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.

---

Sunday 23rd 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, Sunday 23rd 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 $\theta$. We learn parameters $\theta$ 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 $\theta$ 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. [14, 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 2.7.10.

## 2.3 Latent Variables

We can think of each observed datapoint $\mathbf{x} \in \mathcal{D}$ as being represented or generated via $N_{\mathbf{z}} \geq 1$ associated *latent variables*:

$$\{\mathbf{z}_1, \ldots, \mathbf{z}_{N_{\mathbf{z}}}\} = \{\mathbf{z}_i \mid i \leq 1 \leq N_{\mathbf{z}}, i \in \mathbb{N}\} \tag{2.4}$$

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 and the latent variables by $p_\theta(\mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_{N_{\mathbf{z}}})$; the marginal distribution over the observed variable $p_\theta(\mathbf{x})$ is given by:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_{N_{\mathbf{z}}}) d\mathbf{z} \tag{2.5}$$

In the context of a latent-variable model, *generation* refers to the process of sampling the latent variables from the joint distribution $p_\theta(\mathbf{z}_1, \ldots, \mathbf{z}_{N_{\mathbf{z}}})$, and then sampling the observed variable from the conditional distribution $p_\theta(\mathbf{x}|\mathbf{z}_1, \ldots, \mathbf{z}_{N_{\mathbf{z}}})$. In the simplest case, we may only have a single latent variable; we omit the index 1 in such cases for notational simplicity.

## 2.4 Likelihood-Based Generative Models

As mentioned in Section 2.1, the goal of a generative model is to learn parameters $\theta$ 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 $\theta$ that minimise the Kullback–Leibler (KL) divergence of the true distribution $p^*(\mathbf{x})$ from our model distribution $p_\theta(\mathbf{x})$:

$$\arg\min_\theta \; D_{KL}(p^*(\mathbf{x}) \| p_\theta(\mathbf{x})) \tag{2.6}$$

The KL divergence $D_{KL}$ 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:

$$\begin{aligned} D_{KL}(p^*(\mathbf{x}) \| p_\theta(\mathbf{x})) &= \mathbb{E}_{\mathbf{x} \sim p^*(\mathbf{x})} \left[ \log \left( \frac{p^*(\mathbf{x})}{p_\theta(\mathbf{x})} \right) \right] && (2.7) \\ &= \mathbb{E}_{\mathbf{x} \sim p^*(\mathbf{x})} \left[ \log p^*(\mathbf{x}) \right] + \mathbb{E}_{\mathbf{x} \sim p^*(\mathbf{x})} \left[ -\log p_\theta(\mathbf{x}) \right] && (2.8) \\ &= -\mathcal{H}(p^*(\mathbf{x})) + \mathbb{E}_{\mathbf{x} \sim p^*(\mathbf{x})} \left[ -\log p_\theta(\mathbf{x}) \right] && (2.9) \end{aligned}$$

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 $\mathbf{x} \sim p^*(\mathbf{x})$; formally:

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

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:

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

In other words, under the i.i.d assumption of $\mathcal{D}$, the mean negative log-likelihood of our model with respect to $\mathcal{D}$ serves as an unbiased estimator of the expected negative 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:

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

As such, by transitivity, the mean negative log-likelihood of our model with respect to each mini-batch $\mathcal{M}$ is itself an unbiased estimator of the expected negative 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 minimise the expected negative 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 Components of the Basic Variational Autoencoder

The variational autoencoder (VAE) [8, 13] is an important example of a likelihood-based generative model. 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 $\mathbf{x}$ 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 $p^*(\mathbf{x}|\mathbf{z})$. Thus, our model $p_\theta(\mathbf{x}, \mathbf{z})$ we seek to optimise such that $p_\theta(\mathbf{x}) \approx p^*(\mathbf{x})$ takes the following factorised form:

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

where, naturally, we need to specify our two distributions: $p_\theta(\mathbf{z})$ and $p_\theta(\mathbf{x}|\mathbf{z})$. We refer to $p_\theta(\mathbf{z})$ as the prior over $\mathbf{z}$, and one common choice is the standard Gaussian:

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

Furthermore, we refer to $p_\theta(\mathbf{x}|\mathbf{z})$ as the stochastic *decoder* since given a latent $\mathbf{z}$ it produces a distribution over the possible corresponding values of $\mathbf{x}$. As an example, we can select $p_\theta(\mathbf{x}|\mathbf{z})$ be a multivariate Gaussian with diagonal covariance:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_\theta(\mathbf{z}), \mathrm{diag}(\boldsymbol{\sigma}_\theta(\mathbf{z}))^{\circ 2}) \tag{2.15}$$

where $\boldsymbol{\mu}_\theta(\mathbf{z})$ and $\boldsymbol{\sigma}_\theta(\mathbf{z})$ are outputs from a neural network with parameters $\theta$. One final, crucial defining feature of the VAE is the stochastic *encoder* $q_\phi(\mathbf{z}|\mathbf{x})$, also referred to as the *inference model*, with variational parameters $\phi$. The stochastic encoder $q_\phi(\mathbf{z}|\mathbf{x})$ approximates the intractable posterior $p_\theta(\mathbf{z}|\mathbf{x})$ of the generative model. Again, a common choice is for $q_\phi(\mathbf{z}|\mathbf{x})$ to be a multivariate Gaussian with diagonal covariance:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}, \boldsymbol{\mu}_\phi(\mathbf{z}), \mathrm{diag}(\boldsymbol{\sigma}_\phi(\mathbf{z}))^{\circ 2}) \tag{2.16}$$

Figure 2.1 provides a graphical depiction of the VAE.

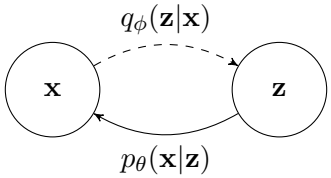



Figure 2.1: Graphical depiction of basic VAE with one observed variable $\mathbf{x}$ and one latent variable $\mathbf{z}$. Solid lines depict the Bayesian network of the generative model; dashed lines depict the Bayesian network of the approximate inference model.

### 2.5.2 Evidence Lower Bound Objective

The VAE falls into the broad class of likelihood-based generative models. However, the likelihood $p_\theta(\mathbf{x})$ cannot be optimised directly, as the VAE model does not make common simplifying assumptions about marginal and posterior probabilities. Notably, we assume the model's marginal likelihood $p(\mathbf{x})$ given by:

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

does not have an analytic solution or efficient estimator. In addition, we assume the model's posterior density $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable, so we cannot employ the expectation-maximisation algorithm.

Not making these simplifying assumptions is the reason for introducing the inference model $q_\phi(\mathbf{z}|\mathbf{x})$ to approximate the model's intractable posterior $p_\theta(\mathbf{z}|\mathbf{x})$. With the introduction of the inference model, we can derive a variational bound on the negative log-likelihood:

$$\begin{aligned}
-\log p_\theta(\mathbf{x}) &= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[-\log p_\theta(\mathbf{x})\right] && (2.18)\\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[-\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right)\right] && (2.19)\\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[-\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})q_\phi(\mathbf{z}|\mathbf{x})}\right)\right] && (2.20)\\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[-\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right)\right] - \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{q(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right)\right] && (2.21)\\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[-\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right)\right] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) && (2.22)
\end{aligned}$$

The second term in Equation 2.22 is the KL divergence of $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{z}|\mathbf{x})$ and is non-negative:

$$D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) \geq 0 \tag{2.23}$$

and zero if and only if $q_\phi(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}|\mathbf{x})$. The first term in Equation 2.22 is the additive inverse of the *evidence lower bound objective* (ELBO); in this work, we will refer to this as the ELBO loss, denoted $\mathcal{L}_{\text{ELBO}}$. By the non-negativity of the KL divergence, it serves as a variational bound on the negative log-likelihood of the observed variable $\mathbf{x}$:

$$\begin{aligned}
-\log p_\theta(\mathbf{x}) &\leq \mathcal{L}_{\text{ELBO}}(\mathbf{x}) && (2.24)\\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[-\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right)\right] && (2.25)
\end{aligned}$$

As such, minimisation of the ELBO loss accomplishes two things. Firstly, it will approximately minimise the negative 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 Markovian Hierarchical Variational Autoencoders

The Markovian hierarchical variational autoencoder (MHVAE) [9, 17] is a versatile extension of the VAE, accommodating an unrestricted number $N_{\mathbf{z}} \geq 1$ of latent variables. Notably, the joint distribution of the observed variable and the latent variables is Markovian:

$$p_\theta(\mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_{N_{\mathbf{z}}}) = p(\mathbf{x}|\mathbf{z}_1) p_\theta(\mathbf{z}_{N_{\mathbf{z}}}) \prod_{i=1}^{N_{\mathbf{z}}-1} p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1}) \tag{2.26}$$

Figure 2.2 illustrates the MHVAE model. For the generative model, the observed variable $\mathbf{x}$ is conditionally independent of $\mathbf{z}_2$ and $\mathbf{z}_3$ given $\mathbf{z}_1$. Similarly, $\mathbf{z}_1$ is conditionally independent of $\mathbf{z}_3$ given $\mathbf{z}_2$.

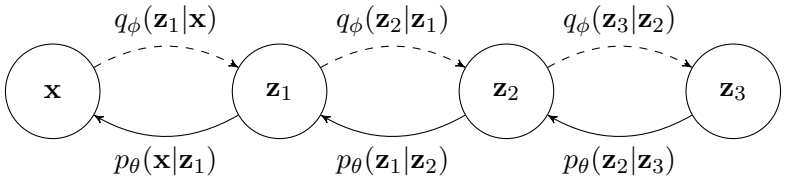



Figure 2.2: Graphical depiction of Markovian hierarchical VAE with one observed variable $\mathbf{x}$ and three latent variables $\mathbf{z}_1$, $\mathbf{z}_2$ and $\mathbf{z}_3$. Solid lines depict the Bayesian network of the generative model; dashed lines depict the Bayesian network of the approximate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

### 2.5.4 Infinitely Deep Markovian Hierarchical Variational Autoencoders

In the limit of $N_{\mathbf{z}} \to \infty$, we instead notationally write our latent variables in terms of a continuous-time variable $t \in [0, 1]$ as:

$$\{\mathbf{z}_0, \ldots, \mathbf{z}_1\} = \{\mathbf{z}_t \mid t \in [0, 1]\} \tag{2.27}$$

We can formulate the ELBO loss for the continuous-time MHVAE as follows:

$$-\log p_\theta(\mathbf{x}) \leq \mathcal{L}_{\text{ELBO}}(\mathbf{x}) \tag{2.28}$$

$$= \mathbb{E}_{\mathbf{z}_0,\ldots,\mathbf{z}_1 \sim q_\phi(\mathbf{z}_0,\ldots,\mathbf{z}_1|\mathbf{x})} \left[ -\log \left( \frac{p_\theta(\mathbf{x}, \mathbf{z}_0, \ldots, \mathbf{z}_1)}{q_\phi(\mathbf{z}_0, \ldots, \mathbf{z}_1|\mathbf{x})} \right) \right] \tag{2.29}$$

$$= \mathbb{E}_{\mathbf{z}_0 \sim q(\mathbf{z}_0,\mathbf{x})} \left[ -\log p_\theta(\mathbf{x}|\mathbf{z}_0) \right] + D_{KL}(q(\mathbf{z}_0, \ldots, \mathbf{z}_1|\mathbf{x}) \| p_\theta(\mathbf{z}_0, \ldots, \mathbf{z}_1)) \tag{2.30}$$

Per datapoint $\mathbf{x}$, we define $\mathcal{L}_T(t)$ as the KL divergence of $q_\phi(\mathbf{z}_t, \ldots, \mathbf{z}_1|\mathbf{x})$ from $p_\theta(\mathbf{z}_t, \ldots, \mathbf{z}_1)$ for a subset of timesteps from $t$ to 1, and its corresponding time derivative $\mathcal{L}'_T(t)$ as:

$$\mathcal{L}_T(t) = D_{KL}(q_\phi(\mathbf{z}_t, \ldots, \mathbf{z}_1|\mathbf{x}) \| p_\theta(\mathbf{z}_t, \ldots, \mathbf{z}_1)) \tag{2.31}$$

$$\mathcal{L}'_T(t) = \frac{d}{dt} \mathcal{L}_T(t) \tag{2.32}$$

We can substitute this into the ELBO loss and use the second fundamental theorem of calculus to yield the following form for the ELBO loss, defined per datapoint $\mathbf{x}$ as:

$$\mathcal{L}_{\text{ELBO}}(\mathbf{x}) = \mathbb{E}_{\mathbf{z}_0 \sim q(\mathbf{z}_0,\mathbf{x})} \left[ -\log p_\theta(\mathbf{x}|\mathbf{z}_0) \right] + D_{KL}(q(\mathbf{z}_0, \ldots, \mathbf{z}_1|\mathbf{x}) \| p_\theta(\mathbf{z}_0, \ldots, \mathbf{z}_1)) \tag{2.33}$$

$$= \mathbb{E}_{\mathbf{z}_0 \sim q(\mathbf{z}_0,\mathbf{x})} \left[ -\log p_\theta(\mathbf{x}|\mathbf{z}_0) \right] + \mathcal{L}_T(0) \tag{2.34}$$

$$= \mathbb{E}_{\mathbf{z}_0 \sim q(\mathbf{z}_0,\mathbf{x})} \left[ -\log p_\theta(\mathbf{x}|\mathbf{z}_0) \right] + \mathcal{L}_T(1) - \int_0^1 \mathcal{L}'_T(t) \mathrm{d}t \tag{2.35}$$

This form for the ELBO may seem unconventional. However, we introduce it here to motivate a strong theoretical link between the ELBO and the weighted loss [7], which is used to train diffusion models in the broader literature. We explore the weighted loss in Section 2.7.8.

## 2.6 Score-Based Generative Models

Score-based generative models [18, 19] are an alternative to likelihood-based generative models. We define the *score* of a probability density $p(\mathbf{x})$ to be:

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}) \tag{2.36}$$

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

$$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p^*(\mathbf{x}) \tag{2.37}$$

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

## 2.7 Diffusion Models

### 2.7.1 Overview

Diffusion models [16, 3, 19] are a framework for generative modelling. Diffusion models consist of a *forward diffusion process* that transforms a datapoint into noise, and a *reverse-time generative model* able to transform noise back into a datapoint. As we explore in this section, diffusion models have both likelihood and score-based interpretations.

### 2.7.2 Forward Diffusion Process

Specified in continuous time, the *forward diffusion process* is a Gaussian diffusion process that defines the model's latent variables as a sequence of increasingly noisy versions of $\mathbf{x}$:

$$\{\mathbf{z}_0, \ldots, \mathbf{z}_1\} = \{\mathbf{z}_t \mid t \in [0, 1]\} \tag{2.38}$$

An Itô stochastic differential equation (SDE) defines the time evolution of the diffusion process [19]:

$$d\mathbf{z}_t = \mathbf{f}(\mathbf{z}_t, t)dt + g(t)d\mathbf{w}_t \tag{2.39}$$

where $\mathbf{w}_t$ is the standard Wiener process (i.e. Brownian motion); $\mathbf{f}(\mathbf{z}_t, t) : \mathbb{R}^D \to \mathbb{R}^D$ is a vector-valued function called the *drift* coefficient of $\mathbf{z}$; $g(t) : \mathbb{R} \to \mathbb{R}$ is a scalar-function known as the *diffusion* coefficient of $\mathbf{z}_t$; and $D$ is the dimensionality of our input data. In this work, we use a *variance-preserving* diffusion model, which we define by the following drift and diffusion coefficients:

$$\mathbf{f}(\mathbf{z}_t, t) = -\frac{1}{2}\left(\frac{d}{dt}\log\left(1 + \exp(-\lambda_t)\right)\right)\mathbf{z}_t \tag{2.40}$$

$$g(t)^2 = \frac{d}{dt}\log\left(1 + \exp(-\lambda_t)\right) \tag{2.41}$$

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 $\lambda_t$ in Section 2.7.4. 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 the observed variable $\mathbf{x}$ is given by:

$$q(\mathbf{z}_t|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}_t; \alpha_t\mathbf{x}, \sigma_t^2\mathbf{I}\right) \tag{2.42}$$

where $\alpha_t$ and $\sigma_t$ are functions of $\lambda_t$, such that:

$$\alpha_t^2 = S(\lambda_t) \tag{2.43}$$
$$\sigma_t^2 = S(-\lambda_t) \tag{2.44}$$

where $S$ is the sigmoid function. 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, and thus:

$$q(\mathbf{z}_t|\mathbf{z}_s, \mathbf{z}_r) = q(\mathbf{z}_t|\mathbf{z}_s) = \mathcal{N}\left(\mathbf{z}_t; \alpha_{t|s}\mathbf{z}_s, \sigma_{t|s}^2\mathbf{I}\right) \tag{2.45}$$

where $\alpha_{t|s}$ and $\sigma_{t|s}$ are given by:

$$\alpha_{t|s} = \frac{\alpha_t}{\alpha_s} \tag{2.46}$$
$$\sigma_{t|s}^2 = \sigma_t^2 - \alpha_{t|s}^2\sigma_s^2 \tag{2.47}$$

A full derivation of $q(\mathbf{z}_t|\mathbf{z}_s)$ is given in Appendix A.1. 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 the following joint distribution:

$$q(\mathbf{z}_0, \ldots, \mathbf{z}_1) = \int p^*(\mathbf{x})q(\mathbf{z}_0, \ldots, \mathbf{z}_1|\mathbf{x})d\mathbf{x} \tag{2.48}$$

### 2.7.3 Reverse-Time Generative Model

Anderson [1] shows that the forward SDE of Equation 2.39 is exactly solved by a second diffusion process, running backwards in time and given by the reverse-time SDE:

$$d\mathbf{z}_t = \left[\mathbf{f}(\mathbf{z}_t, t) - g(t)^2\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t)\right]dt + g(t)d\bar{\mathbf{w}}_t \tag{2.49}$$

where $\bar{\mathbf{w}}_t$ is a standard Wiener process when time flows backwards. Let $\mathbf{s}_\theta(\mathbf{z}_t, \lambda_t)$ be a $\lambda_t$-dependent score network [18] that approximates the score of $q(\mathbf{z}_t)$ such that:

$$\mathbf{s}_\theta(\mathbf{z}_t, \lambda_t) \approx \nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t) \tag{2.50}$$

Thus, 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 reverse-time SDE is exactly:

$$d\mathbf{z}_t = \left[\mathbf{f}(\mathbf{z}_t, t) - g(t)^2\mathbf{s}_\theta(\mathbf{z}_t, \lambda_t)\right]dt + g(t)d\bar{\mathbf{w}}_t \tag{2.51}$$

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:

$$p_\theta(\mathbf{z}_0, \ldots, \mathbf{z}_1) = p_\theta(\mathbf{z}_1)\prod_{i=1}^{T} p_\theta(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)}) \tag{2.52}$$

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:

$$D_{KL}(q(\mathbf{z}_1|\mathbf{x})\|p_\theta(\mathbf{z}_1)) \approx 0 \tag{2.53}$$

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

$$p_\theta(\mathbf{z}_1) = \mathcal{N}(\mathbf{z}_1; \mathbf{0}, \mathbf{I}) \tag{2.54}$$

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. [19]. More formally, from a given latent $\mathbf{z}_t$ we generate $\mathbf{z}_s \sim p_\theta(\mathbf{z}_s|\mathbf{z}_t)$ via:

$$p_\theta(\mathbf{z}_s|\mathbf{z}_t) = q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x} = \hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t)) \tag{2.55}$$

$$= \mathcal{N}\left(\tilde{\boldsymbol{\mu}}_{s|t}(\mathbf{z}_t, \mathbf{x} = \hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t)), \tilde{\sigma}_{s|t}\mathbf{I}\right) \tag{2.56}$$

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 $\lambda_t$, and

$$\tilde{\boldsymbol{\mu}}_{s|t}(\mathbf{z}_t, \mathbf{x}) = \frac{\alpha_{t|s}\sigma_s^2}{\sigma_t^2}\mathbf{z}_t + \frac{\alpha_s \sigma_{t|s}^2}{\sigma_t^2}\mathbf{x} \tag{2.57}$$

$$\tilde{\sigma}_{s|t} = \frac{\sigma_{t|s}\sigma_s}{\sigma_t} \tag{2.58}$$

Interestingly, in Equation 2.55 we introduced a denoiser network $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t)$ to define the discrete-time ancestral sampler, while in Equation 2.50 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 [15]; we describe this in detail in Section 2.7.5.

### 2.7.4 Noise Schedule

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

$$\lambda_t = \log\left(\frac{\alpha_t^2}{\sigma_t^2}\right) \in [\lambda_{\min}, \lambda_{\max}] \tag{2.59}$$

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

In this work, we use a truncated continuous-time version of the $\alpha$-cosine schedule [12], introduced in its original discrete-time form by Nichol and Dhariwal [12]. The $\alpha$-cosine schedule was motivated by the fact that the 'linear' schedule introduced in prior work by Ho et al. [3] causes $\alpha_t$ 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 $\alpha$-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:

$$\lambda_t = f_\Lambda(t) = -2\log\left(\tan\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0))\right)\right) \tag{2.60}$$

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:

$$t_0 = \frac{2}{\pi}\arctan\left(\exp\left(-\frac{1}{2}\lambda_{\max}\right)\right) \tag{2.61}$$

$$t_1 = \frac{2}{\pi}\arctan\left(\exp\left(-\frac{1}{2}\lambda_{\min}\right)\right) \tag{2.62}$$

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 $\alpha$-cosine schedule detailed above.

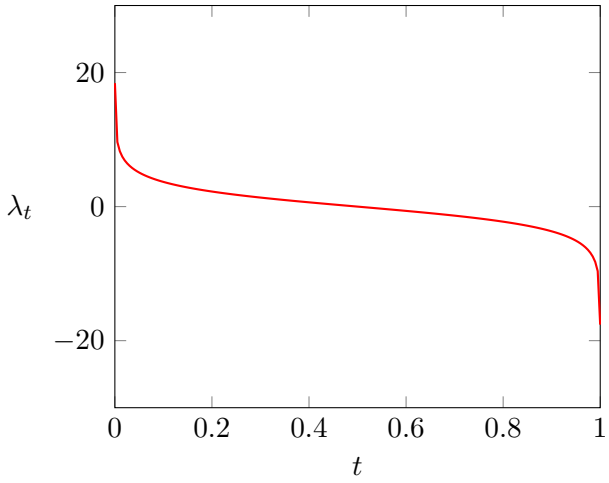



Figure 2.3: Relationship between time $t$ and the log signal-to-noise ratio $\lambda_t$ for the truncated continuous-time $\alpha$-cosine noise schedule $f_\Lambda(t)$ as defined in Equation 2.60 with $\lambda_{\min} = -30$ and $\lambda_{\max} = 30$. The horizontal axis is time $t \in [0, 1]$; the vertical axis is $\lambda_t = f_\Lambda(t) \in [\lambda_{\min}, \lambda_{\max}] = [-30, 30]$.

We can compute $\alpha_t$ and $\sigma_t$ from either $\lambda_t$ or $t$ via the following equations:

$$\alpha_t = \sqrt{S(\lambda_t)} \quad = \cos\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0))\right) \tag{2.63}$$

$$\sigma_t = \sqrt{S(-\lambda_t)} = \sin\left(\frac{\pi}{2}(t_0 + t(t_1 - t_0))\right) \tag{2.64}$$

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

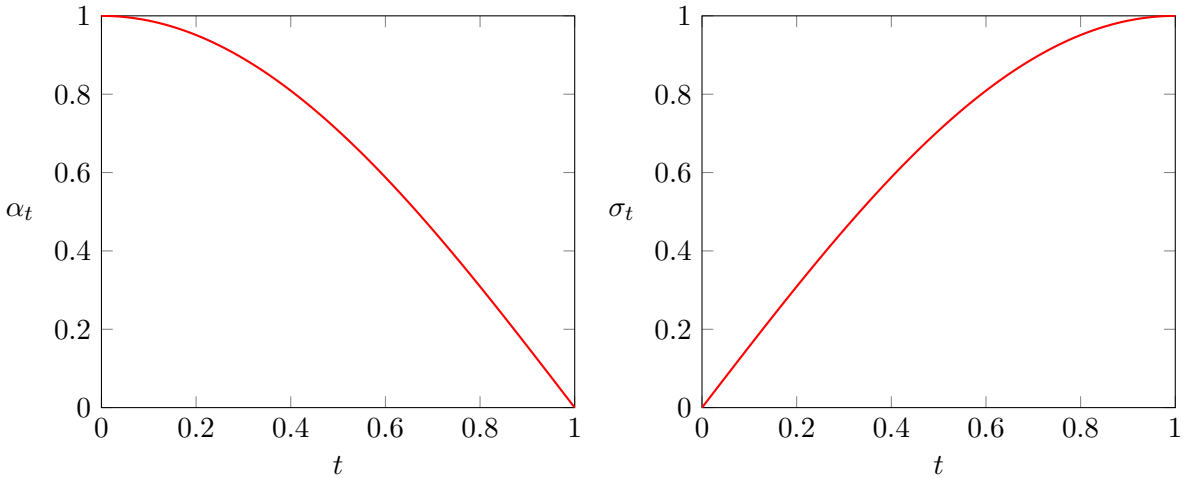



Figure 2.4: Relationship between time $t$ and $\alpha_t$ (left) and $\sigma_t$ (right) for the same truncated continuous-time $\alpha$-cosine noise schedule as that in Figure 2.3. The horizontal axis is time $t \in [0, 1]$; the vertical axis is the value of $\alpha_t$ (left) and $\sigma_t$ (right).

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 $\alpha$-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 $\alpha$-cosine schedule, and given by:

$$p_\Lambda(\lambda) = \frac{1}{2\pi(t_1 - t_0)} \operatorname{sech}\left(\frac{\lambda}{2}\right) \tag{2.65}$$

Figure 2.5 displays the probability density function.

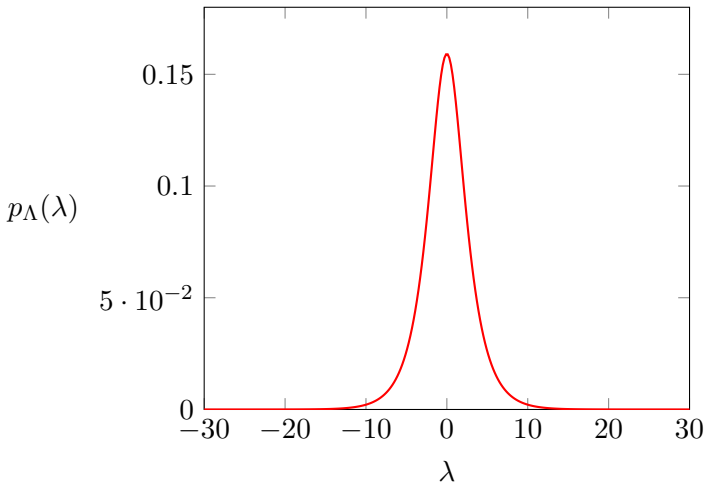



Figure 2.5: Probability density function $p_\Lambda(\lambda)$ for the same truncated continuous-time $\alpha$-cosine schedule as that in Figure 2.3. The horizontal axis is the log signal-to-noise ratio $\lambda \in [\lambda_{\min}, \lambda_{\max}]$; the vertical axis is the corresponding probability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

### 2.7.5 Parameterisations

In Section 2.7.3, 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 $\lambda_t$ 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, 14]), 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:

$$\hat{\mathbf{x}}_\theta(\mathbf{z}_t, \lambda_t) = \frac{1}{\alpha_t}\left(\mathbf{z}_t - \sigma_t \hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t, \lambda_t)\right) \tag{2.66}$$

In this work, we employ the $\mathbf{v}$-prediction parameterisation, introduced originally by Salimans and Ho [15], 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 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:

$$\mathbf{v}_t = \alpha_t \boldsymbol{\epsilon} - \sigma_t \mathbf{x} \tag{2.67}$$

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:

$$\mathbb{E}_{\lambda \sim p_\Lambda(\lambda), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \| \mathbf{v}_t - \hat{\mathbf{v}}_\theta(\mathbf{z}_t, \lambda_t) \|_2^2 \right] \tag{2.68}$$

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:

$$\hat{\mathbf{x}}(\mathbf{z}_t, \lambda_t) = \alpha_t \mathbf{z}_t - \sigma_t \hat{\mathbf{v}}_\theta(\mathbf{z}_t, \lambda_t) \tag{2.69}$$

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

### 2.7.6 Score-Based Interpretation

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

$$\mathbf{z} \sim p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\mathbf{z}, \boldsymbol{\Sigma}_\mathbf{z}) \tag{2.70}$$

Tweedie's formula states that:

$$\mathbb{E}\left[\boldsymbol{\mu}_\mathbf{z} | \mathbf{z}\right] = \mathbf{z} + \boldsymbol{\Sigma}_\mathbf{z} \nabla_\mathbf{z} \log p(\mathbf{z}) \tag{2.71}$$

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

$$\mathbb{E}\left[\boldsymbol{\mu}_{\mathbf{z}_t} | \mathbf{z}_t\right] = \mathbf{z}_t + \sigma_t^2 \nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t) \tag{2.72}$$

From Equation 2.42, 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:

$$\alpha_t \mathbf{x} = \mathbf{z}_t + \sigma_t^2 \nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t) \tag{2.73}$$

### 2.7.7 ELBO for Diffusion Models

We can interpret the variance-preserving diffusion model used in this work as an MHVAE with several additional restrictions. Firstly, the dimensionality of each latent $\mathbf{z}_t$ equals the dimensionality of the observed variable. Secondly, we have pre-defined $q(\mathbf{z}_t | \mathbf{z}_s)$ where $0 \leq s < t \leq 1$ as a Gaussian diffusion process with no learnable inference parameters. Finally, the marginal distribution of the final latent $q(\mathbf{z}_1)$ is approximately the multivariate standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$, and thus holds effectively no information about the observed variable $\mathbf{x}$. VAEs and MHVAEs do not typically have these restrictions. Nonetheless, much like VAEs and MHVAEs, we can optimise the generative parameters $\theta$ of diffusion models by minimising the ELBO loss. As a notable example, Sohl-Dickstein et al. [16] optimised the original discrete-time diffusion model via the ELBO loss.

For a given datapoint $\mathbf{x}$, we define $\mathcal{L}_\Lambda(\lambda)$ as 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 $\mathbf{x}$:

$$\mathcal{L}_\Lambda(\lambda) = D_{KL}(q(\mathbf{z}_t, \dots, \mathbf{z}_1 | \mathbf{x}) \| p_\theta(\mathbf{z}_t, \dots, \mathbf{z}_1)) \tag{2.74}$$

Notably, $\mathcal{L}_\Lambda(\lambda)$ equates to $\mathcal{L}_T(t)$ defined in Equation 2.31 under a simple change of variable:

$$\mathcal{L}_\Lambda(\lambda) = \mathcal{L}_T(t = f_\Lambda^{-1}(\lambda)) \tag{2.75}$$

Similarly, we can reformulate the ELBO loss for a continuous-time MHVAE given in Equation 2.35 to provide the ELBO loss in terms of the log signal-to-noise ratio $\lambda$; it is given per datapoint $\mathbf{x}$ by:

$$\mathcal{L}_{\text{ELBO}}(\mathbf{x}) = \mathbb{E}_{\mathbf{z}_0 \sim q(\mathbf{z}_0, \mathbf{x})} \left[ -\log p_\theta(\mathbf{x}|\mathbf{z}_0) \right] + \mathcal{L}_\Lambda(\lambda_{\max}) \tag{2.76}$$

$$= \underbrace{\mathbb{E}_{\mathbf{z}_0 \sim q(\mathbf{z}_0, \mathbf{x})} \left[ -\log p_\theta(\mathbf{x}|\mathbf{z}_0) \right]}_{\text{Reconstruction Loss}} + \underbrace{\mathcal{L}_\Lambda(\lambda_{\min})}_{\text{Prior Loss}} + \int_{\lambda_{\min}}^{\lambda_{\max}} \mathcal{L}'_\Lambda(\lambda) d\lambda \tag{2.77}$$

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:

$$\lim_{\lambda_{\max} \to \infty} q(\mathbf{z}_0|\mathbf{x}) = \delta(\mathbf{z}_0 - \mathbf{x}) \tag{2.78}$$

where $\delta$ is the Dirac delta distribution. Similarly, with sufficiently small $\lambda_{\min}$, the prior loss is approximately zero; as $\lambda_{\min} \to -\infty$, we have:

$$\lim_{\lambda_{\min} \to -\infty} q(\mathbf{z}_1|\mathbf{x}) = \mathcal{N}(\mathbf{0}, \mathbf{I}) = p_\theta(\mathbf{z}_1) \tag{2.79}$$

so the KL divergence prior loss term likewise approaches zero.

In Appendix C of [7], Kingma and Gao showed that $\mathcal{L}'_\Lambda(\lambda)$—which, with a slight abuse of terminology, they refer to as the *time derivative*—simplifies to a remarkable degree:

$$\mathcal{L}'_\Lambda(\lambda) = \frac{d}{d\lambda}\mathcal{L}_\Lambda(\lambda) = \frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t, \lambda_t)\|_2^2 \right] \tag{2.80}$$

### 2.7.8 Weighted Loss

Most diffusion models in the broader literature—including state-of-the-art models—do not optimise their parameters $\theta$ via minimisation of the ELBO loss. In practice, the various objectives used are all special cases of a *weighted loss* [7], which is defined per datapoint $\mathbf{x}$ as:

$$\mathcal{L}_{\text{WL}} = w(\lambda_{\min})\mathcal{L}_\Lambda(\lambda_{\min}) + \int_{\lambda_{\min}}^{\lambda_{\max}} w(\lambda)\mathcal{L}'_\Lambda(\lambda) d\lambda \tag{2.81}$$

where $w(\lambda)$ is a weighting function. Note that, assuming the reconstruction loss is approximately zero, the ELBO loss given in Equation 2.77 is a special case of the weighted loss $\mathcal{L}_{\text{WL}}$ with $w(\lambda) = 1$. Substituting the form of $\mathcal{L}'_\Lambda(\lambda)$ given in Equation 2.80 yields the following form for the weighted loss:

$$\mathcal{L}_{\text{WL}} = w(\lambda_{\min})\mathcal{L}_\Lambda(\lambda_{\min}) + \frac{1}{2}\int_{\lambda_{\min}}^{\lambda_{\max}} w(\lambda)\mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t, \lambda)\|_2^2 \right] d\lambda \tag{2.82}$$

This form provides several useful insights. Since the first term—the weighted prior loss—contains no learnable parameters, minimisation of the weighted loss $\mathcal{L}_{\text{WL}}$ equates to minimisation of the intractable integral. In practice, we minimise the integral via an importance-weighted Monte Carlo integrator:

$$\int_{\lambda_{\min}}^{\lambda_{\max}} w(\lambda)\mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t, \lambda_t)\|_2^2 \right] d\lambda = \mathbb{E}_{\lambda \sim p_\Lambda(\lambda)} \left[ \frac{w(\lambda)}{p_\Lambda(\lambda)} \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t, \lambda)\|_2^2 \right] \right] \tag{2.83}$$

$$= \mathbb{E}_{\lambda \sim p_\Lambda(\lambda), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \frac{w(\lambda)}{p_\Lambda(\lambda)} \|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t, \lambda)\|_2^2 \right] \tag{2.84}$$

$$\simeq \frac{w(\lambda)}{p_\Lambda(\lambda)} \|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t, \lambda)\|_2^2 \tag{2.85}$$

Notable further analysis by Kingma and Gao [7] shows that the weighted loss also has a likelihood-based interpretation. Simple integration by parts enables us to write the weighted loss as:

$$\mathcal{L}_{\text{WL}} = w(\lambda_{\max})\mathcal{L}_{\Lambda}(\lambda_{\max}) + \int_{\lambda_{\min}}^{\lambda_{\max}} -w'(\lambda)\mathcal{L}_{\Lambda}(\lambda)d\lambda \tag{2.86}$$

The likelihood-based interpretation comes from the fact that $\mathcal{L}_{\Lambda}(\lambda)$ serves as a variational bound on the negative marginal likelihood of the noise-perturbed data $\mathbf{z}_t \sim q(\mathbf{z}_t|\mathbf{x})$:

$$\begin{aligned}\mathcal{L}_{\Lambda}(\lambda) &\geq D_{KL}(q(\mathbf{z}_t|\mathbf{x})\|p(\mathbf{z}_t)) &(2.87)\\ &= \mathbb{E}_{\mathbf{z}_t\sim q(\mathbf{z}_t|\mathbf{x})}\left[-\log p(\mathbf{z}_t)\right] + \mathbb{E}_{\mathbf{z}_t\sim q(\mathbf{z}_t|\mathbf{x})}\left[\log q(\mathbf{z}_t|\mathbf{x})\right] &(2.88)\end{aligned}$$

As such, minimisation of $\mathcal{L}_{\Lambda}(\lambda)$ 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 $\lambda$. 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 $\mathcal{L}_{\text{WL}}$ 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 $-w'(\lambda)$.

Kingma and Gao's [7] analysis directly justifies our use of the $\mathbf{v}$-prediction parameterisation with the truncated continuous-time $\alpha$-cosine noise schedule. In conjunction, their use equates to the weighted loss with:

$$w(\lambda) = \frac{1}{2\pi(t_1 - t_0)} \exp\left(-\frac{\lambda}{2}\right) \tag{2.89}$$

$$-w'(\lambda) = \frac{1}{4\pi(t_1 - t_0)} \exp\left(-\frac{\lambda}{2}\right) \tag{2.90}$$

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 $\lambda$, and as such $-w'(\lambda)$ is positive for all $\lambda \in [\lambda_{\min}, \lambda_{\max}]$. Therefore, during training, we are maximising a weighted expected log-likelihood of noise-perturbed data $\mathbf{z}_t \sim q(\mathbf{z}_t|\mathbf{x})$ for all $\lambda \in [\lambda_{\min}, \lambda_{\max}]$. In contrast, most diffusion models in the broader literature (see e.g. [3, 12, 14]) 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 $\mathbf{z}_t \sim q(\mathbf{z}_t|\mathbf{x})$.

### 2.7.9 Imputation for Conditional Generation

In Section 2.2, we introduced the concept of conditional generation. In this work, however, we do not train a conditional model explicitly. Instead, we utilise *reconstruction-guided sampling* [5]: a sophisticated technique that derives a conditional model approximately from an unconditional model. More specifically, reconstruction-guided sampling facilitates the conditional generation of the unknown dimensions of some observed datapoint $\mathbf{x}$ given the known dimensions. Deriving such a conditional model approximately from an unconditional model is advantageous: it enables us to train only a single unconditional model, which we can then flexibly use to facilitate the conditional generation of any unknown subset of dimensions. In this work, we utilise reconstruction-guided sampling to facilitate three distinct conditional generation tasks: temporal interpolation, forecasting, and autoregressive generation of arbitrarily long samples.

Reconstruction-guided sampling extends a prior technique known as *imputation*, introduced by Song et al. [19]. Thus, we first introduce imputation for completeness and to better motivate our use of reconstruction-guided sampling in this work.

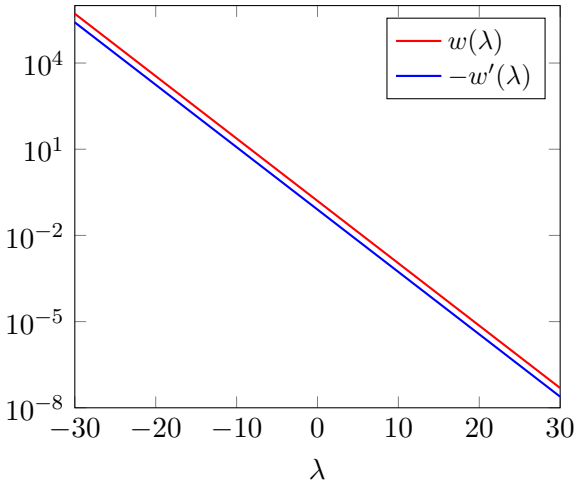



Figure 2.6: Relationship between the log signal-to-noise ratio $\lambda$ and the functions $w(\lambda)$ and $-w'(\lambda)$ for the $\mathbf{v}$-parameterisation with the same truncated continuous-time $\alpha$-cosine schedule as that in Figure 2.3. The horizontal axis is the log signal-to-noise level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

We denote by $\Omega(\mathbf{x})$ and $\bar{\Omega}(\mathbf{x})$ the known and unknown dimensions of some observed datapoint $\mathbf{x}$, respectively. Formally, our goal is to derive the following conditional model without training it explicitly:

$$p_\theta(\bar{\Omega}(\mathbf{x})|\Omega(\mathbf{x})) \tag{2.91}$$

We can write the forward diffusion process for the unknown dimensions as the following SDE:

$$d\bar{\Omega}(\mathbf{z}_t) = \mathbf{f}(\bar{\Omega}(\mathbf{z}_t), t) + g(t)d\mathbf{w}_t \tag{2.92}$$

Anderson [1] shows that the corresponding reverse-time SDE conditioned on the known dimensions $\Omega(\mathbf{x})$ is given by:

$$d\bar{\Omega}(\mathbf{z}_t) = \left[\mathbf{f}(\bar{\Omega}(\mathbf{z}_t), t) - g(t)^2 \nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q(\bar{\Omega}(\mathbf{z}_t)|\Omega(\mathbf{x}))\right] dt + g(t)d\bar{\mathbf{w}}_t \tag{2.93}$$

Although $q(\bar{\Omega}(\mathbf{z}_t)|\Omega(\mathbf{x}))$ is intractable, we can approximate it as follows:

$$\begin{aligned}
q(\bar{\Omega}(\mathbf{z}_t)|\Omega(\mathbf{x})) &= \int q(\bar{\Omega}(\mathbf{z}_t), \Omega(\mathbf{z}_t)|\Omega(\mathbf{x}))d\Omega(\mathbf{z}_t) && (2.94)\\
&= \int q(\bar{\Omega}(\mathbf{z}_t)|\Omega(\mathbf{x}), \Omega(\mathbf{z}_t))q(\Omega(\mathbf{z}_t)|\Omega(\mathbf{x}))d\Omega(\mathbf{z}_t) && (2.95)\\
&= \mathbb{E}_{\Omega(\mathbf{z}_t)\sim q(\Omega(\mathbf{z}_t)|\Omega(\mathbf{x}))}\left[q(\bar{\Omega}(\mathbf{z}_t)|\Omega(\mathbf{x}), \Omega(\mathbf{z}_t))\right] && (2.96)\\
&\approx \mathbb{E}_{\Omega(\mathbf{z}_t)\sim q(\Omega(\mathbf{z}_t)|\Omega(\mathbf{x}))}\left[q(\bar{\Omega}(\mathbf{z}_t)|\Omega(\mathbf{z}_t))\right] && (2.97)
\end{aligned}$$

Song et al. [19] argue that the approximation in Equation 2.97 is appropriate since for small $t$, $\Omega(\mathbf{x})$ is almost the same as $\Omega(\mathbf{z}_t)$; and for larger $t$, $\Omega(\mathbf{x})$ is further away from $\bar{\Omega}(\mathbf{z}_t)$ in the Markov chain, and thus has a smaller impact on $\bar{\Omega}(\mathbf{z}_t)$. Assuming the approximation holds, we can derive an unbiased estimator of $q(\bar{\Omega}(\mathbf{z}_t)|\Omega(\mathbf{x}))$ as:

$$q(\bar{\Omega}(\mathbf{z}_t)|\Omega(\mathbf{x})) \approx \mathbb{E}_{\Omega(\mathbf{z}_t)\sim q(\Omega(\mathbf{z}_t)|\Omega(\mathbf{x}))}\left[q(\bar{\Omega}(\mathbf{z}_t)|\Omega(\mathbf{z}_t))\right] \simeq q(\bar{\Omega}(\mathbf{z}_t)|\Omega(\mathbf{z}_t)) \tag{2.98}$$

The score of the natural logarithm of the unbiased estimator is given by:

$$\begin{aligned}
\nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q(\bar{\Omega}(\mathbf{z}_t)|\Omega(\mathbf{z}_t)) &= \nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q(\bar{\Omega}(\mathbf{z}_t), \Omega(\mathbf{z}_t)) - \nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q(\Omega(\mathbf{z}_t)) && (2.99)\\
&= \nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q(\bar{\Omega}(\mathbf{z}_t), \Omega(\mathbf{z}_t)) && (2.100)\\
&= \nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q([\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)]) && (2.101)
\end{aligned}$$

where $[\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)]$ denotes a vector such that:

$$\Omega([\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)]) = \Omega(\mathbf{z}_t) \tag{2.102}$$
$$\bar{\Omega}([\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)]) = \bar{\Omega}(\mathbf{z}_t) \tag{2.103}$$

Thus, assuming the assumption given in Equation 2.97 holds, we can consequently approximate the reverse-time SDE conditioned on $\Omega(\mathbf{x})$ given in Equation 2.93 as follows:

$$d\bar{\Omega}(\mathbf{z}_t) = \left[\mathbf{f}(\bar{\Omega}(\mathbf{z}_t), t) - g(t)^2 \nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q([\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)])\right] dt + g(t) d\bar{\mathbf{w}}_t \tag{2.104}$$

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 reverse-time SDE is thus given by:

$$d\bar{\Omega}(\mathbf{z}_t) = \left[\mathbf{f}(\bar{\Omega}(\mathbf{z}_t), t) - g(t)^2 \mathbf{s}_\theta([\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)], \lambda_t)\right] dt + g(t) d\bar{\mathbf{w}}_t \tag{2.105}$$

To provide a more intuitive link between Equation 2.105 and the generative procedure in Section 3.2, we also define the imputation method as a conditional denoiser $\hat{\mathbf{x}}_\theta^{\mathrm{C}}(\mathbf{z}_t, \lambda_t, \Omega(\mathbf{x}))$, which takes the known dimensions of $\mathbf{x}$ as input:

$$\hat{\mathbf{x}}_\theta^{\mathrm{C}}(\mathbf{z}_t, \lambda_t, \Omega(\mathbf{x})) = \bar{\Omega}\left(\hat{\mathbf{x}}_\theta([\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)], \lambda_t)\right) \tag{2.106}$$

Thus, the imputation method for conditional generation yields only a slight adjustment to the generative procedure defined in Section 2.7.3. Namely, at each step of the discrete-time ancestral sampler, we replace the dimensions of $\mathbf{z}_t$ corresponding to the known dimensions of $\mathbf{x}$ with an exact sample from the forward process: $\Omega(\mathbf{z}_t) \sim q(\Omega(\mathbf{z}_t)|\Omega(\mathbf{x}))$.

### 2.7.10 Reconstruction-Guided Sampling for Conditional Generation

Ho et al. [5] showed that the imputation process produces incoherent samples when applied to video diffusion models. Namely, although a given $\bar{\Omega}(\mathbf{x})$ will often appear reasonable in isolation, it will often not be coherent with $\Omega(\mathbf{x})$. This incoherency is likely because the assumption in Equation 2.97 does not hold for all $t \in [0, 1]$. Avoiding the assumption, we instead construct an unbiased estimator for $q(\bar{\Omega}(\mathbf{z}_t|\Omega(\mathbf{x})))$ as:

$$q(\bar{\Omega}(\mathbf{z}_t)|\Omega(\mathbf{x})) = \mathbb{E}_{\Omega(\mathbf{z}_t) \sim q(\Omega(\mathbf{z}_t)|\Omega(\mathbf{x}))}\left[q(\bar{\Omega}(\mathbf{z}_t)|\Omega(\mathbf{x}), \Omega(\mathbf{z}_t))\right] \simeq q(\bar{\Omega}(\mathbf{z}_t)|\Omega(\mathbf{x}), \Omega(\mathbf{z}_t)) \tag{2.107}$$

The score of the natural logarithm of the unbiased estimator is given by:

$$\nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q(\bar{\Omega}(\mathbf{z}_t)|\Omega(\mathbf{x}), \Omega(\mathbf{z}_t)) = \nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q(\bar{\Omega}(\mathbf{z}_t), \Omega(\mathbf{x}), \Omega(\mathbf{z}_t)) - \nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q(\Omega(\mathbf{z}_t), \Omega(\mathbf{x})) \tag{2.108}$$
$$= \nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q(\bar{\Omega}(\mathbf{z}_t), \Omega(\mathbf{x}), \Omega(\mathbf{z}_t)) \tag{2.109}$$
$$= \nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q(\bar{\Omega}(\mathbf{z}_t), \Omega(\mathbf{z}_t)) + \nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q(\Omega(\mathbf{x})|\bar{\Omega}(\mathbf{z}_t), \Omega(\mathbf{z}_t)) \tag{2.110}$$
$$= \nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q([\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)]) + \nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q(\Omega(\mathbf{x})|[\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)]) \tag{2.111}$$

The second term in Equation 2.111 is missing in the imputation approach. Plugging in this missing term would make conditional sampling exact. However, since $q(\Omega(\mathbf{x})|[\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)])$ is not available in closed form, we must approximate it. Ho et al. [5] proposed to approximate it with a multivariate Gaussian of the form:

$$q(\Omega(\mathbf{x})|[\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)]) \approx \mathcal{N}\left(\Omega(\mathbf{x}); \Omega\left(\hat{\mathbf{x}}_\theta\left([\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)], \lambda_t\right)\right), \left(\frac{\sigma_t^2}{\alpha_t^2}\right) \mathbf{I}\right) \tag{2.112}$$

Under this approximation, the second term of Equation 2.111 is thus itself approximated by:

$$\nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q(\Omega(\mathbf{x})|[\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)]) \approx -\frac{\alpha_t^2}{2\sigma_t^2} \nabla_{\bar{\Omega}(\mathbf{z}_t)} \left\| \Omega(\mathbf{x}) - \Omega\left(\hat{\mathbf{x}}_\theta\left([\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)], \lambda_t\right)\right)\right\|_2^2 \tag{2.113}$$

Ho et al. [5] interpret the inclusion of this term—absent in the imputation method—as a form of *guidance* based on the model's reconstruction of the conditioning data. They found empirically that—as with other forms of guidance—including a large weighting term $w_r > 1$ tends to improve sample quality further. They refer to this technique as *reconstruction-guided sampling*. Formally, assuming we have a perfect score model $\mathbf{s}_\theta(\mathbf{z}_t, \lambda_t) = \nabla_{\mathbf{z}_t} q(\mathbf{z}_t)$, reconstruction-guided sampling derives a conditional model from an unconditional model by approximating the score given in Equation 2.93 by:

$$\nabla_{\bar{\Omega}(\mathbf{z}_t)} \log q(\bar{\Omega}(\mathbf{z}_t)|\Omega(\mathbf{x})) \approx \mathbf{s}_\theta([\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)], \lambda_t) \tag{2.114}$$

$$- \frac{w_r \alpha_t^2}{2\sigma_t^2} \nabla_{\bar{\Omega}(\mathbf{z}_t)} \left\| \Omega(\mathbf{x}) - \Omega\left(\hat{\mathbf{x}}_\theta\left([\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)], \lambda_t\right)\right)\right\|_2^2 \tag{2.115}$$

Writing the reconstruction-guidance method as a conditional denoiser $\hat{\mathbf{x}}_\theta^{\mathrm{C}}(\mathbf{z}_t, \lambda_t, \Omega(\mathbf{x}))$ yields:

$$\hat{\mathbf{x}}_\theta^{\mathrm{C}}(\mathbf{z}_t, \lambda_t, \Omega(\mathbf{x})) = \bar{\Omega}\left(\hat{\mathbf{x}}_\theta([\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)], \lambda_t)\right) - \frac{w_r \alpha_t}{2} \nabla_{\bar{\Omega}(\mathbf{z}_t)} \left\| \Omega(\mathbf{x}) - \Omega\left(\hat{\mathbf{x}}_\theta\left([\bar{\Omega}(\mathbf{z}_t); \Omega(\mathbf{z}_t)]\right)\right)\right\|_2^2 \tag{2.116}$$

### 2.7.11 U-Net

# Chapter 3

# Climate Background

## 3.1 Climate Simulations

## 3.2 Generative Models for Climate Simulations

# Chapter 4

# Experiments and Results

## 4.1 Dataset

### 4.1.1 UKCP18

United Kingdom Climate Projections 2018 (UKCP18) [11] is the UK Met Office's latest generation of national climate projections. UKCP18 contains simulations for various components of the climate under several Representative Concentration Pathways (RCPs), which are trajectories for greenhouse gas concentration adopted by the Intergovernmental Panel on Climate Change. For example, RCP8.5 represents a scenario whereby greenhouse gas emissions continue to grow unmitigated [10].

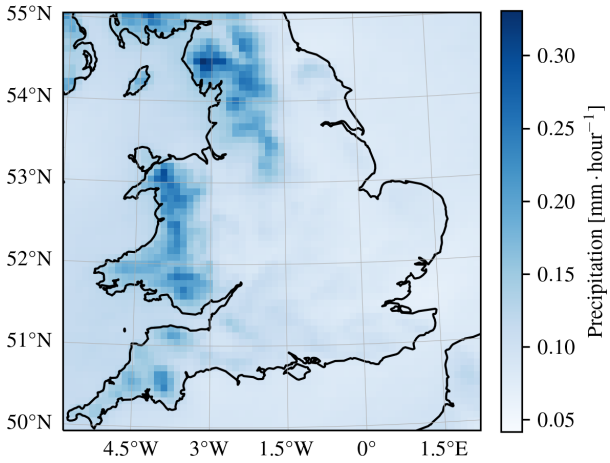



Figure 4.1: Mean precipitation rate for each 8.8 km × 8.8 km subarea of the UKCP18 dataset.

**Change the above. It's not the UKCP18 dataset, we're using a specific ensemble**

### 4.1.2 Train and Test Sets

We scale each input data element to the range $[-1, 1]$ to ensure that our neural network operates on consistently scaled inputs during the reverse-time generative process, starting from the multivariate standard Gaussian prior.

Before proceeding to the results of our model, we first introduce the concept of the *dry threshold*, which is the minimum precipitation rate that we deem non-zero for our analysis. We set our dry threshold to be 0.01 millimetres per hour, representing a value below which we deem precipitation inconsequential for real-world use cases. Without this threshold, we would have to evaluate the degree to which we correctly model the entire range of nonzero precipitation rates in the UKCP18 dataset, which we observed to go as low as $5.4 \times 10^{-17}$ millimetres per

hour. The dry threshold thus streamlines our analysis: it delineates a clear value below which we deem precipitation inconsequential and above which to focus our attention.

## 4.2 Importance of the Transformation

### 4.2.1 No Transformation

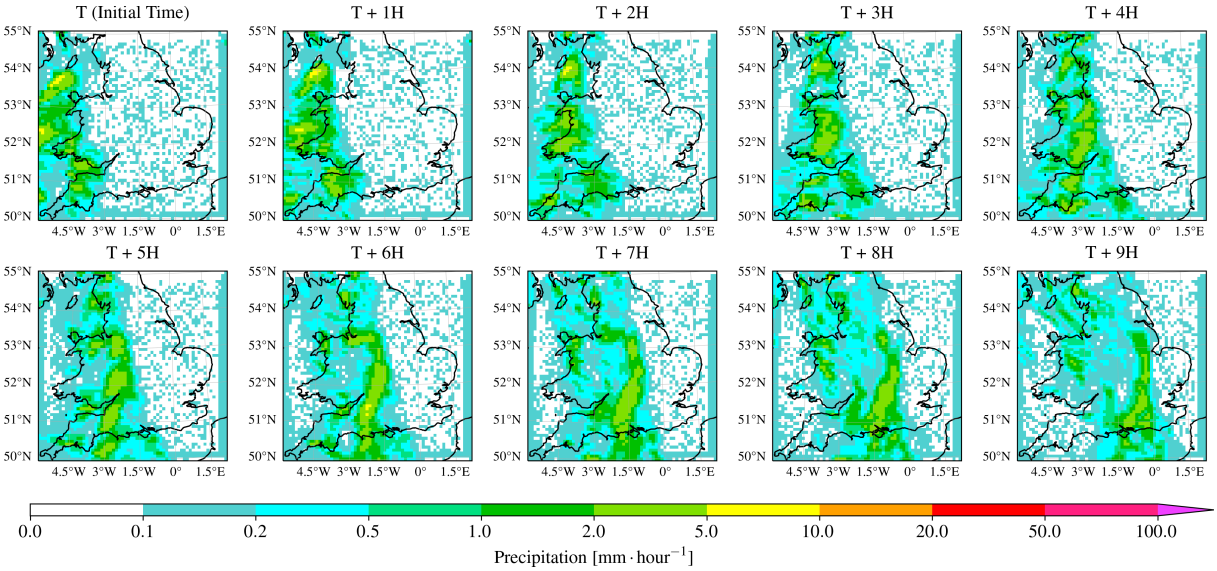
One of the primary contributions of this work is demonstrating that a transformation of the input data is critical to the model's performance. Since no directly comparable results exist in the broader literature, we first establish a benchmark by providing results for our model trained without any transformation to the input data.

Notably, the model produced samples with two significant, related issues. First and foremost, the generated samples all contain significant perceptible noise above the dry threshold at the lower end of the precipitation scale; Figure 4.2a depicts an example of this. Additionally, many samples exhibit a baseline shift in precipitation rates; that is, samples generated by the model have a modal precipitation rate significantly higher than those observed in the test set. Most samples in the test set have a modal precipitation rate below the dry threshold—indicating typically dry conditions. Conversely, many samples the model generates have a modal precipitation rate above the dry threshold, with little to no subareas below it. In practice, this indicates at least a trace quantity of precipitation across the entirety of the 563.2 km $\times$ 563.2 km region for the entire ten hours. Even in isolation, this would be an unnatural scenario and thus should certainly not occur at the frequency we observed in the samples generated via our model. Figure 4.2b depicts a sample exhibiting this baseline shift.

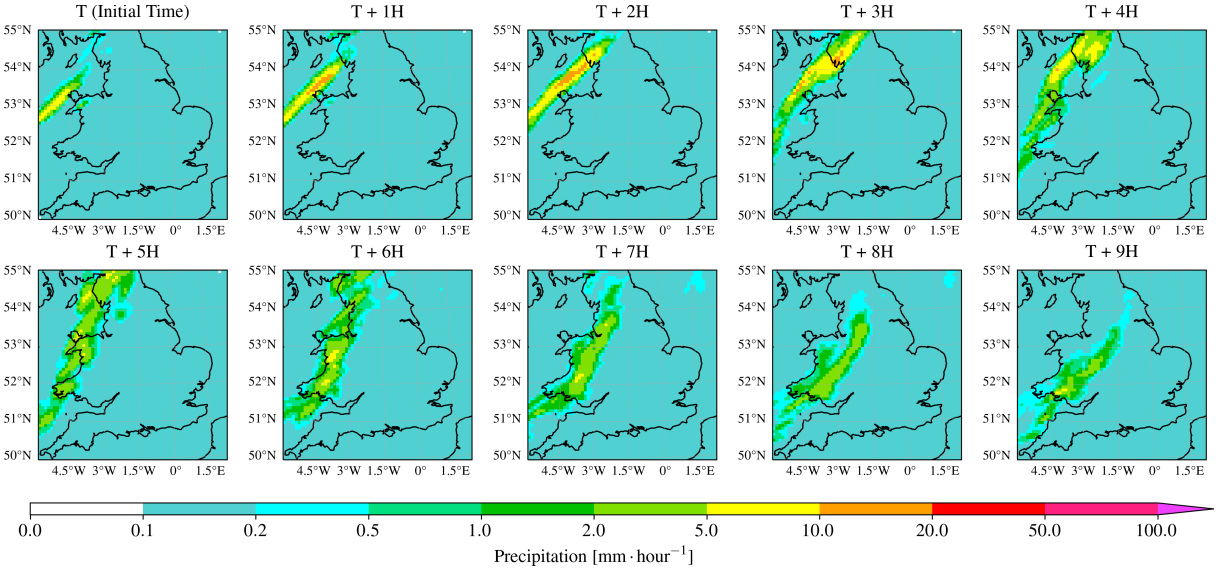
To motivate these phenomena beyond individual examples, Figure 4.3a depicts the distribution of precipitation rates of our generated samples compared to those in our test set at the lower end of the scale (i.e. less than 0.3 millimetres per hour) where the noise is most perceptible. As can be seen, below approximately 0.2 millimetres per hour, the distribution of precipitation rates in our generated samples significantly deviates from that of the test set. While precipitation of this scale equates only to trace amounts, it is still non-negligible and above the dry threshold we set. Thus, using the generated samples in real-world contexts—such as hydrological studies—may present significant challenges.

We can attribute much of this deviation to random noise visible in the generated samples. We cannot determine the precise source of this noise, but we speculate it may be a combination of multiple factors. The first contributing factor is likely that our score model $\mathbf{s}_\theta(\mathbf{z}_t, \lambda_t)$ is not perfect; in other words, there likely exists $\mathbf{z}_t$ such that $\mathbf{s}_\theta(\mathbf{z}_t, \lambda_t)$ does not serve as a good approximation for the true score $\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t)$. In addition, since we can only approximate the reverse-time SDE, the discretised numerical solver likely introduces errors itself—this would be the case even if we had a perfect score model. We speculate that the errors introduced by these two factors likely manifest as the noise at the lower end of the precipitation scale and the baseline shift in precipitation rate.

Noise of the observed magnitude creates unique challenges for diffusion models generating precipitation data—challenges that do not present as significant an issue in the more common domain of image generation. There are two primary reasons for this. First and foremost, the distribution of precipitation rates in our data is far more skewed than the distribution of colours in image data. This skew is evident in Figure 4.3b, which contains the full distribution of precipitation rates in our test set from UKCP18. 83% of the cells in the test set are of a precipitation rate lower than 0.01 millimetres per hour; to put this into perspective, the highest precipitation rate in our test set is approximately 60 millimetres per hour. As a direct result of this extreme distribution, even relatively small amounts of noise at the lower end of the scale or minute inaccuracies in the reverse-time generative model can significantly impact the quality of the generated samples. The second challenge that generating precipitation data presents in contrast to image data is that precipitation is a continuous variable, in contrast to the discrete

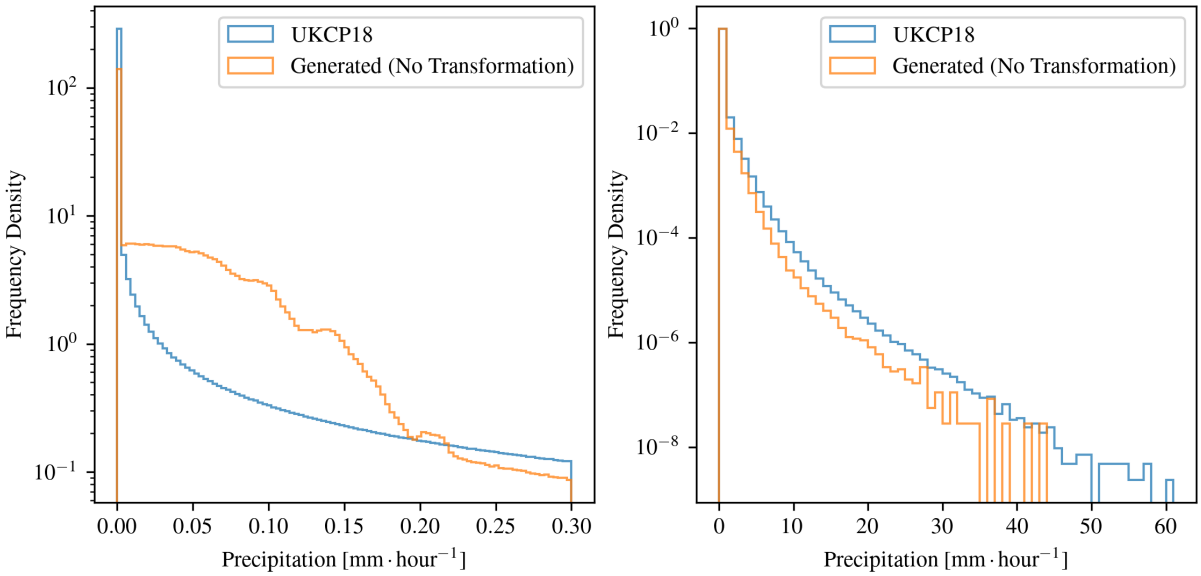



(a) Noise at the lower end of the precipitation scale




(b) Baseline shift in precipitation rate

Figure 4.2: Two samples generated via our model trained with no transformation of the input data. The sample depicts hourly precipitation for the same 563.2 km × 563.2 km region centred on Birmingham, UK, over ten hours. Each grid cell illustrates the mean precipitation rate, measured in millimetres per hour, for an 8.8 km × 8.8 km subarea. The top sample exhibits significant noise at the lower end of the precipitation scale, while the bottom sample exhibits a baseline shift in precipitation rate.




(a) Below 0.3 millimetres per hour

(b) Full distribution

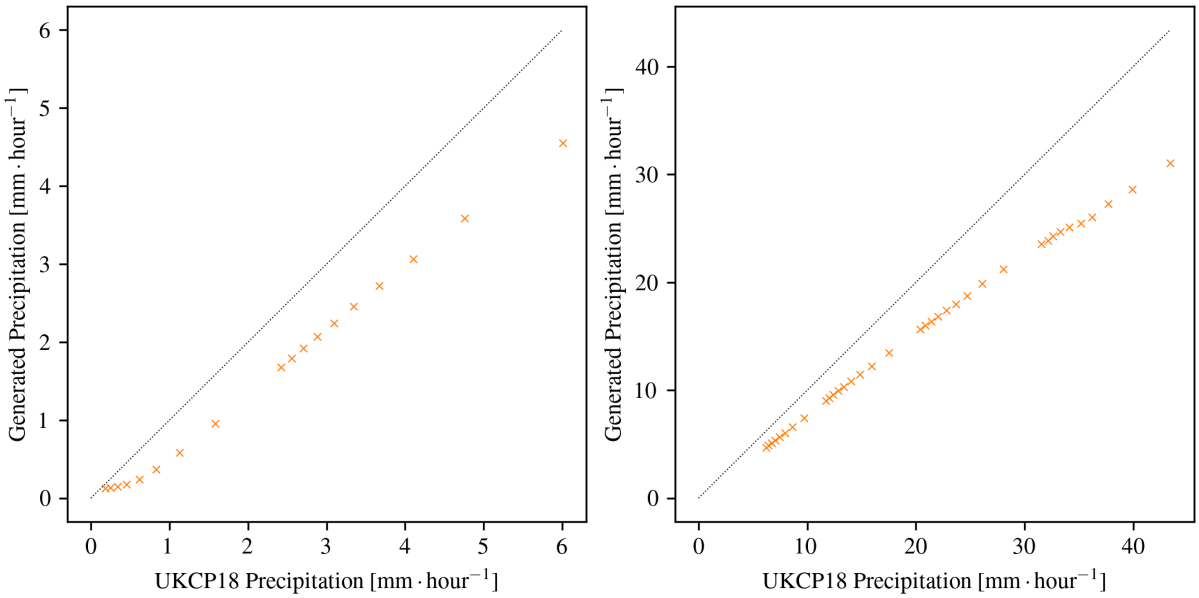
Figure 4.3: Distribution of precipitation rates in samples generated via our model trained with no transformation compared to the test set from UKCP18. The horizontal axis is the precipitation rate, measured in millimetres per hour; the vertical axis is frequency density. The vertical axis is scaled logarithmically. The left plot depicts the distribution of precipitation rates below 0.3 millimetres per hour, while the right plot depicts the full distribution.

values that comprise each colour channel in an image. Thus, a final discretisation step at the end of the generative process for images eliminates much of the noise. We cannot take advantage of this approach for precipitation data. In the context of Figure 4.2a wherein the significant deviation from the true distribution is evident below the 0.2 millimetres per hour threshold, this constitutes approximately 0.2% of our output space. If this were an image with 256 discrete values per colour channel, this relative amount of the output space represents less than a single discrete value. Thus, a final discretisation step would eliminate the problem almost entirely.

Our model trained with no transformation to the input is not entirely without merit. As evident from Figure 4.3b, the distribution of individual cells' values in the samples generated via our model broadly resembles that of individual cells' values in the test set. More specifically, ignoring the divergence at the low end of the scale, the frequency of precipitation events at increasing intensity levels approximately decreases at the same rate in both the generated samples and the test set. We interpret this as the model having broadly learnt the relative frequency of precipitation events at different intensity levels. In simpler terms, the model has correctly identified that more extreme precipitation events are less frequent than milder precipitation events.

However, the generated samples undeniably misrepresent the absolute frequency of precipitation events at most intensity levels. Ideally, above our dry threshold, the quantiles of the generated samples should be approximately equal to the quantiles of the test set, as this would indicate that the model has correctly identified the frequency of precipitation events at each intensity level. However, this is not the case. In the region below 0.2 millimetres per hour with the aforementioned significant noise, most of the quantiles of the generated samples are significantly higher than their corresponding quantiles in the test set. Conversely, above this region—except for a brief crossover point—most of the quantiles are approximately 25% lower than their corresponding quantile in the test set. This underrepresentation of precipitation events at most intensity levels is evident in Figure 4.4. Of the plotted quantiles, the more significant discrepancy is at quantile 0.9999999, the largest plotted and representing the most extreme

precipitation events. In any real-world context, the ability of our model to learn the frequency of the most extreme events is paramount, as these events are the most likely to cause significant damage to infrastructure and property. Thus, the model's inability to learn the frequency of these events is a significant shortcoming.




(a) Quantiles: $\{q \in \mathcal{Q} \mid 0.91 \leq q \leq 0.999\}$ (b) Quantiles: $\{q \in \mathcal{Q} \mid 0.9991 \leq q \leq 0.9999999\}$

Figure 4.4: Quantile-quantile plots for individual cells within the test set from UKCP18 against samples generated via our model trained with no transformation. Both plots are for the same samples, with different quantiles plotted; the quantiles plotted are disjoint subsets of $\mathcal{Q} = \{1 - 10^{-n_1} + n_2 \cdot 10^{-n_1-1} \mid n_1, n_2 \in \mathbb{N}, 1 \leq n_2 \leq 9\}$.

Evaluating our model goes beyond analysing the distribution of individual cells' values; it is essential to determine the extent to which our model has learnt the geographical structure of the true precipitation distribution across the UK. Ideally, our model must capture regional nuances influenced by topography and local climate patterns. For example, Wales experiences more precipitation on average than England, as the former is more mountainous. Accurate representation of this spatial structure is vital for reliable predictions and well-informed decision-making in agriculture, flood management and infrastructure development. To evaluate the effectiveness of our model in capturing the correct amount of precipitation in different subareas, we employ mean-normalised bias plots as an assessment tool. We calculate mean-normalised bias by subtracting the target mean from the sample mean and dividing the result by the target mean. Figure 4.5 depicts the mean-normalised bias for each 8.8 km × 8.8 km subarea. Interestingly, Wales, North West England and South West England are the regions with the largest absolute mean-normalised bias. These regions are those with the largest overall precipitation.

Moving beyond the distribution of individual cells' values, we must assess how successfully our model has learnt the true distribution's underlying spatial and temporal structure. This analysis is crucial: theoretically, we could trivially train a model to generate pure noise whereby the distribution of individual cells' values perfectly matches that of the test set. While an extreme example, this motivates the necessity of ensuring our generated samples also capture the structure and characteristics of the true distribution. To assess the degree to which this is the case, we employ power spectral density (PSD) graphs, which depict a signal's power distribution. In general terms, a PSD graph provides information on the distribution of frequencies present in a signal. In this work, we employ two different PSD graphs: spatial PSD graphs for individual one-hour snapshots of the entire 563.2 km × 563.2 km region; and temporal PSD graphs for

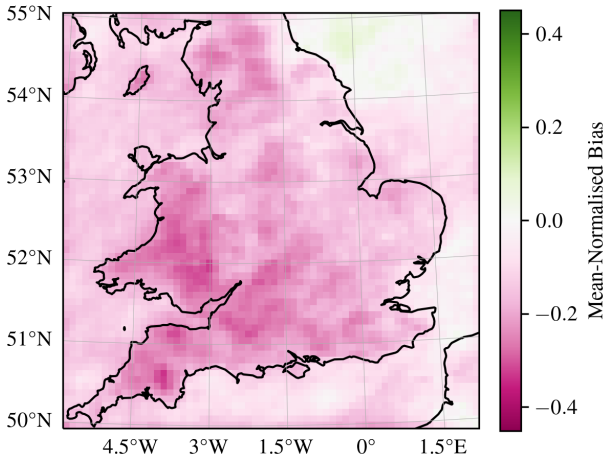



Figure 4.5: Mean-normalised bias at each cell representing the same 8.8 km×8.8 km geographical area in the samples generated via our model trained with no transformation to the input data.

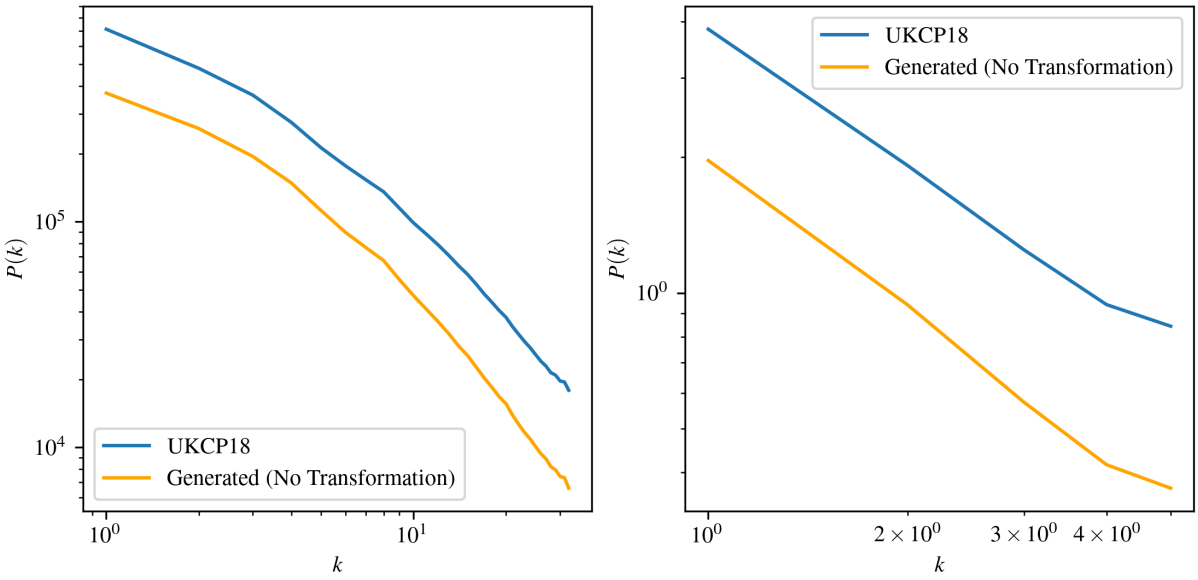
multiple consecutive one-hour snapshots of a single 8.8 km × 8.8 km subarea. Our interpretation of a PSD graph depends on the nature of the signal. For spatial PSD graphs, greater power at higher spatial frequencies reveals sharp boundaries between areas with and without precipitation, likely indicating the presence of localised showers. On the other hand, greater power at lower spatial frequencies points to smooth variations in precipitation intensity, suggesting a sizeable frontal system. For temporal PSD graphs, greater power at higher temporal frequencies reveals rapid fluctuations in precipitation, indicating the presence of briefer precipitation events such as convective showers. Conversely, greater power at lower temporal frequencies suggests stabler events—potentially a continuous dry spell.

From Figure 4.6, it is evident that the samples in the test set contain more power at both lower spatial and temporal frequencies than at higher temporal and spatial frequencies, on average. The samples generated via our model trained with no transformation follow a similar trend in both cases, as indicated by the similar shapes of the PSD graphs. We can interpret this as the model having broadly learnt the true distribution's underlying spatial and temporal structure. In other words, the model has broadly learnt the relative frequency of precipitation events at different spatial and temporal scales, from short localised showers to long-lasting frontal systems. However, the samples generated via our model contain less power than those in the test set at all scales. We can primarily attribute this discrepancy to the model underestimating the intensity of most precipitation events; if we increased the intensity of all precipitation events in the generated samples by an appropriate constant factor, the PSD graphs would be almost identical.

Importantly, these results serve as a benchmark to which we can compare the performance of our model trained with a square root transformation, as explored in Section 4.2.2.

### 4.2.2 Square Root Transformation

We found that performing a square root transformation to the input data eliminated the two primary problems observed in samples generated via our model trained with no transformation. Namely, it eliminated any detectable noise at the lower end of the precipitation scale, and none of its samples contained the unnatural baseline shift in precipitation rates. The improvement is evident in Figure 4.7a. The significant deviation from the test distribution shape below 0.2 millimetres per hour is absent in samples generated via our model trained with the square root transformation. We speculate this is directly related to the fact that the square root transformation provides a more significant proportion of the output space $[-1, 1]$ to the lower end of the precipitation scale. With no transformation, we linearly scale the input data to the range




(a) Spatial Dimensions

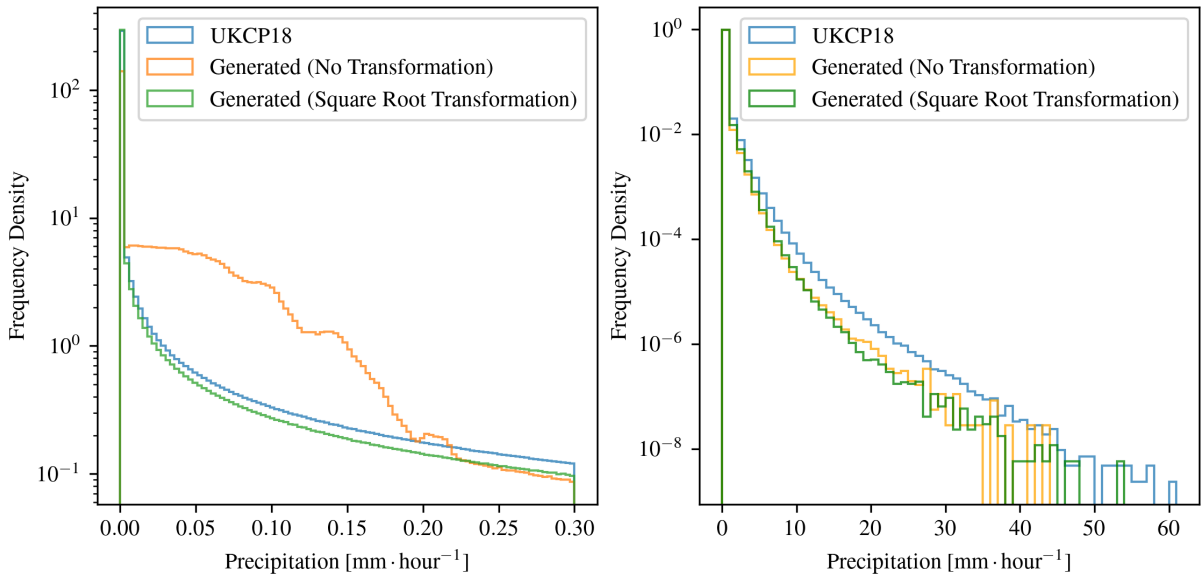
(b) Temporal Dimension

Figure 4.6: Power spectral density (PSD) graphs for samples generated via our model trained with no transformation, and the test set from UKCP18. The horizontal axis is the number of waves, denoted $k$; the vertical axis is the mean power across all samples for a given $k$, denoted $P(k)$. Both axes are logarithmic. The left plot is a PSD graph for two spatial dimensions, and the right plot is a PSD graph for the temporal dimension.

$[-1, 1]$, and as such, the region below 0.2 millimetres per hour only occupies approximately 0.2% of the output space. On the other hand, with the square root transformation, we first compute the square root of each element in the input data before conducting the same linear scaling. As such, the region below 0.2 millimetres per hour occupies approximately 5% of the output space—a 20-fold increase. We cannot definitively identify why allocating a more significant proportion of the output space to the same region eliminates the two issues mentioned. We leave this as a topic for future work.

Conducting the square root transformation improves the quality of our samples beyond simply eliminating the two aforementioned problems. One such improvement is evident in the histogram in Figure 4.3b. Namely, the distribution of individual cells' values better matches the test set's distribution in the entire region up to approximately 10 millimetres per hour, broadly equating to the beginning of what most would consider heavy precipitation. We see a similar improvement in the quantile-quantile plot in Figure 4.9a. However, for more intense precipitation, the dominance of either model is more ambiguous, as evident from the quantile-quantile plot in Figure 4.9b. The model trained via the square root transformation performed significantly better at the highest end of the scale, above 30 millimetres per hour. However, quantiles from the model trained with no transformation were closer to the target, particularly in the region from 10–30 millimetres per hour. However, we note that this is the sole domain wherein we found the square root transformation to yield a minor degradation in performance. Overall, we thus interpret the model trained on the square root transformation to have marginally better learnt the frequency of precipitation rates at most intensities.

Moving beyond the intensity of individual cells, we found that the square root transformation benefited the power in our samples. More specifically, the average power over the spatial and temporal dimensions is closer to that of the test set in samples generated via the square root transformation. The improvement is applicable over all scales, as evident from the spatial and temporal PSD graphs in Figure 4.10. We interpret this as the square root transformation enabling the model to learn the correct spatial and temporal characteristics of the true distribution




(a) Below 0.3 millimetres per hour

(b) Full distribution

Figure 4.7: Distribution of precipitation rates in samples generated via our model trained with the square root transformation, compared to both our model trained with no transformation and the test set from UKCP18. The format is the same as Figure 4.3

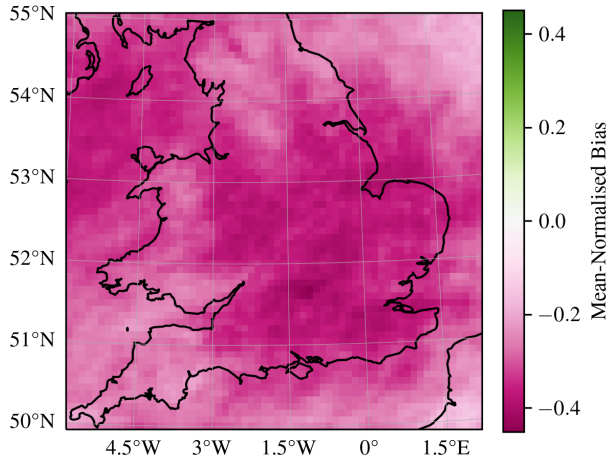
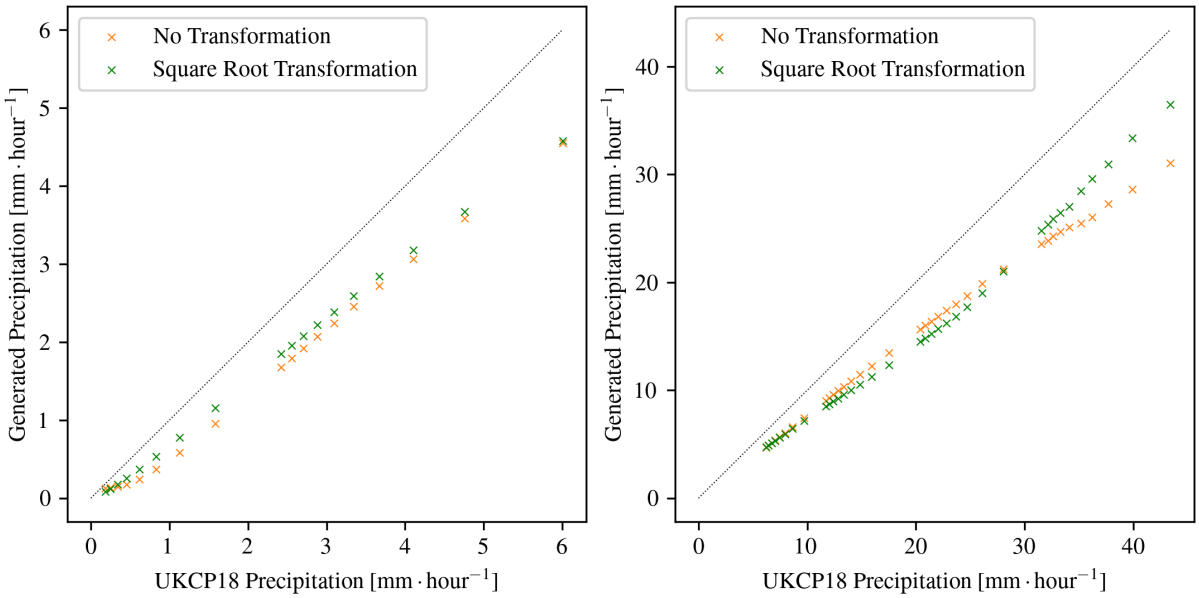



Figure 4.8: Mean-normalised bias at each cell representing the same 8.8 km×8.8 km geographical area in the samples generated via our model trained with the square root transformation.




(a) Quantiles: $\{q \in \mathcal{Q} \mid 0.91 \leq q \leq 0.999\}$ (b) Quantiles: $\{q \in \mathcal{Q} \mid 0.9991 \leq q \leq 0.9999999\}$

Figure 4.9: Quantile-quantile plots for individual cells within the test set from UKCP18 against both samples generated via our model trained with the square root transformation and our model trained with no transformation. The format is the same as Figure 4.4. $\mathcal{Q} = \{1 - 10^{-n_1} + n_2 \cdot 10^{-n_1 - 1} \mid n_1, n_2 \in \mathbb{N}, 1 \leq n_2 \leq 9\}$.

more effectively.

In all, we found the square root transformation to yield a significant improvement in sample quality, thus serving as clear evidence that a transformation of the input can be critical to the performance of a diffusion model.

## 4.3 Learning a Transformation

In this section, we explore the possibility of learning a transformation of the input data instead of explicitly defining one.

### 4.3.1 Change of Variable Rule

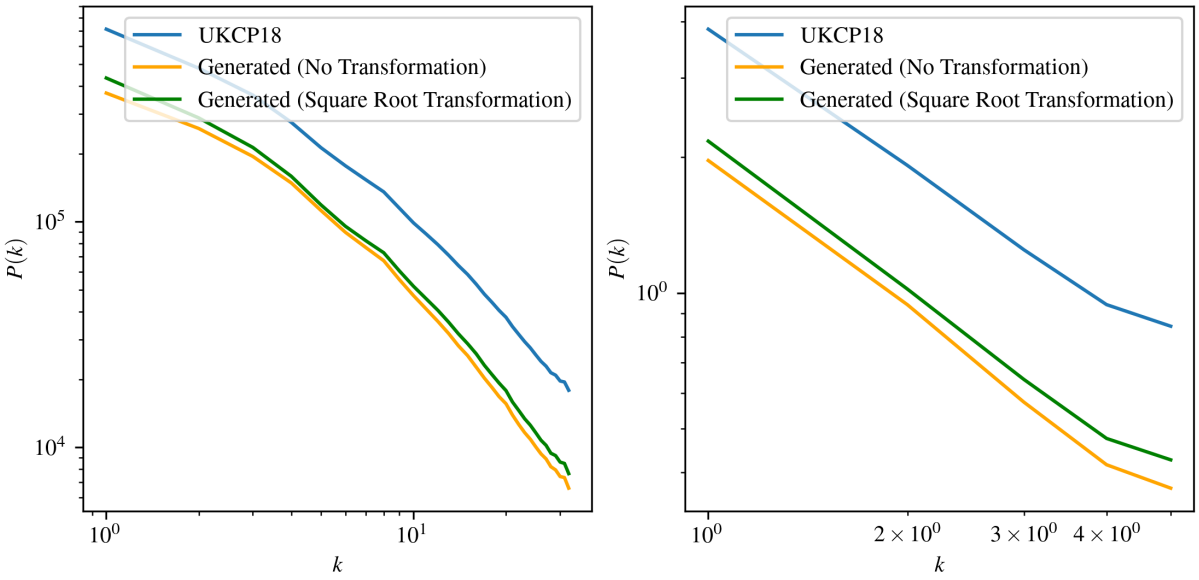
Suppose we transform our observed variable $\mathbf{x}$ via an element-wise vector-valued function $\mathbf{h} : [-1, 1]^D \to [-1, 1]^D$, such that:

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} h(x_1) \\ h(x_2) \\ \vdots \\ h(x_D) \end{bmatrix} \tag{4.1}$$

$x_i$ is the $i$-th element of $\mathbf{x}$, and $h : [-1, 1] \to [-1, 1]$ is a scalar-valued monotonically increasing function applied to each element of $\mathbf{x}$. The change of variable rule states that:

$$p_\theta(\mathbf{x}) = \tilde{p}_\theta(\mathbf{h}(\mathbf{x})) \left| \det \left( \frac{d\mathbf{h}(\mathbf{x})}{d\mathbf{x}} \right) \right| \tag{4.2}$$

where $\tilde{p}_\theta(\mathbf{h}(\mathbf{x}))$ is the probability density of our transformed observed variable $\mathbf{h}(\mathbf{x})$, det is the determinant operator, and $d\mathbf{h}(\mathbf{x})/d\mathbf{x}$ is the Jacobian of transformation $\mathbf{h}$. Since $\mathbf{h}$ is an




(a) Spatial Dimensions

(b) Temporal Dimension

Figure 4.10: PSD graphs for samples generated via our model trained with the square root transformation, samples generated via our model trained with no transformation, and the test set from UKCP18. The format is the same as Figure 4.6.

element-wise transformation, the Jacobian matrix is diagonal:

$$\frac{d\mathbf{h}(\mathbf{x})}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial \tilde{x}_1}{\partial x_1} & 0 & \cdots & 0 \\ 0 & \frac{\partial \tilde{x}_2}{\partial x_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial \tilde{x}_D}{\partial x_D} \end{bmatrix} \tag{4.3}$$

The determinant of a diagonal matrix is simply the product of its diagonal elements; thus, we can write:

$$\det\left(\frac{d\mathbf{h}(\mathbf{x})}{d\mathbf{x}}\right) = \prod_{i=1}^{D} \frac{\partial h(x_i)}{\partial x_i}. \tag{4.4}$$

Therefore, we can rewrite the change of variables rule under transformation $\mathbf{h}$ in the following simplified form:

$$p_\theta(\mathbf{x}) = \tilde{p}_\theta(\mathbf{h}(\mathbf{x})) \prod_{i=1}^{D} \frac{\partial h(x_i)}{\partial x_i} \tag{4.5}$$

We can thus formulate the negative log-likelihood of $\mathbf{x}$ as follows:

$$-\log p_\theta(\mathbf{x}) = -\log\left(\tilde{p}_\theta(\mathbf{h}(\mathbf{x})) \prod_{i=1}^{D} \frac{\partial h(x_i)}{\partial x_i}\right) \tag{4.6}$$

$$= -\log \tilde{p}_\theta(\mathbf{h}(\mathbf{x})) - \log\left(\prod_{i=1}^{D} \frac{\partial h(x_i)}{\partial x_i}\right) \tag{4.7}$$

$$= -\log \tilde{p}_\theta(\mathbf{h}(\mathbf{x})) - \sum_{i=1}^{D} \log\left(\frac{\partial h(x_i)}{\partial x_i}\right) \tag{4.8}$$

### 4.3.2 ELBO Loss with a Change of Variable

The ELBO loss for a transformed datapoint $\mathbf{h}(\mathbf{x})$, denoted $\tilde{\mathcal{L}}_{\text{ELBO}}(\mathbf{h}(\mathbf{x}))$, serves as a variational bound on the negative log-likelihood of $\mathbf{h}(\mathbf{x})$ and is given by:

$$-\log \tilde{p}_\theta(\mathbf{h}(\mathbf{x})) \leq \tilde{\mathcal{L}}_{\text{ELBO}}(\mathbf{h}(\mathbf{x})) = \mathbb{E}_{\mathbf{z}_0,\ldots,\mathbf{z}_1 \sim \tilde{q}(\mathbf{z}_0,\ldots,\mathbf{z}_1|\mathbf{h}(\mathbf{x}))}\left[\log \frac{\tilde{p}_\theta(\mathbf{z}_0,\ldots,\mathbf{z}_1,\mathbf{h}(\mathbf{x}))}{\tilde{q}(\mathbf{z}_0,\ldots,\mathbf{z}_1|\mathbf{h}(\mathbf{x}))}\right] \tag{4.9}$$

Plugging the inequality into Equation 4.8, we obtain:

$$-\log p_\theta(\mathbf{x}) \leq \tilde{\mathcal{L}}_{\text{ELBO}}(\mathbf{h}(\mathbf{x})) - \sum_{i=1}^{D} \log\left(\frac{\partial h(x_i)}{\partial x_i}\right) \tag{4.10}$$

Thus, if we were to parameterise the transformation $\mathbf{h}_\omega$, with transformative parameters $\omega$, we would theoretically be able to learn some optimal transformation that enables us to minimise the negative log-likelihood of the observed variable $\mathbf{x}$. For completeness, the loss with transformative parameters $\omega$ yields only a minor modification to Equation 4.10 and is given by:

$$p_\theta(\mathbf{x}) \leq \tilde{\mathcal{L}}_{\text{ELBO}}(\mathbf{h}_\omega(\mathbf{x})) - \sum_{i=1}^{D} \log\left(\frac{\partial h_\omega(x_i)}{\partial x_i}\right) \tag{4.11}$$

However, as detailed in Section 2.7.8, optimising our generative parameters $\theta$ via the ELBO loss yields suboptimal results. Diffusion models in the broader literature have achieved far superior results via optimisation with the weighted loss $\mathcal{L}_{\text{WL}}$.

### 4.3.3 Weighted Loss with a Change of Variable

The $\lambda$-dependent weighting function $w(\lambda)$ in the weighted loss $\mathcal{L}_{\text{WL}}$ causes it to lose a direct theoretical relationship with the negative log-likelihood of the observed variable $\mathbf{x}$. Thus, we cannot directly apply the change of variable approach to the weighted loss as we did for the ELBO loss. However, we can consider the individual contribution of the negative log-determinant term in the loss given in Equation 4.11 to derive an empirically good loss function for our diffusion model that jointly optimises the generative parameters $\theta$ and the transformative parameters $\omega$. Importantly, since we constrain the output range of $h_\omega$ to be $[-1, 1]$, the neural network cannot infinitely extend the output range. As such, it has to learn some optimal monotonically increasing function to minimise:

$$-\sum_{i=1}^{D} \log\left(\frac{\partial h_\omega(x_i)}{\partial x_i}\right) \tag{4.12}$$

Intuitively, since the log function itself is monotonically increasing, the negative log-determinant term will encourage a steeper gradient for areas of the input space $[-1, 1]$ wherein individual elements $x_i$ occur most often. Conversely, the term will individually encourage a shallower gradient for areas of the input space $[-1, 1]$ wherein individual elements $x_i$ occur least often. Using this intuition, we hypothesised that we could add the negative log-determinant term to the weighted loss to jointly optimise the generative parameters $\theta$ and the transformative parameters $\omega$ to achieve higher-quality samples. We denote the weighted loss for a transformed datapoint $\mathbf{h}_\omega(\mathbf{x})$ by $\tilde{\mathcal{L}}_{\text{WL}}(\mathbf{h}_\omega(\mathbf{x}))$, and thus our new loss function is given by:

$$\tilde{\mathcal{L}}_{\text{WL}}(\mathbf{h}_\omega(\mathbf{x})) - \sum_{i=1}^{D} \log\left(\frac{\partial h_\omega(x_i)}{\partial x_i}\right) \tag{4.13}$$

### 4.3.4 Transformation Neural Network

We utilise a simple neural network with two fully-connected hidden layers to learn the transformation $h_\omega$ from Equation 4.13. The network contains only a single neuron in both the input and output layers and 16 neurons in each hidden layer; we use the tanh activation function for each neuron in both hidden layers and for the output neuron. We restrict all the weights—not including biases—to be positive to ensure the learnt transformation is monotonically increasing. Since the logarithm of zero is undefined, we add a small multiple of the input to the network's output to prevent the gradient from being zero at any point. We scale the output to the range $[-1, 1]$.

### 4.3.5 Improved Weighted Loss with a Change of Variable

We found that, in practice, the weighted loss with a change of variable given in Equation 4.13 gave rise to poor-quality samples. The transformation it induced was too aggressive, with a significantly large gradient at the lowest end of the input space around $x_i = -1$. Consequently, except for that region, the learnt transformation was $h_\omega(x_i) \approx 1$. As such, the loss function gave rise to the effect depicted in Figure 4.11. Constraining much of the input space to $h_\omega(x_i) \approx 1$ prevents the model from learning the distribution of the transformed observed variable $\mathbf{h}_\omega(\mathbf{x})$ correctly. As such, the samples contain unnaturally high precipitation rates: many areas depicted in Figure 4.11 receive over half a metre of precipitation over the ten hours.

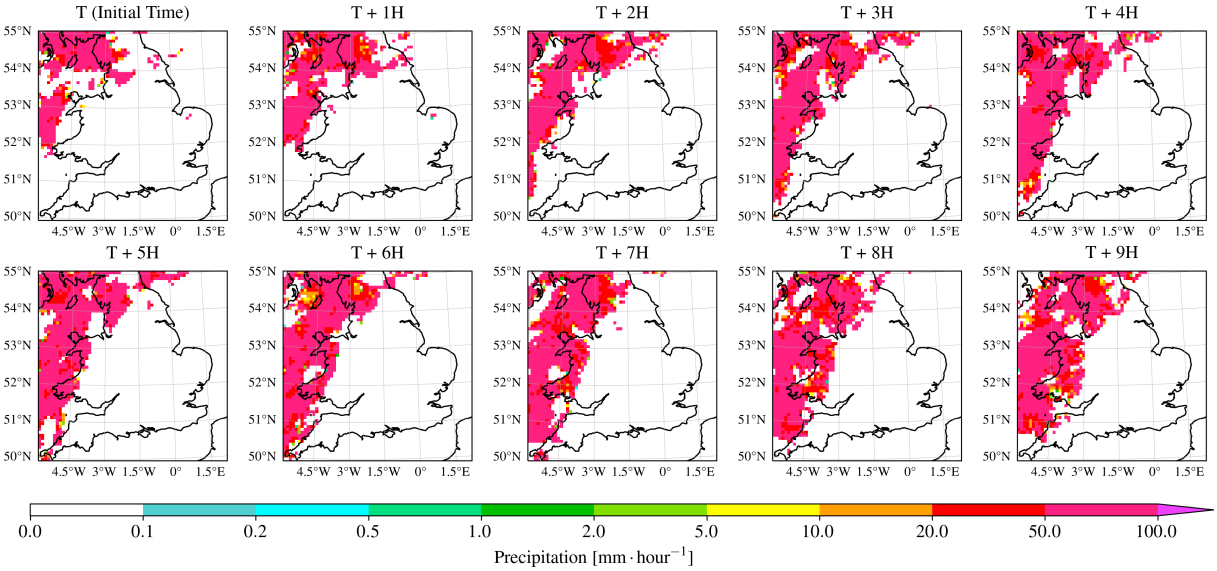



Figure 4.11: Randomly selected sample generated via our model trained with the loss function given in Equation 4.13. The format is the same as Figure 4.2a.

To address this issue, we introduce two interventions which, in conjunction, we found to yield a significant improvement in the quality of the samples generated by our model. Firstly, we significantly increase the multiple by which we add the input to the transformation network to its output. Consequently, at no point is the gradient of the transformation less than a predetermined value. We thus refer to this small multiple as our *minimum gradient coefficient*. Empirically, we found a minimum gradient coefficient of 0.1 yielded good results.

Secondly, we introduce a *masking function*, denoted $m : [-1, 1] \to \{0, 1\}$, that takes a single element $x_i$ of some datapoint $\mathbf{x}$ as input and outputs a binary value:

$$m(x_i) = \begin{cases} 1 & \text{if } x_i \in \mathcal{M} \\ 0 & \text{otherwise} \end{cases} \tag{4.14}$$

where $\mathcal{M} \subseteq [-1, 1]$ is a pre-defined masking set with no learnable parameters, which we use to assert greater control over the transformation learnt by our transformation neural network. We incorporate the masking function into our loss given in Equation 4.13 to yield:

$$\tilde{\mathcal{L}}_{\text{WL}}(\mathbf{h}_\omega(\mathbf{x})) - \sum_{i=1}^{D} \log\left(\frac{\partial h_\omega(x_i)}{\partial x_i}\right) \cdot m(x_i) \tag{4.15}$$

As such, we can use the masking function and masking set to influence the gradient of the learnt transformation. Only elements $x_i$ within the masking set $\mathcal{M}$ will contribute to the negative log-determinant term in the loss. On one end of the scale, if we were to set $\mathcal{M} = [-1, 1]$, this would equate to the loss of Equation 4.13. Conversely, if we were to set $\mathcal{M} = \emptyset$, then the log-determinant term would be redundant—it would never contribute to the loss. Empirically, we found $\mathcal{M} = [-0.99975, 1]$ prevented the transformation network from learning too extreme a transformation. In simple terms, we prevent the elements of our training set that are extremely small—to the extent that they can almost be considered 'dry' for all practical intents and purposes—contributing to the transformation. Figure 4.12 shows the transformation learnt by our model trained with the loss function given in Equation 4.15.

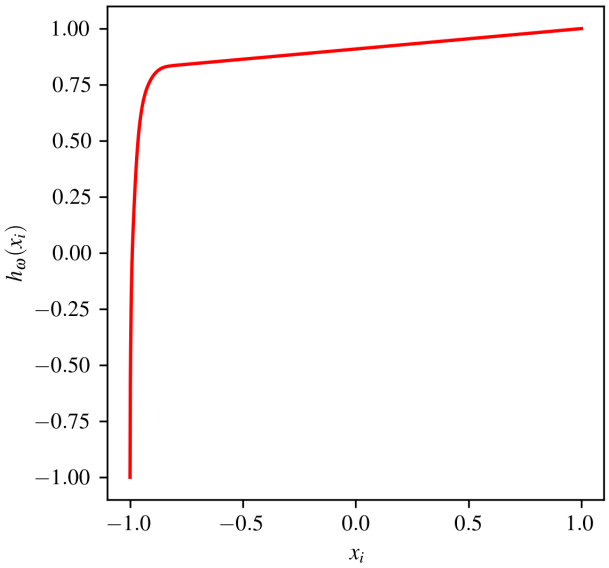


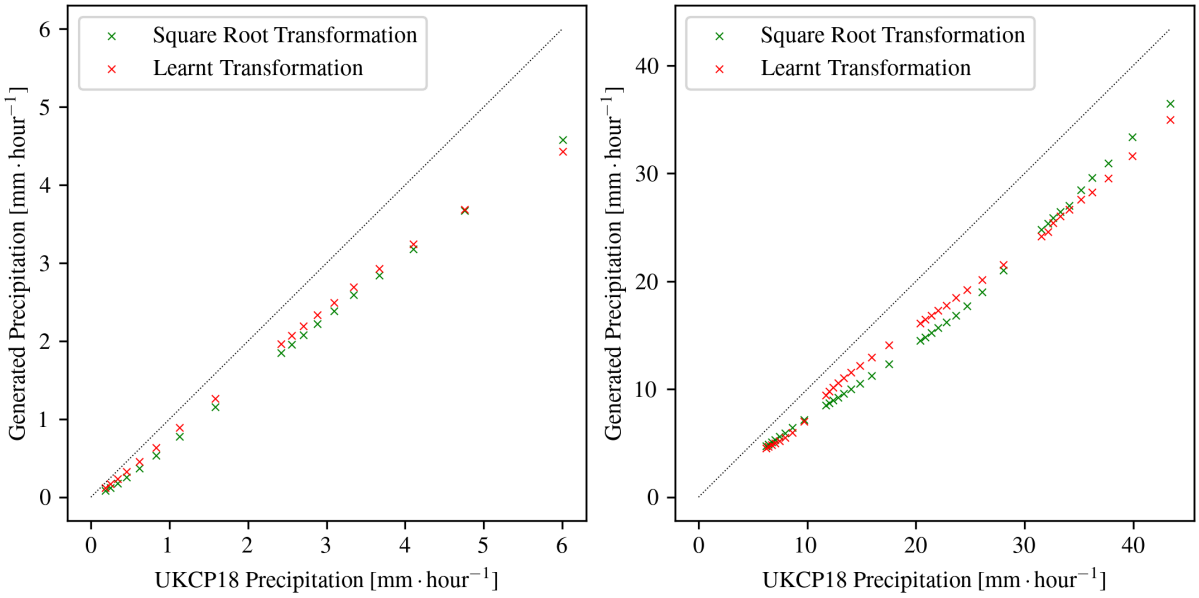

Figure 4.12: The transformation learnt by our model trained with the loss function given in Equation 4.15. The horizontal axis is the value of some element $x_i \in [-1, 1]$ input to the transformation network; the vertical axis is the corresponding output $h_\omega(x_i) \in [-1, 1]$.

Importantly, for both of these solutions, we did not conduct any systematic hyperparameter search; thus, it is likely that more optimal combinations of the minimum gradient coefficient and masking set exist.

### 4.3.6 Evaluation of the Learnt Transformation

To evaluate the performance of our model with the learnt transformation, we primarily focus our analysis against benchmarks set by the square root transformation since it outperformed our model trained with no transformation by most measures. As with the other models, one area of our focus is the model's ability to generate samples wherein the frequency of individual precipitation events is close to the true frequency of those events. We thus compare the quantiles above the dry threshold of the samples generated by our model that learnt its transformation against those generated by our model trained with the square root transformation. Notably, up to the 0.999th quantile, samples generated by our model that learnt its transformation

are closer to the target than those from the square root transformation, as shown in Figure 4.13. The superior performance in this respect suggests that enabling our model to learn its transformation enables it to learn the frequency of most precipitation events more accurately. However, we cannot definitively say that either model has more effectively learnt the frequency of heavier precipitation events, corresponding to quantiles above 0.999. As evident from Figure 4.13, neither model is consistently closer to the target for all quantiles. In one respect, this only represents approximately 0.1% of precipitation events in our dataset. However, as mentioned previously, performing closer to the target in this region may be more desirable in certain use cases—for example, flood forecasting.




(a) Quantiles: $\{q \in \mathcal{Q} \mid 0.91 \leq q \leq 0.999\}$ (b) Quantiles: $\{q \in \mathcal{Q} \mid 0.9991 \leq q \leq 0.9999999\}$

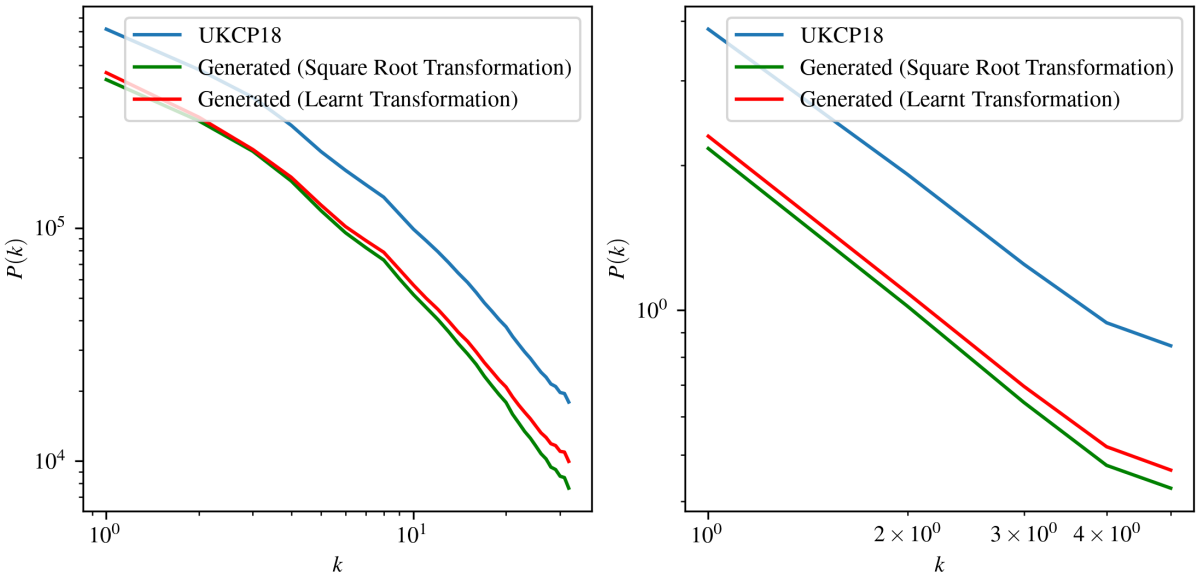
Figure 4.13: Quantile-quantile plots for individual cells within the test set from UKCP18 against both samples generated via our model trained with the square root transformation and samples generated via our model that learnt its transformation. The format is the same as Figure 4.4. $\mathcal{Q} = \{1 - 10^{-n_1} + n_2 \cdot 10^{-n_1-1} \mid n_1, n_2 \in \mathbb{N}, 1 \leq n_2 \leq 9\}$.

Although the dominance of each model may be context-dependent regarding the frequency of individual precipitation events, our study demonstrates that our model employing a learnt transformation outperforms consistently in terms of the power in its samples. More specifically, at any given temporal or spatial resolution, the power in samples produced via our model that learnt its transformation is closer to the power in the test set from UKCP18 than those produced via the square root transformation. The dominance of the learnt transformation is evident in Figure 4.14; the learnt transformation notably outperforms the square root transformation at low spatial scales. The consistent outperformance across all scales suggests that enabling the model to learn its transformation empowers it to learn precipitation's spatial and temporal characteristics more effectively.

We observe a particular improvement in the mean-normalised bias in almost all 8.8 km × 8.8 km subareas. Mean-normalised bias is the sample mean subtracted by the target mean divided by the target mean. As such, it provides a strong indication as to whether

Figure 4.15 depicts the mean-normalised bias at each area for samples produced via our model that learnt its transformation and for the square root transformation.

The superior performance of our model that learnt its transformation is most evident from Figure 4.15, which depicts the mean-normalised bias at each cell representing the same geographical area over time and all samples. The plots indicate the degree to which the models capture




(a) Spatial Dimensions (b) Temporal Dimension

Figure 4.14: PSD graphs for samples generated via our model trained with the square root transformation, our model that learnt its transformation, and the test set from UKCP18. The format is the same as Figure 4.6.

the true amount of precipitation over different geographical areas. For almost all 8.8 km × 8.8km subareas, the mean-normalised bias

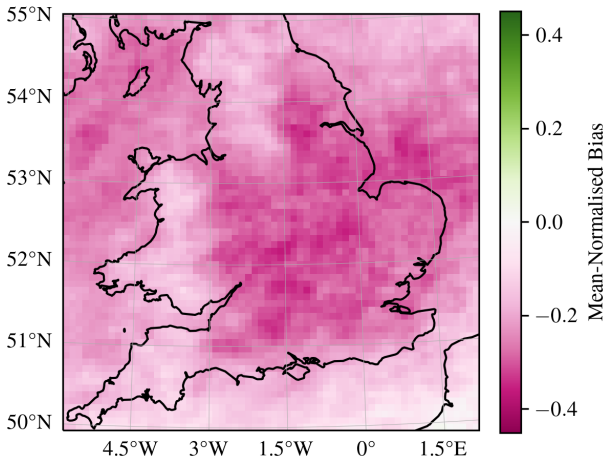



Figure 4.15: Mean-normalised bias at each cell representing the same 8.8 km × 8.8 km geographical area in the samples generated via our model that learnt its transformation.

### 4.3.7 Samples

## 4.4 Temporal Interpolation

In Section 2.7.10, we introduced the notion of deriving a conditional diffusion model approximately from an unconditional model by employing reconstruction-guided sampling [5]. In this section, we illustrate how this method can facilitate *temporal interpolation*: the generation of unknown hourly precipitation samples that maintain temporal coherency with adjacent samples at a lower temporal resolution. Specifically, we showcase the generation of hourly precipita-

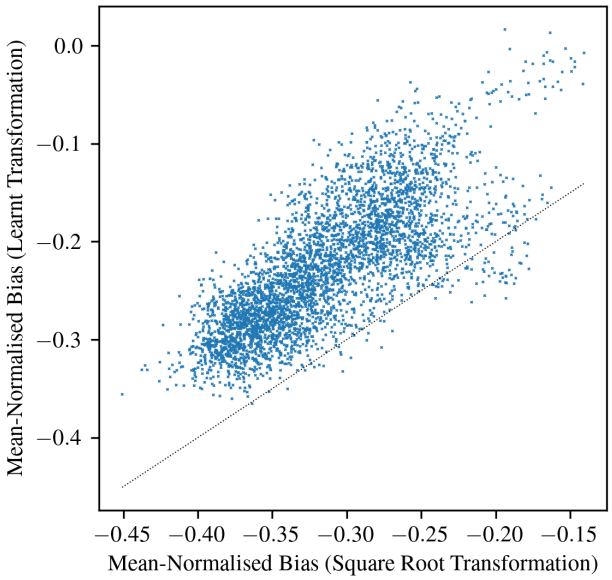



Figure 4.16: Relationship between the mean-normalised bias at each cell representing the same 8.8 km × 8.8 km geographical area in the samples generated via our model trained with the square root transformation and our model that learnt its transformation. The horizontal axis is the mean-normalised bias for the square root transformation; the vertical axis is the mean-normalised bias for the learnt transformation. The black line is $y = x$; points above the line indicate that the mean-normalised bias for the learnt transformation is lower than for the square root transformation.

tion samples that exhibit temporal coherence with adjacent samples at a two-hour temporal resolution.

Temporal interpolation of precipitation data could facilitate many real-world applications. Consider a scenario wherein a region only has precipitation data at a low temporal resolution. With the proposed method, it would be possible to generate hourly precipitation samples that maintain temporal coherency with the existing two-hour interval data. By interpolating the data to a finer temporal resolution, meteorologists, hydrologists, and other professionals could improve their understanding of local climate patterns and thus improve their flood forecasting and management abilities.

## 4.5 Forecasting

## 4.6 Autoregressive Generation of Longer Sequences

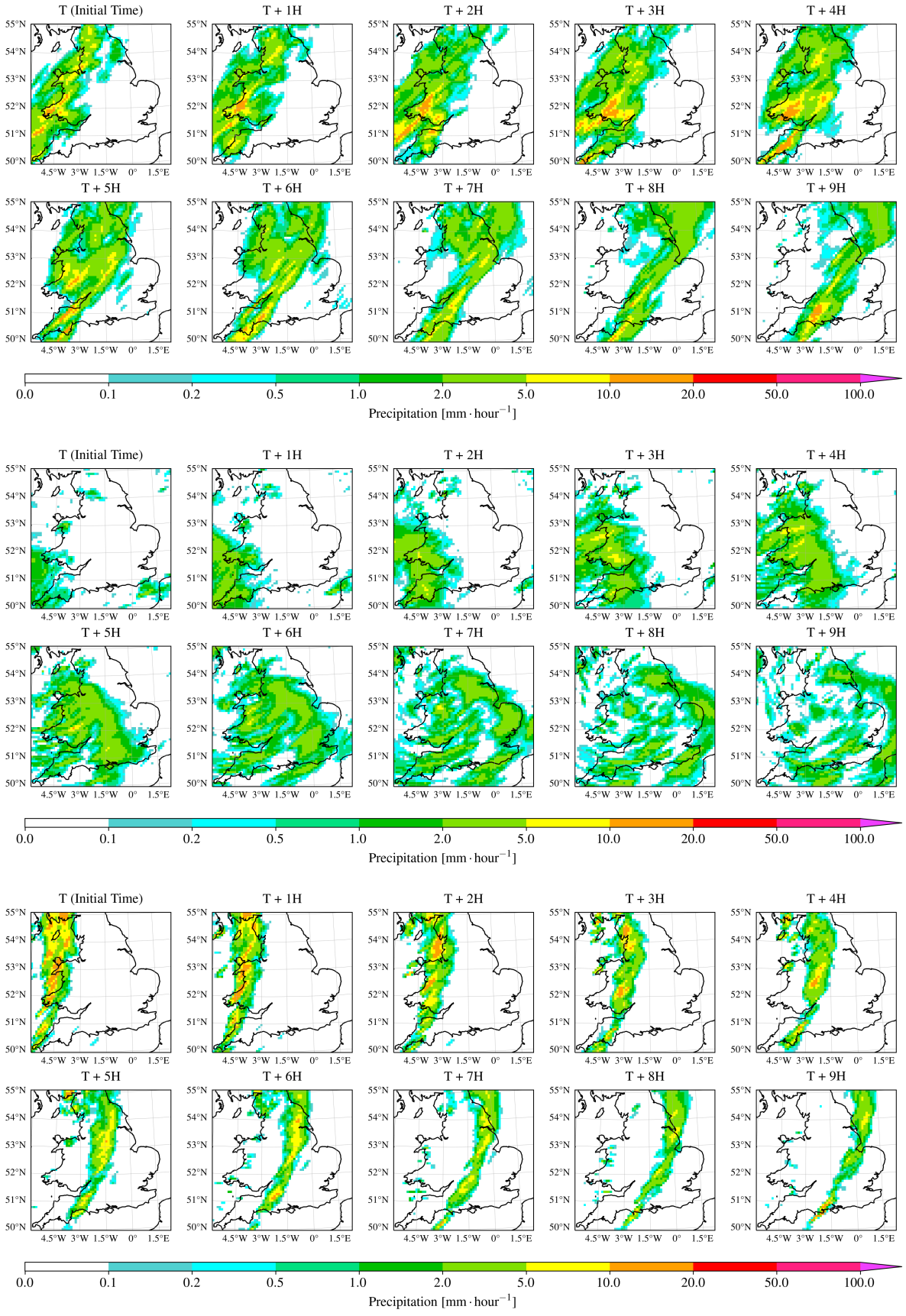



Figure 4.17: Three samples generated via our model trained with the loss function given in Equation 4.15. The format is the same as Figure 4.2a.

# 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] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[10] Met Office Hadley Centre. UKCP18 guidance: Representative concentration pathways. Technical report, 2018.

[11] Met Office Hadley Centre. UKCP18 probabilistic climate projections, 2018.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3738–3746, 2016.

[18] 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.

[19] 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 2.42, 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:

$$\mathbf{z}_t = \alpha_t \mathbf{x} + \sqrt{\sigma_t^2}\boldsymbol{\epsilon}_t \tag{A.2}$$

$$= \alpha_t \mathbf{x} + \sqrt{\sigma_t^2 - \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2 + \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2}\boldsymbol{\epsilon}_t \tag{A.3}$$

$$= \alpha_t \mathbf{x} + \sqrt{\sigma_t^2 - \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2 + \left(\frac{\alpha_t}{\alpha_s}\sigma_s\right)^2}\boldsymbol{\epsilon}_t \tag{A.4}$$

The sum of two independent Gaussian random variables with mean $\mu_1$ and $\mu_2$ and variance $\sigma_1^2$ and $\sigma_2^2$ 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:

$$\mathbf{z}_t = \alpha_t \mathbf{x} + \sqrt{\sigma_t^2 - \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2}\boldsymbol{\epsilon}_t^* + \frac{\alpha_t}{\alpha_s}\sigma_s\boldsymbol{\epsilon}_s \tag{A.5}$$

$$= \alpha_t \mathbf{x} + \frac{\alpha_t}{\alpha_s}\sigma_s\boldsymbol{\epsilon}_s + \sqrt{\sigma_t^2 - \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2}\boldsymbol{\epsilon}_t^* \tag{A.6}$$

$$= \frac{\alpha_s}{\alpha_s}\alpha_t \mathbf{x} + \frac{\alpha_t}{\alpha_s}\sigma_s\boldsymbol{\epsilon}_s + \sqrt{\sigma_t^2 - \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2}\boldsymbol{\epsilon}_t^* \tag{A.7}$$

$$= \frac{\alpha_t}{\alpha_s}(\alpha_s \mathbf{x} + \sigma_s\boldsymbol{\epsilon}_s) + \sqrt{\sigma_t^2 - \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2}\boldsymbol{\epsilon}_t^* \tag{A.8}$$

$$\tag{A.9}$$

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:

$$\mathbf{z}_t = \frac{\alpha_t}{\alpha_s}\mathbf{z}_s + \sqrt{\sigma_t^2 - \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2}\boldsymbol{\epsilon}_t^* \tag{A.10}$$

$$= \alpha_{t|s}\mathbf{z}_s + \sigma_{t|s}\boldsymbol{\epsilon}_t^* \tag{A.11}$$

$$\sim \mathcal{N}\left(\mathbf{z}_t; \alpha_{t|s}\mathbf{z}_s, \sigma_{t|s}^2\mathbf{I}\right) \tag{A.12}$$

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 $\alpha$-Cosine Noise Schedule

Before truncation, the continuous-time version of the $\alpha$-cosine schedule [12] as described in [6] defines $\alpha_t^2$ 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, $\lambda_t$ is unbounded. We follow prior work (see e.g. [6, 5]) by truncating $\lambda_t$ 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 $\psi$, which itself is:

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

when using the truncated continuous-time $\alpha$-cosine noise schedule as per Section 2.7.4. As such, we can formulate the velocity as:

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

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

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

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:

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