CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Designing Loss Functions for Learning Sound Timbre Audio Representations in Variational Autoencoders

A Loss Function Perspective for Learning Audio Synthesis Representations

Master's thesis in Computer science and engineering

Ipek Korkmaz

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Designing Loss Functions for Learning Sound Timbre Audio Representations in Variational Autoencoders

A Loss Function Perspective for Learning Audio Synthesis Representations

Ipek Korkmaz

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Designing Loss Functions for Learning Sound Timbre Audio Representations in Variational Autoencoders
A Loss Function Perspective for Learning Audio Synthesis Representations
Ipek Korkmaz

© Ipek Korkmaz, 2025.

Supervisor: Kivanc Tatar, Department of Computer Science and Engineering
Examiner: Simon Olsson, Department of Computer Science and Engineering

Master's Thesis 2025
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 2025

Designing Loss Functions for Learning Sound Timbre Audio Representations in Variational Autoencoders
A Loss Function Perspective for Learning Audio Synthesis Representations
Ipek Korkmaz
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This study investigates the effect of audio-related loss functions, audio feature extraction methods, and the addition of a synthesis layer on the reconstruction quality and latent space organization of variational autoencoders (VAEs). Three different experiments were conducted to address these questions. The first experiment suggests that different audio-related loss functions do not lead to significant differences in performance, aside from requiring different training durations. Additionally, in the second experiment, while adding a synthesis layer does not substantially improve reconstruction quality, it generally helps the model converge faster during training. Finally in the third experiment, which focuses on feature extraction methods, Mel-Frequency Cepstral Coefficients (MFCC) performed slightly better in terms of reconstruction quality. These findings can potentially guide architectural choices for effective audio representation learning in VAE-based models.

Keywords: timbre representation, audio feature extraction, generative models, variational autoencoder.

Acknowledgements

I would like to thank my supervisor, Kivanc Tatar, for his continuous support and valuable guidance through insightful feedback during this process.

I am also deeply grateful to my family and friends for their support and encouragement throughout this journey.

Ipek Korkmaz, Gothenburg, 2025-06-18

Contents

List of Figures

List of Tables

List of Acronyms

ADSR	Attack Decay Sustain Release
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DBCV	Density-Based Clustering Validation
DDSP	Differentiable Digital Signal Processing
CQT	Constant-Q Transform
FAD	Fréchet Audio Distance
FID	Fréchet Inception Distance
FFT	Fast Fourier Transform
LFO	Low-Frequency Oscillators
LogMSL	Log Magnitude Spectral Loss
LogSLP	Log Magnitude Spectral Loss with Phase
LMSL	Linear Magnitude Spectral Loss
LSLP	Linear Magnitude Spectral Loss with Phase
MFCC	Mel-Frequency Cepstral Coefficients
MSE	Mean Squared Error
PCA	Principal Component Analysis
R-VAE	Rhythm-Variational Autoencoder
SCL	Spectral Convergence Loss
SCLP	Spectral Convergence Loss with Phase
STFT	Short-Time Fourier Transform
t-SNE	t-Distributed Stochastic Neighbor Embedding
UMAP	Uniform Manifold Approximation and Projection for Dimension Reduction
VAE	Variational Autoencoder

1

Introduction

Vibration in a physical environment produces sound [1], and consistent, recurring sound patterns form rhythm [2] which is a fundamental aspect of music. Nowadays, rhythm and sound design are frequently produced using electronic music instruments. One common electronic musical instrument for production and sound design is synthesizer [3], which can be either analog, digital, or a hybrid of both. The fact that every calculable function can be applied by digital synthesizers makes them popular due to limitless capacity of sound production [1]. A typical example of a digital synthesizer is frequency modulation (FM), which is a set of methods for artificial sound creation [4].

The rich sounds generated by synthesizers can be used to form rhythms. An example is Patternarium [5], which is a set of rhythms generated by computers. It uses evolutionary computation to generate unique rhythms. This is done by combining two patterns in each generation to produce new rhythms for every offspring.

1.1 Background

Designing and shaping timbres through synthesizers is a part of sound design process which is an artistic practice. However, the use of synthesizers needs background knowledge due to their complexity and large number of parameters [6]. These can be time consuming, as producers must first learn how to use the synthesizer. Moreover, synthesis parameters are not always connected to the musical aspects and qualities. As exploring vast sound spaces through synthesis parameters takes time, creating a timbre feature space that users can intuitively explore and that reveals relationships between different timbre characteristics can help musicians and producers focus more on their artistic expression rather than technical adjustments.

Deep generative modeling can enable musicians and producers to explore an abstract space of sounds in a more organized way, where similar sounds are positioned closer together. For instance, in this project, a machine learning model, specifically a Variational Autoencoder (VAE), is trained using synthesizer parameters to create a feature (latent) space of timbral characteristics. In this way, musicians and producers can easily generate new sounds in the style they desire by navigating this latent space. However, there are various ways to introduce audio characteristics to the model, such as different audio-related loss functions and audio feature extraction

methods. Therefore, this project focuses on finding the best approach for creating a timbre space using a VAE.

1.2 Purpose of the Project

This thesis aims to explore how generative models can be used for audio synthesis parameter modeling, focusing on timbre representation. A VAE model is developed and trained to learn a latent space of timbral features using differentiable audio synthesis techniques.

This project focuses on analyzing how different design choices impact the quality of audio reconstruction and the latent structure of timbre. These choices include experimenting with various audio loss function formulations and exploring different audio feature extraction methods.

The primary objective of a machine learning model is to minimize the value of a loss function. The loss function defines how a model optimizes its parameters to best fit the data. By evaluating the loss, the model gains insight into its performance, specifically, how adjustments to its parameters can lead to improved outcomes. Consequently, the choice of loss function significantly influences the model's ability to learn from data. Therefore, modifying the formulation of the loss function and the audio feature extraction methods used in the loss function, can have a significant impact on model performance.

The VAE model used in this project is trained using the parameters of a digital synthesizer. However, these parameters are not necessarily directly related to audio features. For instance, some synthesizer parameters may influence the sound in a nonlinear manner, while others may have a linear effect. Therefore, the purpose of employing audio feature extraction methods is to provide the model with deeper insight into the characteristics of the audio signal.

The goal of this project is to enable the model to develop a deep understanding of audio timbre features and their interrelationships by leveraging different audio-related loss functions. Additionally, the project aims to effectively encode and reconstruct audio features using a VAE. Achieving these objectives has the potential to support more intuitive and expressive sound design workflows in the future.

1.2.1 Research Questions

The project aims to answer following research questions:

- **RQ1:** How do different audio-related loss function formulations influence the reconstruction accuracy and organization of the latent space in variational autoencoders for audio synthesis?
- **RQ2:** What is the influence of different feature extraction methods on the effectiveness of variational autoencoders in representing timbre for audio parameter reconstruction?

The first research question focuses on designing and evaluating different loss function formulations to identify the most effective approach for integrating audio features into the model. Specifically, it investigates how these varying definitions influence reconstruction accuracy and the organization of the latent space. In this context, the study assesses the contribution of not only the classical loss components of the VAE, but also audio-specific loss terms, in order to enhance the model's performance.

The second research question aims to identify the most effective audio feature extraction method for capturing timbral characteristics of sound. Particularly, it explores how different representations of audio, derived from various feature extraction techniques, influence the reconstruction quality within the latent space of the VAE. Therefore, this research question seeks to determine which feature extraction methods enable the VAE to learn more essential and musically useful representation of sounds.

1.2.2 Contributions

This project aims to contribute to the field of deep learning for music by developing and analyzing a VAE-based model for audio parameter reconstruction. Specifically, it investigates how different audio-related loss function formulations and audio feature extraction methods affect the model's reconstruction accuracy and the structure of its latent space, including aspects such as cluster formation and continuity. Additionally, the VAE model employed in this project is relatively lightweight compared to commonly used models in the audio synthesis literature, making it more environmentally friendly and computationally efficient.

This research may benefit stakeholders such as musicians, sound designers, and researchers by providing deeper insights into the timbral characteristics of sound.

1.3 Limitations

The most significant limitation of the project was the restricted timeframe. Due to time constraints, existing libraries such as TorchSynth were utilized for sound generation and synthesis. As a result, the digital synthesizer used in this project was not developed from scratch but was based on pre-existing implementations.

2

Theory

This chapter describes important concepts related to the project. Section 2.1 provides essential parts of a synthesizer. Section 2.2 covers important audio feature extraction methods concepts for the project. Section 2.3 explains autoencoders, with a focus on variational autoencoders. Section 2.4 describes latent space and how it can be visualized. Section 2.5 explains a clustering algorithm. Finally, Section 2.6 describes methods for evaluating the quality of both clusters and generated audio.

2.1 Synthesizer Components

Synthesizers typically consist of various components that work together to produce sound. This section explains some of the most common components, including Voltage-Controlled Oscillators (VCOs), Voltage-Controlled Amplifiers (VCAs), and time-variant controllers such as Attack, Decay, Sustain, and Release (ADSR) envelopes, as well as Low-Frequency Oscillators (LFOs).

2.1.1 Voltage-Controlled Oscillator

The essential sound generation component in a synthesizer is the oscillator. A digital oscillator can produce waveforms in various shapes, such as sine, triangle, square, and sawtooth waves. These waveforms are generated at a specific frequency, which determines the pitch of the sound. Controlling the pitch with an arrival voltage signal is typically managed by a Voltage-Controlled Oscillator (VCO) [7].

A simplified illustration of a VCO is shown in Figure 2.1. This diagram illustrates that the VCO takes a control voltage as input and produces a corresponding frequency (audio signal) as output. The level of the control voltage affects the pitch of the sound: higher voltages result in high-pitched sound, and lower voltages result in low-pitched sound.

2.1.2 Voltage-Controlled Amplifier

One of the primitive components of a synthesizer is amplifier that regulates signal's amplitude or level [8]. A common amplifier is a Voltage Controlled Amplifier (VCA), which uses control voltage to estimate its gain [9].

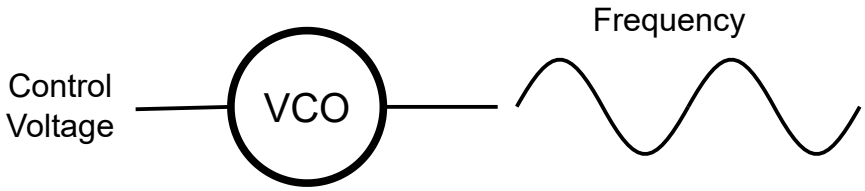

Figure 2.1: Simplified illustration of a VCO.

A simplified illustration of a VCA is shown in Figure 2.2. This diagram demonstrates that the VCA takes both a control voltage and a frequency (audio signal) as inputs and outputs the same frequency with a modified amplitude. The control voltage determines the amplitude of the output: higher voltages result in louder sounds, while lower voltages produce quieter sounds.

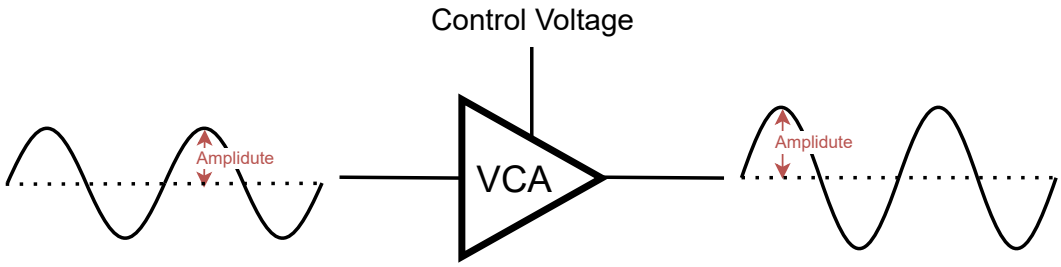

Figure 2.2: Simplified illustration of a VCA.

2.1.3 Time-Variant Controllers

The synthesis components responsible for shifting the sound over time are called time-variant controllers, also referred to as envelope generators. They do not generate any sound themselves, so they are not audible, but they affect the sound. Therefore, the output of time-variant controller is referred to as control signal [8].

Commonly used time-variant controllers in synthesizers include attack, decay, sustain, and release (ADSR) and low-frequency oscillators (LFO).

2.1.3.1 Attack, Decay, Sustain, and Release

A common method of synthesizing sound is to change the envelope of the sound. Envelope expresses the shift in sound throughout its duration [10]. One of the procedures of synthesizing sound using envelope generators is changing ADSR parameters; attack, decay, sustain, and release. Description of these parameters from [11] as follows: Attack is the time period from 0 to highest amplitude. Following the attack phase, the decay is the time period that amplitude decreases from highest amplitude to sustain level. Unlike other parameters, the sustain is not the a period, but the amplitude level that remains constant. Release is the time period during which the amplitude level decreases to 0, starting from the release of the key. An example of ADSR is shown in a waveform in Figure 2.3.

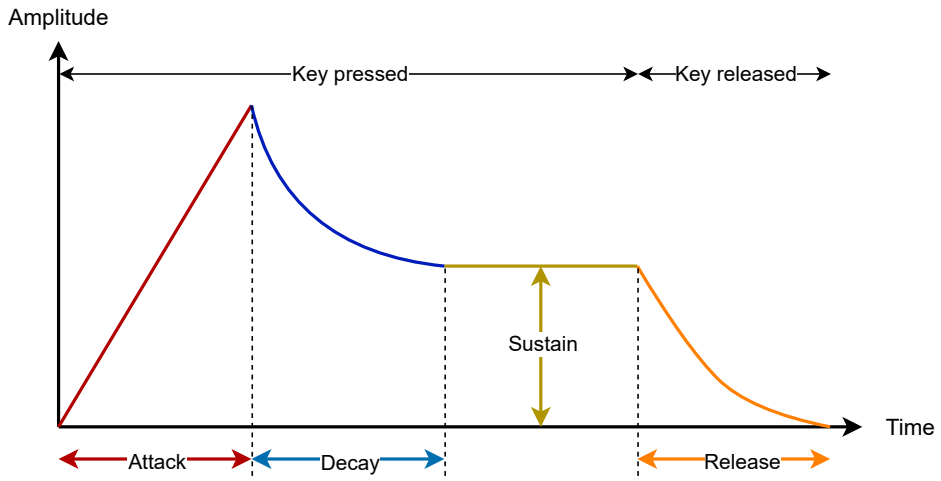

Figure 2.3: An envelope showing ADSR parameters.

2.1.3.2 Low-Frequency Oscillators

A Low-Frequency Oscillator (LFO) adjusts parameters of other modules such as oscillators, filters, or amplifiers based on its control signal. Although the LFO outputs a waveform, its frequency is below the range of human hearing, so it is not directly audible. However, its effect is perceptible through the changes it causes in the sound, such as vibrato or tremolo [8].

A simplified illustration of a LFO is shown in Figure 2.4. This diagram shows that an LFO operates similarly to a VCO but generates much lower frequencies. It takes a control voltage as input and produces a low-frequency signal as output.

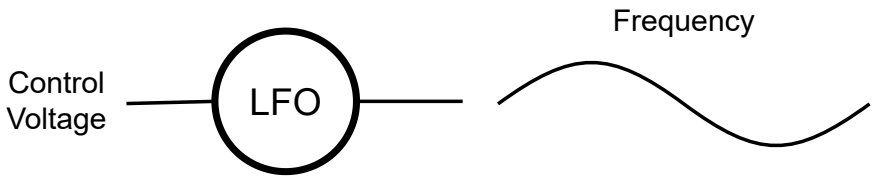

Figure 2.4: Simplified illustration of a LFO.

2.2 Audio Feature Extraction Methods

The audio feature extraction covers detecting and emphasizing the most notable and distinguishing attributes of a signal. A sufficient feature resembles characteristics of a signal in a more intense and concise form [12].

There are three fundamental types of audio features: time-domain, frequency-domain, and time-frequency domain features. Time-domain feature extraction is the earliest and most basic method used for analyzing and classifying audio signals. Subsequently, frequency-domain techniques were developed to study the spectral characteristics

and spectrum of audio. Later, time-frequency domain features were introduced, combining both momentary and spectral information to give richer representation of audio signals [12].

This section introduces some of the most commonly used time-frequency feature extraction methods, including the Short-Time Fourier Transform (STFT), Mel-Frequency Cepstral Coefficients (MFCC), and Constant-Q Transform (CQT).

2.2.1 Short-Time Fourier Transform

The frequency components that form initial signal are uncovered by the Fourier transform, which converts a time-domain signal into its frequency-domain representation [13]. The discrete Fourier transform of a time series can be effectively estimated using the Fast Fourier Transform (FFT) to convert it into the frequency-domain [14].

Dennis Gabor (1946) [15] developed the adjusted version of the Fourier transform, nowadays called the short-time Fourier transform (STFT), in order to recover the masked time information [13]. The sliding-window description of a signal is also provided by STFT [16].

A spectrogram is a visualization of STFT where time is expressed by horizontal axis, frequency is expressed by vertical axis, and the color of the figure is the dimension that shows the value of spectrogram of a specific frequency at a specific moment [13]. Spectrograms of various signals are shown in Figure 2.5, illustrating how the frequency composition of each signal changes over time.

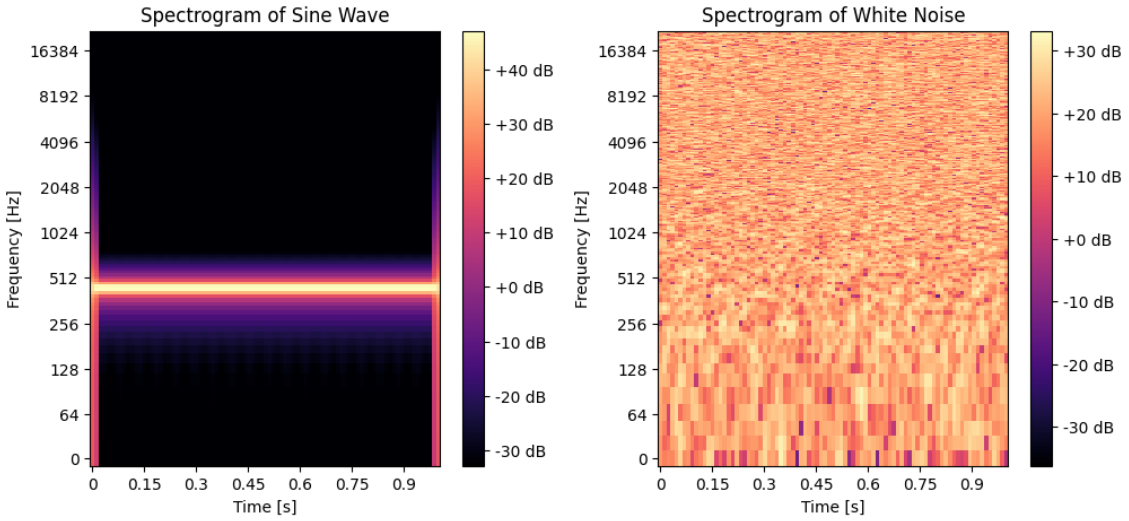

Figure 2.5: The spectrogram of the sine wave is on the left, and the spectrogram of the white noise is on the right.

2.2.2 Mel-Frequency Cepstral Coefficients

The Mel scale is base upon sound perceiving range of human [17]. Mel-Frequency Cepstral Coefficients (MFCC) is utilized STFT and the Mel scale to display a sound's

spectral information. This method is often utilized particularly preprocessing of voice signal [18]. Librosa [19], a specialized python library for signal analysis of audio and music, has a module for implementing MFCC.

Musical timbral features are often displayed using Mel-scaled representations [19]. Timbre is a feature of sound by which a person can detect the difference between two sounds even if their loudness and pitch are the same [1]. Therefore, this feature makes each sound from different instruments unique, even when played at the same note and volume.

2.2.3 Constant-Q Transform

The Constant-Q Transform (CQT) converts time-domain signals into the time-frequency domain by dividing the signal into geometrically spaced frequency bins. The sharpness or selectivity of the frequency components is characterized by the Q-factor, which remains constant across all frequency bins in the CQT. As a result, lower frequencies exhibit higher frequency resolution, while higher frequencies are represented with improved temporal resolution [20].

There are solid musical and perceptual justifications for using the CQT as a feature extraction method. In Western music, notes are spaced geometrically, which aligns with the technique of CQT. Furthermore, human auditory perception operates similar to constant-Q frequency resolution, making CQT a good representation of audio signals for humans [20].

2.3 Autoencoders

Autoencoders [21], a type of neural network, were originally designed to recreate their input. They were developed to represent the data in an informative manner, serving as a form of unsupervised learning. There are different ways autoencoders can be used to learn useful representations of data. To achieve this, diverse types of autoencoder can be fused or altered to build new models for a variety of applications. Some example usages of autoencoders include data generation, clustering, and anomaly detection [22].

The autoencoders usually consist of three parts: encoder, latent space (sometimes referred to as code), and decoder. The encoder processes the input and transforms it into a representation within the latent space, which captures essential features of the input in a reduced-dimensional form. The decoder then uses this compact and informative representation to reproduce the original input [23]. The representation of an autoencoder is shown in Figure 2.6.

The encoder transforms the input $\mathbf{X}$ into a latent representation $\mathbf{Z}$ using weights and biases, as shown in Equation 2.1. Here, $\mathbf{W}$ is the weights and b is the bias [24] and f denotes the activation function. The decoder then recreates an output $\hat{\mathbf{X}}$ using $\mathbf{Z}$ [24].

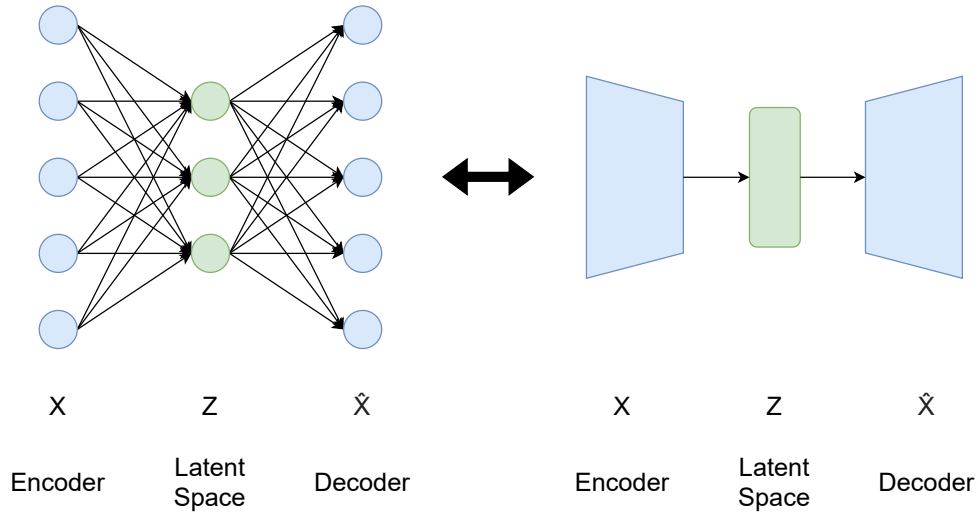

Figure 2.6: (a) An autoencoder network contains a single layer in the encoder and decoder. (b) How an autoencoder is usually illustrated.

$$\mathbf{Z} = f(\mathbf{W}\mathbf{X} + b). \tag{2.1}$$

The loss is computed using the standard L2 loss function [24], as shown in Equation 2.2. It calculates the sum of the squared differences between the original data points x and the reconstructed data points $\hat{x}$.

$$L = ||x - \hat{x}||^2. \tag{2.2}$$

Oussidi and Elhassouny (2018) [25] noted that although autoencoders effectively learn the distribution of input data and encode it into the latent space, the distribution of the latent space itself is still unknown, and autoencoders are not inherently designed to generate new data. As a result, when an autoencoder is used as a generative model, it samples arbitrarily from its latent space, which can lead to poor-quality outputs. This occurs because the generated data does not follow the true distribution of the input data. VAE was designed to solve this problem by learning the distribution of the latent space [25] for generative purposes. The following section focuses on this variation of the autoencoder.

2.3.1 Variational Autoencoders

VAEs use a Bayesian approach to learn the input data's probability distribution [23]. They were proposed to find a mapping between the data distribution $p(x)$ and the latent distribution $p(z)$, as illustrated in Figure 2.7.

VAE aims to estimate the generator $p_\theta(x)$ which represents the likelihood of a sample $x \sim \mathcal{X}$. It uses Bayes' rule to compute this probability, as shown in Equation 2.3 [26].

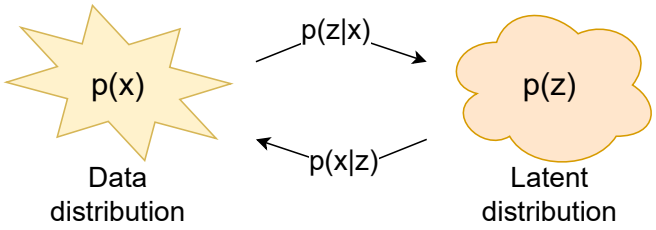

Figure 2.7: The mapping that the VAE tries to understand between data distribution and latent distribution.

$$p_\theta(x) = \frac{p_\theta(x,z)}{p_\theta(z|x)} = \frac{p_\theta(x|z)p_{\mathcal{Z}}(z)}{p_\theta(z|x)} \quad \text{for } z \sim \mathcal{Z}. \tag{2.3}$$

The goal is to maximize this likelihood by optimizing the parameter θ. However, estimating the maximum likelihood using Equation 2.3 is not feasible: although calculating $p_\theta(x|z)$, the conditional probability of a data instance given a latent variable, is uncomplicated, the reverse, $p_\theta(z|x)$ is intractable. This issue is particularly significant in deep generative models, which are typically nonlinear and non-reversible [26].

VAE employs approximation of the intractable posterior $p_\theta(z|x)$ using a tractable family of parameterized probability distributions. This approach, known as variational inference, is used to address the problem of intractability. In this way, sampling from the distribution and computing probabilities becomes more manageable. The encoder estimates the parameters of this distribution using the approximate posterior, as shown in Equation 2.4, where ψ represents the neural network's weights that outputs the approximate posterior's parameters [26]. A simplified illustration of a VAE is presented in Figure 2.8.

$$q_\psi(z|x) \approx p_\theta(z|x) \quad \text{where } \psi \in \mathbb{R}^k. \tag{2.4}$$

The encoder in vanilla autoencoders functions similarly to q_ψ, but it is deterministic, and therefore produces a single point in the latent space. In contrast, the encoder in a VAE outputs a probability distribution, represented as $q_\psi(z|x)$ [26].

VAEs are also referred to as deep latent variable models because they aim to learn a mapping between latent vectors and data instances using deep neural networks [26].

The approximate posterior $q_\psi(z|x)$ is introduced as a manageable substitute problem for the true posterior $p_\theta(z|x)$. To make this solution feasible, two main objectives must be achieved: maximizing the approximate likelihood and minimizing the error in the approximate posterior in Equation 2.4. These two goals are accomplished by optimizing a lower bound on $p_\theta(x)$. Maximizing this lower bound also decreases the error in the approximation of the true posterior. The following expression decomposes the log-likelihood of the true posterior [26]:

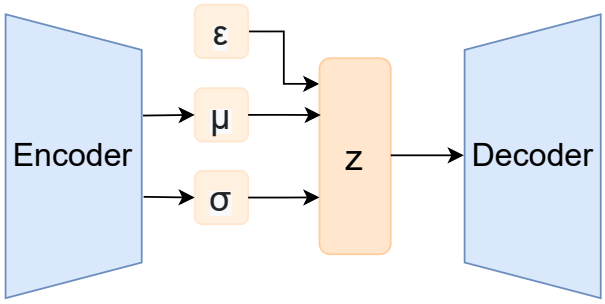

Figure 2.8: Simplified illustration of a VAE. The encoder estimates the distribution parameters (μ and σ) and ϵ is the noisy data that sampled from a standard normal distribution.

$$\log p_\theta(x) = \mathbb{E}_{z\sim q_\psi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\psi(z|x)}\right] + \mathbb{E}_{z\sim q_\psi(z|x)}\left[\log \frac{q_\psi(z|x)}{p_\theta(z|x)}\right]. \tag{2.5}$$

The Kullback–Leibler (KL) divergence between the approximation of the posterior and the true posterior appears as the second term in this decomposition, $\mathrm{KL}(q_\psi(z|x)||p_\theta(z|x))$. The KL divergence measures how different these two distributions are. Therefore, it is always non-negative and equals zero only when the two distributions are identical in 2.4. Because it is non-negative, a lower bound of $p_\theta(x)$ can be obtained by removing this term from the decomposition. For this reason, the first term of the decomposition is referred to as the variational lower bound or the Evidence Lower Bound (ELBO). Minimizing the negative ELBO is essential for determining the parameters ψ and θ, and define the loss function as follows [26]:

$$\begin{aligned} J_{\mathrm{ELBO}}(\psi,\theta) &= -\mathbb{E}_{x\sim\mathcal{X}}\mathbb{E}_{z\sim q_\psi(z|x)}\left[\log p_\theta(x,z) - \log q_\psi(z|x)\right] \\ &= \mathbb{E}_{x\sim\mathcal{X}}\mathbb{E}_{z\sim q_\psi(z|x)}\left[-\log p_\theta(x|z) - \log p_Z(z) + \log q_\psi(z|x)\right] \end{aligned} \tag{2.6}$$

The approximation of negative ELBO in Equation 2.6 is presented in Equation 2.7, where $x^{(i)}$ are i.i.d. samples from $\mathcal{X}$.

$$J_{\mathrm{ELBO}}(\psi,\theta) \approx \frac{1}{s}\sum_{i=1}^{s}\mathbb{E}_{z\sim q_\psi(z|x^{(i)})}\left[-\log p_\theta(x^{(i)}|z) - \log p_Z(z) + \log q_\psi(z|x^{(i)})\right] \tag{2.7}$$

Here, stochastic approximation schemes are employed to avoid directly computing the expectation $\mathbb{E}_{z\sim q_\psi(z|x^{(i)})}$ by instead minimizing J_{ELBO}. In practice, this involves drawing a sample from the posterior approximation to estimate the expected value [26].

Equation 2.6 can be reformulated as follows to provide a clearer understanding of the VAE's objective function [26]:

$$J_{\text{ELBO}}(\psi, \theta) = \mathbb{E}_{x\sim\mathcal{X}}\mathbb{E}_{z\sim q_\psi(z|x)}\left[-\log p_\theta(x|z)\right] + \mathbb{E}_{x\sim\mathcal{X}}\mathbb{E}_{z\sim q_\psi(z|x)}\left[\log q_\psi(z|x) - \log p_Z(z)\right]$$
$$= \mathbb{E}_{x\sim\mathcal{X}}\mathbb{E}_{z\sim q_\psi(z|x)}\left[-\log p_\theta(x|z)\right] + \mathbb{E}_{x\sim\mathcal{X}}\left[\text{KL}\left(q_\psi(z|x) \,\|\, p_Z(z)\right)\right] \tag{2.8}$$

Here, the first term is the reconstruction error, which measures how well the approximate posterior can reconstruct the input data. The second term is the KL divergence, which is thought of as a regularizer. It makes the approximate posterior to remain close to the latent variables. Therefore, there is a trade-off between these two terms. Improving the reconstruction error can cause overfitting of the latent space, resulting in less normally distributed latent representations. On the other hand, minimizing the KL divergence might cause all approximate posteriors to resemble the prior distribution, which can decrease reconstruction quality. The likelihood function $p_\theta(x|z)$ used here is crucial in balancing this trade-off [26].

The implementation of the VAE in the project considers the distribution of the latent space to be a Gaussian distribution, where the mean is μ and the variance is σ^2 [25], as defined by $q_\psi(z|x) = \mathcal{N}(\mu(x), \sigma^2(x))$, since the exact distribution of $p(z)$ is unknown. With this assumption, $p(x|z)$ can be computed, as defined by $p_Z(z) = \mathcal{N}(0, 1)$.

The differentiation of a sample $z \sim q_\psi(z|x)$ becomes possible through the use of the reparameterization trick [26], as shown in Equation 2.9. This trick enables backpropagation by making the sampling operation differentiable. It also makes it possible to apply Monte Carlo estimation in the training process [26].

$$z(\epsilon) = \mu_\psi(x) + \exp(0.5 \cdot \log \sigma^2_\psi(x)) \cdot \epsilon \quad \text{where } \epsilon \sim \mathcal{N}(0, 1) \tag{2.9}$$

Now, the KL divergence term in ELBO loss function in Equation 2.8 can be rewritten in closed form, as shown in Equation 2.10 where J is the number of dimensions of the latent variable z [27]. This is possible by assuming a Gaussian distribution for both the approximate posterior and the prior [27]. The KL divergence showed here corresponds to the implementation used in this project.

$$\text{KL}\left(\mathcal{N}(\mu, \sigma^2) \,\|\, \mathcal{N}(0, 1)\right) = \frac{1}{2}\sum_{j=1}^{J}\left(\mu_j^2 + \sigma_j^2 - \log \sigma_j^2 - 1\right) \tag{2.10}$$

Under the assumption that $p(x|z)$ comes from a Gaussian distribution, the Euclidean distance between, X and $\hat{X}$ corresponds to the negative logarithmic probability of X [28]. Therefore, the reconstruction loss is computed as shown in Equation 2.11. This formulation is used in the implementation of the project.

$$\text{Reconstruction Loss} = ||X - \hat{X}||^2 \tag{2.11}$$

Figure 2.9 illustrates the training process of a VAE, which involves both KL divergence loss and reconstruction loss. The input X is fed into the encoder, which outputs the mean and variance parameters. These parameters are used to compute the KL divergence and to apply the reparameterization trick, enabling backpropagation through the sampling process. The decoder then takes the inferred latent vector and reconstructs the output $\hat{X}$. Finally, the reconstruction loss is computed by comparing X and $\hat{X}$.

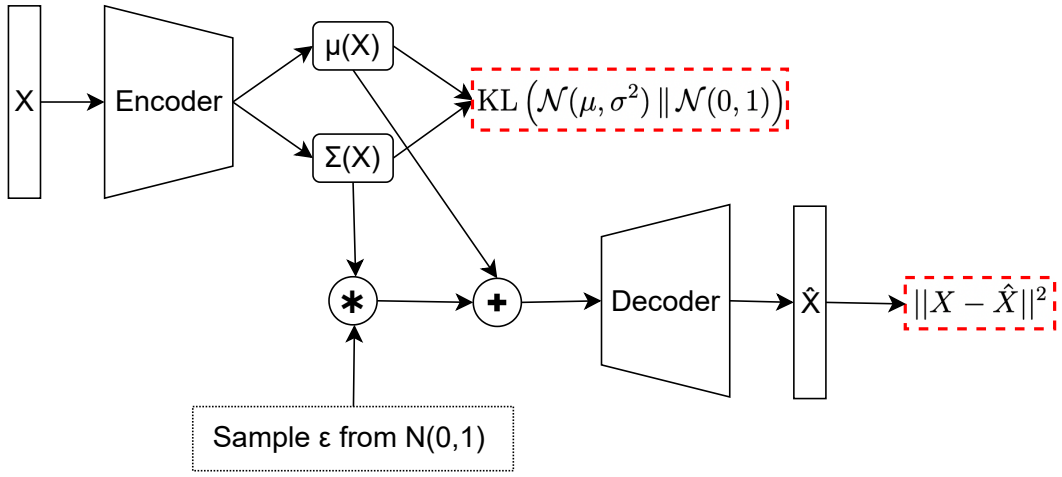

Figure 2.9: The training process of the VAE. The loss components, KL divergence and reconstruction loss, are indicated by red dashed lines.

2.4 Latent Space

As mentioned in Section 2.3, the latent space is one of the components of autoencoders. The hidden information of the input data can be captured by the latent space, while keeping its dimensionality smaller than that of the data space [26], [29]. Consequently, the latent space provides a more condensed and useful representation of the data [29]. Following training, each item in the dataset should have a representation in the latent space that is not directly visible in the higher-dimensional input data [30]. The latent space of a VAE can be used to reconstruct existing data and generate new data through interpolation.

The following section discusses methods for visualizing the latent space in order to analyze its structure.

2.4.1 Visualization of Latent Space

Visualizing the latent space provides insight into the hidden structure or design learned by the model. It also can be meaningful for understanding the smoothness of interpolation. This enables smooth generation of new sounds by interpolating between two original sounds, making sound customization more flexible.

The dimension of the latent space is typically too high to visualize directly. Therefore, dimensionality reduction methods need to be applied. Several techniques exist for this purpose, such as Principal Component Analysis (PCA), t-distributed Stochastic Neighbor Embedding (t-SNE) [31], and Uniform Manifold Approximation and Projection (UMAP) [32].

- **PCA:** A linear dimensionality reduction method that maximizes variance retention and conserves linear connections [33].

- **t-SNE:** A nonlinear method that maps original space's neighbor points in the lower-dimensional embedding [33] to maintain local structure.

- **UMAP:** A nonlinear method that maintains relationships among clusters [33] while reducing dimensionality. Sainburg et al. [34] applied UMAP to the latent space to distinguish between different animal sounds. The authors noted that previous researches suggest UMAP is more suitable than t-SNE for applications in the natural sciences.

2.5 UMAP

UMAP is recently one of the common nonlinear dimension reduction methods using manifold learning [32]. Manifold form of the data is aimed to be understood, and also, embeddings in low dimension are desired to be discovered while preserving the fundamental form. The essential hyperparameters that affect the consequent embeddings are: number of neighbors, minimum distance, number of components, and metric [35].

The number of neighbors determines how much priority is placed on local versus global relationships within the data. The hyperparameter limits the size of the local neighborhood while learning the data's manifold structure. Therefore, when the number of neighbors is small, UMAP focuses more on local structures; when it is larger, it emphasizes more global structures [35].

The minimum distance parameter limits how closely points are arranged. Thus, a small value for minimum distance causes more intense groups in low dimensions, which can be beneficial for analyzing clusters and finer topology of the structure. On the other hand, a larger distance avoids placing points too close together and maintains the broad topology of the structure [35].

The number of components parameter controls the dimensionality of the resulting embeddings, while the metric parameter decides the way to calculate the distance between points, such as Euclidean, Manhattan, or Cosine [35].

2.6 Density-Based Spatial Clustering of Applications with Noise

The density-based spatial clustering of applications with noise (DBSCAN) [36] is a clustering algorithm that clusters data points of any shape in high-dimensional spatial and non-spatial databases with noise [37]. The main principle of the DBSCAN is that the distance between each point in a cluster should be less than ϵ and each cluster should have at least a certain number of points as a threshold [36]. If the distance of a point is not close enough to be included in the nearest cluster and the density of the neighborhood is less than the threshold, that point is considered as noise.

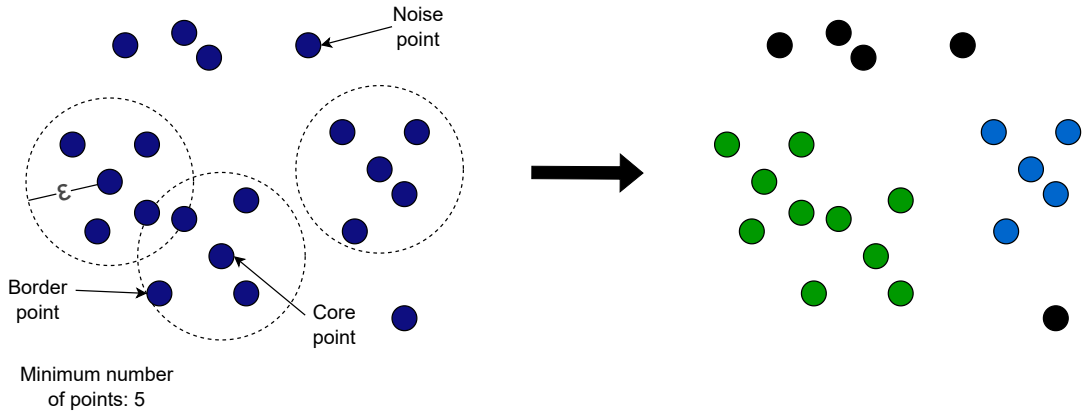

Figure 2.10: How DBSCAN clusters data points.

DBSCAN determines the cluster points with a certain minimum number of points in the epsilon ϵ diameter as core points and cluster points that cannot exceed the minimum number of points around the epsilon as border points. Figure 2.10 shows how DBSCAN works: after defining the minimum point and epsilon ϵ value provided as input, the algorithm identifies core and border points to cluster data points.

2.7 Evaluation Methods

This section outlines the methods used to evaluate the proposed framework. It includes evaluation techniques for both the clustering algorithm and the generated audio.

2.7.1 Clustering Evaluation

Density-based clustering validation (DBCV) [38] considers the arbitrary shape of clusters and the noisiness of data into account; these make DBCV a good candidate for evaluating DBSCAN results.

Density-contour trees [39] are utilized for measuring area with the least population density in a cluster and area with the most population density between the clusters

[38]. In other words, DBCV assess clusters' density connectivity for both inside and between clusters [38]. The key idea is using minimum spanning tree (MST) for measuring a cluster's density sparseness (CDS) and a pair of clusters' density separation (PCDS) for evaluating a cluster's validity index (V_C). A cluster C_i's validity index is described in Equation 2.12 where $1 \leq i \leq l$.

$$V_C(C_i) = \frac{\min\limits_{1 \leq j \leq l, j \neq i}[\text{PCDS}(C_i, C_j)] - \text{CDS}(C_i)}{\max[\min\limits_{1 \leq j \leq l, j \neq i}[\text{PCDS}(C_i, C_j)], \text{CDS}(C_i)]} \tag{2.12}$$

The description of DBCV is shown in Equation 2.13. The weighted average includes cluster size and entire object number including noise, $|C_i|$ and $|O|$ respectively. Therefore, even though the expression does not include noise directly, it includes noise with the weighting average [38]. The output of DBCV is in the range of -1 and 1, where higher values imply well density-based cluster results.

$$\text{DBCV}(C) = \sum_{i=1}^{i=l} \frac{|C_i|}{|O|} V_C(C_i) \tag{2.13}$$

2.7.2 Audio Evaluation

This section introduces the evaluation metrics used to assess the quality of the generated sounds.

2.7.2.1 Fréchet Audio Distance

Fréchet Audio Distance (FAD) [40] is a metric for evaluating of enhancement of generated audio. It is inspired by the Fréchet Inception Distance (FID) [41] that is a metric for assessing quality of generated images. Unlike some other metrics, FAD does not require the original reference audio for evaluating the quality of the audio, which makes FAD suitable for this project since there is no correct original audio.

The original FAD paper introduced the use of VGGish [42] to create a set of qualified embeddings from large dataset with quality music, as well as a set of assessment embeddings from music wanting to evaluate quality. The multivariate Gaussians are calculated for qualified embeddings $\mathcal{N}_q(\mu_q, \Sigma_q)$ and assessment embeddings $\mathcal{N}_a(\mu_a, \Sigma_a)$ and then their Fréchet distance [43] is estimated as shown in Equation 2.14 where the trace of a matrix is represented as tr. A lower FAD score indicates better quality, as the distance between qualified embeddings and evaluation embeddings is smaller.

$$\text{F}(\mathcal{N}_q, \mathcal{N}_a) = ||\mu_q - \mu_a||^2 + tr(\Sigma_q + \Sigma_a - 2\sqrt{\Sigma_q \Sigma_a}) \tag{2.14}$$

The long-distance temporal shifts in a song are not exposed by FAD since the embeddings produced by the method have 1 second windows [40]. Therefore, the method is convenient for evaluating generated sounds rather than rhythm.

Other FAD implementations use models beyond VGGish, such as PANNs [44], CLAP [45], and EnCodec [46]. These are pre-trained models trained on large-scale audio datasets. While they follow the same fundamental logic, they differ in terms of the datasets used for training and their underlying neural network architectures.

2.7.2.2 Spectral Evaluation Methods

Feature extraction methods can be used as evaluation metrics by comparing the original and reconstructed audio. In this project, STFT, MFCC, and CQT are employed for this purpose, using Mean Squared Error (MSE) and L1 loss distribution as the evaluation criterion.

MSE measures the sum of average squared error between the features extracted from the original audio, Y_{true}, and those from the reconstructed audio, $\hat{Y}_{\text{reconstructed}}$ as shown in Equation 2.15 where n is number of samples. This metric emphasizes larger errors made by the model while minimizing the impact of smaller ones. It is useful for identifying whether the model is making significant mistakes.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_{\text{true}_i} - \hat{Y}_{\text{reconstructed}_i})^2 \tag{2.15}$$

L1 loss calculates the magnitude difference between the features extracted from the original audio, Y_{true}, and the reconstructed audio, $\hat{Y}_{\text{reconstructed}}$ as shown in Equation 2.16. The L1 loss is computed for each data point and stored to form a distribution of errors.

$$\text{L1 loss} = \sum_{i=1}^{n}|Y_{\text{true}_i} - \hat{Y}_{\text{reconstructed}_i}| \tag{2.16}$$

To compare the L1 loss distributions, the Kruskal–Wallis test can be used. The Kruskal–Wallis test is a statistical test which is nonparametric [47], meaning it does not assume the distribution of the data, so it is convenient for non-normal distributions. It is used to determine distinctness between three or more independent distributions [47].

The Kruskal–Wallis test checks if there is any statistically significant difference, which means the distinctness is not because of chance. However, even if there is a statistically significant difference, it does not necessarily mean that the difference is relevant. The effect size is used to address this issue in statistics. The value of the effect size can indicate whether the relationship between these distributions is strong and meaningful [48].

These evaluation metrics help assess the quality of the reconstructed audio by emphasizing the most distinguishing attributes of the signals, as they are calculated using audio feature extraction methods.

3

Related Work

This chapter provides an overview of related literature and tools. Section 3.1 outlines techniques for differentiable audio synthesis. Section 3.2 examines TorchSynth, a Python library, with particular emphasis on its synthesizer class. Section 3.3 introduces commonly used audio-related loss functions found in the literature. Finally, Section 3.4 describes an existing tool for exploring sound spaces.

3.1 Differentiable Audio Synthesis Techniques

Differentiable audio synthesis techniques are crucial for the project, as they integrate digital signal processing into neural networks by enabling backpropagation for loss function gradients [49]. This allows the model to utilize gradients from the synthesis process itself, providing deeper insights into the audio signal. There are two essential python libraries for differentiable sound synthesis techniques: DDSP and torchsynth.

DDSP: Differentiable digital signal processing (DDSP) is a library that combines signal processing and deep learning through TensorFlow [50]. High-fidelity audio is produced using differentiable synthesis unit with vocoders [51]. The authors also manage to control audio characters using DDSP units [51].

TorchSynth: A modular synthesizer library that produces pair of samples of sound and its synthesizer parameters more quickly than real-time using only a GPU [52]. The following section discusses the advantages and core synthesizer architecture of the TorchSynth library.

3.2 TorchSynth

TorchSynth was chosen for this project because it is better suited than the DDSP library, particularly due to its ability to generate sound in real time. The models in this project continuously produce audio using synthesizer parameters, making real-time synthesis a critical feature for reducing computational complexity. The following section describes the synthesizer used in this project for sound generation.

3.2.1 The Voice Synthesizer

The synthesizer Voice in Torchsynth was selected for this project since it is a capable synthesis using FM synthesis, noise generation, and other oscillators. Its complex modulation structure enables synthesizing a rich variety of sounds. Moreover, it allows for easy generation of sounds with the desired duration.

The Voice consists of several customizable components: one monophonic keyboard, two LFO modules, each of which has two ADSR envelopes that modulate rate and amplitude, one sine-wave VCO, one squaresaw-wave VCO, and a noise generator. Furthermore, the system integrates VCA modules, a modulation mixer, and an audio mixer [52] for shaping and combining signals as illustrated in Figure 3.1.

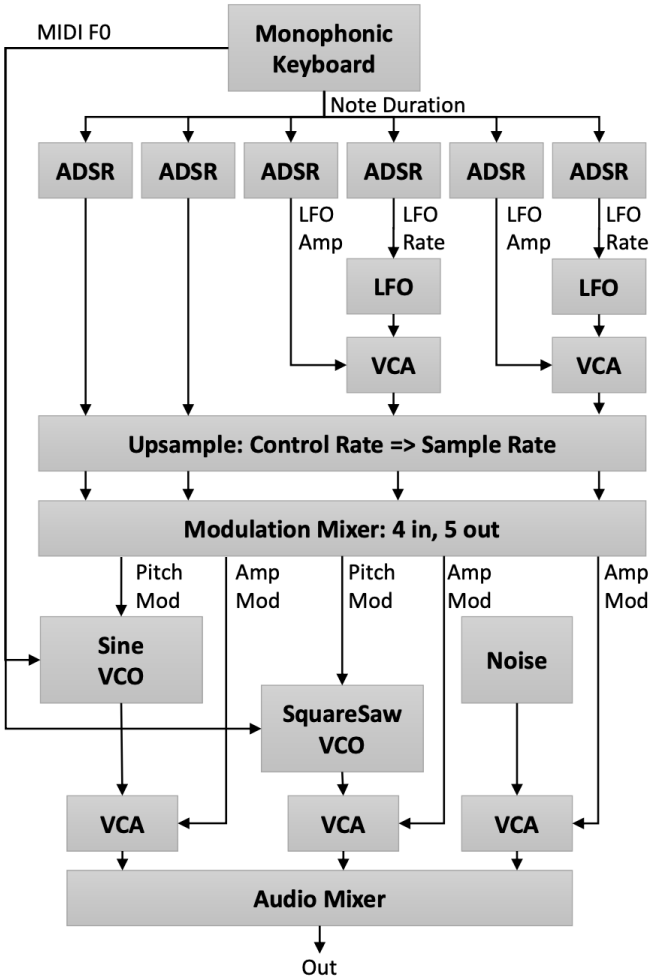

Figure 3.1: Architecture of the Voice synthesizer. Reproduced from [53].

The monophonic keyboard does not generate sound directly; instead, it sends information to the synthesizer, specifying which note to play and for how long. It is referred to as 'monophonic' because it can only play one note at a time. The keyboard provides two parameters: note duration and MIDI F0. The note duration determines how long the note is sustained, while the MIDI F0 value specifies the pitch of the note, such as A4 or C4.

The Voice architecture includes six ADSR modules. Two of these are directly connected to the LFO to modulate its rate, i.e., the oscillator's frequency over time. Another two are connected to the VCA to modulate the amplitude over time. The remaining two are connected to the upsampling module, which converts control-rate signals to sample-rate signals. Following this process, a modulation mixer is used to control the pitch of both the Sine VCO and the SquareSaw VCO, as well as the amplitude of the VCA.

The MIDI F0 information is headed to both the Sine VCO and the SquareSaw VCO, as these are the main oscillators in the architecture responsible for generating sound at the specific pitch indicated by the MIDI F0 value.

The two VCO and the Noise module are connected to the VCA to control the volume of the sounds over time. After this stage, all signals are combined in Audio Mixer module to produce a single unified sound output.

All components of the synthesizer are important for producing rich and diverse sounds. For instance, VCOs and the Noise module are responsible for creating the raw sounds that humans can hear, while components like ADSR and LFO shape these sounds to make them more lively and colorful.

3.3 Loss Functions for Audio

Raw audio waveforms are not suitable for estimating loss through point-wise comparison because two audio signals can have different waveforms but sound similar, and vice versa [50]. Hayes et al. (2024) [49] listed three audio loss functions that frequently appear in the literature, all of which use magnitude spectrograms: Spectral Convergence Loss (SCL) [54], Log Magnitude Spectral Loss (LogMSL) [54], and Linear Magnitude Spectral Loss (LMSL), as outlined below.

- SCL measures the difference between the magnitude STFTs of the true audio y and the reconstructed audio $\hat{y}$ using the Frobenius norm, denoted as $||\cdot||_F$. The result is then normalized by the Frobenius norm of the true audio's magnitude spectrogram, as shown in Equation 3.1.

$$L_{\text{SCL}}(y, \hat{y}) = \frac{\| \, |\text{STFT}(y)| - |\text{STFT}(\hat{y})| \, \|_F}{\| \, |\text{STFT}(y)| \, \|_F} \tag{3.1}$$

- LogMSL calculates the difference between the logarithmic magnitude STFTs of the ground truth audio and the reconstructed audio $\hat{y}$. A small constant ϵ is added to avoid taking the logarithm of zero. The difference is measured using the L_1 norm, denoted as $||\cdot||_1$, as shown in Equation 3.2.

$$L_{\text{LogMSL}}(y, \hat{y}) = \| \log(|\text{STFT}(y)| + \epsilon) - \log(|\text{STFT}(\hat{y})| + \epsilon) \|_1 \tag{3.2}$$

- LMSL computes the difference between the magnitude STFTs of the ground truth audio and the reconstructed audio $\hat{y}$ using the L_1 norm, denoted as $||\cdot||_1$,

as shown in Equation 3.3.

$$L_{\text{LMSL}}(y, \hat{y}) = \| |\text{STFT}(y)| - |\text{STFT}(\hat{y})| \|_1 \tag{3.3}$$

SCL, as shown in Equation 3.1, prioritizes large spectral components, which facilitates the initial training process. On the other hand, LogMSL, as shown in Equation 3.2, is more susceptible, as it uses the logarithm of the magnitude of the STFT. This allows LogMSL to focus on fitting small-amplitude components, which become more critical in the later stages of training [54]. In contrast, LMSL is simple and steady, as it directly computes the difference between the magnitude STFTs of the true and reconstructed signals. However, since all of these loss functions consider only the magnitude spectrogram, they do not account for the phase of the audio.

3.4 Exploring Sound Space

AudioStellar [55] is a recent study to explore the latent space of sound. The authors consecutively applied PCA and t-SNE for dimensionality reduction of the latent space and clustered similar sounds using DBSCAN based on their timbre, amplitude, envelope, or a combination of these features. In this way, the authors provide an interactive visualization of sound's latent space for exploration. The framework of AudioStellar is illustrated in Figure 3.2. However, it is important to note that their framework does not include a generative model for sound synthesis.

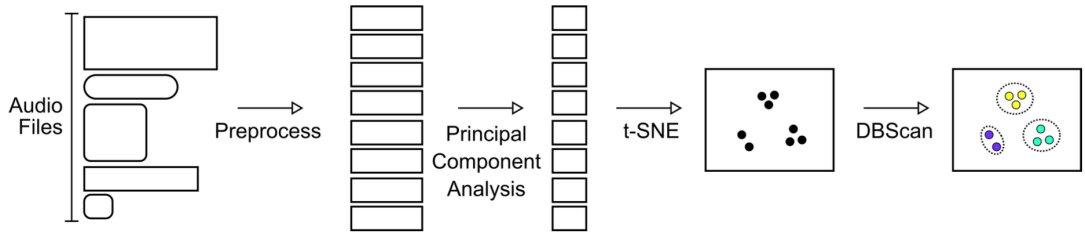

Figure 3.2: The pipeline of AudioStellar. Reproduced from [56].

4

Methods

This chapter provides an overview of the methods and experiments conducted, and also includes details about the dataset and the proposed framework. Section 4.1 explains the dataset generation process and the preprocessing methods applied. Section 4.2 presents the detailed pipeline of the project, including the VAE model and the evaluation methods used to assess the quality of the results. Section 4.3 explains the main components of the loss functions used in the experiments. Section 4.4 describes the first experiment and its cases, focusing on various audio-related loss function formulations. Section 4.5 details the second experiment and its cases, which investigate the impact of adding the TorchSynth layer to the model. Finally, Section 4.6 explains the third experiment, which focuses on evaluating different audio feature extraction methods.

4.1 Dataset

The dataset used to train the VAE consists of synthesized sound parameters generated using the parameter of Voice module from the TorchSynth library. Each sound is one second long, making them well-suited for use within a beat.

As shown in Figure 3.1, the Voice synthesizer is composed of several modules, each with a variety of parameters. In total, there are 63 parameters, as detailed in Table 4.1. These parameters are used to generate the dataset. For each sound, a random value is assigned to each parameter. Using this method, 140,000 sound parameter sets are generated and stored in a JSON file to ensure consistency across experiments. The dataset is then split into training, validation, and test sets, containing 100,000, 20,000, and 20,000 samples, respectively.

To save memory, only the parameter sets are saved, rather than the raw sound and parameter pairs. Since each raw audio file contains 44,100 data points, storing them would require a significant amount of memory. Moreover, this would lead to high dimensionality in the input layer. Therefore, training the model using only the parameters simplifies the process and speeds up training. Additionally, it provides an opportunity to explore the feature space of the synthesizer parameters.

For preprocessing, each parameter is normalized to a range between 0 and 1 before being input into the model. This step is necessary because the parameters span different value ranges, which are detailed in Tables A.1 and A.2 in the appendix.

Module	Parameters
keyboard	midi_f0, duration
adsr_1	attack, decay, sustain, release, alpha
adsr_2	attack, decay, sustain, release, alpha
lfo_1	frequency, mod_depth, initial_phase, sin, tri, saw, rsaw, sqr
lfo_2	frequency, mod_depth, initial_phase, sin, tri, saw, rsaw, sqr
lfo_1_amp_adsr	attack, decay, sustain, release, alpha
lfo_2_amp_adsr	attack, decay, sustain, release, alpha
lfo_1_rate_adsr	attack, decay, sustain, release, alpha
lfo_2_rate_adsr	attack, decay, sustain, release, alpha
mod_matrix	adsr_1→vco_1_pitch, adsr_2→vco_2_pitch, lfo_1→vco_1_amp, lfo_2→vco_2_amp, adsr_1→noise_amp
vco_1	tuning, mod_depth, initial_phase
vco_2	tuning, mod_depth, initial_phase, shape
mixer	vco_1, vco_2, noise

Table 4.1: Modules and their parameters in the Voice synthesizer.

4.2 Pipeline of the Project

This section presents the complete pipeline, which includes dataset generation, sound synthesis, the VAE model, referred to as SynthParam-VAE, and evaluation methods. The overall pipeline is illustrated in Figure 4.1. SynthParam-VAE is implemented using the PyTorch library. While the exact architecture of SynthParam-VAE may vary between experiments, it remains consistent within each individual experiment. This consistency allows for fair comparisons of different loss functions under the same experimental conditions.

To further ensure fairness, the same random seed, as well as identical initial weights and biases, are used within each experiment to eliminate variability caused by randomness. As described in Section 4.1, the model is trained on the sound parameters of the Voice synthesizer.

To evaluate the quality of the model's reconstructions, the predicted parameters are passed through the Voice synthesizer to generate reconstructed sounds. These are then compared with the original sounds using FAD and various spectral metrics, providing a quantitative assessment of reconstruction quality.

Two different FAD implementations are used: one based on VGGish embeddings and another based on PANNs embeddings. Using multiple models helps ensure a more robust evaluation of audio similarity. In addition, several spectral metrics are employed to assess reconstruction quality. MSE and L1 loss distribution are calculated between the ground truth and reconstructed audio using three different representations: STFT, MFCC, and CQT.

To further analyze the distribution of reconstruction errors, L1 loss distributions are

computed for STFT, MFCC, and CQT. To determine whether there are statistically significant differences between these distributions, the Kruskal–Wallis H-test is applied separately to the results from each representation. This metric helps determine whether there are significant differences in reconstruction quality between models.

MSE is used to identify which models produce larger reconstruction errors, while the L1 loss distribution provides a general overview and more stable indication of reconstruction quality.

To assess the organization of the model's latent space, latent vectors are first extracted from the encoder. These vectors are then reduced to two dimensions using UMAP for visualization. DBSCAN is applied to cluster the resulting 2D points, allowing us to examine whether similar sounds are grouped together. This is an indication that the latent space effectively captures sound characteristics. The quality of clustering is evaluated using the DBCV metric.

To analyze how well the model organizes similar sounds in UMAP representations, three medium-sized random clusters are selected. For each cluster, the average FFT values are computed. By analyzing these averaged FFT profiles, it is possible to assess whether the clusters represent distinct sound characteristics. Additionally, listening to random samples from each cluster provides a qualitative sense of how different or similar the sounds are within and across clusters.

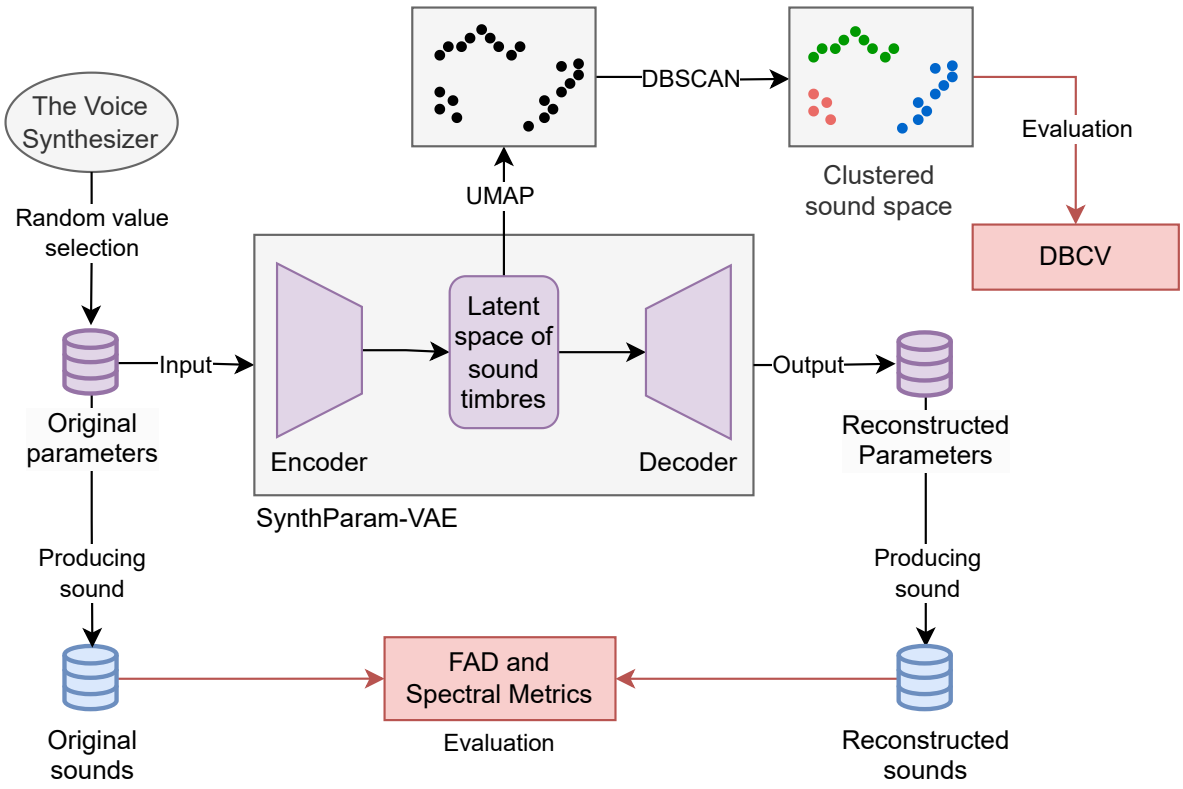

Figure 4.1: Illustration of the pipeline of the project.

4.3 Loss Function Components

Designing the loss function is an important part of addressing the research questions in this project. The exact formulation of the loss function varies depending on the specific experiment and the conditions within each case. This section outlines the main components of the loss function, explaining their roles and the rationale behind their inclusion.

- Reconstruction loss ($L_{recon}(y, \hat{y})$): This is the squared error between the ground truth parameters y and the reconstructed parameters $\hat{y}$. It measures how accurately the model can reproduce the original parameters, indicating the quality of the reconstruction.
- Spectral loss ($L_{spectral}(y, \hat{y})$): This loss function varies across experiments and between different cases within each experiment. Depending on the case, it may use different audio-related formulations or feature extraction methods. Since different spectral loss functions can produce values on different scales, a constant weight β is applied to the spectral loss to ensure a balanced contribution relative to the reconstruction loss. This weighting maintains a consistent ratio of 1:3 between the reconstruction loss and the spectral loss in cases where both are used, enabling fair comparisons across cases within experiments.
- Clamping loss ($L_{clamp}(\hat{y})$): The Voice synthesizer requires parameters to fall within specific valid ranges. However, the model may sometimes predict values outside these ranges, which prevents audio from being generated and causes the training process to crash. In such cases, the out-of-range values are clamped to the nearest valid boundary. This clamping process implies that the model has not fully learned the valid parameter ranges. This loss penalizes such occurrences, encouraging the model to produce values within acceptable limits. It is weighted by a factor α, which is set to 100 in all experiments to emphasize its importance.
- Kullback-Leibler Divergence ($L_{KL}(y, \hat{y})$): This term encourages the encoder to shape the latent space distribution resembling a standard Gaussian distribution. It plays a crucial role in helping the model learn a well-structured latent space. Without this loss term, the model might still achieve good reconstruction performance due to the reconstruction loss, but it would struggle to generate meaningful samples. This is because the latent space would lack proper organization, making sampling from it unreliable or inconsistent.

4.4 Experiment 1

Experiment 1 focuses on evaluating the performance of different audio-related loss function formulations while using the same feature extraction technique, namely STFT. The goal is to determine which audio-related loss formulation is most effective for capturing and learning timbre features of sounds. This experiment is designed to address the first research question.

The encoder and decoder of the architectures mirror each other. The encoder's input layer has 63 neurons, the first hidden layer has 256 neurons, and the second hidden layer has 64 neurons. The latent space has 16 neurons to facilitate easier visualization, making the latent space relatively small. The decoder has layers with 64 neurons, then 256 neurons, and the output layer has 63 neurons.

4.4.1 Spectral Loss Functions

As previously mentioned, one of the research questions is to identify a suitable loss function formulation for the SynthParam-VAE. In total, six different loss functions were tested for this experiment. These include the formulations presented in Equations 3.1, 3.3, and 3.2. In addition to these, three variations that incorporate phase information are also considered, as described below.

- Spectral convergence loss with phase (SCLP): measures the difference between the complex-valued STFTs of the true audio y and the reconstructed audio $\hat{y}$ using the Frobenius norm, denoted as $||\cdot||_F$. The result is then normalized by the Frobenius norm of the true audio's the complex-valued STFT, as shown in Equation 4.1.

$$L_{\text{SCLP}}(y, \hat{y}) = \frac{\| \text{STFT}(y) - \text{STFT}(\hat{y}) \|_F}{\| \text{STFT}(y) \|_F} \tag{4.1}$$

- Log spectral loss with phase (LogSLP): calculates the difference between the complex-valued logarithmic STFTs of the ground truth audio and the reconstructed audio $\hat{y}$. A small constant ϵ is added to avoid taking the logarithm of zero. The difference is measured using the L_1 norm, denoted as $||\cdot||_1$, as shown in Equation 4.2.

$$L_{\text{LogSLP}}(y, \hat{y}) = \| \log(\text{STFT}(y) + \epsilon) - \log(\text{STFT}(\hat{y}) + \epsilon) \|_1 \tag{4.2}$$

- Linear spectral loss with phase (LSLP): computes the difference between the complex-valued STFTs of the ground truth audio and the reconstructed audio $\hat{y}$ using the L_1 norm, denoted as $||\cdot||_1$, as shown in Equation 4.3.

$$L_{\text{LSLP}}(y, \hat{y}) = \| \text{STFT}(y) - \text{STFT}(\hat{y}) \|_1 \tag{4.3}$$

These phase-aware loss functions are introduced because implementing the magnitude of the STFT results in the loss of phase information, as it does not contain this data. This can lead to slight shifts in the reconstructed audio. Incorporating phase-aware loss functions helps the model reconstruct audio more accurately by compensating for the missing phase information.

Table 4.2 summarizes each case in Experiment 1, including the case name, the abbreviation of the loss function formulation, the full name of the loss function, and the corresponding equation reference.

Case Name	Loss Name	Spectral Loss Term	Equation Ref.
Exp1-SCL	SCL	Spectral Convergence Loss	Eq. 3.1
Exp1-LogSL	LogSL	Log Magnitude Spectral Loss	Eq. 3.2
Exp1-LMSL	LMSL	Linear Magnitude Spectral Loss	Eq. 3.3
Exp1-SCLP	SCLP	Spectral Convergence with Phase	Eq. 4.1
Exp1-LogSLP	LogSLP	Log Spectral with Phase	Eq. 4.2
Exp1-LSLP	LSLP	Linear Spectral with Phase	Eq. 4.3

Table 4.2: Overview of the spectral loss functions used in Experiment 1.

4.4.2 The Total Loss Functions

The total loss function used in Experiment 1 is presented in Equation 4.4. It consists of four components: the reconstruction loss ($L_{recon}(y, \hat{y})$), the weighted clamping loss ($L_{clamp}(\hat{y})$), the weighted spectral loss ($L_{spectral}(y, \hat{y})$), and the KL Divergence ($L_{KL}(y, \hat{y})$).

$$L_{total}(y, \hat{y}) = L_{recon}(y, \hat{y}) + \alpha L_{clamp}(\hat{y}) + \beta L_{spectral}(y, \hat{y}) + L_{KL}(y, \hat{y}) \tag{4.4}$$

4.5 Experiment 2

The aim of this experiment is to investigate the effect of using the TorchSynth layer as the final layer after the decoder on both the training process of the VAE model and its reconstruction accuracy while applying different audio-related loss functions. This experiment is also designed to address the first research question, while further exploring the impact of integrating synthesis layer into the learning process.

The architecture used is the same as in Experiment 1, with the addition of a TorchSynth layer following the decoder. Therefore, the decoder reconstructs the synthesizer parameters and passes them to the synth layer to directly generate the reconstructed sounds. This architecture enables an end-to-end model. Moreover, this modification allows the model to benefit from gradients of the synth layer during backpropagation, potentially providing deeper insights into audio characteristics.

This experiment also applies the six spectral loss functions along with the total loss function defined in Equation 4.4, maintaining consistency with Experiment 1 to facilitate easy comparison. The names of the cases corresponding to this architecture are listed in Table 4.3.

4.6 Experiment 3

The main goal of Experiment 3 is to evaluate the effectiveness of STFT, MFCC, and CQT as audio feature extraction methods for capturing timbre-related characteristics. This experiment is designed to address the second research question.

Case Name	Loss Name	Spectral Loss Term	Equation Ref.
Exp2-SCL-ts	SCL	Spectral Convergence Loss	Eq. 3.1
Exp2-LogSL-ts	LogSL	Log Magnitude Spectral Loss	Eq. 3.2
Exp2-LMSL-ts	LMSL	Linear Magnitude Spectral Loss	Eq. 3.3
Exp2-SCLP-ts	SCLP	Spectral Convergence with Phase	Eq. 4.1
Exp2-LogSLP-ts	LogSLP	Log Spectral with Phase	Eq. 4.2
Exp2-LSLP-ts	LSLP	Linear Spectral with Phase	Eq. 4.3

Table 4.3: Overview of names of the cases in Experiment 2 and the corresponding spectral loss functions used.

Initially, the model is trained using only the Voice synthesizer parameter reconstruction loss to establish a baseline. This phase concludes after 1000 epochs. However, these parameters are not necessarily aligned with perceptual audio qualities. Therefore, in the following step, three new models are created by integrating perceptually relevant audio-based loss functions into the baseline model. Each model uses the same spectral loss function, namely SCLP, but differs in the audio feature extraction method applied, specifically STFT, MFCC, or CQT. In this phase, the parameter reconstruction loss is removed to encourage the model to focus more on learning spectral characteristics rather than regressing to the original synthesizer parameters.

For this experiment, the VAE architecture is intentionally kept simpler compared to other experiments, as the initial phase involves only parameter reconstruction without incorporating any spectral loss terms. Both the encoder and decoder consist of a single hidden layer. Given that the number of parameters is 63, the input layer has 63 neurons, followed by a hidden layer with 128 neurons. The latent space has 16 dimensions to facilitate visualization. The decoder is symmetric to the encoder, with a hidden layer of 128 neurons and an output layer of 63 neurons.

4.6.1 Loss Function Strategy Across Training Phases

The third experiment follows a two-phase training approach:

- **Phase 1 (Baseline Training):** The initial total loss function, defined in Equation 4.5, combines the reconstruction loss, the weighted clamping loss, and the KL divergence:

$$L_{\text{baseline}}(y, \hat{y}) = L_{recon}(y, \hat{y}) + \alpha L_{clamp}(\hat{y}) + L_{KL}(y, \hat{y}) \tag{4.5}$$

- **Phase 2 (Fine-tuning):** After early stopping in Phase 1, a spectral loss term is introduced. This term is based on one of three audio feature extraction types (STFT, MFCC, or CQT) each defining a separate case within the experiment. The total loss function used during this phase is described in Equation 4.6 including the weighted clamping loss, the weighted spectral loss, and the KL divergence.

$$L_{\text{fine-tuning}}(y, \hat{y}) = \alpha L_{clamp}(\hat{y}) + L_{spectral}(y, \hat{y}) + L_{KL}(y, \hat{y}) \tag{4.6}$$

4.6.2 Spectral Loss Functions

The spectral loss component $L_{spectral}$ is implemented using SCLP, applied with three different audio feature representations: STFT, MFCC, and CQT. SCLP is chosen for this experiment because it emphasizes large spectral components, which represent more general and perceptible characteristics of a sound. Since the model did not directly learn audio features during the pre-training phase, focusing on the overall structure of the sound rather than fine details becomes more important in the fine-tuning stage. Therefore, this approach helps the model generate audio that is more perceptually similar to the original, aligning better with human auditory perception.

The spectral loss functions used in the different cases are defined as follows. Each function applies SCLP using a different audio feature extraction method: STFT, MFCC, and CQT, as shown in Equations 4.7, 4.8, and 4.9, respectively.

- SCLP applied to STFT features:

$$L_{\text{STFT}}(y, \hat{y}) = \frac{\| \text{STFT}(y) - \text{STFT}(\hat{y}) \|_F}{\| \text{STFT}(y) \|_F} \tag{4.7}$$

- SCLP applied to MFCC features:

$$L_{\text{MFCC}}(y, \hat{y}) = \frac{\| \text{MFCC}(y) - \text{MFCC}(\hat{y}) \|_F}{\| \text{MFCC}(y) \|_F} \tag{4.8}$$

- SCLP applied to CQT features:

$$L_{\text{CQT}}(y, \hat{y}) = \frac{\| \text{CQT}(y) - \text{CQT}(\hat{y}) \|_F}{\| \text{CQT}(y) \|_F} \tag{4.9}$$

Table 4.4 summarizes each case in Experiment 3, including the case name, the audio feature extraction method used, the corresponding loss function name, and the equation reference.

Case Name	**Spectral Feature**	**Loss Name**	**Equation Ref.**
Exp3-STFT	STFT	STFT-SCLP	Eq. 4.7
Exp3-MFCC	MFCC	MFCC-SCLP	Eq. 4.8
Exp3-CQT	CQT	CQT-SCLP	Eq. 4.9

Table 4.4: Spectral loss types applied during Phase 2 training in Experiment 3.

5

Results

This chapter presents the results obtained from the various experiments conducted throughout the project. Sections 5.1, 5.2, and 5.3 show the results of audio evaluation and clustering assessment for Experiment 1, Experiment 2, and Experiment 3, respectively. Section 5.4 presents additional examples, including latent space interpolation, ground truth and reconstruction comparison, and clustering analysis that explores the differences between selected clusters.

As mentioned in Section 4.2, to assess the reconstruction quality of the models, audio samples are generated using both the ground-truth synthesizer parameters and the reconstructed parameters. FAD scores, based on VGGish and PANN embeddings, are computed using 20% of the training data as well as the test data. Additionally, spectral losses such as STFT MSE, MFCC MSE, and CQT MSE are calculated using the test set. Furthermore, the L1 loss distribution of the cases for the test data is also analyzed.

The clustering of the UMAP representations is assessed using DBCV. To facilitate analysis, latent vectors from 10% of the training data were randomly selected and plotted, as clustering all 100,000 data points made it difficult to identify smaller clusters. The training set was chosen for this task to specifically analyze how the models organize the latent space. UMAP plots, both with and without clustering, for each case are also presented in this chapter.

5.1 Outcomes of Experiment 1

All cases in this experiment were trained for 500 epochs and also evaluated using early stopping. The only difference between the two setups is that the early-stopped models used a fixed learning rate, while the models trained for the full 500 epochs employed a learning rate schedule that decreased every 100 epochs after the 200th epoch. Although the models trained for the full duration did not exhibit overfitting on the training set, their results were similar to those obtained with early stopping. Since early-stopped models are more efficient and yield comparable performance, the results presented in this section are based on the early-stopped models.

The early-stopped cases and their corresponding hyperparameters are shown in Table 5.1. All cases used a batch size of 256, a fixed learning rate of 1×10^{-4}, and an early stopping patience of 20 epochs based on validation performance. The number of

epochs each case was trained for is also shown.

Case	Epochs	Batch Size	Learning rate	Patient
Exp1-LMSL	282	256	1×10^{-4}	20
Exp1-SCL	246	256	1×10^{-4}	20
Exp1-LogSL	165	256	1×10^{-4}	20
Exp1-LSLP	282	256	1×10^{-4}	20
Exp1-SCLP	246	256	1×10^{-4}	20
Exp1-LogSLP	165	256	1×10^{-4}	20

Table 5.1: The cases in Experiment 1, along with their corresponding hyperparameters and the number of training epochs.

5.1.1 Audio Evaluation

Table 5.2 shows that FAD scores do not exhibit significant differences between the cases. However, the main distinction lies in the number of training epochs required to achieve these scores. For example, the cases Exp1-LogSL and Exp1-LogSLP reached their results in just 165 epochs, whereas Exp1-LMSL and Exp1-LSLP required 282 epochs. This suggests that Exp1-LogSL and Exp1-LogSLP introduce audio features to the model more effectively than the other audio-related loss functions in this experiment. Moreover, since the evaluation results on subsets of the training and test data show no significant differences, this indicates that the models did not overfit to the training data.

Table 5.2 shows that Exp1-LMSL and Exp1-LSLP perform slightly better on VGGish embeddings for both the training subset and the test set. They also show slightly better performance on PANNs embeddings for the training subset. However, Exp1-LogSL and Exp1-LogSLP perform slightly better on the test set according to PANNs embeddings.

Case	Subset of Train		Test	
	VGGish ↓	PANNs ↓	VGGish ↓	PANNs ↓
Exp1-LMSL	**0.7454**	**0.0923**$\times 10^3$	**0.7325**	0.1163$\times 10^3$
Exp1-SCL	0.7577	0.1292$\times 10^3$	**0.7325**	0.1563$\times 10^3$
Exp1-LogSL	0.7586	0.0933$\times 10^3$	0.7575	**0.1095**$\times 10^3$
Exp1-LSLP	**0.7454**	**0.0923**$\times 10^3$	**0.7325**	0.1163$\times 10^3$
Exp1-SCLP	0.7577	0.1292$\times 10^3$	0.7517	0.1563$\times 10^3$
Exp1-LogSLP	0.7586	0.0933$\times 10^3$	0.7575	**0.1095**$\times 10^3$

Table 5.2: FAD scores of the cases in Experiment 1 on training subsets and test sets, based on VGGish and PANNs embeddings.

Table 5.3 presents the MSE scores on the test set using different feature extraction methods. Consistent with the FAD scores, the cases do not show significant differences

overall. Exp1-SCL and Exp1-SCLP exhibit very similar performance. Exp1-LogSL performs slightly better in terms of STFT MSE, while Exp1-LogSLP shows better performance on MFCC and CQT MSE.

Case	STFT MSE ↓	MFCC MSE ↓	CQT MSE ↓
Exp1-LMSL	20.9128	**881.284**	18178.72
Exp1-SCL	20.8169	896.875	18086.45
Exp1-LogSL	20.6750	879.013	18093.40
Exp1-LSLP	20.9374	885.940	18203.51
Exp1-SCLP	20.8169	896.875	18086.45
Exp1-LogSLP	**20.6182**	875.318	**17659.07**

Table 5.3: The cases in Experiment 1 and their corresponding MSE scores on the test set, using different feature extraction methods.

The L1 loss distribution for each case was calculated based on STFT, MFCC, and CQT features, and the Kruskal–Wallis H-test was applied to assess statistical differences. The test did not reveal any statistically significant differences between the distributions, except for the MFCC L1 loss. However, an effect size analysis indicated that this difference was not meaningful. Therefore, the comparison of L1 loss distributions across feature extraction methods suggests that the cases perform similarly.

5.1.2 Clustering Evaluation

For Experiment 1, the UMAP parameters were set to 50 for the number of neighbors and 0 for the minimum distance. For DBSCAN, the epsilon value was set to 0.04, and the minimum number of samples was set to 4.

Table 5.4 presents the number of clusters and noise samples identified after applying DBSCAN to the latent vectors, along with the corresponding DBCV scores. The case Exp1-LMSL achieved the highest score, with a DBCV value of 0.4136, while Exp1-LSLP recorded the lowest score at 0.3595. Overall, the DBCV scores are relatively close to one another, meaning the quality of clustering is similar across the cases.

Case	Cluster Number	Noise	DBCV ↑
Exp1-LMSL	818	3616	**0.4136**
Exp1-SCL	767	3293	0.3843
Exp1-LogSL	772	3494	0.3778
Exp1-LSLP	755	3202	0.3595
Exp1-SCLP	767	3293	0.3843
Exp1-LogSLP	772	3494	0.3778

Table 5.4: Clustering results for Experiment 1, including the number of clusters, noise points, and DBCV scores.

Figure 5.1 shows the UMAP plots for Exp1-LMSL and Exp1-LogSL, which achieved the highest and one of the lowest clustering quality scores, respectively, according to the DBCV metric. The left column displays the latent vectors before clustering, while the right column shows the results after applying DBSCAN. Due to the large number of clusters, some clusters may share the same color. UMAP plots for the remaining cases can be found in Appendix B, in Figure B.1.

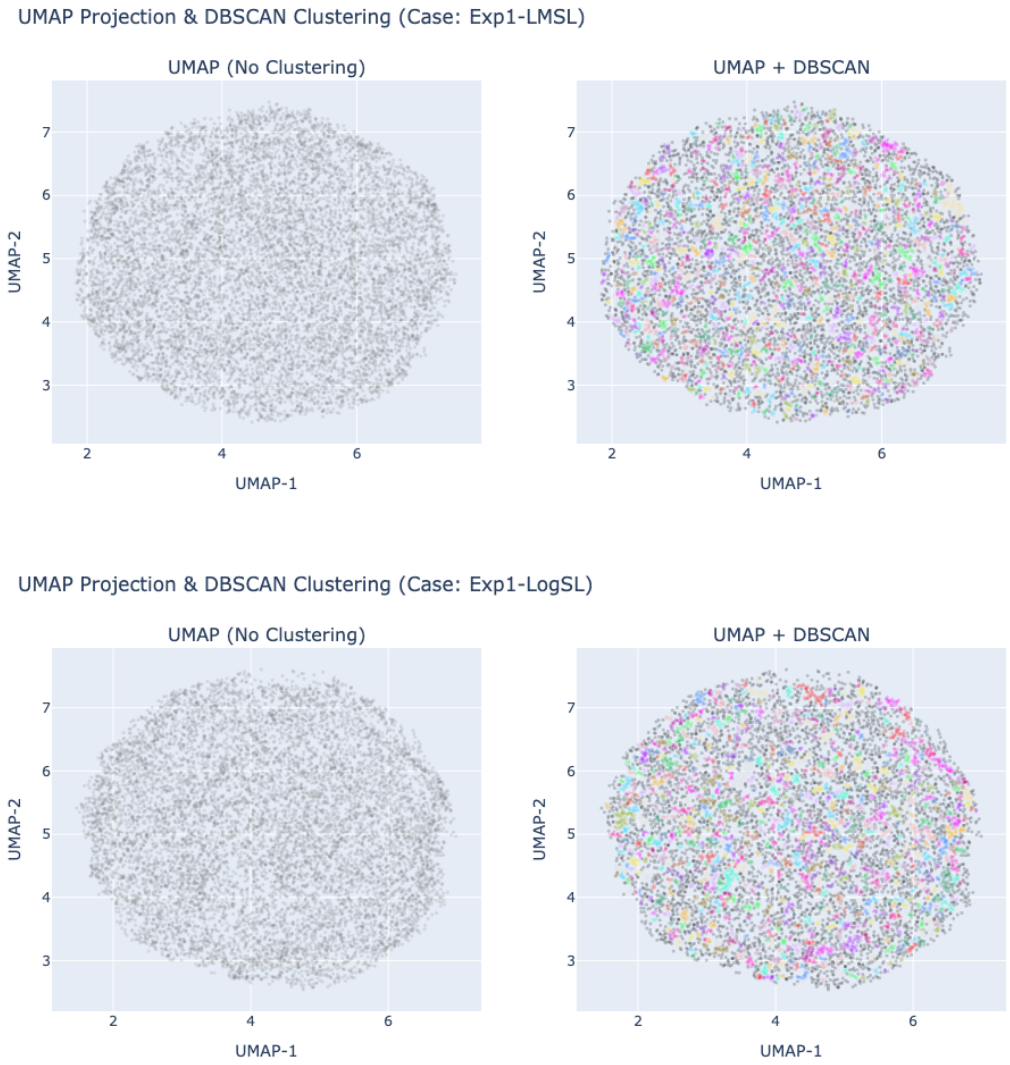

Figure 5.1: UMAP plots of latent vectors (left) and their clustered versions using DBSCAN (right). The first row shows results for Exp1-LMSL, while the second row corresponds to Exp1-LogSL.

5.2 Outcomes of Experiment 2

As in Experiment 1, the models in this experiment were trained both for 500 epochs and with early stopping. The only difference between these two setups was the

learning rate: the early-stopped models used a fixed learning rate, while the models trained for the full 500 epochs had their learning rate decreased every 100 epochs after the 200th epoch. Similar to the results in Experiment 1, the models trained for the full duration did not show signs of overfitting but also did not yield significant performance improvements. Therefore, the early-stopped models were again selected for presenting the results.

Table 5.5 shows the hyperparameters and the number of training epochs for each case. Similar to Experiment 1, all cases used a batch size of 256, a fixed learning rate of 1×10^{-4}, and an early stopping patience of 20 epochs. Compared to Experiment 1, the cases that included the TorchSynth layer generally required fewer training epochs, except for Exp2-LogSL-ts and Exp2-LogSLP-ts.

Case	Epochs	Batch Size	Learning rate	Patient
Exp2-LMSL-ts	228	256	1×10^{-4}	20
Exp2-SCL-ts	228	256	1×10^{-4}	20
Exp2-LogSL-ts	199	256	1×10^{-4}	20
Exp2-LSLP-ts	228	256	1×10^{-4}	20
Exp2-SCLP-ts	228	256	1×10^{-4}	20
Exp2-LogSLP-ts	199	256	1×10^{-4}	20

Table 5.5: The cases in Experiment 2, along with their corresponding hyperparameters and the number of training epochs.

5.2.1 Audio Evaluation

The FAD scores in Table 5.6 do not show significant differences, similar to the results in Experiment 1, even though most cases in Experiment 2 required less training. Exp2-LMSL-ts shows slightly better performance on VGGish embeddings for both the training subset and the test set, while Exp2-SCL-ts performs slightly better on PANNs embeddings for the same sets.

Case	Subset of Train		Test	
	VGGish ↓	PANNs ↓	VGGish ↓	PANNs ↓
Exp2-LMSL-ts	**0.7442**	0.1213×10^3	**0.7435**	0.1349×10^3
Exp2-SCL-ts	0.7467	$\mathbf{0.1135} \times 10^3$	0.7495	$\mathbf{0.1274} \times 10^3$
Exp2-LogSL-ts	0.7500	0.1362×10^3	0.7558	0.1416×10^3
Exp2-LSLP-ts	**0.7442**	0.1213×10^3	0.7495	$\mathbf{0.1274} \times 10^3$
Exp2-SCLP-ts	0.7467	0.1274×10^3	0.7495	$\mathbf{0.1274} \times 10^3$
Exp2-LogSLP-ts	0.7500	0.1362×10^3	0.7558	0.1416×10^3

Table 5.6: FAD scores of the cases in Experiment 2 on training subsets and test sets, based on VGGish and PANNs embeddings.

Table 5.7 presents the spectral MSE scores for the cases in Experiment 2. Exp2-LogSL-ts shows the best performance on STFT MSE, while Exp2-SCLP-ts performs

slightly better on MFCC MSE. For CQT MSE, Exp2-LMSL-ts achieves the best performance.

Case	STFT MSE ↓	MFCC MSE ↓	CQT MSE ↓
Exp2-LMSL-ts	22.3714	918.207	**18946.33**
Exp2-SCL-ts	22.3709	917.534	19331.513
Exp2-LogSL-ts	**22.0708**	927.233	19463.56
Exp2-LSLP-ts	22.4194	918.468	19249.80
Exp2-SCLP-ts	22.2498	**917.315**	19124.00
Exp2-LogSLP-ts	22.0796	921.654	19124.85

Table 5.7: The cases in Experiment 2 and their corresponding MSE scores on the test set, using different feature extraction methods.

The L1 distributions based on STFT, MFCC, and CQT for the cases in Experiment 2 were also compared. However, the Kruskal–Wallis H-test did not reveal any statistically significant differences.

5.2.2 Clustering Evaluation

For Experiment 2, the same parameters from Experiment 1 were used for both UMAP and DBSCAN. In UMAP, the number of neighbors was set to 50 and the minimum distance to 0. For DBSCAN, the epsilon value was set to 0.04, and the minimum number of samples was set to 4.

Table 5.8 presents the cases along with their corresponding cluster numbers, the amount of noise data, and the DBCV scores for each clustering result. The cases Exp2-LMSL-ts, Exp2-SCL-ts, Exp2-LSLP-ts, and Exp2-SCLP-ts all achieved the same DBCV score of 0.3871, indicating similar clustering performance. In contrast, Exp2-LogSL-ts and Exp2-LogSLP-ts yielded slightly lower DBCV scores of 0.3591.

Case	Cluster Number	Noise	DBCV ↑
Exp2-LMSL-ts	775	3143	**0.3871**
Exp2-SCL-ts	775	3143	**0.3871**
Exp2-LogSL-ts	778	3300	0.3591
Exp2-LSLP-ts	775	3143	**0.3871**
Exp2-SCLP-ts	775	3143	**0.3871**
Exp2-LogSLP-ts	778	3300	0.3591

Table 5.8: Clustering results for Experiment 2, including the number of clusters, noise points, and DBCV scores.

Figure 5.2 illustrates Exp2-LMSL-ts and Exp2-LogSL-ts as examples of UMAP visualizations without and with clustering using DBSCAN, respectively. These cases were selected because Exp2-LMSL-ts represents one of the best-performing clustering models, while Exp2-LogSL-ts has one of the lower DBCV scores. Additional UMAP plots for other cases are provided in Appendix B, Figure B.2.

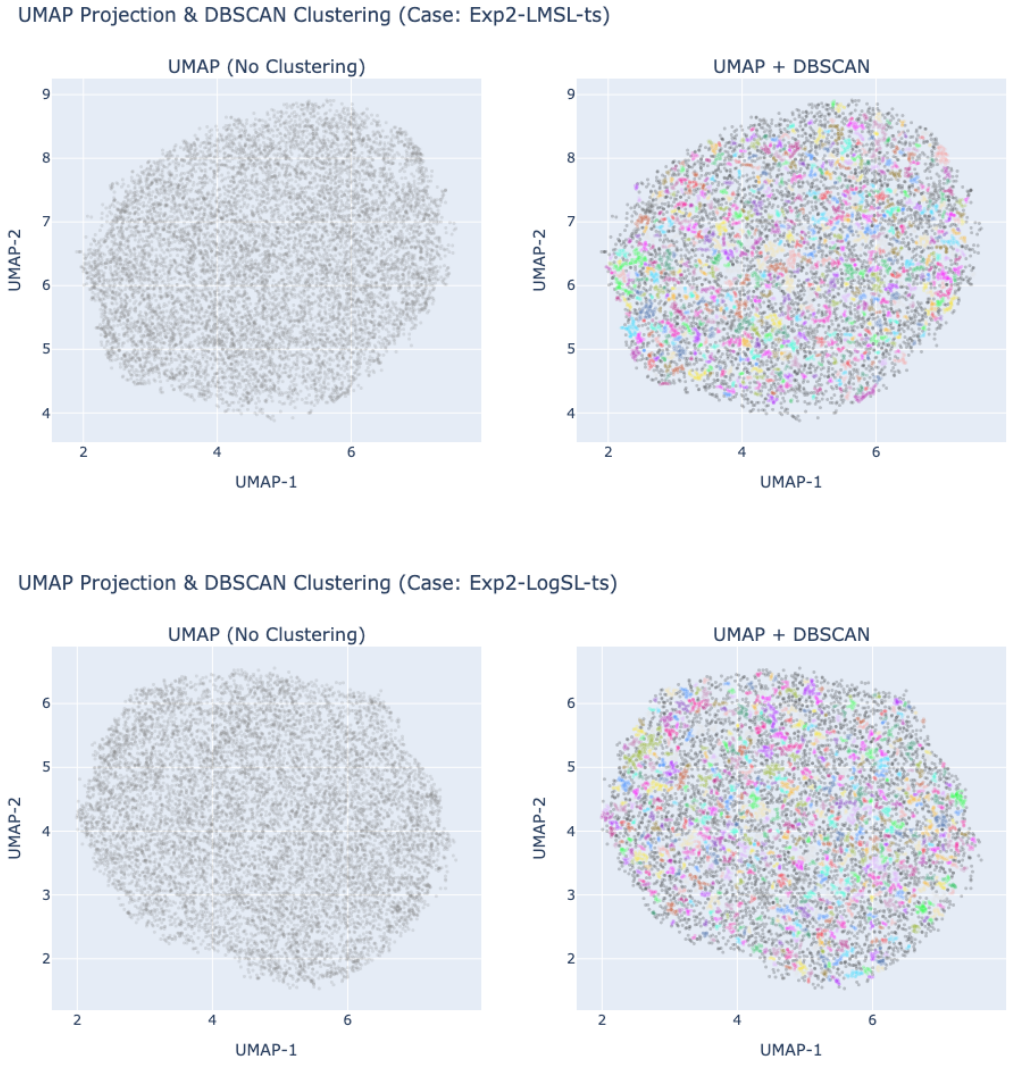

Figure 5.2: UMAP plots of latent vectors (left) and their clustered versions using DBSCAN (right). The first row shows results for Exp2-LMSL-ts, while the second row corresponds to Exp2-LogSL-ts.

5.3 Outcomes of Experiment 3

In Experiment 3, the models were trained using two approaches: for a fixed 250 epochs and with early stopping. For the models trained for 250 epochs, the learning rate was reduced after epoch 200. In contrast, a fixed learning rate was used for the early stopping cases. As observed in previous experiments, the models trained for 250 epochs produced results similar to those with early stopping. Therefore, the outcomes from the early stopping models are also considered representative for this experiment.

Table 5.9 lists the training hyperparameters and the number of epochs each case was trained for. Since these models were fine-tuned on top of a baseline model that had

already been trained for 1000 epochs, the listed epochs represent only the fine-tuning phase. The batch size used in this experiment was larger, 512, compared to previous experiments, due to the availability of more GPU cores. On the other hand, the patience parameter for early stopping was set lower than in earlier experiments, as the models had already undergone extensive pretraining, making them easier to fine-tune.

Case	Epochs	Batch Size	Learning rate	Patient
Exp3-STFT	59	512	1×10^{-4}	10
Exp3-MFCC	55	512	1×10^{-4}	10
Exp3-CQT	70	512	1×10^{-4}	10

Table 5.9: The cases in Experiment 3, along with their corresponding hyperparameters and the number of training epochs.

5.3.1 Audio Evaluation

Table 5.10 presents the FAD scores for the cases in Experiment 3. Among all metrics, Exp3-MFCC demonstrates the best performance, followed by Exp3-STFT, while Exp3-CQT shows the lowest performance, despite being trained for a higher number of epochs.

Case	Subset of Train		Test	
	VGGish ↓	PANNs ↓	VGGish ↓	PANNs ↓
Exp3-STFT	1.6308	0.5406×10^3	1.6091	0.5326×10^3
Exp3-MFCC	**1.6120**	$\mathbf{0.5047} \times 10^3$	**1.6077**	$\mathbf{0.5246} \times 10^3$
Exp3-CQT	1.6466	0.5874×10^3	1.6424	0.6041×10^3

Table 5.10: FAD scores of the cases in Experiment 3 on training subsets and test sets, based on VGGish and PANNs embeddings.

Table 5.11 presents the MSE scores based on different feature extraction methods. The results indicate that each model performed best on the spectral metric it was trained with. Exp3-STFT achieved the lowest MSE on STFT features, Exp3-MFCC on MFCC features, and Exp3-CQT on CQT features. This suggests that models are most effective when evaluated using the same feature representation they were trained on.

The L1 loss distributions for each spectral metric (STFT, MFCC, and CQT) were also compared across the cases. Although the Kruskal–Wallis H-test indicated a statistically significant difference among the distributions, the effect size suggests that the difference is not essentially meaningful.

Case	STFT MSE ↓	MFCC MSE ↓	CQT MSE ↓
Exp3-STFT	**25.5990**	1022.803	35544.23
Exp3-MFCC	25.9966	**1010.494**	35200.91
Exp3-CQT	25.7877	1025.180	**33915.05**

Table 5.11: The cases in Experiment 3 and their corresponding MSE scores on the test set, using different feature extraction methods.

5.3.2 Clustering Evaluation

For this experiment, the same UMAP parameters were used as in the previous experiments: the number of neighbors was set to 50, and the minimum distance was 0. However, the epsilon parameter in DBSCAN was slightly increased to 0.05, while the minimum samples parameter remained at 4. The increase in epsilon was made because the UMAP plots showed more clearly separated latent vectors, particularly for Exp3-STFT and Exp3-MFCC, as illustrated in Figure 5.3.

Table 5.12 presents the number of clusters, noise samples, and the corresponding DBCV scores. Among the cases, Exp3-STFT demonstrates the highest clustering quality with a DBCV score of 0.3626, followed by Exp3-MFCC with a score of 0.3519. However, Exp3-CQT shows relatively lower clustering quality, with a DBCV score of 0.3129, compared to the other cases.

Case	Cluster Number	Noise	DBCV ↑
Exp3-STFT	656	4039	**0.3626**
Exp3-MFCC	646	4449	0.3519
Exp3-CQT	648	2543	0.3129

Table 5.12: Clustering results for Experiment 3, including the number of clusters, noise points, and DBCV scores.

5.4 Answering Research Questions

This section provides direct answers to the research questions:

- **RQ1:** Experiment 1 and Experiment 2 were designed to answer the first research question, which focuses on identifying which loss function formulations perform better in terms of reconstruction accuracy and latent space organization. As detailed in the previous sections, the models did not show significant differences in reconstruction quality or latent space structure. They achieved similar results based on FAD scores and spectral metrics. However, the models differed in training efficiency. In terms of clustering performance, the Exp1-LMSL model achieved the highest DBCV score among all evaluated models.
- **RQ2:** Experiment 3 addresses this research question, which investigates which feature extraction method is most effective at capturing sound features. Accord-

Figure 5.3: UMAP plots of latent vectors (left) and their clustered versions using DBSCAN (right).

ing to the FAD evaluation, MFCC demonstrated slightly better performance. On the other hand, spectral metrics indicated that each feature extraction method performed best when the evaluation used the same representation as the one it was trained on. Additionally, the model using the STFT method achieved slightly better results in terms of clustering quality.

5.5 Example Results

This section presents selected examples to illustrate the quality of reconstruction, the organization of the latent space, and the clustering capabilities. The Exp1-LMSL case was chosen for these examples because it demonstrated slightly better performance across most reconstruction quality metrics and achieved the highest DBCV score.

5.5.1 Latent Space Interpolation

Interpolating between two different audio samples in the latent space can reveal both the organization of the latent space and the model's reconstruction capabilities.

To explore this, two audio samples were selected from the test dataset to evaluate the model's performance with unseen data. The parameters of these samples were encoded into the latent space, and then five linear interpolation steps were generated between them. The resulting latent vectors were decoded to to obtain reconstructed parameters. Next, these parameters are used to synthesize the corresponding audio signals.

Figure 5.4 displays the waveforms of the two original audio samples selected from the test dataset. Waveforms were chosen to represent these signals for easier visual comparison across interpolation steps. In the plots, the y-axis represents the amplitude of the sound, while the x-axis corresponds to time.

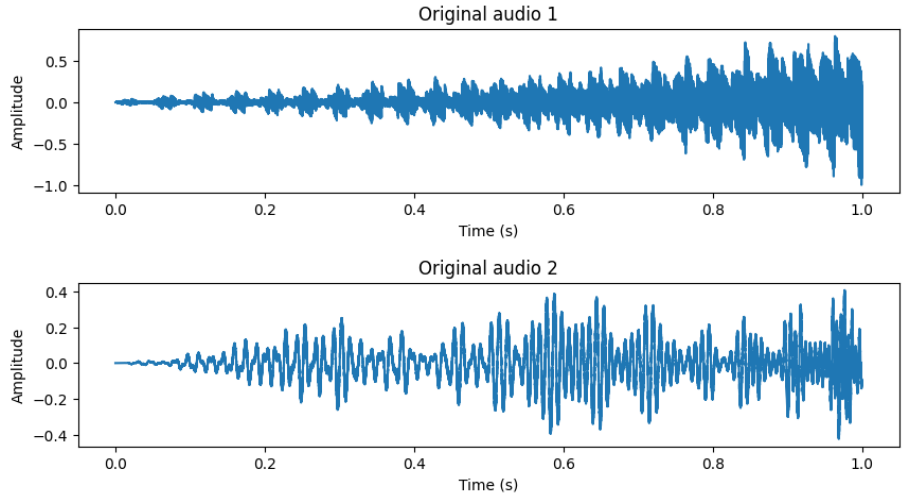

Figure 5.4: The waveform of two samples from test set.

Figure 5.5 illustrates the reconstructed versions of the two original audio samples, along with the five linear interpolation steps between them.

5.5.2 Original and Reconstructed Sound Comparison

To analyze the reconstruction quality of the Exp1-LMSL model, two examples from the test set are presented in Figure 5.6 and Figure 5.7. These examples demonstrate the model's ability to reconstruct audio from unseen data. Both figures include

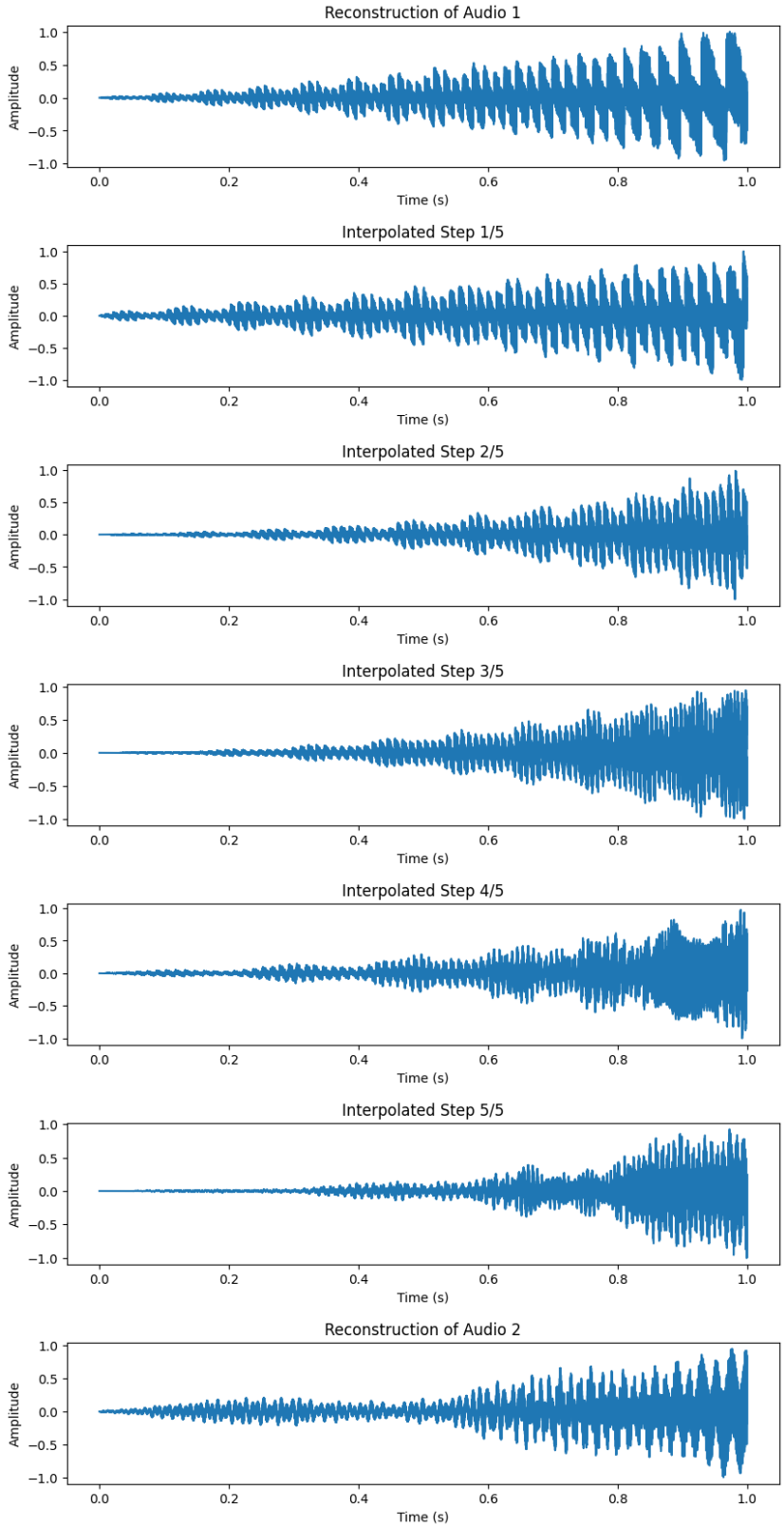

Figure 5.5: Reconstructed audio samples and interpolation steps between them from the Exp1-LMSL model.

overlaid waveform comparisons and spectrograms of the original and reconstructed audio signals, providing detailed understanding of the signals in both the time and frequency domains.

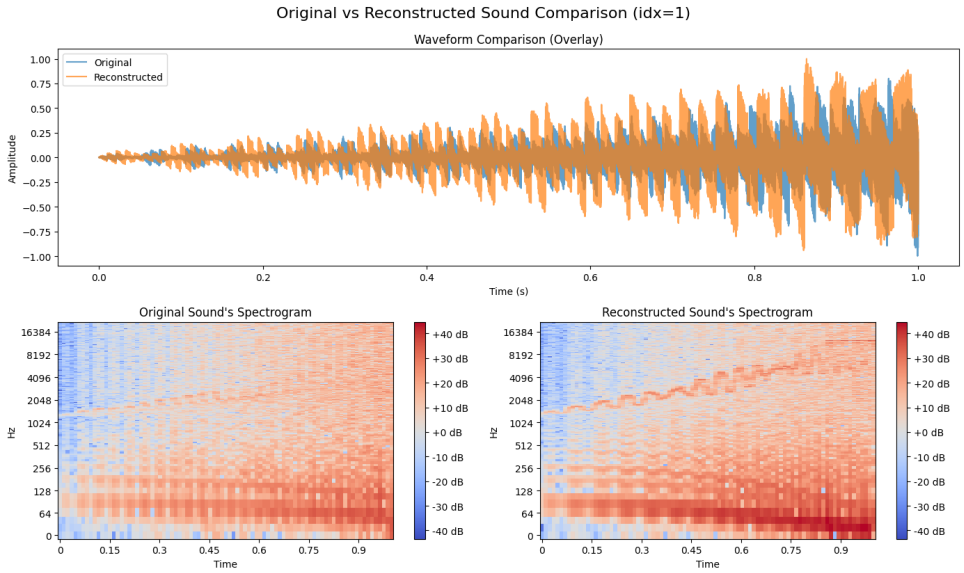

Figure 5.6: Original and reconstructed audio signals of index sound 1, including overlaid waveforms and spectrograms from the Exp1-LMSL model.

Figure 5.7 illustrates the reconstruction capability of the Exp1-LMSL model for test signal index 88, while Figure 5.8 shows the corresponding reconstruction from the Exp1-LSLP model. This comparison highlights differences in their ability to preserve phase information during reconstruction.

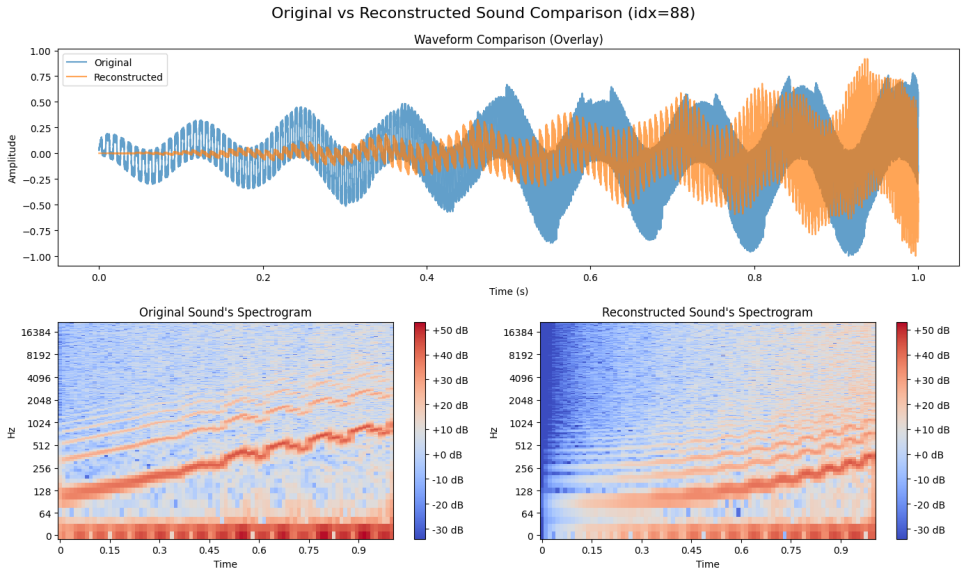

Figure 5.7: Original and reconstructed audio signals of index sound 88, including overlaid waveforms and spectrograms generated by the Exp1-LMSL model.

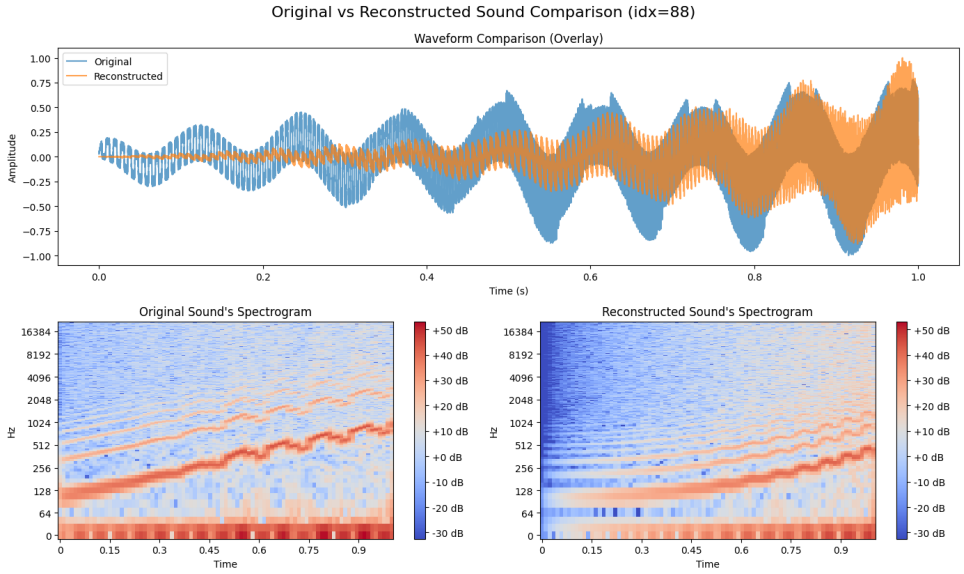

Figure 5.8: Original and reconstructed audio signals of index sound 88, including overlaid waveforms and spectrograms generated by the Exp1-LSLP model.

5.5.3 Clustering Analysis

Three mid-sized clusters were randomly selected from the latent space of the Exp1-LSLP model, using a subset of the training set, to examine the organization of the UMAP representation. The Exp1-LSLP model was chosen because it achieved the highest DBCV score for clustering.

Each selected cluster contains 15 samples. Below, the average FFT of the samples within each cluster is plotted over frequency, and the positions of the clusters on the UMAP are also visualized.

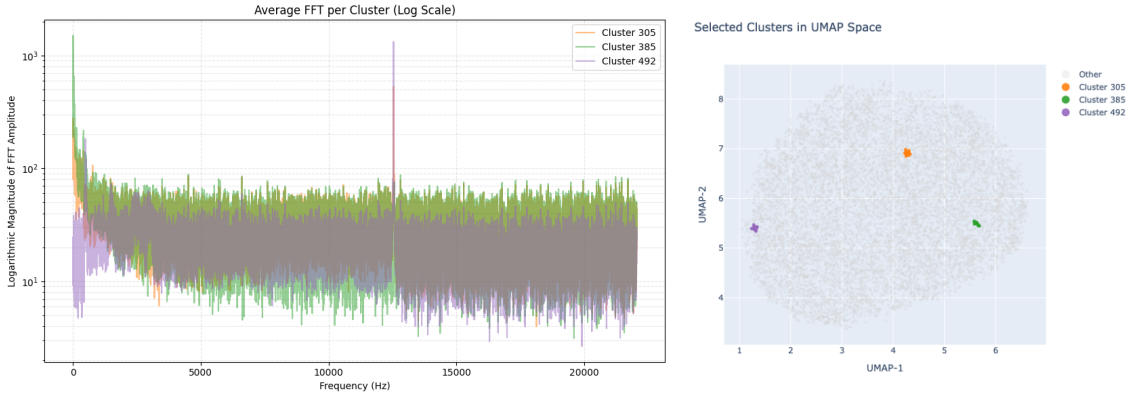

Figure 5.9: Average FFT scores across frequencies (left) and cluster locations visualized on the UMAP (right).

6

Discussion

This chapter discusses the major findings from the results presented in Chapter 5. Section 6.1 provides a detailed analysis of the models' reconstruction capabilities, while Section 6.2 examines the clustering performance.

6.1 Findings on Reconstruction Quality

The results of FAD and spectral MSE metrics within each experiment were generally consistent. Interestingly, in most cases, models trained with a magnitude-only loss function and those trained with the same loss function augmented with phase information (e.g., SCL vs. SCLP) required a similar number of training epochs and produced nearly identical FAD and spectral MSE scores. This suggests that, under the given conditions, incorporating angle information may not significantly impact model performance. However, as shown in the reconstruction comparison between Exp1-LMSL (Figure 5.7) and Exp1-LSLP (Figure 5.8), the inclusion of phase information appeared to improve waveform alignment, particularly at the peaks. Despite this visual improvement, the difference remained minimal when evaluated using FAD and spectral MSE metrics.

In the first experiment, Exp1-LogSL and Exp1-LogSLP required significantly fewer training epochs compared to the other models. This suggests that LogSL and LogSLP enable the models to learn audio features more efficiently.

By comparing the number of training epochs between the models in Experiment 1 and Experiment 2, where the hyperparameters were kept the same but Experiment 2 included an additional TorchSynth layer after the decoder, it appears that the added layer helped the models learn more efficiently. In most cases, models in Experiment 2 required fewer epochs to converge compared to those in Experiment 1, with the exceptions of Exp2-LogSL-ts and Exp2-LogSLP-ts. Although these models required more epochs than their counterparts in Experiment 1, they still needed fewer epochs than the other models in Experiment 2.

When comparing the reconstruction quality of the two experiments, models from Experiment 1 achieved slightly better results in terms of both FAD and spectral MSE scores. This suggests that although adding the synth layer in Experiment 2 improves training efficiency, the models may require more training to reach the same level of accuracy as those in Experiment 1.

In Experiment 3, the model using MFCC as the feature extraction method performed slightly better than the others in terms of FAD scores. This suggests that MFCC may be more effective in helping the model learn perceptual features of sound.

Another finding from Experiment 3 is that models trained with a specific feature extraction method tend to perform better when evaluated using the MSE score of that same method. This suggests that models are most effective when assessed using the same feature representation they were trained on. Additionally, this indicates that the models successfully learned audio features through the feature extraction method used in the loss function.

The experiment 3 showed the lowest FAD and spectral MSE scores compared to the other experiments. This may be attributed to the fact that the models were primarily trained using only reconstruction parameter loss, without incorporating any audio-related loss functions. Since these parameters are not necessarily directly related to audio features, the models may have struggled to capture meaningful audio representations.

Another possible reason for the models' lower performance in Experiment 3, compared to the other experiments, could be the nature of the audio-related loss function used. As shown in Equation 4.1, this loss function produces relatively low output values due to normalization. This may not sufficiently encourage the model to focus on learning audio features. Increasing the weight of this loss component could potentially improve the model's ability to capture spectral characteristics.

Additionally, the model Exp3-CQT required significantly more training time compared to Exp3-STFT and Exp3-MFCC, due to the higher computational cost of the CQT feature extraction. It also took the longest to train, reaching 70 epochs. Under these conditions, its associated loss function, CQT-SCLP, may not be an ideal choice.

As shown in the waveform examples in Figures 5.7 and 5.8, the amplitude of the reconstructed sounds tends to start low and gradually increase toward the end of the sound. This pattern is commonly observed across the reconstructed outputs of the models. One possible explanation is that the dataset contains many examples with similar amplitude envelopes, leading the model to replicate this trend. Additionally, the synthesizer parameters might be insufficient to represent the envelope of the sounds effectively. Another possible explanation is that the feature extraction methods used in the project do not contain envelope information directly, making it difficult for the model to capture this aspect of signals.

One limitation is the example of linear interpolation in the latent space was visualized using only waveforms, as shown in Figure 5.5. This choice was made to simplify the step-by-step comparison and due to space constraints on a single page. However, using only waveform representations may not fully capture or reflect the underlying audio features. Moreover, to obtain more generalized and interpretable interpolation results, additional steps should be taken such as perceptual metrics, or listener evaluations.

A factor that may limit audio quality is the 1:3 ratio between reconstruction loss and spectral loss, which may not be sufficient to strongly encourage the model to learn

the spectral features of sounds. Therefore, assigning a larger weight β to the spectral loss component in the loss function, as shown in Equation 4.4, could help the model place greater emphasis on spectral characteristics during training. Moreover, the latent space used in this project is relatively small compared to those in many studies in the literature. This limited dimensionality may restrict the model's capacity to capture deeper insights from the data.

6.2 Findings on Clustering Quality

The model Exp1-LMSL achieved the highest DBCV score among all models. Most models scored around 0.35–0.38, while Exp3-CQT showed a lower score of 0.31. However, it is important to note that these scores may not directly reflect the true quality of the latent space. Since the evaluation is based on a reduced-dimensional representation of the latent space, some information may be lost during the dimensionality reduction process. Additionally, both the UMAP dimensionality reduction method and the DBSCAN clustering algorithm are sensitive to their respective hyperparameters, which can significantly influence the results.

There were 10,000 samples in the visualized latent spaces, with more than 600 clusters identified in all cases. Additionally, around 3,000 samples were classified as noise. These numbers are reasonable given the high diversity of sounds in the dataset. As a result, it was expected to observe many small clusters and a significant number of samples labeled as noise by the clustering algorithm.

In most cases, the shape of the samples in the UMAP plot for Experiment 1 resembled a circle, as shown in Figure 5.1. In contrast, the shape in Experiment 2 was more elliptical, as illustrated in Figure 5.2. Interestingly, the UMAP plot for Experiment 3 displayed a rougher circular shape compared to Experiment 1. Moreover, the grouping in the Exp3-STFT and Exp3-MFCC plots was noticeably clearer than in the other visualizations.

One notable observation about the general shape of the UMAP projections was that they tend to resemble circular or elliptical forms, rather than appearing arbitrary. One possible explanation is that the KL divergence component of the loss function may lead the model's latent space to be similar to a Gaussian distribution. Another contributing factor could be the dimensionality reduction itself which from a relatively high-dimensional space (16 dimensions) to just 2. This may naturally result in more rounded shapes in the projection.

The clustering analysis presented in Section 5.4.3 highlights three distinct clusters selected from different regions of the UMAP visualization. The average FFT values of these clusters were shown to illustrate their differences in audio features. However, it is difficult to interpret the differences between clusters using this plot, as it appears noisy and lacks clear patterns. Moreover, this example does not provide a generalized view of the overall clustering structure. Therefore, further steps should be taken to analyze the clusters more deeply such as computing the cosine similarity between sounds.

Another approach to analyzing the clustering structure of the latent space would be applying DBSCAN first and then reducing the dimension using UMAP. With this method, the true structure of the latent space could be clustered without losing information during dimensionality reduction. However, since there are many small clusters in the latent space due to sound diversity, it might become harder to observe the general structure. This is because different clusters must share the same colors, as there are not enough distinct colors to represent each cluster individually.

7

Conclusion

In this study, various spectral loss functions and feature extraction methods (STFT, MFCC, and CQT) were explored within a VAE framework based on synthesizer parameters. Three experimental setups were designed, evaluating a total of 15 models for their performance in sound reconstruction and the organization of the latent space. Reconstruction quality was assessed using FAD scores based on VGGish and PANNs embeddings, as well as spectral MSE and L1 loss computed using STFT, MFCC, and CQT features of sounds. Clustering quality in the reduced-dimensional latent space was evaluated using the DBCV metric.

The first experiment focused on comparing different spectral loss functions using STFT as the feature extraction method. These loss functions included LMSL, SCL, LogSL, and their respective versions incorporating phase information. The second experiment assessed the impact of a differentiable audio synthesis technique by incorporating a TorchSynth layer after the decoder, using the same loss functions from the first experiment. The third experiment examined the effectiveness of different feature extraction methods such as STFT, MFCC, and CQT when used within the Spectral Convergence loss. In this setup, models were initially trained using only synthesizer parameters, and the spectral loss functions were introduced afterward to evaluate their impact.

The FAD and spectral loss metrics indicate that there were no significant differences in reconstruction quality between cases in both Experiment 1 and Experiment 2. However, even though the reconstruction quality score of Experiment 2 was slightly lower compared to the scores of Experiment 1, most models in Experiment 2 learned more effectively and required fewer training epochs. Experiment 3 suggested that initially training with only synthesizer parameters and then introducing audio feature extraction methods was less effective for reconstruction quality compared to the other experiments.

The model Exp1-LMSL achieved the highest DBCV score among all models, indicating better clustering quality. However, all models exhibited a significant number of clusters and noise, which may be attributed to the high diversity of sounds represented in the latent space.

In summary, this study systematically compared the effects of different audio-related loss functions and audio feature extraction methods on a VAE framework, evaluating both reconstruction quality and latent space organization through detailed analysis.

Additionally, it explored the effectiveness of incorporating differentiable synthesis layers into the model architecture.

7.1 Future Work

This study can be improved in several ways in the future:

- **Improving the envelope of reconstructed audio:** The envelope feature of reconstructed sounds can be enhanced by incorporating envelope characteristics into future models. This could be achieved by using feature extraction methods specifically designed to capture amplitude envelopes.
- **Focused model exploration:** The results of this study can guide future researchers to focus on the most promising model configurations for more in-depth investigation and optimization.
- **Human listening evaluations:** For a deeper understanding of reconstruction quality, listening tests with human participants could be organized for subjective evaluation. The results of these evaluations can be compared with the metrics used in this study to validate the accuracy of the current evaluation approaches.
- **Deeper latent space analysis:** To gain a better understanding of the latent space's structure and organization, more generalized comparisons between clusters could be performed. For example, using cosine similarity within and across clusters could provide insights into how well similar sounds are grouped in the latent space.
- **Analyzing the effect of synthesizer parameters on the latent space:** To better understand how synthesizer parameters influence the latent space, each parameter can be systematically varied while observing the resulting changes in the latent representations. This analysis could reveal whether the latent space responds linearly or nonlinearly to specific parameter changes.
- **Interactive latent space tools:** Developing interactive tools for exploring the latent space could benefit musicians and producers by providing audio feature based representation of sounds of sounds for creative exploration.

7.2 Ethical Considerations

This project adheres to ethical standards of NIME (New Interfaces for Musical Expression) that is institution acknowledged for highest ethical guideline for research and artistic production. The project involves generating a sound dataset using open-source libraries, ensuring that there are no ethical concerns related to data privacy or the use of proprietary content. Moreover, the project was utilized open-source libraries and the finalized code for the project is available [1] to everyone.

[1] `https://github.com/ipekkorkmz/SynthParam-VAE`

Bibliography

[1] C. Roads, *Composing electronic music: a new aesthetic*. Oxford University Press, 2015.

[2] *Rhythm*, in *Cambridge Learner's Dictionary*, Cambridge University Press. [Online]. Available: `https://dictionary.cambridge.org/dictionary/learner-english/rhythm` (visited on 02/09/2025).

[3] Z. Chen, Y. Jing, S. Yuan, Y. Xu, J. Wu, and H. Zhao, "Sound2synth: Interpreting sound via fm synthesizer parameters estimation," *arXiv preprint arXiv:2205.03043*, 2022.

[4] C. Roads, *The computer music tutorial*. MIT press, 1996.

[5] F. Lidström, M. Lidström, *Patternarium*. [Online]. Available: `https://soniccharge.com/patternarium`.

[6] P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos, "Universal audio synthesizer control with normalizing flows. arxiv 2019," *arXiv preprint arXiv:1907.00971*,

[7] C. Meyer. "Vco." (Nov. 2016), [Online]. Available: `https://learningmodular.com/glossary/vco` (visited on 05/05/2025).

[8] J. Stolet. "Electronic music interactive, 2nd edition." (2011), [Online]. Available: `https://pages.uoregon.edu/emi/index.php` (visited on 04/29/2025).

[9] "What is vca?" (n.d.), [Online]. Available: `https://mackie.com/en/blog/all/what_vca.html` (visited on 05/05/2025).

[10] Wikipedia contributors. "Envelope (music)." (n.d.), [Online]. Available: `https://en.wikipedia.org/wiki/Envelope_(music)` (visited on 02/17/2025).

[11] productionmusiclive. "How to use the adsr envelope." (n.d.), [Online]. Available: `https://www.productionmusiclive.com/blogs/news/how-to-use-the-adsr-envelope` (visited on 02/17/2025).

[12] G. Sharma, K. Umapathy, and S. Krishnan, "Trends in audio signal feature extraction methods," *Applied Acoustics*, vol. 158, p. 107 020, 2020.

[13] M. Müller, "Short-time fourier transform and chroma features," *Lab Course, Friedrich-Alexander-Universität Erlangen-Nürnberg*, 2015.

[14] W. T. Cochran, J. W. Cooley, D. L. Favin, *et al.*, "What is the fast fourier transform?" *Proceedings of the IEEE*, vol. 55, no. 10, pp. 1664–1674, 1967.

[15] D. Gabor, "Theory of communication. part 1: The analysis of information," *Journal of the Institution of Electrical Engineers-part III: radio and communication engineering*, vol. 93, no. 26, pp. 429–441, 1946.

[16] M. Bastiaans, "On the sliding-window representation in digital signal processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 4, pp. 868–873, 1985. DOI: 10.1109/TASSP.1985.1164653.

[17] S. Umesh, L. Cohen, and D. Nelson, "Fitting the mel scale," in *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, IEEE, vol. 1, 1999, pp. 217–220.

[18] Z. K. Abdul and A. K. Al-Talabani, "Mel frequency cepstral coefficient and its applications: A review," *IEEE Access*, vol. 10, pp. 122 136–122 158, 2022. DOI: 10.1109/ACCESS.2022.3223444.

[19] B. McFee, C. Raffel, D. Liang, *et al.*, "Librosa: Audio and music signal analysis in python.," *SciPy*, vol. 2015, pp. 18–24, 2015.

[20] C. Schörkhuber and A. Klapuri, "Constant-q transform toolbox for music processing," in *7th sound and music computing conference, Barcelona, Spain*, SMC, 2010, pp. 3–64.

[21] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, *Learning internal representations by error propagation*, 1985.

[22] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *Machine learning for data science handbook: data mining and knowledge discovery handbook*, pp. 353–374, 2023.

[23] A. Singh and T. Ogunfunmi, "An overview of variational autoencoders for source separation, finance, and bio-signal applications," *Entropy*, vol. 24, no. 1, p. 55, 2021.

[24] G. Harshvardhan, M. K. Gourisaria, M. Pandey, and S. S. Rautaray, "A comprehensive survey and analysis of generative models in machine learning," *Computer Science Review*, vol. 38, p. 100 285, 2020.

[25] A. Oussidi and A. Elhassouny, "Deep generative models: Survey," in *2018 International conference on intelligent systems and computer vision (ISCV)*, IEEE, 2018, pp. 1–8.

[26] L. Ruthotto and E. Haber, "An introduction to deep generative modeling," *GAMM-Mitteilungen*, vol. 44, no. 2, e202100008, 2021.

[27] S. Odaibo, "Tutorial: Deriving the standard variational autoencoder (vae) loss function," *arXiv preprint arXiv:1907.08956*, 2019.

[28] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[29] "Latent space." (n.d.), [Online]. Available: https://www.promptlayer.com/glossary/latent-space (visited on 02/09/2025).

[30] B. M. Dillon, T. Plehn, C. Sauer, and P. Sorrenson, "Better latent spaces for better autoencoders," *SciPost Physics*, vol. 11, no. 3, p. 061, 2021.

[31] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[32] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[33] Y. Liu, E. Jun, Q. Li, and J. Heer, "Latent space cartography: Visual analysis of vector space embeddings," in *Computer graphics forum*, Wiley Online Library, vol. 38, 2019, pp. 67–78.

[34] T. Sainburg, M. Thielk, and T. Q. Gentner, "Latent space visualization, characterization, and generation of diverse vocal communication signals," *BioRxiv*, p. 870 311, 2019.

[35] L. McInnes, *Umap: Uniform manifold approximation and projection for dimension reduction*, 2018. [Online]. Available: `https://umap-learn.readthedocs.io/`.

[36] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, 1996, pp. 226–231.

[37] K. Khan, S. U. Rehman, K. Aziz, S. Fong, and S. Sarasvady, "Dbscan: Past, present and future," in *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, IEEE, 2014, pp. 232–238.

[38] D. Moulavi, P. A. Jaskowiak, R. J. Campello, A. Zimek, and J. Sander, "Density-based clustering validation," in *Proceedings of the 2014 SIAM international conference on data mining*, SIAM, 2014, pp. 839–847.

[39] J. A. Hartigan, *Clustering algorithms*. John Wiley & Sons, Inc., 1975.

[40] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fr\'echet audio distance: A metric for evaluating music enhancement algorithms," *arXiv preprint arXiv:1812.08466*, 2018.

[41] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[42] tensorflow, *Models*. [Online]. Available: `https://github.com/tensorflow/models/tree/master/research/audioset`.

[43] D. Dowson and B. Landau, "The fréchet distance between multivariate normal distributions," *Journal of multivariate analysis*, vol. 12, no. 3, pp. 450–455, 1982.

[44] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[45] Y. Wu, K. Chen, T. Zhang, Y. Hui, T. Berg-Kirkpatrick, and S. Dubnov, "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023, pp. 1–5.

[46] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High fidelity neural audio compression," *arXiv preprint arXiv:2210.13438*, 2022.

[47] P. E. McKight and J. Najab, "Kruskal-wallis test," *The corsini encyclopedia of psychology*, pp. 1–1, 2010.

[48] Wikipedia contributors. "Effect size." (n.d.), [Online]. Available: `https://en.wikipedia.org/wiki/Effect_size` (visited on 06/02/2025).

[49] B. Hayes, J. Shier, G. Fazekas, A. McPherson, and C. Saitis, "A review of differentiable digital signal processing for music and speech synthesis," *Frontiers in Signal Processing*, vol. 3, p. 1 284 100, 2024.

[50] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, “Ddsp: Differentiable digital signal processing,” *arXiv preprint arXiv:2001.04643*, 2020.

[51] K. Tatar, K. Cotton, and D. Bisig, “Sound design strategies for latent audio space explorations using deep learning architectures,” *arXiv preprint arXiv:2305.15571*, 2023.

[52] J. Turian, J. Shier, G. Tzanetakis, K. McNally, and M. Henry, “One billion audio sounds from gpu-enabled modular synthesis,” in *2021 24th International Conference on Digital Audio Effects (DAFx)*, IEEE, 2021, pp. 222–229.

[53] J. Turian, J. Shier, G. Tzanetakis, K. McNally, and M. Henry, *Figure 3: Module configuration for the voice in torchsynth*, Reproduced from the paper "One billion audio sounds from GPU-enabled modular synthesis", Presented at the 24th International Conference on Digital Audio Effects (DAFx), 2021.

[54] S. Ö. Arık, H. Jun, and G. Diamos, “Fast spectrogram inversion using multi-head convolutional neural networks,” *IEEE Signal Processing Letters*, vol. 26, no. 1, pp. 94–98, 2018.

[55] L. Garber, T. Ciccola, and J. C. Amusategui, “Audiostellar, an open source corpus-based musical instrument for latent sound structure discovery and sonic experimentation,” in *Proceedings of ICMC*, 2020.

[56] L. Garber, T. Ciccola, and J. C. Amusategui, *Figure 3: Machine learning pipeline*, Reproduced from the paper "AudioStellar, an open source corpus-based musical instrument for latent sound structure discovery and sonic experimentation", Presented at ICMC 2020, 2020.

A

Appendix

Table A.1: All module parameters and value ranges (1/2)

Module	**Parameter**	**Range**
keyboard	midi_f0	(0, 127)
keyboard	duration	(0.01, 4.0)
adsr_1	attack	(0.0, 2.0)
adsr_1	decay	(0.0, 2.0)
adsr_1	sustain	(0.0, 1.0)
adsr_1	release	(0.0, 5.0)
adsr_1	alpha	(0.1, 6.0)
adsr_2	attack	(0.0, 2.0)
adsr_2	decay	(0.0, 2.0)
adsr_2	sustain	(0.0, 1.0)
adsr_2	release	(0.0, 5.0)
adsr_2	alpha	(0.1, 6.0)
lfo_1	frequency	(0.0, 20.0)
lfo_1	mod_depth	(-10.0, 20.0)
lfo_1	initial_phase	$(-\pi, \pi)$
lfo_1	sin	(0.0, 1.0)
lfo_1	tri	(0.0, 1.0)
lfo_1	saw	(0.0, 1.0)
lfo_1	rsaw	(0.0, 1.0)
lfo_1	sqr	(0.0, 1.0)
lfo_2	frequency	(0.0, 20.0)
lfo_2	mod_depth	(-10.0, 20.0)
lfo_2	initial_phase	$(-\pi, \pi)$
lfo_2	sin	(0.0, 1.0)
lfo_2	tri	(0.0, 1.0)
lfo_2	saw	(0.0, 1.0)
lfo_2	rsaw	(0.0, 1.0)
lfo_2	sqr	(0.0, 1.0)

Table A.2: All module parameters and value ranges (2/2)

Module	**Parameter**	**Range**
lfo_1_amp_adsr	attack	(0.0, 2.0)
lfo_1_amp_adsr	decay	(0.0, 2.0)
lfo_1_amp_adsr	sustain	(0.0, 1.0)
lfo_1_amp_adsr	release	(0.0, 5.0)
lfo_1_amp_adsr	alpha	(0.1, 6.0)
lfo_2_amp_adsr	attack	(0.0, 2.0)
lfo_2_amp_adsr	decay	(0.0, 2.0)
lfo_2_amp_adsr	sustain	(0.0, 1.0)
lfo_2_amp_adsr	release	(0.0, 5.0)
lfo_2_amp_adsr	alpha	(0.1, 6.0)
lfo_1_rate_adsr	attack	(0.0, 2.0)
lfo_1_rate_adsr	decay	(0.0, 2.0)
lfo_1_rate_adsr	sustain	(0.0, 1.0)
lfo_1_rate_adsr	release	(0.0, 5.0)
lfo_1_rate_adsr	alpha	(0.1, 6.0)
lfo_2_rate_adsr	attack	(0.0, 2.0)
lfo_2_rate_adsr	decay	(0.0, 2.0)
lfo_2_rate_adsr	sustain	(0.0, 1.0)
lfo_2_rate_adsr	release	(0.0, 5.0)
lfo_2_rate_adsr	alpha	(0.1, 6.0)
mod_matrix	adsr_1→vco_1_pitch	(0.0, 1.0)
mod_matrix	adsr_2→vco_2_pitch	(0.0, 1.0)
mod_matrix	lfo_1→vco_1_amp	(0.0, 1.0)
mod_matrix	lfo_2→vco_2_amp	(0.0, 1.0)
mod_matrix	adsr_1→noise_amp	(0.0, 1.0)
vco_1	tuning	(-24.0, 24.0)
vco_1	mod_depth	(-96.0, 96.0)
vco_1	initial_phase	$(-\pi, \pi)$
vco_2	tuning	(-24.0, 24.0)
vco_2	mod_depth	(-96.0, 96.0)
vco_2	initial_phase	$(-\pi, \pi)$
vco_2	shape	(0, 1)
mixer	vco_1	(0.0, 1.0)
mixer	vco_2	(0.0, 1.0)
mixer	noise	(0.0, 1.0)

B

Appendix

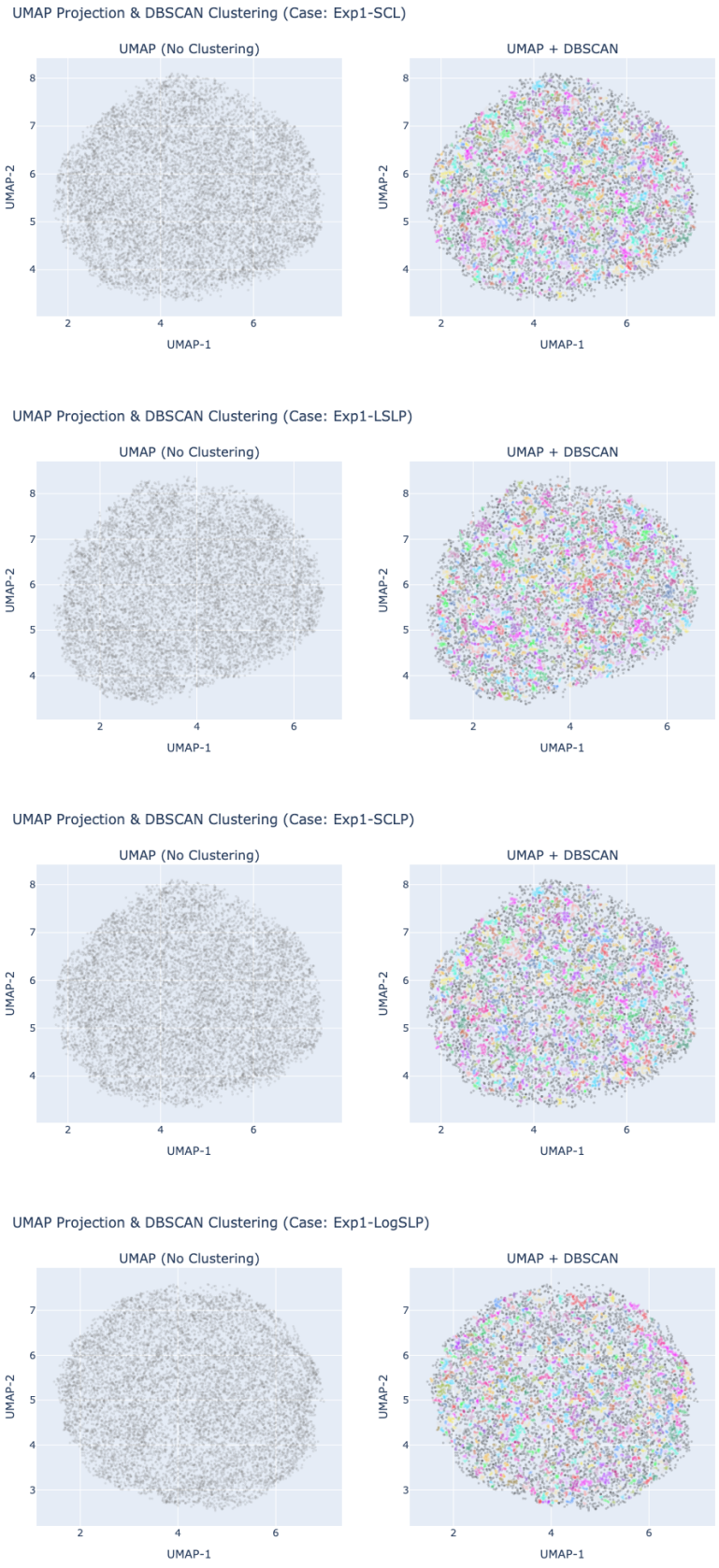

Figure B.1: UMAP plots of latent vectors (left) and their clustered versions using DBSCAN (right).

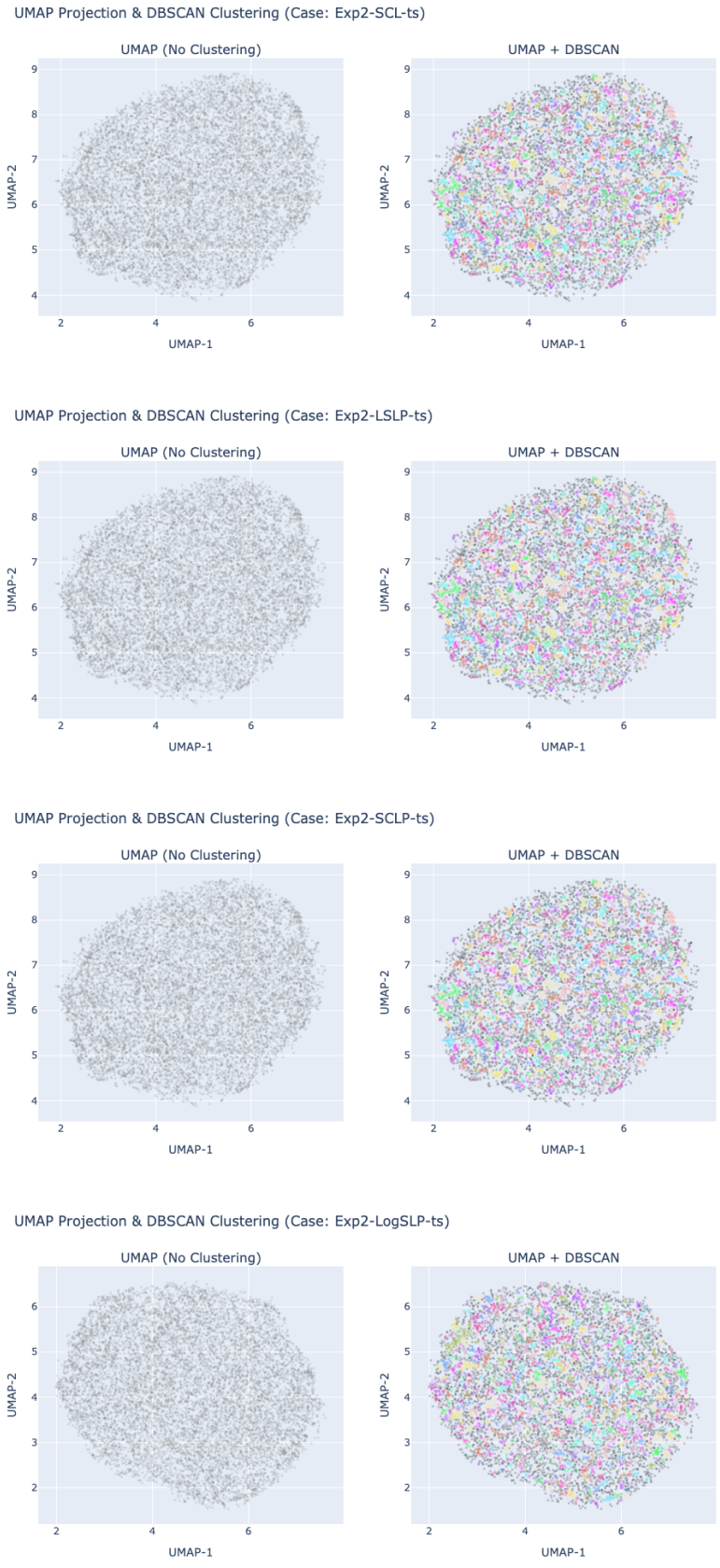

Figure B.2: UMAP plots of latent vectors (left) and their clustered versions using DBSCAN (right).