DEPARTMENT OF COMPUTER SCIENCE

Video Diffusion Models for Climate Simulations

George Herbert

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering.

Wednesday 5th April, 2023

Abstract

Dedication and Acknowledgements

Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted or otherwise incorporated material which is the work of others, I have included the source in the references. Any views expressed in the dissertation, other than referenced material, are those of the author.

George Herbert, Wednesday 5$^{\text{th}}$ April, 2023

Contents

List of Figures

List of Tables

Ethics Statement

Notation and Acronyms

Chapter 1

Introduction

Chapter 2

Background

2.1 Generative Models

Let us consider some dataset $\mathcal{D}$ consisting of $N \geq 1$ datapoints which we assume are independent and identically distributed:

$$\mathcal{D} = \{\mathbf{x}_i \mid 1 \leq i \leq N, i \in \mathbb{N}\} \tag{2.1}$$

We assume each observed datapoint $\mathbf{x} \in \mathcal{D}$ is a random sample from an underlying process, whose true distribution $p^*(\mathbf{x})$ is unknown. The goal of *generative modelling* is to approximate this true distribution with a chosen model $p_\theta(\mathbf{x})$ with parameters θ. We learn parameters θ such that the probability distribution function given by the model $p_\theta(\mathbf{x})$ approximates the true distribution of the data, such that for any observed $\mathbf{x} \in \mathcal{D}$, we have:

$$p_\theta(\mathbf{x}) \approx p^*(\mathbf{x}) \tag{2.2}$$

Once learned, we can *generate* new samples unconditionally from our approximate model at will.

2.2 Variational Autoencoders

The broader literature contains several well-known approaches to learning a model $p_\theta(\mathbf{x})$. Likelihood-based generative modelling, for example, seeks to learn a model that assigns a high likelihood to the observed datapoints $\mathcal{D}$. Variational autoencoders (VAEs) [7, 9] are one such type of likelihood-based generative model.

2.2.1 Latent Variables

VAEs are *latent-variable models*. That is, we can think of each observed datapoint $\mathbf{x} \in \mathcal{D}$ as being represented or generated by an associated *latent* variable $\mathbf{z}$. The latent variable is part of the model, but we do not observe it directly—it is not within the dataset. We model the joint distribution of the observed data and latent variable by $p_\theta(\mathbf{x}, \mathbf{z})$.

2.2.2 Evidence Lower Bound Objective

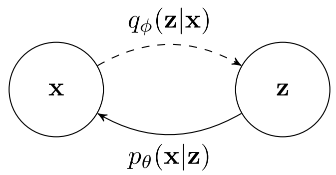

Figure 2.1: Graphical depiction of a VAE

Since VAEs are a likelihood-based generative model, we would ideally train our model to maximise the likelihood of $p_\theta(\mathbf{x})$. However, there is a problem with this approach. Firstly, we cannot simply marginalise out the latent variable $\mathbf{z}$ via:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} \tag{2.3}$$

since the integral does not have an analytic solution or efficient estimator. Secondly, we cannot appeal to the chain rule of probability:

$$p_\theta(\mathbf{x}) = \frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})} \tag{2.4}$$

since it requires access to a ground truth latent encoder $p_\theta(\mathbf{z}|\mathbf{x})$.

2.2.3 Hierarchical Variational Autoencoders

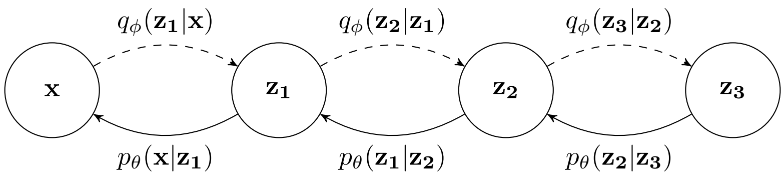

Figure 2.2: Graphical depiction of a hierarchical VAE

2.3 Diffusion Models

Diffusion models are

Given observed datapoints $\mathbf{x}$, the goal of a generative model is to learn to model its true data distribution $q(\mathbf{x})$.

2.3.1 Forward Diffusion Process

The *forward diffusion process* is a Gaussian diffusion process that defines a sequence of increasingly noisy versions of $\mathbf{x}$, which we call the *latent variables*:

$$\mathbf{z} = \{\mathbf{z}_t \mid t \in [0, 1]\} \tag{2.5}$$

The forward process forms a conditional joint distribution $q(\mathbf{z}|\mathbf{x})$, whose marginal distributions of latent variables $\mathbf{z}_t$ given $\mathbf{x} \sim q(\mathbf{x})$ are given by:

$$q(\mathbf{z}_t|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}_t; \alpha_t \mathbf{x}, \sigma_t^2 \mathbf{I}\right) \tag{2.6}$$

where α_t and σ_t are strictly positive scalar-valued functions of t. The joint distribution of latent variables $\mathbf{z}_r, \mathbf{z}_s, \mathbf{z}_t$ at subsequent timesteps $0 \le r < s < t \le 1$ is Markovian:

$$q(\mathbf{z}_t|\mathbf{z}_s, \mathbf{z}_r) = q(\mathbf{z}_t|\mathbf{z}_s) = \mathcal{N}\left(\mathbf{z}_t; \alpha_{t|s}\mathbf{z}_s, \sigma_{t|s}^2 \mathbf{I}\right) \tag{2.7}$$

where $\alpha_{t|s} = \alpha_t \alpha_s^{-1}$ and $\sigma_{t|s}^2 = \sigma_t^2 - \alpha_{t|s}^2 \sigma_s^2$. A full derivation of $q(\mathbf{z}_t|\mathbf{z}_s)$ is given in Appendix A.1.

2.3.2 Noise Schedule

We formalise the notion that $\mathbf{z}_t$ is increasingly noisy by defining the log signal-to-noise ratio

$$\lambda_t = \log\left(\frac{\alpha_t^2}{\sigma_t^2}\right) \in [\lambda_{\min}, \lambda_{\max}] \tag{2.8}$$

as a strictly monotonically decreasing function f_λ of time $t \in [0, 1]$, known as the *noise schedule*.

In this work, we use a truncated continuous-time version of the α-cosine schedule [8], introduced in its original discrete-time form by Nichol and Dhariwal [8]. The α-cosine schedule was motivated by the fact that the 'linear' schedule introduced in prior work by Ho et al. [2] causes α_t to fall to zero more quickly than is optimal. Nichol and Dhariwal empirically found that this induces too much noise in the latter stages of the forward diffusion process; as such, the latent variables $\mathbf{z}_t$ in these stages contribute little to sample quality. In response, they proposed the original discrete-time α-cosine schedule. In this work, we use a continuous-time diffusion model and therefore use an adapted model described in [5]. More formally, we define:

$$f_\lambda(t) = -2\log\left(\tan\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0))\right)\right) \tag{2.9}$$

where t_0 and t_1 truncate $f_\lambda(t)$ to the desired range $[\lambda_{\min}, \lambda_{\max}]$ for $t \in [0,1]$, and are themselves defined as:

$$t_0 = \frac{2}{\pi}\arctan\left(\exp\left(-\frac{1}{2}\lambda_{\max}\right)\right) \tag{2.10}$$

$$t_1 = \frac{2}{\pi}\arctan\left(\exp\left(-\frac{1}{2}\lambda_{\min}\right)\right) \tag{2.11}$$

Figure 2.3 visualises how the log signal-to-noise ratio $\lambda_t \in [\lambda_{\min}, \lambda_{\max}]$ varies with time $t \in [0,1]$ using the α-cosine schedule detailed above.

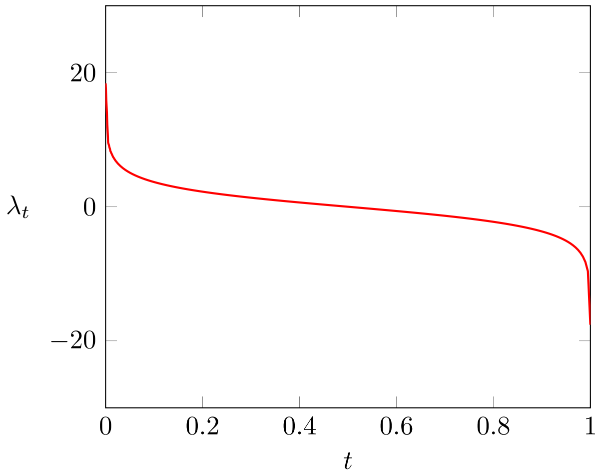

Figure 2.3: Relationship between time t and the log signal-to-noise ratio λ_t for the truncated continuous-time α-cosine noise schedule $f_\lambda(t)$ as defined in Equation 2.9 with $\lambda_{\min} = -30$ and $\lambda_{\max} = 30$. The horizontal axis is time $t \in [0,1]$; the vertical axis is $\lambda_t = f_\lambda(t) \in [\lambda_{\min}, \lambda_{\max}] = [-30, 30]$.

We can compute α_t and σ_t from either λ_t or t via the following equations:

$$\alpha_t = \sqrt{S(\lambda_t)} \quad = \cos\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0)\right) \tag{2.12}$$

$$\sigma_t = \sqrt{S(-\lambda_t)} = \sin\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0)\right) \tag{2.13}$$

where S is the sigmoid function. Figure 2.4 visualises how the values of α_t and σ_t vary with time $t \in [0,1]$ using the α-cosine schedule detailed above. Appendix A.2 provides further details on the form of f_λ and how we can derive the forms for α_t and σ_t.

2.3.3 Generative Model

The *generative model* is a learned hierarchical model that matches the forward process running in reverse-time: in T uniformly-spaced discrete timesteps, we sequentially generate latent variables, starting from $t = 1$ and working backwards to $t = 0$. More formally, our hierarchical generative model defines a joint distribution over latent variables:

$$p_\theta(\mathbf{z}) = p(\mathbf{z}_1)\prod_{i=1}^{T} p_\theta(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)}) \tag{2.14}$$

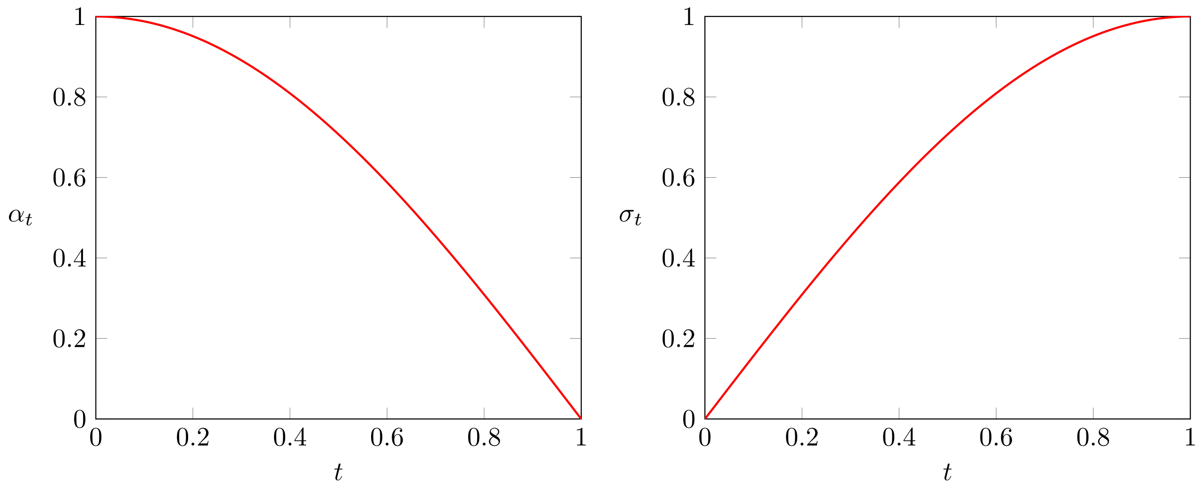

Figure 2.4: Relationship between time t and α_t (left) and σ_t (right) for the same truncated continuous-time α-cosine noise schedule as that in Figure 2.3. The horizontal axis is time $t \in [0, 1]$; the vertical axis is the value of α_t (left) and σ_t (right).

where $s(i) = (i-1) \cdot T^{-1}$ and $t(i) = i \cdot T^{-1}$. For large enough $\lambda_{\max}$, $\mathbf{z}_0$ is almost noiseless, so learning a model $p_\theta(\mathbf{z}_0)$ is practically equivalent to learning a model $p_\theta(\mathbf{x})$.

For sufficiently small $\lambda_{\min}$, $\mathbf{z}_1$ contains almost no information about $\mathbf{x}$. As such, there exists a distribution $p(\mathbf{z}_1)$ such that:

$$D_{KL}(q(\mathbf{z}_1|\mathbf{x}) \| p(\mathbf{z_1})) \approx 0 \tag{2.15}$$

where D_{KL} is the Kullback–Leibler divergence. In this work, we use a variance-preserving diffusion model (i.e. $\alpha_t^2 = 1 - \sigma_t^2$), and as such, we model $p(\mathbf{z}_1)$ as the multivariate standard Gaussian:

$$p(\mathbf{z}_1) = \mathcal{N}(\mathbf{z}_1; \mathbf{0}, \mathbf{I}) \tag{2.16}$$

Once we have sampled $\mathbf{z}_1 \sim p(\mathbf{z}_1)$, we use the discrete-time ancestral sampler [2] to sequentially generate each latent variable $\mathbf{z}_s$ from $\mathbf{z}_t$ where $0 \leq s < t \leq 1$. The discrete-time ancestral sampler samples $\mathbf{z}_s \sim p_\theta(\mathbf{z}_s|\mathbf{z}_t)$ via:

$$p_\theta(\mathbf{z}_s|\mathbf{z}_t) = q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x} = \hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t)) \tag{2.17}$$

$$= \mathcal{N}\left(\tilde{\boldsymbol{\mu}}_{s|t}(\mathbf{z}_t, \mathbf{x} = \hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t)), \tilde{\sigma}_{s|t}\mathbf{I}\right) \tag{2.18}$$

where $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t)$ is our denoised estimate of the original data $\mathbf{x}$ given latent $\mathbf{z}_t$ and log signal-to-noise ratio λ_t, and

$$\tilde{\boldsymbol{\mu}}_{s|t}(\mathbf{z}_t, \mathbf{x}) = \frac{\alpha_{t|s}\sigma_s^2}{\sigma_t^2}\mathbf{z}_t + \frac{\alpha_s \sigma_{t|s}^2}{\sigma_t^2}\mathbf{x} \tag{2.19}$$

$$\tilde{\sigma}_{s|t}^2 = \frac{\sigma_{t|s}\sigma_s}{\sigma_t} \tag{2.20}$$

2.3.4 Parameterisations

In Section 2.3.3, we defined our generative model $p_\theta(\mathbf{x})$ using $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t)$, which takes as input some noisy latent variable $\mathbf{z}_t$ and a log signal-to-noise ratio λ_t and outputs a denoised estimate of the latent. Training a neural network to predict $\mathbf{x} \approx \hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t)$ directly is referred to as the $\mathbf{x}$-prediction parameterisation, but is seldom adopted in the broader literature due to sub-optimal results [2].Recent diffusion models have instead adopted different parameterisations, most commonly the $\boldsymbol{\epsilon}$-prediction parameterisation (e.g. [2, 3, 10]), wherein a neural network is instead trained to predict the noise $\boldsymbol{\epsilon} \approx \hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t, \lambda_t)$, from which we can compute a denoised estimate of noisy latent $\mathbf{z}_t$ via:

$$\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t) = \frac{1}{\alpha_t}\left(\mathbf{z}_t - \sigma_t \hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t, \lambda_t)\right) \tag{2.21}$$

In this work, we employ the $\mathbf{v}$-prediction parameterisation, introduced originally by Salimans and Ho [11], and commonly employed in video diffusion models (e.g. [4, 1]). The $\mathbf{v}$-prediction parameterisation was originally introduced facilitate progressive distillation for faster sampling, though we utilise it here for its additional benefits highlighted by Ho et al. [1], namely faster convergence of sample quality and prevention of temporal colour shifting sometimes observed with $\boldsymbol{\epsilon}$-prediction video diffusion models.

Formally, for a given datapoint $\mathbf{x} \sim q(\mathbf{x})$ we define the velocity of $\mathbf{z}_t \sim q(\mathbf{z}_t|\mathbf{x})$ as:

$$\mathbf{v}_t = \alpha_t \boldsymbol{\epsilon} - \sigma_t \mathbf{x} \tag{2.22}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is multivariate standard Gaussian noise. We train our neural network $\hat{\mathbf{v}}_\theta(\mathbf{z}_t, \lambda_t)$ to minimise the following loss function, defined per datapoint $\mathbf{x}$ as:

$$\mathbb{E}_{\lambda \sim p_\Lambda(\lambda), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})} \left[\|\mathbf{v}_t - \hat{\mathbf{v}}_\theta(\mathbf{z}_t, \lambda_t)\|_2^2 \right] \tag{2.23}$$

During discrete-time ancestral sampling, we convert our estimate $\mathbf{v}_t \approx \hat{\mathbf{v}}_\theta(\mathbf{z}_t, \lambda_t)$ into an estimate of the denoised latent $\mathbf{x} \approx \hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t)$ via:

$$\hat{\mathbf{x}}(\mathbf{z}_t, \lambda_t) = \alpha_t \mathbf{z}_t - \sigma_t \hat{\mathbf{v}}_\theta(\mathbf{z}_t, \lambda_t) \tag{2.24}$$

Appendix A.3 provides further details on the $\mathbf{v}$-prediction parameterisation, including derivations of the velocity and denoised latent.

2.3.5 Objective Function

Diffusion models can be interpreted as a special case of deep variational autoencoders (VAEs) [7, 9] with a particular choice of inference model and generative model.

$$\begin{aligned}
\log p_\theta(\mathbf{x}) &= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}) \right] && (2.25)\\
&= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log \left(\frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})} \right) \right] && (2.26)\\
&= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log \left(\frac{p_\theta(\mathbf{x}, \mathbf{z}) q(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x}) q(\mathbf{z}|\mathbf{x})} \right) \right] && (2.27)\\
&= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log \left(\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right) \right] + \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log \left(\frac{q(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} \right) \right] && (2.28)\\
&= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log \left(\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right) \right] + D_{KL}(q(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) && (2.29)\\
&\geq \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log \left(\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right) \right] && (2.30)\\
&= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log \left(\frac{p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right) \right] && (2.31)\\
&= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) \right] + \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log \left(\frac{p_\theta(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right) \right] && (2.32)\\
&= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) \right] - \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log \left(\frac{q(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z})} \right) \right] && (2.33)\\
&= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) \right] - D_{KL}(q(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) && (2.34)\\
& && (2.35)
\end{aligned}$$

Kingma and Gao [6] discovered that diffusion models in the broader literature are optimised with various objectives that are almost all special cases of a weighted loss, which is defined per datapoint $\mathbf{x}$ as:

$$\begin{aligned}
\mathcal{L}_w &= w(\lambda_{\min})\mathcal{L}(\lambda_{\min}) + \frac{1}{2} \int_{\lambda_{\min}}^{\lambda_{\max}} w(\lambda) \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})} \left[\|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t, \lambda_t)\|_2^2 \right] d\lambda && (2.36)\\
&= w(\lambda_{\min})\mathcal{L}(\lambda_{\min}) + \frac{1}{2} \mathbb{E}_{\lambda \sim p(\lambda), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})} \left[\frac{w(\lambda)}{p(\lambda)} \|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}(\mathbf{z}_t, \lambda_t)\|_2^2 \right] && (2.37)
\end{aligned}$$

where $\mathcal{L}(\lambda)$ is the Kullback–Leibler divergence from the joint distributions q to p for a subset of timesteps from $t = f_\lambda^{-1}(\lambda)$ to 1 for datapoint $\mathbf{x}$; $w(\lambda)$ is a weighting function; and $p(\lambda)$ is determined by the training noise schedule—we can sample from $p(\lambda)$ by first sampling $t \sim \mathcal{U}(0,1)$, then computing $\lambda = f_\lambda(t)$. The first term, $w(\lambda_{\min})\mathcal{L}(\lambda_{\min})$, is constant and close to zero for sufficiently small $\lambda_{\min}$. The second term, however, contains an intractable integral and thus is optimised via an importance-weighted Monte Carlo integrator in practice.

Minimisation of the various loss functions used to optimise diffusion models in the literature equates to minimisation of $\mathcal{L}_w$, with specific choices of $p(\lambda)$ and $w(\lambda)$. Notably, uniform weighting with $w(\lambda) = 1$ for all $\lambda \in [\lambda_{\min}, \lambda_{\max}]$ corresponds to maximisation of the evidence lower bound objective (ELBO). The ELBO—also sometimes known as the variational lower bound—is a lower bound of the log-likelihood of the data. More concretely:

2.3.6 Reconstruction-Guided Sampling

2.4 Climate Simulations

Chapter 3

Results

Chapter 4

Conclusion

Bibliography

[1] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey A. Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models. *CoRR*, abs/2210.02303, 2022.

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[3] Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *J. Mach. Learn. Res.*, 23:47:1–47:33, 2022.

[4] Jonathan Ho, Tim Salimans, Alexey A. Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. *CoRR*, abs/2204.03458, 2022.

[5] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. *CoRR*, abs/2301.11093, 2023.

[6] Diederik P. Kingma and Ruiqi Gao. Understanding the diffusion objective as a weighted integral of elbos. *CoRR*, abs/2303.00848, 2023.

[7] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[8] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 2021.

[9] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1278–1286. JMLR.org, 2014.

[10] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *CoRR*, abs/2205.11487, 2022.

[11] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Appendix A

Diffusion Models

A.1 Derivation of $q(\mathbf{z}_t|\mathbf{z}_s)$

From Equation 2.6, we know $q(\mathbf{z}_t|\mathbf{x})$ is an isotropic Gaussian probability density function. As such, we can sample $\mathbf{z}_t \sim q(\mathbf{z}_t|\mathbf{x})$ by sampling $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ from the multivariate standard Gaussian distribution and computing:

$$\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \boldsymbol{\epsilon}_t \tag{A.1}$$

With some algebraic manipulation, we can show that:

$$\begin{aligned}
\mathbf{z}_t &= \alpha_t \mathbf{x} + \sqrt{\sigma_t^2} \boldsymbol{\epsilon}_t && \text{(A.2)} \\
&= \alpha_t \mathbf{x} + \sqrt{\sigma_t^2 - \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2 + \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2} \boldsymbol{\epsilon}_t && \text{(A.3)} \\
&= \alpha_t \mathbf{x} + \sqrt{\sigma_t^2 - \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2 + \left(\frac{\alpha_t}{\alpha_s}\sigma_s\right)^2} \boldsymbol{\epsilon}_t && \text{(A.4)}
\end{aligned}$$

The sum of two independent Gaussian random variables with mean μ_1 and μ_2 and variance σ_1^2 and σ_2^2 is a Gaussian random variable with mean $\mu_1 + \mu_2$ and variance $\sigma_1^2 + \sigma_2^2$. As such, we can manipulate the above equation further to show that:

$$\begin{aligned}
\mathbf{z}_t &= \alpha_t \mathbf{x} + \sqrt{\sigma_t^2 - \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2} \boldsymbol{\epsilon}_t^* + \frac{\alpha_t}{\alpha_s}\sigma_s \boldsymbol{\epsilon}_s && \text{(A.5)} \\
&= \alpha_t \mathbf{x} + \frac{\alpha_t}{\alpha_s}\sigma_s \boldsymbol{\epsilon}_s + \sqrt{\sigma_t^2 - \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2} \boldsymbol{\epsilon}_t^* && \text{(A.6)} \\
&= \frac{\alpha_s}{\alpha_s}\alpha_t \mathbf{x} + \frac{\alpha_t}{\alpha_s}\sigma_s \boldsymbol{\epsilon}_s + \sqrt{\sigma_t^2 - \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2} \boldsymbol{\epsilon}_t^* && \text{(A.7)} \\
&= \frac{\alpha_t}{\alpha_s}(\alpha_s \mathbf{x} + \sigma_s \boldsymbol{\epsilon}_s) + \sqrt{\sigma_t^2 - \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2} \boldsymbol{\epsilon}_t^* && \text{(A.8)} \\
& && \text{(A.9)}
\end{aligned}$$

where $\boldsymbol{\epsilon}_t^*, \boldsymbol{\epsilon}_s \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ are similarly both sampled from the multivariate standard Gaussian distribution. We can substitute $\mathbf{z}_s = \alpha_s \mathbf{x} + \sigma_s \boldsymbol{\epsilon}_s$ into the above equation to show that:

$$\begin{aligned}
\mathbf{z}_t &= \frac{\alpha_t}{\alpha_s}\mathbf{z}_s + \sqrt{\sigma_t^2 - \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2} \boldsymbol{\epsilon}_t^* && \text{(A.10)} \\
&= \alpha_{t|s}\mathbf{z}_s + \sigma_{t|s}\boldsymbol{\epsilon}_t^* && \text{(A.11)} \\
&\sim \mathcal{N}\left(\mathbf{z}_t; \alpha_{t|s}\mathbf{z}_s, \sigma_{t|s}^2 \mathbf{I}\right) && \text{(A.12)}
\end{aligned}$$

The subscript $t|s$ relates to the fact that $\alpha_{t|s}$ and $\sigma_{t|s}$ define the parameters of the Gaussian probability density function $q(\mathbf{z}_t|\mathbf{z}_s)$.

A.2 α-Cosine Noise Schedule

Before truncation, the continuous-time version of the α-cosine schedule [8] as described in [5] defines α_t^2 at a given timestep $t \in [0, 1]$ as:

$$\alpha_t^2 = \cos^2\left(\frac{\pi}{2}t\right) \tag{A.13}$$

Since our model is a variance-preserving diffusion model, we can show that:

$$\begin{aligned} \sigma_t^2 &= 1 - \alpha_t^2 && \text{(A.14)} \\ &= 1 - \cos^2\left(\frac{\pi}{2}t\right) && \text{(A.15)} \\ &= \sin^2\left(\frac{\pi}{2}t\right) && \text{(A.16)} \end{aligned}$$

As such, we define our noise schedule before truncation $\tilde{f}_\lambda$ for all $t \in [0, 1]$ as:

$$\begin{aligned} \tilde{f}_\lambda(t) &= \log\left(\frac{\alpha_t^2}{\sigma_t^2}\right) && \text{(A.17)} \\ &= \log\left(\frac{\cos^2\left(\frac{\pi}{2}t\right)}{\sin^2\left(\frac{\pi}{2}t\right)}\right) && \text{(A.18)} \\ &= -2\log\left(\tan\left(\frac{\pi}{2}t\right)\right) && \text{(A.19)} \end{aligned}$$

However, the above noise schedule means that $\tilde{f}_\lambda : [0, 1] \to [-\infty, \infty]$; in simpler terms, λ_t is unbounded. We follow prior work (e.g. [5, 4]) by truncating λ_t to the desired range $[\lambda_{\min}, \lambda_{\max}]$. To do so, we first need to define the inverse of the unbounded noise schedule:

$$\tilde{f}_\lambda^{-1}(\lambda) = \frac{2}{\pi}\arctan\left(\exp\left(-\frac{1}{2}\lambda\right)\right) \tag{A.20}$$

From this, we define t_0 and t_1 as:

$$t_0 = \tilde{f}_\lambda^{-1}(0) = \frac{2}{\pi}\arctan\left(\exp\left(-\frac{1}{2}\lambda_{\max}\right)\right) \tag{A.21}$$

$$t_1 = \tilde{f}_\lambda^{-1}(1) = \frac{2}{\pi}\arctan\left(\exp\left(-\frac{1}{2}\lambda_{\min}\right)\right) \tag{A.22}$$

The truncated noise schedule used in this work is then defined as:

$$\begin{aligned} f_\lambda(t) &= \tilde{f}_\lambda(t_0 + t(t_1 - t_0)) && \text{(A.23)} \\ &= -2\log\left(\tan\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0))\right)\right) && \text{(A.24)} \end{aligned}$$

A.3 v-Prediction Parameterisation

From Equation A.1, for a given datapoint $\mathbf{x} \sim q(\mathbf{x})$, we can sample latent variable $\mathbf{z}_t \sim q(\mathbf{z}_t|\mathbf{x})$ via:

$$\mathbf{z}_t = \alpha_t\mathbf{x} + \sigma_t\boldsymbol{\epsilon} \tag{A.25}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is multivariate standard Gaussian noise. We define the velocity of $\mathbf{z}_t$ as

$$\mathbf{v}_t = \frac{d\mathbf{z}_t}{d\phi} \tag{A.26}$$

i.e. the derivative of $\mathbf{z}_t$ with respect to ϕ, which itself is:

$$\begin{aligned} \phi_t &= \arctan\left(\frac{\sigma_t}{\alpha_t}\right) && \text{(A.27)} \\ &= \arctan\left(\frac{\sin\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0))\right)}{\cos\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0))\right)}\right) && \text{(A.28)} \\ &= \arctan\left(\tan\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0))\right)\right) && \text{(A.29)} \\ &= \frac{\pi}{2}(t_0 + t(t_1 - t_0)) && \text{(A.30)} \end{aligned}$$

when using the truncated continuous-time α-cosine noise schedule as per Section 2.3.2. As such, we can formulate the velocity as:

$$\begin{aligned}\mathbf{v}_t = \frac{\mathbf{z}_t}{d\phi} &= \frac{d\cos(\phi)}{d\phi}\mathbf{x} + \frac{d\sin(\phi)}{d\phi}\boldsymbol{\epsilon} && \text{(A.31)}\\ &= -\sin(\phi)\mathbf{x} + \cos(\phi)\boldsymbol{\epsilon} && \text{(A.32)}\\ &= \alpha_t\boldsymbol{\epsilon} - \sigma_t\mathbf{x} && \text{(A.33)}\end{aligned}$$

We can rearrange the above to derive a form for $\mathbf{x}$ in terms of $\mathbf{z}_t$ and $\mathbf{v}_t$ as follows:

$$\begin{aligned}\mathbf{v}_t &= -\sin(\phi)\mathbf{x} + \cos(\phi)\boldsymbol{\epsilon} && \text{(A.34)}\\ \sin(\phi)\mathbf{x} &= \cos(\phi)\boldsymbol{\epsilon} - \mathbf{v}_t && \text{(A.35)}\\ &= \cos(\phi)\left(\frac{\mathbf{z}_t - \cos(\phi)\mathbf{x}}{\sin(\phi)}\right) - \mathbf{v}_t && \text{(A.36)}\\ \sin^2(\phi)\mathbf{x} &= \cos(\phi)\mathbf{z}_t - \cos^2(\phi)\mathbf{x} - \sin(\phi)\mathbf{v}_t && \text{(A.37)}\\ \sin^2(\phi)\mathbf{x} + \cos^2(\phi)\mathbf{x} &= \cos(\phi)\mathbf{z}_t - \sin(\phi)\mathbf{v}_t && \text{(A.38)}\\ (\sin^2(\phi) + \cos^2(\phi))\mathbf{x} &= \cos(\phi)\mathbf{z}_t - \sin(\phi)\mathbf{v}_t && \text{(A.39)}\\ \mathbf{x} &= \cos(\phi)\mathbf{z}_t - \sin(\phi)\mathbf{v}_t && \text{(A.40)}\\ &= \alpha_t\mathbf{z}_t - \sigma_t\mathbf{v}_t && \text{(A.41)}\end{aligned}$$

As per Equation A.33, during training we can

We define the velocity of $\mathbf{z}_t$ as

We rearrange to get:

As such:

$$\mathbf{x} = \alpha_t\mathbf{z}_t - \sigma_t\mathbf{v}_t \tag{A.42}$$

During training, we train the model to minimise:

$$\mathbb{E}_{\mathbf{x},\boldsymbol{\epsilon},t}\left[\|\mathbf{v}_t - \hat{\mathbf{v}}_\theta(\mathbf{z}_t, \lambda_t)\|_2^2\right] \tag{A.43}$$