CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Latent Vector Synthesis

Exploring the Aesthetic Affordances of Latent Audio Spaces

Master's thesis in Computer science and engineering

DAVID HÖGBERG

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

Latent Vector Synthesis

Exploring the Aesthetic Affordances of Latent Audio Spaces

DAVID HÖGBERG

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

Latent Vector Synthesis
Exploring the Aesthetic Affordances of Latent Audio Spaces
DAVID HÖGBERG

Supervisor: Kıvanç Tatar, Computer Science and Engineering
Examiner: Staffan Björk, Computer Science and Engineering

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2023

Latent Vector Synthesis
Exploring the Aesthetic Affordances of Latent Audio Spaces
DAVID HÖGBERG
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Generative deep learning models for sound synthesis applications have gathered interest recently that are able to generate novel sound material based on the characteristics of a given audio dataset. A subcategory of these models are variational autoencoders, which build generative latent spaces of audio where sounds are organised based on similarity. Although expressive uses of these models abound, questions around their practical applicability and aesthetic implications as part of an artistic process remain underexplored. This thesis investigates the technological and aesthetic affordances of latent audio spaces in the context of creative sound design and exploration. To this end, a sound synthesis tool in the form of a latent vector synthesizer is conceptualised and developed from a first-person research through design perspective. The prototype addresses issues around real-time playability of current machine learning models for sound generation by training a variational autoencoder on short samples of audio signals. The generated waveforms are incorporated as part of a wavetable- and vector synthesis engine that enables timbral interpolations and explorations of sonic textures. Positioned at the intersection of sonic art and audio technology the design implementation uncovers some latent potentials and affordances of new technologies for musical tasks.

Keywords: sound synthesis, variational autoencoders, latent audio spaces, vector synthesis, wavetable synthesis, research through design

Acknowledgements

I would like to express my deepest gratitude to my supervisor Kıvanç Tatar for his invaluable feedback and encouragement throughout the process of completing this thesis. His enthusiasm and expertise have been instrumental in guiding me forward towards finalization. I would also like to give special appreciation to Palle Dahlstedt, who in his inspiring role as a researcher and teacher led me down this path in the first place and opened my eyes to novel fields of research. His guidance planted the initial seeds of this project. Finally, a warm thank you to Kelsey Cotton, whose feedback on the synthesis tool elicited many insights and inspirations for future work.

David Högberg, Gothenburg, June 2023

Contents

List of Figures

1

Introduction

1.1 Overview

Recently, a plethora of generative AI models have emerged that challenge the ideas we have about machines within artistic creation. These are models that can learn to generate new content based on given examples of training data in the form of images, text or audio. Popular examples include ChatGPT[1] for text generation and Midjourney[2] or DALL-E 2[3] for text to image generation. In the audio domain, in particular, DDSP[4] facilitates style transfers between instrument classes whereas Dadabots[5] generates music that imitates the style of a specific genre or band. The research on generative models for sound and music applications, however, has yet to fully mature and there are unaddressed questions as to how artists and creators can practically incorporate these new technologies as part of their artistic workflows.

Variational Autoencoders (VAEs) form a particular category of generative models, which build latent representations of data so that data points that share similarities are organised in proximity to each other in a reduced latent space. This latent space is regularised so that new samples "in-between" the original data points can be generated, for example a completely new image or sound that is an interpolation between given images or sounds. In the audio domain, generative models have been used to construct timbre spaces of sounds, but many of them rely on large network architectures or otherwise computationally expensive conversions in the spectral domain. The applicability of these models for real-time sound synthesis is therefore limited and requires a deeper investigation.

This project in a general sense addresses the opportunities and limitations of incorporating machine learning algorithms and latent space representations in the context of sound and music computing. On the one hand, this points to concrete problems of compatibility between sound synthesis tools and machine learning frameworks as well as the effects of specific audio representations on model performance for musical tasks. But more importantly, we have to consider the use of these models as part of an artistic process, with close attention to the aesthetic implications of the models

[1] https://openai.com/blog/chatgpt
[2] https://www.midjourney.com/home/?callbackUrl=%2Fapp%2F
[3] https://openai.com/product/dall-e-2
[4] https://magenta.tensorflow.org/ddsp
[5] https://dadabots.com

on the artistic output and how they can support creative agency.

New technologies both constrain and liberate creative impulses in how they invite or inhibit certain actions. Exploring the adoption of new tools in an artistic context may therefore unlock the creative affordances of those tools, but also shine a light on their particular limiting functions. Artists have always pushed the affordances of new technologies for aesthetic effects in ways that elicit unintended uses of engineered artefacts. Such explorations may in turn may inform the (re-)construction of new tools. Positioned at the intersection of audio technology, interaction design and sound art, this project aims to trace out the hidden affordances of novel synthesis techniques as well as putting a different lens onto those tools that reveal new aspects of their internal functioning.

1.2 Aim & Research Question

This thesis approaches the question:

> *How can we utilise and interface with latent spaces of audio in order to aid real-time navigation and creative exploration of sound textures?*

The general aim of the project is to design a real-time playable sound synthesis tool that uncovers some creative and aesthetic affordances of latent audio spaces for purposes of sound exploration.

This points towards some specific research objectives:

1. To train a generative machine learning model to encode short samples of audio in a latent audio space.
2. To implement a sound synthesis engine that can interface with the latent space of audio in real-time.
3. To develop a prototype of a latent vector synthesizer that enables exploration of the sonic textures and aesthetic affordances of the latent audio space.

1.3 Contributions

The primary contribution of this thesis with respect to machine learning frameworks is to train a variational autoencoder on wavetables, i.e. collections of single cycle waveforms, that can be flexibly utilised in sound synthesis applications. Implementing a lightweight network architecture trained on short samples of raw audio in this way addresses some problems around real-time applicability of generative machine learning models for audio applications. Also, many of the current machine learning models for music applications focus on audio data from standard acoustic instruments, whereas this project takes inspiration from the domain of electronic music and more experimental forms of sonic art.

Furthermore, a sound synthesis tool in the form of a latent vector synthesizer is

developed from a first-person perspective, as a technologist designing musical interfaces for personal artistic expression, in order to trace out the practical utility and aesthetic implications of latent spaces in the context of sound exploration. The developed interface and workflow not only enables real-time interaction with the machine learning model but also suggests strategies for exploring the potentials of the latent audio space. From the perspective of research through design, then, the main research output and contribution is in the form of explicating a particular design process in order to reveal and uncover the creative potentials of new technologies.

2

Background

This project draws on related work in the field of sound synthesis techniques and the use of machine learning algorithms in artistic processes. Positioned at the intersection of sonic art and technology, inspirations are drawn from musical practice and music technology alike.

2.1 Musicking with Algorithms

The starting point of this thesis is that music is a complex form of activity that extends beyond the apparent artefacts of consumption that we denote as music. Music is made, performed, improvised, heard, danced, enacted and acted out as a social activity, in a cultural and historical context, synthesising influences from ourselves, others and the environment that we are in.

Music-making is rather *musicking* [3].

But more importantly, musicking is conditioned by technology [4]. Our musical expressions are constrained and liberated by the tools we use. Not only do we shape technology and tools to fit our needs, but we are also shaped in return.

The materiality of recording inscriptions, electronics and computer circuits, in particular, challenged the conceptions we have had about music and musical composition. For example, recording technology has permitted a decoupling of the sound object from its original source, allowing it to be edited, cut up, copied, pasted, transferred, reversed, stretched and finally played back in perhaps a completely different aural context (see *musique concrète* [5]). Electronic signal flows, in addition, paved the way for the generation of continuous streams of sound as opposed to sequences of notes, through voltage controlled oscillators [6]. Finally, digital analysis, synthesis and processing of sound materials opened up new possibilities of sound control and design, by further decoupling the sound-generating source from the gestural control of that sound source (as opposed to acoustic instruments where they are intimately coupled). In digital sound synthesis, therefore, the inner workings of the musical instruments are even more occluded from the perspective of the performer in that the digital flows of numbers can be transformed and re-mapped through code in any number of ways.

Arguably, one of the most talked about technological developments today is the rise

of AI and machine learning algorithms and their implications within society, which points to a key question for this project: how does AI and machine learning tools allow for and inhibit musical expression and creative agency? What are the current aesthetic affordances of AI tools and what is missing [7]? Because all artistic tools have aesthetic implications, the question is if we are aware of them and how we make use of them.

2.2 Organised Sound

The technological developments mentioned above in electronic and digital music radically challenge the idea that musical material is naturally organised on a finite lattice structure based on pitch, duration and timbre in how it allows for continuous and timbrally transforming streams of sound [8]. Musicking in the computer age, therefore, becomes more about organising sound-events in time rather than organising notes on paper. Equipped with the computer as a musical instrument, the composer becomes a sound designer in the task of sculpting sonic landscapes and textures.

In the words of Edgar Varèse, a pioneer in the conceptualisation of new forms of music [9]: "I decided to call my music 'organized sound' and myself, not a musician, but 'a worker in rhythms, frequencies, and intensities.' "

The question remains, however, whether these free floating sonic textures can be systematically organised in a way that extends beyond conventional forms of *lattice sonics* [8], in order to support creative exploration and expression in sound design and performance. In particular, how can we utilise recent developments in machine learning and deep learning to construct organised *timbre spaces* (see Section 3.1) that allow for continuous movements between different sound textures? Are there fundamental obstructions and discontinuities as part of these timbral movements that cannot be accurately modelled? As Wishart (1996) points towards [8]:

> It will be interesting to see if the flexibility of the computer will allow us to overcome all such topological restrictions in timbre space (in some ways this would be a pity) or will we discover that timbre space has an intrinsic and insurmountable topological structure quite different from the infinite-coloured fog it is usually taken to be. A second and related question is, can there be any qualitative distinctions between the ways we move through this multi-dimensional continuum? Can motion itself in the continuum have any structure?

2.3 Related Research

2.3.1 Timbre Spaces

The technological problem we are faced with, therefore, is that of structuring and organising a set of sounds by imposing some measure of distance between sonic tex-

tures that share characteristics with each other. This issue has been explored in the foundational work of Grey [10] and Wessel [11], who apply dissimilarity ratings of sounds from acoustic instruments to construct a measure of distance between sound objects as input to a multidimensional scaling (MDS) algorithm. The algorithm reduces the original space in a way that as closely as possible preserves the original distances between sounds as provided by listening tests. The resulting low-dimensional timbre space can then be used for learning audio representations and interpolating between existing timbre textures.

2.3.2 Audio Feature Extraction

Rather than relying on subjective ratings of acoustic contrast, audio signals can also be analysed and measured for quantitative properties and summarised in the form of *audio features*. These features can then be used to aid organisation and classification of sounds. The field of Music Information Retrieval, in particular, deals with different methods for "processing, searching, organizing, and accessing music-related data"[1]. In addition, Hoffman and Cook [12] describes a general framework for automatic extraction of audio features based on quantifiable characteristics of sounds that can be used to structure a sonic control space.

2.3.3 Dimensionality Reduction

Similarly to extracting new features from audio data, we may also want to represent the original high-dimensional data in a compressed form that retains the essential characteristics of the original data distribution. The problem of extracting new features for representing audio data is therefore related to that of reducing the dimensionality of the original search space in a way that supports exploration. This can also be used for visualisation purposes.

The most common dimensionality reduction techniques rely on methods that preserve directions of maximum variability in the data (i.e. discards directions of low variability) or preserve pairwise distances between sound objects (according to some measure of variation and distance). Specifically, Fasciani and Wyse apply principal component analysis (PCA) and Isomap [13, 14] (see also [15]) to match the dimensions between a general timbre space D and a control space C. The inverse direction of the mapping is made possible by maintaining pairwise associations between elements of each transformed matrix, or alternatively by training a neural network. Modern forms of (nonlinear) dimensionality reduction techniques include t-SNE [16] and UMAP [17], which learn low-dimensional embeddings of the original high-dimensional data (see Section 3.3).

2.3.4 Generative Models

Recent advances in machine learning have introduced methods for representation learning [18] that build abstract (latent) representations of data sets (encodings)

[1] `https://www.ismir.net`

that can be used to also generate new data with a similar structure (decodings). Specifically, *Autoencoders* (AEs) are trained to reproduce its data input through a low-dimensional bottleneck in the form of a latent space. That is, the algorithm tries to find a coupled encoder and decoder that minimises the reconstruction error based on input from the original data set. In addition, *Variational Autoencoders* (VAEs) [19, 20] instead encode the input as a stochastic distribution with a particular mean and variance, which enables a smoother interpolation and generation of novel outputs from the decoder in-between data points in the training set.

Another category of generative models are *General Adversarial Networks* (GANs) [21], which instead of a coupled encoder-decoder network train a coupled generator and discriminator model. The function of the generator model is to generate new content, whereas the task of the discriminator is to classify the content as either "real" (from the training data) or "fake" (generated). As the generator improves, the discriminator performs worse on content classification.

Finally, since audio is a temporal signal, many generative models for audio content include different forms of *Autoregressive* (AR) techniques, where the prediction of a temporal sequence is conditioned also on its history. *Recurrent Neural Networks* (RNNs), in addition, include recurrent connections within a neural network to include a form of internal state or memory that can model sequential data.

2.3.5 Spectral Models

In the context of sound exploration, Tatar et al. [22] apply variational autoencoders to learn latent encodings of audio textures that enable interpolation and extrapolation of new sounds in what they term Latent Timber Synthesis. The training of the model is based on the spectrogram of the audio, specifically the magnitude Constant-Q Transform (CQT) spectrogram, which is then converted back to the audio domain using (Griffin-Lim) phase reconstruction.

Similarly, Esling et al. [23] apply VAEs on a dataset of acoustic instrument recordings, with a comparison of different spectrogram types as input (the Short-Term Fourier Transform (STFT), Discrete Cosine Transform (DCT) and Non-Stationary Gabor Transform (NSGT), respectively). In addition, they introduce an additional regularisation term in the model in order to ensure that the structure of the learned latent space is perceptually relevant. The regularisation is based on collected perceptual metrics of acoustic sounds in order to ensure that the constructed latent space follows a structure as indicated by these perceptual timbre studies (similarly to Wessel [11]). Furthermore, Esling et al. [24] proposes normalising flows as a way to transform the simple variational representation to a more complex distribution. Sounds are assigned semantic tags that enable disentanglement of perceptual dimensions in a semi-supervised model.

Conducting listening studies and labelling data based on perceptual ratings often require significant amounts of work and also introduces biases regarding which types of sounds that are listened to (e.g. only sounds from acoustic instruments). In contrast, this project relies on no other data than the sounds themselves and focuses

complementarily in the domain of electronic music and digitally synthesised sounds.

Secondly, the models mentioned work with frequency based representations (spectrograms) as input rather than time domain waveforms, which has clear motivations but also restricts the applicability of the models for real-time processing. Therefore, this project focuses on models that are fast and flexible so that inference can run in real-time.

2.3.6 Raw Audio Models

One reason to use frequency domain features in training audio representations is that audio signals change quickly in the time domain, typically tens of thousands of samples per second (depending on the sampling rate). Since the samples in a time series condition also the following samples, a generative model for long-term audio has to learn to encode these time correlations in the signal. Techniques for learning conditional patterns have been successfully implemented in generative image models (such as PixelRNN [25] and PixelCNN [26]), but was recently adopted also for audio in the WaveNet model [27]. The deep learning architecture works by introducing various dilation paths in the convolutional layers that increase the receptive field and allows for learning of long-term dependencies in the signals.

The WaveNet model, however powerful, is computationally costly in both training- and inference times, which puts a barrier on use in real-time sound synthesis applications. To mitigate these computational constraints, Hantrakul et al. [28] experimented with a convolutional neural network in combination with a WaveRNN model [29] as a form of vocoder for generation of instrument sounds. The model works by analysis and re-synthesis of sounds conditioned on frequency and amplitude as control input to the sound synthesis.

These autoregressive/recurrent models work by generating waveforms sample by sample as part of a time-varying signal. Audio, however, is a special form of signal with a bias towards periodic structures and oscillations that the human ear has evolved to detect and decompose. Traditional synthesizers and vocoders are developed based on this knowledge about how sound is generated and perceived to synthesise new oscillations using digital signal processing (DSP) modules. Building on this idea, Engel et al. developed Differential DSP (DDSP) [30] to explicitly include DSP modules as part of a neural network architecture. They trained a model to analyse and encode audio as parameters of a differentiable additive and filtered noise synthesizer that can re-synthesise new audio. The model was similarly conditioned on pitch and loudness to control the synthesis. The training was performed on the NSynth dataset[2] [31] with applications such as transforming the timbre of an input sound between different instrument classes.

Finally, Tatar et al. [2] encodes frames of raw audio in a lightweight VAE architecture trained on audio data that bypasses the problems of learning long correlations in the audio in favour of faster inference times. The move towards real-time run-

[2] `https://magenta.tensorflow.org/datasets/nsynth`

ning models was motivated by an underexplored problem of how to realistically and practically utilise latent spaces as part of real-world artistic practices.

The following approach builds on the the same ideas of training light architectures on raw audio, with the addition of learning to encode only single cycle waveforms as part of a wavetable synthesis engine. Similarly, a shift towards the perspective of the artist or sound designer utilising these techniques puts emphasis more on the practical strategies of exploring the inherent potentials of latent spaces rather than building them perfectly.

2.3.7 Generative Wavetable Synthesis

Some recent works have been devoted particularly to the use of generative models in wavetable synthesis applications.

Hantrakul and Yang [32] applies the WaveNet model pre-trained on the NSynth dataset of instrument sounds to generate short waveforms of 512 samples. The model can generate playable wavetables by interpolation with combinations of sinusoid, saw and triangle waveforms as input. Because of slow inference times (several seconds on a GPU), however, the wavetables need to be pre-rendered.

In addition, Shan et al. [33] replaces the additive synth in the original DDSP autoencoder with a learnable wavetable synth trained on the NSynth dataset. The model is similarly conditioned on pitch and loudness.

Finally, Hyrkas [34] trains a variational autoencoder on STFT frames of audio from the NSynth dataset. The model relies on a pitch estimation algorithm for the frames of audio as well as phase reconstruction using the Griffin-Lim algorithm back to the audio domain.

3

Theory

3.1 Timbre Spaces

Timbre is the perceived quality of a sound that gives it a distinct colour of tone in comparison to other sounds with the same pitch, loudness and duration [35]. In conventional music theory, the timbre of a sound therefore allows us to distinguish different instrument classes from one another. Importantly, then, timbre is not an intrinsic physical property of a sound, but rather a multi-dimensional texture dependent on the specifics of how the sound is generated, how it reverberates in an environment and how it interacts with the physiology of the ear. This multi-dimensionality of timbre can be referred to in different ways using labels such as brightness, roughness, hollowness or inharmoniciy [36].

Since timbre is defined in terms of a perceived distinctness between sounds, scientific approaches to characterise timbre often relies on listening studies where sounds with the same pitch and loudness are rated based on their perceived dissimilarity. The ratings can then be used to construct a *timbre space* structured by dimensions or audio descriptors that most accurately separate sounds perceived as having a distance from each other. The assumption is that we may represent timbre in the form of a reduced set of auditory dimensions that listeners implicitly utilise in comparing sounds.

As mentioned, the work of Grey [10] and Wessel [11] here introduce multi-dimensional scaling (MDS) as a dimension reduction technique to represent audio in a way that maintains perceived distances between sound object. The idea is that the dimensions of the generated timbre space may correspond to measurable properties of the sounds that influence the tone colour in a specific way [36].

3.2 Audio Analysis

Through grounded knowledge in psycho-acoustics and the physicality of sound, we may understand and represent sound and audio in this way by *analysing* its component parts. Through analysis, therefore, we seek to understand complex sonic textures by abstract concepts and representations that capture certain essential characteristics of the perceived phenomenon. To this end, computational tools for audio signal processing and audio representations can be applied that quantitatively

capture certain aspects of a time-varying audio signal [37].

3.2.1 Time- and Frequency Representations

Any sound can be intuitively graphed out as a *waveform* that captures the physical nature of sound as deviations in air pressure through time at a given location. The rate of repeated fluctuations in pressure determines the *frequency* of the audio wave, which in turn relates to the pitch of a perceived tone, whereas the size of the fluctuations (i.e. the *amplitude*) influences the perceived loudness or intensity of the sound.

In theory, any complex waveform can be decomposed into or approximated by a sum of simpler periodic waveforms with distinct amplitudes and frequencies. Therefore, rather than looking at the temporal amplitude variation through time of a waveform, we may consider the different frequencies of basic waveforms that make up the sound. The *Fourier transform* (FT) of a signal here functions as a mathematical tool to convert a time-dependent continuous signal to a representation of the frequency content in that signal.

Figure 3.1 shows an example waveform as a superposition of sine waves and its corresponding frequency content as given by the Fourier transform.

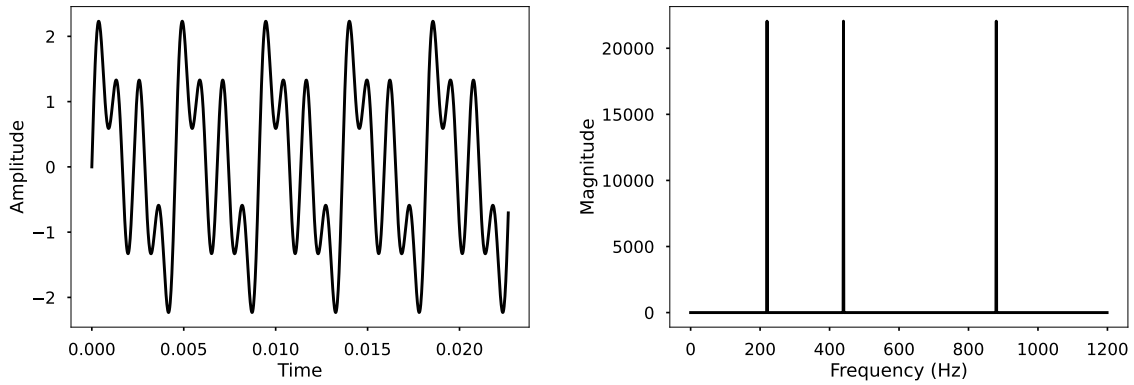

Figure 3.1: Time plot (left) of a waveform and the corresponding frequency plot (right) as given by the Fourier transform of the signal.

Working with digital audio, however, we have to consider ways of discretising a continuous signal for digital processing. In practise, we therefore need to choose a *sampling rate*, i.e. a rate at which to store amplitude values of the time-varying continuous signal. Similarly, there is a corresponding discrete approximation of the continuous Fourier transform termed the *discrete Fourier transform* (DFT) that translates between time- and frequency representations of a sound.

3.2.2 Windowed Fourier Transforms

As we progress from a time-varying waveform representation of sound to an image of the frequency content in the waveform (by applying the Fourier transform), we hide information about when these frequencies are found as part of the waveform.

We therefore have a trade-off between retaining time- vs. frequency information in representations of audio. A method to reduce the effects of averaging the time information of a whole signal is to calculate the Fourier transform on smaller time windows as given by a *window function*. Specifically, the *short-time Fourier transform* (STFT) utilises this notion of short-time windows that are transported across the signal (with a step size determined by the *hop size*) to apply the transform.

This introduced time-dependency in the frequency representation can be visualised in the form of a *spectrogram* that describes the frequency content of the signal through time in a 2D-plot (see Figure 3.2).

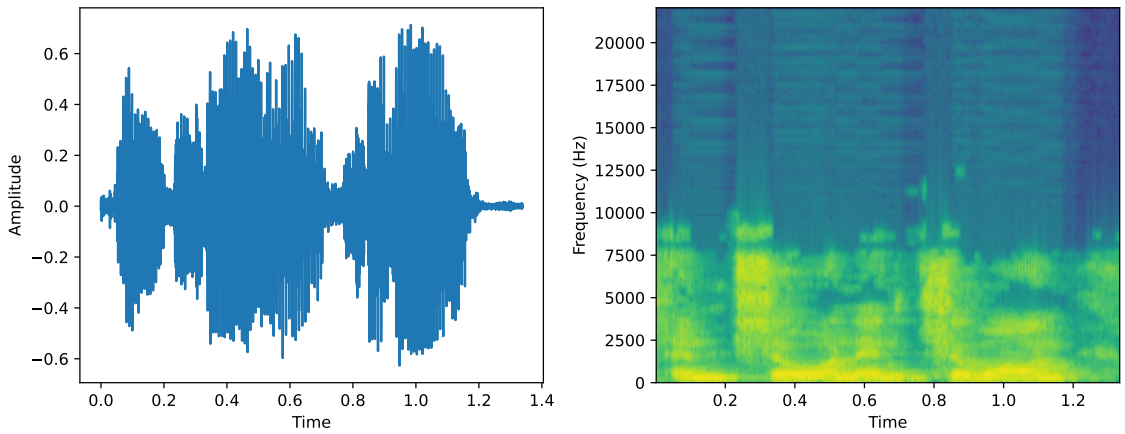

Figure 3.2: Spectrogram (right) of a signal (left) that shows the frequency content through time as given by the short-time Fourier transform.

3.2.3 Wavelet Transforms

With the windowed Fourier transform, we can analyse signals with time-varying frequency content in this way. Since the windows have fixed length, however, we need to decide on a fixed time-resolution and therefore a preferred scale of analysis. The problem is that detecting high frequency content requires shorter time resolution than low frequency components, which introduces the need for multiresolution models that can capture frequency information at different scales. *Wavelet transforms* [38], for example, here include different time-resolutions depending on the frequency range (band) under consideration.

3.3 Dimensionality Reduction

The general idea behind dimension reduction techniques is that we might represent high-dimensional data in a compressed form that still captures the essential characteristics of the true data distribution. This can be used for visualising high-dimensional datasets, but also for extracting new types of features and data representations that defines a latent generative space for data reconstruction and exploration.

These techniques can be broadly divided in two categories depending on how distances between data points are utilised and preserved as part of the algorithm. Either the algorithm aims to preserve pairwise distances globally between all data points (PCA, MDS), or the local distance structure is emphasised (Isomap, t-SNE) [17].

In Principal Component Analysis (PCA) [39], the original data is projected (linearly) onto the directions of highest variance in the data (the principal components). This means that PCA discards directions of low variability in order to preserve the maximum amount of information (variance) in the data. Multidimensional Scaling (MDS) [40], on the other hand, takes a distance matrix of pairwise connections between points as input and builds a lower-dimensional representation that tries to preserve the information contained in the mutual distances between objects in the dataset.

In addition, there are non-linear manifold techniques based on learning an embedding that represents similarities (distances) in the original data. These include for example Isomap, t-SNE and UMAP.

Isomap [41] builds on similar principles as MDS but using geodesic distances instead of a Euclidean metric. The algorithm imposes a graph structure on the original data and then preserves local neighbourhoods of the graph through an isometric mapping. In addition, t-SNE [16] is a nonlinear probabilistic dimensionality reduction method that models the probability that different data points are close to each other in a high-dimensional space and then tries to preserve those probability distributions also in a low-dimensional embedding. Finally, UMAP [17] builds on t-SNE but with lower computation times and a better preserved global structure of the data.

3.4 Audio Synthesis

In addition to *analysing* audio through abstract representations, we can also understand sounds by learning how to generate and produce them, which introduces a more experimental approach of understanding by *synthesising* the complex phenomenon under observation [35]. Acoustic instruments could be conceived of as tools for sound synthesis through manipulating the physical and mechanical properties of objects in order to generate acoustically interesting effects. The introduction of the computer, however, enables new types of analysis and synthesis of sounds in the form of adding, subtracting, filtering and in different ways manipulating the properties of sound generating oscillators and recorded audio samples [6].

In *additive synthesis*, complex waveforms are constructed by adding simpler waveforms together (e.g. sine waves). This follows from principles in Fourier theory that state that all periodic signals can be expressed as a sum of sinusoidal functions, which enables us to construct a desired sound by adding signals together.

Subtractive synthesis takes the opposite approach of starting with a complex waveform and filtering away undesired frequencies and properties of that complex sound.

In contrast to the constructive approach of additive synthesis, then, subtractive synthesis is more about sculpting a sound than building it from scratch.

Additionally, we can add variation and complexity to a sound by *modulation*, which means that e.g. the frequency of the sound is altered based on the current amplitude of another wave. Typically, the modulation signal has low frequency (a low frequency oscillator or LFO), but we can also use high frequency waves to modulate other high frequency waves. This underlies the theory of *frequency modulation synthesis* [42] (or correspondingly, *amplitude modulation synthesis*). The complexity and character of the resulting sounds depends to a large extent on the harmonic relationship between the modulation signal and the original (carrier) signal.

3.5 Generative Models

In practise, there is a feedback loop between methods of analysis and synthesis in that an analysis can be evaluated by (re-)synthesising new observations from the abstract model. If we can decompose a complex sound texture by some abstract representation and then re-synthesise similar textures using that abstract representation, we have to some extent confirmed that the representation carries experimental value [43].

With this perspective of sound analysis and synthesis in mind, we can take advantage of recent advances in machine learning and similar computational methods to first analyse the contents of audio data in order to then (re-)synthesise new samples through a different lower-dimensional data representation that enables novel ways of exploring the sound space.

In the application domain of audio analysis and synthesis, therefore, we are interested in not only compressing, reducing or representing audio data (*encoding* data), but also reconstructing it in order to synthesise new sounds (*decoding* from a reduced representation). Ideally, there is a lossless encoding of the original data that enables perfect data reconstruction from its compressed form (which requires some level of redundancy in the original data representation). In the case of a lossy encoding, however, we may still encode data in a way that minimises the reconstruction error, i.e. the difference between the original data and the reconstruction by some measure of error. This is essentially how *autoencoders* work, a set of techniques for training a coupled encoder and decoder to reconstruct its own input through a lower-dimensional bottleneck [44] (see Figure 3.3). This also means that we can use the decoder directly to synthesise new samples 'in-between' data points of the original data set.

While autoencoders learn a way to encode the input data in the form of a latent representation, there is no guarantee that the input is encoded in an organised way, i.e. that we can generate new content from the latent space that resembles the original data.

Variational Autoencoders (VAEs) [19, 20] extend autoencoders in that instead of

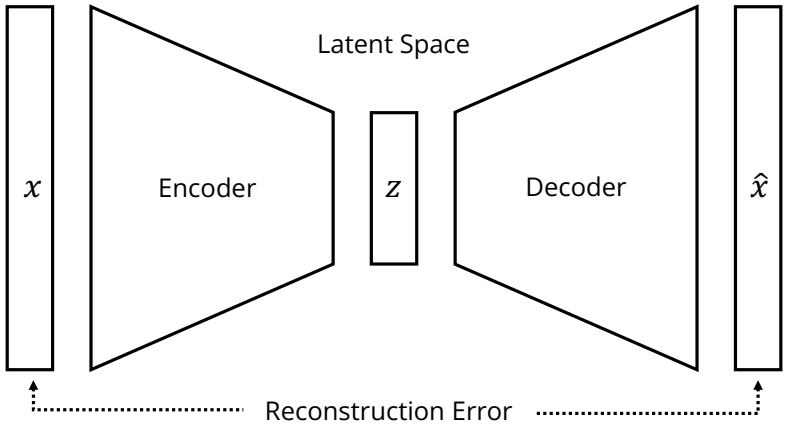

Figure 3.3: General autoencoder architecture with an input vector x that is encoded to a latent space vector z and then decoded as a reconstructed vector $\hat{x}$. The performance of the coupled encoder and decoder is optimised based on the reconstruction error between the input x and the output $\hat{x}$.

encoding data as points in a latent space, we encode the input as parameters for a latent probability distribution that enables a smoother and more meaningful reconstruction of nearby data points (see Figure 3.4). VAEs can represent and reconstruct data points in the training set, while maintaining levels of continuity between samples. VAEs therefore adds an element of *regularisation* of the latent space in order to ensure meaningful and smooth outputs from the decoder.

Formally (see [44]), we assume that the original data samples x comes from an unknown data distribution $p(x)$ that we want to represent or approximate. Furthermore, we assume that the original data share a relationship with and can be generated by some latent (hidden) variables z from a latent distribution $p(z)$. We can then calculate the joint probability $p(x, z) = p(x|z)p(z)$, where $p(x|z)$ is the conditional probability of observing x given the latent variable z. This is the *decoder* part of the network, describing how to generate samples x given a latent z.

Here, we consider $p(z)$ to be a standard Gaussian distribution and $p(x|z)$ to be normally distributed with a mean as given by a function $f(z)$ and a fixed covariance given by a constant c. That is,

$$\begin{aligned} &p(z) \sim \mathcal{N}(0, I), \\ &p(x|z) \sim \mathcal{N}(f(z), cI), \quad f \in F, \quad c > 0, \end{aligned}$$

where f belongs to a family of functions F (to be determined).

Through Bayes theorem, we also know that the posterior distribution

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x|z)p(z)}{\int p(x|u)p(u)du},$$

which constitutes the *encoder* part of the network, i.e. how to generate latent z given an input x.

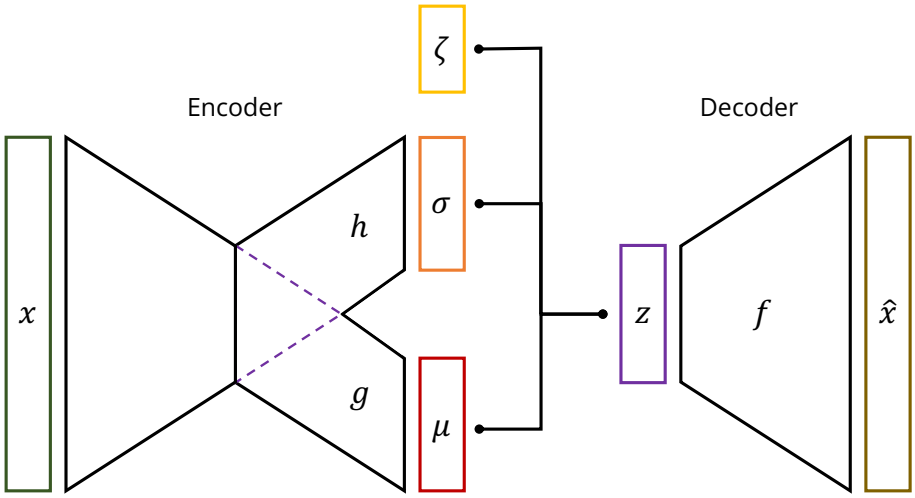

Figure 3.4: General variational autoencoder architecture with an input vector x that is encoded to the parameters μ and σ of a normal distribution. The latent variable z is then sampled from the normal distribution using the reparametrisation trick, with $\zeta \sim \mathcal{N}(0, I)$, before it is decoded as $\hat{x}$.

The integral in the denominator, however, is generally intractable to solve analytically for complex distributions. The idea behind *Variational Inference*, therefore, is to assume that $p(z|x)$ can be approximated using a simpler distribution $q(z|x) \in \mathcal{Q}$ belonging to some family of distributions $\mathcal{Q}$. The problem of inference can therefore be transformed into a problem of optimisation by minimising the "distance" between $p(z|x)$ and $q(z|x)$, which can be measured by the so called Kullback-Leibler (KL) divergence. This gives us a problem of finding

$$q^*(z|x) = \text{argmin}_{q(z|x)\in\mathcal{Q}} D_{KL}[q(z|x)||p(z|x)], \tag{3.1}$$

where we optimise the parameters of the distribution $q(z|x)$.

Specifically, we may assume that $q(z|x)$ belongs also to a family of normal distributions, whose parameters (the mean μ and variance σ) are functions of x. I.e. we assume that

$$q(z|x) \sim \mathcal{N}(g(x), h(x)), \quad g \in G, \quad h \in H,$$

where g and h belong to a family of functions G and H, respectively.

Now, we let the functions f, g and h that determine the parameters of the probability distributions be chosen from a set of functions defined by a neural network architecture. The encoder part of the variational autoencoder therefore seek to approximate the functions $g(x) = \mu_x$ and $h(x) = \sigma_x$ that become the parameters of the normally distributed $q(z|x)$. Similarly, the decoder network determines the function $f(z) = \mu_z$ and the mean of the distribution $p(x|z)$.

The whole model is then a concatenation of the encoder and the decoder. Since the encoder returns a probability distribution rather than a point in a latent space, however, we need to sample a latent vector z from this distribution. This introduces a problem in the training of the network in that a random sampling from a distribution is not differentiable and therefore blocks the error from backpropagating

through the network. The solution is what is called the *reparametrisation trick*, utilising the fact that z can be expressed as a function of $g(x) = \mu_x$ and $h(x) = \sigma_x$ as

$$z = \sigma_x \zeta + \mu_x, \quad \zeta \sim \mathcal{N}(0, I),$$

where ζ is from a standard Gaussian distribution.

Based on our original optimisation problem (equation 3.1), by expanding the KL divergence and re-arranging terms, we can derive the following loss function for training the neural network:

$$\mathcal{L}_{f,g,h} = \underbrace{E_{q(z|x)}[\log p(x|z)]}_{\text{Reconstruction Term}} - \underbrace{\beta \cdot D_{KL}[q(z|x)||p(z)]}_{\text{Regularisation Term}}, \tag{3.2}$$

where the functions f, g and h are determined by the weights of the network. The encoder network approximates the probability $p(z|x)$ through the functions $g(x) = \mu_x$ and $h(x) = \sigma_x$, whereas the decoder approximates the distribution $p(x|z)$ through the function $f(z) = f(\sigma_x \zeta + \mu_x) = \mu_z$. The first expectation term of the loss function corresponds to the reconstruction error of the model whereas the second term regularises the distribution $q(z|x)$ to follow that of the prior $p(z)$. The parameter β is set as a hyperparameter for training that adjusts the trade-off between these two terms, namely between reconstruction error and regularisation of the latent space.

The detailed explication of this loss function is laid out in [19].

4

Methodology

This project lies at the intersection of technology, interaction design and sound art, with an emphasis on the adoptability of engineering tools for artistic purposes. We therefore need to consider the value of doing research within this particular context in order to elucidate what kind of research results we might expect and how to evaluate the results.

There are several ways in which these fields can intersect. From one point of view, the engineer/technologist develops technologies for sound processing (e.g. the back-end of a DAW or VST), the designer constructs a user-friendly interface that enables interaction with these tools, which the artist then utilises for musical creation. Today, however, the lines between these different roles are increasingly blurred with the rapid development and distribution of new digital technologies. With tools such as Pure Data, Max and SuperCollider, a technologically savvy sound artist can develop and design their own instruments. Also, with new forms of multi-media expressions, the artist increasingly becomes a designer of immersive and interactive tools and experiences that hinges on technological knowledge or support.

In terms of methodology, therefore, we necessarily move in the tension between different roles and how they influence the research output.

4.1 First-person Methods

Conventionally, user-centric design takes a third-person perspective in project evaluation based on for examples user interviews, observations and prototype testing. Recently in HCI research, however, first-person perspectives have gathered interest as an approach for using the designers own body and lived experience as an important resource in the design process. This brings the designers own perspectives and intentions to the forefront and tightens the loop between design and user experience in a way that allows for faster iterations and evaluations [45].

The first-person perspective also emphasises how designers and users are not always that easily separated into categories of active creators and passive consumers of a product or tool. In fact, many tools are in a perpetual state of redesign, where user experience and design reconstitute each other in a feedback loop. As Höök et al. notes [45], "[t]he end-user will actively co-construct their experience, the meaning

of the interactions, together shaping practices and engagements over time." This is particularly true of *aesthetic* experiences, which according to the pragmatist philosopher John Dewey have a singularity to them that hinges on active and subjective participation rather than passive reception. Interactive technologies transform our experiences and participation in the world by either "amplifying or reducing reality" as well as through "inviting and inhibiting actions" [45].

The current work explores designing a sound synthesis tool using new technologies in a way that permits certain aesthetic explorations of sounds. Therefore, the work centres on the first-person perspective of the technologist as designer as artist, adapting current technological tools for personal use and developing personalised interfaces to enable interaction with those tools. Furthermore, beyond the functional affordances of these tools, the process centres on expressing a creative intent through new technologies as well as using new technologies creatively.

To this end, *autobiographical design* is about designing with and for the self by tracking the experiences and desires of the designer in utilising the system as it is being developed [46]. Rather than trying to separate functions of designing and using, therefore, autobiographical design takes advantage of the designer being able to take on different roles and perspectives as part of the design process. In this project, therefore, a project log was kept during the design process to trace the specific ideas, experiences and developments that influenced the direction of the design implementation.

4.2 Research through Design

Research and design are two fields that inevitably influence and inform each other in many ways. But what, exactly, is design research? Based on Frayling's influential categorisation [47], we can do research *about* (*into*) design, research *for* design or research *through* design [48]. Research *about* design takes the field of design itself as its object of study and traces the forms and shapes it inhabits. Research *for* design, on the other hand, aims at generating design frameworks, methods, philosophies and recommendations that help designers navigate and frame the problems at hand. In contrast, research *through* design (RtD) is rather about generating knowledge and insight by acting out a particular design process for a specific situated task. Instead of objectively studying and/or developing frameworks for design practises from a distance, RtD actively engages in those practises and includes that particular engagement as part of the research output. Specifically, Gaver states that "reserach through design is likely to produce theories that are provisional, contingent, and aspirational" [49], and in that sense aims at expressing the particularities of a given process and its generative potentials rather than general statements about "what is". In other words, from this perspective, there is no optimal design, since the specifics of the context, the intended user and the situated task changes with each new design implementation.

As Zimmerman et al. point out [48], because of its provisional and situated nature, research through design serves as a viable approach towards so called "wicked

problems" for which there are no optimal solutions that hold for all times and all contexts. Furthermore, research through design is essentially aimed at the future by implementing design artefacts that pushes the world towards a desired state, rather than focusing on what was or what is, which makes it a suitable approach for exploring new fields of knowledge that are not fully formed yet.

In this context, the main research output of the project is the design process itself rather than the particular design artefact that come out of this process (although that is also significant). The act and process of designing interfaces and tools for musical expression in this way functions to draw out the particular affordances of technologies in generating creative content and aesthetic experiences.

There are relevant critiques and calls for further developments of these perspectives on doing design research. In particular, the question is in what ways research through design can produce actual design theory and systematic forms of knowledge [48]. Also, there is still a call for and a need to evaluate what a good outcome and performance means in this context. Importantly, however, first-person methods and research through design practices are complimentary to and not a replacement for other methodologies in Human-Computer Interaction (HCI). Focusing on first-person perspectives, in particular, does not bypass the need for conventional methods of user evaluation and feedback in prototype design.

4.3 Evaluation of Musical Interfaces

In designing interfaces for musical expression, each instantiation of a new interface carries with it the intentions and artistic interests of the creator of the interface, as well as the context in which the musical task is situated. It is therefore a common feature in musical interface design that the creator of the interface is also the main user of the interface [50]. This points to a consideration whether the interface should have broad usability (a common evaluation criterion in HCI), or rather expressive potential for certain use cases that can then be re-appropriated and developed by others. The artistic intentions and context therefore plays an important role in how to approach evaluation in interactive computer music [51]. For example, a sound exploration tool would be valued according to its potential of finding new types of sounds that can be used in sound design and composition tasks, whereas a performance instrument would be evaluating based on the immediacy and expressivity of the interaction and the way it affords musical agency.

With regards to specific evaluation criteria for musical tasks, Wanderley and Orio [51] mentions the importance of learnability, explorability, controllability of audio features and controllability of timing when interacting with a musical interface. Timing, rhythm and haptic feedback are particularly desired characteristics of a musical interface that might not be so relevant in other evaluation contexts. The gesture-sound interaction also requires a level of consistency, continuity and coherence to ensure an intuitive feedback process for learning and exploring an interface [52].

In this context, the evaluation centres around the particular situated musical task of

sound exploration and design of sonic textures with a specific aesthetic quality. To this end, an evaluation study of the prototype was performed that focused on users in a similar context and with similar interests to compliment and extend the first-person perspectives on the musical interface. Evaluation questions were formed to elicit the aesthetic affordances and experience of using the tool, in particular, but also general concerns of usability and navigability. In addition to individual user experiences, the evaluation situated the prototype within a collaborative performance setting.

5

Design Process

This chapter outlines the design process, starting with an initial ideation phase and leading up to a final prototype. As retroactively constructed from the end point, this process takes the shape of a linear movement from beginning to end, which allows us to draw out the significant moments and general trends that contributed to this movement. In practice, however, any design process is of course highly non-linear and iterative and full of contradictions.

The first section describes the inspirations and influences that led up to the idea of a latent vector synthesizer. These influences are conceptual and artistic as well as technological. The next design phase delved into the opportunities and complications of working with sound from a data-centred perspective, as well as building latent spaces of audio. This in the end pointed to a pipeline design that could encompass these data-centred tasks, but also handle low latency audio signal processing. Finally, the whole process of designing the prototype is outlined, from the initial coding to the final graphical interface design.

5.1 Ideation

5.1.1 On Sonic Art

An important influence on this project is the creative opportunities opened up by digital computing technologies for the sonic arts. As mentioned, these technologies push the conceptions we have about music towards a ”liberation of sound” [9] from conventional ”lattice sonics” [8]. Personal experimentation and experience with modular synthesis techniques, in particular, served as a starting point for new forms of sound explorations focusing more on the qualities of the sound textures themselves than their particular musicality and harmonicity as defined by conventional notation systems.

Building on rather than negating these conventional modes of music making and composition, the liberation of sonic textures in this way allows us to conceive of sound as organised without a clear delineation between pitch, noise, timbre, and rhythm. The question remains, however, how these sound textures can be organised in other ways, utilising modern computing technologies. That is, can we construct a timbre space of sounds that allows us to creatively express transitions between any

types of timbres? What kind of structure (if any) does this space exhibit?

A particular genre of sound art dealing with slowly evolving streams of sounds and changing timbres is *drone music*, which functioned as an important aesthetic framing and creative constraint in the design of the synthesis tool. This is a type of music that relies heavily on being able to make such smooth transitions in timbre space.

5.1.2 Sustained Sustain

"In essence, drone equals sustain – sustained sustain, if you will." [53]

Drone music utilises the power of sustained vibrations.

These can be heard today as part of ambient soundscapes, but drones are ancient, reverberating through time from the beginnings of human activity or perhaps even the universe itself. In particular, Harry Sword traces the history of drone from Neolithic burial chambers and the sounds of nature through Buddhist chanting, Indian ragas, Greek aulos pipes and Sufi trance rituals to the modern underground and its experimentation with ringing feedback and noise [53].

Aesthetically, these sustained tones and textures can range from soothing and warm to sharp and agitating. The drone is "an audio carrier vessel capable of evoking womb-like warmth or cavernous dread alike" [53]. In fact, a good drone probably strikes a balance between the smooth and the harsh, between tension and relaxation.

In the late 20th century, drones were perhaps associated with different forms of drone metal, utilising heavily amplified electric guitars, but also as a sub-genre of electronic music, enabled by the sustained tones of voltage controlled oscillators. Today, these oscillators are a fundamental part of any synthesizer or digital audio workstation and the opportunities to generate and modify drones digitally abound.

5.1.3 A Drone Synthesizer

What constitutes a good drone synthesizer? What are its component parts?

With computer music, the drone transitions from physically vibrating sound sources to internally generated electronic or digital signals. A key aspect of a good drone synthesizer, therefore, is the oscillator, the sound generating source, the beating heart.

Secondly, the drone is monotonous but always evolving and transforming. The drone is sustained but with continuous changes in timbre and texture. A good drone synth therefore hinges on these slow transformations of timbre in the sound.

Finally, in acoustic environments and instruments, the generated sounds bounce off of objects and materials and return in the form of echoes and reverberations of filtered aspects of the sounds. These reverberations of the original tone are crucial for a rich drone sound and has to be digitally emulated in a synthesizer.

These three aspects (sustained tones, timbral transitions and reverberation) were

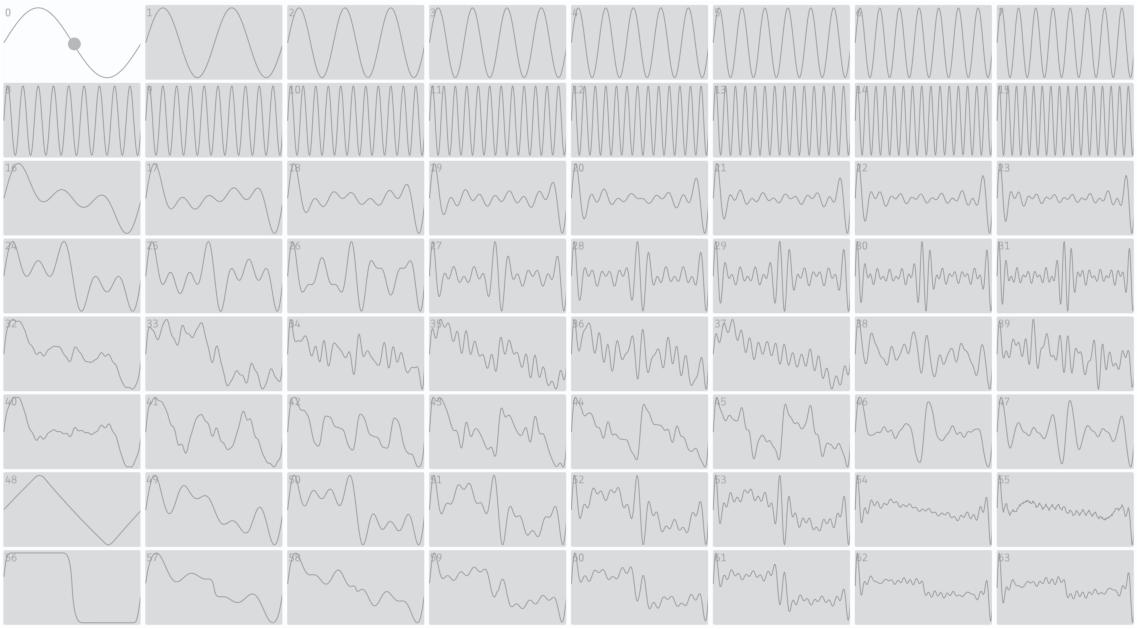

Figure 5.1: A set of 64 wavetables sequentially ordered on a grid from the WaveEdit software. The position of the dot determines which waveform is played to generate a tone, with morphing capabilities between the waveforms.

put down as key components of a drone synthesizer and added technological constraints and requirements to the design of the synthesis tool. In particular, wavetable synthesis and vector synthesis were conceptualised as promising methods of sound synthesis to allow for sustained oscillations with smooth timbral transitions, which in combination with post-processing modules such as filters, delays and reverb would meet the technical requirements of a good drone synth.

5.1.4 Wavetable Synthesis

Wavetable synthesis [54] builds on the principle that in order to generate a sustained tone from a periodic signal, we only need to store one cycle of the waveform in a circular table and then read that table sample by sample in a loop. The frequency of the tone will depend on how fast the table is read through. Additionally, with several waveforms stacked on top of each other comes the ability to cross-fade or interpolate between them to dynamically change the timbre of the tone.

An open source tool, WaveEdit [55], was used in the early stages of the project to explore the potential of wavetables for generating complex evolving sound textures. The tool includes functionality for constructing and editing a set of periodic waveforms as well as morphing between them (see Figure 5.1). The tool also includes functionality for importing audio files to generate custom wavetables.

5.1.5 Vector Synthesis

Vector synthesis, in addition, is a synthesis technique that introduces movement and texture to a sound by cross-fading between a set of predetermined sound sources.

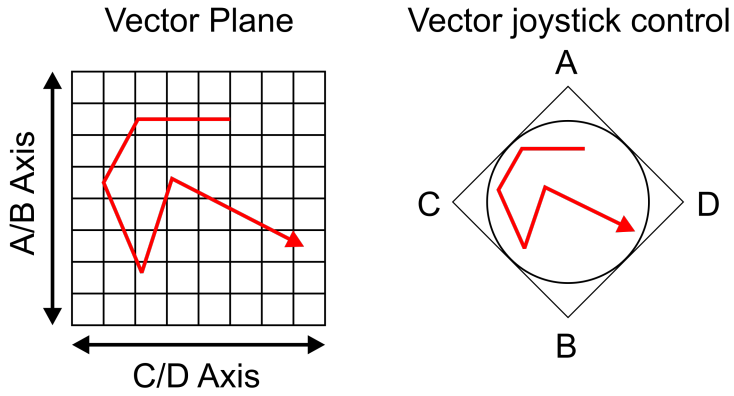

Figure 5.2: Vector synthesis uses the position in a vector plane to control a mix of four sound sources. This control surface can be mapped to for example a joystick. Image from [1].

The technique was introduced by Sequential Circuits in their Prophet VS synthesizer in 1986 [56], which implemented cross-fading in a vector plane between four different wavetable oscillators (labelled A, B, C and D), using a joystick as a controller (see Figure 5.2).

A vector plane of sounds can be explored using a controller in this way to change the position, but the position can also be modulated using envelopes or low frequency oscillators (LFOs). Here, a modern form of vector synthesizer like Vector[1] lets the user define modulation trajectories in the form of shapes on a display. As the point follows the trajectory the sound changes character.

This inspired the implementation of a vector synth with four wavetable oscillators as source waveforms, with the additional functionality to create morphing trajectories in the vector plane for introducing movement to the sounds.

5.1.6 Latent Vector Synthesis

The notion of tracing out a path for a sound texture in a vector plane rhymes well with the idea of a latent space, since each latent vector points to a sound in an organised space of sound textures that can be traversed through interpolation and/or extrapolation in different directions. Any four points in the latent space span a vector plane that can be utilised for sound synthesis in this way. Cross-fading between waveforms then corresponds to interpolation in the latent space.

Additionally, wavetable synthesis generates sustained tones using only short single cycle waveforms, which means that a latent encoder only has to learn to encode short samples of sounds rather than longer tones with timbral fluctuations and time correlations. Utilising neural network architectures in this way for generating single cycle waveforms and interpolating between them therefore enables us to reliably form

[1] `https://www.vectorsynth.com`

longer sustained tones such as drones that evolve slowly as the original waveform is altered.

An important step in the design process of the synthesis tool, therefore, was the idea of revisiting or revitalising old synthesis techniques using new technologies, in this case replacing vector planes with latent vector planes.

5.2 Musicking with Sound Data

Having formed an idea about the aesthetic framing and the technology to support it, the next phase of the design process looked more carefully at data processing tools and machine learning frameworks to address problems of sound exploration. To this end, there are a variety of Python libraries suitable for data retrieval and processing, ranging from audio analysis and feature extraction tools (such as `librosa` [57]) to dimensionality reduction algorithms (as part of `scikit-learn` [58]). For deep learning tasks, PyTorch [59] is one of the most widely used libraries. This makes Python a suitable programming language for data-centred tasks. The question remains, however, how to integrate these frameworks and tools as part of a sound synthesis application (as will become an important consideration).

Any data science or machine learning project starts with data and the exploration of data. An important step in this project was therefore to select a suitable audio datset to work with, that would in turn determine the structure of the sound space that we want to explore. Because, as often pointed out with regards to machine learning applications, the algorithms can only learn what you feed it, which means that the initial data selection and processing will greatly influence the performance and aesthetic output of the model.

5.2.1 Dataset Selection

Sound data come either in the form of recorded audio or synthesised sounds. A website like Freesound[2] offers user uploaded sounds and sound packs, but sometimes researchers publish their data for public use as well.

In order to incorporate sounds as part of a wavetable synthesizer, however, we need data in the form of a single cycle of a periodic tone. Given a sound file, we may attempt to extract these single cycle waveforms by estimating the period of the wave or by cutting up the sound in segments and applying an amplitude envelope. In this project, however, a dataset was selected consisting of waveforms specifically prepared for wavetable synthesis, compiled by Ekstrand [60]. The dataset contains 4358 waveforms in total, that are 600 samples long with a sampling rate of 44100 samples per second. All waveforms also start and end at amplitude 0, which enable looping as part of a wavetable. Furthermore, the dataset is divided in 45 subfolders based on the type of waveform contained in it or how it was generated. Many of these folders have generic names but some show labels such as 'square', 'saw' or

[2] `https://freesound.org`

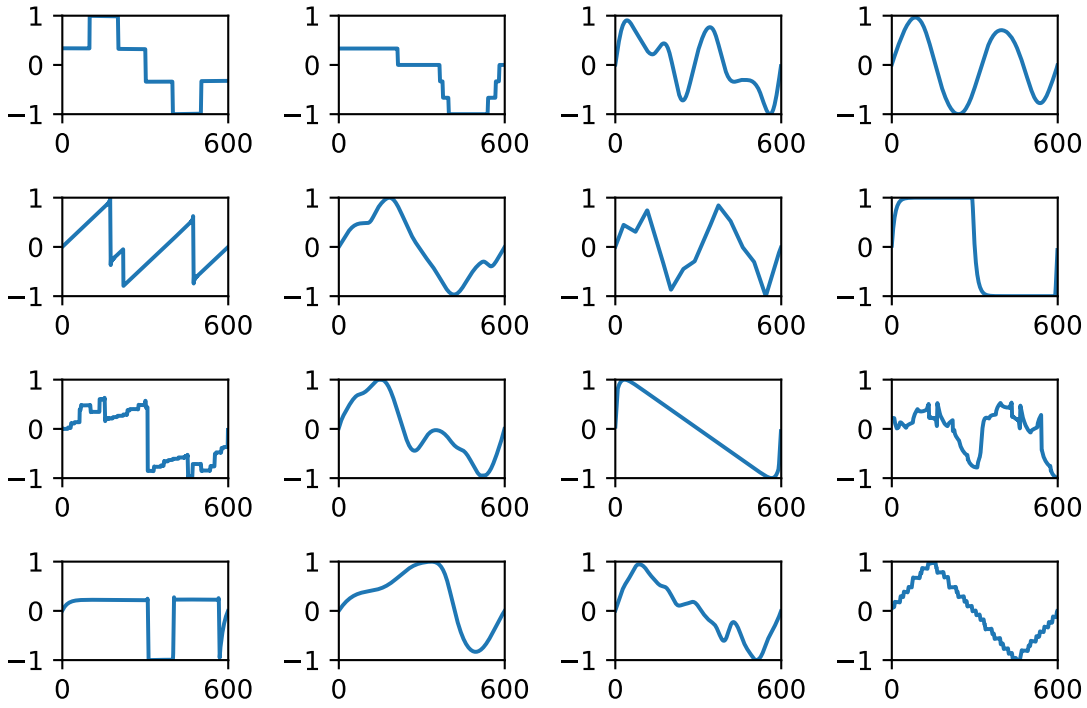

Figure 5.3: A selection of 16 random waveforms from the dataset.

'eorgan' that indicate what type of waveform or shape is contained in the folder.

5.2.2 Random Waveforms

The initial exploration of the dataset was performed interactively in a Jupyter Notebook[3], where the first step was to generate and listen to randomly selected waveforms from the collection to get a view of the range of sounds available. This random waveform selection was also later implemented as a function in the synthesis tool.

The dataset shows a variety of waveforms ranging from smooth sine waves to square waves and more rugged bit-reduced forms (see Figure 5.3 for a random selection).

5.2.3 Data Visualisation

For visualisation, the raw waveforms were reduced to a 2-dimensional embedding using the UMAP dimensionality reduction algorithm [17] (see Figure 5.4), which in its Python implementation includes convenient functions for generating interactive scatter plots of the data. The categorisation of the waveforms into subfolders here enabled a simple form of labelling and colour coding of the data points to indicate where the sounds came from. In particular, the algorithm picked out some clusters of sounds, corresponding to square waves in the 'AKWF_bw_squ'-folder (leftmost yellow cluster), some rugged outliers in the 'AKWF_vgame'-folder (rightmost blue

[3] `https://jupyter.org`

Figure 5.4: Visualisation of the waveforms using the UMAP dimensionality reduction algorithm, with points colour coded according to the subfolder they were in.

cluster) as well as some smoother waves from the 'AKWF_0001'-folder (bottom red cluster).

A further step here would be to incorporate corpus exploration tools to be able to also hear the sounds as they are pointed at in the 2-dimensional space. In addition, there are also many other visualisation and clustering tools to apply in this step that would give additional information about the structure of the sound space.

5.3 Building a Latent Audio Space

The next step in the design process was to figure out whether the abstract concept of a latent space synthesizer carried actual potential. Importantly, then, a proof of concept was needed to assess if the current strategy would be a relevant path to explore further.

For the latent space construction, a variational autoencoder model was implemented, inspired by previous research that showed some promising results regarding these models. Building on the works of others in this step reduced the development time significantly from conceptualisation to having a proof of concept ready to experiment with.

5.3.1 Dataset

The training data for the model was based on the dataset of wavetable audio introduced in Section 5.2.1, i.e. a collection of single cycle waveforms, each 600 samples long and 4358 in total. To be able to test the performance of the model, a random 10/90 train-test split of the dataset was generated (i.e. 436 waveforms were randomly set aside as a test set). Furthermore, 200 silent audio samples were added to let the model learn also to recognise silent audio.

The sample size is rather strange in this case in that most wavetables have lengths in powers of 2, normally 256, 512 or 1024 samples. In order not to interfere with the original waveforms through methods of resampling, however, the waveforms were kept at their original size. For better compatibility with input from external sources, though, it would make sense to adjust the input to a more commonly used wavetable size in future developments of the model.

5.3.2 Network Architecture

The neural network architecture for the VAE was adopted in large parts based on Tatar et al. [2], who use a lightweight model trained on short frames of raw audio, which enables faster than real-time inference.

In comparison, Tatar et al. [22] use a similar network architecture but trained on spectral audio, which in the end requires some form of phase reconstruction algorithm to invert the spectrograms to the audio domain (see Section 3.2.1). These phase reconstruction algorithms usually run slower than real-time and therefore inhibit use in real-time sensitive applications.

Specifically, in this case, the input to the network is a single cycle waveform with 600 samples. The second layer expands the input layer into 2048 units which are then encoded into a latent space of 256 dimensions (see Figure 5.5). The decoder part of the network, in turn, reverses this process from the latent space back to an output of 600 samples that constitutes the reconstructed signal.

5.3.3 Model Training

The model was trained for 1000 epochs on a CPU with a learning rate of 0.0001 and a batch size of 32 samples. The KL divergence parameter was set to 0.0001 in the first run, but then increased to 0.001 to improve the regularisation of the latent space and the extrapolations around the latent vectors. In return, the reconstructions became slightly less accurate.

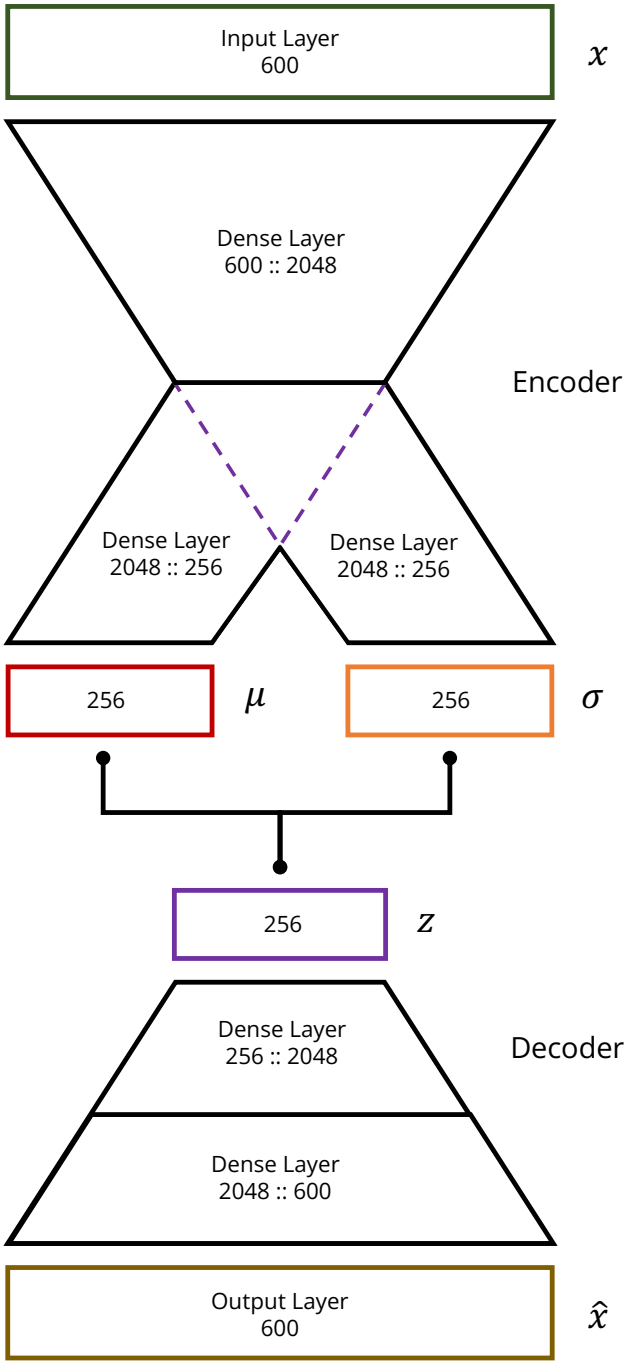

Figure 5.5: Network architecture of the variational autoencoder, adopted from Tatar et al. [2].

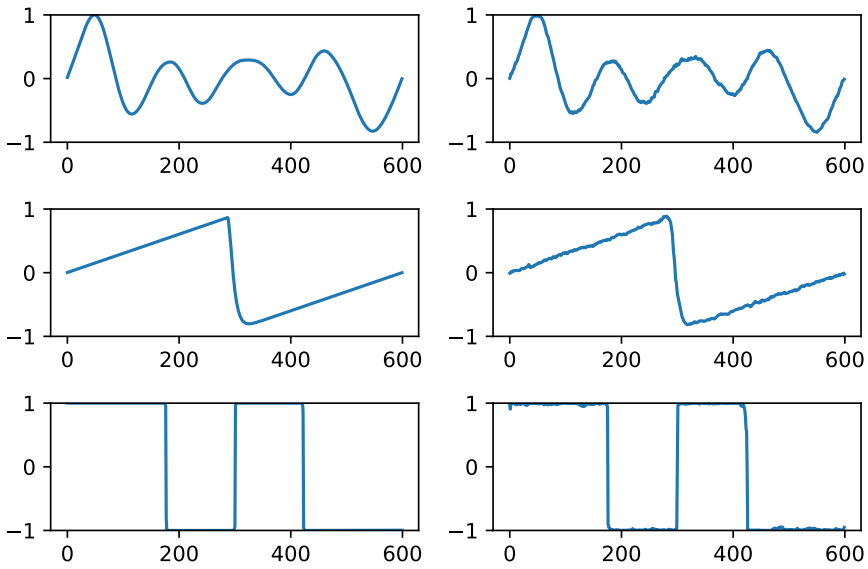

Figure 5.6: Some randomly selected waveforms from the test set (left) and their reconstructions as predicted by the trained variational autoencoder (right).

The training loss function was tracked for each epoch. With more data available, it would have made sense to also generate a validation set and track the validation loss in the training (which helps to identify whether the model over- or under-fits the data). Having other means of evaluating the performance of the model (e.g. through interacting with the synthesizer), however, as much data as possible was reserved for training.

5.3.4 Model Evaluation

At this initial stage, the model performance was evaluated based on the quality of randomly chosen reconstructions. These were plotted and listened to in order to hear the effects of the model on the audio, which included some high-pitched artefacts. Also, interpolations between the test data points were calculated in order to estimate the model's ability to move in-between data samples.

Figure 5.6 shows some randomly selected waveforms and their corresponding reconstructions from the test set.

5.3.5 Model Improvements

Some potential sources of improvement of the model performance were identified, namely regarding the size of the dataset, the model architecture as well as the audio representation used for training.

With regards to alternative datasets, the WaveEdit [55] software includes an online wavetable bank with user generated wavetables in the public domain[4], which was

[4] `https://waveeditonline.com`

explored as a potential alternative or extension to the original dataset. Most of the audio files in that database, however, were not pre-processed into clear single cycle waveforms, which again introduced a problem of modifying the waves to be compatible with wavetable playback. A future implementation would therefore need to consider ways of using general audio samples to extract wavetable data, which would expand the number of available datasets significantly, in turn addressing problems of data sparsity (e.g. using datasets from Freesound etc.).

Secondly, the network architecture of course affects the model's ability to construct a good latent representation of audio. Therefore, an alternative architecture was implemented based on Hyrkas [34], who uses STFT frames of audio as input to a VAE for wavetable synthesis applications. The model takes an input of size 1025 and reduces that to a latent dimension of 16 in two intermediate steps of hidden layers (of size 256 and 64, respectively). In this case, the network architecture was adapted to fit inputs of 600 samples of raw audio. However, the significantly lower dimensionality of the latent audio space made the interpolations more irregular and unpredictable, and therefore the original architecture was preferred.

Finally, the chosen audio representation determines what it is that the model learns to represent and reconstruct. An initial attempt to train the model in the frequency domain was performed, but produced quite low quality reconstructions. This could be due to the specific setup of the model, but also due to the need for phase estimation in the spectrogram conversions to the audio domain. A direction of future work would be to explore the possibilities of utilising frequency representations in ways that produce reliable reconstructions and also work in real-time.

These additional experiments were performed using GPUs accessed remotely through a cloud computing service, which enabled faster iterations and lower training times.

5.4 Pipeline Design

The main question in this project is how to utilise and interface with these latent spaces of audio as part of a real-time playable synthesis application. The next important problem to address was therefore how to move from these data-centred and machine learning focused perspectives to a synthesis tool that could enable and support creative exploration of sound textures.

5.4.1 Latency and Throughput

Approaching sound synthesis from a data-centred perspective in this way illuminated an important trade-off between latency and throughput. The performance of a sound synthesizer relies on carefully crafted digital signal processing lines where small disturbances or delays may cause audible artefacts in the output. A data science project, on the other hand, utilises the power of large datasets to learn statistical models based on batch processing. For large models, then, inference may not be suitable for or even applicable in real-time applications.

Many of the available tools that incorporate machine learning and data exploration tools for audio applications share this problem of being able to generate audio files and samples but not in a particularly interactive or playable way. As an example, Torchsynth [61] is an expansive data-centred synthesis tool that can quickly generate a large number of sounds from a synthesizer that can then be incorporated into machine learning workflows, but the generated sounds come in batches of samples that inhibit real-time exploration and control of the sound output.

The problem here is that audio is inherently temporal. This is also why there is a significant difference in training machine learning models for images vs. audio. Images are static collections of pixel values whereas sounds are signals that evolve through time. The current status on the performance of generative models for audio reflects this discrepancy in that either the inference times are too long for real-time applications (e.g. WaveNet) or the output carries the effects and artefacts from the models necessarily having to cut up the sounds in frames and put them back together again.

This stage of the design process therefore raised questions around how to design interfaces that utilise the power of these high-throughput tools in real-time synthesis applications. Instead of integrating latency and throughput within the same framework, the path in this project was rather to separate the two functions of data retrieval and processing, on the one hand, and audio signal processing, on the other.

5.4.2 Audio Programming Environment

Pure Data (Pd)[5] is a visual and node based programming environment for audio signal processing and interactive music computing. The tool was developed by Miller Puckette in the 1990s [62] to aid programming of signal flows and custom made synthesis engines.

The programs developed in Pure Data as called ”patches”, with reference to the language of modular synthesis where physical synthesis modules are connected through ”patch cables” in more or less complicated arrangements. Similarly, Pd-patches contain different synthesis modules in the form of nodes with (signal) inputs and outputs.

Combining the capabilities of Pure Data patches and Python scripts therefore enabled both handling of real-time audio signal processing as well as tools for processing sound data and training machine learning models. A Pure Data patch can therefore incorporate wavetable- and vector synthesis functionality independently of how the waveforms are generated (i.e. either directly from a dataset or through a latent audio space).

[5] `https://puredata.info`

5.4.3 Open Sound Control

Dividing the different tasks between Python and Pure Data in this way of course requires a form of communication between them.

Open Sound Control (OSC) [63] is a type of protocol that supports communication between networked applications for sound- or media control. The messages can be simple parameter values, but also whole arrays of numbers, which in the current context means that we can send whole waveforms at once from Python to a Pd-patch. In the same way, we can send control messages from the Pd-patch to the Python application, for example to receive new waveforms.

5.4.4 Pipeline Overview

The functionality of the synthesis tool, therefore, was divided in this way between Python scripts and a Pure Data patch, with OSC communication between them.

The Python script in this way provides a link to the original dataset of waveforms, computes cross-fades and interpolations as well as interfaces with the latent space of a trained variational autoencoder.

The role of the Pure Data patch, on the other hand, is to collect and read the waveform output of the Python computations in a wavetable and then channel the audio through different modules such as filters, delays and reverb to add depth and additional texture to the plain oscillator tones.

The whole pipeline design is illustrated in Figure 5.7.

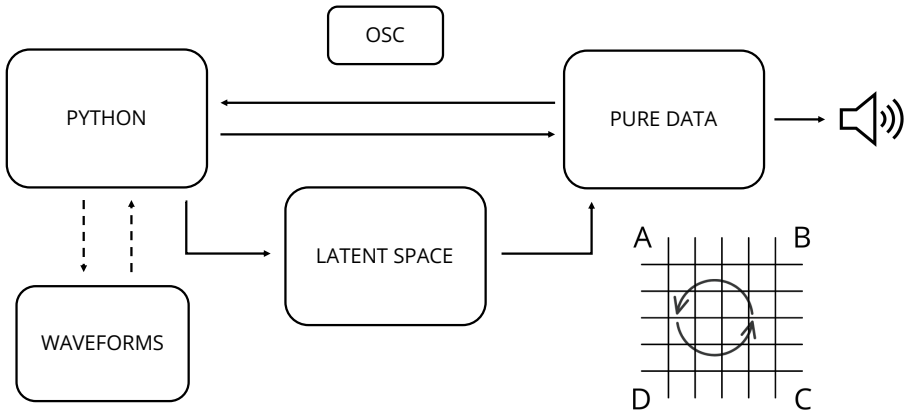

Figure 5.7: Overview of the pipeline for the synthesis tool. A Python script handles waveform selection, interpolations and latent space computations. The waveforms are then sent as OSC messages to a Pure Data patch as part of a wavetable and vector synthesis engine.

5.5 Prototype Design

Finally, the pipeline was to be implemented as an actual prototype, using and interfaceing with the constructed latent audio space. This whole process is outlined below.

5.5.1 Jupyter Notebook

As described, the first step in the prototype design was to set up an interactive Python notebook for loading, modifying and exploring the dataset of waveforms.

5.5.2 Proof of Concept VAE

Secondly, the feasibility of the prototype was evaluated by training a proof of concept VAE on the dataset of audio. As mentioned, the main code for this was adopted from Tatar et al. [2], which also included a Jupyter Notebook tutorial with a pre-trained model to explore the potentials of building latent spaces on raw audio in this way. The code was adjusted to fit the current project and dataset before a model was trained using the default training parameters. The reconstructions were then evaluated in the Jupyter Notebook to get a sense of the audio quality and the particular effects of the model on the audio.

5.5.3 Single Waveform to Pd

At this stage, the audio is only accessible through Python code. As pointed out, this makes sense from a data-centred perspective, but not necessarily as part of a live playable application (although see forms of *live coding* [64]). The next step, therefore, was to establish a connection between Python and Pure Data through OSC communication. To test this out, a simple osc-client was set up in Python that could send waveforms of 600 samples to be received by a Pd-patch and stored in an array. Using methods of wavetable synthesis, this array then formed a basic oscillator with a pure tone as audio output.

5.5.4 Sequence of Waveforms to Pd

The idea of using wavetable synthesis as part of the drone synth was that changing the content of the wavetable would allow us to alter the underlying tone of the oscillator. Therefore, the next step was to send not only a single waveform but a sequence of waves. To this end, an interpolation function was implemented in Python that could cross-fade between two given waveforms, which could then be received and played back by Pd in a sequence, altering the tone of the oscillator.

A 2-dimensional interpolation function was also implemented here that could cross-fade between four source waveforms as inspired by vector synthesis. This also enabled cross-fading by calculating interpolation points following different 2-dimensional morphing shapes (e.g. a circle, triangle or square).

5.5.5 Latent Space Interpolations to Pd

Having formed an interpolation path between waveforms with a simple cross-fade in this way, the following question was whether the prototype could support also latent space interpolations. The same functions were therefore implemented but for interpolations in the (preliminary) latent audio space, to test if there was real-time compatibility in the pipeline. Instead of calculating interpolations as a cross-fade between source waveforms, therefore, the 'in-between' waveforms were generated through the latent audio space. This of course increased the computational demands but could be done in real-time on a laptop CPU.

5.5.6 Synthesis Modules

At this point, the prototype included a simple oscillator with the capacity to change the underlying waveform through different interpolation paths, either by cross-fading between the original waves or between the latent vectors. Aesthetically, however, the sound was still just a simple tone and to make it more 'drone-like' the following development stage focused on the synthesis engine itself, i.e. making the sound more interesting by adding post-processing modules.

Here there is of course plenty of creative freedom in how to design the synthesis engine. The original signal from the wavetable oscillator could be routed in different ways through modules that filter or add effects to the sound. In this case, the synthesis engine was intentionally kept relatively simple so that the sound of the original tone would not be lost in effects, so that the implications of the latent audio space would be heard.

The main synthesis engine was developed as a vector synthesizer with four source waveforms and a wavetable oscillator output (VCO), based on a cross-fade or interpolation between the four source waveforms. These waveforms were received from Python. The oscillator output was then connected to a set of modules in the form of a resonant low-pass/band-pass filer (VCF), a delay, a mixer and a reverb (see Figure 5.8). The filter parameters could be modulated using a low-frequency oscillator (LFO).

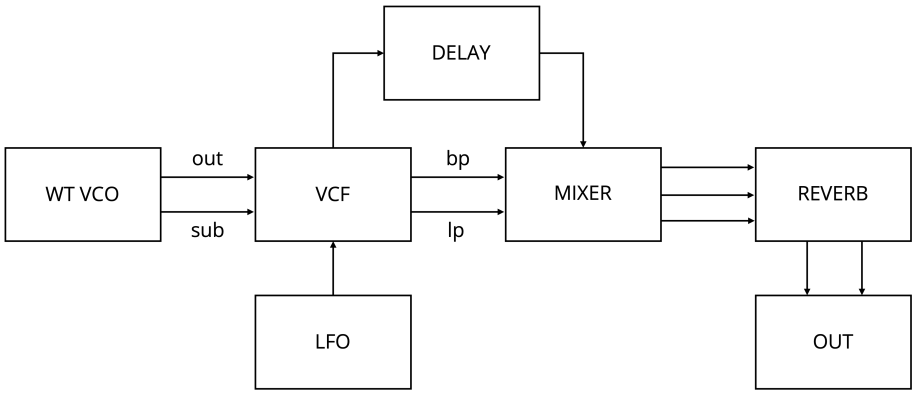

Figure 5.8: Synthesis modules of the Pd-patch.

Worth pointing out here is that any audio programming environment or tool could be used for developing the synthesis engine in this step (i.e. not necessarily Pure Data), as long as it has wavetable compatibility and can receive the waveforms through OSC messages.

5.5.7 Pd to Python

So far, the waveforms were sourced and sent directly from Python, but then modified with post-processing in Pd, which meant that there were two places of interfacing with the synthesizer. Subsequently, in order to gather the control capabilities as part of the same application, a communication line between also Pd and Python was implemented. The Python functions were incorporated as part of a main function that could receive messages from Pure Data regarding sending waveforms and calculating interpolations. The functions were implemented first for the regular waveforms and then for the latent space, with a toggle between the different spaces.

5.5.8 Latent Space Explorations

Next, having the essentials of the pipeline in place, the question of navigating and exploring the latent audio space was emphasised. A purpose with incorporating a latent space as part of the synthesizer is that the latent space itself may open up new trajectories for sound exploration and methods of generating variations on sounds.

Many different strategies for latent space exploration could be conceived and implemented here. In this case, the local neighbourhood of the latent vectors were explored by making a random deviation in the latent space. 'Random' in this context means a Gaussian deviation with a certain mean and variance that was calculated based on the mean and variability of the latent vectors in the whole dataset.

Also, as a second exploration strategy, a feedback loop was established between the output and the input of the model, in order to generate variations on the original waveform by utilising the probabilistic nature of the VAE.

In combination with replacing the waveform completely with a new one, these methods of altering the waveforms was inspired by the principle of exploration vs. exploitation (divergence vs. convergence) that is often employed in research around creativity. This emphasises that creative exploration often functions in the tension between generating completely new ideas and being surprised, on the one hand, and being able to hone in on a particular aesthetic effect and fine tune the details of it, on the other. The exploration of timbre spaces can therefore be aided by these global and local moves within the sonic space.

5.5.9 Refining the Model

At this point, all the parts of the pipeline and the important functionality of the synthesizer were in place, but the latent space was still a rough initial construction that needed refinement. In particular, even though the interpolations were of suffi-

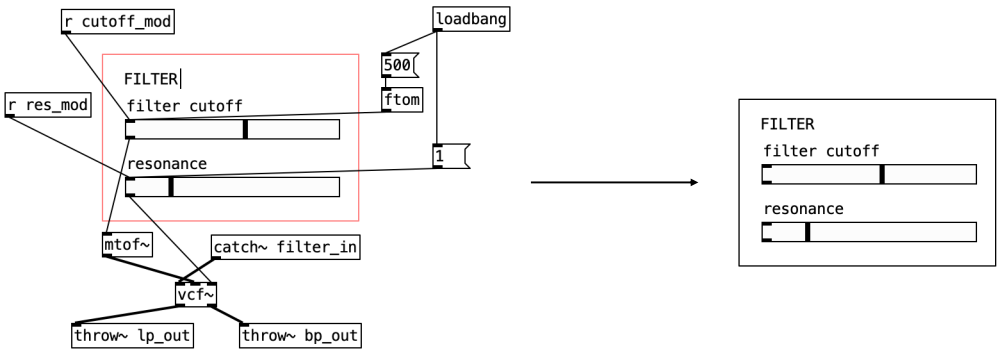

Figure 5.9: An example of how functionality can be embedded in Pure Data as part of a module (in this case a filter) with certain controls exposed to the user.

cient quality, the step of generating extrapolations in the latent audio space revealed some deficiencies in the model's capacity to generate new sound material.

To address these deficiencies, the KL-beta parameter was slightly adjusted, which affects the trade-off in the training between reconstruction and regularisation. Increasing the KL-beta parameter means that the model will put higher emphasis on regularising the posterior distribution to fit that of the latent distribution, in essence making transitions in the latent space more smooth and traversable. This enabled more interesting extrapolations, but also reduced the quality of the reconstructions somewhat.

Additional experimentation with the model settings and architecture was also performed at this stage (see Section 5.3.5), but did not point towards particular improvements in performance. For future work, these paths could be explored more diligently in terms of finding alternative datasets, architectures and audio representations for building the latent audio space.

5.5.10 Graphical User Interface

The final step in the prototype design was to develop the actual user interface. This was done by embedding the internal wiring of the Pd-patch in modules with relevant controls in order to make the tool understandable for the user.

The main design consideration that comes into play here pertains to choices around what is exposed to the user, and what is left hidden or covered within the modules. This will constrain what the user can do in certain ways, but also more clearly outline the relevant points of control in the synthesizer. With this point in mind, the GUI was developed iteratively based on the specific controls and functions that were found particularly expressive in altering the sound textures. Figure 5.9 shows an example of how the functionality of a filter can be exposed to the user in a way that hides the internal wiring and emphasises the relevant points of control (in this case the cutoff and resonance of the filter).

The general layout of the GUI was developed to follow the actual signal flow of

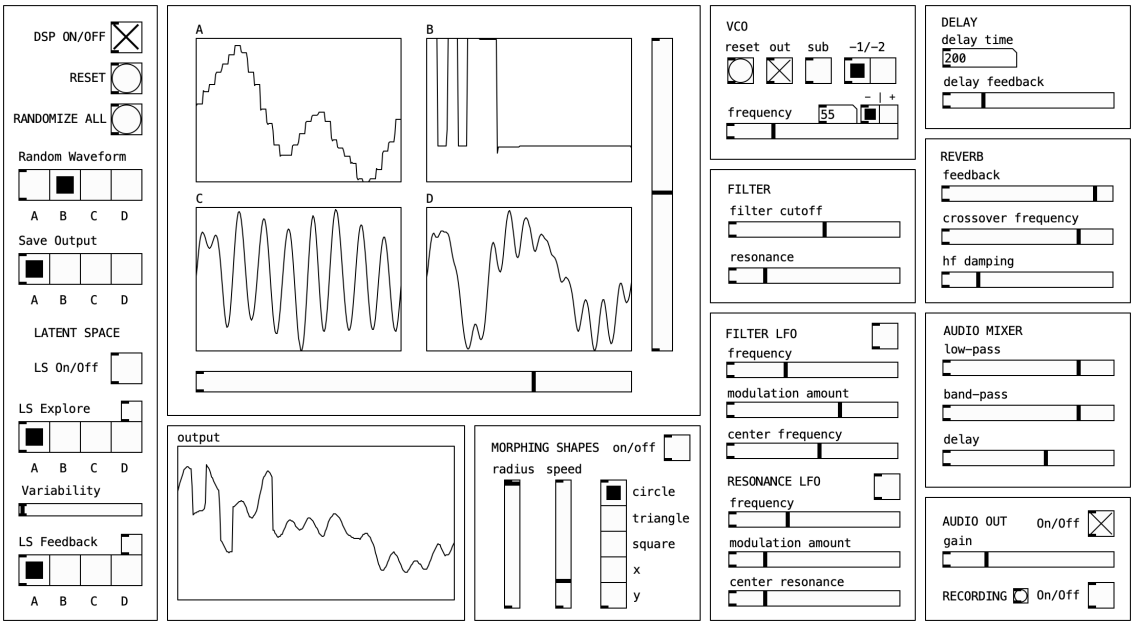

Figure 5.10: Overview of the GUI for the synthesis tool.

audio between the modules, from waveform selection and modification, to vector plane interpolations and finally post-processing modules (see Figure 5.10).

The main functions for communicating with the Python script and sourcing the waveforms were allocated to the leftmost part of the GUI. This part also included the interactivity with the latent space of audio in terms of waveform explorations.

The main visual component of the GUI is the vector synth pad, consisting of a 2x2 grid of the source waveforms and functionality to interpolate between them and modulate the position on the vector grid according to certain morphing trajectories (e.g. a circle, triangle or square). A graphical representation of these morphing shapes imposed on the vector plane would be a useful addition, but was omitted because of time constraints for this iteration of the GUI.

Finally, the post-processing modules were developed and wired together as described in Section 5.5.6.

A final point of imposing design choices in this step that either expand or inhibit the user's agency relates to the fine tuning of control ranges. Some parameter settings were found to produce quite noisy and unreliable output (usually involving large amounts of feedback), which prompted an adjustment of the ranges that could be explored for each parameter. In order not to impose unnecessary constraints on the aesthetic affordances of the synthesizer, however, these considerations were evaluated based primarily on the reliability of the audio output rather than the characteristics of the sounds themselves.

6

Results

The results of the design process is a prototype of a latent vector synthesizer as well as some key considerations that can be extracted from this process.

6.1 Latent Vector Synthesizer

The final GUI for the Pd-patch is divided in three sections (see Figure 6.1):

1. Source waveform selection and modification.
2. Vector synth pad with morphing capabilities.
3. Sound synthesis modules and effects.

6.1.1 Source Waveform Selection and Modification

In the first section, the waveforms for each corner of the vector plane can be selected (A, B, C and D). These are selected randomly from the dataset.

The latent space toggle, in turn, calculates the latent encodings of the source waveforms and then switches to latent space interpolation instead of a simple cross-fade.

Activating the latent space also enables the generation of variations on each source waveform in the form of a local deviation in the latent space. This takes the latent vector and outputs a normally distributed perturbation from the latent vector, where the size of the perturbation can be adjusted.

In addition, the latent space offers the option to run the waveform again through the network in a feedback loop, which will also generate a latent vector close to but not the same as the original vector, due to the probabilistic nature of variational autoencoders.

6.1.2 Vector Synth Pad

The second section of the GUI is the actual vector synth functionality with cross-fading, morphing and interpolation capabilities between the source waveforms, determined by the xy-position in a vector plane. The xy-position can also be modulated through different morphing shapes imposed on the plane, namely a circle,

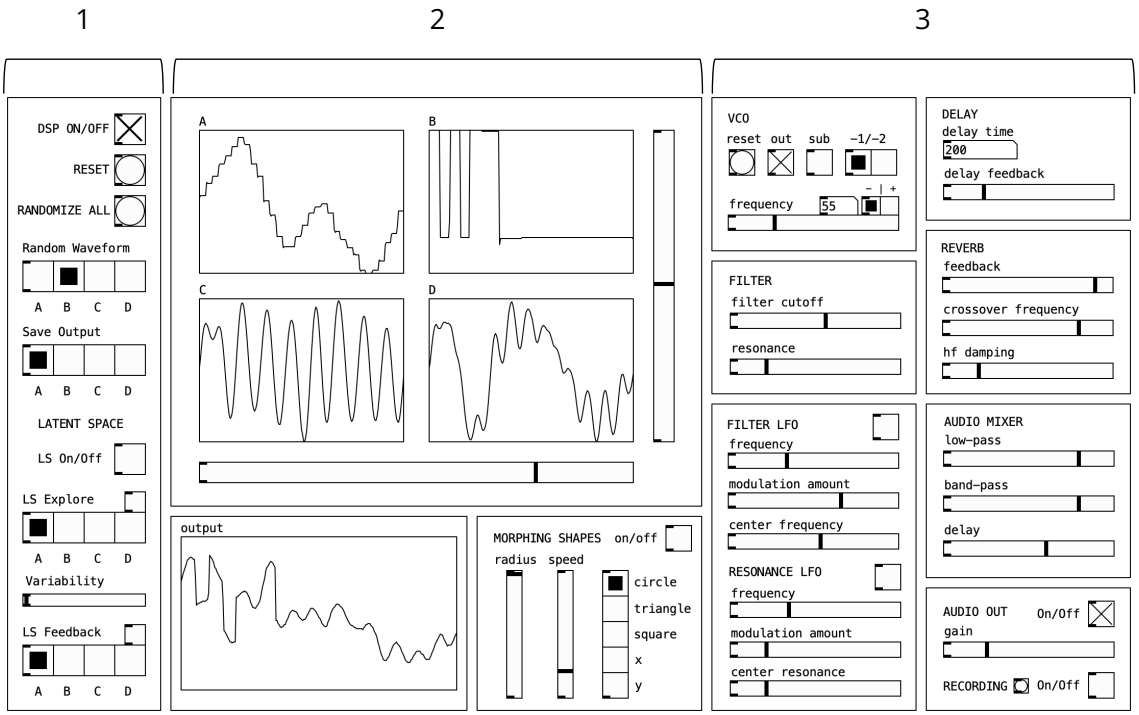

Figure 6.1: The Graphical User Interface (GUI) of the sound synthesis tool in Pure Data, divided in three main sections.

triangle, square or (horizontal/vertical) line. These modulation trajectories come with a speed and a radius parameter.

6.1.3 Sound Synthesis Modules

Finally, the third column of functions offers a few standard sound synthesis modules in the form of an oscillator (VCO), a filter (VCF), a low frequency oscillator (LFO), a delay, an audio mixer and a reverb (see also Figure 5.8).

- The main oscillator is a wavetable (WT) oscillator that reads the output from the vector position, with additional frequency control and a sub oscillator output 1 or 2 octaves down.
- The filter is a resonant low-pass plus band-pass filter that alters the high frequency content of the sound.
- The parameters of the filter can be optionally modulated by an LFO to add movement and texture to the sound.
- The filter output is channelled through a delay line with parameters for delay time and feedback amount.
- Finally, the output passes an audio mixer and a reverb module that adds reverberation to the sound.

6.2 Key Considerations

For future reference, this section summarises some key questions and considerations as part of the design process that may serve as a blueprint for similar projects that explore the use of latent audio spaces in sound synthesis applications.

6.2.1 Musical Task and Aesthetic Framing

What is the situated musical task that the technology is supposed to address? What are the aesthetic expressions that should be afforded?

The process of designing musical interfaces and sound synthesis tools is situated at the intersection of technology and artistic practices. Therefore, being clear on the particular musical context and aesthetic framing of the sound application provides a grounding of technological opportunities in a particular artistic reality. This initial musical and aesthetic framing of course does not limit the use cases of the tool to those particular tasks and expressions, but constrains the subsequent development in important ways.

The latent vector synthesizer, in particular, is developed as a drone synthesizer for purposes of sound exploration and design.

6.2.2 Forming a Sound Space

What sound textures should the model learn to represent? How are these sound textures represented to the model?

Machine learning models will learn what you feed them. Therefore, an initial data selection and exploration phase will put constraints on what sounds can be generated from the synthesis tool. This also includes how the audio is represented to the algorithm, which will determine the coordinate axes of the original sound space (e.g. raw audio, spectrograms or audio features).

The current implementation points to the utility of using short frames of raw wavetable audio to form the original sound corpus, for the benefit of fast model training and inference times.

6.2.3 Building a Latent Audio Space

How is the latent space of audio (iteratively) constructed? How can its structural properties be evaluated?

The process of building latent audio spaces for sound synthesis tasks raises questions around iterative design of network architectures and how to evaluate the model output. As in any prototype design process, the value of being able to quickly set up a proof of concept should not be underestimated, which in the case of training neural networks is significantly aided by having access to GPUs and setting up a structure for iterating through different architecture variations and hyperparameter

settings. The performance of the model variations can be measured quantitatively, but also needs to be evaluated based on the quality and aesthetics of the sound output in relation to the implied musical context.

In this case, an important leverage in the prototype design was building on the works of others in order to set up an initial architecture and evaluate if the abstract ideas bore concrete utility. The model output was then evaluated based on reconstructions, interpolations and extrapolations from a test set of audio.

6.2.4 From Audio Data to Sound Output

How is the machine learning model integrated with audio signal processing in the synthesis tool?

Machine learning models are data-intensive, whereas sound synthesis tools rely on low-latency signal processing. Given that we want to interact with the tool in real-time, therefore, we need to consider how machine learning frameworks can be integrated as part of audio signal processing flows (and vice versa). An important consideration is whether these two domains are divided between separate applications or integrated within the same programming environment.

The pipeline for the latent vector synthesizer relies on communication between a Python script and a Pure Data patch through OSC messages.

6.2.5 Strategies for Latent Explorations

What are the strategies for exploring the latent audio space in the synthesis tool?

A latent audio space is a reordering and restructuring of the original sound space according to a learned latent audio representation. An important consideration, then, is how to utilise the structure of this latent audio space for sound synthesis tasks. Here, the principle of exploration and exploitation comes in handy, which builds on the idea that creativity in musical tasks is supported by finding completely new directions of exploration, but also being able to finely tune and find variations on a current sound texture. Latent audio spaces can support this form of exploration by enabling local and global moves within the sound space.

Strategies for latent space exploration in the current application includes random waveform selection, waveform vector interpolations as well as local (Gaussian) extrapolations around the latent vectors.

6.2.6 Interfacing the Latent

What are the methods of interfacing with the latent audio space in the synthesis tool?

Finally, a sound synthesis tool is only as potent as the user interface allows for. The backend of a machine learning supported sound synthesis engine contains different

control parameters that need to be exposed to a user in order to reveal the affordances of the latent audio space. This interface also provides an important way of evaluating the performance of the model in relation to the musical task at hand.

The functionality of the Latent Vector Synthesizer is embedded within synthesis modules in the Pure Data applications that includes interactive control sliders and buttons exposed to the user.

7

Evaluation

7.1 Collaborative Drone Session

The final prototype was evaluated based on a collaborative drone session with two additional participants, both with extensive experience in developing and utilising musical interfaces for sound exploration and performance. The selection of participants therefore reflected the specific situated task of generating experimental drone based soundscapes for sound exploration and performance.

The purpose of the session was to evaluate the tool and the interface from a first-person perspective in a structured way, but also to receive feedback on how other users approach experimenting with the tool and perhaps use it in ways that extend beyond the particular first-person perspectives of the developer. Finally, the evaluation sought to map out the affordances of the tool in a collaborative setting.

To this end, the tool was evaluated in two instances: first individually one after another (5 minutes each) and then together as a group (15 minutes in total). Each participant downloaded and ran the tool on separate computers but played through the same sound system in a studio lab environment.

After the evaluation session, each participant answered a set of evaluation questions (see Appendix A) aimed to elicit the particular affordances and limitations of the tool and the experience of using it both individually and collaboratively. The reflections were collected in writing after the session to reduce any biases and influences between the participants.

7.2 First-person Evaluation

From a first-person point of view, the final prototype in the end provided a tool for synthesising rich and varying drone-based soundscapes. The vector synth with morphing capabilities, in particular, offered ways to continuously but slowly alter the waveforms in order to change the sound textures. In addition, the subsequent filter, delay and reverb added further characteristics and depth to the sounds.

The experience of using the latent space interpolations was not particularly different from cross-fading between the original sounds, and the latent extrapolations usually

introduced some distortion to the original waveforms. In spite of this, the latent space explorations and feedback options were used quite extensively to add dynamics and harmonics to the sounds. In combination with filtering and modulation, this added variability and contrast to the smoother tones available. The tool was also kept relatively simple in terms of additional functionality and effects in order to be able to hear the effects of the latent interpolations and extrapolations of the sounds. Otherwise, with enough added effects and reverb, any sound can be a drone sound.

Importantly, exploring the tool both as a performer and developer gave a deeper understanding of the relationship between the back-end of the tool and its effect on the output. Playing the instrument therefore provided a lens onto the internal wiring of the technology, in particular around the structure of the latent space and how the model made inferences.

Inhabiting both these roles also drew forth a trade-off between designing tools for purely personal use and for sharing also with others. In particular, from a development point of view, there was a tendency to hide functionality, on the one hand, but also safe-proof the tool for failure, on the other hand, in favour of better navigability and user-friendliness of the interface. In both cases, the possible sounds and interesting textures that could be generated were restricted in some way, while aspects like navigability and reliability were improved.

Finally, the evaluation session put emphasis on the contrast between individual sound exploration and collaborative performance. Since the tool was developed mainly for individual and personal use, the collaborative affordances were not intentionally built into the system. This points to the importance of considering the musical task at hand when designing musical interfaces. A collaborative performance setting, for instance, requires a different level of immediacy and expressivity in the interaction, in order to be able to entice and respond to other musicians, than say a sound design task within a studio setting.

The experience of playing together with others, however, invited each participant to delineate their own expression in using the tool, in order to occupy contrasting positions in the collaboratively generated soundscape.

7.3 Participant Feedback

As mentioned, after the evaluation session, each participant answered questions around their experience using the synthesis tool. The qualitative results of this feedback is summaries below.

7.3.1 Sound Aesthetics

The participants described the generated sound textures as containing a richness in harmonics and overtones, with a bias towards pure tones and waveforms. This reflects the intended application domain of the instrument as a drone synth that can generate sustained tones and slowly varying sound textures. There was also an

experienced variability in the sounds, ranging from ”subtle and smooth” at times to ”harsh and noisy” at other times, attributed to gradual variations in the underlying waveforms as well as to filtering and modulation of the sound output.

7.3.2 Learnability

The evaluation session only permitted a short amount of time for the participants to familiarise themselves with the interface beforehand, which made it difficult to methodically discover all the functions, but an intuitive and familiar labelling of the parameters and a meaningful grouping of similar functions helped navigation. The sequential signal flow from left to right was also reflected in the layout of the interface in a clear way.

7.3.3 Functionality

The waveform mixing and interpolation was found useful for creating long term variation in the sound, whereas the post-processing modules were complementarily explored to generate short-time variations and glitch effects. One participant felt particularly drawn to the filtering and delay options to stir up the sound with rhythmic beatings and noise.

The latent space functionality was considered particularly useful for adding harmonics and texture to the source waveforms, which allowed for movement from smooth waves to richer and harsher sound textures.

7.3.4 Frustrations

All participants experienced some frustrations with working with the mouse to control the sliders, which could be mitigated by mapping a separate tangible interface or controller to the parameters of the interface. In particular, the mouse control did not allow for sufficient precision and dexterity in tuning the parameter values.

7.3.5 Collaboration

In the individual sessions, each participant had their own unique way of using the interface, which brought variability and complementarity to the collaborative session in that the different sonic streams could occupy different layers in the performance. In this way, each participant contributed to the combined soundscape through their personal ways of approaching the tool. Although the tool permitted a certain range of expressivity, the cohesiveness of the generated sound-world also made it difficult ”to provoke or push certain gestures of others to create a more dynamic improvisation”.

Finally, the process of downloading and installing the tool was considered smooth and it was also easy to get started making sounds, which lower the barrier of entry for collaborative uses of the tool and sharing it with others in different ways.

7.3.6 Application Domains

In its current form, the tool was conceived to work well in the context of generating drone textures that could be used as sound material for musical composition or in further audio processing. As to speculations on future applications and improvements of the instrument, one participant longed for a controller for a more expressive control of the output and also thought about ways to spatialise the sound. Additionally, the ability to input live recordings or pre-recorded files in buffers was put forth as a future development.

The ease of installation and navigability in combination with cpu-compatibility also makes the synthesizer available for distribution to others as an example of how to utilise latent spaces as part of a sound synthesis tool, without much necessary prior knowledge of these methods.

8

Discussion

The reserch contribution of this project is a particular design process leading up to a prototype of a latent vector synthesizer that was evaluated from a first-person perspective as well as through feedback from other users. In this chapter, general and specific learnings from this process are emphasised, in particular with regards to the different perspectives on what a "good" performance means in this context.

8.1 Model Performance

First, we may summarise some model specific learnings and outcomes from this project. Here, interestingly, the GUI of the synthesis tool in some ways worked as a visual and aural lens onto the internal workings of the variational autoencoder, that brought forth insight as to how the model handles its inputs. There are of course many ways to measure the performance of a model quantitatively, but for sound synthesis applications, in particular, the value in actually *seeing* and *hearing* the effects of model inferences on the output should not be underestimated.

A first thing to point out is that the model can be incorporated as part of a real-time audio synthesis tool because of a lightweight architecture that takes short samples of raw audio as input. This specifically means that the model organises the waveforms based on similarity in the raw audio domain. In other words, the model uses a notion of similarity that compares the wavforms sample by sample. The question remains, then, if this is a particularly good measure of *timbral* similarity, i.e. if the constructed latent space is in fact comparable to a form of "timbre space" (see Section 3.1).

Since the implementation of the interface included both the option to simply cross-fade between the waveforms as well as the option to interpolate in the latent space, a visual comparison could be made between the two respective methods that indicated a clear similarity between them, i.e. that what the variational autoencoder most likely learns is actually to cross-fade between waveforms (see Figure 8.1). An interesting comparison, therefore, would be to replace the raw audio model with a spectral model (or using alternative audio representations) to delineate if the two different forms of interpolation are perceptually distinguishable, i.e. if interpolation in the spectral domain is in some ways closer to motion in a timbre space. This, however, would require efficient methods for spectrogram inversion back to the audio

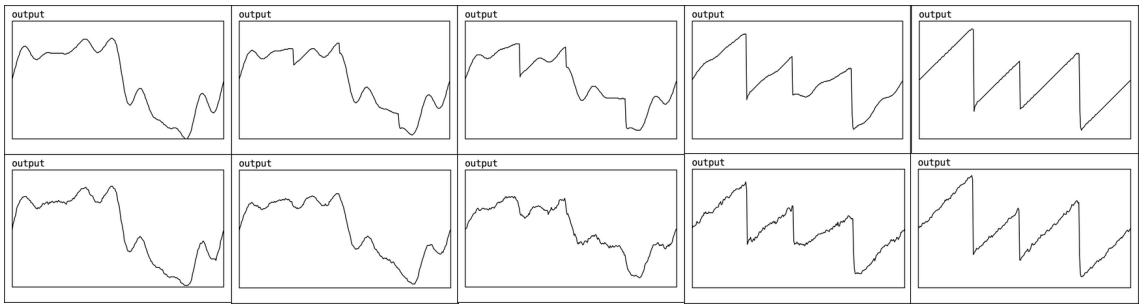

Figure 8.1: Comparison of a cross-fade (top) versus a latent space interpolation (bottom) between two waveforms with an interpolation factor α ranging from 0 to 1 in steps of 0.2. The latent space interpolation here share characteristics with the linear cross-fade.

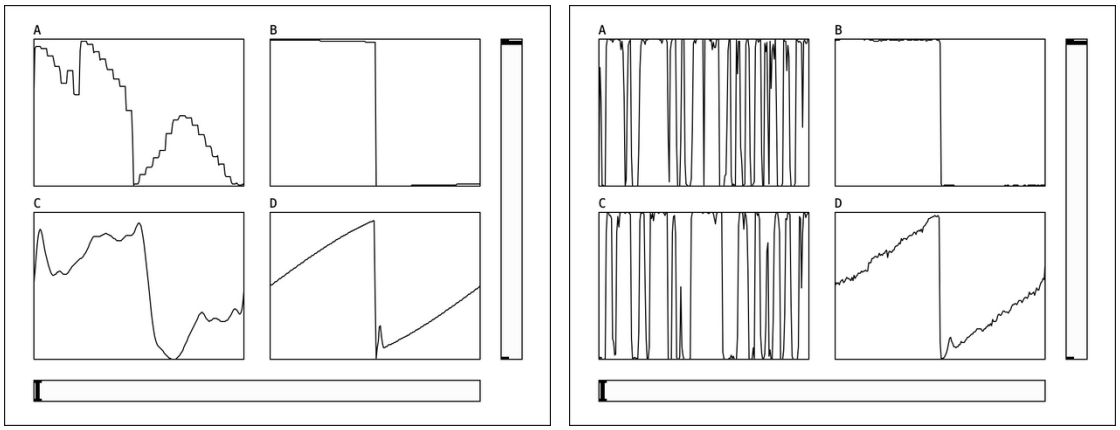

Figure 8.2: Comparison of the original waveforms (left) and the waveforms after many iterations of latent space perturbations (right). Waveforms A and C, in this case, deteriorate into harsh waveforms of alternating values between -1 and 1, whereas waveforms B and D remain stable.

domain, which in most cases limits the real-time applicability of the model.

A second point to address is the sometimes fairly poor performance of the model in terms of extrapolations and perturbations around the latent vectors. The stability of these latent explorations, however, depends to a large extent on the shape of the original waveform. Figure 8.2 shows the transformed waveforms after many iterations of latent space perturbations, which indicates that the waveforms either deteriorate into a quite noisy shape of alternating values between -1 and 1, or remain quite stable even after several iterations. The explanation for this could be many things, but most probably a too sparse and asymmetric dataset in combination with perhaps over-fitting the model to the data that was available. With more data available, the dataset could be split into a training set, a test set as well as a validation set to better track the ability of the model to generalise beyond the training data.

Interesting to note is also that the most stable waveforms seem to be the ones that are square shaped (although this would require a more systematic study). In a

sense, the square wave is on the one extreme of internal waveform variability between samples (the other one being complete silence), which adds to the observation that waveforms often transition into this kind of shape. Square waveforms were also specifically picked out and clustered by the UMAP algorithm in the data exploration part (see Section 5.2.3).

In this particular project, the model could be trained rather conveniently on a laptop with a CPU, but increasing the size of the dataset or the model to improve model performance would realistically require access to GPUs in order to maintain feasible training times. This is also because the hyperparameters of the model generally need tuning based on several runs of the training process. This, in turn, points to an important drawback in using machine learning tools for musical tasks in that the setup of the architectures and sometimes quite extensive computations needed to train the models inhibit quick and iterative prototyping of interfaces and feedback on the particular aesthetic output of the models.

8.2 Aesthetics of Failure

As Dahlstedt points out [7], the use of AI tools in creative tasks points to an important tension between optimisation and exploration. Essentially, engineering works primarily in the mode of optimisation, making systems and processes function more efficiently as to some set performance standard. Art and creativity, on the other hand, hinges on exploration and expanding the field of what is possible, making use of tools and concepts in ways that reveal and extend the affordances in new directions. In fact, in order to be truly creative, we have to let go of set performance standards and goals. Since most AI algorithms are trained to optimise, the question is how we can utilise the computational affordances of these tools for exploratory purposes?

Here we can draw a distinction also between a form of ”logic of success” in contrast to an ”aesthetics of failure” [65]. Whereas the engineer is focused on solving problems and optimising for a specific goal, the artist is oriented rather towards finding interesting problems and exploring the implications of those problems. In the history of electronic music, this tension plays out repeatedly in that the engineer (or designer) first develops a tool to solve a particular problem for an intended user. The artist then takes that tool and uses it in unintended ways, pushes it beyond its apparent affordances, for artistic effects. In a digital context (and the context of industrial society), for example, there is an aesthetic turn from the foreground to the background of technology itself, utilising sonic artefacts like noises, glitches and buzzes that constantly surrounds us for artistic purposes in a form of ”post-digital” aesthetics [65].

Interestingly, the evaluation study of the vector synthesis tool pointed precisely in this direction, that the eventual failure of the latent space to preserve the smoothness and consistency of the waveforms introduced harmonics to the sounds that were positively received as adding richness and texture and contrast to the more predictable sonics of the original waveforms. From an engineering perspective, then,

the model ”failed” to learn a ”good” latent representation that ”performs well” also on extrapolations from data points, but from an artistic perspective, the model ”performed well” in that it generated interesting sound textures.

Secondly, there were certain functions and signal flows in the Pure Data patch that were not sufficiently safe-proofed for failure from a development perspective. For instance, altering the delay time caused audible cracks and scratches in the output and ramping up the feedback too much made the sound textures crackle in unexpected ways. From the perspective of user interface design, these artefacts were deficiencies that the user of the interface should not be exposed to. However, the evaluation rather indicated how the deficiencies were appropriated for aesthetic noise and glitch effects that added crucial intrusions and dynamics to the sustained backdrop of drones that can otherwise become quite monotonous. In fact, one of the participants mentioned that ”my initial instinct was to see how far (and in what ways) I could push the interface to generate a wide palette of sounds”, which delineates a specific strategy when exploring the affordances of a new interface to push it in all sorts of directions until it fails.

This points to a cliché yet important lesson: success is not about not failing, but rather about failing in interesting ways.

8.3 Evaluating Interaction Design Research

The methodology of this project centres around first-person approaches in HCI and research through design (RtD). Although these approaches have generated significant contributions and supported further theories and investigations in HCI research, there is a lack of clarity around how to evaluate contributions produced by these methods systematically. To address these concerns, Zimmerman et al. [66] put forth a set of criteria for evaluating interaction design research focusing on process, invention, relevance and extensibility, respectively. In the following sections, these four criteria are related to the current design implementation.

8.3.1 Process

Research through design aims to generate knowledge and contributions through explicating a particular design process. The process itself, therefore, is a key part of the research output and should be evaluated accordingly. In addition, due to the situated and contextual nature of any design process, conventional expectations on reproducibility have to be addressed. That is, attempts to reproduce a design process would start from different initial conditions that would in turn influence the direction of the process and lead to different results. Rather than focusing on the reproducibility of the research output by way of a specific design process, therefore, RtD should be evaluated based on the rationale behind the process and its methodological underpinnings.

The design process, in this case, consisted of different design phases centred around ideation, data selection and exploration, model development, pipeline design and a

final prototype development. A project log was kept to track and delineate these phases of the process and the important decisions that were made.

The ideation phase imposed an important aesthetic framing on the project and a grounding in a particular sound aesthetics that influenced the subsequent investigation and implementation of synthesis methods as part of the prototype. Conversely, the aesthetic framing was in turn influenced by developments in sound and music technology. This phase of the design process therefore pointed to the influence of artistic inspirations on technological development as well as the influence of novel technologies on artistic imagination. Furthermore, it pointed to the importance of creative constraints in the development of musical interfaces as well as getting clear on the forms of musical tasks that the tool is sought to address.

The data selection and exploration, in turn, was a crucial next step in order to get a sense of the sounds contained in the dataset that in turn would inform the sonic potentials of the final prototype and the structure of the latent audio space. In addition, building a latent space based on the audio data illustrated the potentials of using variational autoencoders in this way and provided a proof of concept for the prototype design. Subsequently, this pointed to the importance of latency considerations in audio processing, in contrast to the high-throughput of data processing tools, and the relevance of real-time compatibility in the pipeline.

Finally, the prototype and GUI was implemented from a first-person RtD perspective that enabled fast iterations and evaluations of design decisions. The evaluations were based on the aesthetic output of the synthesis tool as well as the performance of the model in audio reconstruction and interpolation. In this phase, also, the importance of being able to switch between different roles as developer became apparent.

8.3.2 Invention

The latent vector synthesis tool constitutes a particular invention and novel intervention in how it integrates latent audio spaces with vector synthesis techniques. The project synthesises perspectives on machine learning tools, sound synthesis techniques, musical interface design and the sonic arts to address the situated musical task of generating transforming and sustained drones.

An extensive literature review provided background for the project and situated it within a context and lineage of related research. The history of this lineage can be traced back to the initial attempts to construct timbre spaces of perceptually related sounds, followed by forms of audio feature extraction- and dimensionality reduction techniques, ending in modern variations of generative models that build latent spaces of audio.

The initial explorations of related research, however, were rather biased towards a specific category of generative models, namely variational autoencoders, that perhaps excluded a more open exploration of what methods would best address the task of exploring and navigating latent audio spaces. The literature on dimensionality reduction algorithms and feature extraction tools here deserve a closer investigation

in their potential to aid organisation of sounds.

8.3.3 Relevance

In the context of research through design, the research expectations shift from aspirations towards validity and truth towards instead what is real and relevant [66]. A design implementation cannot be valid for all cases, but only relevant as part of a particular context and framing. In addition, design implementations point towards the future in wanting to bring the world towards some desired state by way of designing artefacts and prototypes.

As mentioned, situating design research within the context of artistic uses of new technologies points towards a tension between optimisation and exploration. On the one hand, design research has to be goal oriented, in order to provide a clear contribution within a particular research field. On the other hand, a design implementation centred on artistic expression has to be open ended to allow for discovery of unseen possibilities and affordances. The question, then, is how research within this context can offer relevant contributions and avoid "to be a self-indulgent, personal exploration that informs the researcher but makes no promise to impact the world" [66]? In this particular case, what is the relevance of exploring new sound synthesis methods? What is the preferred state of the world that the design implementation is thought to bring forth?

As pointed out, novel technologies may both inhibit and support creative expression through its affording and limiting functions. In the context of machine learning techniques and AI for musical tasks, therefore, there is a relevance in examining the aesthetic implications and potentials of using these tools that are perhaps otherwise left unseen and/or undiscovered. Also, as the history of music technology teaches us, novel tools condition musical expression and sometimes even pushes the boundaries of what we consider to be music. Examining the feedback loop between tools and artistic practises may therefore open up for completely new forms of sonic possibilities and musical affordances that would be left unseen within fields of research centred on performance and optimisation.

8.3.4 Extensibility

Lastly, a necessary feature of doing research is that others can build on the research results and leverage it for further discoveries. To this end, the current project is made available open source on GitHub for others to use an/or develop [67]. Also, the design process itself was documented and described in order to outline some general design considerations and affordances of using these technologies for purposes of sound exploration and design. These general learnings are exemplified by this process and can be brought forward into future work and design problems.

8.4 Ethics and Biases

Ethical considerations during the project relate primarily to the use of interview material from other participants in the evaluation study. In this case, the participants were made aware that their answers and feedback would be used as part of the evaluation of the project. The results of the evaluation were also presented in a way that hinders attribution to specific participants. As to data collection and storage, the answers to the questions were collected by email, but otherwise stored offline only.

The main purpose of the evaluation was to receive feedback from a few additional users within the same research context on the extended utility of a personalised musical interface. Therefore, due to the small group size and niche focus of the study, many perspectives were left out that would shine light on the broader usability of the interface. Within this particular situated context, however, participants were selected with a variation in musical background and interests in order to explore and expand the particular affordances of the tool.

The project supports FLOSS (Free/Libre & Open Source Software) and encourages the use and further developments of the tool by others. To this end, the source code of the project is made available open source through GitHub [67]. In addition, attributions have been made to tools and libraries developed by others that were utilised in this project. Furthermore, to mitigate copyright concerns and ensure compliance with intellectual property rights, an audio dataset in the public domain was selected.

9

Conclusion

This thesis set out to explore the opportunities and limitations of utilising latent audio spaces for purposes of creative sound exploration and design. To this end, a latent vector synthesizer was conceptualised, combining a variational autoencoder model, able to generate short samples of audio, with the idea of a vector synthesizer that interpolates between sonic textures using four wavetable source oscillators. This provided a promising framework for interacting with and exploring the generated content of latent audio spaces in real-time.

In contrast and addition to research on the performance and development of these generative models, the main focus of the project was rather to explicate a first-person research through design process that in its particularity traced out some technological and aesthetic affordances of these latent spaces of audio. The artistic framing of the project was inspired by the novel perspectives on music sparked by the developments in electronic and digital computer technology of the 20th century. Rather than focusing on conventional acoustic instrument sounds and systems of notation, the aesthetic focus was rather within the field of the sonic arts and pointed to the development of a drone synthesizer able to generate continuous and slowly transforming streams of sound.

To compliment the first-person perspectives of the design process, the prototype was evaluated as part of a collaborative session with two other sound artists and researchers, where the synthesis tool was explored individually as well as in collaboration. In this way, the aesthetic affordances of the tool could be more clearly mapped out in the different and complementary ways the participants approached the tool.

The latent space functionality of the synthesis tool enabled interpolations between given audio samples as well as local explorations around latent vectors to generate variations in the source waveforms. This provided a proof of concept of a strategy for latent vector exploration of sounds, but also brought some particular characteristics to the sounds in the form of added harmonics that were utilised for aesthetic effects.

From a model performance point of view, however, the latent space was rather unstable in extrapolations from given sound textures. The reason for this might have been the sparsity of the dataset or the model architecture itself. Therefore, further investigations are needed on this point in order to fully map out the utility

of latent audio spaces in these kinds of sound applications.

The use of machine learning models in artistic practices is positioned in the tension between engineering optimisation and artistic exploration, and how technology and aesthetics influence and inform each other in crucial ways. By methods of research through design, therefore, we can navigate this tension in order to uncover the latent potentials of new technologies.

9.1 Future Work

Since the main focus of this work was directed towards providing a proof of concept of an interface utilising latent audio spaces in this way, less focus was directed towards the selection, curation and exploration of a suitable sound dataset as well as experimentation with different model architectures and hyperparameter tunings. These are important considerations that deserve further investigation. In addition, different audio representations and similarity measures will affect how the model organises the sounds. Also, the data selection process greatly influences the aesthetic output of the synthesis in that it constitutes the basis for waveform selection and training of the latent audio space. A different dataset might therefore affect the model performance, but also provide different sonic possibilities.

The idea of using a vector synth with a vector plane in this way also points to further considerations around tangible control interfaces for exploration of that plane of sound textures. A simple extension would be to map the xy-control to a tangible xy-pad as well as mapping parameter controls to a MIDI control interface. This was also expressed as a desired feature of the interface in the evaluation. The importance of expressive gestural control of sound synthesizers, therefore, should be stressed here. In theory, this framework for synthesis could implement also interpolation in a higher dimensional control space or subset of the latent space through interpolation between more sound sources.

Furthermore, the fundamental problem that is addressed here is how to organise sounds by timbre or texture. Variational autoencoders are one method for constructing organised and regularised audio spaces, but there are of course many techniques that may be suitably applied in this context. A simple extension of the tool to this end could implement different dimensionality reduction techniques and clustering algorithms to organise the waveforms by similarity by other means, in order to support functions of waveform selection and variation.

Bibliography

[1] Wikimedia Commons. Vector synthesis diagram, 2007. [Online; accessed 2023-05-20]. URL: https://commons.wikimedia.org/wiki/File:VectorSynthesisDiagram.svg.

[2] Kıvanç Tatar, Kelsey Cotton, and Daniel Bisig. Sound design strategies for latent audio space explorations using deep learning architectures. In *Proceedings of Sound and Music Computing 2023*, 2023.

[3] Christopher Small. *Musicking: The meanings of performing and listening*. Wesleyan University Press, 1998.

[4] Thor Magnusson. *Sonic Writing: Technologies of Material, Symbolic, and Signal Inscriptions*. Bloomsbury Publishing, 2019.

[5] Pierre Schaeffer. *Traité des objets musicaux*. Seuil, nouv. éd edition edition.

[6] Jean-Claude Risset. Fifty years of digital sound for music. In *Proceedings of the 4th Sound and Music Computing Conference (SMC'07)*, Lefkada, Greece, July 2007.

[7] Palle Dahlstedt. Musicking with algorithms: Thoughts on artificial intelligence, creativity, and agency. In Eduardo Reck Miranda, editor, *Handbook of Artificial Intelligence for Music: Foundations, Advanced Approaches, and Developments for Creativity*, pages 873–914. Springer International Publishing, 2021. doi:10.1007/978-3-030-72116-9_31.

[8] Trevor Wishart. *On sonic art*. Contemporary music studies, 12. Harwood Academic, Amsterdam, 1996.

[9] Edgard Varèse and Chou Wen-chung. The liberation of sound. *Perspectives of New Music*, 5:11, 1966.

[10] John M. Grey. Multidimensional perceptual scaling of musical timbres. *The Journal of the Acoustical Society of America*, 61(5):1270–1277, 1977. URL: http://asa.scitation.org/doi/10.1121/1.381428, doi:10.1121/1.381428.

[11] David L. Wessel. Timbre space as a musical control structure. *Computer Music Journal*, 3(2):45, 1979. URL: https://www.jstor.org/stable/3680283?

origin=crossref, doi:10.2307/3680283.

[12] Matthew D. Hoffman and Perry R. Cook. Feature-Based Synthesis: Mapping Acoustic and Perceptual Features onto Synthesis Parameters. In *Proceedings of the 2006 International Computer Music Conference (ICMC 2006)*, New Orleans, Louisiana, USA, November 2006. Michigan Publishing. URL: http://hdl.handle.net/2027/spo.bbp2372.2006.111.

[13] Stefano Fasciani and Lonce Wyse. Adapting General Purpose Interfaces to synthesis Engines using Unsupervised Dimensionality Reduction Techniques and inverse Mapping from Features to parameters. In *Non-Cochlear Sound: Proceedings of the 38th International Computer Music Conference (ICMC 2012)*, Ljubljana, Slovenia, September 2012. Michigan Publishing. URL: http://hdl.handle.net/2027/spo.bbp2372.2012.087.

[14] Stefano Fasciani and Lonce Wyse. Vocal control of sound synthesis personalized by unsupervised machine listening and learning. *Computer Music Journal*, 42(1):37–59, April 2018. URL: https://direct.mit.edu/comj/article/42/1/37-59/94655, doi:10.1162/comj_a_00450.

[15] Stefano Fasciani. Interactive computation of timbre spaces for sound synthesis control. In *ICMA Array, vol 2016, Special Issue: Proceedings of the 2nd International Symposium on Sound & Interactivity*, pages 69–78, Singapore, August 2015. URL: https://journals.qucosa.de/array/article/view/2528, doi:10.25370/array.v20152528.

[16] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[17] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. 2018. URL: https://arxiv.org/abs/1802.03426.

[18] Y. Bengio, A. Courville, and P. Vincent. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, August 2013. URL: http://ieeexplore.ieee.org/document/6472238/, doi:10.1109/TPAMI.2013.50.

[19] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *Proceedings of the 2nd International Conference on Learning Representations (ICLR 2014)*, Banff, AB, Canada, April 2014. URL: http://arxiv.org/abs/1312.6114.

[20] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392, 2019. URL: http://dx.doi.org/10.1561/2200000056, doi:10.1561/2200000056.

[21] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adver-

sarial networks, 2014. arXiv:1406.2661.

[22] Kıvanç Tatar, Daniel Bisig, and Philippe Pasquier. Latent Timbre Synthesis: Audio-based variational auto-encoders for music composition and sound design applications. *Neural Computing and Applications*, 33(1):67–84, January 2021. URL: https://link.springer.com/10.1007/s00521-020-05424-2, doi:10.1007/s00521-020-05424-2.

[23] Philippe Esling, Axel Chemla–Romeu-Santos, and Adrien Bitton. Generative timbre spaces: regularizing variational auto-encoders with perceptual metrics. In Mathew Davies, Aníbal Ferreira, Guilherme Campos, and Nuno Fonseca, editors, *Proceedings of the 21st International Conference on Digital Audio Effects (DAFx-18)*, Aveiro, Portugal, September 2018.

[24] Philippe Esling, Naotake Masuda, Adrien Bardet, Romeo Despres, and Axel Chemla-Romeu-Santos. Flow synthesizer: Universal audio synthesizer control with normalizing flows. *Applied Sciences*, 10(1):302, 2019. URL: https://www.mdpi.com/2076-3417/10/1/302, doi:10.3390/app10010302.

[25] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *CoRR*, abs/1601.06759, 2016. URL: http://arxiv.org/abs/1601.06759, arXiv:1601.06759.

[26] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders, 2016. arXiv:1606.05328.

[27] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio, 2016. arXiv:1609.03499.

[28] Lamtharn (Hanoi) Hantrakul, Jesse Engel, Adam Roberts, and Chenjie Gu. Fast and flexible neural audio synthesis. 2019. URL: https://archives.ismir.net/ismir2019/paper/000063.pdf.

[29] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis, 2018. arXiv:1802.08435.

[30] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. Ddsp: Differentiable digital signal processing, 2020. arXiv:2001.04643.

[31] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders, 2017. arXiv:1704.01279.

[32] Lamtharn Hantrakul and Li-Chia Yang. Neural wavetable: a playable wavetable synthesizer using neural networks, 2018. arXiv:1811.05550.

[33] Siyuan Shan, Lamtharn Hantrakul, Jitong Chen, Matt Avent, and David Trevelyan. Differentiable wavetable synthesis, 2022. arXiv:2111.10003.

[34] Jeremy Hyrkas. Wavaetable synthesis. In *Proceedings of the 15th International Symposium on CMMR*, CMMR 2021, page 263–268, Tokyo, Japan, 2021. CMMR 2021 Organizing Commitee.

[35] Jean-Claude Risset and David L. Wessel. Exploration of timbre by analysis and synthesis. In Diana Deutsch, editor, *The Psychology of Music (Second Edition)*, Cognition and Perception, pages 113–169. Academic Press, San Diego, CA, USA, second edition edition, 1999. URL: https://www.sciencedirect.com/science/article/pii/B9780122135644500068, doi:https://doi.org/10.1016/B978-012213564-4/50006-8.

[36] Stephen McAdams. The perceptual representation of timbre. In Kai Siedenburg, Charalampos Saitis, Stephen McAdams, Arthur N. Popper, and Richard R. Fay, editors, *Timbre: Acoustics, Perception, and Cognition*, pages 23–57. Springer International Publishing, 2019. doi:10.1007/978-3-030-14832-4_2.

[37] Meinard Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer International Publishing, Switzerland, 1st edition, 2015. doi:10.1007/978-3-319-21945-5.

[38] Gerald Kaiser. *A Friendly Guide to Wavelets*. Modern Birkhäuser Classics. Birkhäuser Boston, 2011.

[39] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933. URL: https://psycnet.apa.org/record/1934-00645-001, doi:10.1037/h0071325.

[40] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964. doi:10.1007/BF02289565.

[41] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. URL: https://www.science.org/doi/abs/10.1126/science.290.5500.2319, doi:10.1126/science.290.5500.2319.

[42] John M. Chowning. The synthesis of complex audio spectra by means of frequency modulation. *Computer Music Journal*, 1(2):46–54, 1977. URL: http://www.jstor.org/stable/23320142.

[43] Axel Chemla–Romeu-Santos. *Manifold representations of musical signals and generative spaces*. PhD thesis, Università degli Studi di Milano – Sorbonne Université, Milan, Italy, 2020.

[44] Joseph Rocca. Understanding variational autoencoders (vaes), 2019. [Online; accessed 2023-05-20]. URL: https://towardsdatascience.com/

understanding-variational-autoencoders-vaes-f70510919f73.

[45] Kristina Höök, Baptiste Caramiaux, Cumhur Erkut, Jodi Forlizzi, Nassrin Hajinejad, Michael Haller, Caroline C. M. Hummels, Katherine Isbister, Martin Jonsson, George Khut, Lian Loke, Danielle Lottridge, Patrizia Marti, Edward Melcer, Florian Floyd Müller, Marianne Graves Petersen, Thecla Schiphorst, Elena Márquez Segura, Anna Ståhl, Dag Svanæs, Jakob Tholander, and Helena Tobiasson. Embracing first-person perspectives in soma-based design. *Informatics*, 5(1), 2018. URL: https://www.mdpi.com/2227-9709/5/1/8, doi:10.3390/informatics5010008.

[46] Andrés Lucero, Audrey Desjardins, Carman Neustaedter, Kristina Höök, Marc Hassenzahl, and Marta E. Cecchinato. A sample of one: First-person research methods in hci. In *Companion Publication of the 2019 on Designing Interactive Systems Conference 2019 Companion*, DIS '19 Companion, page 385–388, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3301019.3319996.

[47] Christopher Frayling. Research in art and design. *Royal College of Art Research Papers*, 1(1):1–5, 1993. URL: https://researchonline.rca.ac.uk/384/.

[48] John Zimmerman, Erik Stolterman, and Jodi Forlizzi. An analysis and critique of research through design: Towards a formalization of a research approach. DIS '10, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1858171.1858228.

[49] William Gaver. What should we expect from research through design? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, page 937–946, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2207676.2208538.

[50] Rodrigo Medeiros, Filipe Calegario, Giordano Cabral, and Geber Ramalho. Challenges in designing new interfaces for musical expression. In Aaron Marcus, editor, *Design, User Experience, and Usability. Theories, Methods, and Tools for Designing the User Experience*, pages 643–652. Springer International Publishing, 2014.

[51] Marcelo Mortensen Wanderley and Nicola Orio. Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI. *Computer Music Journal*, 26(3):62–76, 09 2002. doi:10.1162/014892602320582981.

[52] Palle Dahlstedt. Dynamic mapping strategies for expressive synthesis performance and improvisation. In Sølvi Ystad, Richard Kronland-Martinet, and Kristoffer Jensen, editors, *Computer Music Modeling and Retrieval. Genesis of Meaning in Sound and Music*, pages 227–242, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[53] Harry Sword. *Monolithic Undertow: In Search of Oblivion*. White Rabbit, 2021.

[54] Robert Bristow-Johnson. Wavetable synthesis 101, a fundamental perspective. In *Audio Engineering Society Convention 101*, Nov 1996. URL: http://www.aes.org/e-lib/browse.cfm?elib=7379.

[55] Synthesis Technology. Waveedit (1.1), 2018. URL: https://synthtech.com/waveedit/.

[56] Sound on Sound. Sequential prophet vs [retrozone], 2001. [Online; accessed 2023-05-20]. URL: https://www.soundonsound.com/reviews/sequential-prophet-vs-retrozone.

[57] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, 2015.

[58] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[59] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[60] Kristoffer Ekstrand. AKWF FREE (waveform samples). https://www.adventurekid.se/akrt/waveforms/adventure-kid-waveforms/. Accessed: 2023-03-06.

[61] Joseph Turian, Jordie Shier, George Tzanetakis, Kirk McNally, and Max Henry. One billion audio sounds from GPU-enabled modular synthesis. In *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020)*, Vienna, Austria, September 2021.

[62] Miller S. Puckette. Pure data: another integrated computer music environment. In *Proceedings of the Second International Computer Music Conference*, pages 269–272, San Francisco, 1996. International Computer Music Association.

[63] Wikipedia contributors. Open sound control — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Open_Sound_Control&oldid=1130824556, 2023. [Online; accessed 2023-05-20].

[64] Wikipedia contributors. Live coding — Wikipedia, the free encyclope-

dia. https://en.wikipedia.org/wiki/Live_coding, 2023. [Online; accessed 2023-06-19].

[65] Kim Cascone. The aesthetics of failure: "post-digital" tendencies in contemporary computer music. *Computer Music Journal*, 24(4):12–18, 2000. URL: http://www.jstor.org/stable/3681551.

[66] John Zimmerman, Jodi Forlizzi, and Shelley Evenson. Research through design as a method for interaction design research in hci. CHI '07, page 493–502, New York, NY, USA, 2007. Association for Computing Machinery. doi:10.1145/1240624.1240704.

[67] David Högberg. Latent vector synthesis, 2023. [Online]. URL: https://github.com/david-hogberg/latent-vector-synthesis.git.

A

Evaluation Questions

1. What was your overall impression and experience of experimenting with the sound synthesis tool?
2. How would you describe the sound textures that were generated? Could you identify any specific aesthetic affordances?
3. How was the experience of learning and navigating the interface?
4. Did you feel control and agency over the generated sound material?
5. Did you resort to a particular function of the tool?
6. In what ways (if any) did you use the latent space functionality of the interface? Did you find it useful?
7. Were there any moments of frustration while using the tool? How could it be improved?
8. How would you imagine using the tool as part of a creative workflow?
9. How would you imagine using the tool in a performance setting?
10. How was the experience of using the tool in collaboration with others?
11. How did you experience the aesthetic affordances of this tool in collaboration with others?