LEARNING GENERATIVE MODELS USING DENOISING DENSITY ESTIMATORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning generative probabilistic models that can estimate the continuous density given a set of samples, and that can sample from that density, is one of the fundamental challenges in unsupervised machine learning. In this paper we introduce a new approach to obtain such models based on what we call denoising density estimators (DDEs). A DDE is a scalar function, parameterized by a neural network, that is efficiently trained to represent a kernel density estimator of the data. In addition, we show how to leverage DDEs to develop a novel approach to obtain generative models that sample from given densities. We prove that our algorithms to obtain both DDEs and generative models are guaranteed to converge to the correct solutions. Advantages of our approach include that we do not require specific network architectures like in normalizing flows, ODE solvers as in continuous normalizing flows, nor do we require adversarial training as in generative adversarial networks (GANs). Finally, we provide experimental results that demonstrate practical applications of our technique.

1 INTRODUCTION

Learning generative probabilistic models from raw data is one of the fundamental problems in unsupervised machine learning. The defining property of such models is that they provide functionality to sample from the probability density represented by the input data. In other words, such models can generate new content, which has applications in image or video synthesis for example. In addition, generative probabilistic models may include capabilities to perform density estimation or inference of latent variables. Recently, the use of deep neural networks has led to significant advances in this area. For example, generative adversarial networks (Goodfellow et al., 2014) can be trained to sample very high dimensional densities, but they do not provide density estimation or inference. Inference in Boltzman machines (Salakhutdinov & Hinton, 2009) is tractable only under approximations (Welling & Teh, 2003). Variational autoencoders (Kingma & Welling, 2014) provide functionality for both (approximate) inference and sampling. Finally, normalizing flows (Dinh et al., 2014) perform all three operations (sampling, density estimation, inference) efficiently.

In this paper 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 construct 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. A core component of this approach is the density estimator, which we derive from the theory of denoising autoencoders, hence our term *denoising density estimator*. Compared to normalizing flows, 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 ODEs like in continuous normalizing flows. In contrast to GANs, we do not require adversarial training. In summary, our contributions are as follows:

- A novel approach to obtain 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.
- A density estimator based on denoising autoencoders called denoising density estimator (DDE), and its parameterization using neural networks, which we leverage to train our novel generative model.

Property	GAN	Score Matching	Normalizing Flows	Ours
Provides density	-	-	✓	✓
Forward sampling model	✓	iterative	✓	✓
Exact sampling	✓	asymptotic	✓	✓
Free net architecture	✓	✓	-	✓

Table 1: Comparison of different deep generative approaches.

2 Related Work

Generative adversarial networks (Goodfellow et al., 2014) are currently the most widely studied type of generative probabilistic models for very high dimensional data such as images or videos. However, they are often difficult to train in practice, they can suffer from mode-collapse, and they only support sampling, but neither inference nor density estimation. Hence, there has been a renewed interest in alternative approaches to learn generative models. A common approach is to formulate these models as mappings between a latent space and the data domain, and one way to categorize these techniques is to consider the constraints on this mapping. For example, in normalizing flows (Dinh et al., 2014; Rezende & Mohamed, 2015) the mapping is invertible and differentiable, such that the data density can be estimated using the determinant of its Jacobian, and inference can be perfomed by applying the inverse mapping. Normalizing flows can be trained simply using maximum likelihood estimation (Dinh et al., 2017). The challenge for these techniques is to design computational structures so that their inverses and Jacobians, including their determinants, can be computed efficiently (Huang et al., 2018; Kingma & Dhariwal, 2018). Chen et al. (2018) and Grathwohl et al. (2018) derive continuous normalizing flows by parameterizing the dynamics (the time derivative) of an ordinary differential equation (ODE) using a neural network. They show that this implies that the time derivative of the log density can also be expressed as an ODE, which only involves the trace (not the determinant) of the Jacobian of the network. This makes it possible to use arbitrary network architectures to obtain normalizing flows, but it comes at the computation 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 (Kingma & Welling, 2014). As an advantage over normalizing flows, variational methods do not require an invertible mapping between latent and data space. However, Gaussian likelihood functions correspond to an L_2 reconstruction error, which arguably leads to blurriness artifacts. Recently, Li & Malik (2018) have shown that an approximate form of maximum likelihood estimation, which they call implicit maximum likelihood estimation, can also be performed without requiring invertible mappings. A disadvantage of their approach is that it requires nearest neighbor queries in (high dimensional) data space.

Not all generative models include a latent space, including autoregressive models (van den Oord et al., 2016) or denoising autoencoders (DAEs) (Alain & Bengio, 2014). In particular, Alain & Bengio (2014) and Saremi & Hyvärinen (2019) use the well known relation between DAEs and the score of the corresponding data distributions (Vincent, 2011; Raphan & Simoncelli, 2011) to construct an approximate Markov Chain sampling procedure. Our approach also builds on DAEs, but we formulate an estimator for the un-normalized, scalar density, 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. In concurrent work, Song & Ermon (2019) are formulating a generative model using Langevin dynamics based on estimating gradients of the data distribution via score matching, which also requires an iterative sampling procedure and can sample the data density exactly only asymptotically. Table 1 summarises the similarities and differences of our approach to these techniques.

3 Denoising Density Estimators (DDEs)

Here we show how to estimate a density using a variant of denoising autoencoders (DAEs). More precisely, our approach allows us to obtain the density smoothed by a Gaussian kernel, which is

equivalent to kernel density estimation (Parzen, 1962), up to a normalizing factor. Originally, the optimal DAE $r : \mathbb{R}^n \to \mathbb{R}^n$ (Vincent, 2011; Alain & Bengio, 2014) is defined as the function minimizing the following denoising loss,

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

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 (Robbins, 1956; Raphan & Simoncelli, 2011) that the optimal DAE $r^*(x)$ minimizing $\mathcal{L}_{\mathrm{DAE}}$ can be expressed as follows, which is also known as Tweedie's formula,

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

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

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

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is a vector field that estimates the noise vector $-\eta/\sigma_\eta^2$. The following proposition has also been shown by (Vincent, 2011), and we also provide a proof in Appendix A:

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

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

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

A key observation is that the desired vector field is the gradient of a scalar function and 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,

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

A similar formulation has recently been proposed by Saremi & Hyvärinen (2019). Our terminology is motivated by the following corollary:

Corollary 1. *The minimizer $s^*(x) = \arg\min_s \mathcal{L}_{\mathrm{DDE}}(s; p)$ 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 $\mathcal{L}_{\mathrm{DDE}}(s; p)$ we know that $\nabla_x s^*(x) = \nabla_x \log \tilde{p}(x)$, which leads immediately to the corollary. □

In summary, we have shown how modifying the denoising autoencoder loss (Eq. 1) into a noise estimation loss based on the gradients of a scalar function (Eq. 5) allows us to derive a density estimator (Corollary 1), which we call the denoising density estimator (DDE).

In practice, we approximate the DDE using a neural network $s(x; \theta)$. Assuming that the network has enough capacity and is everywhere differentiable both with respect to x and its parameters θ, we can find the unique minimum of Eq. 5 using standard stochastic gradient descent techniques. For illustration, Figure 1 shows 2D distribution examples, which we approximate using a DDE implemented as a multi-layer perceptron. We only use Softplus activations in our network since it is differentiable everywhere.

4 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 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 3. We express our samplers using

mappings $x = g(z)$, where $x \in \mathrm{R}^n, z \in \mathrm{R}^m$ (usually $n > m$), and z is typically a latent variable with standard normal distribution. In contrast to normalizing flows, $g(z)$ 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 $\tilde{q} = q * k$.

We obtain the generator by minimizing the KL divergence $D_{\mathrm{KL}}(\tilde{q}||\tilde{p})$ 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:*

$$D_{\mathrm{KL}}(\tilde{q}||\tilde{p}) = \langle \tilde{q}, \log \tilde{q} - \log \tilde{p} \rangle > \langle \tilde{q} + \Delta, \log \tilde{q} - \log \tilde{p} \rangle \,, \tag{7}$$

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

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

then $D_{\mathrm{KL}}(\tilde{q}||\tilde{p}) > D_{\mathrm{KL}}(\tilde{q} + \Delta||\tilde{p})$ 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 $\langle \cdot, \cdot \rangle$ to denote the inner product, we can write

$$D_{\mathrm{KL}}(\tilde{q} + \Delta||\tilde{p}) = \langle \tilde{q} + \Delta, \log(\tilde{q} + \Delta) - \log \tilde{p} \rangle \tag{10}$$

$$= \left\langle \tilde{q} + \Delta, \log \tilde{q} + \Delta/\tilde{q} + o(\Delta^2) - \log \tilde{p} \right\rangle \tag{11}$$

$$= \langle \tilde{q}, \log \tilde{q} - \log \tilde{p} \rangle + \langle \Delta, \log \tilde{q} - \log \tilde{p} \rangle + \langle \tilde{q}, \Delta/\tilde{q} \rangle + \langle \Delta, \Delta/\tilde{q} \rangle + o(\Delta^2). \tag{12}$$

This means

$$D_{\mathrm{KL}}(\tilde{q} + \Delta||\tilde{p}) - D_{\mathrm{KL}}(\tilde{q}||\tilde{p}) = \langle \Delta, \log \tilde{q} - \log \tilde{p} \rangle + \langle \tilde{q}, \Delta/\tilde{q} \rangle + \langle \Delta, \Delta/\tilde{q} \rangle + o(\Delta^2) < 0 \tag{13}$$

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

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

At the beginning of each iteration in Algorithm 1, 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 $\mathbb{E}_{x=g(z;\phi)+\eta}\left[s^{\tilde{q}}(x) - \log \tilde{p}(x)\right] + C$ with respect to the generator parameters ϕ. The constant C can be ignored since we only need the gradient. A small enough learning rate guarantees that condition one in Proposition 2 is satisfied. The second condition is satisfied because we update the distribution by updating its generator, and the third condition 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.

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 (Section 4.1).

4.1 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:

Algorithm 1: Training steps for the generator.

input : Pre-trained optimal DDE on input data $\log \tilde{p}(x)$, learning rate δ
initialize generator parameters ϕ
initialize DDE $s^{\tilde{q}} = \arg\min_s \mathcal{L}_{\mathrm{DDE}}(s; q, \sigma_\eta)$ with $q \sim g(z; \phi), z \sim \mathcal{N}(0, 1)$
while *not converged* **do**
  $\phi = \phi + \delta \nabla_\phi \mathbb{E}_{x=g(z;\phi)+\eta} \left[s^{\tilde{q}}(x) - \log \tilde{p}(x) \right]$, with $z \sim \mathcal{N}(0, 1), \eta \sim \mathcal{N}(0, \sigma_\eta^2)$
  // $q \sim g(z; \phi)$ now indicates the updated density using the updated ϕ
  $s^{\tilde{q}} = \arg\min_s \mathcal{L}_{\mathrm{DDE}}(s; q, \sigma_\eta)$
  // $s^{\tilde{q}}$ is now the density (up to a constant) of $g(z; \phi) + \eta$

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 non-zero everywhere, that is, Gaussian blur is invertible.

5 Experiments

Visual Comparisons using 2D Toy Datasets. Similar to prior work, we perform experiments for 2D density estimation and visualization over three datasets (Grathwohl et al., 2018). Additionally, we use these datasets to learn generative models. For our DDE networks, we used multi-layer perceptrons with residual connections. All networks have 25 layers, each with 32 channels and Softplus activation. Trainings have 2048 samples per iteration. As shown in Figure 1, the DDEs can estimate the density accurately and capture the underlying complexities of each density. Due to inherent KDE estimation, our method induces a small blur to the distribution to the density compared to BNAF. However, our DDE can estimate the density coherently through the data domain, whereas BNAF produces noisy approximation across the data manifold, where the estimated density is sometimes too small or too large. To demonstrate, we show DDEs trained with both small and large noise standard deviations $\sigma_\eta = 0.05$ and $\sigma_\eta = 0.2$.

Generator training and sampling is demonstrated in Figure 1. The sharp edges of the checkerboard samples implies that the generator learns to sample from the target density although the DDEs estimate noisy densities. The generator update requires DDE networks to be optimal at each gradient step. For faster convergence, we take 10 DDE gradient descent steps for each generator update. In Figure 2 we illustrate the influence of the noise level σ_η on the generated densities. This shows that in practice larger σ_η do not lead to accurate sampling, since inverting the Gaussian blur becomes ill-posed. We summarize the training parameters used in these experiments in Appendix E.

MNIST. Figure 3 illustrates our generative training on the MNIST (LeCun, 1998) dataset using Algorithm 1. We use a dense block architecture with fully connected layers here and refer to Appendix B for the network and training details, including additional results for Fashion-MNIST (Xiao et al., 2017). Figure 3 shows qualitatively that our generator is able to replicate the underlying distributions. In addition, latent-space interpolation demonstrates that the network learns an intuitive and interpretable mapping from noise to samples of the distribution.

CelebA. Figure 4 shows additional experiments on the CelebA dataset (Liu et al., 2015). The images in the dataset have $32 \times 32 \times 3$ dimensions and we normalize the pixel values to be in 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 (Karras et al., 2019) architecture for our generator network. Please refer to Appendix C for network and training details. Figure 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.

Quantitative Evaluation with Stacked-MNIST. We perform a quantitative evaluation of our approach based on the synthetic Stacked-MNIST (Metz et al., 2016) dataset, which was designed to

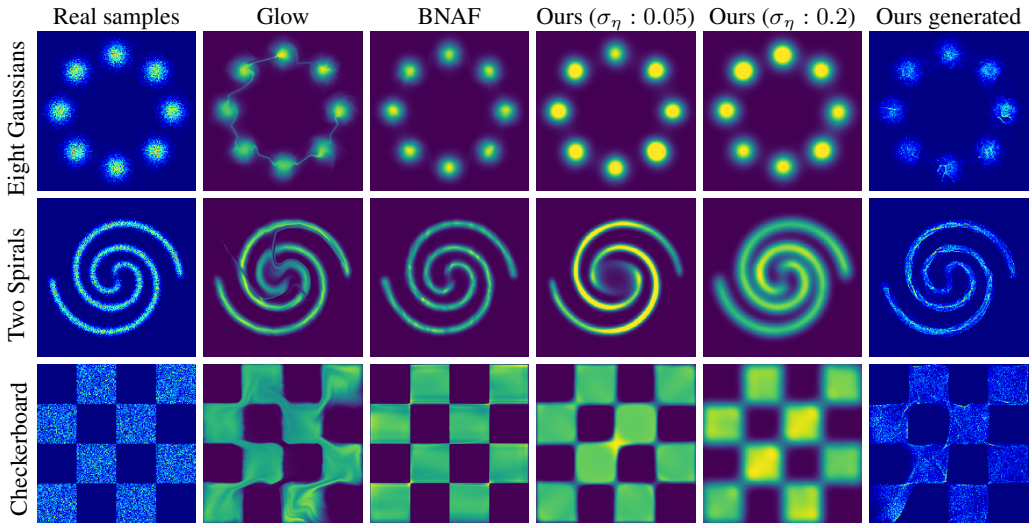

Figure 1: Density estimation on 2D toy data. We show that we can accurately capture these densities with few visual artifacts. We also show samples generated using our generative model training.

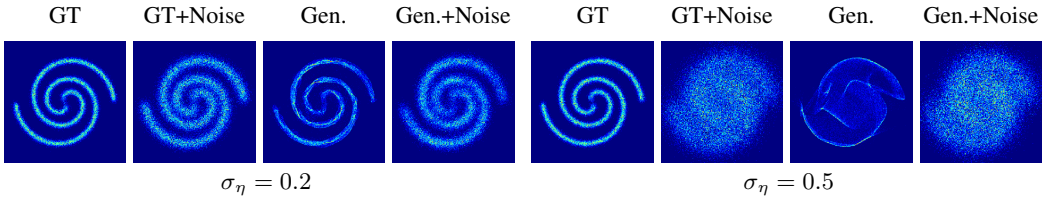

Figure 2: The influence of σ_η on sample generation. We observe that the smoothed sampled density is close to the training density. However, for large σ_η, the sampled density without smoothing can be quite different from the true density because inversion of Gaussian smoothing becomes ill-posed.

analyse mode-collapse in generative models. The dataset is constructed by stacking three randomly chosen digit images from MNIST to generate samples of size $28 \times 28 \times 3$. This augments the number of classes to 10^3, which are considered as 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 (Metz et al., 2016), 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.

Figure 5 reports the quantitative results for this experiment by comparing our method with well-tuned GAN models. DCGAN (Radford et al., 2015) implements a basic GAN training strategy using a stable architecture. WGAN uses the Wasserstein distance (Arjovsky et al., 2017), and WGAN+GP includes a gradient penalty to regularize the discriminator (Gulrajani et al., 2017). 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 (that is, 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 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) parametrized with a UNET architecture, which is then followed

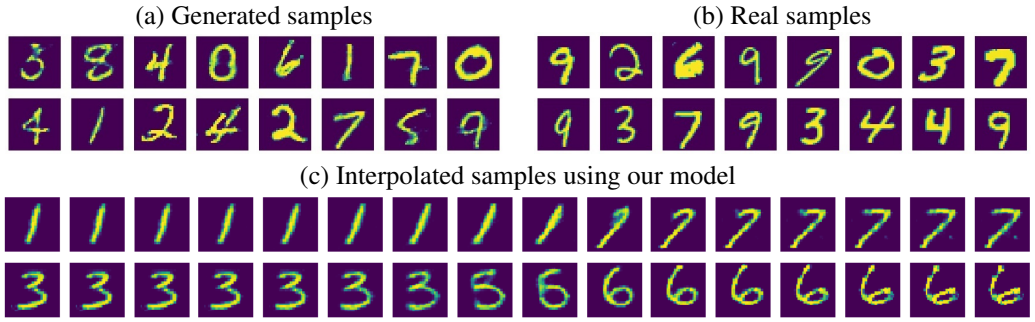

Figure 3: MNIST samples using our generator (a) and from the real dataset (b). Latent space interpolation using our generator (c).

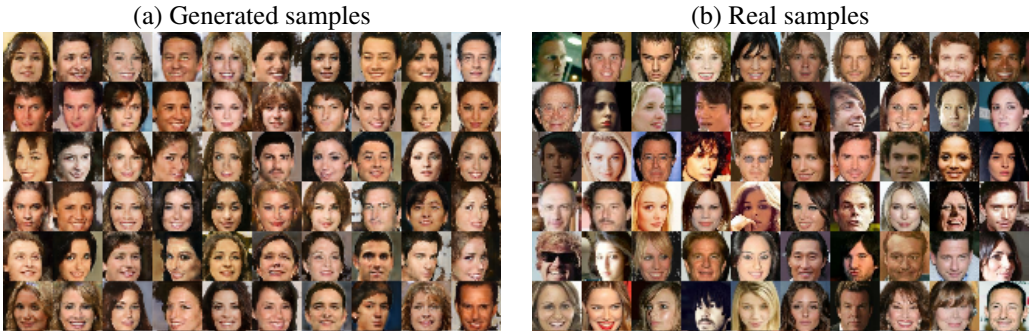

Figure 4: Results of our generator training algorithm on 32×32 images from the celebA dataset (Liu et al., 2015)

by a sampling algorithm using annealed Langevin dynamics (ALD) as described by Song and Ermon (Song & Ermon, 2019). 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. While our training loss is identical 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. ALD+DDE uses the spatial gradient of the DDE for sampling with ALD (Song & Ermon, 2019), instead of our proposed direct, one-step generator. 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 Figure 5. Further, our approach with direct sampling outperforms both UNET+ALD and DDE+ALD.

Real Data Density Estimation. We follow the experiments in BNAF (De Cao et al., 2019) for density estimation on real measured data. This includes POWER, GAS, HEPMASS, and MINIBOON datasets (Asuncion & Newman, 2007). Since DDEs can estimate densities up to their normalizing constant, we approximate the normalizing constant using Monte Carlo estimation for these experiments. We show average log-likelihoods over test sets and compare to state-of-the-art methods for normalized density estimation in Table 2. We have omitted the results of the BSDS300 dataset (Martin et al., 2001), since we could not estimate the normalizing constant reliably (due to high dimensionality of the data).

To train our DDEs, we used Multi-Layer Perceptrons (MLP) with residual connections between each layer. All networks have 25 layers, with 64 channels and Softplus activations, except for GAS and HEPMASS, which employ 128 channels. We trained the models for 400 epochs using learning rate of $2.5e-4$ with linear decay with 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 $5e-2$ for POWER, $4e-2$ for GAS, $2e-2$ for HEPMASS, and $1.5e-1$ for MINIBOON. We estimate the normalizing constant via importance sampling using

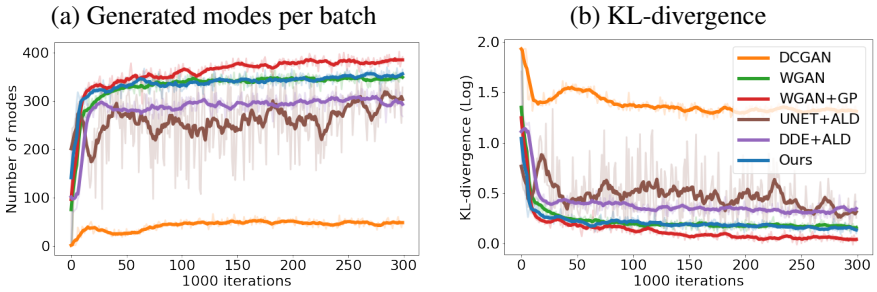

Figure 5: Mode-collapse experiment results on Stacked-MNIST as a function of training iterations (for discriminator or DDE). (a) Number of generated modes per batch of size 512. (b) Reverse KL-divergence between the generated and the data distribution in the logarithmic domain. All methods use the DCGAN architecture with the same capacity except for UNET+ALD.

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	0.17 $\pm.01$	8.33 $\pm.14$	-18.71 $\pm.02$	-13.55 $\pm.49$
Glow	0.17 $\pm.01$	8.15 $\pm.40$	-18.92 $\pm.08$	-11.35 $\pm.07$
MADE MoG	0.40 $\pm.01$	8.47 $\pm.02$	-15.15 $\pm.02$	-12.27 $\pm.47$
MAF-affine	0.24 $\pm.01$	10.08 $\pm.02$	-17.73 $\pm.02$	-12.24 $\pm.45$
MAF-affine MoG	0.30 $\pm.01$	9.59 $\pm.02$	-17.39 $\pm.02$	-11.68 $\pm.44$
FFJORD	0.46 $\pm.01$	8.59 $\pm.12$	-14.92 $\pm.08$	-10.43 $\pm.04$
NAF-DDSF	0.62 $\pm.01$	11.96 $\pm.33$	-15.09 $\pm.40$	-8.86 $\pm.15$
TAN	0.60 $\pm.01$	**12.06** $\pm.02$	-13.78 $\pm.02$	-11.01 $\pm.48$
BNAF	0.61 $\pm.01$	**12.06** $\pm.09$	-14.71 $\pm.38$	-8.95 $\pm.07$
Ours	**0.97** $\pm.18$	9.73 ±1.14	**-11.3** $\pm.16$	**-6.94** ±1.81

Table 2: Average log-likelihood comparison in four datasets (Asuncion & Newman, 2007). The top row indicates input size and dimensionality for each dataset. Best performances are in bold.

a Gaussian distribution with the mean and variance of the DDE input distribution. We average 5 estimations using 51200 samples each (we used 10 times more samples for GAS), and we indicate the variance of this average in Table 2.

5.1 Discussion and Limitations

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 thorough strategies to determine this parameter in a data-dependent manner. An other challenge is to obtain high-quality results using extremely high-dimensional data such as high-resolution images. In practice, one strategy is to combine our approach with latent embedding learning methods (Bojanowski et al., 2018), in a similar fashion as proposed by Hoshen et al. (2019). Finally, our framework uses three networks to learn a generator based on input samples (a DDE for the samples, the generator, and a DDE for the generator). Our generator training approach, however, is independent of the type of density estimator, and techniques other than DDEs could also be used in this step.

6 Conclusions

In conclusion, we presented a novel approach to learn generative models using a novel density estimator, called the denoising density estimator (DDE). We developed simple training algorithms and our theoretical analysis proves their convergence to a unique optimum. Our technique is derived from a reformulation of denoising autoencoders, and does not require specific neural network architectures, ODE integration, nor adversarial training. We achieve state of the art results on a standard log-likelihood evaluation benchmark compared to recent techniques based on normalizing flows, continuous flows, and autoregressive models.

REFERENCES

Guillaume Alain and Yoshua Bengio. What regularized auto-encoders learn from the data-generating distribution. *Journal of Machine Learning Research*, 15:3743–3773, 2014. URL http://jmlr.org/papers/v15/alain14a.html.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Arthur Asuncion and David Newman. UCI machine learning repository, 2007. URL https://archive.ics.uci.edu/ml/index.php.

Piotr Bojanowski, Armand Joulin, David Lopez-Pas, and Arthur Szlam. Optimizing the latent space of generative networks. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 600–609, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/bojanowski18a.html.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6571–6583. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7892-neural-ordinary-differential-equations.pdf.

Nicola De Cao, Ivan Titov, and Wilker Aziz. Block neural autoregressive flow. *arXiv preprint arXiv:1904.04676*, 2019.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. 2014. URL http://arxiv.org/abs/1410.8516,2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *Proc. ICLR*, 2017. URL https://arxiv.org/abs/1605.08803.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: free-form continuous dynamics for scalable reversible generative models. *CoRR*, abs/1810.01367, 2018. URL http://arxiv.org/abs/1810.01367.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pp. 5767–5777, 2017.

Yedid Hoshen, Ke Li, and Jitendra Malik. Non-adversarial image synthesis with generative latent nearest neighbors. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2078–2087, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/huang18d.html.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

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. URL https://openreview.net/forum?id=Hk99zCeAb.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *Proc. ICLR*, 2014. URL https://arxiv.org/abs/1312.6114v10.

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 (eds.), *Advances in Neural Information Processing Systems 31*, pp. 10215–10224. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/8224-glow-generative-flow-with-invertible-1x1-convolutions.pdf.

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

Ke Li and Jitendra Malik. Implicit maximum likelihood estimation. *CoRR*, abs/1809.09087, 2018. URL http://arxiv.org/abs/1809.09087.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

David Martin, Charless Fowlkes, Doron Tal, Jitendra Malik, et al. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. Iccv Vancouver:, 2001.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

Emanuel" Parzen. On estimation of a probability density function and mode. *Ann. Math. Statist.*, 33 (3):1065–1076, Sept 1962. doi: 10.1214/aoms/1177704472. URL https://doi.org/10.1214/aoms/1177704472.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Martin Raphan and Eero P. Simoncelli. Least squares estimation without priors or supervision. *Neural Comput.*, 23(2):374–420, February 2011. ISSN 0899-7667. doi: 10.1162/NECO_a_00076. URL http://dx.doi.org/10.1162/NECO_a_00076.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1530–1538, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/rezende15.html.

Herbert Robbins. An empirical bayes approach to statistics. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pp. 157–163, Berkeley, Calif., 1956. University of California Press. URL https://projecteuclid.org/euclid.bsmsp/1200501653.

Ruslan Salakhutdinov and Geoffrey Hinton. Deep Boltzmann machines. In *Artificial intelligence and statistics*, pp. 448–455, 2009.

Saeed Saremi and Aapo Hyvärinen. Neural empirical bayes. *ArXiv*, abs/1903.02334, 2019.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *CoRR*, abs/1907.05600, 2019. URL http://arxiv.org/abs/1907.05600.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with PixelCNN decoders. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 4790–4798. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6527-conditional-image-generation-with-pixelcnn-decoders.pdf.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Comput.*, 23(7):1661–1674, July 2011. ISSN 0899-7667. doi: 10.1162/NECO_a_00142. URL http://dx.doi.org/10.1162/NECO_a_00142.

Max Welling and Yee Whye Teh. Approximate inference in boltzmann machines. *Artificial Intelligence*, 143(1):19 – 50, 2003. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(02)00361-2. URL http://www.sciencedirect.com/science/article/pii/S0004370202003612.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

A PROOF OF SCORE MATCHING VIA NOISE ESTIMATION

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

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

$$\mathcal{L}_{\text{NEs}}(f; p, \sigma_\eta) = \int_{\mathbb{R}^n} \mathbb{E}_{\eta \sim \mathcal{N}(0, \sigma_\eta^2)} \left[p(x) \| f(x + \eta) + \eta / \sigma_\eta^2 \|^2 \right] dx, \tag{14}$$

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

$$\mathcal{L}_{\text{NEs}}(f; p, \sigma_\eta) = \int_{\mathbb{R}^n} \mathbb{E}_{\eta \sim \mathcal{N}(0, \sigma_\eta^2)} \left[p(\tilde{x} - \eta) \| f(\tilde{x}) + \eta / \sigma_\eta^2 \|^2 \right] d\tilde{x}. \tag{15}$$

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

$$\mathbb{E}_{\eta \sim \mathcal{N}(0, \sigma_\eta^2)} \left[p(\tilde{x} - \eta) f(\tilde{x}) \right] = -\frac{1}{\sigma_\eta^2} \mathbb{E}_{\eta \sim \mathcal{N}(0, \sigma_\eta^2)} \left[p(\tilde{x} - \eta) \eta \right], \tag{16}$$

and hence

$$f(\tilde{x}) = -\frac{1}{\sigma_\eta^2} \frac{\mathbb{E}_{\eta \sim \mathcal{N}(0, \sigma_\eta^2)} \left[p(\tilde{x} - \eta) \eta \right]}{\mathbb{E}_{\eta \sim \mathcal{N}(0, \sigma_\eta^2)} \left[p(\tilde{x} - \eta) \right]} \tag{17}$$

$$= \nabla_{\tilde{x}} \log [p * k](\tilde{x}) = \nabla_{\tilde{x}} \log \tilde{p}(\tilde{x}), \tag{18}$$

which follows from basic calculus and has also been used by Raphan & Simoncelli (2011). □

B VISUAL RESULTS ON FASHION-MNIST

For the experiments on MNIST and Fashion-MNIST, we used the Dense Block architecture (Huang et al., 2017) 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 dimensional 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. Figure 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 (Karras et al., 2019) architecture. We use Swish activations (Ramachandran et al., 2017) 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 (Karras et al., 2018) with learning rate $5e-3$ for the DDEs, and a slightly lower learning rate for the generator $3e-3$. 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 (Karras et al., 2019) 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 $2e-2$, the Adam optimizer with $\beta_1 = 0.9$, and the generator updates took place after every 10th

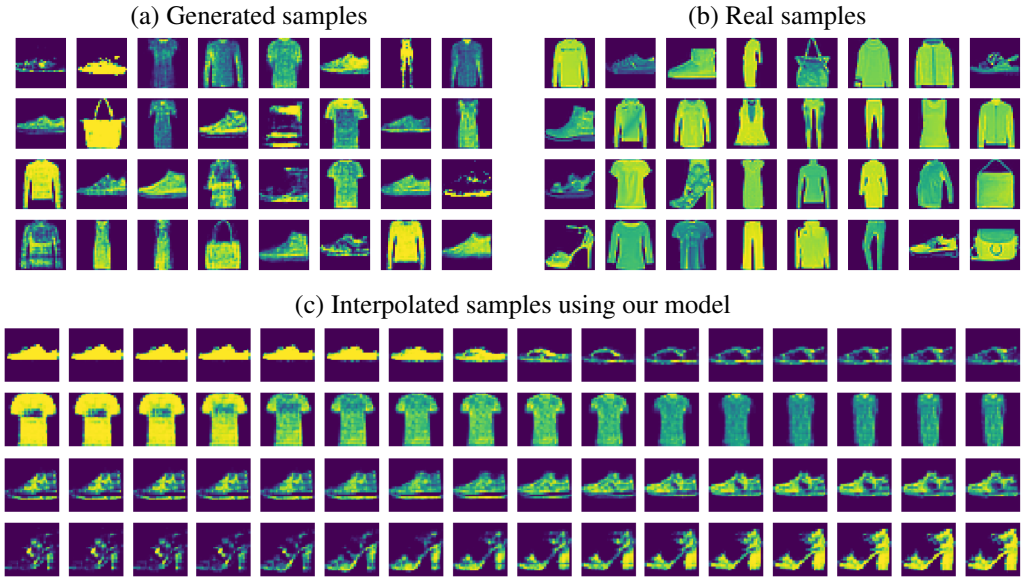

Figure 6: Fashion-MNIST results using our generator training algorithm (a). Samples from the real dataset (b). Interpolated samples using our Generator (c).

DDE step. We use standard parameters for the other methods (DCGAN, WGAN, WGAN+GP), including a learning rate of $2e-4$, the Adam optimizer with $\beta_1 = 0.5$, and we trained the generator every 5th 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 include 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 test. 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 $1e-4$.

E Details for 2D Dataset Training

Table 3 lists the hyper-parameters we used for different experiments on 2D datasets.

Experiment	σ_η	Dataset	Learning rate	Iterations
Figure 1 (density estimation)	0.05, 0.2	Checkerboard Two spirals	0.001	15000
	0.2	Eight Gaussians	0.005	15000
	0.05	Eight Gaussians	0.0005	23750
Figure 1 (generative), Figure 2	0.2	Checkerboard	0.001	200000
	0.2	Two spirals	Decrease by half	250000
	0.1	Eight Gaussians	every 1000 epochs	250000

Table 3: Training parameters for 2D datasets.