Diffusion Models With Learned Adaptive Noise

Anonymous Author(s)
Affiliation
Address
email

Abstract

Diffusion models have gained traction as powerful algorithms for synthesizing high-quality images. Central to these algorithms is the diffusion process, which maps data to noise according to equations inspired by thermodynamics and can significantly impact performance. A widely held assumption is that the ELBO objective of a diffusion model is invariant to the noise process (Kingma et al., 2021). In this work, we dispel this assumption—we propose multivariate learned adaptive noise (MuLAN), a learned diffusion process that applies Gaussian noise at different rates across an image. Our method consists of three components—a multivariate noise schedule, instance-conditional diffusion, and auxiliary variables—which ensure that the learning objective is no longer invariant to the choice of the noise schedule as in previous works. Our work is grounded in Bayesian inference and casts the learned diffusion process as an approximate variational posterior that yields a tighter lower bound on marginal likelihood. Empirically, MuLAN sets a new state-of-the-art in density estimation on CIFAR-10 and ImageNet compared to classical diffusion.

1 Introduction

A diffusion process q transforms an input datapoint denoted by $\mathbf{x}_0$ and sampled from a distribution $q(\mathbf{x}_0)$ into a sequence of noisy data instances $\mathbf{x}_t$ for $t \in [0, 1]$ by progressively adding Gaussian noise of increasing magnitude. (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020). The marginal distribution of each latent is defined by $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \alpha_t\mathbf{x}_0, \sigma_t\boldsymbol{I})$ where the diffusion parameters $\alpha_t, \sigma_t \in \mathbb{R}^+$ implicitly define a noise schedule as a function of t, such that $\nu(t) = \alpha_t^2/\sigma_t^2$ is a monotonically decreasing function in t. Given any discretization of time into T timesteps of width $1/T$, we define $t(i) = i/T$ and $s(i) = (i-1)/T$ and we use $\mathbf{x}_{0:1}$ to denote the subset of variables associated with these timesteps; the forward process q can be shown to factorize into a Markov chain $q(\mathbf{x}_{0:1}) = q(\mathbf{x}_0)\left(\prod_{i=1}^{T} q(\mathbf{x}_{t(i)}|\mathbf{x}_{s(i)})\right)$.

The diffusion model p_θ is defined by a neural network (with parameters θ) used to denoise the forward process q. Given a discretization of time into T steps, p factorizes as $p_\theta(\mathbf{x}_{0:1}) = p_\theta(\mathbf{x}_1)\prod_{i=1}^{T} p_\theta(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)})$. We treat the $\mathbf{x}_t$ for $t > 0$ as latent variables and fit p_θ by maximizing the evidence lower bound (ELBO) on the marginal log-likelihood given by:

$$\log p_\theta(\mathbf{x}_0) = \text{ELBO}(p_\theta, q) + \text{D}_{\text{KL}}[q(\mathbf{x}_{t(1):t(T)}|\mathbf{x}_0)\|p_\theta(\mathbf{x}_{t(1):t(T)}|\mathbf{x}_0)] \geq \text{ELBO}(p_\theta, q) \quad (1)$$

In most works, the noise schedule, as defined by $\nu(t)$, is either fixed or treated as a hyperparameter (Ho et al., 2020; Chen, 2023; Hoogeboom et al., 2023). Chen (2023); Hoogeboom et al. (2023) show that the noise schedule can have a significant impact on sample quality. Kingma et al. (2021) consider learning $\nu(t)$, but argue that the KL divergence terms in the ELBO are invariant to the choice of function ν, except for the initial values $\nu(0), \nu(1)$, and they set these values to

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

hand-specified constants in their experiments. They only consider learning ν for the purpose of minimizing the variance of the gradient of the ELBO. In this work, we show that the ELBO is not invariant to more complex forward processes.

2 Diffusion Models With Multivariate Learned Adaptive Noise

Why Learned Diffusion? Perhaps the most direct motivation for our work comes from Bayesian inference. Notice that the gap between the evidence lower bound $\text{ELBO}(p, q)$ and the marginal log-likelihood (MLL) in Eq. 1 is precisely the KL divergence $\text{D}_{\text{KL}}[q(\mathbf{x}_{t(1):t(T)}|\mathbf{x}_0)\|p_\theta(\mathbf{x}_{t(1):t(T)}|\mathbf{x}_0)]$ between the diffusion process q over the latents $\mathbf{x}_t$ and the true posterior of the diffusion model. The diffusion process takes the role of an approximate variational posterior in $\text{ELBO}(p, q)$. This observation suggests that the ELBO can be made tighter by choosing a diffusion processes q that is closer to the true posterior $p_\theta(\mathbf{x}_{t(1):t(T)}|\mathbf{x}_0)$; this in turn brings the learning objective of closer to $\log p(\mathbf{x})$, which is often the ideal objective that we wish to optimize. In fact, the key idea of variational inference is to optimize $\max_{q\in\mathcal{Q}} \text{ELBO}(p, q)$ over a family of approximate posteriors $\mathcal{Q}$ to induce a tighter ELBO (Kingma & Welling, 2013). Most diffusion algorithms, however optimize $\max_{p\in\mathcal{P}} \text{ELBO}(p, q)$ within some family $\mathcal{P}$ with a fixed q. Our work seeks to jointly optimize $\max_{p\in\mathcal{P},q\in\mathcal{Q}} \text{ELBO}(p, q)$; we will show in our experiments that this improves the likelihood estimation.

2.1 A Forward Diffusion Process With Multivariate Adaptive Noise

This subsection focuses on defining $\mathcal{Q}$; the next sections will show how to parameterize and train a reverse model $p \in \mathcal{P}$.

Notation. Given two vectors **a** and **b**, we use the notation **ab** to represent the Hadamard product (element-wise multiplication). Additionally, we denote element-wise division of **a** by **b** as **a / b**. We denote the mapping diag(.) that takes a vector as input and produces a diagonal matrix as output.

2.1.1 Multivariate Gaussian Noise Schedule Conditioned on Context

Formally, our definition of a forward diffusion process with a multivariate noise schedule follows previous work (Kingma et al., 2021; Hoogeboom & Salimans, 2022) and defines q via the marginal for each latent noise variable $\mathbf{x}_t$ for $t \in [0, 1]$, where the marginal is given by: $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\alpha}_t\mathbf{x}_0, \text{diag}(\boldsymbol{\sigma}_t^2))$, where $\mathbf{x}_t, \mathbf{x}_0 \in \mathbb{R}^d$, $\boldsymbol{\alpha}_t, \boldsymbol{\sigma}_t \in \mathbb{R}^d_+$ and d is the dimensionality of the input data. For more details please refer Sec. B. In Sec. D.4, **we argue that this class of diffusion process $\mathcal{Q}$ induces an ELBO that is not invariant to $q \in \mathcal{Q}$.** The ELBO consists of a line integral along the diffusion trajectory specified by $\boldsymbol{\nu}(t)$. A line integrand is almost always path-dependent, unless its integral corresponds to a conservative force field, which is rarely the case for a diffusion process (Spinney & Ford, 2012). See Sec. D.4 for details.

Next, we extend the diffusion process to support context-adaptive noise. This enables injecting noise in a way that is dependent on the features of an image. Formally, we introduce a context variable $\mathbf{c} \in \mathbb{R}^m$ which encapsulates high-level information regarding $\mathbf{x}_0$. Examples of $\mathbf{c}$ could be a class label, a vector of attributes (e.g., features characterizing a human face), or even the input $\mathbf{x}_0$ itself. We define the marginal of the latent $\mathbf{x}_t$ in the forward process as $q(\mathbf{x}_t|\mathbf{x}_0, \mathbf{c}) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\alpha}_t(\mathbf{c})\mathbf{x}_0, \boldsymbol{\sigma}_t^2(\mathbf{c}))$; the reverse process kernel can be similarly derived as Hoogeboom & Salimans (2022):

$$q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0, \mathbf{c}) = \mathcal{N}\left(\boldsymbol{\mu}_q = \frac{\boldsymbol{\alpha}_{t|s}(\mathbf{c})\boldsymbol{\sigma}_s^2(\mathbf{c})}{\boldsymbol{\sigma}_t^2(\mathbf{c})}\mathbf{x}_t + \frac{\boldsymbol{\sigma}_{t|s}^2(\mathbf{c})\boldsymbol{\alpha}_s(\mathbf{c})}{\boldsymbol{\sigma}_t^2(\mathbf{c})}\mathbf{x}_0,\ \boldsymbol{\Sigma}_q = \text{diag}\left(\frac{\boldsymbol{\sigma}_s^2(\mathbf{c})\boldsymbol{\sigma}_{t|s}^2(\mathbf{c})}{\boldsymbol{\sigma}_t^2(\mathbf{c})}\right)\right) \tag{2}$$

where the diffusion parameters $\boldsymbol{\alpha}_t, \boldsymbol{\sigma}_t$ are now conditioned on $\mathbf{c}$ via a neural network. Specifically, we parameterize the diffusion parameters $\boldsymbol{\alpha}_t(\mathbf{c}), \boldsymbol{\sigma}_t(\mathbf{c}), \boldsymbol{\nu}(t, \mathbf{c})$ as $\boldsymbol{\alpha}_t^2(\mathbf{c}) = \text{sigmoid}(-\boldsymbol{\gamma}_\phi(\mathbf{c}, t))$, $\boldsymbol{\sigma}_t^2(\mathbf{c}) = \text{sigmoid}(\boldsymbol{\gamma}_\phi(\mathbf{c}, t))$, and $\boldsymbol{\nu}(\mathbf{c}, t) = \exp(-\boldsymbol{\gamma}_\phi(\mathbf{c}, t))$. Here, $\boldsymbol{\gamma}_\phi(\mathbf{c}, t) : \mathbb{R}^m \times [0, 1] \to [\gamma_{\text{min}}, \gamma_{\text{max}}]^d$ is a neural network with the property that $\boldsymbol{\gamma}_\phi(\mathbf{c}, t)$ is monotonic in t. Following Kingma et al. (2021); Zheng et al. (2023), we set $\gamma_{\text{min}} = -13.30$, $\gamma_{\text{max}} = 5.0$. We express $\boldsymbol{\gamma}_\phi(\mathbf{c}, t)$ as a monotonic degree 5 polynomial in t. Details about the exact functional form of this polynomial and its implementation can be found in Suppl. D.2. More such parameterizations are discussed in C.

2.2 Auxiliary-Variable Reverse Diffusion Processes

We introduce a class of approximate reverse processes $\mathcal{P}$ that match the structure of $\mathcal{Q}$ and that are naturally suited to the joint optimization $\max_{p\in\mathcal{P},q\in\mathcal{Q}} \text{ELBO}(p,q)$.

Formally, we define a diffusion model where the reverse diffusion process is conditioned on the context $\mathbf{c}$. Specifically, given any discretization of $t \in [0,1]$ into T time steps as in Sec. 1, we introduce a context-conditional diffusion model $p_\theta(\mathbf{x}_{0:1}|\mathbf{c})$ that factorizes as the Markov chain

$$p_\theta(\mathbf{x}_{0:1}|\mathbf{c}) = p_\theta(\mathbf{x}_1|\mathbf{c}) \prod_{i=1}^{T} p_\theta(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)}, \mathbf{c}). \tag{3}$$

Given that the true reverse process is a Gaussian as specified in Eq. 2, the ideal p_θ matches this parameterization (the proof mirrors that of regular diffusion models; Suppl. C), which yields

$$p_\theta(\mathbf{x}_s|\mathbf{c},\mathbf{x}_t) = \mathcal{N}\left(\boldsymbol{\mu}_p = \frac{\boldsymbol{\alpha}_{t|s}(\mathbf{c})\boldsymbol{\sigma}_s^2(\mathbf{c})}{\boldsymbol{\sigma}_t^2(\mathbf{c})}\mathbf{x}_t + \frac{\boldsymbol{\sigma}_{t|s}^2(\mathbf{c})\boldsymbol{\alpha}_s(\mathbf{c})}{\boldsymbol{\sigma}_t^2(\mathbf{c})}\mathbf{x}_\theta(\mathbf{x}_t,t),\ \boldsymbol{\Sigma}_p = \text{diag}\left(\frac{\boldsymbol{\sigma}_s^2(\mathbf{c})\boldsymbol{\sigma}_{t|s}^2(\mathbf{c})}{\boldsymbol{\sigma}_t^2(\mathbf{c})}\right)\right), \tag{4}$$

where $\mathbf{x}_\theta(\mathbf{x}_t,t)$, is a neural network that approximates $\mathbf{x}_0$. Instead of parameterizing $\mathbf{x}_\theta(\mathbf{x}_t,t)$ directly using a neural network, we consider 2 other parameterizations. One is the noise parameterization (Ho et al., 2020) where $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,\mathbf{c},t)$ is the denoising model which is parameterized as $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t) = (\mathbf{x}_t - \boldsymbol{\alpha}_t(\mathbf{c})\mathbf{x}_\theta(\mathbf{x}_t,t,\mathbf{c}))/\boldsymbol{\sigma}_t(\mathbf{c})$; see Suppl. D.1.1 and the other is velocity parameterization (Salimans & Ho, 2022) where $\mathbf{v}_\theta(\mathbf{x}_t,\mathbf{c},t)$ is a neural network that models $\mathbf{v}_\theta(\mathbf{x}_t,\mathbf{c},t) = (\boldsymbol{\alpha}_t(\mathbf{c})\mathbf{x}_t - \mathbf{x}_\theta(\mathbf{x}_t,\mathbf{c},t))/\boldsymbol{\sigma}_t(\mathbf{c})$; see Suppl. D.1.2.

2.2.1 Conditioning Noise on an Auxiliary Latent Variable

In Suppl. C.3, we highlight the challenges when $\mathbf{c}$ is deterministic, and hence, propose an alternative strategy for learning conditional forward and reverse processes p, q that feature the same structure and hence support efficient noise parameterization. Our approach is based on the introduction of auxiliary variables (Wang et al., 2023), which lift the distribution p_θ into an augmented latent space.

Specifically, we define $\mathbf{z} \in \mathbb{R}^m$ as a low-dimensional auxiliary latent variable and define a lifted $p_\theta(\mathbf{x},\mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$, where $p_\theta(\mathbf{x}|\mathbf{z})$ is the conditional diffusion model from Eq. 3 (with context $\mathbf{c}$ set to $\mathbf{z}$) and $p_\theta(\mathbf{z})$ is a simple prior (e.g., unit Gaussian or fully factored Bernoulli). The latents $\mathbf{z}$ can be interpreted as a high-level semantic representation of $\mathbf{x}$ that conditions both the forward and the reverse processes. Unlike $\mathbf{x}_{0:1}$, the $\mathbf{z}$ are not constrained to have a particular dimension and can be a low-dimensional vector of latent factors of variation. They can be continuous or discrete.

We form a learning objective for the lifted p_θ by applying the ELBO twice to obtain:

$$\log p_\theta(\mathbf{x}_0) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0|\mathbf{z})] - \text{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}_0)\|p_\theta(\mathbf{z})) \tag{5}$$

$$\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_0)}[\text{ELBO}(p_\theta(\mathbf{x}_{0:1}|\mathbf{z}), q_\phi(\mathbf{x}_{0:1}|\mathbf{z}))] - \text{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}_0)\|p_\theta(\mathbf{z})), \tag{6}$$

where $\text{ELBO}(p_\theta(\mathbf{x}_{0:1}|\mathbf{z}), q_\phi(\mathbf{x}_{0:1}|\mathbf{z}))$ denotes the variational lower bound of a diffusion model (defined in Eq. 1) with a forward process $q_\phi(\mathbf{x}_{0:1}|\mathbf{z})$ (defined in Eq. 2 and Sec. 2.1.1) and and an approximate reverse process $p_\theta(\mathbf{x}_{0:1}|\mathbf{z})$ (defined in Eq. 3), both conditioned on $\mathbf{z}$. The distribution $q_\phi(\mathbf{z}|\mathbf{x}_0)$ is an approximate posterior for $\mathbf{z}$ parameterized by a neural network with parameters ϕ.

Crucially, note that in the learning objective (Eq. 6), the context, which in this case is $\mathbf{z}$, is available at training time in both the forward and reverse processes. At generation time, we can still obtain a valid context vector by sampling an auxiliary latent from $p_\theta(\mathbf{z})$. Thus, this approach addresses the aforementioned challenges and enables us to use the noise parameterization in Eq. 4.

2.3 Variational Lower Bound

Next, we derive a precise formula for the learning objective (6) of the auxiliary-variable diffusion model. Using the objective of a diffusion model in (1) we can write (6) as the sum of four terms:

$$\log p_\theta(\mathbf{x}_0) \geq \mathbb{E}_{q_\phi}[\mathcal{L}_{\text{recons}} + \mathcal{L}_{\text{diffusion}} + \mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{latent}}], \tag{7}$$

The reconstruction loss, $\mathcal{L}_{\text{recons}}$, can be (stochastically and differentiably) estimated using standard techniques; see (Kingma & Welling, 2013), $\mathcal{L}_{\text{prior}} = -\text{D}_{\text{KL}}[q_\phi(\mathbf{x}_1|\mathbf{x}_0,\mathbf{z})\|p_\theta(\mathbf{x}_1)]$ is the

Table 1: Likelihood in bits per dimension (BPD) on the test set of CIFAR-10 and ImageNet. Results with "/" means they are not reported in the original papers. Model types are autoregressive (AR), normalizing flows (Flow), diffusion models (Diff). We only compare with results achieved without data augmentation.

Model	Type	CIFAR-10 (↓)	ImageNet (↓)
PixelCNN (Van den Oord et al., 2016)	AR	3.03	3.83
Image Transformer (Parmar et al., 2018)	AR	2.90	3.77
DDPM (Ho et al., 2020)	Diff	≤ 3.69	/
ScoreFlow (Song et al., 2021)	Diff	2.83	3.76
VDM (Kingma et al., 2021)	Diff	≤ 2.65	≤ 3.72
Flow Matching (Lipman et al., 2022)	Flow	2.99	/
Reflected Diffusion Models (Lou & Ermon, 2023)	Diff	2.68	3.74
MuLAN (**Ours**)	Diff	**2.55** $\pm 10^{-3}$	**3.67** $\pm 10^{-3}$

diffusion prior term, $\mathcal{L}_{\text{latent}} = -\mathrm{D}_{\mathrm{KL}}[q_\phi(\mathbf{z}|\mathbf{x}_0)\|p_\theta(\mathbf{z})]$ is the latent prior term, and $\mathcal{L}_{\text{diffusion}} = -\frac{1}{2}\mathbb{E}_{t\sim[0,1]}\left[(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t))^\top \operatorname{diag}\left(\nabla_t \boldsymbol{\gamma}(\mathbf{z}, t)\right)(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t))\right]$ where $\nabla_t \boldsymbol{\gamma}(\mathbf{z}, t) \in \mathbb{R}^d$ denotes the Jacobian of $\boldsymbol{\gamma}(\mathbf{z}, t)$ with respect to the scalar t. We try two different kinds of priors for $p_\theta(\mathbf{z})$: discrete ($\mathbf{z} \in \{0, 1\}^m$) and continuous ($\mathbf{z} \in \mathbb{R}^m$). The exact expression for $\mathcal{L}_{\text{prior}}$ can be found in Suppl. D.3.

3 Experiments

This section reports experimental results on the CIFAR-10 (Krizhevsky et al., 2009) and ImageNet-32 (Van Den Oord et al., 2016) datasets. More details can be found in Sec. F.

Likelihood Estimation. In Table 1, we present likelihood estimation results for MuLAN and recent methods on CIFAR-10 and ImageNet-32 using the VLB-estimate; details in Sec. H.1. Trained for 10M steps on CIFAR-10 and 2M steps on ImageNet-32, this MuLAN version employs noise parameterization, akin to VDM (Kingma et al., 2021). Applied on VDM, MuLAN with a learned multivariate noising schedule conditioned on auxiliary latent variables significantly improves BPD over vanilla VDM. Additionally, using the ODE-based exact likelihood estimate, MuLAN outperforms existing methods in density estimation on both datasets, trained for 8M steps on CIFAR-10 and 2M steps on ImageNet-32. In inference, we extract the underlying probability flow ODE, similar to Zheng et al. (2023). While Zheng et al. (2023) used additional techniques, combining them with MuLAN could enhance its performance.

Ablations and Noise Schedule. In Fig. 3 we ablate each component of MuLAN. As $\boldsymbol{\gamma}_\phi(\mathbf{z}, t)$ is multivariate, diverse noise schedules are expected for distinct input dimensions and $\mathbf{z} \sim p_\theta(\mathbf{z})$. In Fig. 2 with our CIFAR-10 model, we visualize the time-dependent variance of the noise schedule for different pixels, based on 128 samples $\mathbf{z} \sim p_\theta(\mathbf{z})$. Early schedule segments show increased variation, yet absolute variance is smaller than anticipated. Attempts to visualize noise schedules across diverse dataset images and areas (see Fig. 11) and with synthetic datasets exhibit no interpretable patterns, despite observable differences in likelihood estimation. We posit that alternative architectures and conditioning forms may unveil interpretable variations, a direction for future exploration.

4 Conclusion

In this study, we introduce MuLAN, a context-adaptive noise process that applies Gaussian noise at varying rates across input data. We present theoretical arguments challenging the prevailing notion that the likelihood of diffusion models remains independent of the noise schedules. We contend that this independence only holds true for univariate schedules, and in the case of multivariate schedules like MuLAN, different diffusion trajectories yield distinct likelihood estimates. Our evaluation of MuLAN spans multiple image datasets, where it outperforms state-of-the-art generative diffusion models.In general, MuLAN represents a promising avenue for advancing generative modeling.

References

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Ting Chen. On the importance of noise scheduling for diffusion models. *arXiv preprint arXiv:2301.10972*, 2023.

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*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

J.R. Dormand and P.J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980. ISSN 0377-0427. doi: https://doi.org/10.1016/0771-050X(80)90013-3. URL https://www.sciencedirect.com/science/article/pii/0771050X80900133.

Pavlos S. Efraimidis and Paul G. Spirakis. Weighted random sampling with a reservoir. *Information Processing Letters*, 97(5):181–185, 2006. ISSN 0020-0190. doi: https://doi.org/10.1016/j.ipl.2005.11.003. URL https://www.sciencedirect.com/science/article/pii/S002001900500298X.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Emiel Hoogeboom and Tim Salimans. Blurring diffusion models. *arXiv preprint arXiv:2209.05557*, 2022.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. *arXiv preprint arXiv:2301.11093*, 2023.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

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

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Aaron Lou and Stefano Ermon. Reflected diffusion models. In *International Conference on Machine Learning*, pp. 22675–22701. PMLR, 2023.

Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021.

Mathias Niepert, Pasquale Minervini, and Luca Franceschi. Implicit MLE: backpropagating through discrete exponential family distributions. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 14567–14579, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/7a430339c10c642c4b2251756fd1b484-Abstract.html.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International conference on machine learning*, pp. 4055–4064. PMLR, 2018.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Subham Sekhar Sahoo, Anselm Paulus, Marin Vlastelica, Vít Musil, Volodymyr Kuleshov, and Georg Martius. Backpropagation through combinatorial algorithms: Identity with projection works. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=JZMR727029.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

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.

John Skilling. The eigenvalues of mega-dimensional matrices. 1989. URL https://api.semanticscholar.org/CorpusID:117844915.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in neural information processing systems*, 34:1415–1428, 2021.

Richard E. Spinney and Ian J. Ford. Fluctuation relations: a pedagogical overview, 2012.

Benigno Uria, Iain Murray, and Hugo Larochelle. Rnade: The real-valued neural autoregressive density-estimator. *Advances in Neural Information Processing Systems*, 26, 2013.

Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pp. 1747–1756. PMLR, 2016.

Yingheng Wang, Yair Schiff, Aaron Gokaslan, Weishen Pan, Fei Wang, Christopher De Sa, and Volodymyr Kuleshov. Infodiffusion: Representation learning using information maximizing diffusion models. In *International Conference on Machine Learning*, pp. xxxx–xxxx. PMLR, 2023.

Sang Michael Xie and Stefano Ermon. Reparameterizable subset sampling via continuous relaxations. *arXiv preprint arXiv:1901.10517*, 2019.

Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. Improved techniques for maximum likelihood estimation for diffusion odes. *arXiv preprint arXiv:2305.03935*, 2023.

A Standard Diffusion Models

We have a Gaussian diffusion process that begins with the data $\mathbf{x}_0$, and defines a sequence of increasingly noisy versions of $\mathbf{x}_0$ which we call the latent variables $\mathbf{x}_t$, where t runs from $t = 0$ (least noisy) to $t = 1$ (most noisy). Given, T, we discretize time uniformly into T timesteps each with a width $1/T$. We define $t(i) = i/T$ and $s(i) = (i-1)/T$.

A.1 Forward Process

$$q(\mathbf{x}_t|\mathbf{x}_s) = \mathcal{N}(\alpha_{t|s}\mathbf{x}_s, \sigma^2_{t|s}\mathbf{I}_n) \tag{8}$$

where

$$\alpha_{t|s} = \frac{\alpha_t}{\alpha_s} \tag{9}$$

$$\sigma^2_{t|s} = \sigma^2_t - \frac{\alpha^2_{t|s}}{\sigma^2_s} \tag{10}$$

A.2 Reverse Process

Kingma et al. (2021) show that the distribution $q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)$ is also gaussian,

$$q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\boldsymbol{\mu}_q = \frac{\alpha_{t|s}\sigma^2_s}{\sigma^2_t}\mathbf{x}_t + \frac{\sigma^2_{t|s}\alpha_s}{\sigma^2_t}\mathbf{x}_0,\ \boldsymbol{\Sigma}_q = \frac{\sigma^2_s\sigma^2_{t|s}}{\sigma^2_t}\mathbf{I}_n\right) \tag{11}$$

Since during the reverse process, we don't have access to $\mathbf{x}_0$, we approximate it using a neural network $\mathbf{x}_\theta(\mathbf{x}_t, t)$ with parameters θ. Thus,

$$p_\theta(\mathbf{x}_s|\mathbf{x}_t) = \mathcal{N}\left(\boldsymbol{\mu}_p = \frac{\alpha_{t|s}\sigma^2_s}{\sigma^2_t}\mathbf{x}_t + \frac{\sigma^2_{t|s}\alpha_s}{\sigma^2_t}\mathbf{x}_\theta(\mathbf{x}_t, t),\ \boldsymbol{\Sigma}_p = \frac{\sigma^2_s\sigma^2_{t|s}}{\sigma^2_t}\mathbf{I}_n\right) \tag{12}$$

A.3 Variational Lower Bound

This corruption process q is the following markov-chain as $q(\mathbf{x}_{0:1}) = q(\mathbf{x}_0)\left(\prod_{i=1}^T q(\mathbf{x}_{t(i)}|\mathbf{x}_{s(i)})\right)$. In the reverse Rrocess, or the denoising process, p_θ, a neural network (with parameters θ) is used to denoise the noising process q. The reverse Rrocess factorizes as: $p_\theta(\mathbf{x}_{0:1}) = p_\theta(\mathbf{x}_1)\prod_{i=1}^T p_\theta(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)})$. Let $\mathbf{x}_\theta(\mathbf{x}_t, t)$ be the reconstructed input by a neural network from $\mathbf{x}_t$. Similar to Sohl-Dickstein et al. (2015); Kingma et al. (2021) we decompose the negative lower bound (VLB) as:

$$\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq \mathbb{E}_{q_\phi}\left[-\log\frac{p_\theta(\mathbf{x}_{t(0):t(T)})}{q_\phi(\mathbf{x}_{t(1):t(T)}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_{\mathbf{x}_{t(1)}\sim q(\mathbf{x}_{t(1)}|\mathbf{x}_0)}[-\log p_\theta(\mathbf{x}_0|\mathbf{x}_{t(1)})] \\
&\quad + \sum_{i=2}^T \mathbb{E}_{\mathbf{x}_{t(i)}|\mathbf{x}_0}\mathrm{D}_{\mathrm{KL}}[p_\theta(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)})\|q_\phi(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)}, \mathbf{x}_0)] \\
&\quad + \mathrm{D}_{\mathrm{KL}}[p_\theta(\mathbf{x}_1)\|q_\phi(\mathbf{x}_1|\mathbf{x}_0)] \\
&= \underbrace{\mathbb{E}_{\mathbf{x}_{t(1)}\sim q(\mathbf{x}_{t(1)}|\mathbf{x}_0)}[-\log p_\theta(\mathbf{x}_0|\mathbf{x}_{t(1)})]}_{\mathcal{L}_{\text{recons}}} \\
&\quad + \underbrace{\frac{T}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n), i\sim U\{2,T\}}\mathrm{D}_{\mathrm{KL}}[p_\theta(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)})\|q_\phi(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)}, \mathbf{x}_0)]}_{\mathcal{L}_{\text{diffusion}}} \\
&\quad + \underbrace{\mathrm{D}_{\mathrm{KL}}[p_\theta(\mathbf{x}_1)\|q_\phi(\mathbf{x}_1|\mathbf{x}_0)]}_{\mathcal{L}_{\text{prior}}}
\end{aligned} \tag{13}$$

The prior loss, $\mathcal{L}_{\text{prior}}$, and reconstruction loss, $\mathcal{L}_{\text{recons}}$, can be (stochastically and differentiably) estimated using standard techniques; see Kingma & Welling (2013). The diffusion loss, $\mathcal{L}_{\text{diffusion}}$, varies with the formulation of the noise schedule. We provide an exact formulation for it in the subsequent sections.

A.4 Diffusion Loss

For brevity, we use the notation s for $s(i)$ and t for $t(i)$. From Eq. 25 and Eq. 26 we get the following expression for $q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)$:

$$
\begin{aligned}
&\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)\|p_\theta(\mathbf{x}_s|\mathbf{x}_t)) \\
&= \frac{1}{2}\left((\boldsymbol{\mu}_q - \boldsymbol{\mu}_p)^\top \boldsymbol{\Sigma}_\theta^{-1}(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p) + \operatorname{tr}\left(\boldsymbol{\Sigma}_q\boldsymbol{\Sigma}_p^{-1} - \mathbf{I}_n\right) - \log\frac{|\boldsymbol{\Sigma}_q|}{|\boldsymbol{\Sigma}_p|}\right) \\
&= \frac{1}{2}(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p)^\top \Sigma_\theta^{-1}(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p)
\end{aligned}
$$

Substituting $\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q, \boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p$ from equation 12 and equation 11; for the exact derivation see Kingma et al. (2021)

$$
= \frac{1}{2}\left(\nu(s) - \nu(t)\right)\|(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t))\|_2^2 \tag{14}
$$

Thus $\mathcal{L}_{\text{diffusion}}$ is given by

$$
\begin{aligned}
&\mathcal{L}_{\text{diffusion}} \\
&= \lim_{T\to\infty}\frac{T}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n),i\sim U\{2,T\}}\mathrm{D}_{\mathrm{KL}}[p_\theta(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)})\|q_\phi(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)}, \mathbf{x}_0)] \\
&= \lim_{T\to\infty}\frac{1}{2}\sum_{i=2}^{T}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n)}\left(\nu(s) - \nu(t)\right)\|\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t)\|_2^2 \\
&= \frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n)}\left[\lim_{T\to\infty}\sum_{i=2}^{T}\left(\nu(s) - \nu(t)\right)\|\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t)\|_2^2\right] \\
&= \frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n)}\left[\lim_{T\to\infty}\sum_{i=2}^{T}T\left(\nu(s) - \nu(t)\right)\|\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t)\|_2^2\frac{1}{T}\right]
\end{aligned}
$$

Substituting $\lim_{T\to\infty} T(\nu(s) - \nu(t)) = \frac{\mathrm{d}}{\mathrm{d}t}\nu(t) \equiv \nu'(t)$; see Kingma et al. (2021)

$$
= \frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n)}\left[\int_0^1 \nu'(t)\|\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t)\|_2^2\right]\mathrm{d}t \tag{15}
$$

In practice instead of computing the integral is computed by MC sampling.

$$
= -\frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n),t\sim U[0,1]}\left[\nu'(t)\|\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t)\|_2^2\right] \tag{16}
$$

B Multivariate noise schedule

Our proposed forward diffusion process progressively induces varying amounts of Gaussian noise across different areas of the image. We introduce two new components relative to previous work: multivariate noise scheduling and context-adaptive noise.

Intuitively, a multivariate noise schedule injects noise at different rates for each pixel of an input image. This enables adapting the diffusion process to spatial variations within the image. We will also see that this change is sufficient to make the ELBO no longer invariant in q.

Formally, our definition of a forward diffusion process with a multivariate noise schedule follows previous work (Kingma et al., 2021; Hoogeboom & Salimans, 2022) and defines q via the marginal for each latent noise variable $\mathbf{x}_t$ for $t \in [0, 1]$, where the marginal is given by:

$$
q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\alpha}_t\mathbf{x}_0, \operatorname{diag}(\boldsymbol{\sigma}_t^2)), \tag{17}
$$

where $\mathbf{x}_t, \mathbf{x}_0 \in \mathbb{R}^d$, $\boldsymbol{\alpha}_t, \boldsymbol{\sigma}_t \in \mathbb{R}^d_+$ and d is the dimensionality of the input data. The $\boldsymbol{\alpha}_t, \boldsymbol{\sigma}_t$ denote varying amounts of signal and associated with each component (i.e., each pixel) of $\mathbf{x}_0$ as a function of time $t(i)$. Similarly to Kingma et al. (2021), we may define the multivariate signal-to-noise ratio as $\boldsymbol{\nu}(t) = \boldsymbol{\alpha}_t^2/\boldsymbol{\sigma}_t^2$ and we choose $\boldsymbol{\alpha}_t, \boldsymbol{\sigma}_t$ such that $\boldsymbol{\nu}(t)$ is monotonically decreasing in t along all dimensions and is differentiable in $t \in [0, 1]$. Let $\boldsymbol{\alpha}_{t|s} = \boldsymbol{\alpha}_t/\boldsymbol{\alpha}_s$ and $\boldsymbol{\sigma}^2_{t|s} = \boldsymbol{\sigma}^2_t - \boldsymbol{\alpha}^2_{t|s}/\boldsymbol{\sigma}^2_s$ with all operations applied elementwise. In Hoogeboom & Salimans (2022), show that these marginals induce transition kernels of the true reverse process between steps $s < t$ that are given by:

$$q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_s; \boldsymbol{\mu}_q = \frac{\boldsymbol{\alpha}_{t|s}\boldsymbol{\sigma}^2_s}{\boldsymbol{\sigma}^2_t}\mathbf{x}_t + \frac{\boldsymbol{\sigma}^2_{t|s}\boldsymbol{\alpha}_s}{\boldsymbol{\sigma}^2_t}\mathbf{x}_0,\ \boldsymbol{\Sigma}_q = \text{diag}\left(\frac{\boldsymbol{\sigma}^2_s\boldsymbol{\sigma}^2_{t|s}}{\boldsymbol{\sigma}^2_t}\right)\right) \tag{18}$$

In Sec. D.4, we argue that this class of diffusion process $\mathcal{Q}$ induces an ELBO that is not invariant to $q \in \mathcal{Q}$. The ELBO consists of a line integral along the diffusion trajectory specified by $\boldsymbol{\nu}(t)$. A line integrand is almost always path-dependent, unless its integral corresponds to a conservative force field, which is rarely the case for a diffusion process (Spinney & Ford, 2012). See Sec. D.4 for details.

For a multivariate noise schedule we have $\boldsymbol{\alpha}_t, \boldsymbol{\sigma}_t \in \mathbb{R}^{n\times n}$ where $t \in [0, 1]$. $\boldsymbol{\alpha}_t, \boldsymbol{\sigma}_t$ are diagonal matrices. The timesteps s, t satisfy $0 \le s < t \le 1$. Furthermore, we use the following notations where arithmetic division represents element wise division between 2 diagonal matrices:

$$\boldsymbol{\alpha}_{t|s} = \frac{\boldsymbol{\alpha}_t}{\boldsymbol{\alpha}_s} \tag{19}$$

$$\boldsymbol{\sigma}^2_{t|s} = \boldsymbol{\sigma}^2_t - \frac{\boldsymbol{\alpha}^2_{t|s}}{\boldsymbol{\sigma}^2_s} \tag{20}$$

B.1 Forward Process

$$q(\mathbf{x}_t|\mathbf{x}_s) = \mathcal{N}\left(\boldsymbol{\alpha}_{t|s}\mathbf{x}_s, \boldsymbol{\sigma}^2_{t|s}\right) \tag{21}$$

Change of variables. We can write $\mathbf{x}_t$ explicitly in terms of the signal-to-noise ratio, $\boldsymbol{\nu}(t)$, and input $\mathbf{x}_0$ in the following manner:

$$\boldsymbol{\nu}_t = \frac{\boldsymbol{\alpha}^2_t}{\boldsymbol{\sigma}^2_t}$$

We know $\alpha^2_t = 1 - \sigma^2_t$ for Variance Preserving process; see Sec. 1.

$$\implies \frac{1 - \boldsymbol{\sigma}^2_t}{\boldsymbol{\sigma}^2_t} = \boldsymbol{\nu}_t$$

$$\implies \boldsymbol{\sigma}^2_t = \frac{1}{1 + \boldsymbol{\nu}_t} \quad \text{and} \quad \boldsymbol{\alpha}^2_t = \frac{\boldsymbol{\nu}_t}{1 + \boldsymbol{\nu}_t} \tag{22}$$

$$\nu_t = \frac{\alpha^2_t}{\sigma^2_t}$$

We know $\alpha^2_t = 1 - \sigma^2_t$ for Variance Preserving process; see Sec. 1.

$$\implies \frac{1 - \sigma^2_t}{\sigma^2_t} = \nu_t$$

$$\implies \sigma^2_t = \frac{1}{1 + \nu_t} \quad \text{and} \quad \alpha^2_t = \frac{\nu_t}{1 + \nu_t} \tag{23}$$

Thus, we write $\mathbf{x}_t$ in terms of the signal-to-noise ratio in the following manner:

$$\begin{aligned}\mathbf{x}_{\boldsymbol{\nu}(t)} &= \boldsymbol{\alpha}_t\mathbf{x}_0 + \boldsymbol{\sigma}_t\boldsymbol{\epsilon}_t;\ \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \mathbf{I}_n) \\ &= \frac{\sqrt{\boldsymbol{\nu}(t)}}{\sqrt{1 + \boldsymbol{\nu}(t)}}\mathbf{x}_0 + \frac{1}{\sqrt{1 + \boldsymbol{\nu}(t)}}\boldsymbol{\epsilon}_t \qquad \text{Using Eq. 22}\end{aligned} \tag{24}$$

B.2 Reverse Process

The distribution of $\mathbf{x}_t$ given $\mathbf{x}_s$ is given by:

$$q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\boldsymbol{\mu}_q = \frac{\boldsymbol{\alpha}_{t|s}\boldsymbol{\sigma}_s^2}{\boldsymbol{\sigma}_t^2}\mathbf{x}_t + \frac{\boldsymbol{\sigma}_{t|s}^2\boldsymbol{\alpha}_s}{\boldsymbol{\sigma}_t^2}\mathbf{x}_0,\ \boldsymbol{\Sigma}_q = \text{diag}\left(\frac{\boldsymbol{\sigma}_s^2\boldsymbol{\sigma}_{t|s}^2}{\boldsymbol{\sigma}_t^2}\right)\right) \tag{25}$$

Let $\mathbf{x}_\theta(\mathbf{x}_t, t)$ be the neural network approximation for $\mathbf{x}_0$. Then we get the following reverse process:

$$p_\theta(\mathbf{x}_s|\mathbf{x}_t) = \mathcal{N}\left(\boldsymbol{\mu}_p = \frac{\boldsymbol{\alpha}_{t|s}\boldsymbol{\sigma}_s^2}{\boldsymbol{\sigma}_t^2}\mathbf{x}_t + \frac{\boldsymbol{\sigma}_{t|s}^2\boldsymbol{\alpha}_s}{\boldsymbol{\sigma}_t^2}\mathbf{x}_\theta(\mathbf{x}_t, t),\ \boldsymbol{\Sigma}_p = \text{diag}\left(\frac{\boldsymbol{\sigma}_s^2\boldsymbol{\sigma}_{t|s}^2}{\boldsymbol{\sigma}_t^2}\right)\right) \tag{26}$$

B.3 Diffusion Loss

For brevity we use the notation s for $s(i)$ and t for $t(i)$. From Eq. 25 and Eq. 26 we get the following expression for $q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)$:

$$\begin{aligned}
&\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)\|p_\theta(\mathbf{x}_s|\mathbf{x}_t)) \\
&= \frac{1}{2}\left((\boldsymbol{\mu}_q - \boldsymbol{\mu}_p)^\top\boldsymbol{\Sigma}_\theta^{-1}(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p) + \text{tr}\left(\boldsymbol{\Sigma}_q\boldsymbol{\Sigma}_p^{-1} - \mathbf{I}_n\right) - \log\frac{|\boldsymbol{\Sigma}_q|}{|\boldsymbol{\Sigma}_p|}\right) \\
&= \frac{1}{2}(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p)^\top\Sigma_\theta^{-1}(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p)
\end{aligned}$$

Substituting $\boldsymbol{\mu}_q, \boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p$ from equation 26 and equation 25.

$$\begin{aligned}
&= \frac{1}{2}\left(\frac{\boldsymbol{\sigma}_{t|s}^2\boldsymbol{\alpha}_s}{\boldsymbol{\sigma}_t^2}\mathbf{x}_0 - \frac{\boldsymbol{\sigma}_{t|s}^2\boldsymbol{\alpha}_s}{\boldsymbol{\sigma}_t^2}\mathbf{x}_\theta(\mathbf{x}_t, t)\right)^\top\text{diag}\left(\frac{\boldsymbol{\sigma}_s^2\boldsymbol{\sigma}_{t|s}^2}{\boldsymbol{\sigma}_t^2}\right)^{-1}\left(\frac{\boldsymbol{\sigma}_{t|s}^2\boldsymbol{\alpha}_s}{\boldsymbol{\sigma}_t^2}\mathbf{x}_0 - \frac{\boldsymbol{\sigma}_{t|s}^2\boldsymbol{\alpha}_s}{\boldsymbol{\sigma}_t^2}\mathbf{x}_\theta(\mathbf{x}_t, t)\right) \\
&= \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t))^\top\text{diag}\left(\frac{\boldsymbol{\sigma}_{t|s}^2\boldsymbol{\alpha}_s}{\boldsymbol{\sigma}_t^2}\right)^\top\text{diag}\left(\frac{\boldsymbol{\sigma}_s^2\boldsymbol{\sigma}_{t|s}^2}{\boldsymbol{\sigma}_t^2}\right)^{-1}\text{diag}\left(\frac{\boldsymbol{\sigma}_{t|s}^2\boldsymbol{\alpha}_s}{\boldsymbol{\sigma}_t^2}\right)(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t)) \\
&= \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t))^\top\text{diag}\left(\frac{\boldsymbol{\sigma}_{t|s}^2\boldsymbol{\alpha}_s}{\boldsymbol{\sigma}_t^2}\odot\frac{\boldsymbol{\sigma}_t^2}{\boldsymbol{\sigma}_s^2\boldsymbol{\sigma}_{t|s}^2}\odot\frac{\boldsymbol{\sigma}_{t|s}^2\boldsymbol{\alpha}_s}{\boldsymbol{\sigma}_t^2}\right)(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t)) \\
&= \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t))^\top\text{diag}\left(\frac{\boldsymbol{\sigma}_{t|s}^2\boldsymbol{\alpha}_s^2}{\boldsymbol{\sigma}_t^2\boldsymbol{\sigma}_s^2}\right)(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t))
\end{aligned}$$

Simplifying the expression using eq. 19 and eq. 20 we get,

$$= \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t))^\top\text{diag}\left(\frac{\boldsymbol{\alpha}_s^2}{\boldsymbol{\sigma}_s^2} - \frac{\boldsymbol{\alpha}_t^2}{\boldsymbol{\sigma}_t^2}\right)(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t))$$

Using the relation $\boldsymbol{\nu}(t) = \boldsymbol{\alpha}_t^2/\boldsymbol{\sigma}_t^2$ we get,

$$= \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t))^\top\text{diag}\left(\boldsymbol{\nu}(s) - \boldsymbol{\nu}(t)\right)(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t)) \tag{27}$$

Like Kingma et al. (2021) we train the model in the continuous domain with $T \to \infty$.

$$
\begin{aligned}
&\mathcal{L}_{\text{diffusion}} \\
&= \lim_{T\to\infty} \frac{1}{2}\sum_{i=2}^{T} \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n)} \mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)},\mathbf{x}_0)\|p_\theta(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)})) \\
&= \lim_{T\to\infty} \frac{1}{2}\sum_{i=2}^{T} \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n)} (\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_{t(i)}, t(i)))^\top \mathrm{diag}\left(\boldsymbol{\nu}_{s(i)} - \boldsymbol{\nu}_{t(i)}\right)(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_{t(i)}, t)) \\
&= \frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n)}\left[\lim_{T\to\infty}\sum_{i=2}^{T} (\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_{t(i)}, t(i)))^\top \mathrm{diag}\left(\boldsymbol{\nu}_{s(i)} - \boldsymbol{\nu}_{t(i)}\right)(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_{t(i)}, t))\right] \\
&= \frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n)}\left[\lim_{T\to\infty}\sum_{i=2}^{T} T(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_{t(i)}, t(i)))^\top \mathrm{diag}\left(\boldsymbol{\nu}_{s(i)} - \boldsymbol{\nu}_{t(i)}\right)(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_{t(i)}, t))\frac{1}{T}\right]
\end{aligned}
$$

Let $\lim_{T\to\infty} T(\boldsymbol{\nu}_{s(i)} - \boldsymbol{\nu}_{t(i)}) = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\nu}(t)$ denote the scalar derivative of the vector $\boldsymbol{\nu}(t)$ w.r.t t

$$
= \frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n)}\left[\int_0^1 (\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t))^\top \mathrm{diag}\left(\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\nu}(t)\right)(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t))\mathrm{d}t\right] \tag{28}
$$

In practice instead of computing the integral is computed by MC sampling.

$$
= -\frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n), t\sim U[0,1]}\left[(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t))^\top \mathrm{diag}\left(\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\nu}(t)\right)(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t))\right] \tag{29}
$$

B.4 Vectorized Representation of the diffusion loss

Let $\boldsymbol{\nu}(t)$ be the vectorized representation of the diagonal entries of the matrix $\boldsymbol{\nu}(t)$. We can rewrite the integral in eq. 28 in the following vectorized form where $\odot$ denotes element wise multiplication and $\langle , \rangle$ denotes dot product between 2 vectors.

$$
\begin{aligned}
&\mathcal{L}_{\text{diffusion}} \\
&= -\frac{1}{2}\int_0^1 (\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t))^\top \mathrm{diag}\left(\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\nu}(t)\right)(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t))\mathrm{d}t \\
&= -\frac{1}{2}\int_0^1 \left\langle (\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t)) \odot (\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t)), \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\nu}(t)\right\rangle \mathrm{d}t
\end{aligned}
$$

Using change of variables as mentioned in Sec. 2.1 we have

$$
= -\frac{1}{2}\int_0^1 \left\langle (\mathbf{x}_0 - \tilde{\mathbf{x}}_\theta(\mathbf{x}_{\boldsymbol{\nu}(t)}, \boldsymbol{\nu}(t))) \odot (\mathbf{x}_0 - \tilde{\mathbf{x}}_\theta(\mathbf{x}_{\boldsymbol{\nu}(t)}, \boldsymbol{\nu}(t))), \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\nu}(t)\right\rangle \mathrm{d}t
$$

Let $\mathbf{f}_\theta(\mathbf{x}_0, \boldsymbol{\nu}(t)) = (\mathbf{x}_0 - \tilde{\mathbf{x}}_\theta(\mathbf{x}_{\boldsymbol{\nu}(t)}, \boldsymbol{\nu}(t))) \odot (\mathbf{x}_0 - \tilde{\mathbf{x}}_\theta(\mathbf{x}_{\boldsymbol{\nu}(t)}, \boldsymbol{\nu}(t)))$

$$
= \int_0^1 \left\langle \mathbf{f}_\theta(\mathbf{x}_0, \boldsymbol{\nu}(t)), \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\nu}(t)\right\rangle \mathrm{d}t \tag{30}
$$

Thus $\mathcal{L}_{\text{diffusion}}$ can be interpreted as the amount of work done along the trajectory $\boldsymbol{\nu}(0) \to \boldsymbol{\nu}(1)$ in the presence of a vector field $\mathbf{f}_\theta(\mathbf{x}_0, \boldsymbol{\nu}(\mathbf{z}, t))$. From the perspective of thermodynamics, this is precisely equal to the amount of heat lost into the environment during the process of transition between 2 equilibria via the noise schedule specified by $\boldsymbol{\nu}(t)$.

B.5 Log likelihood and Noise Schedules: A Thermodynamics perspective

A diffusion model characterizes a quasi-static process that occurs between two equilibrium distributions: $q(\mathbf{x}_0) \to q(\mathbf{x}_1)$, via a stochastic trajectory (Sohl-Dickstein et al., 2015). According to Spinney & Ford (2012), it is demonstrated that the diffusion schedule or the noising process plays a pivotal role in determining the "measure of irreversibility" for this stochastic trajectory which is expressed as $\log \frac{P_F(\mathbf{x}_{0:1})}{P_B(\mathbf{x}_{1:0})}$. $P_F(\mathbf{x}_{0:1})$ represents the probability of observing the forward path $\mathbf{x}_{0:1}$ and $P_B(\mathbf{x}_{1:0})$ represents the probability of observing the reverse path $\mathbf{x}_{1:0}$. It's worth noting that

$\log \frac{P_F(\mathbf{x}_{0:1})}{P_B(\mathbf{x}_{1:0})}$ corresponds precisely to the ELBO Eq. 1 that we optimize when training a diffusion model. Consequently, thermodynamics asserts that the noise schedule indeed has an impact on the log-likelihood of the diffusion model which contradicts Kingma et al. (2021).

C Multivariate noise schedule conditioned on context

Let's say we have a context variable $\mathbf{c} \in \mathbb{R}^m$ that captures high level information about $\mathbf{x}_0$. $\boldsymbol{\alpha}_t(\mathbf{c}), \boldsymbol{\sigma}_t(\mathbf{c}) \in \mathbb{R}^{n \times n}$ are diagonal matrices. The timesteps s, t satisfy $0 \leq s < t \leq 1$. Furthermore, we use the following notations:

$$\boldsymbol{\alpha}_{t|s}(\mathbf{c}) = \frac{\boldsymbol{\alpha}_t(\mathbf{c})}{\boldsymbol{\alpha}_s(\mathbf{c})} \tag{31}$$

$$\boldsymbol{\sigma}^2_{t|s}(\mathbf{c}) = \boldsymbol{\sigma}^2_t(\mathbf{c}) - \frac{\boldsymbol{\alpha}^2_{t|s}(\mathbf{c})}{\boldsymbol{\sigma}^2_s(\mathbf{c})} \tag{32}$$

The forward process for such a method is given as:

$$q_\phi(\mathbf{x}_t|\mathbf{x}_s, \mathbf{c}) = \mathcal{N}\left(\boldsymbol{\alpha}_{t|s}(\mathbf{c})\mathbf{x}_s, \boldsymbol{\sigma}^2_{t|s}(\mathbf{c})\right) \tag{33}$$

The distribution of $\mathbf{x}_t$ given $\mathbf{x}_s$ is given by (the derivation is similar to Hoogeboom & Salimans (2022)):

$$\begin{aligned} &q_\phi(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0, \mathbf{c}) \\ &= \mathcal{N}\left(\boldsymbol{\mu}_q = \frac{\boldsymbol{\alpha}_{t|s}(\mathbf{c})\boldsymbol{\sigma}^2_s(\mathbf{c})}{\boldsymbol{\sigma}^2_t(\mathbf{c})}\mathbf{x}_t + \frac{\boldsymbol{\sigma}^2_{t|s}(\mathbf{c})\boldsymbol{\alpha}_s(\mathbf{c})}{\boldsymbol{\sigma}^2_t(\mathbf{c})}\mathbf{x}_0,\ \boldsymbol{\Sigma}_q = \text{diag}\left(\frac{\boldsymbol{\sigma}^2_s(\mathbf{c})\boldsymbol{\sigma}^2_{t|s}(\mathbf{c})}{\boldsymbol{\sigma}^2_t(\mathbf{c})}\right)\right) \end{aligned} \tag{34}$$

We explore various parameterizations for $\boldsymbol{\gamma}_\phi(\mathbf{c}, t)$. These schedules are designed in a manner that guarantees $\boldsymbol{\gamma}_\phi(\mathbf{c}, 0) = \gamma_{\text{min}}\mathbf{1_d}$ and $\boldsymbol{\gamma}_\phi(\mathbf{c}, 1) = \gamma_{\text{max}}\mathbf{1_d}$, Below, we list these parameterizations. The polynomial parameterization is novel to our work and yields significant performance gains.

Monotonic Neural Network. (Kingma et al., 2021) We use the monotonic neural network $\boldsymbol{\gamma}_{\text{vdm}}(t)$, proposed in VDM to express $\boldsymbol{\gamma}$ as a function of t such that $\boldsymbol{\gamma}_{\text{vdm}}(t) : [0, 1] \to [\gamma_{\text{min}}, \gamma_{\text{max}}]^d$. Then we use FiLM conditioning (Perez et al., 2018) in the intermediate layers of this network via a neural network that maps $\mathbf{z}$. The activations of the FiLM layer are constrained to be positive.

Sigmoid. (Ours) We express $\boldsymbol{\gamma}_\phi(\mathbf{c}, t)$ as a sigmoid function in t such that:
$\boldsymbol{\gamma}_\phi(\mathbf{c}, t) = \gamma_{\text{min}} + (\gamma_{\text{max}} - \gamma_{\text{min}})\frac{\text{sigmoid}(\mathbf{a}_\phi(\mathbf{c})t + \mathbf{b}_\phi(\mathbf{c})) - \text{sigmoid}(\mathbf{b}_\phi(\mathbf{c}))}{\text{sigmoid}(\mathbf{a}_\phi(\mathbf{c}) + \mathbf{b}_\phi(\mathbf{c})) - \text{sigmoid}(\mathbf{b}_\phi(\mathbf{c}))}$ where the coefficients $\mathbf{a}_\phi, \mathbf{b}_\phi$ are parameterized by a neural network such that $\mathbf{a}_\phi : \mathbb{R}^m \to \mathbb{R}^d_{>0}$, $\mathbf{b}_\phi : \mathbb{R}^m \to \mathbb{R}^d$.

Polynomial. (Ours) We express $\boldsymbol{\gamma}_\phi(\mathbf{c}, t)$ as a monotonic degree 5 polynomial in t . Details about the exact functional form of this polynomial and its implementation can be found in Suppl. D.2.

C.1 context is available during the inference time.

Even though $\mathbf{c}$ represents the input $\mathbf{x}_0$, it could be available during during inference. For example $\mathbf{c}$ could be class labels (Dhariwal & Nichol, 2021) or prexisting embeddings from an auto-encoder (Preechakul et al., 2022).

C.1.1 Reverse Process: Approximate

Let $\mathbf{x}_\theta(\mathbf{x}_t, \mathbf{c}, t)$ be an approximation for $\mathbf{x}_0$. Then we get the following reverse process (for brevity we write $\mathbf{x}_\theta(\mathbf{x}_t, \mathbf{c}, t)$ as $\mathbf{x}_\theta$):

$$p_\theta(\mathbf{x}_s|\mathbf{x}_t, \mathbf{c}) = \mathcal{N}\left(\boldsymbol{\mu}_p = \frac{\boldsymbol{\alpha}_{t|s}(\mathbf{c})\boldsymbol{\sigma}^2_s(\mathbf{c})}{\boldsymbol{\sigma}^2_t(\mathbf{c})}\mathbf{x}_t + \frac{\boldsymbol{\sigma}^2_{t|s}(\mathbf{c})\boldsymbol{\alpha}_s(\mathbf{c})}{\boldsymbol{\sigma}^2_t(\mathbf{c})}\mathbf{x}_\theta,\ \boldsymbol{\Sigma}_p = \text{diag}\left(\frac{\boldsymbol{\sigma}^2_s(\mathbf{c})\boldsymbol{\sigma}^2_{t|s}(\mathbf{c})}{\boldsymbol{\sigma}^2_t(\mathbf{c})}\right)\right) \tag{35}$$

C.1.2 Diffusion Loss

Similar to the derivation of multi-variate $\mathcal{L}_{\text{diffusion}}$ in Eq. 27 we can derive $\mathcal{L}_{\text{diffusion}}$ for this case too:

$$\mathcal{L}_{\text{diffusion}} = -\frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n),t\sim U[0,1]}\left[(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t,\mathbf{c},t))^\top \text{diag}\left(\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\nu}(t)\right)(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t,\mathbf{c},t))\right] \tag{36}$$

C.1.3 Limitations of this method

This approach is very limited where the diffusion process is only conditioned on class labels. Using pre-existing embeddings like Diff-AE (Preechakul et al., 2022) is also not possible in general and is only limited to tasks such as attribute manipulation in datasets.

C.2 context isn't available during the inference time.

If the context, $\mathbf{c}$ is an explicit function of the input $\mathbf{x}_0$ things become challenging because $\mathbf{x}_0$ isn't available during the inference stage. For this reason, Eq. 34 can't be used to parameterize $\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p$ in $p_\theta(\mathbf{x}_s|\mathbf{x}_t)$. Let $p_\theta(\mathbf{x}_s|\mathbf{x}_t) = \mathcal{N}(\boldsymbol{\mu}_p(\mathbf{x}_t,t), \boldsymbol{\Sigma}_p(\mathbf{x}_t,t))$ where $\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p$ are parameterized directly by a neural network. Using Eq. 2 we get the following diffusion loss:

$$\begin{aligned}\mathcal{L}_{\text{diffusion}} &= T\ \mathbb{E}_{i\sim U[0,T]}\mathrm{D}_{\mathrm{KL}}\left(q(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)},\mathbf{x}_0)\|p_\theta(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)})\right)\\ &= \mathbb{E}_{q_\phi}\left(\underbrace{\frac{T}{2}(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p)^\top\boldsymbol{\Sigma}_\theta^{-1}(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p)}_{\text{term 1}} + \underbrace{\frac{T}{2}\left(\text{tr}\left(\boldsymbol{\Sigma}_q\boldsymbol{\Sigma}_p^{-1} - \mathbf{I}_n\right) - \log\frac{|\boldsymbol{\Sigma}_q|}{|\boldsymbol{\Sigma}_p|}\right)}_{\text{term 2}}\right)\end{aligned} \tag{37}$$

C.2.1 Reverse Process: Approximate

Due to the challenges associated with parameterizing $\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p$ directly using a neural network we parameterize $\mathbf{c}$ using a neural network that approximates $\mathbf{c}$ in the reverse process. Let $\mathbf{x}_\theta(\mathbf{x}_t,t)$ be an approximation for $\mathbf{x}_0$. Then we get the following reverse Rrocess (for brevity we write $\mathbf{x}_\theta(\mathbf{x}_t,t)$ as $\mathbf{x}_\theta$, and $\mathbf{c}_\theta$ denotes an approximation to $\mathbf{c}$ in the reverse process.):

$$\begin{aligned}&p_\theta(\mathbf{x}_s|\mathbf{x}_t)\\ &= \mathcal{N}\left(\boldsymbol{\mu}_p = \frac{\boldsymbol{\alpha}_{t|s}(\mathbf{c}_\theta)\boldsymbol{\sigma}_s^2(\mathbf{c}_\theta)}{\boldsymbol{\sigma}_t^2(\mathbf{c}_\theta)}\mathbf{x}_t + \frac{\boldsymbol{\sigma}_{t|s}^2(\mathbf{c}_\theta)\boldsymbol{\alpha}_s(\mathbf{c}_\theta)}{\boldsymbol{\sigma}_t^2(\mathbf{c}_\theta)}\mathbf{x}_\theta,\ \boldsymbol{\Sigma}_p = \text{diag}\left(\frac{\boldsymbol{\sigma}_s^2(\mathbf{c}_\theta)\boldsymbol{\sigma}_{t|s}^2(\mathbf{c}_\theta)}{\boldsymbol{\sigma}_t^2(\mathbf{c}_\theta)}\right)\right)\end{aligned} \tag{38}$$

Consider the limiting case where $T \to \infty$. Let's analyze the 2 terms in Eq. 37 separately.

Using Eq. 2 and Eq. 4, **term 1** in Eq. 37 simplifies in the following manner:

$$\begin{aligned}&\lim_{T\to\infty}\frac{T}{2}(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p)^\top\boldsymbol{\Sigma}_\theta^{-1}(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p)\\ &\lim_{T\to\infty}\frac{T}{2}\sum_{i=1}^{d}\frac{((\boldsymbol{\mu}_q)_i - (\boldsymbol{\mu}_p)_i)^2}{(\boldsymbol{\Sigma}_\theta)_i}\end{aligned} \tag{39}$$

Substituting 1 / T as δ

$$\begin{aligned}&\lim_{\delta\to 0^+}\sum_{i=1}^{d}\frac{1}{\delta\boldsymbol{\sigma}_i{}^2(\mathbf{x}_\theta,t-\delta)\left(1-\frac{\boldsymbol{\nu}_i(\mathbf{x}_\theta,t)}{\boldsymbol{\nu}_i(\mathbf{x}_\theta,t-\delta)}\right)}\times\\ &\quad\left[\frac{\boldsymbol{\alpha}_i(\mathbf{x},t-\delta)}{\boldsymbol{\alpha}_i(\mathbf{x},t)}\frac{\boldsymbol{\nu}_i(\mathbf{x},t)}{\boldsymbol{\nu}_i(\mathbf{x},t-\delta)}\mathbf{z}_t + \boldsymbol{\alpha}_i(\mathbf{x},t-\delta)\left(1-\frac{\boldsymbol{\nu}_i(\mathbf{x},t)}{\boldsymbol{\nu}_i(\mathbf{x},t-\delta)}\right)x_i\right.\\ &\quad\left.-\frac{\boldsymbol{\alpha}_i(\mathbf{x}_\theta,t-\delta)}{\boldsymbol{\alpha}_i(\mathbf{x}_\theta,t)}\frac{\boldsymbol{\nu}_i(\mathbf{x}_\theta,t)}{\boldsymbol{\nu}_i(\mathbf{x}_\theta,t-\delta)}\mathbf{z}_t + \boldsymbol{\alpha}_i(\mathbf{x}_\theta,t-\delta)\left(1-\frac{\boldsymbol{\nu}_i(\mathbf{x}_\theta,t)}{\boldsymbol{\nu}_i(\mathbf{x}_\theta,t-\delta)}\right)(x_\theta)_i\right]^2\end{aligned} \tag{40}$$

Consider the scalar case: substituting $\delta = 1/T$,

$$\lim_{\delta \to 0} \frac{1}{\delta \sigma^2(\mathbf{x}_\theta, t-\delta)\left(1 - \frac{\nu(\mathbf{x}_\theta,t)}{\nu(\mathbf{x}_\theta,t-\delta)}\right)} \times$$
$$\left[\frac{\alpha(\mathbf{x},t-\delta)}{\alpha(\mathbf{x},t)}\frac{\nu(\mathbf{x},t)}{\nu(\mathbf{x},t-\delta)}\mathbf{z}_t + \alpha(\mathbf{x},t-\delta)\left(1 - \frac{\nu(\mathbf{x},t)}{\nu(\mathbf{x},t-\delta)}\right)\mathbf{x}\right.$$
$$\left. - \frac{\alpha(\mathbf{x}_\theta,t-\delta)}{\alpha(\mathbf{x}_\theta,t)}\frac{\nu(\mathbf{x}_\theta,t)}{\nu(\mathbf{x}_\theta,t-\delta)}\mathbf{z}_t + \alpha(\mathbf{x}_\theta,t-\delta)\left(1 - \frac{\nu(\mathbf{x}_\theta,t)}{\nu(\mathbf{x}_\theta,t-\delta)}\right)\mathbf{x}_\theta\right]^2 \tag{41}$$

Notice that this equation is in indeterminate for when we substitute $\delta = 0$. One can apply L'Hospital rule twice or break it down into 3 terms below. For this reason let's write it as

$$\text{expression 1: } \lim_{\delta \to 0} \frac{1}{\delta} \times \left[\frac{\alpha(\mathbf{x},t-\delta)}{\alpha(\mathbf{x},t)}\frac{\nu(\mathbf{x},t)}{\nu(\mathbf{x},t-\delta)}\mathbf{z}_t + \alpha(\mathbf{x},t-\delta)\left(1 - \frac{\nu(\mathbf{x},t)}{\nu(\mathbf{x},t-\delta)}\right)\mathbf{x}\right.$$
$$\left. - \frac{\alpha(\mathbf{x}_\theta,t-\delta)}{\alpha(\mathbf{x}_\theta,t)}\frac{\nu(\mathbf{x}_\theta,t)}{\nu(\mathbf{x}_\theta,t-\delta)}\mathbf{z}_t + \alpha(\mathbf{x}_\theta,t-\delta)\left(1 - \frac{\nu(\mathbf{x}_\theta,t)}{\nu(\mathbf{x}_\theta,t-\delta)}\right)\mathbf{x}_\theta\right] \tag{42}$$
$$\text{expression 2: } \lim_{\delta \to 0} \frac{1}{\left(1 - \frac{\nu(\mathbf{x}_\theta,t)}{\nu(\mathbf{x}_\theta,t-\delta)}\right)} \times \left[\frac{\alpha(\mathbf{x},t-\delta)}{\alpha(\mathbf{x},t)}\frac{\nu(\mathbf{x},t)}{\nu(\mathbf{x},t-\delta)}\mathbf{z}_t + \alpha(\mathbf{x},t-\delta)\left(1 - \frac{\nu(\mathbf{x},t)}{\nu(\mathbf{x},t-\delta)}\right)\mathbf{x}\right.$$
$$\left. - \frac{\alpha(\mathbf{x}_\theta,t-\delta)}{\alpha(\mathbf{x}_\theta,t)}\frac{\nu(\mathbf{x}_\theta,t)}{\nu(\mathbf{x}_\theta,t-\delta)}\mathbf{z}_t + \alpha(\mathbf{x}_\theta,t-\delta)\left(1 - \frac{\nu(\mathbf{x}_\theta,t)}{\nu(\mathbf{x}_\theta,t-\delta)}\right)\mathbf{x}_\theta\right]^2 \tag{43}$$

Applying L'Hospital rule in expression 1 we get,

$$\frac{d}{d\delta}\left(\frac{\alpha(\mathbf{x},t-\delta)}{\alpha(\mathbf{x},t)}\frac{\nu(\mathbf{x},t)}{\nu(\mathbf{x},t-\delta)}\right)\Bigg|_{\delta=0} = \frac{\nu(\mathbf{x},t)}{\alpha(\mathbf{x},t)}\frac{-\nu(\mathbf{x},t)\alpha'(\mathbf{x},t) + \alpha(\mathbf{x},t)\nu'(\mathbf{x},t)}{\nu^2(\mathbf{x},t)}$$
$$= \frac{-\alpha'(\mathbf{x},t)}{\alpha(\mathbf{x},t)} + \frac{\nu'(\mathbf{x},t)}{\nu(\mathbf{x},t)} \tag{44}$$
$$\frac{d}{d\delta}\alpha(\mathbf{x},t-\delta)\left(1 - \frac{\nu(\mathbf{x},t)}{\nu(\mathbf{x},t-\delta)}\right)\Bigg|_{\delta=0} = -\alpha(\mathbf{x},t)\frac{\nu'(\mathbf{x},t)}{\nu(\mathbf{x},t)} \tag{45}$$

$$\left[\left(\frac{-\alpha'(\mathbf{x},t)}{\alpha(\mathbf{x},t)} + \frac{\nu'(\mathbf{x},t)}{\nu(\mathbf{x},t)} + \frac{\alpha'(\mathbf{x}_\theta,t)}{\alpha(\mathbf{x}_\theta,t)} - \frac{\nu'(\mathbf{x}_\theta,t)}{\nu(\mathbf{x}_\theta,t)}\right)\mathbf{z}_t \tag{46}$$
$$-\alpha(\mathbf{x},t)\frac{\nu'(\mathbf{x},t)}{\nu(\mathbf{x},t)}\mathbf{x} + \alpha(\mathbf{x}_\theta,t)\frac{\nu'(\mathbf{x}_\theta,t)}{\nu(\mathbf{x}_\theta,t)}\mathbf{x}_\theta\Big]^2 \times \frac{\nu(\mathbf{x},t)}{\nu'(\mathbf{x},t)} \tag{47}$$

Thus the final result:

$$\sum_{i=1}^{d}\Bigg[\left(\frac{-\boldsymbol{\alpha}_i{}'(\mathbf{x},t)}{\boldsymbol{\alpha}_i(\mathbf{x},t)}+\frac{\boldsymbol{\nu}_i{}'(\mathbf{x},t)}{\boldsymbol{\nu}_i(\mathbf{x},t)}+\frac{\boldsymbol{\alpha}_i{}'(\mathbf{x}_\theta,t)}{\boldsymbol{\alpha}_i(\mathbf{x}_\theta,t)}-\frac{\boldsymbol{\nu}_i{}'(\mathbf{x}_\theta,t)}{\boldsymbol{\nu}_i(\mathbf{x}_\theta,t)}\right)\mathbf{z}_t$$
$$-\boldsymbol{\alpha}_i(\mathbf{x},t)\frac{\boldsymbol{\nu}_i{}'(\mathbf{x},t)}{\boldsymbol{\nu}_i(\mathbf{x},t)}\mathbf{x}+\boldsymbol{\alpha}_i(\mathbf{x}_\theta,t)\frac{\boldsymbol{\nu}_i{}'(\mathbf{x}_\theta,t)}{\boldsymbol{\nu}_i(\mathbf{x}_\theta,t)}\mathbf{x}_\theta\Bigg]^2\times\frac{\boldsymbol{\nu}_i(\mathbf{x},t)}{\boldsymbol{\nu}_i{}'(\mathbf{x},t)}$$
$$=\Lambda^\top\text{diag}\left(\frac{\boldsymbol{\nu}(\mathbf{x},t)}{\boldsymbol{\nu}'(\mathbf{x},t)}\right)\Lambda$$
$$\text{where }\Lambda=\left[\left(\frac{-\boldsymbol{\alpha}'(\mathbf{x},t)}{\boldsymbol{\alpha}(\mathbf{x},t)}+\frac{\boldsymbol{\nu}'(\mathbf{x},t)}{\boldsymbol{\nu}(\mathbf{x},t)}+\frac{\boldsymbol{\alpha}'(\mathbf{x}_\theta,t)}{\boldsymbol{\alpha}(\mathbf{x}_\theta,t)}-\frac{\boldsymbol{\nu}'(\mathbf{x}_\theta,t)}{\boldsymbol{\nu}(\mathbf{x}_\theta,t)}\right)\mathbf{z}_t-\boldsymbol{\alpha}(\mathbf{x},t)\frac{\boldsymbol{\nu}'(\mathbf{x},t)}{\boldsymbol{\nu}(\mathbf{x},t)}\mathbf{x}+\boldsymbol{\alpha}(\mathbf{x}_\theta,t)\frac{\boldsymbol{\nu}'(\mathbf{x}_\theta,t)}{\boldsymbol{\nu}(\mathbf{x}_\theta,t)}\mathbf{x}_\theta\right] \tag{48}$$

For the second term we have the following:

$$\lim_{T\to\infty}\frac{T}{2}\left(\text{tr}\left(\boldsymbol{\Sigma}_q\boldsymbol{\Sigma}_p^{-1}-\mathbf{I}_n\right)-\log\frac{|\boldsymbol{\Sigma}_q|}{|\boldsymbol{\Sigma}_p|}\right)$$
$$=\lim_{T\to\infty}\frac{T}{2}\Bigg[\text{tr}\left(\text{diag}\left(\boldsymbol{\sigma}^2(\mathbf{c},s)\left(1-\frac{\boldsymbol{\nu}(\mathbf{c},t)}{\boldsymbol{\nu}(\mathbf{c},s)}\right)\right)\Big/\text{diag}\left(\boldsymbol{\sigma}^2(\mathbf{c}_\theta,s)\left(1-\frac{\boldsymbol{\nu}(\mathbf{c}_\theta,t)}{\boldsymbol{\nu}(\mathbf{c}_\theta,s)}\right)\right)-\mathbf{I}_n\right)$$
$$-\log\frac{\left|\text{diag}\left(\boldsymbol{\sigma}^2(\mathbf{c},s)(1-\frac{\boldsymbol{\nu}(\mathbf{c},t)}{\boldsymbol{\nu}(\mathbf{c},s)})\right)\right|}{\left|\text{diag}\left(\boldsymbol{\sigma}^2(\mathbf{c}_\theta,s)(1-\frac{\boldsymbol{\nu}(\mathbf{c}_\theta,t)}{\boldsymbol{\nu}(\mathbf{c}_\theta,s)})\right)\right|}\Bigg]$$
$$=\lim_{T\to\infty}\frac{T}{2}\sum_{i=1}^{d}\left(\frac{{\boldsymbol{\sigma}_i}^2(\mathbf{c},s)\left(1-\frac{\boldsymbol{\nu}_i(\mathbf{c},t)}{\boldsymbol{\nu}_i(\mathbf{c},s)}\right)}{{\boldsymbol{\sigma}_i}^2(\mathbf{c}_\theta,s)\left(1-\frac{\boldsymbol{\nu}_i(\mathbf{c}_\theta,t)}{\boldsymbol{\nu}_i(\mathbf{c}_\theta,s)}\right)}-1-\log\frac{{\boldsymbol{\sigma}_i}^2(\mathbf{c},s)\left(1-\frac{\boldsymbol{\nu}_i(\mathbf{c},t)}{\boldsymbol{\nu}_i(\mathbf{c},s)}\right)}{{\boldsymbol{\sigma}_i}^2(\mathbf{c}_\theta,s)\left(1-\frac{\boldsymbol{\nu}_i(\mathbf{c}_\theta,t)}{\boldsymbol{\nu}_i(\mathbf{c}_\theta,s)}\right)}\right) \tag{49}$$
$$\tag{50}$$

Let $p_i=\frac{{\boldsymbol{\sigma}_i}^2(\mathbf{c},s)\left(1-\frac{\boldsymbol{\nu}_i(\mathbf{c},t)}{\boldsymbol{\nu}_i(\mathbf{c},s)}\right)}{{\boldsymbol{\sigma}_i}^2(\mathbf{c}_\theta,s)\left(1-\frac{\boldsymbol{\nu}_i(\mathbf{c}_\theta,t)}{\boldsymbol{\nu}_i(\mathbf{c}_\theta,s)}\right)}$

The sequence $\lim_{T\to\infty}\frac{T}{2}\sum_{i=1}^{d}(p_i-1-\log p_i)$ converges iff $\lim_{T\to\infty}\sum_{i=1}^{d}(p_i-1-\log p_i)=0$. Notice that the function $f(x)=x-1-\log x\geq 0\ \ \forall x\in\mathbb{R}$ and the equality holds for $x=1$. Thus, the condition $\lim_{T\to\infty}\frac{T}{2}\sum_{i=1}^{d}(p_i-1-\log p_i)$ holds iff $\lim_{T\to\infty}p_i=0\ \ \forall i\in\{1,\ldots,d\}$. Thus,

$$\lim_{T\to\infty} p_i = 1$$

$$\implies \lim_{T\to\infty} \left(\frac{{\boldsymbol{\sigma}_i}^2(\mathbf{c}, s)\left(1 - \frac{\boldsymbol{\nu}_i(\mathbf{c},t)}{\boldsymbol{\nu}_i(\mathbf{c},s)}\right)}{{\boldsymbol{\sigma}_i}^2(\mathbf{c}_\theta, s)\left(1 - \frac{\boldsymbol{\nu}_i(\mathbf{c}_\theta,t)}{\boldsymbol{\nu}_i(\mathbf{c}_\theta,s)}\right)} \right) = 1$$

Substituting 1/T as δ,

$$\implies \lim_{\delta\to 0^+} \left(\frac{{\boldsymbol{\sigma}_i}^2(\mathbf{c}, t-\delta)\left(1 - \frac{\boldsymbol{\nu}_i(\mathbf{c},t)}{\boldsymbol{\nu}_i(\mathbf{c},t-\delta)}\right)}{{\boldsymbol{\sigma}_i}^2(\mathbf{c}_\theta, t-\delta)\left(1 - \frac{\boldsymbol{\nu}_i(\mathbf{c}_\theta,t)}{\boldsymbol{\nu}_i(\mathbf{c}_\theta,t-\delta)}\right)} \right) = 1$$

$$\implies \frac{{\boldsymbol{\sigma}_i}^2(\mathbf{c}, t)}{{\boldsymbol{\sigma}_i}^2(\mathbf{c}_\theta, t)} \lim_{\delta\to 0^+} \left(\frac{1 - \frac{\boldsymbol{\nu}_i(\mathbf{c},t)}{\boldsymbol{\nu}_i(\mathbf{c},t-\delta)})}{1 - \frac{\boldsymbol{\nu}_i(\mathbf{c}_\theta,t)}{\boldsymbol{\nu}_i(\mathbf{c}_\theta,t-\delta)}} \right) = 1$$

Applying L'Hospital rule,

$$\implies \frac{{\boldsymbol{\sigma}_i}^2(\mathbf{c}, t)}{{\boldsymbol{\sigma}_i}^2(\mathbf{c}_\theta, t)} \left(\frac{\frac{-{\boldsymbol{\nu}_i}'(\mathbf{c},t)}{\boldsymbol{\nu}_i(\mathbf{c},t)})}{\frac{-{\boldsymbol{\nu}_i}'(\mathbf{c}_\theta,t)}{\boldsymbol{\nu}_i(\mathbf{c}_\theta,t)}} \right) = 1$$

$$\implies \frac{{\boldsymbol{\sigma}_i}^2(\mathbf{c}, t)}{{\boldsymbol{\sigma}_i}^2(\mathbf{c}_\theta, t)} \left(\frac{{\boldsymbol{\nu}_i}'(\mathbf{c}, t)\boldsymbol{\nu}_i(\mathbf{c}_\theta, t)}{\boldsymbol{\nu}_i(\mathbf{c}, t){\boldsymbol{\nu}_i}'(\mathbf{c}_\theta, t))} \right) = 1 \tag{51}$$

In the vector form the above equation can be written as,

$$\frac{\boldsymbol{\sigma}_t^2(\mathbf{c})\boldsymbol{\nu}_t(\mathbf{c}_\theta)\nabla_t\boldsymbol{\nu}(\mathbf{c}, t)}{\boldsymbol{\sigma}_t^2(\mathbf{c}_\theta)\boldsymbol{\nu}_t(\mathbf{c})\nabla_t\boldsymbol{\nu}(\mathbf{c}_\theta, t)} \to \mathbf{1_d} \tag{52}$$

Eq. 52 holds if:

- $x_\theta = x_0$ i.e. the unet can perfectly map $\mathbf{x}_t$ to $\mathbf{x}_0$ $\forall t \in [0, 1]$ which is unrealistic.
- Clever parameterizations for $\boldsymbol{\sigma}, \boldsymbol{\alpha}, \boldsymbol{\nu}$ that ensure Eq. 52 holds.

Because of aforementioned challenges we evaluate this method with finite $T = 1000$. We demonstrate the performance of the model empirically in Fig. 1.

C.2.2 Recovering VDM

If we substitute $\boldsymbol{\nu}_t(\mathbf{c}), \boldsymbol{\nu}_t(\mathbf{c}_\theta)$ with $\boldsymbol{\nu}(t)$ (since the SNR isn't conditioned on the context $\mathbf{c}$), $\boldsymbol{\sigma}_t(\mathbf{c}_\theta), \boldsymbol{\sigma}_t(\mathbf{c})$ with σ_t and $\boldsymbol{\alpha}_t(\mathbf{c}_\theta), \boldsymbol{\alpha}_t(\mathbf{c})$ with $\boldsymbol{\alpha}_t$, Eq. 39 reduces to the intermediate loss in VDM i.e. $\frac{1}{2}(\mathbf{x}_\theta - \mathbf{x}_0)^\top(\nabla_t\boldsymbol{\nu}(t))\ (\mathbf{x}_\theta - \mathbf{x}_0)$ and Eq. 49 reduces to 0.

C.3 Challenges in Conditioning on Context

Note that the model $p_\theta(\mathbf{x}_{0:1}|\mathbf{c})$ implicitly assumes the availability of $\mathbf{c}$ at generation time. Sometimes, this context may be available, such as when we condition on a label. We may then fit a conditional diffusion process with a standard diffusion objective $\mathbb{E}_{\mathbf{x}_0,c}[\text{ELBO}(\mathbf{x}_0, p_\theta(\mathbf{x}_{0:1}|\mathbf{c}), q_\phi(\mathbf{x}_{0:1}|\mathbf{c})]$, in which both the forward and the backward processes are conditioned on $\mathbf{c}$ (see Sec. 2.3).

When $\mathbf{c}$ is not known at generation time, we may fit a model p_θ that does not condition on $\mathbf{c}$. Unfortunately, this also forces us to define $p_\theta(\mathbf{x}_s|\mathbf{x}_t) = \mathcal{N}(\boldsymbol{\mu}_p(\mathbf{x}_t, t), \boldsymbol{\Sigma}_p(\mathbf{x}_t, t))$ where $\boldsymbol{\mu}_p(\mathbf{x}_t, t), \boldsymbol{\Sigma}_p(\mathbf{x}_t, t)$ is parameterized directly by a neural network. We can no longer use a noise parameterization $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) = (\mathbf{x}_t - \boldsymbol{\alpha}_t(\mathbf{c})\mathbf{x}_\theta(\mathbf{x}_t, t, \mathbf{c}))/\boldsymbol{\sigma}_t(\mathbf{c})$ because it requires us to compute $\boldsymbol{\alpha}_t(\mathbf{c})$ and $\boldsymbol{\sigma}_t(\mathbf{c})$, which we do not know. Since noise parameterization plays a key role in the sample quality of diffusion models (Ho et al., 2020), this approach limits performance.

The other approach is to approximate $\mathbf{c}$ using a neural network, $\mathbf{c}_\theta(\mathbf{x}_t, t)$. This would allow us to write $p_\theta(\mathbf{x}_s|\mathbf{x}_t) = q_\phi(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0 = \mathbf{x}_\theta(\mathbf{x}_t, t), \mathbf{c} = \mathbf{c}_\theta(\mathbf{x}_t, t))$. Unfortunately, this introduces instability in the learning objective, which we observe both theoretically and empirically. Specifically,

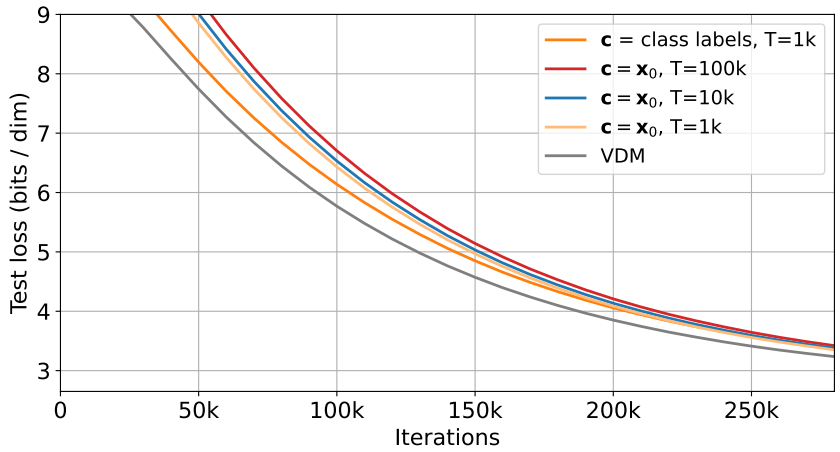


Figure 1: For $\mathbf{c} =$ "class labels" or $\mathbf{c} = \mathbf{x}_0$ the likelihood estimates are worse than VDM. For $\mathbf{c} = \mathbf{x}_0$ we see that the VLB degrades with increasing T whereas for VDM and MuLAN it improves for increasing T; see Kingma et al. (2021). This empirical observation is consistent with our mathematical insights earlier. As these models consistently exhibit inferior performance w.r.t VDM, in line with our initial conjectures, we refrain from training them beyond 300k iterations due to the substantial computational cost involved.

in Suppl. C we show that the learning objective diverges unless the following condition holds true: $\lim_{T\to\infty} T\frac{\boldsymbol{\sigma}_t^2(\mathbf{x}_0)\boldsymbol{\nu}_t(\mathbf{x}_0)\boldsymbol{\nu}'_t(\mathbf{x}_\theta)}{\boldsymbol{\sigma}_t^2(\mathbf{x}_\theta)\boldsymbol{\nu}_t(\mathbf{x}_\theta)\boldsymbol{\nu}'_t(\mathbf{x}_0)} \to \mathbf{I}_d$ pointwise across t. Experiments in Suppl. C.4 confirm this issue.

C.4 Experimental results

In Fig. 1 we demonstrate that the multivariate diffusion processes where $\mathbf{c} =$ "class labels" or $\mathbf{c} = \mathbf{x}_0$ perform worse than VDM. Since a continuous time formulation i.e. $T \to \infty$ for the case when $\mathbf{c} = \mathbf{x}_0$ isn't possible (unlike MuLAN or VDM) we evaluate these models in the discrete time setting where we use $T = 1000$. Furthermore we also ablate $T = 10k, 100k$ for $\mathbf{c} = \mathbf{x}_0$ to show that the VLB degrades with increasing T whereas for VDM and MuLAN it improves for increasing T; see Kingma et al. (2021). This empirical observation is consistent with our mathematical insights earlier. As these models consistently exhibit inferior performance w.r.t VDM, in line with our initial conjectures, we refrain from training them beyond 300k iterations due to the substantial computational cost involved.

D MuLAN: MUltivariate Latent Auxiliary variable Noise Schedule

D.1 Parameterization in the reverse process

D.1.1 Noise parameterization

Since the forward pass is given by $\mathbf{x}_t = \boldsymbol{\alpha}_t(\mathbf{z})\mathbf{x}_0 + \boldsymbol{\sigma}_t(\mathbf{z})\boldsymbol{\epsilon}_t$, we can write the noise $\boldsymbol{\epsilon}_t$ in terms of $\mathbf{x}_0, \mathbf{x}_t$ in the following manner:

$$\boldsymbol{\epsilon}_t = \frac{\mathbf{x}_t - \boldsymbol{\alpha}_t(\mathbf{z})\mathbf{x}_0}{\boldsymbol{\sigma}_t(\mathbf{z})} \tag{53}$$

Following Dhariwal & Nichol (2021); Kingma et al. (2021), instead of parameterizing $\mathbf{x}_\theta(\mathbf{x}_t, \mathbf{z}, t)$ using a neural network, we use Eq. 53 to parameterize the denoising model in terms of a noise prediction model $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)$,

$$\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t) = \frac{\mathbf{x}_t - \boldsymbol{\alpha}_t(\mathbf{z})\mathbf{x}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\boldsymbol{\sigma}_t(\mathbf{z})} \tag{54}$$

Table 2: Likelihood in bits per dimension (BPD) (mean and 95% confidence interval), on the test set of CIFAR-10 computed using VLB estimate.

parameterization	Num training steps	CIFAR-10 (↓)
Noise parameterization	10M	2.60 ± 10^{-3}
Velocity parameterization	8M	2.59 ± 10^{-3}

D.1.2 Velocity parameterization

Following Salimans & Ho (2022); Zheng et al. (2023), we explore another parameterization of the denoising network which is given by

$$\mathbf{v}_\theta(\mathbf{x}_t, \mathbf{z}, t) = \frac{\boldsymbol{\alpha}_t(\mathbf{z})\mathbf{x}_t - \mathbf{x}_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\boldsymbol{\sigma}_t(\mathbf{z})} \tag{55}$$

In practice, Velocity parameterization leads to a better performance than noise parameterization; as illustrated in Table 2.

D.2 Polynomial Noise Schedule

Let $f(x;\psi)$ be a scalar-valued polynomial of degree n with coefficients $\psi \in \mathbb{R}^{n+1}$ expressed as:

$$f(x;\psi) = \psi_n x^n + \psi_{n-1}x^{n-1} + \cdots + \psi_1 x + \psi_0,$$

and denote its derivative with respect to x as $\frac{d}{dx}f(x;\psi)$, represented by $f'(x;\psi)$. Now we'd like to find least n such that $f(x;\psi)$ satisfies the following properties:

1. $f(x;\psi)$ is monotonically increasing, i.e. $f'(x;\psi) \geq 0\ \forall x \in \mathbb{R}, \psi \in \mathbb{R}^{n+1}$.
2. $f'(x_1;\psi) = 0, f'(x_2;\psi) = 0\ \exists x_1, x_2 \in \mathbb{C}, x_1 \neq x_2, \forall \psi \in \mathbb{R}^{n+1}$.

For the **first** condition to hold, we can design $f'(x;\psi)$ such that it's a perfect square with real / imaginary roots. That way $f'(x;\psi) \geq 0\ \forall x \in \mathbb{R}, \psi \in \mathbb{R}^{n+1}$. This means that $f'(x;\psi)$ is an even degree polynomial, i.e. the degree of $f'(x;\psi)$ can take the following values: $2, 4, \ldots$. Also, note that at least half of the roots of $f'(x;\psi)$ are repeated since $f'(x;\psi)$ can be expressed as a perfect square, i.e., if $f'(x;\psi)$ has a degree 2 then it has exactly 1 unique root (real / imaginary), if $f'(x;\psi)$ has a degree 4 then it has at most 2 unique roots (real / imaginary), and so on.

For the **second** condition to hold, $f'(x;\psi)$ needs to have at least 2 unique roots $\exists \psi \in \mathbb{R}^{n+1}$. For this reason $f'(x;\psi)$ is a polynomial of degree 4. Thus, $f'(x;\psi)$ can be written as $f'(x;\psi) = (\psi_3 x^2 + \psi_2 x + \psi_1)^2$. This ensures that $\exists \psi \in \mathbb{R}^5$ s.t. $f'(x;\psi) = 0$ twice in $x \in \mathbb{R}$, and $f'(x;\psi) \geq 0\ \forall \psi \in \mathbb{R}^5$.

Thus, $f(x;\psi)$ takes the following functional form:

$$\begin{aligned} f(x;\psi) &= \int (\psi_3 x^2 + \psi_2 x + \psi_1)^2 dx \\ &= \frac{\psi_3^2}{5}x^5 + \frac{\psi_3\psi_2}{2}x^4 + \frac{\psi_2^2 + 2\psi_3\psi_1}{3}x^3 + \psi_2\psi_1 x^2 + \psi_1^2 x + \text{constant}. \end{aligned} \tag{56}$$

For the above-mentioned reasons we express $\boldsymbol{\gamma}(\mathbf{c}, t) : \mathbb{R}^m \times [0,1] \to \mathbb{R}^d$ as a degree 5 polynomial in t. We define neural networks $\mathbf{a}_\phi(\mathbf{c}) : \mathbb{R}^m \to \mathbb{R}^d$, $\mathbf{b}_\phi(\mathbf{c}) : \mathbb{R}^m \to \mathbb{R}^d$, and $\mathbf{d}_\phi(\mathbf{c}) : \mathbb{R}^m \to \mathbb{R}^d$ with parameters ϕ. Let $f_\phi : \mathbb{R}^m \times [0,1] \to \mathbb{R}^d$ be defined as:

$$f_\phi(\mathbf{c}, t) = \frac{\mathbf{a}_\phi^2(\mathbf{c})}{5}t^5 + \frac{\mathbf{a}_\phi(\mathbf{c})\mathbf{b}_\phi(\mathbf{c})}{2}t^4 + \frac{\mathbf{b}_\phi^2(\mathbf{c}) + 2\mathbf{a}_\phi(\mathbf{c})\mathbf{d}_\phi(\mathbf{c})}{3}t^3 + \mathbf{b}_\phi(\mathbf{c})\mathbf{d}_\phi(\mathbf{c})t^2 + \mathbf{d}_\phi^2(\mathbf{c})t$$

where the multiplication and division operations are elementwise. The the noise schedule, $\boldsymbol{\gamma}(\mathbf{c}, t)$, is given as follows:

$$\boldsymbol{\gamma}_\phi(\mathbf{c}, t) = \gamma_{\text{min}} + (\gamma_{\text{max}} - \gamma_{\text{min}})\frac{f_\phi(\mathbf{c}, t)}{f_\phi(\mathbf{c}, t=1)} \tag{57}$$

Notice that $\boldsymbol{\gamma}_\phi(\mathbf{c}, t)$ has these interesting properties:

- Is increasing in $t \in [0, 1]$ which is crucial as mentioned in Sec. D.4.

- $\boldsymbol{\gamma}_\phi(\mathbf{c}, t)$ has end points at $t = 0$ and $t = 1$ which the user can specify via γ_{min} and γ_{max}. Specificaly, $\boldsymbol{\gamma}_\phi(\mathbf{c}, t = 0) = \gamma_{\text{min}}$ and $\boldsymbol{\gamma}_\phi(\mathbf{c}, t = 1) = \gamma_{\text{max}}$.

- Its time-derivative i.e. $\nabla_t \boldsymbol{\gamma}_\phi(\mathbf{c}, t)$ **can** be zero twice in $t \in [0, 1]$. This isn't a necessary condition but it's nice to have a flexible noise schedule whose time-derivative can be 0 at the beginning and the end of the diffusion process.

D.3 Variational Lower Bound

In this section we derive the VLB. For ease of reading we use the notation $\mathbf{x}_t$ to denote $\mathbf{x}_{t(i)}$ and $\mathbf{x}_{t-1}$ to denote $\mathbf{x}_{t(i-1)} \equiv \mathbf{x}_{s(i)}$ in the following derivation.

$$
\begin{aligned}
&- \log p_\theta(\mathbf{x}_0) \\
&\leq \mathbb{E}_{q_\phi}\left[-\log \frac{p_\theta(\mathbf{z}, \mathbf{x}_{0:T})}{q_\phi(\mathbf{z}, \mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_{q_\phi}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T-1}|\mathbf{z}, \mathbf{x}_T)}{q_\phi(\mathbf{z}, \mathbf{x}_{1:T}|\mathbf{x}_0)} - \log p_\theta(\mathbf{x}_T) - \log p_\theta(\mathbf{z})\right] \\
&= \mathbb{E}_{q_\phi}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T-1}|\mathbf{z}, \mathbf{x}_T)}{q_\phi(\mathbf{x}_{1:T}|\mathbf{z}, \mathbf{x}_0)} - \log \frac{1}{q_\phi(\mathbf{z}|\mathbf{x}_0)} - \log p_\theta(\mathbf{x}_T) - \log p_\theta(\mathbf{z})\right] \\
&= \mathbb{E}_{q_\phi}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T-1}|\mathbf{z}, \mathbf{x}_T)}{q_\phi(\mathbf{x}_{1:T}|\mathbf{z}, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_T) - \log \frac{p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_{q_\phi}\left[-\sum_{t=1}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{z}, \mathbf{x}_t)}{q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_T) - \log \frac{p_\theta(z)}{q_\phi(z|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_{q_\phi}\left[-\log \frac{p_\theta(\mathbf{x}_0|\mathbf{z}, \mathbf{x}_1)}{q_\phi(\mathbf{x}_1|\mathbf{x}_0, \mathbf{z})} - \sum_{t=2}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{z}, \mathbf{x}_t)}{q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_T) - \log \frac{p_\theta(z)}{q_\phi(\mathbf{z}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_{q_\phi}\left[-\log \frac{p_\theta(\mathbf{x}_0|\mathbf{z}, \mathbf{x}_1)}{q_\phi(\mathbf{x}_1|\mathbf{x}_0, \mathbf{z})} - \sum_{t=2}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{z}, \mathbf{x}_t) q_\phi(\mathbf{x}_{t-1}|\mathbf{z}, \mathbf{x}_0)}{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{z}, \mathbf{x}_0) q_\phi(\mathbf{x}_t|\mathbf{z}, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_T) - \log \frac{p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_{q_\phi}\left[-\log \frac{p_\theta(\mathbf{x}_0|\mathbf{z}, \mathbf{x}_1)}{q_\phi(\mathbf{x}_1|\mathbf{x}_0, \mathbf{z})} - \sum_{t=2}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{z}, \mathbf{x}_t)}{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{z}, \mathbf{x}_0)} - \sum_{t=2}^{T} \log \frac{q_\phi(\mathbf{x}_{t-1}|\mathbf{z}, \mathbf{x}_0)}{q_\phi(\mathbf{x}_t|\mathbf{z}, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_T) - \log \frac{p_\theta(z)}{q_\phi(\mathbf{z}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_{q_\phi}\left[-\log \frac{p_\theta(\mathbf{x}_0|\mathbf{z}, \mathbf{x}_1)}{q_\phi(\mathbf{x}_1|\mathbf{x}_0, \mathbf{z})} - \sum_{t=2}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{z}, \mathbf{x}_t)}{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{z}, \mathbf{x}_0)} - \log \frac{q(\mathbf{x}_1|\mathbf{z}, x_0)}{q_\phi(\mathbf{x}_T|\mathbf{z}, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_T) - \log \frac{p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_{q_\phi}\left[-\log p_\theta(\mathbf{x}_0|\mathbf{z}, \mathbf{x}_1) - \sum_{t=2}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{z}, \mathbf{x}_t)}{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{z}, \mathbf{x}_0)} - \log \frac{1}{q_\phi(\mathbf{x}_T|\mathbf{z}, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_T) - \log \frac{p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_{q_\phi}\left[-\log p_\theta(\mathbf{x}_0|\mathbf{z}, \mathbf{x}_1) - \sum_{t=2}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1}|z, \mathbf{x}_t)}{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{z}, \mathbf{x}_0)} - \log \frac{p_\theta(\mathbf{x}_T)}{q_\phi(\mathbf{x}_T|\mathbf{z}, \mathbf{x}_0)} - \log \frac{p_\theta(\mathbf{z})}{q_\phi(z|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_{q_\phi}\left[\underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{z}, \mathbf{x}_1)}_{\mathcal{L}_{\text{recons}}} + \underbrace{\sum_{t=2}^{T} \mathrm{D}_{\mathrm{KL}}[p_\theta(\mathbf{x}_{t-1}|\mathbf{z}, \mathbf{x}_t) \| q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{z}, \mathbf{x}_0)]}_{\mathcal{L}_{\text{diffusion}}}\right] \\
&\quad + \mathbb{E}_{q_\phi}\left[\underbrace{\mathrm{D}_{\mathrm{KL}}[p_\theta(\mathbf{x}_T) \| q_\phi(\mathbf{x}_T|\mathbf{z}, \mathbf{x}_0)]}_{\mathcal{L}_{\text{prior}}} + \underbrace{\mathrm{D}_{\mathrm{KL}}[p_\theta(\mathbf{z}) \| q(\mathbf{z}|\mathbf{x}_0)]}_{\mathcal{L}_{\text{latent}}}\right]
\end{aligned}
\tag{58}
$$

Switching back to the notation used throughout the paper, the VLB is given as:

$$
\begin{aligned}
&-\log p_\theta(\mathbf{x}_0) \\
&\leq \mathbb{E}_{q_\phi}\left[\underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{z},\mathbf{x}_1)}_{\mathcal{L}_{\text{recons}}} + \underbrace{\sum_{i=2}^{T} \mathrm{D}_{\mathrm{KL}}[p_\theta(\mathbf{x}_{s(i)}|\mathbf{z},\mathbf{x}_{t(i)})\|q_\phi(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)},\mathbf{z},\mathbf{x}_0)]}_{\mathcal{L}_{\text{diffusion}}}\right] \\
&+ \mathbb{E}_{q_\phi}\left[\underbrace{\mathrm{D}_{\mathrm{KL}}[p_\theta(\mathbf{x}_1)\|q_\phi(\mathbf{x}_1|\mathbf{z},\mathbf{x}_0)]}_{\mathcal{L}_{\text{prior}}} + \underbrace{\mathrm{D}_{\mathrm{KL}}[p_\theta(\mathbf{z})\|q_\phi(\mathbf{z}|\mathbf{x}_0)]}_{\mathcal{L}_{\text{latent}}}\right]
\end{aligned} \tag{59}
$$

Next, we derive a precise formula for the learning objective (6) of the auxiliary-variable diffusion model. Using the objective of a diffusion model in (1) we can write (6) as the sum of four terms:

$$
\log p_\theta(\mathbf{x}_0) \geq \mathbb{E}_{q_\phi}[\mathcal{L}_{\text{recons}} + \mathcal{L}_{\text{diffusion}} + \mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{latent}}], \tag{60}
$$

The reconstruction loss, $\mathcal{L}_{\text{recons}}$, can be (stochastically and differentiably) estimated using standard techniques; see (Kingma & Welling, 2013), $\mathcal{L}_{\text{prior}} = -\mathrm{D}_{\mathrm{KL}}[q_\phi(\mathbf{x}_1|\mathbf{x}_0,\mathbf{z})\|p_\theta(\mathbf{x}_1)]$ is the diffusion prior term, $\mathcal{L}_{\text{latent}} = -\mathrm{D}_{\mathrm{KL}}[q_\phi(\mathbf{z}|\mathbf{x}_0)\|p_\theta(\mathbf{z})]$ is the latent prior term, and $\mathcal{L}_{\text{diffusion}}$ is the diffusion loss term, which we examine below. The complete derivation is given in Suppl. D.3.

D.3.1 Diffusion Loss

Discrete-Time Diffusion. We start by defining p_θ in discrete time, and as in Sec. 1, we let $T > 0$ be the number of total time steps and define $t(i) = i/T$ and $s(i) = (i-1)/T$ as indexing variables over the time steps. We also use $\mathbf{x}_{0:1}$ to denote the subset of variables associated with these timesteps. Starting with the expression in Eq. 1 and following the steps in Suppl. D, we can write $\mathcal{L}_{\text{diffusion}}$ as:

$$
\begin{aligned}
\mathcal{L}_{\text{diffusion}} &= -\sum_{i=2}^{T} \mathrm{D}_{\mathrm{KL}}[q_\phi(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)},\mathbf{x}_0,\mathbf{z})\|p_\theta(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)},\mathbf{z})] \\
&= \frac{1}{2}\sum_{i=2}^{T} \left[(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,\mathbf{z},t(i)))^\top \mathrm{diag}\left(\boldsymbol{\gamma}(\mathbf{z},s(i)) - \boldsymbol{\gamma}(\mathbf{z},t(i))\right)(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,\mathbf{z},t(i)))\right]
\end{aligned} \tag{61}
$$

Continuous-Time Diffusion. We can also consider the limit of the above objective as we take an infinitesimally small partition of $t \in [0,1]$, which corresponds to the limit when $T \to \infty$. In Suppl. D we show that taking this limit of Eq. 61 yields the continuous-time diffusion loss:

$$
\mathcal{L}_{\text{diffusion}} = -\frac{1}{2}\mathbb{E}_{t\sim[0,1]}\left[(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,\mathbf{z},t))^\top \mathrm{diag}\left(\nabla_t\boldsymbol{\gamma}(\mathbf{z},t)\right)(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,\mathbf{z},t))\right] \tag{62}
$$

where $\nabla_t\boldsymbol{\gamma}(\mathbf{z},t) \in \mathbb{R}^d$ denotes the Jacobian of $\boldsymbol{\gamma}(\mathbf{z},t)$ with respect to the scalar t. We observe that the limit of $T \to \infty$ yields improved performance, matching the existing theoretical argument by Kingma et al. (2021).

D.3.2 Auxiliary latent loss

We try two different kinds of priors for $p_\theta(\mathbf{z})$: discrete ($\mathbf{z} \in \{0,1\}^m$) and continuous ($\mathbf{z} \in \mathbb{R}^m$).

Continuous Auxiliary Latents. In the case where $\mathbf{z}$ is continuous, we select $p_\theta(\mathbf{z})$ as $\mathcal{N}(\mathbf{0},\mathbf{I}_m)$. This leads to the following KL loss term:
$\mathrm{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}_0)\|p_\theta(\mathbf{z})) = \frac{1}{2}(\boldsymbol{\mu}^\top(\mathbf{x}_0)\boldsymbol{\mu}(\mathbf{x}_0)) + \mathrm{tr}(\boldsymbol{\Sigma}^2(\mathbf{x}_0) - \mathbf{I}_m) - \log|\boldsymbol{\Sigma}^2(\mathbf{x}_0)|)$.

Discrete Auxiliary Latents. In the case where $\mathbf{z}$ is discrete, we select $p_\theta(\mathbf{z})$ as a uniform distribution. Let $\mathbf{z} \in \{0,1\}^m$ be a k-hot vector sampled from a discrete Exponential Family distribution $p_\theta(\mathbf{z};\theta)$ with logits θ. Niepert et al. (2021) show that $\mathbf{z} \sim p_\theta(\mathbf{z};\theta)$ is equivalent to $\mathbf{z} = \arg\max_{y\in Y}\langle\theta + \epsilon_g, y\rangle$ where ϵ_g denotes the sum of gamma distribution Suppl. E, Y denotes the set of all k-hot vectors of some fixed length m. For $k > 1$, To differentiate through the $\arg\max$ we use a relaxed estimator, Identity, as proposed by Sahoo et al. (2023). This leads to the following KL loss term: $\mathrm{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}_0)\|p_\theta(\mathbf{z})) = -\sum_{i=1}^{m} q_\phi(\mathbf{z}|\mathbf{x}_0)_i(\log q_\phi(\mathbf{z}|\mathbf{x}_0)_i + \log m)$.

D.4 The Variational Lower Bound as a Line Integral Over The Noise Schedule

Having defined our loss, we now return to the question of whether it is invariant to the choice of diffusion process. Notice that we may rewrite Eq. 62 in the following vectorized form:

$$\mathcal{L}_{\text{diffusion}} = -\frac{1}{2}\int_0^1 (\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, \mathbf{z}, t))^2 \cdot \nabla_t \boldsymbol{\nu}(\mathbf{z}, t)\mathrm{d}t \tag{63}$$

where the square is applied elementwise. We seek to rewrite (63) as a line integral $\int_a^b \mathbf{f}(\mathbf{r}(t)) \cdot \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{r}(t)\mathrm{d}t$ for some vector field $\mathbf{f}$ and trajectory $\mathbf{r}(t)$. Recall that $\boldsymbol{\nu}(\mathbf{z}, t)$ is monotonically decreasing in each coordinate as a function of t; hence, it is invertible on its image, and we can write $t = \boldsymbol{\nu}_{\mathbf{z}}^{-1}(\boldsymbol{\nu}(\mathbf{z}, t))$ for some $\boldsymbol{\nu}_z^{-1}$. Let $\bar{\mathbf{x}}_\theta(\mathbf{x}_{\boldsymbol{\nu}(\mathbf{z},t)}, \mathbf{z}, \boldsymbol{\nu}(\mathbf{z}, t)) = \mathbf{x}_\theta(\mathbf{x}_{\boldsymbol{\nu}_z^{-1}(\boldsymbol{\nu}(\mathbf{z},t))}, \mathbf{z}, \boldsymbol{\nu}_{\mathbf{z}}^{-1}(\boldsymbol{\nu}(\mathbf{z}, t)))$ and note that for all t, we can write $\mathbf{x}_t$ as $\mathbf{x}_{\boldsymbol{\nu}(\mathbf{z},t)}$; see Eq. 24, and have $\bar{\mathbf{x}}_\theta(\mathbf{x}_{\boldsymbol{\nu}(\mathbf{z},t)}, \mathbf{z}, \boldsymbol{\nu}(\mathbf{z}, t)) = \mathbf{x}_\theta(\mathbf{x}_t, \mathbf{z}, t)$. We can then write the integral in (63) as $\int_0^1 (\mathbf{x}_0 - \bar{\mathbf{x}}_\theta(\mathbf{x}_{\boldsymbol{\nu}(\mathbf{z},t)}, \mathbf{z}, \boldsymbol{\nu}(\mathbf{z}, t)))^2 \cdot \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\nu}(\mathbf{z}, t)\rangle \mathrm{d}t$, which is a line integral with $\mathbf{f}(\mathbf{r}(t)) \equiv (\mathbf{x}_0 - \bar{\mathbf{x}}_\theta(\mathbf{x}_{\boldsymbol{\nu}(\mathbf{z},t)}, \mathbf{z}, \boldsymbol{\nu}(\mathbf{z}, t)))^2$ and $\mathbf{r}(t) \equiv \boldsymbol{\nu}(\mathbf{z}, t)$ and .

Thus the diffusion loss, $\mathcal{L}_{\text{diffusion}}$, can be interpreted as a measure of work done along the trajectory $\boldsymbol{\nu}(\mathbf{z}, t)$ in the presence of a vector field $\mathbf{f}$. Different "trajectories" yield different results for most integrands, unless its integral corresponds to a conservative force field, which is rarely the case for a diffusion process (Spinney & Ford, 2012). We empirically observe this in our experiments where swapping out different multivariate $\boldsymbol{\nu}$ yields different values of the ELBO. In Sec. D.6, we show that variational diffusion models can be viewed as following only linear trajectories $\boldsymbol{\nu}(t)$, hence their objective is invariant to the noise schedule. Our method learns a multivariate $\boldsymbol{\nu}$ that yields paths corresponding to a better ELBO.

D.5 Diffusion Loss

To derive the diffusion loss, $\mathcal{L}_{\text{diffusion}}$ in Eq. 60, we first derive an expression for $\mathrm{D}_{\mathrm{KL}}(q_\phi(\mathbf{x}_s|\mathbf{z}, \mathbf{x}_t, \mathbf{x}_0)\|p_\theta(\mathbf{x}_s|\mathbf{z}, \mathbf{x}_t))$ using Eq. 2 and Eq. 4 in the following manner (details in Suppl. D):

$$\begin{aligned}
&\mathrm{D}_{\mathrm{KL}}(q_\phi(\mathbf{x}_s|\mathbf{z}, \mathbf{x}_t, \mathbf{x}_0)\|p_\theta(\mathbf{x}_s|\mathbf{z}, \mathbf{x}_t)) \\
&= \frac{1}{2}\left((\boldsymbol{\mu}_{q_\phi} - \boldsymbol{\mu}_p)^\top \boldsymbol{\Sigma}_\theta^{-1}(\boldsymbol{\mu}_{q_\phi} - \boldsymbol{\mu}_p) + \mathrm{tr}\left(\boldsymbol{\Sigma}_{q_\phi}\boldsymbol{\Sigma}_p^{-1} - \mathbf{I}_n\right) - \log\frac{|\boldsymbol{\Sigma}_{q_\phi}|}{|\boldsymbol{\Sigma}_p|}\right) \\
&= \frac{1}{2}\left((\mathbf{x}_0 - \mathbf{x}_\theta)^\top \mathrm{diag}(\boldsymbol{\nu}(\mathbf{z}, s) - \boldsymbol{\nu}(\mathbf{z}, t))(\mathbf{x}_0 - \mathbf{x}_\theta)\right)
\end{aligned} \tag{64}$$

Let $\lim_{T\to\infty} T(\boldsymbol{\nu}_s(z) - \boldsymbol{\nu}_t(z)) = -\nabla_t \boldsymbol{\nu}(\mathbf{z}, t)$ be the partial derivative of the vector $\boldsymbol{\nu}(\mathbf{z}, t)$ w.r.t scalar t. Then we derive the diffusion loss, $\mathcal{L}_{\text{diffusion}}$, for the continuous case in the following manner

(for brevity we use the notation s for $s(i) = (i-1)/T$ and t for $t(i) = i/T$):

$$
\begin{aligned}
&\mathcal{L}_{\text{diffusion}} \\
&= \lim_{T\to\infty} \frac{1}{2}\sum_{i=2}^{T} \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n)} \mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_s|\mathbf{x}_t,\mathbf{x}_0,\mathbf{z})\|p_\theta(\mathbf{x}_s|\mathbf{x}_t,\mathbf{z})) \\
&\quad \text{Using Eq. 64 we get,} \\
&= \lim_{T\to\infty} \frac{1}{2}\sum_{i=2}^{T} \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n)} (\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t(i)))^\top \mathrm{diag}\left(\boldsymbol{\nu}(s(i),\mathbf{z}) - \boldsymbol{\nu}(t(i),\mathbf{z})\right)(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t(i))) \\
&= \frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}_n)}\left[\lim_{T\to\infty}\sum_{i=2}^{T} T(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t(i)))^\top \mathrm{diag}\left(\boldsymbol{\nu}(s(i),\mathbf{z}) - \boldsymbol{\nu}(t(i),\mathbf{z})\right)(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t(i)))\frac{1}{T}\right] \\
&\quad \text{Using the fact that } \lim_{T\to\infty} T\left(\boldsymbol{\nu}(s,\mathbf{z}) - \boldsymbol{\nu}(\mathbf{z},t)\right) = -\nabla_t\boldsymbol{\nu}(t,\mathbf{z}) \text{ we get,} \\
&= -\frac{1}{2}\mathbb{E}_{t\sim\{0,\ldots,1\}}\left[(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t,t))^\top\left(\nabla_t\boldsymbol{\nu}_t(z)\right)(\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t,t))\right] \\
&\quad \text{Substituting } \mathbf{x}_0 = \boldsymbol{\alpha}_t^{-1}(\mathbf{z})(\mathbf{x}_t - \boldsymbol{\sigma}_t(\mathbf{z})\boldsymbol{\epsilon}_t) \text{ from Eq. 53 and} \\
&\quad \text{Substituting } \mathbf{x}_\theta(\mathbf{x}_t,\mathbf{z},t) = \boldsymbol{\alpha}_t^{-1}(\mathbf{z})(\mathbf{x}_t - \boldsymbol{\sigma}_t(\mathbf{z})\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)) \text{ from Eq. 54 we get,} \\
&= -\frac{1}{2}\mathbb{E}_{t\sim[0,1]}\left[(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t))^\top\left(\frac{\boldsymbol{\sigma}_t^2(\mathbf{z})}{\boldsymbol{\alpha}_t^2(\mathbf{z})}\times\nabla_t\boldsymbol{\nu}_t(\mathbf{z})\right)(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t))\right] \\
&\quad \text{Let } \boldsymbol{\nu}^{-1}(\mathbf{z},t) \text{ denote the reciprocal of the values in the vector } \boldsymbol{\nu}(\mathbf{z},t). \\
&= -\frac{1}{2}\mathbb{E}_{t\sim[0,1]}\left[(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t))^\top\mathrm{diag}\left(\boldsymbol{\nu}^{-1}(t)(\mathbf{z})\nabla_t\boldsymbol{\nu}_t(\mathbf{z})\right)(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t))\right] \\
&\quad \text{Substituting } \boldsymbol{\nu}(\mathbf{z},t) = \exp(-\boldsymbol{\gamma}(\mathbf{z},t)) \text{ from Sec. D.1.1} \\
&= -\frac{1}{2}\mathbb{E}_{t\sim[0,1]}\left[(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t))^\top\mathrm{diag}\left(\exp\left(\boldsymbol{\gamma}(\mathbf{z},t)\right)\nabla_t\exp\left(-\boldsymbol{\gamma}(\mathbf{z},t)\right)\right)(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t))\right] \\
&= \frac{1}{2}\mathbb{E}_{t\sim[0,1]}\left[(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t))^\top\mathrm{diag}\left(\exp\left(\boldsymbol{\gamma}(\mathbf{z},t)\right)\exp\left(-\boldsymbol{\gamma}(\mathbf{z},t)\right)\nabla_t\boldsymbol{\gamma}(\mathbf{z},t)\right)(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t))\right] \\
&= \frac{1}{2}\mathbb{E}_{t\sim[0,1]}\left[(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t))^\top\mathrm{diag}\left(\nabla_t\boldsymbol{\gamma}(\mathbf{z},t)\right)(\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t))\right] \qquad (65)
\end{aligned}
$$

D.6 Recovering VDM from the Vectorized Representation of the diffusion loss

Notice that we recover the loss function in VDM when $\boldsymbol{\nu}(\mathbf{z},t) = \nu(t)\mathbf{1_d}$ where $\nu_t \in \mathbb{R}^+$ and $\mathbf{1_d}$ represents a vector of 1s of size d and the noising schedule isn't conditioned on $\mathbf{z}$.

$$
\begin{aligned}
\int_0^1 \langle \mathbf{f}_\theta(\mathbf{x}_0, \boldsymbol{\nu}(\mathbf{z},t)), \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\nu}(t)\rangle \mathrm{d}t &= \int_0^1 \langle \mathbf{f}_\theta(\mathbf{x}_0, \boldsymbol{\nu}(t)), \frac{\mathrm{d}}{\mathrm{d}t}(\nu(t)\mathbf{1_n})\rangle \mathrm{d}t \\
&= \int_0^1 \langle \mathbf{f}_\theta(\mathbf{x}_0, \boldsymbol{\nu}(t)), \mathbf{1_d}\rangle \nu'(t)\mathrm{d}t \\
&= \int_0^1 \nu'(t)\|\mathbf{f}_\theta(\mathbf{x}_0, \boldsymbol{\nu}(t))\|_1^1 \mathrm{d}t \\
&= \int_0^1 \nu'(t)\|(\mathbf{x}_0 - \tilde{\mathbf{x}}_\theta(\mathbf{x}_{\boldsymbol{\nu}(t)}, \boldsymbol{\nu}(t)))\|_2^2 \mathrm{d}t \qquad (66)
\end{aligned}
$$

$\int_0^1 \frac{d}{dt}\nu(t)\|(\mathbf{x}_0 - \tilde{\mathbf{x}}_\theta(\mathbf{x}_{\boldsymbol{\nu}(t)}, \boldsymbol{\nu}(t)))\|_2^2 dt$ denotes the diffusion loss, $\mathcal{L}_{\text{diffusion}}$, as used in VDM; see Kingma et al. (2021).

E Subset Sampling

Sampling a subset of k items from a collection of collection of n items, $x_1, x_2, \ldots, x_3$ belongs to a category of algorithms called reservoir algorithms. In weighted reservoir sampling, every

x_i is associated with a weight $w_i \geq 0$. The probability associated with choosing the sequence $S_{\text{wrs}} = [i_1, i_2, \ldots, i_k]$ be a tuple of indices. Then the probability associated with sampling this sequence is

$$p(S_{\text{wrs}}|\mathbf{w}) = \frac{w_{i_1}}{Z} \frac{w_{i_2}}{Z - w_{i_1}} \cdots \frac{w_{i_k}}{Z - \sum_{j=1}^{k-1} w_{i_j}} \tag{67}$$

Efraimidis & Spirakis (2006) give an algorithm for weighted reservoir sampling where each item is assigned a random key $r_i = u_i^{\frac{1}{w_i}}$ where u_i is drawn from a uniform distribution [0, 1] and w_i is the weight of item x_i. Let TopK($\mathbf{r}, k$) which takes keys $\mathbf{r} = [r_1, r_2, \ldots, r_n]$ and returns a sequence $[i_1, i_2, \ldots, i_k]$. Efraimidis & Spirakis (2006) proved that TopK($\mathbf{r}, k$) is distributed according to $p(S_{\text{wrs}}|\mathbf{w})$.

Let's represent a subset $S \in \{0, 1\}^n$ with exactly k non-zero elements that are equal to 1. Then the probability associated with sampling S is given as,

$$p(S|\mathbf{w}) = \sum_{S_{\text{wrs}} \in \Pi(S)} p(S_{\text{wrs}}|\mathbf{w}) \tag{68}$$

where $\Pi(S)$ denotes all possible permutations of the sequence S. By ignoring the ordering of the elements in S_{wrs} we can sample using the same algorithm. Xie & Ermon (2019) show that this sampling algorithm is equivalent to TopK($\hat{\mathbf{r}}, k$) where $\hat{\mathbf{r}} = [\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_n]$ where $\hat{r}_i = -\log(-\log(r_i)) = \log w_i +$ Gumbel(0, 1). This holds true because the monotonic transformation $-\log(-\log(x))$ preserves the ordering of the keys and thus TopK($\mathbf{r}, k$) $\equiv$ TopK($\hat{\mathbf{r}}, k$).

Sum of Gamma Distribution. Niepert et al. (2021) show that adding SOG noise instead of Gumbel noise leads to better performance.

Niepert et al. (2021) show that $\mathbf{z} \sim p_\theta(\mathbf{z}; \theta)$ is equivalent to $\mathbf{z} = \arg\max_{y \in Y} \langle \theta + \epsilon_g, y \rangle$ where ϵ_g is a sample from Sum-of-Gamma distribution given by

$$\text{SoG}(k, \tau, s) = \frac{\tau}{k} \left(\sum_{i=1}^{s} \text{Gamma}\left(\frac{1}{k}, \frac{k}{i}\right) - \log s \right), \tag{69}$$

where s is a positive integer and $\text{Gamma}(\alpha, \beta)$ is the Gamma distribution with (α, β) as the shape and scale parameters.

And hence, given logits $\log \mathbf{w}$, we sample a k-hot vector using TopK($\log \mathbf{w} + \epsilon$). We choose a categorical prior with uniform distribution across n classes. Thus the KL loss term is given by:

$$-\sum_{i=1}^{n} \frac{w_i}{Z} \log\left(n \frac{w_i}{Z}\right) \tag{70}$$

F Experiment Details

F.1 Model Architecture

Denoising network. Our model architecture is extremely similar to VDM. The UNet of our pixel-space diffusion has an unchanged architecture from Kingma et al. (2021).This structure is specifically designed for optimal performance in maximum likelihood training. We employ features from VDM such as the elimination of internal downsampling/upsampling processes and the integration of Fourier features to enhance fine-scale prediction accuracy. In alignment with the configurations suggested by Kingma et al. (2021), our approach varies depending on the dataset: For CIFAR-10, we employ a U-Net with a depth of 32 and 128 channels; for ImageNet-32, the U-Net also has a depth of 32, but the channel count is increased to 256. Additionally, all these models incorporate a dropout rate of 0.1 in their intermediate layers.

Encoder network. $q_\phi(\mathbf{z}|\mathbf{x})$ is modeled using a sequence of 4 Resnet blocks with a channel count of 128 for CIFAR-10 and 256 for ImageNet-32 with a drop out of 0.1 in their intermediate layers.

Table 3: Likelihood in bits per dimension (BPD) on the test set of CIFAR-10 and ImageNet. Results with "/" means they are not reported in the original papers. Model types are autoregressive (AR), normalizing flows (Flow), variational autoencoders (VAE), diffusion models (Diff), diffusion ODEs (Diff ODE). The likelihood for "Diff" type models is computed using the VLB-based method described in appendix Sec. H.1, while "Diff ODE" type models utilize an ODE-based exact likelihood estimate as detailed in appendix Sec. H.2. Additionally, for MuLAN , we present the mean and a 95% confidence interval.

Model	Type	CIFAR-10 (↓)	ImageNet (↓)
PixelCNN (Van den Oord et al., 2016)	AR	3.03	3.83
PixelCNN++ (Salimans et al., 2017)	AR	2.92	/
Glow (Kingma & Dhariwal, 2018)	Flow	/	4.09
Image Transformer (Parmar et al., 2018)	AR	2.90	3.77
DDPM (Ho et al., 2020)	Diff	3.69	/
Score SDE (Song et al., 2020)	Diff	2.99	/
Improved DDPM (Nichol & Dhariwal, 2021)	Diff	2.94	/
VDM (Kingma et al., 2021)	Diff	2.65	3.72
Flow Matching (Lipman et al., 2022)	Flow	2.99	/
i-DODE (Zheng et al., 2023) (VLB-based)	Diff	2.61	/
i-DODE (Zheng et al., 2023) (ODE-based)	Diff ODE	2.56	3.69
MuLAN (**Ours**, VLB-based; see Sec. H.1)	Diff	2.60	3.71
MuLAN (**Ours**, ODE-based; see Sec. H.2)	Diff ODE	**2.55** $\pm 10^{-3}$	**3.67** $\pm 10^{-3}$

Noise schedule. For polynomial noise schedule, we use an MLP that maps the latent vector $\mathbf{z}$ to $\mathbf{a}_\phi(\mathbf{z}), \mathbf{b}_\phi(\mathbf{z}), \mathbf{c}(\mathbf{z})$; see Eq. D.2 for details. The MLP has 2 hidden layers of size 3072 with `swish` activation function. The final layer is a linear mapping to 3×3072 values corresponding to $\mathbf{a}_\phi(\mathbf{z}), \mathbf{b}_\phi(\mathbf{z}), \mathbf{c}(\mathbf{z})$. Note that $\mathbf{a}_\phi(\mathbf{z}), \mathbf{b}_\phi(\mathbf{z}), \mathbf{c}(\mathbf{z})$ have the same dimensionality of 3072.

F.2 Hardware.

For the ImageNet experiments, we used a single GPU node with 8-A100s. For the cifar-10 experiments, we used several GPUs types including V100, A5000s, A6000s, A100-40GBs, and 3090s but the experiments were trained on 4 GPUs with `float32` precision.

F.3 Training

This section reports experimental results on the CIFAR-10 (Krizhevsky et al., 2009) and ImageNet-32 (Van Den Oord et al., 2016) datasets. We chose to employ a discrete prior for the auxiliary latent space rather than a Gaussian prior due to training instability issues that frequently led to NaNs. In all our experiments, we set the parameters for the discrete latent distribution as $m = 50$ and $k = 15$.

F.4 Likelihood Estimation.

In Table 1, we present the likelihood estimation results for MuLAN, and other recent methods on CIFAR-10 and ImageNet-32 using the VLB-estimate; details in Sec. H.1. This version of MuLAN was trained with noise parameterization for 10M steps on CIFAR-10 and 2M steps on Imagenet-32, similar to VDM (Kingma et al., 2021). We apply MuLAN on top of the VDM, endowing it with a learned multivariate noising schedule conditioned on auxiliary latent variables. We find that these new components result in a significant improvement in BPD over a vanilla VDM.

We also compute the likelihood using ODE-based exact likelihood estimate with which we outperform all existing methods in denstity estimation on CIFAR-10 and ImageNet-32. We train MuLAN for 8M steps using velocity parameterization on CIFAR-10 and 2M steps on Imagenet-32. During inference, we extract the underlying probability flow ODE; see Sec. H.2, just like Zheng et al. (2023). Although Zheng et al. (2023) used various other techniques, such as importance-sampled training, variance minimization, and higher-order score matching, MuLAN did not incorporate these. Combining these strategies with MuLAN could further improve its performance.

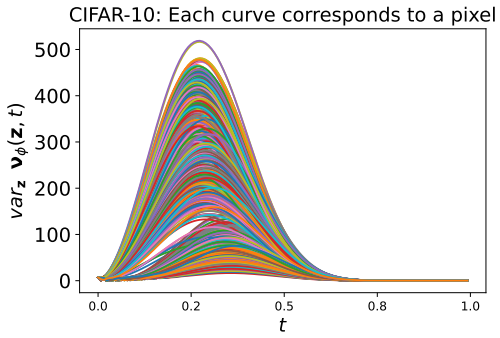
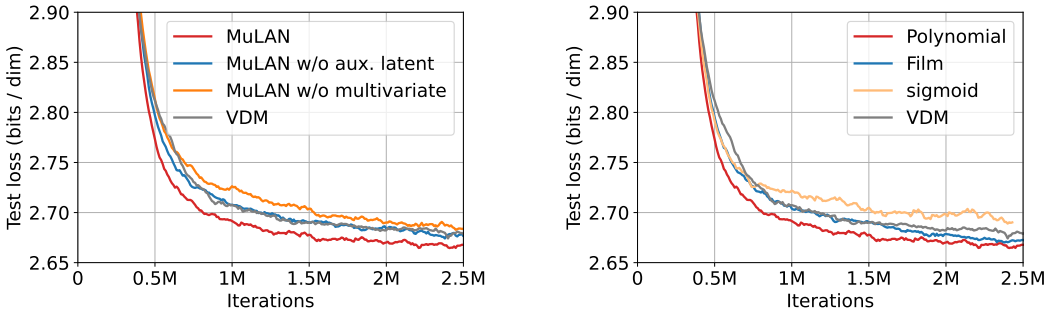


Figure 2: Noise schedule visualizations for MuLAN on CIFAR-10. In this figure, we plot the variance of $\boldsymbol{\nu}_\phi(\mathbf{z}, t)$ across different $\mathbf{z} \sim p_\theta(\mathbf{z})$ where each curve represents the SNR corresponding to an input dimension.



(a) Ablations for different components of MuLAN. MuLAN w/o aux. latent refers to MuLAN where the noise schedule isn't conditioned on an auxililary latent variable. MuLAN w/o multivariate refers to MuLAN with a scalar noise schedule. VDM uses a scalar noise schedule and doesn't have an auxiliary latent space. We see that MuLAN performs the best.

(b) Ablations for noise schedules with different functional parameterizations: sigmoid, polynomial, and a monotonic neural network with FiLM conditioning. We observe that the polynomial parameterization performs the best.

Figure 3: CIFAR-10 ablation studies with a reduced batch size and fewer training steps.

F.5 Ablation Analysis

Due to the expensive cost of training, we only performed ablation studies on CIFAR-10 with a reduced batch size of 64 and trained the model for 2.5M training steps. In Fig. 3 we ablate each component of MuLAN: when we remove the conditioning on an auxiliary latent space from MuLAN so that we have a multivariate noise schedule that is solely conditioned on time t, our performance becomes comparable to that of VDM, on which our model is based. Modifying our method to have a scalar noise schedule conditioned on the auxiliary latent variable $\mathbf{z}$ leads to slightly lower performance than VDM in the initial training stages. However, it gradually converges toward VDM.

Loss curves for different noise schedules. We investigate different parameterizations of the noise schedule in Fig. 3. Among polynomial, sigmoid, and monotonic neural network, we find that the polynomial parameterization yields the best performance. The polynomial noise schedule is a novel component introduced in our work.

Replacing the noise schedules in a trained denoising model. We also wish to confirm experimentally our claim that the learning objective is not invariant to our choice of multivariate noise schedule. To investigate this, we replace the noise schedule in the trained denoising model with two alternatives: MuLAN with scalar noise schedule, and a linear noise schedule: $\boldsymbol{\gamma}_\phi(\mathbf{z}, t) = \gamma_{\text{min}} + t(\gamma_{\text{max}} - \gamma_{\text{min}})\mathbf{1}_\mathbf{d}$; see (Kingma et al., 2021). For both the noise schedules the likelihood worsens to the same value as that of the VDM: 2.65. This experimental result strongly supports our theory that all scalar noise schedules are equivalent, as they compute the likelihood along the same diffusion trajectory. It also

underscores that it is not the multivariate nature or the auxiliary latent space individually, but the combination of both, that makes MuLAN effective.

Examining the noise schedule. Since the noise schedule, $\boldsymbol{\gamma}_\phi(\mathbf{z}, t)$ is multivariate, we expect to learn different noise schedules for different input dimensions and different inputs $\mathbf{z} \sim p_\theta(\mathbf{z})$. In Fig. 2, we take our best trained model on CIFAR-10 and visualize the variance of the noise schedule at each point in time for different pixels, where the variance is taken on 128 samples $\mathbf{z} \sim p_\theta(\mathbf{z})$. We note an increased variation in the early portions of the noise schedule. However, on an absolute scale, the variance of this noise is smaller than we expected. We also tried to visualize noise schedules across different dataset images and across different areas of the same image; refer to Fig. 11. We also generated synthetic datasets in which each datapoint contained only high frequencies or only low frequencies, and with random masking applied to parts of the data points; see Sec. G. Surprisingly, none of these experiments revealed human-interpretable patterns in the learned schedule, although we did observe clear differences in likelihood estimation. We hypothesize that other architectures and other forms of conditioning may reveal interpretable patterns of variation; however, we leave this exploration to future work.

G Datasets and Visualizations

In this section we provide a brief description of the datasets used in the paper and visualize the generated samples and the noise schedules.

G.1 CIFAR-10

The CIFAR-10 dataset (Krizhevsky et al., 2009) is a collection of images consisting of 60,000 32×32 color images in 10 different classes, with each class representing a distinct object or scene. The dataset is divided into 50,000 training images and 10,000 test images, with each class having an equal representation in both sets. The classes in CIFAR-10 include: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck.

Randomly generated samples for the CIFAR-10 datasaet are provided in Fig. 4a for MuLAN and Fig. 4b for VDM. We visualize the noise schedule in Fig. 11.

(a) MuLAN with velocity reparameterization after 8M training iterations.

(b) VDM after 10M training iterations.

Figure 4: CIFAR-10 samples generated by different methods.

G.2 ImageNet-32

ImageNet-32 is a dataset derived from ImageNet Deng et al. (2009), where the original images have been resized to a resolution of 32×32. This dataset comprises 1,281,167 training samples and 50,000 test samples, distributed across 1,000 labels.

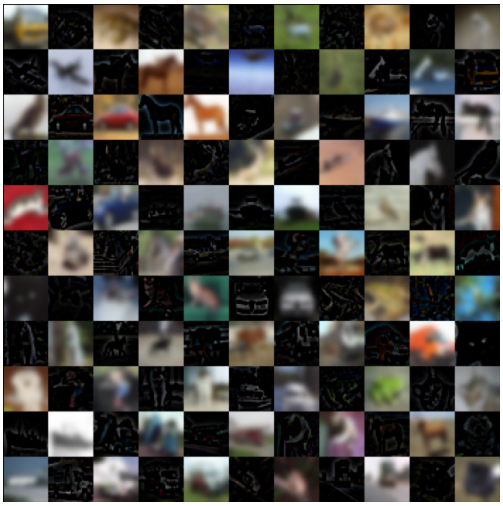
Randomly generated samples for the ImageNet datasaet are provided in Fig. 5 for MuLAN and Fig. 6 for VDM. We visualize the noise schedule in Fig. 11.

Figure 5: MuLAN with noise parameterization after 2M training iterations.

Figure 6: VDM after 2M training iterations.

G.3 Frequency

To see if MuLAN learns different noise schedules for images with different intensities, we modify the images in the CIFAR-10 dataset where we modify an image where we randomly remove the low frequency component an image or remove the high frequency with equal probability. Fig. 7a shows the training samples. MuLAN was trained for 500K steps. The samples generated by MuLAN is shown in Fig. 7b. The corresponding noise schedules is shown in Fig. 11. As compared to CIFAR-10, we notice that the spatial variation in the noise schedule increases (SNRs for all the pixels form a wider band) while the variance of the noise schedule across instances decreases slightly.



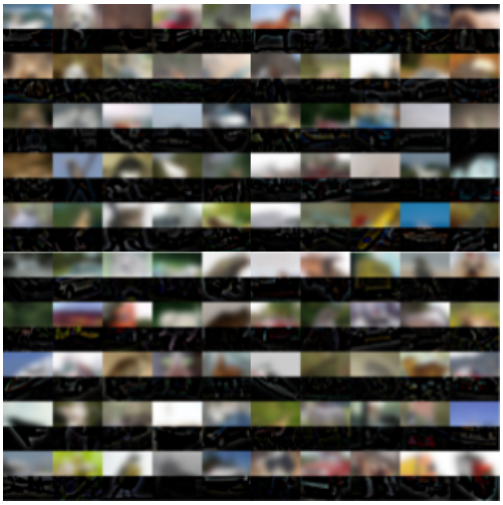
(a) Training samples.

(b) Samples generated by MuLAN with noise parameterization after 500K training iterations.

Figure 7: Frequency Split CIFAR-10 dataset.

G.4 Frequency-2

To see if MuLAN learns different noise schedules for images with different intensities, we modify the images in the CIFAR-10 dataset where we modify an image where we randomly remove the low frequency component an image or remove the high frequency with equal probability. Fig. 7a shows the training samples. MuLAN was trained for 500K steps. The samples generated by MuLAN is shown in Fig. 7b. The corresponding noise schedules is shown in Fig. 11. As compared to CIFAR-10, we notice that the spatial variation in the noise schedule increases (SNRs for all the pixels form a wider band) and the variance of the noise schedule across instances increases as well.



(a) Training samples.

(b) Samples generated by MuLAN with noise parameterization after 500K training iterations.

Figure 8: Frequency Split-2 CIFAR-10 dataset.

G.5 CIFAR-10: Intensity

To see if MuLAN learns different noise schedules for images with different intensities, we modify the images in the CIFAR-10 dataset where we randomly convert an image into a low intensity or

a high intensity image with equal probability. Originally, the CIFAR10 images are in the range [0, 255]. To convert an image into a low intensity image we multiply all pixel values by 0.5. To convert an image into a high intensity image we multiply all the pixel values by 0.5 and add 127.5 to them. Fig. 9a shows the training samples. MuLAN was trained for 500K steps. The samples generated by MuLAN is shown in Fig. 9b. The corresponding noise schedules is shown in Fig. 11. As compared to CIFAR-10, we notice that the spatial variation in the noise schedule slightly increases (SNRs for all the pixels form a wider band) while the variance of the noise schedule across instances slightly decreases.

(a) Training samples.

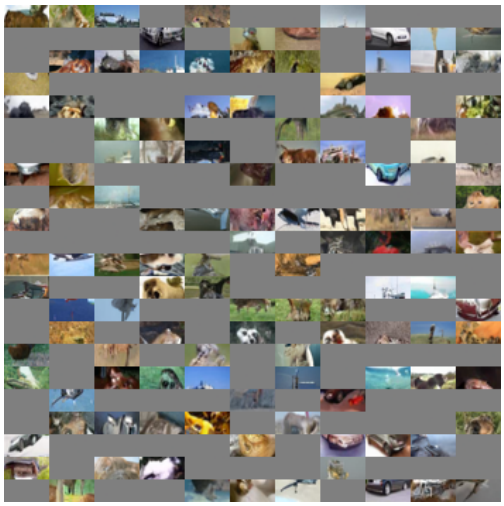
(b) Samples generated by MuLAN with noise parameterization after 500K training iterations.

Figure 9: Intensity CIFAR-10 dataset.

G.6 Mask

We modify the CIFAR-10 dataset where we randomly mask (i.e. replace with **0**s) the top of an image or the bottom half of an image with equal probability. Fig. 10a shows the training samples. MuLAN was trained for 500K steps. The samples generated by MuLAN is shown in Fig. 10b. The corresponding noise schedules is shown in Fig. 11. As compared to CIFAR-10, we notice that the spatial variation in the noise schedule slightly increases (SNRs for all the pixels form a wider band) while the variance of the noise schedule across instances decreases.

(a) Training samples.



(b) Samples generated by MuLAN with noise parameterization after 500K training iterations.

Figure 10: Intensity CIFAR-10 dataset.

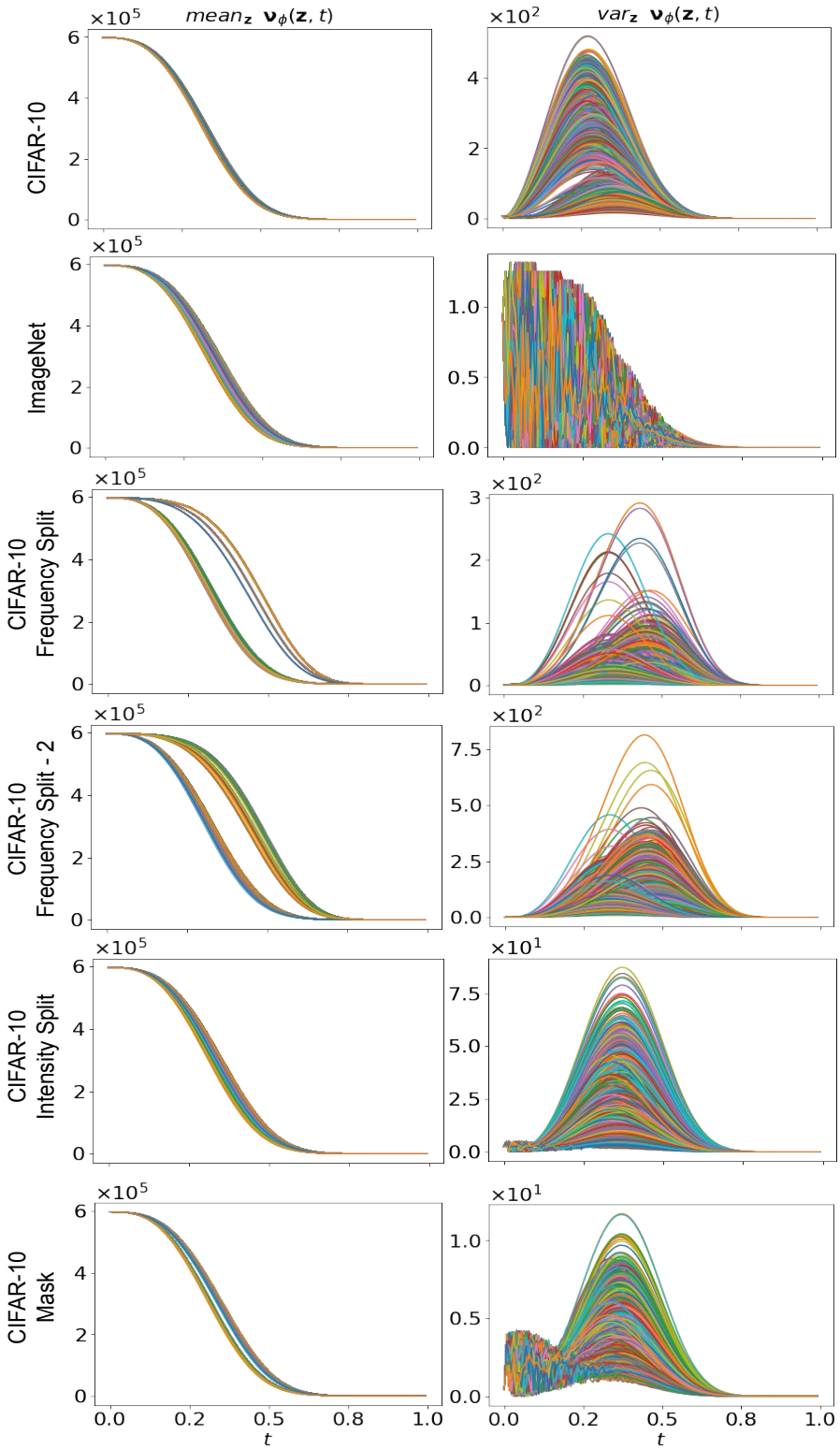


Figure 11: signal-to-noise ratio for different datasets.

H Likelihood Estimation

We used both Variance Lower Bound (VLB) and ODE-based methods to compute BPD.

H.1 VLB Estimate

In the VLB-based approach, we employ Eq. 60. To compute $\mathcal{L}_{\text{diffusion}}$, we use $T = 128$ in Eq. 61, discretizing the timesteps, $t \in [0, 1]$ into 128 bins.

H.2 Exact likelihood computation using Probability Flow ODE

A diffusion process whose marginal is given by (the same as in Eq. 71),

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\alpha}_t\mathbf{x}_0, \text{diag}(\boldsymbol{\sigma}_t^2)), \tag{71}$$

can be modeled as the solution to an Itô Stochastic Differential Equation (SDE):

$$\mathrm{d}\mathbf{x}_t = \mathbf{f}(t)\mathbf{x}_t\mathrm{d}t + \mathbf{g}(t)\mathrm{d}\mathbf{w}_t,\ \mathbf{x}_0 \sim q_0(\mathbf{x}_0), \tag{72}$$

where $\mathbf{f}(t) \in \mathbb{R}^d, \mathbf{g}(t) \in \mathbb{R}^d$ take the following expressions (Song et al., 2020):

$$\mathbf{f}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\log\boldsymbol{\alpha}_t,$$
$$\mathbf{g}^2(t) = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\sigma}_t^2 - 2\boldsymbol{\sigma}_t^2\frac{\mathrm{d}}{\mathrm{d}t}\log\boldsymbol{\alpha}_t$$

The corresponding reverse process, Eq. 2, can also be modelled by an equivalent reverse-time SDE:

$$\mathrm{d}\mathbf{x}_t = [\mathbf{f}(t) - \mathbf{g}(t)^2\nabla_{\mathbf{x}_t}\log q(\mathbf{x}_t|\mathbf{x}_0)]\mathrm{d}t + \mathbf{g}(t)\mathrm{d}\bar{\mathbf{w}}_t,\ \mathbf{x}_1 \sim p_\theta(\mathbf{x}_1), \tag{73}$$

where $\bar{\mathbf{w}}$ is a standard Wiener process when time flows backwards from $1 \to 0$, and $\mathrm{d}t$ is an infinitesimal negative timestep. Song et al. (2020) show that the marginals of Eq. 73 can be described by the following Ordinary Differential Equation (ODE) in the reverse process:

$$\mathrm{d}\mathbf{x}_t = \left[\mathbf{f}(t)\mathbf{x}_t - \frac{1}{2}\mathbf{g}^2(t)\nabla_{\mathbf{x}_t}\log q(\mathbf{x}_t|\mathbf{x}_0)\right]\mathrm{d}t. \tag{74}$$

This ODE, also called the probablity flow ODE, allows us to compute the exact likelihood on any input data via the instantaneous change of variables formula as proposed in Chen et al. (2018). Note that during the reverse process, the term $q(\mathbf{x}_t|\mathbf{x}_0)$ is unknown and is approximated by parameterized by $p_\theta(\mathbf{x}_t)$. For the probability flow defined in Eq. 74, Chen et al. (2018) show that the log- likelihood of $p_\theta(\mathbf{x}_0)$ can be computed using the following equation:

$$\log p_\theta(\mathbf{x}_0) = \log p_\theta(\mathbf{x}_1) - \int_{t=0}^{t=1}\text{tr}\left(\nabla_{\mathbf{x}_t}\mathbf{h}_\theta(\mathbf{x}_t, t)\right)\mathrm{d}t, \tag{75}$$
$$\text{where } \mathbf{h}_\theta(\mathbf{x}_t, t) \equiv \mathbf{f}(t)\mathbf{x}_t - \frac{1}{2}\mathbf{g}^2(t)\nabla_{\mathbf{x}_t}\log p_\theta(\mathbf{x}_t)$$

H.2.1 Probability Flow ODE for MuLAN.

Similarly for the forward process conditioned on the auxiliary latent variable, $\mathbf{z}$,

$$q_\phi(\mathbf{x}_t|\mathbf{x}_0, \mathbf{z}) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\alpha}_t(\mathbf{z})\mathbf{x}_0, \text{diag}(\boldsymbol{\sigma}_t^2(\mathbf{z}))),\ \mathbf{x}_0 \sim q_0(\mathbf{x}_0),\ \mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}_0), \tag{76}$$

we can extend Eq. 72 in the following manner,

$$\mathrm{d}\mathbf{x}_t = \mathbf{f}(\mathbf{z}, t)\mathbf{x}_t\mathrm{d}t + \mathbf{g}(\mathbf{z}, t)\mathrm{d}\mathbf{w}_t,\ \mathbf{x}_0 \sim q_0(\mathbf{x}_0),\ \mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}_0), \tag{77}$$

to obtain the corresponding SDE formulation. Notice that the random variable $\mathbf{z}$ in the above equation doesn't have a subscript t, and hence, $\mathbf{z}$ is drawn from $q_\phi(\mathbf{z}|\mathbf{x}_0)$ once and the same $\mathbf{z}$ is used as $\mathbf{x}_0$ diffuses to $\mathbf{x}_1$. The expressions for $\mathbf{f}(\mathbf{z}, t) : \mathbb{R}^m \times [0, 1] \to \mathbb{R}^d$, $\mathbf{g}(\mathbf{z}, t) : \mathbb{R}^m \times [0, 1] \to \mathbb{R}^d$ is given as follows:

$$\mathbf{f}(\mathbf{z}, t) = \frac{\mathrm{d}}{\mathrm{d}t}\log\boldsymbol{\alpha}_t(\mathbf{z}),$$
$$\mathbf{g}^2(\mathbf{z}, t) = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\sigma}_t^2(\mathbf{z}) - 2\boldsymbol{\sigma}_t^2(\mathbf{z})\frac{\mathrm{d}}{\mathrm{d}t}\log\boldsymbol{\alpha}_t(\mathbf{z})$$

Recall that $\boldsymbol{\alpha}_t^2(\mathbf{z}) = \text{sigmoid}(-\boldsymbol{\gamma}_\phi(\mathbf{z}, t))$, $\boldsymbol{\sigma}_t^2(\mathbf{z}) = \text{sigmoid}(\boldsymbol{\gamma}_\phi(\mathbf{z}, t))$. Substituting these in the above equations, the expressions for $\mathbf{f}(\mathbf{z}, t)$ and $\mathbf{g}^2(\mathbf{z}, t)$ simplify to the following:

$$\mathbf{f}(\mathbf{z}, t) = -\frac{1}{2}\text{sigmoid}(\boldsymbol{\gamma}_\phi(\mathbf{z}, t))\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\gamma}_\phi(\mathbf{z}, t),$$
$$\mathbf{g}^2(\mathbf{z}, t) = \text{sigmoid}(\boldsymbol{\gamma}_\phi(\mathbf{z}, t))\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\gamma}_\phi(\mathbf{z}, t)$$

The corresponding reverse-time SDE is given as:

$$\mathrm{d}\mathbf{x}_t = [\mathbf{f}(t) - \mathbf{g}(t)^2\nabla_{\mathbf{x}_t}\log q_\phi(\mathbf{x}_t|\mathbf{x}_0, \mathbf{z})]\mathrm{d}t + \mathbf{g}(t)\mathrm{d}\bar{\mathbf{w}}_t,\ \mathbf{x}_1 \sim p_\theta(\mathbf{x}_1),\ \mathbf{z} \sim p_\theta(\mathbf{z}), \tag{78}$$

where $\bar{\mathbf{w}}$ is a standard Wiener process when time flows backwards from $1 \to 0$, and $\mathrm{d}t$ is an infinitesimal negative timestep. Given, $\mathbf{s}_\theta(\mathbf{x}_t, \mathbf{z})$, an approximation to the true score function, $\nabla_{\mathbf{x}_t}\log q_\phi(\mathbf{x}_t|\mathbf{x}_0, \mathbf{z})$, Song et al. (2020) show that the marginals of Eq. 78 can be described by the following Ordinary Differential Equation (ODE):

$$\mathrm{d}\mathbf{x}_t = \left[\mathbf{f}(\mathbf{z}, t) - \frac{1}{2}\mathbf{g}^2(\mathbf{z}, t)\mathbf{s}_\theta(\mathbf{x}_t, \mathbf{z})\right]\mathrm{d}t, \tag{79}$$

Zheng et al. (2023) show that the score function, $\mathbf{s}_\theta(\mathbf{x}_t, \mathbf{z})$, for the noise and the velocity parameterization is given as follows:

$$\mathbf{s}_\theta(\mathbf{x}_t, \mathbf{z}) = \begin{cases} -\dfrac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}{\boldsymbol{\sigma}_t(\mathbf{z})} & \text{Noise parameterization; see Sec. D.1.1} \quad (80a) \\ -\mathbf{x}_t - \exp\left(-\dfrac{1}{2}\boldsymbol{\gamma}_\phi(\mathbf{z}, t)\right)\mathbf{v}_\theta(\mathbf{x}_t, \mathbf{z}, t) & \text{Velocity parameterization; see Sec. D.1.2} \quad (80b) \end{cases}$$

Applying the change of variables formula (Chen et al., 2018) on Eq. 79, $\log p_\theta(\mathbf{x}_0|\mathbf{z})$ can be computed in the following manner:

$$\log p_\theta(\mathbf{x}_0|\mathbf{z}) = \log p_\theta(\mathbf{x}_1) - \int_{t=0}^{t=1}\text{tr}\left(\nabla_{\mathbf{x}_t}\mathbf{h}_\theta(\mathbf{x}_t, \mathbf{z}, t)\right)\mathrm{d}t, \tag{81}$$
$$\text{where } \mathbf{h}_\theta(\mathbf{x}_t, \mathbf{z}, t) \equiv \mathbf{f}(\mathbf{z}, t) - \frac{1}{2}\mathbf{g}^2(\mathbf{z}, t)\mathbf{s}_\theta(\mathbf{x}_t, \mathbf{z})$$

The expression for log-likelihood (Eq. 6) is as follows,

$$\begin{aligned} \log p_\theta(\mathbf{x}_0) &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0|\mathbf{z})] - \mathrm{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}_0)\|p_\theta(\mathbf{z})) \\ &\quad \text{Using Eq. 81,} \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_1) - \int_{t=0}^{t=1}\text{tr}\left(\nabla_{\mathbf{x}_t}\mathbf{h}_\theta(\mathbf{x}_t, t, \mathbf{z})\right)\mathrm{d}t\right] - \mathrm{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}_0)\|p_\theta(\mathbf{z})) \end{aligned} \tag{82}$$

Computing $\text{tr}\left(\nabla_{\mathbf{x}_t}\mathbf{h}_\theta(\mathbf{x}_t, t, \mathbf{z})\right)$ is expensive and we follow Chen et al. (2018); Zheng et al. (2023); Grathwohl et al. (2018) to estimate it with Skilling-Hutchinson trace estimator (Skilling, 1989; Hutchinson, 1989). In particular, we have

$$\text{tr}\left(\nabla_{\mathbf{x}_t}\mathbf{h}_\theta(\mathbf{x}_t, t, \mathbf{z})\right) = \mathbb{E}_{p(\epsilon)}\left[\epsilon^\top\nabla_{\mathbf{x}_t}\mathbf{h}_\theta(\mathbf{x}_t, t, \mathbf{z})\epsilon\right], \tag{83}$$

where the random variable ϵ satisfies $\mathbb{E}_{p(\epsilon)}[\epsilon] = \mathbf{0}$ and $\text{Cov}_{p(\epsilon)}[\epsilon] = \mathbf{I}$. Common choices for $p(\epsilon)$ include Rademacher or Gaussian distributions. Notably, the term $\nabla_{\mathbf{x}_t}\mathbf{h}_\theta(\mathbf{x}_t, t, \mathbf{z})\epsilon$ can be computed efficiently using "Jacobian-vector-product" computation in JAX. In our experiments, we follow the exact evaluation procedure for computing likelihood as outlined in Song et al. (2020); Grathwohl et al. (2018). Specifically, for the computation of Eq. 83, we employ a Rademacher distribution for $p(\epsilon)$. To calculate the integral in Eq. 82, we utilize the RK45 ODE solver (Dormand & Prince, 1980) provided by `scipy.integrate.solve_ivp` with `atol=1e-5` and `rtol=1e-5`.

H.2.2 Dequantization.

Real-world datasets for images or texts often consist of discrete data. Attempting to learn a continuous density model directly on these discrete data points can lead to degenerate outcomes (Uria et al., 2013) and fail to provide meaningful density estimations. Dequantization (Salimans et al., 2017; Ho et al., 2020; Zheng et al., 2023) is a common solution in such cases. To elaborate, let x_0 represent 8-bit discrete data scaled to [-1, 1]. Dequantization methods assume that we have trained a continuous model distribution p_θ for x_0, and define the discrete model distribution by

$$P_\theta(\mathbf{x}_0) = \int_{[-\frac{1}{256}, \frac{1}{256})^d} p_\theta(\mathbf{x}_0 + u)\mathrm{d}u.$$

To train $P_\theta(\mathbf{x}_0)$ by maximum likelihood estimation, variational dequantization (Ho et al., 2020; Zheng et al., 2023) introduces a dequantization distribution $q(u|\mathbf{x}_0)$ and jointly train p_{model} and $q(u|\mathbf{x}_0)$ by a variational lower bound:

$$\log P_\theta(\mathbf{x}_0) \geq \mathbb{E}_{q(u|\mathbf{x}_0)}[p_\theta(\mathbf{x}_0 + u) - \log q(u|\mathbf{x}_0)]. \tag{84}$$

Truncated Normal Dequantization. Zheng et al. (2023) show that truncated Normal distribution,

$$q(u|\mathbf{x}_0) = \mathcal{TN}\left(\mathbf{0}, \mathbf{I}, -\frac{1}{256}, \frac{1}{256}\right)$$

with mean $\mathbf{0}$, covariance $\mathbf{I}$, and bounds $\left[-\frac{1}{256}, \frac{1}{256}\right]$ along each dimension, leads to a better likelihood estimate. Thus, Eq. 84 simplifies to the following (for details please refer to section A. in Zheng et al. (2023)):

$$\log P_\theta(\mathbf{x}_0) \geq \mathbb{E}_{\hat{\epsilon}\sim\mathcal{TN}(\mathbf{0},\mathbf{I},-\tau,\tau)}\left[\log p_\theta\left(\mathbf{x}_0 + \frac{\sigma_\epsilon}{\alpha_\epsilon}\hat{\epsilon}\right)\right] + \frac{d}{2}(1 + \log(2\pi\sigma_\epsilon^2)) - 0.01522 \times d \tag{85}$$
$$\text{with } \frac{\sigma_\epsilon}{\alpha_\epsilon} = \exp(-\frac{1}{2} \times 13.3),$$
$$\sigma_\epsilon = \texttt{sqrt}(\text{sigmoid}(-13.3)), \text{ and } \tau = 3.$$

$\log p_\theta\left(\mathbf{x}_0 + \frac{\sigma_\epsilon}{\alpha_\epsilon}\hat{\epsilon}\right)$ is evaluated using Eq. 82.

Importance Weighted Estimator. Eq. 85 can also be extended to obtain an importance weighted likelihood estimator to get a tighter bound on the likelihood. The variational bound is given by (for details please refer to section A. in Zheng et al. (2023)):

$$\log P_\theta(\mathbf{x}_0) \geq \mathbb{E}_{\hat{\epsilon}^{(1)},\ldots,\hat{\epsilon}^{(K)}\sim\mathcal{TN}(\mathbf{0},\mathbf{I},-\tau,\tau)}\left[\log\left(\frac{1}{K}\sum_{i=1}^{K}\frac{p_\theta\left(\mathbf{x}_0 + \frac{\sigma_\epsilon}{\alpha_\epsilon}\hat{\epsilon}^{(k)}\right)}{q(\hat{\epsilon}^{(i)})}\right)\right] + d\log\sigma_\epsilon \tag{86}$$
$$\text{with } \frac{\sigma_\epsilon}{\alpha_\epsilon} = \exp(-\frac{1}{2} \times 13.3), \log\sigma_\epsilon = \frac{1}{2}(-13.3 + \text{softplus}(-13.3)),$$
$$q(\hat{\epsilon}) = \frac{1}{(2\pi Z)^2}\exp\left(-\frac{1}{2}\|\hat{\epsilon}\|_2^2\right), Z = 0.9974613, \text{ and } \tau = 3.$$

Note that for $K = 1$, Eq. 86 is equivalent to Eq. 85; see Zheng et al. (2023). $\log p_\theta\left(\mathbf{x}_0 + \frac{\sigma_\epsilon}{\alpha_\epsilon}\hat{\epsilon}\right)$ is evaluated using Eq. 82. In Table 4, we report BPD values for MuLAN on CIFAR10 (8M training steps, velocity parameterization) and ImageNet (2M training steps, noise parameterization) using both the VLB-based approach, and the ODE-based approach with $K = 1$ and $K = 20$ importance samples.

I MuLAN vs other methods

MuLAN is a noise schedule that can be integrated into any diffusion model such as VDM (Kingma et al., 2021), InfoDiffusion (Wang et al., 2023), or i-DODE (Zheng et al., 2023). The shared components among these models are summarized and compared in Table 5.

Table 4: NLL (mean and 95% Confidence Interval for MuLAN) on CIFAR10 (8M training steps, velocity parameterization) and ImageNet (2M training steps, noise parameterization) using both the VLB-based approach, and the ODE-based approach. $K = 1$ means that we do not use importance weighted estimator since Eq. 86 is equivalent to Eq. 85 for this case; see Zheng et al. (2023).

Likelihood Estimation type	CIFAR-10 (↓)	Imagenet (↓)
VLB-based	2.59 $\pm 10^{-3}$	3.71 $\pm 10^{-3}$
ODE-based ($K = 1$; Eq. 85)	2.59 $\pm 3 \times 10^{-4}$	3.71 $\pm 10^{-3}$
ODE-based ($K = 20$; Eq. 86)	2.55 $\pm 3 \times 10^{-4}$	3.67 $\pm 10^{-3}$

Table 5: MuLAN is a noise schedule that can be integrated into any diffusion model such as VDM (Kingma et al., 2021), InfoDiffusion (Wang et al., 2023), or i-DODE (Zheng et al., 2023). The shared components between MuLAN and these models are summarized and compared in this table.

Method	learned noise	multivariate noise	input conditioned noise	auxiliary latents	noise parameterization
VDM (Kingma et al., 2021)	**Yes**	No	No	No	Monotonic neural network
Blurring Diffusion Model (Hoogeboom & Salimans, 2022)	No	**Yes**	No	No	Frequency scaling
InfoDiffusion (Wang et al., 2023)	No	No	No	In denoising process	Cosine schedule
MuLAN (**Ours**)	**Yes**	**Yes**	**Yes**	In noising and denoising process	Polynomial, sigmoid