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 12th 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 12$^{\text{th}}$ April, 2023

Contents

List of Figures

List of Tables

Ethics Statement

Notation and Acronyms

i.i.d.	:	Independent and identically distributed
KL	:	Kullback–Leibler
VAE	:	Variational Autoencoder
	⋮	
$\mathbf{X}^{\circ n}$	:	Element-wise exponentiation of matrix $\mathbf{X}$ with power n
$\text{diag}(\mathbf{x})$	:	Diagonal matrix with the values of vector $\mathbf{x}$ on the diagonal
$\log(x)$	:	Natural logarithm function (i.e. logarithm with base e) applied to x
$f \simeq g$	:	g is an unbiased estimator of f

Chapter 1

Introduction

Chapter 2

Technical Background

2.1 Generative Models

Let us consider some dataset $\mathcal{D}$ consisting of $N_{\mathcal{D}} \geq 1$ datapoints which we assume are independent and identically distributed (i.i.d.):

$$\mathcal{D} = \{\mathbf{x}^{(i)} \mid 1 \leq i \leq N_{\mathcal{D}}, i \in \mathbb{N}\} \tag{2.1}$$

We assume each observed datapoint $\mathbf{x}^{(i)} \in \mathcal{D}$ is a realisation of the observed random variable $\mathbf{x}$ from an underlying process, whose true distribution $p^*(\mathbf{x})$ is unknown. We will omit the index (i) whenever it is clear we are referring to a single datapoint. 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 $\mathbf{x}$:

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

Once learnt, we can generate new samples *unconditionally* from our approximate model at will. We thus refer to the model $p_\theta(\mathbf{x})$ as an unconditional generative model.

2.2 Conditional Generation

We can extend generative modelling to the conditional setting. We consider each observed $\mathbf{x}$ to have some corresponding conditioning information $\mathbf{c}$. In this context, we wish to approximate the conditional distribution $p^*(\mathbf{x}|\mathbf{c})$. Similar to the unconditional setting, we learn parameters θ for our model $p_\theta(\mathbf{x}|\mathbf{c})$ such that for any observed $\mathbf{x}$ and conditioning information $\mathbf{c}$:

$$p_\theta(\mathbf{x}|\mathbf{c}) \approx p^*(\mathbf{x}|\mathbf{c}) \tag{2.3}$$

Once learnt, we can generate new samples *conditionally* from our approximate model at will.

One of the most basic cases is a class-conditional generative model, where the conditioning variable $\mathbf{c}$ is simply a class label. In such cases, our conditional model $p_\theta(\mathbf{x}|\mathbf{c})$ has an interpretation as the reverse of a discriminative classification model—a more traditional form of machine learning. As opposed to inputting an observed $\mathbf{x}$ and the model outputting the predicted corresponding class label $\mathbf{c}$, we input a class label $\mathbf{c}$ and use the model to generate a new sample $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{c})$.

Significantly, the conditioning variable $\mathbf{c}$ is not limited to class labels; it can be flexible and take the form of any additional information we wish to condition on to generate samples. A powerful tool in the case of image generation, $\mathbf{c}$ may be a text encoding to facilitate text-to-image synthesis (see e.g. [11, 6]). Alternatively, $\mathbf{c}$ may be a lower-resolution image from which we wish to upscale to add higher-resolution details, known as image super-resolution (see e.g. [4]).

In this work, we do not explicitly use a conditional model. However, we do derive one approximately from an unconditional model. We discuss this further in Section 2.7.9.

2.3 Latent Variables

We can think of each observed datapoint $\mathbf{x} \in \mathcal{D}$ as being represented or generated via one or more associated *latent variables*, typically denoted $\mathbf{z}$. The latent variables are part of the model, but we do not observe them directly, and they are not within the dataset. We model the joint distribution of the observed variable $\mathbf{x}$ and the latent variables $\mathbf{z}$ by $p_\theta(\mathbf{x}, \mathbf{z})$; the marginal distribution over the observed variable $p_\theta(\mathbf{x})$ is given by:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} \tag{2.4}$$

In the context of a latent-variable model, *generation* refers to the process of sampling the latent variables $\mathbf{z}$ from some prior $p_\theta(\mathbf{z})$, and then sampling the observed variable $\mathbf{x}$ from the likelihood $p_\theta(\mathbf{x}|\mathbf{z})$.

2.4 Likelihood-Based Generative Models

As mentioned in Section 2.1, the goal of a generative model is to learn parameters θ such that $p_\theta(\mathbf{x}) \approx p^*(\mathbf{x})$. One way to interpret this is as a minimisation problem. Namely, we wish to learn parameters θ that minimise the Kullback–Leibler (KL) divergence of the true distribution $p^*(\mathbf{x})$ from our model distribution $p_\theta(\mathbf{x})$:

$$\arg\min_\theta \; D_{KL}(p^*(\mathbf{x}) \| p_\theta(\mathbf{x})) \tag{2.5}$$

The KL divergence, denoted D_{KL}, measures the dissimilarity between two probability distributions; in our case, it provides a measure of the information lost when we approximate the true distribution $p^*(\mathbf{x})$ with our model distribution $p_\theta(\mathbf{x})$.

We can reformulate the KL divergence of the true distribution $p^*(\mathbf{x})$ from our model distribution $p_\theta(\mathbf{x})$ to provide a likelihood-based interpretation:

$$\begin{aligned} D_{KL}(p^*(\mathbf{x}) \| p_\theta(\mathbf{x})) &= \mathbb{E}_{\mathbf{x} \sim p^*(\mathbf{x})} \left[\log \left(\frac{p^*(\mathbf{x})}{p_\theta(\mathbf{x})} \right) \right] && (2.6) \\ &= \mathbb{E}_{\mathbf{x} \sim p^*(\mathbf{x})} \left[\log p^*(\mathbf{x}) \right] + \mathbb{E}_{\mathbf{x} \sim p^*(\mathbf{x})} \left[-\log p_\theta(\mathbf{x}) \right] && (2.7) \\ &= -\mathcal{H}(p^*(\mathbf{x})) + \mathbb{E}_{\mathbf{x} \sim p^*(\mathbf{x})} \left[-\log p_\theta(\mathbf{x}) \right] && (2.8) \end{aligned}$$

where $\mathcal{H}(p^*(\mathbf{x}))$ is the entropy of the true distribution $p^*(\mathbf{x})$ and is constant. As such, minimisation of the KL divergence in this context equates to minimisation of the expected negative log-likelihood of our model distribution $p_\theta(\mathbf{x})$ with respect to $\mathbf{x} \sim p^*(\mathbf{x})$. We can equivalently interpret this as maximisation of the expected log-likelihood:

$$\begin{aligned} \arg\min_\theta \; D_{KL}(p^*(\mathbf{x}) \| p_\theta(\mathbf{x})) &= \arg\min_\theta \; \mathbb{E}_{\mathbf{x} \sim p^*(\mathbf{x})} \left[-\log p_\theta(\mathbf{x}) \right] && (2.9) \\ &= \arg\max_\theta \; \mathbb{E}_{\mathbf{x} \sim p^*(\mathbf{x})} \left[\log p_\theta(\mathbf{x}) \right] && (2.10) \end{aligned}$$

Under the assumption that each of the $N_\mathcal{D}$ samples in our dataset $\mathcal{D}$ are i.i.d. according to $p^*(\mathbf{x})$, we can construct an unbiased estimator:

$$\mathbb{E}_{\mathbf{x} \sim p^*(\mathbf{x})} \left[\log p_\theta(\mathbf{x}) \right] \simeq \frac{1}{N_\mathcal{D}} \log p_\theta(\mathcal{D}) = \frac{1}{N_\mathcal{D}} \sum_{\mathbf{x} \in \mathcal{D}} \log p_\theta(\mathbf{x}) \tag{2.11}$$

In other words, under the i.i.d assumption of $\mathcal{D}$, the mean log-likelihood of our model with respect to $\mathcal{D}$ serves as an unbiased estimator of the expected log-likelihood of our model with respect to $\mathbf{x} \sim p^*(\mathbf{x})$. In practice, for computational efficiency reasons—as well as GPU memory limitations—we learn via mini-batches $\mathcal{M} \subset \mathcal{D}$ of size $N_\mathcal{M} < N_\mathcal{D}$, which is itself an unbiased estimator:

$$\frac{1}{N_\mathcal{D}} \log p_\theta(\mathcal{D}) \simeq \frac{1}{N_\mathcal{M}} \log p_\theta(\mathcal{M}) = \frac{1}{N_\mathcal{M}} \sum_{\mathbf{x} \in \mathcal{M}} \log p_\theta(\mathbf{x}) \tag{2.12}$$

As such, by transitivity the mean log-likelihood of our model with respect to each mini-batch $\mathcal{M}$ is itself an unbiased estimator of the expected log-likelihood of our model with respect to $\mathbf{x} \sim p^*(\mathbf{x})$.

We refer to the broad class of generative models trained to maximise the expected log-likelihood of $p_\theta(\mathbf{x})$ with respect to $\mathbf{x} \sim p^*(\mathbf{x})$ as *likelihood-based generative models*.

2.5 Variational Autoencoders

2.5.1 Overview

The variational autoencoder (VAE) [8, 10] is one example of a model that utilises latent variables. In its simplest form, the VAE is a latent-variable model $p_\theta(\mathbf{x}, \mathbf{z})$ with a single latent $\mathbf{z}$. We assume that each observed datapoint $\mathbf{x} \in \mathcal{D}$ is generated via a two-step process. First, a latent $\mathbf{z}$ is generated from some true prior distribution $p^*(\mathbf{z})$, followed by an observed value $\mathbf{x}$ generated from some true conditional distribution $p^*(\mathbf{x}|\mathbf{z})$.

ur model takes the form:

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z}) \tag{2.13}$$

The prior $p_\theta(\mathbf{z})$ over the latent variable is often chosen to be the multivariate standard Gaussian:

$$p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I}) \tag{2.14}$$

The true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable, but we commonly assume it takes on the form of a multivariate Gaussian with diagonal covariance. As such, we introduce a variational approximate posterior, defined as:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}, \boldsymbol{\mu}_\phi(\mathbf{z}), \text{diag}(\boldsymbol{\sigma}_\phi(\mathbf{z}))^{\circ 2}) \tag{2.15}$$

with variational parameters ϕ.

$p_\theta(\mathbf{x}|\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$ are often referred to as the *decoder* and *encoder*, respectively.

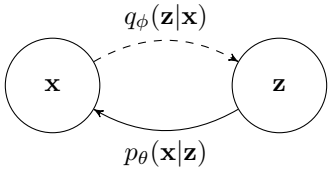

Figure 2.1: Graphical depiction of basic VAE with one observed variable $\mathbf{x}$ and one latent variable $\mathbf{z}$. Solid lines depict the Bayesian network of the generative model; dashed lines depict the Bayesian network of the approximate inference model.

2.5.2 Evidence Lower Bound Objective

For the basic VAE with a single latent variable, we sample from $p_\theta(\mathbf{x})$ by first sampling the latent $\mathbf{z}$ from some prior $p_\theta(\mathbf{z})$ and then sampling $\mathbf{x}$ from $p_\theta(\mathbf{x}|\mathbf{z})$; this defines a generative model of the form:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})d\mathbf{z} \tag{2.16}$$

However,

Firstly, we cannot simply marginalise out the latent variable $\mathbf{z}$ via:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})d\mathbf{z} \tag{2.17}$$

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

$$p_\theta(\mathbf{x}) = \frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})} \tag{2.18}$$

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

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x})\right] \tag{2.19}$$

$$= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right)\right] \tag{2.20}$$

$$= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})q_\phi(\mathbf{z}|\mathbf{x})}\right)\right] \tag{2.21}$$

$$= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right)\right] + \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{q(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right)\right] \tag{2.22}$$

$$= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right)\right] + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) \tag{2.23}$$

The second term in Equation 2.23 is the KL divergence of $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{z}|\mathbf{x})$ and is non-negative:

$$D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) \geq 0 \tag{2.24}$$

and zero if and only if $q_\phi(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}|\mathbf{x})$. The first term in Equation 2.23 is the *evidence lower bound objective* (ELBO), and by the non-negativity of the KL divergence, serves as a lower bound on the log-likelihood of the observed variable $\mathbf{x}$:

$$\log p_\theta(\mathbf{x}) \geq \text{ELBO}(\mathbf{x}) \tag{2.25}$$

$$= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right)\right] \tag{2.26}$$

As such, maximisation of the ELBO accomplishes two things. Firstly, it will approximately maximise the log-likelihood of the observed variable $\mathbf{x}$—the overriding goal of a likelihood-based generative model. Secondly, it will minimise the KL divergence of $q_\phi(\mathbf{z}|\mathbf{x})$ from $p_\theta(\mathbf{z}|\mathbf{x})$, thus encouraging the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ to approximate the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ as closely as possible.

2.5.3 Hierarchical Variational Autoencoders

2.5.4 Markovian Hierarchical Variational Autoencoders

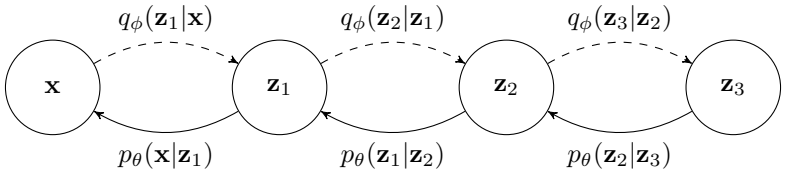

Figure 2.2: Graphical depiction of Markovian hierarchical VAE with one observed variable $\mathbf{x}$ and three latent variables $\mathbf{z}_1$, $\mathbf{z}_2$ and $\mathbf{z}_3$. Solid lines depict the Bayesian network of the generative model; dashed lines depict the Bayesian network of the approximate inference model.

2.6 Score-Based Generative Models

2.6.1 Score Networks

Score-based generative models [14, 15] are an alternative to likelihood-based generative models. We define the *score* of a probability density $p(\mathbf{x})$ to be:

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}) \tag{2.27}$$

The *score network* $\mathbf{s}_\theta$ is a neural network parameterised by θ trained to approximate the score of the true data distribution $p^*(\mathbf{x})$, such that for any observed $\mathbf{x}$:

$$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p^*(\mathbf{x}) \tag{2.28}$$

Importantly, we accomplish this without training a model to directly approximate the true data distribution $p^*(\mathbf{x})$ in advance.

2.6.2 Example: Denoising Score Matching with Langevin Dynamics

Denoising score matching with Langevin dynamics (SMLD) [14] is an example of a score-based generative model that enables us to generate data starting from Gaussian noise. While we do not use SMLD explicitly in this work, we introduce it here to better motivate techniques employed in the diffusion model described in Section 2.7. We introduce SMLD in continuous time. Let us consider a set of positive noise scales:

$$\{\sigma_t \mid t \in [0,1]\} \tag{2.29}$$

such that σ_t is a strictly monotonically increasing function of time $t \in [0,1]$. For each noise scale σ_t, we define a corresponding Gaussian noise perturbation kernel:

$$q(\mathbf{z}_t|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}_t; \mathbf{x}, \sigma_t^2\mathbf{I}\right) \tag{2.30}$$

For all $t \in [0,1]$ the marginal distribution $q(\mathbf{z}_t)$ is given by:

$$q(\mathbf{z}_t) = \int p^*(\mathbf{x})q(\mathbf{z}_t|\mathbf{x}) \tag{2.31}$$

We define $\sigma_0 = \sigma_{\min}$ and $\sigma_1 = \sigma_{\max}$ such that:

$$q(\mathbf{z}_0) \approx p^*(\mathbf{x}) \tag{2.32}$$

$$q(\mathbf{z}_1) \approx \mathcal{N}(\mathbf{z}_1; \mathbf{0}, \sigma_1^2\mathbf{I}) \tag{2.33}$$

We train a noise-conditional score network $\mathbf{s}_\theta(\mathbf{z}_t, \sigma)$ such that:

$$\mathbf{s}_\theta(\mathbf{z}_t, \sigma_t) \approx \nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t) \tag{2.34}$$

For our generative model, we start by sampling $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{z}_1; \mathbf{0}, \sigma_1^2\mathbf{I})$ from an isotropic Gaussian distribution with variance $\sigma_1^2 = \sigma_{\max}^2$. Then, we sequentially generate $\mathbf{z}_s$ from $\mathbf{z}_t$ where $0 \le s < t \le 1$ via Langevin Markov chain Monte Carlo:

$$\mathbf{z}_s = \mathbf{z}_t + \frac{1}{2}\tau_t \mathbf{s}_\theta(\mathbf{z}_t, \sigma_t) + \sqrt{\tau_t}\boldsymbol{\epsilon} \tag{2.35}$$

where $\tau_t > 0$ is a pre-defined time-dependent step size and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is multivariate standard Gaussian noise.

2.7 Diffusion Models

2.7.1 Forward Diffusion Process

Specified in continuous time, 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:

$$\{\mathbf{z}_0, \ldots, \mathbf{z}_1\} = \{\mathbf{z}_t \mid t \in [0,1]\} \tag{2.36}$$

Song et al. [15] showed that we can use an Itô stochastic differential equation (SDE) to define the time evolution of the diffusion process:

$$d\mathbf{z} = \mathbf{f}(\mathbf{z}, t)dt + g(t)d\mathbf{w} \tag{2.37}$$

where $\mathbf{w}$ is the standard Wiener process, $\mathbf{f}(\mathbf{z}, t)$ is a vector-valued function called the *drift* coefficient of $\mathbf{z}$, and $g(t)$ is a scalar function known as the *diffusion* coefficient of $\mathbf{z}$. In this work, we use a *variance-preserving* diffusion model, which is defined by:

$$\mathbf{f}(\mathbf{z}, t) = -\frac{1}{2}\left(\frac{d}{dt}\log\left(1 + \exp(-\lambda_t)\right)\right)\mathbf{z} \tag{2.38}$$

$$g(t)^2 = \frac{d}{dt}\log\left(1 + \exp(-\lambda_t)\right) \tag{2.39}$$

where $\lambda_t \in [\lambda_{\min}, \lambda_{\max}]$ is a monotonically decreasing scalar-valued function of time $t \in [0,1]$. We provide more details on λ_t in Section 2.7.3.

The forward process forms a conditional joint distribution $q(\mathbf{z}_0, \ldots, \mathbf{z}_1|\mathbf{x})$, whose marginal distribution of each latent variable $\mathbf{z}_t$ given $\mathbf{x} \sim p^*(\mathbf{x})$ is given by:

$$q(\mathbf{z}_t|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}_t; \alpha_t \mathbf{x}, \sigma_t^2 \mathbf{I}\right) \tag{2.40}$$

where α_t and σ_t are monotonic functions of λ_t, with the exact relationship also described in Section 2.7.3. The true distribution $p^*(\mathbf{x})$ plus the conditional joint distribution of the forward model $q(\mathbf{z}_0, \ldots, \mathbf{z}_1|\mathbf{x})$ defines a joint distribution:

$$q(\mathbf{z}_0, \ldots, \mathbf{z}_1) = \int p^*(\mathbf{x}) q(\mathbf{z}_0, \ldots, \mathbf{z}_1|\mathbf{x}) d\mathbf{x} \tag{2.41}$$

with marginals $q(\mathbf{z}_t)$. The joint distribution of latent variables $\mathbf{z}_r, \mathbf{z}_s, \mathbf{z}_t$ at subsequent timesteps $0 \leq r < s < t \leq 1$ is Markovian:

$$q(\mathbf{z}_t|\mathbf{z}_s, \mathbf{z}_r) = q(\mathbf{z}_t|\mathbf{z}_s) = \mathcal{N}\left(\mathbf{z}_t; \alpha_{t|s}\mathbf{z}_s, \sigma_{t|s}^2 \mathbf{I}\right) \tag{2.42}$$

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.

2.7.2 Generative Model

Let $\mathbf{s}_\theta(\mathbf{z}_t, \lambda_t)$ be a λ-dependent score network [14] that approximates the score of $q(\mathbf{z}_t)$ such that:

$$\mathbf{s}_\theta(\mathbf{z}_t, \lambda_t) \approx \nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t) \tag{2.43}$$

Anderson [1] showed that if we have a perfect score model $\mathbf{s}_\theta(\mathbf{z}_t, \lambda_t) = \nabla_{\mathbf{z}_t} q(\mathbf{z}_t)$ then the forward SDE of Equation 2.37 is exactly solved by the reverse-time SDE:

$$d\mathbf{z} = \left[\mathbf{f}(\mathbf{z}, t) - g(t)^2 \mathbf{s}_\theta(\mathbf{z}_t, \lambda_t)\right] dt + g(t) d\mathbf{w} \tag{2.44}$$

In the broader literature, diffusion models utilise a variety of numerical solvers to provide approximate trajectories of the reverse-time SDE. In this work, we sequentially generate latent variables starting from $t = 1$ and working backwards to $t = 0$, over T uniformly-spaced discrete timesteps. More formally, this comprises a hierarchical generative model that defines a joint distribution over latent variables as follows:

$$p_\theta(\mathbf{z}_0, \ldots, \mathbf{z}_1) = p_\theta(\mathbf{z}_1) \prod_{i=1}^{T} p_\theta(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)}) \tag{2.45}$$

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 $p_\theta(\mathbf{z}_1)$ such that:

$$D_{KL}(q(\mathbf{z}_1|\mathbf{x}) \| p_\theta(\mathbf{z}_1)) \approx 0 \tag{2.46}$$

For variance-preserving diffusion models, as used in this work, we model $p_\theta(\mathbf{z}_1)$ as the multivariate standard Gaussian:

$$p_\theta(\mathbf{z}_1) = \mathcal{N}(\mathbf{z}_1; \mathbf{0}, \mathbf{I}) \tag{2.47}$$

Once we have sampled $\mathbf{z}_1 \sim p_\theta(\mathbf{z}_1)$, we use the discrete-time ancestral sampler [3] to sequentially generate each latent variable $\mathbf{z}_s$ from $\mathbf{z}_t$ where $0 \leq s < t \leq 1$. This corresponds to a particular discretisation of the reverse-time variance-preserving SDE, as shown by Song et al. [15]. More formally, from a given latent $\mathbf{z}_t$ we generate $\mathbf{z}_s \sim p_\theta(\mathbf{z}_s|\mathbf{z}_t)$ via:

$$p_\theta(\mathbf{z}_s|\mathbf{z}_t) = q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x} = \hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t)) \tag{2.48}$$

$$= \mathcal{N}\left(\tilde{\boldsymbol{\mu}}_{s|t}(\mathbf{z}_t, \mathbf{x} = \hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t)), \tilde{\sigma}_{s|t} \mathbf{I}\right) \tag{2.49}$$

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

$$\tilde{\boldsymbol{\mu}}_{s|t}(\mathbf{z}_t, \mathbf{x}) = \frac{\alpha_{t|s}\sigma_s^2}{\sigma_t^2} \mathbf{z}_t + \frac{\alpha_s \sigma_{t|s}^2}{\sigma_t^2} \mathbf{x} \tag{2.50}$$

$$\tilde{\sigma}_{s|t} = \frac{\sigma_{t|s}\sigma_s}{\sigma_t} \tag{2.51}$$

Interestingly, in Equation 2.48 we introduced a denoiser network $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t)$ to define the discrete-time ancestral sampler, while in Equation 2.43 we described diffusion models as learning a score network $\mathbf{s}_\theta(\mathbf{z}_t, \lambda_t)$. One of the powerful aspects of diffusion models is that we can freely switch between different parameterisations. For example, we can train a neural network $\mathbf{s}_\theta(\mathbf{z}_t, \lambda_t)$ to predict the score of $\mathbf{z}_t$ and then convert the output to a denoised estimate of $\mathbf{z}_t$, as if we had trained a denoiser network $\mathbf{x}_\theta(\mathbf{z}_t, \lambda_t)$ directly. This reparameterisation property can be exceptionally advantageous. In this work, we explicitly train neither a score nor a denoiser network but rather use the $\mathbf{v}$-prediction parameterisation [12]; we describe this in detail in Section 2.7.4.

2.7.3 Noise Schedule

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

$$\lambda_t = \log\left(\frac{\alpha_t^2}{\sigma_t^2}\right) \in [\lambda_{\min}, \lambda_{\max}] \tag{2.52}$$

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 [9], introduced in its original discrete-time form by Nichol and Dhariwal [9]. The α-cosine schedule was motivated by the fact that the 'linear' schedule introduced in prior work by Ho et al. [3] 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 [6]. More formally, we define:

$$f_\lambda(t) = -2\log\left(\tan\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0))\right)\right) \tag{2.53}$$

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:

$$t_0 = \frac{2}{\pi}\arctan\left(\exp\left(-\frac{1}{2}\lambda_{\max}\right)\right) \tag{2.54}$$

$$t_1 = \frac{2}{\pi}\arctan\left(\exp\left(-\frac{1}{2}\lambda_{\min}\right)\right) \tag{2.55}$$

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.

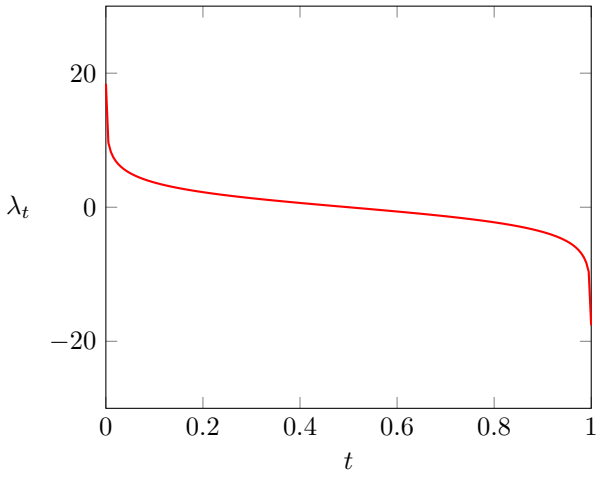

Figure 2.3: Relationship between time t and the log signal-to-noise ratio λ_t for the truncated continuous-time α-cosine noise schedule $f_\lambda(t)$ as defined in Equation 2.53 with $\lambda_{\min} = -30$ and $\lambda_{\max} = 30$. The horizontal axis is time $t \in [0, 1]$; the vertical axis is $\lambda_t = f_\lambda(t) \in [\lambda_{\min}, \lambda_{\max}] = [-30, 30]$.

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

$$\alpha_t = \sqrt{S(\lambda_t)} \quad = \cos\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0)\right) \tag{2.56}$$

$$\sigma_t = \sqrt{S(-\lambda_t)} = \sin\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0)\right) \tag{2.57}$$

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.

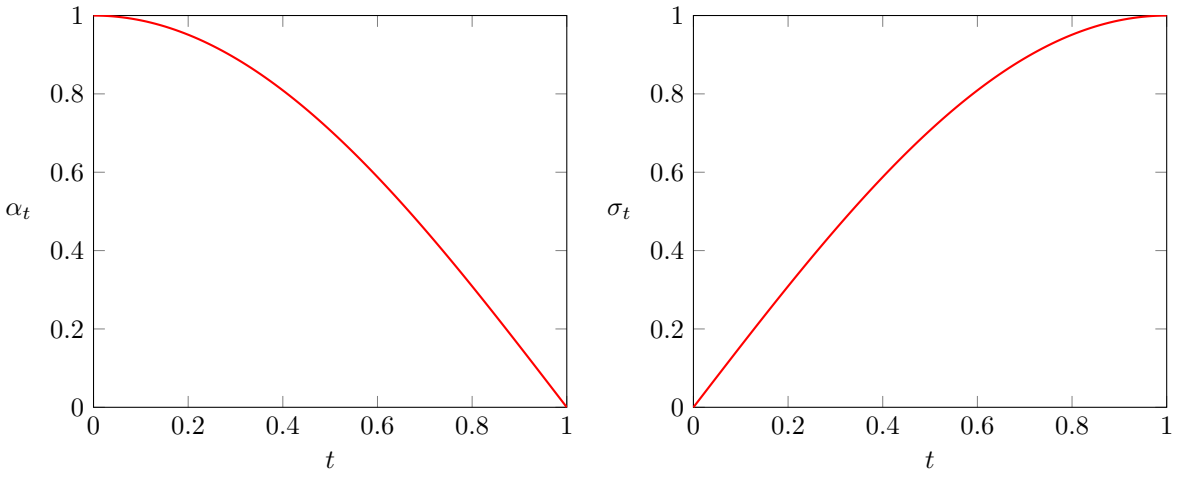

Figure 2.4: Relationship between time t and α_t (left) and σ_t (right) for the same truncated continuous-time α-cosine noise schedule as that in Figure 2.3. The horizontal axis is time $t \in [0, 1]$; the vertical axis is the value of α_t (left) and σ_t (right).

We can, in theory, use two different noise schedules: one to train the model and another to generate new samples. During training, the noise schedule affects the variance of the gradients [7]. During generation, we typically want to use a noise schedule that optimises the quality of the generated samples. However, in this work, we use the truncated continuous-time α-cosine schedule for training and generation, as we found it to provide empirically good results for both. During training, we sampling $t \sim \mathcal{U}(0, 1)$ uniformly at random, then compute $\lambda = f_\lambda(t)$, which equates to sampling $\lambda \sim p_\Lambda(\lambda)$, where $p_\Lambda(\lambda)$ is the probability density function for the truncated continuous-time α-cosine schedule, and given by:

$$p_\Lambda(\lambda) = \frac{1}{2\pi(t_1 - t_0)} \operatorname{sech}\left(\frac{\lambda}{2}\right) \tag{2.58}$$

Figure 2.5 displays the probability density function.

2.7.4 Parameterisations

In Section 2.7.2, 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 [3]. Recent diffusion models have instead adopted different parameterisations, most commonly the $\boldsymbol{\epsilon}$-prediction parameterisation (see e.g. [3, 4, 11]), 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:

$$\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t) = \frac{1}{\alpha_t}\left(\mathbf{z}_t - \sigma_t \hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t, \lambda_t)\right) \tag{2.59}$$

In this work, we employ the $\mathbf{v}$-prediction parameterisation, introduced originally by Salimans and Ho [12], and commonly employed in video diffusion models (see e.g. [5, 2]). The $\mathbf{v}$-prediction parameterisation was introduced initially to facilitate progressive distillation for faster sampling, though we utilise it here for its additional benefits highlighted by Ho et al. [2], namely faster convergence of sample quality and prevention of temporal colour shifting sometimes observed with $\boldsymbol{\epsilon}$-prediction video diffusion models.

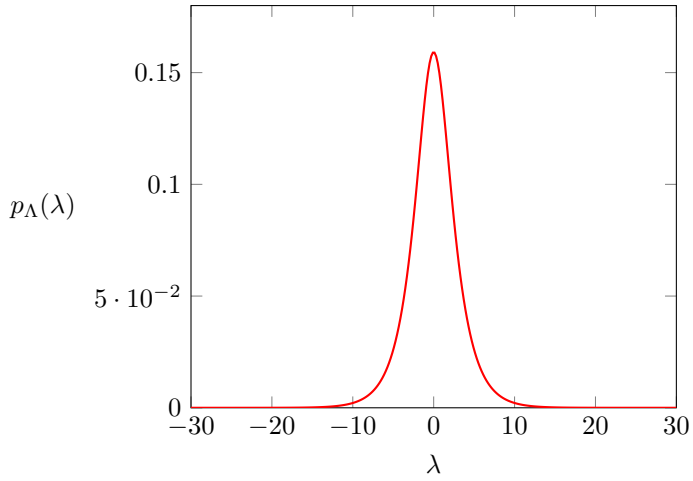

Figure 2.5: Probability density function $p_\Lambda(\lambda)$ for the same truncated continuous-time α-cosine schedule as that in Figure 2.3. The horizontal axis is the log signal-to-noise ratio $\lambda \in [\lambda_{\min}, \lambda_{\max}]$; the vertical axis is the corresponding probability density p_Λ.

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:

$$\mathbf{v}_t = \alpha_t \boldsymbol{\epsilon} - \sigma_t \mathbf{x} \tag{2.60}$$

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:

$$\mathbb{E}_{\lambda \sim p_\Lambda(\lambda), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\|\mathbf{v}_t - \hat{\mathbf{v}}_\theta(\mathbf{z}_t, \lambda_t)\|_2^2 \right] \tag{2.61}$$

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:

$$\hat{\mathbf{x}}(\mathbf{z}_t, \lambda_t) = \alpha_t \mathbf{z}_t - \sigma_t \hat{\mathbf{v}}_\theta(\mathbf{z}_t, \lambda_t) \tag{2.62}$$

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

2.7.5 Score-Based Interpretation

Suppose we have a multivariate Gaussian variable with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$:

$$\mathbf{z} \sim p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\mathbf{z}, \boldsymbol{\Sigma}_\mathbf{z}) \tag{2.63}$$

Tweedie's formula states that:

$$\mathbb{E}\left[\boldsymbol{\mu}_\mathbf{z} | \mathbf{z}\right] = \mathbf{z} + \boldsymbol{\Sigma}_\mathbf{z} \nabla_\mathbf{z} \log p(\mathbf{z}) \tag{2.64}$$

As such, for a given latent:

$$\mathbf{z}_t \sim q() \tag{2.65}$$

$\mathbf{z}_t \sim \mathcal{N}(\mathbf{z}_t; \alpha_t \mathbf{x}, \sigma_t^2 \mathbf{I})$, the expected value of the mean $\boldsymbol{\mu}_{\mathbf{z}_t}$ given $\mathbf{z}_t$ is given by:

$$\mathbb{E}\left[\boldsymbol{\mu}_{\mathbf{z}_t} | \mathbf{z}_t\right] = \mathbf{z}_t + \sigma_t^2 \nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t) \tag{2.66}$$

From Equation 2.40, we have $\boldsymbol{\mu}_{\mathbf{z}_t} = \alpha_t \mathbf{x}$. Thus, we can reformulate the score of $q(\mathbf{z}_t)$ in terms of $\mathbf{z}_t$ and $\mathbf{x}$ as:

$$\alpha_t \mathbf{x} = \mathbf{z}_t + \sigma_t^2 \nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t) \tag{2.67}$$

2.7.6 ELBO for Diffusion Models

We can interpret diffusion models as a special case of Markovian hierarchical VAEs, with several notable restrictions. Namely, the dimensionality of each latent $\mathbf{z}_t$ must be equal to the dimensionality of the observed variable $\mathbf{x}$; we pre-define $q(\mathbf{z}_t|\mathbf{z}_s)$ where $0 \leq s < t \leq 1$ as a Gaussian diffusion process with no learnable inference parameters; and, the distribution of the final latent $\mathbf{z}_t$ is approximately the multivariate standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$, and thus holds no information about the observed variable $\mathbf{x}$. Much like the VAE, the original diffusion model introduced by Sohl-Dickstein et al. [13] was trained by optimising the ELBO—equivalently minimising the negative ELBO—which we can derive into three constituent terms: a reconstruction loss, a prior loss and a diffusion loss:

$$
\begin{aligned}
-\log p_\theta(\mathbf{x}) &\leq -\text{ELBO}(\mathbf{x}) && (2.68)\\
&= \mathbb{E}_{\mathbf{z}_0,\dots\mathbf{z}_1\sim q(\mathbf{z}_0,\dots,\mathbf{z}_1|\mathbf{x})}\left[-\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z}_0,\dots,\mathbf{z}_1)}{q(\mathbf{z}_0,\dots,\mathbf{z}_1|\mathbf{x})}\right)\right] && (2.69)\\
&= \mathbb{E}_{\mathbf{z}_0\sim q(\mathbf{z}_0|\mathbf{x})}\left[-\log p_\theta(\mathbf{x}|\mathbf{z}_0)\right] + D_{KL}(q(\mathbf{z}_0,\dots,\mathbf{z}_1|\mathbf{x})\|p_\theta(\mathbf{z}_0,\dots,\mathbf{z}_1)) && (2.70)\\
&= \underbrace{\mathbb{E}_{\mathbf{z}_0\sim q(\mathbf{z}_0|\mathbf{x})}\left[-\log p_\theta(\mathbf{x}|\mathbf{z}_0)\right]}_{\text{Reconstruction Loss}} + \underbrace{D_{KL}(q(\mathbf{z}_1|\mathbf{x})\|p_\theta(\mathbf{z}_1))}_{\text{Prior Loss}} + \underbrace{\frac{1}{2}\int_{\lambda_{\min}}^{\lambda_{\max}} \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\|\boldsymbol{\epsilon}-\hat{\boldsymbol{\epsilon}}(\mathbf{z}_t,\lambda_t)\|_2^2\right] d\lambda}_{\text{Diffusion Loss}} && (2.71)
\end{aligned}
$$

With sufficiently large $\lambda_{\max}$, the reconstruction loss is approximately zero since we can almost perfectly reconstruct $\mathbf{x}$ from $\mathbf{z}_0$—this is particularly true for discrete $\mathbf{x}$. Mathematically, as $\lambda_{\max} \to \infty$, we have:

$$\lim_{\lambda_{\max}\to\infty} q(\mathbf{z}_0|\mathbf{x}) = \delta(\mathbf{z}_0 - \mathbf{x}) \tag{2.72}$$

where δ is the Dirac delta distribution, in which case $\mathbf{z}_0 = \mathbf{x}$. Similarly, with sufficiently small $\lambda_{\min}$, the prior loss is approximately zero; as $\lambda_{\min} \to -\infty$, we have:

$$\lim_{\lambda_{\min}\to-\infty} q(\mathbf{z}_1|\mathbf{x}) = \mathcal{N}(\mathbf{0},\mathbf{I}) = p_\theta(\mathbf{z}_1) \tag{2.73}$$

so the KL divergence prior loss term likewise approaches zero.

2.7.7 Weighted Loss

Most diffusion models in the broader literature—including state-of-the-art models—are not optimised using the ELBO. Kingma and Gao [7] showed that the various objectives used are all special cases of a *weighted loss*, which is defined per datapoint $\mathbf{x}$ as:

$$\mathcal{L}_w = w(\lambda_{\min})\mathcal{L}(\lambda_{\min}) + \int_{\lambda_{\min}}^{\lambda_{\max}} w(\lambda)\mathcal{L}'(\lambda)d\lambda \tag{2.74}$$

where $w(\lambda)$ is a weighting function and $\mathcal{L}(\lambda)$ is the KL divergence of $q(\mathbf{z}_t,\dots,\mathbf{z}_1|\mathbf{x})$ from $p_\theta(\mathbf{z}_t,\dots,\mathbf{z}_1)$ for a subset of timesteps from $t = f_\lambda^{-1}(\lambda)$ to 1 for datapoint $\mathbf{x}$:

$$\mathcal{L}(\lambda) = D_{KL}(q(\mathbf{z}_t,\dots,\mathbf{z}_1|\mathbf{x})\|p_\theta(\mathbf{z}_t,\dots,\mathbf{z}_1)) \tag{2.75}$$

To maintain consistency with Kingma and Gao [7], we refer to $\mathcal{L}'(\lambda)$ as the *time derivative*, despite actually being the derivative of $\mathcal{L}(\lambda)$ with respect to the log signal-to-noise ratio; it is given by:

$$\mathcal{L}'(\lambda) = \frac{d}{d\lambda}\mathcal{L}(\lambda) = \frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\|\boldsymbol{\epsilon}-\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t,\lambda)\|_2^2\right] \tag{2.76}$$

Substituting the form of $\mathcal{L}(\lambda)$ and $\mathcal{L}'(\lambda)$ and into $\mathcal{L}_w$ gives us:

$$\mathcal{L}_w = w(\lambda_{\min})D_{KL}(q(\mathbf{z}_1|\mathbf{x})\|p_\theta(\mathbf{z}_1)) + \frac{1}{2}\int_{\lambda_{\min}}^{\lambda_{\max}} w(\lambda)\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\|\boldsymbol{\epsilon}-\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t,\lambda)\|_2^2\right] d\lambda \tag{2.77}$$

This form provides several useful insights. We can see that the first term, the weighted prior loss, contains no learnable parameters. Thus, minimisation of the weighted loss $\mathcal{L}_w$ equates to minimisation of

the intractable integral. In practice, we minimise the integral via an importance-weighted Monte Carlo integrator:

$$\int_{\lambda_{\min}}^{\lambda_{\max}} w(\lambda)\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\|\boldsymbol{\epsilon}-\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t,\lambda_t)\|_2^2\right]d\lambda = \mathbb{E}_{\lambda\sim p_\Lambda(\lambda)}\left[\frac{w(\lambda)}{p_\Lambda(\lambda)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\|\boldsymbol{\epsilon}-\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t,\lambda)\|_2^2\right]\right] \tag{2.78}$$

$$= \mathbb{E}_{\lambda\sim p_\Lambda(\lambda),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\frac{w(\lambda)}{p_\Lambda(\lambda)}\|\boldsymbol{\epsilon}-\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t,\lambda)\|_2^2\right] \tag{2.79}$$

$$\simeq \frac{w(\lambda)}{p_\Lambda(\lambda)}\|\boldsymbol{\epsilon}-\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t,\lambda)\|_2^2 \tag{2.80}$$

where $p_\Lambda(\lambda)$ is determined by the training noise schedule—we can sample from $p_\Lambda(\lambda)$ by first sampling $t\sim\mathcal{U}(0,1)$, then computing $\lambda = f_\lambda(t)$.

Notably, further analysis by Kingma and Gao [7] shows that the weighted loss also has a likelihood-based interpretation. The fundamental theorem of calculus enables us to write the weighted loss as:

$$\mathcal{L}_w = w(\lambda_{\max})\mathcal{L}(\lambda_{\max}) + \int_{\lambda_{\min}}^{\lambda_{\max}} -w'(\lambda)\mathcal{L}(\lambda)d\lambda \tag{2.81}$$

The likelihood-based interpretation comes from the fact that $\mathcal{L}(\lambda)$ serves as a variational bound on the negative marginal likelihood of the noise-perturbed data $\mathbf{z}_t\sim q(\mathbf{z}_t|\mathbf{x})$, plus a constant; this is given by:

$$\mathcal{L}(\lambda) \geq D_{KL}(q(\mathbf{z}_t|\mathbf{x})\|p(\mathbf{z}_t)) \tag{2.82}$$

$$= \mathbb{E}_{\mathbf{z}_t\sim q(\mathbf{z}_t|\mathbf{x})}\left[-\log p(\mathbf{z}_t)\right] + \mathbb{E}_{\mathbf{z}_t\sim q(\mathbf{z}_t|\mathbf{x})}\left[\log q(\mathbf{z}_t|\mathbf{x})\right] \tag{2.83}$$

As such, minimisation of $\mathcal{L}(\lambda)$ equates to maximisation of the expected log-likelihood of the noise-perturbed data $\mathbf{z}_t\sim q(\mathbf{z}_t|\mathbf{x})$ with noise level λ. If the weighting function $w(\lambda)$ is a monotonically decreasing function of $\lambda\in[\lambda_{\min},\lambda_{\max}]$, then by definition $-w'(\lambda)$ will be positive for all $\lambda\in[\lambda_{\min},\lambda_{\max}]$. In which case, minimisation of $\mathcal{L}_w$ itself equates to maximisation of the weighted expected log-likelihood of the noise-perturbed data $\mathbf{z}_t\sim q(\mathbf{z}_t|\mathbf{x})$ with weighting $-w'(\lambda)$. For this reason, we use a monotonically decreasing weighting function in this work. In contrast, most diffusion models in the broader literature (see e.g. [3, 9, 11]) undergo training with non-monotonic weighting functions. In such cases, for noise levels whereby $-w'(\lambda)$ is negative, the weighted loss has a counterintuitive interpretation of minimisation of the weighted expected log-likelihood of the noise-perturbed data $\mathbf{z}_t\sim q(\mathbf{z}_t|\mathbf{x})$.

An important observation made by Kingma and Gao [7] is that the ELBO in Equation 2.71 is a special case of the weighted loss $\mathcal{L}_w$ with:

$$w(\lambda) = 1 \tag{2.84}$$

The $\mathbf{v}$-prediction parameterisation loss given in Equation 2.61 equates to the weighted loss $\mathcal{L}_w$ with:

$$w(\lambda) = \frac{1}{2\pi(t_1-t_0)}\exp\left(-\frac{\lambda}{2}\right) \tag{2.85}$$

$$-w'(\lambda) = \frac{1}{4\pi(t_1-t_0)}\exp\left(-\frac{\lambda}{2}\right) \tag{2.86}$$

Figure 2.6 shows the weighting function $w(\lambda)$ and the negative of its derivative $-w'(\lambda)$ for the $\mathbf{v}$-parameterisation loss. As evident from the graphs, the weighting function $w(\lambda)$ is a monotonically decreasing function of λ, and as such $-w'(\lambda)$ is positive for all $\lambda\in[\lambda_{\min},\lambda_{\max}]$.

2.7.8 Replacement Sampling

2.7.9 Reconstruction-Guided Sampling

2.7.10 U-Net

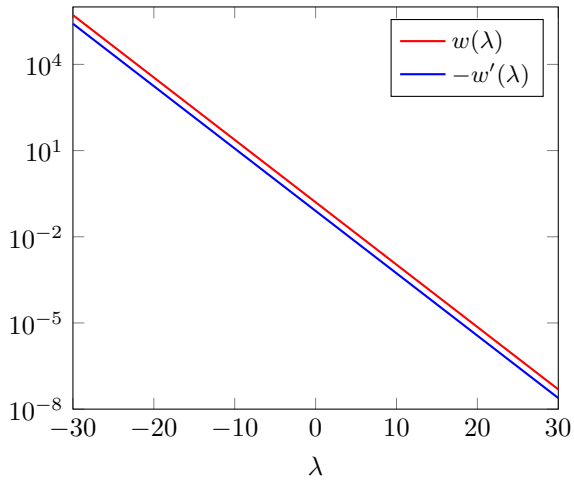

Figure 2.6: Relationship between the log signal-to-noise ratio λ and the functions $w(\lambda)$ and $-w'(\lambda)$ for the $\mathbf{v}$-parameterisation with the same truncated continuous-time α-cosine schedule as that in Figure 2.3. The horizontal axis is the log signal-to-noise level λ; the vertical axis is the value of $w(\lambda)$ and $-w'(\lambda)$; the vertical axis is logarithmic.

Chapter 3

Climate Background

Chapter 4

Results

Chapter 5

Conclusion

Bibliography

[1] Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[2] 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.

[3] 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.

[4] 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.

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

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

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2256–2265. JMLR.org, 2015.

[14] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11895–11907, 2019.

[15] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Appendix A

Diffusion Models

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

From Equation 2.40, 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 [9] as described in [6] 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:

$$\sigma_t^2 = 1 - \alpha_t^2 \tag{A.14}$$

$$= 1 - \cos^2\left(\frac{\pi}{2}t\right) \tag{A.15}$$

$$= \sin^2\left(\frac{\pi}{2}t\right) \tag{A.16}$$

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

$$\tilde{f}_\lambda(t) = \log\left(\frac{\alpha_t^2}{\sigma_t^2}\right) \tag{A.17}$$

$$= \log\left(\frac{\cos^2\left(\frac{\pi}{2}t\right)}{\sin^2\left(\frac{\pi}{2}t\right)}\right) \tag{A.18}$$

$$= -2\log\left(\tan\left(\frac{\pi}{2}t\right)\right) \tag{A.19}$$

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 (see e.g. [6, 5]) 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:

$$f_\lambda(t) = \tilde{f}_\lambda(t_0 + t(t_1 - t_0)) \tag{A.23}$$

$$= -2\log\left(\tan\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0))\right)\right) \tag{A.24}$$

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

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

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

$$\psi_t = \arctan\left(\frac{\sigma_t}{\alpha_t}\right) \tag{A.27}$$

$$= \arctan\left(\frac{\sin\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0))\right)}{\cos\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0))\right)}\right) \tag{A.28}$$

$$= \arctan\left(\tan\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0))\right)\right) \tag{A.29}$$

$$= \frac{\pi}{2}(t_0 + t(t_1 - t_0)) \tag{A.30}$$

when using the truncated continuous-time α-cosine noise schedule as per Section 2.7.3. As such, we can formulate the velocity as:

$$\mathbf{v}_t = \frac{\mathbf{z}_t}{d\psi} = \frac{d\cos(\psi)}{d\psi}\mathbf{x} + \frac{d\sin(\psi)}{d\psi}\boldsymbol{\epsilon} \tag{A.31}$$

$$= -\sin(\psi)\mathbf{x} + \cos(\psi)\boldsymbol{\epsilon} \tag{A.32}$$

$$= \alpha_t\boldsymbol{\epsilon} - \sigma_t\mathbf{x} \tag{A.33}$$

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

$$\mathbf{v}_t = -\sin(\psi)\mathbf{x} + \cos(\psi)\boldsymbol{\epsilon} \tag{A.34}$$

$$\sin(\psi)\mathbf{x} = \cos(\psi)\boldsymbol{\epsilon} - \mathbf{v}_t \tag{A.35}$$

$$= \cos(\psi)\left(\frac{\mathbf{z}_t - \cos(\psi)\mathbf{x}}{\sin(\psi)}\right) - \mathbf{v}_t \tag{A.36}$$

$$\sin^2(\psi)\mathbf{x} = \cos(\psi)\mathbf{z}_t - \cos^2(\psi)\mathbf{x} - \sin(\psi)\mathbf{v}_t \tag{A.37}$$

$$\sin^2(\psi)\mathbf{x} + \cos^2(\psi)\mathbf{x} = \cos(\psi)\mathbf{z}_t - \sin(\psi)\mathbf{v}_t \tag{A.38}$$

$$(\sin^2(\psi) + \cos^2(\psi))\mathbf{x} = \cos(\psi)\mathbf{z}_t - \sin(\psi)\mathbf{v}_t \tag{A.39}$$

$$\mathbf{x} = \cos(\psi)\mathbf{z}_t - \sin(\psi)\mathbf{v}_t \tag{A.40}$$

$$= \alpha_t\mathbf{z}_t - \sigma_t\mathbf{v}_t \tag{A.41}$$

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:

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