DIFFUSION-BASED DENOISING OF HISTORICAL RECORDINGS

Bernardo Vieira de Miranda

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Luiz Wagner Pereira Biscainho

Rio de Janeiro
Julho de 2024

DIFFUSION-BASED DENOISING OF HISTORICAL RECORDINGS

Bernardo Vieira de Miranda

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Orientador: Luiz Wagner Pereira Biscainho

Aprovada por: Prof. Luiz Wagner Pereira Biscainho, D.Sc.
Prof. João Baptista de Oliveira e Souza Filho, D.Sc.
Prof. Hugo Tremonte de Carvalho, D.Sc.
Prof. Paulo Antonio Andrade Esquef, D.Sc.
Prof. Martín Rocamora Martínez, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2024

Miranda, Bernardo Vieira de

Diffusion-Based Denoising of Historical Recordings/Bernardo Vieira de Miranda. – Rio de Janeiro: UFRJ/COPPE, 2024.

XIII, 135 p.: il.; 29, 7cm.

Orientador: Luiz Wagner Pereira Biscainho

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2024.

Referências Bibliográficas: p. 126 – 135.

1. denoising. 2. diffusion. 3. neural networks. 4. restoration. I. Biscainho, Luiz Wagner Pereira. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

À minha família

Agradecimentos

Antes de mais nada, gostaria de agradecer à CAPES e ao governo brasileiro pela bolsa recebida no meu segundo ano de mestrado. Em muitos países ao redor do mundo a formação em nível de mestrado não é remunerada, e me enche de orgulho saber que nosso país em alguma medida ainda investe na formação de jovens pesquisadores. Sou grato ao meu orientador e amigo, Luiz Wagner Biscainho, pela paciência comigo, pelos almoços na UFRJ, pelas recomendações musicais, e sobretudo pela confiança depositada em mim desde os tempos de graduação. Espero que muitos outros alunos ainda tenham a sorte de tê-lo como exemplo em suas formações profissionais. Agradeço aos colegas de SMT pelo companheirismo no dia a dia do laboratório. Em especial, sou grato a Pedro Henrique, Wallace e João Parracho pelos conselhos na programação de redes neurais; a Leonardo, Bernardo Boechat, Victor Massatieze e Gutemberg pelas matérias que cursamos juntos; e a Pedro Donadio, Victor Costa e Luiz Gustavo pela companhia no SMT-II. Agradeço também a todos os voluntários dos testes subjetivos pelo tempo contribuído para este trabalho, e especialmente a Rafael Deslandes pela inestimável ajuda na condução desses testes. Sem o auxílio dele, essa dissertação certamente não teria sido concluída. Sou grato também aos meus amigos de fora da universidade. Agradeço a Eduardo Alves e Pedro Lisboa pela amizade que começou na escola e perdurou na faculdade; a Bernardo Fernandes e Rodrigo Armando pelos bares, blocos e sambas que frequentamos; a Enzo, Eduardo Santos, Rodrigo Acquarone, Gabriel Ferreira, Henrique e Paulo Feghali pelos churrascos e discussões sobre futebol; a Arthur, Rodrigo Moura e Marcel pela hospedagem e camaradagem em terras estrangeiras; a Gabriel Martins, Bruno Mesquita e Eric pelos ensaios e pela nossa apresentação em Botafogo; a Paulo Castro, Alexandre e Bruno Barroso pelos jantares e casamentos; e a Clara Faria pelo crescimento pessoal e relação que tivemos juntos. Por fim, gostaria de agradecer à minha família pelo apoio incondicional em todos os momentos. Agradeço ao meu pai Paulo André e à minha madrasta Suzana por serem um exemplo de família e por terem me dado dois irmãos incríveis, e à minha mãe Silvia e ao meu padrasto Jorge por terem sido uma referência na minha formação pessoal e intelectual. Ter dois pais e duas mães é uma sorte e um privilégio enorme, e a eles sou profundamente grato.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

REMOÇÃO DE RUÍDO DE GRAVAÇÕES HISTÓRICAS USANDO MODELOS DE DIFUSÃO

Bernardo Vieira de Miranda

Julho/2024

Orientador: Luiz Wagner Pereira Biscainho

Programa: Engenharia Elétrica

Dentro do campo da restauração de áudio, a remoção de ruído de fundo em gravações históricas de música é um problema relevante, cujas soluções atuais envolvem métodos de processamento de sinais ou de aprendizado profundo supervisionado. Nesta dissertação, uma nova abordagem para a resolução desse problema é investigada adaptando a geração condicional de sinais de áudio com modelos de difusão para a remoção *zero-shot* de ruído de fundo em versões digitalizadas de gravações de piano solo clássico em discos de 78 RPM. Abordagens comuns de geração condicional com difusão para restauração de áudio precisam de um modelo determinístico do defeito a ser removido, algo que não é possível para as gravações em 78 RPM. Esse obstáculo é superado pelo método proposto neste trabalho com o uso de um conjunto de exemplos de ruído, que é usado ao longo do processo de geração condicional para simular degradações comuns em discos de 78 RPM. Apesar da aplicação em gravações de 78 RPM, o método é genérico, e potencialmente serviria para qualquer degradação aleatória aditiva. Uma avaliação cuidadosa é feita através de experimentos com gravações históricas reais e com exemplos construídos artificialmente. Os resultados mostram que a restauração através de geração condicional com modelos de difusão possui desempenho similar ao estado da arte utilizando aprendizado profundo supervisionado. Há, no entanto, uma troca: enquanto modelos usando aprendizado profundo supervisionado tendem a remover partes do sinal junto com o ruído, o método proposto costuma preservar a integridade do sinal ao custo da permanência de ruído residual.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

DIFFUSION-BASED DENOISING OF HISTORICAL RECORDINGS

Bernardo Vieira de Miranda

July/2024

Advisor: Luiz Wagner Pereira Biscainho

Department: Electrical Engineering

In the context of audio restoration, background noise removal in historical music recordings is a relevant topic for which the use of signal processing and traditional supervised deep learning methods has been previously studied. In this work, a generative approach for removing perceptually distributed noise is investigated by adapting diffusion conditional sampling for zero-shot background noise removal in digitized classical solo piano 78 RPM recordings. Existing diffusion conditional sampling methods used in audio restoration typically require a deterministic model of the degradation being removed, which would not be possible for background noise removal in 78 RPM recordings. The proposed method overcomes this by using a set of noise examples to simulate 78 RPM defects during conditional sampling, which also makes the method potentially generalizable to any random additive degradation. Experiments using both real historical recordings and artificially generated examples show that diffusion models have comparable restoration performance to state-of-the-art supervised deep learning methods in most settings. However, there is a trade-off: while supervised deep learning methods tend to remove all noise at the cost of cutting off high-frequency signal components, the proposed diffusion approach preserves signal integrity, but leaves some residual noise.

Contents

List of Figures

List of Tables

Chapter 1

Presentation

Since the invention of the phonograph in 1877 by Thomas Edison [1], recorded music has been important evidence of past cultural events. Many unique presentations are preserved through historical recordings, and there has long been nothing extraordinary about listening to an artist who is no longer alive.

Recorded music has greatly benefited from the technological advances following its invention. The history of recorded music is usually divided in three eras: acoustic, electric, and digital.

During the acoustic era (1877 - 1925), storage means were brass or wax cylinders and vulcanite discs [1], and recording and reproduction equipment were both exclusively mechanic. The electric era (1925 - 1982) started with the introduction of valve amplifiers and microphones, which greatly improved recording quality. Other technological advances of this era were the appearance of vinyl long-plays (LPs) and of commercial magnetic tape recorders.

Current sound storage standards first appeared in the digital era, which began in 1982 with the commercial introduction of the compact disc (CD) and lasts until today. CDs popularized music storage in binary format, which enabled extraordinary breakthroughs in the manipulation and distribution of sound recordings thanks to the use of digital signal processing techniques [1].

All recorded sound shows some degree of degradation associated with the recording, storage, or reproduction stages of its production. For example, common defects associated with the recording and reproduction phases are frequency range limits, equipment non-linear behavior, and presence of noise. A few examples of storage defects are wearing and deformation of the storage mean through time, quantization (for digital recordings), and information loss during transmission.

The study of techniques to correct defects in previously recorded audio defines the domain of audio restoration, in which this work in inserted. Preservation of important musical recordings alone would justify research in audio restoration; however, there is also a growing commercial interest in restored historical recordings, as

can be seen by some recent releases of known recording companies [2] and by the popularity of re-recordings created using automatically controlled pianos [3, 4].

Restoration techniques prior to the digital era mostly used linear filters for background noise removal, but even physical methods, such as transcribing a disc with a tape recorder and cutting undesired parts of the magnetic tape, were not uncommon. With the advent of digital music formats, unprecedented results were obtained using a wide range of digital signal processing techniques. Still, up to this day, different types of degradations require different restoration methods.

One way of classifying recording defects is dividing them into local and global degradations. Local degradations affect small parts of the recording, whereas global ones affect the whole signal. Thumps produced by scratches in a disc are an example of a local defect, while magnetic tape hiss is an example of a global degradation.

This taxonomy is useful in the sense that restoration practices should preserve the musical content of the original signal as much as possible. Under this assumption, local defects can be treated with processing applied to few samples of a digitized recording, and more pervasive (and potentially harmful) methods can be used only for global degradations.

The problem with this division is that the frontier between local and global defects is not always evident. Impulsive noise, for example, is characterized by short pulses (clicks) that can be identified individually in the signal waveform; however the ensemble of pulses is perceived by the listener as a distributed background noise.

An alternative taxonomy is classifying degradations as: additive noises (e.g. clicks and hiss), linear distortions (e.g low-pass filtering), non-linear distortions (e.g saturation and clipping), and temporal distortions (e.g. tape or disc speed variations).

This work studies the use of diffusion models [5–7] for background noise removal in piano recordings from digitized 78 RPM discs. The interest in removing 78 RPM noise is that, while it is mostly made of clicks, it can also include thumps, motor rumble, and non-linear distortions. It has both local and global characteristics, and is statistically non-stationary [8].

Diffusion models are generative probabilistic models that have become the state-of-the-art for image and video generation [7, 9–11]. They are used in most commercial image generators [12–14], and have also been applied to image editing and denoising [15, 16].

The use of diffusion models for restoration problems is particularly interesting because they can be used in a zero-shot setting, i.e., without needing pairs of clean and degradated samples for training. Diffusion models are trained to generate samples from a distribution, and can be adapted to perform conditional sampling without new data [7].

In a restoration setting, a diffusion model can be trained to create generic piano samples, and a degraded sample can be used as a conditioning factor. With small changes to the sampling procedure, the model can output a clean version of the conditioner [17]. The drawback to this advantage is that this approach is maximally pervasive. Since diffusion models create new (conditional) samples, all parts of the degraded signal are subject to change.

1.1 An Overview of Background Noise Removal

Background noise removal is an extensively studied topic in signal processing, and the literature on this subject dates back to the late 1940s [18]. It is a general topic, but specialized methods for speech and general audio have been developed.

Traditionally, background noise is modelled with wideband spectrum, and processing is applied globally. This assumption is particularly true for hiss, but does not necessarily apply to all types of additive background noise, such as mains hum interference or 78 RPM noise in general.

Early methods for background noise removal were based exclusively on manipulating the degraded signal, but gradually evolved to depend on detailed statistical models for the clean signal [1, 8]. In recent years, with the unprecedented availability of speech and audio data, deep learning [19] denoising techniques grew popular as well, although works specialized in speech appear more often than those dedicated to musical signals.

The earliest methods for background noise removal in musical signals are quite possibly spectral subtraction and Wiener filtering [1, 8]. Both methods consist in dividing the signal into frames, and then performing signal processing on each frame. In spectral subtraction, frequency content attributed to the noise is subtracted from the degraded signal; in Wiener filtering, the clean signal is modelled as a wide-sense stationary process uncorrelated to the noise, and the restored output is the least squares estimate of the clean signal [20].

Both spectral subtraction and Wiener filtering can be shown [1, 8] to be just variable real gains for the frequency components of the noisy signal. As such, they may suffer from the problem of incorrectly estimating the phase of the restored signals, which can create audible artifacts. Furthermore, the two methods need an estimate of the frequency content of the noise to work, a drawback that is partially solved by using noise-only parts of the degraded signal to estimate the noise spectrum. One of the issues with this strategy is that, in cases where the noise is non-stationary, the noise spectrum in musical sections of the degraded signal might differ from the noise estimate. Random errors in the noise spectrum estimate may cause the methods to randomly create short-duration audible tones, generating an audible artifact dubbed

as "musical noise" [8].

Another technique for background noise removal which operates on the degraded signal alone is wavelet shrinkage [1, 21]. In wavelet representations [22], a signal is decomposed into coefficients describing increasing levels of signal detail. One of the properties of the wavelet coefficient representation is that it is sparse, and most signal information is kept on a few coefficients. For audio signals corrupted by wideband additive noise, the coefficients with small amplitudes mostly correspond to noise information, and denoising can be made by thresholding them.

In [1, 21], the authors make an extensive study of wavelet shrinkage denoising using real signals corrupted artificially with white noise. They also explore different kinds of wavelet representations and thresholding techniques. Their work suggests that wavelet shrinkage denoising works best in situations where the signal-to-noise ratio is higher, and that performance decreases sharply as noise power increases. Furthermore, it appears that the residual noise from wavelet shrinkage is signal-dependent, which complicates further processing steps on the signal.

There are examples of denoising methods using detailed statistical models for the clean signal dating back to the 1980s. In [23–25] a Gaussian distribution is assumed for the frequency components of the noisy signal. The clean signal components are assumed to be the mean of this distribution, and the squared magnitude of the noise spectrum is used as its variance. From this assumption, mean estimators can be derived to predict the clean components, which are later used to reconstruct the denoised signal. In [23] a maximum likelihood estimator is used, while minimum mean squared error estimators for the coefficient amplitudes and log-amplitudes are used in [24, 25].

It should be noted that the techniques in [23–25] focus on speech background noise removal, and are not specialized in musical audio. Also, like spectral subtraction and Wiener filtering, they require some knowledge of the noise spectrum, which must be estimated separately from noise-only frames of the signal.

More recent works using detailed statistical models for the clean signal include [26], which adopts a Bayesian approach to denoising musical audio. The authors suggest decomposing the noisy signal in two different bases, corresponding to its tonal and transient components. For each signal frame, the coefficients of the bases vectors are then modelled with a Bernoulli random variable indicating the presence of noise. If the variable is zero, the coefficient from the original decomposition is used, otherwise it is modelled as a Gaussian distribution with zero mean and variance given by an inverse gamma distribution. Since noise is likely to occur in adjacent frames or spread across similar basis vectors, a Markov model is adopted to determine the Bernoulli probability of the noise indicator. The posterior distribution of the remaining parameters is sampled using the Gibbs sampler [27]. Once

many samples are drawn, it is possible to obtain point estimates of the coefficients of the bases vectors, which can then be used to calculate the denoised output. The experiments of [26] use piano signals corrupted with white noise as input, and measure their results only through the variation in signal-to-noise ratio after restoration.

Other recent statistical approaches, such as [28, 29], rely on Kalman filters [20] to simultaneously remove clicks and additive broadband noise. Kalman filters can be used to predict the next clean sample in a sequence of observations corrupted by noise using the weighted average of an autoregressive (AR) prediction and the new noisy observation. In Kalman filter theory, the weights of the average are functions of prediction covariance, with less uncertain predictions having greater influence on the resulting average. In [28] an extended Kalman filter (EKF) — a generalization of conventional Kalman filters to non-linear systems that relies on a linearization procedure — was used to restore orchestral music artificially degraded with a mixture of white Gaussian noise and clicks. The authors of [29] improved the results of [28] by using two EKFs, with one making AR predictions of the current sample given past samples, and the other making AR predictions of the current sample given future samples. In [29], subjective tests evaluating the model with two pop music tracks are also included.

The two most relevant deep learning works for background noise removal in historical recordings of music are quite possibly [30, 31]. In both papers the authors try using automated musical activity detectors to extract noise-only excerpts in historical recordings. Then, they build noise datasets which are used to artificially degrade clean, high-quality, digital recordings by simple addition. Having pairs of clean and noisy versions of the same signals, traditional supervised deep learning techniques [19] can be used to build restoration models.

The pioneer work using this approach to historical recording restoration is [30]. In this paper, the authors determine noise-only parts of historical recordings by calculating the rolling standard deviation of digitized historical recording samples. Frames where the local standard deviation is among the 0.5% lowest are considered noise-only frames, and consecutive noise-only frames are kept as excerpts of the noise dataset.

Since noise excerpts would not necessarily match the length of the clean signal excerpts, they were extended through repetition with small overlaps. To avoid periodicity in the resulting noise samples, the authors also added Gaussian noise to the phase of the samples' frequency components, and randomly performed circular shifts before adding them to the clean signals. To emulate limitations of old recording equipment, the clean signals were low-pass filtered before noise addition as well.

The neural network of [30] takes the real and imaginary parts of the short-time Fourier transform (STFT) of a noisy signal as input, and has a standard

U-Net [32] architecture with a downsampling encoder branch, and an upsampling decoder branch. The output is the STFT of the restored signal.

Two training schemes are proposed, one using the absolute difference between input and reference as loss alone, and another combining the mean absolute error loss with an adversarial training regime [33]. In the adversarial scheme, additional discriminator networks, designed to identify if a signal is noisy or not, receive the restored output of the U-Net as input; their loss is backpropagated through the restoring U-Net to try to improve results.

The authors of [30] evaluated their work through a subjective test in which 11 human listeners rated 10 five-second excerpts of restored historical recordings in a $0-100$ continuous scale. Their test included two benchmarks (Wiener filtering and the model from [25]) and models trained with the simple and adversarial training regimes.

Their model outperformed the two benchmarks with both training regimes (the simpler being slightly better). It should be noted, however, that their benchmark models were developed for wideband additive background noise, and that their test included real historical signals with pulses, rumble, and clicks. Since the benchmark models were not designed for these defects, there may be some bias in their results.

In [31] a similar approach was used to restore recordings from 78 RPM discs. The authors manually extracted a small set of noise segments from a collection of publicly available 78 RPM recordings [34], and then used them to repurpose a voice activity detector [35] as a musical activity detector. By automatically detecting noise-only excerpts in a larger set of 78 RPM recordings, they were able to create a new noise dataset (see Section 4.1), which was then used to train a deep learning restoration model.

Their model is made of two U-Net stages. The first stage receives the STFT of the noisy signal as input, and provides an estimate of the noise components in it. This estimate is subtracted from the noisy signal and passed to an attention module, responsible for highlighting the musical features of the noisy signal. The output of the attention module is then concatenated with the STFT of the noisy signal and passed to the second U-Net stage, which is responsible for creating the final version of the restored signal. The sum of the absolute differences between the reference STFT and the outputs of the two stages is used as a loss function, which enables training both U-Nets simultaneously.

The authors evaluate their work objectively using perceptual audio quality evaluation metrics, and subjectively through a test similar to the one in [30]. In the subjective test of [31], 13 participants evaluated six five-second excerpts of real historic recordings restored by three methods: their own, the one from [30], and a composition of traditional signal processing methods (autoregressive click detection

and interpolation followed by the model of [25]).

One important aspect of the model of [31] is that it was designed to restore signals with very high noise levels, corresponding to discs mostly from the acoustic era, or from very early in the electric period. The examples shown on their webpage are all previous to 1920.

From the results, the two stage U-Net model appears to be the state-of-the-art among the deep learning historical recording restoration methods. It was considerably superior to the two other benchmarks in the objective and subjective tests of [31]. Additionally, their results showed that there was no significant difference between the method of [30] and the combination of traditional methods. This suggests that the results of [30] could indeed have been biased by the choice of wideband additive background noise removal techniques alone as benchmarks.

It should be noted that the subjective tests of both [31] and [30] have the drawback of using signals of five seconds only. Even considering that their test excerpts were extracted from real historical recordings, using very short segments can have an impact on the rating given by volunteers. Using longer excerpts would give a better idea of the generalization capability of the models, in addition to providing more information to volunteers for evaluation.

Although generative deep learning approaches are still seldom used for musical audio restoration, there are promising results for speech. Variational autoencoders (VAEs) [36], for example, have been shown to be able to remove noise from speech samples [37].

VAEs are trained to map a data distribution into a normal distribution in a latent space of smaller dimension, while keeping most signal information. This is done during training by adding Gaussian noise with variance conditioned on the input to the latent representations. In [37], the authors show through an ablation study that U-Nets trained with a variational scheme are able to reconstruct the latent representation of a noisy input with better audio quality than those without.

Generative adversarial networks (GANs) [33] not only inspired [30], but also have been used in vocoders which were shown to improve speech intelligibility under noisy circumstances [38, 39]. In [39] reconstruction and adversarial losses were combined to train a state-of-the-art model for speech enhancement from pairs of clean and degraded speech excerpts.

1.2 Contributions

To the best of the author's knowledge, there are no works in the literature that specifically investigate the use of diffusion models for general background noise removal in musical audio. However, in [17] (which is discussed in detail in Section 3.4),

diffusion models have been used successfully to solve general inverse problems in musical audio. In [40], the model of [17] was specialized for audio inpainting. This work aims to fill this research gap by experimentally studying the viability of diffusion models for removing perceptually distributed noise from historical recordings.

This work contributed towards this goal by training the generative diffusion model of [17] with high-quality piano samples, adapting it for removal of a generic additive noise, and evaluating it on the task of restoring historical 78 RPM piano recordings. Because of the large pitch range of the piano, and because classical solo piano pieces can vary widely in terms of style and musical range, working with the restoration of classical solo piano pieces provides solid ground for a proof of concept work, such as this one.

The proposed restoration method relies on conditional sampling (detailed in Section 3.3.2), and is zero-shot in the sense that it does not depend on the type of noise being removed. Because of this, although this work examines only 78 RPM noise removal, the method theoretically could be generalized to handle other additive degradations.

The objective and subjective evaluations of Chapter 4 include both artificially and naturally degraded signals, and are more extensive than those of [30, 31] in the sense that more signals are evaluated over a larger span of noise levels, including electrical recordings among the real-life signals. Additionally, the expressive number of pairs of objective and subjective evaluations of the same signal that were obtained in the course of the tests is a significant by-product of this work that can be used to study audio quality evaluation algorithms [41, 42].

1.3 Development Tools

The diffusion models trained in this work were written in Python's *pytorch* [43] library, adapting the code of [17]. The models require at least Python 3.9.1, *pytorch* 1.11.0, and CUDA [44] 10.2, but have been tested for *pytorch* 2.2.0 and CUDA 11.8. The complete source code can be found with installation instructions on the Github repository for this dissertation[1].

For the benchmarks of Chapter 4, the code of [31] was used mostly as available on their Github repository[2]. All changes are detailed in Chapter 4. Their model ran on Python 3.9.1, *tensorflow* 2.12.0, and CUDA 11.8.

The objective audio quality metric used in Chapter 4 was the basic version of the perceptual evaluation of audio quality (PEAQ) algorithm [41], running on MATLAB version R2017a with the implementation of [42].

[1] `https://github.com/bvm810/diffusion-audio-restoration`

[2] `https://github.com/eloimoliner/denoising-historical-recordings`

The machines used for training and inference of all models in this work belong to the computer cluster of the Signals, Multimedia, and Telecommunications Lab (SMT) of COPPE/UFRJ. The graphics processing units (GPUs) of the machines in the cluster were NVIDIA GeForce GTX and RTX cards. The slowest card was a GeForce GTX 1080, and the fastest a GeForce RTX 3090. All GPUs had video RAM (VRAM) between 8Gb and 24Gb.

1.4 Outline of this Work

The following chapters describe the theory behind audio restoration using diffusion models and the experiments using it for 78 RPM noise removal. Roughly speaking, Chapters 2 and 3 are dedicated to theory, while Chapters 4 and 5 focus on experimental results.

The first of the two theoretical chapters, Chapter 2, is important for understanding how musical signals can be represented for audio restoration applications. It gives a theoretical overview of time-frequency signal representations, starting with the discrete Fourier transform (DFT), going through the short-time Fourier transform (STFT), and ending with the representation used in the diffusion networks of this work: the constant-Q transform (CQT).

Chapter 3 is a thorough review of the theory behind generative diffusion models. It covers the most important works on diffusion models, starting with its two original formulations [5, 6], and ending with the currently used mathematical framework [7]. It also explains common heuristics used in diffusion generative models, and details the model used in this work.

The experimental set up and results are fully described in Chapter 4, which starts with preliminary qualitative results, and moves on to objective and subjective tests evaluating the performance of diffusion models for 78 RPM noise removal on classical solo piano music. Finally, Chapter 5 draws conclusions on the results of the previous chapter and points directions for future work.

Chapter 2

Time-Frequency Signal Representations

In very general terms, sound, images, and other phenomena can be represented mathematically as signals. As defined in [45], signals are functions of one or more variables that carry information about a physical phenomenon. Sound signals, in particular, are commonly represented as functions of time, frequency, or both.

Time signal representations are useful for identifying the location of certain events in time. The moment in which a musical note is played, for example, can usually be identified in a time visualization of a sound recording.

Frequency signal representations, on the other hand, are useful for identifying periodical content in the signal. A musical note of fixed pitch, for example, makes the sound pressure level in the air oscillate with a specific period. A frequency visualization of a sound recording would be useful for identifying which notes were played in a recording.

Time-frequency signal representations show the signal as a function of both time and frequency, and try to display the evolution of periodical content through time. Sheet music, for example, displays notes sequentially in time, and could be seen as a time-frequency representation of a musical piece.

This chapter briefly introduces signal representations in time and in frequency, then moves to discussing time-frequency representations that are useful for music restoration techniques. Section 2.4, in particular, explains the theory and implementation of constant-Q transform (CQT) based time-frequency representations, which are the input for the denoising models discussed later in this work.

2.1 Signals in Time and in Frequency

For restoration purposes, music must be in a format that can be dealt with in a computer. In other words, audio signals recorded in 78 RPM discs or other analog media must be converted to discrete, digital formats[1] (such as *.wav* files) in order to be treated.

Discrete audio signals are made of samples extracted from a sound recording[2]. Usually, samples are taken at fixed time intervals, and the time T_s elapsed between two samples is called the sampling period. Its inverse, $F_s = \frac{1}{T_s}$, is called the sampling frequency.

To reproduce sound, samples must be read in sequence with an interval T_s between them, just like the sequence of pictures in a movie. Therefore, a common way of visualizing an audio signal in time is displaying its waveform, that is, the sequence of all its samples. Figure 2.1 shows the waveform of a piano note sampled at $F_s = 44.1$kHz. It is possible to see that the note starts at roughly one second.

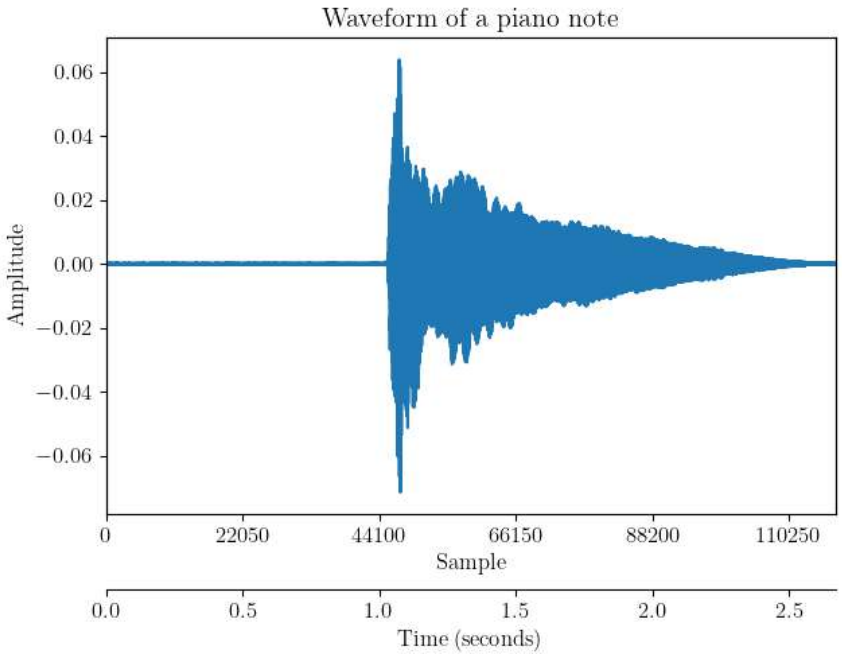

Figure 2.1: Waveform of a C4 piano note. The y-axis is scaled from one to minus one, chosen to represent respectively the maximum and minimum pressure levels that could be recorded without distortion.

Because samples are taken at regular time intervals, each sample index can be

[1]The signal processing theory for continuous signals in time and frequency will not be explained in this work. Only representations that are discrete in both domains will be described here. For a more detailed explanation of different signal representations, see [45, 46]

[2]Before storage in a computer-friendly format, samples from a sound recording are also quantized, i.e., have their continuous range of values mapped into a finite set of possible values. For more details, once again see [45].

converted to an instant of time using $t = nT_s$, where t is the instant of time and n is an integer sample index. This was done in the secondary x-axis of Figure 2.1.

A sequence of samples $x[n]$ of length N representing a discrete audio signal can be converted to and from the frequency domain using the discrete Fourier transform (DFT) [45, 47]. The DFT pair is given by

$$X[k] = \sum_{n=0}^{N-1} x[n] \mathrm{e}^{-j\frac{2\pi}{N}kn}, \tag{2.1}$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \mathrm{e}^{j\frac{2\pi}{N}kn}, \tag{2.2}$$

where $X[k]$ is the frequency representation of $x[n]$, and $0 \leq k \leq N-1$ is the frequency bin index.

The vector x[n] is made of samples extracted at intervals T_s from a continuous signal $x(t)$. If $x(t)$ has frequency content limited to $F_{\max} \leq \frac{1}{2T_s}$, then it is possible to show that $x(t)$ can be decomposed in a basis of functions $b(t) = \mathrm{e}^{j\frac{2\pi}{N}k\frac{t}{T_s}}$, with angular frequency $2\pi f = \frac{2\pi k}{NT_s}$ [45]. The DFT decomposes the vector of samples $x[n] \in \mathbb{R}^N$ in a basis N discrete-time N-periodical functions $b[n] = \mathrm{e}^{j\frac{2\pi}{N}kn}$ $(k = 0, \ldots, N-1)$ which are made of samples of $b(t)$ taken at intervals T_s. This can be seen intuitively through an analogy with the basis change formula from linear algebra [48].

Because of this, using $F_s = \frac{1}{T_s}$, each frequency bin k corresponds to the continuous frequency $f = \frac{kF_s}{N}$. In other words, the range of continuous frequencies is sampled by the DFT bins. It is important to observe that, since k ranges from 0 to $N-1$, the maximum frequency that can be represented by the DFT is $\frac{F_s(N-1)}{F_s}$.

Figure 2.2 shows the magnitude of the DFT of the piano note of Figure 2.1, calculated using Equation (2.1). The x-axis is displayed both in bins and in hertz, using the relation $f = \frac{kF_s}{N}$. The largest peak at 523Hz indicates that the note that was played was a C4, which is the C immediately above the middle C in a piano.

2.2 The Short-Time Fourier Transform

Time information is no longer visible in the coefficients $X[k]$. In order to obtain a mixed representation, a common strategy is to apply the DFT to small chunks (frames) of the signal, weighted by an analysis window function $w[n]$ centered around some sample of the signal $x[n]$.

This is precisely what is done by the short-time Fourier transform (STFT). For

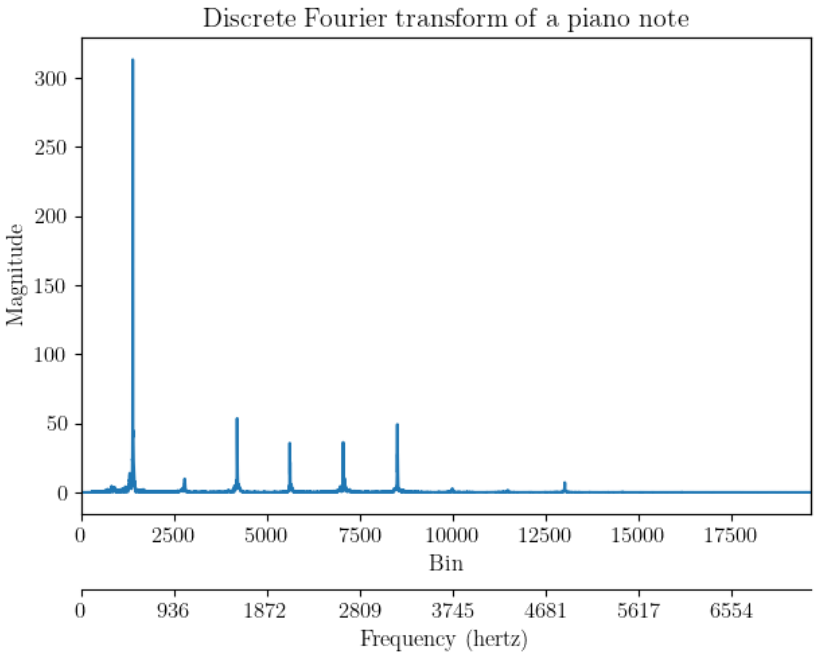

Figure 2.2: Discrete Fourier transform of a piano note. The x-axis was cut just above 7kHz for better visualization.

a signal $x[n]$ of size N, the STFT is traditionally calculated as [49]

$$\mathcal{X}_{\text{STFT}}(m,k) = \sum_{\ell=0}^{L-1} x[\ell + mH]w[\ell]\mathrm{e}^{-j\frac{2\pi}{L}k\ell}, \tag{2.3}$$

where L is the frame size, m is the the frame index, and $H \leq L$ is the number of samples between the center of two consecutive frames (hop size). The analysis window $w[n]$ is a real positive sequence of size L where $w[L-n] = w[n]$. The product $x[\ell + mH]w[\ell]$ selects L samples from $x[n]$, starting at mH, and weights each of those samples in the DFT with the analysis window.

There are a couple of important aspects to be noted in the STFT. Firstly, $\mathcal{X}_{\text{STFT}}(m,k)$ is a matrix belonging to $\mathbb{R}^{M\times L}$, where M is the number of signal frames used. The number of frequency bins L is entirely determined by the frame size because of the DFT. The number of frames M depends on the signal size, frame size, and hop size.

As can be seen in Figure 2.3, if the signal can be divided in an exact number of frames, then $M = \frac{N-L}{H} + 1$. If this is not the case, then zeros can be added at the beginning and/or end of the signal so that an integer number of frames fits into the signal without discarding any samples. The number of zeros to be added will be $H - ((N-L) \mod H)$.

In the STFT, the frequency contributions of $x[n]$ and $w[n]$ in each frame are inseparable because the DFT is applied over their product. Smooth windows lead

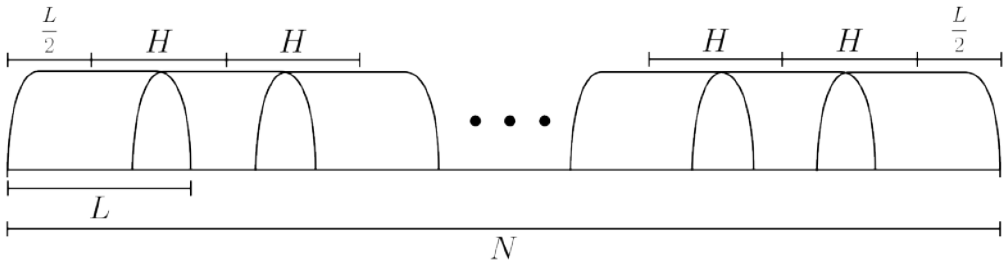

Figure 2.3: Division of a signal into blocks for the STFT. The distance between the middle of each block is H, which makes $N = MH + L$ if the signal can be divided in an exact number of blocks.

to less spectral lekage on the local frequency content of $x[n]$ [47, 50], but any window choice will have some impact on the STFT.

Regardless of the choice of $w[n]$, windowing introduces time-frequency resolution limitations into the STFT. If a long $w[n]$ is used, temporal resolution on the representation of $x[n]$ is lost, since time information cannot be seem in the $X[k]$ of the frames. Using a short $w[n]$, on the other hand, does not provide enough samples to capture low frequency information, i.e., periodical content with long periods.

Computationally, the STFT can be described in three steps: division of $x[n]$ in overlapping blocks of size L, pointwise product with $w[n]$, and application of the DFT. The diagram in Figure 2.4 illustrates the STFT algorithm for a signal of size $N = 12$, with blocks of size $L = 4$ and hop size $H = 2$.

A big advantage of the STFT is that, under certain conditions, it is reversible, that is, it is possible to recover $x[n]$ exactly given $\mathcal{X}_{\text{STFT}}(m, k)$. This is done using the overlap-and-add (OLA) algorithm [50]. The OLA algorithm consists of two steps: applying the inverse DFT to each frame, and adding the overlapping parts of each frame weighted by a synthesis window $w'[n]$ of the same size as $w[n]$. Figure 2.5 illustrates the OLA algorithm for the same signal and parameters of Figure 2.4.

In each block, after inverting the DFT, the product of signal block and analysis window is obtained. Also, if $H \geq \frac{L}{2}$, only adjacent blocks overlap. In this case, except for the first and last blocks, in any frame m, samples from zero to $L - H - 1$ will overlap samples from H to $L - 1$ of frame $m - 1$, and samples from H to $L - 1$ will overlap samples from zero to $L - H - 1$ of frame $m + 1$. This can be seen in Figure 2.5 for $L = 4$ and $H = 3$.

Therefore, OLA computes the samples $\hat{x}[n]$ of all frames except first and last

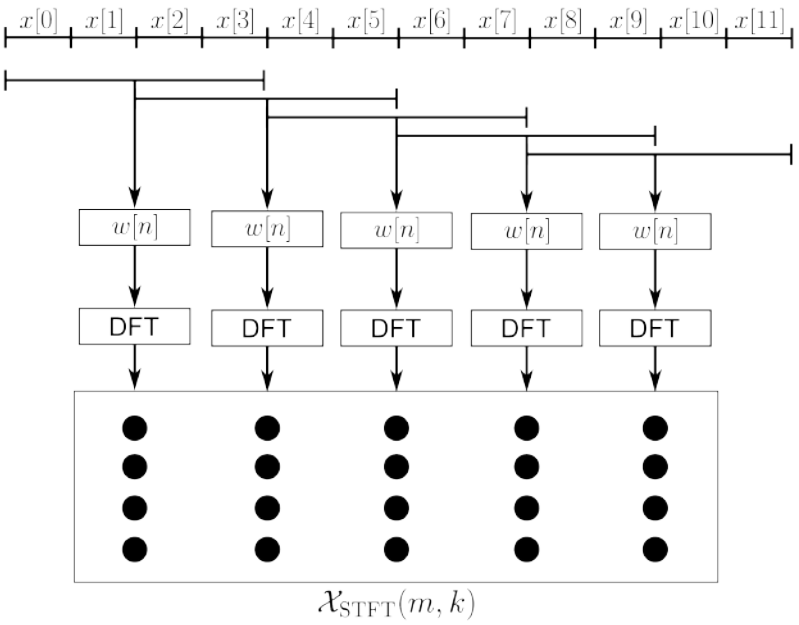

Figure 2.4: Short-time Fourier transform illustration. The $w[n]$ blocks indicate pointwise multiplication of the windows and the chunks of $x[n]$ delimited by the brackets. Each dot in the square in the bottom represents an entry in $\mathcal{X}(m,k)$. Note that it is transposed, i.e., has size four (L) by five (M).

as [50]

$$\hat{x}[n] = \begin{cases} x[n](w[\ell+H]w'[\ell+H] + w[\ell]w'[\ell]) & \text{for } 0 \leq \ell \leq L-H-1, \\ x[n]w[\ell]w'[\ell] & \text{for } L-H \leq \ell \leq H-1, \\ x[n](w[\ell]w'[\ell] + w[\ell-H]w'[\ell-H]) & \text{for } H \leq \ell \leq L-1. \end{cases} \tag{2.4}$$

For those samples, $\hat{x}[n] = x[n]$ as long as

$$\begin{cases} w[\ell+H]w'[\ell+H] + w[\ell]w'[\ell] = 1 & \text{for } 0 \leq \ell \leq L-H-1, \\ w[\ell]w'[\ell] = 1 & \text{for } L-H \leq \ell \leq H-1, \\ w[\ell]w'[\ell] + w[\ell-H]w'[\ell-H] = 1 & \text{for } H \leq \ell \leq L-1. \end{cases} \tag{2.5}$$

The intuition of perfect signal reconstruction with OLA lies in this complementarity of overlapping sections of the product between the analysis and synthesis windows. This is described by Equation (2.5), and can be seen in the middle part of Figure 2.6; whenever there are overlaps, the combined effect of the analysis and synthesis windows must be equal to multiplication by one.

This is broken in the first and last frames, as there could be non-overlapping regions in $w[n]$ and $w'[n]$. To avoid reconstruction issues, a simple solution is padding the signal with H zeros on both sides. This ensures that the first and last effective frames will have complementary preceding and succeeding blocks, and keeps the

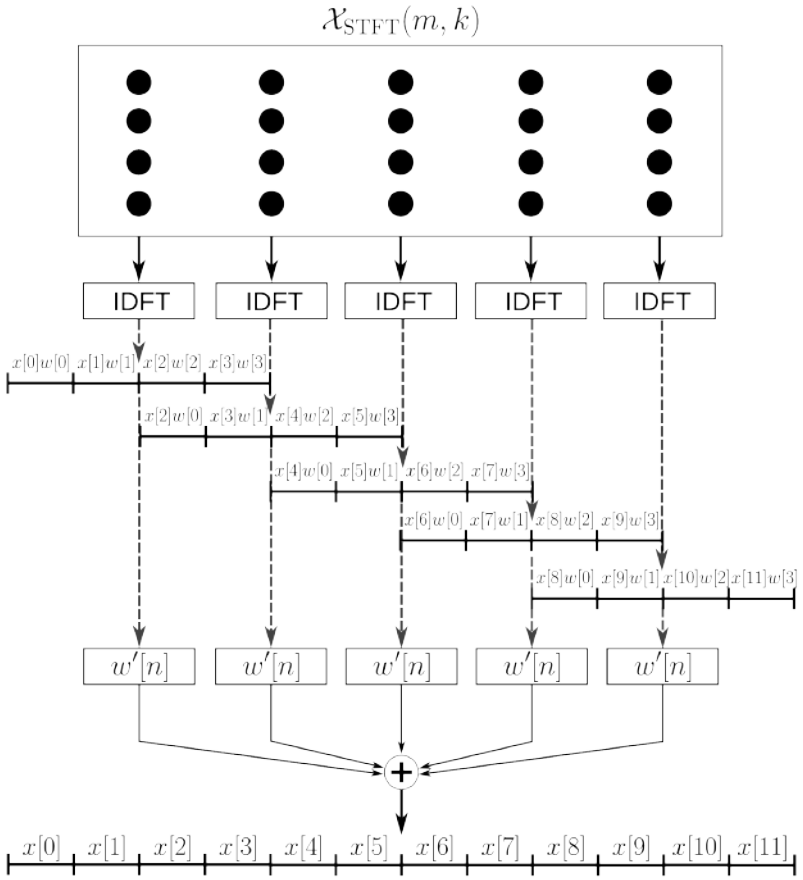

Figure 2.5: Inverse short-time Fourier transform via overlap-and-add. The $w'[n]$ blocks represent pointwise multiplication with the synthesis windows. For clarity, the result of the inverse DFT is explicitly shown for each frame. In the summation block, blocks are added side by side.

signal intact. Note that this should be done after padding to make the signal divisible in M frames.

Perfect reconstruction can be achieved by applying OLA and removing the trailing zeros from the signal. Figure 2.6 describes the intuition behind OLA, and how padding can be used to ensure perfect reconstruction. Once again the signal of Figure 2.4 is used as an example.

The STFT has an interesting alternative interpretation in terms of internal products. For $x[n]$ of length N, consider the set $\{g_{m,k}\}$ made of circular shifts of size mH with $m = 0, \ldots, M$ of the sequence

$$g_k[n] = \begin{cases} w[n]\mathrm{e}^{j\frac{2\pi}{L}kn} & \text{for } 0 \leq n \leq L-1, \\ 0 & \text{for } L \leq n \leq N-1. \end{cases} \tag{2.6}$$

Denoting $\mathbf{x} = x[n]$ and $\mathbf{g}_{m,k} = g_{m,k}[n]$, each entry in $\mathcal{X}_{\mathrm{STFT}}(m,k)$ can be written

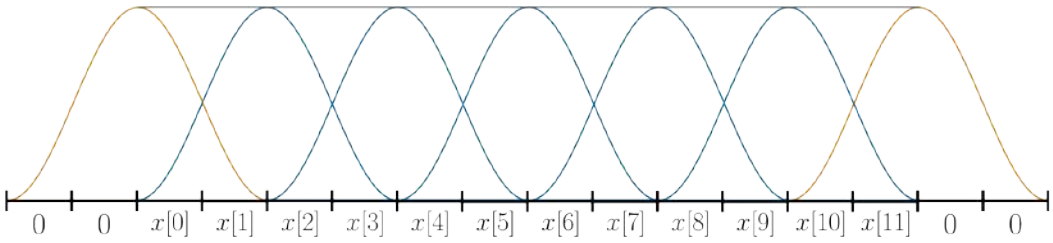

Figure 2.6: Window complementarity in overlap-and-add. The curves represent the combined effects of analysis and synthesis windows. Put side by side, they must add to a constant signal, as shown in the figure. Since the first or last few samples might be affected by the curves, it is common to pad the signal with zeros beforehand, and to add two frames to account for padding.

as the internal product $\langle \mathbf{x}, \mathbf{g}_{m,k} \rangle = \mathbf{g}_{m,k}^H \mathbf{x}$[3]. In other words, the STFT coefficients can be seen as projections of $x[n]$ over a set of atoms.

The atoms $\{g_{m,k}\}$ are traditionally called discrete Gabor frames, and hence the STFT is sometimes called the Gabor transform. As the $\{g_{m,k}\}$ are localized around certain samples and centered around specific frequencies (because of the complex exponential modulation term), the STFT coefficients can be interpreted as the amount of signal energy in the time-frequency region spanned by each atom. Since each atom is shifted by an integer multiple of H, and modulated by a frequency that is an integer multiple of $\frac{2\pi}{L}$, this leads to a uniform division of the time-frequency plane.

Another way to visualize the STFT grid is to realize that, after conversion to continuous time and frequency using $t = \frac{n}{F_s}$ and $f = \frac{kF_s}{L}$, the distance between two consecutive frames is always $\frac{H}{F_s}$ seconds, and the distance between two consecutive bins is always $\frac{F_s}{L}$Hz.

2.3 The Non-Stationary Gabor Transform

The uniform partition of the time-frequency plane generated by the STFT is not always desirable. Consider the case of a musical piece with two parts; one slow and low-pitched, and another fast and high-pitched. A groovy bass line followed by a fast guitar solo, for example.

Using a single frame size (i.e. identical atoms shifted and modulated) would require compromising between the long windows required for the bass part, and the short windows required for the fast part. It would be more interesting to have a

[3]In this equation, the vectors are taken to be column-vectors, and H denotes the conjugate-transpose operator. The same convention is adopted in all other equations of this dissertation, unless stated otherwise.

representation with different analysis windows for each part. This is the idea of the non-stationary Gabor transform [51, 52] (NSGT).

In the NSGT, each matrix entry is calculated as

$$\mathcal{X}_{\mathrm{NSGT}}(m,k) = \sum_{\ell=0}^{L-1} x[\ell + H_m] w_m[\ell] \mathrm{e}^{-j\frac{2\pi}{L}\ell k}, \tag{2.7}$$

where $m = 0, \ldots, M-1$ indexes frames, H_m denotes the start of frame m, and $w_m[n]$ is the real analysis window of length L used for frame m. Note that even though $w_m[n]$ has size L, it may have support $S_m \leq L$. Similarly to the STFT, the non-zero samples of the $w_m[n]$ should be symmetrical.

Customizing the support size for each $w_m[n]$ allows for different window coverage in each frame, while still computing an L-point DFT for all of them. As long as the H_m are set according to the window supports, forcing the whole signal to be covered by some analysis window, no information is lost in the NSGT.

However, since the whole signal needs to be covered, using $S_m < L$ makes the hop between two frames smaller than what it would be if the support of all windows were equal to L. Also, using $S_m < L$ with L sized DFTs implies introducing unnecessary calculations because some points will be multiplied by zero.

Computationally, the NSGT can be described in four steps. First, L is chosen as at least the size required to represent the lowest frequency of interest. Then, the signal is divided in blocks of size L considering the hop sizes H_m. In order to ensure that no part of the signal is skipped, $H_m \leq \sum_{m'=1}^{m} S_{m'-1}$, for $m \geq 1$, with $H_0 = 0$. Finally, pointwise multiplication with $w_m[n]$ is performed, and the DFT is applied to the result.

Depending on the support of each window, and on the signal size, zero-padding might be required on the last block to ensure it has size L. The diagram in Figure 2.7 illustrates the NSGT algorithm for a signal of size $N = 12$ (before padding), with windows of support $S_0 = 2$, $S_1 = 6$, $S_2 = 4$.

The name of the NSGT comes from the fact that it also has an interpretation as an internal product, but with a set of atoms that is not made of circular shifts of the same sequence. Since the atom shape can change over time, the set $\{g_{m,k}\}$ is called a non-stationary Gabor frame. For the NSGT, each atom $g_{m,k}[n]$ can be written as

$$g_{m,k}[n] = \begin{cases} w_m[n - H_m]\mathrm{e}^{j\frac{2\pi}{L}nk} & \text{for } H_m \leq n \leq H_m + L - 1, \\ 0 & \text{for } 0 \leq n \leq H_m - 1 \text{ and } H_m + L \leq n \leq N - 1, \end{cases} \tag{2.8}$$

and $\mathcal{X}_{\mathrm{NSGT}}(m,k) = \langle \mathbf{x}, \mathbf{g}_{m,k} \rangle = \mathbf{g}_{m,k}^H \mathbf{x}$. Note that this formula assumes that the signal has been zero-padded at its end to fit the final L-sized block.

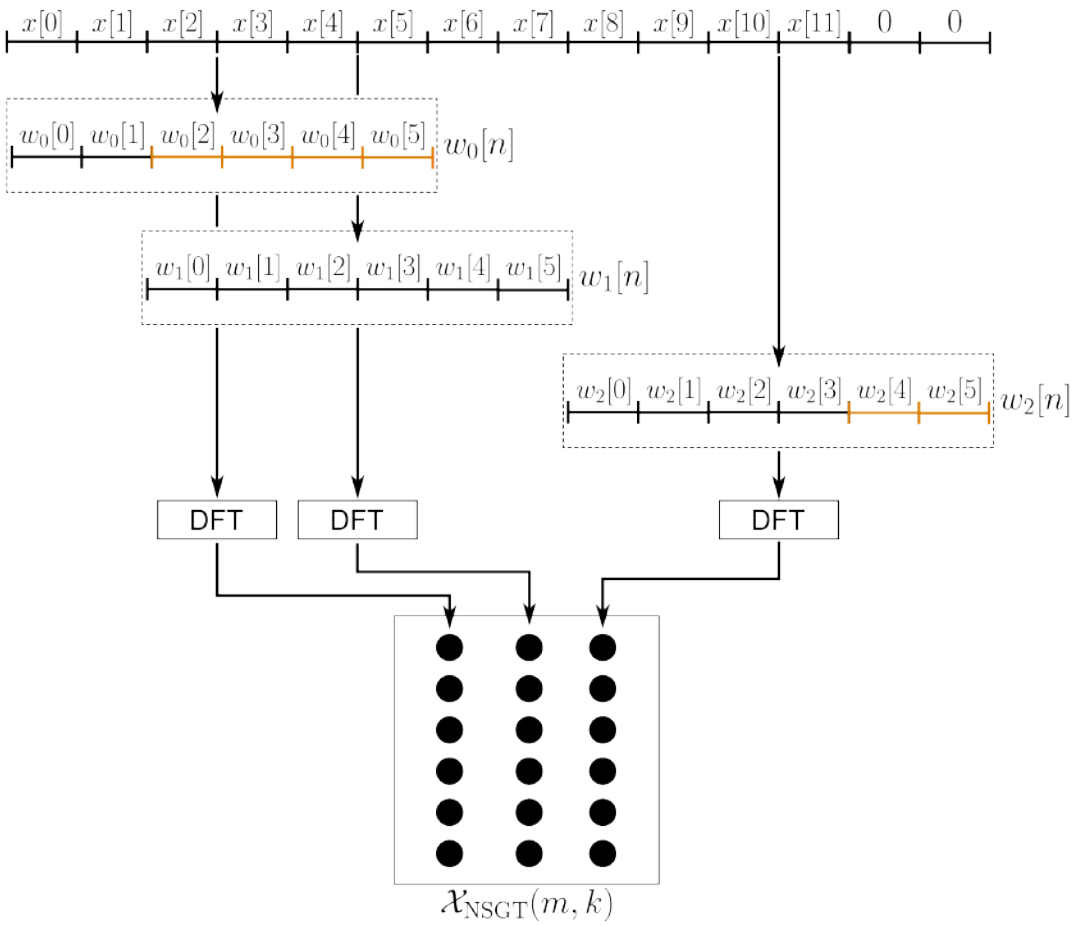

Figure 2.7: Non-stationary Gabor transform. In the illustration, window samples equal to zero are marked in orange. Note that the last frame required zero-padding of two, making $N = 14$.

It is possible to invert the NSGT using OLA [51], as long as the synthesis windows are carefully chosen. The relations of Equation (2.5) are no longer valid because of variable window types and hop sizes, but, as before, the algorithm inverts the DFT and adds overlapping blocks using a synthesis window. Observe that overlaps might happen in non-adjacent blocks, and that the synthesis windows $w'_m[n]$ could be different for each frame.

In order to achieve perfect reconstruction, one needs to use a synthesis window that compensates the effect of the analysis windows on all overlapping block parts. Suppose, for example, that $x[10]$ appears in the tenth sample of the first frame, fifth sample of the second frame, and second sample of the third frame. After inverting the DFT, the reconstructed sample $\hat{x}[10]$ is

$$\hat{x}[10] = x[10]w_1[10]w'_1[10] + x[10]w_2[5]w'_2[5] + x[10]w_3[2]w'_3[2], \tag{2.9}$$

which implies that perfect reconstruction would require

$$\begin{aligned} w_1'[10] &= \frac{w_1[10]}{w_1[10]^2 + w_2[5]^2 + w_3[2]^2}, \\ w_2'[5] &= \frac{w_2[5]}{w_1[10]^2 + w_2[5]^2 + w_3[2]^2}, \\ w_3'[2] &= \frac{w_3[2]}{w_1[10]^2 + w_2[5]^2 + w_3[2]^2}. \end{aligned} \tag{2.10}$$

In the more general setting, it helps to store the result of the inverse DFT of each frame as a sequence of size N where the $N - L$ samples not belonging to the block are zero. This can be done by saving the indexes of the samples belonging to each block before applying the DFT. Then, the L-sized sequence that is obtained after DFT inversion can be added to the same indexes in a zero vector to obtain the inverse DFT of the blocks as sequences of size N.

Using this procedure, the inverse DFTs of the blocks become simply the pointwise product between $x[n]$ and $g_m[n] = |g_{m,k}[n]|$. In this case, the OLA reconstructed signal can be written as

$$\hat{x}[n] = \sum_{m=0}^{M-1} x[n] g_m[n] g_m'[n], \tag{2.11}$$

where $g_m'[n]$ is defined like $g_m[n]$, but using the synthesis windows $w_m'[n]$ instead of $w_m[n]$. Similarly to the example, in order to have $\hat{x}[n] = x[n]$, one must have

$$g_m'[n] = \frac{g_m[n]}{\sum_{m=0}^{M-1} g_m[n]^2}. \tag{2.12}$$

In other words, in each sample the weights of the analysis windows applied to it must be compensated by the synthesis windows. Note that STFT inversion can also be viewed from this standpoint, remembering that complementarity in adjacent windows ensures that $\sum_{m=0}^{M-1} g_m[n] g_m'[n] = 1$.

Considering that the $g_m'[n]$ can be calculated beforehand knowing the $w_m[n]$, NSGT inversion can be resumed to the following steps: application of the inverse DFT in each frame, extension of the results to size N, application of $g_m'[n]$, and pointwise summation of all vectors obtained in the preceding steps. Figure 2.8 illustrates the inverse NSGT algorithm for Figure 2.7.

2.4 The Constant-Q Transform

The NSGT allows one to use different windows in each frame, but does not completely solve the time-frequency resolution issues of the STFT. For any given frame

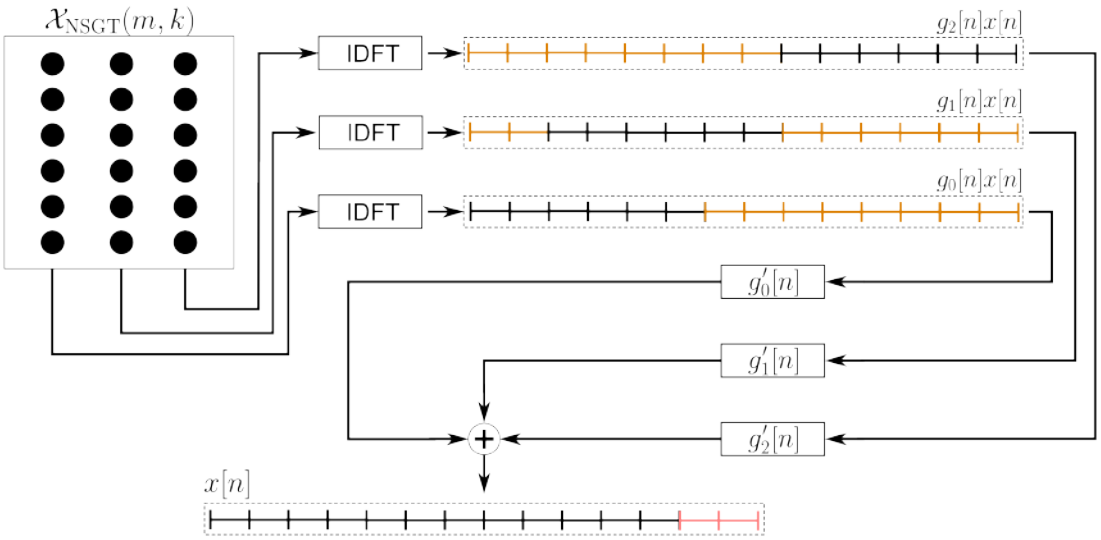

Figure 2.8: Inverse non-stationary Gabor transform. The products $g'_m[n]x[n]$ after the inverse DFTs are shown explicitly, and the extension terms equal to zero are marked in orange. The two pad zeros in $x[n]$ are marked in red at the bottom.

m, one has to choose between windows with short and long support, and representing the whole audible frequency range with good resolution without blurring time information remains a challenge.

However, non-stationary Gabor frames can also be used to efficiently implement redundant, constant-Q transform (CQT) based, time-frequency representations [52]. Such representations avoid the time-frequency resolution trade-off using frequency dependent analysis windows, at the cost of redundancy introduced by the use of non-stationary Gabor frames.

The CQT originally appeared in the context of musical signal processing and has the property of matching note progression in equal temperament[4]. In the equal tempered scale, notes are arranged geometrically, and an octave interval doubles the frequency of a note. Furthermore, in this system octaves are divided in twelve (geometrically spaced) notes, which makes the ratio between the frequencies of adjacent notes equal to $2^{\frac{1}{12}}$. With this in mind, one can attribute to a note with frequency f_n the frequency band Δf_n, calculated as

$$\Delta f_n = \sqrt{2^{\frac{1}{12}} f_n f_n} - \sqrt{f_n 2^{-\frac{1}{12}} f_n} = (2^{\frac{1}{24}} - 2^{-\frac{1}{24}}) f_n \tag{2.13}$$

[4]Equal temperament is the dominant tuning system for Western music since the late 18^{th} century. It appeared in opposition to just intonation tuning, in which note intervals are determined as whole number ratios [53].

In its first appearance in [54], the CQT was defined as

$$X_{\mathrm{CQT}}[\kappa] = \frac{1}{\sqrt{L_\kappa}} \sum_{n=0}^{L_\kappa - 1} w_\kappa[n] x[n] \mathrm{e}^{-j \frac{2\pi}{L_\kappa} Q n}, \tag{2.14}$$

where Q is called the quality factor, and $w_\kappa[n]$ and L_κ are respectively a smoothing analysis window and the number of samples used to calculate bin κ of the CQT.

A parallel can be made between the CQT and the DFT formulations. While in the DFT the ratio between bin frequency and bin width is given by the bin index k, since

$$f = \frac{kF_s}{N} \text{ and } \Delta f = \frac{F_s}{N} \implies \frac{f}{\Delta f} = k, \tag{2.15}$$

in the CQT the ratio between the center frequency f_κ of a bin and its span Δf_κ is kept constant and equal to the quality factor Q (hence the name constant-Q transform).

In the equal tempered scale, the ratio $\frac{f_n}{\Delta f_n}$ is constant, and therefore one can use the CQT to build a representation with frequency resolution matching that of the equal tempered scale. Starting from a predetermined maximum frequency $f_{\max}$, one chooses the number of octaves O to be spanned by the CQT. Then, a minimum frequency $f_{\min}$ is calculated as

$$f_{\min} = \frac{f_{\max}}{2^O}. \tag{2.16}$$

From this minimum frequency, the center frequency f_κ of bin κ is determined as

$$f_\kappa = 2^{\frac{\kappa}{B}} f_{\min}, \tag{2.17}$$

where B is the number of bins per octave, and where $\kappa = 0, \ldots, K-1$, with $K = BO$. The frequency band covered by bin κ is given by $\Delta f_\kappa = f_\kappa(2^{\frac{1}{B}} - 2^{-\frac{1}{B}})$. Therefore choosing

$$Q = \frac{1}{2^{\frac{1}{B}} - 2^{-\frac{1}{B}}} \tag{2.18}$$

achieves logarithmic frequency resolution[5].

If, for example, one wishes to create a representation where each bin corresponds to one musical note, then one could choose $B = 12$ (naturally, at the price of a large overlap between note bandwidths). In practice, it is often convenient to pick higher values of B, which makes each bin represent a fraction of a semitone[6].

The need for using a different number of points for each CQT bin comes from the fact that the CQT has frequency-dependent resolution. Lower bins covering smaller

[5]Note that this implies a small overlap between the frequency bands of adjacent bins.

[6]Interval between two adjacent notes in the Western equal tempered system.

frequency bands need longer observation windows in time to accurately represent low-frequency periodical content, whereas higher bins representing wider frequency ranges are able to use shorter windows in time. In the DFT, all bins are calculated using $N = \frac{F_s}{\Delta f}$ signal samples. By analogy, the number of points needed for each CQT bin is $L_\kappa = \frac{F_s}{\Delta f_\kappa} = \frac{F_s Q}{f_\kappa}$. The different windows $w_\kappa[n]$ are used to smooth possible representation artifacts caused by the use of chunks of the signal [54].

It is possible to create a CQT-based time-frequency representation using Equation (2.14). Using a formula similar to the traditional STFT, one has

$$\mathcal{X}_{\mathrm{CQT}}(m, \kappa) = \frac{1}{\sqrt{L_\kappa}} \sum_{n=0}^{L_\kappa - 1} w_\kappa[n] x[n + mH] \mathrm{e}^{-j \frac{2\pi}{L_\kappa} Q n}, \tag{2.19}$$

where the hop size H is set as the smallest window size $\min_\kappa L_\kappa$. The hop size requirement comes from the fact that using $H > \min_\kappa L_\kappa$ would cause some samples to be skipped for the higher frequency bins of the CQT. Choosing the hop as the smallest window size ensures all samples of the signal are covered. Note also that this implementation might require zero-padding in the last frames, so that the larger windows can remain within the signal.

This implementation comes with two downsides. The first is that it is not easily invertible using OLA [52], as was the case with the STFT and NSGT. The second is that it is computationally very expensive. For L-sized frames, bin values in the STFT and NSGT can be calculated using a fast Fourier transform (FFT) algorithm [47] — e.g. the radix-2 algorithm, which has complexity $\mathcal{O}(L \log L)$. This is not possible for the CQT and, together with the short hop size constraint, induces high computational cost.

With those drawbacks in mind, the authors of [51, 52, 55] suggested an alternative algorithm for computing a constant-Q time frequency representation using non-stationary Gabor frames, named constant-Q non-stationary Gabor transform (CQ-NSGT). Their implementation is invertible, and can take advantage of the FFT.

The intuition is as follows. Instead of using windows of different sizes in time to obtain bins with a constant quality factor, $x[n]$ is first transformed to the discrete frequency domain, where constant-Q frequency windows are applied through point-wise multiplication. The inverse DFT of each product describes the time evolution of the signal content within the respective portion of the spectrum. As a consequence, the ensemble of time functions provides a constant-Q time-frequency representation of the signal.

More formally, for a signal $x[n]$ of length N, first an N-point DFT is applied to obtain $X[k]$ of length N in the frequency domain. Then, the inverse DFT is applied on windowed versions of $X[k]$, similarly to the NSGT. The CQ-NSGT is calculated

as [55]

$$\mathcal{X}_{\mathrm{CQT}}(m,\kappa) = \sum_{\ell=0}^{L-1} X[\ell + H_\kappa] W_\kappa[\ell] \mathrm{e}^{j\frac{2\pi}{L}m\ell}, \tag{2.20}$$

where $m = 0, \ldots, L-1$ because of the length of the inverse DFT.

The idea is similar to the NSGT. The windows $W_\kappa[k]$ have size L, but are allowed to have support $S_\kappa \leq L$. The supports and H_κ should be chosen so that all frequency content of $x[n]$ is covered[7]. In addition, S_κ should be chosen to cover the bins within Δf_κ of the κ-th bin of the traditional CQT, and H_κ should be chosen so that the window supports are centered on the center frequency bins f_κ.

One way to determine $W_\kappa[k]$ is as follows [55]. First, calculate the supports S_κ as the number of bins required to span each Δf_κ. This can be done using $S_\kappa = \lceil \frac{N\Delta f_\kappa}{F_s} \rceil$. Then, choose symmetric, real sequences of length S_κ, and pad them with zeros so that they have length $L = \max_\kappa S_\kappa$ to obtain the windows $W_\kappa[k]$. Setting H_κ is similar: first, the bins k nearest to the center frequencies f_κ are found using $k = \lfloor \frac{Nf_\kappa}{F_s} \rceil$[8]; then, each H_κ is set to $H_\kappa = k - \lfloor \frac{L}{2} \rfloor$.

To recover $x[n]$ from $\mathcal{X}_{\mathrm{CQT}}(m,\kappa)$, first $X[k]$ should be reconstructed using an OLA procedure similar to the one used in the NSGT. Then, an N-point inverse DFT can be applied to obtain $x[n]$. In order to perfectly rebuild $X[k]$, all frequency content of $x[n]$ should be covered by the combination of hops and analysis windows. Since the CQT bands do not cover regions $0 \leq f < f_{\min}$ and $f_K < f \leq f_{\max}$, two additional bins should be added covering the parts of $X[k]$ corresponding to these frequency ranges.

Figure 2.9 illustrates the calculation of $\mathcal{X}_{\mathrm{CQT}}(m,\kappa)$ for a real signal $x[n]$ of size $N = 16$, sampled at $F_s = 16$Hz with frequency content until $f_{\max} = 8$Hz using $O = 2$ and $B = 1$. This choice of $f_{\max}$, O, and B leads to $f_1 = 2$Hz, $f_2 = 4$Hz (using $f_\kappa = \frac{f_{\max}}{2^O} \times 2^{\frac{\kappa}{B}}$) and $\Delta f_1 = 3$Hz, $\Delta f_2 = 6$Hz (using $\Delta f_\kappa = f_\kappa(2^{\frac{1}{B}} - 2^{-\frac{1}{B}})$). Converting these values to bins using $k = \lfloor \frac{Nf_\kappa}{F_s} \rceil$ and $S_\kappa = \lceil \frac{N\Delta f_\kappa}{F_s} \rceil$, one finds that the windows $W_1[k]$ and $W_2[k]$ are centered at bins $k = 2$ and $k = 4$, and have supports $S_1 = 3$ and $S_2 = 6$. Windows $W_0[k]$ and $W_3[k]$ cover the rest of the signal's frequency content. In Figure 2.9, samples of $X[k]$ are shown only until the bin corresponding to $f_{\max}$.

For the CQ-NSGT, the corresponding Gabor atoms are defined in frequency. Through analogy with the NSGT, a set $\{G_{m,\kappa}\}$ of atoms of size N can be defined

[7]The DFT of a real (and perfectly reconstructible in continuous time [45, 47]) signal $x[n]$ of length N is conjugate symmetric around $k = \lceil \frac{N}{2} \rceil$ [47]. This is the case for most practical signals, and means that $X[k]$ needs to be covered only for $0 \leq k \leq \lceil \frac{N}{2} \rceil$. Furthermore, such signals have no content above $\frac{F_s}{2}$ [45, 47], and thus k corresponding to $f_{\max}$ is smaller than or equal to $\lceil \frac{N}{2} \rceil$.

[8]In this work, since $\lfloor \cdot \rfloor$ is being used as the rounding down operator, ans since $\lceil \cdot \rceil$ is being used as the rounding up operator, $\lfloor \cdot \rceil$ was adopted as the round to the nearest integer operator.

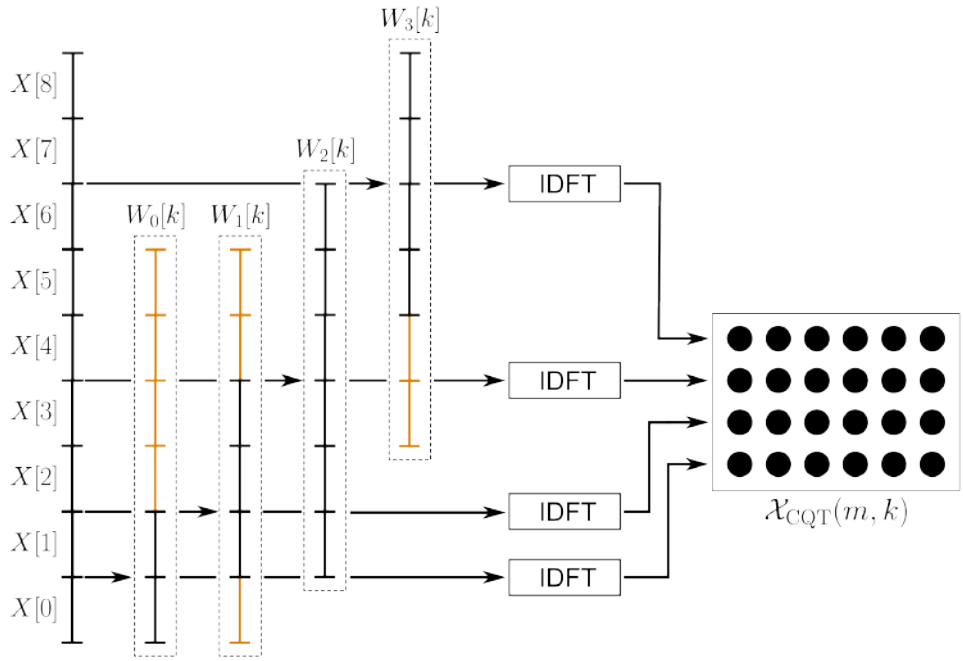

Figure 2.9: Constant-Q non-stationary Gabor transform. Once again, the zero samples in each window are marked in orange. Since $x[n]$ is real, $X[k]$ is symmetric, and all operations can be done using bins zero to eight.

with

$$G_{m,\kappa}[k] = \begin{cases} W_\kappa[k - H_\kappa]\mathrm{e}^{-j\frac{2\pi}{L}mk} & \text{for } H_\kappa \leq k \leq H_\kappa + L - 1, \\ 0 & \text{for } 0 \leq k \leq H_\kappa - 1 \text{ and } H_\kappa + L \leq k \leq N - 1, \end{cases} \tag{2.21}$$

and $\mathcal{X}_{\mathrm{CQT}}(m,\kappa) = \langle \mathbf{X}, \mathbf{G}_{m,\kappa} \rangle = \mathbf{G}_{m,\kappa}^H \mathbf{X}$, where $\mathbf{X} = X[k]$ and $\mathbf{G}_{m,\kappa} = G_{m,\kappa}[k]$[9].

In order to invert $\mathcal{X}_{\mathrm{CQT}}(m,\kappa)$, a procedure similar to that of the NSGT can be used. The DFT is applied to each bin of $\mathcal{X}_{\mathrm{CQT}}(m,k)$ to obtain sequences of size L corresponding to the non-zero samples of the pointwise product between $X[k]$ and $G_\kappa[k] = |G_{m,\kappa}[k]|$. As long as the indexes of those samples are saved beforehand, these products can be padded with $N - L$ zeros to build $K + 1$ sequences of length N. Then, the OLA reconstructed $\hat{X}[k]$ can be written as

$$\hat{X}[k] = \sum_{\kappa=0}^{K+1} X[k] G_\kappa[k] G'_\kappa[k], \tag{2.22}$$

and an inverse DFT of length N can be used to obtain $x[n]$.

The sequences $G'_\kappa[k]$ are synthesis windows created using the same logic of the synthesis windows of NSGT. The combined effect of synthesis an analysis windows

[9]The vectors $\mathbf{G}_{m,\kappa}$ and $\mathbf{X}$ were written with uppercase letters because they were defined in the discrete frequency domain. They should not be confused with matrices.

must make $\hat{X}[k] = X[k]$, and therefore

$$G'_\kappa[k] = \frac{G_\kappa[k]}{\sum_{\kappa=0}^{K+1} G_\kappa[k]^2}. \tag{2.23}$$

Once again, $G'_\kappa[k]$ can be calculated beforehand, since the $W_\kappa[k]$ are known. In summary, inverting $\mathcal{X}_{\mathrm{CQT}}(m, \kappa)$ requires the following steps: application of the DFT on the bins of $\mathcal{X}_{\mathrm{CQT}}(m, \kappa)$, extension of the results to size N, application of $G'_\kappa[k]$, pointwise summation of the results, and application of an inverse DFT of length N on $\hat{X}[k]$. Figure 2.10 illustrates this process for the signal of Figure 2.9.

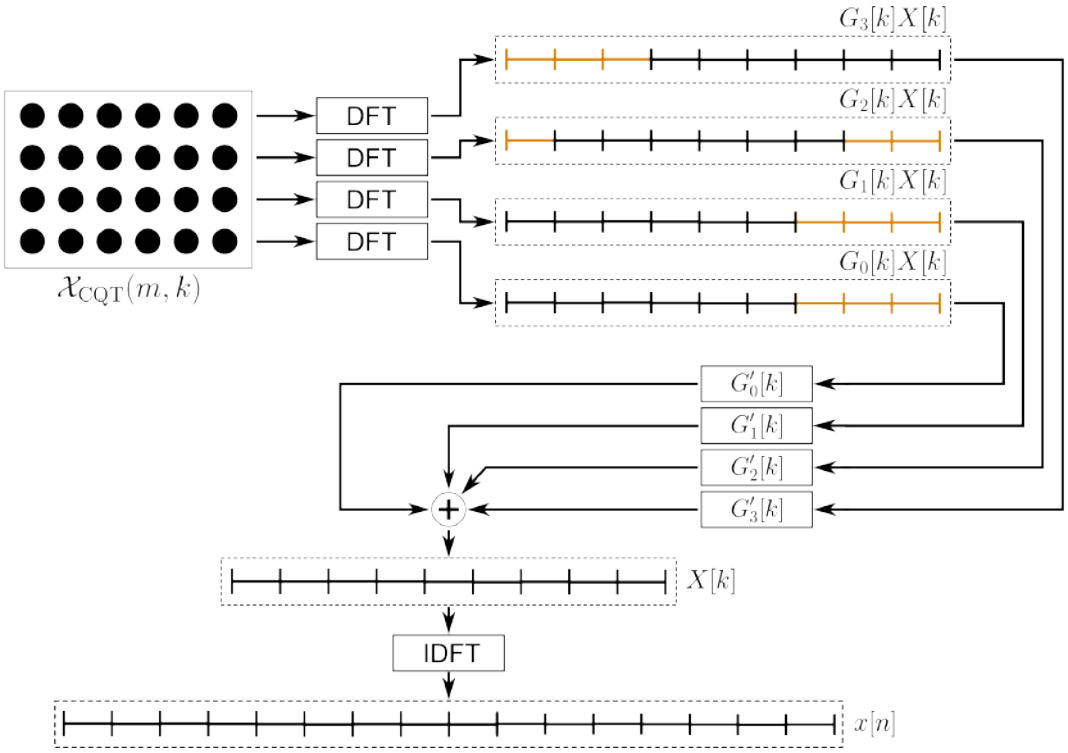

Figure 2.10: Inversion of $\mathcal{X}_{\mathrm{CQT}}(m, \kappa)$ using OLA. The extension samples were marked in orange. Before inverting the DFT, $X[k]$ needs to be mirrored conjugate around its eighth sample (omitting $k = 16$) because of DFT symmetry.

In this work, the CQ-NSGT and its inverse were used as the first and last blocks of the neural networks used for audio restoration. The implementation used was the same of [17], which in turn is a *pytorch* [43] translation of the MATLAB functions *cqt* and *icqt* [56]. These two MATLAB functions were included in version R2018b following the implementations of [51, 52, 55]

The *pytorch* translation comes with two advantages with respect to the original MATLAB version. First, it benefits from GPU acceleration provided by *pytorch*, and therefore runs much faster. Second, it enables adjusting the constant-Q windows used in the CQ-NSGT through backpropagation. This allows for a time-frequency representation that is tailored for a specific application after training.

Following [17], all representations were calculated using Hann windows [47, 50] defined in frequency for each $W_\kappa[k]$. The maximum frequency $f_{\max}$ was set to 22.05kHz, half the sampling rate of the audio signals used, and seven octaves with 64 bins each were used to calculate the center frequencies of the CQ-NSGT.

Figures 2.11 and 2.12 show the CQ-NSGT and STFT of the C4 piano note of Figure 2.1. The constant-Q time-frequency representation was calculated with the parameters of the previous paragraph, and the STFT was calculated using Hann windows with 2048 samples, a hop size of 1024 samples, and 4096-point DFTs. In both figures, the y and x axes were converted from bins and frames to hertz and seconds, respectively. The representations are only shown for frequencies below 8kHz, for better visualization.

Looking at the fundamental frequency of the C4 piano note (which is around 523Hz), it is possible to see that the constant-Q representation is sharper than the STFT at the lower frequency range, but more spread out in time for this part of the spectrum. This is compensated by a better time localization in the upper frequency range. While the STFT represents the signal with a specific time-frequency resolution, the CQ-NSGT tries to use different windows for different parts of the spectrum in order to keep information on the location of events in both time and frequency.

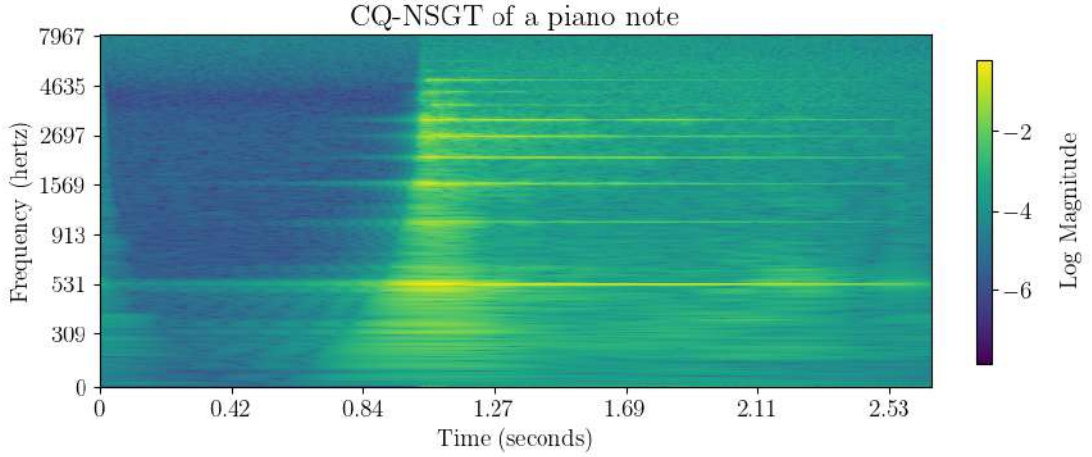

Figure 2.11: CQ-NSGT of a C4 piano note. The CQ-NSGT attempts to solve the time-frequency resolution trade-off by using longer windows (in time) for lower frequencies, and shorter windows (in time) for higher frequencies. The resulting representation maintains sharpness for the lower frequencies, while retaining time localization information.

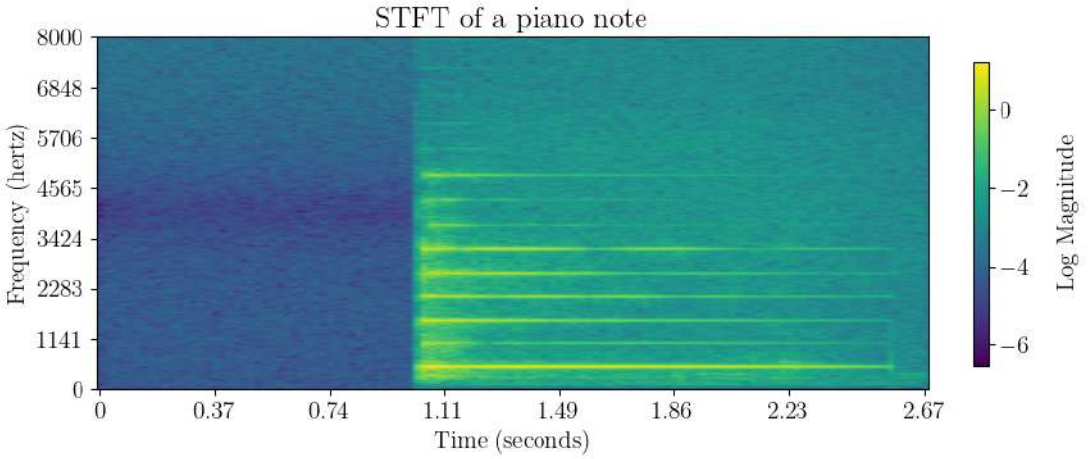

Figure 2.12: STFT of a C4 piano note. It is possible to see that the precise beginning of the note is blurred, and its fundamental frequency is not as well-defined as in the CQ-NSGT.

Chapter 3

Diffusion-Based Generative Models

In machine learning, probabilistic generative models are approaches that try to infer the distribution that originated a dataset [57]. For instance, given a set of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, a generative model would try to estimate the probability density function (PDF) $f(\mathbf{x})$ from which the points are most likely to have come.

In a classification setting, for example, a generative approach would model the distribution $f(\mathbf{x}|\mathcal{C}_k)$ of data point $\mathbf{x}$ given its class $\mathcal{C}_k$, and use it to determine an optimal classification rule for a new point $\mathbf{x}'$. This framework appeared in opposition to discriminative approaches, where $f(\mathcal{C}_k|\mathbf{x})$ is modelled directly, and then used to determine the class of $\mathbf{x}'$.

Recently, generative models received a lot of attention because of their capability to create synthetic data from an inferred distribution. They have achieved state-of-the-art results in artificial image [33, 58] and text [59, 60] creation, and are starting to be used to generate music [61] as well.

Diffusion models are the most recent family of generative models. Their name comes from the physical phenomenon of diffusion, in which particles in a medium spread due to the action of a random external force. During this process, the distribution of the particles changes from $f_0(\mathbf{x})$, at time $t = 0$, to $f_T(\mathbf{x})$, at time $t = T$, and there exists a reverse mathematical process that maps $f_T(\mathbf{x})$ back to $f_0(\mathbf{x})$.

Similarly, diffusion-based models are trained by gradually converting the data distribution $f(\mathbf{x}) = f_0(\mathbf{x})$ into a PDF $f_T(\mathbf{x})$, and learning the reverse process. If $f_T(\mathbf{x})$ is known, sampling from $f(\mathbf{x})$ after training becomes simply a matter of sampling from $f_T(\mathbf{x})$ and using the model to revert the diffusion process.

Much like generative adversarial networks (GANs) and variational autoencoders (VAEs), diffusion-based models maximize the likelihood of their estimate of $f(\mathbf{x})$. However, their training is more stable than that of GANs [62], and they do not assume normality of the data distribution given the latents as in VAEs [36].

3.1 Denoising Diffusion Probabilistic Models

Diffusion models first appeared in 2015 [63], but became popular with denoising diffusion probabilistic models (DDPM), as implemented in [5].

In DDPM, each sample $\mathbf{x}_0$ of a data set is gradually corrupted for $i = 0, \ldots, N$ steps according to the recurrence

$$\mathbf{x}_{i+1} = \sqrt{1-\beta_i}\mathbf{x}_i + \sqrt{\beta_i}\mathbf{z}, \tag{3.1}$$

with $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. The values β_i are a sequence of values called the variance schedule, and they are chosen such that the terminal distribution $f_N(\mathbf{x}_N)$ of the data set after diffusion is approximately normal. Equation (3.1) defines a diffusion process, called the *forward diffusion* process, which maps the initial data distribution $f_0(\mathbf{x}_0)$ into a terminal distribution $f_N(\mathbf{x}_N)$. It is worth mentioning that the forward diffusion process constitutes a Markov chain, since

$$f_{i+1}(\mathbf{x}_{i+1}|\mathbf{x}_i, \ldots, \mathbf{x}_0) = f_{i+1}(\mathbf{x}_{i+1}|\mathbf{x}_i) = \mathcal{N}(\sqrt{1-\beta_i}\mathbf{x}_i, \beta_i\mathbf{I}). \tag{3.2}$$

The *reverse diffusion* process, which maps $f_N(\mathbf{x}_N)$ to $f_0(\mathbf{x}_0)$, is unknown, but could be described through distributions $f_i(\mathbf{x}_i|\mathbf{x}_{i+1})$. DDPM tries to approximate this reverse step distribution with a model $p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{x}_{i+1}) = \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{\theta}}_{i+1}, \boldsymbol{\Sigma}^{\boldsymbol{\theta}}_{i+1})$, where $\{\boldsymbol{\mu}^{\boldsymbol{\theta}}_{i+1}, \boldsymbol{\Sigma}^{\boldsymbol{\theta}}_{i+1}\}$ are the outputs of some function with inputs $\{\mathbf{x}_{i+1},\ i+1\}$, and $\boldsymbol{\theta}$ represents a set of parameters to be optimized. Figure 3.1 illustrates the forward and reverse processes.

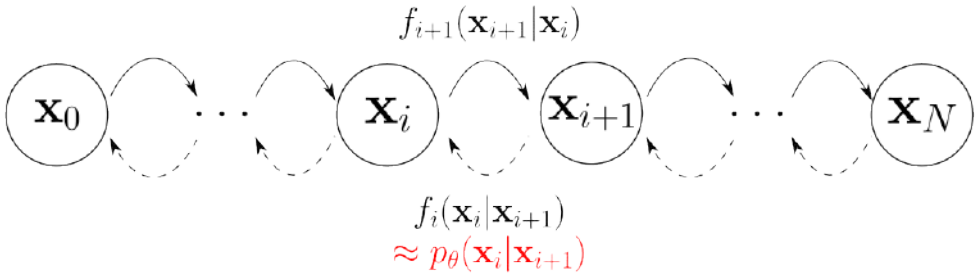

Figure 3.1: Diffusion process Markov chain. Although the reverse process is unknown, it will be approximated step by step using $p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{x}_{i+1})$. Adapted from [5].

Optimization of parameters $\boldsymbol{\theta}$ can be done by minimizing the cross-entropy be-

tween $f_0(\mathbf{x}_0)$ and its approximation $p_{\boldsymbol{\theta}}(\mathbf{x}_0)$, given by

$$\begin{aligned}
H[f_0, p_{\boldsymbol{\theta}}] &= \mathbb{E}_{f_0}[-\log p_{\boldsymbol{\theta}}(\mathbf{x}_0)] \\
&= \mathbb{E}_{f_0}\left[-\log \int_{\mathbb{R}^N} p_{\boldsymbol{\theta}}(\mathbf{x}_0, \dots, \mathbf{x}_N)\, \mathrm{d}\mathbf{x}_1 \dots \mathrm{d}\mathbf{x}_N\right] \\
&= \mathbb{E}_{f_0}\left[-\log \int_{\mathbb{R}^N} \frac{f_{1,\dots,N}(\mathbf{x}_1, \dots, \mathbf{x}_N|\mathbf{x}_0)}{f_{1,\dots,N}(\mathbf{x}_1, \dots, \mathbf{x}_N|\mathbf{x}_0)} p_{\boldsymbol{\theta}}(\mathbf{x}_0, \dots, \mathbf{x}_N)\, \mathrm{d}\mathbf{x}_1 \dots \mathrm{d}\mathbf{x}_N\right] \\
&= \mathbb{E}_{f_0}\left[-\log \mathbb{E}_{f_{1,\dots,N|0}}\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x}_0, \dots, \mathbf{x}_N)}{f_{1,\dots,N}(\mathbf{x}_1, \dots, \mathbf{x}_N|\mathbf{x}_0)}\right]\right] \\
&\leq -\mathbb{E}_{f_{0,\dots,N}}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_0, \dots, \mathbf{x}_N)}{f_{1,\dots,N}(\mathbf{x}_1, \dots, \mathbf{x}_N|\mathbf{x}_0)}\right] = L_{\mathrm{VLB}}. \qquad (3.3)
\end{aligned}$$

The last step of Equation (3.3) comes from Jensen's inequality, and $-L_{\mathrm{VLB}}$ is the variational lower bound [57] for the likelihood of $\boldsymbol{\theta}$.

In [63] the authors showed that L_{VLB} can be rewritten in a simpler form for optimization. First of all, the Markov property of the forward and reverse diffusion processes can be used to write the distributions in the last line of Equation (3.3) as

$$\begin{aligned}
p_{\boldsymbol{\theta}}(\mathbf{x}_0, \dots, \mathbf{x}_N) &= p_{\boldsymbol{\theta}}(\mathbf{x}_N) \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i) = f_N(\mathbf{x}_N) \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i), \\
f_{1,\dots,N}(\mathbf{x}_1, \dots, \mathbf{x}_N|\mathbf{x}_0) &= \prod_{i=1}^{N} f_i(\mathbf{x}_i|\mathbf{x}_{i-1}). \qquad (3.4)
\end{aligned}$$

Note that $p_{\boldsymbol{\theta}}(\mathbf{x}_N) = f_N(\mathbf{x}_N)$ is the starting point of the reverse diffusion process.

Using this and inverting the fraction inside the expectation with the initial minus sign, L_{VLB} can be reduced to three terms:

$$\begin{aligned}
L_{\mathrm{VLB}} &= \mathbb{E}_{f_{0,\dots,N}}\left[\log \frac{\prod_{i=1}^{N} f_i(\mathbf{x}_i|\mathbf{x}_{i-1})}{p_{\boldsymbol{\theta}}(\mathbf{x}_N) \prod_{i=1}^{N} p(\mathbf{x}_{i-1}|\mathbf{x}_i)}\right] \\
&= \mathbb{E}_{f_{0,\dots,N}}\left[-\log p_{\boldsymbol{\theta}}(\mathbf{x}_N) + \sum_{i=1}^{N} \log \frac{f_i(\mathbf{x}_i|\mathbf{x}_{i-1})}{p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i)}\right] \\
&= \mathbb{E}_{f_{0,\dots,N}}\left[-\log p_{\boldsymbol{\theta}}(\mathbf{x}_N) + \sum_{i=2}^{N} \log \frac{f_i(\mathbf{x}_i|\mathbf{x}_{i-1})}{p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i)} + \log \frac{f_1(\mathbf{x}_1|\mathbf{x}_0)}{p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1)}\right]. \qquad (3.5)
\end{aligned}$$

Then, using Bayes' theorem and the Markov property, $f_i(\mathbf{x}_i|\mathbf{x}_{i-1})$ can be written as

$$f_i(\mathbf{x}_i|\mathbf{x}_{i-1}) = f_i(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{x}_0) = \frac{f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_i, \mathbf{x}_0) f_i(\mathbf{x}_i|\mathbf{x}_0)}{f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_0)}, \qquad (3.6)$$

and it is possible to further develop the middle summation as

$$\begin{aligned}\sum_{i=2}^{N}\log\frac{f_i(\mathbf{x}_i|\mathbf{x}_{i-1})}{p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i)} &= \sum_{i=2}^{N}\log\left(\frac{f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_i,\mathbf{x}_0)}{p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i)}\cdot\frac{f_i(\mathbf{x}_i|\mathbf{x}_0)}{f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_0)}\right)\\ &= \sum_{i=2}^{N}\log\frac{f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_i,\mathbf{x}_0)}{p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i)}+\log\prod_{i=2}^{N}\frac{f_i(\mathbf{x}_i|\mathbf{x}_0)}{f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_0)}\\ &= \sum_{i=2}^{N}\log\frac{f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_i,\mathbf{x}_0)}{p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i)}+\log\frac{f_N(\mathbf{x}_N|\mathbf{x}_0)}{f_1(\mathbf{x}_1|\mathbf{x}_0)},\end{aligned}\tag{3.7}$$

and combine the first and last terms of Equation (3.5) with the numerator and denominator of the last term of Equation (3.7) to obtain

$$L_{\text{VLB}}=\mathbb{E}_{f_{0,\dots,N}}\left[\log\frac{f_N(\mathbf{x}_N|\mathbf{x}_0)}{p_{\boldsymbol{\theta}}(\mathbf{x}_N)}+\sum_{i=2}^{N}\log\frac{f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_i,\mathbf{x}_0)}{p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i)}-\log p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1)\right].\tag{3.8}$$

The distribution $f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_i,\mathbf{x}_0)$ is Gaussian because of the nature of incremental perturbations of the diffusion process, and it is possible to show that its mean and variance have a closed form [5]. Reordering Equation (3.6) one has

$$f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_i,\mathbf{x}_0)=f_i(\mathbf{x}_i|\mathbf{x}_{i-1},\mathbf{x}_0)\frac{f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_0)}{f_i(\mathbf{x}_i|\mathbf{x}_0)}.\tag{3.9}$$

In addition, through recursive application of Equation (3.1), it is possible to show that $f_i(\mathbf{x}_i|\mathbf{x}_0)=\mathcal{N}(\sqrt{\bar{\alpha}_i}\mathbf{x}_0,(1-\bar{\alpha}_i)\mathbf{I})$, where $\bar{\alpha}_i=\prod_{k=1}^{i}(1-\beta_k)$. Using this and Equation (3.9), and noting that all distributions involved are Gaussians with covariance matrices proportional to the identity matrix, one can write that

$$\begin{aligned}f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_i,\mathbf{x}_0) &\propto \mathrm{e}^{-\frac{1}{2}\left[\frac{||\mathbf{x}_i-\sqrt{1-\beta_i}\mathbf{x}_{i-1}||^2}{\beta_i}+\frac{||\mathbf{x}_{i-1}-\sqrt{\bar{\alpha}_{i-1}}\mathbf{x}_0||^2}{1-\bar{\alpha}_{i-1}}-\frac{||\mathbf{x}_i-\sqrt{\bar{\alpha}_i}\mathbf{x}_0||^2}{1-\bar{\alpha}_i}\right]}\\ &= \mathrm{e}^{-\frac{1}{2}\left[\frac{||\mathbf{x}_i||^2-2\sqrt{1-\beta_i}\mathbf{x}_{i-1}^T\mathbf{x}_i+(1-\beta_i)||\mathbf{x}_{i-1}||^2}{\beta_i}+\frac{||\mathbf{x}_{i-1}||^2-2\sqrt{\bar{\alpha}_{i-1}}\mathbf{x}_{i-1}^T\mathbf{x}_0+\bar{\alpha}_{i-1}||\mathbf{x}_0||^2}{1-\bar{\alpha}_{i-1}}-\frac{||\mathbf{x}_i||^2-2\sqrt{\bar{\alpha}_i}\mathbf{x}_i^T\mathbf{x}_0+\bar{\alpha}_i||\mathbf{x}_0||^2}{1-\bar{\alpha}_i}\right]}\\ &= \mathrm{e}^{-\frac{1}{2}\left[\left(\frac{1-\beta_i}{\beta_i}+\frac{1}{1-\bar{\alpha}_{i-1}}\right)||\mathbf{x}_{i-1}||^2-2\mathbf{x}_{i-1}^T\left(\frac{\sqrt{1-\beta_i}}{\beta_i}\mathbf{x}_i+\frac{\sqrt{\bar{\alpha}_{i-1}}}{1-\bar{\alpha}_{i-1}}\mathbf{x}_0\right)+\frac{(1-\bar{\alpha}_i)-\beta_i}{\beta_i(1-\bar{\alpha}_i)}||\mathbf{x}_i||^2+\frac{2\sqrt{\bar{\alpha}_i}\mathbf{x}_i^T\mathbf{x}_0}{1-\bar{\alpha}_i}+\left(\frac{\bar{\alpha}_{i-1}}{1-\bar{\alpha}_{i-1}}-\frac{\bar{\alpha}_i}{1-\bar{\alpha}_i}\right)||\mathbf{x}_0||^2\right]}\\ &= \mathrm{e}^{-\frac{1}{2}\left[\frac{1-\bar{\alpha}_i}{(1-\bar{\alpha}_{i-1})\beta_i}||\mathbf{x}_{i-1}||^2-2\mathbf{x}_{i-1}^T\left(\frac{\sqrt{1-\beta_i}}{\beta_i}\mathbf{x}_i+\frac{\sqrt{\bar{\alpha}_{i-1}}}{1-\bar{\alpha}_{i-1}}\mathbf{x}_0\right)+\frac{1-\bar{\alpha}_i}{(1-\bar{\alpha}_{i-1})\beta_i}\left\|\frac{\sqrt{1-\beta_i}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i}\mathbf{x}_i+\frac{\sqrt{\bar{\alpha}_{i-1}}\beta_i}{1-\bar{\alpha}_i}\mathbf{x}_0\right\|^2\right]}.\end{aligned}\tag{3.10}$$

Furthermore, remembering that a d-dimensional Gaussian random variable $\mathbf{x}$ of mean $\boldsymbol{\mu}$ and variance $\sigma^2\mathbf{I}$ has PDF

$$\mathcal{N}(\boldsymbol{\mu},\sigma^2\mathbf{I})=\frac{1}{(2\pi\sigma)^{\frac{d}{2}}}\mathrm{e}^{-\frac{||\mathbf{x}-\boldsymbol{\mu}||^2}{2\sigma^2}}=\frac{1}{(2\pi\sigma)^{\frac{d}{2}}}\mathrm{e}^{-\frac{1}{2}\left[\frac{||\mathbf{x}||^2}{\sigma^2}-\frac{2\mathbf{x}^T\boldsymbol{\mu}}{\sigma^2}+\frac{||\boldsymbol{\mu}||^2}{\sigma^2}\right]},\tag{3.11}$$

it is possible to say, through comparison of the Gaussian PDF with the last line of Equation (3.10), that $f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_i, \mathbf{x}_0) = \mathcal{N}(\tilde{\boldsymbol{\mu}}_i, \tilde{\beta}_i \mathbf{I})$, where

$$\begin{aligned}
\tilde{\beta}_i &= \left(\frac{1-\beta_i}{\beta_i} + \frac{1}{1-\bar{\alpha}_{i-1}}\right)^{-1} = \frac{1-\bar{\alpha}_{i-1}}{1-\bar{\alpha}_i}\beta_i, \\
\frac{\tilde{\boldsymbol{\mu}}_i}{\tilde{\beta}_i} &= \left(\frac{\sqrt{1-\beta_i}}{\beta_i}\mathbf{x}_i + \frac{\sqrt{\bar{\alpha}_{i-1}}}{1-\bar{\alpha}_{i-1}}\mathbf{x}_0\right), \\
\tilde{\boldsymbol{\mu}}_i &= \left(\frac{\sqrt{1-\beta_i}}{\beta_i}\mathbf{x}_i + \frac{\sqrt{\bar{\alpha}_{i-1}}}{1-\bar{\alpha}_{i-1}}\mathbf{x}_0\right)\tilde{\beta}_i \\
&= \frac{\sqrt{1-\beta_i}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i}\mathbf{x}_i + \frac{\sqrt{\bar{\alpha}_{i-1}}\beta_i}{1-\bar{\alpha}_i}\mathbf{x}_0.
\end{aligned} \tag{3.12}$$

The summands of Equation (3.8) can be further simplified by marginalizing the variables that are not used in the expected values. In the end, the result is $L_{\text{VLB}} = L_N + \sum_{i=2}^{N} L_{i-1} + L_0$, where L_N, L_{i-1}, L_0 are given by

$$\begin{aligned}
L_N &= \mathbb{E}_{f_{0,\ldots,N}}\left[\log \frac{f_N(\mathbf{x}_N|\mathbf{x}_0)}{p_{\boldsymbol{\theta}}(\mathbf{x}_N)}\right] = \mathbb{E}_{f_{0,N}}\left[\log \frac{f_N(\mathbf{x}_N|\mathbf{x}_0)}{p_{\boldsymbol{\theta}}(\mathbf{x}_N)}\right] \\
&= \mathbb{E}_{f_0}\left[\mathbb{E}_{f_{N|0}}\left[\log \frac{f_N(\mathbf{x}_N|\mathbf{x}_0)}{p_{\boldsymbol{\theta}}(\mathbf{x}_N)}\right]\right] = \mathbb{E}_{f_0}\left[D_{\text{KL}}(f_N(\mathbf{x}_N|\mathbf{x}_0)||p_{\boldsymbol{\theta}}(\mathbf{x}_N))\right],
\end{aligned} \tag{3.13}$$

$$\begin{aligned}
L_{i-1} &= \mathbb{E}_{f_{0,\ldots,N}}\left[\log \frac{f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_i, \mathbf{x}_0)}{p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i)}\right] \\
&= \mathbb{E}_{f_{0,i-1,i}}\left[\log \frac{f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_i, \mathbf{x}_0)}{p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i)}\right] = \mathbb{E}_{f_{0,i}}\left[\mathbb{E}_{f_{i-1|0,i}}\left[\log \frac{f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_i, \mathbf{x}_0)}{p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i)}\right]\right] \\
&= \mathbb{E}_{f_{0,i}}\left[D_{\text{KL}}(f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_i, \mathbf{x}_0)||p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i))\right],
\end{aligned} \tag{3.14}$$

$$L_0 = \mathbb{E}_{f_{0,\ldots,N}}\left[-\log p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1)\right] = -\,\mathbb{E}_{f_{0,1}}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1)\right], \tag{3.15}$$

where $D_{\text{KL}}(f||p) = \mathbb{E}_f\left[\log \frac{f(\mathbf{x})}{p(\mathbf{x})}\right]$ is the Kullback-Leibler (KL) divergence [57] from distribution f to distribution p.

This simplification enables finding the parameters $\boldsymbol{\theta}$ that minimize L_{VLB} for a few reasons. Firstly, because the only part of L_N that depends on $\boldsymbol{\theta}$ is $p_{\boldsymbol{\theta}}(\mathbf{x}_N)$, there is no need to optimize this term, since $p_{\boldsymbol{\theta}}(\mathbf{x}_N) = f_N(\mathbf{x}_N)$, which is known. Furthermore, the middle term L_{i-1} can be calculated adopting $\boldsymbol{\Sigma}_i^{\boldsymbol{\theta}} = \beta_i \mathbf{I}$ and using

the formula for the KL divergence between two Gaussians[1], which results in

$$L_{i-1} = \mathbb{E}_{f_{0,i}}\left[\frac{1}{2\beta_i}||\tilde{\boldsymbol{\mu}}_i - \boldsymbol{\mu}_i^{\boldsymbol{\theta}}||^2\right] + C, \tag{3.17}$$

where C is a term that can be ignored for optimization, as it depends only on β_i and $\tilde{\beta}_i$[2] . Finally, if β_0 is taken to be zero, then $\bar{\alpha}_0 = 1$, and $\tilde{\boldsymbol{\mu}}_1 = \mathbf{x}_0$, which implies that L_0 can be calculated in the same way as L_{i-1}.

The authors of [5] suggest a different parameterization of Equation (3.17). Because $f_i(\mathbf{x}_i|\mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_i}\mathbf{x}_0, (1-\bar{\alpha}_i)\mathbf{I})$, it is possible to write

$$\mathbf{x}_i = \sqrt{\bar{\alpha}_i}\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_i)}\mathbf{z} \implies \mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_i}}\left(\mathbf{x}_i - \sqrt{(1-\bar{\alpha}_i)}\mathbf{z}\right), \tag{3.18}$$

with $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. When $\mathbf{x}_0$ is substituted in this form into $\tilde{\boldsymbol{\mu}}_i$, Equation (3.17) can be rewritten as

$$\begin{aligned} L_{i-1} &= \mathbb{E}_{f_{0,i}}\left[\frac{1}{2\beta_i}\left|\left|\frac{1}{\sqrt{1-\beta_i}}\left(\mathbf{x}_i - \frac{\beta_i}{\sqrt{1-\bar{\alpha}_i}}\mathbf{z}\right) - \boldsymbol{\mu}_i^{\boldsymbol{\theta}}\right|\right|^2\right] + C \\ &= \mathbb{E}_{f_0,\mathcal{N}(0,\mathbf{I})}\left[\frac{1}{2\beta_i}\left|\left|\frac{1}{\sqrt{1-\beta_i}}\left(\mathbf{x}_i - \frac{\beta_i}{\sqrt{1-\bar{\alpha}_i}}\mathbf{z}\right) - \boldsymbol{\mu}_i^{\boldsymbol{\theta}}\right|\right|^2\right] + C, \end{aligned} \tag{3.19}$$

where the change in the expectation in the second line comes from the fact that $\mathbf{x}_i$ is a function of $\mathbf{x}_0$ and $\mathbf{z}$. Since $\boldsymbol{\mu}_i^{\boldsymbol{\theta}}$ is an arbitrary function of $\{\mathbf{x}_i, i\}$, in [5] the authors propose adopting $\boldsymbol{\mu}_i^{\boldsymbol{\theta}} = \frac{1}{\sqrt{1-\beta_i}}\left(\mathbf{x}_i - \frac{\beta_i}{\sqrt{1-\bar{\alpha}_i}}\mathbf{z}_i^{\boldsymbol{\theta}}\right)$, where now $\mathbf{z}_i^{\boldsymbol{\theta}}$ is the function of $\{\mathbf{x}_i, i\}$ to be approximated. Ignoring the constant C, this allows writing $L_{i-1} = \mathbb{E}_{f_0,\mathcal{N}(0,\mathbf{I})}\left[k_i\left|\left|\mathbf{z} - \mathbf{z}_i^{\boldsymbol{\theta}}\right|\right|^2\right]$, with k_i being a function of β_i and $\bar{\alpha}_i$. Removing k_i, and considering the summation of all L_{i-1} as an expected value over the indexes, the final cost function of the original DDPM becomes

$$L_{\text{DDPM}}(\boldsymbol{\theta}) = \mathbb{E}_i\left[\mathbb{E}_{f_0,\mathcal{N}(0,\mathbf{I})}\left[\left|\left|\mathbf{z} - \mathbf{z}_i^{\boldsymbol{\theta}}\right|\right|^2\right]\right]. \tag{3.20}$$

The authors show that this simplified formulation has better performance than the formulation with $\boldsymbol{\mu}_i^{\boldsymbol{\theta}}$ for image generation [3].

Training a neural network to calculate $\mathbf{z}_i^{\boldsymbol{\theta}}$ is possible through stochastic gradient

[1]For two d-dimensional Gaussians $g_1 = \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $g_2 = \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$

$$D_{\text{KL}}(g_1||g_2) = \frac{1}{2}\left[\log\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} - d + \mathbf{tr}\{\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\} + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T\right], \tag{3.16}$$

where $|\cdot|$ denotes the determinant and $\mathbf{tr}\{\cdot\}$ denotes the trace operator.

[2]Observe that C can only be ignored because the hypothesis $\boldsymbol{\Sigma}_i^{\boldsymbol{\theta}} = \beta_i\mathbf{I}$ was made. If generic $\boldsymbol{\Sigma}_i^{\boldsymbol{\theta}}$ was assumed, the remaining terms of $D_{\text{KL}}(f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_i, \mathbf{x}_0)||p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i))$ would also have to be considered in L_{i-1}.

[3] According to the FID (Fréchet inception distance) objective image quality metric [64]. Other comparative results for image generation mentioned in this work also use this metric.

descent (SGD) [57] with Algorithm 3.1. In Algorithm 3.1, a data point $\mathbf{x}_0$ from $f_0(\mathbf{x_0})$, a noise sample $\mathbf{z}$ and a diffusion step i are sampled at each iteration, and the neural network is used with the current parameters $\boldsymbol{\theta}_{\text{old}}$ to calculate an estimate $\mathbf{z}_i^{\boldsymbol{\theta}}$ of $\mathbf{z}$ based on a degraded version $\mathbf{x}_i$ of $\mathbf{x}_0$ at diffusion step i. The gradient of the squared estimation error is then used to update $\boldsymbol{\theta}_{\text{old}}$ according to a learning rate η [57].

Algorithm 3.1: DDPM Training

while $||\boldsymbol{\theta}_{\text{new}} - \boldsymbol{\theta}_{\text{old}}||^2 \geq$ tolerance **do**
  $\mathbf{x}_0 \sim f_0(\mathbf{x}_0)$;
  $i \sim \mathcal{U}(\{0, \ldots, N\})$;
  $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$;
  $\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_{\text{old}} - \eta \nabla_{\boldsymbol{\theta}} \left|\left|\mathbf{z} - \mathbf{z}_i^{\boldsymbol{\theta}}(\mathbf{x}_i(\mathbf{x}_0, i), i)\right|\right|^2$;
end
return $\boldsymbol{\theta}_{\text{new}}$;

With a function approximator for $\mathbf{z}_i^{\boldsymbol{\theta}}$, sampling from $f_0(\mathbf{x}_0)$ is relatively simple. Starting from $p_{\boldsymbol{\theta}}(\mathbf{x}_N) = \mathcal{N}(0, \mathbf{I})$, the reverse diffusion model $p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i) = \mathcal{N}(\boldsymbol{\mu}_i^{\boldsymbol{\theta}}, \beta_i \mathbf{I})$ can be used iteratively from N down to 0 with the recurrence $\mathbf{x}_{i-1} = \boldsymbol{\mu}_i^{\boldsymbol{\theta}} + \sqrt{\beta_i}\mathbf{z}$, $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. Algorithm 3.2 shows this: it starts with a Gaussian sample corresponding to $\mathbf{x}_N$, and performs reverse diffusion using the recurrence for $\mathbf{x}_{i-1}$ with $\boldsymbol{\mu}_i^{\boldsymbol{\theta}}$ replaced by its formulation as a function of $\mathbf{z}_i^{\boldsymbol{\theta}}$, and β_i and $\bar{\alpha}_i$ as predefined by the diffusion variance schedule.

Algorithm 3.2: DDPM Sampling

$\mathbf{x} \sim \mathcal{N}(0, \mathbf{I})$;
for $i = N : 0$ **do**
  $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$;
  $\mathbf{x} = \frac{1}{\sqrt{1-\beta_i}} \left(\mathbf{x}_i - \frac{\beta_i}{\sqrt{1-\bar{\alpha}_i}} \mathbf{z}_i^{\boldsymbol{\theta}}\right) + \sqrt{\beta_i}\mathbf{z}$;
end
return $\mathbf{x}$;

Given a noisy sample and a time index, denoising diffusion probabilistic models are trained to find out the amount of noise introduced during the forward diffusion process. This information is used to estimate a mean value for the sample one step back during the reverse diffusion process, which enables sampling when used iteratively. There are variations to the original DDPM model [65, 66], but they mostly enhance the sampling process, using less reverse diffusion steps to generate samples faster. In further sections, a more mathematical interpretation of diffusion models will be presented, along with some heuristics that will improve both training and sampling of DDPM.

3.2 Score Matching with Langevin Dynamics

Models using score matching with Langevin dynamics (SMLD) initially appeared in 2019 [6] as a novel approach for generative modelling of data distributions. They were later proven to be equivalent to diffusion models [5, 7], and are the starting point for the interpretation that will be given in Section 3.3.

Langevin dynamics is a Markov chain Monte Carlo (MCMC) approach for approximating a desired distribution $f(\mathbf{x})$ from an initial distribution $f_0(\mathbf{x})$. If $\nabla_{\mathbf{x}} \log f(\mathbf{x})$ is available, then it is possible to show [27, 67] that the recurrence

$$\mathbf{x}^{m+1} = \mathbf{x}^m + \epsilon \nabla_{\mathbf{x}} \log f(\mathbf{x}) + \sqrt{2\epsilon}\mathbf{z}, \tag{3.21}$$

with $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$, and $\epsilon > 0$, constitutes a Markov process that converts samples $\mathbf{x}^0 \sim f_0(\mathbf{x})$ into samples $\mathbf{x} \sim f(\mathbf{x})$, as long as enough steps are applied.

For real world data, $\nabla_{\mathbf{x}} \log f(\mathbf{x})$ is unknown most of the times, but it can be estimated through *denoising score matching* [68]. Denoising score matching is a technique for estimating the score function $\nabla_{\mathbf{x}} \log f(\mathbf{x})$ by minimizing the quantity

$$L_{\text{SM}} = \mathbb{E}_{f_{\tilde{\mathbf{x}},\mathbf{x}}} \left[\left|\left| s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log f_{\tilde{\mathbf{x}}|\mathbf{x}}(\tilde{\mathbf{x}}|\mathbf{x}) \right|\right|^2 \right], \tag{3.22}$$

where $s^{\boldsymbol{\theta}}(\cdot)$ is a function approximator parameterized by $\boldsymbol{\theta}$, and where $\tilde{\mathbf{x}}$ is a randomly perturbed version of data point $\mathbf{x}$. The distribution $f_{\tilde{\mathbf{x}}|\mathbf{x}}(\tilde{\mathbf{x}}|\mathbf{x})$ is usually chosen to be $\mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I})$, with σ being the perturbation level. This makes calculating L_{SM} using Monte Carlo straightforward, as sampling from $f_{\tilde{\mathbf{x}},\mathbf{x}}(\tilde{\mathbf{x}}, \mathbf{x})$ requires only applying Gaussian noise to a sample from the data set.

In [68], the authors show that optimizing L_{SM} makes $s^{\boldsymbol{\theta}}(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log f(\mathbf{x})$. The proof is as follows. The distribution $f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}})$ corresponding to a set of perturbed versions of the original data points $\mathbf{x}$ should have approximately the same shape as $f_{\mathbf{x}}(\mathbf{x})$, as long as the perturbation is small. Therefore, minimizing

$$L_{\text{noisy}} = \mathbb{E}_{f_{\tilde{\mathbf{x}}}} \left[\left|\left| s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}) \right|\right|^2 \right] \tag{3.23}$$

should make $s^{\boldsymbol{\theta}}(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log f(\mathbf{x})$. Equation (3.23) can be expanded as

$$L_{\text{noisy}} = \mathbb{E}_{f_{\tilde{\mathbf{x}}}} \left[\left|\left| s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}}) \right|\right|^2 \right] - 2\,\mathbb{E}_{f_{\tilde{\mathbf{x}}}} \left[s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}})^T \nabla_{\tilde{\mathbf{x}}} \log f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}) \right] + C_1, \tag{3.24}$$

where $C_1 = \mathbb{E}_{f_{\tilde{\mathbf{x}}}} \left[||\nabla_{\tilde{\mathbf{x}}} \log f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}})||^2 \right]$ is not dependent on $\boldsymbol{\theta}$, and where the middle

term $\mathbb{E}_{f_{\tilde{\mathbf{x}}}}\left[s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}})^T \nabla_{\tilde{\mathbf{x}}} \log f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}})\right]$ can be rewritten as

$$\begin{aligned}
\mathbb{E}_{f_{\tilde{\mathbf{x}}}}\left[s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}})^T \nabla_{\tilde{\mathbf{x}}} \log f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}})\right] &= \int_{\mathbb{R}^N} f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}) \left[s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}})^T \nabla_{\tilde{\mathbf{x}}} \log f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}})\right] \mathrm{d}\tilde{\mathbf{x}} \\
&= \int_{\mathbb{R}^N} s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}})^T \nabla_{\tilde{\mathbf{x}}} f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}) \mathrm{d}\tilde{\mathbf{x}} \\
&= \int_{\mathbb{R}^N} s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}})^T \nabla_{\tilde{\mathbf{x}}} \left[\int_{\mathbb{R}^N} f(\mathbf{x}) f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}|\mathbf{x}) \mathrm{d}\mathbf{x}\right] \mathrm{d}\tilde{\mathbf{x}} \\
&= \int_{\mathbb{R}^N} s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}})^T \left[\int_{\mathbb{R}^N} f(\mathbf{x}) f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}|\mathbf{x}) \nabla_{\tilde{\mathbf{x}}} \log f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}|\mathbf{x}) \mathrm{d}\mathbf{x}\right] \mathrm{d}\tilde{\mathbf{x}} \\
&= \int_{\mathbb{R}^N} \int_{\mathbb{R}^N} f(\mathbf{x}) f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}|\mathbf{x}) s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}})^T \nabla_{\tilde{\mathbf{x}}} \log f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}|\mathbf{x}) \mathrm{d}\mathbf{x}\, \mathrm{d}\tilde{\mathbf{x}} \\
&= \mathbb{E}_{f_{\tilde{\mathbf{x}},\mathbf{x}}}\left[s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}})^T \nabla_{\tilde{\mathbf{x}}} \log f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}|\mathbf{x})\right]. \qquad (3.25)
\end{aligned}$$

Thus,

$$L_{\text{noisy}} = \mathbb{E}_{f_{\tilde{\mathbf{x}}}}\left[\left|\left|s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}})\right|\right|^2\right] - 2\,\mathbb{E}_{f_{\tilde{\mathbf{x}},\mathbf{x}}}\left[s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}})^T \nabla_{\tilde{\mathbf{x}}} \log f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}|\mathbf{x})\right] + C_1, \qquad (3.26)$$

which has the same form of the expansion of L_{SM}

$$\begin{aligned}
L_{\text{SM}} &= \mathbb{E}_{f_{\tilde{\mathbf{x}},\mathbf{x}}}\left[\left|\left|s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}})\right|\right|^2\right] - 2\,\mathbb{E}_{f_{\tilde{\mathbf{x}},\mathbf{x}}}\left[s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}})^T \nabla_{\tilde{\mathbf{x}}} \log f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}|\mathbf{x})\right] + C_2 \\
&= \mathbb{E}_{f_{\tilde{\mathbf{x}}}}\left[\left|\left|s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}})\right|\right|^2\right] - 2\,\mathbb{E}_{f_{\tilde{\mathbf{x}},\mathbf{x}}}\left[s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}})^T \nabla_{\tilde{\mathbf{x}}} \log f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}|\mathbf{x})\right] + C_2, \qquad (3.27)
\end{aligned}$$

where $C_2 = \mathbb{E}_{f_{\tilde{\mathbf{x}},\mathbf{x}}}\left[||\nabla_{\tilde{\mathbf{x}}} \log f_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}|\mathbf{x})||^2\right]$ is also independent of $\boldsymbol{\theta}$. Because L_{noisy} and L_{SM} are equal except for terms independent of $\boldsymbol{\theta}$, optimizing L_{SM} gives the same result as optimizing L_{noisy}, which is $s^{\boldsymbol{\theta}}(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log f(\mathbf{x})$.

In theory, a single perturbation level is enough to approximate the score and use it for Langevin sampling regardless of the initial $\mathbf{x}^0$. In practice, however, the score estimate may be inaccurate in the neighborhood of $\mathbf{x}^0$. This is because the distance inside L_{SM} is weighted by $f(\mathbf{x})$, which makes the estimate error in low-density regions irrelevant. If $\mathbf{x}^0$ happens to fall in a place where the score estimate is inaccurate, Langevin dynamics will not converge to the desired distribution. Figure 3.2 illustrates this idea in two dimensions.

To solve this issue, the authors of [6] suggested the alternative cost function

$$L_{\text{SMLD}}(\boldsymbol{\theta}) = \mathbb{E}_i\left[\sigma_i^2\, \mathbb{E}_{f^i_{\tilde{\mathbf{x}},\mathbf{x}}}\left[\left|\left|s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, i) - \nabla_{\tilde{\mathbf{x}}} \log f^i_{\tilde{\mathbf{x}}|\mathbf{x}}(\tilde{\mathbf{x}}|\mathbf{x})\right|\right|^2\right]\right], \qquad (3.28)$$

where $f^i_{\tilde{\mathbf{x}}|\mathbf{x}}(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\mathbf{x}, \sigma_i^2 \mathbf{I})$, and where $\{\sigma_i\}_{i=1}^N$ is a growing sequence of perturbation levels. With this loss function, $s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, i)$ becomes a function of both the noisy input and an index i representing the noise level, much like DDPM.

Using higher noise levels helps in the convergence problem because it enlarges the

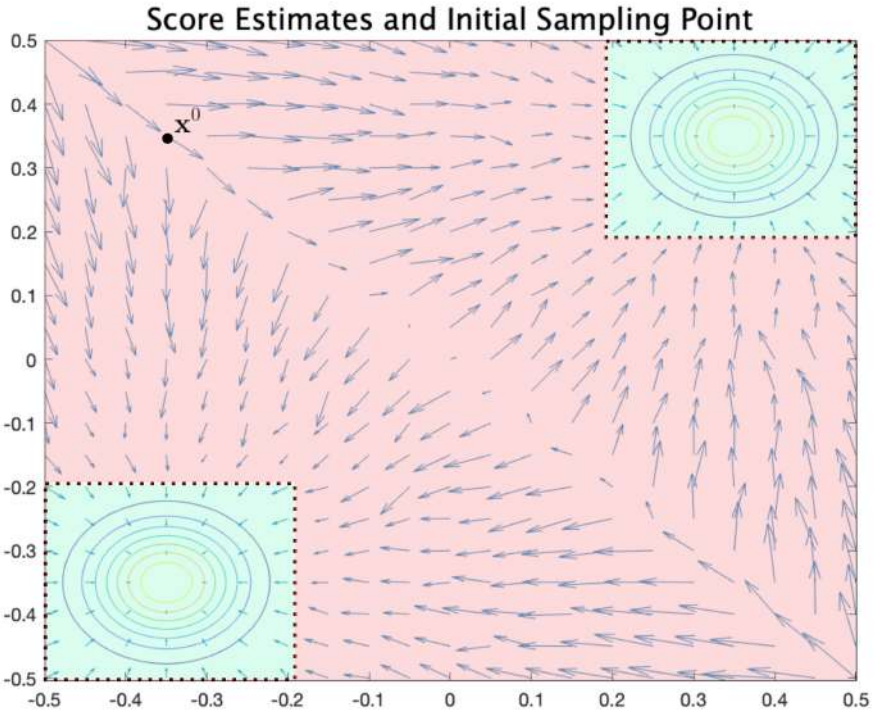

Figure 3.2: Possible score estimates for $f(\mathbf{x})$ composed of the sum of two Gaussian distributions, along with the initial point for Langevin sampling and the contour curves of $f(\mathbf{x})$. The regions with accurate score estimates are shown in green, whereas red indicates the regions with poor estimates. Adapted from [6].

original data distribution, populating low-density regions where the score estimate was not accurate. However, choosing σ too large corrupts the original distribution, and makes the approximation $s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}}) \approx \nabla_{\mathbf{x}} \log f(\mathbf{x})$ invalid. By averaging over multiple noise levels, the authors of [6] intended to keep the best of both worlds to make the score estimate as precise as possible.

Algorithm 3.3 shows a routine for training a neural network to calculate $s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, i)$. Before each update of parameters $\boldsymbol{\theta}$ the loss function L_{SMLD} is accumulated over noise levels one to N. In the algorithm, $\mathbf{x}$ is sampled from the data distribution to be modelled, and its perturbed version $\tilde{\mathbf{x}}$ from a Gaussian distribution of covariance matrix $\sigma_i^2\mathbf{I}$ centered around $\mathbf{x}$. The gradient $\nabla_{\tilde{\mathbf{x}}} \log f^i_{\tilde{\mathbf{x}}|\mathbf{x}}(\tilde{\mathbf{x}}|\mathbf{x})$ is calculated considering the Gaussian distribution of $\tilde{\mathbf{x}}$ as $\frac{\tilde{\mathbf{x}}-\mathbf{x}}{\sigma_i^2}$, and the learning rate is once again represented by η. It is important to observe that Algorithm 3.1 could be recast in a similar fashion as Algorithm 3.3 if all noise levels were used at each iteration.

Considering this alternative cost function, the authors of [6] suggested a new sampling algorithm inspired on simulated annealing [69]. Several Langevin MCMC iterations of M steps are performed in sequence, with decreasing noise level at every

Algorithm 3.3: SMLD Training

while $||\boldsymbol{\theta}_{\text{new}} - \boldsymbol{\theta}_{\text{old}}||^2 \leq$ tolerance **do**
  $L_{\text{SMLD}} = 0$;
  for $i = 1 : N$ **do**
    $\mathbf{x} \sim f(\mathbf{x})$;
    $\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \sigma_i^2 \mathbf{I})$;
    $L_{\text{SMLD}} = L_{\text{SMLD}} + \sigma_i \left(\left|\left| s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, i) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma_i^2} \right|\right|^2 \right)$;
  end
  $\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_{\text{old}} - \eta \nabla_{\boldsymbol{\theta}} L_{\text{SMLD}}$;
end
return $\boldsymbol{\theta}_{\text{new}}$;

round. The new sampling recurrence is

$$\mathbf{x}_i^{m+1} = \mathbf{x}_i^m + \epsilon s^{\boldsymbol{\theta}}(\mathbf{x}_i^m, i) + \sqrt{2\epsilon}\mathbf{z}, \tag{3.29}$$

with $m = 0, \ldots, M$ being the current Langevin step, $i = N, \ldots, 1$ the current noise level, $\epsilon \geq 0$, and $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. The last value of Langevin iteration i is the starting point of the iteration with immediately inferior noise level $i - 1$, i.e., $\mathbf{x}_i^M = \mathbf{x}_{i-1}^0$.

The authors recommend choosing the noise schedule σ_i such that $f_{\tilde{\mathbf{x}}}^1(\tilde{\mathbf{x}}) \approx f(\mathbf{x})$, and that $f_{\tilde{\mathbf{x}}}^N(\tilde{\mathbf{x}}) \approx \mathcal{N}(0, \sigma_N^2 \mathbf{I})$, which implies $\mathbf{x}_N^0 \sim \mathcal{N}(0, \sigma_N^2 \mathbf{I})$. They also state that M, N, and ϵ are hyperparameters that should be validated depending on the application. In [7] some guidelines for choosing these parameters are presented for image generation. Algorithm 3.4 shows the full sampling algorithm for SMLD which consists of applying the recurrence of Equation (3.29) over all (decreasing) noise levels.

Algorithm 3.4: SMLD Sampling

$\mathbf{x} \sim \mathcal{N}(0, \sigma_N \mathbf{I})$;
for $i = N : 1$ **do**
  for $m = 1 : M$ **do**
    $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$;
    $\mathbf{x} = \mathbf{x} + \epsilon s^{\boldsymbol{\theta}}(\mathbf{x}_i^m, i) + \sqrt{2\epsilon}\mathbf{z}$;
  end
end
return $\mathbf{x}$;

In SMLD models, each Langevin MCMC round acts as a corrector to an initial prediction of a sample from $f(\mathbf{x})$. Using score matching with multiple noise levels, the Langevin iterations are gradually annealed to converge to the data distribution being modelled. In Section 3.3, the connection between DDPM and SMLD models

will be established, and the benefits of stochastic correction during sampling will be examined in more detail.

3.3 Stochastic Differential Equation Diffusion

When DDPM was first published in [5], the authors showed that there was a connection between DDPM and score matching with multiple noise levels. In [7] this idea was further developed, and a general framework using stochastic differential equations (SDEs) was proposed for both DDPM and SMLD[4].

To see the connection between DDPM and SMLD, the first step is to rewrite the loss functions of both models in a common form. For denoising diffusion probabilistic models, remembering that $f_i(\mathbf{x}_i|\mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_i}\mathbf{x}_0, (1-\bar{\alpha}_i)\mathbf{I})$, it is possible to write

$$\nabla_{\mathbf{x}_i} \log f_i(\mathbf{x}_i|\mathbf{x}_0) = \nabla_{\mathbf{x}_i} \left[\frac{-\left|\left|\mathbf{x}_i - \sqrt{\bar{\alpha}_i}\mathbf{x}_0\right|\right|^2}{2(1-\bar{\alpha}_i)} \right] = \frac{-\mathbf{z}}{\sqrt{1-\bar{\alpha}_i}}, \tag{3.30}$$

with $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. In the first passage, the normalization terms in the Gaussian can be ignored because the gradient is taken with respect to $\mathbf{x}_i$. Then, Equation (3.20) can be rewritten as

$$\begin{aligned} L_{\text{DDPM}}(\boldsymbol{\theta}) &= \mathbb{E}_i \left[\mathbb{E}_{f_0, \mathcal{N}(0,\mathbf{I})} \left[\left|\left| \mathbf{z} - \mathbf{z}_i^{\boldsymbol{\theta}} \right|\right|^2 \right] \right] \\ &= \mathbb{E}_i \left[(1-\bar{\alpha}_i)\, \mathbb{E}_{f_0, \mathcal{N}(0,\mathbf{I})} \left[\left|\left| \frac{\mathbf{z}}{\sqrt{1-\bar{\alpha}_i}} - \frac{\mathbf{z}_\mathbf{i}^{\boldsymbol{\theta}}}{\sqrt{1-\bar{\alpha}_i}} \right|\right|^2 \right] \right] \\ &= \mathbb{E}_i \left[\lambda_i\, \mathbb{E}_{f_{0,i}} \left[\left|\left| s^{\boldsymbol{\theta}}(\mathbf{x}_i, i) - \nabla_{\mathbf{x}_i} \log f_i(\mathbf{x}_i|\mathbf{x}_0) \right|\right|^2 \right] \right], \end{aligned} \tag{3.31}$$

where $\lambda_i = 1 - \bar{\alpha}_i$ and $s_i^{\boldsymbol{\theta}}(\mathbf{x}_i, i) = \frac{-\mathbf{z}_\mathbf{i}^{\boldsymbol{\theta}}}{\sqrt{1-\bar{\alpha}_i}}$. Written like this, L_{DDPM} has the same shape as

$$L_{\text{SMLD}}(\boldsymbol{\theta}) = \mathbb{E}_i \left[\sigma_i^2\, \mathbb{E}_{f^i_{\tilde{\mathbf{x}},\mathbf{x}}} \left[\left|\left| s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, i) - \nabla_{\tilde{\mathbf{x}}} \log f^i_{\tilde{\mathbf{x}}|\mathbf{x}}(\tilde{\mathbf{x}}|\mathbf{x}) \right|\right|^2 \right] \right], \tag{3.32}$$

with $\lambda_i = \sigma_i^2$, $\tilde{\mathbf{x}} = \mathbf{x}_i$ and $f^i_{\tilde{\mathbf{x}}|\mathbf{x}}(\tilde{\mathbf{x}}|\mathbf{x}) = f_i(\mathbf{x}_i|\mathbf{x}_0)$.

In both DDPM and SMLD, $f_i(\mathbf{x}_i|\mathbf{x}_0)$ is Gaussian with covariance matrix proportional to the identity, which makes it possible to write that the score $\nabla_{\mathbf{x}_i} \log f_i(\mathbf{x}_i|\mathbf{x}_0) = -\frac{\mathbf{z}}{\sigma}$, with $\sigma^2\mathbf{I}$ being the covariance matrix of $f_i(\mathbf{x}_i|\mathbf{x}_0)$. For DDPM, $\sigma^2 = 1 - \bar{\alpha}_i$, while for SMLD $\sigma^2 = \sigma_i^2$. Having $\nabla_{\mathbf{x}_i} \log f_i(\mathbf{x}_i|\mathbf{x}_0) = -\frac{\mathbf{z}}{\sigma}$ im-

[4]The intent of this section was providing detailed explanations of results for SDE diffusion that are partially derived, or derived with extreme conciseness, in [7] and [70]. Because of this, a different notation was adopted in relation to the original papers, and most derivations and algorithms were reformulated by the author of this dissertation for clarity and detail. Unless explicitly referenced before being presented, all algorithms and demonstrations in this section are reformulated versions of the ones in [7] and [70].

plies that $\mathbb{E}_{f_{i|0}}\left[||\nabla_{\mathbf{x}_i}\log f_i(\mathbf{x}_i|\mathbf{x}_0)||^2\right] = \mathbb{E}_{\mathcal{N}(0,\mathbf{I})}\left[\left|\left|-\frac{\mathbf{z}}{\sigma}\right|\right|^2\right] = \frac{1}{\sigma^2}$, which in turn means that $\lambda_i = \left(\mathbb{E}_{f_{i|0}}\left[||\nabla_{\mathbf{x}_i}\log f_i(\mathbf{x}_i|\mathbf{x}_0)||^2\right]\right)^{-1}$ for DDPM and SMLD.

Noticing this, and also the fact that the Langevin sampling algorithm is originally derived from the discretization of a stochastic differential equation, the authors of [7] proposed a model with cost function

$$L_{\text{SDE}}(\boldsymbol{\theta}) = \mathbb{E}_t\left[\lambda(t)\,\mathbb{E}_{f_{0,t}}\left[\left|\left|s^{\boldsymbol{\theta}}(\mathbf{x}(t),t) - \nabla_{\mathbf{x}(t)}\log f_t(\mathbf{x}(t)|\mathbf{x}(0))\right|\right|^2\right]\right], \tag{3.33}$$

where continuous $t \sim \mathcal{U}(0,T)$ indexes time, and where $\mathbf{x}(t)$ is now a continuous-time random process. Inspired by the previous loss functions, $\lambda(t) \propto \left(\mathbb{E}_{f_{t|0}}\left[\left|\left|\nabla_{\mathbf{x}(t)}\log f_t(\mathbf{x}(t)|\mathbf{x}(0))\right|\right|^2\right]\right)^{-1}$. The process $\mathbf{x}(t)$ can be seen as the output of a system modelled by the SDE [7]

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = F(\mathbf{x}(t),t) + g(t)\mathbf{z}, \tag{3.34}$$

where $\mathbf{z} \sim \mathcal{N}(0,\mathbf{I})$. Equation (3.34) is the SDE responsible for the forward diffusion process, mapping the data distribution $f_0(\mathbf{x}(0))$ to a known noise distribution $f_T(\mathbf{x}(T))$.

Stochastic differential equations in the form of Equation (3.34) are reversible [71]; that is, there exists a SDE capable of mapping $f_T(\mathbf{x}(T))$ back into $f_0(\mathbf{x}(0))$. Furthermore, the reverse SDE has the form

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = F(\mathbf{x}(t),t) - g(t)^2\nabla_{\mathbf{x}(t)}\log f_t(\mathbf{x}(t)) + g(t)\mathbf{z}, \tag{3.35}$$

where once again $\mathbf{z} \sim \mathcal{N}(0,\mathbf{I})$.

By the same arguments of score matching, optimizing Equation (3.33) makes $s^{\boldsymbol{\theta}}(\mathbf{x}(t),t) \approx \nabla_{\mathbf{x}(t)}\log f_t(\mathbf{x}(t))$, and therefore the backward SDE can be used to sample the original data distribution starting from $f_T(\mathbf{x}(T))$. It is important to note, however, that this requires algorithms to simulate Equations (3.34) and (3.35) in discrete time during training and sampling, respectively.

In DDPM and SMLD, two different pairs of SDEs are defined by the recursions between noise levels [7]. Using them, one can derive the $F(\cdot,\cdot)$ and $g(\cdot)$ corresponding to each of the two models.

In SMLD, for example, one has

$$\begin{cases}\mathbf{x}_i = \mathbf{x}_0 + \sigma_i\mathbf{z}\\ \mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{y}\end{cases} \implies \begin{cases}\mathbf{x}_0 + \sigma_i\mathbf{z} + \mathbf{y} = \mathbf{x}_0 + \sigma_{i+1}\mathbf{z}\\ \mathbf{y} \sim \mathcal{N}(0,(\sigma_{i+1}^2 - \sigma_i^2)\mathbf{I})\end{cases} \tag{3.36}$$

with $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$, and therefore

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \sqrt{\sigma_{i+1}^2 - \sigma_i^2}\,\mathbf{z}. \tag{3.37}$$

Using Δt as the step size for discretizing the SDE, and considering $\sigma(t) : [0, T] \to \mathbb{R}$ a continuous function where $\sigma_i = \sigma(i\Delta t)$, Equation (3.37) can be rewritten as [7]

$$\mathbf{x}(t + \Delta t) - \mathbf{x}(t) = \sqrt{\sigma(t + \Delta t)^2 - \sigma(t)^2}\,\mathbf{z}, \tag{3.38}$$

which implies

$$\mathbf{x}(t + \Delta t) - \mathbf{x}(t) \approx \sqrt{\frac{\mathrm{d}[\sigma(t)]^2}{\mathrm{d}t}}\sqrt{\Delta t}\,\mathbf{z}. \tag{3.39}$$

Finally, the SDE is found noticing that Equation (3.39) [72] is the Euler-Maruyama discretization of

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \sqrt{\frac{\mathrm{d}[\sigma(t)]^2}{\mathrm{d}t}}\mathbf{z}. \tag{3.40}$$

A more complete description of Euler-Maruyama (and other numerical methods for solving SDEs) can be found in [72]. For the purposes of this thesis, it suffices to know that it is the stochastic equivalent of the Euler method for solving ordinary differential equations. Just like the Euler method, Euler-Maruyama assumes $F(\cdot, \cdot)$ and $g(\cdot)$ are constant over an interval Δt and calculates an integral as the product between the function value and Δt. However, because the stochastic term cannot be approximated by a constant, it is instead approximated by the sum of Δt standard Gaussians, resulting in the term $\sqrt{\Delta t}\,\mathbf{z}$.

For SMLD, the sequence $\{\sigma_i\}_{i=1}^N$ is usually chosen as a geometric progression starting at $\sigma_{\min}$ and ending at $\sigma_{\max}$. Techniques for properly choosing both values are given in [6]. Therefore, one has

$$\sigma(i\Delta t) = \sigma_{\min}\left(\frac{\sigma_{\max}}{\sigma_{\min}}\right)^{\frac{i-1}{N-1}}, \tag{3.41}$$

which implies, when $\Delta t \to 0$, that

$$\sigma(t) = \sigma_{\min}\left(\frac{\sigma_{\max}}{\sigma_{\min}}\right)^{t/T}. \tag{3.42}$$

In [7], Equation (3.40) is named the variance exploding (VE) SDE, because it can be shown that the random process that solves the equation has covariance matrix $\sigma(t)^2\mathbf{I}$. Since $\sigma(t)$ represents increasing noise levels, the variance of the elements of

$\mathbf{x}(t)$ increases indefinitely over time[5].

Simulating forward diffusion during training always requires discretizing the forward SDE. For sampling, however, there are multiple options. The first one is discretizing the backward SDE similarly to the forward SDE. Using Euler-Maruyama, the forward recurrence is [7]

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \underbrace{F(\mathbf{x}_i, i\Delta t)\Delta t}_{F_i} + \underbrace{g(i\Delta t)\sqrt{\Delta t}}_{g_i}\,\mathbf{z}, \tag{3.45}$$

which implies that the sampling recurrence (i.e., the discretized backwards SDE) is

$$\mathbf{x}_i = \mathbf{x}_{i+1} - F_{i+1} + g_{i+1}^2 s^{\boldsymbol{\theta}}(\mathbf{x}_{i+1}, (i+1)\Delta t) - g_{i+1}\mathbf{z}, \tag{3.46}$$

where $s^{\boldsymbol{\theta}}$ was used to approximate the score $\nabla_{\mathbf{x}(t)} \log f_t(\mathbf{x}(t))$. Note that in both discretizations the differential $\mathrm{d}\mathbf{x}$ is approximated assuming the deterministic equation terms as constants over Δt, and using $\sqrt{\Delta t}\,\mathbf{z}$ — the sum of Δt standard Gaussians — to approximate the random multiplier of the stochastic terms. This is the standard procedure for Euler-Maruyama, and the reader is invited to read about the method in more detail in [72]

The second approach is the one used in DDPM, where sampling is performed based on the recurrence $\mathbf{x}_i = \boldsymbol{\mu}_{i+1}^{\boldsymbol{\theta}} + \sigma_{i+1}\mathbf{z}$ that comes from $p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{x}_{i+1}) = \mathcal{N}(\boldsymbol{\mu}_{i+1}^{\boldsymbol{\theta}}, \sigma_{i+1}^2\mathbf{I})$. This approach requires solving the KL divergence in the L_{i-1} terms of L_{VLB} in order to find a suitable parametrization for $\boldsymbol{\mu}_i^{\boldsymbol{\theta}}$, which might be difficult in a general case, since it requires an analytical expression for $f_{i-1}(\mathbf{x}_{i-1}|\mathbf{x}_i, \mathbf{x}_0)$.

The third approach involves finding an ordinary differential equation (ODE) that replaces the pair of SDEs corresponding to the model [7]. Instead of having a forward SDE that maps $f(\mathbf{x}(0))$ into $f(\mathbf{x}(T))$, and a backward SDE that maps $f(\mathbf{x}(T))$ into $f(\mathbf{x}(0))$, one can use a single ODE capable of mapping a sample of $f(\mathbf{x}(0))$ into a sample of $f(\mathbf{x}(T))$. During sampling, this ODE can be integrated backwards in time with a sample from $f(\mathbf{x}(T))$ in order to obtain a sample from $f(\mathbf{x}(0))$. This approach is not directly available with SDEs because of the addition of random terms in the differential equation.

[5]The demonstration that the covariance of the solution of the VE SDE is $\sigma(t)^2\mathbf{I}$ is actually mentioned, but not explicitly shown in [7]. The formulas for the mean ($\mathbf{m}(t)$) and covariance ($\boldsymbol{\Sigma}(t)$) of the solution of an SDE of the form of Equation (3.34) are deduced in [72] as

$$\frac{\mathrm{d}\mathbf{m}(t)}{\mathrm{d}t} = \mathbb{E}[F(\mathbf{x}(t), t)], \tag{3.43}$$

$$\frac{\mathrm{d}\boldsymbol{\Sigma}(t)}{\mathrm{d}t} = \mathbb{E}[F(\mathbf{x}(t), t)(\mathbf{x}(t) - \mathbf{m}(t))^T] + \mathbb{E}[(\mathbf{x}(t) - \mathbf{m}(t))F(\mathbf{x}(t), t)^T] + g(t)^2\mathbf{I}. \tag{3.44}$$

From there, it follows with $F(\mathbf{x}(t), t) = 0$ and $g(t) = \sqrt{\frac{\mathrm{d}[\sigma(t)]^2}{\mathrm{d}t}}$, that $\boldsymbol{\Sigma}(t) = \sigma(t)^2\mathbf{I}$ for the VE SDE.

For an SDE of the form of Equation (3.34), the equivalent ODE is given by

$$\frac{\mathrm{d}\mathbf{x}(\mathrm{t})}{\mathrm{d}t} = F(\mathbf{x}(t), t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}(t)} \log f_t(\mathbf{x}(t)). \tag{3.47}$$

A detailed proof showing that this ODE is indeed equivalent to the SDEs in Equations (3.34) and (3.35) can be seen in Appendix D of [7]. However, an intuition is given as follows.

It can be shown that the probability density $f(\mathbf{x}(t))$ of the solution of Equation (3.34) at time t is given by the Fokker-Planck partial differential equation [72]

$$\frac{\partial f(\mathbf{x}(t))}{\partial t} = -\nabla_{\mathbf{x}(t)} \left[F(\mathbf{x}(t), t) f(\mathbf{x}(t)) \right] + \frac{1}{2} g(t)^2 \nabla^2_{\mathbf{x}(t)} f(\mathbf{x}(t)). \tag{3.48}$$

This equation can be rewritten in the form

$$\frac{\partial f(\mathbf{x}(t))}{\partial t} = -\nabla_{\mathbf{x}(t)} \left[F(\mathbf{x}(t), t) f(\mathbf{x}(t)) - \frac{1}{2} g(t)^2 f(\mathbf{x}(t)) \nabla_{\mathbf{x}(t)} \log f(\mathbf{x}(t)) \right], \tag{3.49}$$

which is the Fokker-Planck equation for the SDE

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \tilde{F}(\mathbf{x}(t), t) + \tilde{g}(t)\, \mathbf{z}, \tag{3.50}$$

with $\tilde{F}(\mathbf{x}(t), t) = F(\mathbf{x}(t), t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}(t)} \log f(\mathbf{x}(t))$, and $\tilde{g}(t) = 0$. Equation (3.50) performs the same mapping as the forward SDE, but is, in fact, equivalent to the ODE of Equation (3.47).

Using one of these three approaches allows one to obtain approximate samples of $\mathbf{x}_{i-1}$ given $\mathbf{x}_i$, that is, reversing a time step Δt of the forward diffusion process. However, it is worth noticing that this backward recursion is not present in the original sampling algorithm for SMLD. For every noise level in SMLD, an initial arbitrary prediction is corrected using Langevin dynamics. When the noise level changes, the initial prediction is simply the last MCMC sample from the previous noise level.

This suggests that initial predictions given by SDE sampling can be used as the starting point for MCMC procedures to obtain better samples. The authors of [7] propose "predictor-corrector" algorithms consisting of a numerical SDE/ODE step (prediction) followed by a Langevin MCMC step (correction). In SMLD, one can see the prediction step as the identity function, and in DDPM the correction step can be seen as the identity. Algorithm 3.5 contains an example of what an algorithm of this type would look like for the variance exploding SDE.

In Algorithm 3.5, for each noise level i the Euler-Maruyama discretized version of the variance exploding backward SDE is used to obtain a predicted sample $\mathbf{x}_i$.

Algorithm 3.5: Predictor-Corrector Sampling

$\mathbf{x} \sim \mathcal{N}(0, \sigma_N \mathbf{I})$;
for $i = N - 1 : 0$ **do**
  $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$;
  $\mathbf{x} = \mathbf{x} + (\sigma_{i+1}^2 - \sigma_i^2) s^{\boldsymbol{\theta}}(\mathbf{x}, i+1) + \sqrt{\sigma_{i+1}^2 - \sigma_i^2}\, \mathbf{z}$;
  for $m = 1 : M$ **do**
    $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$;
    $\mathbf{x} = \mathbf{x} + \epsilon s^{\boldsymbol{\theta}}(\mathbf{x}, i) + \sqrt{2\epsilon}\mathbf{z}$;
  end
end
return $\mathbf{x}$;

This prediction is then corrected by M steps of Langevin MCMC (the nested for loop). This discretization of the variance exploding backward SDE can be obtained by applying the $F(\cdot, \cdot)$ and $g(\cdot)$ of Equation (3.40) in Equation (3.46). Once again $\epsilon > 0$ is a hyperparameter that needs to be tuned, but some heuristics for choosing ϵ can be found in [7].

3.3.1 Heuristics for Diffusion with SDEs

As presented above, diffusion models using SDEs are rather complicated in the sense of having multiple design choices available. There are different plausible choices for $F(\cdot, \cdot)$ and $g(\cdot)$, different sampling techniques, and different neural networks that could be used to approximate $s^{\boldsymbol{\theta}}(\cdot, \cdot)$.

In [70], the authors tried to better understand diffusion models in order to suggest heuristics for designing them successfully. Their first contribution concerns the choice of the SDE, that is, the choice of the functions $F(\cdot, \cdot)$ and $g(\cdot)$ that characterize the diffusion process. If the forward diffusion SDE can be written in the form

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = F(t)\mathbf{x}(t) + g(t)\mathbf{z}, \tag{3.51}$$

with $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$, as is the case with DDPM and SMLD, then the mean $\mathbf{m}(t)$ and covariance $\boldsymbol{\Sigma}(t)$ of $f(\mathbf{x}(t))$ can be calculated using Equations (3.43) and (3.44), with

$$\frac{\mathrm{d}\mathbf{m}(t)}{\mathrm{d}t} = F(t)\,\mathbb{E}\left[\mathbf{x}(t)\right] = F(t)\,\mathbf{m}(t) \implies \mathbf{m}(t) = s(t)\mathbf{m}(0), \tag{3.52}$$

where $s(t) = \mathrm{e}^{\int_0^t F(\tau)\mathrm{d}\tau}$, and

$$\begin{aligned}
\frac{\mathrm{d}\boldsymbol{\Sigma}(t)}{\mathrm{d}t} &= F(t)\,\mathbb{E}\left[\mathbf{x}(t)(\mathbf{x}(t)-\mathbf{m}(t))^T\right] + F(t)\,\mathbb{E}\left[(\mathbf{x}(t)-\mathbf{m}(t))\mathbf{x}(t)^T\right] + g(t)^2\mathbf{I} \\
&= 2F(t)\left\{\mathbb{E}\left[\mathbf{x}(t)\mathbf{x}(t)^2\right] - \mathbf{m}(t)\mathbf{m}(t)^T\right\} + g(t)^2\mathbf{I} \\
&= 2F(t)\boldsymbol{\Sigma}(t) + g(t)^2\mathbf{I} \\
\implies \boldsymbol{\Sigma}(t) &= s(t)^2\tilde{\sigma}(t)^2\mathbf{I} + s(t)^2\boldsymbol{\Sigma}(0), \qquad (3.53)
\end{aligned}$$

where $\tilde{\sigma}(t)^2 = \int_0^t \frac{g(\tau)^2}{s(\tau)^2}\mathrm{d}\tau$.

Because $F(\mathbf{x}(t), t) = F(t)\mathbf{x}(t)$, the conditional distribution $f(\mathbf{x}(t)|\mathbf{x}(0))$ is Gaussian [7]. Intuitively, this can be seen by noticing that $F(\mathbf{x}(t), t) = F(t)\mathbf{x}(t)$ makes every differential step $\mathrm{d}t$ in the forward equation simply an addition of Gaussian noise. If $\mathbf{x}(0)$ is given, then $\mathbf{m}(0) = \mathbf{x}(0)$ and $\boldsymbol{\Sigma}(0) = \mathbf{0}$, which makes $f(\mathbf{x}(t)|\mathbf{x}(0)) = \mathcal{N}(s(t)\mathbf{x}(0), s(t)^2\tilde{\sigma}(t)^2\mathbf{I})$.

Therefore, as long as $F(\mathbf{x}(t), t) = F(t)\mathbf{x}(t)$ holds, there is a way to parameterize the equations of a diffusion model in terms of the desired mean and variance of the distribution for $\mathbf{x}(t)$ given some $\mathbf{x}(0)$. This is more intuitive than choosing arbitrary $F(\cdot,\cdot)$ and $g(\cdot)$ because it gives an idea of the evolution of the forward diffusion process.

In [70], the authors name $s(t)$ the scaling function, and $\tilde{\sigma}(t)$ the noise schedule function. The scaling function has this name because it scales the original data point $\mathbf{x}(0)$ to generate the sample mean during forward diffusion. The schedule function has this name because it represents the noise level progression $\sigma(t)$, ignoring the scaling factor.

To see this, one can calculate $\tilde{\sigma}(t)$ for SMLD as an example. From Equation (3.40), $F(t) = 0$, and $g(t) = \sqrt{\frac{\mathrm{d}[\sigma(t)]^2}{\mathrm{d}t}}$, which implies

$$s(t) = \mathrm{e}^{\int_0^t F(\tau)\mathrm{d}\tau} = 1, \qquad (3.54)$$

$$\tilde{\sigma}(t)^2 = \int_0^t \frac{g(\tau)^2}{s(\tau)^2}\mathrm{d}\tau = \int_0^t \frac{\mathrm{d}[\sigma(\tau)]^2}{\mathrm{d}\tau}\mathrm{d}\tau = \sigma(t)^2 - \sigma(0)^2. \qquad (3.55)$$

Considering $\sigma(0) = 0$ (which is consistent with the previous definition of $\sigma(i\Delta t)$), $\tilde{\sigma}(t)$ gives $\sigma(t)$, the perturbation level at time t for SMLD.

One can rewrite $F(t)$ and $g(t)$ in terms of $s(t)$ and $\sigma(t)$ to reformulate the forward SDE, backward SDE, and equivalent ODE in the same framework. Since $s(t) = \mathrm{e}^{\int_0^t F(\tau)\mathrm{d}\tau}$, it is possible to write [70]

$$\log s(t) = \int_0^t F(\tau)\mathrm{d}\tau \implies F(t) = \frac{\mathrm{d}\log s(t)}{\mathrm{d}t} = \frac{\dot{s}(t)}{s(t)}, \qquad (3.56)$$

where the dot represents the derivative with respect to t. Furthermore [70],

$$\begin{aligned}\tilde{\sigma}(t)^2 &= \int_0^t \frac{g(\tau)^2}{s(\tau)^2}\mathrm{d}\tau \\ \implies \frac{g(t)^2}{s(t)^2} &= \frac{\mathrm{d}\left[\tilde{\sigma}(t)^2\right]}{\mathrm{d}t} \\ \implies g(t) &= s(t)\sqrt{2\dot{\tilde{\sigma}}(t)\tilde{\sigma}(t)}.\end{aligned} \tag{3.57}$$

Substituting $F(t)$ and $g(t)$ in the forward SDE equation leads to

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \frac{\dot{s}(t)}{s(t)}\mathbf{x}(t) + s(t)\sqrt{2\dot{\tilde{\sigma}}(t)\tilde{\sigma}(t)}\,\mathbf{z}. \tag{3.58}$$

Doing the same in the backward SDE gives

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \frac{\dot{s}(t)}{s(t)}\mathbf{x}(t) - 2s(t)^2\dot{\tilde{\sigma}}(t)\tilde{\sigma}(t)\nabla_{\mathbf{x}(t)}\log f_t(\mathbf{x}(t)) + s(t)\sqrt{2\dot{\tilde{\sigma}}(t)\tilde{\sigma}(t)}\,\mathbf{z}. \tag{3.59}$$

And finally substituting $F(t)$ and $g(t)$ in the equivalent ODE yields

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \frac{\dot{s}(t)}{s(t)}\mathbf{x}(t) - s(t)^2\dot{\tilde{\sigma}}(t)\tilde{\sigma}(t)\nabla_{\mathbf{x}(t)}\log f_t(\mathbf{x}(t)). \tag{3.60}$$

Therefore, all three equations can be synthesized in the form

$$\begin{aligned}\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = &\underbrace{\frac{\dot{s}(t)}{s(t)}\mathbf{x}(t) - s(t)^2\dot{\tilde{\sigma}}(t)\tilde{\sigma}(t)\nabla_{\mathbf{x}(t)}\log f_t(\mathbf{x}(t))}_{\text{ODE term}} \quad + \\ &\underbrace{s(t)\sqrt{2\dot{\tilde{\sigma}}(t)\tilde{\sigma}(t)}\,\mathbf{z} \pm s(t)^2\dot{\tilde{\sigma}}(t)\tilde{\sigma}(t)\nabla_{\mathbf{x}(t)}\log f_t(\mathbf{x}(t)),}_{\text{Stochastic term}}\end{aligned} \tag{3.61}$$

where the stochastic term appears only in the SDEs and the plus sign and minus sign are used for forward and backward diffusion, respectively.

Equation (3.61) shows that both SDEs and the equivalent ODE can be seen as (almost) the same equation after being recast in terms of $s(t)$ and $g(t)$. Moreover, in this form it is easier to see why the equivalent ODE induces the same mapping for $f(\mathbf{x}(t))$ as the pair of SDEs. The stochastic term can be seen as a correction term for the ODE term prediction for a sample $\mathbf{x}(t)$ at time t.

In [70] the authors suggest

$$s(t) = 1, \tag{3.62}$$

$$\tilde{\sigma}(t) = \left(\sigma_{\min}^{1/\rho} + \frac{t}{T}\left(\sigma_{\max}^{1/\rho} - \sigma_{\min}^{1/\rho}\right)\right)^{\rho}, \tag{3.63}$$

and also propose using a second order Euler method, instead of a first order one

as used in [7]. Note that using Equation (3.62), the discretized version of Equation (3.61) is

$$\mathbf{x}_{i+1} - \mathbf{x}_i = \underbrace{-\left(\tilde{\sigma}_{i+1} - \tilde{\sigma}_i\right)\tilde{\sigma}_{i+1} s^{\boldsymbol{\theta}}(\mathbf{x}_{i+1}, i+1)}_{\text{ODE term}} + \underbrace{\sqrt{2\left(\tilde{\sigma}_{i+1} - \tilde{\sigma}_i\right)\tilde{\sigma}_{i+1}}\,\mathbf{z} \pm \left(\tilde{\sigma}_{i+1} - \tilde{\sigma}_i\right)\tilde{\sigma}_{i+1} s^{\boldsymbol{\theta}}(\mathbf{x}_{i+1}, i+1)}_{\text{Stochastic term}}, \tag{3.64}$$

where $\dot{\tilde{\sigma}}(t)$ was approximated by $\frac{(\tilde{\sigma}_{i+1} - \tilde{\sigma}_i)}{\Delta t}$, and where Δt was already incorporated into the right-hand side.

The idea behind the choice of $\tilde{\sigma}(t)$ is having more points near the lowest noise level. Empirically, the authors of [70] showed that prediction errors near the lowest perturbations have a larger impact in sample quality. Choosing $s(t) = 1$ removes the first term of the SDE/ODE, which makes the prediction more robust to errors accumulated during the sampling steps. Finally, using a second order method for solving the SDE/ODE offers a better trade off between computational complexity and prediction errors in the differential equation.

To validate their suggestion, the authors compared samples generated from pretrained models for DDPM and SMLD using their combination of $s(t)$, $\tilde{\sigma}(t)$ and second order Euler with samples generated from the same models using their original sampling schemes. The comparison was made using the equivalent ODE for their suggestion and SMLD/DDPM. In both cases, their method generated more realistic samples[6] with a smaller number of evaluations of $s^{\boldsymbol{\theta}}(\cdot, \cdot)$.

Algorithm 3.6 shows the sampling algorithm for the equivalent ODE applying the suggestions of [70]. At each reverse diffusion step, it makes an initial prediction $\tilde{\mathbf{x}}$ of $\mathbf{x}_i$ using the ODE part of Equation (3.64). This initial prediction is then used to calculate an approximation for $\mathbf{x}_i$ with a trapezoidal rule, as in standard second order Euler methods. Note that the sampling recurrence is different from that in Algorithm 3.5 because the discretized SDEs are different in the two cases due to the reformulation in terms of $s(t)$ and $\tilde{\sigma}(t)$.

A second contribution of [70] is a different algorithm for sampling with SDEs. If using Euler-Maruyama, in Algorithm 3.5 both prediction and correction steps require evaluating $s^{\boldsymbol{\theta}}(\cdot, \cdot)$. The authors of [70] note, however, that an Euler-Maruyama step is simply an ODE prediction step followed by noise addition, and that the correction steps in Algorithm 3.5 have the same form as the prediction steps. Indeed, for $\epsilon = \left(\sigma_{i+1}^2 - \sigma_i^2\right)$ the two are equal up to a $\sqrt{2}$ constant.

An alternative sampling scheme using fewer evaluations of $s^{\boldsymbol{\theta}}(\cdot, \cdot)$ can be made by removing the correction steps, and performing the ODE prediction in Euler-

[6] See note 3

Algorithm 3.6: Second Order ODE Sampling

$\mathbf{x} \sim \mathcal{N}(0, \sigma_{\max}\mathbf{I})$;
for $i = N-1:0$ **do**
  $\tilde{\mathbf{x}} = \mathbf{x} + (\tilde{\sigma}_{i+1} - \tilde{\sigma}_i)\tilde{\sigma}_{i+1} s^{\boldsymbol{\theta}}(\mathbf{x}, i+1)$;
  if $\tilde{\sigma}_i \neq 0$ **then**
    $\mathbf{x} = \mathbf{x} + \frac{1}{2}\left((\tilde{\sigma}_{i+1} - \tilde{\sigma}_i)\tilde{\sigma}_{i+1} s^{\boldsymbol{\theta}}(\mathbf{x}, i+1) + (\tilde{\sigma}_{i+1} - \tilde{\sigma}_i)\tilde{\sigma}_i s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, i)\right)$;
  else
    $\mathbf{x} = \tilde{\mathbf{x}}$
  end
end
return $\mathbf{x}$;

Maruyama using a randomly perturbed $\mathbf{x}$. In this scenario, the random perturbation in $\mathbf{x}$ replaces the random component in the MCMC correction step, and only simple ODE steps are taken. However, to account for the random perturbation, an accordingly higher noise level is used in the ODE steps.

Algorithm 3.7. shows a basic version of this alternative sampling scheme. At each step an increased noise level is calculated by multiplying the current noise level by a factor $1 + \gamma$, where γ represents a (constant) percentual noise increase. The increased noise level is then used to randomly change $\mathbf{x}$ in Langevin MCMC fashion, similarly to the correction steps in Algorithm 3.5. Finally, the disturbed $\mathbf{x}$ is used to perform a second order Euler ODE step. By fusing together the prediction and correction steps, while still maintaining random changes to $\mathbf{x}$, it is possible to keep the benefits of Algorithm 3.5 while making fewer evaluations of $s^{\boldsymbol{\theta}}(\cdot, \cdot)$.

Algorithm 3.7: Basic Stochastic Sampling

$\mathbf{x} \sim \mathcal{N}(0, \sigma_{\max}\mathbf{I})$;
for $i = N-1:0$ **do**
  $\hat{\sigma}_{i+1} = \tilde{\sigma}_{i+1}(1+\gamma)$;
  $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$;
  $\mathbf{x} = \mathbf{x} + \sqrt{\hat{\sigma}_{i+1}^2 - \tilde{\sigma}_{i+1}^2}\,\mathbf{z}$;
  $\tilde{\mathbf{x}} = \mathbf{x} + (\tilde{\sigma}_{i+1} - \tilde{\sigma}_i)\tilde{\sigma}_{i+1} s^{\boldsymbol{\theta}}(\mathbf{x}, i+1)$;
  if $\tilde{\sigma}_i \neq 0$ **then**
    $\mathbf{x} = \mathbf{x} + \frac{1}{2}\left((\tilde{\sigma}_{i+1} - \tilde{\sigma}_i)\tilde{\sigma}_{i+1} s^{\boldsymbol{\theta}}(\mathbf{x}, i+1) + (\tilde{\sigma}_{i+1} - \tilde{\sigma}_i)\tilde{\sigma}_i s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, i)\right)$;
  else
    $\mathbf{x} = \tilde{\mathbf{x}}$
  end
end
return $\mathbf{x}$;

In [70] the authors propose a variation of Algorithm 3.7 adopting the following heuristics: a range $[S_{\min}, S_{\max}]$ outside of which the random perturbation is not applied, variance $S_{\mathbf{z}}^2$ slightly higher than one for $\mathbf{z}$, and $\gamma = \min\left(S_{\text{total}}/N, \sqrt{2} - 1\right)$,

where S_{total} is a parameter indicating the amount of perturbation to be used considering all steps.

Algorithm 3.8 shows the full algorithm with heuristics included. The first conditional is used to decide whether or not to increase the noise level by the factor γ; noise levels above and below $S_{\min}$ and $S_{\max}$ are not changed. The value of γ is now a function of S_{total} and the number of diffusion steps, with a larger number of diffusion steps implying smaller noise level increases for stability purposes. The rest of the algorithm is identical to Algorithm 3.7, with the exception that the random perturbation applied to $\mathbf{x}$ now has its variance multiplied by $S_{\mathbf{z}}$.

Algorithm 3.8: Heuristic Stochastic Sampling

$\mathbf{x} \sim \mathcal{N}(0, \sigma_{\max}\mathbf{I})$;
for $i = N : 1$ **do**
  if $\tilde{\sigma}_{i+1} \in [S_{\min}, S_{\max}]$ **then**
    $\hat{\sigma}_{i+1} = \tilde{\sigma}_{i+1}(1 + \min\left(S_{\text{total}}/N, \sqrt{2} - 1\right))$;
  else
    $\hat{\sigma}_{i+1} = \tilde{\sigma}_{i+1}$;
  end
  $\mathbf{z} \sim \mathcal{N}(0, S_{\mathbf{z}}^2\mathbf{I})$;
  $\mathbf{x} = \mathbf{x} + \sqrt{\hat{\sigma}_{i+1}^2 - \tilde{\sigma}_{i+1}^2}\,\mathbf{z}$;
  $\tilde{\mathbf{x}} = \mathbf{x} + (\tilde{\sigma}_{i+1} - \tilde{\sigma}_i)\tilde{\sigma}_{i+1}s^{\boldsymbol{\theta}}(\mathbf{x}, i+1)$;
  if $\tilde{\sigma}_i \neq 0$ **then**
    $\mathbf{x} = \mathbf{x} + \frac{1}{2}\left((\tilde{\sigma}_{i+1} - \tilde{\sigma}_i)\tilde{\sigma}_{i+1}s^{\boldsymbol{\theta}}(\mathbf{x}, i+1) + (\tilde{\sigma}_{i+1} - \tilde{\sigma}_i)\tilde{\sigma}_i s^{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, i)\right)$;
  else
    $\mathbf{x} = \tilde{\mathbf{x}}$
  end
end
return $\mathbf{x}$;

To validate Algorithm 3.8, the authors of [70] adopt the same strategy of using the suggested sampling scheme with pre-trained models for DDPM and SMLD. Once again, using their algorithm results in better samples[7] for a fixed number of evaluations of $s^{\boldsymbol{\theta}}(\cdot, \cdot)$. This time, however, the authors note that the choice of $S_{\min}$, $S_{\max}$, S_{total}, and $S_{\mathbf{z}}$ should be done on a case-by-case basis. Their results are presented only for their optimized choice of parameters.

Still, obtaining better results [8] with the same pre-trained models both for ODE and SDE sampling suggests that, for diffusion models, neural network training and sample generation are largely independent tasks. The best sampling scheme can be chosen independently of the neural network used to calculate $s^{\boldsymbol{\theta}}(\cdot, \cdot)$, and the best model for $s^{\boldsymbol{\theta}}(\cdot, \cdot)$ can be chosen independently of the sampling scheme.

[7] See note 3
[8] See note 3

The final contribution of [70] concerns the training regime for $s^{\boldsymbol{\theta}}(\cdot,\cdot)$. Since, for $s(t)=1$,

$$\nabla_{\mathbf{x}(t)} \log f_t(\mathbf{x}(t)|\mathbf{x}(0)) = \frac{\mathbf{x}(0)-\mathbf{x}(t)}{\tilde{\sigma}(t)^2}, \tag{3.65}$$

it makes sense to see $s^{\boldsymbol{\theta}}(\mathbf{x}(t),t)$ as

$$s^{\boldsymbol{\theta}}(\mathbf{x}(t),t) = \frac{\mathbf{x}^{\boldsymbol{\theta}}-\mathbf{x}(t)}{\tilde{\sigma}(t)^2}, \tag{3.66}$$

where $\mathbf{x}^{\boldsymbol{\theta}}$ is a prediction for $\mathbf{x}(0)$ created using the model with parameters $\boldsymbol{\theta}$. Substituting Equations (3.65) and (3.66) into Equation (3.33) gives

$$\begin{aligned} L_{\text{SDE}}(\boldsymbol{\theta}) &= \mathbb{E}_t\left[\lambda(t)\,\mathbb{E}_{f_{0,t}}\left[\left|\left|\frac{1}{\tilde{\sigma}(t)^2}\left(\mathbf{x}^{\boldsymbol{\theta}}-\mathbf{x}(t)-\mathbf{x}(0)+\mathbf{x}(t)\right)\right|\right|^2\right]\right] \\ &= \mathbb{E}_t\left[\frac{\lambda(t)}{\tilde{\sigma}(t)^4}\,\mathbb{E}_{f_{0,t}}\left[\left|\left|\mathbf{x}^{\boldsymbol{\theta}}-\mathbf{x}(0)\right|\right|^2\right]\right]. \end{aligned} \tag{3.67}$$

In [70] the authors propose modelling $\mathbf{x}^{\boldsymbol{\theta}}$ as

$$\mathbf{x}^{\boldsymbol{\theta}} = c_{\text{skip}}(\tilde{\sigma}(t))\mathbf{x}(t) + c_{\text{out}}(\tilde{\sigma}(t))F^{\boldsymbol{\theta}}(c_{\text{in}}(\tilde{\sigma}(t))\mathbf{x}(t), c_{\text{noise}}(\tilde{\sigma}(t))), \tag{3.68}$$

where c_{skip}, c_{out}, c_{in}, and c_{noise} are functions of $\tilde{\sigma}(t)$ to be chosen before training. In this setting, $F^{\boldsymbol{\theta}}(\cdot,\cdot)$ is the model being optimized, and the approximation $s^{\boldsymbol{\theta}}$ for the score can be calculated for sampling with Equations (3.68) and (3.66). Using Equation (3.68) in Equation (3.67) gives

$$\begin{aligned} L_{\text{SDE}}(\boldsymbol{\theta}) &= \mathbb{E}_t\left[\frac{\lambda(t)}{\tilde{\sigma}(t)^4}\,\mathbb{E}_{f_{0,t}}\left[\left|\left|c_{\text{skip}}\mathbf{x}(t) + c_{\text{out}}F^{\boldsymbol{\theta}}(c_{\text{in}}\mathbf{x}(t), c_{\text{noise}}) - \mathbf{x}(0)\right|\right|^2\right]\right] \\ &= \mathbb{E}_t\left[w(t)\,\mathbb{E}_{f_{0,t}}\left[\left|\left|F^{\boldsymbol{\theta}}(c_{\text{in}}\mathbf{x}(t), c_{\text{noise}}) - F_{\text{target}}(\mathbf{x}(t),t)\right|\right|^2\right]\right], \end{aligned} \tag{3.69}$$

where $w(t) = \frac{\lambda(t)c_{\text{out}}^2}{\tilde{\sigma}(t)^4}$, and $F_{\text{target}}(\mathbf{x}(t),t) = \frac{\left(\mathbf{x}(0)-c_{\text{skip}}\mathbf{x}(t)\right)}{c_{\text{out}}}$. The arguments of c_{skip}, c_{out}, c_{in}, and c_{noise} were omitted.

There are two advantages of reformulating the loss function as in Equation (3.69). The first is making sure that the neural network being trained receives input with unit variance [73]. This was not possible with $\mathbf{x}(t)$ given directly as input to $s^{\boldsymbol{\theta}}(\mathbf{x}(t),t)$, but is possible using $c_{\text{in}}(\tilde{\sigma}(t))\mathbf{x}(t)$ as input to $F^{\boldsymbol{\theta}}(c_{\text{in}}(\tilde{\sigma}(t))\mathbf{x}(t), c_{\text{noise}}(\tilde{\sigma}(t)))$, as long as $c_{\text{in}}(\tilde{\sigma}(t))$ is chosen correctly.

To force unit variance [70],

$$\begin{aligned}
&\text{Cov}\left[c_{\text{in}}(\tilde{\sigma}(t))\mathbf{x}(t)\right] = \mathbf{I} \\
&\implies c_{\text{in}}(\tilde{\sigma}(t))^2 \,\text{Cov}\left[\mathbf{x}(t)\right] = \mathbf{I} \\
&\implies c_{\text{in}}(\tilde{\sigma}(t))^2 \left(\tilde{\sigma}(t)^2\mathbf{I} + \boldsymbol{\Sigma}(0)\right) = \mathbf{I},
\end{aligned} \tag{3.70}$$

where the last implication comes from Equation (3.53). Another way to see this would be writing $\mathbf{x}(t) = \mathbf{x}(0) + \mathbf{n}$, with $\mathbf{n} \sim \mathcal{N}(0, \tilde{\sigma}(t)\mathbf{I})$ independent of $\mathbf{x}(0)$. Inverting $(\tilde{\sigma}(t)^2\mathbf{I} + \boldsymbol{\Sigma}(0))$ to find the correct $c_{\text{in}}(\tilde{\sigma}(t))$ might be unfeasible because of high data dimensionality, but one can assume $\boldsymbol{\Sigma}(0) = \sigma_{\text{data}}^2\mathbf{I}$, with σ_{data}^2 being the sample variance across all elements of $\mathbf{x}(0)$. This leads to [70]

$$c_{\text{in}}(\tilde{\sigma}(t)) = \frac{1}{\sqrt{\tilde{\sigma}(t)^2 + \sigma_{\text{data}}^2}}. \tag{3.71}$$

The second advantage of Equation (3.69) is avoiding amplifying errors in the estimate $\mathbf{x}^{\boldsymbol{\theta}}$ of $\mathbf{x}(0)$. Using $s^{\boldsymbol{\theta}}(\mathbf{x}(t), t)$ to model the score means implicitly using $\mathbf{x}^{\boldsymbol{\theta}}$ to calculate $\nabla_{\mathbf{x}(t)} \log f_t(\mathbf{x}(t)|\mathbf{x}(0))$. However,

$$\mathbf{x}^{\boldsymbol{\theta}} = \mathbf{x}(t) + \tilde{\sigma}(t)^2 s^{\boldsymbol{\theta}}(\mathbf{x}(t), t), \tag{3.72}$$

meaning that the score is calculated based on an estimation of $\mathbf{x}(0)$ that depends on the network output times $\tilde{\sigma}(t)$. By using Equation (3.69) it is possible to calculate a minimum $c_{\text{out}}(\tilde{\sigma}(t)) > 0$ that multiplies the new neural network output $F^{\boldsymbol{\theta}}(c_{\text{in}}(\tilde{\sigma}(t))\mathbf{x}(t), c_{\text{noise}}(\tilde{\sigma}(t)))$.

Before finding the best $c_{\text{out}}(\tilde{\sigma}(t))$, first the variance of $F_{\text{target}}(\mathbf{x}(t), t)$ must be set to one. This leads to [70]

$$\begin{aligned}
&\text{Cov}\left[\frac{\mathbf{x}(0) - c_{\text{skip}}(\tilde{\sigma}(t))\mathbf{x}(t)}{c_{\text{out}}(\tilde{\sigma}(t))}\right] = \mathbf{I} \\
&\implies \text{Cov}\left[\frac{\mathbf{x}(0) - c_{\text{skip}}(\tilde{\sigma}(t))\left(\mathbf{x}(0) + \mathbf{n}\right)}{c_{\text{out}}(\tilde{\sigma}(t))}\right] = \mathbf{I} \\
&\implies \frac{(1 - c_{\text{skip}}(\tilde{\sigma}(t))^2\boldsymbol{\Sigma}(0) + c_{\text{skip}}(\tilde{\sigma}(t))^2\tilde{\sigma}(t)^2}{c_{\text{out}}(\tilde{\sigma}(t))^2} = \mathbf{I} \\
&\implies c_{\text{out}}(\tilde{\sigma}(t))^2 = (1 - c_{\text{skip}}(\tilde{\sigma}(t)))^2\sigma_{\text{data}}^2 + c_{\text{skip}}(\tilde{\sigma}(t))^2\tilde{\sigma}(t)^2,
\end{aligned} \tag{3.73}$$

where $\boldsymbol{\Sigma}(0) = \sigma_{\text{data}}^2\mathbf{I}$ was used in the last passage.

Then $c_{\text{skip}}(\tilde{\sigma}(t))$ can be manipulated to minimize $c_{\text{out}}(\tilde{\sigma}(t))$. Because $c_{\text{out}}(\tilde{\sigma}(t))$ is simply a scaling factor greater than zero, minimizing it is equivalent to minimizing

its square. Furthermore, $c_{\text{out}}(\tilde{\sigma}(t))^2$ is convex on $c_{\text{skip}}(\tilde{\sigma}(t))$, and therefore [70]

$$\begin{aligned}
&\left.\frac{\mathrm{d}\left[c_{\text{out}}(\tilde{\sigma}(t))^2\right]}{\mathrm{d}c_{\text{skip}}(\tilde{\sigma}(t))}\right|_{c_{\text{skip}}(\tilde{\sigma}(t))=c^*_{\text{skip}}(\tilde{\sigma}(t))} = 0 \\
&\implies -2\left(1 - c^*_{\text{skip}}(\tilde{\sigma}(t))\right)\sigma^2_{\text{data}} + 2c^*_{\text{skip}}(\tilde{\sigma}(t))\tilde{\sigma}(t)^2 = 0 \\
&\implies c^*_{\text{skip}}(\tilde{\sigma}(t)) = \frac{\sigma^2_{\text{data}}}{\sigma^2_{\text{data}} + \tilde{\sigma}(t)^2}. \qquad (3.74)
\end{aligned}$$

Substituting Equation (3.74) into Equation (3.73) gives [70]

$$\begin{aligned}
c_{\text{out}}(\tilde{\sigma}(t))^2 &= \left(1 - \left(\frac{\sigma^2_{\text{data}}}{\sigma^2_{\text{data}} + \tilde{\sigma}(t)^2}\right)^2\right)\sigma^2_{\text{data}} + \left(\frac{\sigma^2_{\text{data}}}{\sigma^2_{\text{data}} + \tilde{\sigma}(t)^2}\right)^2\tilde{\sigma}(t)^2 \\
&= \frac{\tilde{\sigma}(t)^4\sigma^2_{\text{data}} + \sigma^4_{\text{data}}\tilde{\sigma}(t)^2}{\left(\sigma^2_{\text{data}} + \tilde{\sigma}(t)^2\right)^2} \\
&= \frac{\sigma^2_{\text{data}}\tilde{\sigma}(t)^2\left(\sigma^2_{\text{data}} + \tilde{\sigma}(t)^2\right)}{\left(\sigma^2_{\text{data}} + \tilde{\sigma}(t)^2\right)^2} \\
\implies c_{\text{out}}(\tilde{\sigma}(t)) &= \frac{\sigma_{\text{data}}\tilde{\sigma}(t)}{\sqrt{\sigma^2_{\text{data}} + \tilde{\sigma}(t)^2}}. \qquad (3.75)
\end{aligned}$$

The authors of [70] determine the value of $c_{\text{noise}}(\tilde{\sigma}(t))$ empirically as $\frac{1}{4}\ln(\tilde{\sigma}(t))$. They also set $w(t) = 1$, which is equivalent to adopting $\lambda(t) = \frac{\tilde{\sigma}(t)^2(\sigma^2_{\text{data}}+\tilde{\sigma}(t)^2)}{\sigma^2_{\text{data}}}$. With the weights of all noise levels being equal, the best way to emphasize certain noise levels during training is by changing $\tilde{\sigma}(t)$ used during training, similarly to what was suggested for the inference. Then, t can be sampled uniformly, as was done in Algorithm 3.1. Alternatively, one could change $\lambda(t)$ and use uniformly sampled noise levels, but this was not explored in [70].

Algorithm 3.9: SDE Diffusion Training

while $||\boldsymbol{\theta}_{\text{new}} - \boldsymbol{\theta}_{\text{old}}||^2 \leq$ tolerance **do**
- $\mathbf{x}_0 \sim f_0(x(0))$;
- $t \sim \mathcal{U}(0, T)$;
- $\mathbf{z} \sim \mathcal{N}(0, \tilde{\sigma}(t)\mathbf{I})$;
- $\mathbf{x}_i = \mathbf{x}_0 + \mathbf{z}$;
- $\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_{\text{old}} - \eta\nabla_{\boldsymbol{\theta}}\left|\left|F^{\boldsymbol{\theta}}(c_{\text{in}}(\tilde{\sigma}(t))\mathbf{x}(t), c_{\text{noise}}(\tilde{\sigma}(t))) - \frac{\left(\mathbf{x}(0) - c_{\text{skip}}\tilde{\sigma}(t)\mathbf{x}(t)\right)}{c_{\text{out}}\tilde{\sigma}(t)}\right|\right|^2$;

end

return $\boldsymbol{\theta}_{\text{new}}$;

In [70] the authors show empirically that choosing $\tilde{\sigma}(t)$ during training such that $\ln(\tilde{\sigma}(t))$ is Gaussian for uniformly drawn t enhances sample quality if combined with the adjustments of Equation (3.69). In [17], an audio focused diffusion work, the authors prefer to use $\tilde{\sigma}(t)$ of Equation (3.63) for both training and sampling.

Algorithm 3.9 shows the training algorithm for diffusion models using the techniques of [70]. In it, $\tilde{\sigma}(\cdot)$, $c_{\text{in}}(\cdot)$, $c_{\text{out}}(\cdot)$, $c_{\text{skip}}(\cdot)$ and $c_{\text{noise}}(\cdot)$ correspond to function calls for calculating each of those values according to their formulae. It is very similar to Algorithm 3.3, with the difference that continuous noise levels are used, and that the loss function is parameterized using $F^{\boldsymbol{\theta}}(\cdot,\cdot)$, and the terms $\tilde{\sigma}(\cdot)$, $c_{\text{in}}(\cdot)$, $c_{\text{out}}(\cdot)$, $c_{\text{skip}}(\cdot)$ and $c_{\text{noise}}(\cdot)$.

3.3.2 Conditional Generation

Up until now, sampling with diffusion models has only been described for unconditional generation, that is, generating a point $\mathbf{x}$ by approximating the distribution $f(\mathbf{x})$ of all data points used during training. However, one might be interested in conditional generation, i.e., generating samples by approximating the distribution $f(\mathbf{x}|\mathcal{Y})$ of the data points given the random variable $\mathcal{Y}$.

When $\mathcal{Y}$ is discrete, it can encode a class $\mathcal{C}$ to which $\mathbf{x}$ belongs, for example. In the context of audio restoration, $\mathcal{Y} = \mathbf{y}$ with continuous magnitude can model a degraded version of the signal to be restored, making $f(\mathbf{x}|\mathbf{y})$ the distribution of possible restored signals given the corrupted version. In this approach $\mathbf{y}$ can be a generic corrupted signal, and the same technique can be adapted to different restoration problems [17].

If the initial distribution for the forward diffusion process is $f_0(\mathbf{x}(0)|\mathcal{Y})$ instead of $f_0(\mathbf{x}(0))$, then the distribution at time t will be $f_t(\mathbf{x}(t)|\mathcal{Y})$ instead of $f_t(\mathbf{x}(t))$, and the backward diffusion equation will require $\nabla_{\mathbf{x}(t)} \log f_t(\mathbf{x}(t)|\mathcal{Y})$ instead of $\nabla_{\mathbf{x}(t)} \log f_t(\mathbf{x}(t))$ [7]. The backward SDE of Equation 3.35 becomes

$$\begin{aligned}
\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} &= F(\mathbf{x}(t), t) - g(t)^2 \nabla_{\mathbf{x}(t)} \log f_t(\mathbf{x}(t)|\mathcal{Y}) + g(t)\mathbf{z} \\
&= F(\mathbf{x}(t), t) - g(t)^2 \nabla_{\mathbf{x}(t)} \log \frac{f(\mathcal{Y}|\mathbf{x}(t)) f_t(\mathbf{x}(t))}{f(\mathcal{Y})} + g(t)\mathbf{z} \\
&= F(\mathbf{x}(t), t) - g(t)^2 \left[\nabla_{\mathbf{x}(t)} \log f_t(\mathbf{x}(t)) + \nabla_{\mathbf{x}(t)} \log f(\mathcal{Y}|\mathbf{x}(t))\right] + g(t)\mathbf{z}. \quad (3.76)
\end{aligned}$$

The term $\nabla_{\mathbf{x}(t)} \log f_t(\mathbf{x}(t))$ can be approximated using $F^{\boldsymbol{\theta}}(\cdot,\cdot)$ trained for unconditional sampling. Therefore, conditional sampling can be made from a pre-trained unconditional diffusion model, as long as an estimate for $\nabla_{\mathbf{x}(t)} \log f(\mathcal{Y}|\mathbf{x}(t))$ is available.

If $\mathcal{Y}$ represents a class, and the pairs $(\mathbf{x}(0), \mathcal{Y})$ are available, then modelling $\nabla_{\mathbf{x}(t)} \log f(\mathcal{Y}|\mathbf{x}(t))$ is fairly straightforward [7]. The forward diffusion equation can be used to obtain pairs $(\mathbf{x}(t), \mathcal{Y})$, and a discriminative model can be trained to obtain $f(\mathcal{Y}|\mathbf{x}(t))$.

For a general $\mathcal{Y}$, estimating $\nabla_{\mathbf{x}(t)} \log f(\mathcal{Y}|\mathbf{x}(t))$ is not so simple. In the context

of audio restoration, two methods for estimating $\nabla_{\mathbf{x}(t)} \log f(\mathcal{Y}|\mathbf{x}(t))$ have been used successfully for different restoration tasks [17]. They are explained in more details below.

Data Consistency Method

The first of the two conditional generation techniques, the data consistency (DC) method [16, 17], is fairly straightforward. The score $\nabla_{\mathbf{x}(t)} \log f(\mathbf{x}(t)|\mathbf{y})$ is once again approximated by Equation (3.66), but with an alternative estimate $\bar{\mathbf{x}}^{\boldsymbol{\theta}}$ of $\mathbf{x}(0)$, given by $\bar{\mathbf{x}}^{\boldsymbol{\theta}} = \mathbf{y} + \mathbf{x}^{\boldsymbol{\theta}} - \mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}})$ — with $\mathcal{A}$ being the degradation applied to the signal — used in place of $\mathbf{x}^{\boldsymbol{\theta}}$. Note that $\mathbf{y} = \mathcal{A}(\mathbf{x}(0))$.

As $\bar{\mathbf{x}}^{\boldsymbol{\theta}}$ is an estimate of the clean, non-degraded, signal $\mathbf{x}(0)$, using the DC method implies keeping the parts of $\mathbf{y}$ that were not altered by $\mathcal{A}(\cdot)$, and using $\mathbf{x}^{\boldsymbol{\theta}}$ to estimate the degraded parts.

Audio inpainting, or gap filling, is a good example to understand the intuition behind the DC method. For inpainting, the degradation $\mathcal{A}(\mathbf{x}(0))$ is a mask that removes certain elements from $\mathbf{x}(0)$. Thus $\mathbf{x}^{\boldsymbol{\theta}} - \mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}})$ only contains the elements of $\mathbf{x}^{\boldsymbol{\theta}}$ that are masked by $\mathcal{A}(\cdot)$; the unmasked elements are taken directly from $\mathbf{y}$.

It is important to note that this line of thought does not apply to any $\mathcal{A}(\cdot)$. If $\mathcal{A}(\cdot)$ is random, for example, then $\bar{\mathbf{x}}^{\boldsymbol{\theta}}$ will not be necessarily consistent with $\mathbf{x}(0)$.

An interesting aspect about the DC method is that, since it adopts

$$\nabla_{\mathbf{x}(t)} \log f(\mathbf{x}(t)|\mathbf{y}) = \frac{\bar{\mathbf{x}}^{\boldsymbol{\theta}} - \mathbf{x}(t)}{\tilde{\sigma}(t)^2}, \tag{3.77}$$

it implicitly assumes that $f(\mathbf{x}(t)|\mathbf{y}) = \mathcal{N}(\bar{\mathbf{x}}^{\boldsymbol{\theta}}, \tilde{\sigma}(t)^2\mathbf{I})$. Furthermore,

$$\nabla_{\mathbf{x}(t)} \log f(\mathbf{x}(t)|\mathbf{y}) = \underbrace{\frac{\mathbf{x}^{\boldsymbol{\theta}} - \mathbf{x}(t)}{\tilde{\sigma}(t)^2}}_{\nabla_{\mathbf{x}(t)} \log f_t(\mathbf{x}(t))} + \underbrace{\frac{\mathbf{y} - \mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}})}{\tilde{\sigma}(t)^2}}_{\nabla_{\mathbf{x}(t)} \log f(\mathbf{y}|\mathbf{x}(t))}, \tag{3.78}$$

which means that the DC method is also equivalent to taking

$$\nabla_{\mathbf{x}(t)} \log f(\mathbf{y}|\mathbf{x}(t)) = \frac{\mathbf{y} - \mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}})}{\tilde{\sigma}(t)^2}. \tag{3.79}$$

This can be seen as assuming $f(\mathbf{y}|\mathbf{x}(t)) = \mathcal{N}(\mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}}(\mathbf{x}(t))), \tilde{\sigma}(t)^2\mathbf{I})$ jointly with $\nabla_{\mathbf{x}(t)}\mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}}(\mathbf{x}(t))) \approx 1$. In the last sentence, the dependence of $\mathbf{x}^{\boldsymbol{\theta}}$ in $\mathbf{x}(t)$ was made explicit.

Reconstruction Guidance Method

The reconstruction guidance (RG) method [17, 74] takes a different approach and uses

$$\nabla_{\mathbf{x}(t)} \log f(\mathbf{y}|\mathbf{x}(t)) = \xi(t) \nabla_{\mathbf{x}(t)} ||\mathbf{y} - \mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}}(\mathbf{x}(t)))||^2, \tag{3.80}$$

where $\xi(t)$ is a function of time to be chosen by the user. The underlying assumption in the RG method is also that $f(\mathbf{y}|\mathbf{x}(t)) = \mathcal{N}(\mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}}(\mathbf{x}(t))), \tilde{\sigma}(t)^2\mathbf{I})$, and the term $\xi(t)$ is there to correct errors with this approximation if needed.

Unlike the data consistency method, reconstruction guidance numerically calculates $\nabla_{\mathbf{x}(t)} ||\mathbf{y} - \mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}}(\mathbf{x}(t)))||^2$, and does not assume $\nabla_{\mathbf{x}(t)} \mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}}(\mathbf{x}(t))) \approx 1$. Furthermore, it is compatible with random $\mathcal{A}(\cdot)$, as will be seen in Chapter 4.

In [74], the original paper suggesting the RG method, $\xi(t) = -\frac{1}{\tilde{\sigma}(t)^2}$. This makes

$$\nabla_{\mathbf{x}(t)} \log f_t(\mathbf{x}(t)|\mathbf{y}) = \frac{\overbrace{2\left[\nabla_{\mathbf{x}(t)} \mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}})\right] \mathbf{y} + \mathbf{x}^{\boldsymbol{\theta}} - 2\left[\nabla_{\mathbf{x}(t)} \mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}})\right] \mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}})}^{\bar{\mathbf{x}}^{\boldsymbol{\theta}}} - \mathbf{x}(t)}{\tilde{\sigma}(t)^2}, \tag{3.81}$$

and therefore this choice of $\xi(t)$ implies Gaussian $f_t(\mathbf{x}(t)|\mathbf{y})$ like the DC method, but with adjusted $\bar{\mathbf{x}}^{\boldsymbol{\theta}} = 2\left[\nabla_{\mathbf{x}(t)} \mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}})\right] \mathbf{y} + \mathbf{x}^{\boldsymbol{\theta}} - 2\left[\nabla_{\mathbf{x}(t)} \mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}})\right] \mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}})$. An important detail is that the authors of [74] only worked with inverse problems on images, and did not examine audio inverse problems.

The authors of [17] worked on audio inverse problems, and empirically showed that

$$\xi(t) = \frac{-\xi' \sqrt{N}}{\tilde{\sigma}(t) ||\nabla_{\mathbf{x}(t)}||\mathbf{y} - \mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}})||^2||^2}, \tag{3.82}$$

with N being the length of the generated audio in samples and ξ' being a constant to be set by the user, produced results comparable to the state of the art for bandwidth extension, audio inpainting, and declipping. Their choice of ξ' is dependent on the inverse problem being solved. For bandwidth extension, they use $\xi' = 0.1$, while for inpainting and audio declipping $\xi' = 0.35$ is used.

Both choices of $\xi(t)$ increase guidance as the reverse diffusion approaches $t = 0$. The second choice forces the norm of $\nabla_{\mathbf{x}(t)} \log f(\mathbf{y}|\mathbf{x}(t))$ to be $\frac{\xi' \sqrt{N}}{\tilde{\sigma}(t)}$, which can be an advantage if $\nabla_{\mathbf{x}(t)} ||\mathbf{y} - \mathcal{A}(\mathbf{x}^{\boldsymbol{\theta}})||^2$ is excessively small or excessively large in comparison with $\nabla_{\mathbf{x}(t)} \log f(\mathbf{x}(t))$. However, it adds a parameter to the sampling process.

The DC and RG methods can be combined as well [17]. To do this, the procedure is: obtaining the score $\nabla_{\mathbf{x}(t)} \log f_t(\mathbf{x}(t)|\mathbf{y})$ with the RG method; converting the score into a new $\tilde{\mathbf{x}}^{\boldsymbol{\theta}} = \mathbf{x}(t) + \tilde{\sigma}(t)^2 \nabla_{\mathbf{x}(t)} \log f_t(\mathbf{x}(t)|\mathbf{y})$; and recalculating the score with $\bar{\mathbf{x}}^{\boldsymbol{\theta}} = \mathbf{y} + \tilde{\mathbf{x}}^{\boldsymbol{\theta}} - \mathcal{A}(\tilde{\mathbf{x}}^{\boldsymbol{\theta}})$. In [17] the authors show that combining the two methods improves results for audio inpainting, but not for bandwidth extension.

3.4 Model Architecture and Training in this Work

In this work, the CQTDiff neural network architecture was used as $F^{\boldsymbol{\theta}}(\cdot, \cdot)$. This architecture was introduced in [17], and is an adaptation of a U-Net [32] enclosed with differentiable implementations of the CQ-NSGT and its inverse (as described in Section 2.4).

This section is divided in two parts. The first describes the network architecture used in this work with a bottom-up approach, that is, explaining first the blocks that compose the network and then giving an overview of the full architecture. The second part gives details on the training dataset and explains the adopted training regime.

3.4.1 Neural Network Architecture

As mentioned above, this work adopts the CQTDiff U-Net as the backbone of an unconditional diffusion model. U-Nets in general are widely popular in applications for both image [5, 7, 14, 32] and audio [17, 31], and are a special type of convolutional autoencoder [19]. Like traditional convolutional autoencoders, U-Nets are composed of an encoder — responsible for mapping an input into a latent space of lower dimension — and a decoder — responsible for reconstructing the latent representation as the input. In U-Nets, however, the pooling and unpooling operations used in the encoder and decoder are replaced by downsampler and upsampler blocks, meaning that the latent space representation of U-Nets can be interpreted as downsampled versions of the input (to some extent).

In this sense, the inner structure of the CQTDiff architecture is not very different from other U-Nets. It uses downsampling and upsampling structures to map inputs to and from a latent space of lower dimensions. However, in the CQTDiff architecture, the U-Net inputs are based on CQ-NSGTs of the input, which provides advantages related to the logarithmic resolution of these representations.

Because musical notes are geometrically spaced in frequency (in the equal tempered scale), logarithmic resolution allows one to represent the full range of audible musical notes[9] while avoiding time-frequency resolution issues to some extent. Since bin width decreases with frequency, it is possible to represent both bass and treble notes sharply with good time resolution in the upper frequency range. It is worth noting that, as discussed in Section 2.4, the perfect invertibility of the CQ-NSGT requires redundancy, which implies extra computational cost.

Broadly speaking, the CQTDiff architecture can be divided in four major blocks: the signal representation block, the noise level embedding block, the downsampler,

[9] Assuming there are no limitations regarding the sampling rate.

and the upsampler. These four blocks are described in order below, and an overview of the complete architecture (which can be seen for reference in Figure 3.8) is shown afterwards.

Signal Representation Block

As the CQTDiff U-Net is enclosed with CQ-NSGT and inverse CQ-NSGT blocks, the inputs of its inner blocks are based on constant-Q time-frequency representations. These representations are first processed by the signal representation block, and then are passed along to the inner U-Net blocks of CQTDiff.

Magnitude and phase of the constant-Q time-frequency representations are considered two independent channels in the signal representation block. Because of this, just after the transform, the input has shape $(K, N, 2)$, where K is the number of bins used, and N is the number of frames. Here and in [17], seven octaves are used, with 64 bins in each of them.

With the CQ-NSGT alone, there is no inherent ordering of frequency bins in the inner neural network blocks. To the neural network, the values in each bin are just entries in a matrix, and no distinction is made between low and high frequencies. To solve this, frequency embeddings for each bin are concatenated to the signal time-frequency representation.

These embeddings are obtained similarly to random Fourier features (RFF) [75]. For a vector input $\mathbf{v} \in \mathbb{R}^d$, traditional RFFs are obtained by calculating the vector

$$\mathbf{r} = [\cos(2\pi \mathbf{B}\mathbf{v}), \sin(2\pi \mathbf{B}\mathbf{v})]\,, \tag{3.83}$$

where $\mathbf{B} \in \mathbb{R}^{m \times d}$ is a matrix with elements sampled from $\mathcal{N}(0, \sigma^2)$. The brackets indicate concatenation between the two vector elements, and the sine and cosine functions are applied to each element in the vector. Also, $2m$ is the desired feature dimensionality, and σ is a parameter to be chosen by the user.

In the CQTDiff architecture, frequency embeddings are calculated adapting traditional RFFs to incorporate frequency bin ordering. For each bin index k, an embedding $\mathbf{r} \in \mathbb{R}^{2m}$ is calculated with

$$\mathbf{r} = [\cos(2\pi \mathbf{b}k), \sin(2\pi \mathbf{b}k)]\,, \tag{3.84}$$

where $\mathbf{b} \in \mathbb{R}^m$ is a vector with elements sampled from $\mathcal{N}(0, m^2)$. In [17], $m = 16$ for the frequency embeddings, which makes the network input have shape $(K, N, 34)$. The embeddings calculated with this formula are updated through backpropagation during training.

Including any function of k alone as a frequency embedding is necessary to

allow the neural network to be able to distinguish between different regions of the spectrum. Using RFFs as a function of the bin index k as in Equation (3.84) has the advantage of making the internal product of the embeddings of any two bins k_1 and k_2 Gaussian-shaped with peak at $k_1 = k_2$. This is desirable because it makes adjacent bins — representing adjacent notes, or fractions of notes — highly correlated.

An intuition for why this formulation has this property can be seen for the case where $m = 1$. In this case, noticing that $b \sim \mathcal{N}(0, 1)$, and using the formula for the characteristic function of a Gaussian random variable [76]

$$\begin{aligned}
\mathbb{E}_b\left[\mathbf{r}_1^T \mathbf{r}_2\right] &= \mathbb{E}_b\left[\cos(2\pi b k_1)\cos(2\pi b k_2) + \sin(2\pi b k_1)\sin(2\pi b k_2)\right] \\
&= \mathbb{E}_b\left[\cos(2\pi b(k_1 - k_2)\right] \\
&= \frac{1}{2}\,\mathbb{E}_b\left[\mathrm{e}^{j2\pi b(k_1-k_2)} + \mathrm{e}^{-j2\pi b(k_1-k_2)}\right] \\
&= \mathrm{e}^{-2\pi^2(k_1-k_2)^2}.
\end{aligned} \tag{3.85}$$

After concatenating the frequency embeddings, an initial convolutional layer with a $(5, 3)$ kernel is applied to generate the signal representation used in the following blocks. Zero-padding is applied to the input so that tensor dimensions remain unchanged. The complete signal representation block is shown in Figure 3.3. In the diagram, $\mathbf{x}(t)$ is pre-multiplied by $c_{\text{in}}(\tilde{\sigma}(t))$ according to the heuristics seen in Section 3.3.1.

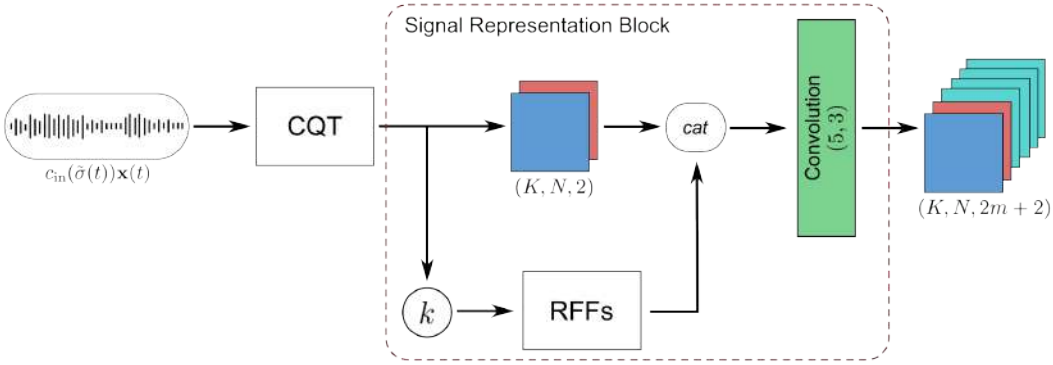

Figure 3.3: Signal representation block in CQTDiff. The blue and red squares represent the real and imaginary part of the CQ-NSGT, k represents the bin indexes, and *cat* represents tensor concatenation. For this block $m = 16$.

Noise Level Embedding Block

To create embeddings for the noise level, $c_{\text{noise}}(\tilde{\sigma}(t)) = \frac{1}{4}\ln(\tilde{\sigma}(t))$ is used as input to traditional RFFs followed by a MLP. In this case, the RFF input is a scalar,

and so $\mathbf{B} \in \mathbb{R}^{m \times 1}$. In [17], $m = 32$ for the noise embeddings, which makes the MLP input a vector with dimension 64. The MLP is composed of three linear layers with dimensions $(64, 128), (128, 256), (256, 512)$, each followed by a rectified linear unit (ReLU) [19] non-linearity. The full noise level embedding block is shown in Figure 3.4.

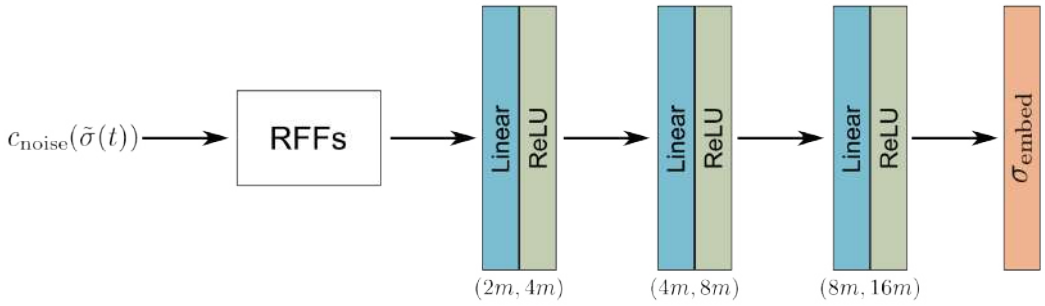

Figure 3.4: Noise level embedding in CQTDiff. For this block $m = 32$, and σ_{embed} represents the noise level embeddings.

Noise embeddings are included in the downsampler and upsampler blocks through feature-wise linear modulation (FiLM) [77] layers. In the most general setting, FiLM layers combine an input $\mathbf{x}$ and a conditioning input $\mathbf{z}$ linearly through

$$\text{FiLM}(\mathbf{x}, \mathbf{z}) = \alpha(\mathbf{z}) \odot \mathbf{x} + \beta(\mathbf{z}), \tag{3.86}$$

where $\odot$ denotes pointwise multiplication, and where $\alpha(\mathbf{z})$ and $\beta(\mathbf{z})$ are neural networks. Usually, $\alpha(\cdot)$ and $\beta(\cdot)$ are trained together as a single neural network whose output is divided in parts of equal size. In [17] and here, $\mathbf{z}$ is the noise level embedding, $\alpha(\cdot)$ is a linear layer, and $\beta(\mathbf{z}) = 0, \forall \mathbf{z}$.

Downsampler and Upsampler Blocks

The downsampler and upsampler blocks both rely on residual blocks (RBlocks). Each RBlock takes an input $\mathbf{x}$, and combines it with the noise level embedding σ_{embed} using a FiLM layer. The output $\mathbf{y}$ of the FiLM layer is fed to a Gaussian error linear unit (GELU) [78] non-linearity[10] followed by a linear layer. The GELU operation is defined as

$$\text{GELU}(\mathbf{y}) = \mathbf{y} \odot \Phi(\mathbf{y}), \tag{3.87}$$

where $\Phi(\cdot)$ is the cumulative distribution function of $\mathcal{N}(0, 1)$. Note that $\Phi(\cdot)$ is applied to each element of $\mathbf{y}$

The result then goes through a sequence of eight blocks made of GELU non-linearities followed by convolutions dilated in the frame dimension, each of which

[10] Using the default implementation in *pytorch*.

has an additive skip connection between input and output. The convolutions use a $(5, 3)$ kernel, and zero-padding is applied to avoid changing input dimensions. Dilation increases by a factor of two for every convolution, meaning that the first convolution has dilation factor $(1, 1)$, and the last convolution has dilation factor $(1, 256)$. The output of this block stack is summed to $\mathbf{x}$ to produce the output of the residual block. Figure 3.5 shows a diagram with the full RBlock.

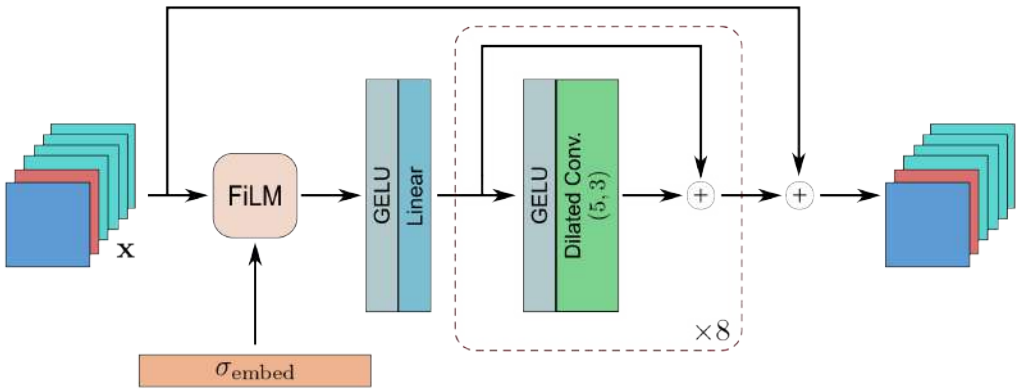

Figure 3.5: Residual block in CQTDiff.

The downsampler blocks (Figure 3.6) propagate the signal through the U-Net via two paths: a factor-2 decimator followed by a linear layer (left path), and a residual block followed by factor-2 decimation (right path). Both decimation operations are coupled with low-pass filtering to avoid aliasing.

The input of the right paths is the sum of the outputs of both paths in the previous downsampler block; in the first layer, it is simply the output of the signal representation block. The input of the left paths is the output of the previous left path; in the first layer, it is the $(K, N, 2)$ CQ-NSGT of the input. This implies that the linear layer maps the real and imaginary parts of a downsampled CQ-NSGT into 34 features that can be added to the output of the RBlock. Figure 3.6 shows the complete downsampler block.

The upsampler blocks (Figure 3.7) are the dual of the downsampler blocks, and therefore use factor-2 interpolations instead of decimations. In the upsampler blocks, the linear layers map the features coming out of the RBlocks into two features representing the real and imaginary parts of the CQT spectrogram. This output is added to the previous upsampler's right path output.

Also, the RBlock inputs in the upsampler blocks are the concatenation of the previous upsampler's left path output and the output of the equivalent downsampler's RBlock. This is done by using skip connections to bridge downsampler and upsampler blocks of equivalent resolution. Figure 3.7 shows the upsampler block.

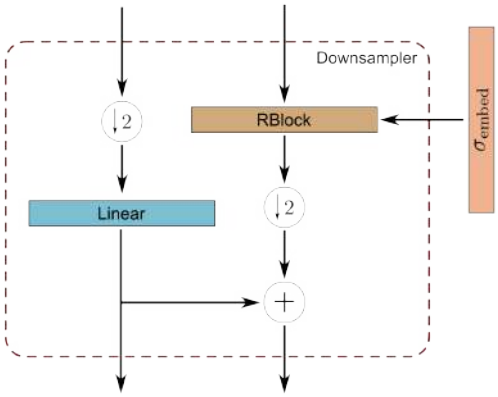

Figure 3.6: Downsampler block in CQTDiff. All decimation operations include anti-aliasing filtering.

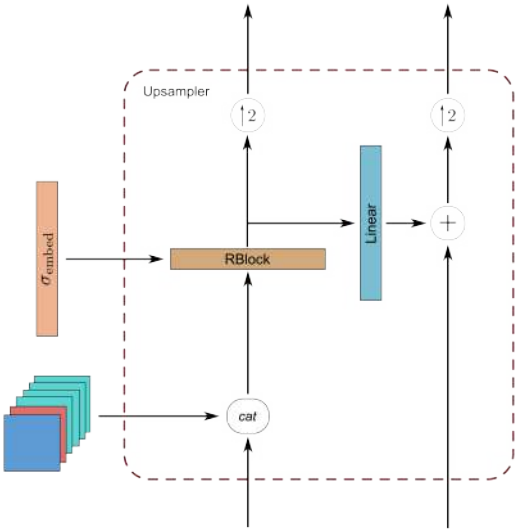

Figure 3.7: Upsampler block in CQTDiff. The tensor input on the left comes from a skip connection with the corresponding downsampler block.

Overview

To obtain the U-Net output, the right hand output of the last upsampler block goes through an inverse CQ-NSGT. As long as the number of downsampler and upsampler blocks is the same, the result of this operation will be a vector with the same shape as $c_{\text{in}}(\tilde{\sigma}(t))\mathbf{x}(t)$.

The middle of the U-Net is composed of three RBlocks, with the top two being bridged by a skip connection. The right and left paths in the first upsampler block receive the output of this trio as input. However, a linear mapping is required for the right side, as its dimension is supposed to have size two. Figure 3.8 shows the full CQTDiff architecture.

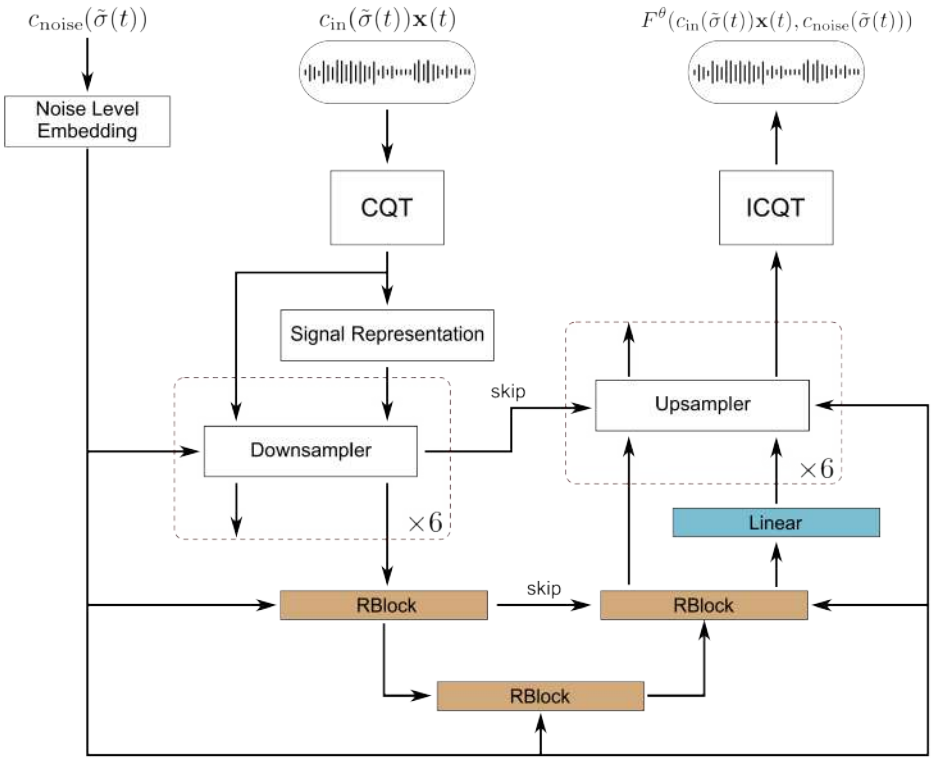

Figure 3.8: Complete CQTDiff architecture. In the last downsampler and upsampler blocks, only the right paths are used in the blocks following.

3.4.2 Training Details

In this work, an unconditional diffusion model using the CQTDiff architecture was trained using the MAESTRO dataset [79]. The MAESTRO dataset is a collection of about 200 hours of recordings of classical piano played by different performers in Yamaha Disklaviers.

Yamaha Disklaviers grand pianos are capable of registering a performance in MIDI[11] format, and MAESTRO was originally meant to be a dataset of audio recordings paired with corresponding MIDI transcriptions. However, its sheer size makes it a good candidate for training generative models of piano as well.

The MAESTRO dataset recordings are all 16-bit PCM, with sampling rate of 44.1kHz or 48kHz. Before training, all 48kHz pieces were resampled to 44.1kHz using the *sox* command line tool [80].

MAESTRO is originally split into train, validation and test sets, but they were all mixed for training the CQTDiff architecture. None of the MAESTRO recordings were used during validation or testing; however, four musical pieces in the test sets have alternative versions in MAESTRO (played by different artists on different

[11] Musical Instrument Digital Interface, or MIDI, is a standard for digitally representing musical notes. For more information, see [49].

instruments and in different acoustical settings).

Considering the very different recording conditions in MAESTRO[12], and the fact that small audio segments were used during training, treating this as a test data leakage is debatable. Still, the results of Sections 4.4 and 4.5 were calculated including and excluding the excerpts extracted from these pieces, reporting differences whenever necessary.

In [17], the MAESTRO dataset is used to train an unconditional model based on CQTDiff. However, the authors chose to resample the training set to 22.05kHz, and only performed experiments using this sample rate. High-quality audio is sampled at a rate of at least 44.1kHz, and so the goal of the training procedure here was to obtain an unconditional model similar to the one in [17], but capable of generating raw audio at a higher sampling rate.

In addition, it is important to have a high sampling rate unconditional model to properly restore historic 78 RPM recordings. Although high-frequency content may appear severely attenuated in historical recordings, it is still present. Training an unconditional model with low sampling rate for usage in this restoration context would mean giving up recovering high-frequency content originally recorded.

As in [17], training was made with Algorithm 3.9. The noise schedule was set using following Equation (3.63), with $\sigma_{\min} = 10^{-6}$, $\sigma_{\max} = 10$, and $\rho = 10$. For simplicity, $T = 1$. It is important to note that Algorithm 3.9 implicitly sets $s(t) = 1$.

In [17], eight random samples of approximately 3s were extracted from a randomly selected recording of the MAESTRO train split each time the dataset was accessed. Their batch size is set to four, meaning that a single dataset access builds two batches. They train the model for 320,000 iterations, and use a learning rate of 2×10^{-4}, with a decay rate of 0.8 every 60,000 iterations. The final network weights are averaged with an exponential moving average[13] of rate 0.9999.

Due to computational constraints, reproducing this exact training setup was not possible in this work. Using samples of around 3s at a sampling rate of 44.1kHz with a batch size of four required more memory than was available. The criterion for changing the batch size and sample length was trying to guarantee the same number of seconds of audio seen in [17] during training, despite the change in sample rate.

[12]Three of the four overlapping test pieces were historical recordings from before 1950, while MAESTRO recordings are from after the year 2000.

[13]As stochastic gradient descent tends to produce noisy estimates of the parameters, smoothing through averaging is a common procedure to improve network performance [81]. In exponential moving averaging of the network parameters, the final version of parameter p is given by

$$p = (1 - \alpha) \sum_{i=1}^{N} \alpha^{N-i} p_i, \tag{3.88}$$

where α is the moving average rate, N is the total number of iterations, i is the iteration number, and p_i is the value of the parameter at iteration i.

In this work, audio samples of 1.5s were used during training. Since the original training setup had $3 \times 4 \times 320{,}000 = 3{,}840{,}000$ seconds of audio, satisfying the equivalent length criterion with a batch size of one required $3{,}840{,}000/1.5 = 2{,}560{,}000$ iterations.

A first model was trained using these smaller batch size and sample length while keeping the rest of the setup of [17]. For this model, using the inference parameters of Chapter 4[14], no significant difference was observed in the outputs after iteration number 2,303,999, and so training stopped at this iteration. Total training time using a NVIDIA GeForce RTX 2080 was a little less than a month, and required around 11Gb of GPU memory. Inference was possible using smaller audio blocks with around 8Gb of GPU memory.

A second model was also trained adopting a similar proportionality criterion for adapting the learning rate decay schedule. Since in [17] the learning rate was updated in steps of $60{,}000/320{,}000 = 18.75\%$ of the total training time, this second model had a decay schedule of $18.75\% \times 2{,}560{,}000 = 480{,}000$ steps.

The second model was trained for 3,072,023 iterations (more than the 2,560,000 that would be equivalent to [17]), but outputted degenerate, noise-like, audio samples when using the inference parameters of Chapter 4. The smaller batch size might have made the gradients too noisy, which was compensated by a faster learning rate decay in the first model. In the end, only the first model was adopted, but memory requirements for the second model were similar to the first model, with training taking about two weeks longer.

Other training parameters were kept the same as in [17], in order to reproduce the original CQTDiff model for higher sampling rates. Sampling parameters were mostly kept the same, as will be seen in Chapter 4.

[14]Which were the same as in [17], except for $\sigma_{\min}$ and $\sigma_{\max}$ covered during training

Chapter 4

Audio Denoising with Diffusion Models

In [17] the authors study the use of diffusion models for solving different inverse problems for audio signals. Using a single pre-trained, unconditional, diffusion model, they try to solve the problems of audio bandwidth extension, declipping, and gap filling.

For all three problems, conditional sampling with diffusion obtains competitive results. Audio bandwidth extension with diffusion is evaluated objectively[1] and subjectively for signals low-passed at 1kHz and 3kHz, and the method proposed in [17] outperforms the generative models of [82] and [83]. For inpainting, a subjective test asking volunteers to evaluate the plausibility of gap fills created with conditional sampling using diffusion shows that the technique of [17] is comparable to that of [84] and [85]. Finally, an objective test with psyacoustically inspired objective audio quality metrics shows that diffusion declipping is comparable to traditional sparsity based methods [86] and [87] for highly distorted signals.

The biggest innovation in [17] is that each of the problems addressed in the paper required specialized models before, whereas with their solution, a single diffusion model is capable of solving them all without requiring new data or training. The downside, however, is that specific information about the degradation $\mathcal{A}(\cdot)$ applied to the signal must be known beforehand. In their paper, the low-pass filter cutoffs, clip thresholds, and gap positions are given to the diffusion model for conditional generation.

For 78 RPM noise removal, this is not possible. Noise in digitized gramophone recordings depends on the original recording conditions, on the medium condition,

[1] It should be noted that the objective metrics used for bandwidth extension evaluation in [17] (log-spectral distance, Fréchet audio distance, and F1 score of a pre-trained transcription model) do not take psychoacoustical models into account. The advantages of using a psychacoustically inspired objective quality metric are discussed later in this chapter.

and also on the equipment used in analog-to-digital conversion. Each 78 RPM noise sample is therefore unique, and restoration using the exact same procedure as [17] would require an exemplary instance of the noise corrupting the specific musical recording being restored.

This chapter shows how the method of [17] was adapted for 78 RPM noise removal. It begins by describing the methodology that was used, and displays preliminary results that were useful for identifying inference parameters relevant for this denoising task. Following this, it shows results that were obtained applying the method to signals not seen during training of the diffusion models, evaluated through objective and subjective audio quality measures.

4.1 Proposed Method

Although each 78 RPM noise sample is unique, to some extent they sound more or less similar. This suggests that they must share statistical characteristics in some circumstances. Considering this, one idea to adapt conditional diffusion sampling for 78 RPM noise removal is to use a random $\mathcal{A}(\cdot)$ in each diffusion step.

Having a representative set of 78 RPM noise samples available, it is possible to randomly select one of them in each diffusion step, and add the selected sample to the current $\mathbf{x}(t)$. Even though the noise added at each step might be different from the noise in the musical recording being restored, if the noise samples that are used are statistically similar to the noise in the recording, then over many steps $\mathcal{A}(\cdot)$ should approach the degradation that needs to be removed.

One important detail is that $\mathcal{A}(\cdot)$ should preserve the signal-to-noise ratio (SNR) of the signal being restored. Since $\mathcal{A}(\cdot)$ should approximate the 78 RPM noise degradation in the signal being restored, it should also preserve the power ratios between the signal and the noise being added artificially.

To ensure this, one possible approach is to normalize all 78 RPM noise samples in the dataset, and calculate the gain necessary to add the noise while preserving the SNR of the musical piece being restored. The noise gain then becomes an additional parameter in diffusion inference.

In a practical setting, the SNR of the signal to be restored is most likely unknown. Therefore, using this approach for restoration requires an accurate estimate of the SNR or of the noise gain that should be used. A suitable heuristic for overcoming this obstacle will be presented in Section 4.2.

This adaptation of the diffusion setting of [17] was tested using a curated version of the noise dataset of [31]. In [31], the authors built a dataset of 78 RPM noise samples by extracting noise-only excerpts of gramophone recordings uploaded to the Great 78 Project [34], a non-profit initiative for preserving and digitizing 78

RPM records. In their work, the noise samples were used to artificially corrupt a dataset of clean, high-quality polyphonic musical pieces, and train a neural network specialized in 78 RPM noise removal.

Their noise dataset is publicly available on the author's page, and consists of 139 minutes of 78 RPM noise divided in 2430 segments from 1386 recordings. The shortest segment in the dataset has around 2s. The dataset was divided[2] in train, validation and test splits. In the preliminary experiments of Section 4.2, all splits were used for $\mathcal{A}(\cdot)$, but for the objective and subjective tests of Sections 4.4 and 4.5, only the test split was used.

The noise dataset of [31] was created adapting the voice activity detector of [35] to automatically extract the noise-only parts of the Great 78 Project recordings. As a consequence, there are a few noise samples in the dataset which include *fade-in* or *fade-out* effects, and that could not be directly used in $\mathcal{A}(\cdot)$.

Because of this, a subset of the dataset was created selecting only the noise samples with original length larger than 2.6s, and removing the first and last 0.5s of them. In other words, only samples containing at least 1.6s of gramophone noise in its middle were kept.

Another important practical note of the restoration system in this work is the division of the signal to be restored in small blocks. Diffusion models work from their first iteration with vectors with the desired output size. Because high-fidelity music recordings are usually sampled at 44.1kHz, this implies loading vectors of very high dimension into memory for restoration. To avoid this problem, the musical pieces being restored were divided into several small blocks, which were then combined to produce the restored signal.

In this work, overlapping blocks of length 1s were used, with an overlap length of 0.25s. No analysis window was used, but for reconstruction with OLA, synthesis windows made of a flat-top with Hann-like transition regions were used. Figure 4.1 illustrates this synthesis window, which is essentially a Hann window split in two with ones in the middle. The transition regions cover the overlapping parts of the blocks, and the windows satisfy the perfect reconstruction conditions.

Since the blocks are shorter than the shortest noise sample in the curated dataset, the selected noise sample can be truncated to block length, and $\mathcal{A}(\cdot)$ could be simply defined as

$$\mathcal{A}(\mathbf{x}) = \mathbf{x} + G\mathbf{n}, \tag{4.1}$$

where $\mathbf{x}$ is the current $\mathbf{x}(t)$, $\mathbf{n}$ is the selected noise sample and G is a noise power

[2]The original dataset of [31] was already divided into three splits, but some files were misnamed in the dataset's metadata. This made the contents of the test split unclear, and the whole dataset was shuffled and split again for this work. The splits and corresponding metadata can be found in this work's Github repository.

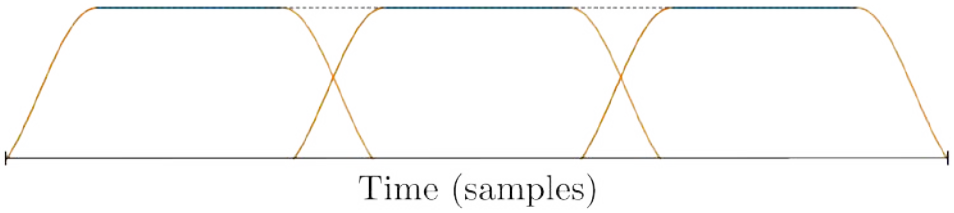

Figure 4.1: Illustration of three flat-top synthesis windows that were used during restoration. Transition regions are painted in orange. It is possible to see that the windows are complementary. The signal was padded to avoid transition effects in the denoised output.

adjustment factor. However, a final detail to be taken into account is that the result of applying $\mathcal{A}(\cdot)$ might not be directly convertible into *.wav* format.

Wave files are limited to values in $[-1, 1]$, which represent the maximum and minimum sound pressure levels that could be registered by the recording equipment that was used. Depending on the values of $\mathbf{x}$, the resulting $\mathcal{A}(\mathbf{x})$ might exceed the *.wav* file range.

To solve this, a simple heuristic was adopted[3]. Instead of defining $\mathcal{A}(\cdot)$ as in Equation (4.1), $\mathcal{A}(\cdot)$ was defined as

$$\mathcal{A}(\mathbf{x}) = \frac{\mathbf{x} + G\mathbf{n}}{2} \tag{4.2}$$

to ensure that the *.wav* range will not be exceeded for SNRs down to around 0dB, as explained below.

In a setting with 0dB SNR — which is far below most restoration scenarios — the signal and noise powers will be the same. If $\mathbf{x}$ contains a maximum (or minimum) value sample, then dividing it by two leaves half of the *.wav* dynamic range for the noise. Since signal and gain times noise should have equal power, it is reasonable to assume that $G\mathbf{n}$ will at most reach the edges of the *.wav* file range. This implies that the sum divided by two will not exceed the desired range.

With this heuristic, $\mathcal{A}(\cdot)$ can simulate 78 RPM noise degradations with SNRs down to approximately 0dB without creating invalid vectors. This is enough considering that, for most recordings from the electric era, the SNR of the signal being restored is well above 10dB [88].

The pseudocode in Algorithm 4.1 summarizes the restoration procedure used in

[3]An alternative to the proposed heuristic would be to allow $\mathbf{x}$ and $\mathcal{A}(\mathbf{x})$ to exceed the *.wav* range, and to apply peak normalization to the restored output. This approach would have the advantage of simplifying the restoration procedure, and would avoid the volume adjustments that are mentioned in Sections 4.2, 4.4, and 4.5. However, this would change the original magnitude range of the digitized 78 RPM recordings, and so it was decided to use the heuristic for $\mathcal{A}(\cdot)$ described in the current section.

this work. The degraded signal is first divided into overlapping blocks of one second. Then, conditional diffusion sampling is used to restore each block, and OLA with flat-top windows is used to reconstruct the output

Algorithm 4.1: Additive Noise Removal

```
noisyBlocks = divideInBlocks(noisySignal);
cleanBlocks = [ ];
for noisyBlock in noisyBlocks do
    block ~ N(0, σ_max I);
    for i = N : 1 do
        n = getNoiseSample();
        function A(v):
            return (v + Gn)/2;
        function score(block, noisyBlock, stepIndex):
            unconditional = s^θ(block, stepIndex);
            conditional = conditioning(block, noisyBlock, stepIndex, A);
            return unconditional + conditional;
        block = reverseStep(block, noisyBlock, score, stepIndex, schedule)
    end
    append block to cleanBlocks
end
return OLA(cleanBlocks);
```

To properly simulate gramophone degradations during reverse diffusion, a new gramophone noise sample $\mathbf{n}$ needs to be obtained at every reverse step. Once $\mathbf{n}$ is sampled the degradation $\mathcal{A}(\cdot)$ can be redefined considering the iteration's noise sample and the pre-defined noise gain G. With updated $\mathcal{A}(\cdot)$, the score function can be redefined as the sum of the unconditional score $s^{\theta}(\cdot,\cdot)$ with a DC or RG estimate of the conditioning term $\nabla_{\mathbf{x}(t)} \log f(\mathbf{y}|\mathbf{x}(t))$. This redefined version of the score can then be used as a drop-in replacement for $s^{\theta}(\cdot,\cdot)$ in the reverse diffusion steps of any of the SDE sampling algorithms.

In Algorithm 4.1, the inner for loop represents the iterations for conditional diffusion sampling of a single block. The first line in this loop calls a function that fetches a new gramophone noise sample, and the two function definitions inside the for loop correspond to the redefinitions of $\mathcal{A}(\cdot)$ according to the new noise sample $\mathbf{n}$, and of the conditional score according to the new $\mathcal{A}(\cdot)$. The last function call in the inner for loop of the algorithm does a reverse diffusion step, as described in the inner for loops of Algorithms 3.6, 3.7, and 3.8

4.2 Preliminary Experiments

Before applying the method of the previous section in a real setting, some preliminary experiments were made to identify which diffusion inference parameters were

most relevant for the denoising task, and to verify the method's robustness to different musical pieces, levels of degradation, and noise gain estimation errors. Ideally, the parameters used in [17] for different tasks could be reused, or slightly modified, for most denoising settings.

The preliminary experiments were conduced by artificially degrading clean, high-quality recordings of Frédéric Chopin's preludes [89]. More specifically Cyprien Katsaris' 1993 versions of preludes 7, 16, and 20 were used, with a special emphasis on prelude 20 because of its variations in playing dynamics.

Prelude 20 has both loud and quiet parts, in which 78 RPM noise is noticed differently by the listener. As such, it was a particularly interesting test sample. Prelude 16 is mostly loud, and is a fast paced piece, while prelude 7 is slower and quieter.

To pollute C. Katsaris' recordings, one noise sample from the train split of the gramophone noise dataset was selected and removed from the set of possible noise samples for $\mathcal{A}(\cdot)$ (inference set). Since the recording was much larger than the noise sample, the latter had to be extended through repetition, taking into account the auditory impression of periodicity caused by rotation.

This was done by calculating the amount of full single 78 RPM rotation periods inside the noise sample, truncating it to this length, and then repeating the truncated noise sample with a cross-fade of 0.1 second between repetitions.

After this, noise was directly added to the preludes using the division by two trick to avoid saturation. Note however, that the division by two is also reverted by conditional diffusion, as it is a part of $\mathcal{A}(\cdot)$.

Regardless of the mix of train, validation and test samples in the inference set, because the noise sample used to pollute the recordings was removed from the inference set, the preliminary experiments emulated a real denoising setting, since the model did not have access to the polluting noise during inference.

4.2.1 Inference Parameter Variations

In [17] a similar set of inference parameters was used for reverting low-pass filtering, inpainting, and clipping. In this work, the same set of parameters was used with minimal modifications.

The authors of [17] used the heuristic stochastic sampling of Algorithm 3.8, with scaling factor and noise schedule as defined by Equations (3.62) and (3.63). For sampling in [17], the values of σ_{min} and σ_{max} are set to 10^{-4} and 1, and $\rho = 13$[4]. For sampling in this work, ρ was kept at 13 to ensure more points near the lowest noise

[4]Note that this is not a problem because $t \sim \mathcal{U}([0, 1])$ during training, which implies that the noise levels used during inference are covered during training, despite the difference between training σ_{min} and σ_{max} and inference σ_{min} and σ_{max}.

limit, but $\sigma_{\min}$ and $\sigma_{\max}$ were set to their training values. As the first results were satisfactory for these values of $\sigma_{\min}$ and $\sigma_{\max}$ (see Figure 4.5), no further exploration was carried out on them.

Here, as in [17], $S_{\max}$ and $S_{\min}$ are not used, and therefore set to values respectively above and below $\sigma_{\max}$ and $\sigma_{\min}$. Similarly, $S_{\mathbf{z}}$ is not used and set to one. However, S_{total} is used and set to 5.

A different number of steps is used for declipping and the other two inverse problems in [17]. For low-pass filtering and inpainting, 35 diffusion steps are used, whereas 140 are used for clipping. In this work, 140 steps were used considering the stochasticity of $\mathcal{A}(\cdot)$, and the fact that the outputs needed a higher sampling rate (44.1kHz here *versus* 22.05kHz in [17]). It should be noted, however, that choosing an elevated number of diffusion steps had a significant impact on the inference time due to the sequential nature of reverse diffusion. This is mentioned again in Section 4.4. Table 4.1 summarizes the diffusion inference parameters that were used in this work.

Table 4.1: Diffusion inference parameters used for sampling in this work. This table does not include conditional sampling specific parameters such as G and reconstruction guidance ξ'. Note that $S_{\max}$ and $S_{\min}$ are outside the range defined by $\sigma_{\min}$ and $\sigma_{\max}$, and are therefore not used, as per Algorithm 3.8.

$\sigma_{\min}$	10^{-6}	$\sigma_{\max}$	10
ρ	13	N_{steps}	140
$S_{\max}$	50	$S_{\min}$	0
$S_{\mathbf{z}}$	1	S_{total}	5

Both data consistency and reconstruction guidance conditional sampling methods were used for the inverse problems in [17]. The only exception was hard clipping, for which the RG method was used uniquely. For the RG method, $\xi' = 0.35$.

As mentioned in Section 3.3.2, the DC method is not appropriate for random $\mathcal{A}(\cdot)$. In the denoising case, $\bar{\mathbf{x}}^{\theta}$ is not consistent with $\mathbf{x}(0)$ because $\mathbf{x}^{\theta} - \mathcal{A}(\mathbf{x}^{\theta})$ does not necessarily correct the corrupted samples of $\mathbf{y}$.

To verify this, one can observe the denoising results of the DC and RG methods in an aritficially degraded version of prelude 20, as shown in the spectrograms[5] of Figures 4.4 and 4.5. C. Katsaris' recording was polluted with 30dB SNR 78 RPM noise, and the inference parameters described in the previous paragraphs were used for restoration. Figures 4.2 and 4.3 show the original and noisy signals, respectively. The same noise gain that was used to artificially degrade the clean recorded was used as G, attempting to isolate the gain's effect in noise removal.

[5]All spectrograms in this chapter were made with 4096 DFT points and Hann windows with 2048 samples and 50% overlap.

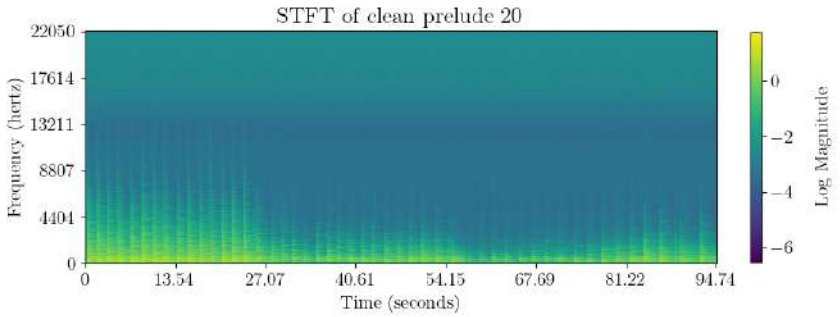

Figure 4.2: Spectrogram of C. Katsaris' recording of prelude 20.

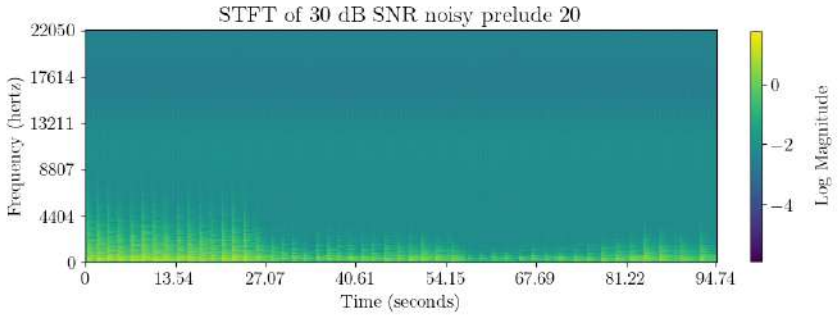

Figure 4.3: Spectrogram of an artificially corrupted version of prelude 20, using a global SNR of 30dB. The signal was multiplied by two (verifying that there was no clipping) to keep the original dynamic range.

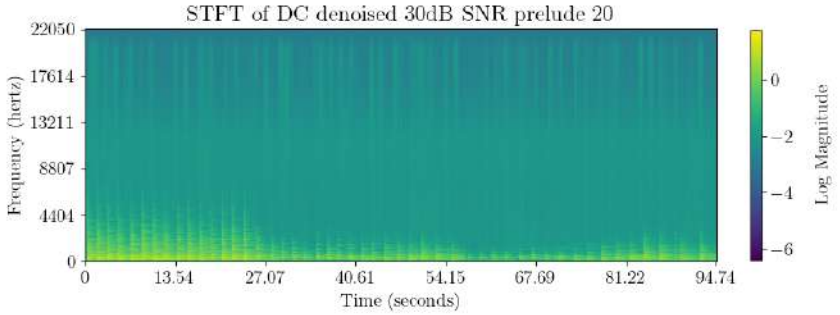

Figure 4.4: Spectrogram of the DC reconstructed version of prelude 20. The output signal was obtained using the sampling parameters described above, while applying the DC method exclusively. The residual noise is visible throughout the spectrum.

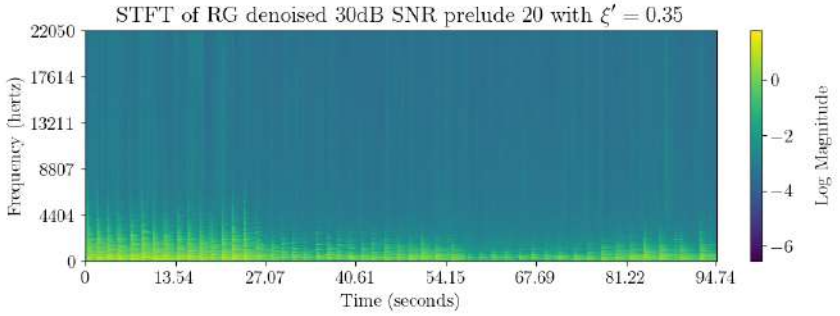

Figure 4.5: Spectrogram of a RG reconstructed version of prelude 20, using the same parameters of Figure 4.4 and $\xi' = 0.35$. Less noise remains with this method.

The DC method leaves considerable residual noise when compared to the RG method. This suggests that indeed the DC method is not suited for the proposed $\mathcal{A}(\cdot)$. For this reason, only the RG method was used in the following experiments.

The parameter that had the most impact in the restored output was ξ'. As seen in Section 3.3.2, ξ' is the user-defined RG parameter that controls the norm of the conditioning term of the score function. Excessively low values of ξ' (i.e., samples with very weak conditioning) tended to remove high frequency content from the signal, and were not always successful in completely removing 78 RPM noise. Excessively high values of ξ' (i.e., samples with very strong conditioning) preserved the musical content in the original signal, but failed to remove 78 RPM noise.

Figures 4.6 to 4.8 show the denoising results for the noisy example of Figure 4.3 using $\xi' = 0.15$, $\xi' = 2.45$, and $\xi' = 4.55$. The results for $\xi' = 0.35$ can also be seen on Figure 4.5. Once again, the noise gain that was used to corrupt the original recording was used as G, to try to isolate the gain estimation effect in restoration.

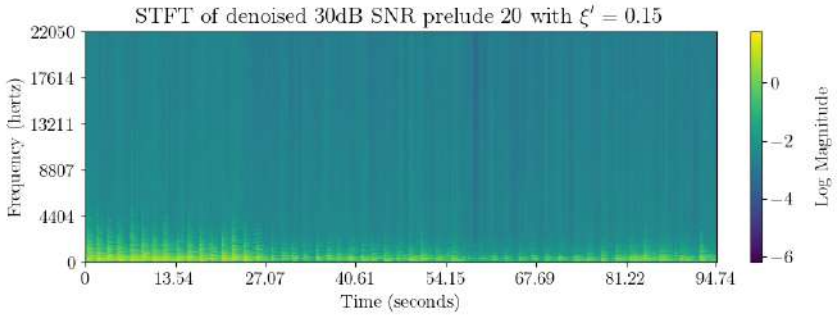

Figure 4.6: Spectrogram of a RG reconstructed version of prelude 20 using $\xi' = 0.15$. The remaining parameters were the same as described in the beginning of this section. It is possible to see residual noise, and reduction of high frequency peaks.

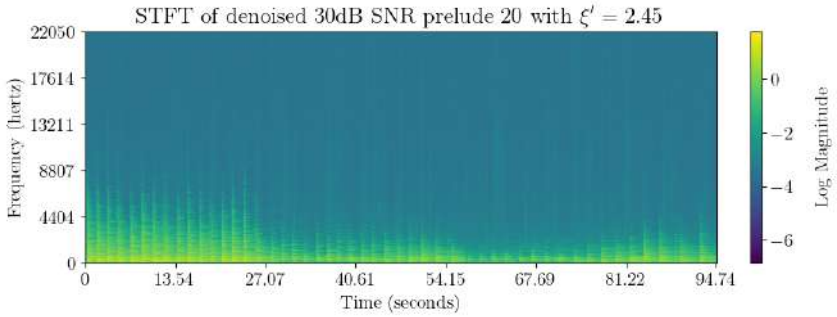

Figure 4.7: Spectrogram of a RG reconstructed version of prelude 20 using $\xi' = 2.45$. The remaining parameters were those described in the beginning of this section.

Listening to the results and comparing Figures 4.6 and 4.5 to Figure 4.7 and to Figure 4.2, it is possible to note that the lower values of ξ' reduce the high-frequency peaks in the beginning of prelude 20, which produces a muffled, low-passed sound.

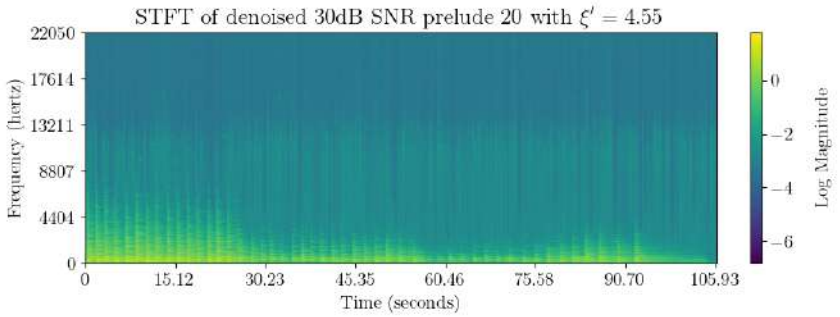

Figure 4.8: Spectrogram of a RG reconstructed version of prelude 20 using $\xi' = 4.55$. The remaining parameters were the same as described in the beginning of this section. It is possible to see that there is residual noise.

Additionally, $\xi' = 0.15$ leaves considerable residual noise in the restored signal, as can be seen by the vertical traces throughout Figure 4.6.

In contrast, it is possible to note that while high-frequency content is preserved for $\xi' = 4.55$ in Figure 4.8, it is accompanied by residual noise. The noise appears to follow note attacks, which results in a particularly bothersome listening experience, and makes further treatment of the signal very hard.

Considering the empirical results above, and the fact that other parameter changes did not have significant impact on the restored output, a preliminary experiment was conduced to find a close to optimal value of ξ' for prelude 20 corrupted with 30dB signal-to-noise ratio. To objectively evaluate and compare different restored outputs, the perceptual evaluation of audio quality (PEAQ) [41, 42] algorithm was used.

The PEAQ algorithm is the International Telecommunication Union's (ITU) recommended standard for the objective evaluation of general audio quality [41]. PEAQ was originally developed in the context of audio coding, and it combines different psycho-acoustical measures to quantify the perceived quality difference between a test signal and a reference signal. Its output is an objective difference grade (ODG) between -4 (worst, very annoying impairment) and 0 (best, imperceptible impairment).

A detailed study of the ITU recommendation describing PEAQ can be found in [42], but it suffices to know that it is a widely adopted [17, 31] general audio quality evaluation metric, despite the fact that it was developed specifically for lossy audio coding. Because PEAQ was developed to detect subtle artifacts introduced by lossy audio codecs, even small differences between the test and reference signals can cause it to give low grades. As PEAQ was not specifically designed to handle additive noise corruption, it may attribute ODGs close to -4 even to signals degraded with 40dB SNR 78 RPM noise (which is barely perceptible).

Despite its sensitivity, PEAQ was a valuable tool while developing the methods

proposed in this work because it allowed for a preliminary evaluation of restoration performance without resorting to time-consuming subjective tests. Furthermore, it is particularly good at capturing high frequency content loss during restoration, which tended to happen for low values of ξ'.

Because PEAQ evaluates a test signal with respect to a reference, and since this work wanted to evaluate the quality improvement of a restoration technique, the measure that was used was actually the ODG variation

$$\Delta\text{ODG} = \text{ODG}_{\text{restored,original}} - \text{ODG}_{\text{noisy,original}}. \tag{4.3}$$

This allowed to measure how much noise annoyance was reduced through this work's denoising technique.

Another important detail that was taken into account is the difference in PEAQ ODG's depending on the loudness of the piece being evaluated. With louder volume, nuisances and artefacts are more perceptible, and PEAQ tends to be more strict[6].

Considering this, and having in mind that the restored output had the same magnitude range as the original signal, the noisy version of prelude 20 was multiplied by two (verifying that no clipping had happened) before being fed into PEAQ. This ensured that both restored and noisy versions would be compared to the reference at the same volume level, and ensured that comparisons in a lower volume level would not make PEAQ grades artificially higher. In all other tests involving PEAQ in this work, a similar procedure was adopted.

Table 4.2: Objective difference grades for restored versions of prelude 20 using different values of ξ'. The noise gain that was used to degrade the original recording was used as G.

		ODG	ΔODG
Noisy		−3.9088	-
ξ'	0.15	−3.5624	0.3465
	0.35	−2.3416	1.5672
	0.875	−1.0169	2.8919
	1.4	−0.8018	3.1071
	1.925	−0.6929	3.2159
	2.45	**−0.6581**	**3.2507**
	2.975	−0.7357	3.1731
	3.5	−1.4574	2.4515
	4.025	−2.7330	1.1758
	4.55	−3.4102	0.4987

Table 4.2 shows the ODG and ΔODG for different values of ξ', as well as the

[6]This was verified through informal experiments, and is likely a consequence of the decay in the psychoacoustical masking models employed in PEAQ. A thorough explanation can be found in [42].

starting ODG between the noisy and original versions of prelude 20. The values of ξ' were chosen with 0.525 increases starting at 0.35 to linearly explore possible values of ξ', and $\xi' = 0.15$ was also included to see the impact of a value close to zero. The table shows that indeed values too big or too small for ξ' cause issues in restoration. For prelude 20, with a global SNR of 30dB, the optimal value of ξ' seems to be around 2.45 (marked in bold).

4.2.2 Signal-to-Noise Ratio Variations

The first preliminary experiments suggest that it is possible to find an optimal value for ξ' when using the RG method for restoration. However, further testing was necessary to see if the method in general was robust to different noise levels in the degraded recording, and to see if the relation between quality[7] and ξ' changed with increasing noise power.

Figures 4.9, 4.10 and 4.11 show the denoised versions, using $\xi' = 2.45$, of C. Katsaris' recording of prelude 20 degraded with SNRs of 10, 20, and 40dB. In all three cases, the same noise sample used in the examples of the previous subsection was used to degrade the original version of prelude 20. The noise power was adjusted in each case to match the required SNR.

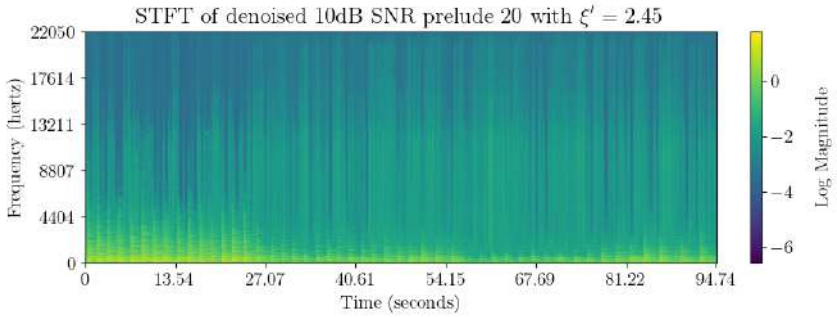

Figure 4.9: Spectrogram of the denoised version of 10dB SNR prelude 20 using $\xi' = 2.45$. The remaining parameters are the same as described in the previous subsection.

In each of the examples, the restoration noise gains G were adjusted to match the noise power, in an attempt to isolate the effect of the power adjustment factor in the outputs. It is possible to see that using $\xi' = 2.45$ left some residual noise for the two lowest SNRs. This suggests that the optimal ξ' could be a function of the SNR, or of the absolute noise level present in the recording.

To evaluate this possibility, another sweep over ξ' values was performed using PEAQ to evaluate restoration over recordings with different SNRs. The setup was similar to the one used previously: the variation in ODG was measured with noisy

[7] As measured by PEAQ.

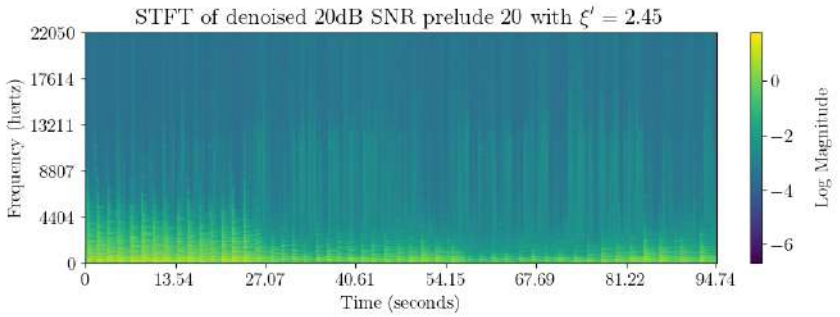

Figure 4.10: Spectrogram of the denoised version of 20dB SNR prelude 20 using $\xi' = 2.45$. The remaining parameters are the same as described in the previous subsection.

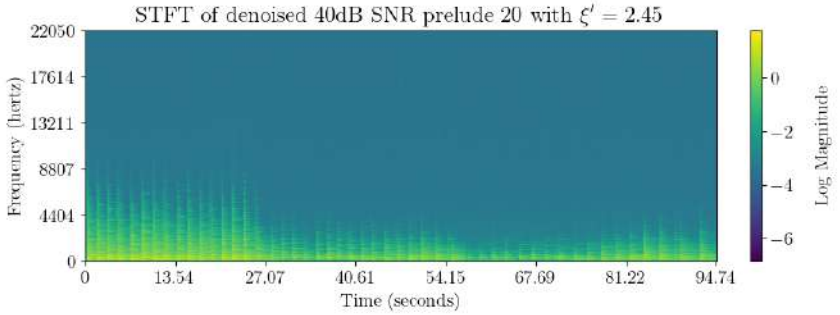

Figure 4.11: Spectrogram of the denoised version of 40dB SNR prelude 20 using $\xi' = 2.45$. The remaining parameters are the same as described in the previous subsection.

and restored versions of prelude 20 as test signals. Several restored versions were created using different values of ξ'. This procedure was repeated for SNRs of 10, 20, and 40dB. For each SNR, the noise adjustment factors G were set to match the noise powers, in order to try to isolate the effect of G on restoration. For convenience, the results for 30dB SNR are also included in the tables below.

Table 4.3 shows the ODGs between the noisy and original versions of the preludes. It is possible to see that PEAQ attributes low grades even for the 40dB SNR noisy version, which showcases the method's sensitivity to additive noise.

Table 4.3: ODG for noisy versions of prelude 20 degraded with different SNR levels.

	SNR			
	10	20	30	40
ODG	−3.9132	−3.9130	−3.9088	−3.7507

Table 4.4 displays the ΔODG using different values of ξ' over the four SNRs. The best values of ξ' for the different SNRs are marked in bold with their corresponding ΔODG. For all SNRs, there is a clear downward trend in restoration performance for increasing ξ' after a certain value. Because of this, it seemed safe to ignore some combinations of ξ' and SNR, as there would be no gains in restoration quality. The

untested combinations of ξ' and SNR are marked with dashes in Table 4.4.

Table 4.4: ΔODG for the versions of prelude 20 degraded with different SNR levels and restored using various values of ξ'.

ΔODG		SNR			
		10	20	30	40
ξ'	0.15	0.4182	0.6126	0.3465	0.0216
	0.35	**0.8275**	0.6788	1.5672	1.7022
	0.875	0.2750	1.8450	2.8919	3.0698
	1.4	0.0696	**2.0728**	3.1071	3.3086
	1.925	0.0116	1.5069	3.2159	3.3930
	2.45	0.0016	0.4615	**3.2507**	3.4427
	2.975	-	0.0797	3.1731	3.4973
	3.5	-	0.0045	2.4515	**3.5104**
	4.025	-	-	1.1758	3.5066
	4.55	-	-	0.4987	3.4891
	5.075	-	-	-	3.4485
	5.6	-	-	-	3.4120
	6.125	-	-	-	3.3485

Observing Table 4.4, there appears to be a relation between the optimal value of ξ' and the SNR of the noisy recording. Noisier recordings appear to require lower values of ξ', whereas higher SNR recordings have better results with higher values of ξ'. Furthermore, the relation between SNR and ξ' appears to be linear, with ξ' increasing 1.05 for every 10dB in SNR.

However, a few points should be noted. First, since this test included a single test signal, it could be possible for ξ' to be in fact related to the noise power in the recording, and not to the signal-to-noise power ratio. Secondly, the decrease in ΔODG after the optimal ξ' happens faster for lower SNRs, i.e. the penalty for overestimating ξ' is much worse for lower SNRs. Finally, the proposed method appears to be much less effective for lower SNRs, regardless of the choice of ξ'.

4.2.3 Test Sample Variation

To see if the results obtained previously were dependent on the test sample, another sweep over ξ' values was made for 30dB noisy versions of preludes 16 and 7. Ideally, if the method is robust, the optimal values of ξ' should be somewhat close to the one obtained for prelude 20.

Preludes 7 and 16 were chosen because they have very different dynamics. Prelude 16 is a *forte*, fast-paced piece, whereas prelude 7 is slower and quieter. These two characteristics are present in parts of prelude 20, but evaluating restoration performance with preludes 7 and 16 allows to see if specific playing dynamics can impact restoration performance.

The procedure for the experiments with preludes 7 and 16 was the same of prelude 20. Both pieces were degraded with the same noise sample used for previous experiments, and the noise power was adjusted to make the SNR equal to 30dB in both cases. In all restored versions, the noise adjustment factor G was set to match the noise power, attempting to isolate the effect of G in the restored output.

Table 4.5 shows the PEAQ results for the 30dB SNR degraded versions of preludes 7 and 16 using different values of ξ'. It is possible to see that the optimal values for ξ' (in bold) remain reasonably close to the one obtained for prelude 20 in both cases. This suggests that the optimal value of ξ' could be reasonably independent of the signal being restored. Signal independence is important because (if true) it implies that diffusion-based restoration performance is only dependent on the amount of noise present in the signal being restored.

Table 4.5: Objective difference grades for restored versions of preludes 7 and 16 using different values of ξ'. Both preludes were degraded with a 30dB SNR, using the same noise sample of previous experiments. The noise gain that was used to degrade each of the original recordings was chosen as G for $\mathcal{A}(\cdot)$.

		Prelude 7		Prelude 16	
		ODG	ΔODG	ODG	ΔODG
Noisy		−3.8673	-	−3.8954	-
ξ'	0.15	−3.6447	0.2227	−2.2922	1.6032
	0.35	−2.0856	1.7817	−2.3757	1.5197
	0.875	−1.5365	2.3308	−2.1377	1.7577
	1.4	−1.3930	2.4743	−1.7321	2.1633
	1.925	−1.2671	2.6002	**−1.6247**	**2.2707**
	2.45	−1.2583	2.6091	−1.6365	2.2589
	2.975	**−1.2131**	**2.6543**	−2.1521	1.7433
	3.5	−1.2360	2.6313	−3.4982	0.3972
	4.025	−1.3954	2.4719	−3.8227	0.0727
	4.55	−1.5879	2.2794	−3.8594	0.0360

A more complete evaluation of the method's robustness to the signal being restored will be provided in the section describing the validation of the restoration method, and in the objective and subjective tests. In all three of them, the method was evaluated using PEAQ with a larger quantity of test signals, with a wider variety of playing styles.

4.2.4 Restoration Heuristics

The preliminary experiments seen until now mostly dealt with finding the optimal value of ξ' for a known global SNR and investigating if restoration using RG is somewhat robust to different musical signals and noise levels. In all cases, the noise

power adjustment factor G was known, which is not the case in a real restoration setting.

Considering this, two experiments were designed to assess the method's robustness to variations in G. The first of them was simply a study of the method's robustness to an estimated noise adjustment factor: for a noisy piano piece with a known "correct" value of G (the value used to create the noisy piece with specified SNR), different restored versions were created using a fixed ξ' while multiplying G for different relative error factors.

The second experiment tried to see if it was possible to denoise a solo piano recording varying G throughout the diffusion steps. Since choosing a value for G is equivalent to estimating a global SNR for the signal, in this experiment a few ranges of possible SNRs were chosen. With the range defined, in each diffusion step a SNR value S was uniformly sampled in the range. The gain G was set so that $\mathcal{A}(\cdot)$ would result in a signal with SNR equal to S.

In the first experiment, the 30dB SNR degraded version of prelude 20 was restored using the optimal value of $\xi' = 2.45$ (see Table 4.2). Different restored versions were created using

$$G \in \{0.1G', 0.2G', 0.316G', 0.5G', 0.9G', 1.1G', 2G', 3.16G', 5G', 10G'\}, \tag{4.4}$$

where G' was the gain required to make the noisy signal have 30dB SNR. The rationale for choosing this set of values for G was evaluating the performance with relative errors of ± 10, ± 7, ± 5, ± 3, and ± 0.5dB with respect to G'.

The implicit assumption is that the best restoration happens when $\mathcal{A}(\cdot)$ is adjusted to add noise with the same power as the noise in the signal that is being restored. In this controlled setting, G' is known to be the value that correctly adjusts $\mathcal{A}(\cdot)$. Considering that G' would not be known in a real restoration scenario, this test verifies the method's robustness to errors in the estimation of the best possible gain factor.

Table 4.6 shows the ODGs of the restored versions with these values of G, and the ΔODG remembering that in this case $\text{ODG}_{\text{noisy,original}} = -3.9088$ (see Table 4.2). For convenience, the results for $G = G'$ are also included in Table 4.6

This single experiment seems to indicate that the ΔODGs remain within 90% of the value for $G = G'$ for up to a relative error factor of two in both directions. That is roughly equivalent to an error tolerance of ± 3dB, which can be considered fairly narrow given that estimating G is equivalent to estimating the noise power. If 75% of the performance for $G = G'$ is considered the acceptable performance loss, then the error tolerance increases to ± 5dB.

In the second experiment, the 30dB SNR degraded version of prelude 20 was also

Table 4.6: Restoration performance for the 30dB SNR prelude 20 considering non-ideal values of G. It is possible to see that performance drops over 25% for relative errors above ±5dB in the estimation of G (below $0.316G'$ and above $3.16G'$).

Gain	ODG	ΔODG
$0.1G'$	-2.7183	1.1905
$0.2G'$	-2.2264	1.6825
$0.316G'$	-1.2768	2.6320
$0.5G'$	-0.7907	3.1181
$0.9G'$	-0.6540	3.2548
G'	-0.6581	3.2507
$1.1G'$	-0.6710	3.2379
$2G'$	-0.8610	3.0478
$3.16G'$	-1.0828	2.8261
$5G'$	-1.4670	2.4419
$10G'$	-2.5874	1.3214

restored using $\xi' = 2.45$. However, as mentioned earlier, different ranges of possible SNRs were chosen, and a different gain for $\mathcal{A}(\cdot)$ was calculated in each diffusion step based on an SNR sampled uniformly from the chosen range. For a sampled SNR S, the gain G of $\mathcal{A}(\cdot)$ was computed noting that:

$$S = 10 \log \frac{P_{\text{signal}}}{G^2 P_{\text{noise}}} \implies G = \sqrt{\frac{1}{10^{\frac{S}{10}}} \frac{P_{\text{signal}}}{P_{\text{noise}}}}, \tag{4.5}$$

where P_{signal} and P_{noise} are respectively the signal and noise powers.

This experiment tried to see if it was possible to clean a noisy signal using an estimate of a range of possible global SNRs instead of a gain estimate. The advantage of this approach is that it can be more intuitively tuned by a human user through listening and guessing, removing the need to evaluate the quality of the gain estimate.

For example, one can informally pair SNRs and noise descriptions (e.g. slight noise: 40dB, regular noise: 30dB, intense noise: 20dB, very intense noise: 10dB), and define a number of uncertainty levels about the description (e.g. unsure: ±10dB, fairly unsure: ±5dB, fairly certain: ±3dB, certain: ±0dB). Then the user would be able to intuitively select the restoration parameters according to those predefined categories.

One drawback of this approach is the fact that the SNR varies throughout the noisy piece. In a real scenario, the noise would likely have approximately the same power throughout the whole recording, but the signal can have parts with lower intensity. Since this approach relies on guessing a range for the global SNR, it could be possible to use inadequate gain values on certain blocks of the signal being

restored, as the local and global SNRs might differ.

Four SNR ranges were tested: $[0, 40]$dB, $[20, 40]$dB, $[27, 33]$dB, and $[30, 30]$dB. The range of $[0, 40]$dB covers both extremely noisy and subtly degraded excerpts, and can be seen as equivalent to assuming nothing about the piece being restored. The two middle ranges (both centered at 30dB for the sake of this experiment) are equivalent to assuming different levels of certainty regarding the global SNR of the piece being restored. Finally, the (single value) range of $[30, 30]$dB corresponds to the case where a precisely known global SNR is used in all blocks, regardless of possible differences with respect to the local SNRs.

Table 4.7 shows the results for the four SNR ranges on the 30dB SNR noisy prelude 20. The ODG variations decrease as the SNR range narrows down to 30dB. This is likely caused by the difference between the global SNR of prelude 20, and the local SNRs of each block.

Table 4.7: Restoration performance for the 30dB SNR prelude 20 using four SNR ranges.

Range (dB)	ODG	ΔODG
$[0, 40]$	-1.2200	2.6888
$[20, 40]$	-1.4597	2.4492
$[27, 33]$	-1.5828	2.3261
$[30, 30]$	-1.5658	2.3431

Prelude 20 starts in *fortissimo* and moves to a quieter register for its second half. Because of this, the local SNR for the quieter blocks is likely much smaller than 30dB, which explains why the performance decreased as the SNR range narrows down. This suggests that the use of SNR ranges is not appropriate unless the local SNR of each block is considered.

Considering the results of the two previous experiments, two heuristics were designed to try to adjust the gain factor G. The first one tried to directly estimate G by estimating the noise power, while the second explored the idea of using a different SNR range for each block.

For a normalized (unit power) noise sample, the value of G that makes the sample used in $\mathcal{A}(\cdot)$ have the same power as the noise in the recording is simply the square root of the noise power. To estimate the noise power, a sliding window with stride of one sample can be used to calculate the local power $P_{\text{noisy}}[n]$ of small chunks of the noisy signal. In the first heuristic, the noise power is estimated as $\min P_{\text{noisy}}$, and $\tilde{G} = \sqrt{\min P_{\text{noisy}}}$

The intuition behind this idea is that in moments where there is no musical content in the noisy signal, only noise should be present[8]. Therefore, the minimum

[8]Assuming a studio recording and not a live one.

local power should approximate the noise power. This estimate should be accurate as long as the window size is properly adjusted, and as long as there are noise-only parts in the signal being restored.

While these requirements are certainly a drawback, as one cannot guarantee the existence of noise-only parts in all digitized 78 RPM recordings, in most real scenarios it is possible to estimate G using noise-only excerpts in the beginning or end of the tracks being restored.

The second heuristic also relied on estimating the noise power as $\min P_{\text{noisy}}$, but used it to estimate local SNR ranges. Inside each one second block being restored using diffusion, a sliding window with stride one was used to calculate the local power $P_{\text{block}}[n]$ of small chunks of the block. Then, the local SNR was approximated as

$$\tilde{S} = 10 \log \left(\frac{\max P_{\text{block}} - \min P_{\text{noisy}}}{\min P_{\text{noisy}}} \right). \tag{4.6}$$

The idea is that, even for low SNRs such as 10dB, the noise component of the degraded signal will be much smaller than the original signal. It follows that the peak local power of each block should approximate the original signal power, as the noise content inside the peak chunk should be negligible. Therefore, $\max P_{\text{block}} - \min P_{\text{noisy}} \approx P_{\text{signal}}$. As before, $\min P_{\text{noisy}}$ approximates P_{noise}.

This approximation works better when $\max P_{\text{noisy}} \gg \min P_{\text{noisy}}$. If this is the case, then $\min P_{\text{noisy}}$ is more likely to be a noise-only chunk, and accurately represent noise power. Furthermore, if there are no noise-only chunks, the best noise power estimation will be made by taking the power of the chunk with the least amount of original signal. If the signal being restored has a large dynamic range, this is more likely to happen in $\min P_{\text{noisy}}$.

This suggests that it could be a good idea to use a narrower SNR range for recordings where $\max P_{\text{noisy}} \gg \min P_{\text{noisy}}$. Additionally, considering that $\min P_{\text{noisy}}$ might contain original signal content, it is also important to note that $\tilde{S}$ tends to underestimate the local SNR of each block.

With these factors in mind, the second heuristic used $S_{\text{low}} = \tilde{S}$ as the low end of the SNR range, and calculated the high end as

$$S_{\text{high}} = S_{\text{low}} + \left(43 + \max \left(-40, 10 \log_{10} \left(\frac{\min P_{\text{noisy}}}{\max P_{\text{noisy}}} \right) \right) \right). \tag{4.7}$$

This formula for the range considers the possible underestimation of the local SNR by $\tilde{S}$, and extends the range beyond the initial estimation. The range narrows down to a minimum of 3dB as the relative distance between $\max P_{\text{noisy}}$ and $\min P_{\text{noisy}}$ increases in decibels. This captures the idea of better estimation for noisy signals where $\max P_{\text{noisy}} \gg \min P_{\text{noisy}}$.

For signals with small dynamic range, the estimate $\tilde{S}$ is unable distinguish between a *fortissimo* piece with little noise and no silent parts and a *pianissimo* piece with very low SNR. Both pieces will have consistently low SNR estimates for their blocks, even though the first probably has very high SNR overall. This heuristic increases the SNR range in both cases, and approximates the no-assumptions SNR range of $[0, 40]$dB.

For signals with large dynamic range, the second heuristic avoids excessively large ranges for high SNRs. In a signal where $\max P_{\text{noisy}} \gg \min P_{\text{noisy}}$ the variations in $\tilde{S}$ between blocks can be very large. For the blocks with higher $\tilde{S}$, such as 30dB or 40dB, it would not make sense to have a SNR range extending to 50dB or 60dB, as noise is imperceptible for these SNRs[9]. By using a narrower range, useless computations with imperceptible SNRs are avoided for these blocks.

These two heuristics were compared to the no-assumptions ($[0, 40]$dB) SNR range method, and to the known gain case using preludes 7 and 20 degraded with global SNRs of 10, 20, 30, and 40dB. For both heuristics, a sliding window size of 8192 samples (roughly 0.2s at 44.1kHz sampling rate) was used. The optimal values of ξ' according to Table 4.4 were used for each SNR, and the same noise sample of previous experiments was used to degrade the original preludes.

Table 4.8 shows the ODGs for the noisy versions of prelude 7. The ODGs for noisy prelude 20 can be seen in Table 4.3. Table 4.9 shows the ΔODGs per SNR for the heuristics with prelude 7, and Table 4.10 shows the same results for prelude 20.

Table 4.8: ODG for noisy versions of prelude 7 degraded with different SNR levels.

	SNR			
	10	20	30	40
ODG	−3.9132	−3.9128	−3.8887	−3.1883

Table 4.9: Variations in ODG for versions of prelude 7 restored using the two proposed heuristics. The noisy signals were prepared with four different SNRs.

ΔODG	SNR			
	10	20	30	40
Known gain	0.6864	**2.1105**	2.6305	2.4904
SNR range $[0, 40]$dB	0.8009	2.0061	2.3588	1.8079
Gain estimation heuristic	0.8592	1.9685	**2.7618**	**2.5865**
SNR range heuristic	**1.1737**	2.0410	2.7184	2.5786

[9]This was tested informally by degrading clean recordings with noise samples from the Great 78 dataset and listening to the results.

Table 4.10: Variations in ODG for versions of prelude 20 restored using the two proposed heuristics. The noisy signals were prepared with four different SNRs.

ΔODG	SNR			
	10	20	30	40
Known gain	**0.8275**	**2.0728**	**3.2507**	3.5104
SNR range $[0, 40]$dB	0.2164	1.2636	2.6888	2.9641
Gain estimation heuristic	0.6037	1.3462	2.8767	3.5265
SNR range heuristic	0.3546	1.1963	3.1165	**3.5293**

The best values obtained for each SNR are marked in bold in both tables. Interestingly, denoising using the same gain used to degrade the recordings did not always[10] produce the best results. While restoration with the proper gain had the overall best performance for prelude 20 (with only 0.02 ΔODG difference to the winner for 40dB), for prelude 7 this was not the case for 10, 20, and 40dB SNRs. Additionally, for 30dB SNR, the difference between the ΔODGs of the heuristics and of the known gain method was of just around 0.1 ΔODG.

A possible explanation is that the values of ξ' that were used for both preludes were the ones in Table 4.4. These values were optimized for prelude 20, and as it is possible to see in Table 4.5, they were not exactly the same for preludes 16 and 7 (although close).

Perhaps the estimation errors in $\tilde{G}$ and in the SNR range for prelude 7 somehow compensated the sub-optimal values of ξ'. Prelude 7 required a slightly larger value of ξ' in Table 4.5, possibly because its quieter dynamics made $\xi' = 2.45$ too restrictive. For the SNRs of 10, 20, and 40dB a similar effect might exist, and a slight overestimation of the noise power could have made the values of ξ' that were used just right.

However, it is important to observe that the ΔODG values for denoising with the proper gain in prelude 7 were (in most cases) close to the best values obtained. This is even more evident when comparing the distance between the results using the heuristics and using the correct gain in prelude 20.

Table 4.11 shows the mean ΔODG for each of the methods considering all SNRs for both prelude 7 and prelude 20. Denoising with the correct adjustment factor obtained the best mean ODG variations, despite the possible use of sub-optimal values of ξ' for prelude 7. The heuristics obtained similar results, and were reevaluated on a validation set later on (see Section 4.3).

[10]Strictly speaking, it would be necessary to carry out a statistical test to verify this, as the method is stochastic. However, this was not possible due to computational constraints, and in all experiments in this work the results are reported for a single denoising run. An informal test was conducted using prelude 20, and the PEAQ differences between the results were insignificant (on the order of 0.001).

Table 4.11: Mean ODG variations for the two proposed heuristics over all SNRs with preludes 7 and 20.

	Mean ΔODG
Known gain	2.1974
SNR range $[0, 40]$dB	1.7633
Gain estimation heuristic	2.0665
SNR range heuristic	2.0882

Considering the possible compensation of an incorrect $\tilde{G}$ by certain values of ξ', and also the fact that the optimal value of ξ' would be unknown in a real restoration scenario (as the SNR of the signal being restored is likely unknown), two strategies for estimating ξ' were also designed.

The first one relied on the estimate $\tilde{S}$ of the local SNR of each block and on the linear relation visible in Table 4.4, and used

$$\xi' = \max\left(0.105 \times \tilde{S} - 0.7, 0.35\right), \tag{4.8}$$

where the maximum was used to avoid values of ξ' lower than 0.35, as these appeared to be too restrictive in the other preliminary tests (see Figure 4.6).

This strategy can be employed per block (using the block's $\tilde{S}$ directly), or per iteration (using the SNR sampled from the range defined by $[S_{\text{low}}, S_{\text{high}}]$). Figures 4.12 and 4.13 show the results for denoising the 30dB SNR prelude 20 estimating ξ' per block and per iteration, respectively. In both cases, the variable range method for estimating G was used.

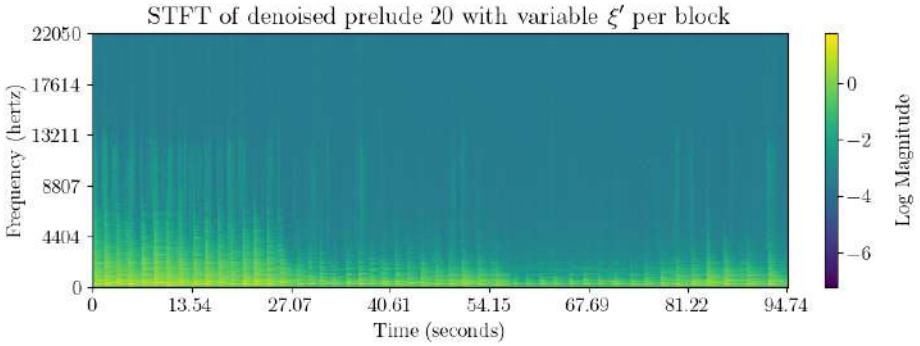

Figure 4.12: Spectrogram of denoised 30dB SNR prelude 20 estimating ξ' per block. The residual noise is apparent on the peaks of the first half of the signal.

In the two cases, there is clearly some residual noise in the peaks at the start of prelude 20. Since the estimate for ξ' for this strategy relied on the local SNRs, the first blocks of prelude 20, which are played in *fortissimo*, could have had values of ξ' excessively high. The result is a residual noise that follows the piece's dynamics, which is particularly annoying to hear.

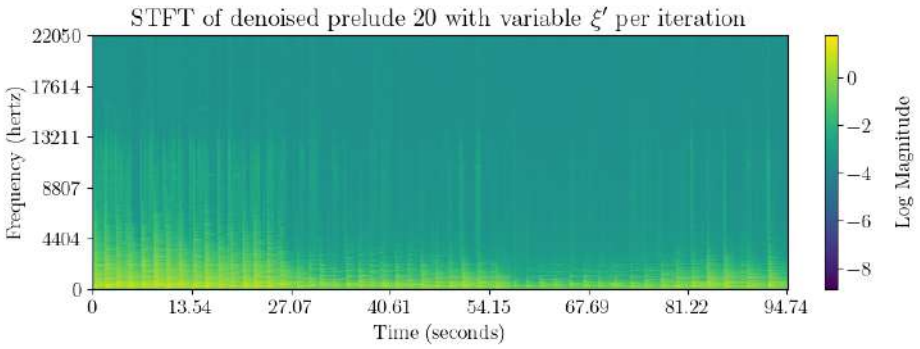

Figure 4.13: Spectrogram of denoised 30dB SNR prelude 20 estimating ξ' per iteration. The residual noise is also apparent on the peaks of the first half of the signal.

These results suggested that perhaps ξ' should not be a function of the local SNRs, but of the noise power alone. With this in mind, for the second strategy for estimating ξ' a linear regression was used to find the relation between the noise power and ξ'.

For the previous test signals, i.e. prelude 20 with 10, 20, 30, and 40dB SNR, and preludes 7 and 16 with 30dB SNR, the noise powers and optimal ξ' were known. Because the noise sample that was used to generate the degraded versions had unit power, the noise powers were simply G^2.

In addition, since the results in Table 4.4 showed the best values of ξ' growing linearly for increases in the SNR in decibels, it seemed appropriate to use the logarithm of the noise power as the independent variable. The relation between ξ' and the log noise power, obtained after linear regression, was

$$\xi' = \max(0.35, -1.0246 \times \log_{10} G^2 - 4.0884), \tag{4.9}$$

where G^2 is the estimate of the noise power. The p-value for the slope coefficient was 4.3407×10^{-5}, and the R^2 of the regression was 0.9892. Once again the lowest ξ' was capped at 0.35. Table 4.12 shows the values that were used for the regression.

Table 4.12: Relation between noise powers and ξ' for selected preludes. There appears to be a linear relation between the noise powers and the optimal value of ξ'.

Prelude	SNR	G	$\log G^2$	ξ'
20	10	0.0062	−4.4113	0.35
20	20	0.0020	−5.4113	1.4
20	30	0.0006	−6.4113	2.45
20	40	0.0002	−7.4113	3.5
16	30	0.0015	−5.6410	1.925
7	30	0.0003	−6.9539	2.975

The second strategy for estimating ξ' was tested in combination with the gain estimation and SNR range heuristics. As before, restoration quality on preludes 7 and 20 degraded with 10, 20, 30, and 40dB SNR was evaluated. The sliding window used to estimate $\tilde{P}_{\text{noise}}$ was the same that was used to estimate $\tilde{G}$ and $\tilde{S}$, a flat window of length 8192 samples. Table 4.13 and 4.14 show ΔODGs for preludes 7 and 20, respectively.

Table 4.13: Variations in ODG for versions of prelude 7 restored using estimated ξ' combined with gain estimation and SNR range heuristics.

ΔODG	SNR			
	10	20	30	40
ξ' estimation + gain estimation	0.0092	0.0808	1.7031	2.6265
ξ' estimation + SNR range	0.0016	0.0655	1.8832	2.6122

Table 4.14: Variations in ODG for versions of prelude 20 restored using estimated ξ' combined with gain estimation and SNR range heuristics.

ΔODG	SNR			
	10	20	30	40
ξ' estimation + gain estimation	0.0115	0.0160	0.6740	3.3436
ξ' estimation + SNR range	0.0022	0.0137	1.1315	3.3750

It is possible to see that the results were very poor overall. For 10 and 20dB SNR the ODG variation was near zero, and for 30dB it was below the results obtained using ξ' according to Table 4.4. For 40dB SNR, the results were comparable, but not significantly better.

Given the unsatisfactory results with both techniques for estimating ξ', at this point the decision was made to work with predefined values for ξ' only. Whenever the global SNR S was known, the following pairs of bands and values were used

$$\xi' = \begin{cases} 0.35, \text{ if } S \leq 20; \\ 1.4, \text{ if } 20 < S \leq 30; \\ 2.45, \text{ if } 30 < S \leq 40; \\ 3.5, \text{ if } S \geq 40. \end{cases} \tag{4.10}$$

For cases where the global SNR was unknown, ξ' was manually set to one of these values after careful listening of the signal being restored.

Up until now, all experiments focused on determining the best strategies for 78 RPM noise removal starting from the parameters described in Table 4.1. First, the two methods for conditional sampling were compared, in order to determine which one to use with the proposed method. Then, experiments trying to verify

the robustness of the method to different signals and different noise levels followed. Finally, heuristics for determining the conditioning factor ξ' and the gain G to be used in $\mathcal{A}(\cdot)$ were proposed. Sections 4.4 and 4.5 are dedicated to testing the method, using a gain heuristic that is validated in the following section.

4.3 Validation

After the preliminary experiments, a validation round was set up to determine which of the gain heuristics to use, as well as to evaluate the performance of the method in a more diverse set of signals. For the heuristics, a validation stage also provided the chance to explore windows of different sizes when calculating $\tilde{S}$ and $\tilde{G}$.

Artificially degraded signals were used once again in order to evaluate the performance over different SNRs. In total, sixteen different signals were used for validation, with four signals being degraded with each of the SNRs of 10, 20, 30, and 40dB.

The original versions were 9s to 24s excerpts of eight solo piano pieces written by Samuel Barber, Gyorgy Ligeti, and Robert Schumann, recorded between 1993 and 1995. The pieces were chosen trying to cover different playing dynamics and pitch ranges, and also trying to include possible restoration pitfalls, such as silent parts. Two excerpts were extracted from each piece, and an identification number was attributed to each of them. For the validation files, the IDs were numbers between 33 and 48. The Github repository of this dissertation[11] contains a CSV file with the metadata of the excerpts, and the SNR that was applied to each of them.

The training dataset did not have any recordings of the pieces that were used in the validation set. The validation noise samples of the gramophone noise dataset were used to degrade the original signals, applying extension as described earlier whenever necessary. To simulate the case where a noise sample from the recording being restored is unavailable, only the train split of the noise dataset was included in the inference set of the diffusion model.

Validation evaluated only the heuristics for setting up G. All inferences used RG conditional sampling with ξ' according to Equation (4.10), and diffusion parameters as described in the beginning of Section 4.2, i.e., $\sigma_{\min} = 10^{-6}$, $\sigma_{\max} = 10$, $\rho = 13$, and $S_{\text{total}} = 5$. The remaining stochastic sampling parameters ($S_{\min}$, $S_{\max}$, and $S_{\mathbf{z}}$) were not used.

Three versions, using flat windows of length of 4096, 8192, and 16384 samples, of the two heuristics were evaluated. Denoising with a fixed SNR range of $[0, 40]$dB and using the gains required to degrade each excerpt with its corresponding SNR were also included in validation for comparison. In total, eight inference configurations were evaluated. For convenience, they were numbered as shown in Table 4.15.

[11] `https://github.com/bvm810/diffusion-audio-restoration`

Table 4.15: Numbering of the validated heuristics.

	Heuristic	Window Length
Configuration #1	SNR range	4096
Configuration #2	SNR range	8192
Configuration #3	SNR range	16384
Configuration #4	Gain estimation	4096
Configuration #5	Gain estimation	8192
Configuration #6	Gain estimation	16384
Configuration #7	SNR range $[0, 40]$dB	-
Configuration #8	Known gain	-

The evaluation method was the same used in the preliminary experiments. Since inference was made for artificially degraded signals, it was possible to use PEAQ's ODG variation after restoration as an audio quality metric.

Table 4.16 shows the mean ΔODGs obtained over all excerpts for each configuration. Detailed results for the sixteen excerpts can be found in the companion web page. It is interesting to observe that denoising with the proper gain did not obtain the best results overall, possibly because of mismatches between the optimal values of ξ' for each excerpt and the values that were used.

Table 4.16: Validation mean ΔODGs. Configuration #5, gain estimation with window of 8192 samples, obtained the best results and is marked in bold.

	Mean ΔODG
Configuration #1	1.0597
Configuration #2	1.2262
Configuration #3	1.3481
Configuration #4	1.2768
Configuration #5	**1.3897**
Configuration #6	1.2993
Configuration #7	1.3213
Configuration #8	1.1351

Configuration #5 obtained the best results (although not by a large margin), and so was selected as the model to be used during objective and subjective testing. Its mean ΔODG is marked in bold in Table 4.16.

Because of its more sound theoretical background, denoising with the same gain that was used to degrade the original signals was also not discarded for testing. Considering the possible effect of ξ' mismatches, and also the fact that each signal was evaluated for only one of the four SNRs, it seemed a good idea to include this method of denosing in order to have some reference values during testing.

Table 4.17 shows the mean ΔODG per SNR level. The best mean ΔODG for

each SNR level is marked in bold. For all configurations, the worst ΔODGs occurred for the SNR of 10dB, which shows the method's poorer performance for low SNRs.

Table 4.17: Validation mean ΔODG per SNR. The worst performances for all methods happened for 10dB SNR.

Mean ΔODG	SNR			
	10	20	30	40
Configuration #1	**0.4441**	2.0632	0.8320	0.8996
Configuration #2	0.4279	2.1534	1.3899	0.9337
Configuration #3	0.3932	**2.1926**	1.9556	0.8509
Configuration #4	0.3174	2.1414	2.0948	0.5536
Configuration #5	0.2502	2.1178	**2.3431**	0.8477
Configuration #6	0.2355	2.0696	2.2299	0.6622
Configuration #7	0.2610	1.9584	1.9862	1.0797
Configuration #8	0.2768	1.9821	1.1285	**1.1529**

It also seems counter-intuitive that the average denoising quality improvement for 40dB SNR was considerably lower than for 20 and 30dB SNR. However, two of the 40 dB SNR excerpts (IDs 36 and 48) had a high ODG between the original and noisy versions (-1.5035, and -0.2200, respectively, *versus* < -3 for the other excerpts in all SNRs). This made the ODG variations for these two excerpts much smaller, which affected the mean for this SNR.

The big fluctuation in ODG for the four noisy excerpts with 40dB SNR is due to the fact that the SNR is only a measure of relative noise power, and does not consider perceptual factors. Differences in dynamics can make two degraded musical signals with the same SNR have radically different absolute noise power levels, which impacts the perception of noise.

Because of this, since PEAQ takes masking and other psychoacoustical phenomena into account, two excerpts with the same SNR can have very different ODGs. This point is somewhat illustrated by Figures 4.14 and 4.15, which show the (very similar) spectrograms of the clean and noisy versions of excerpt 48. The reader is also invited to see the detailed results for each excerpt in the companion web page.

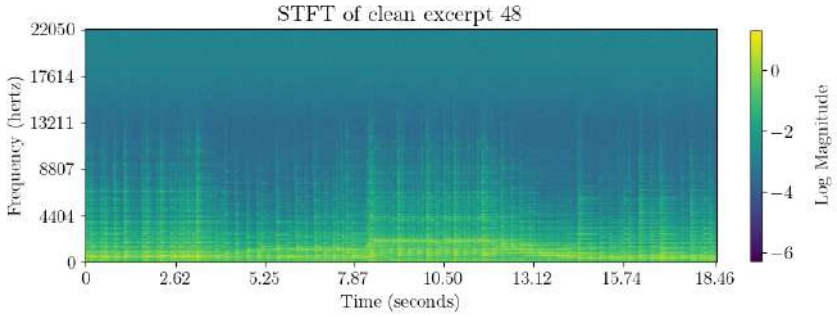

Figure 4.14: Spectrogram of the clean version of excerpt 48.

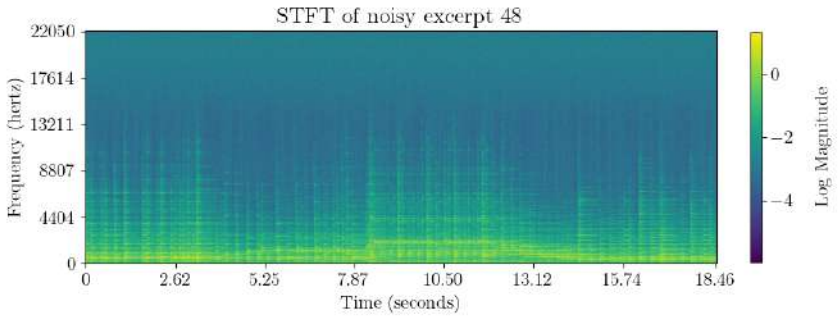

Figure 4.15: Spectrogram of the noisy version of excerpt 48. It is possible to see that it is very similar to the clean spectrogram.

4.4 Objective Evaluation

With the validation results in hand, the last step in evaluating the denoising techniques presented in this work was performing objective and subjective tests on unseen audio signals. In summary, the idea of both objective and subjective tests was to compare the restoration techniques presented in this work against a known benchmark for denoising of historical recordings.

For the objective tests, the setup was very similar to the one used in validation. Artificially degraded signals were used to evaluate performance over 10, 20, 30, and 40dB SNRs using ΔODG as the evaluation metric.

The original excerpts were also extracted from solo piano recordings of pieces written by Samuel Barber, Gyorgy Ligeti, and Robert Schumann. All recordings were made between 1993 and 1995, and all excerpts had length between 9s and 23s.

Thirty-two excerpts were extracted from sixteen different recordings (two per recording). The criteria for choosing the sixteen recordings was similar to the criteria for selecting recordings for validation. An effort was made to cover a large range of playing dynamics with variations in loudness and speed in the selection, and musical pieces with large silent parts were also included to see the impact of non-musical frames in restoration performance.

Each excerpt was given a numerical ID ranging from one to thirty-two. Excerpts one to sixteen were also used for the subjective tests, as will be further explained in Section 4.5.

For the objective evaluation, all excerpts were combined with all SNRs (10, 20, 30, and 40dB). In other words, for each excerpt four degraded versions were created, one for each SNR. In total, 128 artificial noisy samples were restored for the test.

The noise samples used to create the noisy excerpts were taken from the test split of the gramophone noise dataset. Only the train split was included in the inference set of the denoising model.

For all four noisy versions created from an original excerpt, the same noise sample

was used. This made it possible to analyze the results of increasing the noise level in each of the excerpts.

Once again, the Github repository of this dissertation contains CSV files with the metadata of the excerpts, the noise sample that was used to create their noisy versions, and the ODGs for each of them.

A retrained version of the model of [31] (mentioned in Chapter 1) was used as the test's benchmark, with a few adjustments to the training procedure. As a reminder, the model of [31] consists of a two-stage U-Net where the first stage estimates the noise components in the signal to be restored and the second stage reconstructs the clean signal based on an attention-enhanced version of the first stage's output.

The inputs of the network of [31] are the real and imaginary parts of the STFT of the noisy signal (using 2048-point Hann windows and hop of 512 samples), and a combined absolute error loss on the elements of the STFT matrix is used to train the two stages simultaneously. Both U-Net stages rely on residual blocks, and the positional encodings of [90] are used as frequency embeddings instead of RFFs.

In [31], the model was trained using artificially degraded recordings of multi-instrument classical music [91] with SNRs in the range $[2, 20]$dB. While this might be appropriate for restoring very old (1890-1920) gramophone recordings, it could be excessively restrictive for general purpose gramophone noise removal including well preserved post-1920 recordings. Considering this, and the fact that the test signals in this work were degraded with SNRs in the range of $[10, 40]$dB, it seemed fair to re-purpose the model for denoising with higher SNRs, as using the original weights could mean applying the model to out-of-distribution data.

In addition, the benchmark model was retrained using the new splits of the gramophone noise dataset. Some noise samples appeared to be missing from the noise dataset metadata available online, and because of this using the new splits was a way to ensure reproducibility. If the old splits (or the original checkpoint of [31]) were used, there would be no way to know if certain excerpts were in the training set or not.

Aside from these two changes, the training procedure was mostly the same as described in [31], with small changes only to the batch size, number of iterations, and learning rate decay step, due to computational constraints. In [31], the model was trained using audio samples of 5s and a batch size of 8 samples for 300,000 iterations, with a learning rate of 10^{-4} and a learning rate decay factor of 0.1 every 100,000 step. Here, the batch size was reduced from 8 to one, while keeping the same sample length. Because of this, the number of iterations was increased to 2,400,000 according to the same proportionality criterion that was used to adapt the number of iterations of the retrained diffusion model.

The learning rate decay factor was applied after every 800,000 steps for the same

reason, but the rest of the training setup was kept exactly as in [31]. The reader is invited to see [31] if any additional details on the benchmark model are needed.

A positive aspect of the benchmark model that should be highlighted is its inference speed in comparison to the proposed diffusion framework. Diffusion inference is slow due to the sequential nature of the sampling process, which cannot be parallelized. With the same hardware[12], inference for a 5s musical signal took a few seconds for the benchmark model, and around 20 minutes with the diffusion pipeline.

Additionally, because of the difficulty of unconditional generation, the diffusion models in this work require more data in absolute terms, despite not needing pairs of noisy and clean signals. The dataset used to train the benchmark model [91] has around 40 hours of music, which is five times less than MAESTRO.

A noteworthy point of the model of [31] is that it was developed for denoising signal blocks of fixed duration of 5s. When treating longer signals, this means dividing the signals being denoised into (possibly overlapping) blocks of 5s for treatment. The default behavior of the code of [31] is padding noisy signals with zeros, so that they can be divided in an integer number of blocks.

During test preparation, it became clear that both the original pretrained model and the retrained version behaved unexpectedly for blocks which consisted mostly of zeros. For these blocks, which appeared at the end of the signals because of the padding procedure the authors of [31] applied, noise increased instead of being removed. This may have gone unnoticed due to the use of 5s test signals in [31], which did not need to be divided in sub-blocks. Although their website presents also some examples of longer signals restored with overlap-and-add, in most cases this effect is not evident because the last block after padding the signal to be restored had a short trail of zeros.

This noise increase effect can be reproduced by using the code of [31][13] on a signal of length 10.005s. With their implementation of OLA, there will be three 5s blocks to be denoised: the first two consisting of the signal until a little before 10s, and the last consisting of the remaining part of the signal followed by a long sequence of zeros (needed to make the last block have length 5s). The last block will consist mainly of zeros, and the unexpected noise will be visible at the end of the output signal.

This issue with the benchmark was the reason for choosing test signals with length between 9s and 23s. The excerpts were extracted from the original recordings being careful to include complete musical phrases, but also making sure that they

[12] A server with the fastest GPU available for the author, an NVIDIA GeForce RTX 3090.

[13] A demo for executing the code of [31] can be found at `https://colab.research.google.com/github/eloimoliner/denoising-historical-recordings/blob/master/colab/demo.ipynb` as of 24/07/2024. The code is identical to the one on the Github repository, and can be used to reproduce the bug online.

would all require little or no padding. The duration of the excerpts is explained by the fact that this work used 5s blocks with 0.5s overlap, and that excerpts were chosen with length between two and five blocks. Blocks were created using the same window as the diffusion restoration methods discussed previously.

Another important detail concerning the benchmark method is that it preserved the dynamic range of the noisy signal. Since noisy excerpts were created using division by two to avoid saturation, the restored excerpts of the benchmark method had half the amplitude of the original signals. Because PEAQ is sensitive to loudness, the noisy and benchmark restored versions were both multiplied by two (making sure that there was no saturation), so that they could be compared to the original excerpts. This was not necessary for the diffusion restored signals, as reverting $\mathcal{A}(\cdot)$ removed the $\frac{1}{2}$ gain of the noisy versions.

Three restoration methods were evaluated by this test: neural network restoration with the benchmark model, diffusion restoration with gain estimation (validation configuration #5), and diffusion restoration using the gains required to degrade the noisy samples with their corresponding SNRs (configuration #8).

Although not useful in practice, the third method served for comparison with diffusion restoration with an estimated gain, as it is an upper bound on the latter's performance assuming optimal ξ'. Since the test set evaluated all samples with all SNRs, the ξ' compensation effect was supposed to have a smaller impact, and the third method was expected to outperform the second.

Excerpts number seven and eight of the test set were taken from a recording of Schumann's Op. 7, "Toccata in C Major", which is also present in MAESTRO. As mentioned earlier, the MAESTRO versions were recorded in different conditions, with different pianists, and their inclusion in the test set did not significantly alter the results.

Excluding such tracks from the test set, the average results for diffusion restoration with known gain increased by around 0.1ΔODG, and the average results for diffusion restoration with estimated gain decreased by around 0.1ΔODG. Also, the results per SNR for these two excerpts were not very different from the results for the other excerpts, as will be seen in Figure 4.23. All results below are given including these two excerpts in the dataset.

Table 4.18 shows the mean ΔODGs for each of the three methods. The confidence intervals were set using Gaussian distributions and a 95% degree of confidence [92]. It is possible to see that diffusion based restoration was unable to outperform the benchmark.

Figure 4.16 shows boxplots corresponding to the ΔODG distributions of the three restoration methods. The three lines on the boxes represent the first, second, and third quartiles of the samples. The second quartile is the sample median, and

Table 4.18: Test mean ΔODGs for the restoration methods. Diffusion based restoration was unable to outperform the benchmark. Confidence intervals were set using the z-scores for a 95% degree of confidence.

	Mean ΔODG
Benchmark	1.9993 ± 0.1277
Diffusion + estimated gain	1.2602 ± 0.1827
Diffusion + known gain	1.3142 ± 0.1930

the distance between third and first quartile is called the interquartile range (IQR). The whiskers extend to samples that lie within $1.5 \times$ IQR of the lower and upper quartiles. Samples outside of this range are considered outliers, shown as circles.

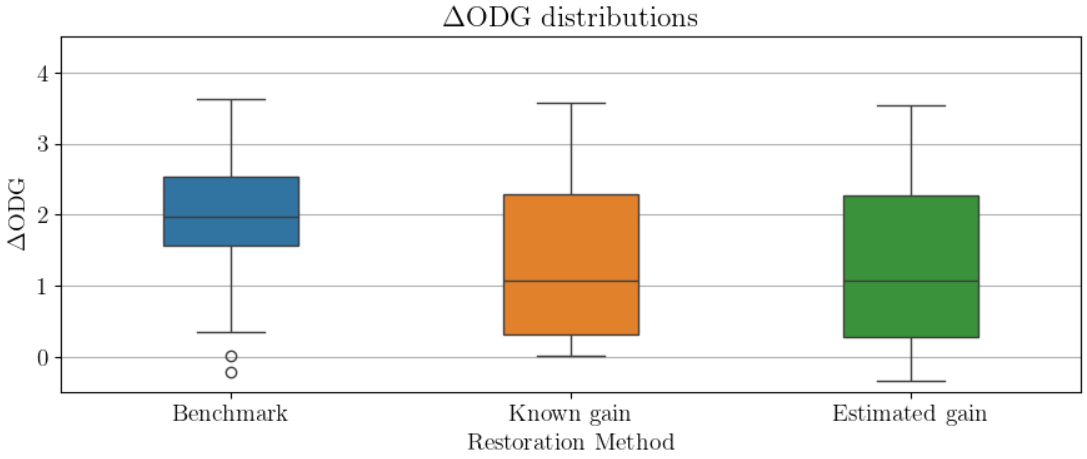

Figure 4.16: Boxplots of the ΔODG distributions of the three restoration methods. The benchmark method's distribution is narrower and has higher median value.

It is possible to see that the benchmark method's distribution is narrower, and also that it is concentrated mostly above the median of the other two methods. However, it is important to consider from the results of Sections 4.2.3 and 4.2.4 that ξ' appears to be dependent on the absolute noise power. Since the values ξ' were set based on the SNR of each excerpt, the larger width of the diffusion boxplots could be a result of suboptimal values of ξ'.

The distributions for restoration with the proper gain and with gain estimation overlap considerably. This suggests that restoration with the estimated gain accurately models $\mathcal{A}(\cdot)$, without considering possible errors in ξ'.

All three methods had outputs with ΔODG close to (or worse) than zero. This was probably caused by certain examples where the ODG between the original and noisy versions of the excerpt was already on the upper range of PEAQ. The reader is invited to verify this in the detailed results CSV file, which can be found in the Github repository.

In the second and third methods, the smaller distance between the minimum ΔODG and the first quartile can be explained by the poor performance for diffusion based restoration for 10dB SNR excerpts.

The higher range samples in all three cases mostly correspond to 40dB SNR excerpts which originally had very low ODGs with respect to their original versions. Restoration in these cases left less artifacts than in most 20 or 30dB SNR situations, which explains the long whiskers on top of the three boxplots. This can be verified looking at the results for individual tracks (available in this work's Github repository).

Listening to the restored excerpts, it seems that the benchmark model removes more noise, at the cost of potentially cutting out signal content. Diffusion models, on the other hand, appear to preserve signal content, at the price of not removing noise is some signal blocks. This is illustrated in Figures 4.17 to 4.20.

Figures 4.17 and 4.18 show the clean and 30dB SNR versions of excerpt 19 for reference. The output spectrograms for the benchmark and diffusion restoration with estimated gain methods can be seen in Figures 4.19 and 4.20, respectively. It is possible to see that the benchmark method removes more noise, but that it also cuts some frequency peaks around 15s. The estimated gain method keeps those same peaks at the cost of also keeping noise content.

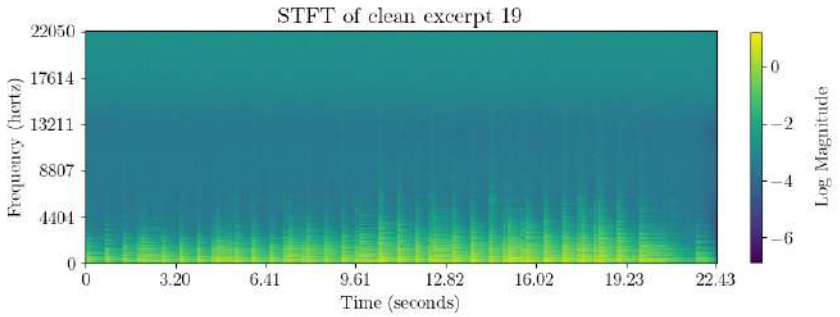

Figure 4.17: Spectrogram of the clean version of excerpt 19.

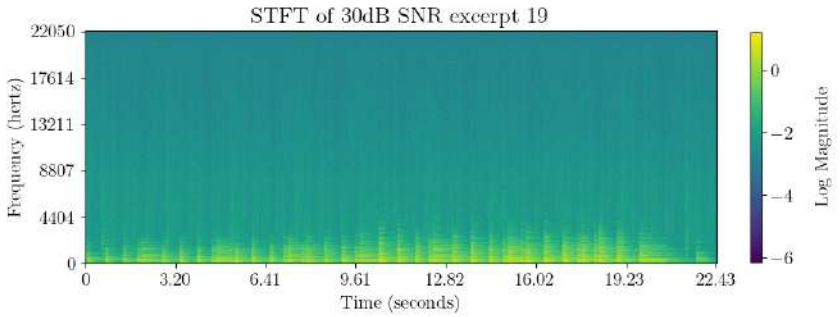

Figure 4.18: Spectrogram of the 30dB SNR version of excerpt 19.

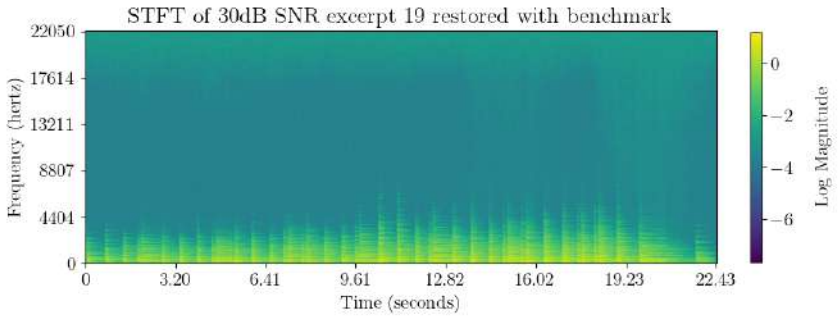

Figure 4.19: Spectrogram of the 30dB SNR version of excerpt 19 restored with benchmark. It is possible to see the reduction of frequency peaks around 15s.

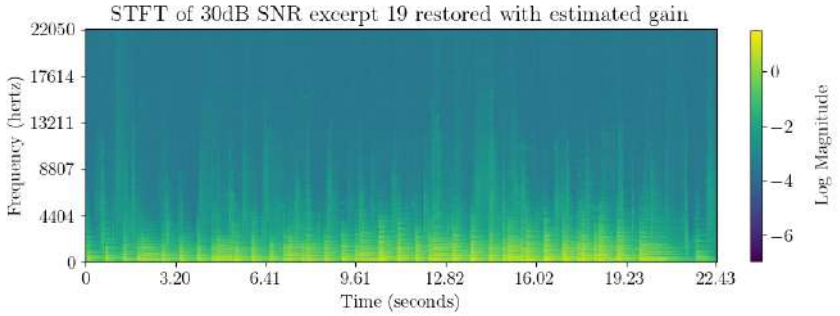

Figure 4.20: Spectrogram of the 30dB SNR version of excerpt 19 restored with estimated gain. Residual noise is visible in the spectrogram.

For lower SNRs, it can be perceptually advantageous to be more rigorous with respect to noise removal, particularly if the alternative is removing noise unevenly across blocks, as seems to be the case with the diffusion methods. On the other hand, for higher SNRs preserving most of the signal pays off, even if there is some residual noise.

This can be seen on Figure 4.21, which shows the ΔODG boxplots divided by method and SNR. From the image, it is possible to observe that the benchmark method consistently achieved better results for lower SNRs. In the boxplots for 10 and 20dB SNRs, the benchmark distribution is concentrated above the third quartile of the other two distributions.

For 30 and 40dB, benchmark superiority is less pronounced. Diffusion based denoising improves as the SNRs increase, as can be seen by the increasing median values for these methods in Figure 4.21. For 40dB SNR, there is considerable overlap between the three distributions.

Additionally, because the diffusion methods seem to be dependent on the absolute noise power through ξ', grouping the results by SNR might result in large variance distributions for them. Variance increase due to SNR grouping accumulates with the variance increase brought by potentially suboptimal values of ξ', and to the natural widening of the distributions caused by the larger spread of ODG values between

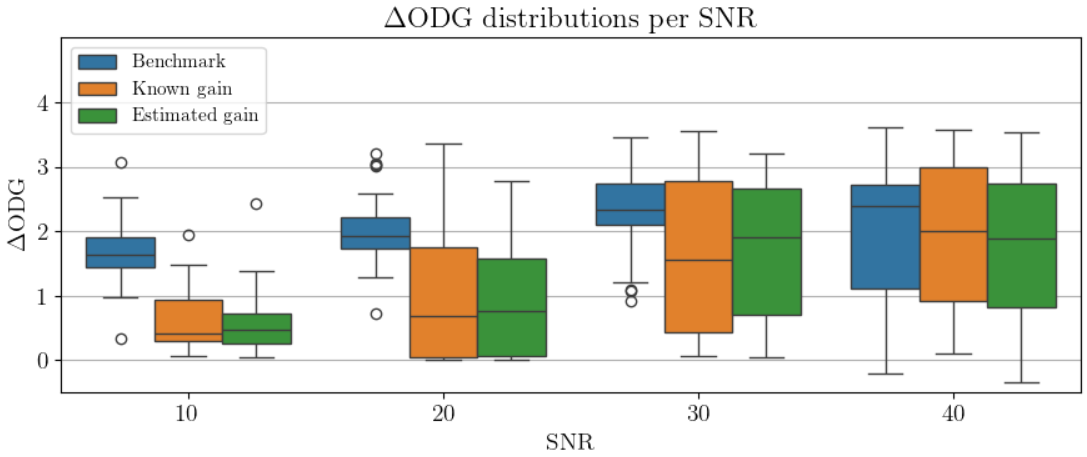

Figure 4.21: Boxplots of the ΔODG distributions of the three restoration methods per SNR.

noisy and clean excerpts for high SNRs[14].

To account for the dependency of the diffusion methods on the absolute noise power, the relation between ΔODG and the noise gain of the degraded excerpts was also studied. The plot in Figure 4.22 shows the relation between ΔODG and the logarithm of the noise gain for the three methods.

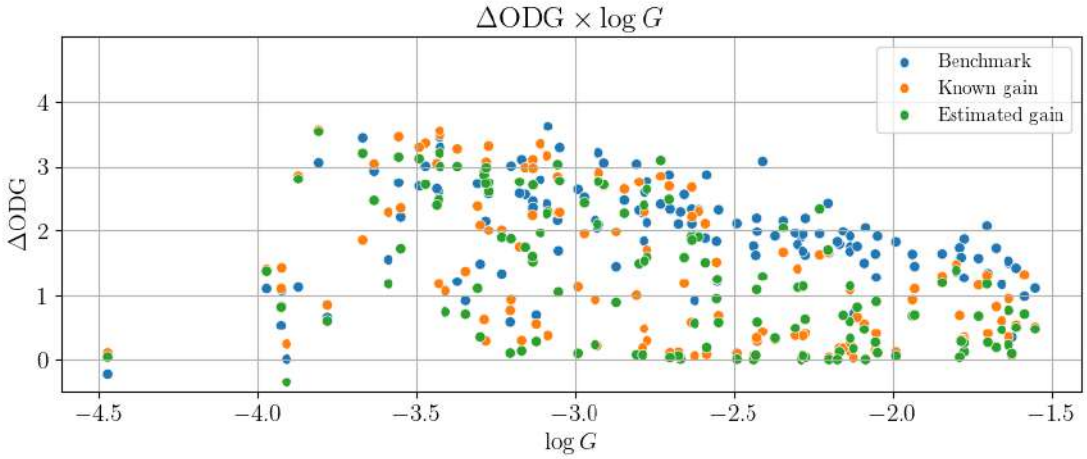

Figure 4.22: ΔODG *versus* log noise gain scatterplot for the three methods. The benchmark is superior for higher gains, but comparable as the noise gain decreases.

Looking at the blue dots on the graph from right to left, it is possible to see that the benchmark method is superior for higher gains, but that diffusion methods start to show comparable results as the noise gain decreases. This suggests that diffusion based denoising might be more suited to cases with less noise, where it might be beneficial to maintain maximum signal integrity, at the cost of leaving residual noise.

[14]This larger spread can be seen looking at the individual track results for this test.

This hypothesis goes in the same direction of the example shown in Figures 4.17 to 4.20, where it was shown that diffusion methods were less strict than the benchmark with respect to noise removal. The preliminary experiments of Section 4.2.2, where the best ΔODG decreased along with the SNR for a single signal, also seem to corroborate this idea.

Another way to confirm this hypothesis is to look at the variation of ΔODG versus SNR for individual signals. If the signal remains unchanged, reducing the SNR implies increasing the noise power only, which allows to see the methods' response to absolute noise level.

Furthermore, knowing the dynamics of the excerpts, more information can be inferred about the methods' relation to noise power. To achieve the same SNR, quiet excerpts will have less noise power than loud excerpts. Therefore, if diffusion methods are comparable to the benchmark in lower noise power settings, they should have similar results in *piano* excerpts.

Figure 4.23 shows the ΔODG × SNR plots for tracks one to sixteen. The excerpts are ordered from top-left to bottom-right according to increasing noise power at 30dB SNR.

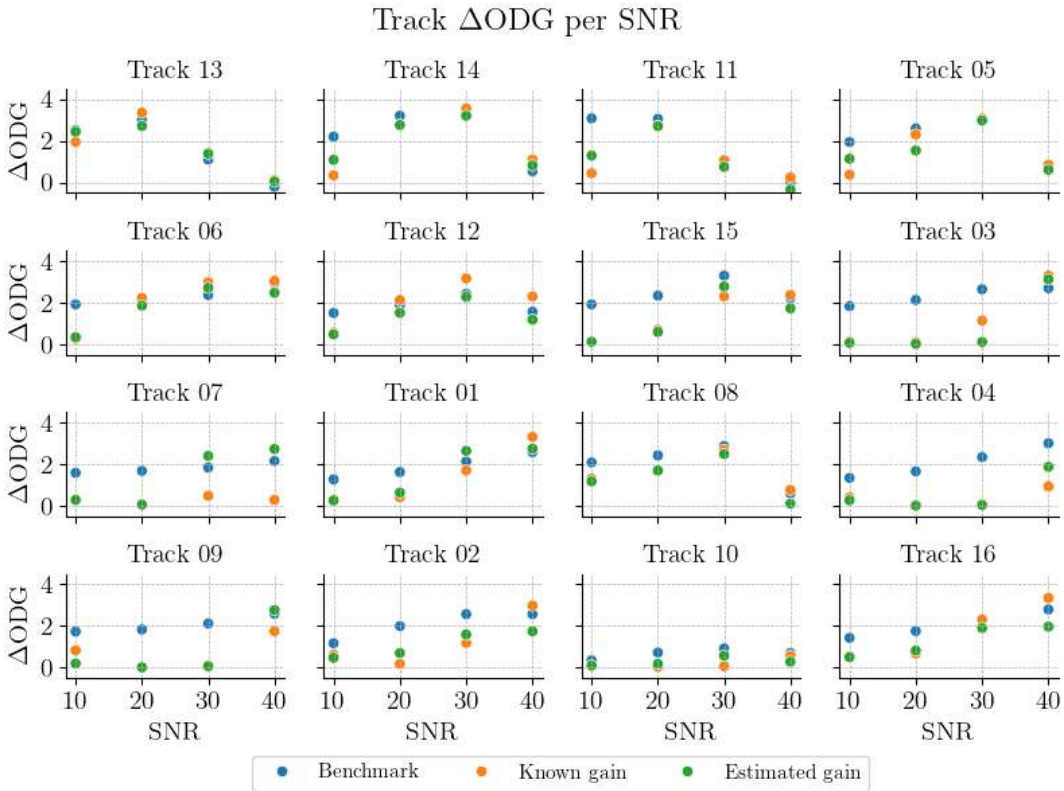

Figure 4.23: Track ΔODG per SNR. The excerpts are ordered according to increasing noise power at 30dB SNR.

Looking at the plots individually it is possible to see that the diffusion methods

show an overall tendency to increase their performance as the noise power decreases. There are exceptions (notably excerpts thirteen and eleven), but the benchmark performance below 2ΔODG above 20dB SNR for them suggests that these cases already had an ODG between noisy and clean versions near PEAQ's upper range.

Observing the plots in order shows that the performance of the benchmark and diffusion methods is comparable only for *piano* excerpts. From excerpt number three (which is the median track in terms of noise power) onwards, the benchmark dominates the diffusion methods.

Certain individual excerpts are noteworthy as well. Tracks number three and four are interesting because they have long silences. Since diffusion methods tend to remove less noise, the restored versions of these excerpts had chunks of unmasked noise, which caused a considerable negative impact in their scores.

In conclusion, the objective tests seem to indicate that the diffusion denoising methods are more lenient with respect to noise. This can be useful in cases in which maintaining signal integrity is the top priority. Well-preserved, late electric-era gramophone recordings are examples of such situations.

On the other hand, this property makes the methods perform poorly when noise power is elevated. In such cases, it is perceptually more inconvenient to leave noise than to remove signal content. Acoustic-era recordings provide good examples of this.

4.5 Subjective Evaluation

Although objective audio quality measurements, such as PEAQ, are very useful for obtaining an estimate of audio quality in short time, they are often developed with specific applications in mind [92]. PEAQ, for example, was developed for audio codec evaluation, and was not formally tested for audio restoration. Therefore, the evaluation of this work would not be complete without subjective tests.

Two tests were conducted to compare diffusion based restoration with the benchmark presented in Section 4.4. The first test used artificially degraded recordings, as in the objective tests and preliminary experiments. The second test evaluated the performance of the methods on real historical recordings.

4.5.1 Tests with Artificially Degraded Signals

For the first test, excerpts one to sixteen of the objective test were artificially degraded with 10, 20, 30, and 40dB SNRs. Once again, for each clean excerpt, four noisy versions were created using the same noise sample. The noise samples used in each clean excerpt were extracted from the test split of the gramophone noise

dataset, as with the objective tests.

The 64 resulting noisy excerpts were divided in four panels of 16 excerpts, each with four examples of each SNR. Panel splitting was done in circular fashion, i.e. if an excerpt was corrupted with 40dB SNR in panel one, it would be degraded with 10dB SNR in panel two, 20dB SNR in panel three, and so on. This made sure that all combinations of signal and SNR would be evaluated.

A set of anchor signals was also created by corrupting the clean excerpts with 7dB SNR. The noise samples used to create the anchors were the same used to create the noisy excerpts. The anchor signals represented the worst possible version of each excerpt, and were meant to serve as a reference point for result analysis.

Test details

Volunteers answered sixteen questions, corresponding to the noisy excerpts in one of the four panels. In each question, they were presented with the reference clean excerpt, followed by six signals: the panel's noisy excerpt, a copy of the reference, the excerpt's anchor signal, and restored signals using the three methods seen in Section 4.4. They were then asked to evaluate the quality of the six signals with respect to the reference.

In order to compare all signals in the dynamic range of the reference, the noisy excerpt, the anchor, and the benchmark restored version had their amplitudes multiplied by two. This was not necessary for the diffusion restored versions. All signals were checked for saturation after this procedure, and the same volume was used when reproducing all excerpts.

Before the test, each volunteer was presented with an introductory text explaining that quality was to be evaluated in two aspects: presence of noise, and integrity of the original signal. To illustrate each of them, a reference (clean prelude 7) was presented along with two degraded versions, one corrupted with gramophone noise at 10dB SNR, and one low-passed at 3kHz.

Grading was done using a $0 - 100$ continuous scale where only the edges of the scale were marked. Figure 4.24 shows an image of the interface used for the test's questions, which was created using [93].

The order of the six signals evaluated in each question was random, in order to avoid bias from the volunteers, and to remove ordering effects from the test. However, the sixteen excerpts were shown to the volunteers in the same order, across panels, and according to their IDs. In other words, question one always corresponded to excerpt number one, question number two to excerpt number two, and so on.

The test had around 25 minutes of audio, but average completion time was around 50 minutes, likely due to the small differences between restored and clean

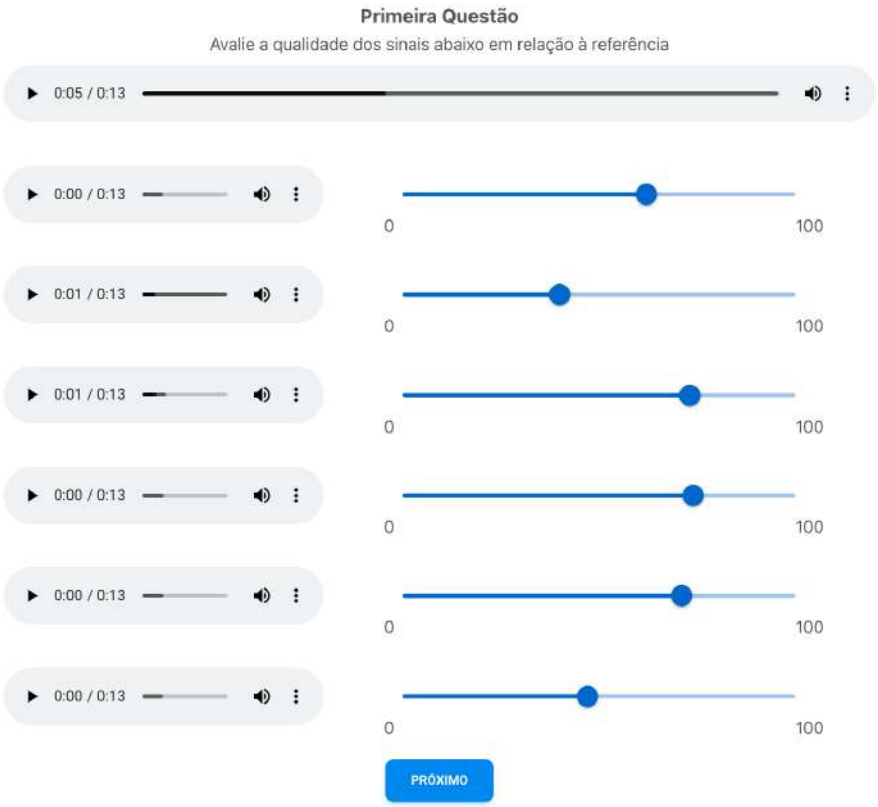

Figure 4.24: Interface of the first subjective test. The order of the signals was random for each question.

versions of the excerpts with higher SNR, and due to the need of listening to the excerpts several times. The audio samples were presented to the volunteers in the acoustically isolated listening room of the Signals, Multimedia, and Telecommunications (SMT) laboratory at UFRJ, using Sennheiser HD 265 linear headphones. There were no audio monitors, and volume level was set to a fixed value (tested informally) for all excerpts. Volunteers were instructed not to change the volume level.

In total, there were 20 volunteers for this test, 5 for each panel. None of them reported any kind of hearing condition or loss, except for one, which had self-diagnosed light tinnitus. Volunteers were mostly amateur musicians, audiophiles, or people who worked with audio signal processing before. Their experience with music is reported in the test's results file, which is available in this work's Github repository. However, although all volunteers had some musical expertise, their level of experience with music varied. With this setup, each combination of signal and SNR was evaluated five times.

The goal of the first test was to evaluate average performance of the three methods across pieces of different styles, with different recording environments, and with

different noise levels. However, since all combinations of signal and SNR were evaluated, this test also allowed to study the performance variation of the three methods as noise increased. Furthermore, as the questions included a hidden reference and an anchor, there were maximum and minimum reference values for each excerpt, and the sensibility of the methods to particular signals could also be studied.

Another interesting by-product of this test was creating a dataset useful for studying the accuracy of PEAQ outside of the audio codec application domain. Because the same excerpts were evaluated for both objective and subjective tests, it is possible to compare the two of them to estimate the accuracy of the PEAQ's ODGs. This is outside of the scope of this work, and was not greatly explored here, but is an interesting future contribution nevertheless.

Results

The results were evaluated using two different metrics. The first was simply the grade given by the volunteers for each of the test signals, while the second was a measure of the improvement in the restored versions, similar to the ΔODG metric that was used in the objective tests.

The advantage of using the subjective grades (SGs) directly is that they have a simple interpretation. A better grade should translate to an audio signal with better quality. Also, the hidden reference and anchor ratings provide reference points for evaluating the restoration methods.

However, when using the grades directly, one must be mindful that since the noisy signals and anchors were created using SNRs, the perceived quality of those signals fluctuates between questions. This effect increases the spreads of the SGs distributions, which are already considerable due to the differences in the grading criteria of the volunteers.

The improvement in a restored version of a noisy excerpt can be measured as

$$\Delta\text{SDG} = (G_{\text{ref}} - G_{\text{noisy}}) - (G_{\text{ref}} - G_{\text{test}}) = G_{\text{test}} - G_{\text{noisy}}, \tag{4.11}$$

where SDG stands for subjective difference grade, and where G_{ref}, G_{noisy}, and G_{test} represent the scores assigned by one volunteer to the hidden reference, the noisy signal, and the restored signal, respectively. Intuitively, ΔSDG measures how much the distance between the score of the hidden reference and the noisy signal was reduced by the restoration procedure.

The advantage of using ΔSDG is that it is easier to compare it to the ΔODG used in the objective tests. However, it is not an absolute measure of quality. An improvement of 10 score points for a signal which is slightly degraded could be harder to achieve than an improvement of 50 score points for a strongly degraded

signal.

Table 4.19 shows the mean SG and ΔSDG for each of the three restoration methods. Once again, the confidence intervals were set using Gaussian distributions and a 95% degree of confidence.

Table 4.19: Test mean SG and ΔSDGs for the restoration methods. The relative difference between the methods decreased for the two subjective metrics.

	Mean SG	Mean ΔSDG
Benchmark	82.468 ± 2.233	40.990 ± 2.705
Diffusion + estimated gain	72.178 ± 2.534	30.700 ± 2.445
Diffusion + known gain	69.287 ± 2.598	27.809 ± 2.277

Overall, the benchmark method had once again better results, albeit with a smaller relative difference[15] with respect to the diffusion methods in the improvement metric. Before, the mean ΔODG relative difference between the benchmark and the best diffusion method was of 34%. In the subjective tests this was reduced to 25% for ΔSDG and to 12% for the direct scores.

Figure 4.25 shows the boxplots of the direct scores of the three methods. In this plot, the scores for the hidden references and anchors are also provided for comparison. Figure 4.26 shows the boxplots for the ΔSDGs of the three methods. In both figures, the whiskers cover samples that lie within $1.5 \times$ IQR of the top and bottom quartiles. The small circles represent outlier points.

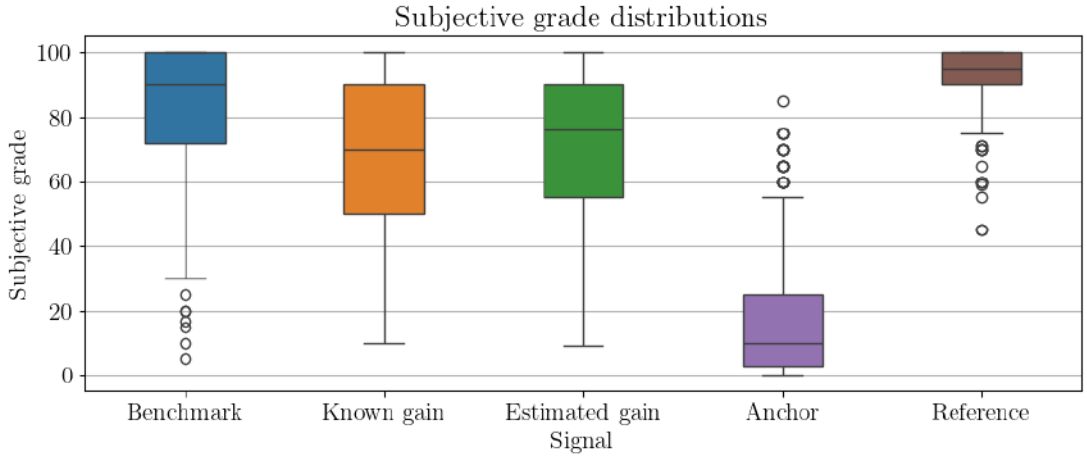

Figure 4.25: Boxplots of the subjective test scores. Once again, the benchmark method was superior.

In both figures, it is possible to see that the benchmark method was superior to the diffusion based methods. However, once again it is necessary to consider that

[15]Defined as the absolute metric difference between the benchmark and the target method divided by the benchmark metric.

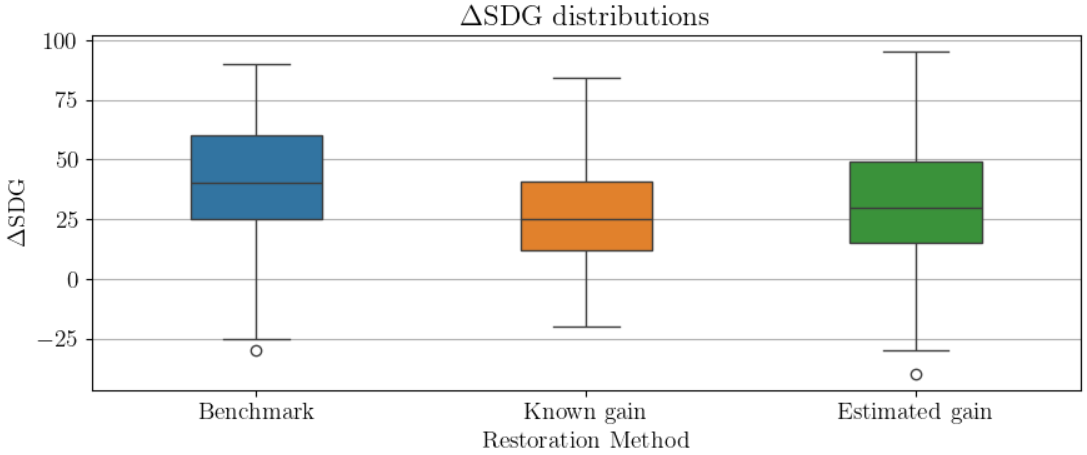

Figure 4.26: Boxplots of the ΔSDG distributions of the three restoration methods. The ΔSDG distributions were narrower than their ΔODG counterparts.

optimal ξ' seems to be dependent on absolute noise power. The poorer results and slightly larger width of the boxplots for the diffusion methods could be a result of sub-optimal ξ'.

The diffusion methods presented similar results. This time, diffusion restoration with estimated gain had slightly better results than restoration using the correct gain value. This is another evidence that ξ' values could have been sub-optimal; the estimated gains could have compensated the chosen values of ξ' to some extent.

The big spread of the distributions and the number of outliers in Figure 4.25 were expected for a test with a large pool of volunteers. Still, it is worth noting that the higher anchor grades were given mostly for 40dB SNR signals, and that the lower reference grades were given mostly by volunteers with less musical experience. The level of musical experience of each volunteer was included in the results file.

In Figure 4.25, the lower whiskers on the three restoration methods correspond to 10dB SNR test signals, in general. For these excerpts, the benchmark method removed too much content of the original signal, while the diffusion methods were unable to remove the noise completely.

The higher whiskers on the subjective grade boxplots correspond mostly to 30dB and 40dB SNR signals. In these cases, residual noise was less bothersome for the diffusion methods, and the benchmark method removed less original signal content.

For the ΔSDG distributions, the higher whiskers represent lower SNR excerpts. The noisy version at lower SNRs had very low starting scores, which caused the quality improvement after treatment to be large, even if the resulting score was still not close to the hidden reference grades.

Similarly, the bottom whiskers on the same figure correspond to higher SNR excerpts. The starting quality of these test signals was already considerably high, making improvement harder than it was for lower SNR signals. For some 40dB SNR

signals, ΔSDG was negative, meaning that the treatments worsened signal quality.

It is important to remember that volunteers were asked to evaluate both noise removal and original signal content loss. Because of this, the subjective test was more tolerant to residual noise presence than the objective test. In the latter, PEAQ penalized residual noise presence more strongly than content loss.

This is particularly visible in Figure 4.27, which shows the histogram of the ODGs of excerpts one to sixteen considering 30 and 40dB SNR on the left, and the histogram of $\mathrm{SDG}_{\mathrm{noisy,reference}} = G_{\mathrm{noisy}} - G_{\mathrm{ref}}$[16] for the same noisy signals on the right. Although the scales are different, it is possible to notice that PEAQ grades for the noisy signals concentrate much more on the lower scale range.

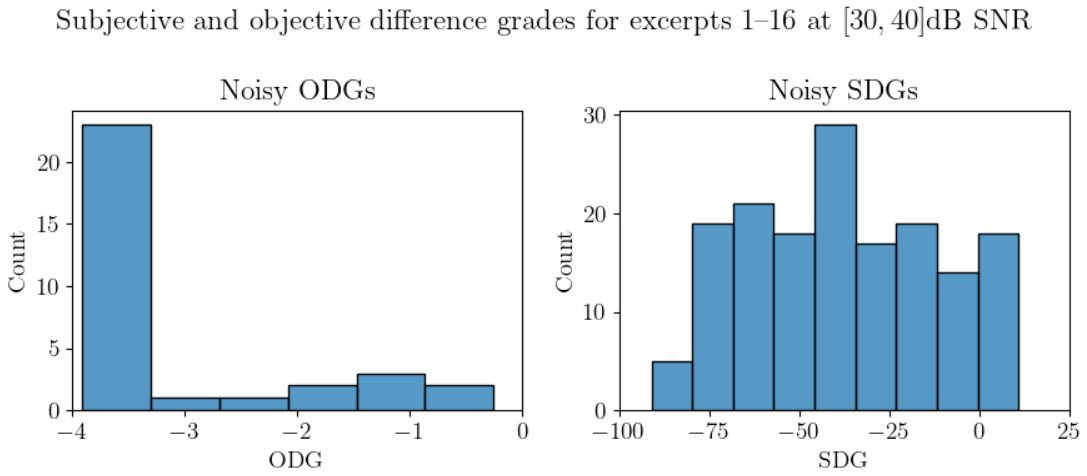

Figure 4.27: Histograms of objective and subjective difference grades for noisy test excerpts at 30 and 40dB SNR. The number of bins in each graph was set to the the smallest integer larger than the square root of the number of entries. As the subjective test has more evaluations per stimulus, its graph also has more bins.

Another way of seeing PEAQ's strictness with respect to residual noise is observing that, for the diffusion methods, the distributions for ΔSDG are narrower than they were for ΔODG. As mentioned earlier, the diffusion methods had a tendency to preserve signal content at the cost of leaving some residual noise; this made the objective evaluation of some signals considerably worse than their subjective evaluation.

Figures 4.28 and 4.29 show the boxplots for the SGs and ΔSDGs divided per SNR. Whiskers and outliers are drawn using the same criteria as before. Considering the SNR segmentation, this time the boxplots for the noisy excerpts were also included in the SG.

In Figure 4.28, the scores for the noisy signals increase along with the SNR. But

[16]The subjective equivalent of the noisy ODG. It will range from −100 to 100, but positive values will be rare, as they indicate that the noisy stimulus was perceived as better than the reference.

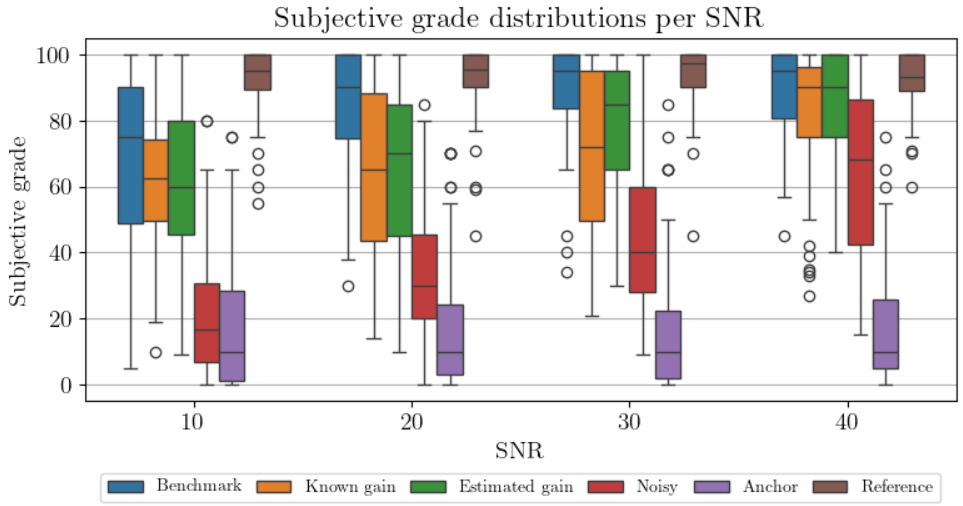

Figure 4.28: Boxplots of the subjective test scores per SNR.

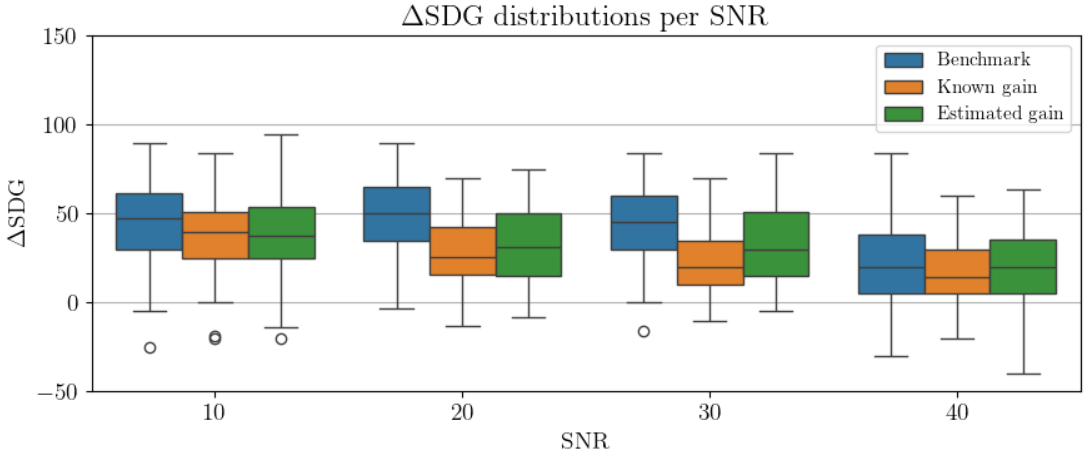

Figure 4.29: Boxplots of the ΔSDG distributions of the three restoration methods.

so does their variance; each volunteer's individual tolerance to the presence of noise plays a big role in the scoring of pieces lightly degraded.

As in the objective tests, the performance of diffusion inspired methods gets better as the SNR increases. It is possible to see that the median score for them gradually approaches that of the benchmark method. The diffusion models apparently have their performance tied to the absolute noise power in the signal being denoised; since all test signals were evaluated with all SNRs, this increase in scores was expected.

The fact that the noisy signal grades increased significantly with the SNR made the ΔSDG plot different from the ΔODG plot. For higher SNRs, it is more difficult to obtain large improvements, and so the median ΔSDGs for all methods falls slightly for 30 and 40dB SNR. This effect makes it harder to see the improvement of the

diffusion models, but it is possible to see in Figure 4.29 that the diffusion models approach the benchmark model for 40dB.

To account for the dependence on absolute noise power for the diffusion methods, scores and ΔSDGs were also evaluated for individual tracks. Figures 4.30 and 4.31 show the plots of the median values of the two metrics *versus* SNR for the individual tracks of the subjective test.

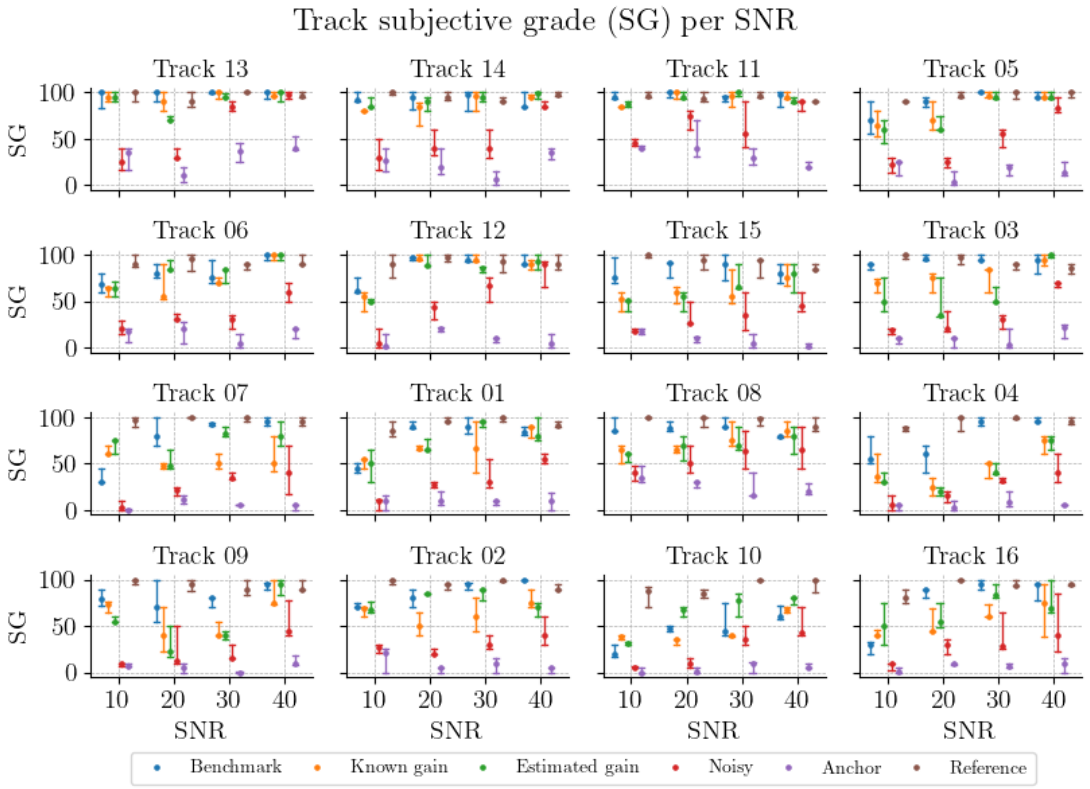

Figure 4.30: Track scores per SNR. The excerpts are ordered according to increasing noise power at 30dB SNR.

Once again, the excerpts are ordered from top-left to bottom-right according to increasing noise power at 30dB SNR. In the two figures, the error bars for each point are drawn discarding the highest and lowest values for each signal. Since there were five evaluations for each signal, the error bars represent quantiles 20% to 80%.

Looking at Figure 4.30, a few of the same tendencies of Figure 4.23 can be observed. Starting from the top-left and proceeding to the bottom-right, it is possible to see that the diffusion methods become gradually more distant to the benchmark model. This showcases the method's dependency on the absolute noise power; for lower noise gains the performance of the diffusion methods is comparable to the benchmark method.

It is also possible to look at the progression of the scores for the individual plots. The performance of the diffusion based methods generally increases with the SNRs,

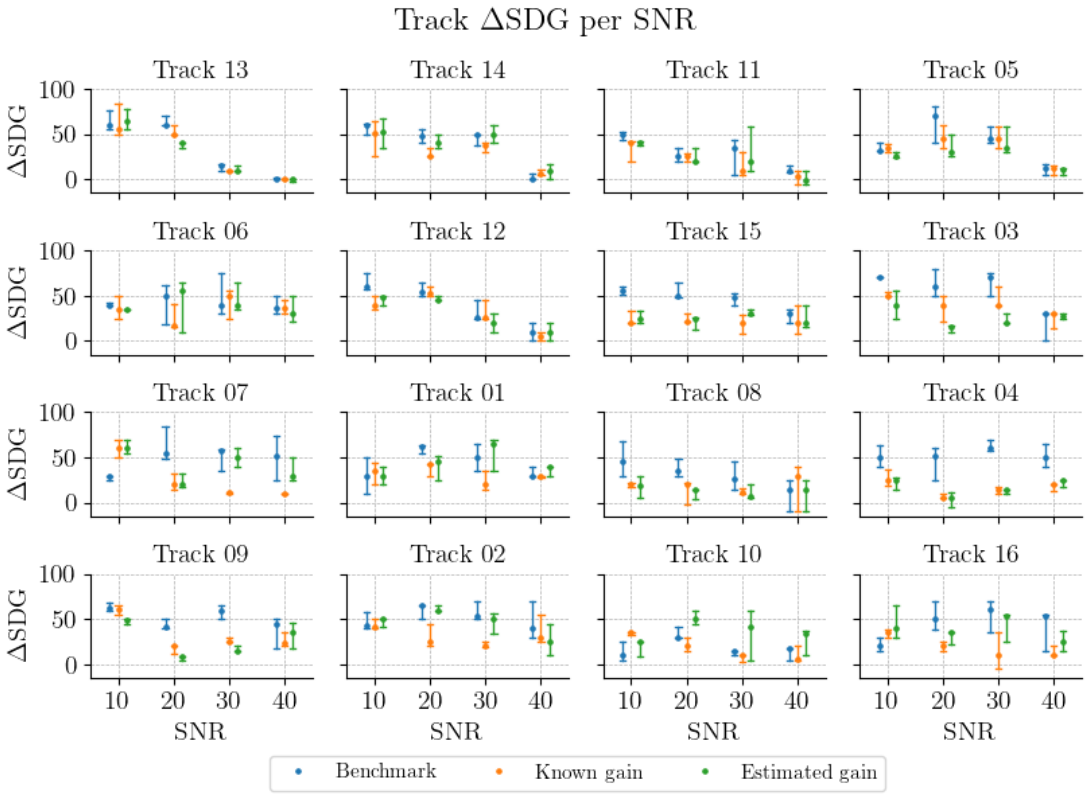

Figure 4.31: Track ΔSDG per SNR. The excerpts are ordered according to increasing noise power at 30dB SNR.

and is comparable to the benchmark for most tracks at 40dB. Considering tracks individually, the correlation between the performance of the diffusion methods and the SNR reinforces the hypothesis of their dependence on the power of the noise in each signal.

The scores for the noisy excerpts increase for higher SNRs. As mentioned earlier, this probably suggests that subjective evaluation was in general more tolerant to some noise presence than objective evaluation using PEAQ. This also explains why the ΔSDG values of Figure 4.31 show a slight downward tendency as the SNR increases. For initially higher scores, large improvements are more difficult to achieve.

Once again, it is interesting to observe the examples of tracks three and four. Their silent parts made any residual noise particularly evident, which explains the worse performance of the diffusion methods for these two tracks. This was expected, and could also be seen in Figure 4.23.

Figure 4.31 endorses the observations made for Figure 4.30 to some extent. Observing the plots in order, it is possible to see that the benchmark method gradually obtains the upper hand as noise power increases. Looking the individual plots, the results appear worse for 30 and 40dB SNRs because of the increase in the scores of the noisy excerpts, despite the performance gain for the diffusion methods at higher

SNRs.

Comparing Figure 4.31 to Figure 4.23, the difference between the benchmark and the diffusion methods became smaller in the subjective test. Although the benchmark method was still better overall, there was considerable overlap between some distributions of the three methods.

A two-tailed Student's t-test with 95% confidence interval was applied to the SG results for the benchmark and diffusion restoration with estimated gains on the individual track distributions, considering all SNRs. For tracks six, seven, eleven, fourteen, and sixteen, it was not possible to reject the hypothesis that the results for the two methods were equal (p-values of $0.21, 0.37, 0.20, 0.55$ and 0.44, respectively).

Additionally, applying the same test considering only 10dB SNR, the SGs for the diffusion (with estimated gain) and benchmark methods were only statistically distinguishable for tracks five, seven, eight, eleven, and fifteen (p-values $0.03, 0.01, 0.01, 0.0003$, and 0.008, respectively). Perhaps for very low SNRs, the amount of signal content removed by the benchmark method was perceptually as bothersome as the residual noise left by the diffusion methods. This could have caused volunteers to give similar scores to the three methods in some cases, despite of the difference between the artifacts created by the two methods.

Figures 4.32 to 4.34 illustrate this point. They show the STFTs of the clean, restored with benchmark method, and restored with diffusion and estimated gain versions of excerpt number one.

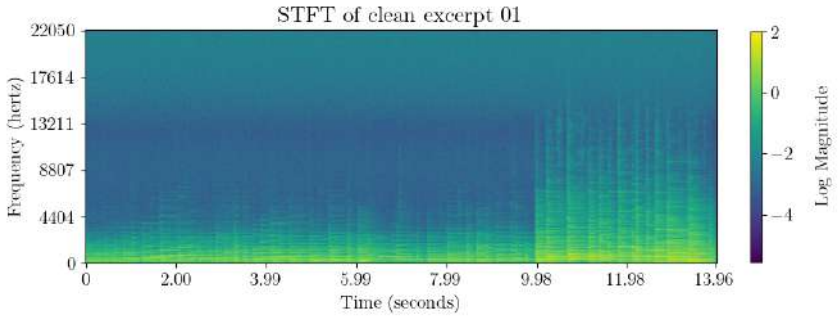

Figure 4.32: Spectrogram of the clean version of excerpt 01.

Comparing Figures 4.32 and 4.34, it is possible to observe that diffusion restoration leaves considerable residual noise. On the other hand, looking at Figure 4.33, it is also clear that the benchmark method removed original signal content in the higher frequencies.

In general terms, the subjective tests endorse the results obtained in the objective tests. Diffusion based methods tend to be more lenient with respect to noise removal, but also seem to preserve original signal content better. This property could be useful in situations where the signal need to be preserved as much as possible.

The performance of diffusion based methods appears to be somewhat dependent

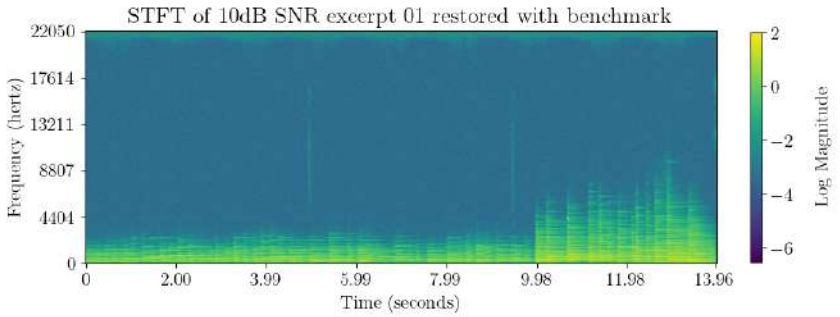

Figure 4.33: Spectrogram of the benchmark restored version of 10dB SNR excerpt 01.

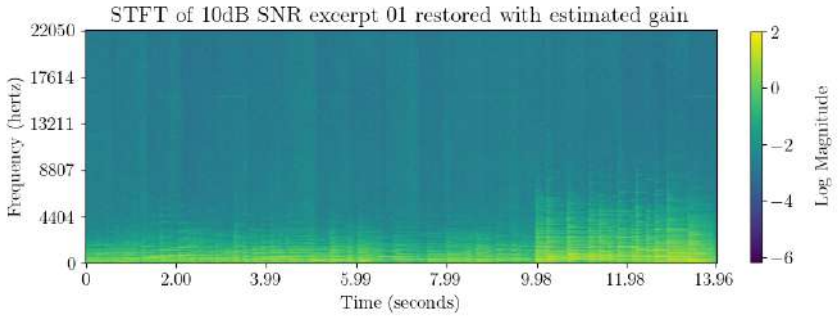

Figure 4.34: Spectrogram of the estimated gain diffusion restored version of 10dB SNR excerpt one.

on the quantity of noise in the signal being restored. The best results were obtained for tracks corrupted with lower noise power, as was possible to see in Figure 4.30. In addition, similar results for the estimated gain and known gain diffusion methods suggest that the heuristic for choosing ξ' could be improved.

In a practical setting, considering that the estimated gain can compensate for sub-optimal values of ξ' to some extent, perhaps it would be a good idea to fine-tune the value of ξ' to the piece being restored. The values of Equation 4.10 could be used as a reference, and multiple inferences could be made in order to obtain a satisfactory result.

Overall, the subjective scores for the benchmark method were superior (as per Figure 4.25), but the diffusion methods showed competitive results in some cases. This is positive considering that diffusion restoration is zero-shot, and does not require specialized training for 78 RPM noise. However, it should also be taken into account that diffusion models are usually larger, and that generative training could be computationally expensive.

4.5.2 Tests with Historical Recordings

For the second subjective test, excerpts of real historical recordings were extracted from six different musical pieces. Five of them were taken from the companion

CD to the book "The Art of the Piano: Its Performers, Literature, and Recordings Revised" [94], and one was taken from Friedrich Gulda's Complete Decca Recordings [95] collection.

The book "The Art of the Piano" is a piano encyclopedia that covers the repertoire of several major piano composers, and also details some of the most famous recordings of their pieces. The companion CD to the book contains rare recordings from 1916 up to 1976, most of which are contaminated by 78 RPM noise. Friedrich Gulda was a famous jazz and classical pianist, and the collection of his recordings by Decca includes pieces recorded between 1949 and 1971.

The five pieces taken from "The Art of the Piano" for the subjective test were Eugene D'Albert's 1920 recording of Franz Lizst's "Au Bord d'une Source" (excerpt 49), Arthur De Greef's 1929 recording of Edvard Grieg's Op. 12 No. 1 (excerpt 50), Walter Gieseking's 1938 recording of Claude Debussy's "Mouvement", from the first book of "Images" (excerpt 51), Bela Bartok's 1945 recording of his own piece "Evening in Transylvania" (excerpt 52), and Ignaz Friedman's 1930 recording of Felix Mendelssohn's Op. 102 No. 5 (excerpt 53). The sixth test track (excerpt 54) was extracted from Friedrich Gulda's 1949 recording of the first movement of Ludwig van Beethoven's Sonata No. 31. All excerpts were 23s long, and were cut minding the padding requirements of the benchmark model. On the CDs from which the tracks were extracted, there is no explicit mention to the application of noise removal techniques on these signals.

The choice of recordings was made to cover important technological innovations in recording techniques, and listening to the excerpts it is possible to see that the background noise gradually diminishes for more recent excerpts. Avoiding as much as possible the occurrence of non-additive defects, such as hard-clipping, was also a selection criterion.

Excerpts 49, 51, and 54 had alternative versions in the MAESTRO dataset. However, as before, the performance of the diffusion methods for these excerpts was not very different from the one obtained for the other tracks. This will be visible while analyzing the results for the tracks individually later on.

Test details

Volunteers answered six questions, one per excerpt. In each question, they were presented with the original historical recording (reference), followed by three restored signals. This blind setup, also with six test signals, was the same as that used in [31]. Among the restored signals, one was treated using the benchmark method while the other two were restored with diffusion based denoising. One of the diffusion restored signals was treated using fixed $\xi' = 0.35$, while the other was denoised using a customized ξ' value for each test track.

The reason for choosing this test setup was as follows. While preparing the test, it became clear by listening to the results that using $\xi' \geq 1.4$ left too much residual noise in "The Art of the Piano" test tracks. This is exemplified by Figure 4.35, which shows the STFT for excerpt 49 restored using $\xi' = 1.4$. It is clear that no noise was removed, except in a few short segments.

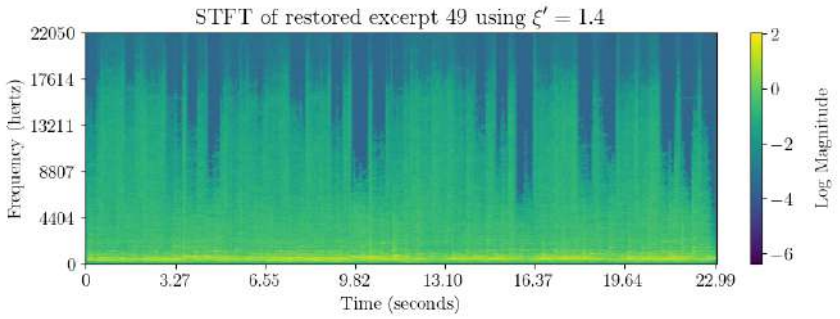

Figure 4.35: Spectrogram of the restored version of excerpt 49 using $\xi' = 1.4$.

On the other hand, $\xi' = 0.35$ was a bit too restrictive for excerpt 54, likely because it was the most recent and least noisy test file. Considering this, it seemed like a good idea to evaluate the method using the best possible ξ' values, i.e. ξ' fine-tuned through listening, for each test track. This approaches a real-life scenario, in which it would be possible to experiment using the four ξ' values that were suggested in Equation (4.10), and then fine tune ξ' by listening to the outputs and subjectively choosing the best of the four ξ' as a starting point.

The fine tuning strategy above was adopted for the restored versions with customized ξ'. After trying different values and listening to the results, $\xi' = 0.30$ was used for test tracks 49 to 53, and $\xi' = 1.4$ was used for excerpt 54.

Although using the best ξ' approach is closer to what would be done in a practical setting, it introduced some bias in the test, as the restored excerpts needed to be examined before being given to the volunteers. Ideally, restored versions using the four heuristic ξ' values of Equation (4.10) would need to be included in the test to avoid this bias.

However $\xi' = 2.45$ and $\xi' = 3.5$ left too much noise in all excerpts, and did not merit being subjectively evaluated. For excerpts 49 to 53, $\xi' = 1.4$ also resulted in outputs with too much residual noise. Since $\xi' = 1.4$ was already the customized ξ' for excerpt 54, including versions restored with $\xi' = 0.35$ in all questions was enough to evaluate diffusion restoration blindly.

To compare all signals in the dynamic range of the original historic recordings, the diffusion restored versions were obtained from a half-amplitude version of the noisy references. Reverting $\mathcal{A}(\cdot)$ with conditional diffusion included removing the $\frac{1}{2}$ gain factor, which meant doubling the amplitude of the signals being restored. Therefore, the references had to be divided by two before applying the diffusion

restoration method.

Volunteers were asked to evaluate the quality of the restored excerpts considering two aspects: presence of noise, and integrity of the reference's musical content. This time, no introductory examples were given, but the test was preceded by an introductory text explaining that an ideal restoration method should remove all noise without cutting signal content or introducing new signal artifacts. There were 13 volunteers in total, 12 of which participated in the first test.

Once again, volunteers were presented to the stimuli in the SMT listening room using Sennheiser 265 linear headphones. While listening to the references to adjust the testing equipment's volume, it was possible to notice that excerpt 52 was recorded at a higher volume than the others. To take this into account, volunteers were instructed to reduce the testing equipment's volume by 20% for this excerpt only. The remaining excerpts were evaluated at the same output volume level.

Grading was done on a continuous scale of $0 - 100$ created using [93], as in the previous test. Only the edges of the scale were labeled. Figure 4.36 shows an image of the interface used for the second test.

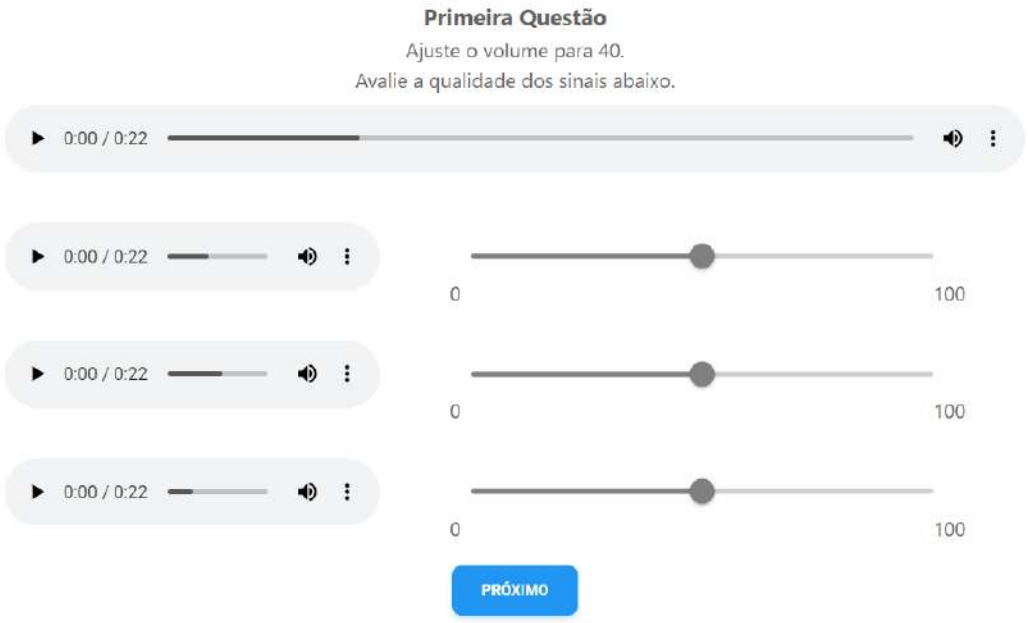

Figure 4.36: Interface of the second subjective test. The order of the signals was random for each question.

The position of the restored signals was random in each question, but volunteers evaluated all excerpts in the same order, which was: excerpt 52, excerpt 50, excerpt 51, excerpt 49, excerpt 55, and excerpt 53.

The test had around 10 minutes of audio, but most candidates took a little over 20 minutes to complete it. Once again, this difference was likely due to the need to listen to the restored versions many times for comparison.

Results

For this test, since clean references were unavailable, the subjective grades alone were used as a quality metric. Table 4.20 shows the mean SGs for the benchmark, fixed $\xi' = 0.35$, and best ξ' restoration methods using 95% confidence intervals set with Gaussian distributions. Excluding the excerpts with alternative versions, the mean SGs for $\xi' = 0.35$ and for the best ξ' change to 72.359 and 80.153.

Table 4.20: Test mean subjective grades. In the test with real historical recordings, the diffusion methods showed better mean scores than the benchmark method.

	Mean SG
Benchmark	65.436 ± 5.257
$\xi' = 0.35$	72.974 ± 3.453
Best ξ'	77.833 ± 3.257

The table shows that the diffusion based methods in this test were, on average, better than the benchmark. This was also verified with a one-tailed, 95% confidence interval, Student's t-test with null hypothesis that the mean benchmark results are greater to or equal than each of the diffusion models. For the best ξ' setup, the null hypothesis was rejected with $p < 0.0001$; for the fixed ξ' setup, the null hypothesis was rejected with $p = 0.001$. If excerpts 49, 51, and 54 are excluded, the best ξ' setup is still significantly better ($p = 0.01$), but the fixed ξ' regime cannot be distinguised from the benchmark.

Figure 4.37 shows the boxplots of the grades of the three methods. It is possible to see that the benchmark method has a lower median value and is more spread than the diffusion methods.

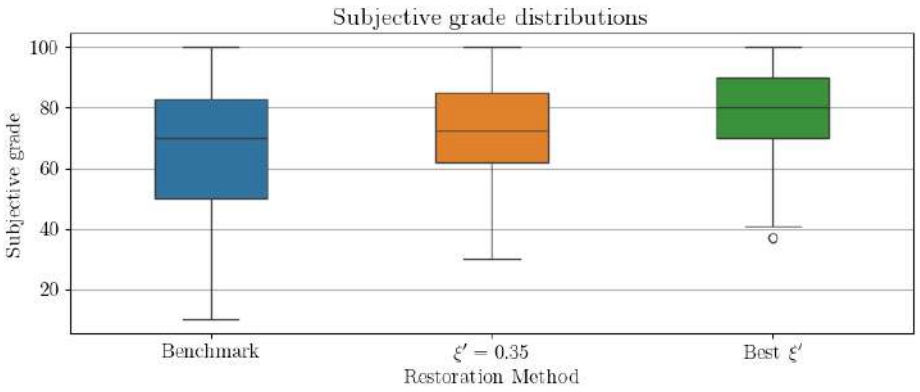

Figure 4.37: Boxplots of the subjective test scores. This time, the diffusion based methods obtained better results.

A few different hypotheses can be made to explain why the benchmark method did not perform as well in the test with real historical recordings. The first is that the benchmark model did not generalize well enough to the realistic setting of restoring

historical recordings. As it was trained on synthetic pairings of clean and degraded signals, it is possible that the excerpts used in this test were out-of-distribution data, to some extent. This idea is supported by the fact that the subjective grades of the diffusion methods did not change as sharply between the two tests.

The second hypothesis is that most of the tracks used in the historical recordings test could actually be in the range of 30 and 40dB SNR, where the benchmark method showed similar performance to diffusion based methods. All test excerpts were extracted from electric-era recordings, and perhaps 10 and 20dB SNRs are more adequate for acoustic-era records.

Finally, the third hypothesis is related to the setups of both tests. The presence of a clean reference in the artificial signal test could have biased the results in favor of the benchmark method, since the baseline to which the volunteers were paying attention did not have any noise whatsoever. In other words, volunteers could have excessively penalized residual noise presence over removal of high-frequency content simply because the former is more noticeable when compared to a clean reference.

Additionally, eventual residual noise in the historical recordings test could have been mistaken for high-frequency content in the signal. Very low amounts of background noise can make some restored excerpts sound brighter than the others [1]. As there were no clean references in the second test, this brightness might have been considered musical content, and not noise.

Since the benchmark method favours noise removal over content preservation, and since the diffusion models have a tendency to leave some residual noise while preserving most of the original signal's content, it makes sense to assume that the presence of a clean reference could have affected the results to some extent.

The subjective grade distribution for the individual tracks of the second test can be seen in Figure 4.38. It is possible to see that the distributions for the diffusion methods excerpts 49, 51, and 54 were very similar to the others, reinforcing that their presence in MAESTRO did not incur in overfitting for the diffusion techniques.

Looking at the individual results, it is possible to see that the benchmark method had particularly bad results for excerpts 49 and 54. In the case of excerpt 54, this could have been caused by a strong motor rumble noise present in addition to the typical crackles of gramophone noise. None of the three methods was able to completely remove the motor rumble, but the benchmark also failed to remove the other noise components in the start and end of the track.

Excerpt 54 starts in the lower piano range, which might have made it difficult for the benchmark method to distinguish between musical and noise content when combined with the low-frequency rumble. In a sense, the rumble could have masked certain parts of the other noise content for the benchmark method. As the diffusion methods inverted a generic gramophone noise, they were able to avoid this problem.

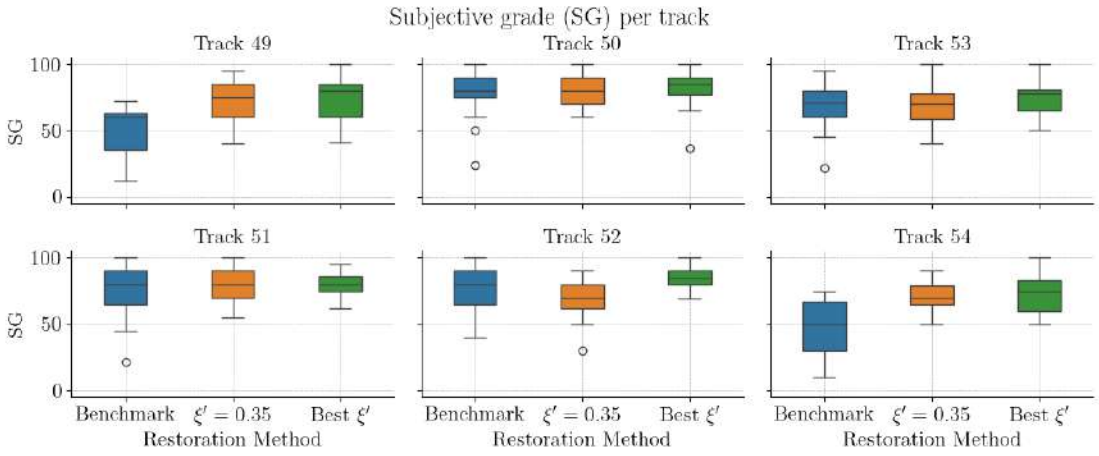

Figure 4.38: Historical recordings individual track scores. The diffusion methods are more consistent across the excerpts.

In the case of excerpt 49, a possible explanation is that the original recording itself is originally very much band-limited. While removing the background noise, the benchmark method also removes the little high-frequency content that was originally present in the recording, making it sound particularly muffled.

In the remaining excerpts, the performance of the benchmark and diffusion methods was similar. Listening to the examples, it is clear that preferring one over the other is a matter of choosing between leaving some residual noise or removing some parts of the original signal's content.

An advantage of the diffusion methods, however, is the possibility to customize the value of ξ' according to the piece being restored. Although the results for $\xi' = 0.35$ were also superior to the benchmark in most tracks, the customized ξ' strategy was the one that showed the best scores. This is a flexibility that is not available to the benchmark model.

Considering the results of the previous tests, and the results of the test with historical recordings, it is possible to say that diffusion based restoration provides an alternative to current state-of-the-art methods for removing perceptually distributed noise. Despite being slower, diffusion based restoration is interesting because of its zero-shot nature, and its relative controllability with respect to current neural network methods for this task.

Although the diffusion methods suggested in this work were only applied to gramophone noise removal, they could be used in theory to any type of additive degradation model. The approach presented in this work is only dependent on having an inference noise set that contains samples similar to the one in the musical piece being restored. In the most general case, silent parts of a track being restored could be used to enhance the inference set and try to obtain better restoration

results.

On the other hand, diffusion methods seem to be more lenient with respect to the presence of noise, as the ensemble of objective and subjective evaluations with artificial signals seem to indicate. Therefore, their use is most recommended for situations in which the priority is preserving most of the signal content, and in which noise presence is not excessively annoying from the start. Well-preserved electric-era recordings, such as the ones that were used in the subjective test with historical recordings, are good examples of such cases.

Chapter 5

Conclusions and Future Work

This dissertation investigated the use of diffusion models for removing perceptually distributed noise from historical recordings through a case study with classical solo piano music. A summary of the contributions of this work and possible directions of future work are provided below.

5.1 Conclusions

Despite the current surge in works studying the use of generative models for different tasks in the audio domain, the use of diffusion models for musical audio restoration is still a relatively understudied topic. Although there are promising results using diffusion for solving general inverse problems in the audio domain, they commonly require specific knowledge about the degradation being restored.

Additionally, general background noise removal is a subtopic of musical audio restoration that has not been studied in previous works using diffusion. Current state-of-the-art approaches for removing perceptually distributed noise from historical recordings rely on traditional supervised learning with neural networks, and require datasets of pairs of clean and degraded recordings.

This supervised learning strategy has two important drawbacks in the context of musical audio restoration. First, it makes restoration models specialized on the defects they were presented to during training, which creates the need of training separate models for different degradations. Second, the lack of clean reference data for historical recordings forces the use of synthetic examples while training, which can hinder restoration performance with real historical recordings.

Considering this context, this work provided a solution that could be used for general background noise removal using generative diffusion models. The idea is training a model capable of generating a plausible musical piece from scratch (ideally), and then using it to generate clean versions of a noisy recording through conditioning techniques. This generative approach is zero-shot, meaning that it can

be used to restore different types of degradations without needing new noise data, as long as conditioning is properly adapted.

Although the method does not depend on a specific noise type, this dissertation focused on gramophone noise removal in historical classical solo piano recordings. Gramophone noise is an interesting object of study because it is a combination of several different additive defects, such as clicks, crackles, motor rumble, and thumps. Furthermore, it is random and non-stationary, which made adapting existing diffusion restoration techniques for this problem a challenging task.

This work provided a comprehensive review of diffusion models, conditioning techniques, and useful heuristics for unconditional and conditional generation in Chapter 3. The conditional generation section was of particular interest, as it explained two methods — data consistency (DC), and reconstruction guidance (RG) — for creating a clean audio sample conditioned on a degraded version of it.

Both conditional generation techniques originally relied on a deterministic model $\mathcal{A}(\cdot)$ of the degradation, and so Chapter 4 started by introducing a method to use the DC and RG techniques with a random model for additive $\mathcal{A}(\cdot)$. It consisted of drawing a random noise sample from a set of possible examples, normalizing it, and adding it to the intermediate output of reverse diffusion with a pre-defined gain. Ideally, over the course of the reverse diffusion process, $\mathcal{A}(\cdot)$ would emulate the corruption of a clean sample with noise drawn from a specific distribution, at a fixed SNR.

This technique was evaluated on the problem of gramophone noise removal, and a study of different design choices for conditional generation using random $\mathcal{A}(\cdot)$ came next. First, it was shown that the DC conditional generation method was not suited for removing additive defects with a random degradation model. Then, the importance of the RG parameter ξ' was explored, and it was shown that it was directly correlated to restoration performance.

A study of different heuristics to determine which pre-defined gain G to use on $\mathcal{A}(\cdot)$ followed. Some of them relied on estimating a range of possible SNRs for $\mathcal{A}(\cdot)$ based on the noisy signal and sampling G according to a random SNR within the range. Others tried to directly estimate G by computing the evolution of the noisy signal's power with a sliding window, and using the square root of the minimum value as an estimate. Some proposals on techniques to estimate optimal ξ' values were also evaluated.

The study seemed to suggest that the optimal value of ξ' was not dependent on the noisy signal's SNR, but on the absolute power of the noise component in the degraded recording. As $\mathcal{A}(\cdot)$ tries to emulate the degradation, this meant that the choices of ξ' and G were coupled; an excessively large estimate of G could be compensated by a more restrictive choice of ξ', and vice-versa.

As the proposals for estimating ξ' showed poor results, in the end a validation round was set up to compare the different heuristics for estimating G, and a rule of thumb approach was adopted for determining ξ' based on the SNR, knowing that it was a sub-optimal approach. The sliding window heuristic for determining G yielded the best results, and so it was selected for a batch of objective and subjective tests meant to compare diffusion based restoration with a state-of-the-art neural network method for gramophone noise removal.

Two types of tests were conducted. In the first, artificially degraded recordings at four SNR levels were restored and compared objectively and subjectively to the benchmark method of [31]. In the second, real historical solo piano recordings were restored by diffusion methods and by the benchmark method, and evaluated subjectively.

Although this was not the objective of this work, it is important to stress that that the pairs of objective and subjective evaluations of the first test could be an important data source for future development and evaluation of objective audio quality measurement algorithms. Therefore, the test results alone are a significant by-product of this dissertation.

Listening to the restored test signals, it became clear that the benchmark method and the diffusion strategies had complementary properties. The benchmark model removed most of the noise, even at very low SNRs, but muffled the high-frequency content of the original signals more agressively. Diffusion based restoration, on the other hand, was better at maintaining the sharpness of the original signals, at the cost of leaving residual noise (non-uniformly).

This fact led the objective results of the first test to be better for the benchmark model. The objective quality metric that was used, PEAQ, penalized more heavily the presence of noise and other artifacts than the loss of spectral components during the restoration process. However, for signals lightly degraded, the difference in the results of the benchmark and diffusion methods was not so stark.

Subjective evaluation with the artificially degraded signals confirmed that part of the advantage of the benchmark indicated by PEAQ was indeed caused by its partiality towards complete noise removal, at the cost of content loss. For higher corruption levels the results of diffusion methods and the benchmark model were closer and, despite sounding very different, diffusion methods had comparable results to the benchmark in some cases.

In the test using historical recordings, diffusion based restoration showed a slight advantage. A few hypothesis were made to explain the difference in results in the first and second evaluations. The first was that the neural network model did not generalize sufficiently well to the case of real recordings. The second was that most electrical era recordings — which were the ones used in the second test — had SNRs

closer to the higher ones in the first test. Finally, the third hypothesis was related to the test setups: the presence of clean references in the first test could have biased volunteers to choose total noise removal over signal preservation, while the lack of clean references in the second test could have made volunteers more tolerant towards residual noise.

Considering the ensemble of the results, a reasonable conclusion is that diffusion based restoration techniques could offer a viable, zero-shot, alternative for general background noise removal, albeit slower, and with different properties when compared to traditional neural network methods. Further testing is necessary to see if the results stand for other types of additive noise and music, but this work showed the viability of diffusion inspired restoration as a research topic. Diffusion restoration is more adapted to lightly degraded musical pieces, and cases where preserving signal integrity is crucial. Well-preserved electrical era recordings are great examples of such cases.

5.2 Future Work

Immediate future contributions include taking advantage of the zero-shot nature of diffusion based restoration for removing other additive random defects from historical recordings. Hiss removal, for example, is a good candidate as traditional signal processing methods provide a solid benchmark for comparison.

Another way to profit from the zero-shot characteristic of diffusion restoration is to try to use specialized inference noise sets, crafted from noise-only parts of the musical recording being restored. In the tests of this dissertation, the diffusion restoration methods drew samples for $\mathcal{A}(\cdot)$ from an inference set crafted from noise samples from other recordings. In theory, if noise samples from the piece being restored are included in the inference set, restoration performance should improve. Therefore, exploring ways to craft inference sets specific to certain pieces is a valuable investigation direction.

Further parameter exploration is also a possible line of work. Inference with diffusion models is notoriously slow because of the sequential nature of reverse diffusion, and it would be useful to explore how much the number of steps could be reduced while still being able to restore the degraded signals. This would probably mean also investigating the impacts of σ_{min}, σ_{max}, ρ, S_{min}, S_{max}, $S_{\mathbf{z}}$ and S_{total} more in-depth.

Investigating latent diffusion models is a particularly promising line of research as well. In latent diffusion, an autoencoder is used to map the input to a latent space of lower dimensionality, in which diffusion can be performed much faster. This approach is currently used in commercial image generators and editors [12–14],

and improved generated image quality besides reducing inference time. However, denoising with latent diffusion would require investigating how to adapt the gradient calculation of RG to the latent space.

In this work the unconditional diffusion model that was used was specialized in generating solo piano samples. This was mainly due to the availability of a large dataset of high-quality piano recordings (MAESTRO), and investigating the use of diffusion models for restoration of recordings with different (or multiple) instruments would also be a valuable contribution. Additionally, it would also open up the path for studying the impact of the dataset used for unconditional training in conditional generation.

References

[1] BISCAINHO, L. W. P. *Restauração Digital de Sinais de Áudio Provenientes de Gravações Musicais Degradadas*. PhD Thesis, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil, 2000.

[2] DEUTSCHE GRAMOPHON. "The Shellac Project". 2018. Disponível em: <https://artsandculture.google.com/story/OAXRla_BJlVMJg>.

[3] ROTH, E. "Is It Live ... or Yamaha? Channeling Glenn Gould", *New York Times*, March 2007. Disponível em: <https://www.nytimes.com/2007/03/12/arts/music/12conn.html>.

[4] STEINWAY & SONS. "Steinway & Sons Spirio: The World's Finest Resolution Player Piano". 2015. Accessed on 2024-05-03.

[5] HO, J., JAIN, A., ABBEEL, P. "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6840 – 6851, Montreal, Canada, December 2020. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf>.

[6] SONG, Y., ERMON, S. "Generative Modelling by Estimating Gradients of the Data Distribution". In: *Advances in Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, December 2019. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2019/file/3001ef257407d5a371a96dcd947c7d93-Paper.pdf>.

[7] SONG, Y., SOHL-DICKSTEIN, J., KINGMA, D., et al. "Score-Based Generative Modeling through Stochastic Differential Equations". In: *International Conference on Learning Representations (ICLR)*, Vienna, Austria, May 2021. Disponível em: <https://openreview.net/forum?id=PxTIG12RRHS>.

[8] GODSILL, S. J., RAYNER, P. J. *Digital Audio Restoration*. Cambridge, United Kingdom, Springer, 1998.

[9] VAHDAT, A., KREIS, K., KAUTZ, J. "Score-based Generative Modeling in Latent Space". In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 11287 – 11302, December 2021.

[10] ROMBACH, R., BLATTMANN, A., LORENZ, D., et al. "High-Resolution Image Synthesis With Latent Diffusion Models". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684 – 10695, New Orleans, Louisiana, USA, June 2022.

[11] HO, J., SALIMANS, T., GRITSENKO, A., et al. "Video Diffusion Models". In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8633 – 8646, New Orleans, Louisiana, USA, 2022.

[12] MIDJOURNEY INC. "Midjourney". 2022. Disponível em: <`https://www.midjourney.com/showcase`>.

[13] OPENAI. "DALL-E 3". 2023. Disponível em: <`https://openai.com/dall-e-3`>.

[14] STABILITY AI. "Stable Diffusion 3". 2024. Disponível em: <`https://stability.ai/news/stable-diffusion-3`>.

[15] MENG, C., HE, Y., SONG, Y., et al. "SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations". In: *International Conference on Learning Representations (ICLR)*, April 2022.

[16] LUGMAYR, A., DANELLJAN, M., ROMERO, A., et al. "RePaint: Inpainting Using Denoising Diffusion Probabilistic Models". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11461– 11471, New Orleans, Louisiana, USA, June 2022.

[17] MOLINER, E., LEHTINEN, J., VÄLIMÄKI, V. "Solving Audio Inverse Problems with a Diffusion Model". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Rhodes Island, Greece, June 2023. Disponível em: <`https://ieeexplore.ieee.org/document/10095637`>.

[18] WIENER, N. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series: With Engineering Applications*. 1st ed. Cambridge, United States, MIT Press, 1949.

[19] GOODFELLOW, I., BENGIO, Y., COURVILLE, A. *Deep Learning*. Montreal, Canada, MIT Press, 2016.

[20] HAYES, M. H. *Statistical Digital Signal Processing and Modeling.* 2nd ed. Fairfax, United States, John Wiley & Sons, Ltd., 1996.

[21] BISCAINHO, L. W. P., FREELAND, F. P., ESQUEF, P. A. A., et al. "Wavelet Shrinkage Denoising Applied to Real Audio Signals under Perceptual Evaluation". In: *European Signal Processing Conference (EUSIPCO)*, pp. 2061 – 2064, Tampere, Finland, September 2000. EURASIP.

[22] MALLAT, S. *A Wavelet Tour of Signal Processing: The Sparse Way.* 3rd ed. Burlington, United States, Academic Press, 2009.

[23] MCAULAY, R. J., MALPASS, M. "Speech Enhancement Using a Soft-decision Noise Suppression Filter", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 28, n. 2, pp. 137–145, 1980.

[24] EPHRAIM, Y., MALAH, D. "Speech Enhancement Using a Minimum-Mean Square Error Short-Time Spectral Amplitude Estimator", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 32, n. 6, pp. 1109 – 1121, 1984.

[25] EPHRAIM, Y., MALAH, D. "Speech Enhancement Using a Minimum-Mean Square Error Log Spectral Amplitude Estimator", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 33, n. 2, pp. 443 – 445, 1985.

[26] FEVOTTE, C., DAUDET, L., GODSILL, S., et al. "Sparse Regression with Structured Priors: Application to Audio Denoising". In: *IEEE International Conference on Acoustics Speech and Signal Processing Proceedings (ICASSP)*, pp. 57 – 60, Toulouse, France, September 2006. IEEE.

[27] Brooks, S., Gelman, A., Jones, G. (Eds.). *Handbook of Markov Chain Monte Carlo.* Boca Raton, United States, CRC Press, 2011.

[28] NIEDZWIECKI, M., CISOWSKI, K. "Adaptive scheme for elimination of broadband noise and impulsive disturbances from AR and ARMA signals", *IEEE Transactions on Signal Processing*, v. 44, n. 3, pp. 528–537, 1996. Disponível em: <`https://ieeexplore.ieee.org/document/489026`>.

[29] CANAZZA, S., DE POLI, G., MIAN, G. A. "Restoration of Audio Documents by Means of Extended Kalman Filter", *IEEE Transactions on Audio, Speech, and Language Processing*, v. 18, n. 6, pp. 1107–1115, 2010. Disponível em: <`https://ieeexplore.ieee.org/document/5200506`>.

[30] LI, Y., GFELLER, B., TAGLIASACCHI, M., et al. “Learning to Denoise Historical Music”. In: *International Society for Music Information Retrieval Congress*, pp. 504 – 511, Montreal, Canada, October 2020.

[31] MOLINER, E., VÄLIMÄKI, V. “A Two-Stage U-Net for High-Fidelity Denoising of Historical Recordings”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 841–845, Singapore, June 2022. Disponível em: <https://ieeexplore.ieee.org/document/9746977>.

[32] RONNEBERGER, O., FISCHER, P., BROX, T. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 234–241, Munich, Germany, October 2015.

[33] GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M., et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems (NeurIPS)*, Montreal, Canada, 2014. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf>.

[34] THE INTERNET ARCHIVE. “The Great 78 Project”. 2017. Disponível em: <https://great78.archive.org/>.

[35] SEHGAL, A., KEHTARNAVAZ, N. “A Convolutional Neural Network Smartphone App for Real-Time Voice Activity Detection”, *IEEE Access*, v. 6, pp. 9017–9026, 2018. Disponível em: <https://ieeexplore.ieee.org/document/8278160>.

[36] KINGMA, D. P., WELLING, M. “An Introduction to Variational Autoencoders”, *Foundations and Trends in Machine Learning*, v. 12, n. 4, pp. 307 – 392, 2019.

[37] NUSTEDE, E. J., ANEMÜLLER, J. “Towards Speech Enhancement Using a Variational U-Net Architecture”. In: *European Signal Processing Conference (EUSIPCO)*, pp. 481 – 485, Dublin, Ireland, August 2021.

[38] PASCUAL, S., BONAFONTE, A., SERRÀ, J. “SEGAN: Speech Enhancement Generative Adversarial Network”. In: *Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3642 – 3646, Stockholm, Sweden, August 2017.

[39] KONG, J., KIM, J., BAE, J. "HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis". In: *Advances in Neural Information Processing Systems*, pp. 17022–17033, December 2020. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2020/file/c5d736809766d46260d816d8dbc9eb44-Paper.pdf>.

[40] ELOI, M., VESA, V. "Diffusion-based Audio Inpainting", *Journal of the Audio Engineering Society (JAES)*, v. 72, pp. 100–113, 2024.

[41] RADIOCOMMUNICATION SECTOR OF THE ITU. *Recommendation ITU-R BS.1387-2ITU-R BS.1387*. Tech report, International Telecommunications Union, Geneva, Switzerland, 2023. Disponível em: <https://www.itu.int/dms_pubrec/itu-r/rec/bs/R-REC-BS.1387-2-202305-I!!PDF-E.pdf>.

[42] KABAL, P. *An Examination and Interpretation of ITU-R BS.1387: Perceptual Evaluation of Audio Quality*. Technical Report, McGill University, Montreal, Canada, 2003. Disponível em: <https://www.mmsp.ece.mcgill.ca/Documents/Reports/2002/KabalR2002v2.pdf>.

[43] THE PYTORCH FOUNDATION. "Pytorch Doumentation". https://pytorch.org/docs/stable/index.html, 2016. Accessed on 2024-01-29.

[44] NVIDIA. "NVIDIA CUDA Toolkit". https://developer.nvidia.com/cuda-toolkit, 2006. Accessed on 2024-05-13.

[45] HAYKIN, S., VEEN, B. V. *Signals and Systems*. 2nd ed. Hoboken, United States, John Wiley & Sons, 2002.

[46] DE MIRANDA, B. V. *Synchronization and Score Alignment of Musical Recordings*. Undegraduate Final Project, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil, 2021. Disponível em: <http://www.repositorio.poli.ufrj.br/monografias/projpoli10035746.pdf>.

[47] DINIZ, P. S. R., DA SILVA, E. A. B., NETTO, S. L. *Digital Signal Processing: System Analysis and Design*. 2nd ed. Rio de Janeiro, Brazil, Cambridge University Press, 2010.

[48] NICHOLSON, W. K. *Linear Algebra with Applications*. 3rd ed. Boston, United States, PWS, 1995.

[49] MÜLLER, M. *Fundamentals of Music Processing*. 2nd ed. Erlangen, Germany, Springer, 2021.

[50] BOSI, M., GOLDBERG, R. E. *Introduction to Digital Audio Coding and Standards*. Palo Alto, United States, Springer, 2003.

[51] BALAZS, P., DÖRFLER, M., JAILLET, F., et al. "Theory, Implementation and Applications of Nonstationary Gabor Frames", *Journal of Computational and Applied Mathematics*, v. 236, n. 6, pp. 1481–1496, 2011. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377042711004900>.

[52] VELASCO, G. A., HOLIGHAUS, N., DÖRFLER, M., et al. "Constructing an Invertible Constant-Q Transform with Nonstationary Gabor Frames". In: *International Conference on Digital Audio Effects (DAFx)*, pp. 93 – 99, Paris, France, September 2011.

[53] LOY, G. *Musimathics: The Mathematical Foundations of Music*, v. 1. Cambridge, United States, MIT Press, 2006.

[54] BROWN, J. C. "Calculation of a Constant Q Spectral Transform", *The Journal of the Acoustical Society of America*, v. 89, n. 1, pp. 425–434, January 1991. Disponível em: <https://doi.org/10.1121/1.400476>.

[55] HOLIGHAUS, N., DÖRFLER, M., VELASCO, G. A., et al. "A Framework for Invertible, Real-Time Constant-Q Transforms", *IEEE Transactions on Audio, Speech, and Language Processing*, v. 21, n. 4, pp. 775–785, 2013. Disponível em: <https://ieeexplore.ieee.org/document/6384709>.

[56] MATHWORKS. "CQ-NSGT Manual Page". https://www.mathworks.com/help/wavelet/ref/cqt.html, 2018.

[57] BISHOP, C. M. *Pattern Recognition and Machine Learning*. Cambridge, United Kingdom, Springer, 2006.

[58] KARRAS, T., LAINE, S., AILA, T. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4401 – 4410, Long Beach, California, USA, June 2019.

[59] BROWN, T., MANN, B., RYDER, N., et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1877 – 1901, 2020. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf>.

[60] TOUVRON, H., LAVRIL, T., IZACARD, G., et al. "LLaMA: Open and Efficient Foundation Language Models". 2023.

[61] DHARIWAL, P., JUN, H., PAYNE, C., et al. "Jukebox: A Generative Model for Music". 2020.

[62] SALIMANS, T., GOODFELLOW, I., ZAREMBA, W., et al. "Improved Techniques for Training GANs". In: *Advances in Neural Information Processing Systems (NeurIPS)*, Barcelona, Spain, 2016. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf>.

[63] SOHL-DICKSTEIN, J., WEISS, E., MAHESWARANATHAN, N., et al. "Deep Unsupervised Learning using Nonequilibrium Thermodynamics". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pp. 2256 – 2265, Lille, France, July 2015. Disponível em: <https://proceedings.mlr.press/v37/sohl-dickstein15.html>.

[64] HEUSEL, M., RAMSAUER, H., UNTERTHINER, T., et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in Neural Information Processing Systems (NeurIPS)*, p. 6629–6640, Long Beach, USA, December 2017. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf>.

[65] SONG, J., MENG, C., ERMON, S. "Denoising Diffusion Implicit Models". In: *International Congress on Learning Representations (ICLR)*, Vienna, Austria, May 2021. Disponível em: <https://openreview.net/forum?id=St1giarCHLP>.

[66] NICHOL, A., DHARIWAL, P. "Improved Denoising Diffusion Probabilistic Models". In: *Proceedings of the 38th International Conference on Machine Learning (ICML)*, p. 8162–8171, July 2021. Disponível em: <http://proceedings.mlr.press/v139/nichol21a/nichol21a.pdf>.

[67] WELLING, M., TEH, Y. W. "Bayesian Learning via Stochastic Gradient Langevin Dynamics". In: *Proceedings of the 28th International Conference on Machine Learning (ICML)*, p. 681–688, Bellevue, USA, July 2011. Disponível em: <https://www.stats.ox.ac.uk/~teh/research/compstats/WelTeh2011a.pdf>.

[68] VINCENT, P. "A Connection Between Score Matching and Denoising Autoencoders", *Neural Computation*, v. 23, n. 7, pp. 1661–1674, July 2011.

[69] KIRKPATRICK, S., GELATT, C. D., VECCHI, M. P. "Optimization by Simulated Annealing", *Science*, v. 220, n. 4598, pp. 671–680, 1983. Disponível em: <https://www.science.org/doi/abs/10.1126/science.220.4598.671>.

[70] KARRAS, T., AITTALA, M., AILA, T., et al. "Elucidating the Design Space of Diffusion-Based Generative Models". In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 26565–26577, New Orleans, USA, December 2022. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2022/file/a98846e9d9cc01cfb87eb694d946ce6b-Paper-Conference.pdf>.

[71] ANDERSON, B. D. "Reverse-time Diffusion Equation Models", *Stochastic Processes and their Applications*, v. 12, n. 3, pp. 313–326, 1982. Disponível em: <https://www.sciencedirect.com/science/article/pii/0304414982900515>.

[72] SÄRKKÄ, S., SOLIN, A. *Applied Stochastic Differential Equations*. Cambridge, United Kingdom, Cambridge University Press, 2019.

[73] HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. 2nd ed. Upper Saddle River, United States, Prentice-Hall, 1999.

[74] CHUNG, H., KIM, J., MCCANN, M. T., et al. "Diffusion Posterior Sampling for General Noisy Inverse Problems". In: *International Conference on Learning Representations (ICLR)*, Kigali, Rwanda, May 2023. Disponível em: <https://openreview.net/forum?id=OnD9zGAGT0k>.

[75] TANCIK, M., SRINIVASAN, P. P., MILDENHALL, B., et al. "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains". In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 7537–7547, Vancouver, Canada, December 2020. Disponível em: <https://proceedings.neurips.cc/paper/2020/file/55053683268957697aa39fba6f231c68-Paper.pdf>.

[76] PAPOULIS, A. *Probability, Random Variables, and Stochastic Processes*. 3rd ed. New York, United States, McGraw-Hill, 1991.

[77] PEREZ, E., STRUB, F., DE VRIES, H., et al. "FiLM: Visual Reasoning with a General Conditioning Layer". In: *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 3942–3951, New Orleans, Louisiana, USA, April 2018. Disponível em: <https://ojs.aaai.org/index.php/AAAI/article/view/11671>.

[78] HENDRYCKS, D., GIMPEL, K. "Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units". 2016. Disponível em: <http://arxiv.org/abs/1606.08415>.

[79] HAWTHORNE, C., STASYUK, A., ROBERTS, A., et al. "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset". In: *International Conference on Learning Representations (ICLR)*, New Orleans, Louisiana, USA, May 2019. Disponível em: <https://openreview.net/forum?id=r1lYRjC9F7>.

[80] "SoX Arch Linux Man Page". https://man.archlinux.org/man/sox.1.en, 1991. Accessed on 2023-12-13.

[81] MORALES-BROTONS, D., VOGELS, T., HENDRIKX, H. "Exponential Moving Average of Weights in Deep Learning: Dynamics and Benefits", *Transactions on Machine Learning Research*, 2024. Disponível em: <https://openreview.net/forum?id=2M9CUnYnBA>.

[82] MOLINER, E., VÄLIMÄKI, V. "BEHM-GAN: Bandwidth Extension of Historical Music Using Generative Adversarial Networks", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, v. 31, pp. 943–956, 2023. Disponível em: <https://ieeexplore.ieee.org/document/9829821>.

[83] GOEL, K., GU, A., DONAHUE, C., et al. "It's Raw! Audio Generation with State-Space Models". In: *Proceedings of the 39th International Conference on Machine Learning (ICML)*, pp. 7616–7633, Honolulu, United States, July 2022. Disponível em: <https://proceedings.mlr.press/v162/goel22a/goel22a.pdf>.

[84] MARAFIOTI, A., MAJDAK, P., HOLIGHAUS, N., et al. "GACELA: A Generative Adversarial Context Encoder for Long Audio Inpainting of Music", *IEEE Journal of Selected Topics in Signal Processing*, v. 15, n. 1, pp. 120–131, 2021. Disponível em: <https://ieeexplore.ieee.org/document/9257074>.

[85] GRESHLER, G., SHAHAM, T., MICHAELI, T. "Catch-A-Waveform: Learning to Generate Audio from a Single Short Example". In: *Advances in Neural Information Processing Systems*, pp. 20916–20928, December 2021. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2021/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf>.

[86] KITIĆ, S., BERTIN, N., GRIBONVAL, R. "Sparsity and Cosparsity for Audio Declipping: A Flexible Non-convex Approach". In: *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, LVA/ICA 2015, pp. 243—-250, Liberec, Czech Republic, August 2015. Disponível em: <https://doi.org/10.1007/978-3-319-22482-4_28>.

[87] SIEDENBURG, K., KOWALSKI, M., DÖRFLER, M. "Audio declipping with social sparsity". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1577–1581, Florence, Italy, May 2014. IEEE.

[88] STOTZER, S. *Phonographic Record Sound Extraction by Image Processing*. PhD Thesis, University of Fribourg, Fribourg, Switzerland, 2006.

[89] CHOPIN, F. *Préludes, Op. 28*. Leipzig, Germany, Breitkopf & Härtel, 1865.

[90] ISIK, U., GIRI, R., PHANSALKAR, N., et al. "PoCoNet: Better Speech Enhancement with Frequency-Positional Embeddings, Semi-Supervised Conversational Data, and Biased Loss". In: *Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 2487–2491, October 2020.

[91] THICKSTUN, J., HARCHAOUI, Z., KAKADE, S. "Learning Features of Music From Scratch". In: *International Conference on Learning Representations (ICLR)*, Toulon, France, April 2017. Disponível em: <https://openreview.net/forum?id=rkFBJv9gg>.

[92] BECH, S., ZACHAROV, N. *Perceptual Audio Evaluation – Theory, Method, and Application*. Chichester, United Kingdom, John Wiley & Sons, 2006.

[93] LEITE, P. H. L. "Audio Subjective Tests Web Interface". https://github.com/pedrohlopes/web-audio-subjective-tests, 2023. Accessed on 2024-04-25.

[94] DUBAL, D. *The Art of the Piano: Its Performers, Literature, and Recordings Revised*. 3rd ed. Milwaukee, United States, Amdadeus, 2005.

[95] FRIEDRICH GULDA. "Friedrich Gulda: Complete Decca Recordings". 2021. Disponível em: <https://www.deccaclassics.com/en/catalogue/products/friedrich-gulda-complete-decca-recordings-12420>.