Denoising Deep Generative Models

Gabriel Loaiza-Ganem
Layer 6 AI
gabriel@layer6.ai

Brendan Leigh Ross
Layer 6 AI
brendan@layer6.ai

Luhuan Wu
Columbia University
lw2827@columbia.edu

John P. Cunningham
Columbia University
jpc2181@columbia.edu

Jesse C. Cresswell
Layer 6 AI
jesse@layer6.ai

Anthony L. Caterini
Layer 6 AI
anthony@layer6.ai

Abstract

Likelihood-based deep generative models have recently been shown to exhibit pathological behaviour under the manifold hypothesis as a consequence of using high-dimensional densities to model data with low-dimensional structure. In this paper we propose two methodologies aimed at addressing this problem. Both are based on adding Gaussian noise to the data to remove the dimensionality mismatch during training, and both provide a denoising mechanism whose goal is to sample from the model as though no noise had been added to the data. Our first approach is based on Tweedie’s formula, and the second on models which take the variance of added noise as a conditional input. We show that surprisingly, while well motivated, these approaches only sporadically improve performance over not adding noise, and that other methods of addressing the dimensionality mismatch are more empirically adequate.

1 Introduction

The manifold hypothesis [Bengio et al., 2013], which states that high-dimensional data often lies on an unknown low-dimensional manifold embedded in ambient space, aims to explain the success of deep learning: neural networks would be unable to learn good low-dimensional representations if there was no low-dimensional structure to begin with. There have also been empirical studies estimating the intrinsic dimension of commonly-used image datasets, finding it is indeed much lower than its corresponding ambient dimension [Pope et al., 2021, Tempczyk et al., 2022, Brown et al., 2022]. Along with these empirical verifications of the low-dimensional structure present in data, there has also been a surge in research in deep generative models (DGMs) attempting to directly account for the manifold hypothesis [Gemici et al., 2016, Dai and Wipf, 2019, Saremi and Hyvärinen, 2019, Rezende et al., 2020, Brehmer and Cranmer, 2020, Mathieu and Nickel, 2020, Arbel et al., 2021, Kothari et al., 2021, Caterini et al., 2021, Ross and Cresswell, 2021, De Bortoli et al., 2022, Loaiza-Ganem et al., 2022, Ross et al., 2022]. This is a relevant line of research, especially for likelihood-based models, which have been shown to suffer from manifold overfitting under the manifold hypothesis [Dai and Wipf, 2019, Loaiza-Ganem et al., 2022], a surprising phenomenon where likelihoods can become arbitrarily large without recovering the ground truth distribution, even in the presence of an infinite amount of data.

Current DGMs that account for the manifold hypothesis require either non-trivial modifications from their corresponding fully-dimensional counterparts [Brehmer and Cranmer, 2020, Arbel et al., 2021, Kothari et al., 2021, Caterini et al., 2021, Ross and Cresswell, 2021, Ross et al., 2022], or require training more models [Dai and Wipf, 2019, Loaiza-Ganem et al., 2022]. In this paper we propose two slight modifications to existing full-dimensional likelihood-based models so as to enable them to directly account for the manifold hypothesis. In our first proposed method, we train off-the-shelf
We will also assume that the manifold hypothesis holds; i.e., that \( p \) where \( c \) with the intention of recovering \( \hat{p} \).

Throughout this work we will assume that we have access to samples from a distribution \( p \) values of \( \sigma \) consistency with most of the DGM literature. We nonetheless opt for this notation for conditional models that we use in appendix A.2.

Many DGMs, including VAEs and NFs, admit conditional variants [Sohn et al., 2015, Agrawal and Dukkipati, 2016] to learn the conditional distribution of the (noisy) data given \( \sigma \), and denoising is carried out by using \( \sigma = 0 \) when sampling from the model.

In spite of being strongly motivated, both of our proposed procedures do not obtain consistent improvements over simply using full-dimensional models, unlike some of the aforementioned more involved manifold-aware models. We hope that this surprising result will lead into further research aiming to understand the interplay between the manifold hypothesis and DGMs.

2 Background

2.1 Likelihood-based DGMs and Tweedie’s formula

Throughout this work we will assume that we have access to samples from a distribution \( p(x) \) in \( \mathbb{R}^D \). We will also assume that the manifold hypothesis holds; i.e., that \( p(x) \) is supported on an embedded submanifold of \( \mathbb{R}^D \) of dimension less than \( D \). Our discussions apply to all continuous likelihood-based models such as variational autoencoders (VAEs) [Kingma and Welling, 2014, Rezende et al., 2014], normalizing flows (NFs) [Dinh et al., 2017, Kingma and Dhariwal, 2018, Behrmann et al., 2019, Chen et al., 2019, Durkan et al., 2019, Cornish et al., 2020], energy-based models [Du and Mordatch, 2019], and continuous autoregressive models [Uria et al., 2013, Theis andBethge, 2015], in which a density \( p_\eta(x) \) over \( \mathbb{R}^D \) is constructed through neural networks parameterized by \( \eta \), and trained through maximum-likelihood

\[
\eta^* = \arg \max_\eta \mathbb{E}_{p(x)}[\log p_\eta(x)]
\]  

(1)

with the intention of recovering \( p(x) \). Note that we slightly abuse notation, as depending on the model being used, \( p_\eta(x) \) might not be directly available. For example, VAEs maximize a lower bound of the log-likelihood, and energy-based models do not directly have access to \( p_\eta(x) \) although they still aim to solve (1) through gradient estimates. We nonetheless keep the notation \( p_\eta(x) \) for the sake of generality and provide a review of the DGMs that we will use in our experiments section in appendix A.1. Likelihood-based DGMs do not properly account for the manifold hypothesis, since \( p_\eta(x) \) is a high-dimensional density. As we will see in subsection 2.2, this modelling choice results in pathological behaviour, which we aim to address through Tweedie’s formula and conditional DGMs.

**Tweedie’s formula** Assume we are given a sample \( x_\sigma \), obtained by first sampling \( x \) from \( p(x) \), and then adding Gaussian noise \( x_\sigma := x + \sigma \epsilon \), where \( \sigma > 0 \) and \( \epsilon \sim \mathcal{N}(0, I_D) \). Tweedie’s formula [Robbins, 1956] provides the best estimate \( \hat{x}_\sigma \) (in mean squared error) of \( x \) obtainable from \( x_\sigma \):

\[
\hat{x}_\sigma := \mathbb{E}_{p(x|x_\sigma)}[x] = x_\sigma + \sigma^2 \nabla_{x_\sigma} \log p(x_\sigma).
\]  

(2)

Surprisingly, computing \( \hat{x}_\sigma \) does not require access to \( p(x) \) nor to \( p(x|x_\sigma) \), only the marginal of \( x_\sigma \), \( p(x_\sigma) \), is involved.

**Conditional models** Many DGMs, including VAEs and NFs, admit conditional variants [Sohn et al., 2015, Agrawal and Dukkipati, 2016]. These models are trained to maximize

\[
\eta^* = \arg \max_\eta \mathbb{E}_{p(x,c)}[\log p_\eta(x|c)],
\]  

(3)

where \( c \) is a conditioning variable. For example, \( c \) might be a class label, in which case \( p(x, c) = p(x)p(c|x) \) where \( p(c|x) \) is a point mass at the class label corresponding to \( x \); or \( c \) could also specify a subset of coordinates of \( x \), in which case \( p(c|x) \) selects the conditioning coordinates, often independently of \( x \), i.e. \( p(c|x) = p(c) \). Here \( p_\eta(x|c) \) is now a density defined through neural networks parameterized by \( \eta \), whose inputs now also include \( c \). Once again, we include details of the conditional models that we use in appendix A.2.

1While the notation \( p(x) \) suggests this is a density in the Lebesgue sense, we highlight that formally \( p \) is a probability measure as it is supported on a low-dimensional manifold. We nonetheless opt for this notation for consistency with most of the DGM literature.
2.2 Manifold overfitting

Manifold overfitting [Dai and Wipf, 2019, Loaiza-Ganem et al., 2022] shows that solving (1) will in general not result in \( p_\eta^*(x) \) recovering \( p(x) \), as the likelihood \( p_\eta^*(x) \) can achieve arbitrarily large values by concentrating around the manifold over which \( p(x) \) is supported, without getting the correct distribution on the manifold. Figure 1 illustrates this phenomenon. Here \( p(x) \) is supported on a 1-dimensional curve (manifold) in \( \mathbb{R}^2 \), and the plotted choice of \( p_\eta(x) \) concentrates around the correct manifold, but does so in an incorrect way, assigning more probability to the wrong region of the curve. If \( p_\eta(x) \) is flexible enough, this spiking behaviour can increase, resulting in unbounded likelihoods even if the model is not close to \( p(x) \). Manifold overfitting strongly motivates the development of likelihood-based DGMs which properly account for the manifold hypothesis.

3 Methods

3.1 Tweedie Denoising DGMs

Here we propose to train a DGM not to learn \( p(x) \) directly, but rather its noisy version \( p(x_\sigma) \), which is the density obtained after convolving \( p(x) \) with Gaussian noise with variance \( \sigma^2 \), where \( \sigma \) is treated as a hyperparameter. This amounts to solving

\[
\eta^* = \arg \max_{\eta} \mathbb{E}_{p(x_\sigma)}[\log p_\eta(x_\sigma)]
\]

instead of (1). The intuition is simple: by adding noise, the target distribution \( p(x_\sigma) \) is not supported on a low-dimensional manifold anymore (formally, it becomes absolutely continuous with respect to the Lebesgue measure in \( \mathbb{R}^D \)), which should theoretically avoid manifold overfitting. We point out that adding Gaussian noise is a common practice (see section 4), although Loaiza-Ganem et al. [2022] found that, by itself, doing so does not fully avoid manifold overfitting in practice. We thus propose to add an additional denoising step through Tweedie’s formula in the hope of improving empirical performance by properly accounting for the fact that the learned distribution is not directly \( p(x) \). Once we have the trained model \( p_\eta^*(x_\sigma) \), given a sample \( x_\sigma \) from the model, we correct it through Tweedie’s formula (2):

\[
\hat{x}_\sigma \leftarrow x_\sigma + \sigma^2 \nabla_{x_\sigma} \log p_\eta^*(x_\sigma).
\]

We highlight the simplicity of using Tweedie denoising DGMs: we only have to add Gaussian noise to training data, train an off-the-shelf likelihood-based DGM \( p_\eta(x_\sigma) \), and do a post-hoc correction through (5) at sample time.

3.2 Conditional Denoising DGMs

We also propose to use conditional models (3) to learn the conditional distribution of noisy data, conditional on the standard deviation of the added noise by maximizing

\[
\eta^* = \arg \max_{\eta} \mathbb{E}_{p(x_\sigma|\sigma)p(\sigma)}[\log p_\eta(x_\sigma|\sigma)],
\]

where \( p(\sigma) \) is an arbitrary distribution over \( \sigma \), e.g. uniform on \((0, C)\) for some hyperparameter \( C > 0 \). Note that since we are now treating \( \sigma \) as random instead of a fixed hyperparameter, we write \( p(x_\sigma|\sigma) \) instead of \( p(x_\sigma) \) for the distribution of noisy data at a given noise level. We also highlight the simplicity of using conditional denoising DGMs: during training we sample \( \sigma \) along with each datapoint, add corresponding Gaussian noise to the data, and condition the DGM on
σ. When sampling from a trained model, we simply sample from \( p_{\eta_1}(x_\sigma | \sigma = 0) \). The intuition is similar to that of Tweedie denoising DGMs: by adding noise, we hope the model manages to properly learn \( p(x_\sigma | \sigma) \) for every \( \sigma \) in the appropriate range, e.g. \((0, C)\), and that the \( \sigma = 0 \) case \( p(x) = p(x_\sigma | \sigma = 0) \) is also learned by “continuity” over \( \sigma \).

4 Related work

As mentioned in the introduction, most deep generative modelling methods that account for the manifold hypothesis without explicitly adding noise to the data deviate substantially from their full-dimensional counterparts. While not in itself a problem, this property does prevent “plugging in” any likelihood-based DGM from the vast existing literature to our context of interest. [Dai and Wipf 2019] and [Loaiza-Ganem et al. 2022] propose to first obtain low-dimensional representations of the data, and then train a likelihood-based DGM on these representations, which results in the added complexity of having to specify two models and not having a single end-to-end training procedure. Indeed, our original motivation was to tackle the same problem in a simpler way.

Adding noise to data before training DGMs is a common practice [Vincent et al. 2008, Vincent 2011, Alain and Bengio 2014, Theis et al. 2016, Chae et al. 2021], albeit not always directly motivated as a way to account for the manifold hypothesis. In the context of accounting for the manifold hypothesis within likelihood-based DGMs, several methods based on adding noise have been proposed, although these tend to be model-specific. For example, the method of [Zhang et al. 2020] can only be applied to VAEs, those of [Horvat and Plister 2021a,b] and [Cunningham and Fiterau 2021] to NFs, and that of [Meng et al. 2021a] to autoregressive models. [Song and Ermon 2019] follow a similar approach for score-based models [Hyvärinen 2005]. Similarly, Tweedie’s formula has been used in the context of DGMs before [Saremi and Hyvärinen 2019, Meng et al. 2021b], although these uses are once again model-specific. Our motivation for this paper was to propose a widely applicable methodology, compatible with any likelihood-based DGM.

We were also motivated by diffusion models, which have extremely strong empirical performance. These models can be understood as likelihood-based models [Ho et al. 2020, Song et al. 2021a], or as score-based models in a stochastic differential equation setting [Song et al. 2021b]. Diffusion models learn how to slowly transform noisy samples into samples from the data distribution (i.e., to denoise them). In our notation this roughly translates to transforming samples \( x_{\sigma_2} \) from \( p(x_{\sigma_2} | \sigma_2) \) into samples \( x_{\sigma_1} \) from \( p(x_{\sigma_1} | \sigma_1) \) for a multitude of values \( \sigma_1 < \sigma_2 \). Importantly, the structure of diffusion models implies that these models learn not only the target distribution \( p(x) \), but also noisy versions of it \( p(x_{\sigma_1} | \sigma) \) at different noise levels \( \sigma \). Furthermore, in contrast to likelihood-based models which can experience manifold overfitting, diffusion models are known to converge under the manifold hypothesis [Pidsstrigach 2022, De Bortoli 2022]. All these properties of diffusion models motivated our conditional models, with the hope that learning \( p(x_{\sigma} | \sigma) \) for a continuum of values of \( \sigma \) could address manifold overfitting.

5 Experiments

5.1 Results

Although as mentioned previously our methods can in principle be applied to any likelihood-based model, in this section we focus on VAEs and NFs as both have commonly-used conditional versions. For example, energy-based models [Du and Mordatch 2019] keep a sample buffer during training, and naively adding the conditioning variable as input to the energy function would result in the buffer containing samples at different noise levels: this would confound any observed poor performance of conditional denoising, and attempting to improve upon the buffer falls outside of the scope of this work. Similarly, we also omit autoregressive models from our experiments, as most well-performing versions of these models are discrete rather than continuous [van den Oord et al. 2016, Salimans et al. 2017] (and are thus not susceptible to manifold overfitting), and proposing performant continuous autoregressive models also falls outside the scope of our work. We use the prefixes “ND-”, “TD-”, and “CD-” to denote models trained with added Gaussian noise with no denoising step, Tweedie denoising, and conditional denoising, respectively. All experimental details are provided in appendix A.3 and our code is publicly available at https://github.com/layer6ai-labs/denoising_dgms.
Table 1 shows comparisons of all the considered models using the FID score [Heusel et al., 2017] for the MNIST [LeCun, 1998], F MNIST [Xiao et al., 2017], SVHN [Netzer et al., 2011], and CIFAR-10 [Krizhevsky et al., 2009] datasets. We opted to use FID scores as a measure of how well the models recover \( p(x) \) instead of test log-likelihoods since the latter are, by definition, unable to detect manifold overfitting. We highlight that we tuned \( \sigma \) for the Tweedie denoising models, as well as \( C \) for the conditional denoising models. We can see that, surprisingly, the TD-V AE (TD-NF) and CD-V AE (CD-NF) models do not consistently outperform their V AE (NF) and ND-V AE (ND-NF) baselines: the only instances of denoising models obtaining a non-marginal improvement over their baselines are the CD-V AE on SVHN and the CD-NF on MNIST. We also tried annealing \( \sigma \) (for Tweedie denoising models) and \( C \) (for conditional denoising models), but found results did not significantly change. Not only do our denoising models not outperform simply adding Gaussian noise, but in some cases denoising can even hamper performance: for example CD-VAEs on FMNIST and TD-NFs on MNIST both do not consistently outperform simply adding Gaussian noise, but in some cases denoising the problem significantly – albeit marginally – underperform their non-denoised alternatives.

<table>
<thead>
<tr>
<th>MODEL</th>
<th>MNIST</th>
<th>FMNIST</th>
<th>SVHN</th>
<th>CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAE</td>
<td>197.4 ± 1.5</td>
<td>188.9 ± 1.8</td>
<td>311.5 ± 6.9</td>
<td>270.3 ± 3.2</td>
</tr>
<tr>
<td>ND-VAE</td>
<td>199.9 ± 1.4</td>
<td>185.7 ± 2.0</td>
<td>317.8 ± 3.3</td>
<td>264.5 ± 0.5</td>
</tr>
<tr>
<td>TD-VAE</td>
<td>199.1 ± 0.8</td>
<td>190.4 ± 3.3</td>
<td>309.9 ± 8.9</td>
<td>263.9 ± 0.9</td>
</tr>
<tr>
<td>CD-VAE</td>
<td>197.4 ± 0.2</td>
<td>195.8 ± 2.1</td>
<td>290.0 ± 4.4</td>
<td>262.4 ± 0.3</td>
</tr>
<tr>
<td>NF</td>
<td>137.2 ± 3.4</td>
<td>110.5 ± 0.9</td>
<td>231.9 ± 22.0</td>
<td>222.7 ± 3.9</td>
</tr>
<tr>
<td>ND-NF</td>
<td>103.2 ± 0.4</td>
<td>72.3 ± 0.8</td>
<td>224.2 ± 4.4</td>
<td>222.8 ± 2.2</td>
</tr>
<tr>
<td>TD-NF</td>
<td>105.6 ± 0.5</td>
<td>70.6 ± 0.4</td>
<td>222.0 ± 5.7</td>
<td>222.9 ± 1.2</td>
</tr>
<tr>
<td>CD-NF</td>
<td>87.4 ± 0.5</td>
<td>73.3 ± 0.3</td>
<td>206.0 ± 7.1</td>
<td>225.4 ± 0.7</td>
</tr>
</tbody>
</table>

5.2 Discussion

We conjecture that the issue with these methods remains related to manifold overfitting: the target noisy distribution might still be very peaked around the manifold and difficult to learn, in which case \( p_{\sigma}(x_{\sigma}) \) (or \( p_{\sigma^*}(x_{\sigma} | \sigma) \) for conditional models) might not be close to its target \( p(x_{\sigma}) \) (or \( p(x_{\sigma} | \sigma) \)) and just concentrate around the manifold. If this is the case, we should expect neither Tweedie’s formula nor conditioning on \( \sigma = 0 \) to properly sample from \( p(x) \). In other words, the noise being added might not be enough to numerically overcome manifold overfitting. We hypothesize that using more powerful models and further tuning how the noise is added might alleviate the situation.

Another potential explanation for the performance of Tweedie denoising models is that the value of \( \sigma \) that we used (0.01 for most models, which as previously mentioned was found by tuning this hyperparameter) is too small, and thus the update from (5) ends up barely correcting the samples. We find this explanation is not fully satisfactory as due to manifold overfitting, one should expect the \( \| \nabla_{x_{\sigma}} \log p_{\sigma^*}(x_{\sigma}) \|_2 \) term to become larger as \( \sigma \) becomes smaller (since the density should become “spiker” around the manifold), potentially offsetting the small correction size \( \sigma \). Additionally, this explanation does not provide any intuition for the observed performance of our conditional models.

Yet another explanation would be that the manifold hypothesis does not hold, and that thus manifold overfitting does not happen to begin with. We find this hypothesis particularly hard to believe: first, the manifold hypothesis is a sensible and intuitive way to think about high-dimensional data [Bengio et al., 2013] that been empirically verified in various ways [Pope et al., 2021, Tempczyk et al., 2022, Brown et al., 2022]. Second, the works of [Dai and Wipf, 2019] and [Loaiza-Ganem et al., 2022] show very clear empirical improvements by avoiding manifold overfitting.

Finally, it is also likely the case that observed data already represents noisy observations from a true low-dimensional manifold, and that injecting further noise to accommodate our approaches makes the problem significantly harder at the outset. However, the relative good performance of the no denoising (ND) models over their vanilla versions without any added noise suggests that adding noise is not making the problem harder.

6 Conclusions

In this paper we propose Tweedie denoising and conditional denoising with the goal of alleviating manifold overfitting. Our methodologies are based on adding Gaussian noise to the data before
training a likelihood-based DGM, along with a way of denoising samples from the resulting trained models. Unexpectedly, our denoising approaches do not provide meaningful empirical improvements; we suspect that manifold overfitting remains a culprit in the failure of these models. We hope that our work will incentivize the community to further understand this intriguing result, as well as the role of the manifold hypothesis in DGMs.

References


## Appendix

### A.1 Likelihood-based DGMs

**VAEs** Variational autoencoders [Kingma and Welling, 2014, Rezende et al., 2014] model \( x \in \mathbb{R}^D \) through a lower-dimensional latent variable \( z \in \mathbb{R}^d \). A prior \( p(z) \), often taken as a standard Gaussian, is specified, and the conditional distribution \( p_\theta(x|z) \) is parameterized by a neural network with parameters \( \theta \). In Gaussian VAEs, \( p_\theta(x|z) \) (often referred to as the decoder) is given by a Gaussian whose parameters are given by the output of the neural network parameterized by \( \theta \). Since the \( x \)-marginal of the model, \( \int p(z)p_\theta(x|z)dz \), is not tractable, VAEs cannot be trained through maximum-likelihood directly. Instead, an auxiliary distribution \( q_\phi(z|x) \) (the encoder) is introduced, with the aim of approximating the posterior \( p_\theta(z|x) \). The approximate posterior \( q_\phi(z|x) \) is often taken as a low-dimensional Gaussian whose parameters are given by the output of a neural network parameterized by \( \phi \). A lower bound to the low-likelihood is then maximized over \( \eta = (\theta, \phi) \):

\[
\eta^* = \arg \max_{\eta} \mathbb{E}_{p(x)} \left[ \mathbb{E}_{q_\phi(z|x)} \left[ \log p_\theta(x|z) \right] - \text{KL}(q_\phi(z|x)||p(z)) \right].
\]  

(7)
NFs Normalizing flows [Dinh et al., 2017] construct \( p_\eta(x) \) as the density obtained by transforming \( x = f_\eta(z) \), where \( f_\eta : \mathbb{R}^D \rightarrow \mathbb{R}^D \) is a bijective neural network parameterized by \( \eta \), and \( z \sim p(z) \), where \( p(z) \) is often taken as a standard Gaussian. By the change of variable formula, we have that

\[
p_\eta(x) = p(z) \left| \det \frac{\partial f_\eta^{-1}}{\partial x} \right|,
\]

where \( z = f_\eta^{-1}(x) \) and \( \frac{\partial f_\eta^{-1}}{\partial x} \) is the Jacobian matrix of \( f_\eta^{-1} \) evaluated at \( x \). As long as \( f_\eta \) is constructed in such a way that its inverse and the determinant in (8) can be efficiently computed, NFs can be trained through maximum-likelihood [1]. In practice, \( f_\eta^{-1} \) is constructed (rather than \( f_\eta \)) by stacking coupling layers. Each coupling layer is itself a bijective transformation with the aforementioned properties enabling tractability. Coupling layers proceed by partitioning their input into two blocks \( x = (x_A, x_B) \), and the corresponding output \( z = (z_A, z_B) \) is given by:

\[
\begin{aligned}
z_A &= x_A \\
z_B &= g(x_B; h_\eta(x_A)),
\end{aligned}
\]

where \( g(\cdot; h) : \mathbb{R} \rightarrow \mathbb{R} \) is an invertible function parameterized by \( h \) which is applied element-wise to \( x_B \), and \( h_\eta \) is a neural network mapping \( x_A \) to the parameters of \( g \). For example, \( g \) could be an affine transformation [Dinh et al., 2017] parameterized by two scalars \( h \in \mathbb{R}^2 \), or a monotonic rational quadratic spline [Durkan et al., 2019]. It is easy to check that the determinant of a coupling layer is triangular (up to a permutation), and can thus be computed efficiently. Finally, \( f_\eta^{-1} \) is constructed by stacking multiple coupling layers, each with its own partition, on top of each other.

A.2 Conditional likelihood-based DGMs

VAEs Variational autoencoders can straightforwardly be made into conditional models [Sohn et al., 2015] by modifying (7) as follows:

\[
\eta^* = \arg \max_\eta \mathbb{E}_{p(x, c)} \left[ \mathbb{E}_{q_\phi(z|x, c)} [\log p_\theta(x|z)] - \mathbb{KL}(q_\phi(z|x, c)||p(z|c)) \right],
\]

where the prior \( p(z|c) \) can now also depend on the conditioning variable \( c \), although this dependency can be omitted, i.e. \( p(z|c) = p(z) \). In practice, this change amounts to having the encoder and decoder neural networks take \( c \) as input in addition to \( x \) and \( z \), respectively. We omitted the dependency of \( p(z|c) \) in our experiments for the sake of simplicity, although in some preliminary experiments we did not observe any significant changes by including this dependency.

NFs Similarly to VAEs, NFs can be made into conditional models [Agrawal and Dukkipati, 2016] defining a conditional density \( p_\eta(x|c) \) simply by using the conditioning variable \( c \) as an input to coupling layers, i.e. modifying (9) to

\[
\begin{aligned}
z_A &= x_A \\
z_B &= g(x_B; h_\eta(x_A, c)),
\end{aligned}
\]

which then enables the model to be trained through (5). This strategy has been used in different contexts [Atanov et al., 2019, Ardizzone et al., 2019, Winkler et al., 2019].

A.3 Experimental details

For all experiments, we used the Adam optimizer [Kingma and Ba, 2015], and gradient clipping with a value of 10. For ND- models, we tuned \( \sigma \) by reporting the best value out of \( \sigma \in \{0.005, 0.01, 0.05, 0.1, 0.5\} \). We use the same value of \( \sigma \) for the TD- versions of the models. For the CD- models, we similarly tune \( C \) by selecting the best value in \( \{0.005, 0.01, 0.05, 0.1, 0.5\} \).

VAEs We preprocessed the data by scaling it to \([0, 1]\). The latent space is 20-dimensional, the learning rate is 0.001, and we train the models for 100 epochs. We use fully-connected architectures for MNIST and FMNIST, both for the encoder and decoder, which both have a single hidden layer with 256 units. For SVHN and CIFAR-10, we use convolutional encoders and decoders, which have \((32, 32, 16, 16)\) and \((16, 16, 32, 32)\) hidden channels, respectively, with a fully-connected layer at the
end of the encoder, and one at the beginning of the decoder. We use ReLU activations throughout.
For the conditional versions on MNIST and FMNIST, we found that simply concatenating the 1-
dimensional $\sigma$ with a 784-dimensional datapoint $x$ resulted in the network just ignoring $\sigma$. In order
to provide the inductive bias that the conditioning is a relevant feature to the model, we used two
additional networks, which we call conditioning networks. The encoder conditioning network takes $\sigma$
as input and outputs a 64-dimensional representation, which is then concatenated with $x$ before being
fed into the encoder. Similarly, for the decoder, the conditioning network takes $\sigma$ as input and outputs
an 8-dimensional representation, which is concatenated with $z$ before being fed into the decoder.
Both conditioning networks have a single hidden layer with 256 units and use ReLU activations.
The conditioning network for the encoder on SVHN and CIFAR-10 remains the same, although the
conditioning network for the encoder now outputs a $32 \times 32$ channel, which is concatenated with the
input $x$ before being fed to the encoder.

**NFs** We also preprocessed the data by scaling it to $[0, 1]$ for MNIST and FMNIST, and by whitening
it for SVHN and CIFAR-10. We used a learning rate of 0.0005, and train for 100 epochs with early
stopping on validation log-likelihood with a patience of 30 epochs. We use rational quadratic spline
flows [Durkan et al., 2019] with 128 units, 4 layers, and 3 blocks per layer. For the conditional models,
as in VAEs, we use a conditioning network which takes $\sigma$ as input and outputs a 64-dimensional
representation, which is used as additional input in the coupling layers (11). The conditioning network
has a single hidden layer with 256 units and uses a ReLU activation.