Learning Generative Models Using Denoising Density Estimators

Bigdeli, Siavash A.; Lin, Geng; Dunbar, L. Andrea; Portenier, Tiziano; Zwicker, Matthias

Published in:
IEEE Transactions on Neural Networks and Learning Systems

Link to article, DOI:
10.1109/TNNLS.2023.3308191

Publication date:
2024

Document Version
Peer reviewed version

Learning Generative Models Using Denoising Density Estimators

Siavash A. Bigdeli, Geng Lin, L. Andrea Dunbar, *Member, IEEE*, Tiziano Portenier, and Matthias Zwicker, *Associate Member, IEEE*

***Abstract*— Learning probabilistic models that can estimate the density of a given set of samples, and generate samples from that density, is one of the fundamental challenges in unsupervised machine learning. We introduce a new generative model based on denoising density estimators (DDEs), which are scalar functions parametrized by neural networks, that are efficiently trained to represent kernel density estimators of the data. Leveraging DDEs, our main contribution is a novel technique to obtain generative models by minimizing the Kullback–Leibler (KL)-divergence directly. We prove that our algorithm for obtaining generative models is guaranteed to converge consistently to the correct solution. Our approach does not require specific network architecture as in normalizing flows (NFs), nor use ordinary differential equation (ODE) solvers as in continuous NFs. Experimental results demonstrate substantial improvement in density estimation and competitive performance in generative model training.**

***Index Terms*— Denoising autoencoders (DAEs), density estimation, energy models, generative modeling, score-matching.**

I. Introduction

LEARNING generative probabilistic models from raw data are one of the fundamental problems in unsupervised machine learning. Such models can generate new content, which has various applications such as image editing [1], anomaly detection [2], and localization [3]. These models enable sampling from the probability density represented by the input data or also perform density estimation and inference of latent variables. The recent use of deep neural networks has led to significant advances in this area. For example, generative adversarial networks (GANs) [4] can be trained to sample very high-dimensional densities, without explicitly providing density estimation or inference. Inference in Boltzmann machines [5] is tractable only under approximations [6]. Variational autoencoders [7] provide functionality for both (approximate) inference and sampling. NF [8] performs all three operations (sampling, density estimation, and inference) through highly constrained network architectures. Denoising diffusion models (DDMs) [9] overcome many of the challenges of previous approaches, but often require many iterations for sampling to converge.

In this article, we introduce a novel type of generative model based on what we call denoising density estimators (DDEs), which supports efficient sampling and density estimation. Our approach to constructing a sampler is straightforward: assuming we have a density estimator that can be efficiently trained and evaluated, we learn a sampler by forcing its generated density to be the same as the input data density via minimizing their Kullback–Leibler (KL) divergence. In particular, we use the *reverse* KL divergence, which has a mode-seeking behavior and is better at avoiding saddle points when the two distributions have a small overlap. In our approach, the density estimator is derived from the theory of denoising autoencoders (DAEs), hence our term DDEs. Recent nonparametric models that use neural networks [10], [11] can still perform on the order of 10-D, while our density estimation model can perform well on 3000-D image datasets. Compared to normalizing flows (NFs), a key advantage of our theory is that it does not require any specific network architecture, except differentiability, and we do not need to solve ordinary differential equations (ODEs) like in continuous NFs. In summary, our main contribution is a novel approach to obtaining a generative model by explicitly estimating the energy (unnormalized density) of the generated and true data distributions and minimizing the statistical divergence of these densities. We also provide extensive experiments to compare our approach with other techniques for generative modeling and density estimation.[1] We discuss the possibility of scaling our results to higher resolutions in Section V-H.

II. Related Work

In this section, we discuss prior work on learning generative modeling and density estimation. Table I summarizes the differences between our approach to GANs, stein variational gradient descent, denoising diffusion models, and normalizing flows.

Manuscript received 22 September 2022; revised 2 February 2023 and 17 April 2023; accepted 28 July 2023. *(Corresponding author: Siavash A. Bigdeli.)*

Siavash A. Bigdeli is with the Compute Department, Denmark Technical University, 2800 Copenhagen, Denmark (e-mail: sarbi@dtu.dk).

Geng Lin and Matthias Zwicker are with the Computer Science Department, University of Maryland, College Park, MD 20742 USA.

L. Andrea Dunbar is with the CSEM, 2002 Neuchâtel, Switzerland.

Tiziano Portenier is with the ETH Zurich, 8092 Zürich, Switzerland.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3308191.

Digital Object Identifier 10.1109/TNNLS.2023.3308191

[1]Links to online code repositories:
Experiments and results: https://github.com/logchan/dde
DDE example: https://github.com/siavashbigdeli/dde
Generative model example: https://github.com/siavashbigdeli/egm-dde

TABLE I
COMPARISON OF DIFFERENT DEEP GENERATIVE APPROACHES BASED ON GANS, SVGD, DDM, NF, AND OUR PROPOSED TECHNIQUE. ADVERSARIAL DENSITY ESTIMATION CAN BE ACHIEVED USING THE APPROACH BY ABBASNEJAD ET AL. [12] USING A SUITABLE TRAINING OBJECTIVE

Property	GAN	SVGD	DDM	NF	Ours
Provides density	(-)	-	(-)	✓	✓
Forward sampling model	✓	✓	iterative	✓	✓
Exact sampling	✓	asymptotic	asymptotic	✓	✓
Convergence guarantee	-	-	✓	✓	✓
Free network architecture	✓	✓	✓	-	✓

A. Maximum Likelihood Estimators

A common approach is to formulate generative models as mappings between a latent space and the data domain, and one way to categorize them is to consider the constraints on this mapping. In NFs [8] and [13], the mapping is invertible and differentiable, such that the data density can be estimated using the determinant of its Jacobian, and inference can be performed via the inverse mapping. NFs can be trained simply using maximum likelihood estimation (MLE) [14]. The challenge, however, is to design efficient computational structures for the required operations [15], [16]. Chen et al. [17] and Grathwohl et al. [18] derive continuous NFs by parameterizing the dynamics (the time derivative) of an ODE using a neural network, but it comes at the cost of solving ODEs to produce outputs. In contrast, in variational techniques, the relation between the latent variables and data is probabilistic, usually expressed as a Gaussian likelihood function. Hence, computing the marginal likelihood requires integration over latent space. To make this tractable, it is common to bound the marginal likelihood using the evidence lower bound [7]. Recently, Li and Malik [19] proposed an approximate form of MLE, which they call implicit MLE (IMLE), that can also be performed without requiring invertible mappings. As a disadvantage, IMLE requires nearest-neighbor queries in (high-dimensional) data space.

Following restricted Boltzmann machines, Liu et al. [20], [21], and [22] propose Hebb-inspired energy models of the neural network features. This allows to perform Markov chain Monte Carlo (MCMC) sampling of the joint input-label distribution for various machine learning tasks. Our energy model uses kernel density estimation (KDE) and does not rely on Hebb intuitions for learning.

B. Denoising Autoencoders

Not all generative models include a latent space, for example, autoregressive models [23] or DAEs [24]. In particular, Alain and Bengio [24] and Saremi and Hyv̈arinen [25] use the well-known relation between DAEs and the score of the corresponding data distributions [26], [27] to construct an approximate Markov chain sampling procedure. Similarly, many techniques [28], [29], [30] use DAEs to learn the gradient of image densities for optimizing maximum a posteriori problems in image restoration. In our previous work [31], [32], [33], we used DAEs to build priors based on the gradient of the log-likelihood for image restoration. In this work, we instead directly learn the log-likelihood (not its gradients) and use it to learn a generator model. We build on DAEs, but formulate an estimator for the unnormalized, scalar density (DDE), rather than for the score (a vector field). This is crucial to allow us to train a generator instead of requiring Markov chain sampling, which has the disadvantages of requiring sequential sampling and producing correlated samples.

C. Denoising Diffusion and Score-Matching Models

Diffusion models have been the most successful recently in generating high-quality samples. They start with a noise sample and form a Markov chain to produce a sample within the distribution. For this, they build conditional models that sample new output based on previously generated ones. Based on score-matching, Song and Ermon [34] formulate a generative model using Langevin dynamics, which uses an iterative sampling procedure for sampling that converges asymptotically. Similarly, Dai et al. [35] use adversarial training to learn dynamics for generating samples. Most score-based generative modeling techniques [34], [36], [37], including denoising diffusion models [9], [38], [39], [40], [41], require iterative steps with scheduling for noise levels in their sampling algorithms, which produce samples asymptotically as noise level reaches zero [42]. Even though we use a denoising score-matching loss, we learn an explicit sampling model that can be used in single forward passes and does not require an iterative sampling scheme in the training or the sampling step.

Instead of using a denoising objective, score-matching can also be achieved by minimizing Stein's loss for the true and estimated density gradients. Kingma and LeCun [43] use a regularized version of the loss to parameterize a product-of-experts model for images, and Li et al. [44] train deep density estimators based on exponential family kernels. These techniques require computation of third-order derivatives, however, limiting the dimensionality of their models. Song and Ermon [34] extend this approach by introducing a sliced score-matching objective that leads to more efficient training. Li and Turner [45] learn an energy model using Stein's score-matching and propose a gradient-free sampling of this energy using MCMC. Because of the complexity of computing the Jacobian of the energy model in the original score-matching objective, these methods would often fail to scale or produce visually competitive results [34]. Unlike these techniques, DDEs are optimized using a denoising objective, hence they can be optimized without approximations or higher-order derivatives (with respect to parameters). These properties allow us to efficiently train an exact generator that scales well with the data dimensionality.

An alternative approach to using the score in a diffusion process is the work of Wang et al. [46] that joins a GAN training with Langevin dynamics using Stein variational gradient descent (SVGD) to regularize the training by making MCMC sampling. Similarly, Tao et al. [47] used score-matching to regularize a GAN model training. Most similar to our work is the work of Liu and Wang [48] who propose SVGD that uses the Stein objective to learn gradients for minimizing divergence between a large set of particles (i.e., samples) and

a target distribution. Feng et al. [49] realize an amortized version of the Stein variational gradients and use it to update a generative model for sampling from a target distribution. These techniques use kernelized Stein discrepancy [50] to estimate the gradients (score) of the distribution. The main limiting factor of these techniques is the parameterization of the Stein variation gradients estimator that cannot be guaranteed to follow a conservative vector field. Unfortunately, this could, and, in practice, will,[2] lead to instabilities in the training of the generator as the gradients do not help find an equilibrium (inference). Our model guarantees this by forcing the gradients to be explicit of a scalar field and, therefore, eliminating this drawback.

D. Energy-Based Models

Other energy-based techniques for generative models include the work by Kim and Bengio [51], who use directed graphs to learn densities in latent space and to train their generator. The approximation in this approach limits their generalization to complex and higher-dimensional datasets. Using kernel exponential families, Dai et al. [52] train a density estimator at the same time as their dual generator. Similar to other score-matching optimizations, their approach requires quadratic computations with respect to the input dimensions at each gradient calculation. Moreover, they only report generated results on 2-D toy examples. Other energy-based models include the works of Du and Mordatch [53] and Nijkamp et al. [54], where they generate samples using learned energy models. However, they require MCMC sampling of their energy model both during training and inference, which compromises efficiency. More similar to our work, Tao et al. [55] used a second parametrization and trained a generator using their energy model. These techniques require sample proposals for importance-weighting of their energy model estimation, which renders them inefficient compared to our approach. Moreover, we do not need negative proposal samples to train our DDEs and therefore are not prone to the quality of such proposals.

E. Generative Adversarial Models

GANs [4] are currently one of the most widely studied types of generative probabilistic models for very high-dimensional data. GANs are often difficult to train, however, and they can suffer from mode-collapse, sparking renewed interest in alternatives.

Similar to GANs, we use multiple neural networks to train a generator. However, our strategy is not based on a zero-sum game. In the original GANs [4], the generator is trained to minimize the Jensen–Shannon divergence between generated and real data distributions. Our model is optimized to minimize the KL-divergence instead, which has been shown to achieve better likelihood scores compared to GANs [56]. Moreover, we use the reverse KL-divergence loss in our training, which compared to forward KL-divergence, avoids saddle points when the two distributions have a small overlap. This is because minimizing the reverse KL-divergence can be formulated as

$$\arg\min_{\tilde{q}} D_{\mathrm{KL}}\left(\tilde{q}\,\|\,\tilde{p}\right) = \arg\max_{\tilde{q}} E_{x\sim\tilde{q}}\left[\log \tilde{p}(x)\right] + \mathcal{H}\left(\tilde{q}(x)\right)$$

which includes a term that attempts to maximize the entropy $\mathcal{H}$ of the generated distribution $\tilde{q}$. Wasserstein-GANs address the same issue by using the Wasserstein distance between the two distributions to formulate their loss. These models, however, require the discriminator network to guarantee Lipschitz continuity, which is imposed either by weight-clipping [57] or gradient penalty methods [58]. Our DDEs explicitly impose Gaussian smoothness on the data distribution, which guarantees that the density is nonzero everywhere and, therefore, stabilizes the training [59]. Additionally, the DDEs are trained to exactly constrain their gradients with respect to their inputs (5), without requiring additional techniques to control gradient magnitudes or weight clipping. Another important difference is that GANs' objectives are known to form a nonconservative vector field [60], [61], [62], which breaks the convergence guarantees when applying gradient-based optimization techniques. DDEs are models of scalar fields, which guarantee by construction that their gradients are conservative vector fields.

III. Background on Energy Estimation Using Score Matching

We start by first describing DAEs and second showing how we estimate a density using a variant of DAEs.

A. Denoising Autoencoders

Score-matching energy models were introduced by Kingma and LeCun [43], where they used Stein's objective. The noise-estimation (or denoising) objective was later shown to be equivalent to the score-matching objective by Vincent [26] and used as DAEs by Alain and Bengio [24]. DAEs allow us to obtain the gradient of the density, smoothed by a Gaussian kernel, which is equivalent to KDE [63]. Originally, the optimal DAE $r{:}\mathbb{R}^n \rightarrow \mathbb{R}^n$ [24], [26] is defined as the function minimizing the following denoising loss:

$$\mathcal{L}_{\mathrm{DAE}}\left(r;\, p, \sigma_\eta\right) = \mathbb{E}_{x,\eta\sim\mathcal{N}\left(0,\sigma_\eta^2\right)}\left[\|r(x+\eta) - x\|^2\right] \quad (1)$$

where the data x is distributed according to a density p over $\mathbb{R}^n$, and $\eta \sim \mathcal{N}(0, \sigma_\eta^2)$ represents n-dimensional, isotropic additive Gaussian noise with variance σ_η^2. It has been shown [27], [64], and [32] that the optimal DAE $r^*(x)$ minimizing $\mathcal{L}_{\mathrm{DAE}}$ can be expressed as follows, which is also known as Tweedie's formula:

$$r^*(x) = x + \sigma_\eta^2 \nabla_x \log \tilde{p}(x) \quad (2)$$

where ∇_x is the gradient with respect to the input x, $\tilde{p}(s) = [p * k](x)$ denotes the convolution result of the data and noise distributions $p(x)$, and $k = \mathcal{N}(0, \sigma_\eta^2)$. Inspired by this result, we reformulate the DAE-loss as a noise estimation loss

$$\mathcal{L}_{\mathrm{NEs}}\left(f;\, p, \sigma_\eta\right) = \mathbb{E}_{x,\eta\sim\mathcal{N}\left(0,\sigma_\eta^2\right)}\left[\|f(x+\eta) + \eta/\sigma_\eta^2\|^2\right] \quad (3)$$

[2]We have failed to train an MNIST generative model based on learning only the gradients (scores).

where $f{:}\mathbb{R}^n \to \mathbb{R}^n$ is a vector field that estimates the noise vector $-\eta/\sigma_\eta^2$. This is proportional to the objective in 1, where $r(x) = \sigma^2 f(x) + x$. Similar to [26] and [24], we formulate the following proposition and provide the proof in the Appendix.

Proposition 1: There is a unique minimizer $f^*(x) = \arg\min_f \mathcal{L}_{\text{NEs}}(f; p, \sigma_\eta)$ that satisfies

$$f^*(x) = \nabla_x \log \tilde{p}(x) = \nabla_x \log\big[p * k\big](x). \tag{4}$$

That is, the optimal estimator corresponds to the gradient of the logarithm of the Gaussian smoothed density $\tilde{p}(x)$, that is, the score of the density.

B. Denoising Density Estimators

A key observation is that the desired vector-field f^* is the gradient of a scalar function and is conservative. Hence, we can write the noise estimation loss in terms of a scalar function $s : \mathbb{R}^n \to \mathbb{R}$ instead of the vector field f, which we call the denoising density estimation loss

$$\mathcal{L}_{\text{DDE}}\big(s; p, \sigma_\eta\big) = \mathbb{E}_{x,\eta\sim\mathcal{N}\left(0,\sigma_\eta^2\right)}\Big[\big\|\nabla_x s(x+\eta) + \eta/\sigma_\eta^2\big\|^2\Big]. \tag{5}$$

A similar formulation has recently been proposed by Saremi and Hyv̈arinen [25]. Our terminology is motivated by the following corollary.

Corollary 1: The minimizer $s^*(x) = \arg\min_s \mathcal{L}_{\text{DDE}}(s; p)$ satisfies

$$s^*(x) = \log \tilde{p}(x) + C \tag{6}$$

with some constant $C \in \mathbb{R}$.

Proof: From Proposition 1 and the definition of $\mathcal{L}_{\text{DDE}}(s; p)$ we know that $\nabla_x s^*(x) = \nabla_x \log \tilde{p}(x)$, which leads immediately to the corollary. □

Following Hyv̈arinen and Dayan [65] and Raphan and Simoncelli [27], it is straightforward to show that the noise estimation objective will lead to a consistent DDE estimator.

In summary, we see that modifying the DAEs loss (1) into a noise estimation loss based on the gradients of a scalar function (5) allows us to derive a density estimator (Corollary 1), which we call the DDE.

Our parametrization of DDEs is different from prior works that use score-matching. We approximate the DDE using a neural network $s(x; \theta)$ trained using gradient-based optimizers. In other words, we look for a neural network s_w parameterized with weights w, such that $\nabla_{w_i}\nabla_x s_w(x)$ exists for all input x and network parameters w_i. We omit the subscript w in future references. This requires us to use neural network architectures (or other parameterization) that are twice differentiable everywhere, for example, with using SoftPlus activations. For illustration, Fig. 1 shows 2-D distribution examples, which we approximate using a DDE implemented as a multilayer perceptron (MLP).

We provide quantitative experiments for DDEs in Section V-F, where we make comparisons for estimating log-likelihoods on real datasets.

IV. Learning Generative Models Using DDEs

By leveraging DDEs, our key contribution is to formulate a novel training algorithm to obtain generators for given densities, which can be represented by a set of samples or as a given continuous function. In either case, we denote the smoothed data density $\tilde{p}$, which is obtained by training a DDE in case the input is given as a set of samples as described in Section III. We express our samplers using mappings $x = g(z)$, where $x \in R^n$, and $z \in R^m$ (usually $n > m$) is a latent variable, which typically has a standard normal distribution. In contrast to NFs, $g(z)$ does not need to be invertible. Let us denote the distribution of x induced by the generator as q, that is, $q \sim g(z)$, and also its Gaussian smoothed version $\tilde{q} = q * k$.

A. Optimizing a Generative Model Using Gradients

We obtain the generator by minimizing the KL divergence $D_{\text{KL}}(\tilde{q}||\tilde{p})$ between the density induced by the generator $\tilde{q}$ and the data density $\tilde{p}$. Our algorithm is based on the following observation.

Proposition 2: Given a scalar function $\Delta : \mathbb{R}^n \to \mathbb{R}$ that satisfies the following conditions.

$$\langle\tilde{q}, \log\tilde{q} - \log\tilde{p}\rangle > \langle\tilde{q}+\Delta, \log\big(\tilde{q}+\Delta\big) - \log\tilde{p}\rangle \tag{7}$$

$$\langle\Delta, \mathbb{1}\rangle = 0 \tag{8}$$

$$\Delta^2 < \epsilon, \quad \text{(pointwise exponentiation)} \tag{9}$$

then $D_{\text{KL}}(\tilde{q}||\tilde{p}) > D_{\text{KL}}(\tilde{q}+\Delta||\tilde{p})$ for small enough ϵ.

Proof: We will use the first-order approximation $\log(\tilde{q}+\Delta) = \log\tilde{q} + \Delta/\tilde{q} + O(\Delta^2)$, where the division is pointwise. Using $<\cdot,\cdot>$ to denote the inner product, we can write

$$\begin{aligned} D_{\text{KL}}\big(\tilde{q}+\Delta||\tilde{p}\big) &= \big\langle\tilde{q}+\Delta, \log\big(\tilde{q}+\Delta\big) - \log\tilde{p}\big\rangle \\ &= \big\langle\tilde{q}+\Delta, \log\tilde{q} + \Delta/\tilde{q} + O\big(\Delta^2\big) - \log\tilde{p}\big\rangle \\ &= \big\langle\tilde{q}, \log\tilde{q} - \log\tilde{p}\big\rangle + \big\langle\Delta, \log\tilde{q} - \log\tilde{p}\big\rangle \\ &\quad + \big\langle\tilde{q}, \Delta/\tilde{q}\big\rangle + \big\langle\Delta, \Delta/\tilde{q}\big\rangle + O\big(\Delta^2\big). \end{aligned} \tag{10}$$

This means

$$\begin{aligned} &D_{\text{KL}}\big(\tilde{q}+\Delta||\tilde{p}\big) - D_{\text{KL}}\big(\tilde{q}||\tilde{p}\big) \\ &\quad= \big\langle\Delta, \log\tilde{q} - \log\tilde{p}\big\rangle + \big\langle\tilde{q}, \Delta/\tilde{q}\big\rangle + \big\langle\Delta, \Delta/\tilde{q}\big\rangle + O\big(\Delta^2\big) < 0 \end{aligned} \tag{11}$$

because the first term on the right-hand side is negative [first assumption in (7)], the second term is zero [second assumption in (8)], and the third and fourth terms are quadratic in Δ and can be ignored for $\Delta < \epsilon$ when ϵ is small enough [third assumption in (9)]. □

The gradients of our KL-divergence objective satisfy the assumptions in the proposition with a small enough step size. Therefore, if we take steps using these gradients, we reduce the KL-divergence between the generated and real distributions $\tilde{q}$, $\tilde{p}$. Because KL-divergence is bounded from below by zero, then by iterating over these gradient steps our density $\tilde{q}$ will converge to the desired density $\tilde{p}$ with small enough steps. This requires that at any step of computing the gradient densities $\tilde{q}$ and $\tilde{p}$ to be optimal with respect to the data. $\tilde{p}$ can be estimated once and used during the optimization, but $\tilde{q}$

Algorithm 1 Training Steps for the Generator. The Input to the Algorithm Is a Pretrained Optimal DDE on Input Data $\log \tilde{p}(x)$ and a Learning Rate δ

1: Initialize generator parameters ϕ
2: Initialize DDE $s^{\tilde{q}} = \arg\min_s \mathcal{L}_{\text{DDE}}(s; q, \sigma_\eta)$ with $q \sim g(z; \phi), z \sim \mathcal{N}(0, 1)$
3: **while** not converged **do**
4: $\phi = \phi - \delta \nabla_\phi \mathbb{E}_{x=g(z;\phi)+\eta}[s^{\tilde{q}}(x) - \log \tilde{p}(x)]$, with $z \sim \mathcal{N}(0, 1), \eta \sim \mathcal{N}(0, \sigma_\eta^2)$
5: // $q \sim g(z; \phi)$ now indicates the updated density using the updated ϕ
6: $s^{\tilde{q}} = \arg\min_s \mathcal{L}_{\text{DDE}}(s; q, \sigma_\eta)$ // In practice, we only take few optimization steps
7: // $s^{\tilde{q}}$ is now the density (up to a constant) of $g(z; \phi)+\eta$
8: **end while**

changes after taking each gradient step. Therefore, we have to re-estimate $\tilde{q}$ after the gradient step by optimizing its objective, which we do using few steps in practice.

Proposition 2 shows that one can minimize KL divergence by taking a step Δ with given properties to change the generated distribution (that is the exact KL and not an upper bound). Note that the step Δ is with respect to the generated distribution and not the generator model parameters. Algorithm 1 shows our approach to finding the gradient steps to minimize the generator model and satisfy the conditions of the proposition for minimizing the divergence.

Based on the above observation, Algorithm 1 minimizes $D_{\text{KL}}(\tilde{q}||\tilde{p})$ by iteratively computing updated densities $\tilde{q} + \Delta$ that satisfy the conditions from Proposition 2, hence $D_{\text{KL}}(\tilde{q}||\tilde{p}) > D_{\text{KL}}(\tilde{q} + \Delta||\tilde{p})$. This is guaranteed to converge to a global minimum because $D_{\text{KL}}(\tilde{q}||\tilde{p})$ is convex in $\tilde{q}$.

At the beginning of each iteration in Algorithm 1 (line 3), by definition q is the density obtained by sampling our generator $x = g(z; \phi), z \sim \mathcal{N}(0, 1)$ (n-dimensional standard normal distribution), and the generator is a neural network with parameters ϕ. In addition, $\tilde{q} = q * k$ is defined as the density obtained by sampling $x = g(z; \phi) + \eta, z \sim \mathcal{N}(0, 1), \eta \sim \mathcal{N}(0, \sigma_\eta^2)$. Finally, the DDE $s^{\tilde{q}}$ correctly estimates $\tilde{q}$, that is, $\log \tilde{q}(x) = s^{\tilde{q}}(x) + C$. In each iteration, we update the generator such that its density is changed by a small Δ that satisfies the conditions from Proposition 2. We achieve this by computing a gradient descent step of $\mathbb{E}_{x=g(z;\phi)+\eta}[s^{\tilde{q}}(x) - \log \tilde{p}(x)] + C$ with respect to the generator parameters ϕ (line 4). A small enough learning rate guarantees that condition one (7) in Proposition 2 is satisfied. The second condition (8) is satisfied because we update the distribution by updating its generator, and the third condition (9) is also satisfied under a small enough learning rate (and assuming the generator network is Lipschitz continuous). After updating the generator, we update the DDE to correctly estimate the new density produced by the updated generator (line 6). Note that, in practice, we perform a fixed number of iterations (5)-ten steps similar to GANs) to optimize the DDE, which did not lead to any instabilities.

Note that it is crucial in the first step in the iteration in Algorithm 1 that we sample using $g(z; \phi)+\eta$ and not $g(z; \phi)$. This allows us, in the second step, to use the updated $g(z; \phi)$ to train a DDE $s^{\tilde{q}}$ that exactly (up to a constant) matches the density generated by $g(z; \phi)+\eta$. Even though in this approach, we only minimize the KL divergence with the "noisy" input density $\tilde{p}$, the sampler $g(z; \phi)$ still converges to a sampler of the underlying density p in theory (exact sampling).

B. Normalizing Constant C in the DDE Estimator

The constant C depends on the parameters of DDE because it indicates the drift of our estimate from the partition function. However, this constant is independent of the generator parameters as it can vary for a fixed generator parametrization, without influencing the DDE loss. The key result is that this constant can be ignored during the training of the generator because it does not influence the gradients of the loss wrt the generator parameters. Therefore, the estimated $\tilde{q}$ always integrates to one after any generator update in Algorithm 1.

C. Consistency of the Distribution Estimation

DDE s is a consistent estimator (Section III-B), and we assume that $\tilde{p}$ and $\tilde{q}$ remain optimal during the training of the generator (Section IV-A). Following the law of large numbers, as the number of samples increases, the expected KL divergence loss converges asymptotically to the true KL divergence between the two distributions. Therefore, the generated distribution will converge to the true distribution. Assuming the existence of a unique solution for the generator, the generator has weak consistency. The generator will have strong consistency assuming the compactness of its parameter space. We refer the reader to Wald [66] for a formal proof of the consistency with assumptions.

D. Exact Sampling

Our objective involves reducing the KL divergence between the Gaussian smoothed generated density $\tilde{q}$ and the data density $\tilde{p}$. This also implies that the density q obtained from sampling the generator $g(z; \phi)$ is identical with the data density p, without Gaussian smoothing, which can be expressed as the following corollary:

Corollary 2: Let $\tilde{p}$ and $\tilde{q}$ be related to densities p and q, respectively, via convolutions using a Gaussian k, that is, $\tilde{p} = p * k, \tilde{q} = q * k$. Then the smoothed densities $\tilde{p}$ and $\tilde{q}$ are the same if and only if the data density p and the generated density q are the same.

This follows immediately from the convolution theorem and the fact that the Fourier transform of Gaussian functions is nonzero everywhere, that is, Gaussian blur is invertible.

E. Gradient Computation Complexity

The second-order derivatives needed to solve the DDE objective in (5) are only wrt the DDE input x and not its parameters. This is unlike approaches where one needs the Hessian wrt the parameters (e.g., in some normalizing flow techniques). A similar operation is used for the "gradient penalty" in the WGAN-GP model where the model is regularized by the norm of its gradients wrt. the input. In practice,

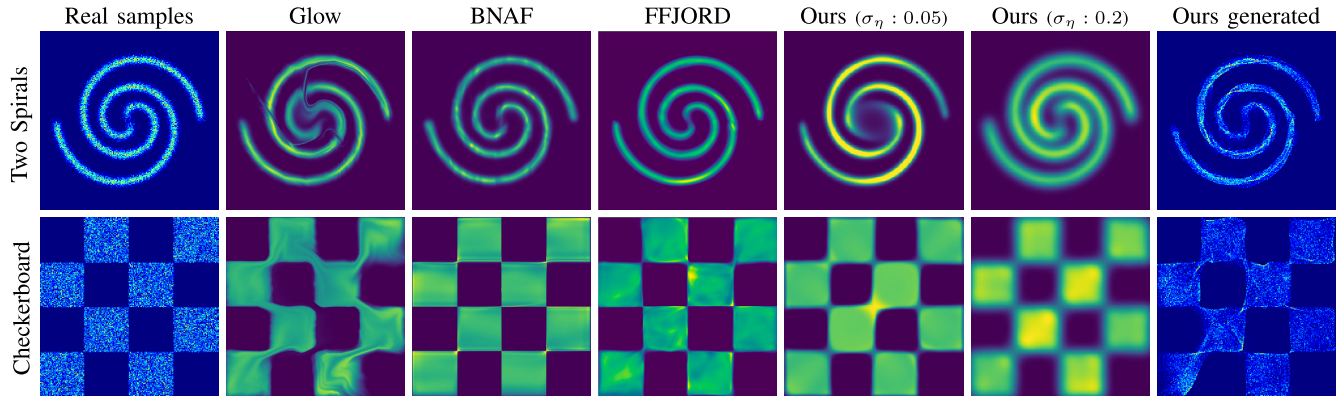

Fig. 1. Density estimation in 2-D, showing that we can accurately capture these densities with few visual artifacts. The rightmost column shows samples generated using our generative model training.

this costs no more than $O(2N)$ and is negligible in the training of the DDEs. Additionally, the second-order derivatives do not take part in the gradient step in Algorithm 1 (line 4) where we take gradients of DDE output and not their loss objective.

V. Experiments

We include the results of our generative sampling approach and compare quantitative results with state-of-the-art models. We also show the competitiveness of our DDE density estimation approach and provide evaluation on real datasets.

A. Two-Dimensional Dataset Comparisons

Similar to Grathwohl et al. [18], we perform experiments for 2-D density estimation and visualization over three datasets. Additionally, we learn generative models. For our DDE networks, we used MLPs with residual connections. All networks have 25 layers, each with 32 channels and Softplus activation. For training, we use 2048 samples per iteration to estimate the expected values. Fig. 1 shows the comparison of our method with Glow [16], BNAF [67], and FFJORD [18]. Our DDEs can estimate the density accurately and capture the underlying complexities of each density. Due to inherent smoothing as in KDE, our method induces a small blur to the density compared to BNAF. To demonstrate this effect, we show DDEs trained with both small and large noise standard deviations $\sigma_\eta = 0.05$ and $\sigma_\eta = 0.2$. However, our DDE can estimate the density coherently through the data domain, whereas BNAF produces noisy approximation across the data (these artifacts are visible in close-up zoom of two spirals data in Fig. 1).

Generator training and sampling are demonstrated in Fig. 1 on the right. The sharp edges of the checkerboard samples imply that, due to the invertibility of a small Gaussian blur, the generator learns to sample from the sharp target density even though the DDEs estimate noisy densities. While the generator update in theory requires DDE networks to be optimal at each gradient step, we take a limited number of ten DDE gradient descent steps for each generator update to accelerate convergence.

B. Stability of Convergence

Alternating optimization with a few steps of training for DDE in the Algorithm 1 might lead to some instabilities.

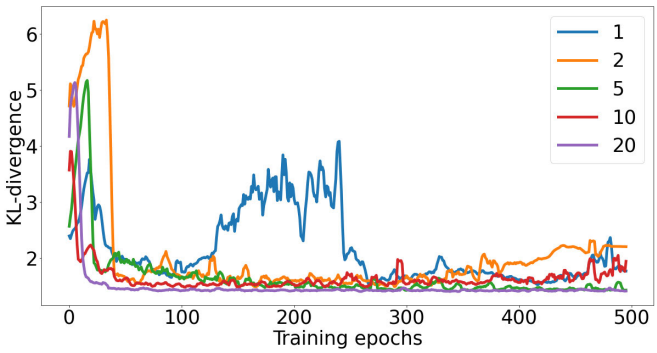

Fig. 2. Convergence of the generator model with respect to varying DDE optimization steps for generating a 1-D data distribution (a symmetric mixture of eight Gaussian). We show the KL-divergence of the generator network parameters during the training steps for each model trained with a different number of inner DDE optimization steps (data are smoothed for clarity using a moving average filter with a width of five iterations).

We have performed an ablation test to analyze the best tradeoff for the number of inner iterations of DDE. We have synthesized a 1-D data distribution using a mixture of eight Gaussian densities (the common eight Gaussians 2-D dataset projected to 1-D). And we tested the convergence of Algorithm 1 by varying the number of inner optimizations for the DDE optimization.

Fig. 2 bottom shows the results of these optimizations where we tested the algorithm for 1, 2, 5, 10, and 20 inner iterations. We show the KL-divergence as a means of understanding the convergence and possible instabilities during training. Note that time scales proportionally with respect to the number of inner iterations of DDE training. All models reduce the KL-divergence. The cases with 1 and 2 inner iterations are fast but have instabilities in the earlier iterations. The cases with 10 and 20 inner iterations take proportionally longer time to estimate the correct density. We can see that using five inner iterations leads to a good tradeoff in terms of convergence stability, quality, and time.

C. MNIST Dataset

Fig. 3 illustrates our generative training on MNIST [68] using Algorithm 1. We use a dense block architecture with fully connected layers here and refer to the Appendix for

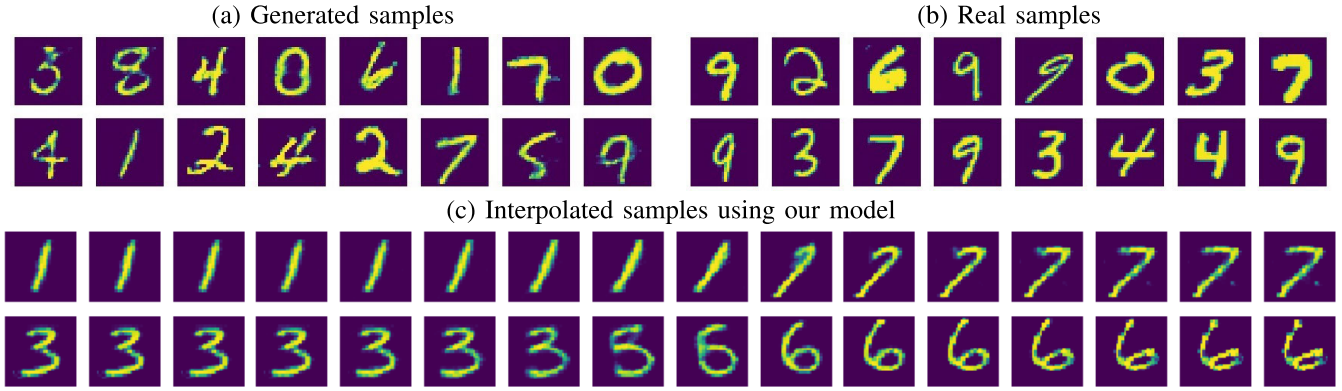

Fig. 3. Generated MNIST (a), from the dataset (b), and latent space interpolation (c).

Fig. 4. Generated results on 32 × 32 images from the CelebA dataset [70]. (a) Generated samples. (b) Real samples.

the network and training details, including additional results for Fashion-MNIST [69]. Fig. 3 shows qualitatively that our generator can replicate the underlying distributions. In addition, latent-space interpolation demonstrates that the network learns an intuitive and interpretable mapping from normally distributed latent variables to samples of the data distribution.

D. CelebA Dataset

Fig. 4 shows additional experiments on the CelebA dataset [70]. The images in the dataset have 32 × 32 × 3 dimensions and we normalize the pixel values to be in the range $[-0.5, 0.5]$. To show the flexibility of our algorithm with respect to neural network architectures, here we use a style-based generator [71] architecture for our generator network. Refer to the Appendix for network and training details. Fig. 4 shows that our approach can produce natural-looking images, and the model has learned to replicate the global distribution with a diverse set of images and different characteristics.

E. Quantitative Evaluation With Stacked-MNIST

We perform a quantitative evaluation of our approach based on the synthetic Stacked-MNIST [72] dataset, which was designed to analyze mode-collapse in generative models. The dataset is constructed by stacking three randomly chosen digit images from MNIST to generate samples of size 28 × 28 × 3. This augments the number of classes to 10^3, which are considered distinct modes of the dataset. Mode-collapse can be quantified by counting the number of nodes generated by a model. Additionally, the quality of the distribution can be measured by computing the KL-divergence between the generated class distribution and the original dataset, which has a uniform distribution in terms of class labels. Similar to prior work [72], we use an external classifier to measure the number of classes that each generator produces by separately inferring the class of each channel of the images.

Fig. 5 reports the quantitative results for this experiment by comparing our method with well-tuned GAN models. DCGAN [73] implements a basic GAN training strategy using a stable architecture. WGAN uses the Wasserstein distance [57], and WGAN+GP includes a gradient penalty to regularize the discriminator [58]. For a fair comparison, all methods use the DCGAN network architecture. Since our method requires two DDE networks, we have used fewer parameters in the DDEs so that in total we preserve the same number of parameters and capacity as the other methods. For each method, we generate batches of 512 samples per training iteration and count the number of classes within each batch (i.e., the maximum number of different labels in each batch is 512). We also plot the reverse KL-divergence to the uniform ground-truth class distribution. Using the two measurements we can see how well each method replicates the distribution in terms of diversity and balance. Without fine-tuning and changing the capacity of our network models, our approach is comparable to modern GANs such as WGAN

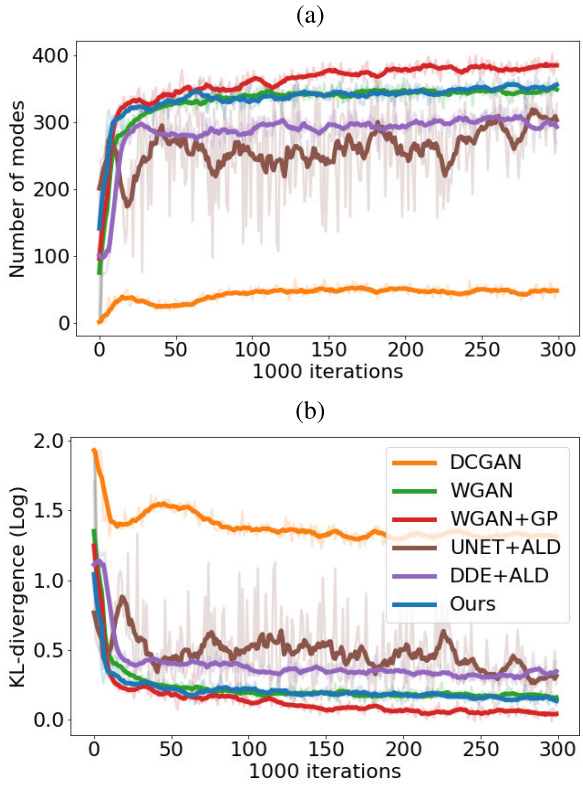

Fig. 5. Mode-collapse experiment results on Stacked-MNIST as a function of training iterations (for discriminator or DDE). (a) Number of generated modes per batch of size 512. (b) Reverse KL-divergence between the generated and the data distribution in the logarithmic domain.

and WGAN+GP, which outperform DCGAN by a large margin in this experiment.

We also report results for sampling techniques based on Score-Matching. We trained a noise conditional score network (NCSN) parameterized with a UNET architecture [74], which is then followed by a sampling algorithm using the annealed Langevin dynamics (ALD) as described by Song and Ermon [34]. We refer to this method as UNET+ALD. We also implemented a model based on our approach called DDE+ALD, where we used our DDE network in combination with iterative Langevin sampling. While our training loss is equivalent to the score-matching objective, the DDE network outputs a scalar and explicitly enforces the score to be a conservative vector field by computing it as the gradient of its scalar output. DDE+ALD uses the spatial gradient of the DDE for iterative sampling with ALD [34], instead of our proposed direct, one-step generator as described in Section IV. We observe that DDE+ALD is more stable compared to the UNET+ALD baseline, even though the UNET achieves a lower loss during training. We believe that this is because DDEs guarantee conservativeness of the distribution gradients (i.e., scores), which leads to more diverse and stable data generation as we see in Fig. 5. Furthermore, our approach with direct sampling outperforms both UNET+ALD and DDE+ALD.

F. Real Data Density Estimation

In this section, we evaluate the DDE models, excluding the generative models, and compare their performance in learning the densities and generalize to test sets for estimating log-likelihoods. We follow the experiments in BNAF [67] for density estimation, which includes the POWER, GAS, HEPMASS, and MINIBOON datasets [75]. Since DDEs estimate densities up to their normalizing constant, we approximate the constant using Monte Carlo estimation here. Similarly, Li et al. [44] use sampling to estimate the normalizing constant. We show average log-likelihoods over test sets and compare them to state-of-the-art methods for normalized density estimation in Table II. Average log-likelihood refers to the average density values of the test set (in log-domain) and measures the generalization performance of a density estimation model, where higher indicates better fitting to the test set. We have omitted the results of the BSDS300 dataset [76] since we could not estimate the normalizing constant reliably (due to the high dimensionality of the data).

In this experiment, we only use the DDE trained on the training dataset to evaluate the density, and we do not train a second DDE or a generator. To train our DDEs, we used MLPs with residual connections between each layer. All networks have 25 layers, with 64 channels and Softplus activation, except for GAS and HEPMASS, which employ 128 channels. We trained the models for 400 epochs using a learning rate of 2.5×10^{-4} with linear decay with a scale of 2 every 100 epochs. Similarly, we started the training by using noise standard deviation $\sigma_\eta = 0.1$ and decreased it linearly with the scale of 1.1 up to a dataset-specific value, which we set to 5×10^{-2} for POWER, 4×10^{-2} for GAS, 2×10^{-2} for HEPMASS, and 0.15 for MINIBOON.

For evaluation, we use the original, noise-free, test dataset. We estimate the normalizing constant via importance sampling using a Gaussian distribution with the mean and variance of the DDE input distribution. We average five estimations using 51 200 samples each (we used ten times more samples for GAS), and we indicate the variance of this average in Table II. The average log-likelihood results indicated in Table II shows better generalization using our trained DDEs for high-dimensional datasets HEPMASS and MINIBOON, and competitive values for POWER and GAS datasets.

As indicated by the results in the table, DKEF [44] (2019) does not generalize well compared to DDEs. While both approaches use score-matching for density estimation, their use of Stein's objective makes their estimator only asymptotically equal to our noise-estimation loss for learning the unnormalized KDEs. Moreover, they require other smoothness terms in their optimization that adds more bias to the estimation.

G. Discussion on Parametrization of KDEs Using Neural Networks

Nonparametric KDEs are known to perform poorly for high-dimensional data [82], [83]. This is mostly due to larger distances in higher dimensions and failure to calculate accurate kernel weights for samples in the database. Recent nonparametric models that use neural networks [10], [11] can still perform on datasets with around 10-D. In our experiments, DDEs where successfully trained on CelebA image datasets

TABLE II

AVERAGE LOG-LIKELIHOOD COMPARISON IN FOUR DATASETS [75]. THE TOP ROWS INCLUDES DATASET SIZE AND DIMENSIONALITY, BOTTOM ROWS ARE NORMALIZED BY SAMPLING. THE UPPER SECTION INCLUDES METHODS THAT ESTIMATE NORMALIZED DENSITIES. RESULTS OF LI ET AL. [44] ARE READ FROM THE BAR PLOTS REPORTED IN THEIR ARTICLE. BEST PERFORMANCES ARE IN BOLD

Model	POWER $d=6, N\approx 2M$	GAS $d=8, N\approx 1M$	HEPMASS $d=21, N\approx 500K$	MINIBOON $d=43, N\approx 36K$
RealNVP, Dinh et al. [14] (2017)	0.17 $\pm.01$	8.33 $\pm.14$	-18.71 $\pm.02$	-13.55 $\pm.49$
Glow, Kingma and Dhariwal [16] (2018)	0.17 $\pm.01$	8.15 $\pm.40$	-18.92 $\pm.08$	-11.35 $\pm.07$
MADE MoG, Germain et al. [77]* (2015)	0.40 $\pm.01$	8.47 $\pm.02$	-15.15 $\pm.02$	-12.27 $\pm.47$
MAF-affine, Papamakarios et al. [78] (2017)	0.24 $\pm.01$	10.08 $\pm.02$	-17.73 $\pm.02$	-12.24 $\pm.45$
MAF-affine MoG, Papamakarios et al. [78] (2017)	0.30 $\pm.01$	9.59 $\pm.02$	-17.39 $\pm.02$	-11.68 $\pm.44$
FFJORD, Grathwohl et al. [18] (2019)	0.46 $\pm.01$	8.59 $\pm.12$	-14.92 $\pm.08$	-10.43 $\pm.04$
NAF-DDSF, Huang et al. [15] (2018)	0.62 $\pm.01$	11.96 $\pm.33$	-15.09 $\pm.40$	-8.86 $\pm.15$
TAN Oliva et al. [79] (2018)	0.60 $\pm.01$	**12.06** $\pm.02$	-13.78 $\pm.02$	-11.01 $\pm.48$
BNAF, De Cao et al. [67] (2019)	0.61 $\pm.01$	**12.06** $\pm.09$	-14.71 $\pm.38$	-8.95 $\pm.07$
nMDMA, Gilboa et al. [80] (2021)	**1.78** $\pm.12$	8.43 ±0.04	-18.0 $\pm.91$	-18.6 ±0.47
AdaCat, Li et al. [81] (2022)	0.56 $\pm NA$	11.27 $\pm NA$	-18.17 $\pm NA$	-14.14 $\pm NA$
DKEF, Li et al. [44] (2019)	-	-	≤ -20	≤ -40
Ours	0.97 $\pm.18$	9.73 ±1.14	**-11.3** $\pm.16$	**-6.94** ±1.81

with 3000 dimensions. In our experiments, we observe that our model can mitigate this problem using the neural network parametrization with no instabilities during training and reliable results while testing. We know that neural networks can generalize well due to their implicit regularities. This will in turn make more stable density estimates for unlikely samples. Moreover, in our model, we are not computing the KDE explicitly using our model, but only regressing to its estimate using out network and ignoring the normalizing constant. This property also allows us to mitigate further issues regarding the overflow of floating point variables and therefore instabilities of nonparametric methods.

The DDE model, however, is constrained to output a single scalar value, that is the log-probability. We enforce this constraint in the network architecture by setting the last layer to map its features to one output value. This, in turn, makes the model very restrictive in capacity. Note that this is not the case as in the discriminator in GANs, where the discriminator must only learn the difference between the two distributions, whereas, in the case of DDEs, they must learn the full data distribution and its complexities. Therefore, DDEs require models with much higher capacity, in practice, with the increasing dimensions of the input domain. An alternative would be to use product-of-experts to reduce this effect as in the work of Saremi and Hyv̈arinen [25].

H. Discussion and Future Work

Our approach relies on a key hyperparameter σ_η that determines the training noise for the DDE, which we currently set manually. In the future, we will investigate strategies to determine this parameter in a data-dependent manner.

Another challenge is to obtain high-quality results using complex, higher-dimensional data such as CIFAR or high-resolution images. In practice, one strategy is to combine our approach with latent embedding learning methods [84], in a similar fashion as proposed by Hoshen et al. [85]. The robustness of our technique with very high-dimensional data could potentially also be improved by leveraging slicing techniques [86], [87]. We use three networks to learn a generator (a DDE each for the input and generated data, and the generator) and it would be more efficient if one could use a single model for estimating both DDEs. Finally, our generator training approach is independent of the type of density estimator, and techniques other than DDEs could also be used.

VI. CONCLUSION

We presented a novel approach to learning generative models using DDEs, and our theoretical analysis proves that our training algorithm converges consistently to a unique optimum. Furthermore, our technique does not require specific neural network architectures or ODE integration. A quantitative evaluation using the stacked MNIST dataset shows that our approach avoids mode collapse similarly to state-of-the-art Wasserstein GANs. Finally, our DDE parameterization achieves state-of-the-art results on a standard log-likelihood evaluation benchmark compared to recent techniques based on NFs, continuous flows, and autoregressive models, and we demonstrate successful generators on diverse image datasets.

APPENDIX

A. Proof of Score Matching via Noise Estimation

This is a proof for Proposition 1 in the main article.

Proof: Clearly, $\mathcal{L}_{\text{NEs}}$ is convex in f hence the minimizer is unique. We can rewrite the noise estimation loss from (3) as

$$\mathcal{L}_{\text{NEs}}\big(f;\, p, \sigma_\eta\big) = \int_{\mathbb{R}^n} \mathbb{E}_{\eta\sim\mathcal{N}(0,\sigma_\eta^2)}\big[p(x)\|f(x+\eta)+\eta/\sigma_\eta^2\|^2\big]dx \tag{12}$$

which we minimize with respect to the vector-valued function $f:\mathbb{R}^n\to\mathbb{R}^n$. Substituting $\tilde{x}=x+\eta$ yields

$$\mathcal{L}_{\text{NEs}}\big(f;\, p, \sigma_\eta\big) = \int_{\mathbb{R}^n} \mathbb{E}_{\eta\sim\mathcal{N}(0,\sigma_\eta^2)}\big[p(\tilde{x}-\eta)\|f(\tilde{x})+\eta/\sigma_\eta^2\|^2\big]d\tilde{x}. \tag{13}$$

We can minimize this with respect to $f(\tilde{x})$ by differentiating and setting the derivative to zero, which leads to

$$\mathbb{E}_{\eta\sim\mathcal{N}(0,\sigma_\eta^2)}\big[p(\tilde{x}-\eta)f(\tilde{x})\big] = -\frac{1}{\sigma_\eta^2}\mathbb{E}_{\eta\sim\mathcal{N}(0,\sigma_\eta^2)}\big[p(\tilde{x}-\eta)\eta\big] \tag{14}$$

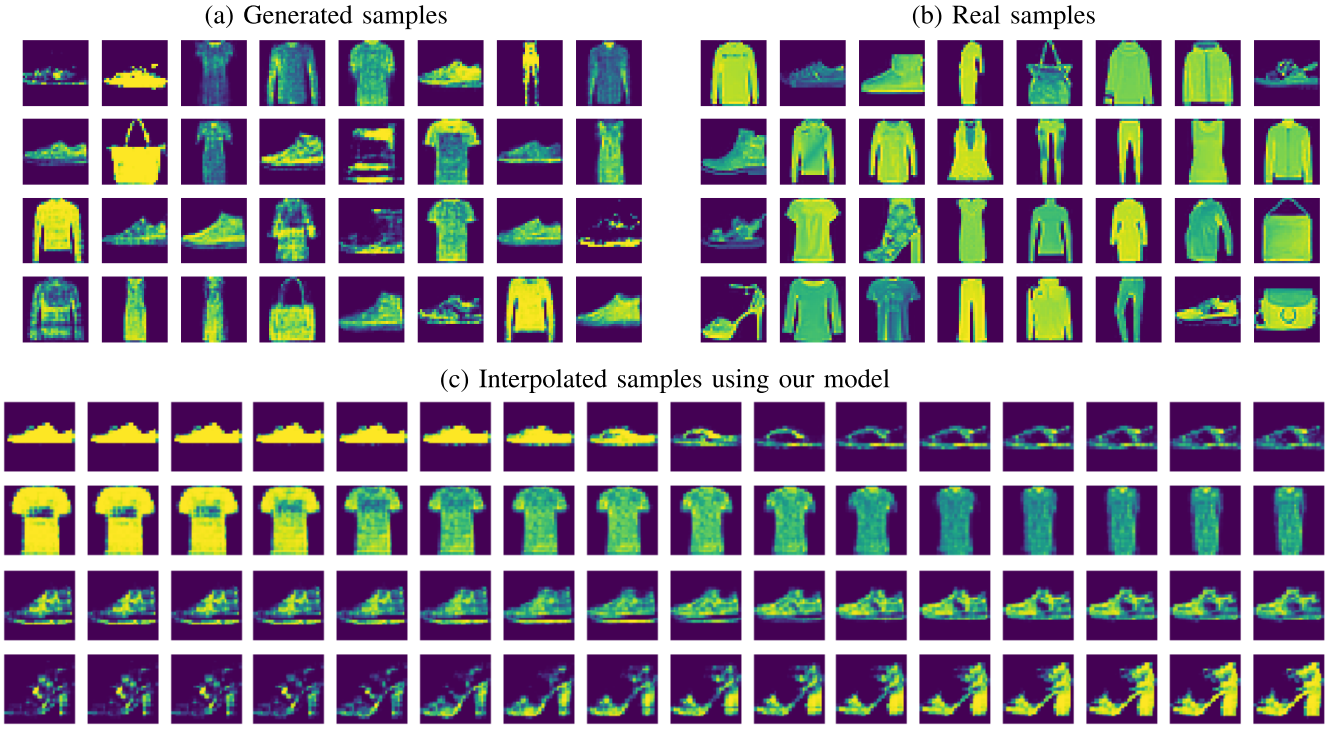

Fig. 6. Fashion-MNIST samples from our generator (a), the dataset (b), and latent space interpolation using our generator (c).

and hence

$$f(\tilde{x}) = -\frac{1}{\sigma_\eta^2}\frac{\mathbb{E}_{\eta\sim\mathcal{N}(0,\sigma_\eta^2)}\big[p(\tilde{x}-\eta)\eta\big]}{\mathbb{E}_{\eta\sim\mathcal{N}(0,\sigma_\eta^2)}\big[p(\tilde{x}-\eta)\big]} \tag{15}$$

$$= \nabla_{\tilde{x}} \log\big[p * k\big](\tilde{x}) = \nabla_{\tilde{x}} \log \tilde{p}(\tilde{x}) \tag{16}$$

which follows from basic calculus and has also been used by Raphan and Simoncelli [27]. □

B. Visual Results on Fashion-MNIST

For the experiments on MNIST and Fashion-MNIST, we used the Dense Block architecture [88] with 15 fully connected layers and 256 additional neurons each. The last layer of the network maps all its inputs to one value, which we train to approximate the density of input images. For the generator network, we used Dense Blocks with 15 fully connected layers and 256 additional neurons each. The last layer maps all outputs to the image size of $28 \times 28 = 784$. For the input of the generator, we used noise with a 16-D standard normal distribution. In addition, the DDEs were trained with noise standard deviation $\sigma_\eta = 0.5$, where pixel values were scaled to range between 0 and 1.

In addition to the MNIST results, here we include visual results on the Fashion-MNIST dataset, where we have used the exact setup as in our experiments on MNIST for training our generator. Fig. 6 shows our generated images and interpolations in the latent space of Fashion-MNIST.

C. Network and Training Details for Experiments on CelebA

For our experiments on CelebA, we use a style-based generator [71] architecture. We use Swish activations [89] in all hidden layers of our networks except for their last layer, which we set to be linear. Additionally, we normalized each output of the generator to be in the accepted range $[-0.5, 0.5]$. We used equalized learning rate [90] with a learning rate 5×10^{-3} for the DDEs, and a slightly lower learning rate for the generator 3×10^{-3}. We trained our DDEs using $\sigma_\eta = 0.5$ and set the truncation parameter in the style-based generator to $\phi = 0.7$ when feeding the generator with random noise [71] at test time.

D. Network Models and Training for Stacked-MNIST Experiment

In our experiments with Stacked-MNIST, our generative networks are trained using a learning rate of 2×10^{-2}, the Adam optimizer with $\beta_1 = 0.9$, and the generator updates took place after every 10th DDE step. We use standard parameters for the other methods (DCGAN, WGAN, and WGAN+GP), including a learning rate of 2×10^{-4}, the Adam optimizer with $\beta_1 = 0.5$, and we trained the generator every fifth iteration of the discriminator training.

The NCSN models are trained to remove Gaussian noise at ten different noise standard deviations within the range $[1.0, 0.01]$ (geometric interpolation). The input to the NCSN models includes also the noise level. To further improve the quality of the networks, we use separate last layers for each noise standard deviation for training and testing. This way we can increase the capacity of the network significantly, while we keep the same order of parameters as in the other methods. We used the Adam optimizer with original parameters and a learning rate of 1×10^{-4}.

REFERENCES

[1] T. Portenier, Q. Hu, A. Szabó, S. A. Bigdeli, P. Favaro, and M. Zwicker, "FaceShop: Deep sketch-based face image editing," 2018, *arXiv:1804.08972*.

[2] Z. Yang, T. Zhang, I. S. Bozchalooi, and E. Darve, "Memory-augmented generative adversarial networks for anomaly detection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 6, pp. 2324–2334, Jun. 2022.

[3] X.-Y. Liu and X. Wang, "Real-time indoor localization for smartphones using tensor-generative adversarial nets," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3433–3443, Aug. 2021.

[4] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014.

[5] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," in *Proc. 11th Int. Conf. Artif. Intell. Statist.*, 2009, pp. 448–455.

[6] M. Welling and Y. W. Teh, "Approximate inference in Boltzmann machines," *Artif. Intell.*, vol. 143, no. 1, pp. 19–50, Jan. 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370202003612

[7] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.

[8] L. Dinh, D. Krueger, and Y. Bengio, "NICE: Non-linear independent components estimation," 2014, *arXiv:1410.8516*.

[9] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 6840–6851.

[10] Y. Nakamura and O. Hasegawa, "Nonparametric density estimation based on self-organizing incremental neural network for large noisy data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 1, pp. 8–17, Jan. 2017.

[11] B. Fang, S. Chen, and Z. Dong, "Density distillation for fast nonparametric density estimation," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 1, 2022, doi: 10.1109/TNNLS.2022.3160939.

[12] M. E. Abbasnejad, Q. Shi, A. van den Hengel, and L. Liu, "A generative adversarial density estimator," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10774–10783.

[13] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proc. 32nd Int. Conf. Mach. Learn.*, vol. 37, F. Bach and D. Blei, Eds. Lille, France, Jul. 2015, pp. 1530–1538. [Online]. Available: http://proceedings.mlr.press/v37/rezende15.html

[14] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," in *Proc. ICLR*, 2017, pp. 1–32.

[15] C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville, "Neural autoregressive flows," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, J. Dy and A. Krause, Eds. Stockholmsmässan, Stockholm Sweden, Jul. 2018, pp. 2078–2087. [Online]. Available: http://proceedings.mlr.press/v80/huang18d.html

[16] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1×1 convolutions," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[17] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[18] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "FFJORD: Free-form continuous dynamics for scalable reversible generative models," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–13.

[19] K. Li and J. Malik, "Implicit maximum likelihood estimation," 2018, *arXiv:1809.09087*.

[20] J. Liu, M. Gong, and H. He, "Deep associative neural network for associative memory based on unsupervised representation learning," *Neural Netw.*, vol. 113, pp. 41–53, May 2019.

[21] J. Liu, W. Zhang, F. Liu, and L. Xiao, "Deep associative learning for neural networks," *Neurocomputing*, vol. 443, pp. 222–234, Jul. 2021.

[22] J. Liu, W. Zhang, F. Liu, and L. Xiao, "A probabilistic model based on bipartite convolutional neural network for unsupervised change detection," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 4701514.

[23] A. Van den Oord et al., "Conditional image generation with PixelCNN decoders," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016.

[24] G. Alain and Y. Bengio, "What regularized auto-encoders learn from the data-generating distribution," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3743–3773, 2014. [Online]. Available: http://jmlr.org/papers/v15/alain14a.html

[25] S. Saremi and A. Hyvärinen, "Neural empirical Bayes," 2019, *arXiv:1903.02334*.

[26] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural Comput.*, vol. 23, no. 7, pp. 1661–1674, Jul. 2011, doi: 10.1162/NECO_a_00142.

[27] M. Raphan and E. P. Simoncelli, "Least squares estimation without priors or supervision," *Neural Comput.*, vol. 23, no. 2, pp. 374–420, Feb. 2011, doi: 10.1162/NECO_a_00076.

[28] C. Kaae Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár, "Amortised MAP inference for image super-resolution," 2016, *arXiv:1610.04490*.

[29] E. T. Reehorst and P. Schniter, "Regularization by denoising: Clarifications and new interpretations," *IEEE Trans. Comput. Imag.*, vol. 5, no. 1, pp. 52–67, Mar. 2019.

[30] R. Laumont, V. D. Bortoli, A. Almansa, J. Delon, A. Durmus, and M. Pereyra, "Bayesian imaging using plug & play priors: When Langevin meets tweedie," *SIAM J. Imag. Sci.*, vol. 15, no. 2, pp. 701–737, Jun. 2022.

[31] S. A. Bigdeli, M. Zwicker, P. Favaro, and M. Jin, "Deep mean-shift priors for image restoration," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 763–772.

[32] S. Arjomand Bigdeli and M. Zwicker, "Image restoration using autoencoding priors," 2017, *arXiv:1703.09964*.

[33] S. Bigdeli, D. Honzátko, S. Süsstrunk, and L. A. Dunbar, "Image restoration using plug-and-play CNN MAP denoisers," 2019, *arXiv:1912.09299*.

[34] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," 2019, *arXiv:1907.05600*.

[35] B. Dai, Z. Liu, H. Dai, N. He, A. Gretton, L. Song, and D. Schuurmans, "Exponential family estimation via adversarial dynamics embedding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 10977–10988.

[36] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," 2020, *arXiv:2011.13456*.

[37] Z. Kadkhodaie and E. Simoncelli, "Stochastic solutions for linear inverse problems using the prior implicit in a denoiser," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 13242–13254.

[38] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2256–2265.

[39] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," 2020, *arXiv:2010.02502*.

[40] Z. Kadkhodaie and E. P. Simoncelli, "Solving linear inverse problems using the prior implicit in a denoiser," 2020, *arXiv:2007.13640*.

[41] G. Daras, M. Delbracio, H. Talebi, A. G. Dimakis, and P. Milanfar, "Soft diffusion: Score matching for general corruptions," 2022, *arXiv:2209.05442*.

[42] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," 2022, *arXiv:2206.00364*.

[43] D. P. Kingma and Y. LeCun, "Regularized estimation of image statistics by score matching," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1126–1134.

[44] W. Li, D. J. Sutherland, H. Strathmann, and A. Gretton, "Learning deep kernels for exponential family densities," in *Proc. ICML*, 2019.

[45] Y. Li and R. E. Turner, "Gradient estimators for implicit models," 2017, *arXiv:1705.07107*.

[46] D. Wang, X. Qin, F. Song, and L. Cheng, "Stabilizing training of generative adversarial nets via Langevin stein variational gradient descent," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 7, pp. 2768–2780, Jul. 2022.

[47] C. Tao et al., "Variational annealing of GANs: A Langevin perspective," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6176–6185.

[48] Q. Liu and D. Wang, "Stein variational gradient descent: A general purpose Bayesian inference algorithm," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–9.

[49] Y. Feng, D. Wang, and Q. Liu, "Learning to draw samples with amortized stein variational gradient descent," 2017, *arXiv:1707.06626*.

[50] Q. Liu, J. Lee, and M. Jordan, "A kernelized stein discrepancy for goodness-of-fit tests," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 276–284.

[51] T. Kim and Y. Bengio, "Deep directed generative models with energy-based probability estimation," 2016, *arXiv:1606.03439*.

[52] B. Dai, H. Dai, A. Gretton, L. Song, D. Schuurmans, and N. He, "Kernel exponential family estimation via doubly dual embedding," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 2321–2330.

[53] Y. Du and I. Mordatch, "Implicit generation and generalization in energy-based models," 2019, *arXiv:1903.08689*.

[54] E. Nijkamp, M. Hill, S.-C. Zhu, and Y. N. Wu, "Learning non-convergent non-persistent short-run MCMC toward energy-based model," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.

[55] C. Tao et al., "On Fenchel mini-max learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–13.

[56] S. Nowozin, B. Cseke, and R. Tomioka, "F-GAN: Training generative neural samplers using variational divergence minimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 271–279.

[57] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, D. Precup and Y. W. Teh, Eds. Aug. 2017, pp. 214–223. [Online]. Available: http://proceedings.mlr.press/v70/arjovsky17a.html

[58] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5767–5777.

[59] S. Jenni and P. Favaro, "On stabilizing generative adversarial training with noise," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12137–12145.

[60] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2234–2242.

[61] F. Huszar. (Oct. 2017). *GANs are Broken in More Than One Way: The Numerics of GANs*. [Online]. Available: https://www.inference.vc/my-notes-on-the-numerics-of-gans/

[62] B. Franci and S. Grammatico, "Training generative adversarial networks via stochastic Nash games," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 3, pp. 1319–1328, Mar. 2023.

[63] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statist.*, vol. 33, no. 3, pp. 1065–1076, Sep. 1962, doi: 10.1214/aoms/1177704472.

[64] H. Robbins, "An empirical Bayes approach to statistics," in *Proc. 3rd Berkeley Symp. Math. Statist. Probab.*, vol. 1. 1956, pp. 157–163. [Online]. Available: https://projecteuclid.org/euclid.bsmsp/1200501653

[65] A. Hyvärinen and P. Dayan, "Estimation of non-normalized statistical models by score matching," *J. Mach. Learn. Res.*, vol. 6, no. 4, pp. 1–15, 2005.

[66] A. Wald, "Note on the consistency of the maximum likelihood estimate," *Ann. Math. Statist.*, vol. 20, no. 4, pp. 595–601, Dec. 1949.

[67] N. De Cao, I. Titov, and W. Aziz, "Block neural autoregressive flow," 2019, *arXiv:1904.04676*.

[68] Y. LeCun. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[69] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.

[70] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3730–3738.

[71] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4396–4405.

[72] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," 2016, *arXiv:1611.02163*.

[73] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*.

[74] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Cham, Switzerland: Springer, 2015, pp. 234–241.

[75] A. Asuncion and D. Newman. (2007). *UCI Machine Learning Repository*. [Online]. Available: https://archive.ics.uci.edu/ml/index.php

[76] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. ICCV*, 2001, pp. 416–423.

[77] M. Germain, K. Gregor, I. Murray, and H. Larochelle, "MADE: Masked autoencoder for distribution estimation," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 881–889.

[78] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2338–2347.

[79] J. B. Oliva et al., "Transformation autoregressive networks," 2018, *arXiv:1801.09819*.

[80] D. Gilboa, A. Pakman, and T. Vatter, "Marginalizable density models," 2021, *arXiv:2106.04741*.

[81] Q. Li, A. Jain, and P. Abbeel, "AdaCat: Adaptive categorical discretization for autoregressive models," in *Proc. 38th Conf. Uncertainty Artif. Intell.*, 2022, pp. 1188–1198.

[82] W. Hardle, M. Müller, S. Sperlich, and A. Werwatz, *Nonparametric and Semiparametric Models*, vol. 1. Berlin, Germany: Springer, 2004.

[83] V. Vapnik, *Statistical Learning Theory*. New York, NY, USA: Wiley, 1998.

[84] P. Bojanowski, A. Joulin, D. Lopez-Pas, and A. Szlam, "Optimizing the latent space of generative networks," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, J. Dy and A. Krause, Eds. Stockholmsmässan, Stockholm Sweden, Jul. 2018, pp. 600–609. [Online]. Available: http://proceedings.mlr.press/v80/bojanowski18a.html

[85] Y. Hoshen, K. Li, and J. Malik, "Non-adversarial image synthesis with generative latent nearest neighbors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5804–5812.

[86] Y. Song, S. Garg, J. Shi, and S. Ermon, "Sliced score matching: A scalable approach to density and score estimation," 2019, *arXiv:1905.07088*.

[87] J. Wu et al., "Sliced Wasserstein generative models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3708–3717.

[88] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.

[89] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017, *arXiv:1710.05941*.

[90] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–26. [Online]. Available: https://openreview.net/forum?id=Hk99zCeAb