple base distribution followed by iterative denoising, and then transforming this embedding via a decoder to data space (see Fig. 1). We can consider this model a VAE with a SGM prior. Our approach has several key advantages:

Synthesis Speed: By pretraining the VAE with a Normal prior first, we can bring the marginal distribution over encodings (the *aggregate posterior*) close to the Normal prior, which is also the SGM's base distribution. Consequently, the SGM only needs to model the remaining mismatch, resulting in a less complex model from which sampling becomes easier. Furthermore, we can tailor the latent space according to our needs. For example, we can use hierarchical latent variables and apply the diffusion model only over a subset of them, further improving synthesis speed.

Expressivity: Training a regular SGM can be considered as training a neural ODE directly on the data [2]. However, previous works found that augmenting neural ODEs [16, 17] and more generally generative models [18, 19, 20, 21] with latent variables improves their expressivity. Consequently, we expect similar performance gains from combining SGMs with a latent variable framework.

Tailored Encoders and Decoders: Since we use the SGM in latent space, we can utilize carefully designed encoders and decoders mapping between latent and data space, further improving expressivity. Additionally, the LSGM method can therefore be naturally applied to non-continuous data.

LSGMs can be trained end-to-end by maximizing the variational lower bound on the data likelihood. Compared to regular score matching, our approach comes with additional challenges, since both the score-based denoising model and its target distribution, formed by the latent space encodings, are learnt simultaneously. To this end, we make the following technical contributions: **(i)** We derive a new denoising score matching objective that allows us to efficiently learn the VAE model and the latent SGM prior at the same time. **(ii)** We introduce a new parameterization of the latent space score function, which mixes a Normal distribution with a learnable SGM, allowing the SGM to model only the mismatch between the distribution of latent variables and the Normal prior. **(iii)** We propose techniques for variance reduction of the training objective by designing a new SDE and by analytically deriving importance sampling schemes, allowing us to stably train deep LSGMs. Experimentally, we achieve state-of-the-art 2.10 FID on CIFAR-10 and 7.22 FID on CelebA-HQ-256, and significantly improve upon likelihoods of previous SGMs. On CelebA-HQ-256, we outperform previous SGMs in synthesis speed by two orders of magnitude. We also model binarized images, MNIST and OMNIGLOT, achieving state-of-the-art likelihood on the latter.

2 Background

Here, we review continuous-time score-based generative models (see [2] for an in-depth discussion). Consider a forward diffusion process $\{\mathbf{z}_t\}_{t=0}^{t=1}$ for continuous time variable $t \in [0, 1]$, where $\mathbf{z}_0$ is the starting variable and $\mathbf{z}_t$ its perturbation at time t. The diffusion process is defined by an Itô SDE:

$$\mathrm{d}\mathbf{z} = f(t)\mathbf{z}\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w} \tag{1}$$

where $f : \mathbb{R} \rightarrow \mathbb{R}$ and $g : \mathbb{R} \rightarrow \mathbb{R}$ are scalar drift and diffusion coefficients, respectively, and $\mathbf{w}$ is the standard Wiener process. $f(t)$ and $g(t)$ can be designed such that $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{z}_1; \mathbf{0}, \mathbf{I})$ follows a Normal distribution at the end of the diffusion process.[1] Song et al. [2] show that the SDE in Eq. 1

[1] Other distributions at $t = 1$ are possible; for instance, see the "variance-exploding" SDE in [2]. In this paper, however, we use only SDEs converging towards $\mathcal{N}(\mathbf{z}_1; \mathbf{0}, \mathbf{I})$ at $t = 1$.

can be converted to a generative model by first sampling from $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{z}_1; \mathbf{0}, \mathbf{I})$ and then running the reverse-time SDE $\mathrm{d}\mathbf{z} = [f(t)\mathbf{z} - g(t)^2 \nabla_{\mathbf{z}} \log q_t(\mathbf{z})]\,\mathrm{d}t + g(t)\,\mathrm{d}\bar{\mathbf{w}}$, where $\bar{\mathbf{w}}$ is a reverse-time standard Wiener process and $\mathrm{d}t$ is an infinitesimal negative time step. The reverse SDE requires knowledge of $\nabla_{\mathbf{z}_t} \log q_t(\mathbf{z}_t)$, the score function of the marginal distribution under the forward diffusion at time t. One approach for estimating it is via the score matching objective[2]:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}[0,1]} \left[\lambda(t) \mathbb{E}_{q(\mathbf{z}_0)} \mathbb{E}_{q(\mathbf{z}_t|\mathbf{z}_0)} [||\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t) - \nabla_{\mathbf{z}_t} \log p_{\boldsymbol{\theta}}(\mathbf{z}_t)||_2^2] \right] \tag{2}$$

that trains the parameteric score function $\nabla_{\mathbf{z}_t} \log p_{\boldsymbol{\theta}}(\mathbf{z}_t)$ at time $t \sim \mathcal{U}[0,1]$ for a given weighting coefficient $\lambda(t)$. $q(\mathbf{z}_0)$ is the $\mathbf{z}_0$-generating distribution and $q(\mathbf{z}_t|\mathbf{z}_0)$ is the diffusion kernel, which is available in closed form for certain $f(t)$ and $g(t)$. Since $\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t)$ is not analytically available, Song et al. [2] rely on denoising score matching [22] that converts the objective in Eq. 2 to:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}[0,1]} \left[\lambda(t) \mathbb{E}_{q(\mathbf{z}_0)} \mathbb{E}_{q(\mathbf{z}_t|\mathbf{z}_0)} [||\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{z}_0) - \nabla_{\mathbf{z}_t} \log p_{\boldsymbol{\theta}}(\mathbf{z}_t)||_2^2] \right] + C \tag{3}$$

Vincent [22] shows $C = \mathbb{E}_{t\sim\mathcal{U}[0,1]} [\lambda(t) \mathbb{E}_{q(\mathbf{z}_0)} \mathbb{E}_{q(\mathbf{z}_t|\mathbf{z}_0)} [||\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t)||_2^2 - ||\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{z}_0)||_2^2]]$ is independent of $\boldsymbol{\theta}$, making the minimizations in Eq. 3 and Eq. 2 equivalent. Durkan and Song [4] show that for $\lambda(t) = g(t)^2/2$, the minimizations correspond to maximum likelihood training. For instance, the objective of Eq. 2 can then be written in terms of a Kullback-Leibler (KL) divergence:

$$\mathrm{KL}\big(q(\mathbf{z}_0)||p_{\boldsymbol{\theta}}(\mathbf{z}_0)\big) = \mathbb{E}_{t\sim\mathcal{U}[0,1]} \left[\frac{g(t)^2}{2} \mathbb{E}_{q(\mathbf{z}_0)} \mathbb{E}_{q(\mathbf{z}_t|\mathbf{z}_0)} \left[||\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t) - \nabla_{\mathbf{z}_t} \log p_{\boldsymbol{\theta}}(\mathbf{z}_t)||_2^2 \right] \right] \tag{4}$$

which can again be transformed into denoising score matching (Eq. 3) following Vincent [22].

3 Score-based Generative Modeling in Latent Space

The LSGM framework in Fig. 1 consists of the encoder $q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})$, SGM prior $p_{\boldsymbol{\theta}}(\mathbf{z}_0)$, and decoder $p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z}_0)$. The SGM prior leverages a diffusion process as defined in Eq. 1 and diffuses $\mathbf{z}_0 \sim q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})$ samples in latent space to the standard Normal distribution $p(\mathbf{z}_1) = \mathcal{N}(\mathbf{z}_1; \mathbf{0}, \mathbf{I})$. Generation uses the reverse SDE to sample from $p_{\boldsymbol{\theta}}(\mathbf{z}_0)$ with time-dependent score function $\nabla_{\mathbf{z}_t} \log p_{\boldsymbol{\theta}}(\mathbf{z}_t)$, and the decoder $p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z}_0)$ to map the synthesized encodings $\mathbf{z}_0$ to data space. Formally, the generative process is written as $p(\mathbf{z}_0, \mathbf{x}) = p_{\boldsymbol{\theta}}(\mathbf{z}_0) p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z}_0)$. The goal of training is to learn $\{\boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{\psi}\}$, the parameters of the encoder $q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})$, score function $\nabla_{\mathbf{z}_t} \log p_{\boldsymbol{\theta}}(\mathbf{z}_t)$, and decoder $p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z}_0)$, respectively.

We train LSGM by minimizing the variational upper bound on negative data log-likelihood $\log p(\mathbf{x})$:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{\psi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})} \left[-\log p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z}_0) \right] + \mathrm{KL}\big(q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}_0)\big) \tag{5}$$

$$= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})} \left[-\log p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z}_0) \right]}_{\text{reconstruction term}} + \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})} \left[\log q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x}) \right]}_{\text{negative encoder entropy}} + \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})} \left[-\log p_{\boldsymbol{\theta}}(\mathbf{z}_0) \right]}_{\text{cross entropy}} \tag{6}$$

following a VAE approach [14, 15], where $q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})$ approximates the true posterior $p(\mathbf{z}_0|\mathbf{x})$.

In this paper, we use Eq. 6 with decomposed KL divergence into its entropy and cross entropy terms. The reconstruction and entropy terms are estimated easily for any explicit encoder as long as the reparameterization trick is available [14]. The challenging part in training LSGM is to train the cross entropy term that involves the SGM prior. We motivate and present our expression for the cross-entropy term in Sec. 3.1, the parameterization of the SGM prior in Sec. 3.2, different weighting mechanisms for the training objective in Sec. 3.3, and variance reduction techniques in Sec. 3.4.

3.1 The Cross Entropy Term

One may ask, why not train LSGM with Eq. 5 and rely on the KL in Eq. 4. Directly using the KL expression in Eq. 4 is not possible, as it involves the marginal score $\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t)$, which is unavailable analytically for common non-Normal distributions $q(\mathbf{z}_0)$ such as Normalizing flows. Transforming into denoising score matching does not help either, since in that case the problematic $\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t)$ term appears in the C term (see Eq. 3). In contrast to previous works [2, 22], we cannot simply drop C, since it is, in fact, not constant but depends on $q(\mathbf{z}_t)$, which is trainable in our setup.

To circumvent this problem, we instead decompose KL in Eq. 5 and rather work directly with the cross entropy between the encoder distribution $q(\mathbf{z}_0|\mathbf{x})$ and the SGM prior $p(\mathbf{z}_0)$. We show that:

[2] We omit the t-subscript of the diffused distributions q_t in all score functions of the form $\nabla_{\mathbf{z}_t} \log q_t(\mathbf{z}_t)$

Theorem 1. *Given two distributions $q(\mathbf{z}_0|\mathbf{x})$ and $p(\mathbf{z}_0)$, defined in the continuous space $\mathbb{R}^D$, denote the marginal distributions of diffused samples under the SDE in Eq. 1 at time t with $q(\mathbf{z}_t|\mathbf{x})$ and $p(\mathbf{z}_t)$. Assuming mild smoothness conditions on $\log q(\mathbf{z}_t|\mathbf{x})$ and $\log p(\mathbf{z}_t)$, the cross-entropy is:*

$$CE(q(\mathbf{z}_0|\mathbf{x})||p(\mathbf{z}_0)) = \mathbb{E}_{t\sim\mathcal{U}[0,1]}\left[\frac{g(t)^2}{2}\mathbb{E}_{q(\mathbf{z}_t,\mathbf{z}_0|\mathbf{x})}\left[||\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{z}_0) - \nabla_{\mathbf{z}_t}\log p(\mathbf{z}_t)||_2^2\right]\right] + \frac{D}{2}\log\left(2\pi e\sigma_0^2\right),$$

with $q(\mathbf{z}_t,\mathbf{z}_0|\mathbf{x}) = q(\mathbf{z}_t|\mathbf{z}_0)q(\mathbf{z}_0|\mathbf{x})$ and a Normal transition kernel $q(\mathbf{z}_t|\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_t;\boldsymbol{\mu}_t(\mathbf{z}_0),\sigma_t^2\mathbf{I})$, where $\boldsymbol{\mu}_t$ and σ_t^2 are obtained from $f(t)$ and $g(t)$ for a fixed initial variance σ_0^2 at $t=0$.

A proof with generic expressions for $\boldsymbol{\mu}_t$ and σ_t^2 as well as an intuitive interpretation are in App. A.

Importantly, unlike for the KL objective of Eq. 4, no problematic terms depending on the marginal score $\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{x})$ arise. This allows us to use this denoising score matching objective for the cross entropy term in Theorem 1 not only for optimizing $p(\mathbf{z}_0)$ (which is commonly done in the score matching literature), but also for the $q(\mathbf{z}_0|\mathbf{x})$ encoding distribution. It can be used even with complex $q(\mathbf{z}_0|\mathbf{x})$ distributions, defined, for example, in a hierarchical fashion [20, 21] or via Normalizing flows [23, 24]. Our novel analysis shows that, for diffusion SDEs following Eq. 1, only the cross entropy can be expressed purely with $\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{z}_0)$. Neither KL nor entropy in [4] can be expressed without the problematic term $\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{x})$ (details in the Appendix).

3.2 Mixing Normal and Neural Score Functions

In VAEs [14], $p(\mathbf{z}_0)$ is often chosen as a standard Normal $\mathcal{N}(\mathbf{z}_0;\mathbf{0},\mathbf{I})$. For recent hierarchical VAEs [20, 21], using the reparameterization trick, the prior can be converted to $\mathcal{N}(\mathbf{z}_0;\mathbf{0},\mathbf{I})$ (App. E).

Considering a single dimensional latent space, we can assume that the prior at time t is in the form of a geometric mixture $p(z_t) \propto \mathcal{N}(z_t;0,1)^{1-\alpha}p'_{\boldsymbol{\theta}}(z_t)^{\alpha}$ where $p'_{\boldsymbol{\theta}}(z_t)$ is a trainable SGM prior and $\alpha \in [0,1]$ is a learnable scalar mixing coefficient. Formulating the prior this way has crucial advantages: (i) We can pretrain LSGM's autoencoder networks assuming α=0, which corresponds to training the VAE with a standard Normal prior. This pretraining step will bring the distribution of latent variable as close as possible to $\mathcal{N}(z_0;0,1)$, allowing the SGM prior to learn a much simpler distribution in the following end-to-end training stage. (ii) The score function for this mixture is of the form $\nabla_{z_t}\log p(z_t) = -(1-\alpha)z_t + \alpha\nabla_{z_t}\log p'_{\boldsymbol{\theta}}(z_t)$. When the score function is dominated by the linear term, we can expect that the reverse SDE is easy to solve, allowing for faster sampling.

For our multivariate latent space, we obtain diffused samples at time t by sampling $\mathbf{z}_t \sim q(\mathbf{z}_t|\mathbf{z}_0)$ with $\mathbf{z}_t = \boldsymbol{\mu}_t(\mathbf{z}_0) + \sigma_t\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})$. Since we have $\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{z}_0) = -\boldsymbol{\epsilon}/\sigma_t$, similar to [1], we parameterize the score function by $\nabla_{\mathbf{z}_t}\log p(\mathbf{z}_t) := -\boldsymbol{\epsilon}_\theta(\mathbf{z}_t,t)/\sigma_t$, where $\boldsymbol{\epsilon}_\theta(\mathbf{z}_t,t) := \sigma_t(1-\boldsymbol{\alpha})\odot\mathbf{z}_t + \boldsymbol{\alpha}\odot\boldsymbol{\epsilon}'_\theta(\mathbf{z}_t,t)$ is defined by our *mixed score parameterization* that is applied elementwise to the components of the score. With this, we simplify the cross entropy expression to:

$$\mathrm{CE}(q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}_0)) = \mathbb{E}_{t\sim\mathcal{U}[0,1]}\left[\frac{w(t)}{2}\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_t,\mathbf{z}_0|\mathbf{x}),\boldsymbol{\epsilon}}\left[||\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t,t)||_2^2\right]\right] + \frac{D}{2}\log\left(2\pi e\sigma_0^2\right), \quad (7)$$

where $w(t) = g(t)^2/\sigma_t^2$ is a time-dependent weighting scalar. See Appendix for details.

3.3 Training with Different Weighting Mechanisms

Table 1: Weighting mechanisms

Mechanism	Weights
Weighted	$w_{\text{ll}}(t) = g(t)^2/\sigma_t^2$
Unweighted	$w_{\text{un}}(t) = 1$
Reweighted	$w_{\text{re}}(t) = g(t)^2$

The weighting term $w(t)$ in Eq. 7 trains the prior with maximum likelihood. Similar to [1, 2], we observe that when $w(t)$ is dropped while training the SGM prior (i.e., $w(t)=1$), LSGM often yields higher quality samples at a small cost in likelihood. However, in our case, we can only drop the weighting when training the prior. When updating the encoder parameters, we still need to use the maximum likelihood weighting to ensure that the encoder $q(\mathbf{z}_0|\mathbf{x})$ is brought closer to the true posterior $p(\mathbf{z}_0|\mathbf{x})$[3]. Tab. 1 summarizes three weighting mechanisms we consider in this paper: $w_{\text{ll}}(t)$ corresponds to maximum likelihood, $w_{\text{un}}(t)$ is the unweighted objective used by [1, 2], and $w_{\text{re}}(t)$ is a variant obtained by dropping only $1/\sigma_t^2$. This weighting mechanism has a similar affect on the sample quality as $w_{\text{un}}(t)=1$; however, in Sec. 3.4, we show that it is easier to define a variance reduction scheme for this weighting mechanism.

The following summarizes our training objectives (with $t\sim\mathcal{U}[0,1]$ and $\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})$):

[3]Minimizing $\mathcal{L}(\mathbf{x},\boldsymbol{\phi},\boldsymbol{\theta},\boldsymbol{\psi})$ w.r.t $\boldsymbol{\phi}$ is equivalent to minimizing $\mathrm{KL}\big(q(\mathbf{z}_0|\mathbf{x})||p(\mathbf{z}_0|\mathbf{x})\big)$ w.r.t $q(\mathbf{z}_0|\mathbf{x})$.

$$\min_{\boldsymbol{\phi},\boldsymbol{\psi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[-\log p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z}_0)\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[\log q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})\right] + \mathbb{E}_{t,\boldsymbol{\epsilon},q(\mathbf{z}_t|\mathbf{z}_0),q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[\frac{w_{\text{ll}}(t)}{2}||\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t,t)||_2^2\right] \quad (8)$$

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t,\boldsymbol{\epsilon},q(\mathbf{z}_t|\mathbf{z}_0),q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[\frac{w_{\text{ll/un/re}}(t)}{2}||\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t,t)||_2^2\right] \quad \text{with} \quad q(\mathbf{z}_t|\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_t;\boldsymbol{\mu}_t(\mathbf{z}_0),\sigma_t^2\mathbf{I}), \quad (9)$$

where Eq. 8 trains the VAE encoder and decoder parameters $\{\boldsymbol{\phi},\boldsymbol{\psi}\}$ using the variational bound $\mathcal{L}(\mathbf{x},\boldsymbol{\phi},\boldsymbol{\theta},\boldsymbol{\psi})$ from Eq. 6. Eq. 9 trains the prior with one of the three weighting mechanisms. Since, the SGM prior participates in the objective only in the cross entropy term, we only consider this term when training the prior. Efficient algorithms for training with these objective are presented in App. G.

3.4 Variance Reduction

The objectives in Eqs. 8 and 9 involve sampling of the time variable t, which has high variance [25]. We introduce several techniques for reducing this variance for all three objective weightings. We focus on the "variance preserving" SDEs (VPSDEs) [2, 1, 26], defined by $\mathrm{d}\mathbf{z} = -\frac{1}{2}\beta(t)\mathbf{z}\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\mathbf{w}$ where $\beta(t) = \beta_0 + (\beta_1 - \beta_0)t$ linearly interpolates in $[\beta_0, \beta_1]$ (other SDEs discussed in App. B).

We denote the marginal distribution of latent variables by $q(\mathbf{z}_0) := \mathbb{E}_{p_{\text{data}}(\mathbf{x})}[q(\mathbf{z}_0|\mathbf{x})]$. Here, we derive variance reduction techniques for $\text{CE}(q(\mathbf{z}_0)||p(\mathbf{z}_0))$, assuming that both $q(\mathbf{z}_0) = p(\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_0;\mathbf{0},\mathbf{I})$. This is a reasonable simplification for our analysis because pretraining our LSGM model with a $\mathcal{N}(\mathbf{z}_0;\mathbf{0},\mathbf{I})$ prior brings $q(\mathbf{z}_0)$ close to $\mathcal{N}(\mathbf{z}_0;\mathbf{0},\mathbf{I})$ and our SGM prior is often dominated by the fixed Normal mixture component. We empirically observe that the variance reduction techniques developed with this simplification still work well when $q(\mathbf{z}_0)$ and $p(\mathbf{z}_0)$ are not exactly $\mathcal{N}(\mathbf{z}_0;\mathbf{0},\mathbf{I})$.

Variance reduction for likelihood weighting: In App. B, for $q(\mathbf{z}_0) = p(\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_0;\mathbf{0},\mathbf{I})$, we show $\text{CE}(q(\mathbf{z}_0)||p(\mathbf{z}_0))$ is given by $\frac{D}{2}\mathbb{E}_{t\sim\mathcal{U}[0,1]}[\mathrm{d}\log\sigma_t^2/\mathrm{d}t] + \text{const}$. We consider two approaches:

(1) *Geometric VPSDE*: To reduce the variance sampling uniformly form t, we can design the SDE such that $\mathrm{d}\log\sigma_t^2/\mathrm{d}t$ is constant for $t \in [0,1]$. We show in App. B that a $\beta(t) = \log(\sigma_{\text{max}}^2/\sigma_{\text{min}}^2)\frac{\sigma_t^2}{(1-\sigma_t^2)}$ with geometric variance $\sigma_t^2 = \sigma_{\text{min}}^2(\sigma_{\text{max}}^2/\sigma_{\text{min}}^2)^t$ satisfies this condition. We call a VPSDE with this $\beta(t)$ a geometric VPSDE. σ_{min}^2 and σ_{max}^2 are the hyperparameters of the SDE, with $0<\sigma_{\text{min}}^2<\sigma_{\text{max}}^2<1$. Although our geometric VPSDE has a geometric variance progression similar to the "variance exploding" SDE (VESDE) [2], its variance throughout the process is bounded, $\sigma_t^2 < \sigma_{\text{max}}^2$ for $t \in [0,1]$. In App. B, we show that the VESDE does not come with a reduced variance by default.

(2) *Importance sampling (IS):* We can keep $\beta(t)$ and σ_t^2 unchanged for the original linear VPSDE, and instead use IS to minimize variance. The theory of IS shows that the proposal $r(t) \propto \mathrm{d}\log\sigma_t^2/\mathrm{d}t$ has minimum variance [27]. In App. B, we show that we can sample from $r(t)$ using inverse transform sampling $t = \text{var}^{-1}((\sigma_1^2)^\rho(\sigma_0^2)^{1-\rho})$ where var^{-1} is the inverse of σ_t^2 and $\rho \sim \mathcal{U}[0,1]$. This variance reduction technique is available for any VPSDE with arbitrary $\beta(t)$.

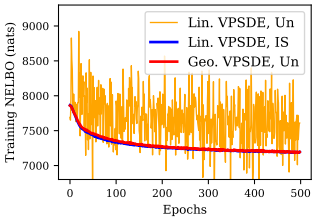

Figure 2: Variance reduction

In Fig. 2, we train a small LSGM on CIFAR-10 with w_{ll} weighting using (i) the original VPSDE with uniform t sampling, (ii) the same SDE but with our IS from t, and (iii) the proposed geometric VPSDE. Note how both (ii) and (iii) significantly reduce the variance and allow us to monitor the progress of the training objective. In this case, (i) has difficulty minimizing the objective due to the high variance. In App. B, we show how IS proposals can be formed for other SDEs, including the VESDE and Sub-VPSDE from [2].

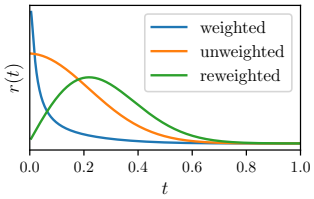

Figure 3: IS distributions

Variance reduction for unweighted and reweighted objectives: When training with w_{un}, analytically deriving IS proposal distributions for arbitrary $\beta(t)$ is challenging. For linear VPSDEs, we provide a derivation in App. B to obtain the optimal IS distribution. In contrast, defining IS proposal distributions is easier when training with w_{re}. In App. B, we show that the optimal distribution is in the form $r(t) \propto \mathrm{d}\sigma_t^2/\mathrm{d}t$ which is sampled by $t = \text{var}^{-1}((1-\rho)\sigma_0^2 + \rho\sigma_1^2)$ with $\rho \sim \mathcal{U}[0,1]$. In Fig. 3, we visualize three IS distributions for the three weighting mechanisms for the linear VPSDE with the original $[\beta_0, \beta_1]$ parameters from [2]. $r(t)$ for the likelihood weighting is more tilted towards $t = 0$ due to the $1/\sigma_t^2$ term in w_{ll}.

When using differently weighted objectives for training, we can either sample separate t with different IS distributions for each objective, or use IS for the SGM objective (Eq. 9) and reweight the samples according to the likelihood objective for encoder training (Eq. 8). See App. G for details.

4 Related Work

Our work builds on score-matching [28, 29, 30, 31, 32, 33, 34, 35, 36], specifically denoising score matching [22], which makes our work related to recent generative models using denoising score matching- and denoising diffusion-based objectives [3, 37, 1, 2, 6]. Among those, [1, 6] use a discretized diffusion process with many noise scales, building on [26], while Song et al. [2, 4] introduce the continuous time framework using SDEs. Experimentally, these works focus on image modeling and, contrary to us, work directly in pixel space. Various works recently tried to address the slow sampling of these types of models and further improve output quality. [38] add an adversarial objective, [5] introduce non-Markovian diffusion processes that allow to trade off synthesis speed, quality, and sample diversity, [39] learn a sequence of conditional energy-based models for denoising, [40] distill the iterative sampling process into single shot synthesis, and [41] learn an adaptive noise schedule, which is adjusted during synthesis to accelerate sampling. Further, [25] propose empirical variance reduction techniques for discretized diffusions and introduce a new, heuristically motivated, noise schedule. In contrast, our proposed noise schedule and our variance reduction techniques are analytically derived and directly tailored to our learning setting in the continuous time setup.

Concurrently with us, [10] proposed to learn a denoising diffusion model in the latent space of a VAE for symbolic music generation. This work does not introduce an end-to-end training framework of the combined VAE and denoising diffusion model and instead trains in two separate stages. Additionally, a discretized diffusion objective is used, similar to [1], while we build on the continuous time framework. Most importantly, a different domain, music generation, is targeted. Recently, [11] presented a method to generate graphs using score-based models, relaxing the entries of adjacency matrices to continuous values, treating the task as a continuous data modeling problem. LSGM would allow to model graph data more naturally using encoders and decoders tailored to graphs [42, 43, 44, 45].

Since our model can be considered a VAE [14, 15] with score-based prior, it is related to approaches that improve VAE priors. For example, Normalizing flows and hierarchical distributions [23, 24, 46, 47, 20, 21], as well as energy-based models [48, 49, 50, 51, 52] have been proposed as VAE priors. Furthermore, classifiers [53, 54, 55], adversarial methods [56], and other techniques [57, 58] have been used to define prior distributions implicitly. In two-stage training a separate generative model is trained in latent space as a new prior after training the VAE itself [59, 60, 61, 62, 63, 10].

5 Experiments

Here, we examine the efficacy of LSGM in learning generative models for images.

Implementation details: We implement LSGM using the NVAE [20] architecture as VAE backbone and NCSN++ [2] as SGM backbone. NVAE has a hierarchical latent structure. The diffusion process input $\mathbf{z}_0$ is constructed by concatenating the latent variables from all groups in the channel dimension. For NVAEs with multiple spatial resolutions in latent groups, we only feed the smallest resolution groups to the SGM prior and assume that the remaining groups have a standard Normal distribution.

Sampling: To generate samples from LSGM at test time, we use a black-box ODE solver [64] to sample from the prior. Prior samples are then passed to the decoder to generate samples in data space.

Evaluation: We measure NELBO, an upper bound on negative log-likelihood (NLL), using Eq. 6. For estimating $\log p(\mathbf{z}_0)$, we rely on the probability flow ODE [2], which provides an unbiased but stochastic estimation of $\log p(\mathbf{z}_0)$. This stochasticity prevents us from performing an importance weighted estimation of NLL [65] (see App. F for details). For measuring sample quality, Fréchet inception distance (FID) [66] is evaluated with 50K samples. Implementation details in App. G.

5.1 Main Results

Unconditional Color Image Generation: Here, we present our main results for unconditional image generation on CIFAR-10 [80] (Tab. 2) and CelebA-HQ-256 (5-bit quantized) [79] (Tab. 3). For CIFAR-10, we train 3 different models: *LSGM (FID)* and *LSGM (balanced)* both use the VPSDE with linear $\beta(t)$ and w_{un}-weighting for the SGM prior in Eq. 9, while performing IS as derived in Sec. 3.4. They only differ in how the backbone VAE is trained. *LSGM (NLL)* is a model that is trained with

Table 2: Generative performance on CIFAR-10.

	Method	NLL↓	FID↓
Ours	LSGM (FID)	≤3.43	**2.10**
	LSGM (NLL)	≤**2.87**	6.89
	LSGM (balanced)	≤2.95	2.17
	VAE Backbone	2.96	43.18
VAEs	VDVAE [21]	**2.87**	-
	NVAE [20]	2.91	23.49
	VAEBM [67]	-	12.19
	NCP-VAE [55]	-	24.08
	BIVA [47]	3.08	-
	DC-VAE [68]	-	17.90
Score	NCSN [3]	-	25.32
	Rec. Likelihood [39]	3.18	9.36
	DSM-ALS [38]	3.65	-
	DDPM [1]	3.75	3.17
	SDE (DDPM++) [2]	2.99	2.92
	SDE (NCSN++) [2]	-	2.20
Flows	VFlow [19]	2.98	-
	ANF [18]	3.05	-
Aut. Reg.	DistAug aug [69]	2.53	42.90
	Sp. Transformers [70]	2.80	-
	δ-VAE [71]	2.83	-
	PixelSNAIL [72]	2.85	-
	PixelCNN++ [73]	2.92	-
GANs	AutoGAN [74]	-	12.42
	StyleGAN2-ADA [75]	-	2.92

Table 3: Generative results on CelebA-HQ-256.

	Method	NLL↓	FID↓
Ours	LSGM	≤**0.70**	**7.22**
	VAE Backbone	**0.70**	30.87
VAEs	NVAE [20]	**0.70**	29.76
	VAEBM [67]	-	20.38
	NCP-VAE [55]	-	24.79
	DC-VAE [68]	-	15.80
Score	SDE [2]	-	**7.23**
Flows	GLOW [76]	1.03	68.93
Aut. Reg.	SPN [77]	0.61	-
GANs	Adv. LAE [78]	-	19.21
	VQ-GAN [63]	-	10.70
	PGGAN [79]	-	8.03

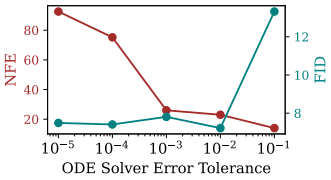

Figure 4: FID and number of function evaluations (NFEs) for different ODE solver error tolerances on CelebA-HQ-256. LSGM takes 2.74 sec. for sampling while the original SGM [2] takes 44 min. with PC and 3.9 min. with ODE-based sampling.

Table 4: Dyn. binarized OMNIGLOT results.

	Method	NELBO↓	NLL↓
Ours	LSGM	**87.79**	≤**87.79**
VAEs	NVAE [20]	93.92	90.75
	BIVA [47]	93.54	91.34
	DVAE++ [50]	-	92.38
	Ladder VAE [81]	-	102.11
Aut. Reg.	VLVAE [46]	-	89.83
	VampPrior [58]	-	89.76
	PixelVAE++ [82]	-	88.29

Table 5: Dynamically binarized MNIST results.

	Method	NELBO↓	NLL↓
Ours	LSGM	**78.47**	≤78.47
VAEs	NVAE [20]	79.56	**78.01**
	BIVA [47]	80.06	78.41
	IAF-VAE [24]	80.80	79.10
	DVAE++ [50]	-	78.49
Aut. Reg.	PixelVAE++ [82]	-	78.00
	VampPrior [58]	-	78.45
	MAE [83]	-	77.98

our novel geometric VPSDE, using w_{ll}-weighting in the prior objective (further details in App. G). When set up for high image quality, LSGM achieves a new state-of-the-art FID of 2.10. When tuned towards NLL, we achieve a NELBO of 2.87, which is significantly better than previous score-based models. Only autoregressive models, which come with very slow synthesis, and VDVAE [21] reach similar or higher likelihoods, but they usually have much poorer image quality.

For CelebA-HQ-256, we observe that when LSGM is trained with different SDE types and weighing mechanisms, it often obtains similar NELBO potentially because of applying SGM prior only to small latent variable groups and using Normal priors at the larger groups. With w_{re}-weighting and linear VPSDE, LSGM obtains the state-of-the-art FID score of 7.22 on a par with the original SGM [2].

For both datasets, we also report results for the VAE backbone used in our LSGM. Although this baseline achieves competitive NLL, its sample quality is behind our LSGM and the original SGM.

Modeling Binarized Images: Next, we examine LSGM on dynamically binarized MNIST [84] and OMNIGLOT [65]. We apply LSGM to binary images using a decoder with pixel-wise independent Bernoulli distributions. For these datasets, we report both NELBO and NLL in nats in Tab. 4 and Tab. 5. On OMNIGLOT, LSGM achieves state-of-the-art likelihood of ≤87.79 nat, outperforming previous models including VAEs with autoregressive decoders, and even when comparing its NELBO against importance weighted estimation of NLL for other methods. On MNIST, LSGM outperforms previous VAEs in NELBO, reaching a NELBO 1.09 nat lower than the state-of-the-art NVAE.

(a) CIFAR-10 (b) CelebA-HQ-256 (c) OMNIGLOT (d) MNIST

Figure 5: Generated samples for different datasets. For binary datasets, we visualize the decoder mean. LSGM successfully generates sharp, high-quality, and diverse samples (additional samples in appendix).

Qualitative Results: We visualize qualitative results for all datasets in Fig. 5. On the complex multimodal CIFAR-10 dataset, LSGM generates sharp and high-quality images. On CelebA-HQ-256, LSGM generates diverse samples from different ethnicity and age groups with varying head poses and facial expressions. On MNIST and OMNIGLOT, the generated characters are sharp and high-contrast.

Sampling Time: We compare LSGM against the original SGM [2] trained on the CelebA-HQ-256 dataset in terms of sampling time and number of function evaluations (NFEs) of the ODE solver. Song et al. [2] propose two main sampling techniques including predictor-corrector (PC) and probability flow ODE. PC sampling involves 4000 NFEs and takes 44.6 min. on a Titan V for a batch of 16 images. It yields 7.23 FID score (see Tab. 3). ODE-based sampling from SGM takes 3.91 min. with 335 NFEs, but it obtains a poor FID score of 128.13 with 10^{-5} as ODE solver error tolerance[4].

In a stark contrast, ODE-based sampling from our LSGM takes 0.07 min. with average of 23 NFEs, yielding 7.22 FID score. LSGM is 628$\times$ and 56$\times$ faster than original SGM's [2] PC and ODE sampling, respectively. In Fig. 4, we visualize FID scores and NFEs for different ODE solver error tolerances. Our LSGM achieves low FID scores for relatively large error tolerances.

We identify three main reasons for this significantly faster sampling from LSGM: (i) The SGM prior in our LSGM models latent variables with 32$\times$32 spatial dim., whereas the original SGM [2] directly models 256$\times$256 images. The larger spatial dimensions require a deeper network to achieve a large receptive field. (ii) Inspecting the SGM prior in our model shows that the score function is heavily dominated with the linear term at the end of training, as the mixing coefficients $\boldsymbol{\alpha}$ are all < 0.02. This makes our SGM prior smooth and numerically faster to solve. (iii) Since SGM is formed in the latent space in our model, errors from solving the ODE can be corrected to some degree using the VAE decoder, while in the original SGM [2] errors directly translate to artifacts in pixel space.

5.2 Ablation Studies

SDEs, Objective Weighting Mechanisms and Variance Reduction. In Tab. 6 we analyze the different weighting mechanisms and variance reduction techniques and compare the geometric VPSDE with the regular VPSDE with linear $\beta(t)$ [1, 2]. Previous work [20, 21] have reported instability in training large VAEs. We find that our method inherits similar instabilities from VAEs, however, we observe that importance sampling often stabilizes training our LSGM. As expected, we obtain the best NELBOs (red) when training with the weighted, maximum likelihood objective (w_{ll}). Importantly, our new geometric VPSDE achieves the best NELBO. Furthermore, the best FIDs (blue) are obtained either by unweighted (w_{un}) or reweighted (w_{re}) SGM prior training, with only slightly worse NELBOs. These experiments were run on the CIFAR10 dataset, using a smaller model than for our main results above (details in App. G).

End-to-End Training. We proposed to train LSGM end-to-end, in contrast to the previous work [10]. Using a similar setup as above we compare end-to-end training of LSGM during the second stage with freezing the VAE encoder and decoder and only training the SGM prior in latent space during

[4]We use the VESDE checkpoint at `https://github.com/yang-song/score_sde_pytorch`. Song et al. [2] report that ODE-based sampling yields worse FID scores for their models (see D.4 in [2]). The problem is more severe for VESDEs. Unfortunately, at submission time only a VESDE model was released.

Table 6: Ablation study on SDEs, training objectives, weighting mechanisms, and variance reduction. *SGM-obj.-weighting* denotes the weighting mechanism used when training the SGM prior (via Eq. 9). *t-sampling (SGM-obj.)* indicates the sampling approach for t, where $r_{\text{ll}}(t)$, $r_{\text{un}}(t)$ and $r_{\text{re}}(t)$ denote the IS distributions for the weighted (likelihood), the unweighted, and the reweighted objective, respectively. For training the VAE encoder $q_\phi(\mathbf{z}_0|\mathbf{x})$ (last term in Eq. 8), we either sample a separate batch t with importance sampling following $r_{\text{ll}}(t)$ (only necessary when the SGM prior is not trained with w_{ll} itself), or we simply *reweight* the samples drawn for training the prior according to the likelihood objective (denoted by *rew.*). *n/a* indicates fields that do not apply: The geometric VPSDE has optimal variance for the weighted (likelihood) objective already with uniform sampling; there is no additional IS distribution. Also, we did not derive IS distributions for the geometric VPSDE for w_{un}. *NaN* indicates experiments that failed due to training instabilities. Further details in App. G.

SGM-obj.-weighting		w_{ll}		w_{un}				w_{re}			
t-sampling (SGM-obj.)		$\mathcal{U}[0,1]$	$r_{\text{ll}}(t)$	$\mathcal{U}[0,1]$		$r_{\text{un}}(t)$		$\mathcal{U}[0,1]$		$r_{\text{re}}(t)$	
t-sampling (q-obj.)		*rew.*	*rew.*	*rew.*	$r_{\text{ll}}(t)$	*rew.*	$r_{\text{ll}}(t)$	*rew.*	$r_{\text{ll}}(t)$	*rew.*	$r_{\text{ll}}(t)$
Geom.-VPSDE	**FID↓**	10.18	*n/a*	*NaN*	*NaN*	*n/a*	*n/a*	22.21	*NaN*	7.29	7.18
	NELBO↓	2.96	*n/a*	*NaN*	*NaN*	*n/a*	*n/a*	3.04	*NaN*	2.99	2.99
VPSDE	**FID↓**	6.15	8.00	*NaN*	*NaN*	5.39	5.39	*NaN*	4.99	15.12	6.19
	NELBO↓	2.97	2.97	*NaN*	*NaN*	2.98	2.98	*NaN*	2.99	3.03	2.99

the second stage. When training the model end-to-end, we achieve an FID of 5.19 and NELBO of 2.98; when freezing the VAE networks during the second stage, we only get an FID of 9.00 and NELBO of 3.03. These results clearly motivate our end-to-end training strategy.

Mixing Normal and Neural Score Functions. We generally found training LSGM without our proposed "mixed score" formulation (Sec. 3.2) to be unstable during end-to-end training, highlighting its importance. To quantify the contribution of the mixed score parametrization for an stable model, we train a small LSGM with only one latent variable group. In this case, without the mixed score, we reached an FID of 34.71 and NELBO of 3.39; with it, we got an FID of 7.60 and NELBO of 3.29.

6 Conclusions

We proposed the *Latent Score-based Generative Model*, a novel framework for end-to-end training of score-based generative models in the latent space of a variational autoencoder. Moving from data to latent space allows us to form more expressive generative models, model non-continuous data, and reduce sampling time using smoother SGMs. To enable training latent SGMs, we made three core contributions: (i) we derived a simple expression for the cross entropy term in the variational objective, (ii) we parameterized the SGM prior by mixing Normal and neural score functions, and (iii) we proposed several techniques for variance reduction in the estimation of the training objective. Experimental results show that latent SGMs outperform recent pixel-space SGMs in terms of both data likelihood and sample quality, and they can also be applied to binary datasets. In large image generation, LSGM generates data several orders of magnitude faster than recent SGMs. Nevertheless, LSGM's synthesis speed does not yet permit sampling at interactive rates, and our implementation of LSGM is currently limited to image generation. Therefore, future work includes further accelerating sampling, applying LSGMs to other data types, and designing efficient networks for LSGMs.

7 Broader Impact

Generating high-quality samples while fully covering the data distribution has been a long-standing challenge in generative learning. A solution to this problem will likely help reduce biases in generative models and lead to improving overall representation of minorities in the data distribution. SGMs are perhaps one of the first deep models that excel at both sample quality and distribution coverage. However, the high computational cost of sampling limits their widespread use. Our proposed LSGM reduces the sampling complexity of SGMs by a large margin and improves their expressivity further. Thus, in the long term, it can enable the usage of SGMs in practical applications.

Here, LSGM is examined on the image generation task which has potential benefits and risks discussed in [85, 86]. However, LSGM can be considered as a generic framework that extends SGMs to non-continuous data types. In principle LSGM could be used to model, for example, language [87], music [88, 10], or molecules [89]. Furthermore, like other deep generative models, it can potentially be used also for non-generative tasks such as semi-supervised and representation learning [90, 91, 92]. This makes the long-term social impacts of LSGM dependent on the downstream applications.

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. *arXiv:2006.11239*, 2020.

[2] 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*, 2021.

[3] Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems 32*, 2019.

[4] Conor Durkan and Yang Song. On maximum likelihood training of score-based generative models. *arXiv preprint arXiv:2101.09258*, 2021.

[5] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[6] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *arXiv preprint arXiv:2105.05233*, 2021.

[7] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J. Weiss, Mohammad Norouzi, and William Chan. WaveGrad: Estimating Gradients for Waveform Generation. *arXiv:2009.00713*, 2020.

[8] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A Versatile Diffusion Model for Audio Synthesis. *arXiv:2009.09761*, 2020.

[9] 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:2104.01409*, 2021.

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

[11] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, 2020.

[12] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge J. Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part III*, volume 12348 of *Lecture Notes in Computer Science*, pages 364–381. Springer, 2020.

[13] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. *arXiv:2103.01458*, 2021.

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

[15] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.

[16] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural ODEs. In *Advances in Neural Information Processing Systems*, pages 3140–3150, 2019.

[17] Zhifeng Kong and Kamalika Chaudhuri. The expressive power of a class of normalizing flow models. *arXiv preprint arXiv:2006.00392*, 2020.

[18] 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.

[19] Jianfei Chen, Cheng Lu, Biqi Chenli, Jun Zhu, and Tian Tian. Vflow: More expressive generative flows with variational data augmentation. *arXiv preprint arXiv:2002.09741*, 2020.

[20] Arash Vahdat and Jan Kautz. NVAE: A Deep Hierarchical Variational Autoencoder. *arXiv:2007.03898*, 2020.

[21] Rewon Child. Very deep VAEs generalize autoregressive models and can outperform them on images. In *International Conference on Learning Representations*, 2021.

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

[23] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[24] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

[25] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672*, 2021.

[26] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning Using Nonequilibrium Thermodynamics. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2256–2265. JMLR.org, 2015.

[27] Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

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

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

[30] Durk P Kingma and Yann L. Cun. Regularized estimation of image statistics by Score Matching. In *Advances in Neural Information Processing Systems 23*, 2010.

[31] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized Denoising Auto-Encoders as Generative Models. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, 2013.

[32] Krzysztof J. Geras and Charles A. Sutton. Scheduled denoising autoencoders. In *3rd International Conference on Learning Representations, ICLR*, 2015.

[33] Saeed Saremi, Arash Mehrjou, Bernhard Schölkopf, and Aapo Hyvärinen. Deep Energy Estimator Networks. *arXiv:1805.08306*, 2018.

[34] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced Score Matching: A Scalable Approach to Density and Score Estimation. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*, 2019.

[35] Zengyi Li, Yubei Chen, and Friedrich T. Sommer. Learning Energy-Based Models in High-Dimensional Spaces with Multi-scale Denoising Score Matching. *arXiv:1910.07762*, 2019.

[36] Tianyu Pang, Kun Xu, Chongxuan Li, Yang Song, Stefano Ermon, and Jun Zhu. Efficient Learning of Generative Models via Finite-Difference Score Matching. *arXiv:2007.03317*, 2020.

[37] Yang Song and Stefano Ermon. Improved Techniques for Training Score-Based Generative Models. *arXiv:2006.09011*, 2020.

[38] 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*, 2021.

[39] 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.

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

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

[42] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. *arXiv:1802.03480*, 2018.

[43] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[44] Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[45] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. In *Advances in Neural Information Processing Systems*, 2019.

[46] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

[47] Lars Maaløe, Marco Fraccaro, Valentin Liévin, and Ole Winther. BIVA: A very deep hierarchy of latent variables for generative modeling. In *Advances in neural information processing systems*, pages 6548–6558, 2019.

[48] Jason Tyler Rolfe. Discrete variational autoencoders. *arXiv preprint arXiv:1609.02200*, 2016.

[49] Arash Vahdat, Evgeny Andriyash, and William G Macready. DVAE#: Discrete variational autoencoders with relaxed Boltzmann priors. In *Neural Information Processing Systems*, 2018.

[50] Arash Vahdat, William G. Macready, Zhengbing Bian, Amir Khoshaman, and Evgeny Andriyash. DVAE++: Discrete variational autoencoders with overlapping transformations. In *International Conference on Machine Learning (ICML)*, 2018.

[51] Arash Vahdat, Evgeny Andriyash, and William G Macready. Undirected graphical models as approximate posteriors. In *International Conference on Machine Learning (ICML)*, 2020.

[52] Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning latent space energy-based prior model. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21994–22008. Curran Associates, Inc., 2020.

[53] Jesse Engel, Matthew Hoffman, and Adam Roberts. Latent constraints: Learning to generate conditionally from unconditional generative models. In *International Conference on Learning Representations*, 2018.

[54] Matthias Bauer and Andriy Mnih. Resampled priors for variational autoencoders. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 66–75. PMLR, 16–18 Apr 2019.

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

[56] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders, 2016.

[57] Hiroshi Takahashi, Tomoharu Iwata, Yuki Yamanaka, Masanori Yamada, and Satoshi Yagi. Variational autoencoder with implicit optimal priors. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5066–5073, Jul. 2019.

[58] Jakub Tomczak and Max Welling. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223, 2018.

[59] Bin Dai and David Wipf. Diagnosing and enhancing VAE models. In *International Conference on Learning Representations*, 2019.

[60] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2018.

[61] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, pages 14837–14847, 2019.

[62] Partha Ghosh, Mehdi S. M. Sajjadi, Antonio Vergari, Michael Black, and Bernhard Scholkopf. From variational to deterministic autoencoders. In *International Conference on Learning Representations*, 2020.

[63] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. *arXiv preprint arXiv:2012.09841*, 2020.

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

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

[66] 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. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

[67] 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.

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

[69] 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*, pages 5006–5019. PMLR, 2020.

[70] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[71] Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. Preventing posterior collapse with delta-vaes. In *The International Conference on Learning Representations (ICLR)*, 2019.

[72] XI Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. PixelSNAIL: An improved autoregressive generative model. In *International Conference on Machine Learning*, 2018.

[73] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. PixelCNN++: Improving the pixelCNN with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[74] Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. Autogan: Neural architecture search for generative adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019.

[75] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *NeurIPS*, 2020.

[76] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10236–10245, 2018.

[77] Jacob Menick and Nal Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. In *International Conference on Learning Representations*, 2019.

[78] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[79] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[80] Alex Krizhevsky et al. Learning multiple layers of features from tiny images, 2009.

[81] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in neural information processing systems*, pages 3738–3746, 2016.

[82] Hossein Sadeghi, Evgeny Andriyash, Walter Vinci, Lorenzo Buffoni, and Mohammad H Amin. Pixelvae++: Improved pixelvae with discrete prior. *arXiv preprint arXiv:1908.09948*, 2019.

[83] Xuezhe Ma, Chunting Zhou, and Eduard Hovy. MAE: Mutual posterior-divergence regularization for variational autoencoders. In *The International Conference on Learning Representations (ICLR)*, 2019.

[84] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

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

[86] 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.

[87] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, 2016.

[88] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

[89] Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

[90] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

[91] Augustus Odena. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016.

[92] Daiqing Li, Junlin Yang, Karsten Kreis, Antonio Torralba, and Sanja Fidler. Semantic segmentation with generative models: Semi-supervised learning and strong out-of-domain generalization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[93] 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.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec. 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Sec. 7.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] For example, we state that our proposed techniques for reducing the variance of the sample-based estimate of the training objective assume that the marginal distribution of latent variables is Normal.
   (b) Did you include complete proofs of all theoretical results? [Yes] We include the proof of Theorem 1 and derivations for all other theoretical results, i.e. the novel variance reduction techniques, in the Appendix. We also compare our theoretical results to previous, related works [25, 4].
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See App. G.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Following standard conventions in the deep generative modeling literature, we do not report error bars. We avoid running similar setups repeatedly to save computational resources (our main models are quite large and require substantial GPU resources for training, see App. G).
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See App. G.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] For baseline comparisons we run publicly available code (`https://github.com/yang-song/score_sde_pytorch`) from a previous, publicly available paper, which we cite [2]. We also rely on NVAE [20] available at `https://github.com/NVlabs/NVAE`
   (b) Did you mention the license of the assets? [Yes] The submitted code comes with the license of the assets.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We are only using publicly available datasets, which have been widely used in the generative modeling literature as standard benchmarks (CIFAR-10, CelebA-HQ, MNIST, OMNIGLOT).
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]