Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference implementation of the algorithm.

1. Introduction

Historically, probabilistic models suffer from a tradeoff between two conflicting objectives: *tractability* and *flexibility*. Models that are *tractable* can be analytically evaluated and easily fit to data (e.g. a Gaussian or Laplace). However, these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we can define models in terms of any (non-negative) function $\phi(\mathbf{x})$ yielding the flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where Z is a normalization constant. However, computing this normalization constant is generally intractable. Evaluating, training, or drawing samples from such flexible models typically requires a very expensive Monte Carlo process.

A variety of analytic approximations exist which ameliorate, but do not remove, this tradeoff–for instance mean field theory and its expansions (T, 1982; Tanaka, 1998), variational Bayes (Jordan et al., 1999), contrastive divergence (Welling & Hinton, 2002; Hinton, 2002), minimum probability flow (Sohl-Dickstein et al., 2011b;a), minimum KL contraction (Lyu, 2011), proper scoring rules (Gneiting & Raftery, 2007; Parry et al., 2012), score matching (Hyvärinen, 2005), pseudolikelihood (Besag, 1975), loopy belief propagation (Murphy et al., 1999), and many, many more. Non-parametric methods (Gershman & Blei, 2012) can also be very effective[1].

1.1. Diffusion probabilistic models

We present a novel way to define probabilistic models that allows:

1. extreme flexibility in model structure,
2. exact sampling,

[1]Non-parametric methods can be seen as transitioning smoothly between tractable and flexible models. For instance, a non-parametric Gaussian mixture model will represent a small amount of data using a single Gaussian, but may represent infinite data as a mixture of an infinite number of Gaussians.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

3. easy multiplication with other distributions, e.g. in order to compute a posterior, and
4. the model log likelihood, and the probability of individual states, to be cheaply evaluated.

Our method uses a Markov chain to gradually convert one distribution into another, an idea used in non-equilibrium statistical physics (Jarzynski, 1997) and sequential Monte Carlo (Neal, 2001). We build a generative Markov chain which converts a simple known distribution (e.g. a Gaussian) into a target (data) distribution using a diffusion process. Rather than use this Markov chain to approximately evaluate a model which has been otherwise defined, we explicitly define the probabilistic model as the endpoint of the Markov chain. Since each step in the diffusion chain has an analytically evaluable probability, the full chain can also be analytically evaluated.

Learning in this framework involves estimating small perturbations to a diffusion process. Estimating small perturbations is more tractable than explicitly describing the full distribution with a single, non-analytically-normalizable, potential function. Furthermore, since a diffusion process exists for any smooth target distribution, this method can capture data distributions of arbitrary form.

We demonstrate the utility of these *diffusion probabilistic models* by training high log likelihood models for a two-dimensional swiss roll, binary sequence, handwritten digit (MNIST), and several natural image (CIFAR-10, bark, and dead leaves) datasets.

1.2. Relationship to other work

The wake-sleep algorithm (Hinton, 1995; Dayan et al., 1995) introduced the idea of training inference and generative probabilistic models against each other. This approach remained largely unexplored for nearly two decades, though with some exceptions (Sminchisescu et al., 2006; Kavukcuoglu et al., 2010). There has been a recent explosion of work developing this idea. In (Kingma & Welling, 2013; Gregor et al., 2013; Rezende et al., 2014; Ozair & Bengio, 2014) variational learning and inference algorithms were developed which allow a flexible generative model and posterior distribution over latent variables to be directly trained against each other.

The variational bound in these papers is similar to the one used in our training objective and in the earlier work of (Sminchisescu et al., 2006). However, our motivation and model form are both quite different, and the present work retains the following differences and advantages relative to these techniques:

1. We develop our framework using ideas from physics, quasi-static processes, and annealed importance sampling rather than from variational Bayesian methods.
2. We show how to easily multiply the learned distribution with another probability distribution (eg with a conditional distribution in order to compute a posterior)
3. We address the difficulty that training the inference model can prove particularly challenging in variational inference methods, due to the asymmetry in the objective between the inference and generative models. We restrict the forward (inference) process to a simple functional form, in such a way that the reverse (generative) process will have the same functional form.
4. We train models with thousands of layers (or time steps), rather than only a handful of layers.
5. We provide upper and lower bounds on the entropy production in each layer (or time step)

There are a number of related techniques for training probabilistic models (summarized below) that develop highly flexible forms for generative models, train stochastic trajectories, or learn the reversal of a Bayesian network. Reweighted wake-sleep (Bornschein & Bengio, 2015) develops extensions and improved learning rules for the original wake-sleep algorithm. Generative stochastic networks (Bengio & Thibodeau-Laufer, 2013; Yao et al., 2014) train a Markov kernel to match its equilibrium distribution to the data distribution. Neural autoregressive distribution estimators (Larochelle & Murray, 2011) (and their recurrent (Uria et al., 2013a) and deep (Uria et al., 2013b) extensions) decompose a joint distribution into a sequence of tractable conditional distributions over each dimension. Adversarial networks (Goodfellow et al., 2014) train a generative model against a classifier which attempts to distinguish generated samples from true data. A similar objective in (Schmidhuber, 1992) learns a two-way mapping to a representation with marginally independent units. In (Rippel & Adams, 2013; Dinh et al., 2014) bijective deterministic maps are learned to a latent representation with a simple factorial density function. In (Stuhlmüller et al., 2013) stochastic inverses are learned for Bayesian networks. Mixtures of conditional Gaussian scale mixtures (MCGSMs) (Theis et al., 2012) describe a dataset using Gaussian scale mixtures, with parameters which depend on a sequence of causal neighborhoods. There is additionally significant work learning flexible generative mappings from simple latent distributions to data distributions – early examples including (MacKay, 1995) where neural networks are introduced as generative models, and (Bishop et al., 1998) where a stochastic manifold mapping is learned from a latent space to the data space. We will compare experimentally against adversarial networks and MCGSMs.

Related ideas from physics include the Jarzynski equality (Jarzynski, 1997), known in machine learning as An-

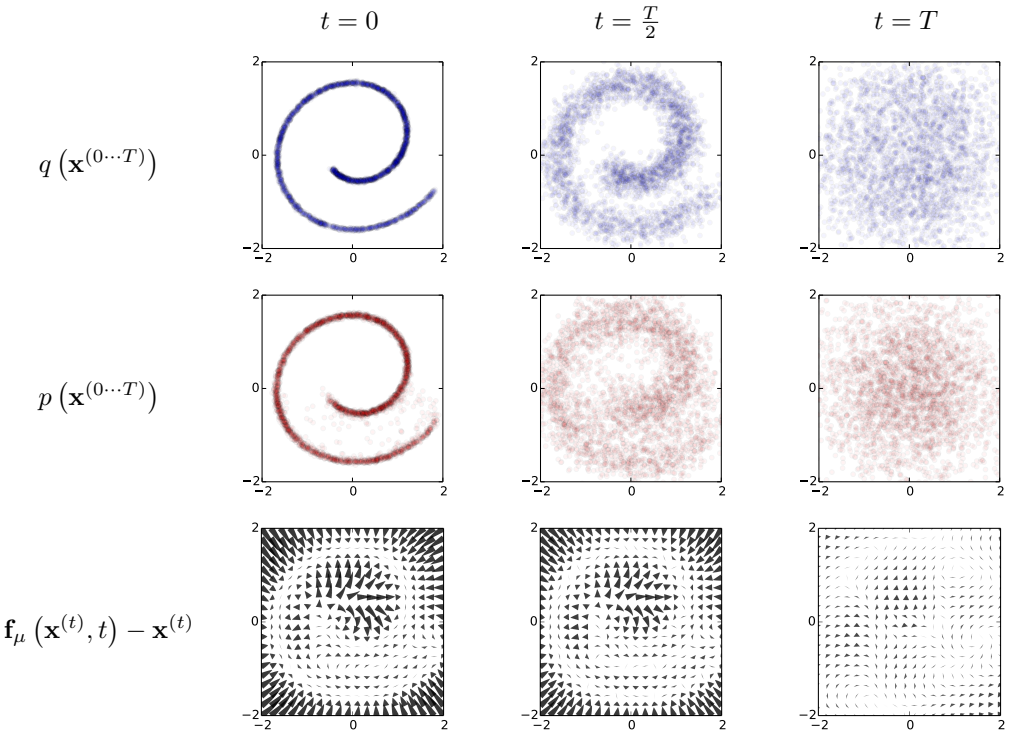

Figure 1. The proposed modeling framework trained on 2-d swiss roll data. The top row shows time slices from the forward trajectory $q\left(\mathbf{x}^{(0\cdots T)}\right)$. The data distribution (left) undergoes Gaussian diffusion, which gradually transforms it into an identity-covariance Gaussian (right). The middle row shows the corresponding time slices from the trained reverse trajectory $p\left(\mathbf{x}^{(0\cdots T)}\right)$. An identity-covariance Gaussian (right) undergoes a Gaussian diffusion process with learned mean and covariance functions, and is gradually transformed back into the data distribution (left). The bottom row shows the drift term, $\mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right) - \mathbf{x}^{(t)}$, for the same reverse diffusion process.

nealed Importance Sampling (AIS) (Neal, 2001), which uses a Markov chain which slowly converts one distribution into another to compute a ratio of normalizing constants. In (Burda et al., 2014) it is shown that AIS can also be performed using the reverse rather than forward trajectory. Langevin dynamics (Langevin, 1908), which are the stochastic realization of the Fokker-Planck equation, show how to define a Gaussian diffusion process which has any target distribution as its equilibrium. In (Suykens & Vandewalle, 1995) the Fokker-Planck equation is used to perform stochastic optimization. Finally, the Kolmogorov forward and backward equations (Feller, 1949) show that for many forward diffusion processes, the reverse diffusion processes can be described using the same functional form.

2. Algorithm

Our goal is to define a forward (or inference) diffusion process which converts any complex data distribution into a simple, tractable, distribution, and then learn a finite-time reversal of this diffusion process which defines our generative model distribution (See Figure 1). We first describe the forward, inference diffusion process. We then show how the reverse, generative diffusion process can be trained and used to evaluate probabilities. We also derive entropy bounds for the reverse process, and show how the learned distributions can be multiplied by any second distribution (e.g. as would be done to compute a posterior when inpainting or denoising an image).

2.1. Forward Trajectory

We label the data distribution $q\left(\mathbf{x}^{(0)}\right)$. The data distribution is gradually converted into a well behaved (analytically tractable) distribution $\pi\left(\mathbf{y}\right)$ by repeated application of a Markov diffusion kernel $T_\pi\left(\mathbf{y}|\mathbf{y}';\beta\right)$ for $\pi\left(\mathbf{y}\right)$, where β is the diffusion rate,

$$\pi\left(\mathbf{y}\right) = \int d\mathbf{y}' T_\pi\left(\mathbf{y}|\mathbf{y}';\beta\right)\pi\left(\mathbf{y}'\right) \quad (1)$$

$$q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}\right) = T_\pi\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)};\beta_t\right). \quad (2)$$

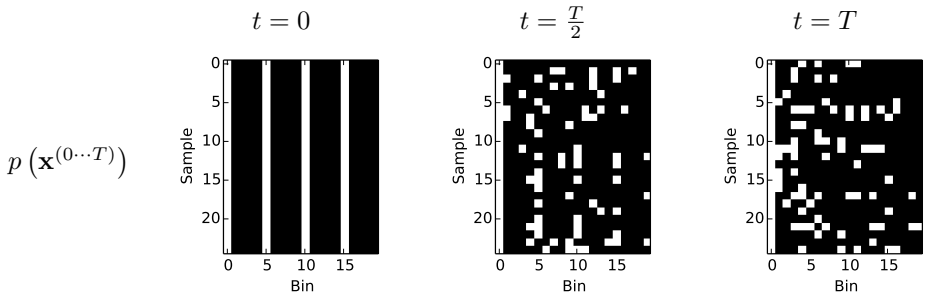

Figure 2. Binary sequence learning via binomial diffusion. A binomial diffusion model was trained on binary 'heartbeat' data, where a pulse occurs every 5th bin. Generated samples (left) are identical to the training data. The sampling procedure consists of initialization at independent binomial noise (right), which is then transformed into the data distribution by a binomial diffusion process, with trained bit flip probabilities. Each row contains an independent sample. For ease of visualization, all samples have been shifted so that a pulse occurs in the first column. In the raw sequence data, the first pulse is uniformly distributed over the first five bins.

Figure 3. The proposed framework trained on the CIFAR-10 (Krizhevsky & Hinton, 2009) dataset. *(a)* Example holdout data (similar to training data). *(b)* Holdout data corrupted with Gaussian noise of variance 1 (SNR = 1). *(c)* Denoised images, generated by sampling from the posterior distribution over denoised images conditioned on the images in (b). *(d)* Samples generated by the diffusion model.

The forward trajectory, corresponding to starting at the data distribution and performing T steps of diffusion, is thus

$$q\left(\mathbf{x}^{(0\cdots T)}\right) = q\left(\mathbf{x}^{(0)}\right)\prod_{t=1}^{T} q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}\right) \qquad (3)$$

For the experiments shown below, $q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}\right)$ corresponds to either Gaussian diffusion into a Gaussian distribution with identity-covariance, or binomial diffusion into an independent binomial distribution. Table App.1 gives the diffusion kernels for both Gaussian and binomial distributions.

2.2. Reverse Trajectory

The generative distribution will be trained to describe the same trajectory, but in reverse,

$$p\left(\mathbf{x}^{(T)}\right) = \pi\left(\mathbf{x}^{(T)}\right) \tag{4}$$

$$p\left(\mathbf{x}^{(0\cdots T)}\right) = p\left(\mathbf{x}^{(T)}\right)\prod_{t=1}^{T} p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right). \tag{5}$$

For both Gaussian and binomial diffusion, for continuous diffusion (limit of small step size β) the reversal of the diffusion process has the identical functional form as the forward process (Feller, 1949). Since $q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}\right)$ is a Gaussian (binomial) distribution, and if β_t is small, then $q\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)$ will also be a Gaussian (binomial) distribution. The longer the trajectory the smaller the diffusion rate β can be made.

During learning only the mean and covariance for a Gaussian diffusion kernel, or the bit flip probability for a binomial kernel, need be estimated. As shown in Table App.1, $\mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right)$ and $\mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right)$ are functions defining the mean and covariance of the reverse Markov transitions for a Gaussian, and $\mathbf{f}_b\left(\mathbf{x}^{(t)}, t\right)$ is a function providing the bit flip probability for a binomial distribution. The computational cost of running this algorithm is the cost of these functions, times the number of time-steps. For all results in this paper, multi-layer perceptrons are used to define these functions. A wide range of regression or function fitting techniques would be applicable however, including nonparameteric methods.

2.3. Model Probability

The probability the generative model assigns to the data is

$$p\left(\mathbf{x}^{(0)}\right) = \int d\mathbf{x}^{(1\cdots T)} p\left(\mathbf{x}^{(0\cdots T)}\right). \tag{6}$$

Naively this integral is intractable – but taking a cue from annealed importance sampling and the Jarzynski equality, we instead evaluate the relative probability of the forward and reverse trajectories, averaged over forward trajectories,

$$p\left(\mathbf{x}^{(0)}\right) = \int d\mathbf{x}^{(1\cdots T)} p\left(\mathbf{x}^{(0\cdots T)}\right)\frac{q\left(\mathbf{x}^{(1\cdots T)}|\mathbf{x}^{(0)}\right)}{q\left(\mathbf{x}^{(1\cdots T)}|\mathbf{x}^{(0)}\right)} \tag{7}$$

$$= \int d\mathbf{x}^{(1\cdots T)} q\left(\mathbf{x}^{(1\cdots T)}|\mathbf{x}^{(0)}\right)\cdot$$

$$p\left(\mathbf{x}^{(T)}\right)\prod_{t=1}^{T}\frac{p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)}{q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}\right)}. \tag{9}$$

This can be evaluated rapidly by averaging over samples from the forward trajectory $q\left(\mathbf{x}^{(1\cdots T)}|\mathbf{x}^{(0)}\right)$. For infinitesimal β the forward and reverse distribution over trajectories can be made identical (see Section 2.2). If they are identical then only a *single* sample from $q\left(\mathbf{x}^{(1\cdots T)}|\mathbf{x}^{(0)}\right)$ is required to exactly evaluate the above integral, as can be seen by substitution. This corresponds to the case of a quasi-static process in statistical physics (Spinney & Ford, 2013; Jarzynski, 2011).

2.4. Training

Training amounts to maximizing the model log likelihood,

$$L = \int d\mathbf{x}^{(0)} q\left(\mathbf{x}^{(0)}\right)\log p\left(\mathbf{x}^{(0)}\right) \tag{10}$$

==>

$$L \geq \int d\mathbf{x}^{(0\cdots T)} q\left(\mathbf{x}^{(0\cdots T)}\right)\cdot$$

$$\log\left[p\left(\mathbf{x}^{(T)}\right)\prod_{t=1}^{T}\frac{p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)}{q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}\right)}\right]. \tag{12}$$

As described in Appendix B, for our diffusion trajectories this reduces to,

$$L \geq K \tag{13}$$

$$= -\sum_{t=2}^{T}\int d\mathbf{x}^{(0)} d\mathbf{x}^{(t)} q\left(\mathbf{x}^{(0)}, \mathbf{x}^{(t)}\right)\cdot$$

$$D_{KL}\left(q\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}, \mathbf{x}^{(0)}\right)\middle\|p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)\right)$$

$$+ H_q\left(\mathbf{X}^{(T)}|\mathbf{X}^{(0)}\right) - H_q\left(\mathbf{X}^{(1)}|\mathbf{X}^{(0)}\right) - H_p\left(\mathbf{X}^{(T)}\right). \tag{14}$$

where the entropies and KL divergences can be analytically computed. The derivation of this bound parallels the derivation of the log likelihood bound in variational Bayesian methods.

As in Section 2.3 if the forward and reverse trajectories are identical, corresponding to a quasi-static process, then the inequality in Equation 13 becomes an equality.

Training consists of finding the reverse Markov transitions which maximize this lower bound on the log likelihood,

$$\left\{\hat{p}\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)\right\} = \underset{\left\{p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)\right\}}{\operatorname{argmax}} K. \tag{15}$$

The specific targets of estimation for Gaussian and binomial diffusion are given in Table App.1.

Thus, the task of estimating a probability distribution has been reduced to the task of performing regression on the functions which set the mean and covariance of a sequence of Gaussians (or set the state flip probability for a sequence of Bernoulli trials).

2.4.1. Setting the Diffusion Rate β_t

The choice of β_t in the forward trajectory is important for the performance of the trained model. In AIS, the right schedule of intermediate distributions can greatly improve the accuracy of the log partition function estimate (Grosse et al., 2013). In thermodynamics the schedule taken when moving between equilibrium distributions determines how much free energy is lost (Spinney & Ford, 2013; Jarzynski, 2011).

In the case of Gaussian diffusion, we learn[2] the forward diffusion schedule $\beta_{2\cdots T}$ by gradient ascent on K. The variance β_1 of the first step is fixed to a small constant to prevent overfitting. The dependence of samples from $q\left(\mathbf{x}^{(1\cdots T)}|\mathbf{x}^{(0)}\right)$ on $\beta_{1\cdots T}$ is made explicit by using 'frozen noise' – as in (Kingma & Welling, 2013) the noise is treated as an additional auxiliary variable, and held constant while computing partial derivatives of K wrt the parameters.

For binomial diffusion, the discrete state space makes gradient ascent with frozen noise impossible. We instead choose the forward diffusion schedule $\beta_{1\cdots T}$ to erase a constant fraction $\frac{1}{T}$ of the original signal per diffusion step, yielding a diffusion rate of $\beta_t = (T-t+1)^{-1}$.

2.5. Multiplying Distributions, and Computing Posteriors

Tasks such as computing a posterior in order to do signal denoising or inference of missing values requires multiplication of the model distribution $p\left(\mathbf{x}^{(0)}\right)$ with a second distribution, or bounded positive function, $r\left(\mathbf{x}^{(0)}\right)$, producing a new distribution $\tilde{p}\left(\mathbf{x}^{(0)}\right) \propto p\left(\mathbf{x}^{(0)}\right) r\left(\mathbf{x}^{(0)}\right)$.

Multiplying distributions is costly and difficult for many techniques, including variational autoencoders, GSNs, NADEs, and most graphical models. However, under a diffusion model it is straightforward, since the second distribution can be treated either as a small perturbation to each step in the diffusion process, or often exactly multiplied into each diffusion step. Figures 3 and 5 demonstrate the use of a diffusion model to perform denoising and inpainting of natural images. The following sections describe how to multiply distributions in the context of diffusion probabilistic models.

[2]Recent experiments suggest that it is just as effective to instead use the same fixed β_t schedule as for binomial diffusion.

2.5.1. Modified Marginal Distributions

First, in order to compute $\tilde{p}\left(\mathbf{x}^{(0)}\right)$, we multiply each of the intermediate distributions by a corresponding function $r\left(\mathbf{x}^{(t)}\right)$. We use a tilde above a distribution or Markov transition to denote that it belongs to a trajectory that has been modified in this way. $\tilde{p}\left(\mathbf{x}^{(0\cdots T)}\right)$ is the modified reverse trajectory, which starts at the distribution $\tilde{p}\left(\mathbf{x}^{(T)}\right) = \frac{1}{\tilde{Z}_T} p\left(\mathbf{x}^{(T)}\right) r\left(\mathbf{x}^{(T)}\right)$ and proceeds through the sequence of intermediate distributions

$$\tilde{p}\left(\mathbf{x}^{(t)}\right) = \frac{1}{\tilde{Z}_t} p\left(\mathbf{x}^{(t)}\right) r\left(\mathbf{x}^{(t)}\right), \tag{16}$$

where $\tilde{Z}_t$ is the normalizing constant for the tth intermediate distribution.

2.5.2. Modified Diffusion Steps

The Markov kernel $p\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t+1)}\right)$ for the reverse diffusion process obeys the equilibrium condition

$$p\left(\mathbf{x}^{(t)}\right) = \int d\mathbf{x}^{(t+1)} p\left(\mathbf{x}^{t)} \mid \mathbf{x}^{(t+1)}\right) p\left(\mathbf{x}^{t+1)}\right). \tag{17}$$

We wish the perturbed Markov kernel $\tilde{p}\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t+1)}\right)$ to instead obey the equilibrium condition for the perturbed distribution,

$$\tilde{p}\left(\mathbf{x}^{(t)}\right) = \int d\mathbf{x}^{(t+1)} \tilde{p}\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t+1)}\right) \tilde{p}\left(\mathbf{x}^{t+1)}\right), \tag{18}$$

$$p\left(\mathbf{x}^{(t)}\right) = \int d\mathbf{x}^{(t+1)} \tilde{p}\left(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t+1)}\right) \cdot \frac{\tilde{Z}_t r\left(\mathbf{x}^{(t+1)}\right)}{\tilde{Z}_{t+1} r\left(\mathbf{x}^{(t)}\right)} p\left(\mathbf{x}^{(t+1)}\right). \tag{20}$$

Eq (20) will be satisfied if

$$\tilde{p}\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t+1)}\right) = p\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t+1)}\right) \frac{\tilde{Z}_{t+1} r\left(\mathbf{x}^{(t)}\right)}{\tilde{Z}_t r\left(\mathbf{x}^{(t+1)}\right)}. \tag{21}$$

Equation 21 may not correspond to a normalized probability distribution, so we choose $\tilde{p}\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t+1)}\right)$ to be the corresponding normalized distribution

$$\tilde{p}\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t+1)}\right) = \frac{1}{\tilde{Z}_t\left(\mathbf{x}^{(t+1)}\right)} p\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t+1)}\right) r\left(\mathbf{x}^{(t)}\right), \tag{22}$$

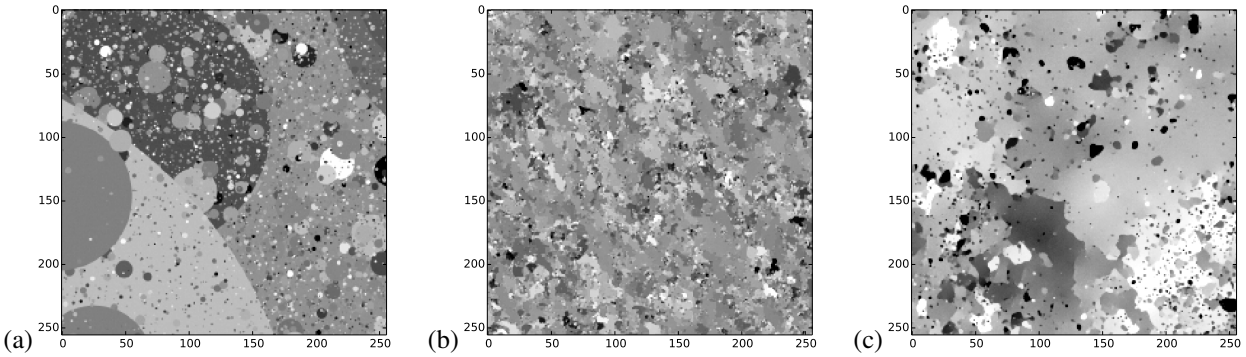

Figure 4. The proposed framework trained on dead leaf images (Jeulin, 1997; Lee et al., 2001). *(a)* Example training image. *(b)* A sample from the previous state of the art natural image model (Theis et al., 2012) trained on identical data, reproduced here with permission. *(c)* A sample generated by the diffusion model. Note that it demonstrates fairly consistent occlusion relationships, displays a multiscale distribution over object sizes, and produces circle-like objects, especially at smaller scales. As shown in Table 2, the diffusion model has the highest log likelihood on the test set.

where $\tilde{Z}_t\left(\mathbf{x}^{(t+1)}\right)$ is the normalization constant.

For a Gaussian, each diffusion step is typically very sharply peaked relative to $r\left(\mathbf{x}^{(t)}\right)$, due to its small variance. This means that $\frac{r\left(\mathbf{x}^{(t)}\right)}{r\left(\mathbf{x}^{(t+1)}\right)}$ can be treated as a small perturbation to $p\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t+1)}\right)$. A small perturbation to a Gaussian effects the mean, but not the normalization constant, so in this case Eq (21) == (22) (see Appendix C).

2.5.3. Applying $r\left(\mathbf{x}^{(t)}\right)$

If $r\left(\mathbf{x}^{(t)}\right)$ is sufficiently smooth, then it can be treated as a small perturbation to the reverse diffusion kernel $p\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t+1)}\right)$. In this case $\tilde{p}\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t+1)}\right)$ will have an identical functional form to $p\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t+1)}\right)$, but with perturbed mean for the Gaussian kernel, or with perturbed flip rate for the binomial kernel. The perturbed diffusion kernels are given in Table App.1, and are derived for the Gaussian in Appendix C.

If $r\left(\mathbf{x}^{(t)}\right)$ can be multiplied with a Gaussian (or binomial) distribution in closed form, then it can be directly multiplied with the reverse diffusion kernel $p\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t+1)}\right)$ in closed form. This applies in the case where $r\left(\mathbf{x}^{(t)}\right)$ consists of a delta function for some subset of coordinates, as in the inpainting example in Figure 5.

2.5.4. Choosing $r\left(\mathbf{x}^{(t)}\right)$

Typically, $r\left(\mathbf{x}^{(t)}\right)$ should be chosen to change slowly over the course of the trajectory. For the experiments in this paper we chose it to be constant,

$$r\left(\mathbf{x}^{(t)}\right) = r\left(\mathbf{x}^{(0)}\right). \tag{23}$$

Another convenient choice is $r\left(\mathbf{x}^{(t)}\right) = r\left(\mathbf{x}^{(0)}\right)^{\frac{T-t}{T}}$. Under this second choice $r\left(\mathbf{x}^{(t)}\right)$ makes no contribution to the starting distribution for the reverse trajectory. This guarantees that drawing the initial sample from $\tilde{p}\left(\mathbf{x}^{(T)}\right)$ for the reverse trajectory remains straightforward.

2.6. Entropy of Reverse Process

Since the forward process is known, we can derive upper and lower bounds on the conditional entropy of each step in the reverse trajectory, and thus on the log likelihood,

$$H_q\left(\mathbf{X}^{(t)}|\mathbf{X}^{(t-1)}\right) + H_q\left(\mathbf{X}^{(t-1)}|\mathbf{X}^{(0)}\right) - H_q\left(\mathbf{X}^{(t)}|\mathbf{X}^{(0)}\right) \leq H_q\left(\mathbf{X}^{(t-1)}|\mathbf{X}^{(t)}\right) \leq H_q\left(\mathbf{X}^{(t)}|\mathbf{X}^{(t-1)}\right), \tag{24}$$

where both the upper and lower bounds depend only on $q\left(\mathbf{x}^{(1\cdots T)}|\mathbf{x}^{(0)}\right)$, and can be analytically computed. The derivation is provided in Appendix A.

3. Experiments

We train diffusion probabilistic models on a variety of continuous datasets, and a binary dataset. We then demonstrate sampling from the trained model and inpainting of missing data, and compare model performance against other techniques. In all cases the objective function and gradient were computed using Theano (Bergstra & Breuleux, 2010). Model training was with SFO (Sohl-Dickstein et al., 2014), except for CIFAR-10. CIFAR-10 results used the

[3] An earlier version of this paper reported higher log likelihood bounds on CIFAR-10. These were the result of the model learning the 8-bit quantization of pixel values in the CIFAR-10 dataset. The log likelihood bounds reported here are instead for data that has been pre-processed by adding uniform noise to remove pixel quantization, as recommended in (Theis et al., 2015).

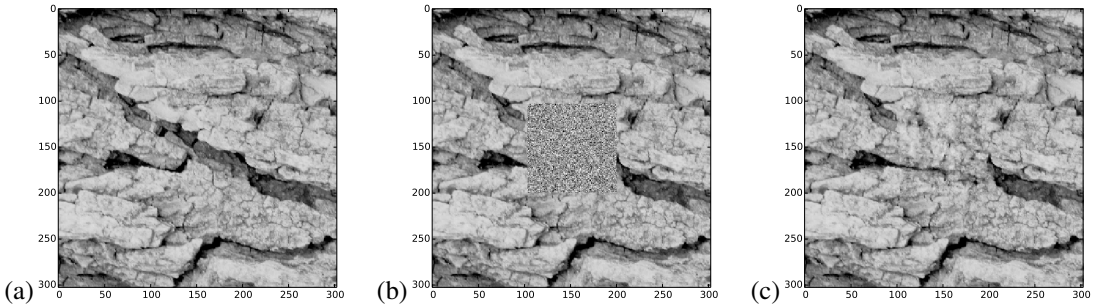

Figure 5. Inpainting. *(a)* A bark image from (Lazebnik et al., 2005). *(b)* The same image with the central 100×100 pixel region replaced with isotropic Gaussian noise. This is the initialization $\tilde{p}\left(\mathbf{x}^{(T)}\right)$ for the reverse trajectory. *(c)* The central 100×100 region has been inpainted using a diffusion probabilistic model trained on images of bark, by sampling from the posterior distribution over the missing region conditioned on the rest of the image. Note the long-range spatial structure, for instance in the crack entering on the left side of the inpainted region. The sample from the posterior was generated as described in Section 2.5, where $r\left(\mathbf{x}^{(0)}\right)$ was set to a delta function for known data, and a constant for missing data.

Dataset	K	$K - L_{null}$
Swiss Roll	2.35 bits	6.45 bits
Binary Heartbeat	-2.414 bits/seq.	12.024 bits/seq.
Bark	-0.55 bits/pixel	1.5 bits/pixel
Dead Leaves	1.489 bits/pixel	3.536 bits/pixel
CIFAR-10[3]	5.4 ± 0.2 bits/pixel	11.5 ± 0.2 bits/pixel
MNIST	See table 2	

Table 1. The lower bound K on the log likelihood, computed on a holdout set, for each of the trained models. See Equation 12. The right column is the improvement relative to an isotropic Gaussian or independent binomial distribution. L_{null} is the log likelihood of $\pi\left(\mathbf{x}^{(0)}\right)$. All datasets except for Binary Heartbeat were scaled by a constant to give them variance 1 before computing log likelihood.

Model	*Log Likelihood*
Dead Leaves	
MCGSM	1.244 bits/pixel
Diffusion	**1.489 bits/pixel**
MNIST	
Stacked CAE	174 ± 2.3 bits
DBN	199 ± 2.9 bits
Deep GSN	309 ± 1.6 bits
Diffusion	$\mathbf{317 \pm 2.7}$ **bits**
Adversarial net	325 ± 2.9 bits
Perfect model	349 ± 3.3 bits

Table 2. Log likelihood comparisons to other algorithms. Dead leaves images were evaluated using identical training and test data as in (Theis et al., 2012). MNIST log likelihoods were estimated using the Parzen-window code from (Goodfellow et al., 2014), with values given in bits, and show that our performance is comparable to other recent techniques. The perfect model entry was computed by applying the Parzen code to samples from the training data.

open source implementation of the algorithm, and RMSprop for optimization. The lower bound on the log likelihood provided by our model is reported for all datasets in Table 1. A reference implementation of the algorithm utilizing Blocks (van Merriënboer et al., 2015) is available at `https://github.com/Sohl-Dickstein/Diffusion-Probabilistic-Models`.

3.1. Toy Problems

3.1.1. Swiss Roll

A diffusion probabilistic model was built of a two dimensional swiss roll distribution, using a radial basis function network to generate $\mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right)$ and $\mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right)$. As illustrated in Figure 1, the swiss roll distribution was successfully learned. See Appendix Section D.1.1 for more details.

3.1.2. Binary Heartbeat Distribution

A diffusion probabilistic model was trained on simple binary sequences of length 20, where a 1 occurs every 5th time bin, and the remainder of the bins are 0, using a multilayer perceptron to generate the Bernoulli rates $\mathbf{f}_b\left(\mathbf{x}^{(t)}, t\right)$ of the reverse trajectory. The log likelihood under the true distribution is $\log_2\left(\frac{1}{5}\right) = -2.322$ bits per sequence. As can be seen in Figure 2 and Table 1 learning was nearly perfect. See Appendix Section D.1.2 for more details.

3.2. Images

We trained Gaussian diffusion probabilistic models on several image datasets. The multi-scale convolutional archi-

tecture shared by these experiments is described in Appendix Section D.2.1, and illustrated in Figure D.1.

3.2.1. DATASETS

MNIST In order to allow a direct comparison against previous work on a simple dataset, we trained on MNIST digits (LeCun & Cortes, 1998). Log likelihoods relative to (Bengio et al., 2012; Bengio & Thibodeau-Laufer, 2013; Goodfellow et al., 2014) are given in Table 2. Samples from the MNIST model are given in Appendix Figure App.1. Our training algorithm provides an asymptotically consistent lower bound on the log likelihood. However most previous reported results on continuous MNIST log likelihood rely on Parzen-window based estimates computed from model samples. For this comparison we therefore estimate MNIST log likelihood using the Parzen-window code released with (Goodfellow et al., 2014).

CIFAR-10 A probabilistic model was fit to the training images for the CIFAR-10 challenge dataset (Krizhevsky & Hinton, 2009). Samples from the trained model are provided in Figure 3.

Dead Leaf Images Dead leaf images (Jeulin, 1997; Lee et al., 2001) consist of layered occluding circles, drawn from a power law distribution over scales. They have an analytically tractable structure, but capture many of the statistical complexities of natural images, and therefore provide a compelling test case for natural image models. As illustrated in Table 2 and Figure 4, we achieve state of the art performance on the dead leaves dataset.

Bark Texture Images A probabilistic model was trained on bark texture images (T01-T04) from (Lazebnik et al., 2005). For this dataset we demonstrate that it is straightforward to evaluate or generate from a posterior distribution, by inpainting a large region of missing data using a sample from the model posterior in Figure 5.

4. Conclusion

We have introduced a novel algorithm for modeling probability distributions that enables exact sampling and evaluation of probabilities and demonstrated its effectiveness on a variety of toy and real datasets, including challenging natural image datasets. For each of these tests we used a similar basic algorithm, showing that our method can accurately model a wide variety of distributions. Most existing density estimation techniques must sacrifice modeling power in order to stay tractable and efficient, and sampling or evaluation are often extremely expensive. The core of our algorithm consists of estimating the reversal of a Markov diffusion chain which maps data to a noise distribution; as the number of steps is made large, the reversal distribution of each diffusion step becomes simple and easy to estimate. The result is an algorithm that can learn a fit to any data distribution, but which remains tractable to train, *exactly* sample from, and evaluate, and under which it is straightforward to manipulate conditional and posterior distributions.

Acknowledgements

We thank Lucas Theis, Subhaneil Lahiri, Ben Poole, Diederik P. Kingma, Taco Cohen, Philip Bachman, and Aäron van den Oord for extremely helpful discussion, and Ian Goodfellow for Parzen-window code. We thank Khan Academy and the Office of Naval Research for funding Jascha Sohl-Dickstein, and we thank the Office of Naval Research and the Burroughs-Wellcome, Sloan, and James S. McDonnell foundations for funding Surya Ganguli.

References

Barron, J. T., Biggin, M. D., Arbelaez, P., Knowles, D. W., Keranen, S. V., and Malik, J. Volumetric Semantic Segmentation Using Pyramid Context Features. In *2013 IEEE International Conference on Computer Vision*, pp. 3448–3455. IEEE, December 2013. ISBN 978-1-4799-2840-8. doi: 10.1109/ICCV.2013.428.

Bengio, Y. and Thibodeau-Laufer, E. Deep generative stochastic networks trainable by backprop. *arXiv preprint arXiv:1306.1091*, 2013.

Bengio, Y., Mesnil, G., Dauphin, Y., and Rifai, S. Better Mixing via Deep Representations. *arXiv preprint arXiv:1207.4404*, July 2012.

Bergstra, J. and Breuleux, O. Theano: a CPU and GPU math expression compiler. *Proceedings of the Python for Scientific Computing Conference (SciPy)*, 2010.

Besag, J. Statistical Analysis of Non-Lattice Data. *The Statistician, 24(3), 179-195*, 1975.

Bishop, C., Svensén, M., and Williams, C. GTM: The generative topographic mapping. *Neural computation*, 1998.

Bornschein, J. and Bengio, Y. Reweighted Wake-Sleep. *International Conference on Learning Representations*, June 2015.

Burda, Y., Grosse, R. B., and Salakhutdinov, R. Accurate and Conservative Estimates of MRF Log-likelihood using Reverse Annealing. *arXiv:1412.8566*, December 2014.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear Independent Components Estimation. *arXiv:1410.8516*, pp. 11, October 2014.

Feller, W. On the theory of stochastic processes, with particular reference to applications. In *Proceedings of the [First] Berkeley Symposium on Mathematical Statistics and Probability*. The Regents of the University of California, 1949.

Gershman, S. J. and Blei, D. M. A tutorial on Bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1–12, 2012.

Gneiting, T. and Raftery, A. E. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Nets. *Advances in Neural Information Processing Systems*, 2014.

Gregor, K., Danihelka, I., Mnih, A., Blundell, C., and Wierstra, D. Deep AutoRegressive Networks. *arXiv preprint arXiv:1310.8499*, October 2013.

Grosse, R. B., Maddison, C. J., and Salakhutdinov, R. Annealing between distributions by averaging moments. In *Advances in Neural Information Processing Systems*, pp. 2769–2777, 2013.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8): 1771–1800, 2002.

Hinton, G. E. The wake-sleep algorithm for unsupervised neural networks). *Science*, 1995.

Hyvärinen, A. Estimation of non-normalized statistical models using score matching. *Journal of Machine Learning Research*, 6:695–709, 2005.

Jarzynski, C. Equilibrium free-energy differences from nonequilibrium measurements: A master-equation approach. *Physical Review E*, January 1997.

Jarzynski, C. Equalities and inequalities: irreversibility and the second law of thermodynamics at the nanoscale. *Annu. Rev. Condens. Matter Phys.*, 2011.

Jeulin, D. Dead leaves models: from space tesselation to random functions. *Proc. of the Symposium on the Advances in the Theory and Applications of Random Sets*, 1997.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Kavukcuoglu, K., Ranzato, M., and LeCun, Y. Fast inference in sparse coding algorithms with applications to object recognition. *arXiv preprint arXiv:1010.3467*, 2010.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. *International Conference on Learning Representations*, December 2013.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. *Computer Science Department University of Toronto Tech. Rep.*, 2009.

Langevin, P. Sur la théorie du mouvement brownien. *CR Acad. Sci. Paris*, 146(530-533), 1908.

Larochelle, H. and Murray, I. The neural autoregressive distribution estimator. *Journal of Machine Learning Research*, 2011.

Lazebnik, S., Schmid, C., and Ponce, J. A sparse texture representation using local affine regions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27 (8):1265–1278, 2005.

LeCun, Y. and Cortes, C. The MNIST database of handwritten digits. 1998.

Lee, A., Mumford, D., and Huang, J. Occlusion models for natural images: A statistical study of a scale-invariant dead leaves model. *International Journal of Computer Vision*, 2001.

Lyu, S. Unifying Non-Maximum Likelihood Learning Objectives with Minimum KL Contraction. *Advances in Neural Information Processing Systems 24*, pp. 64–72, 2011.

MacKay, D. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 1995.

Murphy, K. P., Weiss, Y., and Jordan, M. I. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pp. 467–475. Morgan Kaufmann Publishers Inc., 1999.

Neal, R. Annealed importance sampling. *Statistics and Computing*, January 2001.

Ozair, S. and Bengio, Y. Deep Directed Generative Autoencoders. *arXiv:1410.0630*, October 2014.

Parry, M., Dawid, A. P., Lauritzen, S., and Others. Proper local scoring rules. *The Annals of Statistics*, 40(1):561–592, 2012.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, January 2014.

Rippel, O. and Adams, R. P. High-Dimensional Probability Estimation with Deep Density Models. *arXiv:1410.8516*, pp. 12, February 2013.

Schmidhuber, J. Learning factorial codes by predictability minimization. *Neural Computation*, 1992.

Sminchisescu, C., Kanaujia, A., and Metaxas, D. Learning joint top-down and bottom-up processes for 3D visual inference. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pp. 1743–1752. IEEE, 2006.

Sohl-Dickstein, J., Battaglino, P., and DeWeese, M. New Method for Parameter Estimation in Probabilistic Models: Minimum Probability Flow. *Physical Review Letters*, 107(22):11–14, November 2011a. ISSN 0031-9007. doi: 10.1103/PhysRevLett.107.220601.

Sohl-Dickstein, J., Battaglino, P. B., and DeWeese, M. R. Minimum Probability Flow Learning. *International Conference on Machine Learning*, 107(22):11–14, November 2011b. ISSN 0031-9007. doi: 10.1103/PhysRevLett.107.220601.

Sohl-Dickstein, J., Poole, B., and Ganguli, S. Fast large-scale optimization by unifying stochastic gradient and quasi-Newton methods. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 604–612, 2014.

Spinney, R. and Ford, I. Fluctuation Relations : A Pedagogical Overview. *arXiv preprint arXiv:1201.6381*, pp. 3–56, 2013.

Stuhlmüller, A., Taylor, J., and Goodman, N. Learning stochastic inverses. *Advances in Neural Information Processing Systems*, 2013.

Suykens, J. and Vandewalle, J. Nonconvex optimization using a Fokker-Planck learning machine. In *12th European Conference on Circuit Theory and Design*, 1995.

T, P. Convergence condition of the TAP equation for the infinite-ranged Ising spin glass model. *J. Phys. A: Math. Gen. 15 1971*, 1982.

Tanaka, T. Mean-field theory of Boltzmann machine learning. *Physical Review Letters E*, January 1998.

Theis, L., Hosseini, R., and Bethge, M. Mixtures of conditional Gaussian scale mixtures applied to multiscale image representations. *PloS one*, 7(7):e39857, 2012.

Theis, L., van den Oord, A., and Bethge, M. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Uria, B., Murray, I., and Larochelle, H. RNADE: The real-valued neural autoregressive density-estimator. *Advances in Neural Information Processing Systems*, 2013a.

Uria, B., Murray, I., and Larochelle, H. A Deep and Tractable Density Estimator. *arXiv:1310.1757*, pp. 9, October 2013b.

van Merriënboer, B., Chorowski, J., Serdyuk, D., Bengio, Y., Bogdanov, D., Dumoulin, V., and Warde-Farley, D. Blocks and Fuel. *Zenodo*, May 2015. doi: 10.5281/zenodo.17721.

Welling, M. and Hinton, G. A new learning algorithm for mean field Boltzmann machines. *Lecture Notes in Computer Science*, January 2002.

Yao, L., Ozair, S., Cho, K., and Bengio, Y. On the Equivalence Between Deep NADE and Generative Stochastic Networks. In *Machine Learning and Knowledge Discovery in Databases*, pp. 322–336. Springer, 2014.

Appendix

A. Conditional Entropy Bounds Derivation

The conditional entropy $H_q\left(\mathbf{X}^{(t-1)}|\mathbf{X}^{(t)}\right)$ of a step in the reverse trajectory is

$$H_q\left(\mathbf{X}^{(t-1)},\mathbf{X}^{(t)}\right) = H_q\left(\mathbf{X}^{(t)},\mathbf{X}^{(t-1)}\right) \tag{25}$$

$$H_q\left(\mathbf{X}^{(t-1)}|\mathbf{X}^{(t)}\right) + H_q\left(\mathbf{X}^{(t)}\right) = H_q\left(\mathbf{X}^{(t)}|\mathbf{X}^{(t-1)}\right) + H_q\left(\mathbf{X}^{(t-1)}\right) \tag{26}$$

$$H_q\left(\mathbf{X}^{(t-1)}|\mathbf{X}^{(t)}\right) = H_q\left(\mathbf{X}^{(t)}|\mathbf{X}^{(t-1)}\right) + H_q\left(\mathbf{X}^{(t-1)}\right) - H_q\left(\mathbf{X}^{(t)}\right) \tag{27}$$

An upper bound on the entropy change can be constructed by observing that $\pi\left(\mathbf{y}\right)$ is the maximum entropy distribution. This holds without qualification for the binomial distribution, and holds for variance 1 training data for the Gaussian case. For the Gaussian case, training data must therefore be scaled to have unit norm for the following equalities to hold. It need not be whitened. The upper bound is derived as follows,

$$H_q\left(\mathbf{X}^{(t)}\right) \geq H_q\left(\mathbf{X}^{(t-1)}\right) \tag{28}$$

$$H_q\left(\mathbf{X}^{(t-1)}\right) - H_q\left(\mathbf{X}^{(t)}\right) \leq 0 \tag{29}$$

$$H_q\left(\mathbf{X}^{(t-1)}|\mathbf{X}^{(t)}\right) \leq H_q\left(\mathbf{X}^{(t)}|\mathbf{X}^{(t-1)}\right). \tag{30}$$

A lower bound on the entropy difference can be established by observing that additional steps in a Markov chain do not increase the information available about the initial state in the chain, and thus do not decrease the conditional entropy of the initial state,

$$H_q\left(\mathbf{X}^{(0)}|\mathbf{X}^{(t)}\right) \geq H_q\left(\mathbf{X}^{(0)}|\mathbf{X}^{(t-1)}\right) \tag{31}$$

$$H_q\left(\mathbf{X}^{(t-1)}\right) - H_q\left(\mathbf{X}^{(t)}\right) \geq H_q\left(\mathbf{X}^{(0)}|\mathbf{X}^{(t-1)}\right) + H_q\left(\mathbf{X}^{(t-1)}\right) - H_q\left(\mathbf{X}^{(0)}|\mathbf{X}^{(t)}\right) - H_q\left(\mathbf{X}^{(t)}\right) \tag{32}$$

$$H_q\left(\mathbf{X}^{(t-1)}\right) - H_q\left(\mathbf{X}^{(t)}\right) \geq H_q\left(\mathbf{X}^{(0)},\mathbf{X}^{(t-1)}\right) - H_q\left(\mathbf{X}^{(0)},\mathbf{X}^{(t)}\right) \tag{33}$$

$$H_q\left(\mathbf{X}^{(t-1)}\right) - H_q\left(\mathbf{X}^{(t)}\right) \geq H_q\left(\mathbf{X}^{(t-1)}|\mathbf{X}^{(0)}\right) - H_q\left(\mathbf{X}^{(t)}|\mathbf{X}^{(0)}\right) \tag{34}$$

$$H_q\left(\mathbf{X}^{(t-1)}|\mathbf{X}^{(t)}\right) \geq H_q\left(\mathbf{X}^{(t)}|\mathbf{X}^{(t-1)}\right) + H_q\left(\mathbf{X}^{(t-1)}|\mathbf{X}^{(0)}\right) - H_q\left(\mathbf{X}^{(t)}|\mathbf{X}^{(0)}\right). \tag{35}$$

Combining these expressions, we bound the conditional entropy for a single step,

$$H_q\left(\mathbf{X}^{(t)}|\mathbf{X}^{(t-1)}\right) \geq H_q\left(\mathbf{X}^{(t-1)}|\mathbf{X}^{(t)}\right) \geq H_q\left(\mathbf{X}^{(t)}|\mathbf{X}^{(t-1)}\right) + H_q\left(\mathbf{X}^{(t-1)}|\mathbf{X}^{(0)}\right) - H_q\left(\mathbf{X}^{(t)}|\mathbf{X}^{(0)}\right), \tag{36}$$

where both the upper and lower bounds depend only on the conditional forward trajectory $q\left(\mathbf{x}^{(1\cdots T)}|\mathbf{x}^{(0)}\right)$, and can be analytically computed.

B. Log Likelihood Lower Bound

The lower bound on the log likelihood is

$$L \geq K \tag{37}$$

$$K = \int d\mathbf{x}^{(0\cdots T)} q\left(\mathbf{x}^{(0\cdots T)}\right) \log\left[p\left(\mathbf{x}^{(T)}\right)\prod_{t=1}^{T}\frac{p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)}{q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}\right)}\right] \tag{38}$$

B.1. Entropy of $p\left(\mathbf{X}^{(T)}\right)$

We can peel off the contribution from $p\left(\mathbf{X}^{(T)}\right)$, and rewrite it as an entropy,

$$K = \int d\mathbf{x}^{(0\cdots T)} q\left(\mathbf{x}^{(0\cdots T)}\right) \sum_{t=1}^{T} \log\left[\frac{p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)}{q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}\right)}\right] + \int d\mathbf{x}^{(T)} q\left(\mathbf{x}^{(T)}\right) \log p\left(\mathbf{x}^{(T)}\right) \tag{40}$$

$$= \int d\mathbf{x}^{(0\cdots T)} q\left(\mathbf{x}^{(0\cdots T)}\right) \sum_{t=1}^{T} \log\left[\frac{p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)}{q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}\right)}\right] + \int d\mathbf{x}^{(T)} q\left(\mathbf{x}^{(T)}\right) \log \pi\left(\mathbf{x}^{T}\right) \tag{41}$$

By design, the cross entropy to $\pi\left(\mathbf{x}^{(t)}\right)$ is constant under our diffusion kernels, and equal to the entropy of $p\left(\mathbf{x}^{(T)}\right)$. Therefore,

$$K = \sum_{t=1}^{T} \int d\mathbf{x}^{(0\cdots T)} q\left(\mathbf{x}^{(0\cdots T)}\right) \log\left[\frac{p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)}{q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}\right)}\right] - H_p\left(\mathbf{X}^{(T)}\right). \tag{43}$$

B.2. Remove the edge effect at $t = 0$

In order to avoid edge effects, we set the final step of the reverse trajectory to be identical to the corresponding forward diffusion step,

$$p\left(\mathbf{x}^{(0)}|\mathbf{x}^{(1)}\right) = q\left(\mathbf{x}^{(1)}|\mathbf{x}^{(0)}\right) \frac{\pi\left(\mathbf{x}^{(0)}\right)}{\pi\left(\mathbf{x}^{(1)}\right)} = T_\pi\left(\mathbf{x}^{(0)}|\mathbf{x}^{(1)}; \beta_1\right). \tag{44}$$

We then use this equivalence to remove the contribution of the first time-step in the sum,

$$K = \sum_{t=2}^{T} \int d\mathbf{x}^{(0\cdots T)} q\left(\mathbf{x}^{(0\cdots T)}\right) \log\left[\frac{p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)}{q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}\right)}\right] + \int d\mathbf{x}^{(0)} d\mathbf{x}^{(1)} q\left(\mathbf{x}^{(0)}, \mathbf{x}^{(1)}\right) \log\left[\frac{q\left(\mathbf{x}^{(1)}|\mathbf{x}^{(0)}\right) \pi\left(\mathbf{x}^{(0)}\right)}{q\left(\mathbf{x}^{(1)}|\mathbf{x}^{(0)}\right) \pi\left(\mathbf{x}^{(1)}\right)}\right] - H_p\left(\mathbf{X}^{(T)}\right) \tag{45}$$

$$= \sum_{t=2}^{T} \int d\mathbf{x}^{(0\cdots T)} q\left(\mathbf{x}^{(0\cdots T)}\right) \log\left[\frac{p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)}{q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}\right)}\right] - H_p\left(\mathbf{X}^{(T)}\right), \tag{46}$$

where we again used the fact that by design $-\int d\mathbf{x}^{(t)} q\left(\mathbf{x}^{(t)}\right) \log \pi\left(\mathbf{x}^{(t)}\right) = H_p\left(\mathbf{X}^{(T)}\right)$ is a constant for all t.

B.3. Rewrite in terms of posterior $q\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(0)}\right)$

Because the forward trajectory is a Markov process,

$$K = \sum_{t=2}^{T} \int d\mathbf{x}^{(0\cdots T)} q\left(\mathbf{x}^{(0\cdots T)}\right) \log\left[\frac{p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)}{q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}, \mathbf{x}^{(0)}\right)}\right] - H_p\left(\mathbf{X}^{(T)}\right). \tag{47}$$

Using Bayes' rule ==>

$$K = \sum_{t=2}^{T} \int d\mathbf{x}^{(0\cdots T)} q\left(\mathbf{x}^{(0\cdots T)}\right) \log\left[\frac{p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)}{q\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}, \mathbf{x}^{(0)}\right)} \frac{q\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(0)}\right)}{q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(0)}\right)}\right] - H_p\left(\mathbf{X}^{(T)}\right). \tag{48}$$

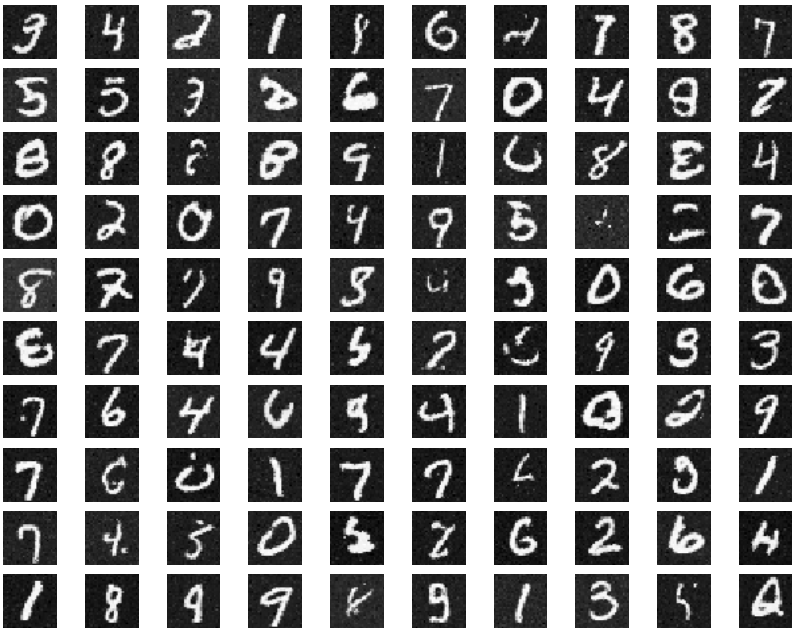

Figure App.1. Samples from a diffusion probabilistic model trained on MNIST digits. Note that unlike many MNIST sample figures, these are true samples rather than the mean of the Gaussian or binomial distribution from which samples would be drawn.

B.4. Rewrite in terms of KL divergences and entropies

We then recognize that several terms are conditional entropies,

$$K = \sum_{t=2}^{T} \int d\mathbf{x}^{(0\cdots T)} q\left(\mathbf{x}^{(0\cdots T)}\right) \log\left[\frac{p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)}{q\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}, \mathbf{x}^{(0)}\right)}\right] + H_q\left(\mathbf{X}^{(T)}|\mathbf{X}^{(0)}\right) - H_q\left(\mathbf{X}^{(1)}|\mathbf{X}^{(0)}\right) - H_p\left(\mathbf{X}^{(T)}\right). \tag{49}$$

Finally we transform the log ratio of probability distributions into a KL divergence,

$$\begin{aligned} K = &-\sum_{t=2}^{T} \int d\mathbf{x}^{(0)} d\mathbf{x}^{(t)} q\left(\mathbf{x}^{(0)}, \mathbf{x}^{(t)}\right) D_{KL}\left(q\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}, \mathbf{x}^{(0)}\right) \middle|\middle| p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)\right) \\ &+ H_q\left(\mathbf{X}^{(T)}|\mathbf{X}^{(0)}\right) - H_q\left(\mathbf{X}^{(1)}|\mathbf{X}^{(0)}\right) - H_p\left(\mathbf{X}^{(T)}\right). \end{aligned} \tag{51}$$

Note that the entropies can be analytically computed, and the KL divergence can be analytically computed given $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(t)}$.

		Gaussian	*Binomial*
Well behaved (analytically tractable) distribution	$\pi\left(\mathbf{x}^{(T)}\right) =$	$\mathcal{N}\left(\mathbf{x}^{(T)}; \mathbf{0}, \mathbf{I}\right)$	$\mathcal{B}\left(\mathbf{x}^{(T)}; 0.5\right)$
Forward diffusion kernel	$q\left(\mathbf{x}^{(t)}\vert\mathbf{x}^{(t-1)}\right) =$	$\mathcal{N}\left(\mathbf{x}^{(t)}; \mathbf{x}^{(t-1)}\sqrt{1-\beta_t}, \mathbf{I}\beta_t\right)$	$\mathcal{B}\left(\mathbf{x}^{(t)}; \mathbf{x}^{(t-1)}\left(1-\beta_t\right) + 0.5\beta_t\right)$
Reverse diffusion kernel	$p\left(\mathbf{x}^{(t-1)}\vert\mathbf{x}^{(t)}\right) =$	$\mathcal{N}\left(\mathbf{x}^{(t-1)}; \mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right), \mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right)\right)$	$\mathcal{B}\left(\mathbf{x}^{(t-1)}; \mathbf{f}_b\left(\mathbf{x}^{(t)}, t\right)\right)$
Training targets		$\mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right), \mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right), \beta_{1\cdots T}$	$\mathbf{f}_b\left(\mathbf{x}^{(t)}, t\right)$
Forward distribution	$q\left(\mathbf{x}^{(0\cdots T)}\right) =$	$q\left(\mathbf{x}^{(0)}\right)\prod_{t=1}^{T} q\left(\mathbf{x}^{(t)}\vert\mathbf{x}^{(t-1)}\right)$	
Reverse distribution	$p\left(\mathbf{x}^{(0\cdots T)}\right) =$	$\pi\left(\mathbf{x}^{(T)}\right)\prod_{t=1}^{T} p\left(\mathbf{x}^{(t-1)}\vert\mathbf{x}^{(t)}\right)$	
Log likelihood	$L =$	$\int d\mathbf{x}^{(0)} q\left(\mathbf{x}^{(0)}\right)\log p\left(\mathbf{x}^{(0)}\right)$	
Lower bound on log likelihood	$K =$	$-\sum_{t=2}^{T}\mathbb{E}_{q\left(\mathbf{x}^{(0)},\mathbf{x}^{(t)}\right)}\left[D_{KL}\left(q\left(\mathbf{x}^{(t-1)}\vert\mathbf{x}^{(t)}, \mathbf{x}^{(0)}\right)\vert\vert p\left(\mathbf{x}^{(t-1)}\vert\mathbf{x}^{(t)}\right)\right)\right] + H_q\left(\mathbf{X}^{(T)}\vert\mathbf{X}^{(0)}\right) - H_q\left(\mathbf{X}^{(1)}\vert\mathbf{X}^{(0)}\right) - H_p\left(\mathbf{X}^{(T)}\right)$	
Perturbed reverse diffusion kernel	$\tilde{p}\left(\mathbf{x}^{(t-1)}\vert\mathbf{x}^{(t)}\right) =$	$\mathcal{N}\left(x^{(t-1)}; \mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right) + \mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right)\left.\frac{\partial \log r\left(\mathbf{x}^{(t-1)'}\right)}{\partial \mathbf{x}^{(t-1)'}}\right\vert_{\mathbf{x}^{(t-1)'}=f_\mu\left(\mathbf{x}^{(t)},t\right)}, \mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right)\right)$	$\mathcal{B}\left(x_i^{(t-1)}; \frac{c_i^{t-1}d_i^{t-1}}{x_i^{t-1}d_i^{t-1}+(1-c_i^{t-1})(1-d_i^{t-1})}\right)$

Table App.1.
for the perturbed Bernoulli trials $b_i^t = \mathbf{x}^{(t-1)}\left(1-\beta_t\right) + 0.5\beta_t$, $c_i^t = \left[\mathbf{f}_b\left(\mathbf{x}^{(t+1)}, t\right)\right]_i$, and $d_i^t = r\left(x_i^{(t)} = 1\right)$, and the distribution is given for a single bit i.

C. Perturbed Gaussian Transition

We wish to compute $\tilde{p}\left(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)}\right)$. For notational simplicity, let $\mu = \mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right)$, $\Sigma = \mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right)$, and $\mathbf{y} = \mathbf{x}^{(t-1)}$. Using this notation,

$$\tilde{p}\left(\mathbf{y} \mid \mathbf{x}^{(t)}\right) \propto p\left(\mathbf{y} \mid \mathbf{x}^{(t)}\right) r\left(\mathbf{y}\right) \tag{52}$$

$$= \mathcal{N}\left(\mathbf{y}; \mu, \Sigma\right) r\left(\mathbf{y}\right). \tag{53}$$

$$\propto \exp\left[-E\left(\mathbf{y}\right)\right]$$

$$\text{where } E_r\left(\mathbf{y}\right) = -\log r\left(\mathbf{y}\right),\ E\left(\mathbf{y}\right) = \frac{1}{2}\left(\mathbf{y}-\mu\right)^T \Sigma^{-1}\left(\mathbf{y}-\mu\right) + E_r\left(\mathbf{y}\right). \tag{54}$$

If $E_r\left(\mathbf{y}\right)$ is smooth relative to $\frac{1}{2}\left(\mathbf{y}-\mu\right)^T \Sigma^{-1}\left(\mathbf{y}-\mu\right)$, then we can approximate it using its Taylor expansion around μ. One sufficient condition is that the eigenvalues of the Hessian of $E_r\left(\mathbf{y}\right)$ are everywhere much smaller magnitude than the eigenvalues of Σ^{-1}. We then have

$$E_r\left(\mathbf{y}\right) \approx E_r\left(\mu\right) + \left(\mathbf{y}-\mu\right)\mathbf{g} \tag{56}$$

where $\mathbf{g} = \left.\frac{\partial E_r\left(\mathbf{y}'\right)}{\partial \mathbf{y}'}\right|_{\mathbf{y}'=\mu}$. ==>

$$E\left(\mathbf{y}\right) \approx \frac{1}{2}\left(\mathbf{y}-\mu\right)^T \Sigma^{-1}\left(\mathbf{y}-\mu\right) + \left(\mathbf{y}-\mu\right)^T \mathbf{g} + \text{constant} \tag{57}$$

$$= \frac{1}{2}\mathbf{y}^T\Sigma^{-1}\mathbf{y} - \frac{1}{2}\mathbf{y}^T\Sigma^{-1}\mu - \frac{1}{2}\mu^T\Sigma^{-1}\mathbf{y} + \frac{1}{2}\mathbf{y}^T\Sigma^{-1}\Sigma\mathbf{g} + \frac{1}{2}\mathbf{g}^T\Sigma\Sigma^{-1}\mathbf{y} + \text{constant} \tag{58}$$

$$= \frac{1}{2}\left(\mathbf{y}-\mu+\Sigma\mathbf{g}\right)^T \Sigma^{-1}\left(\mathbf{y}-\mu+\Sigma\mathbf{g}\right) + \text{constant}. \tag{59}$$

==>

$$\tilde{p}\left(\mathbf{y} \mid \mathbf{x}^{(t)}\right) \approx \mathcal{N}\left(\mathbf{y}; \mu - \Sigma\mathbf{g}, \Sigma\right). \tag{60}$$

Substituting back in the original formalism, ==>

$$\tilde{p}\left(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)}\right) \approx \mathcal{N}\left(x^{(t-1)}; \mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right) + \mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right) \left.\frac{\partial \log r\left(\mathbf{x}^{(t-1)'}\right)}{\partial \mathbf{x}^{(t-1)'}}\right|_{\mathbf{x}^{(t-1)'}=f_\mu\left(\mathbf{x}^{(t)},t\right)}, \mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right)\right). \tag{61}$$

D. Experimental Details

D.1. Toy Problems

D.1.1. Swiss Roll

A probabilistic model was built of a two dimensional swiss roll distribution. The generative model $p\left(\mathbf{x}^{(0\cdots T)}\right)$ consisted of 40 time steps of Gaussian diffusion initialized at an identity-covariance Gaussian distribution. A (normalized) radial basis function network with a single hidden layer and 16 hidden units was trained to generate the mean and covariance functions $\mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right)$ and a diagonal $\mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right)$ for the reverse trajectory. The top, readout, layer for each function was learned independently for each time step, but for all other layers weights were shared across all time steps and both functions. The top layer output of $\mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right)$ was passed through a sigmoid to restrict it between 0 and 1. As can be seen in Figure 1, the swiss roll distribution was successfully learned.

D.1.2. Binary Heartbeat Distribution

A probabilistic model was trained on simple binary sequences of length 20, where a 1 occurs every 5th time bin, and the remainder of the bins are 0. The generative model consisted of 2000 time steps of binomial diffusion initialized at an independent binomial distribution with the same mean activity as the data ($p\left(x_i^{(T)} = 1\right) = 0.2$). A multilayer perceptron with sigmoid nonlinearities, 20 input units and three hidden layers with 50 units each was trained to generate the Bernoulli rates $\mathbf{f}_b\left(\mathbf{x}^{(t)}, t\right)$ of the reverse trajectory. The top, readout, layer was learned independently for each time step, but for all other layers weights were shared across all time steps. The top layer output was passed through a sigmoid to restrict it between 0 and 1. As can be seen in Figure 2, the heartbeat distribution was successfully learned. The log likelihood under the true generating process is $\log_2\left(\frac{1}{5}\right) = -2.322$ bits per sequence. As can be seen in Figure 2 and Table 1 learning was nearly perfect.

D.2. Images

D.2.1. Architecture

Readout In all cases, a convolutional network was used to produce a vector of outputs $\mathbf{y}_i \in \mathcal{R}^{2J}$ for each image pixel i. The entries in $\mathbf{y}_i$ are divided into two equal sized subsets, $\mathbf{y}^\mu$ and $\mathbf{y}^\Sigma$.

Temporal Dependence The convolution output $\mathbf{y}^\mu$ is used as per-pixel weighting coefficients in a sum over time-dependent "bump" functions, generating an output $\mathbf{z}_i^\mu \in \mathcal{R}$ for each pixel i,

$$\mathbf{z}_i^\mu = \sum_{j=1}^{J} \mathbf{y}_{ij}^\mu g_j\left(t\right). \tag{62}$$

The bump functions consist of

$$g_j\left(t\right) = \frac{\exp\left(-\frac{1}{2w^2}\left(t-\tau_j\right)^2\right)}{\sum_{k=1}^{J}\exp\left(-\frac{1}{2w^2}\left(t-\tau_k\right)^2\right)}, \tag{63}$$

where $\tau_j \in (0, T)$ is the bump center, and w is the spacing between bump centers. $\mathbf{z}^\Sigma$ is generated in an identical way, but using y^Σ.

For all image experiments a number of timesteps $T = 1000$ was used, except for the bark dataset which used $T = 500$.

Mean and Variance Finally, these outputs are combined to produce a diffusion mean and variance prediction for each pixel i,

$$\Sigma_{ii} = \sigma\left(z_i^\Sigma + \sigma^{-1}\left(\beta_t\right)\right), \tag{64}$$

$$\mu_i = \left(x_i - z_i^\mu\right)\left(1 - \Sigma_{ii}\right) + z_i^\mu. \tag{65}$$

where both Σ and μ are parameterized as a perturbation around the forward diffusion kernel $T_\pi\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}; \beta_t\right)$, and z_i^μ is the mean of the equilibrium distribution that would result from applying $p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)$ many times. Σ is restricted to be a diagonal matrix.

Multi-Scale Convolution We wish to accomplish goals that are often achieved with pooling networks – specifically, we wish to discover and make use of long-range and multi-scale dependencies in the training data. However, since the network output is a vector of coefficients for every pixel it is important to generate a full resolution rather than down-sampled feature map. We therefore define multi-scale-convolution layers that consist of the following steps:

1. Perform mean pooling to downsample the image to multiple scales. Downsampling is performed in powers of two.
2. Performing convolution at each scale.
3. Upsample all scales to full resolution, and sum the resulting images.
4. Perform a pointwise nonlinear transformation, consisting of a soft relu ($\log\left[1 + \exp\left(\cdot\right)\right]$).

The composition of the first three linear operations resembles convolution by a multiscale convolution kernel, up to blocking artifacts introduced by upsampling. This method of achieving multiscale convolution was described in (Barron et al., 2013).

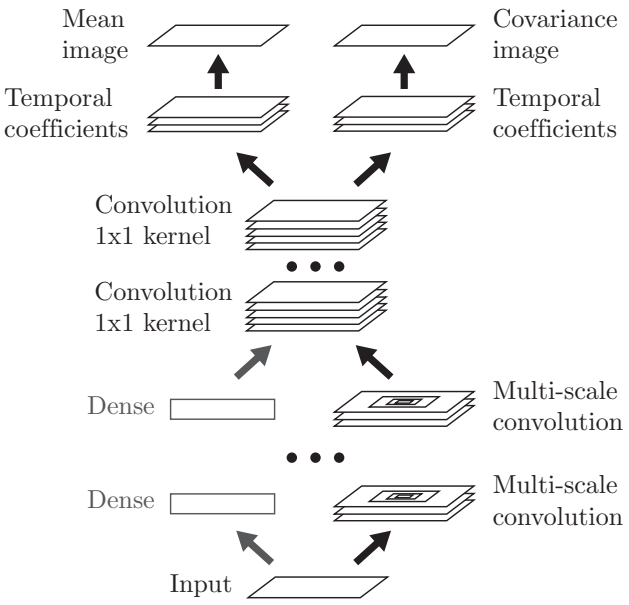

Figure D.1. Network architecture for mean function $\mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right)$ and covariance function $\mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right)$, for experiments in Section 3.2. The input image $\mathbf{x}^{(t)}$ passes through several layers of multi-scale convolution (Section D.2.1). It then passes through several convolutional layers with 1×1 kernels. This is equivalent to a dense transformation performed on each pixel. A linear transformation generates coefficients for readout of both mean $\mu^{(t)}$ and covariance $\Sigma^{(t)}$ for each pixel. Finally, a time dependent readout function converts those coefficients into mean and covariance images, as described in Section D.2.1. For CIFAR-10 a dense (or fully connected) pathway was used in parallel to the multi-scale convolutional pathway. For MNIST, the dense pathway was used to the exclusion of the multi-scale convolutional pathway.

Dense Layers Dense (acting on the full image vector) and kernel-width-1 convolutional (acting separately on the feature vector for each pixel) layers share the same form. They consist of a linear transformation, followed by a tanh nonlinearity.