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. In contrast, Boltzman machines (Salakhutdinov & Hinton, 2009) allow for efficient inference. 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 density estimator based on denoising autoencoders called denoising density estimator (DDE), and its parameterization using neural networks.
- An algorithm to train samplers for generative models via minimizing the KL divergence between the DDE of the sampler and the DDE of the input data. The training algorithm is guaranteed to converge.

2 RELATED

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) use the relation between DAEs and the score of the corresponding data distributions to construct an approximate Markov Chain sampling procedure. Our approach also builds on DAEs, but we train a generator instead of requiring Markov chain sampling, which has the disadvantages of requiring sequential sampling and producing correlated samples.

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 $r : \mathbb{R}^n \to \mathbb{R}^n$ (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 (Raphan & Simoncelli, 2011) that the optimal DAE $r^*(x)$, which minimizes $\mathcal{L}_{\mathrm{DAE}}$, can be expressed as follows,

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

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 formulate the following 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 $-\eta/\sigma_\eta^2$.

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 noise estimator is the gradient of the logarithm of the Gaussian smoothed density $\tilde{p}(x)$.

Proof. Clearly $\mathcal{L}_{\mathrm{DAE}}$ is convex in f hence the minimizer is unique. We can rewrite the noise estimation loss from Equation 10 as

$$\mathcal{L}_{\mathrm{NEs}}(f; p, \sigma_\eta) = \int_{\mathbb{R}^n} \mathbb{E}_{\eta \sim \mathcal{N}(0, \sigma_\eta^2)} \left[p(x) \| f(x+\eta) + \eta/\sigma_\eta^2 \|^2 \right] dx, \tag{5}$$

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

$$\mathcal{L}_{\mathrm{NEs}}(f; p, \sigma_\eta) = \int_{\mathbb{R}^n} \mathbb{E}_{\eta \sim \mathcal{N}(0, \sigma_\eta^2)} \left[p(\tilde{x} - \eta) \| f(\tilde{x}) + \eta/\sigma_\eta^2 \|^2 \right] d\tilde{x}. \tag{6}$$

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

$$\mathbb{E}_{\eta \sim \mathcal{N}(0, \sigma_\eta^2)} \left[p(\tilde{x} - \eta) f(\tilde{x}) \right] = \frac{1}{\sigma_\eta^2} \mathbb{E}_{\eta \sim \mathcal{N}(0, \sigma_\eta^2)} \left[p(\tilde{x} - \eta)\eta \right], \tag{7}$$

and hence

$$\begin{aligned} f(\tilde{x}) &= \frac{1}{\sigma_\eta^2} \frac{\mathbb{E}_{\eta \sim \mathcal{N}(0, \sigma_\eta^2)} \left[p(\tilde{x} - \eta)\eta \right]}{\mathbb{E}_{\eta \sim \mathcal{N}(0, \sigma_\eta^2)} \left[p(\tilde{x} - \eta) \right]} && (8) \\ &= \nabla_{\tilde{x}} \log[p * k](\tilde{x}) = \nabla_{\tilde{x}} \log \tilde{p}(\tilde{x}), && (9) \end{aligned}$$

which follows from basic calculus. □

The last step also proofs 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,

$$\mathcal{L}_{\mathrm{DDE}}(s; p, \sigma_\eta) = \mathbb{E}_{x \sim p, \eta \sim \mathcal{N}(0, \sigma_\eta^2)} \left[\| \nabla_x s(x+\eta) + \eta/\sigma_\eta^2 \|^2 \right]. \tag{10}$$

The name is motivated by the following corollary:

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

$$s^*(x) = \log \tilde{p}(x) + C, \tag{11}$$

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. 10) allows us to derive a density estimator (Corollary 1), which we call the denoising density estimator (DDE).

3.1 Parametrization using Neural Nets

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. 10 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

We next describe how to leverage DDEs to construct samplers 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 divergenc $D_{\mathrm{KL}}(\tilde{q}||\tilde{p})$ 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:*

$$D_{\mathrm{KL}}(\tilde{q}||\tilde{p}) = \langle \tilde{q}, \log \tilde{q} - \log \tilde{p} \rangle > \langle \tilde{q} + \Delta, \log \tilde{q} - \log \tilde{p} \rangle , \tag{12}$$

$$\langle \Delta, 1 \rangle = 0, \tag{13}$$

$$\Delta^2 < \epsilon, \quad \textit{(pointwise exponentiation)} \tag{14}$$

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. Then we can write

$$D_{\mathrm{KL}}(\tilde{q} + \Delta||\tilde{p}) = \langle \tilde{q} + \Delta, \log(\tilde{q} + \Delta) - \log \tilde{p} \rangle \tag{15}$$

$$= \left\langle \tilde{q} + \Delta, \log \tilde{q} + \Delta/\tilde{q} + o(\Delta^2) - \log \tilde{p} \right\rangle \tag{16}$$

$$= \langle \tilde{q}, \log \tilde{q} - \log \tilde{p} \rangle + \langle \Delta, \log \tilde{q} - \log \tilde{p} \rangle + \langle \tilde{q}, \Delta/\tilde{q} \rangle + \langle \Delta, \Delta/\tilde{q} \rangle + o(\Delta^2). \tag{17}$$

This means

$$D_{\mathrm{KL}}(\tilde{q} + \Delta||\tilde{p}) - D_{\mathrm{KL}}(\tilde{q}||\tilde{p}) = \langle \Delta, \log \tilde{q} - \log \tilde{p} \rangle + \langle \tilde{q}, \Delta/\tilde{q} \rangle + \langle \Delta, \Delta/\tilde{q} \rangle + o(\Delta^2) < 0 \tag{18}$$

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 determine an updated generator 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).

Algorithm 1: Training steps for the generator.

```
input : Pre-trained optimal DDE on input data $\log \tilde{p}(x)$, learning rate $\delta$
initialize generator parameters $\phi$
initialize DDE $s^{\tilde{q}} = \arg\min_s \mathcal{L}_{\text{DDE}}(s; q, \sigma_\eta)$ with $q \sim g(z;\phi), z \sim \mathcal{N}(0,1)$
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 $\phi$
    $s^{\tilde{q}} = \arg\min_s \mathcal{L}_{\text{DDE}}(s; q, \sigma_\eta)$
    // $s^{\tilde{q}}$ is now the density (up to a constant) of $g(z;\phi) + \eta$
```

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:

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

5.1 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 $\sigma_\eta = 0.05, 0.2$.

Generator training and sampling is demonstrated in Figure 2. 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 3 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 in the table in Figure 2.

5.2 High Dimensional Data

To experiment with higher dimensional signals, we tested our generative training using the MNIST(LeCun, 1998) and Fashion-MNIST (Xiao et al., 2017) datasets. 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 distribtion. In addition, the DDEs were trained with noise standard deviation $\sigma_\eta = 0.5$, where

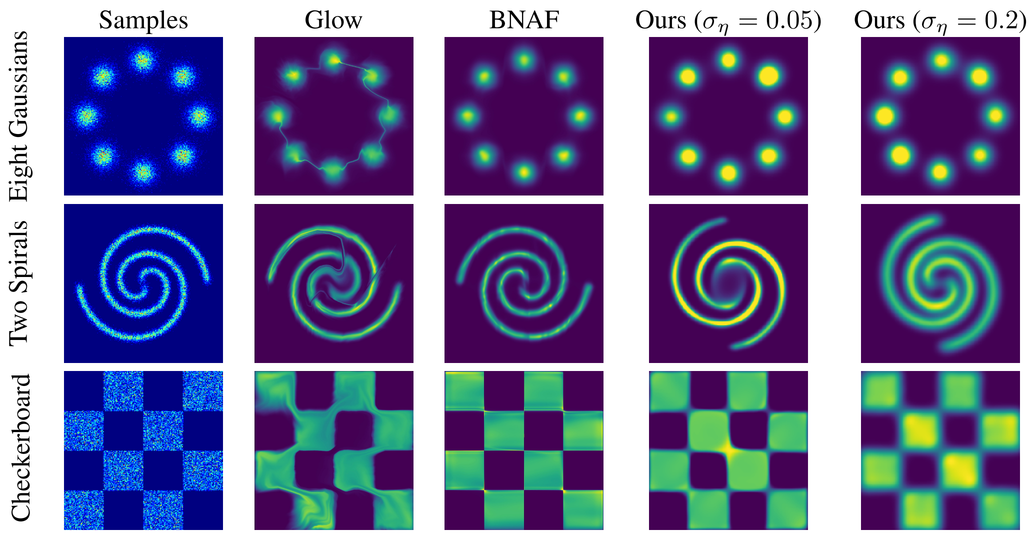

Figure 1: Density estimation on toy 2D data. We show that we can accurately capture these densities with few visual artifacts. The size of the 2D domain shown is 4×4.

Fig.	σ_η	Dataset	Lrng. rate	Its.
1	0.05, 0.2	Checkerbrd. Two spirals	0.001	15k
	0.2	8 Gaussians	0.005	15k
	0.05	8 Gaussians	0.0005	23k
2,3	0.2	Checkerbrd.	0.001 $1/2$ every 1000 it.	200k
	0.2	Two spirals		250k
	0.1	8 Gaussians		250k

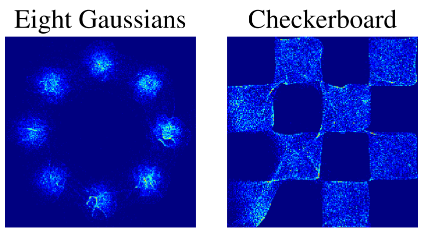

Figure 2: Training parameters for 2D datasets (left). Generating samples from 2D densities (right).

pixel values were scaled to range between 0 and 1. Figures 4 and 5 show the results for these two datasets. As we show, the generator is able to replicate the underlying distribution for both datasets. Finally, we show results with latent-sapce interpolation for the generator input, and we can see that the network learns an intuitive and interpretable mapping from noise to samples of the distribution.

5.3 Real Data Density Estimation

We follow the experiments in BNAF (De Cao et al., 2019) on real data measurements. 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 1. 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. In order to estimate the normalizing constant, we use importance sampling using a Gaussian distribution as the base density. For all datasets we used the mean and variance of the DDE input distribution for the base model. We average 5 instances of estimation using 51200

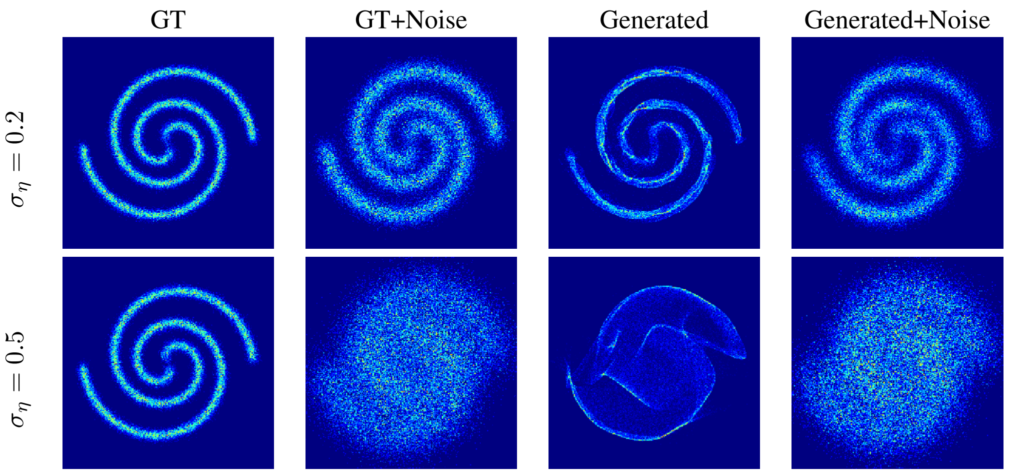

Figure 3: The influence of σ_η on sample generation. We observe that the smoothed sampled density is close to the training density. However, for large σ_η, the sampled density without smoothing can be quite different from the true density because inversion of Gaussian smoothing becomes ill-posed.

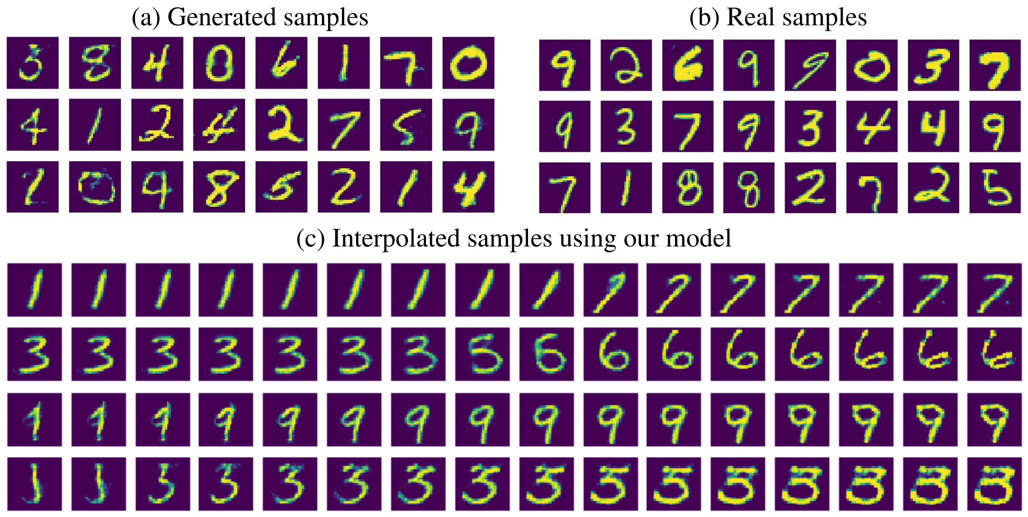

Figure 4: MNIST samples using our generator (a) and from the real dataset (b). Latent space interpolation using our generator (c).

samples (we used 10 times more samples for GAS) to estimate the normalizing constant, and we indicate the variance of this average in Table 1.

5.4 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.

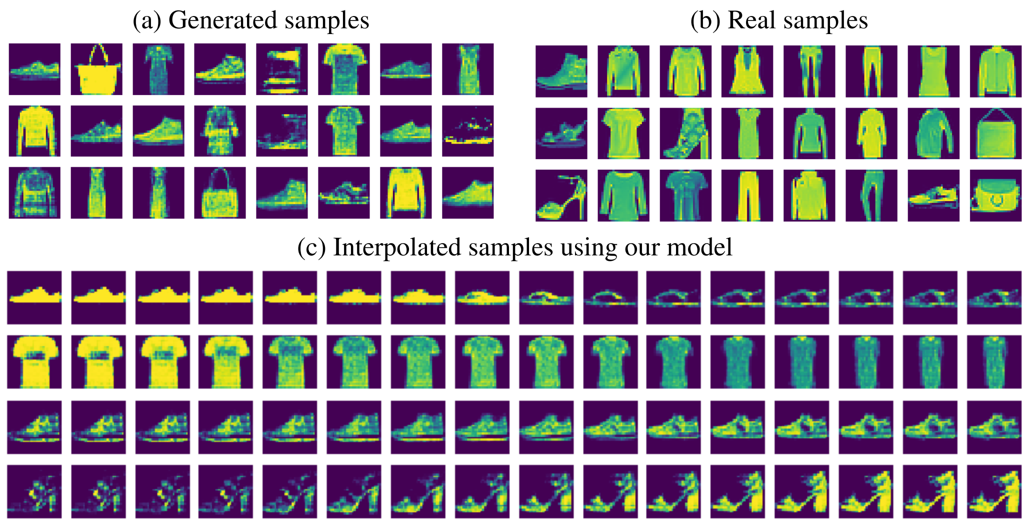

Figure 5: Fashion-MNIST results using our generator training algorithm (a). Samples from the real dataset (b). Interpolated samples using our Generator (c).

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	$0.60 \pm .01$	$\mathbf{12.06} \pm .02$	$-13.78 \pm .02$	$-11.01 \pm .48$
BNAF	$0.61 \pm .01$	$\mathbf{12.06} \pm .09$	$-14.71 \pm .38$	$-8.95 \pm .07$
Ours	$\mathbf{0.97} \pm .18$	9.73 ± 1.14	**-11.3** $\pm .16$	**-6.94** ± 1.81

Table 1: Average log-likelihood comparison in four datasets (Asuncion & Newman, 2007). Input size and dimensionality is shown below each dataset name. Best performances are shown in bold.

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`.

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, Stockholmsmssan, 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.

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, Stockholmsmssan, 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.

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.

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.

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`.

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`.

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

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`.

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