Wavelets for diffusion models

Felix Blanke

Born August 30, 1999 in Frankfurt am Main

January 22, 2024

Master's Thesis Mathematics

Advisor: Prof. Dr. Jochen Garcke

Second Advisor: Prof. Dr. Christian Bauckhage

Institut für Numerische Simulation

Fraunhofer-Institut für Algorithmen und Wissenschaftliches Rechnen

Mathematisch-Naturwissenschaftliche Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Contents

Acronyms

CNN convolutional neural network

DDPM denoising diffusion probabilistic model

DWT discrete wavelet transform

EMA exponential moving average

FFT fast Fourier transform

FID Fréchet Inception distance

FIR filter finite impulse response filter

FWT fast wavelet transform

GAN generative adversarial network

GPU graphics processing unit

iFWT inverse fast wavelet transform

KL divergence Kullback-Leibler divergence

MRA multiresolution analysis

ODE ordinary differential equation

SDE stochastic differential equation

SMLD denoising score matching with Langevin dynamics

WPT wavelet packet transform

WSGM wavelet score-based generative model

Introduction

Diffusion models are a class of probabilistic models that allow sampling from complex data distributions. They are the current state of the art models for image generation [DN21]. They have also gathered attention from the general public through the astonishing quality of images that current models are able to generate [Pod+23]. In particular, they are known for the sampling of images conditioned on text input [Ram+22]. However, we focus on unconditional image generation.

As diffusion models are at the cutting edge of research into generative models the field is very fast moving. Introduced by Sohl-Dickstein et al. [SWMG15] they started to gain traction in 2020 [HJA20].

In contrast, wavelets are a field-tested concept from functional analysis [SN96]. Their name means "small wave" which describes their nature: a function that allows for localization in space *and* frequency. This very useful notion has wide applications in classical signal processing as well as "modern" deep learning. One particular application wavelets are used in is denoising, due to the separation of frequency components they enable. As we will see, this is the task that diffusion models essentially have to solve in order to generate image samples. Furthermore, we will see that wavelets help us to address a longstanding problem for generative image models: the frequency bias. This means that generative images tend to show discrepencies in the spectral distribution to the distribution of real images, particularly in the high frequencies.

In addition to being local in space and time, wavelets have another characteristic property: they work on multiple scales, i.e. they successively approximate signals by adding finer and finer details. We will see that embracing this multiscale structure allows us to spend significantly less time to sample images of the same fidelity. So far, this approach was only tested with the most simple wavelet and the same approximation effort on all scales. We will evaluate possible ways to extend the method.

However, quantifying the fidelity of generated images is a hard task. As we will discuss in this work, wavelets are the key to one approach to quantitative comparison of the generated images to real ones. However, it will turn out that the originally proposed way is flawed, requiring us to construct a mitigation. This is probably the most exciting section of this work.

Structure of the work

This work is structured into five chapters. The first chapter introduces foundational notions of wavelet theory. To this end, we first introduce filters and filter banks (section 1.1). We discuss conditions posed on filter banks which allow us to construct a basis of $L^2(\mathbb{R})$ that is localized in space and frequency—the so-called wavelet basis (section 1.2). Further,

we introduce two important wavelet transformations: the fast wavelet transform and the wavelet packet transform. In section 1.3 we introduce multiple boundary handling techniques that enable us to apply wavelet transformations to signals of finite length.

The second chapter is concerned with the introduction of diffusion models. Starting with a in discrete-time diffusion model (section 2.1) we derive in detail how it is trained and consider different interpretations of the optimization goal. This allows us to draw a connection to diffusion models in continuous-time (section 2.2) which use SDEs to formulate the diffusion model. Lastly, we discuss different choices of the noise schedule which dictates the behaviour of the diffusion model.

In the third chapter we combine the foundational notions introduced in the previous sections and discuss how wavelets can be used in the context of diffusion models. First, we introduce a model that moves the diffusion model into a multiscale wavelet domain (section 3.1). As we will see, this allows for the formulation of theoretical guarantees on required number of steps. Then, we will discuss a way of also adopting the wavelet domain structure into the underlying architecture (section 3.2).

The fourth chapter experimentally examines extensions of the multiscale wavelet diffusion model we have introduced to wavelets of higher order, different boundary handling techniques and varying number of discretization steps. To be able to measure the effects, we discuss the standard metric for quantitatively evaluation of generated images and its flaws. We then analyze a method based on the comparison of wavelet signal representations and propose two variants.

Finally, chapter five contains a short summary of the work and the presented results, discusses the findings and offers an outlook.

Own contribution

The following contributions can be found in this work:

- review of the basics of diffusion models, in particular in discrete-time (chapter 2),
- theoretical analysis of the Fourier and wavelet packet power spectrum Kullback-Leibler divergences [VWG23]; identification and empirical validation of a flaw in the application of the metric to image batches
- proposal of (i) a mitigation of the flaw in the metrics and (ii) of an extension to evaluate the spectral and spatial energy distribution simultanuously,
- implementation of WSGMs as well as frequency-spatial convolution and attention blocks in the modular diffusers framework [Pla+22] to facilitate reusability,
- extension of the wavelet-based diffusion model WSGM to higher-order wavelets with different boundary handling techniques and discretizations
- discussion of the obtained results.

The own contribution in chapter 1 is limited as we follow Blanke [Bla21] very closely in the representation of the results.

Acknowledgments

I would like to take this opportunity to express my gratitude for the support I have received. First of all, I would like to thank Prof. Dr. Jochen Garcke for his supervision, patience and the opportunity to work on such a dynamic research topic. I would also like to thank Prof. Dr. Christian Bauckhage for taking over the second correction and to Dr. Moritz Wolter for the stimulating discussions and inspiration. To the Fraunhofer SCAI I would like to express my gratitude for the access to the Löwenburg computing cluster for this project in the scale necessary for the training of diffusion models. My heartfelt thanks go to Thomas Blanke, Hanna Blanke, Lorenzo Conti, and Gina Muuss for attentive proofreading at a late hour and Katharina Axtmann for her indulgence.

1 Foundations of wavelet theory

This chapter introduces foundational notions of wavelet theory. The presentation of contents is adapted closely from Blanke [Bla21] which is based on Strang and Nguyen [SN96].

The starting point for wavelet theory are signals with finite energy, i.e. functions $f \in L^2(\mathbb{R}^d)$. Our goal is to construct an orthonormal basis of $L^2(\mathbb{R}^d)$ from functions localized in time and frequency, so-called wavelet functions. Fundamental for this construction is an important duality between the wavelet transform on continuous-time signals and operators on discrete-time signals. Hence, we first introduce some basic notions from signal theory on discrete-time signals in section 1.1. In section 1.2 we consider continuous-time signals and highlight said duality which leads us to the wavelet basis and a fast algorithm for its calculation.

In this chapter we first constrict ourselves to one-dimensional signals. Thinking of audio signals as an intuition might be helpful. In section 1.2.5, we discuss how the introduced notions can be extended to higher-dimensional signals. In particular we discuss two-dimensional signals as our experimental setup is concerned with the synthesis of images.

1.1 Filters and filterbanks

Definition 1.1. A (discrete-time) *signal* $\mathbf{x}$ is a finite-element sequence, i.e. an element of

$$\ell^2(\mathbb{Z}) := \Big\{\mathbf{x}\colon \mathbb{Z} \to \mathbb{R} \;\Big|\; \sum_{n\in\mathbb{Z}} |\mathbf{x}(n)|^2 < \infty\Big\}.$$

We note ℓ^2 as a shorthand for $\ell^2(\mathbb{Z})$. As an index set we utilize $\mathbb{Z}$ so that the sequence does not end for both rising and falling indices.

Remark. We can interpret discrete-time signals as the result of a discretization of continuous-time signals $f \in L^2(\mathbb{R}^d)$ on a regular grid [SN96], as depicted in figure 1.1.

An important signal is the so-called unit impulse δ:

Definition 1.2. The *unit impulse* $\delta \in \ell^2$ is defined as

$$\delta(n) := \begin{cases} 1 & n = 0, \\ 0 & \text{else.} \end{cases}$$

Definition 1.3. A *filter* $\mathbf{H}\colon \ell^2 \to \ell^2$ is a linear operator that transforms a signal by convolution with a fixed signal $\mathbf{h} \in \ell^2$. For a signal $\mathbf{x} \in \ell^2$ we have

$$[\mathbf{H}\mathbf{x}](n) = [\mathbf{h} * \mathbf{x}](n) = \sum_{k\in\mathbb{Z}} \mathbf{h}(k)\mathbf{x}(n-k).$$

We call the fixed signal $\mathbf{h}$ *impulse response* as it is the result of applying a filter to the unit impulse. The length of a filter is defined as the length of its impulse response.

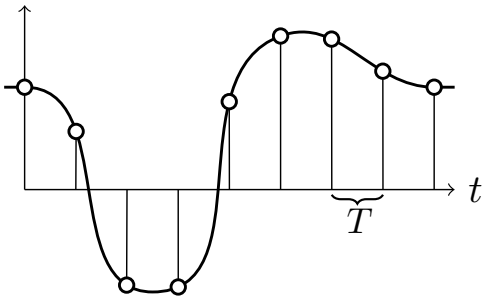


Figure 1.1: An exemplary signal on $\mathbb{R}$ as well a discretized signal with a sampling rate of T.

Definition 1.4. An important special case are *finite impulse response filters* (*FIR filters*), i.e. filters with a finite length.

These filters are important in practice, as encoding filters by their impulse response and calculating the convolution with the impulse response is infeasible for impulse responses of infinite length.

Definition 1.5. We call a filter *causal* if its impulse response is zero for all negative indices.

Example 1.6. Two filter that will appear repeatedly in this thesis are the *moving average* $\mathbf{H}_0$ and the *moving difference* $\mathbf{H}_1$. The moving average is a causal FIR filter with impulse response

$$\mathbf{h}_0(n) = \begin{cases} \frac{1}{2} & n = 0 \text{ or } n = 1, \\ 0 & \text{else.} \end{cases} \tag{1.1}$$

Thus applying this filter corresponds to the average of the current signal value $\mathbf{x}(n)$ and the previous value $\mathbf{x}(n-1)$:

$$\mathbf{H}_0\mathbf{x}(n) = \frac{1}{2}\mathbf{x}(n) + \frac{1}{2}\mathbf{x}(n-1)$$

The moving difference has the impulse response

$$\mathbf{h}_1(n) = \begin{cases} \frac{1}{2} & n = 0, \\ -\frac{1}{2} & n = 1, \\ 0 & \text{else.} \end{cases} \tag{1.2}$$

and is thus also a causal FIR filter. The output of the filter is the difference of the current signal value $\mathbf{x}(n)$ and the previous value $\mathbf{x}(n-1)$:

$$\mathbf{H}_1\mathbf{x}(n) = \frac{1}{2}\mathbf{x}(n) - \frac{1}{2}\mathbf{x}(n-1)$$

A visualization of the application of both filters to the exemplary signal from figure 1.1 is depicted in figure 1.2.

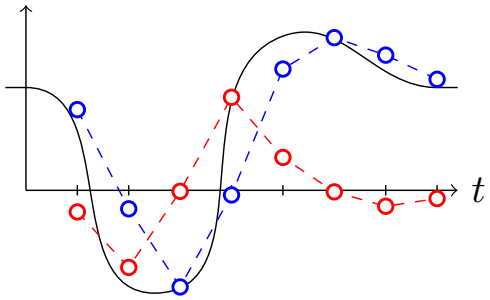


Figure 1.2: Application of the moving aveage (blue) and the moving difference (red) to the examplary signal from figure 1.1.

The above representation of a filter is set in the "time domain", associating a filter with its impulse response. The application of the filter corresponds to a convolution of the input signal with the impulse response.

There are two further ways of representing a filter that will be of use in this work: the representation in the "frequency domain" using the discrete Fourier transform and the representation as an infinite so-called Toeplitz matrix (cf. section 1.1.2).

1.1.1 Filters in the frequency domain

Definition 1.7. The *Fourier transformation* of a signal $\mathbf{x} \in \ell^2$ is

$$X(\omega) = \sum_{k\in\mathbb{Z}} \mathbf{x}(k)\mathrm{e}^{-\mathrm{i}k\omega}.$$

Lemma 1.8. *For $\mathbf{x} \in \ell^2$ exists a unique Fourier transformation $X \in L^2([-\pi, \pi])$.*

Proof. This follows directly from the Fischer-Riesz theorem [e.g. Bea04, Theorem 13.12]. □

Lemma 1.9 (Strang and Nguyen [SN96, p. 5])**.** *The Fourier transformation of a filtered signal equals the multiplication of the input signal's Fourier transformation with the Fourier transformation of the filter's impulse response.*

We can thus also associate a filter with the Fourier transformation H of its impulse response. We call H the *frequency response* of the filter.

Example 1.10. The frequency response of the moving average $\mathbf{H}_0$ is

$$H_0(\omega) = \sum_{k\in\mathbb{Z}} \mathbf{h}_0(k)\mathrm{e}^{-\mathrm{i}k\omega} = \frac{1}{2} + \frac{1}{2}\mathrm{e}^{-\mathrm{i}\omega} = \frac{\mathrm{e}^{\mathrm{i}\omega/2} + \mathrm{e}^{-\mathrm{i}\omega/2}}{2}\mathrm{e}^{-\mathrm{i}\omega/2} = \cos\left(\frac{\omega}{2}\right)\mathrm{e}^{-\mathrm{i}\omega/2}.$$

Example 1.11. The frequency response of the moving difference $\mathbf{H}_1$ is

$$H_1(\omega) = \sum_{k\in\mathbb{Z}} \mathbf{h}_1(k)\mathrm{e}^{-\mathrm{i}k\omega} = \frac{1}{2} - \frac{1}{2}\mathrm{e}^{-\mathrm{i}\omega} = \frac{\mathrm{e}^{\mathrm{i}\omega/2} - \mathrm{e}^{-\mathrm{i}\omega/2}}{2}\mathrm{e}^{-\mathrm{i}\omega/2} = \mathrm{i}\sin\left(\frac{\omega}{2}\right)\mathrm{e}^{-\mathrm{i}\omega/2}.$$

1.1.2 Filter representation with Toeplitz matrices

Definition 1.12. An (*infinite*) *Toeplitz matrix* $A = (a_{ij})_{i,j\in\mathbb{Z}}$ is a matrix with constant diagonal entries, i.e.

$$a_{ij} = a_{kl} \quad \text{for all } i,j,k,l \in \mathbb{Z} \text{ with } i-j = k-l.$$

Lemma 1.13. *The application of a filter* $\mathbf{H}$ *to a discrete signal* $\mathbf{x} \in \ell^2$ *corresponds to the* (*infinite*) *matrix vector product of the Toeplitz matrix* $H = (h_{ij})_{i,j\in\mathbb{Z}}$ *with* $\mathbf{x}$ *where* $h_{ij} := \mathbf{h}(i-j)$.

Proof. We can write the matrix vector product $H\mathbf{x}$ as

$$H\mathbf{x} = \begin{bmatrix} \ddots & & & & \\ \cdots & \mathbf{h}(0) & \mathbf{h}(-1) & \mathbf{h}(-2) & \cdots \\ \cdots & \mathbf{h}(1) & \mathbf{h}(0) & \mathbf{h}(-1) & \cdots \\ \cdots & \mathbf{h}(2) & \mathbf{h}(1) & \mathbf{h}(0) & \cdots \\ & & & & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ \mathbf{x}(-1) \\ \mathbf{x}(0) \\ \mathbf{x}(1) \\ \vdots \end{bmatrix}$$

$$= \begin{bmatrix} & & & \vdots & & & \\ \cdots + & \mathbf{h}(0)\mathbf{x}(-1) & + & \mathbf{h}(-1)\mathbf{x}(0) & + & \mathbf{h}(-2)\mathbf{x}(1) & + \cdots \\ \cdots + & \mathbf{h}(1)\mathbf{x}(-1) & + & \mathbf{h}(0)\mathbf{x}(0) & + & \mathbf{h}(-1)\mathbf{x}(1) & + \cdots \\ \cdots + & \mathbf{h}(2)\mathbf{x}(-1) & + & \mathbf{h}(1)\mathbf{x}(0) & + & \mathbf{h}(0)\mathbf{x}(1) & + \cdots \\ \cdots + & \mathbf{h}(3)\mathbf{x}(-1) & + & \mathbf{h}(2)\mathbf{x}(0) & + & \mathbf{h}(1)\mathbf{x}(1) & + \cdots \\ & & & \vdots & & & \end{bmatrix}.$$

For any $n \in \mathbb{Z}$, the n-th entry of $H\mathbf{x}$ is therefore

$$[H\mathbf{x}](n) = \sum_{j\in\mathbb{Z}} h_{nj}\mathbf{x}(j) = \sum_{j\in\mathbb{Z}} \mathbf{h}(n-j)\mathbf{x}(j) = [\mathbf{h} * \mathbf{x}](n),$$

which is the result of applying $\mathbf{h}$ to $\mathbf{x}$. □

Thus, we can identify a filter $\mathbf{H}$ with the corresponding Toeplitz matrix H. If $\mathbf{H}$ is causal, then H has the form of an infinite lower triangular matrix; if $\mathbf{H}$ is an FIR filter, each row of H contains a finite number of entries, namely the entries of the impulse response in reverse order. Hence, the Toeplitz matrix representation of causal FIR filters, to which we will mainly restrict ourselves, has the form of an infinite lower triangular band matrix.

Remark. For a filter $\mathbf{H}$ we have become familiar with three different representations: the impulse response, the frequency response and the Toeplitz matrix. For the rest of the work, we use the following notations: we notate the impulse response as a bold lowercase letter, e.g. $\mathbf{h}$, and the frequency response as an italicized uppercase letter, e.g. H. For the Toeplitz matrix, we overload the notation and write it - just like the filter itself - as a bold capital letter, e.g. $\mathbf{H}$.

1.1.3 Low-pass and high-pass filter

As the name filter suggests, filters are used to remove certain unwanted components from a signal. The two filter types relevant to wavelet theory are the so-called low-pass and

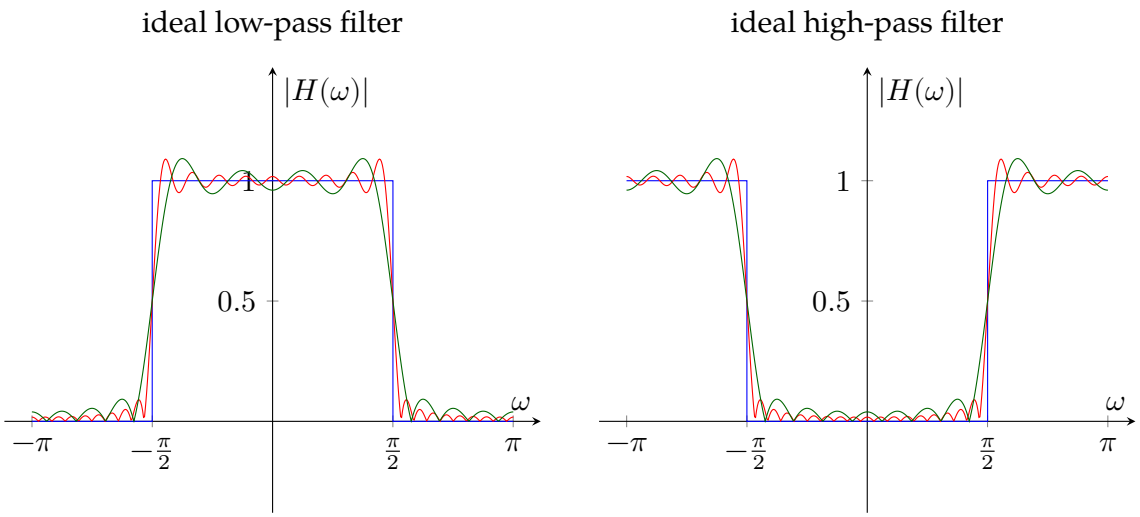


Figure 1.3: Absolute frequency response of the ideal low-pass filter (left) and high-pass filter (right). The absolute frequency response of the ideal filter (blue) and the approximation with the first 10 (green) and 20 (red) terms are shown. The deflections at the jump points $\pm\pi/2$, which are described as the Gibbs phenomenon, are clearly visible.

high-pass filters. These remove low-frequency and high-frequency signal components respectively and leave the remaining signal components unchanged. To do this, we divide the frequency range $[-\pi, \pi]$ into two equal parts: the frequency interval $(-\frac{\pi}{2}, \frac{\pi}{2})$ forms the low frequencies, the rest the high frequencies.

How a filter $\mathbf{H}$ affects different frequency components can be seen in the frequency response H of the filter. The removal of a frequency ω by the filter corresponds to $|H(\omega)| = 0$; retaining the frequency, on the other hand, corresponds to $|H(\omega)| = 1$.

This allows us to define ideal high-pass and low-pass filters:

Definition 1.14. The *ideal low-pass filter* is a filter that leaves the frequency band $(-\frac{\pi}{2}, \frac{\pi}{2})$ unchanged and removes the remaining frequencies. The *ideal high-pass filter* $\mathbf{H}_1$ in turn is a filter that removes the frequency band $(-\frac{\pi}{2}, \frac{\pi}{2})$ and leaves the remaining frequencies unchanged. The frequency responses of $\mathbf{H}_0$ and $\mathbf{H}_1$ have the form

$$H_0(\omega) = \begin{cases} 1 & \omega \in (-\frac{\pi}{2}, \frac{\pi}{2}), \\ 0 & \text{else;} \end{cases} \qquad H_1(\omega) = \begin{cases} 0 & \omega \in (-\frac{\pi}{2}, \frac{\pi}{2}), \\ 1 & \text{else.} \end{cases} \tag{1.3}$$

Lemma 1.15 (Strang and Nguyen [SN96, pp. 45])**.** *The ideal high-pass and low-pass filters have the impulse responses*

$$\mathbf{h}_0(n) = \frac{\sin\frac{\pi n}{2}}{\pi n} = \begin{cases} \frac{1}{2} & n = 0, \\ 0 & n \text{ even}, n \neq 0, \\ (-1)^{k+1}\frac{1}{\pi n} & n = 2k+1; \end{cases} \qquad \mathbf{h}_1(n) = (-1)^n \mathbf{h}_0(n) \tag{1.4}$$

and are thus no FIR filter.

Remark. As the ideal high-pass and low-pass filters are no FIR filter, they are unsuitable for applications. Even finite approximations of the ideal filters, e.g. by restricting them to the first N filter coefficients, are not an optimal solution due to the Gibbs phenomenon [cf. Bea04, section 14A; SN96, pp. 46], as illustrated in figure 1.3. For this reason, various methods for constructing non-ideal FIR high-pass and low-pass filters have been proposed [see SN96, Section 2.3].

Our minimum requirement for non-ideal low-pass filters is that the lowest-frequency signal, i.e. the constant signal, remains unchanged and the highest-frequency discrete signal, i.e. the alternating signal $\mathbf{x}(n) = (-1)^n$, is set to 0. Alternatively, the following must apply to the frequency response H

$$|H(0)| = 1, \qquad |H(\pm\pi)| = 0.$$

This minimum requirement applies in reverse for non-ideal high-pass filters:

$$|H(0)| = 0, \qquad |H(\pm\pi)| = 1.$$

These are minimum criteria. In the course of this work, we will get to know further conditions that a filter must fulfill in order to be suitable for the construction of wavelets.

Example 1.16. We have already seen an example of a non-ideal low-pass and high-pass filter: the moving average and the moving difference. The former calculates the average of two consecutive signal values. Thus, figuratively speaking, rapid changes in the signal are averaged out, resulting in a low-pass filter. As a high-pass filter, the moving difference acts in the opposite way. By forming differences of successive signal values, changes in the signal are highlighted and the "base level" of the signal is removed.

A visualization of the magnitude of the frequency responses of both filters is shown in figure 1.4. We can see that the filters meet the minimum requirements for low-pass and high-pass filters.

1.1.4 Filter banks

A common scheme of signal processing applications is the *analysis* of a signal. This involves transforming a signal into a form that facilitates the extraction of relevant information or further processing. We have already seen an example of such an analysis with the high-pass and low-pass filters. Here, a signal is reduced to its high-frequency or low-frequency component. However, information is lost during this reduction as certain frequency ranges are removed from the signal. The original signal can therefore not be reconstructed.

In some applications, however, it is necessary that no loss of information occurs during the analysis. Another representation of the signal with desirable properties should therefore be found, from which the signal can be reconstructed. We call this reconstruction process *synthesis*.

As discussed, individual high-pass and low-pass filters do not fulfill this property by definition, as certain frequency ranges are removed from the signal. However, if we take a suitable pair of high-pass and low-pass filters together, we do not lose any information. Instead, we simply split the signal into a high-frequency and a low-frequency component,

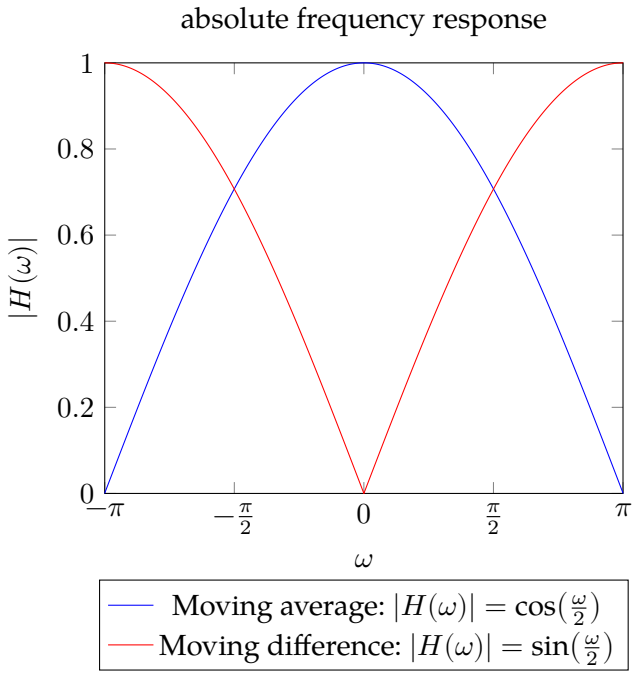


Figure 1.4: Magnitude of the frequency response of the moving average (blue) and the moving difference (red). We recognize that the designations "low-pass filter" or "high-pass filter" are justified in each case

.

which we can then recombine to form the original signal. This brings us to the concept of a *filter bank*. This uses a suitable pair of filters for analysis and synthesis, with the analysis pair consisting of a high-pass filter and a low-pass filter. We also call the analysis pair *analysis bank* and the synthesis pair *synthesis bank*. The analysis takes place by applying both filters of the analysis bank to the signal and thus dividing the signal into a high-frequency and a low-frequency signal component. For the synthesis, we apply a filter from the synthesis bank to each of these two signal components and add them together. This is shown schematically in figure 1.5.

In the following section, we will learn about criteria for choosing suitable filters so that the synthesis part of a filter bank inverts the analysis part.[1].

1.1.5 Sampling in the filter bank

This first concept of a filter bank still has a crucial problem: For an input signal with finite length[2] the filters of the analysis bank each generate a filtered signal of at least the same length.

If both are applied, the required disk space doubles, although the information content

[1]Specifically, we do not require direct invertibility, but allow a delay in the signal during reconstruction to enable causal filters. More on this later.

[2]We discuss exactly how to apply filters to a finite signal in section 1.3.

remains constant. The solution to this problem is downsampling: we discard every second entry of the filter results.

Definition 1.17. The *downsampling* operator $(\downarrow 2)$ removes all odd components from a signal $\mathbf{x}$. For $\mathbf{v} := (\downarrow 2)\mathbf{x}$ we have $\mathbf{v}(k) = \mathbf{x}(2k)$. We thus obtain

$$\mathbf{v} = (\downarrow 2)\begin{bmatrix}\cdots & \mathbf{x}(-1) & \mathbf{x}(0) & \mathbf{x}(1) & \mathbf{x}(2) & \cdots\end{bmatrix} = \begin{bmatrix}\cdots & \mathbf{x}(-2) & \mathbf{x}(0) & \mathbf{x}(2) & \cdots\end{bmatrix}.$$

Lemma 1.18. *Let* $\mathbf{H}$ *be a filter with impulse response* $\mathbf{h}$ *and* $\mathbf{x}$ *a signal. Then*

$$[(\downarrow 2)\mathbf{H}\mathbf{x}](n) = \sum_{k\in\mathbb{Z}} \mathbf{h}(2n-k)\mathbf{x}(k).$$

Proof. Let $\mathbf{v} := \mathbf{H}\mathbf{x} = \sum_{k\in\mathbb{Z}} \mathbf{h}(n-k)\mathbf{x}(k)$. We then have

$$[(\downarrow 2)\mathbf{v}](n) = \mathbf{v}(2n) = \sum_{k\in\mathbb{Z}} \mathbf{h}(2n-k)\mathbf{x}(k). \qquad \square$$

During reconstruction, the synthesis should generate a signal of the same length as the output signal. To compensate for the halving of the signal length through downsampling, we use the upsampling operator.

Definition 1.19. The *upsampling* operator $(\uparrow 2)$ is the counterpart to the downsampling operator and inserts a 0 between all components of a signal $\mathbf{y}$. For $\mathbf{u} := (\uparrow 2)\mathbf{y}$ we have

$$\begin{cases}\mathbf{u}(2k) = \mathbf{y}(k)\\ \mathbf{u}(2k+1) = 0\end{cases}.$$

So we get

$$(\uparrow 2)\begin{bmatrix}\cdots & \mathbf{y}(-1) & \mathbf{y}(0) & \mathbf{y}(1) & \cdots\end{bmatrix} = \begin{bmatrix}\cdots & \mathbf{y}(-1) & 0 & \mathbf{y}(0) & 0 & \mathbf{y}(1) & 0 & \cdots\end{bmatrix}.$$

Remark. For the rest of this work, we will see these two operators as part of the analysis bank and the synthesis bank respectively. The analysis bank thus consists of the application of the analysis filters followed by the downsampling operator; in the synthesis bank, the upsampling operator is followed by the synthesis filters.

Remark. To construct a wavelet basis in section 1.2, we will iterate the analysis bank of a filter bank several times. However, we currently discard half of the signal entries and thus significantly reduce the energy of the analysis result, especially when applied repeatedly. We will compensate for this loss of energy by scaling the output of the analysis bank by the factor $\sqrt{2}$ and thus doubling the energy.

We will integrate this factor of $\sqrt{2}$ directly into the high-pass and low-pass filters used. Instead of the low-pass filter $\mathbf{H}_0$ and high-pass filter $\mathbf{H}_1$ of the analysis bank, we use the filters

$$\mathbf{C} := \sqrt{2}\mathbf{H}_0 \quad \text{and} \quad \mathbf{D} := \sqrt{2}\mathbf{H}_1.$$

In the context of filter banks, we also refer to these scaled filters as high-pass or low-pass filters and always notate them with $\mathbf{C}$ for the low-pass filter and $\mathbf{D}$ for the high-pass filter. We will write the corresponding synthesis filters with $\tilde{\mathbf{C}}$ for the low-pass channel and $\tilde{\mathbf{D}}$ for the high-pass channel.

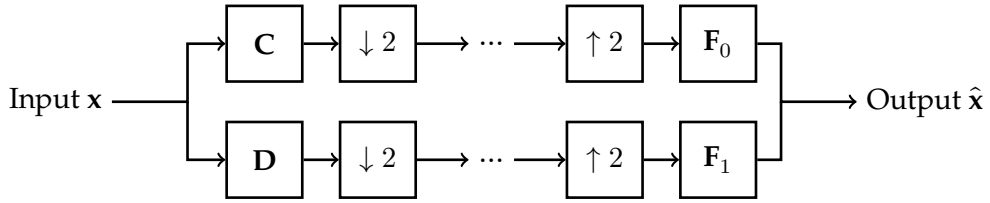


Figure 1.5: Schematic of a filter bank. We can see the analysis bank (left) and the synthesis bank (right) with the normalized low-pass and high-pass filters **C** and **D**. The analysis bank splits the signal **x** into a low-frequency and a high-frequency component, each with half the length; the synthesis bank recombines these components to produce the output $\hat{\mathbf{x}}$. The dots between the two parts indicate further processing such as compression. For the reconstruction, we assume no further processing (can close the gap mentally) and expect $\hat{\mathbf{x}}(n) = \mathbf{x}(n-l)$ for $l \geq 0$.

1.1.6 Matrix representations of filter banks

Let $\mathbf{L} := (\downarrow 2)\mathbf{C}$ and $\mathbf{B} := (\downarrow 2)\mathbf{D}$. Then we can write **L** and **B** as infinite matrices by deleting the odd indexed rows from the Toeplitz matrices of **C** and **D** respectively. If we take both matrices together, we obtain the representation of the analysis bank as an infinite matrix

$$\mathbf{H}_t := \begin{bmatrix}\mathbf{C}\\ \mathbf{D}\end{bmatrix} = \begin{bmatrix}
 & & & \vdots & & & \\
\cdots & \mathbf{c}(0) & \mathbf{c}(-1) & \mathbf{c}(-2) & \mathbf{c}(-3) & \mathbf{c}(-4) & \cdots \\
\cdots & \mathbf{c}(2) & \mathbf{c}(1) & \mathbf{c}(0) & \mathbf{c}(-1) & \mathbf{c}(-2) & \cdots \\
 & & & \vdots & & & \\
\cdots & \mathbf{d}(0) & \mathbf{d}(-1) & \mathbf{d}(-2) & \mathbf{d}(-3) & \mathbf{c}(-4) & \cdots \\
\cdots & \mathbf{d}(2) & \mathbf{d}(1) & \mathbf{d}(0) & \mathbf{d}(-1) & \mathbf{c}(-2) & \cdots \\
 & & & \vdots & & &
\end{bmatrix}.$$

For a signal **x** and $\mathbf{a} := \mathbf{L}\mathbf{x}$, $\mathbf{b} := \mathbf{B}\mathbf{x}$ we have in block notation

$$\mathbf{H}_t\mathbf{x} = \begin{bmatrix}\mathbf{L}\\ \mathbf{B}\end{bmatrix}\mathbf{x} = \begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix}.$$

A further matrix representation of the analysis bank is the block Toeplitz matrix:

Definition 1.20. We then obtain the infinite matrix $\mathbf{H}_b$ by interleaving the rows of **L** and **B**. The even rows of $\mathbf{H}_b$ form the rows of **L**; the odd rows form the rows of **B**. The following applies

$$\begin{cases}(\mathbf{H}_b)_{2i,j} = (\mathbf{L})_{i,j},\\ (\mathbf{H}_b)_{2i+1,j} = (\mathbf{B})_{i,j}.\end{cases}$$

$\mathbf{H}_b$ has thus the form

$$\mathbf{H}_b = \begin{bmatrix} \ddots & & & & & & & \\ \cdots & \mathbf{c}(0) & \mathbf{c}(-1) & \mathbf{c}(-2) & \mathbf{c}(-3) & \mathbf{c}(-4) & \mathbf{c}(-5) & \cdots \\ \cdots & \mathbf{d}(0) & \mathbf{d}(-1) & \mathbf{d}(-2) & \mathbf{d}(-3) & \mathbf{c}(-4) & \mathbf{d}(-5) & \cdots \\ \cdots & \mathbf{c}(2) & \mathbf{c}(1) & \mathbf{c}(0) & \mathbf{c}(-1) & \mathbf{c}(-2) & \mathbf{c}(-3) & \cdots \\ \cdots & \mathbf{d}(2) & \mathbf{d}(1) & \mathbf{d}(0) & \mathbf{d}(-1) & \mathbf{c}(-2) & \mathbf{d}(-3) & \cdots \\ & & & & & & & \ddots \end{bmatrix}.$$

We call $\mathbf{H}_b$ a *block Toeplitz matrix*, since the 2×2 block diagonals of $\mathbf{H}_b$ are constant.

1.1.7 Perfect reconstruction and orthonormal filter banks

Definition 1.21. Let $\mathbf{x}$ be a signal and $\mathbf{a} = \mathbf{L}\mathbf{x}$ and $\mathbf{b} = \mathbf{B}\mathbf{x}$ the outputs of the analysis bank. Applying the synthesis bank then yields

$$\hat{\mathbf{x}} := \tilde{\mathbf{C}}(\uparrow 2)\mathbf{a} + \tilde{\mathbf{D}}(\uparrow 2)\mathbf{b} = \tilde{\mathbf{C}}(\uparrow 2)(\downarrow 2)\mathbf{C}\mathbf{x} + \tilde{\mathbf{D}}(\uparrow 2)(\downarrow 2)\mathbf{D}\mathbf{x}.$$

We say that a filter bank allows for *perfect reconstruction* if the synthesis bank inverts the analysis bank up to a delay of $l \geq 0$. This means that for all $n \in \mathbb{Z}$ we have

$$\mathbf{x}(n) = \hat{\mathbf{x}}(n - l).$$

Remark. We allow for a delay during inversion so that we can ensure the causality of the synthesis filters used. Let's assume that we have FIR filters as synthesis filters that allow perfect reconstruction without delay. If these are not causal filters, there exists an $l \geq 0$ due to the finite impulse response, so that a delay of l makes the filters causal. However, this also leads to a delay of the reconstructed signal by l.

In the following, we will construct synthesis filters from the analysis filters. The calculations for this are simpler if the constructed synthesis filters are not causal. However, we note that causality can be created by inserting a delay.

Definition 1.22. Let $\mathbf{H}$ be a filter. We call the filter $\mathbf{H}^T$ with impulse response

$$\mathbf{h}^T(k) = \mathbf{h}(-k)$$

the *transposed filter* of $\mathbf{H}$. The Toeplitz matrix of $\mathbf{H}^T$ can be obtained by transposing the Toeplitz matrix $\mathbf{H}$.

Definition 1.23. We call a filter bank *orthonormal* if it allows for perfect reconstruction and

$$\tilde{\mathbf{C}} = \mathbf{C}^T \quad \tilde{\mathbf{D}} = \mathbf{D}^T,$$

i.e. the synthesis filters result from the analysis filters through transposition.

Remark. To construct an orthonormal filter bank, it is therefore sufficient to determine the filters of the analysis bank in such a way that the filter bank enables perfect reconstruction. The synthesis bank then results from the transposed filters.

Theorem 1.24 (Strang and Nguyen [SN96, pp. 147])**.** *Let* $\mathbf{c}$ *be a low-pass filter of even length* N*. If we choose*

$$\mathbf{d}(k) = (-1)^k \mathbf{c}(N-1-k), \tag{1.5}$$

then is the resulting filter bank orthonormal if and only if

$$\sum_{k\in\mathbb{Z}} \mathbf{c}(k)\mathbf{c}(k-2n) = \delta_{n,0}. \tag{1.6}$$

Remark. With the help of this theorem, an orthonormal filter bank even results from the choice of a suitable low-pass filter. The high-pass filter follows from equation (1.5) and the synthesis bank from the transposed analysis filters.

Example 1.25. Let's look at our running example of the moving average and the moving difference. We want to show that both filters together form a filter bank. To do this, we scale the impulse responses by $\sqrt{2}$ and obtain

$$\sqrt{2}\mathbf{h_0} = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right), \quad \sqrt{2}\mathbf{h_1} = \left(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right).$$

We note that the filter pair fulfills the equation (1.5). Furthermore,

$$\sum_{k\in\mathbb{Z}} (\sqrt{2}\mathbf{h}_0(k))^2 = \frac{1}{\sqrt{2}^2} + \frac{1}{\sqrt{2}^2} = 1.$$

In addition, due to their small length, the even displacements of $\mathbf{h}_0$ do not overlap, so that equation (1.6) is fulfilled. The filter bank with scaled moving average and moving difference as analysis filters is therefore orthonormal.

1.2 Wavelets

In the previous section, we used filter banks to divide a discrete signal into a high-frequency part ("details") and a low-frequency part ("base level"). We want to use this concept to approximate signals in $L^2(\mathbb{R})$. Starting from a base level, we gradually add more details at different resolutions, also called scales, to approximate the signal. The functions we use to add the details are our wavelets. If we take the details at all resolutions together, no matter how fine, we get a basis of $L^2(\mathbb{R})$ – the wavelet basis.

We again follow the presentation of Blanke [Bla21] closely which is based on Strang and Nguyen [SN96] and Daubechies [Dau92].

1.2.1 Multiresolution analysis

Definition 1.26 (Multiresolution analysis)**.** A familiy $(V_j)_{j\in\mathbb{Z}}$ of subspaces of $L^2(\mathbb{R})$ is called a *multiresolution analysis* (MRA), if they form a nested chain

$$\{0\} \subset \cdots \subset V_{-1} \subset V_0 \subset V_1 \subset \cdots \subset L^2(\mathbb{R}) \tag{1.7}$$

with the following properties:

(MS1) completeness:

$$\overline{\bigcup_{j\in\mathbb{Z}} V_j} = L^2(\mathbb{R}) \quad \text{and} \quad \bigcap_{j\in\mathbb{Z}} V_j = \{0\}, \tag{1.8}$$

where the closure is formed w.r.t. the $L^2(\mathbb{R})$ norm,

(MS2) shift invariance:

$$f(t) \in V_0 \iff f(t-k) \in V_0 \quad \text{for all } k \in \mathbb{Z}, \tag{1.9}$$

(MS3) scale invariance:

$$f(t) \in V_j \iff f(2t) \in V_{j+1} \quad \text{for all } j \in \mathbb{Z}, \tag{1.10}$$

(MS4) shift invariant basis: There exists a function $\phi \in V_0$ such that

$$\{\phi(t-k) \mid k \in \mathbb{Z}\}$$

forms an orthonormal basis of V_0.

The function ϕ is called a *scaling function*. The orthogonal complement W_j of two subsequent subspaces of the MRA

$$V_{j+1} = V_j \oplus W_j, \tag{1.11}$$

is called a *wavelet subspace*.

MRAs are the central functional analytical concept of wavelet theory. The subspaces V_j correspond to all functions that we have approximated up to a resolution of 2^j. Wavelet subspaces W_j represent the details described at the beginning, which are added at each scale.

Remark. For $f \in L^2(\mathbb{R})$ let $\mathcal{P}_{V_j}(f)$ be the orthogonal projection of f onto V_j. Then item (MS1) implies on the one hand that the MRA is not redundant and on the other hand that

$$\lim_{j\to\infty} \mathcal{P}_{V_j}(f) = f.$$

Lemma 1.27 (Properties of a MRA)**.** *Let $\{V_j\}_{j\in\mathbb{Z}}$ be a MRA. The following holds:*

(*i*) $W_j \perp W_k$ *for all* $k \neq j$,

(*ii*) $V_{j+1} = W_j \oplus \cdots \oplus W_1 \oplus W_0 \oplus V_0$ *for all* $j \geq 0$,

(*iii*) $f(t) \in W_j \iff f(2t) \in W_{j+1}$ *for all* $j \in \mathbb{Z}$,

(*iv*) *for all* $j \in \mathbb{Z}$ *we have* $\{\phi_{jk} \mid k \in \mathbb{Z}\}$ *where* $\phi_{jk}(t) := 2^{j/2}\phi(2^j t - k)$ *form an orthonormal basis of* V_j.

Proof. (i) Without loss of generality let k be the smaller index of k and j. Then $W_k \subset V_j \perp W_j$.

(ii) follows directly from the repeated application of the definition of wavelet subspaces.

(iii) follows from the definition of wavelet subspaces and item (MS3).

(iv) Being a basis follows directly from the properties (MS2) to (MS4). We show the orthonormality of the basis by reducing it via substitution to item (MS4):

$$\|2^{j/2}\phi(2^j t-k)\|^2_{L^2(\mathbb{R})} = \int_{\mathbb{R}} 2^j \phi(2^j t-k)^2 \,\mathrm{d}t = \int_{\mathbb{R}} \phi(t)^2 \,\mathrm{d}t = \|\phi\|^2_{L^2(\mathbb{R})} = 1$$

For $k \neq k'$ we get by substitution

$$\begin{aligned}\langle 2^{j/2}\phi(2^j t-k), 2^{j/2}\phi(2^j t-k')\rangle_{L^2(\mathbb{R})} &= \int_{\mathbb{R}} 2^j \phi(2^j t-k)\phi(2^j t-k') \,\mathrm{d}t \\ &= \int_{\mathbb{R}} \phi(t-k)\phi(t-k') \,\mathrm{d}t \\ &= \langle \phi(t-k), \phi(t-k')\rangle_{L^2(\mathbb{R})} = 0\end{aligned}$$

□

Theorem 1.28 (Dilation equation). *Let $\{V_j\}_{j\in\mathbb{Z}}$ be a MRA. Then there exist coefficients $\mathbf{c} \in l^2$ such that the scaling function ϕ fulfills the dilation equation:*

$$\phi(t) = \sqrt{2}\sum_{k\in\mathbb{Z}} \mathbf{c}(k)\phi(2t-k) \qquad \text{with } \mathbf{c}(k) \in \mathbb{R}. \qquad \text{(dilation equation)}$$

For the coefficients $\mathbf{c}(k)$ we have

$$\sum_{k\in\mathbb{Z}} \mathbf{c}(k)\mathbf{c}(k-2m) = \delta_{m,0}. \tag{1.12}$$

Proof according to Strang and Nguyen [SN96]. It holds $\phi \in V_0 \subset V_1$. Thus, we can write ϕ using the orthonormal basis $\{\sqrt{2}\phi(2t-k)\}$ of V_1 and get the dilation equation. We obtain equation (1.12) by multiplying the dilation equations for $\phi(t)$ and $\phi(t-m)$, then integrating and using the orthonormality of the basis:

$$\begin{aligned}\delta_{m,0} = \int_{\mathbb{R}} \phi(t)\phi(t-m) \,\mathrm{d}t &= \int_{\mathbb{R}} 2\Big(\sum_{k\in\mathbb{Z}} \mathbf{c}(k)\phi(2t-k)\Big)\Big(\sum_{k\in\mathbb{Z}} \mathbf{c}'(k)\phi(2(t-m)-k)\Big) \,\mathrm{d}t \\ &= \int_{\mathbb{R}} 2\Big(\sum_{k\in\mathbb{Z}} \mathbf{c}(k)\phi(2t-k)\Big)\Big(\sum_{k\in\mathbb{Z}} \mathbf{c}(k-2m)\phi(2t-k)\Big) \,\mathrm{d}t \\ &= \sum_{k\in\mathbb{Z}} \mathbf{c}(k)\mathbf{c}(k-2m)\end{aligned}$$

□

Theorem 1.29 (Wavelet basis, [Dau92, Theorem 5.1.1]). *Let $\{V_j\}_{j\in\mathbb{Z}}$ be a MRA. There exists $w \in W_0$ such that $\{w(t-k) \mid k \in \mathbb{Z}\}$ forms an orthonormal basis of W_0 and*

$$\{\phi(t-k) \mid k \in \mathbb{Z}\} \quad \cup \quad \bigcup_{j\geq 0}\{w_{jk}(t) \mid k \in \mathbb{Z}\} \qquad \text{with} \qquad w_{jk}(t) := 2^{j/2}w(2^j t-k) \tag{1.13}$$

an orthonormal basis of $L^2(\mathbb{R})$. The function w has the form

$$w(t) = \sqrt{2} \sum_{k \in \mathbb{Z}} \mathbf{d}(k) \phi(2t - k). \qquad \text{(wavelet equation)}$$

A possible construction of w results if we choose **d** *as in equation* (1.5)*:*

$$\mathbf{d}(k) := (-1)^k \mathbf{c}(N - 1 - k) \qquad \textit{with } N \textit{ even} \tag{1.14}$$

where **c** *are the coefficients of the dilation equation.*

We call the function w the "mother wavelet" as all wavelet subspaces are spanned from scaled translations of this function.

Remark. The dilation equation and its counterpart, the wavelet equation, are central to wavelet theory, as they allow us to write scaling functions or wavelets of one scale in the basis of the next-higher scale – with basis coefficients that do not depend on the scale used. This relationship between the different scales is fundamental to wavelet theory and allows us, for example, to construct fast methods for switching bases between scales in the following sections.

Lemma 1.30. *Let $\{V_j\}_{j \in \mathbb{Z}}$ be a MRA with a scaling function ϕ whose Fourier transform is bounded and continuous and non-zero in $\omega = 0$. Further, let* **c** *be the coefficients from the dilation equation and* **d** *the coefficients from the wavelet equation after the construction* (1.14)*. Then* **c** *is a low-pass filter that forms an orthonormal filter bank together with* **d**.

Proof. One can show that under these conditions on the scaling function, **c** is a low-pass filter scaled by $\sqrt{2}$ [VK95, Problem 4.3], i.e.

$$|C(0)| = \sqrt{2} \quad \text{and} \quad |C(\pm\pi)| = 0.$$

As we choose **d** as in (1.5), the orthonormality follows from (1.12) because of theorem 1.24. □

Remark. The conditions posed on the scaling functions are in practice always fulfilled [VK95, S. 226].

We have shown how we can construct an orthonormal basis of $L^2(\mathbb{R})$ based on a MRA. The central element here is the choice of a suitable scaling function, from which everything else follows.

However, it is not easy to find a suitable scaling function. For example, the scaling functions of two important classes of wavelets, which we introduce in section 1.2.2, cannot be represented in closed-form [Dau92].

A more promising approach is to choose a suitable low-pass filter **c**, which we insert as a coefficient in the dilation equation. This gives us a fixed-point equation that we can solve for the scaling function—provided the low-pass filter fulfills certain properties that ensure solvability.

We calculate this approach using a pair of low-pass and high-pass filters as an example:

Example 1.31 (Haar wavelets)**.** We choose the moving average $\mathbf{h}_0 = (1/2, 1/2)$ as a low-pass filter and want to construct the scaling function ϕ from it. First, we scale $\mathbf{h}_0$ by $\sqrt{2}$, receiving the low-pass filter of the filter bank

$$\mathbf{c} = \sqrt{2}\mathbf{h}_0 = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right).$$

From equation (1.14) then results with $N = 2$ the high-pass filter

$$\mathbf{d}(k) := (-1)^k \mathbf{c}(1-k) = \left(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right) = \sqrt{2}\left(\frac{1}{2}, -\frac{1}{2}\right),$$

which we recognize as the scaled moving differences. We apply the dilation equation to $\mathbf{c}$ and get the fixed point equation

$$\phi(t) = \phi(2t) + \phi(2t-1).$$

The solution to this simple equation can be just read off as the "rectangular" function

$$\phi(t) = \phi_0 := \mathbb{1}_{[0,1)}(t) = \begin{cases} 1 & 0 \le t < 1, \\ 0 & \text{else.} \end{cases}$$

We calculate that this is indeed a solution to dilation equation:

$$\phi(2t) + \phi(2t-1) = \mathbb{1}_{[0,\frac{1}{2})}(t) + \mathbb{1}_{[\frac{1}{2},1)}(t) = \mathbb{1}_{[0,1)} = \phi(t).$$

Using the wavelet equation we can now construct the mother wavelet corresponding to ϕ:

$$w(t) = \sqrt{2}\sum_{k\in\mathbb{Z}} \mathbf{d}(k)\phi(2t-k) = \phi(2t) - \phi(2t-1) = \begin{cases} 1 & 0 \le t < \frac{1}{2}, \\ -1 & \frac{1}{2} \le t < 1, \\ 0 & \text{else.} \end{cases}$$

The wavelets derived from ϕ and w are called *Haar wavelets* according to Haar [Haa10]. In figure 1.6 ϕ and w are depicted.

The Haar wavelets are the simplest of all wavelets due to the short length of the underlying filters. Nevertheless, they are of great practical relevance. One reason for this is that they do not require any boundary treatment due to their short length on finite signals (see section 1.3).

For Haar wavelets, we were able to directly determine the scaling function ϕ that solves the dilation equation. In general, we can only determine ϕ with a fixed point iteration, applying the dilation equation repeatedly. Details on this construction can be found in Strang and Nguyen [SN96, Section 7.2].

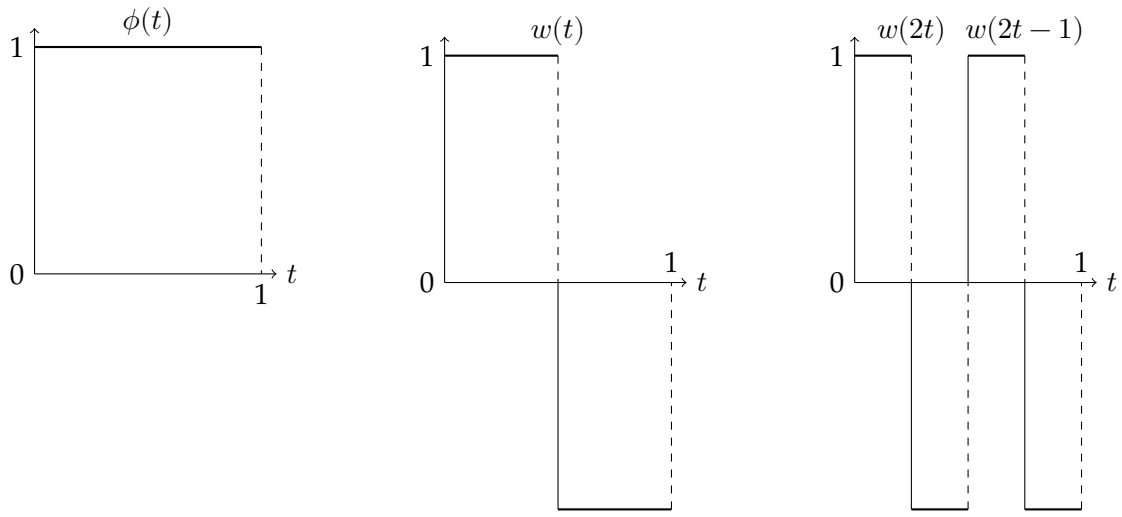


Figure 1.6: Scaling function (left), mother wavelet (center) and wavelets of the first scale (right) of the Haar wavelets

.

1.2.2 Daubechies-Wavelets and Symlets

So far, we have only come across a single wavelet in the form of the simplest of all wavelets—the Haar wavelet. In the following, we will outline the construction of two related wavelet families: the *Daubechies wavelets* and the *symlets*. The Daubechies wavelets are named after Daubechies [Dau88], who first described them. The symlets are a more symmetrical variation of the Daubechies wavelets. We denote a Daubechies wavelets and symlets of length $2p$ as *dbp* respectively *symp*. For the derivation we follow Strang and Nguyen [SN96] and Daubechies [Dau92].

As we have seen in the previous section, constructing a wavelet amounts to the choice of a suitable low-pass filter as the high-pass filter can be obtained by applying equation (1.5). As the ideal low-pass filter is not obtainable due to not being a FIR filter, we aim to use a good approximation while only using filters with a finite impulse response.

As shown in figure 1.3 the absolute frequency response of an ideal low-pass filter is a step function, which is flat everywhere. In particular, being flat at the extreme ends of the frequency bands yields a good seperation of the very low and very high frequencies. This idea is core to the low-pass filters used for both wavelet families: so-called *maxflat filters*. These are causal FIR filters with an absolute frequency response that is maximally flat at 0 and $\pm\pi$ for a given filter length.

We now want to construct the filter coefficients of such a maxflat low-pass filter $\mathbf{C}$ of length $2p$. As we want to obtain an orthonormal filter bank, the filter coefficients $\mathbf{C}$ need to fulfill the p conditions of equation (1.6). We use the remaining p degrees of freedom to choose $C(\omega)$ at $\omega = 0$ and $\omega = \pi$ maximally flat. This corresponds to the p conditions

$$C(\pi) = C'(\pi) = \cdots = C^{(p-1)}(\pi) = 0,$$

so π being a p-fold zero of C. For symmetry reasons this also gives the same derivatives for

$\omega = 0$. Thus, we can write $C(\omega)$ as

$$C(\omega) = \sum_{n=0}^{2p-1} \mathbf{c}(n)\mathrm{e}^{-\mathrm{i}n\omega} = \left(\frac{1+\mathrm{e}^{-\mathrm{i}\omega}}{2}\right)^p R(\mathrm{e}^{-\mathrm{i}\omega}),$$

where R is a polynomial of degree $p-1$. One can determine the coefficients of R by spectral factorization

$$|R(z)|^2 = R(z)\overline{R(z)}.$$

of the trigonometric polynomial $|R(z)|^2$ [cf. Dau92, pp. 171]. However, this factorization is not unique. Choosing it such that all roots of R lie in or on the unit circle yields the Daubechies wavelets. It is remarkable that for the special case of $p = 1$, the Daubechies wavelets coincide with the Haar wavelets. However, the Daubechies filters with large lengths are very asymmetrical.

For $p > 1$, it is impossible to choose the roots of the polynomial in a way that results in a symmetric filter [Dau92]. The choice of roots that gets closest to a symmetric filter results in the family of symlets. In figure 2 the scaling functions and mother wavelets are depicted for Daubechies wavelets and symlets of length $2p = 8, 16, 24$.

So far, we did not address the question why the choice of a wavelet family matters. The following lemma shows that using longer Daubechies wavelets or longer symlets improves the approximation accuracy. We thus refer to longer wavelets of these families as being of "higher order".

Lemma 1.32 (approximation accuracy [SN96])**.** *For Daubechies wavelets and symlets of length $2p$ we have:*

(*i*) *all polynomials of degree $p-1$ or smaller are linear combination of $\{\phi(t-k) \mid k \in \mathbb{Z}\}$,*

(*ii*) *the mother wavelet w has p vanishing moments, i.e.*

$$\langle w, t^k \rangle = 0 \qquad \text{for } k \in \{0, \dots, p-1\}.$$

1.2.3 Fast wavelet transform

We have seen in item (ii) of lemma 1.27 that we can decompose the subspaces V_j of a MRA into a direct sum of V_0 and wavelet subspaces. Similarly, we have constructed two orthonormal bases of V_j:

$$\{\phi_{Jk} \mid k \in \mathbb{Z}\} \qquad \text{and} \qquad \{\phi_{0k} \mid k \in \mathbb{Z}\} \cup \bigcup_{i=0}^{J} \{w_{jk} \mid k \in \mathbb{Z}\}$$

A change of basis from the scaling function basis into the wavelet basis is called discrete wavelet transform (DWT). In the following, we will introduce an efficient procedure to calculate the DWT: the fast wavelet transform (FWT). It was first described by Mallat [Mal89]. For the derivation, we follow Strang and Nguyen [SN96]. We use the notation $y_{j,-}$ for a sequence $\{y_{j,k}\}_{k\in\mathbb{Z}}$.

We first consider the step from V_1 to $V_0 \oplus W_0$. The further steps then follow inductively due to the properties of MRAs. For $f_1(t) \in V_1$ let $a_{1,-}$ be the coefficients of the basic representation

$$f_1(t) = \sum_{k\in\mathbb{Z}} a_{1k}\phi_{1k}(t) = \sqrt{2}\sum_{k\in\mathbb{Z}} a_{1k}\phi(2t-k).$$

We aim to construct a change in basis. We therefore look for $a_{0,-}$ and $b_{0,-}$ such that

$$f_1(t) = \sum_{k\in\mathbb{Z}} a_{1k}\phi_{1k}(t) = \sum_{k\in\mathbb{Z}} a_{0,k}\phi_{0,k}(t) + \sum_{k\in\mathbb{Z}} b_{0,k}w_{0,k}(t) = \sum_{k\in\mathbb{Z}} a_{0,k}\phi(t-k) + \sum_{k\in\mathbb{Z}} b_{0,k}w(t-k).$$

For the calculation of the coefficients we consider the dilation equation for $\phi(t-n)$ and the wavelet equation for $w(t-n)$ with $n \in \mathbb{Z}$. We substitute $k = l - 2n$ and get

$$\begin{aligned}
\phi_{0,n}(t) &= \sqrt{2}\sum_{l\in\mathbb{Z}} \mathbf{c}(l-2n)\phi(2(t-n)-(l-2n)) = \sqrt{2}\sum_{l\in\mathbb{Z}} \mathbf{c}(l-2n)\phi_{1,l}(t);\\
w_{0,n}(t) &= \sqrt{2}\sum_{l\in\mathbb{Z}} \mathbf{d}(l-2n)\phi(2(t-n)-(l-2n)) = \sqrt{2}\sum_{l\in\mathbb{Z}} \mathbf{d}(l-2n)\phi_{1,l}(t).
\end{aligned}$$

Due to the orthonormality of the basis, calculating the scalar product of $\phi_{0,n}$ respectively $w_{0,n}$ with f_1 yields an expression for the coefficients a_{0n} respectively b_{0n}:

$$\begin{aligned}
a_{0n} &= \langle \phi_{0n}, f_1\rangle_{L^2(\mathbb{R})} = \sum_{l\in\mathbb{Z}} \mathbf{c}(l-2n)a_{1l},\\
b_{0n} &= \langle w_{0n}, f_1\rangle_{L^2(\mathbb{R})} = \sum_{l\in\mathbb{Z}} \mathbf{d}(l-2n)a_{1l}
\end{aligned} \qquad \text{with } n \in \mathbb{Z}.$$

We compare this to lemma 1.18 and note the coefficients $a_{0,-}$ and $b_{0,-}$ stemming directly from a filter bank applied to $a_{1,-}$. However, it is important to note that the sign of the filter indices is opposite to lemma 1.18. Therefore the analysis bank of the filter bank consists of the transposed filters $\mathbf{C}^T$ and $\mathbf{D}^T$ with impulse responses

$$\mathbf{c}^T(k) = \mathbf{c}(-k) \qquad \text{and} \qquad \mathbf{d}^T(k) = \mathbf{d}(-k).$$

We can analogously apply the considerations from above for all $j \in \mathbb{Z}$ in order to calculate the coefficients $a_{j-1,-}$ and $b_{j-1,-}$ starting from the basis representation in V_j. If we then apply the method to $a_{j-1,-}$, we obtain a representation of $a_{j,-}$ in the basis of $V_{j-2} \oplus W_{j-2} \oplus W_{j-1}$. We can now repeat this to successively calculate the coefficients of further wavelet subspaces. In total, this results in the fast wavelet transform. In figure 1.7 you can find a schematic representation of the FWT.

Remark. (i) In practice, the FWT is stopped after a fixed number J of recusion steps. In this case, we call the FWT to have *J levels* or *scales*. For a three level FWT, we start with $a_{3,-}$, calculate the coefficients $b_{2,-}$, $b_{1,-}$, $b_{0,-}$ and stop at $a_{0,-}$.

(ii) As long as the filters in use are of finite length, at each recursion step we generate from the coefficient vector $a_{j,-}$ of finite length two vectors $a_{j-1,-}$ and $b_{j-1,-}$ with half the length each.

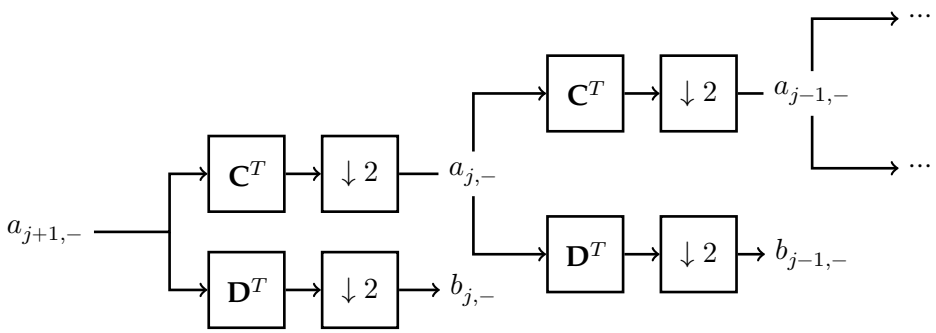


Figure 1.7: Scheme of the fast wavelet transform.

(iii) Reverting the construction of the FWT, i.e. calculating the coefficients $a_{j+1,-}$ based on $a_{j,-}$ and $b_{j,-}$ yields the inverse fast wavelet transform (iFWT). This corresponds to repeatedly applying the synthesis bank of the filter bank used in the FWT. The resulting scheme is shown in figure 1.

(iv) The starting point of the FWT are the basis coefficients $a_{j,k}$. Given a signal $f \in L^2(\mathbb{R})$ or a discretization thereof, to apply the FWT we have to first calculate the base coefficients of f in the wavelet basis by orthognally projecting f to V_j. This can be done by calculating the scalar product of f and the basis elements $\phi_{j,k}$. If we are given a discretized signal an approximation of the basis coefficients is required; different approaches are discussed e.g. by Strang and Nguyen [SN96, S. 232f.].

However, this subtlety is often overlooked in practice, e.g. by popular implementations of the FWT [Lee+19; WBGH24]. As ignoring this subtlety is the norm in application we follow accordingly to ensure comparability.

Theorem 1.33 (Runtime complexity of the FWT)**.** *For a filter length N and a signal length $L = 2^J$, the FWT for J scales has a runtime complexity of $\mathcal{O}(NL)$.*

Proof. Applying a filter to coefficients on level j has a cost of $NL2^{j-J}$ as the coefficients on level j have a length of $2^{j-J}L$. Summing these costs over all levels yields

$$NL + \frac{1}{2}NL + \frac{1}{4}NL + \cdots + 2^{-(J-1)}NL < 2NL = \mathcal{O}(NL). \qquad \square$$

Remark. For a constant filter length N the FWT has a linear runtime complexity and is thus asymptotically faster than the fast Fourier transform (FFT), which has an asymptotic runtime of $\mathcal{O}(L \log L)$ [SN96].

1.2.4 Wavelet packet transform

We look at the schematic representation of fast wavelet transform in figure 1.7. If we take the calculated base coefficients as nodes, the FWT generates a directed binary tree. The root is formed by the input coefficients $a_{J,-}$. The depth of the nodes in the tree corresponds to the scale of the coefficients. To obtain the nodes of the next scale, the filter bank is applied

only to the scaling function coefficients $a_{j,-}$; the wavelet coefficients $b_{j,-}$ are leaves of the binary tree. Thus, the resulting binary tree consists of a linear number of nodes.

We now want to introduce another wavelet transformation with the *wavelet packet transformation*. To do this, we complete the binary tree of FWT by also applying the filter bank to the wavelet coefficients $b_{j,-}$. Then the coefficients of the j-th level of the complete tree are the result of the j-scale wavelet packet transform. If the input coefficients have a length of L, the j-scale wavelet packet transform generates 2^j coefficients of length $L/2^J$. It can be shown that this corresponds to a change of basis to the so-called *Walsh basis* [SN96, pp. 72]. The basis functions result from the scaling function ϕ by a recursive application of dilation equation and wavelet equation corresponding to the path in the complete tree.

1.2.5 Wavelets on 2d signals

So far we constructed a wavelet basis over $\mathbb{R}$. However, we are in particular interested in a wavelet basis of $L^2(\mathbb{R}^2)$ as we are concerned with the generation of images. Therefore, we introduce a simple extension of wavelet theory to two dimensions by applying a 1d wavelet transform seperately to both axes. For this reason, the introduced extension is called a *separable* 2d wavelet transformation. For the construction we are guided by Jensen and la Cour-Harbo [Jl01, section 6.1]. Note that other constructions are possible, in particular in a non-seperable fashion that genuinely uses the two-dimensionality of $\mathbb{R}^2$ [SN96].

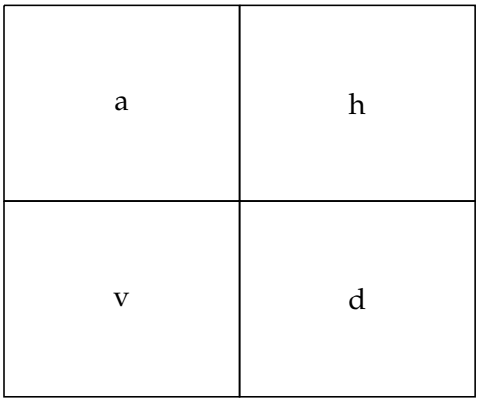
Let $\mathbf{X} \in \mathbb{R}^{N \times N}$ be a two-dimensional quadratic signal of finite length, with N rows and columns. First, we apply a one-dimensional wavelet transform along the columns of $\mathbf{X}$. Let $\mathbf{W}$ be the $N \times N$ matrix representing the one-timensional wavelet transform, that can be constructed e.g. using Gram-Schmidt boundary filters (cf. section 1.3.2). For each column of $\mathbf{X}$ the wavelet transformation is obtained by multiplication with $\mathbf{W}_c$. Thus, the matrix product

$$\mathbf{Y}_c := \mathbf{W}\mathbf{X}$$

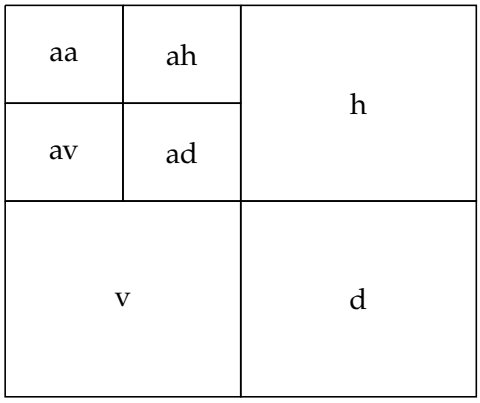
yields a matrix with all columns being the wavelet transformations of columns of $\mathbf{X}$. Repeating this procedure along the rows of $\mathbf{Y}_c$ can be done by multiplying the wavelet transform matrix $\mathbf{W}$ to $\mathbf{Y}_c^T$. Together, we get the result of the seperable two-dimensional wavelet transform

$$\mathbf{Y} := (\mathbf{W}\mathbf{Y}_c^T)^T = \mathbf{W}\mathbf{X}\mathbf{W}^T.$$

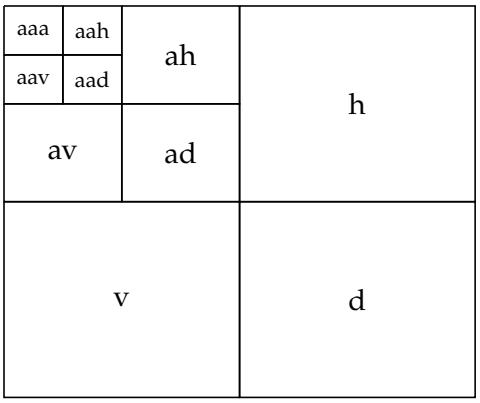
With 2d FWT, the signal is divided into a high-frequency and a low-frequency part along each axis, resulting in a total of four parts. The low-frequency part in both axes is called the approximation coefficient and is noted as “a”. The other parts correspond to directions of edges in the image, which are highlighted in this part: if the part of one axis is high-frequency and the other is low-frequency, horizontal (“h”) or vertical (“v”) edges in the image are highlighted. If the parts of both axes are high-frequency, this corresponds to diagonal edges (“d”). This is shown schematically in figure 1.8a. We obtain the second scale of the 2D-FWT by repeating this on the approximation coefficient of the previous scale (see figures 1.8b and 1.8c). We note the coefficients obtained in this way by appending the letter of the current scale (“a”, “d” etc.) to the letters of the previous scale. If we



(a) First scale of a 2d FWT



(b) Second scale of a 2d FWT



(c) Third scale of a 2d FWT

aaa	aah	aha	ahh	haa	hah	hha	hhh
aav	aad	ahv	ahd	hav	had	hhv	hhd
ava	avh	ada	adh	hva	hvh	hda	hdh
avv	avd	adv	add	hvv	hvd	hdv	hdd
vaa	vah	vha	vhh	daa	dah	dha	dhh
vav	vad	vhv	vhd	dav	dad	dhv	dhd
vva	vvh	vda	vdh	dva	dvh	dda	ddh
vvv	vvd	vdv	vdd	dvv	dvd	ddv	ddd

(d) 2d wavelet packet transformation using three scales in frequency order.

Figure 1.8: Schemes of two-dimensional wavelet transforms.

repeat this decomposition on all coefficients of the previous scale, this corresponds to a 2D wavelet-packet transformation (cf. figure 1.8d).

1.3 Handling signal boundaries for wavelets

In numerical applications, especially those related to machine learning, the signals often have finite lengths. Images, for example, a common use case in signal processing, are two-dimensional discrete signals with finite width and height. The theory introduced in section 1.2 is formulated on unbounded signals in $L^2(\mathbb{R})$. Simply moving to finite signals leads to problems: the application of a causal FIR filter—the core of wavelet transforms—to a signal $\sum_{k=0}^{N-1} \mathbf{h}(k)\mathbf{x}(n-k)$ is not possible at the signal edge, as undefined signal entries would have to be accessed.

Methods for realizing the filter application at the signal edges are called *boundary handling*. We will learn about two approaches in this section: signal continuation (section 1.3.1) and boundary filters (section 1.3.2).

1.3.1 Boundary handling by signal extension

An obvious approach for the boundary treatment is to continue the finite signal to an infinite signal and apply the developed methods to this continued signal. To do this, we look at how a filter bank can be applied to the finite signal, as the other use cases such as DWT can be derived from this. We will introduce this with an example that we will return to in later sections.

Example 1.34. Suppose we want to apply the FWT with FIR filters of length N to a finite signal $\mathbf{x}$ of length L. Let $\mathbf{H}_b$ be the block Toeplitz matrix of the filter bank of the transformation. Then we would actually have to form the matrix-vector product $\mathbf{H}_b\mathbf{x}$. However, this is impossible as $\mathbf{H}_b$ is an infinite matrix.

To remedy this, we continue $\mathbf{x}$ to an infinite signal in $l^2(\mathbb{Z})$ by adding zeros to the edges of the signal. This gives us the infinite signal $\tilde{\mathbf{x}}$ with

$$\tilde{\mathbf{x}}(k) = \begin{cases} 0 & k < 0 \\ \mathbf{x}(k) & 0 \leq k < L \\ 0 & k \geq L \end{cases}.$$

This conitinuation is equivalent to the removal of all columns of $\mathbf{H}_b$ with a negative index or an index larger than $L-1$. This gives us a matrix with a suitable finite number of columns and we can form the matrix-vector product. However, the adjusted matrix still contains an infinite number of rows, so the filter result is infinitely long.

We remember that $\mathbf{H}_b$ is a block Toeplitz matrix and the filter bank consists of FIR filters. Therefore, only finitely many (block) diagonals of $\mathbf{H}_b$ are non-zero and all rows of the cropped matrix below or above a certain index consist only of zeros. If we remove these "zero rows", a finite $K \times L$ matrix $\tilde{\mathbf{H}}_b$ remains, which we can multiply by the signal. For a filter length $N = 6$ and a signal length $L = 8$ we thus have

$$\tilde{\mathbf{H}}_b\mathbf{x} = \begin{bmatrix} \mathbf{c}(4) & \mathbf{c}(5) & & & & & & \\ \mathbf{d}(4) & \mathbf{d}(5) & & & & & & \\ \mathbf{c}(2) & \mathbf{c}(3) & \mathbf{c}(4) & \mathbf{c}(5) & & & & \\ \mathbf{d}(2) & \mathbf{d}(3) & \mathbf{d}(4) & \mathbf{d}(5) & & & & \\ \mathbf{c}(0) & \mathbf{c}(1) & \mathbf{c}(2) & \mathbf{c}(3) & \mathbf{c}(4) & \mathbf{c}(5) & & \\ \mathbf{d}(0) & \mathbf{d}(1) & \mathbf{d}(2) & \mathbf{d}(3) & \mathbf{d}(4) & \mathbf{d}(5) & & \\ & & \mathbf{c}(0) & \mathbf{c}(1) & \mathbf{c}(2) & \mathbf{c}(3) & \mathbf{c}(4) & \mathbf{c}(5) \\ & & \mathbf{d}(0) & \mathbf{d}(1) & \mathbf{d}(2) & \mathbf{d}(3) & \mathbf{d}(4) & \mathbf{d}(5) \\ & & & & \mathbf{c}(0) & \mathbf{c}(1) & \mathbf{c}(2) & \mathbf{c}(3) \\ & & & & \mathbf{d}(0) & \mathbf{d}(1) & \mathbf{d}(2) & \mathbf{d}(3) \\ & & & & & & \mathbf{c}(0) & \mathbf{c}(1) \\ & & & & & & \mathbf{d}(0) & \mathbf{d}(1) \end{bmatrix} \begin{bmatrix} \mathbf{x}(0) \\ \mathbf{x}(1) \\ \mathbf{x}(2) \\ \mathbf{x}(3) \\ \mathbf{x}(4) \\ \mathbf{x}(5) \\ \mathbf{x}(6) \\ \mathbf{x}(7) \end{bmatrix} = \begin{bmatrix} \mathbf{y}(0) \\ \mathbf{y}(1) \\ \mathbf{y}(2) \\ \mathbf{y}(3) \\ \mathbf{y}(4) \\ \mathbf{y}(5) \\ \mathbf{y}(6) \\ \mathbf{y}(7) \\ \mathbf{y}(8) \\ \mathbf{y}(9) \\ \mathbf{y}(10) \\ \mathbf{y}(11) \end{bmatrix}. \quad (1.15)$$

In this case, the resulting matrix has $K = 12$ rows, which means that the result of the filter application is longer than the output signal.

The boundary handling method in this example works with a continuation of the signal with zeros. Such a continuation with a constant value is also referred to as *padding*. There

are also other methods of signal continuation [cf. KV89]. In figure 1.9 four frequently used methods of signal continuation are sketched. All of these methods are very similar to the initial example. Only the matrix $\tilde{\mathbf{H}}_b$ is continued also to the left and right so that none of the FIR filters are "cut through" in the rows, as happened in the example.

With these continuations of the signal, discontinuities occur at the ends of the signal in the continuation or in its derivative. These discontinuities lead to large coefficients in the FWT and thus to boundary artifacts [Jl01, p. 144]. Furthermore, we have already encountered another problem for some of these extension techniques. With the construction above[3] applying a single FWT step to a signal will lead to a longer output signal for all orthonormal wavelets besides the Haar wavelet, where the elongation is proportional to the filter length [Jl01]. This is caused by the rows for which only a part of the filter overlaps with the signal still being relevant for the result, leading to a corresponding elongation on both sides of the signal. Due to its short length, this does not apply to the Haar wavelet. It is important to note that the fully extended output signal is necessary for perfect reconstructability and cannot simply be shortened.

1.3.2 Boundary handling with boundary filters

All approaches to boundary handling discussed so far have amounted to continuing the finite signal to an infinite signal in various ways. The boundary handling that we introduce below is fundamentally different. It is not the signal that is adapted to the filters, but the filters to the signal.

We do this by neglecting a property that all filters have implicitly had up to now: *time invariance*. The matrix representation of the filters were Toeplitz matrices, i.e. they had constant diagonal entries. They therefore filtered all parts of the signal equally. We now want to adjust the filters at the edges of the signal in order to avoid boundary effects despite the finite signal length. The adapted filters are no longer time-invariant; their matrix representation will contain some modified rows.

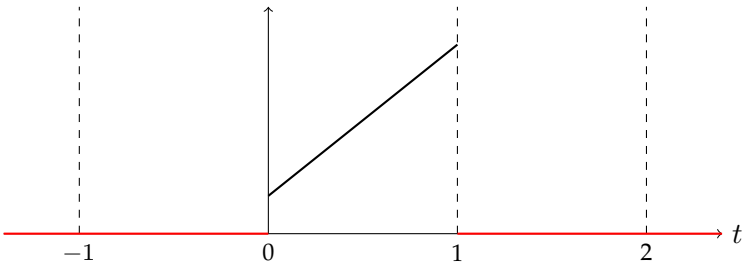
The construction of the so-called Gram-Schmidt boundary filters discussed here follows Blanke [Bla21] closely. Its contents is based on Herley and Vetterli [HV94], Jensen and la Cour-Harbo [Jl01, section 10.3], and Strang and Nguyen [SN96, section 8.5].

The boundary filters also correspond to a wavelet basis. However, it is is not a basis of $L^2(\mathbb{R})$ but of $L^2(I)$ for an interval I. For details on this we also refer to Herley and Vetterli [HV94].

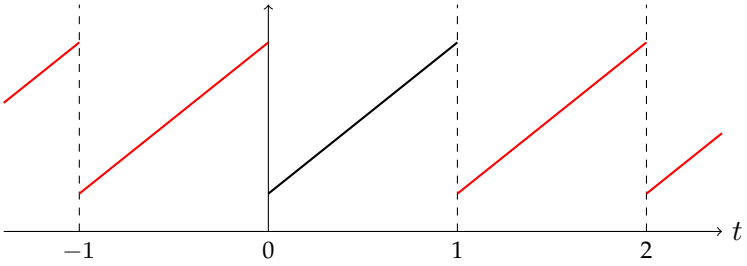
As with signal continuation, our aim is to construct a finite matrix from the infinite Block-Toeplitz matrix $\mathbf{H}_b$ with which we can multiply a finite signal. However, we want to avoid the problem of a signal length elongation and at the same time maintain the orthogonality of the filter bank. We therefore want to construct a square, orthonormal $L \times L$ matrix $\mathbf{H}_L$ for a signal of length L. For the construction, we require L and the filter length N to be even and $L \gg N$.

The reason for the signal length elongation for extension techniques are the "incomplete" rows at the top and bottom of the matrix. Therefore, our starting point is the L-column

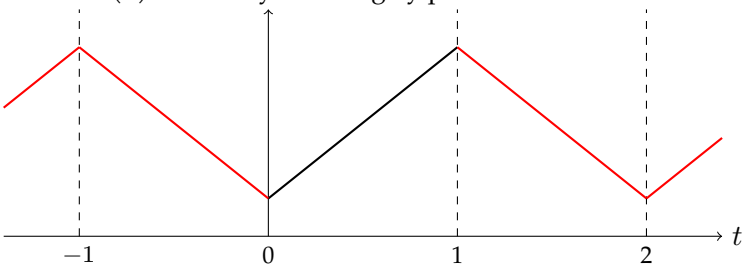
[3] Alternative constructions are available for some variants like the periodic extension, mitigating this problem [Jl01, section 10.4].



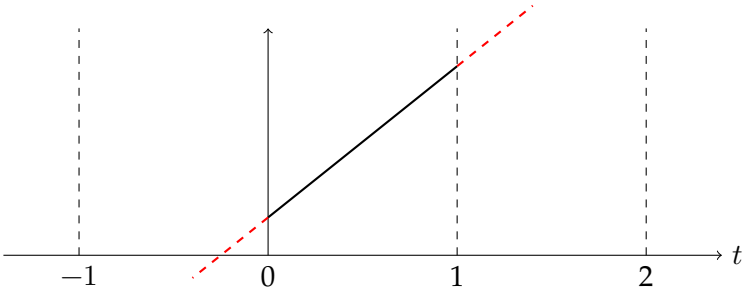
(a) Boundary handling by padding with zeros.



(b) Boundary handling by periodic extension.



(c) Boundary handling by symmetric extension.



(d) Boundary handling by extrapolation.

Figure 1.9: Four types of boundary handling of an example signal with bounded support. The example signal is black, its extension red.

matrix

$$\mathbf{H}_{\text{in}} := \begin{bmatrix} \mathbf{c}(0) & \mathbf{c}(1) & \dots & \dots & \mathbf{c}(N-2) & \mathbf{c}(N-1) & & & \\ \mathbf{d}(0) & \mathbf{d}(1) & \dots & \dots & \mathbf{d}(N-2) & \mathbf{d}(N-1) & & & \\ & \ddots & \ddots & & & \ddots & \ddots & & \\ & & \ddots & \ddots & & & \ddots & \ddots & \\ & & & \mathbf{c}(0) & \mathbf{c}(1) & \dots & \dots & \mathbf{c}(N-2) & \mathbf{c}(N-1) \\ & & & \mathbf{d}(0) & \mathbf{d}(1) & \dots & \dots & \mathbf{d}(N-2) & \mathbf{d}(N-1) \end{bmatrix}, \tag{1.16}$$

which is obtained by extracting the L columns of $\mathbf{H}_b$ with complete rows. By selecting only complete rows $\mathbf{H}_{\text{in}}$ is not a quadratic matrix.[4] One can easily show that $\mathbf{H}_{\text{in}}$ consists of $(L - N + 2)$ rows.

As the matrix $\mathbf{H}_{\text{in}}$ is obtained from an orthonormal filter bank we have

$$(\mathbf{H}_{\text{in}}) \cdot (\mathbf{H}_{\text{in}})^T = \mathbf{I}.$$

Thus, the rows of $\mathbf{H}_{\text{in}}$ form an orthonormal system. We now want to construct $\mathbf{H}_L$ from $\mathbf{H}_{\text{in}}$ by adding further rows at the top and bottom without breaking the orthonormality. This is equivalent to completing the orthonormal system to an orthonormal basis of $\mathbb{R}^L$ which can easily be done using the Gram-Schmidt procedure on any basis completion. We call the added rows *boundary filters*.

Lemma 1.35 (Herley and Vetterli [HV94])**.** *Let* $\mathbf{H}_{\text{in}}$ *as in* (1.16) *and* $\mathbf{x} \in \mathbb{R}^L$ *with* $\mathbf{x}$ *orthogonal to all rows of* $\mathbf{H}_{\text{in}}$*. Then, all entries of* $\mathbf{x}$ *besides the outer* $N - 2$ *entries are zero where* N *denotes the filter length.*

Thus, applying the Gram-Schmidt procedure leads to boundary filters with non-zero entries only on the signal boundary. However, we would prefer the boundary filters added on top to only have entries on the left boundary and analogously for the filters added at the bottom. To achieve this we start with a basis completion with non-zero entries on only one of the signal ends. We choose our candidates from the rows of $\tilde{\mathbf{H}}_b$ as defined in equation (1.15). The inner rows of $\tilde{\mathbf{H}}_b$ form $\mathbf{H}_{\text{in}}$. Adding the next $(N-2)/2$ rows on each side yields the quadratic matrix

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{\text{li}} \\ \mathbf{H}_{\text{in}} \\ \mathbf{H}_{\text{re}} \end{bmatrix} \in \mathbb{R}^{L \times L},$$

where

$$\mathbf{H}_{\text{li}}, \mathbf{H}_{\text{re}} \in \mathbb{R}^{\frac{N-2}{2} \times L}$$

are the rows added on top respectively on the bottom. By their construction only the first or last $N - 2$ entries of $\mathbf{H}_{\text{li}}$ and $\mathbf{H}_{\text{re}}$ can be non-zero, allowing us to write

$$\mathbf{H}_{\text{li}} = \begin{bmatrix} \mathbf{L} & \mathbf{0} \end{bmatrix}, \quad \mathbf{H}_{\text{re}} = \begin{bmatrix} \mathbf{0} & \mathbf{R} \end{bmatrix} \qquad \text{with } \frac{N-2}{2} \times (N-2) \text{ matrices } \mathbf{L} \text{ and } \mathbf{R}. \tag{1.17}$$

[4] Besides the special case of Haar wavelets in which $\mathbf{H}_{\text{in}}$ already constitutes the transformation matrix we want to construct.

Theorem 1.36 ([Bla21])**.** *Let* $\mathbf{H}_{\text{in}}$ *as in* (1.16)*,* $\mathbf{L}$ *and* $\mathbf{R}$ *as in* (1.17)*. Furthermore, let* $\mathbf{L}'$ *and* $\mathbf{R}'$ *be the orthonormalized rows of* $\mathbf{L}$ *respectively* $\mathbf{R}$*, which are obtained using the Gram-Schmidt procedure. Then,*

$$\begin{bmatrix} [\mathbf{L}' \quad \mathbf{0}] \\ \mathbf{H}_{\text{in}} \\ [\mathbf{0} \quad \mathbf{R}'] \end{bmatrix}$$

is an orthonormal $L \times L$ *matrix.*

We thus have constructed a procedure for the calculation of the boundary filters. One can show that the runtime of this procedure is linear in the signal length if we assume the filter length to be constant [Bla21].

2 Diffusion models

Diffusion models are a class of probabilistic generative models that allow sampling from highly complex probabilty distributions, introduced by Sohl-Dickstein et al. [SWMG15]. At the core of these models is a diffusion process that progressively morphs an input sample into white noise. Diffusion models aim to reverse that process in order to sample from the data distribution.

In this section we will first introduce a Markov chain based approach to diffusion models. Then we will discuss a different ansatz based on stochastic differential equations, which will turn out to be a generalization.

For the representation of diffusion models in this chapter, we follow Ho, Jain, and Abbeel [HJA20] for the discrete-time case while the continuous-time case is based on Song et al. [Son+21].

Definition 2.1 (Forward process)**.** Let q be the data probabilty distribution over $\mathbb{R}^d$ and $T > 0$ a sufficiently large time horizon. Let p be a well-behaved prior distribution, usually $p := \mathcal{N}(\mathbf{0}, \mathbf{I})$. The *forward process* $(\mathbf{x}_t)_{t\in\mathcal{T}}$ is an inhomogeneous Markov process that morphs q into p, i.e. $\mathbf{x}_0 \sim q$ and $\mathbf{x}_T \sim p$. We denote its transition kernel as $q(\mathbf{x}_t \mid \mathbf{x}_s)$ and its marginal densities as $q(\mathbf{x}_t)$.

In the *discrete-time* case we have $\mathcal{T} = \{0, \dots, T\}$. In the *continuous-time* case we set $\mathcal{T} = [0, T]$ and require $(\mathbf{x}_t)$ to have almost surely continuous sample paths, i.e. we require $(\mathbf{x}_t)$ to be a diffusion process.

Remark. The forward process is at the core of a diffusion model. Its role is to gradually destroy structure in the signals. If we learn to reverse the forward process, we learn to reconstruct this structure – the fundamental idea of diffusion models.

A visualization of a forward process is depicted in figure 2.1.

2.1 Diffusion models in discrete-time

There are two main formulations for diffusion models in discrete-time: denoising score matching with Langevin dynamics (SMLD) [SE19] and denoising diffusion probabilistic model (DDPM) [HJA20]. Both methods can be understood as score-based (cf. section 2.1.3) and both can be generalized in a similar fashion in continuous-time, as discussed in section 2.2. In this section we will introduce DDPMs.

A DDPM models both the forward process as well as its time reversal as discrete-time Markov chains. The forward chain gradually adds Gaussian noise to the data according to a fixed variance schedule $0 < \beta_1, \dots, \beta_T < 1$. This leads to the transition kernel

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}), \tag{2.1}$$

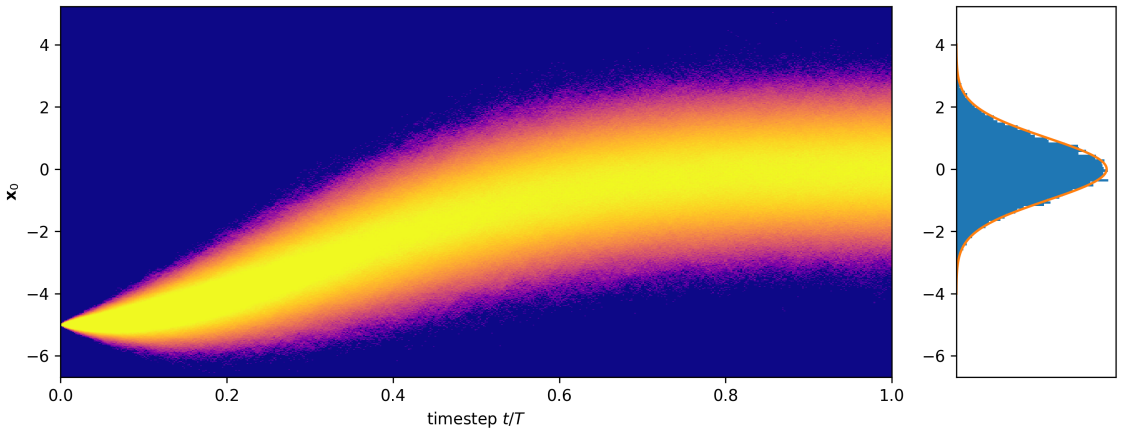


Figure 2.1: Visualization of a one-dimensional forward process. Left: estimation of $q(\mathbf{x_t} \mid \mathbf{x}_0)$ by generating 25000 sample paths with $\mathbf{x}_0 = -5$ on the left and $\mathbf{x}_T$ on the right. Right: Histogram of the generated samples $\mathbf{x}_T$ (blue) and the standard Gaussian density (orange). Inspired by Strümke and Langseth [SL23b].

or equivalently

$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_t \quad \text{for } \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{2.2}$$

Using the chain rule of probabilty and the Markov property yields

$$q(\mathbf{x}_1, \dots, \mathbf{x}_T \mid \mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}). \tag{2.3}$$

Lemma 2.2 (Closed-form sampling of the forward process)**.** *Using* $\alpha_t := 1 - \beta_t$ *and* $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$ *we have*

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{2.4}$$

or equivalently with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}. \tag{2.5}$$

Proof. We calculate using (2.3)

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t = \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{(1-\alpha_{t-1})\alpha_t}\boldsymbol{\epsilon}_{t-1} + \sqrt{(1-\alpha_t)}\boldsymbol{\epsilon}_t \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{(1-\alpha_{t-1})\alpha_t + (1-\alpha_t)}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}_{t-1}.
\end{aligned}$$

In the second step we used the fact that since $\boldsymbol{\epsilon}_t$ and $\boldsymbol{\epsilon}_{t-1}$ are independent and centered Gaussians, their sum is a centered Gaussian as well with variance equal to the sum of the variances of $\boldsymbol{\epsilon}_t$ and $\boldsymbol{\epsilon}_{t-1}$. Repeating this step yields the result. □

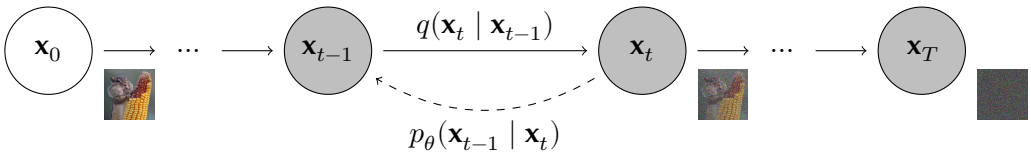


Figure 2.2: Scheme of a DDPM [HJA20].

2.1.1 Learning to reverse the diffusion

Figure 2.2 shows the forward and backward Markov chain schematically. A central result from Feller [Fel49], applied to this setting by Sohl-Dickstein et al. [SWMG15], allows us to construct the reverse transitions:

Theorem 2.3 ([Fel49; SWMG15]). *If all β_t are chosen sufficiently small, then the reversal of the diffusion process has the same functional form as the forward process and the reverse transitions follow Gaussian distributions.*

Thus, the goal of a DDPM is to estimate the parameters of the Gaussian reverse transition kernel p_θ:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)). \tag{2.6}$$

Following Ho, Jain, and Abbeel [HJA20], we set $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) := \sigma_t^2 \mathbf{I}$ with σ_t^2 as fixed hyperparameters.[1] The rest of this section is concerned with deriving the mean approximating function $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ in detail. The structure is based on Sohl-Dickstein et al. [SWMG15] and Ho, Jain, and Abbeel [HJA20], some calculations are adapted from Strümke and Langseth [SL23b].

Remark. In the definition of the forward process we defined the data distribution over $\mathbb{R}^d$, i.e. over a continuous space. However, images as the application we are interested in are of discrete nature, usually modeled with values in $\{0, 1, \dots, 255\}$ or as quantized values in the interval $[-1, 1]$. This can be addressed by introducing a discrete decoder bridging from the continuous endpoint of the reverse process $\mathcal{N}(\mathbf{x}_0; \boldsymbol{\mu}_\theta(\mathbf{x}_1, 1), \sigma_1^2 \mathbf{I})$ to the discrete data distribution $q(\mathbf{x}_0)$. As the addition of the discrete decoder does not affect the simplified loss function used in training we omit this detail here and refer to Ho, Jain, and Abbeel [HJA20] instead.

To learn $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ we aim to minimize the model's negative log likelihood (NLL). However, as optimizing the log likelihood directly is generally intractable we introduce the so-called evidence lower bound (ELBO) as our optimization target.

Lemma 2.4 (Evidence lower bound (ELBO)). *We have*

$$\log p_\theta(\mathbf{x}_0) = \underbrace{\int q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) \log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \, \mathrm{d}\mathbf{x}_{1:T}}_{=:L} + \underbrace{\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{1:T} \mid \mathbf{x}_0))}_{\geq 0} \tag{2.7}$$

[1] There are formulations that also train a model for the variances $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)$ and incorporate this into the loss function accordingly[ND21].

where $\mathrm{D}_{\mathrm{KL}}(\cdot \,\|\, \cdot)$ *denotes the Kullback-Leibler divergence (KL divergence) [KL51]. Thus,* L *is a lower bound on the log likelihood. This lower bound is called the evidence lower bound (ELBO).*

Proof. We calculate

$$\begin{aligned}
\log p_\theta(\mathbf{x}_0) &= \int q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) \log p_\theta(\mathbf{x}_0) \, \mathrm{d}\mathbf{x}_{1:T} \\
&= \int q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) \log \left[\frac{p_\theta(\mathbf{x}_{0:T})}{p_\theta(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \frac{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \right] \mathrm{d}\mathbf{x}_{1:T} \\
&= \int q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) \log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \, \mathrm{d}\mathbf{x}_{1:T} + \int q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) \log \frac{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}{p_\theta(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \, \mathrm{d}\mathbf{x}_{1:T} \\
&= L + \mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{1:T} \mid \mathbf{x}_0))
\end{aligned}$$

□

Using the ELBO we can derive an upper bound on the negative log likelihood in our setting:

Theorem 2.5. *We can bound the negative log likelihood*

$$-\log p_\theta(\mathbf{x}_0) \leq L_T + \sum_{t=1}^{T-1} L_t - L_0 \tag{2.8}$$

with

$$L_t := \mathbb{E}_{q(\mathbf{x}_{t+1} \mid \mathbf{x}_0)} \Big[\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_t \mid \mathbf{x}_{t+1}, \mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1})) \Big] \quad t = 1, \dots, t-1, \tag{2.9}$$

$$L_T := \mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_T \mid \mathbf{x}_0) \,\|\, p(\mathbf{x}_T)), \tag{2.10}$$

$$L_0 := \mathbb{E}_{q(\mathbf{x}_1 \mid \mathbf{x}_0)} \Big[\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1) \Big]. \tag{2.11}$$

Proof. Using $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0) = \frac{q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) q(\mathbf{x}_t \mid \mathbf{x}_0)}{q(\mathbf{x}_{t-1} \mid \mathbf{x}_0)}$, the Markov property of the forward process and a telescope sum property, we first calculate

$$\begin{aligned}
\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} &= \log \left[p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})} \right] \\
&= \log p_\theta(\mathbf{x}_T) + \sum_{t=1}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})} \\
&= \log p_\theta(\mathbf{x}_T) + \log \frac{p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)}{q(\mathbf{x}_1 \mid \mathbf{x}_0)} + \sum_{t=2}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0)} \\
&= \log p_\theta(\mathbf{x}_T) + \log \frac{p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)}{q(\mathbf{x}_1 \mid \mathbf{x}_0)} + \sum_{t=2}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)} \frac{q(\mathbf{x}_{t-1} \mid \mathbf{x}_0)}{q(\mathbf{x}_t \mid \mathbf{x}_0)} \\
&= \log \frac{p_\theta(\mathbf{x}_T)}{q(\mathbf{x}_T \mid \mathbf{x}_0)} + \log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1) + \sum_{t=2}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)}.
\end{aligned}$$

Plugging this into the ELBO yields

$$
\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq -\int q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) \log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \, \mathrm{d}\mathbf{x}_{1:T} \\
&= -\int q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) \left[\log \frac{p_\theta(\mathbf{x}_T)}{q(\mathbf{x}_T \mid \mathbf{x}_0)} + \log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1) + \sum_{t=2}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)} \right] \mathrm{d}\mathbf{x}_{1:T} \\
&= -\int q(\mathbf{x}_T \mid \mathbf{x}_0) \log \frac{p_\theta(\mathbf{x}_T)}{q(\mathbf{x}_T \mid \mathbf{x}_0)} \, \mathrm{d}\mathbf{x}_T - \int q(\mathbf{x}_1 \mid \mathbf{x}_0) \log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1) \, \mathrm{d}\mathbf{x}_1 \\
&\quad - \sum_{t=2}^{T} \int q(\mathbf{x}_{t-1}, \mathbf{x}_t \mid \mathbf{x}_0) \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)} \, \mathrm{d}\mathbf{x}_{t-1} \, \mathrm{d}\mathbf{x}_t \\
&= \overbrace{\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_T \mid \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))}^{=:L_T} - \overbrace{\int q(\mathbf{x}_1 \mid \mathbf{x}_0) \log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1) \, \mathrm{d}\mathbf{x}_1}^{=:L_0} \\
&\quad + \sum_{t=2}^{T} \underbrace{\int q(\mathbf{x}_t \mid \mathbf{x}_0) \, \mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)) \, \mathrm{d}\mathbf{x}_t}_{=:L_{t-1}} .
\end{aligned}
$$

□

Remark. L_T represents how close the result of the forward process is to our assumed prior distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. As we fixed the variance schedule β_t as hyperparameters this is a constant which we ignore in training.

We will now show how we can reparametrize L_t to formulate a simple loss function.

Lemma 2.6. *We have*

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$$

where

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\mathbf{x}_t \quad \textit{and} \quad \tilde{\beta}_t := \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t. \tag{2.12}$$

Proof. Using Bayes' rule we calculate

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_{t-1} \mid \mathbf{x}_0) q(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0)}{q(\mathbf{x}_t \mid \mathbf{x}_0)} \overset{\text{Markov}}{=} \frac{q(\mathbf{x}_{t-1} \mid \mathbf{x}_0) q(\mathbf{x}_t \mid \mathbf{x}_{t-1})}{q(\mathbf{x}_t \mid \mathbf{x}_0)}.$$

All of these terms can be evaluated using equations (2.1) and (2.4). □

Thus, all KL divergences in L_t compare Gaussians and are tractable in close form expressions:

Lemma 2.7 ([Duc07])**.** *Let $\mathcal{N}_a := \mathcal{N}(\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a)$ and $\mathcal{N}_b := \mathcal{N}(\boldsymbol{\mu}_b, \boldsymbol{\Sigma}_b)$ be d-dimensional multivariate Gaussian distributions. The KL divergence between both can be calculated as*

$$\mathrm{D}_{\mathrm{KL}}(\mathcal{N}_a \parallel \mathcal{N}_b) = \frac{1}{2}\left[\log \frac{\det \boldsymbol{\Sigma}_b}{\det \boldsymbol{\Sigma}_a} - d + \operatorname{tr}(\boldsymbol{\Sigma}_b^{-1}\boldsymbol{\Sigma}_a) + (\boldsymbol{\mu}_b - \boldsymbol{\mu}_a)^T \boldsymbol{\Sigma}_b^{-1}(\boldsymbol{\mu}_b - \boldsymbol{\mu}_a) \right] \tag{2.13}$$

Lemma 2.8. *For $t = 2, \dots, t$ we can reformulate L_{t-1} as*

$$L_{t-1} = \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\right\|^2\right] + C \tag{2.14}$$

where C is a constant that does not depend on θ.

Proof. The reverse kernel $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$ and the forward posterior $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I})$ are both Gaussians (equation (2.6) and lemma 2.6). We explicitly calculate their KL divergence:

$$\begin{aligned}
&\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)) \\
&= \frac{1}{2\sigma_t^2}\left\|\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) - \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0)\right\|^2 + \underbrace{\frac{1}{2}\left[\log\frac{\det\sigma_t^2\mathbf{I}}{\det\tilde{\beta}_t\mathbf{I}} - d + d\frac{\tilde{\beta}_t}{\sigma_t^2}\right]}_{=:C}.
\end{aligned}$$

□

Lemma 2.8 offers a straightforward parametrization of $\boldsymbol{\mu}_\theta$ as an approximation of the mean of the forward process posterior $\tilde{\boldsymbol{\mu}}$. Nevertheless, we continue by reparametrizing $\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) = \sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}$ for $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ using (2.5). Plugging this into equation (2.12) yields

$$\begin{aligned}
&\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}), \mathbf{x}_0) = \tilde{\boldsymbol{\mu}}_t\Big(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}), \frac{1}{\sqrt{\overline{\alpha}_t}}(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon})\Big) \\
&= \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t}\frac{1}{\sqrt{\overline{\alpha}_t}}\left(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}\right) + \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) \\
&= \left[\frac{\beta_t}{(1-\overline{\alpha}_t)\sqrt{\alpha_t}} + \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\right]\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon} \\
&= \left[\frac{\beta_t + \alpha_t - \overline{\alpha}_t}{(1-\overline{\alpha}_t)\sqrt{\alpha_t}}\right]\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon} \\
&= \frac{1}{\sqrt{\alpha_t}}\left[\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}\right]
\end{aligned}$$

This allows us to reformulate L_{t-1} as follows:

$$L_{t-1} - C = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}\Big) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}), t)\right\|^2\right]. \tag{2.15}$$

Thus, $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ should predict $\frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}\Big)$. As $\mathbf{x}_t$ is available to the model we can choose to approximate the noise $\boldsymbol{\epsilon}$ from $\mathbf{x}_t$ instead using a noise prediction model $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$. This leads to the parametrization

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\Big). \tag{2.16}$$

If we apply this parametrization to (2.15) we get

$$L_{t-1} - C = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\overline{\alpha}_t)}\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]. \tag{2.17}$$

The formulation (2.17) of the loss function, i.e. learning a noise predictor $\boldsymbol{\epsilon}_\theta$ and substracting the predicted noise as the core of the reverse diffusion step (cf. equation (2.16)), lends denoising diffusion probabilistic models their name.

Remark. The computations above can be repeated in a similar fashion for L_0 using $\mathbf{x}_0$ instead of $\tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0)$.

2.1.2 A simplified loss function and sampling from a DDPM

In the previous section we have derived that training a DDPM amounts to learning the mean $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ of the Gaussian reverse transition kernel. We have seen that this can be either done directly by approximating the mean of the forward process posterior $\tilde{\boldsymbol{\mu}}$ or by learning predictor for the added white noise $\boldsymbol{\epsilon}$. For both cases minimizing the ELBO of the negative log likelihood corresponds to minimizing a weighted mean squared error loss.

Ho, Jain, and Abbeel [HJA20] found that training the noise predictor to be more stable in practice. Additionally, they found training on an unweighted variant of the loss beneficial to sample quality, i.e. to minimize

$$L_{\text{simple}} := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right] \tag{2.18}$$

with $t \sim \mathcal{U}(\{1, \dots, T\})$. Besides simplicity the usage of an unweighted variant of the ELBO is advantageous as the weighting term is large for small t. Removing these large weights emphasizes timesteps with a larger amount of noise that are more difficult to denoise [HJA20]. The resulting training algorithm for a DDPM is displayed in algorithm 1.

Algorithm 1 DDPM training [HJA20]

1: **repeat**
2: Draw $\mathbf{x}_0 \sim q$.
3: Sample $t \sim \mathcal{U}(\{1, \dots, T\})$.
4: Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
5: Take a gradient descent step on $\nabla_\theta\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$.
6: **until** converged

To sample from $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ with a noise predictor $\boldsymbol{\epsilon}_\theta$ we apply equations (2.6) and (2.16) and have

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z} \qquad \text{with } \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

Guided by experimental evaluation [ND21] we use $\sigma_t^2 = \tilde{\beta}_t$ as defined in equation (2.12). In the case of $t = 1$, we follow Ho, Jain, and Abbeel [HJA20] and use the calculated mean of the forward process posterior directly as our sample, skipping the addition of noise. The resulting algorithm is displayed in algorithm 2.

However, this sampling procedure requires T evaluations of $\boldsymbol{\epsilon}_\theta$. As evaluating a large deep learning model hundreds of times is very expensive, a major focus in research on

Algorithm 2 DDPM full sampling [HJA20]

Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
for $t = T, \dots, 1$ **do**
  Set $\mathbf{x}_{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right)$
  if $t > 1$ **then**
    Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
    $\mathbf{x}_{t-1} \leftarrow \mathbf{x}_{t-1} + \sqrt{\widetilde{\beta}_t}\mathbf{z}$.
  end if
end for
return $\mathbf{x}_0$

diffusion models is reducing the cost of sampling, preferably without affecting the training procedure [SME21; Lu+22; Rom+22; Jol+21].

A simple approach is to perform inference on a coarser variance schedule derived from the variance schedule used in training [SME21; ND21]. Let $(S_k)_{k=1,\dots,K}$ for $K < T$ be an increasing subsequence of $(1, \dots, T)$ of length K. We construct the sampling variance schedule from the training noise schedule $\overline{\alpha}_t$ as

$$\beta_k^{\mathrm{s}} := 1 - \frac{\overline{\alpha}_{S_k}}{\overline{\alpha}_{S_{k-1}}}, \qquad \widetilde{\beta}_k^{\mathrm{s}} := \frac{1 - \overline{\alpha}_{S_{k-1}}}{1 - \overline{\alpha}_{S_k}}\beta_k^{\mathrm{s}}$$

The sampling is done using this coarser schedule, which results in algorithm 3. Multiple approaches to construct the subsequence (S_k) are used in practice [LLLY24], e.g. dividing the uniformly discretizing the interval $[1, T]$ and rounding.

Algorithm 3 DDPM reduced step sampling [SME21; ND21]

Require: Number of inference steps $1 \leq K \leq T$.
Construct sampling timesteps (S_k) of length K.
Sample $\mathbf{x}_{S_K} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
for $t = K, \dots, 1$ **do**
  Set $\mathbf{x}_{S_{t-1}} \leftarrow \frac{1}{\sqrt{1-\beta_t^{\mathrm{s}}}}\left(\mathbf{x}_{S_t} - \frac{\beta_t^{\mathrm{s}}}{\sqrt{1-\overline{\alpha}_{S_t}}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_{S_t}, S_t)\right)$
  if $t > 1$ **then**
    Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
    $\mathbf{x}_{S_{t-1}} \leftarrow \mathbf{x}_{S_{t-1}} + \sqrt{\widetilde{\beta}_t^{\mathrm{s}}}\mathbf{z}$.
  end if
end for
return $\mathbf{x}_0$

2.1.3 DDPM as a score-based model

We have seen that training a DDPM is done by optimizing the unweighted variational bound L_{simple}. Our interpretation in the previous sections of this goal is training a noise

predictor to gradually reconstruct $\mathbf{x}_0$ by removing the predicted noise—a process that can be described by "denoising". In this section, will introduce a different interpretation of the loss function, helping us in section 2.2 to generalize to continuous-time.

Definition 2.9 (Score)**.** Let p be the probability density of a distribution $\mathcal{P}$. We call $\nabla_{\mathbf{x}_t} \log p$ the *score*[2] of the distribution $\mathcal{P}$ where log denotes the natural logarithm.

If we now calculate the score of $q(\mathbf{x}_t \mid \mathbf{x}_0)$ of a DDPM using (2.4), we get

$$-\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t \mid \mathbf{x}_0) = \nabla_{\mathbf{x}_t} \frac{\|\mathbf{x}_t - \sqrt{\overline{\alpha}_t}\mathbf{x}_0\|^2}{2(1-\overline{\alpha}_t)} = \frac{\mathbf{x}_t - \sqrt{\overline{\alpha}_t}\mathbf{x}_0}{1-\overline{\alpha}_t} \overset{(2.5)}{=} \frac{1}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}.$$

Hence, the reparametrization equation (2.18) can also be interpreted as

$$\begin{aligned} L_{\text{simple}} &= \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}}\boldsymbol{\epsilon}, t)\right\|^2\right] \\ &= \mathbb{E}_{t,\mathbf{x}_0,q(\mathbf{x}_t|\mathbf{x}_0)}\left[(1-\overline{\alpha}_t)\left\|\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t \mid \mathbf{x}_0) - \mathbf{s}_\theta(\mathbf{x}_t, t)\right\|^2\right] \end{aligned} \tag{2.19}$$

with $\mathbf{s}_\theta(\mathbf{x}_t, t) := -\sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$.

We can therefore view the optimization goal as estimating the score function. Models with this goal are called *score-based* [SE19]. The particular formulation of the score estimation is known as denoising score matching [Vin11] which builds a bridge between the denoising task we have considered so far and score estimations. There are other score estimation techniques available that can be used instead [HD05; Vin11; SGSE20].

2.1.4 Taking the time horizon to infinity

The noise schedule in the construction of a DDPM depends implicitly on the choice of the time horizon T. The larger we choose T and the more diffusion steps we thus make, the smaller the variance of the added noise β_t must be. Expanding the time horizon T leads to improvements in practice while also increasing the required amount of compute effort [ND21]. We are interested in the result of making infinitesimal small steps in the Markov chain. We sketch a derivation based on Song et al. [Son+21].

We consider the forward transition kernels equation (2.2)

$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\mathbf{z}_{t-1} \quad \text{for } \mathbf{z}_{t-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

First, we define an auxillary variance schedule $\widehat{\beta}_i^{(T)} := T\beta_i^{(T)}$ where the superscript denotes that $\widehat{\beta}^{(T)}$ is defined with respect to a variance schedule for time horizon T. We assume that $\beta_i^{(T)}$ were constructed by uniformly discretizing a sufficiently regular variance schedule function $\beta\colon [0,1] \to [0,1]$, i.e.

$$\beta\left(\frac{i}{T}\right) = \widehat{\beta}_i^{(T)} \quad \text{for all } T > 0.$$

[2] Note that in context of diffusion models the score is defined contrary to statistics, where the gradient w.r.t. the model parameters is used.

For a given timescale T, let $\Delta t := T^{-1}$ and $\mathbf{x}, \mathbf{z}$ be functions over $[0, 1]$ with

$$\mathbf{x}(i\Delta t) = \mathbf{x}_i \quad \text{and} \quad \mathbf{z}(i\Delta t) = \mathbf{z}_i.$$

Now, inserting this into equation (2.2) and doing a Taylor approximation yields

$$\begin{aligned}\mathbf{x}(t+\Delta t) &= \sqrt{1-\beta(t+\Delta t)\Delta t}\mathbf{x}(t) + \sqrt{\beta(t+\Delta t)\Delta t}\mathbf{z}(t) \\ &= \mathbf{x}(t) - \frac{1}{2}\beta(t+\Delta t)\Delta t\mathbf{x}(t) + \sqrt{\beta(t+\Delta t)\Delta t}\mathbf{z}(t) + o(\Delta t) \\ &= \mathbf{x}(t) - \frac{1}{2}\beta(t)\Delta t\mathbf{x}(t) + \sqrt{\beta(t)\Delta t}\mathbf{z}(t) + o(\Delta t)\end{aligned}$$

Rearanging the terms leads to the difference quotient

$$\frac{\mathbf{x}(t+\Delta t) - x(\mathbf{x}(t))}{\Delta t} = -\frac{1}{2}\beta(t)\mathbf{x}(t) + \sqrt{\beta(t)}\frac{\mathbf{z}(t)}{\sqrt{\Delta t}} + \frac{o(\Delta t)}{\Delta t}$$

which results in the following SDE for $\Delta t \to 0$:

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\mathbf{w} \tag{VP SDE}$$

where $\mathbf{w}$ denotes a Wiener process.

Remark. The calculation shows that DDPMs can be regarded as a discretization of a SDE using a Euler-Maruyama scheme [KP92]. We call the SDE *variance preserving* (*VP*) as its solution has a constant unit variance if $\mathbf{x}(0)$ has unit variance [Son+21].

2.2 Diffusion models in continuous-time

In the previous section we have seen that DDPMs can be understood as a discretization of a particular stochastic differential equation (SDE). The goal of this section is to introduce a more general framework for this notion based on Song et al. [Son+21].

Recalling definition 2.1, the forward diffusion process in continuous-time is defined as $\{\mathbf{x}_t\}_{t=0}^T$ with $\mathbf{x}_0 \sim q$, $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and almost surely continuous sample paths. We now model $\mathbf{x}$ as the solution of an Itô SDE

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w} \tag{2.20}$$

where $\mathbf{w}$ is the standard Wiener process, $\mathbf{f}(\cdot, t)\colon \mathbb{R}^d \to \mathbb{R}^d$ the so-called *drift* coefficient of $\mathbf{x}_t$ and $g\colon \mathbb{R} \to \mathbb{R}$ a scalar-valued function called *diffusion* coefficient of $\mathbf{x}_t$.[3] This SDE has a unique strong solution in both state and time if the coefficients are globally Lipschitz in both state and time [Øks03, Theorem 5.2.1].

Example 2.10 (VP forward SDE)**.** The (VP SDE) can be described in this framework with

$$\mathbf{f}(\mathbf{x}_t, t) = -\frac{1}{2}\beta(t)\mathbf{x}_t, \qquad g(t) = \sqrt{\beta(t)}.$$

[3] The framework can be extended to matrix-valued diffusion functions as well as diffusion functions depending on $\mathbf{x}$ [Son+21].

Example 2.11 (VE forward SDE)**.** In section 2.1 we introduced DDPMs, one of the two main formulations of diffusion models in discrete-time, and associated it to the (VP SDE). The other formulation, denoising score matching with Langevin dynamics (SMLD) [SE19], can be associated to a SDE as well, the so-called *variance exploding* (*VE*) SDE [Son+21]. In the continuous-time diffusion model framework, the VE SDE can be described as

$$\mathrm{d}\mathbf{x} = \sqrt{\frac{\mathrm{d}[\sigma^2(t)]}{\mathrm{d}t}}\,\mathrm{d}\mathbf{w} \tag{VE SDE}$$

for an increasing noise schedule $\sigma\colon [0,T] \to \mathbb{R}_+$. The name of the VE SDE stems from the fact that it always yields a process with exploding variance when $t \to \infty$ [Son+21].

A central result by Anderson [And82] allows us to construct the reverse stochastic process, playing the role of theorem 2.3 in continuous-time:

Theorem 2.12 ([And82])**.** *The reversal of a diffusion process is also a diffusion process given by the reverse-time SDE:*

$$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}_t, t) - g(t)^2 \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)]\,\mathrm{d}t + g(t)\,\mathrm{d}\bar{\mathbf{w}}. \tag{2.21}$$

with $\bar{\mathbf{w}}$ *reverse-time Wiener process and* $\mathrm{d}t$ *negative timestep.*

Given a model to estimate the score of the marginal distribution, we can thus sample from the diffusion model by simulating the SDE (2.21).

The score function can be estimated using denoised score matching [Vin11], an approach we have encountered in section 2.1.3. To estimate the score function $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)$ we train a time-dependent score-based model $\mathbf{s}_\theta(\mathbf{x}, t)$ to minimize

$$\mathbb{E}_{t,\mathbf{x}_0,q(\mathbf{x}_t|\mathbf{x}_0)}\left[\lambda(t)\left\|\mathbf{s}_\theta(\mathbf{x}_t,t) - \nabla_{\mathbf{x}_t}\log q(\mathbf{x}_t \mid \mathbf{x}_0)\right\|^2\right] \tag{2.22}$$

with $t \sim \mathcal{U}([0,T])$ and $\lambda\colon [0,T] \to \mathbb{R}_{>0}$ a weighting function. Equation (2.22) can be considered a continuous generalization of equation (2.19) with

$$\lambda \propto \frac{1}{\mathbb{E}[\|\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t \mid \mathbf{x}_0)\|^2]}.$$

To evaluate equation (2.22) we typically need to know the transition kernel $q(\mathbf{x}_t \mid \mathbf{x}_0)$. When $f(\cdot, t)$ is affine, the transition kernel is a Gaussian which parameters can often be obtained in closed form [Son+21].

Example 2.13 (VP reverse SDE)**.** The reversal of the (VP SDE) is

$$\mathrm{d}\mathbf{x} = \left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\nabla_{\mathbf{x}_t}\log q(\mathbf{x}_t)\right]\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\bar{\mathbf{w}}. \tag{2.23}$$

As the drift is an affine function, we can calculate the transition kernel in closed form [Son+21]:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t; \mathbf{x}_0 \exp\left(-\frac{1}{2}\int_0^t \beta(s)\,\mathrm{d}s\right), \left(1 - \exp\left(-\int_0^t \beta(s)\,\mathrm{d}s\right)\right)\mathbf{I}\right). \tag{2.24}$$

2.2.1 Sampling from continuous-time diffusion models

Sampling from continuous-time diffusion model is done by solving the reverse SDE (2.21). This allows for a variety of choices regarding the solver. Using a simple Euler-Maruyama solver for the (VP SDE) leads to DDPMs as discussed in section 2.1.4. We also have a multitude of general purpose SDE solvers available, e.g. stochastic Runge-Kutta methods [KP92].

To improve on general purpose solvers Song et al. [Son+21] propose so-called predictor-corrector samplers. The general purpose numerical solvers acts as a predictor, proposing an estimate of the sample at the next timestep. Using the information that we use a score-based model $\mathbf{s}_\theta(\mathbf{x}, t)$, we can employ score-based Markov chain Monte-Carlo approaches to directly sample from $q(\mathbf{x}_t)$[Par81; GM94] and correct the sample from the numerical solver.

Another approach considers a *deterministic* ordinary differential equation (ODE) that matches the marginals of the SDE at each timestep instead:

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

This ODE is called the *probability flow ODE* [Son+21]. Solving it allows us to deterministically map noise vectors $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to samples $\mathbf{x}_0$ from the data distribution q. In case of a time-dependent score-based model $\mathbf{s}_\theta(\mathbf{x}, t)$ the probability flow ODE is an instance of a neural ODE [CRBD18], allowing to deploy various solvers [Kid21].

2.3 Choosing the noise schedule

An important detail in the design of diffusion models is the choice of the noise schedule which governs the interpolation between the input sample $\mathbf{x}_0$ and white noise $\boldsymbol{\epsilon}$ at x_T. The noise schedule determines how much information of the input sample is available at intermediate timesteps t, deciding how fast the structure is destroyed in the forward diffusion. In practice the choice of the noise schedule has high effect on the performance of a diffusion model [ND21].

There are two main ways to parametrize a noise schedule in discrete time. Either by choosing the variance schedule β_t of the transition kernels or by choosing the paramters $\overline{\alpha}_t$ used in the closed-form sampling of the forward process equation (2.4). Both can derived from the other as

$$\beta_t = 1 - \frac{\overline{\alpha}_t}{\overline{\alpha}_{t-1}} \quad \text{with } \overline{\alpha}_0 = 1 \qquad \text{and} \qquad \overline{\alpha}_t = \prod_{s=1}^{t} (1 - \beta_s).$$

To ensure $\mathbf{x}_0 \sim q$ and $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we require $\overline{\alpha}_0 \approx 1$ and $\overline{\alpha}_T \approx 0$.

Ho, Jain, and Abbeel [HJA20] propose to derive a noise schedule for DDPMs by linear interpolation between fixed β_1 and β_T. For $T = 1000$ they choose $\beta_1 = 10^{-4}$ and $\beta_T = 0.02$. The resulting values of $\overline{\alpha}_t$ are depicted in figure 2.3 together with further noise schedules.

Nichol and Dhariwal [ND21] observed that for this linear schedule $\overline{\alpha}_t$ was approximately zero for the last 20 % of timesteps, making this part of the reverse process an ineffecient

use of compute. To address this Nichol and Dhariwal [ND21] propose the so called *cosine schedule*, parametrizing $\overline{\alpha}_t$ directly:

$$\overline{\alpha}_t = \frac{f(t)}{f(0)}, \quad f(t) := \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)^2 \tag{2.25}$$

where $s = 0.008$ is an offset value introduced to improve training stability for small t.

In case of a discretization of a continuous-time diffusion model with an affine drift function, the implicitely used noise schedule can be derived from the forward transition kernel. For the (VP SDE) this yields

$$\overline{\alpha}_k = \exp\left(-\int_0^{t_k} \beta(s)\,\mathrm{d}s\right)$$

where $\{t_k\}_{k=1}^K$ is a discretization of $[0, T]$.

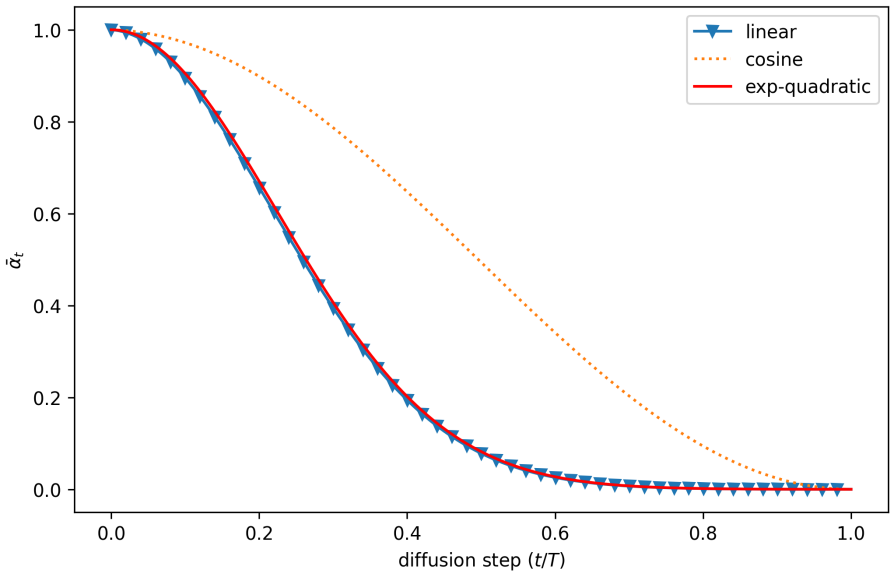


Figure 2.3: $\bar{\alpha}_t$ during the forward diffusion process for a linear schedule[HJA20], a cosine schedule[ND21] and an exponential schedule with quadratic timestep sampling with $T = 5$ [GCDM22] for 1000 timesteps each. The curves for the cosine schedule and the exponential schedule lie on top of each other.

A particular instance of the (VP SDE) used in the next chapter uses $\beta \equiv 2$ together with a quadratic discretization, i.e.

$$t_k = \left(\frac{k}{K}\right)^2 T$$

In this case, we have

$$\overline{\alpha}_k = \mathrm{e}^{-2k^2TK^{-2}} \qquad \text{and} \qquad \beta_k = 1 - \mathrm{e}^{-2TK^{-2}[k^2-(k-1)^2]} = 1 - \mathrm{e}^{-2TK^{-2}(2k-1)}.$$

Using the identity $1 - \mathrm{e}^{-z} \approx z$ for small z, we get $\beta_k \approx 2T\frac{2k-1}{K^2}$ which for $K = 1000$ and $T = 5$ approximately corresponds to the linear schedule [HJA20] as can be seen in figure 2.3.

3 Wavelets for diffusion models

Diffusion models have shown remarkable success in generating high-fidelity images. Architecturally, they are built around a score-based model $\mathbf{s}_\theta$ approximating the score of a marginal distribution. As we use denoising score matching for the score approximation (cf. section 2.1.3) the model is a noise predictor.

Wavelet representations have a long standing track record in signal processing. They have been used successfully for denoising in the context of classical signal processing [SN96] as well as in model approaches [HD22; Tia+23], motivating their use for denoising score matching. In the context of deep learning architectures, wavelets see increasingly frequent use, e.g. in image colorization [Li+23], image superresolution [HHST17; Mos+23; SSLZ23], image enhancement [Liu+20; Jia+23; Hua+23] and video enhancement [Wan+20], style transfer [Yoo+19] and generative image models [Zha+22; GHBC21; GCDM22; PDT23; VWG23; Yua+23; YZFW23]. To explain the widespread adoption of UNets [RFB15] as the backbone of diffusion models Williams et al. [Wil+23] show that a representation in the Haar wavelet basis is learned through maxpooling layers which leads to a good inductive bias.

For diffusion models alike other generative image models such as generative adversarial networks (GANs) [Goo+14], discrepencies in capturing the spectral data distributions are reported [DKK20; WBHG22; Fra+20; GHBC21; Rah+19]. In particular, DDPMs have been observed to be biased towards the dominant frequency bands of the data distribution leading to defects in the recovery of high frequencies, especially for DDPMs with a small backbone model [YZFW23]. This frequency bias motivates the use of signal representations in the spectral domain, e.g. using a DWT or WPT.

One idea to address the frequency bias is to run the diffusion process in wavelet space instead of pixel space, learning the structure of frequency bands directly. To that end the data distribution $\mathbf{x}_0 \sim q$ is reformulated over wavelet coefficient space and the forward process corrupts the wavelet coefficients instead of the image pixels. To sample new images, new wavelet coefficients are generated and reassembled using the iFWT. Multiple variants of this approach have been examined: Yuan et al. [Yua+23] generate all coefficients of a single level Haar wavelet transform and observe increased image quality. Phung, Dao, and Tran [PDT23] also uses the single level Haar wavelet transform, focussing on reducing the inference time while retaining high image quality as applying the wavelet transformation halves the spatial dimensions. Training diffusion models on the low-frequency coefficients of multilevel Haar wavelet transforms have been used in related tasks like image restoration [Hua+23] and image enhancement [Jia+23].

These results limit the generated wavelet coefficients to one decomposition level and are thus no multiscale approach However, using multiple scales in diffusion models has been found to be advantagous, e.g. in so-called *cascaded diffusion models* [Sah+22; Ho+22a] that successively sample images of a higher resolution, conditioned on the generated

image from the previous resolution. While not using wavelets explicitely this approach is closely related to a multiscale wavelet approach successively generating the low-frequency coefficients.

In section 3.1 we introduce an approach that uses a multiscale wavelet approach that successively generates the *high-frequency coefficients* conditioned on the previous level low-frequency coefficients, based on Guth et al. [GCDM22]. As we will discuss, this offers provable advantages for multiscale Gaussian processes which in practice also extend to image synthesis.

In addition to the change of domains, wavelets are used to improve the noise estimation backbone of a diffusion model, either in the wavelet domain [PDT23; Yua+23] or by including the wavelet transform into the archtitecure also in pixel domain [YZFW23]. Furthermore, wavelets are used to adapt the loss function, e.g. to improve training convergence [PDT23] or to address the spectral bias [VWG23]. In section 3.2 we will introduce one approach of using the wavelet domain structure in the noise estimation and propose an extension using the WPT.

3.1 Multiscale diffusion models in the wavelet domain

We consider the continuous-time diffusion model determined by the Itô SDE

$$\mathrm{d}\mathbf{x} = -\mathbf{x}\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}\mathbf{w} \tag{3.1}$$

which we recognize as the (VP SDE) with $\beta \equiv 2$. Using equation (2.24) we get that the solution is the Ornstein-Uhlenbeck process [SL23a]:

$$\mathbf{x} = \mathrm{e}^{-t}\mathbf{x}_0 + \sqrt{1-\mathrm{e}^{-2t}}\mathbf{z} \qquad \text{for } \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

Sampling from this diffusion model is done by solving the reverse SDE (2.23):

$$\mathrm{d}\mathbf{x} = -[\mathbf{x} + 2\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)]\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}\bar{\mathbf{w}}$$

with a negative infinitesimal timestep $\mathrm{d}t$ and a reverse Wiener process $\bar{\mathbf{w}}$. Given a score-approximating model $\mathbf{s}(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)$, we can approximate the SDE by discretizing at times $\{t_k\}_{k=0}^N$ with $t_N = T$ and $t_0 = 0$ and a stepsize $\delta_k := t_k - t_{k-1}$:

$$\tilde{\mathbf{x}}_{t_{k-1}} = \tilde{\mathbf{x}}_{t_k} + \delta_k[\tilde{\mathbf{x}}_{t_k} + \mathbf{s}(\tilde{\mathbf{x}}_{t_k}, t_k)] + \sqrt{2\delta_k}\boldsymbol{\epsilon}_k \qquad \text{with } \boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \text{ and } \tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{3.2}$$

In the following, we will ignore the approximation error of the score, i.e. we assume $\mathbf{s}(\mathbf{x}_t, t) = \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)$.

3.1.1 Discretization error and score regularity

We are interested how the regularity of the score function $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)$ affects the discretization $\tilde{\mathbf{x}}$. Denoting the distribution of the sampled images $\tilde{\mathbf{x}}_0$ as $\tilde{q}$, we thus look for a bound of the distance between $\tilde{q}$ and the data distribution q as function of the score.

First, we consider the simple case of a Gaussian data distribution $\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{\Sigma})$ in d dimensions. Let κ be the condition of the covariance matrix $\mathbf{\Sigma}$. Assuming a normalized signal energy, i.e. $\operatorname{tr} \mathbf{\Sigma} = d$, and a uniform discretization with $\delta_k = \delta$ Guth et al. [GCDM22, Theorem 1] show that

$$\mathrm{D}_{\mathrm{KL}}(q \parallel \tilde{q}) \leq E_T + E_\delta + E_{T,\delta}$$

with the following error terms: E_T stemming from the mismatch between $\mathbf{x}_T$ and $\tilde{\mathbf{x}}_T$, E_δ representing the error through the time discretization and $E_{T,\delta}$ a higher-order term with $E_{T,\delta} = \mathcal{O}(\delta + \mathrm{e}^{-4T})$ when $\delta \to 0$ and $T \to \infty$. Importantly, for any approximation error $\varepsilon > 0$ there exists $T, \delta \geq 0$ such that

$$\frac{E_T + E_\delta}{d} \leq \varepsilon \quad \text{and} \quad T/\delta \leq C\varepsilon^{-2}\kappa^3$$

with $C \geq 0$ a constant. This gives us a direct bound on the number of time steps $N = T/\delta$ necessary to achieve a fixed error depending on the condition κ of the covariance matrix. If κ increases we can thus expect the necessary number of timesteps to follow suit. Guth et al. [GCDM22, Theorem 2] also extend this result to non-Gaussian processes, relating the bound on the discretization error on the regularity of the score function. As the bound on the discretization error is determined by the condition κ in the Gaussian case, Guth et al. [GCDM22] conjecture for non-Gaussian processes that an ill-conditioned covariance matrix requires a high number of discretization steps to reach a small discretization error.

Natural images contain a wide range of frequency bands as they contain high-level structures as well as fine details. Their power spectrum is often modeled to follow a power law decay [GCDM22]

$$P(\omega) \sim (\xi^\eta + |\omega|)^{-1}$$

with $\eta = 1$ and $\eta \approx 2\pi/L$ for $L \times L$ images. The score of the resulting process is not well-conditioned [GCDM22].

3.1.2 Preconditioning through normalized wavelet coefficients

As we have seen in the previous section the number of discretization steps necessary depends on the condition of the covariance matrix. We will now show that using normalized wavelet coefficients acts as a preconditioner, thus lowering the bound on the number of timesteps. We will focus on the case of quadratic 2d images but more general formulations are possible [GCDM22].

Let $\mathbf{L}$ and $\mathbf{B}$ be the subsampled convolutional operators of the 2d FWT for an orthonormal wavelet and $\mathbf{L}^T$, $\mathbf{B}^T$ their inverse counterparts. The FWT is applied to an $L \times L$ input image $\mathbf{x}$, successively decomposing the signal into a low-frequency component with $(2^{-j}L)^2$ entries and three high-frequency components of the same size. We denote the normalized low-frequency component on level j with $2^{j-1} \geq 1$ as $\mathbf{x}_j$ with $\mathbf{x}_0 = \mathbf{x}$ and the corresponding normalized detail coefficients as $\bar{\mathbf{x}}_j$. We calculate $\mathbf{x}_j$ and $\bar{\mathbf{x}}_j$ as

$$\mathbf{x}_j = \gamma_j^{-1}\mathbf{L}\mathbf{x}_{j-1} \qquad \text{and} \qquad \bar{\mathbf{x}}_j = \gamma_j^{-1}\mathbf{B}\mathbf{x}_{j-1}.$$

Note that $\mathbf{x}_j$ and $\bar{\mathbf{x}}_j$ differ from the regular FWT by the inclusion of a normalization factor γ_j guaranteeing that $\mathbb{E}[\|\bar{x}_j\|^2] = 3(2^{-j}L)^2$. The decomposition is calculated for J levels where typically $J \approx \log_2 L$.

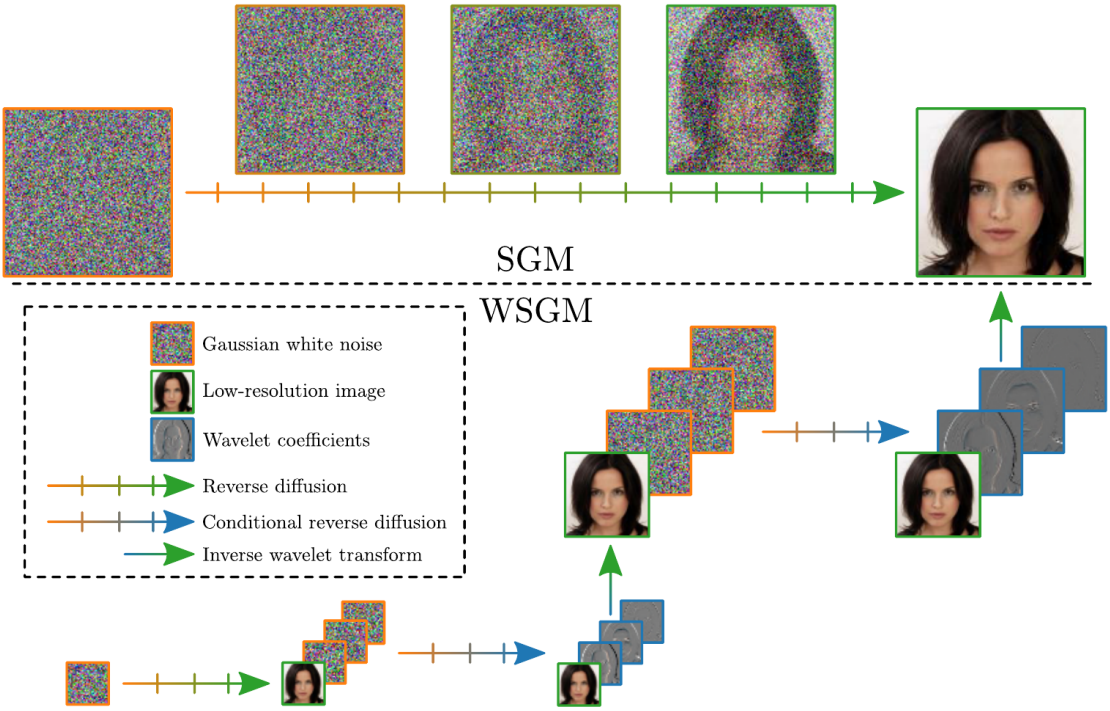


Figure 3.1: Scheme of a wavelet score-based generative model (WSGM) [GCDM22].

Definition 3.1 (Conditional wavelet renormalization, [GCDM22])**.** The *conditional wavelet renormalization* of a data distribution $\mathbf{x} \sim q$ is a factorization into conditional probabilities over normalized wavelet coefficients:

$$q(\mathbf{x}) = \alpha q_J(\mathbf{x}_J) \prod_{j=1}^{J} \bar{q}_j(\bar{\mathbf{x}}_j \mid \mathbf{x}_j) \tag{3.3}$$

where the scalar α depends on the normalization factors γ_j.

If $q(\mathbf{x})$ is a distribution over image data, typically q is highly non-Gaussian as are $\bar{q}_j(\bar{\mathbf{x}}_j)$ due to the sparse nature of wavelet detail coefficients. However, it has been observed that the conditional distrbutions $\bar{q}_j(\bar{\mathbf{x}}_j)$ are much closer to Gaussians [WS99]. Furthermore, the normalized wavelet coefficients $\bar{\mathbf{x}}_j$ have a white spectrum by choice of normalization. Therefore, $\bar{q}_j(\bar{\mathbf{x}}_j \mid \mathbf{x}_j)$ should be closer to a white Gaussian distribution, leading to a well-conditioned covariance matrix. This allows us to use fewer discretization steps if to sample from $\bar{q}_j(\bar{\mathbf{x}}_j \mid \mathbf{x}_j)$.

The coefficient $\mathbf{x}_J$ of the largest scale J is the result of repeated applications of a low-pass filter. We can consider low-pass filtering as averaging (cf. section 1.1.3). Hence, $\mathbf{x}_J$ is close to Gaussian if the image $\mathbf{x}$ has independent structures [GCDM22]. Note that in

practice this does not hold, as we stop the FWT early to avoid the domination of boundary effects. Nevertheless, in theory equation (3.3) gives us a factorization of q into a product of distributions that are all close to Gaussian.

A wavelet score-based generative model (WSGM) [GCDM22] embraces this notion by successively sampling from all factors of (3.3). First, an unconditional diffusion model is used to sample $\mathbf{x}_J$. Then, conditional diffusion models successively sample $\bar{\mathbf{x}}_j$ conditioned on the previously generated $\mathbf{x}_j$. We apply the normalized iFWT to construct $\mathbf{x}_{j-1}$ from $\mathbf{x}_j$ and $\bar{\mathbf{x}}_j$.

For each scale $j \leq J$, we define the forward processes as

$$\mathrm{d}\bar{\mathbf{x}}_j = -\bar{\mathbf{x}}_j \,\mathrm{d}t + \sqrt{2}\,\mathrm{d}\bar{\mathbf{w}}_j \quad \text{and} \quad \mathrm{d}\mathbf{x}_J = -\mathbf{x}_J \,\mathrm{d}t + \sqrt{2}\,\mathrm{d}\mathbf{w}_J \tag{3.4}$$

where $\bar{\mathbf{w}}_j$ and $\mathbf{w}_J$ are Wiener processes. For each forward process we train a score-based model to approximate the score of the marginals of the forward processes, i.e.

$$\mathbf{s}_{\theta,J}(\mathbf{x}_{J,t}, t) \approx \nabla_{\mathbf{x}_{J,t}} \log q(\mathbf{x}_{J,t}) \quad \text{and} \quad \bar{\mathbf{s}}_{\theta,j}(\bar{\mathbf{x}}_{j,t}, t, \mathbf{x}_j) \approx \nabla_{\bar{\mathbf{x}}_{j,t}} \log q(\bar{\mathbf{x}}_{j,t} \mid \mathbf{x}_j).$$

To sample we adapt the discretization (3.2). First, $\mathbf{x}_J$ is sampled unconditionally using

$$\mathbf{x}_{J,t_{J,k-1}} = \mathbf{x}_{J,t_{J,k}} + \delta_{J,k}[\mathbf{x}_{J,t_{J,k}} + \mathbf{s}_{\theta,J}(\mathbf{x}_{J,t_{J,k}}, t_{J,k})] + \sqrt{2\delta_{J,k}}\boldsymbol{\epsilon}_{J,k} \quad \text{with } \boldsymbol{\epsilon}_{J,k}, \mathbf{x}_{J,T} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

Starting at $j = J$, we successively sample $\bar{\mathbf{x}}_j$ conditioned on $\mathbf{x}_j$:

$$\bar{\mathbf{x}}_{j,t_{j,k-1}} = \bar{\mathbf{x}}_{j,t_{j,k}} + \delta_{j,k}[\bar{\mathbf{x}}_{j,t_{j,k}} + \mathbf{s}_{\theta,j}(\bar{\mathbf{x}}_{j,t_{j,k}}, t_{j,k}, \mathbf{x}_j)] + \sqrt{2\delta_{j,k}}\boldsymbol{\epsilon}_{j,k} \quad \text{with } \boldsymbol{\epsilon}_{j,k}, \bar{\mathbf{x}}_{j,T} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

and apply the normalized iFWT to calculate

$$\mathbf{x}_{j-1} = \gamma_j \mathbf{L}^T \mathbf{x}_j + \gamma_j \mathbf{B}^T \bar{\mathbf{x}}_j.$$

An illustration of WSGMs can be found in figure 3.1. The pseudocode for training a WSGM and sampling from it is given in algorithms 4 and 5. Using a similar reasoning as in the runtime analysis of the FWT one can show that the asymptotic complexity of WSGMs is $\mathcal{O}(NL^2)$ for $L \times L$ images.

As we have discussed, due to the whitening of the normalization and the conditioning on the low-frequency coefficients, we expect the necessary number of discretization steps to reach an error ε to be significantly smaller for WSGMs compared to a standard diffusion model. Guth et al. [GCDM22] have proved this for Gaussian processes if a constant discretization step size δ is used for all scales. In particular, they have shown that the normalization of wavelet coefficients performs a preconditioning of the covariance, whose eigenvalues then remain of the order of 1. As a consequence, the number of discretization steps necessary to reach an error ε for WSGMs is independent of the size of the generated images.

Theorem 3.2 ([GCDM22])**.** *Let $\mathbf{x}$ be a Gaussian stationary process of power spectrum $P(\omega) = c(\xi^\eta + |\omega|^\eta)^{-1}$ with $\eta, \xi > 0$. If the wavelet has a compact support, $q \geq \eta$ vanishing moments and is C^q then the first-order terms E_T and E_δ in the sampling error of WSGM $\mathrm{D}_{\mathrm{KL}}(q \parallel \tilde{q})$ are such that for any $\varepsilon > 0$, there exists $C > 0$ such that for any δ, T:*

$$(1/d)(E_T + E_\delta) \leq \varepsilon \quad \textit{and} \quad N = T/\delta \leq C\varepsilon^{-2}.$$

Remark. All wavelets of the Daubechies family with a length of $N = 2q$ have q vanishing moments. In particular, the Haar wavelet has only one vanishing moment. Since for the power spectrum of natural images we usually have $\eta = 2$ [GCDM22], the conditions are fulfilled for all wavelets of the Daubechies wavelet family except the Haar wavelet.

Remark. WSGMs can be considered as a cascaded diffusion model [Sah+22; Ho+22a] as multiple diffusion models are used in a cascaded fashion, generating increasingly larger images. However, the wavelet analogue of other cascaded diffusion model approaches is to sample from $q(\mathbf{x}_{j+1} \mid \mathbf{x}_j)$. Sampling from $q(\bar{\mathbf{x}}_j \mid \mathbf{x}_j)$ allows us to explicitely exploit the preconditioning discussed above.

Algorithm 4 WSGM training [GCDM22]

Require: Training data $\{\mathbf{x}_0^m\}_{m=1}^M$, normalization .
for $j \in \{1, \dots, J\}$ **do**
  for $m \in \{1, \dots, M\}$ **do**
    $\mathbf{x}_j^m, \bar{\mathbf{x}}_j^m \leftarrow \texttt{FWT}(\mathbf{x}_{j-1}^m)$. ▷ Wavelet transform the dataset
  end for
  Calculate normalization factor γ_j.
end for
for $j \in \{J, \dots, 1\}$ **do** ▷ Can be run in parallel
  repeat
    Draw $(\mathbf{x}_j, \bar{\mathbf{x}}_{j,0})$ from $\{(\mathbf{x}_j^m, \bar{\mathbf{x}}_j^m)\}_{m=1}^M$.
    Sample t from $[0, T]$.
    Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
    $\bar{\mathbf{x}}_{j,t} \leftarrow \mathrm{e}^{-t}\gamma_j^{-1}\bar{\mathbf{x}}_{j,0} + \sqrt{1 - \mathrm{e}^{-2t}}\boldsymbol{\epsilon}$.
    Take a gradient step on $\nabla_{\bar{\theta}_j}\|\boldsymbol{\epsilon} - \bar{\mathbf{s}}_{\bar{\theta}_j}(\bar{\mathbf{x}}_{j,t}, t \mid \gamma_j^{-1}\mathbf{x}_j)\|^2$.
  until converged
end for

Algorithm 5 WSGM sampling [GCDM22]

$x_J \leftarrow \texttt{unconditional\_sample}(T, N, \theta_J)$ ▷ Unconditional DM sampling
for $j \in \{J, \dots, 1\}$ **do**
  $\bar{\mathbf{x}}_j \leftarrow \texttt{EulerMaruyama}(T, N, \bar{\mathbf{s}}_{\bar{\theta}_j}(\cdot, \cdot \mid \mathbf{x}_j))$. ▷ Discretization (3.2)
  $\mathbf{x}_{j-1} \leftarrow \texttt{iFWT}(\gamma_j\mathbf{x}_j, \gamma_j\bar{\mathbf{x}}_j)$ ▷ Wavelet reconstruction
end for
return $\mathbf{x}_0$

3.2 Incorporating wavelets into diffusion model backbones

In the previous section we have introduced how diffusion models can be applied in wavelet space. In particular, WSGMs offer a way to embrace the multiscale structure of the wavelet transform which gives strong theoretical guarantees.

In this section, we want to discuss a different aspect of using wavelets to improve diffusion models which is often "orthogonal" to the choice of domain: the backbone model, i.e. the model used to approximate the score function. Usually, a UNet [RFB15] is used for this purpose [Ho+22a; GCDM22; Ho+22a].

While a single score-based model $\mathbf{s}_{\theta,j}$ used in a WSGM acts on noised wavelet coefficients instead of pixels, its task is still denoising as WSGMs use denoising score matching. The difference between a WSGM and a standard diffusion model is that a WSGM operates in a different structure, e.g. having multiple wavelet coefficients that correspond to one spatial location available. However, Guth et al. [GCDM22] use a vanilla conditional UNet where the three generated detail coefficients are treated as channels, similar to colors. The conditioning on the the approximation coefficients $\mathbf{x}_j$ is done by a simple concatenation with the input channels, resulting in 12 input image channels for images with three color channels. This approach does not make use of the structure of the wavelet coefficients, as the convolutional and attention operators of the network are only applied *spatially*.

As an obvious alternative, we could use 3d operators over the dimensions (F, H, W) instead of 2d operators on (H, W), where F denotes the number of wavelet coefficients and H and W the height and width of the coefficients respectively. For now, we set $F = 4$ as we consider a single-level wavelet transform. Using 3d operators would allow to model relationships between different wavelet coefficients. However, moving all operators to be fully 3d would significantly increase the number of model parameters and inference time of the model.

To mitigate this we look at video-based tasks for inspiration. While seemingly unrelated, a video consists of a sequence of images called frames that are usually spatially correlated—as are wavelet coefficients. The particular task of denoising a sequence of images simultaneously is also faced by video diffusion models [Ho+22b]. There the 3d operators are separated into a 2d spatial operator and a 1d "temporal" operator. For the convolutional operator, multiple variants of this separation have been proposed in the video context [Tra+18].

For the single-level WSGMs, Yuan et al. [Yua+23] implement these considerations in a model they dub spatial-frequency UNet. Let $\mathbf{x} \in \mathbb{R}^{B\times C_i\times F\times H\times W}$ be a batch of wavelet coefficients with a batch size of B and C_i channels. For inputs to the model, we have $C_0 = 3$. We consider a convolutional operator that first applies a spatial convolution followed by a frequency convolution. These operators were introduced as $(2{+}1)$d convolutions in the context of videos [Tra+18]. The spatial convolution uses a $1 \times k \times k_s$ kernel while a $f \times 1 \times 1$ kernel is used for the frequency convolution, with k and f being kernel sizes. The number of input channels C_i and output channels C_{i+1} are fixed as model hyperparameters. As an intermediate channel count Yuan et al. [Yua+23] use

$$\frac{fk^2C_iC_{i+1}}{k^2C_i + fC_{i+1}}$$

to get approximately the same number of parameters as a full 3d convolution. Although no computational speedup is achieved by this choice, the number of activation functions is effectively doubled and an inductive bias is introduced. Both have been found to be benefitial [Tra+18].

For the attention operators Yuan et al. [Yua+23] propose to also use a (2+1)d split by first using a spatial attention layer followed by a frequency attention layer. For the spatial attention layer, the batch is temporarilly reshaped to be of the shape $(B \cdot F, C_i, H \cdot W)$. Similarly, the frequency attention is applied to a signal of the shape $(B \cdot H \cdot W, C, F)$. A similar (2+1)d attention using a FFT instead has also been proposed by Guo et al. [Guo+23].

Applied to a diffusion model that unconditionally generates all coeffients of a single-level Haar wavelet transform, Yuan et al. [Yua+23] found the inclusion of frequency-spatial blocks into the noise predictor to be benefitial to the image quality.

4 Numerical Experiments

In this chapter we examine the application of diffusion models in wavelet coefficient space by extending WSGMs described in section 3.1. As in Guth et al. [GCDM22] our chosen exemplary application is generating portrait photographies of human faces.

In section 4.1 we discuss metrics to evaluate the quality of the generated images. Starting with the Fréchet Inception distance (FID) [Heu+17a] as the current standard metric in the field we discuss its flaws, especially for the chosen task of generated images of human faces. Motivated by an observed frequency bias of generative models we analyze metrics based on KL divergences of power spectra of wavelet packet and Fourier representations as introduced by Veeramacheneni, Wolter, and Gall [VWG23] and propose a variant addressing shortcomings.

In section 4.2 we introduce the setup of the experiments and the implementation. In the following sections we examine various ways to extend WSGMs: using higher order wavelets with different boundary handling techniques (section 4.3), choosing different numbers of inference steps per decomposition level (section 4.4) and increasing the number of decomposition levels (section 4.6).

4.1 Evaluation metrics

Quantitatively measuring the performance of generative models is a difficult but important task. Ideally, the model would generate samples indistinguishable from the data distribution while covering all its modes.

In the context of generative models for images the current de facto standard metric is the Fréchet Inception distance (FID) [Heu+17a]. Its core concept is to map a set of generated images and a set of images from the data distribution separately to some *vision-related feature space* [Kyn+23]. Multivariate Gaussians $\mathcal{P} = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\mathcal{P}_{\text{data}} = \mathcal{N}(\boldsymbol{\mu}_{\text{data}}, \boldsymbol{\Sigma}_{\text{data}})$ are then fitted to the feature space representations. The FID is then calculated as the Fréchet distance [Fré57] between both Gaussians [DL82]:

$$\mathrm{d}(\mathcal{P}, \mathcal{P}_{\text{data}})^2 = \|\mu - \mu_{\text{data}}\|_2^2 + \operatorname{tr}(\boldsymbol{\Sigma} + \boldsymbol{\Sigma}_{\text{data}} - 2\sqrt{\boldsymbol{\Sigma}\boldsymbol{\Sigma}_{\text{data}}})$$

As a feature space representation, the activations of the last pooling layer of an Inception-V3 model[Sze+16] trained on the ImageNet dataset [Den+09] are used.

FID has seen widespread adoption and has been found to correlate reasonably well with human judgement [Kyn+23; Heu+17b]. However, FID has some major shortcomings. The calculated FID is sensitive to low-level choices in implementation, e.g. rounding [VWG23], compression or resizing [PZZ22]. Additionally, several authors [Kar+20; MVB20; NMDB21; Bor22; Alf+22] observe a difference in model ranking between FID and human judgement on datasets other than ImageNet.

Kynkäänniemi et al. [Kyn+23] show the FID feature space—one linear layer away from the output layer—to be very close to the ImageNet classifications, i.e. the FID score mainly depends on the ImageNet classification of the images. This is in particular problematic for the use case of generating human face images as they do not constitute an ImageNet class. Therefore, the ImageNet classification and thus the FID of human faces might depend on background objects that resemble ImageNet classes such as a bow tie, leading to a high distance between perceptually similar images [VWG23].

Generative image models have been observed to fail in reproducing spectral distributions in particular in the high-frequency bands [DKK20; WBHG22; Fra+20; GHBC21; Rah+19]. The discussed drawbacks of the FID motivate the search for a different feature space. To address the low-frequency bias of generative models we consider the Fourier decompositon of the signal as well as a wavelet packet representation. Both allow us to directly compare energies between frequency bands. While a Fourier representation has a higher spectral resolution a wavelet packet representation also incorporates spatial information, allowing us to e.g. differentiate between the image boundaries and its center.

Following Veeramacheneni, Wolter, and Gall [VWG23] we use the KL divergence to compare the energy in the Fourier or wavelet packet representation of the real and generated images. As the KL divergence compares probability distributions, we interpret our feature space probabilisticly by normalization. For an image $\mathbf{A} \in \mathbb{R}^{C\times H\times W}$ with $C = 3$ color channels the *spatially normalized* wavelet packet energy is defined as

$$\mathcal{P}(\mathbf{A}; c, p)_{[i,j]} := \frac{\mathcal{W}(\mathbf{A})^2_{[c,p,i,j]}}{\sum_{h=1}^{F_h}\sum_{w=1}^{F_w}\mathcal{W}(\mathbf{A})^2_{[c,p,h,w]}} \tag{4.1}$$

where $\mathcal{W}$ denotes the wavelet packet transform used. F_h and F_w denote the spatial resolution of the wavelet packet; $F = 4^l$ the number of wavelet packets for a wavelet packet transform with l levels. The indices $i \in \{1, \dots, F_h\}$ and $j \in \{1, \dots, F_w\}$ specify the wavelet packet coefficient for a specific packet $p \in \{1, \dots, F\}$ and color channel $c \in \{1, \dots, C\}$.

We can interpret $\mathcal{P}(\mathbf{A}; c, p)$ as a probability distribution describing the spatial energy distribution for a given color channel c and wavelet packet p. To compare two images $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{C\times H\times W}$ we average over the KL divergences of these probability distributions:

$$\mathcal{D}_{\mathcal{W}}(\mathbf{A}, \mathbf{B}) := \frac{1}{FC}\sum_{\substack{1\le p\le F\\ 1\le c\le C}} \mathrm{D}_{\mathrm{KL}}(\mathcal{P}(\mathbf{A}; c, p) \parallel \mathcal{P}(\mathbf{B}; c, p)). \tag{4.2}$$

This divergence was introduced by Veeramacheneni, Wolter, and Gall [VWG23] as the wavelet packet power spectrum KL divergence. However, as our target is to compare image distributions instead of single images we need to extend this notion to comparing batches $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{B\times C\times H\times W}$ of B images each. Veeramacheneni, Wolter, and Gall [VWG23] propose to incorporate the batch dimension in the normalization, i.e. replacing $\mathcal{P}(\mathbf{A}; c, p)$ with

$$\mathcal{P}_{\mathrm{b}}(\mathbf{A}; c, p)_{[b,i,j]} := \frac{\mathcal{W}(\mathbf{A})^2_{[b,c,p,i,j]}}{\sum_{b=1}^{B}\sum_{h=1}^{F_h}\sum_{w=1}^{F_w}\mathcal{W}(\mathbf{A})^2_{[b,c,p,h,w]}}.$$

Denoting the proportion of total energy of image b for channel c in packet p as

$$w(\mathbf{A}; c, p)_b := \frac{\sum_{h=1}^{F_h} \sum_{w=1}^{F_w} \mathcal{W}(\mathbf{A})^2_{[b,c,p,h,w]}}{\sum_{b=1}^{B} \sum_{h=1}^{F_h} \sum_{w=1}^{F_w} \mathcal{W}(\mathbf{A})^2_{[b,c,p,h,w]}}$$

and the b-th image in $\mathbf{A}$ as $\mathbf{A}_b$ we have

$$\mathcal{P}_{\mathrm{b}}(\mathbf{A}; c, p)_{[b,i,j]} = w(\mathbf{A}; c, p)_b \mathcal{P}(\mathbf{A}_b; c, p). \tag{4.3}$$

For a specific color channel c and wavelet packet p we can thus interpret $\mathcal{P}_{\mathrm{b}}$ as sampling a random image b from the batch weighted by the proportion of total energy in the packet and then sampling a random spatial location in the packet of this image. Using (4.3) we can relate the resulting divergence to (4.2) as

$$\begin{aligned}
&\mathcal{D}_{\mathcal{W},\mathrm{b}}(\mathbf{A}, \mathbf{B}) := \frac{1}{FC} \sum_{\substack{1 \le p \le F \\ 1 \le c \le C}} \mathrm{D}_{\mathrm{KL}}(\mathcal{P}_{\mathrm{b}}(\mathbf{A}; c, p) \parallel \mathcal{P}_{\mathrm{b}}(\mathbf{B}; c, p)) \\
&= \frac{1}{FC} \sum_{\substack{1 \le p \le F \\ 1 \le c \le C}} \sum_{b=1}^{B} \sum_{i=1}^{F_h} \sum_{j=1}^{F_w} \mathcal{P}_{\mathrm{b}}(\mathbf{A}; c, p)_{[b,i,j]} \log \frac{\mathcal{P}_{\mathrm{b}}(\mathbf{A}; c, p)_{[b,i,j]}}{\mathcal{P}_{\mathrm{b}}(\mathbf{B}; c, p)_{[b,i,j]}} \\
&= \frac{1}{FC} \sum_{\substack{1 \le p \le F \\ 1 \le c \le C}} \sum_{b=1}^{B} w(\mathbf{A}; c, p)_b \sum_{i=1}^{F_h} \sum_{j=1}^{F_w} \mathcal{P}(\mathbf{A}_b; c, p)_{[i,j]} \left[\log \frac{w(\mathbf{A}; c, p)_b}{w(\mathbf{B}; c, p)_b} + \log \frac{\mathcal{P}(\mathbf{A}_b; c, p)_{[i,j]}}{\mathcal{P}(\mathbf{B}_b; c, p)_{[i,j]}} \right] \\
&= \frac{1}{FC} \sum_{\substack{1 \le p \le F \\ 1 \le c \le C}} \sum_{b=1}^{B} w(\mathbf{A}; c, p)_b \, \mathrm{D}_{\mathrm{KL}}(\mathcal{P}(\mathbf{A}_b; c, p) \parallel \mathcal{P}(\mathbf{B}_b; c, p)) \\
&\quad + \frac{1}{FC} \sum_{\substack{1 \le p \le F \\ 1 \le c \le C}} \sum_{b=1}^{B} w(\mathbf{A}; c, p)_b \log \frac{w(\mathbf{A}; c, p)_b}{w(\mathbf{B}; c, p)_b} \underbrace{\sum_{i=1}^{F_h} \sum_{j=1}^{F_w} \mathcal{P}(\mathbf{A}_b; c, p)_{[i,j]}}_{=1} \\
&= \frac{1}{FC} \sum_{\substack{1 \le p \le F \\ 1 \le c \le C}} \left[\mathrm{D}_{\mathrm{KL}}(w(\mathbf{A}; c, p) \parallel w(\mathbf{B}; c, p)) + \sum_{b=1}^{B} w(\mathbf{A}; c, p)_b \, \mathrm{D}_{\mathrm{KL}}(\mathcal{P}(\mathbf{A}_b; c, p) \parallel \mathcal{P}(\mathbf{B}_b; c, p)) \right].
\end{aligned}$$

We note that $\mathcal{D}_{\mathcal{W},\mathrm{b}}$ consists of an average over the sum of two parts: the first is the KL divergence of the total energy proportion between $\mathbf{A}$ and $\mathbf{B}$, the second the expected KL divergence of the spatially normalized wavelet packet energies of packets from $\mathbf{A}$ to its correspondents from $\mathbf{B}$ if images are sampled according to the proportion of the total energy in the packet. If all packets in $\mathbf{A}$ and $\mathbf{B}$ are normalized to the same total energy, $\mathcal{D}_{\mathcal{W},\mathrm{b}}$ calculates the average pairwise KL divergence

$$\frac{1}{FC} \sum_{\substack{1 \le p \le F \\ 1 \le c \le C}} \frac{1}{B} \sum_{b=1}^{B} \mathrm{D}_{\mathrm{KL}}(\mathcal{P}(\mathbf{A}_b; c, p) \parallel \mathcal{P}(\mathbf{B}_b; c, p))$$

which can be interpreted as an average over the expected pairwise KL divergence between the spatially normalized wavelet packet energies.

However, the following lemma shows that the expected pairwise KL divergence itself is not a statistical divergence. This is because a distribution compared with itself has a non-zero value if the spatially normalized wavelet packet energies are not constant between images:

Lemma 4.1. *Let $\mathcal{D}$ be a divergence, $S = \{\mathcal{P}_x \mid x \in X\}$ a familiy of probability measures and q_1, q_2 probability distributions over X. Let*

$$\mathcal{D}'(q_1, q_2) := \mathbb{E}_{x\sim q_1, y\sim q_2}\big[\mathcal{D}(\mathcal{P}_x, \mathcal{P}_y)\big]$$

be the expected pairwise divergence. We have

$$\mathcal{D}'(q_1, q_1) = 0 \iff \mathcal{P}_x = \mathcal{P}_y \text{ a.s. for } x, y \sim q_1 \text{ iid.}$$

Proof. As $\mathcal{D}$ is a divergence, it is non-negative and $\mathcal{D}(\mathcal{P}_x, \mathcal{P}_y) = 0$ if and only if $\mathcal{P}_x = \mathcal{P}_y$. □

This is not just a theoretical problem, e.g. comparing a batch of real images to a batch of generated images might lead to a smaller score than comparing a batch of real images to the same batch but with a different image order (cf. table 4.1).

Furthermore, both components of $\mathcal{D}_{\mathcal{W},\mathrm{b}}$ are dependent on the order of images. In particular, comparing a batch of images to a reordered batch of the same images might lead to a high divergence if aligned image pairs have high pairwise divergences or if the reordering significantly changes the total energy distribution.

As an alternative we propose to incorporate the batch dimension into (4.1) by averaging the spatially normalized wavelet packet energy for all images, yielding the probability distribution

$$\mathcal{P}_{\text{spat}}(\mathbf{A}; c, p)_{[i,j]} := \frac{1}{B}\sum_{b=1}^{B} \mathcal{P}(\mathbf{A}_b; c, p)_{[i,j]} \tag{4.4}$$

which can be interpreted as an estimation of the expected spatially normalized wavelet packet energy. The corresponding divergence

$$\mathcal{D}_{\text{spat}}(\mathbf{A}, \mathbf{B}) := \frac{1}{FC}\sum_{\substack{1\le p\le F\\ 1\le c\le C}} \mathrm{D}_{\mathrm{KL}}(\mathcal{P}_{\text{spat}}(\mathbf{A}; c, p) \parallel \mathcal{P}_{\text{spat}}(\mathbf{B}; c, p)). \tag{4.5}$$

is a measure how close the spatial energy distribution of the synthetic images matches the real data's distribution accross all frequency bands. While $\mathcal{D}_{\mathcal{W},\mathrm{b}}$ estimates the expected pairwise KL divergence, $\mathcal{D}_{\text{spat}}$ estimates the KL divergence between expected probabilities. In the setting of lemma 4.1, $\mathcal{D}_{\text{spat}}$ approximates $\mathcal{D}(\mathbb{E}_{x\sim q_1}[\mathcal{P}_x], \mathbb{E}_{y\sim q_2}[\mathcal{P}_y])$. It is independent of the order of images in the batch and a statistical divergence.

We stress that both $\mathcal{D}_{\mathcal{W},\mathrm{b}}$ and $\mathcal{D}_{\text{spat}}$ are measures for the approximation of the spatial energy distribution on each frequency band but not for the energy distibution between frequency bands as each frequency band is normalized separately. To address this we construct a Fourier-based divergence in a similar fashion, using the absolute value of the Fourier transform instead of the wavelet packet transform and removing all indexing over

Table 4.1: Results of different scores comparing 30k images from the CelebA dataset to the same 30k images but in a different order as well as 30k images generated from a 2-level WSGM using Haar wavelets and 100 inference steps on each level. The mean and standard deviation are reported for 5 runs with a different order of real images each.[1]

	$\mathcal{D}^{\text{symm}}_{\mathcal{W},\text{b}}$	$\mathcal{D}^{\text{symm}}_{\mathcal{W},\text{bf}}$	$\mathcal{D}^{\text{symm}}_{\mathcal{F},\text{b}}$
CelebA-reordered	4.731 ± 0.004	1.570 ± 0.003	0.311 ± 0.0004
Haar WSGM	4.335 ± 0.004	1.234 ± 0.001	0.278 ± 0.0008

the wavelet packet dimension. We denote the Fourier variant of $\mathcal{D}_{\mathcal{W},\text{b}}$ as $\mathcal{D}_{\mathcal{F},\text{b}}$ and the variant of $\mathcal{D}_{\text{spat}}$ as $\mathcal{D}_{\text{freq}}$.

However, the Fourier variants *only* capture how the energy levels of frequent bands match, spatial distribution is not regarded. To address both, we adapt $\mathcal{P}$ by extending the state space to include the wavelet packet:

$$\mathcal{P}_{\text{f}}(\mathbf{A}; c)_{[p,i,j]} := \frac{\mathcal{W}(\mathbf{A})^2_{[c,p,i,j]}}{\sum_{f=1}^{F}\sum_{h=1}^{H}\sum_{w=1}^{W}\mathcal{W}(\mathbf{A})^2_{[c,f,h,w]}}. \tag{4.6}$$

We repeat the constructions above using $\mathcal{P}_{\text{f}}$ instead of $\mathcal{P}$ yielding

$$\mathcal{D}_{\text{spat-freq}}(\mathbf{A}, \mathbf{B}) := \frac{1}{C}\sum_{c=1}^{C} \mathrm{D}_{\mathrm{KL}}\left(\frac{1}{B}\sum_{b=1}^{B}\mathcal{P}_{\text{f}}(\mathbf{A}_b; c) \,\middle\|\, \frac{1}{B}\sum_{b=1}^{B}\mathcal{P}_{\text{f}}(\mathbf{B}_b; c)\right).$$

We denote the variant of $\mathcal{D}_{\mathcal{W},\text{b}}$ which uses $\mathcal{P}_{\text{f}}$ as $\mathcal{D}_{\mathcal{W},\text{bf}}$.

As the KL divergence is not symmetric, we define the symmetrized variant of each introduced divergence as

$$\mathcal{D}^{\text{symm}}(\mathbf{A}, \mathbf{B}) := \frac{\mathcal{D}(\mathbf{A}, \mathbf{B}) + \mathcal{D}(\mathbf{B}, \mathbf{A})}{2}.$$

4.2 Setup

As we extend WSGMs we follow Guth et al. [GCDM22] closely for our setup to allow comparisons. The task is to generate images from the distribution of the CelebA dataset [LLWT15], depicting faces of celebrities that are centered and aligned by their eye position. The images are cropped and resized to a resolution of 128×128, following [WBHG22]. We randomly split the dataset into 150k training samples, 30k test samples and 20k validation samples.

[1] Note that the range of values shown for $\mathcal{D}^{\text{symm}}_{\mathcal{W},\text{b}}$ does not match the ranges reported in Veeramacheneni, Wolter, and Gall [VWG23] due to an error in the implementation, as confirmed in private discussion with one of the authors.

Separable 2D Wavelet transforms are applied, separately for each color channel. For each level and color channel, the wavelet coefficients are normalized to have zero mean and unit variance.

As Guth et al. [GCDM22] we base our model on a discretization of the (VP SDE) and optimize the denoising loss L^{simple} but differ in the choice of the noise schedule. Guth et al. [GCDM22] use an exponential noise schedule with quadratic timestep sampling for training and linear timestep sampling for inference, i.e.

$$T_{\text{train}} = \left\{ \left(\frac{k}{N}\right)^2 T \;\middle|\; k = 1, \dots, N \right\}, \quad T_{\text{inference}} = \left\{ \frac{k}{N} T \;\middle|\; k = 1, \dots, N \right\} \quad \text{for } N \text{ steps.}$$

As seen in section 2.3 the former corresponds approximately to the linear noise schedule as used by Ho, Jain, and Abbeel [HJA20]. We choose to use the computationally more efficient cosine noise schedule (cf. section 2.3) and DDPM reduced step sampling with $\sigma_t^2 = \tilde{\beta}_t$ as defined in (2.12) [ND21].

Evaluation To evaluate the performance of the trained generative models quantitatively we report the FID. Additionally, we report $\mathcal{D}^{\text{symm}}_{\text{spat}}$, $\mathcal{D}^{\text{symm}}_{\text{spat-freq}}$, $\mathcal{D}^{\text{symm}}_{\text{freq}}$ introduced in section 4.1 as metrics how the real and synthetic distributions match spatially across frequency bands and spectrally. Following Veeramacheneni, Wolter, and Gall [VWG23] we use a *sym5* wavelet packet transform using 3 levels with a reflecting extension for boundary handling.

Architecture Following Guth et al. [GCDM22], the architecture of our noise estimator is a UNet [RFB15] with 3 residual blocks at each scale and group normalization[WH18]. The number of scales and the channel size for each scale depends on the input resolution, cf. table 4.2. The input resolution is halved in between layers; scales corresponding to an unpadded resolution of 16×16 or 8×8 include attention layers [Vas+17]. The input and output channel count for unconditional models is 3 each. For conditional models, the output channel count is 9 corresponding to the three color channels per detail coefficient. Conditioning on the low frequencies is done with a simple input concatenation along channels, i.e. the input channel count is 12. As in Ho, Jain, and Abbeel [HJA20] we use a sinusoidal positional encoding [Vas+17] to specify the diffusion time t.

Training was performed on four graphics processing units (GPUs) of type A100 or V100. Inference times are reported for a single A100 GPU. All models were trained for 500k optimization steps with a batch size of 64 for spatial resolutions of at least 64×64 and 128 below. We use the Adam optimizer [KB15] with a fixed learning rate of 10^{-4} and no weight decay. As in Ho, Jain, and Abbeel [HJA20] we used exponential moving average (EMA) [KB15] on model parameters with a decay factor of 0.9999.

Implementation We implemented our model in a *PyTorch* [Pas+19] based implementation using the *diffusers* [Pla+22] framework. To accelerate training, we made use of the *accelerate* library [Gug+22] to train distributedly on multiple GPUs and applied mixed precision training [Nar+17]. For the calculation of wavelet transforms we used the *PyTorch Wavelet Toolbox* [WBGH24]. Our implementation is available at `github.com/felixblanke/wavelet-dm`.

Table 4.2: Parametrization of the UNet used in the experiments. With a base channel count of $C = 128$, the number of channels at scale k equals $a_k C$ with multipliers (a_k) depending on the wavelet decomposition level. The input resolution depends on the mode of boundary handling: "unpadded" applies to Gram-Schmidt boundary wavelets or Haar wavelets, "padded" is calculated for wavelets of length 8, e.g. *db4* or *sym4*, using a boundary extension technique.

		resolution	
Level	channel size multipliers (a_k)	unpadded	padded
1	(2, 2, 4, 4)	64×64	67×67
2	(4, 4)	32×32	37×37
3	(4, 4)	16×16	22×22
0 (uncond.)	(1, 2, 2, 4, 4)	128×128	–
2 (uncond.)	(1, 2, 2, 2)	32×32	37×37
3 (uncond.)	(2, 4, 4)	16×16	22×22

4.3 Higher-order wavelets and boundary handling

The use of wavelets in the context of generative models for images is not a new concept. However, most existing work is based on Haar wavelets[GHBC21; Li+23; Liu+20; PDT23; Wan+20; Yoo+19; Zha+22]. The two main reasons given for this choice are the simplicity of the Haar wavelet and the lack of boundary effects due to their short support. Overcoming problems with boundary handling might enable us to make use of higher order wavelets and their better approximation accuracy (cf. lemma 1.32 and [SN96]).

Guth et al. [GCDM22] introduced the WSGM and gave theoretical results that also applied to wavelets of higher order, but only tested their model on images with Haar wavelets. This motivates us to extend the setup to wavelets of higher order.

We select the *db4* wavelet with zero padding and extension by reflection as well as the *sym4* Gram-Schmidt boundary wavelets. Additionally, we also include a model using Haar wavelets and a model that unconditionally generates images of full 128×128 resolution without any wavelet transforms as a baseline. To choose these pairings, we ran a subset of the experiments on all combinations and selected for each boundary technique the better performing wavelet. We note however that we observed no significant difference between the *db4* and *sym4* wavelet in this regard.

All wavelet transforms used are separable 2D transforms with two levels, i.e. low frequency coefficients of size 32×32 are generated unconditionally; normalized detail coefficients of size 32×32 and 64×64 are generated conditioned on the low frequency coefficients of the respective size.

Examplary images sampled using *sym4* boundary wavelets as well as the baseline are shown in figure 4.1. Further images are depicted in appendix 1.

As a first experiment we evaluate all models using the same number of discretization steps for each level. We report the FID and different power spectrum based metrics as introduced in section 4.1. The resulting curves are depicted in figure 4.2. We observe

that the FID curves for our models follow the same form and lie very close to each other. Comparing to the cited results from Guth et al. [GCDM22] we note the missing distinct gap between our baseline model and the WSGMs. Our baseline model performs significantly better than the baseline from Guth et al. [GCDM22]. Additionally, we note that our models plateau around a FID value of 10 compared to 20 reported by Guth et al. [GCDM22] and roughly an order of magnitude later at approximately 250 steps. We conjecture that these observations might be caused by differences in the used backbone architectures. For the curves of the $\mathcal{D}^{\text{symm}}_{\text{spat}}$ metric measuring the divergences of the spatial distribution of wavelet packet coefficients, we note all curves mathching except the *db4* model using zero padding. We will address this case later in detail.

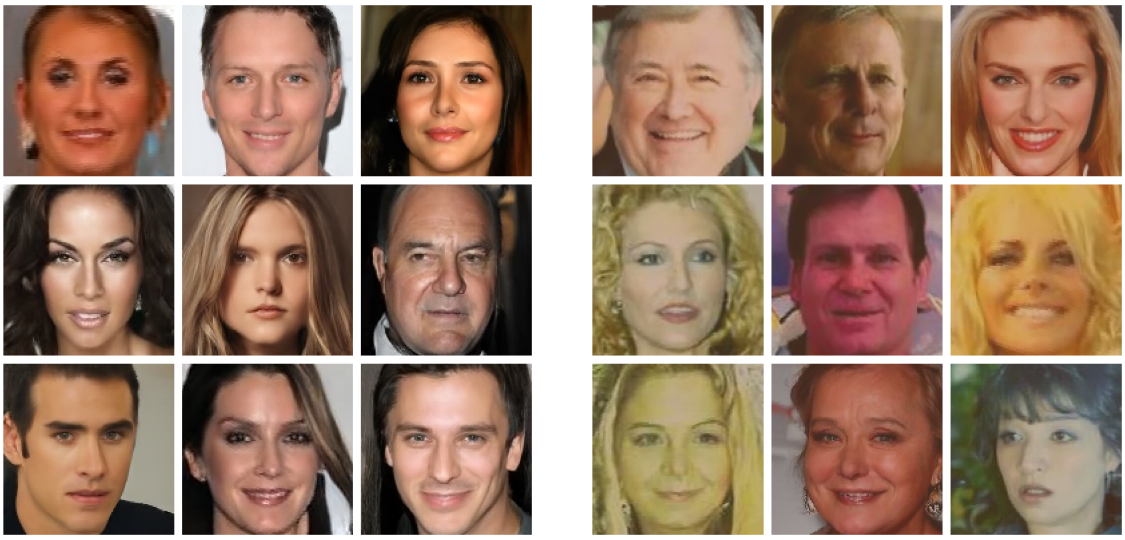
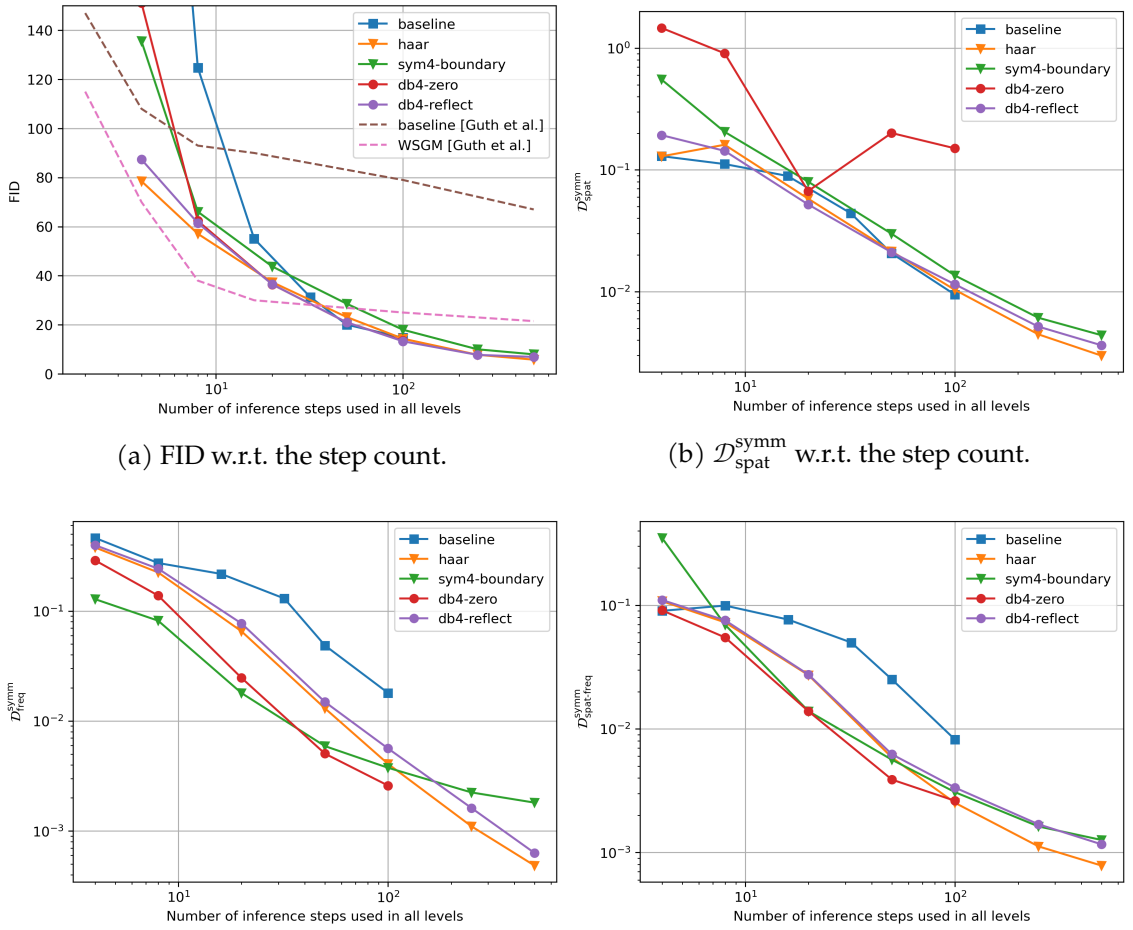


Figure 4.1: Generated samples for 2-level *sym4* wavelet using Gram-Schmidt boundary wavelets (left) and the baseline model trained in pixel space (right). All models were sampled with 100 inference steps.

For both power spectrum based metrics that take the frequency distributions into account the curves are not overlayed as in the spatial curve hinting at differences in the spectral distribution of the generated samples. In both metrics we report larger values for the baseline model showing a reduction of the frequency bias for wavelet-based models.

Between the wavelet models we observe the same behaviour under the spatio-frequency metric. However, for smaller timesteps the boundary wavelet and zero padding models seem to better approximate the energy distribution between frequency bands as their Fourier-based metric is smaller.

Coming back to *db4* with zero padding, we note seemingly contradicting scores. For 100 steps it performs worst in the purely spatial metric by more than an order of magnitude while leading the combined metric $\mathcal{D}^{\text{symm}}_{\text{spat-freq}}$. To examine this further, we split the combined metric into a low-frequency part using only the packets corresponding to the lower half of the frequency bands and a high-frequency part using the other half of packets. Thus, the low-frequency component measures the divergence of energy distributions accross



(a) FID w.r.t. the step count.

(b) $\mathcal{D}^{\text{symm}}_{\text{spat}}$ w.r.t. the step count.

(c) $\mathcal{D}^{\text{symm}}_{\text{freq}}$ w.r.t. the step count.

(d) $\mathcal{D}^{\text{symm}}_{\text{spat-freq}}$ w.r.t. the step count.

Figure 4.2: Comparison of two-scale WSGMs for different wavelets and boundary handling techniques. All models use two scales and are sampled using the same number of discretization steps across scales. We plot the influence of the number of steps to different metrics: the FID (a) and power spectrum-based metric comparing energy distributions of wavelet packets spatially (b), Fourier coefficient (c) and wavelet packets accross frequency bands and spatially (d). Markers indicate the type of wavelet transform used: padded (circle), unpadded (triangle) and a standard DDPM model (square).
The dashed lines in (a) are taken from Guth et al. [GCDM22, Figure 4] by eye as a baseline for a differing implementation of a standard score-based model ("baseline") and their WSGM implementation with Haar wavelets.
No metrics are reported for samples from the baseline model for more than 100 steps as the model failed to sample from the data distribution, showing only noisy shapes not resembling human faces.

the lower half of the frequency spectrum and accross the full spatial dimensions, and vice versa. Both values as well as the numerical values for the FID and Fourier-based power spectrum metric are reported in table 4.3. We notice a remarkably high value of 297.7 for the high-frequency combined metric for the *db4* wavelet with zero padding with 100 steps while the corresponding low-frequency value is two orders smaller with 2.0. Calculating the mean packet energy for both halves of the frequency spectrum reveals a large gap in energy levels, with the value of the lower frequency half being over 350 times larges. This explains how the combined metric for the full frequency band with 2.6 is so close to the lower half's value.

Table 4.3: Comparison of WSGMs for different wavelets and boundary handling techniques, using two and three levels and a fixed number of discretization steps. We report the spatio-frequency wavelet packet power spectrum metric on the packets of the lower half ("lo") and of the upper half ("hi") of spectrum, the Fourier-baset power spectrum-based metric and the FID. For each metric, we compare the results for 50 respectively 100 discretization steps on all levels of the model. "baseline" denotes a standard DDPM. We use "bd" as a shorthand for Gram-Schmidt boundary filters.

		$\mathcal{D}^{\text{symm}}_{\text{spat-freq}}$ lo $[\cdot 10^3]$		$\mathcal{D}^{\text{symm}}_{\text{spat-freq}}$ hi $[\cdot 10^3]$		$\mathcal{D}^{\text{symm}}_{\text{freq}}$ $[\cdot 10^3]$		FID	
Wavelet	# Level	50st.	100st.	50st.	100st.	50st.	100st.	50st.	100st.
baseline	–	23.3	7.2	75.7	38.1	48.6	18.0	20.1	14.7
Haar	2	4.5	*1.8*	64.6	28.9	13.0	4.1	23.2	14.3
db4-reflect	2	4.7	2.4	79.2	39.8	14.9	5.6	21.0	*13.3*
db4-zero	2	*3.0*	2.0	392.8	297.7	*5.0*	*2.6*	21.0	13.5
sym4-bd	2	4.3	2.3	*60.1*	*26.5*	5.9	3.8	28.6	18.0
Haar	3	**2.9**	**1.6**	**21.1**	**13.7**	3.9	**1.4**	18.2	**11.0**
db4-zero	3	4.3	2.5	31.5	23.2	**3.0**	1.8	**17.2**	11.9
sym4-bd	3	4.6	2.4	43.8	20.3	4.1	2.2	29.5	17.9

To explain the high metric value for the upper frequencies we visually inspect samples. A representative image is depicted in figure 4.3. We notice a black border surrounding the image as well darker edges parallel to the left and top image boundary. We suspect that these artefacts are caused by the boundary handling technique. To check this we crop the sampled images to the center 100×100 square, removing 14 pixels from each boundary, and recalcuate all metrics. The results are reported in table 4.4. On the image centers *db4* with zero padding approximates the packet coefficients for the higher-frequencies well performing best under all 2-level models. This confirms boundary effects being the cause.

Comparing the other values in table 4.3, we note that the WSGMs fare significantly better under the power-spectrum based metrics while achieving comparable yet slightly better results for the FID. Comparing between wavelets shows no clear best 2-level model.

Table 4.4: Comparison of WSGMs for different wavelets and boundary handling techniques, using two and three levels and a fixed number of discretization steps. All metrics are calculated on the center 100×100 pixels of the generated images, cropping 14 pixels from each boundary. This is indicated by the suffix "-c14". We report the spatio-frequency wavelet packet power spectrum metric on the packets of the lower half ("lo") and of the upper half ("hi") of spectrum, the Fourier-baset power spectrum-based metric and the FID. For each metric, we compare the results for 50 respectively 100 discretization steps on all levels of the model. "baseline" denotes a standard DDPM. We use "bd" as a shorthand for Gram-Schmidt boundary filters.

		$\mathcal{D}^{\text{symm}}_{\text{spat-freq}}$ lo $[\cdot 10^3]$		$\mathcal{D}^{\text{symm}}_{\text{spat-freq}}$ hi $[\cdot 10^3]$		$\mathcal{D}^{\text{symm}}_{\text{freq}}$ $[\cdot 10^3]$		FID	
Wavelet	# Level	50st.	100st.	50st.	100st.	50st.	100st.	50st.	100st.
baseline-c14	–	26.0	6.9	74.9	38.0	28.8	10.1	16.9	13.0
Haar-c14	2	4.1	*1.8*	64.4	29.9	10.5	3.8	21.4	13.2
db4-rf-c14	2	4.5	2.3	82.4	41.8	11.7	5.1	19.6	11.9
db4-zr-c14	2	*3.3*	2.0	*30.8*	*21.8*	*5.2*	*3.1*	18.9	11.4
sym4-bd-c14	2	4.3	2.2	65.6	30.7	5.8	3.7	28.6	17.8
Haar-c14	3	**2.9**	**1.6**	**21.7**	**13.2**	3.8	**1.5**	17.9	10.9
db4-zr-c14	3	3.7	1.9	23.2	13.9	**2.9**	1.7	16.3	11.0
sym4-bd-c14	3	4.1	2.0	48.5	23.5	3.6	2.0	29.6	17.4

4.4 A finer discretization of the unconditional model

Guth et al. [GCDM22] theoretically derived that preconditioning allows us to use fewer sampling steps while maintaining the same error (cf. section 3.1.2). Conditionally generating normalized wavelet detail coefficients acts as a whitening and thus as a preconditioner. For the unconditional model in a WSGM that generates the approximation coefficients on the largest scale $J \approx \log_2 L$ for $L \times L$ images, Guth et al. [GCDM22] argue that the unconditional distrubution should be close to Gaussian. However, this assumption fails in practice as we stop the wavelet decomposition early to avoid boundary effects dominating the full signal. Additionally, to reasonably use a deep convolutional neural network (CNN) as a backbone we cannot make the spatial dimensions arbitrarily small. Guth et al. [GCDM22] observe the second level Haar approximation coefficients with a 32×32 resolution to also follow a power law decay of their spectrum, confirming that the distribution is in fact *not* close to Gaussian. Therefore, the theoretical results derived in section 3.1.2 do not apply to the unconditional model.

We are thus motivated to increase the number of discretization steps selectively for the unconditional model, testing 100 and 500 steps. We vary the number of inference steps on the other scales. The results for Haar wavelets and *sym4* wavelets with boundary filters can be found in table 4.6. We report the low- and high-frequency variant of $\mathcal{D}^{\text{symm}}_{\text{spat-freq}}$ as introduced in the previous section. Furthermore, we compute all metrics on images

Figure 4.3: Examplary image generated by an 2-scale WSGM using *db4* wavelets with zero padding for 100 steps on each level.

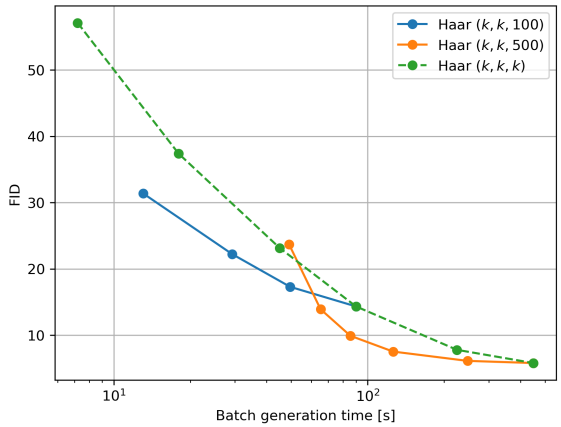
cropped to their 100×100 pixel center to control for boundary artefacts.

We find a monotonous improvement with an increasing step count accross all experiments. As we would expect, varying the discretization on the detail coefficients and with it the approximation error mainly influences the quality of high-frequency packets. Nevertheless, we also observe an improvement for the low-frequency packets, especially if the step count is low.

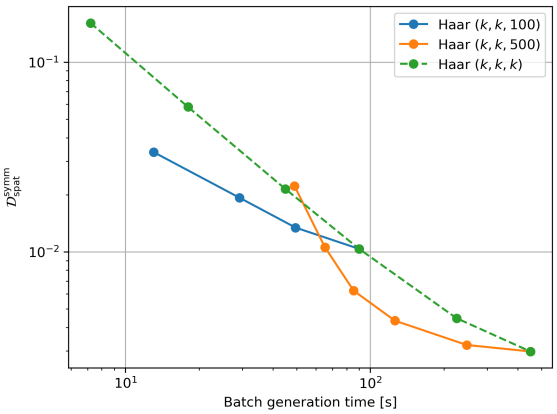
For 5 diffusion steps on the *sym4* wavelet we note a high spike in the metric for low- and high-frequency packets. Comparing to the cropped images reveals in the low frequencies the spike to be caused by boundary effects; for the high frequencies we also see a noticable drop on the cropped images. For Haar wavelets, this spike is only apparent in the high frequencies, suggesting that 5 steps simply are not sufficient for a descent approximation. For larger step counts we observe no significant improvements by cropping, making further boundary artefacts unlikely.

To fairly plot metrics of the different models we calculate the batch generation time, i.e. the average time spent to generate a single batch of 250 images on an A100 GPU. The calculated times are reported in table 4.5. The resulting plots for the Haar wavelet are shown in figure 4.4; the plot for the *sym4* wavelet can be found in the appendix in figure 3.

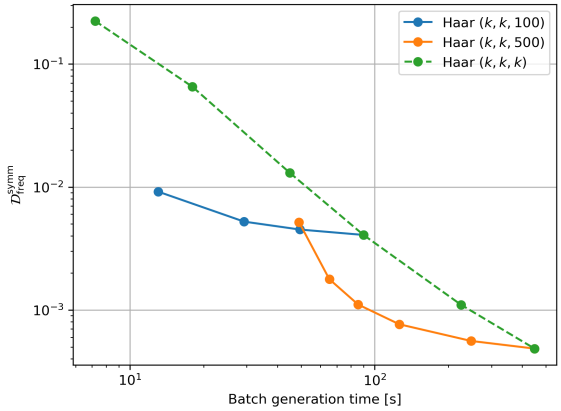
We have already found the use of the smallest step count to be insufficient, which is supported by the plots. For all other step counts, the curves run below the reference curve. This indicates a better error to runtime ratio throughout. To highlight the possible improvements we note the Haar wavelet run using 500 base model steps and 50 steps for the conditional models. This total runtime is comparable to using 100 steps on all scales. Compared to this 100-step reference curve we see a drop in FID from approximately 15 to 10 and in $\mathcal{D}_{\text{freq}}^{\text{symm}}$ by half an order of magnitude while also seeing some improvements for the spatial and combined frequency-spatial packet energy distribution.



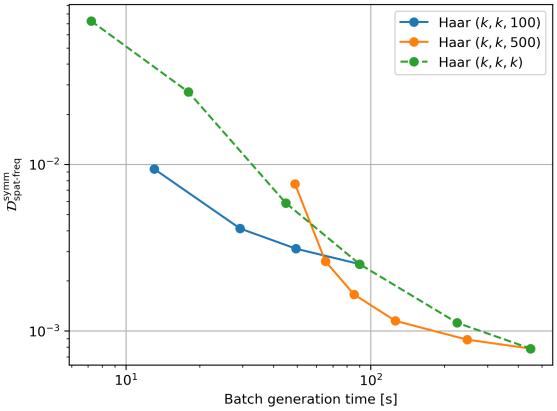
(a) FID w.r.t. the step count.



(b) $\mathcal{D}^{\text{symm}}_{\text{spat}}$ w.r.t. the step count.



(c) $\mathcal{D}^{\text{symm}}_{\text{freq}}$ w.r.t. the step count.



(d) $\mathcal{D}^{\text{symm}}_{\text{spat-freq}}$ w.r.t. the step count.

Figure 4.4: Comparison of two-scale Haar wavelet WSGMs for a different number of discretization steps between the unconditional base model and the conditional models. We plot the influence of the number of sampling steps for the conditional models for 100 and for 500 steps in the unconditional model using: the FID (a) and power spectrum-based metric comparing energy distributions of wavelet packets spatially (b), Fourier coefficient (c) and wavelet packets accross frequency bands and spatially (d). As a baseline we add a two-scale Haar WSGM with the same number of steps on all scales.

Table 4.5: Average inference time for a single inference step with a batch size of 250. The padded sizes are given for a wavelet with a filter length of 8 like *db4* or *sym4*.

	level	size $[C \times H \times W]$	sample time [s / batch]
Wavelet unpadded	1	$9 \times 64 \times 64$	0.51
Wavelet unpadded	2	$9 \times 32 \times 32$	0.30
Wavelet unpadded	2 (uncond.)	$3 \times 32 \times 32$	0.09
Wavelet unpadded	3	$9 \times 16 \times 16$	0.12
Wavelet unpadded	3 (uncond.)	$3 \times 16 \times 16$	0.07
Wavelet padded	1	$9 \times 67 \times 67$	0.62
Wavelet padded	2	$9 \times 37 \times 37$	0.46
Wavelet padded	2 (uncond.)	$3 \times 37 \times 37$	0.14
Pixel baseline	–	$3 \times 128 \times 128$	1.20

4.5 Progressively decreasing the number of inference steps per level

In the experiments we considered in the previous section, we used a different number of sampling steps between the unconditional and the conditional models, but kept the number constant for all conditional models. In particular, the step count for the unconditional model was kept fixed. In this section we vary the number of steps for all levels, calculating the count as a multiple of the next level. That means that for a scaling factor of α and k steps on the 64×64 conditional model, we do αk steps on the 32×32 conditional model and $\alpha^2 k$ on the unconditional model. To ensure a uniform discretization we choose the step count of the 64×64 model such that k, αk and αk^2 are divisors of $T = 1000$.

This setup is motivated by two observations. First, the batch generation time scales with the spatial resolution of the coefficients. Thus, steps for the 64×64 model are significantly more expensive than for the other models. Second, Ho et al. [Ho+22a] observe an error propagation across models for cascaded diffusion models. We address this by increasing the approximation accuracy for earlier models.

The results for $\alpha \in \{2, 2.5, 5\}$ are plotted in figure 4.5 for the Haar wavelets and for the *sym4* boundary wavelets in figure 4 in the appendix.

For the Haar wavelets, we note that the curves are all below the reference curve, showing a higher efficiency of the progressive step counts. Furthermore, in all four reported metrics all curves run approximately parallel to the reference curve while being offset. This suggests that the progressive step schedule has a similar approximation order as the reference curve. Remarkably, all progressive step are close to parallel showing only small deviations in the parts where their support overlaps, showing no difference between $\alpha = 2$ and $\alpha = 5$.

The *sym4* wavelets paint a similar picture although less pronounced for the FID and the Fourier-based power spectrum metric. However, in both power spectrum metrics that consider spatial distributions we see sudden jumps for smaller step sizes. We speculate that this is caused by the introduction of boundary artefacts for small step sizes, as we

have seen in previous experiments.

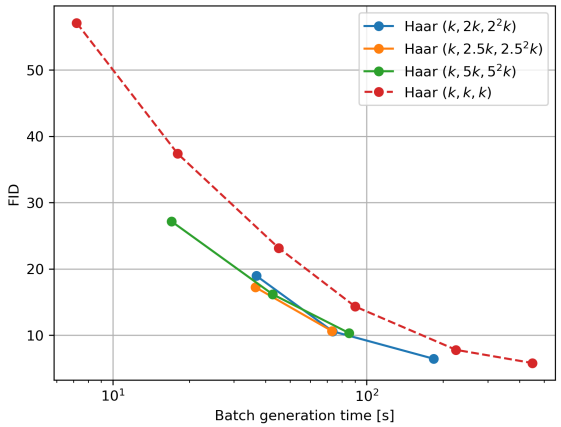
4.6 Increasing the number of wavelet levels

So far, all experiments used WSGMs with two scales which consist of an unconditional model generating the approximation coefficients and two conditional models generating the detail coefficients for both scales. As discussed in section 4.4 the distribution of the approximation coefficients is not close to Gaussian since we stopped the FWT early. But we expect it to get successively closer to Gaussian for an increasing number of scales as it is further averaged. The added detail coefficients in turn are by construction always close to Gaussian (cf. section 3.1.2). With respect to the condition, adding more scales should thus always be benefitial.

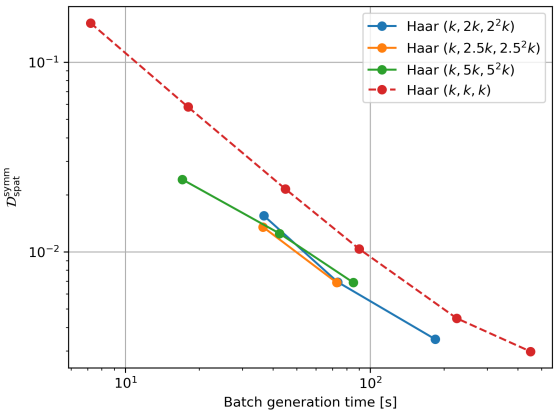
However, we have to balance the number of scales with the boundary effects and the design of the noise prediction backbone. As boundary filters have a length of $N - 2$, incrementing the number of scales by one for *db4* or *sym4* wavelets would leave only the central 4×4 coefficients independent of boundary filters. So adding more than one level is infeasible for these wavelets.

We test the addition of a single scale, i.e. evaluating WSGMs with three scales. The results are reported in tables 4.3 and 4.4. We note significant improvements for the energy distributions between frequency bands and in the high-frequencies combined metric with reductions by more than a factor of 2 for Haar wavelets. The FID and low-frequency metric are also improved although more modestly. Especially for the Haar wavelet, which is not affected by the boundary problematic, we see clear improvement for all metrics.

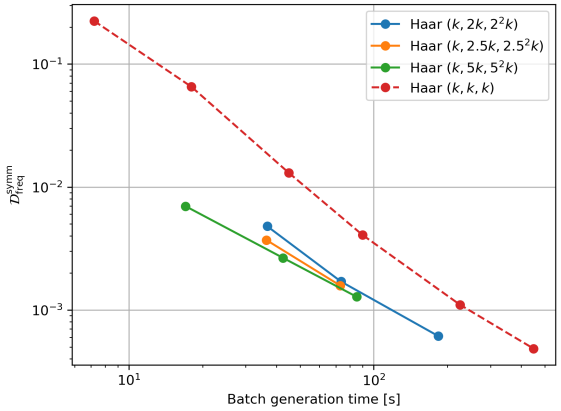
We further note no boundary artefacts for the *db4* wavelet with zero padding.



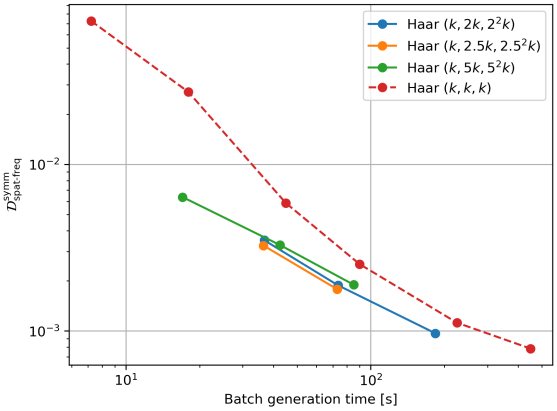
(a) FID w.r.t. the step count.



(b) $\mathcal{D}^{\text{symm}}_{\text{spat}}$ w.r.t. the step count.



(c) $\mathcal{D}^{\text{symm}}_{\text{freq}}$ w.r.t. the step count.



(d) $\mathcal{D}^{\text{symm}}_{\text{spat-freq}}$ w.r.t. the step count.

Figure 4.5: Comparison of two-scale Haar wavelet WSGMs for an increasing number of discretization steps between models. The number of steps is chosen as the step count on the previous level multiplied by a scaling factor α. We plot the influence of the step count for $\alpha \in \{2, 2.5, 5\}$ using: the FID (a) and power spectrum-based metric comparing energy distributions of wavelet packets spatially (b), Fourier coefficient (c) and wavelet packets accross frequency bands and spatially (d). As a baseline we add a two-scale Haar WSGM with the same number of steps on all scales.

Table 4.6: Comparison of WSGMs with a different step count for the unconditional and conditional models. We show results for Haar wavelets and *sym4* wavelets using boundary filters, each using two levels. We report the spatio-frequency wavelet packet power spectrum metric on the packets of the lower half ("lo") and of the upper half ("hi") of spectrum, evaluated on the full image as well as a 100×100 pixel center crop ("crop-14"). We conclude the experiment for 100 steps (a) and 500 steps (b) in the baseline model.

(a) Results for $(k, k, 100)$ steps.

	$\mathcal{D}^{\text{symm}}_{\text{spat-freq}}$ lo $[\cdot 10^3]$				$\mathcal{D}^{\text{symm}}_{\text{spat-freq}}$ hi $[\cdot 10^3]$			
	no crop		crop-14		no crop		crop-14	
k	Haar	*sym4*-bd.	Haar	*sym4*-bd.	Haar	*sym4*-bd.	Haar	*sym4*-bd.
5	6.3	172.4	5.1	4.7	193.4	565.5	181.9	254.2
25	2.7	3.6	2.5	2.8	62.3	70.8	63.0	74.7
50	2.1	2.6	2.1	2.4	37.9	40.6	38.8	45.3
100	1.8	2.3	1.8	2.2	28.9	26.5	29.9	30.7

(b) Results for $(k, k, 500)$ steps.

	$\mathcal{D}^{\text{symm}}_{\text{spat-freq}}$ lo $[\cdot 10^3]$				$\mathcal{D}^{\text{symm}}_{\text{spat-freq}}$ hi $[\cdot 10^3]$			
	no crop		crop-14		no crop		crop-14	
k	Haar	*sym4*-bd.	Haar	*sym4*-bd.	Haar	*sym4*-bd.	Haar	*sym4*-bd.
5	4.9	183.2	3.7	3.7	524.0	162.0	152.1	243.6
25	1.5	2.6	1.4	1.7	52.9	40.5	41.9	55.6
50	1.0	1.5	1.0	1.3	24.7	21.2	22.7	28.6
100	0.8	1.3	0.8	1.1	14.3	12.8	14.0	17.1
250	0.7	1.1	0.7	0.9	9.3	8.3	9.0	11.3
500	0.6	1.0	0.6	0.9	7.9	7.6	8.1	9.4

5 Discussion and outlook

In this work we have discussed uses of wavelet transforms in diffusion models. For this matter we first introduced foundations of wavelet theory on infinite signals and possible boundary handling techniques to allow applications to finite signals.

Then, we introduced DDPMs which are discrete-time diffusion models and derived different interpretations for the optimization goal. Starting from approximating the forward process posterior mean and equivalently predicting the added noise, we showed a connection to score-based models through denoising score matching. Taking the limit of the time horizon, we have seen how a DDPM can be seen as a discretization of an SDE. Conclusively, we have introduced an SDE-based framework for diffusion models in continuous-time. We closed the chapter by discussing the noise schedule used in diffusion models, observing the equivalence between two noise schedules used in practice.

We then focussed on the use of wavelets for diffusion models. In particular, we introduced an SDE-based multiscale approach to running diffusion models in the wavelet domain called WSGM. We discussed how this approach can be seen as a preconditioning to reduce bounds on the necessary number of discretization steps for an error. We then introduced how the score estimation backbone can be adapted to the structure of the wavelet domain.

To measure the quality of generated samples we discussed the current standard metric—the FID—and its flaws. We investigated a recently introduced metric based on the KL divergence between the power spectra of signal representation using wavelet packets and the FFT. In the analysis of the approach we derived that the proposed way to apply the metric to image *distributions* is equivalent to computing the expected pairwise divergence. We have argued theoretically that this is no useful metric and supported this empirically. Yet, we have shown that this flaw can be mitigated by calculating the divergence of expected probabilities instead, leading to a different divergence. Additionally, we extended this divergence by introducing a variant that compares the energy distributions spatially as well as across frequency bands.

We then extended WSGMs to wavelets of higher order and different step counts. We observed that depending on the discretization, the use of a higher order wavelet might lead to improvements, in particular in the approximation of higher frequencies. However, no wavelet or boundary handling technique performed clearly best for two-scale WSGMs.

The investigation of different discretization schemes was fruitful. In particular, choosing more steps for the unconditional base model while using a reasonable but smaller step count for the conditional model was clearly more efficient than choosing a step size which is constant across scales, offering in parts significant approximation improvements. Similarly, we could show that choosing a progressively increasing step count also performed better with a similar approximation order as the constant step count while maintaining an offset in each metric.

Furthermore, we could show that increasing the number of decomposition scales in

a WSGM was strictly superior for the Haar wavelet, improving all metrics, especially the approximation in the high-frequencies. For wavelets of higher order we only see clear improvements in closing the spectral gap, in particular in the high frequencies. The approximation in the low frequencies and the FID showed mixed developments. We speculate that this might be caused by the significant length of the higher order wavelets when compared to the base coefficients size.

We noticed that the choice of boundary handling mattered. We observed boundary artefacts for zero padding as well boundary filters. However, the artefacts of the boundary filters occured consistently but only for very small step sizes of the conditional models. The artefacts of zero padding were maintained even for higher step counts but could be avoided when the scale count was increased.

To conclude, we were able to show clear benefits of choosing less steps for conditional models and for increasing the scale count for Haar wavelets.

However, these remarks only apply to sampled images of size 128×128 and two to three scales—although the clear theoretical advantages of WSGMs lie in large image sizes and their multiscale nature. Furthermore, we were also only able to conduct experiments on a single dataset. The reason for both is the large computational cost of training and evaluating diffusion models. The experiments that were conducted for this work amounted to roughly estimated 6 months of GPU hours, stretching the possible scope of a Master's thesis. This corresponds to approximately emitted 200 kgCO_2eq [LLSD19]. Additionally, this cost increases the development and testing times of new ideas, slowing development cycles. In particular, testing WSGMs requires a large image size which makes small image datasets like CIFAR-10 unsuitable.

One idea that we were not able to evaluate for this work due to computational cost is the inclusion of frequency-spatial convolutional and attention operators in the U-Net, testing the potential of applying them in a multiscale environment and on wavelets of higher order. Note that these experiments have been implemented but the increase of computational cost due to the additional dimension of the operators made the experiments infeasible for the scope of this work.

Nevertheless, frequency-spatial blocks allow for an exciting extension. In the context of FWT coefficients the frequency dimension F in the frequency-spatial operators is always 4 as we need all coefficients to be of the same spatial size, binding us to single-level FWTs. This limits the spectral resolution available to the 1d frequency operators, since a single-level 2d wavelet transform only halves the spectrum along each axis. Addressing this limitation would be possible with a structural change to WSGMs, generating the coefficients of a WPT instead of FWT coefficients. This would allow for an increase in spectral resolution by using a multi-level WPT, strengthening the potential of the frequency operators. But this potential would come at a cost. We would have to leave the original setting of WSGMs with their theoretical guarantees. Also, we would either lose the multiscale property by generating all packet coefficients of a certain scale or we could explore the whole quaternary tree of the 2d WPT, requiring the generation of 4^J coefficients where J is the number of levels. Nevertheless, the former approach has the appeal of generating a signal in a spectrally localized representation by discretion of a single SDE.

For the application of WSGMs on FWT coefficients with boundary wavelets we used an existing boundary filter implementation [WBGH24] following the construction of

Herley and Vetterli [HV94]. The construction theoretically allows for an optimization of the boundary filters although this is not implemented. This optimization might allow us to mitigate the boundary artefacts occurring for very small step sizes. Alternatively, we could use a boundary filter formulation by Cohen, Daubechies, and Vial [CDV93] that is harder to implement but allows for better control of the corresponding wavelet basis of $L^2([0,1])$, e.g. retaining the maxflat properties of Daubechies wavelts.

To conclude, this work is concerned with a very exciting and fast moving topic. Our results support the use of wavelets for diffusion models—as a multiscale approach for the generation as well as in a quantitative measure of the sample distribution. However, as always further research is necessary. We hope that we could sketch potential next avenues.

Appendix

1 Further plots and samples

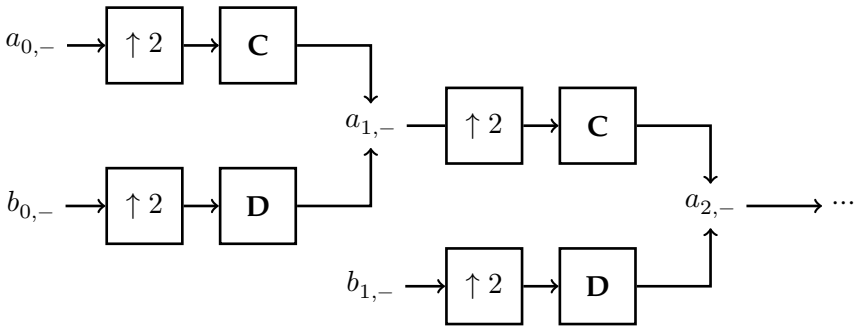
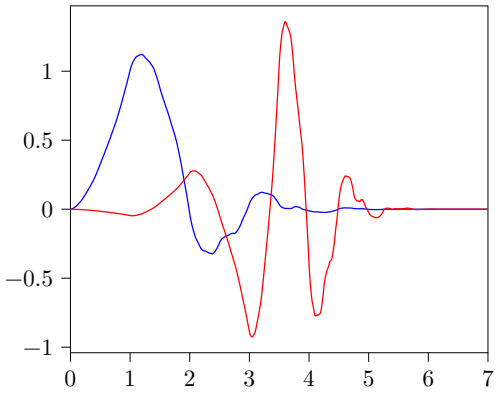
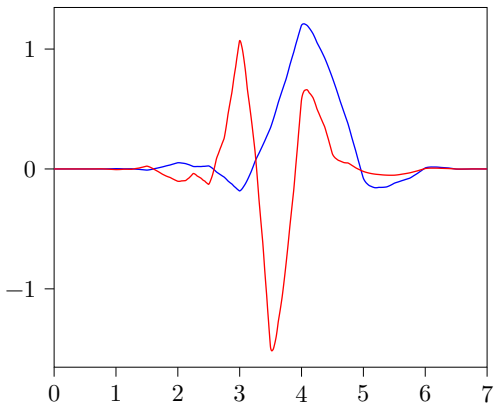


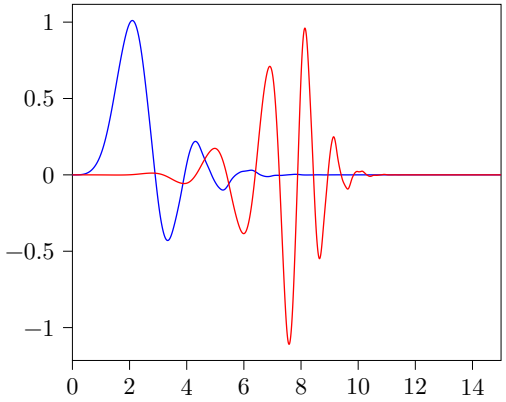
Figure 1: Scheme of the inverse fast wavelet transform



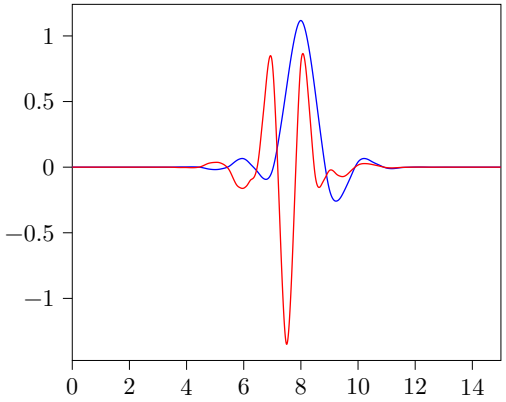
(a) Daubechies wavelet of length 8.



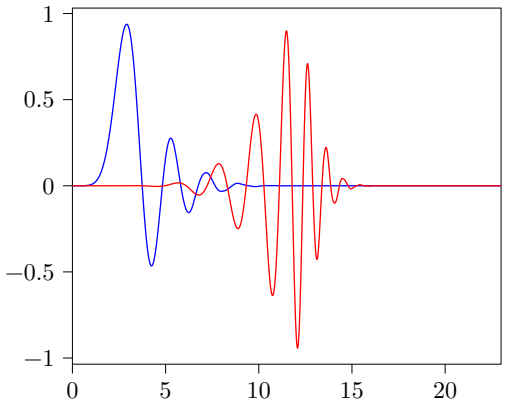
(b) Symlet of length 8.



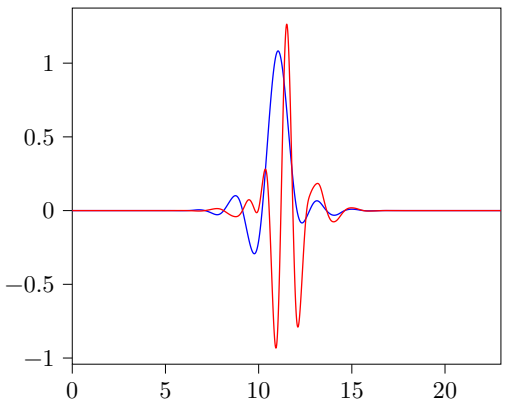
(c) Daubechies wavelet of length 8.



(d) Symlet of length 8.



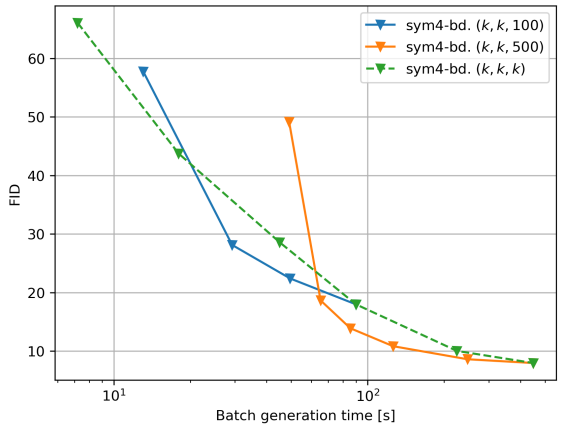
(e) Daubechies wavelet of length 12.



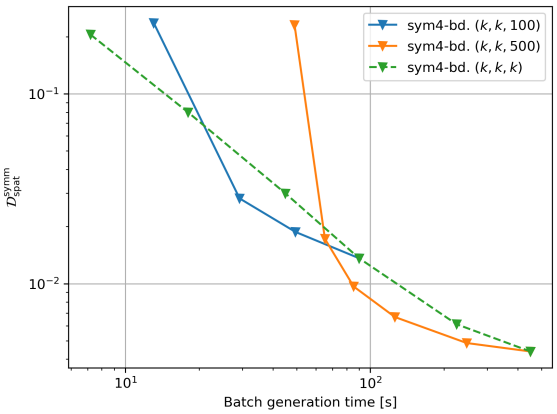
(f) Symlet of length 12.

Figure 2: Scaling functions (red) and mother wavelets (blue) for Daubechies wavelets (top) and symlets (bottom) of lengths 8, 16 and 24 (from left to right)

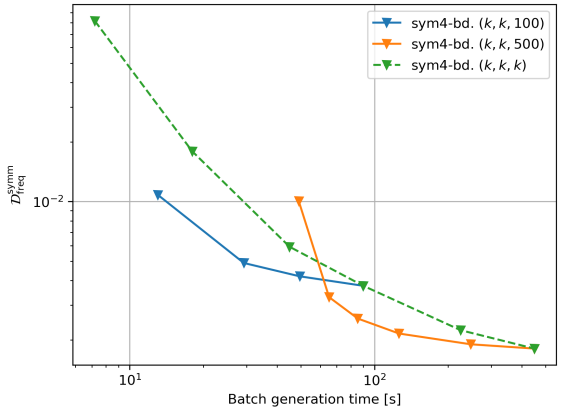
.



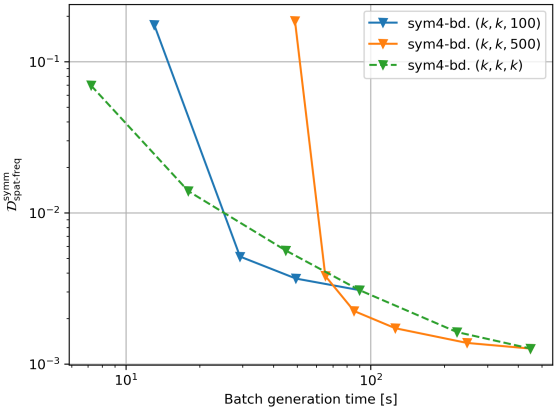
(a) FID w.r.t. the step count.



(b) $\mathcal{D}^{\text{symm}}_{\text{spat}}$ w.r.t. the step count.

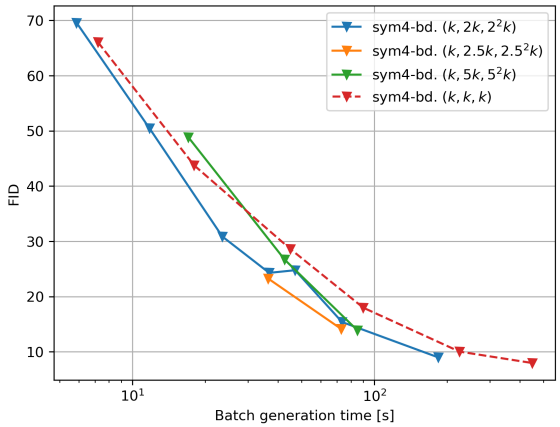


(c) $\mathcal{D}^{\text{symm}}_{\text{freq}}$ w.r.t. the step count.

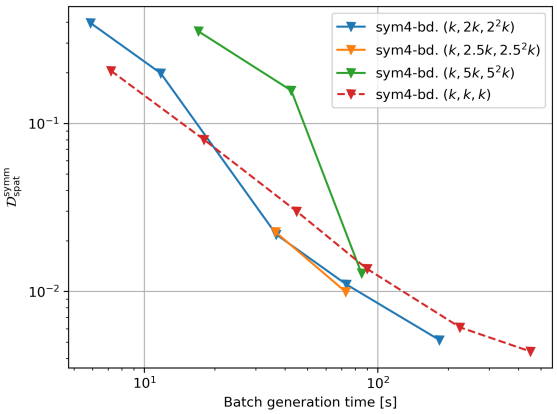


(d) $\mathcal{D}^{\text{symm}}_{\text{spat-freq}}$ w.r.t. the step count.

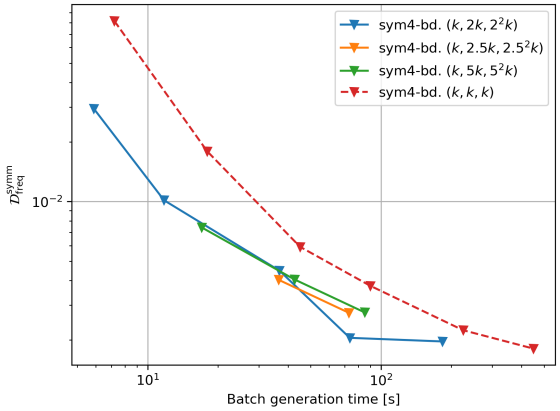
Figure 3: Comparison of two-scale *sym4* boundary wavelet WSGMs for a different number of discretization steps between the unconditional base model and the conditional models. We plot the influence of the number of sampling steps for the conditional models for 100 and for 500 steps in the unconditional model using: the FID (a) and power spectrum-based metric comparing energy distributions of wavelet packets spatially (b), Fourier coefficient (c) and wavelet packets accross frequency bands and spatially (d). As a baseline we add a two-scale *sym4* boundary WSGM with the same number of steps on all scales.



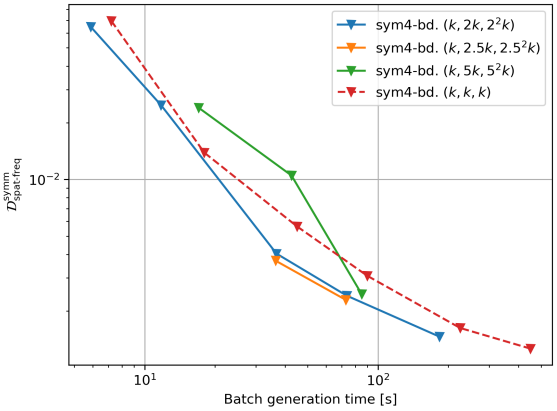
(a) FID w.r.t. the step count.



(b) $\mathcal{D}^{\text{symm}}_{\text{spat}}$ w.r.t. the step count.



(c) $\mathcal{D}^{\text{symm}}_{\text{freq}}$ w.r.t. the step count.



(d) $\mathcal{D}^{\text{symm}}_{\text{spat-freq}}$ w.r.t. the step count.

Figure 4: Comparison of two-scale *sym4* wavelet WSGMs for an increasing number of discretization steps between models. The number of steps is chosen as the step count on the previous level multiplied by a scaling factor α. We plot the influence of the step count for $\alpha \in \{2, 2.5, 5\}$ using: the FID (a) and power spectrum-based metric comparing energy distributions of wavelet packets spatially (b), Fourier coefficient (c) and wavelet packets accross frequency bands and spatially (d). As a baseline we add a two-scale *sym4* boundary WSGM with the same number of steps on all scales.

Figure 5: Samples for a DDPM with 100 discretization steps.

Figure 6: Samples for a 2-scale WSGM with 100 discretization steps for Haar wavelets.

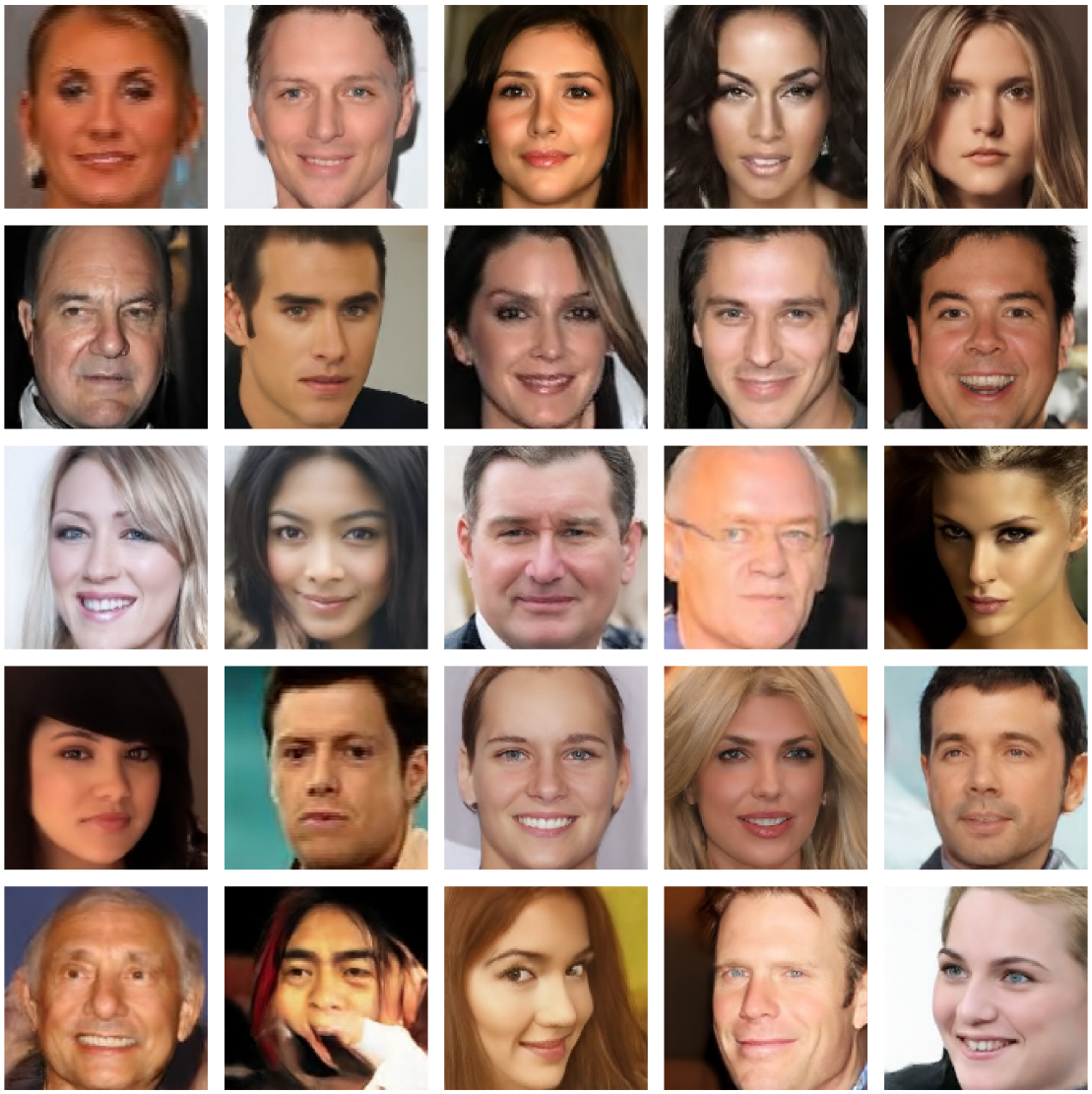


Figure 7: Samples for a 2-scale WSGM with 100 discretization steps for *sym4* wavelets using boundary wavelets.

Figure 8: Samples for a 2-scale WSGM with 100 discretization steps for *db4* wavelets using extension by reflection.

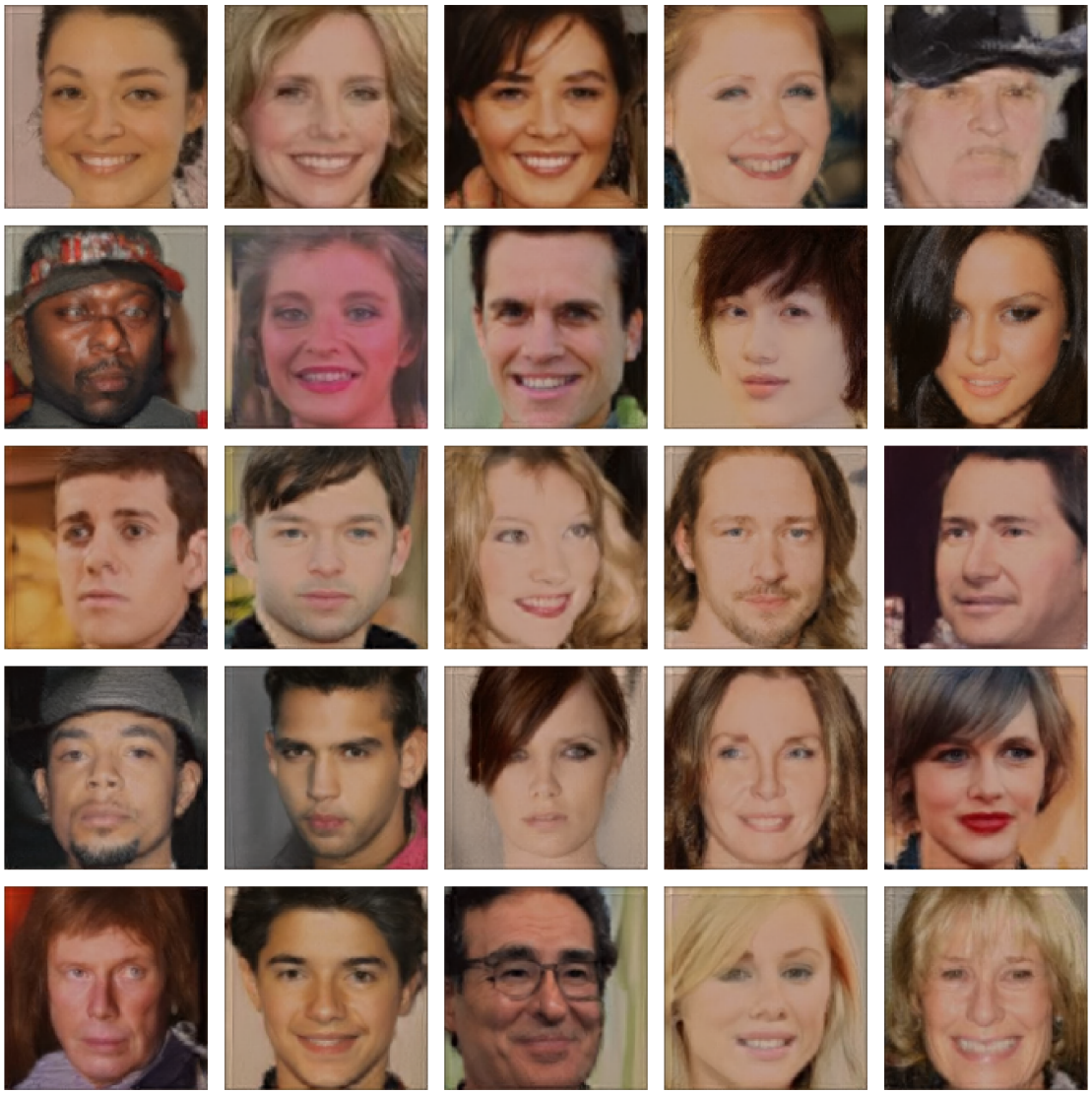


Figure 9: Samples for a 2-scale WSGM with 100 discretization steps for *db4* wavelets using zero padding.

Bibliography

[Alf+22] Motasem Alfarra et al. "On the robustness of quality measures for gans." In: *European Conference on Computer Vision*. Springer. 2022, pp. 18–33.

[And82] Brian D.O. Anderson. "Reverse-time diffusion equation models." In: *Stochastic Processes and their Applications* 12.3 (1982), pp. 313–326.

[Bea04] Richard Beals. *Analysis: an introduction*. Cambridge u.a.: Cambridge Univ. Press, 2004.

[Bla21] Felix Blanke. *Randbehandlung bei Wavelets für Faltungsnetzwerke*. Bachelor's Thesis. 2021.

[Bor22] Ali Borji. "Pros and cons of GAN evaluation measures: New developments." In: *Computer Vision and Image Understanding* 215 (2022), p. 103329.

[CDV93] Albert Cohen, Ingrid Daubechies, and Pierre Vial. "Wavelets on the Interval and Fast Wavelet Transforms." In: *Applied and Computational Harmonic Analysis* 1.1 (1993), pp. 54–81.

[CRBD18] Ricky TQ Chen et al. "Neural ordinary differential equations." In: *Advances in neural information processing systems* 31 (2018).

[Dau88] Ingrid Daubechies. "Orthonormal bases of compactly supported wavelets." In: *Communications on Pure and Applied Mathematics* 41.7 (1988), pp. 909–996.

[Dau92] Ingrid Daubechies. *Ten Lectures on Wavelets*. SIAM, 1992.

[Den+09] Jia Deng et al. "ImageNet: A large-scale hierarchical image database." In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255.

[DKK20] Ricard Durall, Margret Keuper, and Janis Keuper. "Watch your up-convolution: CNN based generative deep neural networks are failing to reproduce spectral distributions." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 7890–7899.

[DL82] D.C Dowson and B.V Landau. "The Fréchet distance between multivariate normal distributions." In: *Journal of Multivariate Analysis* 12.3 (1982), pp. 450–455.

[DN21] Prafulla Dhariwal and Alexander Nichol. "Diffusion models beat gans on image synthesis." In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794.

[Duc07] John Duchi. "Derivations for linear algebra and optimization." In: *Berkeley, California* 3.1 (2007), pp. 2325–5870.

[Fel49] W. Feller. *On the theory of stochastic processes, with particular reference to applications*. Proc. Berkeley Sympos. Math. Statist. and Probability 1946, 403-432 (1949). 1949.

[Fra+20] Joel Frank et al. "Leveraging frequency analysis for deep fake image recognition." In: *International conference on machine learning*. PMLR. 2020, pp. 3247–3258.

[Fré57] Maurice Fréchet. "Sur la distance de deux lois de probabilité." French. In: *Annales de l'ISUP* VI.3 (1957), pp. 183–198.

[GCDM22] Florentin Guth et al. "Wavelet Score-Based Generative Modeling." In: *Advances in Neural Information Processing Systems*. Vol. 35. 2022, pp. 478–491.

[GHBC21] Rinon Gal et al. "SWAGAN: A style-based wavelet-driven generative model." In: *ACM Transactions on Graphics (TOG)* 40.4 (2021), pp. 1–11.

[GM94] Ulf Grenander and Michael I Miller. "Representations of knowledge in complex systems." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 56.4 (1994), pp. 549–581.

[Goo+14] Ian Goodfellow et al. "Generative Adversarial Nets." In: *Advances in Neural Information Processing Systems*. Vol. 27. 2014.

[Gug+22] Sylvain Gugger et al. *Accelerate: Training and inference at scale made simple, efficient and adaptable.* https://github.com/huggingface/accelerate. 2022.

[Guo+23] Shi Guo et al. *Spatial-Frequency Attention for Image Denoising*. 2023. arXiv: 2302.13598 [cs.CV].

[Haa10] Alfred Haar. "Zur Theorie der orthogonalen Funktionensysteme." German. In: *Mathematische Annalen* 69.3 (1910), pp. 331–371.

[HD05] Aapo Hyvärinen and Peter Dayan. "Estimation of non-normalized statistical models by score matching." In: *Journal of Machine Learning Research* 6.4 (2005).

[HD22] Jun-Jie Huang and Pier Luigi Dragotti. "WINNet: Wavelet-inspired invertible network for image denoising." In: *IEEE Transactions on Image Processing* 31 (2022), pp. 4377–4392.

[Heu+17a] Martin Heusel et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium." In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[Heu+17b] Martin Heusel et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium." In: *Advances in neural information processing systems* 30 (2017).

[HHST17] Huaibo Huang et al. "Wavelet-SRNet: A wavelet-based CNN for multi-scale face super resolution." In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 1689–1697.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models." In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 6840–6851.

[Ho+22a] Jonathan Ho et al. "Cascaded Diffusion Models for High Fidelity Image Generation." In: *Journal of Machine Learning Research* 23.47 (2022), pp. 1–33.

[Ho+22b] Jonathan Ho et al. *Video Diffusion Models*. 2022. arXiv: 2204.03458 [cs.CV].

[Hua+23] Yi Huang et al. *WaveDM: Wavelet-Based Diffusion Models for Image Restoration*. 2023. arXiv: 2305.13819 [cs.CV].

[HV94] Cormac Herley and Martin Vetterli. "Orthogonal time-varying filter banks and wavelet packets." In: *IEEE Transactions on Signal Processing* 42.10 (1994), pp. 2650–2663.

[Jia+23] Hai Jiang et al. "Low-light image enhancement with wavelet-based diffusion models." In: *ACM Transactions on Graphics (TOG)* 42.6 (2023), pp. 1–14.

[Jl01] Arne Jensen and Anders la Cour-Harbo. *Ripples in mathematics: the discrete wavelet transform*. 2001.

[Jol+21] Alexia Jolicoeur-Martineau et al. *Gotta Go Fast When Generating Data with Score-Based Models*. 2021. arXiv: 2105.14080 [cs.LG].

[Kar+20] Tero Karras et al. "Analyzing and improving the image quality of stylegan." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 8110–8119.

[KB15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[Kid21] Patrick Kidger. "On Neural Differential Equations." PhD thesis. University of Oxford, 2021.

[KL51] S. Kullback and R. A. Leibler. "On Information and Sufficiency." In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86.

[KP92] Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Springer Berlin Heidelberg, 1992.

[KV89] Gunnar Karlsson and Martin Vetterli. "Extension of finite length signals for sub-band coding." In: *Signal Processing* 17.2 (1989), pp. 161–168.

[Kyn+23] Tuomas Kynkäänniemi et al. "The Role of ImageNet Classes in Fréchet Inception Distance." In: *Proc. ICLR*. 2023.

[Lee+19] Gregory R. Lee et al. "PyWavelets: A Python package for wavelet analysis." In: *Journal of Open Source Software* 4.36 (2019), p. 1237.

[Li+23] Jin Li et al. "Wavelet Transform-Assisted Adaptive Generative Modeling for Colorization." In: *IEEE Transactions on Multimedia* 25 (2023), pp. 4547–4562.

[Liu+20] Lin Liu et al. "Wavelet-based dual-branch network for image demoiréing." In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*. Springer. 2020, pp. 86–102.

[LLLY24] Shanchuan Lin et al. "Common diffusion noise schedules and sample steps are flawed." In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2024, pp. 5404–5411.

[LLSD19] Alexandre Lacoste et al. "Quantifying the Carbon Emissions of Machine Learning." In: *arXiv preprint arXiv:1910.09700* (2019).

[LLWT15] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild." In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015.

[Lu+22] Cheng Lu et al. "DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps." In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 5775–5787.

[Mal89] Stéphane Mallat. "A theory for multiresolution signal decomposition: the wavelet representation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11.7 (1989), pp. 674–693.

[Mos+23] Brian Moser et al. *Waving Goodbye to Low-Res: A Diffusion-Wavelet Approach for Image Super-Resolution*. 2023. arXiv: 2304.01994 [cs.CV].

[MVB20] Stanislav Morozov, Andrey Voynov, and Artem Babenko. "On self-supervised image representations for GAN evaluation." In: *International Conference on Learning Representations*. 2020.

[Nar+17] Sharan Narang et al. "Mixed precision training." In: *Int. Conf. on Learning Representation*. 2017.

[ND21] Alexander Quinn Nichol and Prafulla Dhariwal. "Improved Denoising Diffusion Probabilistic Models." In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 8162–8171.

[NMDB21] Charlie Nash et al. *Generating Images with Sparse Representations*. 2021. arXiv: 2103.03841 [cs.CV].

[Øks03] Bernt Øksendal. *Stochastic Differential Equations*. Springer Berlin Heidelberg, 2003.

[Par81] Giorgio Parisi. "Correlation functions and computer simulations." In: *Nuclear Physics B* 180.3 (1981), pp. 378–384.

[Pas+19] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. 2019, pp. 8024–8035.

[PDT23] Hao Phung, Quan Dao, and Anh Tran. "Wavelet Diffusion Models Are Fast and Scalable Image Generators." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2023, pp. 10199–10208.

[Pla+22] Patrick von Platen et al. *Diffusers: State-of-the-art diffusion models*. https://github.com/huggingface/diffusers. 2022.

[Pod+23] Dustin Podell et al. *SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis*. 2023. arXiv: 2307.01952 [cs.CV].

[PZZ22] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. "On aliased resizing and surprising subtleties in GAN evaluation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 11410–11420.

[Rah+19] Nasim Rahaman et al. "On the spectral bias of neural networks." In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5301–5310.

[Ram+22] Aditya Ramesh et al. "Hierarchical text-conditional image generation with clip latents." In: *arXiv preprint arXiv:2204.06125* 1.2 (2022), p. 3.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. 2015, pp. 234–241.

[Rom+22] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

[Sah+22] Chitwan Saharia et al. "Image super-resolution via iterative refinement." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.4 (2022), pp. 4713–4726.

[SE19] Yang Song and Stefano Ermon. "Generative Modeling by Estimating Gradients of the Data Distribution." In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019.

[SGSE20] Yang Song et al. "Sliced score matching: A scalable approach to density and score estimation." In: *Uncertainty in Artificial Intelligence*. PMLR. 2020, pp. 574–584.

[SL23a] Javier E. Santos and Yen Ting Lin. *Using Ornstein-Uhlenbeck Process to understand Denoising Diffusion Probabilistic Model and its Noise Schedules*. 2023. arXiv: 2311.17673 [stat.ML].

[SL23b] Inga Strümke and Helge Langseth. *Lecture Notes in Probabilistic Diffusion Models*. 2023. arXiv: 2312.10393 [cs.LG].

[SME21] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising Diffusion Implicit Models." In: *International Conference on Learning Representations*. 2021.

[SN96] Gilbert Strang and Truong Nguyen. *Wavelets and Filter Banks*. 1996.

[Son+21] Yang Song et al. "Score-Based Generative Modeling through Stochastic Differential Equations." In: *International Conference on Learning Representations*. 2021.

[SSLZ23] Shuyao Shang et al. *ResDiff: Combining CNN and Diffusion Model for Image Super-Resolution*. 2023. arXiv: 2303.08714 [cs.CV].

[SWMG15] Jascha Sohl-Dickstein et al. "Deep Unsupervised Learning using Nonequilibrium Thermodynamics." In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 2256–2265.

[Sze+16] Christian Szegedy et al. "Rethinking the inception architecture for computer vision." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[Tia+23] Chunwei Tian et al. "Multi-stage image denoising with the wavelet transform." In: *Pattern Recognition* 134 (2023), p. 109050.

[Tra+18] D. Tran et al. "A Closer Look at Spatiotemporal Convolutions for Action Recognition." In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA, June 2018, pp. 6450–6459.

[Vas+17] Ashish Vaswani et al. "Attention is All you Need." In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[Vin11] Pascal Vincent. "A connection between score matching and denoising autoencoders." In: *Neural computation* 23.7 (2011), pp. 1661–1674.

[VK95] Martin Vetterli and Jelena Kovačević. *Wavelets and subband coding*. 1995.

[VWG23] Lokesh Veeramacheneni, Moritz Wolter, and Juergen Gall. *Wavelet Packet Power Spectrum Kullback-Leibler Divergence: A New Metric for Image Synthesis*. 2023. arXiv: 2312.15289 [cs.CV].

[Wan+20] Jianyi Wang et al. "Multi-level wavelet-based generative adversarial network for perceptual quality enhancement of compressed video." In: *European Conference on Computer Vision*. Springer. 2020, pp. 405–421.

[WBGH24] Moritz Wolter et al. "ptwt - The PyTorch Wavelet Toolbox." In: *Journal of Machine Learning Research* (2024).

[WBHG22] Moritz Wolter et al. "Wavelet-packets for deepfake image analysis and detection." In: *Machine Learning* 111.11 (2022), pp. 4295–4327.

[WH18] Yuxin Wu and Kaiming He. "Group Normalization." In: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*. Vol. 11217. Lecture Notes in Computer Science. 2018, pp. 3–19.

[Wil+23] Christopher Williams et al. "A Unified Framework for U-Net Design and Analysis." In: *Advances in Neural Information Processing Systems* 36 (2023).

[WS99] Martin J Wainwright and Eero Simoncelli. "Scale mixtures of Gaussians and the statistics of natural images." In: *Advances in neural information processing systems* 12 (1999).

[Yoo+19] Jaejun Yoo et al. "Photorealistic style transfer via wavelet transforms." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9036–9045.

[Yua+23] Xin Yuan et al. *Spatial-Frequency U-Net for Denoising Diffusion Probabilistic Models*. 2023. arXiv: 2307.14648 [cs.CV].

[YZFW23] Xingyi Yang et al. "Diffusion probabilistic model made slim." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 22552–22562.

[Zha+22] Bowen Zhang et al. "Styleswin: Transformer-based GAN for high-resolution image generation." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 11304–11314.

List of Figures

List of Tables