Variational Diffusion Models

Diederik P. Kingma*, **Tim Salimans***, **Ben Poole, Jonathan Ho**
Google Research

Abstract

Diffusion-based generative models have demonstrated a capacity for perceptually impressive synthesis, but can they also be great likelihood-based models? We answer this in the affirmative, and introduce a family of diffusion-based generative models that obtain state-of-the-art likelihoods on standard image density estimation benchmarks. Unlike other diffusion-based models, our method allows for efficient optimization of the noise schedule jointly with the rest of the model. We show that the variational lower bound (VLB) simplifies to a remarkably short expression in terms of the signal-to-noise ratio of the diffused data, thereby improving our theoretical understanding of this model class. Using this insight, we prove an equivalence between several models proposed in the literature. In addition, we show that the continuous-time VLB is invariant to the noise schedule, except for the signal-to-noise ratio at its endpoints. This enables us to learn a noise schedule that minimizes the variance of the resulting VLB estimator, leading to faster optimization. Combining these advances with architectural improvements, we obtain state-of-the-art likelihoods on image density estimation benchmarks, outperforming autoregressive models that have dominated these benchmarks for many years, with often significantly faster optimization. In addition, we show how to turn the model into a bits-back compression scheme, and demonstrate lossless compression rates close to the theoretical optimum.

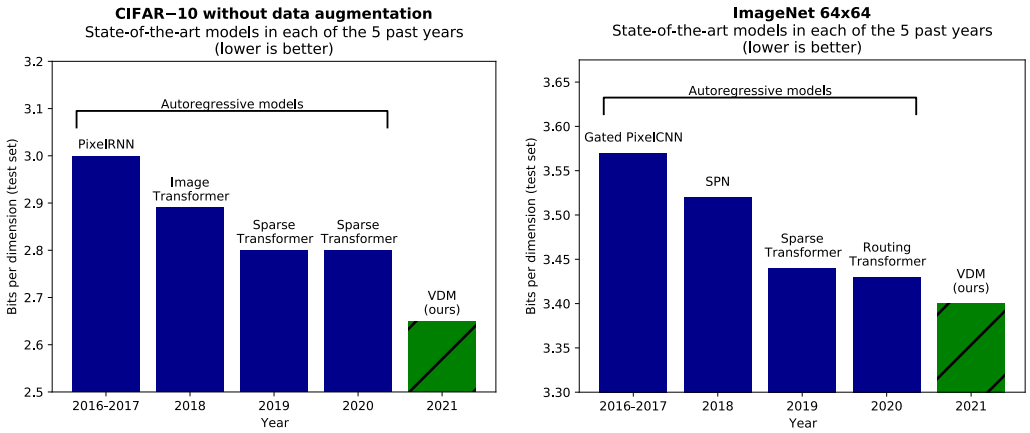
1 Introduction

Likelihood-based generative modeling is a central task in machine learning that is the basis for a wide range of applications ranging from speech synthesis [Oord et al., 2016], to translation [Sutskever et al., 2014], to compression [MacKay, 2003], to many others. Autoregressive models have long been the dominant model class on this task due to their tractable likelihood and expressivity, as shown in Figure 1. Diffusion models have recently shown impressive results in image [Ho et al., 2020, Song et al., 2021b, Nichol and Dhariwal, 2021] and audio generation [Kong et al., 2020, Chen et al., 2020] in terms of perceptual quality, but have yet to match autoregressive models on density estimation benchmarks. In this paper we make several technical contributions that allow diffusion models to challenge the dominance of autoregressive models in this domain. Our main contributions are as follows:

- We introduce a flexible family of diffusion-based generative models that achieve new state-of-the-art log-likelihoods on standard image density estimation benchmarks (CIFAR-10 and ImageNet). This is enabled by incorporating Fourier features into the diffusion model and using a learnable specification of the diffusion process, among other modeling contributions.
- We improve our theoretical understanding of density modeling using diffusion models by analyzing their variational lower bound (VLB), deriving a remarkably simple expression in

* Equal contribution.

Preprint. Under review.



(a) CIFAR-10 without data augmentation

(b) ImageNet 64x64

Figure 1: Autoregressive generative models were long dominant in standard image density estimation benchmarks. In contrast, we propose a family of diffusion-based generative models, *Variational Diffusion Models* (VDMs), that outperforms contemporary autoregressive models in these benchmarks. See Table 1 for more results and comparisons.

terms of the signal-to-noise ratio of the diffusion process. This result delivers new insight into the model class: for the continuous-time (infinite-depth) setting we prove a novel invariance of the generative model and its VLB to the specification of the diffusion process, and we show that various diffusion models from the literature are equivalent up to a trivial time-dependent rescaling of the data.

2 Related work

Our work builds on diffusion probabilistic models (DPMs) [Sohl-Dickstein et al., 2015], or *diffusion models* in short. DPMs can be viewed as a type of variational autoencoder (VAE) [Kingma and Welling, 2013, Rezende et al., 2014], whose structure and loss function allows for efficient training of arbitrarily deep models. Interest in diffusion models has recently reignited due to their impressive image generation results [Ho et al., 2020, Song and Ermon, 2020].

Ho et al. [2020] introduced a number of model innovations to the original DPM, with impressive results on image generation quality benchmarks. They showed that the VLB objective, for a diffusion model with discrete time and diffusion variances shared across input dimensions, is equivalent to multi-scale denoising score matching, up to particular weightings per noise scale. Further improvements were proposed by Nichol and Dhariwal [2021], resulting in better log-likelihood scores.

Song and Ermon [2019] first proposed learning generative models through a multi-scale denoising score matching objective, with improved methods in Song and Ermon [2020]. This was later extended to continuous-time diffusion with novel sampling algorithms based on reversing the diffusion process [Song et al., 2021b].

Concurrent to our work, Song et al. [2021a], Huang et al. [2021], and Vahdat et al. [2021] also derived variational lower bounds to the data likelihood under a continuous-time diffusion model. Where we consider the infinitely deep limit of a standard VAE, Song et al. [2021a] and Vahdat et al. [2021] present different derivations based on stochastic differential equations. Huang et al. [2021] considers both perspectives and discusses the similarities between the two approaches. An advantage of our analysis compared to these other works is that we present an intuitive expression of the VLB in terms of the signal-to-noise ratio of the diffused data, which then leads to new results on the invariance of the generative model and its VLB to the specification of the diffusion process. We empirically compare to these works, as well as others, in Table 1.

Previous approaches to diffusion probabilistic models fixed the diffusion process, while we consider flexible learned diffusion processes. This is enabled by directly parameterizing the mean and variance

of the marginal $q(\mathbf{z}_t|\mathbf{z}_0)$, where previous approaches instead parameterized the individual diffusion steps $q(\mathbf{z}_{t+\epsilon}|\mathbf{z}_t)$. In addition, our denoising models include several architecture changes, the most important of which is the use of Fourier features, which enable us to reach much better likelihoods than previous diffusion probabilistic models.

3 Model

We will focus on the most basic case of generative modeling, where we have a dataset of observations of $\mathbf{x}$, and the task is to estimate the marginal distribution $p(\mathbf{x})$. As with most generative models, the described methods can be extended to the case of multiple observed variables, and/or the task of estimating conditional densities $p(\mathbf{x}|\mathbf{y})$. The proposed latent-variable model consists of a diffusion process (Section 3.1) that we invert to obtain a hierarchical generative model (Section 3.2). We optimize the model parameters by maximizing the variational lower bound of the marginal log-likelihood (Section 4). In contrast with earlier DPMs, we optimize the forward time diffusion process jointly with the rest of the model. This turns the model into a type of VAE [Kingma and Welling, 2013, Rezende et al., 2014].

3.1 Forward time diffusion process

The starting point for our generative model is a diffusion process that begins with the data $\mathbf{x}$, and then samples a sequence of latent variables $\mathbf{z}_t$ given $\mathbf{x}$, where t runs forward in time from $t = 0$ to $t = 1$. The distribution of latent variable $\mathbf{z}_t$ conditioned on $\mathbf{x}$, for any $t \in [0, 1]$ is given by:

$$q(\mathbf{z}_t|\mathbf{x}) = \mathcal{N}\left(\alpha_t \mathbf{x}, \sigma_t^2 \mathbf{I}\right), \tag{1}$$

where α_t and σ_t^2 are scalar-valued functions that define the mean and variance of the marginal distributions with domain $t \in [0, 1]$ and range $\mathbb{R}^+$. Both α_t and σ_t^2 are smooth, such that their derivatives with respect to time t are finite. Furthermore, their ratio α_t^2/σ_t^2 is strictly monotonically decreasing in t, such that $\alpha_t^2/\sigma_t^2 < \alpha_s^2/\sigma_s^2$ for any $t > s$.

The joint distribution of latent variables $(\mathbf{z}_s, \mathbf{z}_t, \mathbf{z}_u)$, at subsequent timesteps $0 \leq s < t < u \leq 1$ are distributed as a first-order Markov chain, such that $q(\mathbf{z}_u|\mathbf{z}_t, \mathbf{z}_s) = q(\mathbf{z}_u|\mathbf{z}_t)$. The distribution of $\mathbf{z}_t$ given $\mathbf{z}_s$, for any $0 \leq s < t \leq 1$, is then:

$$q(\mathbf{z}_t|\mathbf{z}_s) = \mathcal{N}\left(\alpha_{t|s}\mathbf{z}_s, \sigma_{t|s}^2 \mathbf{I}\right), \quad \text{where} \quad \alpha_{t|s} = \alpha_t/\alpha_s, \quad \text{and} \quad \sigma_{t|s}^2 = \sigma_t^2 - \alpha_{t|s}^2 \sigma_s^2. \tag{2}$$

Given the distributions above, it is straightforward to verify that the reverse time inference distribution of $\mathbf{z}_s$ given $\mathbf{z}_t$ and $\mathbf{x}$, for any $0 \leq s < t \leq 1$, is also Gaussian and given by:

$$q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_Q(\mathbf{z}_t, \mathbf{x}; s, t), \sigma_{Q,s,t}^2 \mathbf{I}) \quad \text{with} \quad \sigma_{Q,s,t}^2 = \sigma_{t|s}^2 \sigma_s^2/\sigma_t^2, \tag{3}$$

$$\text{and} \quad \boldsymbol{\mu}_Q(\mathbf{z}_t, \mathbf{x}; s, t) = \frac{1}{\alpha_{t|s}}(\mathbf{z}_t + \sigma_{t|s}^2 \nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{x})) = \frac{\alpha_{t|s}\sigma_s^2}{\sigma_t^2}\mathbf{z}_t + \frac{\alpha_s \sigma_{t|s}^2}{\sigma_t^2}\mathbf{x}. \tag{4}$$

In Appendix E we provide an implementation of $\sigma_{Q,s,t}^2$ that is numerically stable for small $t - s$.

As we show in Section 4, the continuous-time VLB objective that we will propose optimizing is surprisingly invariant to the choice of functions α_t and σ_t, which we refer to as the *noise schedule*. Their only impact on our objective is through their ratio at times $t = 0$ and $t = 1$:

$$\text{SNR}(t) = \alpha_t^2/\sigma_t^2, \tag{5}$$

which we call the *signal-to-noise ratio*.

The specific diffusion processes used by Song and Ermon [2019] and Sohl-Dickstein et al. [2015] can be seen as special cases of the proposed model. Song and Ermon [2019] use $\alpha_t = 1$, called *variance-exploding* diffusion processes by Song et al. [2021b]. In our experiments, we choose to use *variance-preserving* diffusion processes as in [Sohl-Dickstein et al., 2015, Ho et al., 2020] where $\alpha_t = \sqrt{1 - \sigma_t^2}$. Written as a function of $\text{SNR}(t)$, this is:

$$\alpha_t^2 = \text{SNR}(t)/(1 + \text{SNR}(t)) \quad \text{and} \quad \sigma_t^2 = 1/(1 + \text{SNR}(t)). \tag{6}$$

In previous works the signal-to-noise ratio $\text{SNR}(t)$ was a fixed function of time, but here we learn this function jointly with the rest of the model, as we explain in Section 5.

3.2 Reverse time generative model

We define our generative model by inverting the diffusion process of Section 3.1, yielding a hierarchical generative model that samples a sequence of latents $\mathbf{z}_t$, with time running backward from $t = 1$ to $t = 0$. We consider both the case where this sequence consists of a finite number of steps T, as well as a continuous time model corresponding to $T \to \infty$. We start by presenting the discrete-time case.

Given finite T, we discretize time uniformly into T segments of width $\tau = 1/T$. Defining $s(i) = (i-1)/T$ and $t(i) = i/T$, our hierarchical generative model for data $\mathbf{x}$ is then given by:

$$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{z}_1) p(\mathbf{x}|\mathbf{z}_0) \prod_{i=1}^{T} p(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)}). \tag{7}$$

With the variance preserving diffusion specification and sufficiently small $\mathrm{SNR}(1)$, we have that $q(\mathbf{z}_1|\mathbf{x}) \approx \mathcal{N}(\mathbf{z}_1; 0, \mathbf{I})$. We therefore model the marginal distribution of $\mathbf{z}_1$ as a spherical Gaussian:

$$p(\mathbf{z}_1) = \mathcal{N}(\mathbf{z}_1; 0, \mathbf{I}) \tag{8}$$

For the reconstruction term, we wish to choose a model $p(\mathbf{x}|\mathbf{z}_0)$ that is close to the unknown $q(\mathbf{x}|\mathbf{z}_0)$. Let x_i and $z_{0,i}$ be the i-th elements of $\mathbf{x}, \mathbf{z}_0$, respectively. We then use a factorized distribution of the form:

$$p(\mathbf{x}|\mathbf{z}_0) = \prod_i p(x_i|z_{0,i}) \tag{9}$$

where we choose $p(x_i|z_{0,i}) \propto q(z_{0,i}|x_i)$. With sufficiently large $\mathrm{SNR}(0)$, this becomes a very close approximation to the true $q(\mathbf{x}|\mathbf{z}_0)$, as the influence of the unknown data distribution $q(\mathbf{x})$ is overwhelmed by the likelihood $q(\mathbf{z}_0|\mathbf{x})$.

Finally, we choose the conditional model distributions as

$$p(\mathbf{z}_s|\mathbf{z}_t) = q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x} = \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)), \tag{10}$$

i.e. the same as the reverse time inference model $q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})$, but with the original data $\mathbf{x}$ replaced by the output of a denoising model $\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)$ that predicts $\mathbf{x}$ from its noisy version $\mathbf{z}_t$. We then have $p(\mathbf{z}_s|\mathbf{z}_t) = \mathcal{N}(\mathbf{z}_s; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}_t; s, t), \sigma^2_{Q,s,t}\mathbf{I})$, with

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}_t; s, t) = \frac{\alpha_{t|s}\sigma_s^2}{\sigma_t^2}\mathbf{z}_t + \frac{\alpha_s \sigma^2_{t|s}}{\sigma_t^2}\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t) = \frac{1}{\alpha_{t|s}}\mathbf{z}_t - \frac{\sigma^2_{t|s}}{\alpha_{t|s}\sigma_t}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t) = \frac{1}{\alpha_{t|s}}\mathbf{z}_t + \frac{\sigma^2_{t|s}}{\alpha_{t|s}}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t; t), \tag{11}$$

with variance $\sigma^2_{Q,s,t}$ the same as in Equation 3, and with

$$\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t) = (\mathbf{z}_t - \alpha_t \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t))/\sigma_t \quad \text{and} \quad \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t; t) = (\alpha_t \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t) - \mathbf{z}_t)/\sigma_t^2. \tag{12}$$

Equation 11 shows that we can interpret our model in three different ways: 1) In terms of the *denoising model* $\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)$ that recovers $\mathbf{x}$ from its corrupted version $\mathbf{z}_t$. 2) In terms of a *noise prediction model* $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)$ that directly infers the noise $\boldsymbol{\epsilon}$ that was used to generate $\mathbf{z}_t$. 3) In terms of a *score model* $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)$, that at its optimum equals the scores of the marginal density: $\mathbf{s}^*(\mathbf{z}_t; t) = \nabla_{\mathbf{z}} \log q(\mathbf{z}_t)$; see Appendix G. These are three equally valid views on the same model class, that have been used interchangeably in the literature. We find the denoising interpretation the most intuitive, and will therefore mostly use $\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)$ in this paper, although in practice we parameterize our model via $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)$ following Ho et al. [2020].

When we take the number of steps $T \to \infty$, our model for $p(\mathbf{z}_t)$ can best be described as a continuous time diffusion process [Song et al., 2021b], governed by the stochastic differential equation

$$d\mathbf{z}_t = [f(t)\mathbf{z}_t - g^2(t)s_{\boldsymbol{\theta}}(\mathbf{z}_t; t)]dt + g(t)d\mathbf{W}_t, \tag{13}$$

with time running backwards from $t = 1$ to $t = 0$ and

$$f(t) = \frac{d \log \alpha_t}{dt}, \qquad g^2(t) = \frac{d\sigma^2(t)}{dt} - 2\frac{d \log \alpha_t}{dt}\sigma^2(t). \tag{14}$$

As we argue in Section 4.2, we reach the best likelihoods with $T \to \infty$. For most practical purposes however, there is no difference between the continuous time formulation of our model and the discrete-time formulation with large T. For simplicity we therefore use the discrete-time model to derive most of our results in the remaining discussion. For a more detailed discussion of generative modeling via stochastic differential equations see Song et al. [2021b].

4 The variational lower bound

Similar to the original DPMs [Sohl-Dickstein et al., 2015], we optimize the parameters towards the variational lower bound of the marginal likelihood, also called the variational lower bound (VLB). Unlike earlier DPMs, but similar to VAEs [Kingma and Welling, 2013, Rezende et al., 2014], we optimize the inference model parameters (that define the forward time diffusion process) jointly with the rest of the model.

The negative marginal log-likelihood is bounded by:

$$-\log p(\mathbf{x}) \leq -\text{VLB}(\mathbf{x}) = \underbrace{D_{KL}(q(\mathbf{z}_1|\mathbf{x})||p(\mathbf{z}_1))}_{\text{Prior loss}} + \underbrace{\mathbb{E}_{q(\mathbf{z}_0|\mathbf{x})}\left[-\log p(\mathbf{x}|\mathbf{z}_0)\right]}_{\text{Reconstruction loss}} + \underbrace{\mathcal{L}_T(\mathbf{x})}_{\text{Diffusion loss}} \tag{15}$$

The prior loss is a KL divergence between two Gaussians that can be computed in closed form; see Appendix B. The reconstruction loss can be evaluated and optimized using standard reparameterization gradients [Kingma and Welling, 2013]. The diffusion loss, $\mathcal{L}_T(\mathbf{x})$, is more complicated, and depends on the hyperparameter T that determines the depth of the generative model.

4.1 Discrete-time loss

In the case of finite T, using $s(i) = (i-1)/T$, $t(i) = i/T$, the diffusion loss is:

$$\mathcal{L}_T(\mathbf{x}) = \sum_{i=1}^{T} \mathbb{E}_{q(\mathbf{z}_{t(i)}|\mathbf{x})} D_{KL}[q(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)}, \mathbf{x})||p(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)})]. \tag{16}$$

In appendix B we show that this expression simplifies considerably, yielding:

$$\mathcal{L}_T(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I}),i\sim U\{1,T\}}\left[\frac{T}{2}\left(\text{SNR}(s) - \text{SNR}(t)\right)||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)||_2^2\right] \tag{17}$$

where $U\{1, T\}$ is the uniform distribution on the integers $\{1, \ldots, T\}$, and $\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \boldsymbol{\epsilon}$.

4.2 More steps is better

A natural question to ask is what the number of time segments T should be, and whether more segments is always better. In Appendix C we analyze the difference between the diffusion loss with T segments, $\mathcal{L}_T(\mathbf{x})$, and the diffusion loss with double that number of segments, $\mathcal{L}_{2T}(\mathbf{x})$, and find that

$$\mathcal{L}_{2T}(\mathbf{x}) - \mathcal{L}_T(\mathbf{x}) = \mathbb{E}_{t,\boldsymbol{\epsilon}}\left[c(t')\left(||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_{t'}; t')||_2^2 - ||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)||_2^2\right)\right], \tag{18}$$

with $t' = t - 0.5T^{-1}$ and $c(t') = \text{SNR}(t' - 0.5T^{-1}) - \text{SNR}(t')$. Since $t' < t$, latent $\mathbf{z}_{t'}$ is a less noisy version of the data $\mathbf{x}$ from earlier in the diffusion process compared to $\mathbf{z}_t$, which means that predicting the uncorrupted data from $\mathbf{z}_{t'}$ is easier than from $\mathbf{z}_t$. If our trained model $\hat{\mathbf{x}}_{\boldsymbol{\theta}}$ is sufficiently good, we should thus always have that $\mathcal{L}_{2T}(\mathbf{x}) - \mathcal{L}_T(\mathbf{x}) < 0$, i.e. that our VLB will always be better for a larger number of time segments.

4.3 Continuous-time loss

Since taking more time steps leads to a better VLB, we take $T \to \infty$ in the remainder of this paper, effectively treating time t as continuous rather than discrete. In Appendix B we show that in this limit the diffusion loss $\mathcal{L}_T(\mathbf{x})$ simplifies further. Letting $\text{SNR}'(t) = \frac{d\text{SNR}(t)}{dt} = \frac{d\alpha_t^2/\sigma_t^2}{dt}$, we have:

$$\mathcal{L}_\infty(\mathbf{x}) = -\frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\int_0^1 \text{SNR}'(t)\,\|\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)\|_2^2\,dt, \tag{19}$$

$$= -\frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I}),t\sim\mathcal{U}(0,1)}\left[\text{SNR}'(t)\,\|\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)\|_2^2\right]. \tag{20}$$

In terms of predicting the noise $\boldsymbol{\epsilon}$, this can equivalently be written as

$$\mathcal{L}_\infty(\mathbf{x}) = -\frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\int_0^1 \text{log-SNR}'(t)\,\|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)\|_2^2\,dt, \tag{21}$$

$$= -\frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I}),t\sim\mathcal{U}(0,1)}\left[\text{log-SNR}'(t)\,\|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)\|_2^2\right], \tag{22}$$

where $\text{log-SNR}'(t) = \frac{d\log[\text{SNR}(t)]}{dt} = \frac{\text{SNR}'(t)}{\text{SNR}(t)}$.

4.3.1 Invariance to the noise schedule in continuous time

Note that the signal-to-noise function $\text{SNR}(t) = \alpha_t^2/\sigma_t^2$ is invertible due to the monotonicity assumption in Section 3.1. Due to this invertibility, we can perform a change of variables, and make everything a function of $v \equiv \text{SNR}(t)$ instead of t, such that $t = \text{SNR}^{-1}(v)$. Let α_v and σ_v be the functions α_t and σ_t evaluated at $t = \text{SNR}^{-1}(v)$, and correspondingly let $\mathbf{z}_v = \alpha_v \mathbf{x} + \sigma_v \boldsymbol{\epsilon}$. Similarly, we rewrite our noise prediction model as $\tilde{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}, v) \equiv \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}, \text{SNR}^{-1}(v))$. With this change of variables, our continuous-time loss in Equation 19 can equivalently be written as:

$$\mathcal{L}_\infty(\mathbf{x}) = \frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})} \int_{\text{SNR}_{\text{min}}}^{\text{SNR}_{\text{max}}} \|\mathbf{x} - \tilde{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_v, v)\|_2^2 \, dv, \tag{23}$$

where instead of integrating w.r.t. time t we now integrate w.r.t. the signal-to-noise ratio v, and where $\text{SNR}_{\text{min}} = \text{SNR}(1)$ and $\text{SNR}_{\text{max}} = \text{SNR}(0)$.

What this equation shows us is that the only effect the functions $\alpha(t)$ and $\sigma(t)$ have on the diffusion loss is through the values $\text{SNR}(t) = \alpha_t^2/\sigma_t^2$ at endpoints $t = 0$ and $t = 1$. Given these values SNR_{max} and SNR_{min}, the diffusion loss is invariant to the shape of function $\text{SNR}(t)$ between $t = 0$ and $t = 1$.

4.3.2 Equivalence of diffusion models in continuous time

In the last section, we showed that the VLB is only impacted by the function $\text{SNR}(t)$ through its endpoints $\text{SNR}_{\text{min}}, \text{SNR}_{\text{max}}$. We will now show that, apart from these endpoints, the choice of α_t and σ_t actually does not matter at all in continuous time. Since $v = \alpha_v^2/\sigma_v^2$, we have that $\sigma_v = \alpha_v/\sqrt{v}$, which means that $\mathbf{z}_v = \alpha_v \mathbf{x} + \sigma_v \boldsymbol{\epsilon} = \alpha_v(\mathbf{x} + \boldsymbol{\epsilon}/\sqrt{v})$. We can therefore adapt a model $\tilde{\mathbf{x}}_{\boldsymbol{\theta}}^A(\mathbf{z}, v)$ trained with diffusion process $\mathbf{z}_v^A = \alpha_v^A \mathbf{x} +, \sigma_v^A \boldsymbol{\epsilon}$, to a model $\tilde{\mathbf{x}}_{\boldsymbol{\theta}}^B$ corresponding to a different diffusion process $\mathbf{z}_v^B = \alpha_v^B \mathbf{x} +, \sigma_v^B \boldsymbol{\epsilon}$ simply by a time-dependent rescaling its input $\mathbf{z}$: because $(\alpha_v^A/\alpha_v^B)\mathbf{z}_v^B = \mathbf{z}_v^A$, we can simply define $\tilde{\mathbf{x}}_{\boldsymbol{\theta}}^B(\mathbf{z}_v^B, v) \equiv \tilde{\mathbf{x}}_{\boldsymbol{\theta}}^A((\alpha_v^A/\alpha_v^B)\mathbf{z}_v^B, v)$. For $v' < v$, we now have that

$$\begin{aligned} &p_\theta^B(\mathbf{z}_{v'}^B = \mathbf{z} | \mathbf{z}_v^B) = q^B(\mathbf{z}_{v'}^B = \mathbf{z} | \mathbf{z}_v^B, \mathbf{x} = \tilde{\mathbf{x}}_{\boldsymbol{\theta}}^B(\mathbf{z}_v^B, v)) \\ &= q^A((\alpha_v^B/\alpha_v^A)\mathbf{z}_{v'}^A = \mathbf{z} | \mathbf{z}_v^A, \mathbf{x} = \tilde{\mathbf{x}}_{\boldsymbol{\theta}}^A(\mathbf{z}_v^A, v)) = p_\theta^A((\alpha_{v'}^B/\alpha_{v'}^A)\mathbf{z}_{v'}^A = \mathbf{z} | \mathbf{z}_v^A). \end{aligned} \tag{24}$$

In other words, diffusion processes A, B with their corresponding denoising models define the exact same generative model. Assuming these diffusion processes also start and stop at the same signal-to-noise ratios SNR_{max} and SNR_{min}, the previous section tells us that $\mathcal{L}_\infty^A(\tilde{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}^A, \mathbf{x}) = \mathcal{L}_\infty^B(\tilde{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}^B, \mathbf{x})$, i.e. they also both produce the same diffusion loss in continuous time. Any two diffusion models A and B, under the mild constraints set in 3.1 (which includes the variance exploding and variance preserving specifications), can thus be seen as equivalent in continuous time, up to a time-dependent re-scaling of $\mathbf{z}$.

This equivalence between diffusion specifications continues to hold even if, instead of the VLB, these models optimize a *weighted* diffusion loss of the form:

$$\mathcal{L}_\infty(\mathbf{x}, w) = \frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})} \int_{\text{SNR}_{\text{min}}}^{\text{SNR}_{\text{max}}} w(v) \, \|\mathbf{x} - \tilde{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_v, v)\|_2^2 \, dv, \tag{25}$$

which e.g. captures all the different objectives discussed by Song et al. [2021b], see Appendix F. Here, $w(v)$ is a weighting function that generally puts increased emphasis on the noisier data compared to the VLB, and which thereby can sometimes improve perceptual generation quality as measured by certain metrics like FID and Inception Score.

4.3.3 Practical stochastic approximation of the continuous-time loss

Since calculating the integral $\mathcal{L}_\infty(\mathbf{x})$; or its generalization $\mathcal{L}_\infty(\mathbf{x}, w)$, is not analytically tractable, we use its unbiased Monte Carlo estimator in practice. To do this, we construct the VLB in terms of predicting $\boldsymbol{\epsilon}$, which is equivalent to predicting $\mathbf{x}$, but easier to implement in a numerically stable way:

$$\mathcal{L}_\infty^{MC}(\mathbf{x}, w, \gamma) = \frac{1}{2}\gamma'(t) w(\gamma(t)) \, \|\boldsymbol{\epsilon} - \tilde{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{z}_t; \gamma(t))\|_2^2 \,, \tag{26}$$

with $\gamma(t) = -\log \text{SNR}(t) = \log[\sigma_t^2/\alpha_t^2]$, and $\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \boldsymbol{\epsilon}$, with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), t \sim U[0, 1]$. For the models presented in this paper, we further use $w(v) = 1$ as corresponding to the (unweighted)

VLB. The resulting VLB estimate can then be optimized using stochastic gradient descent as usual. The noise schedule $\gamma(t)$ influences the variance of our Monte Carlo estimator, which is why we jointly optimize it with the rest of the model as described in Section 5.

Here, we also found it helpful to sample time t using a low-discrepancy sampler. When processing a minibatch of k examples $\mathbf{x}^i$, $i \in \{1, \dots, k\}$, we require k timesteps t^i sampled from a uniform distribution. Instead of sampling these timesteps independently, we sample a single uniform random number $u_0 \sim U[0, 1]$ and then set $t^i = \text{mod}(u_0 + i/k, 1)$. Each t^i now as the correct uniform marginal distribution, but the minibatch of timesteps covers the space in $[0, 1]$ more equally than when sampling independently, which we find to reduce the variance in our VLB estimate.

5 Learning the noise schedule

So far, we have not discussed how to select the signal-to-noise ratio function, or *noise schedule* $\text{SNR}(t) = \alpha_t^2/\sigma_t^2$ that governs our generative model. In previous work, $\text{SNR}(t)$ has a fixed form (see Appendix D, Fig. 4a). Here, we propose learning this schedule jointly with our denoising model $\hat{\mathbf{x}}_{\boldsymbol{\theta}}$. We parameterize $\text{SNR}(t) = \exp(-\gamma_{\boldsymbol{\eta}}(t))$ using a monotonically increasing neural network $\gamma_{\boldsymbol{\eta}}(t)$, details of which are given in Appendix D.

In the discrete-time case, we then learn parameters $\boldsymbol{\eta}$ by maximizing the VLB, together with our other model parameters. We find this to be especially beneficial when T is small, as we show in Section 7.

The continuous-time case is different: As we showed in Section 4.3.1, the continuous-time diffusion loss is invariant to $\text{SNR}(t)$, except for its endpoints $\text{SNR}_{\text{min}} = \alpha_1^2/\sigma_1^2$ and $\text{SNR}_{\text{max}} = \alpha_0^2/\sigma_0^2$. For this case, we therefore only optimize the VLB with respect to SNR_{min}, SNR_{max}, and not the parameters $\boldsymbol{\eta}$ of the schedule interpolating between them.

Although this interpolating function $\gamma_{\boldsymbol{\eta}}$ does not impact the value of the continuous-time diffusion loss, it does impact the variance of our stochastic estimate of it, given in Equation 26. We therefore propose to learn $\boldsymbol{\eta}$ by minimizing the variance, which we do by performing stochastic gradient descent on our squared diffusion loss $\mathcal{L}_\infty^{MC}(\mathbf{x}, w, \gamma)^2$. We have that $\mathbb{E}[\mathcal{L}_\infty^{MC}(\mathbf{x}, w, \gamma)^2] = \mathcal{L}_\infty(\mathbf{x}, w)^2 + \text{Var}[\mathcal{L}_\infty^{MC}(\mathbf{x}, w, \gamma)]$, where the first part is independent of $\gamma_{\boldsymbol{\eta}}(t)$, and hence that

$$\mathbb{E}[\nabla_{\boldsymbol{\eta}} \mathcal{L}_\infty^{MC}(\mathbf{x}, w, \gamma_{\boldsymbol{\eta}})^2] = \nabla_{\boldsymbol{\eta}} \text{Var}[\mathcal{L}_\infty^{MC}(\mathbf{x}, w, \gamma_{\boldsymbol{\eta}})]. \tag{27}$$

We can calculate this gradient with negligible computational overhead as a by-product of calculating the gradient of the VLB, details of which are given in Appendix D.

This strategy of minimizing the variance of our diffusion loss estimate remains valid for weighted diffusion losses, $w(v) \neq 1$, not corresponding to the VLB, and we therefore expect it to be useful beyond the goal of optimizing for likelihood that we consider in this paper.

6 Fourier features for improved fine scale prediction

Prior work on diffusion models has mainly focused on the perceptual quality of generated samples, which emphasizes coarse scale patterns and global consistency of generated images. Here, we optimize for likelihood, which is sensitive to fine scale details and exact values of individual pixels. Since our reconstruction model $p(\mathbf{x}|\mathbf{z}_0)$ given in Equation 9 is weak, the burden of modeling these fine scale details falls on our denoising diffusion model $\hat{\mathbf{x}}_{\boldsymbol{\theta}}$. In initial experiments, we found that the denoising model had a hard time accurately modeling these details. At larger noise levels, the latents $\mathbf{z}_t$ follow a smooth distribution due to the added Gaussian noise, but at the smallest noise levels the discrete nature of 8-bit image data leads to sharply peaked marginal distributions $q(\mathbf{z}_t)$.

To capture the fine scale details of the data, we propose adding a set of *Fourier features* to the input of our denoising model $\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)$. Such Fourier features consist of a linear projection of the original data onto a set of periodic basis functions with high frequency, which allows the network to more easily model high frequency details of the data. Previous work [Tancik et al., 2020] has used these features for input coordinates to model high frequency details across the *spatial* dimension, and for time embeddings to condition denoising networks over the *temporal* dimension[Song et al., 2021b]. Here we apply it to color channels for single pixels, in order to model fine distributional details at the level of each scalar input.

Concretely, let $z_{i,j,k}$ be the scalar value in the k-th channel in the (i, j) spatial position of network input $\mathbf{z}_t$. We then add additional channels to the input of the denoising model of the form

$$f^n_{i,j,k} = \sin(z_{i,j,k}2^n\pi), \quad \text{and} \quad g^n_{i,j,k} = \cos(z_{i,j,k}2^n\pi), \tag{28}$$

where we used $n \in \{7, 8\}$. These additional channels are then concatenated to $\mathbf{z}_t$ before being used as input in a standard convolutional denoising model similar to that used by Ho et al. [2020]. We find that the presence of these high frequency features allows our network to learn with much higher values of SNR_{max}, or conversely lower noise levels σ_0^2, than is otherwise optimal. This leads to large improvements in likelihood as demonstrated in Section 7 and Figure 3. We did not observe such improvements when incorporating Fourier features into autoregressive models.

7 Experiments

We demonstrate our proposed class of diffusion models, which we call *Variational Diffusion Models* (VDMs), on the CIFAR-10 [Krizhevsky et al., 2009] dataset, and the downsampled ImageNet [Van Oord et al., 2016, Deng et al., 2009] dataset, where we focus on maximizing likelihood. The score models we use closely follow Ho et al. [2020], except that they process the data solely at the original resolution, without any internal downsampling or upsampling. Our score models are also deeper than those used by others in the literature. All reported models incorporate Fourier features (Section 6) as well as a learnable diffusion specification (Section 5). For our result with data augmentation we used random flips, 90-degree rotations, and color channel swapping. Complete details on our model specifications can be found in Appendix A.

Model (Bits per dim on test set)	Type	CIFAR10 no data aug.	CIFAR10 data aug.	ImageNet 32x32	ImageNet 64x64
ResNet VAE with IAF [Kingma et al., 2016]	VAE	3.11			
Very Deep VAE [Child, 2020]	VAE	2.87		3.80	3.52
NVAE [Vahdat and Kautz, 2020]	VAE	2.91		3.92	
CR-NVAE [Sinha and Dieng, 2021]	VAE		$2.51^{(A)}$		
Glow [Kingma and Dhariwal, 2018]	Flow		$3.35^{(B)}$	4.09	3.81
Flow++ [Ho et al., 2019a]	Flow	3.08		3.86	3.69
PixelCNN [Van Oord et al., 2016]	AR	3.03		3.83	3.57
PixelCNN++ [Salimans et al., 2017]	AR	2.92			
Image Transformer [Parmar et al., 2018]	AR	2.90		3.77	
SPN [Menick and Kalchbrenner, 2018]	AR				3.52
Sparse Transformer [Child et al., 2019]	AR	2.80			3.44
Routing Transformer [Roy et al., 2021]	AR				3.43
Sparse Transformer + DistAug [Jun et al., 2020]	AR		$2.53^{(A)}$		
DDPM [Ho et al., 2020]	Diff		$3.69^{(C)}$		
Score SDE [Song et al., 2021b]	Diff	2.99			
Improved DDPM [Nichol and Dhariwal, 2021]	Diff	2.94			3.54
LSGM [Vahdat et al., 2021]	Diff	2.87			
ScoreFlow [Song et al., 2021a] (variational bound)	Diff	2.87		3.84	
ScoreFlow [Song et al., 2021a] (cont. norm. flow)	Diff	2.74		3.76	
VDM (ours) (variational bound)	Diff	**2.65**	**2.49**$^{(A)}$	**3.72**	**3.40**

Table 1: Summary of our findings for density modeling tasks, in terms of bits per dimension (BPD) on the test set. Model types are autoregressive (AR), normalizing flows (Flow), variational autoencoders (VAE), or diffusion models (Diff). Our results were obtained using the continuous-time formulation of our model. CIFAR-10 data augmentations are: (A) extensive, (B) small translations, or (C) horizontal flips. The numbers for VDM are variational bounds, and can likely be improved by estimating the marginal likelihood through importance sampling, or through evaluation of the corresponding continuous normalizing flow as done by Song et al. [2021a].

7.1 Likelihood and samples

Table 1 shows our results on modeling the CIFAR-10 dataset, and the downsampled ImageNet dataset. We establish a new state-of-the-art in terms of test set likelihood on all the benchmarks without data

augmentation, by a significant margin. Our model for CIFAR-10 without data augmentation surpasses the previous best result of 2.80 about 10x faster than it takes the Sparse Transformer to reach this, in wall clock time on equivalent hardware.

On CIFAR-10 with data augmentation we tie the concurrent work by Sinha and Dieng [2021], which obtains impressive results by applying data augmentation and a consistency regularizer to VAEs. The data augmentation we considered is relatively simple compared to their work as we only use permutations of the data (flips, 90 degree rotations, channel shuffling) and not augmentations that change the data itself (zoom, non-integer shift, more general rotations). Training our model with the augmentation procedure used by Sinha and Dieng [2021] is an interesting direction for future work.

Our CIFAR-10 model, whose hyper-parameters were tuned for likelihood, results in a FID (perceptual quality) score of 7.41. This would have been state-of-the-art until recently, but is worse than recent diffusion models that specifically target FID scores [Nichol and Dhariwal, 2021, Song et al., 2021b, Ho et al., 2020]. By instead using a weighted diffusion loss, with the weighting function $w(\text{SNR})$ used by Ho et al. [2020] and described in Appendix F, our FID score improves to 4.0. We did not pursue further tuning of the model to improve FID instead of likelihood.

A random sample of generated images from our model is provided in Figure 2. We provide additional samples from this model, as well as our other models for the other datasets, in Appendix H.

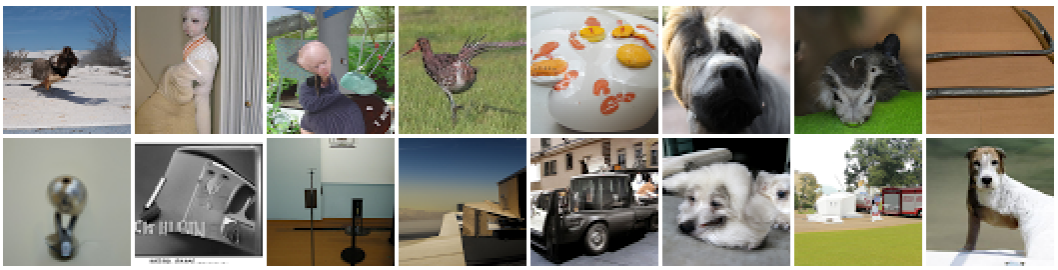


Figure 2: Non cherry-picked unconditional samples from our Imagenet 64x64 model, trained in continuous time and generated using $T = 1000$. The model's hyper-parameters and parameters are optimized w.r.t. the likelihood bound, so the model is not optimized for synthesis quality.

7.2 Ablations

Next, we investigate the relative importance of our contributions. In Table 2 we compare our discrete-time and continuous-time specifications of the diffusion model: When evaluating our model with a small number of steps, our discretely trained models perform better by learning the diffusion schedule to optimize the VLB. However, as argued theoretically in Section 4.2, we find experimentally that more steps T indeed gives better likelihood. When T grows large, our continuously trained model performs best, helped by training its diffusion schedule to minimize variance instead.

Minimizing the variance also helps the continuous time model to train faster, as shown in Figure 3. This effect is further examined in Table 4b, where we find dramatic variance reductions compared to our baselines in continuous time. Figure 4a shows how this effect is achieved: Compared to the other schedules, our learned schedule spends much more time in the high $\text{SNR}(t)$ / low σ_t^2 range.

In Figure 3 we further show training curves for our model including and excluding the Fourier features proposed in Section 6: With Fourier features enabled our model achieves much better likelihood. For comparison we also implemented Fourier features in a PixelCNN++ model [Salimans et al., 2017], where we do not see a benefit.

7.3 Lossless compression

For a fixed number of evaluation timesteps T_{eval}, our diffusion model in discrete time is a hierarchical latent variable model that can be turned into a lossless compression algorithm using bits-back coding [Hinton and Van Camp, 1993]. Bits-back coding encodes a latent and data together, with the latent sampled from the approximate posterior using auxiliary random bits. The net coding cost

T_{train}	T_{eval}	BPD	Bits-Back Net BPD
10	10	4.31	
100	100	2.84	
250	250	2.73	
500	500	2.68	
1000	1000	2.67	
10000	10000	2.66	
∞	10	7.54	7.54
∞	100	2.90	2.91
∞	250	2.74	2.76
∞	500	2.69	2.72
∞	1000	2.67	2.72
∞	10000	**2.65**	
∞	∞	**2.65**	

Table 2: Discrete versus continuous-time training and evaluation with CIFAR-10, in terms of bits per dimension (BPD).

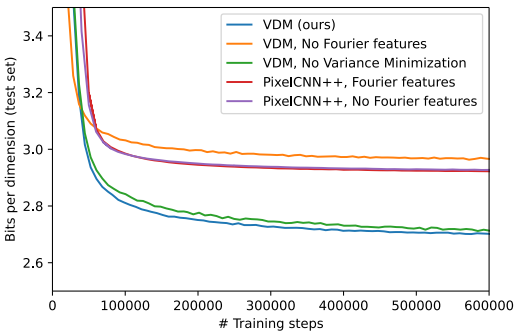
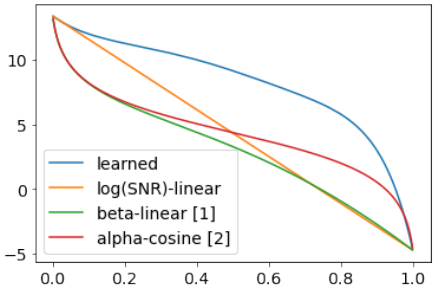


Figure 3: Test set likelihoods during training, with/without Fourier features, and with/without learning the noise schedule to minimize variance.



(a) log SNR vs time t

SNR(t) schedule	Var(BPD)
Learned (ours)	**0.53**
log SNR-linear	6.35
β-Linear [1]	31.6
α-Cosine [2]	31.1

(b) Variance of VLB estimate

Figure 4: Our learned continuous-time variance-minimizing noise schedule $\text{SNR}(t)$ for CIFAR-10, compared to its log-linear initialization and to schedules from the literature: [1] The β-Linear schedule from Ho et al. [2020], [2] The α-Cosine schedule from Nichol and Dhariwal [2021]. All schedules were scaled and shifted on the log scale such that the resulting SNR_{min}, SNR_{max} were the equal to our learned endpoints, resulting in the same VLB estimate of 2.66. We report the variance of our VLB estimate per data point, computed on the test set, and conditional on the data: This does not include the noise due to sampling minibatches of data.

of bits-back coding is given by subtracting the number of bits needed to sample the latent from the number of bits needed to encode the latent and data using the reverse process, so the negative VLB of our discrete time model is the theoretical expected coding cost for bits-back coding.

As a proof of concept for practical lossless compression using our model, Table 2 reports net codelengths on the CIFAR10 test set for various settings of T_{eval} using BB-ANS [Townsend et al., 2018], a practical implementation of bits-back coding based on asymmetric numeral systems [Duda, 2009]. Details of our implementation are given in Appendix I. We achieve state-of-the-art net codelengths, proving our model can be used as the basis of a practical lossless compression algorithm. However, for large T_{eval} a gap remains with the theoretically optimal codelength corresponding to the negative VLB, and compression becomes computationally expensive due to the large number of neural network forward passes required. Closing this gap with more efficient implementations of bits-back coding suitable for very deep models is an interesting avenue for future work.

8 Conclusion

We presented state-of-the-art results on modeling the density of natural images using a new class of diffusion models that incorporates a learnable diffusion specification, Fourier features for fine-scale modeling, as well as other architectural innovations. In addition, we obtained new theoretical insight into likelihood-based generative modeling with diffusion models, showing a surprising invariance of the VLB to the forward time diffusion process in continuous time, as well as an equivalence between various diffusion processes from the literature previously thought to be different.

9 Acknowledgments

We thank Yang Song, Kevin Murphy and Mohammad Norouzi for feedback on drafts of this paper.

References

Nanxin Chen, Yu Zhang, Heiga Zen, Ron J. Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, 2020.

Rewon Child. Very deep VAEs generalize autoregressive models and can outperform them on images. *arXiv preprint arXiv:2011.10650*, 2020.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Jarek Duda. Asymmetric numeral systems. *arXiv preprint arXiv:0902.0271*, 2009.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 5–13, 1993.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pages 2722–2730. PMLR, 2019a.

Jonathan Ho, Evan Lohn, and Pieter Abbeel. Compression with flows via local bits-back coding. In *Advances in Neural Information Processing Systems*, pages 3874–3883, 2019b.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

Chin-Wei Huang, Jae Hyun Lim, and Aaron Courville. A variational perspective on diffusion-based generative models and score matching. *arXiv preprint arXiv:2106.02808*, 2021.

Heewoo Jun, Rewon Child, Mark Chen, John Schulman, Aditya Ramesh, Alec Radford, and Ilya Sutskever. Distribution augmentation for generative modeling. In *International Conference on Machine Learning*, pages 5006–5019. PMLR, 2020.

Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *International Conference on Learning Representations*, 2013.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

Friso Kingma, Pieter Abbeel, and Jonathan Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. In *International Conference on Machine Learning*, pages 3408–3417. PMLR, 2019.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Jacob Menick and Nal Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. *arXiv preprint arXiv:1812.01608*, 2018.

Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672*, 2021.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018.

Danilo J Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Samarth Sinha and Adji B Dieng. Consistency regularization for variational auto-encoders. *arXiv preprint arXiv:2105.14859*, 2021.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pages 11895–11907, 2019.

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *arXiv e-prints*, pages arXiv–2101, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling Through Stochastic Differential Equations. In *International Conference on Learning Representations*, 2021b.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020.

James Townsend, Thomas Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. In *International Conference on Learning Representations*, 2018.

James Townsend, Thomas Bird, Julius Kunze, and David Barber. HiLLoC: lossless image compression with hierarchical latent variable models. In *International Conference on Learning Representations*, 2020.

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *arXiv preprint arXiv:2106.05931*, 2021.

Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756, 2016.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

A Hyperparameters, architecture, and implementation details

In this section we provide details on the exact setup for each of our experiments. In Sections A.1 we describe the choices in common to each of our experiments. Hyperparameters specific to the individual experiments are given in Section A.2. We are currently working towards open sourcing our code.

A.1 Model and implementation

Our denoising models are parameterized in terms of $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{z}_t; \gamma_t)$, where $\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; \gamma_t) = \frac{1}{\alpha_t}(\mathbf{z}_t - \sigma_t \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{z}_t; \gamma_t))$, and where γ_t is the negative log signal-to-noise ratio, i.e. $\gamma_t = \log[\sigma_t^2/\alpha_t^2]$.

Our models $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}$ closely follow the architecture used by Ho et al. [2020], which is based on a U-Net type neural net [Ronneberger et al., 2015] that maps from the input $\mathbf{z} \in \mathbb{R}^d$ to output of the same dimension. As compared to their publically available code at `https://github.com/hojonathanho/diffusion`, our implementation differs in the following ways:

- Our networks don't perform any internal downsampling or upsampling: we process all the data at the original input resolution.
- Our models are deeper than those used by Ho et al. [2020]. Specific numbers are given in Section A.2.
- Instead of taking time t as input to the denoising model, we use $\gamma_t = \log[\sigma_t^2/\alpha_t^2]$, which we rescale to have approximately the same range as t of $[0, 1]$ before using it to form 'time' embeddings in the same way as Ho et al. [2020].
- Our models calculate Fourier features on the input data $\mathbf{z}_t$ as discussed in Section 6, which are then concatenated to $\mathbf{z}_t$ before being fed to the U-Net.
- Apart from the *middle* attention block that connects the upward and downward branches of the U-Net, we remove all other attention blocks from the model. We found that these attention blocks made it more likely for the model to overfit to the training set.
- All of our models use dropout at a rate of 0.1 in the intermediate layers, as did Ho et al. [2020]. In addition we regularize the model by using decoupled weight decay [Loshchilov and Hutter, 2017] with coefficient 0.01.
- We use the Adam optimizer with a learning rate of $2e^{-4}$ and exponential decay rates of $\beta_1 = 0.9, \beta_2 = 0.99$. We found that higher values for β_2 resulted in training instabilities.
- For evaluation, we use an exponential moving average of our parameters, calculated with an exponential decay rate of 0.9999.

We implemented our evaluation of the VLB following Equation 58, which we find the easiest to implement in a numerically stable way. This is equivalent to the VLB expressions in terms of $\hat{\mathbf{x}}_{\boldsymbol{\theta}}$ that we find more intuitive to reason about and which therefore form the core of our presentation here. We use the variance preserving diffusion process with $\alpha_t^2 = \phi(-\gamma(t)), \sigma_t^2 = \phi(\gamma(t))$, with ϕ the sigmoid function. Contrary to earlier works that used a fixed noise schedule, we learn the $\gamma(t)$ function using the approach described in Sections 5 and D.

We regularly evaluate the variational bound on the likelihood on the validation set and find that our models do not overfit during training, using the current settings. We therefore do not use early stopping and instead allow the network to be optimized for 10 million parameter updates for CIFAR-10, and for 2 million updates for ImageNet, before obtaining the test set numbers reported in this paper. It looks like our models keep improving even after this number of updates, in terms of likelihood, but we did not explore this systematically due to resource constraints.

All of our models are trained on TPUv3 hardware (see `https://cloud.google.com/tpu`) using data parallelism. We also evaluated our trained models using CPU and GPU to check for robustness of our reported numbers to possible rounding errors. We found only very small differences when evaluating on these other hardware platforms.

A.2 Settings for each dataset

Our model for CIFAR-10 with no data augmentation uses a U-Net of depth 32, consisting of 32 ResNet blocks in the forward direction and 32 ResNet blocks in the reverse direction, with a single attention layer and two additional ResNet blocks in the middle. We keep the number of channels constant throughout at 128. This model was trained on 8 TPUv3 chips, with a total batch size of 128 examples. Reaching a test-set BPD of 2.65 after 10 million updates takes 9 days, although our model already surpasses sparse transformers (the previous state-of-the-art) of 2.80 BPD after only 2.5 hours of training.

For CIFAR-10 with data augmentation we used random flips, 90-degree rotations, and color channel swapping, which were previously shown to help for density estimation by Jun et al. [2020]. Each of the three augmentations independently were given a 50% probability of being applied to each example, which means that 1 in 8 training examples was not augmented at all. For this experiment, we doubled the number of channels in our model to 256, and decreased the dropout rate from 10% to 5%. Since overfitting was less of a problem with data augmentation, we add back the attention blocks after each ResNet block, following Ho et al. [2020]. We also experimented with conditioning our model on an additional binary feature that indicates whether or not the example was augmented, which can be seen as a simplified version of the augmentation conditioning proposed by Jun et al. [2020]. Conditioning made almost no difference to our results, which may be explained by the relatively large fraction (12.5%) of clean data fed to our model during training. We trained our model for slightly over a week on 128 TPUv3 chips to obtain the reported result.

Our model for 32x32 ImageNet looks similar to that for CIFAR-10 without data augmentation, with a U-Net depth of 32, but uses double the number of channels at 256. It is trained using data parallelism on 32 TPUv3 chips, with a total batch size of 512.

Our model for 64x64 ImageNet uses double the depth at 64 ResNet layers in both the forward and backward direction in the U-Net. It also uses a constant number of channels of 256. This model is trained on 128 TPUv3 chips at a total batch size of 512 examples. The model passes the

B Derivation of the VLB estimators

B.1 Discrete-time VLB

Similar to [Sohl-Dickstein et al., 2015], we decompose the negative variational lower bound (VLB) as:

$$-\log p(\mathbf{x}) \geq \mathcal{L}(\mathbf{x}) = \mathcal{L}_T(\mathbf{x}) + \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})}\left[\log \frac{q(\mathbf{z}_1|\mathbf{x})}{p(\mathbf{z}_1)}\right] + \mathbb{E}_{q(\mathbf{z}_0|\mathbf{x})}\left[-\log p(\mathbf{x}|\mathbf{z}_0)\right] \tag{29}$$

$$\text{where} \quad \mathcal{L}_T(\mathbf{x}) = \sum_{i=1}^{T} \mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})}[D_{KL}(q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})||p(\mathbf{z}_s|\mathbf{z}_t))] \tag{30}$$

where, $s = (i-1)/T$ and $t = i/T$. The second and third right-hand side terms of Equation 29 can be evaluated and optimized using standard techniques. We will now derive an estimator for $\mathcal{L}_T(\mathbf{x})$, the remaining and more challenging term. We will first derive an expression of $D_{KL}(q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})||p(\mathbf{z}_s|\mathbf{z}_t))$.

Recall that $p(\mathbf{z}_s|\mathbf{z}_t) = q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x} = \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t))$, and thus $q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x}) = \mathcal{N}(\mathbf{z}_s; \boldsymbol{\mu}_Q(\mathbf{z}_t, \mathbf{x}; s, t), \sigma^2_{Q,s,t}\mathbf{I})$ and $p(\mathbf{z}_s|\mathbf{z}_t) = \mathcal{N}(\mathbf{z}_s; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}_t; s, t), \sigma^2_{Q,s,t}\mathbf{I})$, with

$$\boldsymbol{\mu}_Q(\mathbf{z}_t, \mathbf{x}; s, t) = \frac{\alpha_{t|s}\sigma_s^2}{\sigma_t^2}\mathbf{z}_t + \frac{\alpha_s \sigma^2_{t|s}}{\sigma_t^2}\mathbf{x} \tag{31}$$

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}_t; s, t) = \frac{\alpha_{t|s}\sigma_s^2}{\sigma_t^2}\mathbf{z}_t + \frac{\alpha_s \sigma^2_{t|s}}{\sigma_t^2}\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t), \tag{32}$$

$$\text{and} \quad \sigma^2_{Q,s,t} = \sigma^2_{t|s}\sigma_s^2/\sigma_t^2. \tag{33}$$

Since $q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})$ and $p(\mathbf{z}_s|\mathbf{z}_t)$ are Gaussians, their KL divergence is available in closed form as a function of their means and variances, which due to their with equal variances simplifies as:

$$D_{KL}(q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})||p(\mathbf{z}_s|\mathbf{z}_t)) = \frac{1}{2\sigma^2_{Q,s,t}}||\boldsymbol{\mu}_Q - \boldsymbol{\mu}_{\boldsymbol{\theta}}||_2^2 \tag{34}$$

$$= \frac{\sigma_t^2}{2\sigma^2_{t|s}\sigma_s^2}\frac{\alpha_s^2\sigma^4_{t|s}}{\sigma_t^4}||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)||_2^2 \tag{35}$$

$$= \frac{1}{2\sigma_s^2}\frac{\alpha_s^2\sigma^2_{t|s}}{\sigma_t^2}||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)||_2^2 \tag{36}$$

$$= \frac{1}{2\sigma_s^2}\frac{\alpha_s^2(\sigma_t^2 - \alpha^2_{t|s}\sigma_s^2)}{\sigma_t^2}||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)||_2^2 \tag{37}$$

$$= \frac{1}{2}\frac{\alpha_s^2\sigma_t^2/\sigma_s^2 - \alpha_t^2}{\sigma_t^2}||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)||_2^2 \tag{38}$$

$$= \frac{1}{2}\left(\frac{\alpha_s^2}{\sigma_s^2} - \frac{\alpha_t^2}{\sigma_t^2}\right)||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)||_2^2 \tag{39}$$

$$= \frac{1}{2}\left(\text{SNR}(s) - \text{SNR}(t)\right)||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)||_2^2 \tag{40}$$

Reparameterizing $\mathbf{z}_t \sim q(\mathbf{z}_t|\mathbf{x})$ as $\mathbf{z}_t = \alpha_t\mathbf{x} + \sigma_t\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$, our diffusion loss becomes:

$$\mathcal{L}_T(\mathbf{x}) = \sum_{i=1}^{T}\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})}[D_{KL}(q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})||p(\mathbf{z}_s|\mathbf{z}_t))] \tag{41}$$

$$= \frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}[\sum_{i=1}^{T}\left(\text{SNR}(s) - \text{SNR}(t)\right)||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)||_2^2] \tag{42}$$

B.2 Estimator of $\mathcal{L}_T(\mathbf{x})$

To avoid having to compute all T terms when calculating the diffusion loss, we construct an unbiased estimator of $\mathcal{L}_T(\mathbf{x})$ using

$$\mathcal{L}_T(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I}),i\sim U\{1,T\}}\left[\frac{T}{2}\left(\text{SNR}(s) - \text{SNR}(t)\right)||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)||_2^2\right] \tag{43}$$

where $U\{1, T\}$ is the discrete uniform distribution from 1 to (and including) T, $s = (i-1)/T$, $t = i/T$ and $\mathbf{z}_t = \sqrt{1-\boldsymbol{\sigma}_t^2}\mathbf{x} + \boldsymbol{\sigma}_t\boldsymbol{\epsilon}$. This trivially yields an unbiased Monte Carlo estimator, by drawing random samples $i \sim U\{1, T\}$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$.

B.3 Infinite depth ($T \to \infty$)

To calculate the limit of the diffusion loss as $T \to \infty$, we express $\mathcal{L}_T(\mathbf{x})$ as a function of $\tau = 1/T$:

$$\mathcal{L}_T(\mathbf{x}) = \frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I}),i\sim U\{1,T\}}\left[\frac{\text{SNR}(t-\tau) - \text{SNR}(t)}{\tau}||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)||_2^2\right]. \tag{44}$$

As $\tau \to 0, T \to \infty$, this then gives

$$\mathcal{L}_\infty(\mathbf{x}) = -\frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I}),t\sim U[0,1]}\left[\text{SNR}'(t)||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)||_2^2\right] \tag{45}$$

$$= -\frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\int_0^1 \text{SNR}'(t)\left\|\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)\right\|_2^2 dt. \tag{46}$$

C Influence of the number of steps T on the VLB

Recall that the diffusion loss for our choice of model p, q, when using T segments of discrete time, is given by

$$\mathcal{L}_T(\mathbf{x}) = \frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[\sum_{i=1}^{T}\left(\text{SNR}(s(i)) - \text{SNR}(t(i))\right)||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_{t(i)}; t(i))||_2^2\right], \tag{47}$$

with $s(i) = (i-1)/T$, $t(i) = i/T$.

This can then be written equivalently as

$$\mathcal{L}_T(\mathbf{x}) = \frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[\sum_{i=1}^{T}\left(\text{SNR}(s) - \text{SNR}(t') + \text{SNR}(t') - \text{SNR}(t)\right)||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)||_2^2\right], \tag{48}$$

with $t' = t - 0.5/T$.

In contrast, the diffusion loss with 2T timesteps can be written as

$$\begin{aligned}\mathcal{L}_{2T}(\mathbf{x}) = \frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})} &\sum_{i=1}^{T}\left(\text{SNR}(s) - \text{SNR}(t')\right)||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_{t'}; t')||_2^2 \\ &+ \sum_{i=1}^{T}\left(\text{SNR}(t') - \text{SNR}(t)\right)||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)||_2^2.\end{aligned} \tag{49}$$

Subtracting the two results, we get

$$\begin{aligned}&\mathcal{L}_{2T}(\mathbf{x}) - \mathcal{L}_T(\mathbf{x}) = \\ &\frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[\sum_{i=1}^{T}(\text{SNR}(s) - \text{SNR}(t'))\left(||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_{t'}; t')||_2^2 - ||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)||_2^2\right)\right].\end{aligned} \tag{50}$$

Since $t' < t$, $\mathbf{z}_{t'}$ is a less noisy version of the data from earlier in the diffusion process compared to $\mathbf{z}_t$. Predicting the original data $\mathbf{x}$ from $\mathbf{z}_{t'}$ is thus strictly easier than from $\mathbf{z}_t$, leading to lower mean squared error if our model $\hat{\mathbf{x}}_{\boldsymbol{\theta}}$ is good enough. We thus have that $\mathcal{L}_{2T}(\mathbf{x}) - \mathcal{L}_T(\mathbf{x}) < 0$, which means that doubling the number of timesteps always improves our diffusion loss. For this reason we argue for using the continuous-time VLB corresponding to $T \to \infty$ in this paper.

D Implementation of monotonic neural net noise schedule $\gamma_\eta(t)$

To learn the signal-to-noise ratio $\text{SNR}(t)$, we parameterize it as $\text{SNR}(t) = \exp(-\gamma_{\boldsymbol{\eta}}(t))$ with $\gamma_{\boldsymbol{\eta}}(t)$ a monotonic neural network. This network consists of 3 linear layers with weights that are restricted to be positive, l_1, l_2, l_3, which are composed as $\tilde{\gamma}_{\boldsymbol{\eta}}(t) = l_1(t) + l_3(\phi(l_2(l_1(t))))$, with ϕ the sigmoid function. Here, the l_2 layer has 1024 outputs, and the other layers have a single output.

At this point, the range of $\tilde{\gamma}_{\boldsymbol{\eta}}(t)$ is unbounded and so the resulting SNR is not yet restricted to $[\text{SNR}_{\text{min}}, \text{SNR}_{\text{max}}]$. We therefore postprocess the monotonic neural network as

$$\gamma_{\boldsymbol{\eta}}(t) = \gamma_0 + (\gamma_1 - \gamma_0)\frac{\tilde{\gamma}_{\boldsymbol{\eta}}(t) - \tilde{\gamma}_{\boldsymbol{\eta}}(0)}{\tilde{\gamma}_{\boldsymbol{\eta}}(1) - \tilde{\gamma}_{\boldsymbol{\eta}}(0)}, \tag{51}$$

with $\gamma_0 = -\log(\text{SNR}_{\text{max}}), \gamma_1 = -\log(\text{SNR}_{\text{min}})$. Now $\text{SNR}(t) = \exp(-\gamma_{\boldsymbol{\eta}}(t))$ has the correct range and interpolates exactly between SNR_{min} and SNR_{max}. We treat γ_0, γ_1 as free parameters that we optimize jointly with the parameters of the denoising model to maximize the VLB. The parameters $\boldsymbol{\eta}$ are learned by minimizing the variance of the stochastic estimate of the VLB.

D.1 Efficient gradient calculation for η

We wish to calculate $\nabla_{\boldsymbol{\eta}}[\mathcal{L}_\infty^{MC}(\mathbf{x}, \gamma_{\boldsymbol{\eta}})^2]$ without performing a second backpropagation pass through the denoising model due to this objective being different than for the other parameters. To do this, we

decompose the gradient as

$$\frac{d}{d\boldsymbol{\eta}}[\mathcal{L}_\infty^{MC}(\mathbf{x},\gamma_{\boldsymbol{\eta}})^2] = \frac{d}{d\text{SNR}}\left[\mathcal{L}_\infty^{MC}(\mathbf{x},\text{SNR})^2\right]\frac{d}{d\boldsymbol{\eta}}\left[\text{SNR}(\boldsymbol{\eta})\right], \tag{52}$$

$$\text{and}\quad \frac{d}{d\text{SNR}}\left[\mathcal{L}_\infty^{MC}(\mathbf{x},\text{SNR})^2\right] = 2\frac{d}{d\text{SNR}}\left[\mathcal{L}_\infty^{MC}(\mathbf{x},\text{SNR})\right]\odot\mathcal{L}_\infty^{MC}(\mathbf{x},\text{SNR}), \tag{53}$$

where $\odot$ denotes elementwise multiplication. Here $\frac{d}{d\text{SNR}}\left[\mathcal{L}_\infty^{MC}(\mathbf{x},\text{SNR})\right]$ is computed along with the other gradients when performing the single backpropagation pass for calculating $\nabla_{\boldsymbol{\theta}}[\mathcal{L}_\infty^{MC}]$. The remaining operations required to get $\nabla_{\boldsymbol{\eta}}[\mathcal{L}_\infty^{MC}(\mathbf{x},\gamma_{\boldsymbol{\eta}})^2]$ have negligible computational cost.

E Numerical stability

Floating point numbers are much worse at representing numbers close to 1, than at representing numbers close to 0. Since a naïve implementation of our model and its discrete-time loss function requires computing intermediate values that are close to 1, those numbers are erroneously rounded to 1, leading to numerical issues and incorrect results.

To circumvent this problem, previous implementations of discrete-time diffusion models (e.g. [Ho et al., 2020]) used 64-bit floating point numbers for part of the computation, complicating the implementation. Luckily, in our case, this can be completely avoided by using numerically stable implementations.

The numerically problematic terms are $\boldsymbol{\sigma}_{t|s}^2$ which is used for sampling, and $\boldsymbol{\sigma}_{t|s}^2/(1-\boldsymbol{\sigma}_{t|s}^2)$ which appears in the discrete-time loss. Defining $\gamma(t) \equiv -\log[\text{SNR}(t)]$, it is straightforward to verify that:

$$\boldsymbol{\sigma}_{t|s}^2 = -\text{expm1}(\text{softplus}(\gamma(s)) - \text{softplus}(\gamma(t))), \tag{54}$$

where $\text{expm1}(x) \equiv \exp(x) - 1$ and $\text{softplus}(x) \equiv \log(1+\exp(x))$ are functions with numerically stable primitives in common numerical computing packages. Likewise, it is straightfoward to verify that:

$$\frac{\boldsymbol{\sigma}_{t|s}^2}{1-\boldsymbol{\sigma}_{t|s}^2} = \text{expm1}(\text{softplus}(\gamma(t)) - \text{softplus}(\gamma(s))) \tag{55}$$

F Comparison to DDPM and NCSN objectives

Previous works using denoising diffusion models [Ho et al., 2020, Song and Ermon, 2019, Nichol and Dhariwal, 2021] used a training objective that can be understood as a *weighted* diffusion loss of the form given in Equation 25:

$$\mathcal{L}_\infty(\mathbf{x},w) = \frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\int_{\text{SNR}_{\text{min}}}^{\text{SNR}_{\text{max}}} w(v)\left\|\mathbf{x}-\tilde{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_v,v)\right\|_2^2 dv \tag{56}$$

$$= -\frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\int_0^1 \text{SNR}'(t)w(\text{SNR}(t))\left\|\mathbf{x}-\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t;t)\right\|_2^2 dt \tag{57}$$

$$= \frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\int_0^1 \gamma'(t)w(\exp(-\gamma(t)))\left\|\boldsymbol{\epsilon}-\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{z}_t;t)\right\|_2^2 dt, \tag{58}$$

where $\gamma(t) = -\log\text{SNR}(t)$.

When using the loss in Equation 56, we set $w(v) = 1$, corresponding to optimization of a variational bound on the likelihood of the data. Ho et al. [2020], Song and Ermon [2019], Nichol and Dhariwal [2021] instead choose to minimize the *simple objective* defined as

$$L_{\text{simple}}(\mathbf{x}) \equiv \int_0^1 \left\|\boldsymbol{\epsilon}-\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{z}_t,t)\right\|_2^2 dt, \tag{59}$$

or a discrete-time version of this.

Comparing Equation 59 with Equation 58, we can see that the loss used by Ho et al. [2020], Song and Ermon [2019], Nichol and Dhariwal [2021] corresponds to a weighting function $w(\text{SNR}(t)) =$

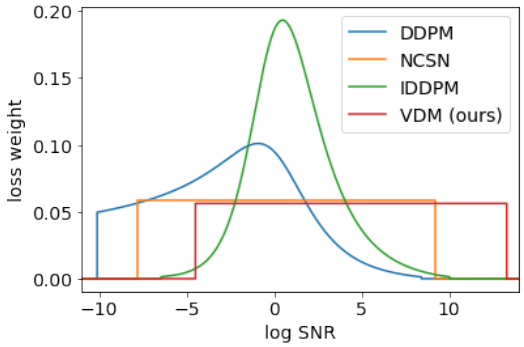


Figure 5: Implied weighting functions corresponding to the losses used by Ho et al. [2020], Song et al. [2020], and Nichol and Dhariwal [2021], as well as our proposed loss. NCSN [Song et al., 2020] uses a constant implied weighting function, and is thus consistent with maximization of the variational bound like we propose in this paper. However, unlike Song and Ermon [2019] we also learn the endpoints SNR_{min}, SNR_{max}, which results in a better optimized VLB value. DDPM [Ho et al., 2020] and improved DDPM [Nichol and Dhariwal, 2021] instead use implied weighting functions that put relatively more weight on the noisy data with low to medium signal-to-noise ratio. The latter two works report better FID and Inception Score than Song et al. [2020] and the current paper, which we hypothesize is due to their loss emphasizing the global consistence and coarse level patterns more than the fine scale features of the data.

$1/\gamma'(t)$. Below, we derive the $\gamma(t)$, and thus the weighting function $w(v)$, corresponding to the diffusion processes used by Ho et al. [2020], Song and Ermon [2019], Nichol and Dhariwal [2021]. We visualize these weighting functions in Figure 5.

For DDPM, Ho et al. [2020] use a diffusion process in discrete time with $\alpha_i = \sqrt{\prod_{j=1}^{i}(1-\beta_j)}$, $\sigma_i^2 = 1-\alpha_i^2$, where β_i linearly interpolates between $\beta_1 = 1e^{-4}$ and $\beta_T = 0.02$ in $T = 1000$ discrete steps. When defining time $t = i/T$, this can be closely approximated as $\alpha_t^2 = \exp(-1e^{-4} - 10t^2)$, and correspondingly with $\text{SNR}(t) = 1/\text{expm1}(1e^{-4} + 10t^2)$ or $\gamma(t) = \log[\text{expm1}(1e^{-4} + 10t^2)]$, where $\text{expm1}(x) = \exp(x) - 1$. This approximation is shown in Figure 6.

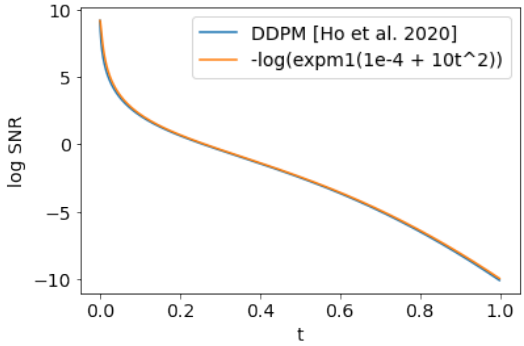


Figure 6: Log signal-to-noise ratio for the discrete-time diffusion process in Ho et al. [2020] and our continous-time approximation.

For NCSNv2, Song and Ermon [2020] instead use $\alpha_t = 1$ and let σ_t be a geometric series interpolating between 0.01 and 50, i.e. $\sigma_t^2 = \exp(\gamma(t))$ with $\gamma(t) = 2\log[0.01] + 2\log[5000]t$. This means that $\gamma'(t) = 2\log[5000]$ and thus that $w(v)$ is a constant. The procedure proposed by Song and Ermon [2020] is thus consistent with maximization of the VLB like we propose here. The same holds for [Song and Ermon, 2019].

For IDDPM, Nichol and Dhariwal [2021] use $\tilde{\alpha}_t = \cos(\frac{t+0.008}{1.008}\frac{\pi}{2})/\cos(\frac{0.008}{1.008}\frac{\pi}{2})$. The values for $\tilde{\alpha}_t$ are then translated into value for β_t, which are then clipped to 0.999. Subsequently we can then derive

the $\alpha_t, \sigma_t, \gamma(t)$ corresponding to those β_t. Due to the clipping these expressions do not simplify, but we include their numerical results in Figure 5.

G Consistency

Let $q(\mathbf{x})$ denote the marginal distribution of data $\mathbf{x}$, and let:

$$q(\mathbf{z}_t) = \int q(\mathbf{z}_t|\mathbf{x})q(\mathbf{x})d\mathbf{x} \tag{60}$$

Here we will show that derived estimators are *consistent* estimators, in the sense that with infinite data, the optimal score model $\mathbf{s}^*_{\boldsymbol{\theta}}(\mathbf{z}_t; t)$ is such that:

$$\mathbf{s}^*_{\boldsymbol{\theta}}(\mathbf{z}_t; t) = \nabla_{\mathbf{z}} \log q(\mathbf{z}_t) \tag{61}$$

Note that $\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{x}) = -\boldsymbol{\epsilon}/\boldsymbol{\sigma}_t$. We can rewrite the diffusion loss (discrete time or continuous time) for timestep t as:

$$\mathcal{L}_T(\mathbf{x}; t) = \frac{1}{2}\mathbb{E}_{q(\mathbf{x},\mathbf{z}_t)}\left[\left\|\sqrt{\mathbf{c}(t)}\left(\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)\right)\right\|_2^2\right] \tag{62}$$

where $\sqrt{\mathbf{c}(t)}$ is a time-dependent weighting factor.

In [Ho et al., 2020], it is noted that the discrete-time VLB, when using equal variances across dimensions, is equivalent to a Denoising Score Matching (DSM) objective [Vincent, 2011]. This is interesting, since it implies consistency. We generalize this original consistency proof of DSM to a more general case of different noises schedules per dimension, and arbitrary multipliers $\sqrt{\mathbf{c}_1}$ and $\sqrt{\mathbf{c}_2}$ in front of the scores, i.e. where the dimensions of $\mathbf{z}$ are differently weighted. Note, however that we'll only need the special case where $\sqrt{\mathbf{c}} = \sqrt{\mathbf{c}_1} = \sqrt{\mathbf{c}_2}$. First, note that:

$$\begin{aligned}\tfrac{1}{2}\mathbb{E}_{q(\mathbf{z}_t)}\left[||\sqrt{\mathbf{c}_1}\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t) - \sqrt{\mathbf{c}_2}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t)||_2^2\right] &= \tfrac{1}{2}\mathbb{E}_{q(\mathbf{z}_t)}\left[||\sqrt{\mathbf{c}_1}\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t)||_2^2\right] \\ &+ \tfrac{1}{2}\mathbb{E}_{q(\mathbf{z}_t)}\left[||\sqrt{\mathbf{c}_2}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t)||_2^2\right] \\ &- \mathbb{E}_{q(\mathbf{z}_t)}\left[\langle\sqrt{\mathbf{c}_1}\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t), \sqrt{\mathbf{c}_2}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t)\rangle\right]\end{aligned} \tag{63}$$

Where $\langle ., . \rangle$ denotes a dot product. Similarly:

$$\begin{aligned}\tfrac{1}{2}\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})}\left[||\sqrt{\mathbf{c}_1}\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{x}) - \sqrt{\mathbf{c}_2}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t)||_2^2\right] &= \tfrac{1}{2}\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})}\left[||\sqrt{\mathbf{c}_1}\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{x})||_2^2\right] \\ &+ \tfrac{1}{2}\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})}\left[||\sqrt{\mathbf{c}_2}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t)||_2^2\right] \\ &- \mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})}\left[\langle\sqrt{\mathbf{c}_1}\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{x}), \sqrt{\mathbf{c}_2}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t)\rangle\right]\end{aligned} \tag{64}$$

The second terms of the right-hand sides of Equation 63 and Equation 64 are equal. The third terms of the right-hand sides of Equation 63 and Equation 64 are also equal:

$$\mathbb{E}_{q(\mathbf{z}_t)}\left[\langle\sqrt{\mathbf{c}_2}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t), \sqrt{\mathbf{c}_1}\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t)\rangle\right] \tag{65}$$

$$= \mathbb{E}_{q(\mathbf{z}_t)}\left[\left\langle\sqrt{\mathbf{c}_2}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t), \sqrt{\mathbf{c}_1}\frac{\nabla_{\mathbf{z}_t} q(\mathbf{z}_t)}{q(\mathbf{z}_t)}\right\rangle\right] \tag{66}$$

$$= \int_{\mathbf{z}_t}\langle\sqrt{\mathbf{c}_2}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t), \sqrt{\mathbf{c}_1}\nabla_{\mathbf{z}_t} q(\mathbf{z}_t)\rangle\, d\mathbf{z}_t \tag{67}$$

$$= \int_{\mathbf{z}_t}\left\langle\sqrt{\mathbf{c}_2}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t), \sqrt{\mathbf{c}_1}\nabla_{\mathbf{z}_t}\int_{\mathbf{x}} q(\mathbf{x})q(\mathbf{z}_t|\mathbf{x})d\mathbf{x}\right\rangle d\mathbf{z}_t \tag{68}$$

$$= \int_{\mathbf{z}_t}\left\langle\sqrt{\mathbf{c}_2}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t), \sqrt{\mathbf{c}_1}\int_{\mathbf{x}} q(\mathbf{x})\nabla_{\mathbf{z}_t} q(\mathbf{z}_t|\mathbf{x})d\mathbf{x}\right\rangle d\mathbf{z}_t \tag{69}$$

$$= \int_{\mathbf{z}_t}\left\langle\sqrt{\mathbf{c}_2}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t), \sqrt{\mathbf{c}_1}\int_{\mathbf{x}} q(\mathbf{x})q(\mathbf{z}_t|\mathbf{x})\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{x})d\mathbf{x}\right\rangle d\mathbf{z}_t \tag{70}$$

$$= \int_{\mathbf{x}}\int_{\mathbf{z}_t} q(\mathbf{x})q(\mathbf{z}_t|\mathbf{x})\left\langle\sqrt{\mathbf{c}_2}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t), \sqrt{\mathbf{c}_1}\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{x})\right\rangle d\mathbf{z}_t d\mathbf{x} \tag{71}$$

$$= \mathbb{E}_{q(\mathbf{x},\mathbf{z}_t)}\left[\langle\sqrt{\mathbf{c}_2}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t), \sqrt{\mathbf{c}_1}\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{x})\rangle\right] \tag{72}$$

So, only the first term of the right-hand sides of Equation 63 and Equation 64 are not equal. It follows that:

$$\frac{1}{2}\mathbb{E}_{q(\mathbf{z}_t)}\left[||\sqrt{\mathbf{c}_1}\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t) - \sqrt{\mathbf{c}_2}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t)||_2^2\right] = \frac{1}{2}\mathbb{E}_{q(\mathbf{x},\mathbf{z}_t)}\left[||\sqrt{\mathbf{c}_1}\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{x}) - \sqrt{\mathbf{c}_2}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t)||_2^2\right] + \text{constant} \tag{73}$$

where $\text{constant} = \frac{1}{2}\mathbb{E}_{q(\mathbf{z}_t)}\left[||\sqrt{\mathbf{c}_1}\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t)||_2^2\right] - \frac{1}{2}\mathbb{E}_{q(\mathbf{x},\mathbf{z}_t)}\left[||\sqrt{\mathbf{c}_1}\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{x})||_2^2\right]$ is constant w.r.t. the EBM $E()$. In the special case where $\sqrt{\mathbf{c}_1} = \sqrt{\mathbf{c}_2}$, we have:

$$\frac{1}{2}\mathbb{E}_{q(\mathbf{z}_t)}\left[||\sqrt{\mathbf{c}_1}(\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t))||_2^2\right] = \frac{1}{2}\mathbb{E}_{q(\mathbf{x},\mathbf{z}_t)}\left[||\sqrt{\mathbf{c}_1}(\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t))||_2^2\right] + \text{constant} \tag{74}$$

Therefore, minimizing the first term on the right-hand side of Equation 74 w.r.t. $E()$ (a denoising score matching objective with differently weighted dimensions) is equivalent to minimizing the left-hand side of Equation 74 w.r.t. $E()$. From this equation, it is clear that at the optimum of this DSM objective, for any positive $\mathbf{c}_1$:

$$\mathbf{s}^*_{\boldsymbol{\theta}}(\mathbf{z}_t) = \nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t) \tag{75}$$

If the score model is parameterized as the gradient of an EBM $E()$, then this implies that for all $t \in [0, 1]$:

$$\exp(-E^*(\mathbf{z}_t; t)) \propto q(\mathbf{z}_t) \tag{76}$$

So, when optimizing for the diffusion loss, the EBM $E(.; t)$ will approximate the correct marginals corresponding the inference model.

H Additional samples from our models

We include additional uncurated random samples from our unconditional models trained on CIFAR-10, 32x32 Imagenet, and 64x64 Imagenet. See Figures 7, 8, and 9.

I Lossless compression

For a fixed number of evaluation timesteps T_{eval}, our diffusion model in discrete time is a hierarchical latent variable model that can be turned into a lossless compression algorithm using bits-back coding [Hinton and Van Camp, 1993]. Assuming a source of auxiliary random bits is available alongside the data, bits-back coding encodes a latent and data together, with the latent sampled from the approximate posterior using the auxiliary random bits. The net coding cost of bits-back coding is given by subtracting the number of bits needed to sample the latent from the number of bits needed to encode the latent and data using the reverse process, so the negative VLB of our discrete time model is the theoretical expected coding cost for bits-back coding.

As a proof of concept for lossless compression using our model, Table 2 reports net codelengths on the CIFAR10 test set for various settings of T_{eval} using BB-ANS [Townsend et al., 2018], a practical implementation of bits-back coding based on asymmetric numeral systems [Duda, 2009]. Since diffusion models have Markov forward and reverse processes, we use the Bit-Swap implementation of BB-ANS [Kingma et al., 2019]. Practical implementations of bits-back coding must discretize continuous latent variables and their associated continuous probability distributions; for simplicity, our implementation uses a uniform discretization of the continuous latents and their associated Gaussian conditionals from the forward and reverse processes. Additionally, we found it crucial to encrypt the ANS bitstream before each decoding operation to ensure clean bits for sampling from the approximate posterior; we did so by applying the XOR operation to the ANS bitstream with pseudorandom bits from a fixed sequence of seeds. For example, without cleaning the bitstream using encryption, compressing a batch of 100 examples using $T_{eval} = 250$ costs 2.74 bits per byte, but with encryption, the cost improves to 2.68 bits per dimension.

For a small number of timesteps T_{eval}, our bits-back implementation attains net codelengths that agree closely with the negative VLB, but there is some discrepancy for large T_{eval}. This is due to inaccuracies in the compression algorithm to represent discretized Gaussians with small standard deviations, and small discrepancies in codelength compound into a gap of up to 0.05 bits per

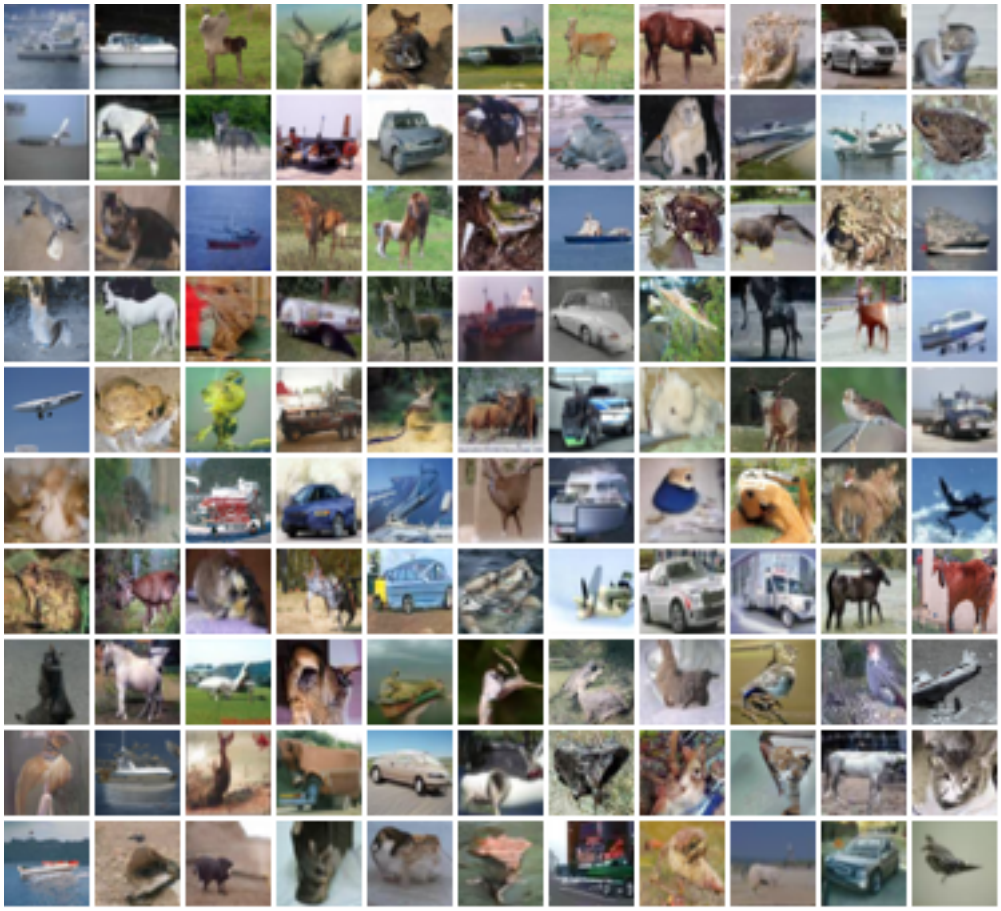


Figure 7: Random samples from an unconditional diffusion model trained on CIFAR-10 for 2 million parameter updates. The model was trained in continuous-time, and sampled using $T = 1000$.

dimension when T is large. (In prior work, e.g. [Kingma et al., 2019, Ho et al., 2019b, Townsend et al., 2020], practical implementations of bits-back coding have been tested on latent variable models with only tens of layers, not hundreds.) In addition, a large number of timesteps makes compression computationally expensive, because a neural network forward pass must be run for each timestep. Closing the codelength gap with an efficient implementation of bits-back coding for a large number of timesteps is an interesting avenue for future work.

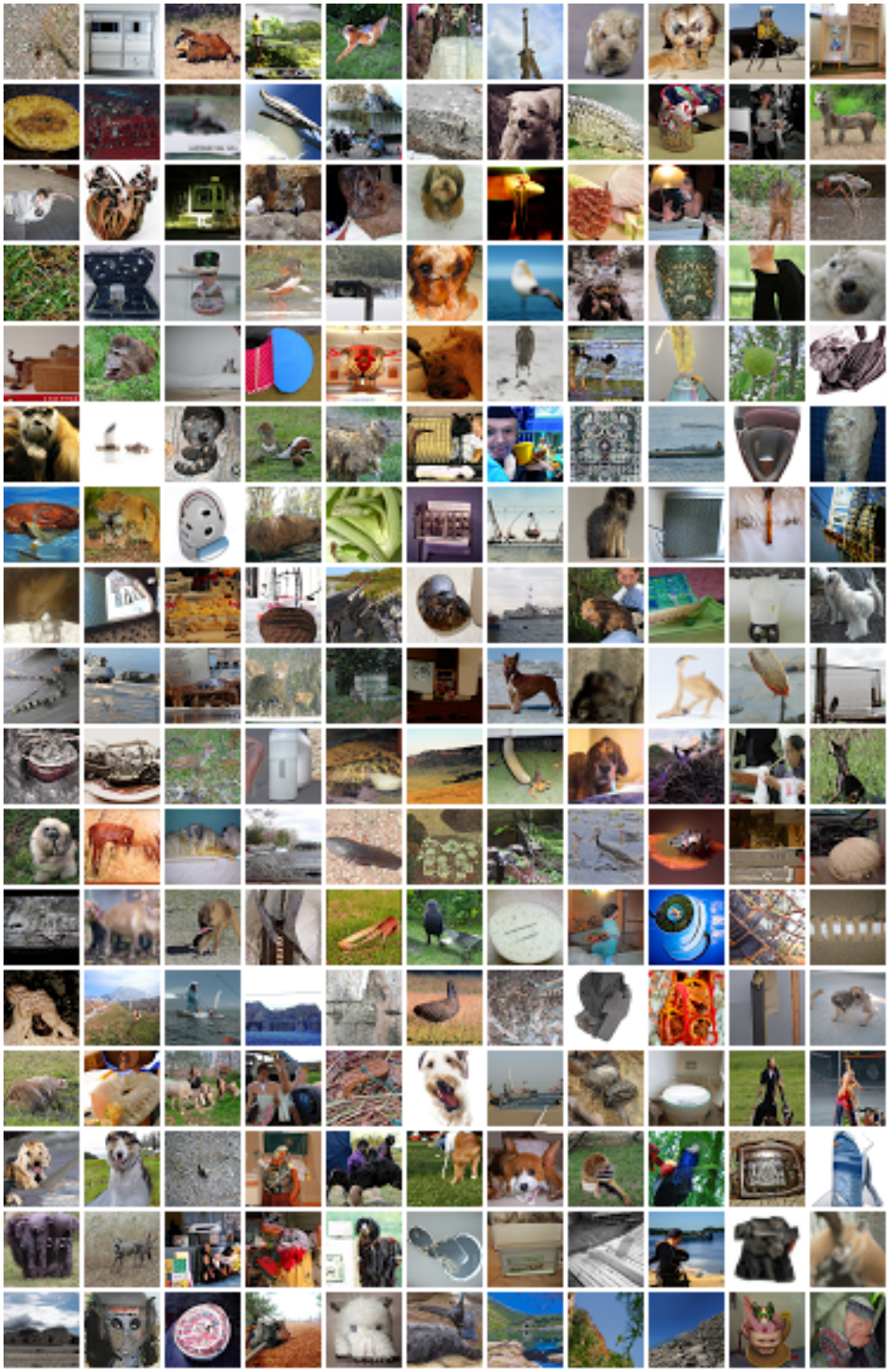


Figure 8: Random samples from an unconditional diffusion model trained on 32x32 ImageNet for 3.7 million parameter updates. The model was trained in continuous-time, and sampled using $T = 1000$.

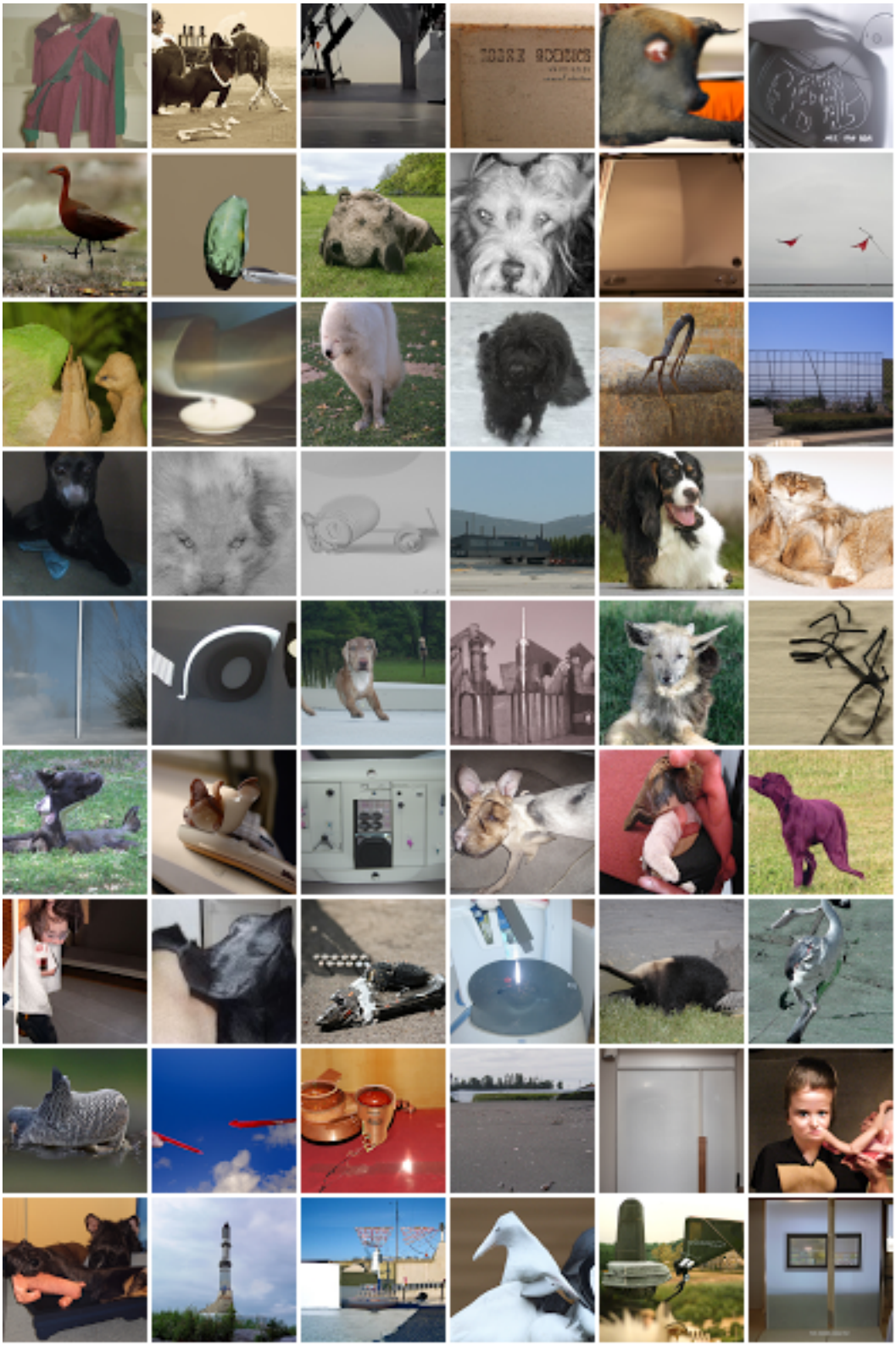


Figure 9: Random samples from an unconditional diffusion model trained on 64x64 ImageNet for 2 million parameter updates. The model was trained in continuous-time, and sampled using $T = 1000$.