



Degree Project in Computer Science and Engineering

Second cycle, 30 credits

# **Developing a Method for Investigating the Population of Vocalists Heard in AI-Generated Music**

**TORE NYLÉN**



# **Developing a Method for Investigating the Population of Vocalists Heard in AI-Generated Music**

TORE NYLÉN

Degree Programme in Computer Science and Engineering  
Date: December 3, 2025

Supervisor: Bob L. T. Sturm  
Examiner: Sten Ternström

School of Electrical Engineering and Computer Science  
Swedish title: Utveckling av en metod som möjliggör undersökning av  
distributionen av sångröster som finns i AI-genererad musik



## Abstract

Despite the increase of music generated by Artificial Intelligence (AI) tools such as Suno and Udio, little research has been done on the songs they create. Previous research has largely focused on techniques for AI music detection, while the potential biases and patterns in the vocals the models generate have been left unanalyzed. This thesis aims to develop a pipeline that allows for population analysis of singing voices present in music generated by Suno and Udio. In order to accomplish this, we investigate two types of methods. One approach uses Mel-Frequency Cepstrum Coefficients (MFCCs) for feature extraction together with Gaussian Mixture Models (GMMs) to model vocal characteristics. The other approach uses deep learning models to extract features directly, with two speaker recognition models and one singing voice representation model. We evaluate both methods through testing with real songs that we process using source separation and silence removal. Based on the initial test results we then apply the most reliable model — the singer representation model — to the dataset of AI singers and use K-medoids clustering as well as Uniform Manifold Approximation and Projection (UMAP) dimensionality reduction to examine the data. The results from the singer representation model showed only limited quantitative success with the K-medoids clustering, while qualitative testing on real songs suggests the approach is somewhat successful. The UMAP projection showed quite distinct separation between the Suno and Udio songs, as well as between female and male vocals. Issues with distortion or incorrectly converted audio files in the datasets we used were discovered, which likely had significant impact on the clustering and visualization results. An implementation error in our UMAP usage of the MFCC approach initially showed it as being much less reliable than further testing seemed to indicate, making this method possibly interesting for more thorough testing. We motivate further research into how the voice characteristics of the generated vocals relate to their textual prompts and vocals of real artists.

## Keywords

Generative AI, Source separation, Speaker recognition, Singing voice, Music information retrieval



## Sammanfattning

Trots att mängden musik som genererats av AI-verktyg som Suno och Udio ökar så har väldigt lite forskning undersökt musiken som genererats. Tidigare forskning har fokuserat på att utveckla metoder för att känna igen AI-genererad musik. System för att upptäcka mönster och likheter i de genererade rösterna har inte varit i fokus. I den här studien utvecklar vi ett system för att kunna analysera fördelningen och variationen av rösterna som förekommer i musiken som genererats av Suno och Udio. För att åstadkomma det har vi undersökt två metoder. En av metoderna använder sig av Mel-Frequency Cepstrum Coefficients (MFCC:er) för att extrahera information från sångerna och Gaussian Mixture Models (GMM:er) för att modellera röstkaraktistiken. Den andra metoden använder sig av förtränade djupinlärningsmodeller för att extrahera data. Vi undersöker två tal- och talarigenkänningsmodeller och en sångaridentifieringsmodell för detta ändamål. Båda metoderna testas först med musik av riktiga sångare, där ljudfilerna har processerats med verktyg för att isolera rösterna och ta bort tystnaden från dem. På de testerna fick modellen som tränats för att känna igen sångare bäst resultat, varpå den användes för att undersöka Suno- och Udio-låtarna. För att analysera och projicera ner datan till en graf används K-medoids-klustering och Uniform Manifold Approximation and Projection (UMAP). Resultaten från de testerna visade på att modellen hade begränsad kvantitativ framgång men kvalitativa lyssningstester indikerade att den underliggande metoden fungerande. Graferna från UMAP-projektionen visar en tydlig separering mellan Suno- och Udio-låtar, och mellan manliga och kvinnliga röster. Dock kan graferna och klustringsresultatet vara missvisande då det förekommer ljudartefakter i datan som orsakats av ett fel i källsepareringssteget. Ett implementationsfel i MFCC-metodens UMAP-projektion gjorde att den initialt fick betydligt sämre resultat än vad den senare korrigerade versionen gav. En vidareutveckling på MFCC-metoden skulle därför vara en intressant fortsättning av studien. Det skulle också vara intressant att undersöka eventuella mönster mellan röstkaraktistiken och texterna som användes för att skapa de AI-genererade låtarna.

## Nyckelord

Generativ AI, Källseparering, Talarigenkänning, Sångröster, Musikdataextrahering



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Background</b>	<b>5</b>
2.1	Source separation . . . . .	5
2.2	Singing voice analysis and recognition . . . . .	6
2.2.1	Mel-Frequency Cepstral Coefficients (MFCCs) . . . . .	6
2.2.2	Singer similarity and classification . . . . .	7
2.3	Clustering . . . . .	8
2.3.1	Kullback–Leibler divergence and Gaussian Mixture Models (GMMs) . . . . .	8
2.3.2	Multi-dimensional scaling . . . . .	9
2.3.3	K-medoids clustering . . . . .	9
2.3.4	Clustering performance . . . . .	10
<b>3</b>	<b>Methodology</b>	<b>11</b>
3.1	Datasets . . . . .	14
3.2	Pre-processing . . . . .	14
3.2.1	Source separation . . . . .	15
3.2.2	Silence removal . . . . .	15
3.3	MFCC Approach . . . . .	16
3.3.1	Correcting errors in multi-dimensional scaler implementations . . . . .	17
3.4	Pre-trained model approach . . . . .	18
3.4.1	HuBERT models approach . . . . .	18
3.4.2	Singer representation model approach . . . . .	19
3.5	Experiments on the Suno and Udio datasets . . . . .	19
3.6	Technologies used . . . . .	20

<b>4</b>	<b>Demonstration and Analysis</b>	<b>21</b>
4.1	Results on the dataset of real singers . . . . .	21
4.2	Udio and Suno . . . . .	22
4.2.1	Clustering performance testing . . . . .	28
<b>5</b>	<b>Discussion</b>	<b>33</b>
5.1	Clustering performance . . . . .	33
5.2	Suno and Udio split . . . . .	34
5.3	Issues with the source separation . . . . .	34
5.4	UMAP . . . . .	35
5.5	Improvements to the MFCC method . . . . .	35
5.6	Ethical considerations regarding Dataset usage . . . . .	36
<b>6</b>	<b>Conclusions and Future work</b>	<b>37</b>
6.1	Conclusions . . . . .	37
6.2	Future Work . . . . .	37
	<b>References</b>	<b>39</b>
<b>A</b>	<b>Additional Figures</b>	<b>45</b>

# List of Figures

1.1	Flowchart illustrating the idea behind the audio pipeline created in this study. . . . .	2
3.1	Flowchart illustrating the audio pipeline used in this study. Only some combinations of paths through the graph were performed for all datasets. . . . .	13
4.1	Scatter plots visualizing the MFCC algorithm clustering of the ‘other’ dataset with the incorrect implementation of KL divergence calculations projected with t-SNE. More separated distinct clusters of colors indicates better separation in the higher dimensional space. . . . .	23
4.2	Scatter plots visualizing the MFCC algorithm clustering of the ‘other’ dataset where a symmetrization of the KL divergence matrix has been applied. More separated distinct clusters of colors indicates better separation in the higher dimensional space. . . . .	24
4.3	Scatter plots visualizing the HuBERT and SUPERB algorithm clustering of the ‘other’ dataset where a symmetrization of the KL divergence matrix has been applied and then scaled using t-SNE. More separated distinct clusters of colors indicates better separation in the higher dimensional space. . . . .	25
4.4	Scatter plots visualizing the ‘BYOL’ singer representation model clustering of the ‘other’ dataset scaled using t-SNE and UMAP. More separated distinct clusters of colors indicates better separation in the higher dimensional space. . . . .	26
4.5	Scatter plots visualizing the ‘BYOL’ singer representation model results of the Suno and Udio datasets dimensionally scaled using UMAP. . . . .	29

4.6	Scatter plots visualizing the ‘BYOL’ singer representation model k-medoids clustering of the Suno dataset scaled using UMAP. . . . .	30
4.7	Scatter plots visualizing the ‘BYOL’ singer representation model k-medoids clustering of the Udio dataset scaled using UMAP. . . . .	31
4.8	Graph showing the impact of changing the number of clusters in the ‘BYOL’ singer representation model K-medoids clustering of the Suno dataset. Every number of groups from 2 up to $2^8$ was tested, then with roughly half exponent increases.	32
4.9	Graph showing the impact of changing the number of clusters in the ‘BYOL’ singer representation model K-medoids clustering of the Udio dataset. Every number of groups from 2 up to $2^4$ was tested, then with roughly half exponent increases.	32
A.1	Scatter plots visualizing projections of the MFCC approach on a subset of 4000 songs from the Suno and Udio datasets. Green is Suno and red is Udio. . . . .	46
A.2	Scatter plots visualizing the real singer dataset and the subset of 1000 Suno and Udio songs, processed using the ‘BYOL’ singer representation model and t-SNE. . . . .	47

# List of Tables

3.1	The number of songs of each artist in the 'other' testing dataset	14
-----	---	----



## List of acronyms and abbreviations

CNN	Convolutional Neural Network
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
KL	Kullback–Leibler
MFCC	Mel-Frequency Cepstral Coefficient
STFT	Short-Time Fourier transform
t-SNE	t-distributed Stochastic Neighbor Embedding
UMAP	Uniform Manifold Approximation and Projection
VRAM	Video Random-Access Memory



# Chapter 1

## Introduction

The field of generative Artificial Intelligence (AI) has seen rapid development and adoption in recent years [1]. One domain of generative AI is AI music, which has seen large strides with research being done by actors such as Google [2] and Meta [3]. It has also seen the establishment of companies such as Udio [4] and Suno [5] providing proprietary commercial platforms.

Both Suno and Udio allow their users to generate songs by inputting text describing the style and genre of music they wish to generate, as well as an optional lyrical prompt for the singing voices in the song. On the platforms both services provide, a significant portion of the generated songs feature vocals. The training data for the models these services use consists largely of copyrighted material [6, 7]. Despite this, not much research has been done on studying the vocals these models generate due to the proprietary nature of the models. Studies have been made attempting to detect songs generated by AI or that used Singing Voice Conversion (SVC) [8, 9, 10] and Suno have published an early version of their text-to-speech/text-to-audio model Bark [11], but little has been done on examining the vocals of the generated songs.

Analyzing the generated songs and population of virtual singers through statistical means could provide indications regarding the model architectures or the underlying datasets used for training, including if certain singers' vocal characteristics have been recreated. It could also visualize biases or specific patterns in either the models, or their usage by the users of the services. In turn, this could be used to better understand and develop similar models, potentially allow for copyright holders to see if their voices has been recreated without proper licensing as well as see how voices relate to the prompts that generated them.

This study attempts to develop a method that allows for estimations of the population of virtual singers in these datasets through statistical analysis of the generated songs. More precisely this study aims to develop an audio pipeline that is able to determine if there are any clusters in the singing voices generated by the commercial models used by Suno and Udio.

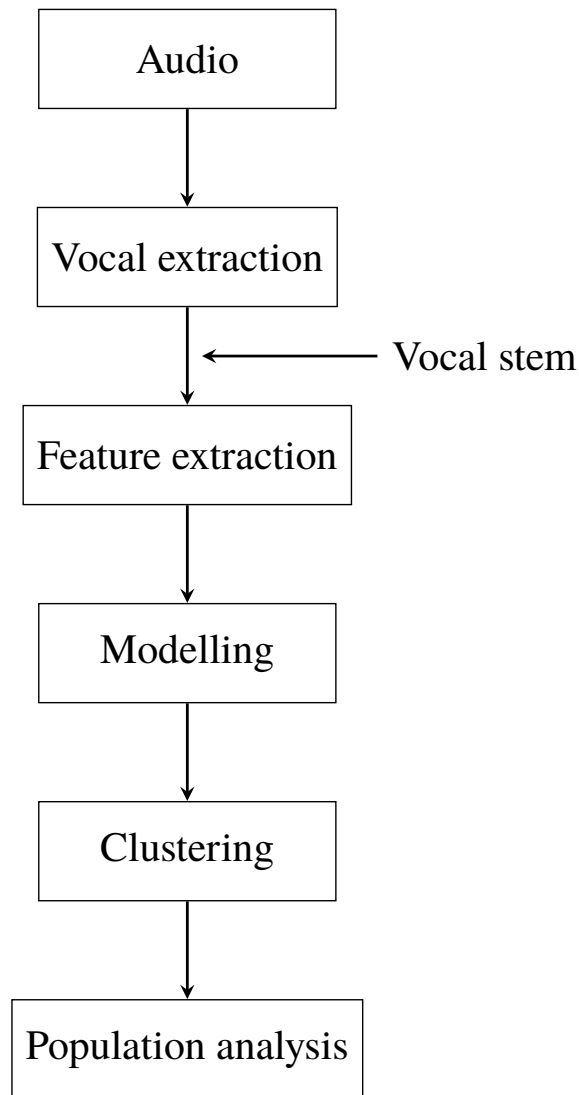


Figure 1.1: Flowchart illustrating the idea behind the audio pipeline created in this study.

The overall structure of the audio pipeline is seen in fig. 1.1, which illustrates the processing steps needed in order to perform the final population analysis. It involves creating the datasets for testing and validation, vocal

extraction to create the raw vocal stems and feature extraction that capture the vocal characteristics present in the audio file. These characteristics must then be modeled to allow for comparisons between the different vocal files. These model representations are then used when attempting to find clusters in the data. Finally population analysis can be performed by analyzing the distribution and variation of the clusters.

We develop the audio pipeline step by step through successive testing on real vocals as well as a subset of AI generated songs, before evaluating the performance on the large datasets of artificially generated songs. Through this development process multiple variations of the audio pipeline are tested. The most significant change is between two approaches, one that uses deep learning for the feature extraction from the preprocessed vocals, and one that does not. For the deep learning feature extraction three models are tested, with two being speech and speaker recognition models and one being a singer representation model. The non-deep-learning approach uses **Mel-Frequency Cepstral Coefficient (MFCC)** features and — similarly to the methods with the two speech recognition models — use **Gaussian Mixture Models (GMMs)** to model the singing voices of each song. For finding the similarities between the **GMMs**, **Kullback–Leibler (KL)** divergence is used. Three different dimensionality reduction techniques are tested as well, **Multidimensional Scaling (MDS)**, **t-distributed Stochastic Neighbor Embedding (t-SNE)** and **Uniform Manifold Approximation and Projection (UMAP)**, which map the dissimilarities from the **KL** divergence to coordinates in two dimensions. The singer representation model uses dimensionality reduction for visualization purposes only, since its output vector is different to the other models.

Our preliminary results from testing on the dataset of real singing voices show the ‘Bootstrap Your Own Latents’ (BYOL) singer representation model as the most promising approach for feature extraction. When we apply that model to the Suno and Udio datasets of artificial generated songs, the results show a distinct separation between the two datasets as well as between male and female voices. When looking at groupings of individual voice features, they are however much less clearly separated compared to the initial testing. The final results are also affected by significant artifacts likely introduced from incorrect sample rate conversion in the source separation process, which could skew the K-medoids clustering performed. Still, the model is able to group genres such as heavy metal voices and choral music separately. This indicates that the approach is functioning, albeit with varying performance.

The theory regarding the vocal pipeline and different audio and vocal processing techniques is presented in chapter 2. Chapter 3 explains the

methodology of how the data is gathered and processed, as well as the reasoning for the choice of method and decisions taken. Chapter 4 provides insight into the performance of the pipeline, by visualizing the results from experiments performed. Chapter 5 discusses potential reasons for why the results from the tests came to be as well as difficulties in implementation. Finally chapter 6 concludes the work and discusses future directions.

# Chapter 2

## Background

This chapter provides background regarding different audio processing techniques that can be used to analyze vocal content in music. These include source separation, **MFCCs** as well as other feature representations of audio along with methods for comparing and clustering arbitrary data.

### 2.1 Source separation

Source separation is the process of extracting sound sources that have been mixed together. For music, that often involves separating singing or speech from accompaniment. Multiple models and approaches have been created for this task such as Spleeter [12] and Demucs [13], with variations in the model structure. Spleeter uses a **Convolutional Neural Network (CNN)** approach, while the latest version of Demucs accomplished significant performance gains through its use of a transformer model. These models allow for either separation into voice and accompaniment, or separating the mix further into voice, bass, drums and other using a four stem model.

While these models often are successful in separating the vocals from the accompaniment, they rarely do so without introducing some amount of artifacts to the stems. These artifacts appear when the model fails to fully separate the sources from the mix, and often occur when parts of an instrumentation share sonic characteristics to vocals or voices. Chants, choirs and heavily processed layered vocals are often included in the vocal stem while not being part of the main singer's melody. If these artifacts make up a large part of a vocal stem, it significantly reduces the quality of the data. This can be problematic when attempting to do automated vocal analysis.

## 2.2 Singing voice analysis and recognition

Singing voices have a lot of variables that change how we perceive them. The vibration frequency of the vocal folds, position of the jaw and the length of the vocal tract are just some of the parameters that can change between singers or between performances that adjusts the sound [14]. In an attempt to model this difficult problem, different approaches have been taken. The source-filter model uses the resonant frequencies found in voices, called formants, in order to recreate the voice. Other approaches use MFCC to recognize the timbre of the voice. These methods are a form of audio feature extraction, which are processes or functions applied to the audio waveform in order to be able to better analyze the track in various ways. They also allow for further processing using other algorithms that enable comparisons between individual voices [15].

### 2.2.1 Mel-Frequency Cepstral Coefficients (MFCCs)

One algorithm often used in audio processing is a **Fast Fourier Transform (FFT)**, which is used to convert time domain audio data to a frequency domain spectrum. Extracting multiple spectra using a short time-axis sliding window allows for a two-dimensional spectrogram to be created. The values of each spectrum act as vertical slices in the y-axis, while the x-axis is the time axis of the sliding window. This allows for better visualization of the signal more aligned with how humans hear audio by showing changes in frequency more clearly [14]. This method is called a **Short-Time Fourier transform (STFT)**.

An **MFCC** is a data reduction method that finds patterns in the frequencies of an audio signal. Calculating it involves first creating a spectrogram. To condense this spectrogram and scale it according to human hearing, a mel-frequency spectrogram is created by using a mel-filterbank. This lowers the vertical resolution of the spectrogram by separating the frequency information into a specified number of bins. An additional **FFT** is then applied to extract the regularities in frequencies across the bins [16]. This can then be further condensed into a number of coefficients, forming the MFCC. While neither the phase nor the frequency information is kept through this process, timbre characteristics are quantified. This method is therefore often used in voice recognition systems [14].

Another approach used in Singing Voice Synthesis (SVS) systems is identifying the frequencies of the formants of a voice in order to use a source-filter model. A formant is a resonance peak that appears when a singer or

speaker makes a vowel sound. They are formed by the shape of the vocal tract and can be adjusted by moving the tongue, lips or other articulators [14]. While the positioning of the formants for specific vowels are similar between different people, they still vary depending on the size and shape of the vocal tract [17]. These formant positions can be used for speech and speaker recognition systems [18], as well as singer recognition tasks [19]. Since the position of the formants affect the timbre, any difference or change in them are reflected in changes to the MFCCs.

## 2.2.2 Singer similarity and classification

The field of singer classification is a subfield of music informatics that aims to be able to group voices using different kinds of features such as MFCCs. While traditionally a statistical approach has been used for singer identification [20], recently research has been made into utilizing deep learning approaches [21].

One established method for singer recognition [20] uses Support Vector Machines (SVM) for singing detection and GMMs to model the singers in the dataset. The approach does not use a deep learning based source separation model to extract the vocals and instead has two GMMs, with one learning the parts of the songs the SVM detected as singing, and the other recognizing the instrumental sections. The singing voice system then preprocesses the vocal sections into shorter segments before finding the vocal tract model coefficients for each segment. This data is then used to train the GMMs in order to identify the singer's unique formant structure. Another approach, suggested by Sten Ternström [22], uses a Long Time Average Spectrum (LTAS) over 30 seconds of speech. That approach looked at the spectral contours of each subject with the results showing that the contour remained consistent to each person in the high frequencies, while still differing among the different subjects. This suggests it could be used for speech or singer identification purposes. However, the authors does not investigate this method further using audio that has been preprocessed in ways where the spectral contours have been affected, making it unknown if this method is suitable for analyzing AI music.

More recently approaches using foundational speech models such as HuBERT [23] for speech and singing related tasks has emerged. SUPERB [24] was developed as a benchmark in which HuBERT among other models were tested on tasks such as phoneme recognition, automatic speech recognition and speaker identification. Furthermore has HuBERT been tested as a foundation

for a system for Singing Voice Conversion (SVC) [25, 26], which is the task of recreating a sung melody through the voice characteristics of another person.

Similar deep learning-based approaches have been used for singer identification as well [21]. One such study focusing on singer related tasks was performed by Sony Computer Science Laboratories Paris [27]. They use an approach that involves feeding a **CNN** with mel-spectrograms of the input audio. Its architecture uses a shallow neural network to provide a one-dimensional vector of 1000 feature representations independent of the length of the original audio file. Using these representations to determine who the singer of certain songs proved quite successful, with 97% identification accuracy on certain datasets, as well as a low Equal Error Rate (EER) on singer similarity tasks.

## 2.3 Clustering

After audio feature extraction using an approach such as a **STFT** or with an **MFCC** there will be a 2-dimensional matrix of features. In the case of using **MFCCs** the output vector shape will depend on the number of mel-coefficients used, as well as window size and hop length of the sliding window. When using a deep-learning model the shape depends on the preprocessing steps used in that model. Typically the vector length depends on the length of the input data, while the width depends on the number of nodes in the final hidden layer of the neural network. This means that these data representations vary in size depending on the length of the audio.

When comparing songs of different length it is often beneficial to use a representation that is not time-dependent. One approach for this involves representing the sequence as a distribution of time-independent values. Such a distribution can be modelled using a **GMM** [28], which is a data representation that uses one or many normal distributions. These normal distributions of features can be compared in order to determine similarities and dissimilarities between them. This in turn allows each distribution to be mapped to a coordinate in a space according to how dissimilar they are to each other.

### 2.3.1 Kullback–Leibler divergence and Gaussian Mixture Models (GMMs)

For comparing the **GMM** distributions of feature vectors, an algorithmic approach is needed. Assuming the data is formed as a single-component

multivariate Gaussian distribution, **KL** divergence can be used [29]. In the case where the distribution is modelled using multiple normal distributions, as in the case with a multi-component **GMM**, this process is much more difficult.

A single-component multivariate **GMM** is synonymous with a multi-dimensional normal distribution. For calculating the **KL** divergence of multi-component **GMMs** there is no precise algorithm. Instead, sampling using the Monte-Carlo method is traditionally used when high accuracy is required. Due to this being very computationally intensive, approximative methods have been developed to accelerate the calculations for large datasets [30]. Given that the **GMM** only uses one Gaussian component, finding the Kullback-Liebr divergence becomes much simpler compared to when using a **GMM** consisting of multiple normal distributions.

While **KL** divergence is non-symmetrical making it unsuitable for use as a distance metric, it can be used to create a symmetrical Jeffrey's divergence matrix by adding the corresponding diagonally mirrored divergence [31]. This is required for use in algorithms such as **UMAP** or **t-SNE** which require symmetrical matrices.

### 2.3.2 Multi-dimensional scaling

In order to convert the **KL** divergence into coordinate representations that allow for clustering, algorithms such as Multidimensional Scaling (MDS), **t-SNE** or **UMAP** [32] can be used. These algorithms can be used for two different methods. One is to reduce the number of feature dimensions so that the same data can be represented using a lower dimensional space, retaining the most significant information and relations between the points. They can also be used to convert data from a pre-calculated distance matrix to coordinates in a specified number of dimensions. When providing a Jeffrey's divergence, it is this method that is used.

### 2.3.3 K-medoids clustering

K-medoids clustering is an algorithm similar to K-means clustering. K-means involves clustering using centroids, which is the position that is in the middle of the data points belonging to a cluster. This position is not necessarily that of one of the data points, but is found using the means of the coordinates of the data points in that cluster. K-medoids on the other hand uses the data points in the dataset as center points for each cluster, which reduces the impact of outliers [33].

Both K-medoids and K-means are clustering algorithms that are fully unsupervised and require the number of clusters to be specified beforehand. In order to determine the optimal number of clusters for an unlabeled dataset, heuristic metrics can be used together with the elbow method, which is the process of continuously increasing the number of clusters tested until a plateau in the metrics is reached.

### **2.3.4 Clustering performance**

The heuristics used to determine the elbow of a curve based on clusters can be the inter-intra distance of the clusters, which is known as the Davies–Bouldin index [34]. Another heuristic is the Dunn index which compares the max internal distance in the cluster to the distances to the other clusters. Both methods measure cluster distinction by comparing the variance of clusters to their separation.

# Chapter 3

## Methodology

This chapter explains the methods we use for creating the datasets necessary for developing and testing the audio pipeline. It also covers the preprocessing steps, specific models used for the deep learning based approaches as well as the techniques for the MFCC approach and specifications of the hardware that was used.

The experiments were mostly performed as a step in developing the audio pipeline by testing on a dataset of real singers. Through a step-by-step process the data was processed and then analyzed visually or aurally to determine if the system was performant enough to develop the next step. Through this fashion multiple configurations of silence removal, MFCC parameters and MDS configurations were tested successively as the pipeline grew in length. As the pipeline reached a more mature state, it was tested using a subset of 500 Suno and Udio songs before the GPU servers were used to run the pipeline on the entire Suno and Udio datasets. To provide more insight into the results throughout the process, clustering methods such as Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) was used, but rarely provided better than visual analysis due to the parameters not being set optimally. Due to difficulty in configuring this method correctly, other clustering methods were explored for the final tests, landing on K-medoids.

The flowchart in figure 3.1 shows the overall structure of the audio pipeline. Beginning with the three datasets of real singers, Suno and Udio, sources are separated using Demucs. Thereafter the audio is processed to remove the silence that was introduced in the stem separation, shortening the files to contain only vocal content. These new files form the basis for all further processing. After removing files shorter than ten seconds, the rest are sent to four feature extraction techniques: MFCC extraction, the two

HuBERT models, and the singer representation model. From there, different actions are taken based on which approach was used, due to the different vector shapes output from the different processes. Regardless of method used, dimensionality reduction is used as a final step to convert the data to coordinates in two dimensions. These coordinates are then used to create plots, which allow for manual analysis.

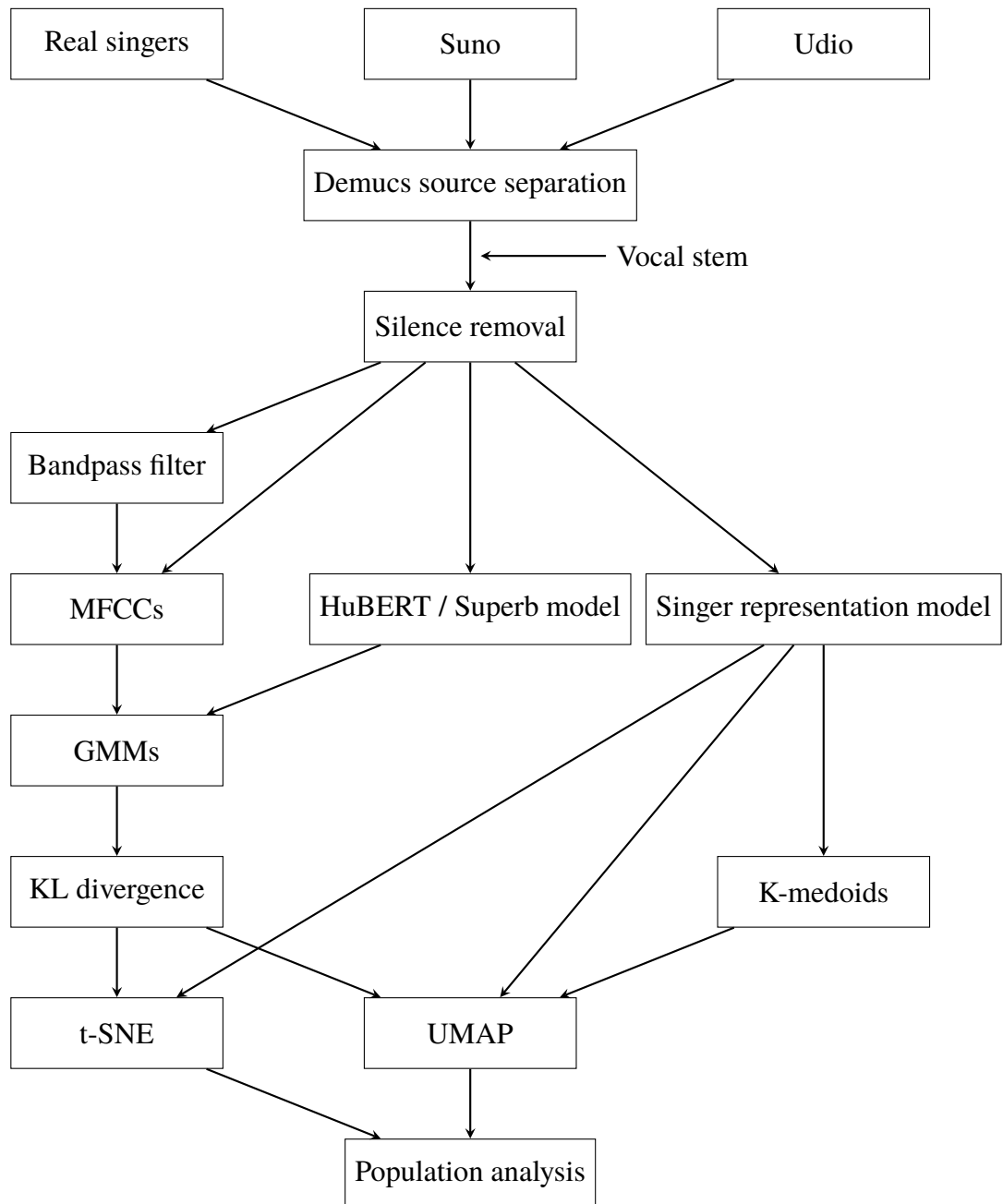


Figure 3.1: Flowchart illustrating the audio pipeline used in this study. Only some combinations of paths through the graph were performed for all datasets.

### 3.1 Datasets

The data used for this study consists of three datasets, a small dataset of 426 real songs by six different artists and two much larger datasets of artificially generated songs by Suno and Udio. We created the Suno and Udio datasets by scraping songs publicly available on their sharing platforms [9], while the real singer dataset was created by downloading the singers' discographies.

In order to cluster artificially created voices, a similar method must be developed to work on real labeled voices as a baseline. For testing and validation purposes we acquired 432 songs from 6 singers with a distribution according to 3.1. After the preprocessing step removed songs with little to no vocal content, 426 songs remained. All files were encoded as 320 kbps MP3 files with 44.1 kHz sampling rate. We labeled this dataset 'other', due to the voices not being created by either Suno or Udio.

Artist	Nr. of songs
Adele	47
Benjamin Ingrosso	74
Dua Lipa	58
Ed Sheeran	82
Justin Bieber	101
Veronica Maggio	70

Table 3.1: The number of songs of each artist in the 'other' testing dataset

After removing songs that featured no or limited singing, the two datasets of AI generated music consisted of 46 049 Udio songs and 51 692 Suno songs. For these two datasets additional metadata such as prompts used, username, tags and lyrics are collected but not processed in this study.

### 3.2 Pre-processing

We perform the pre-processing step on each audio file in the datasets to transform them into files that solely contain singing vocal content. This is done through two steps, source separation to eliminate all instrumentation from the songs, and silence removal to remove the parts of the songs without vocals.

### 3.2.1 Source separation

In order to extract the vocal parts from the songs, the 2-stem version of the hybrid transformer Demucs model [13] is used on all audio files. The files are then re-encoded as MP3 files. The different datasets are stored to separate folders, with the 'other' dataset featuring real singers maintaining the singer labels. The instrumental files created by Demucs are ignored, as processing is only done on the source separated files containing mostly vocals.

### 3.2.2 Silence removal

The source separated vocal files are reimported and mixed down to one-channel mono files using the Pydub Python library. In this process, all files containing a peak amplitude lower than 0.2 are removed from the datasets as these files were mostly silent except for artifacts from the source separation step. Thereafter all the remaining songs are normalized to 0 dB before they have their silence removed using the `strip_silence` Pydub function. The function was used with a silence threshold of -28 dB, silence length parameter of 500 ms and padding of 200 ms. For songs that featured no vocals this resulted in the exported mp3 files being very short – less than 10 seconds – and only containing separation artifacts. To exclude these, songs that were shorter than 10 seconds after the silence removal step were removed from the dataset. The other files were saved as saved to disk as “vocal only” files.

The values of the silence removal algorithm were tested in a couple of configurations by repeatedly exporting a limited selection of the files, listening to them and determining whether or not the majority of the vocal were present throughout the files without separation artifacts or long reverb decay tails being included. The padding value was chosen to avoid introducing clicking noises in the resulting audio files. With a low padding value the amplitude rapidly changed between different parts of the song being edited together, resulting in artifacts. Due to the audio files having different mixing between different artists and songs, finding a silence length and a silence threshold that consistently worked proved quite difficult. It is likely the chosen values are not optimal for all songs. Its impact on the final result has not been evaluated, even though a change in how much reverb tail is included in the testing data likely could affect the results.

When testing methods that filter the audio as part of its preprocessing, it was done in a final step in the silence removal process before saving to disk. A tag in the filename specifying the file has been filtered was added to those

files. The filter applied was a band-pass filter removing frequencies below 100 Hz and above 3000 Hz.

### 3.3 MFCC Approach

In order to extract the **MFCCs** from the source separated files, the Librosa Python library [35] is used. The **MFCCs** approach was configured with 20 **MFCC** coefficients, a 40 ms window length and a hop length of 20 ms. This resulted in a data vector with 20 values per time step, and the number of time steps being dependent on the length of the song. Principal Component Analysis (PCA) plotting of the **MFCC** values of the different songs was added for debugging purposes, visualizing the distribution of the most significant feature values of the vocals. This was solely done to provide insight into the performance of the approach. Since the distributions seemed to vary depending on the singer, this method was further developed.

In order to model the characteristics of the vocals, we added 20 single-component **GMMs** for each song. The means and covariance matrices of the **GMMs** were then used to calculate the **KL** divergences in order to determine the similarities between the different vocals. These calculations resulted in a matrix of non-symmetrical similarity values between the vocals present in the dataset. Using Multi Dimensional Scaling (MDS) to reduce the matrix to two dimensions, the similarities of the songs could be plotted in a graph. These graphs showed some massive outliers. The same method was then tested on a subset of the 'other' dataset with the outliers removed, which made the distribution slightly better but still not showing any distinct clusters. Increasing the number of components used in the **GMMs** was briefly tested as well, but was dropped due to it not substantially improving the results while having the terrible time complexity of the Monte Carlo method. In order to see if there was an issue with the dimensional scaling, **t-SNE** was tested as well as **UMAP** as ways of reducing the dimensionality of the data instead of using MDS. Both algorithms slightly reduced the number of outliers, but still did not distribute the vocals in any proper clusters. This was however implemented using the **KL** divergence matrix as coordinates for the clustering instead of using it as a distance metric. This implementation error was due to the small number of negative values in the matrix making the algorithms crash when using it as a distance metric, but it was not discovered at this point. Instead additional remedying steps were taken in an attempt to improve performance.

As a first attempt to improve performance the number of features were changed to use more or fewer **MFCCs** per time step, as well as including, or

only using, delta values of the MFCCs to provide more context. The spectral contrast feature from librosa was also tested briefly, before implementing a bandpass filter that removed the low-end and high-end of the vocals.

Using Audacity [36], a long time average spectrum was used to compare the frequency distribution between the outliers and the more clustered points. From this, it seemed as if the variation of the mixing of the songs shifted the overall spectral slope resulting in widely different MFCC-distributions, especially if the high-end was boosted. To remove the impact of these mixing differences, the vocals were processed to remove the low and high end using a low-pass and a high-pass filter. Frequencies below 100 Hz and above 3000 Hz were removed, as described in section 3.2.2. While providing slightly improved results in reducing outliers compared to the first method, these changes to the method did not seem to work as consistently as would be required if it were to be used on the large AI-generated dataset. Therefore different approaches using pre-trained deep-learning models were explored instead.

### 3.3.1 Correcting errors in multi-dimensional scaler implementations

When re-running the scripts in order to generate graphs for this thesis, we discovered issues with the implementation of the multi-dimensional scaling in combination with the KL divergence. The algorithms had previously failed to run when using the precomputed distance matrix parameters due to the Jeffrey's divergence matrix having negative values in some slots. At closer inspection these values seemed to only appear on the diagonal and were very close to zero. To fix this, they were replaced with zeroes. With these modifications the original approach performed significantly better on the 'other' dataset.

The corrected MFCC method was then applied to a larger subset of AI songs, 2000 from Udio and 2000 from Suno. During this run only 10 MFCC coefficients were used in order to speed up the KL divergence process, though that might not have had a significant impact on the execution time. The method also differed in that the MFCCs were created from the processed vocals using torchaudio rather than librosa for this experiment. The results from these additional tests were not very exciting, and can be seen in fig. A.1 in the appendix.

## 3.4 Pre-trained model approach

After multiple configurations of the MFCC approach had been tested, it was believed the lack of performance could be due to the MFCC features not being able to take into account the differences in mixing between different songs and artists. To combat this, pretrained speech and speaker recognition models were explored, before finding a singer representation model that finally provided distinct clusters on the ‘other’ dataset.

### 3.4.1 HuBERT models approach

When looking for suitable deep learning based pretrained models, we initially found HuBERT by Facebook Research [37]. It is fine tuned on speech recognition rather than speaker or singer recognition but was tested to determine if the method provided a better distribution for clustering than the MFCC approach. Using this model involved cropping the first 40 seconds of the audio to allow the model and audio to fit in the Video Random-Access Memory (VRAM) of the local testing computer as well as transforming the input data to tensors. It also involved using the last hidden layer to get the embedded features of the input audio as a tensor. This was then fed through the same process as before; creating GMMs and calculating KL divergence, but was changed to use logarithmic values due to the large number of values close to zero in the KL divergence calculations. The multivariate single component GMM was used with covariance type set to ‘full’ and the KL divergence matrix was used to create a Jeffrey’s divergence matrix with the diagonal set to zero. The Jeffrey’s divergence matrix was then used as a distance matrix for UMAP and t-SNE, with their metrics parameters being set as precomputed. The results did not perform well enough to allow proper clustering, but seemed to have fewer problems with outliers compared to the MFCC model approach. Therefore the HuBERT SUPERB SI model was tested, as it is a fine tuned HuBERT model created for speech identification benchmarking [38]. We tested this model because speaker identification have applied similar methods to singer identification in previous research and it required minor changes to the program in order to test. Due to VRAM limitations only the first 30 seconds of each vocal was used for this model, resulting in a max vector size of 1499 by 1024, compared to the 1999 by 1024 shape of the original HuBERT model. As this model provided results very similar to the previously tested HuBERT model, similar approaches using more singer and music tuned models were explored.

### 3.4.2 Singer representation model approach

Since the HuBERT model approach seemed promising but did not perform as well as hoped, we explored other models. Torres et. al. [27] had trained and compared four different singer representation models, with the ‘BYOL’ model having consistently good results on vocal-only content with a sample rate of 44.1 Hz. We therefore picked it as the next model to test.

The input and output shapes of these models differed to the HuBERT models by providing a 1000-length vector independent of the input length. These output values are a feature representation that corresponds to a point in a 1000-dimensional space. In this space the pairwise distances between all songs are calculated in order to find the closest neighbors, as well as to cluster the vocals in similarity. Therefore no GMMs or KL divergence calculations are needed. However, UMAP and t-SNE is still used for dimensionality reduction in order to create the visualizations in two dimensions. As with the SUPERB model, the first 30 second of the extracted and preprocessed vocals were used for testing to allow the model and the data to fit in the VRAM of the local testing computer.

This approach seemed to provide decent results on both the ‘other’ dataset as well as on a subset of the AI generated songs, as can be seen in fig. A.2 in the appendix. Steps were therefore taken to be able to run the entire Suno and Udio datasets on the GPU server, as well as to perform the final tests.

## 3.5 Experiments on the Suno and Udio datasets

The final tests performed on the Suno and Udio datasets were run using the ‘BYOL’ singer representation model. When scaling the data from roughly 1000 songs to around 90000, some issues arose with the processes that were CPU-heavy. The Sci kit Python library was replaced with similar functions that ran on the GPU and most Numpy functions were converted to its PyTorch equivalents. With the increased available VRAM the full songs were able to be used for the ‘BYOL’ model.

For doing clustering, K-medoids was chosen due to there being existing libraries that ran on the GPU, as well as it assigning a center point of a specific song for each cluster. The elbow method was used in order to find the optimal number of clusters, with Dunn index and Davies-Bouldin score being the heuristics measured. When testing the Suno dataset, the number of clusters

were increased by one for each step at first. This resulted in an unnecessarily high degree of accuracy and was decreased slightly for the Udio dataset.

An interactive plot was then created to allow for playback of the processed vocals, as well as traversal of the dataset. This made it possible to find the closest neighbors of each song. We also converted the final audio pipeline to be able to run as one script in order to more easily be able to compare individual songs to the rest of the data. Finally the data was also compared to the 'other' dataset in order to see how the real voices placed in relation to the generated ones.

### 3.6 Technologies used

Python 3 was used as the programming language for the data processing, together with the libraries numpy, sci kit-learn and Pytorch (including torchaudio) for scientific calculations, together with CUDA. Librosa was used for generating the MFCCs for the initial algorithm. Pydub was used for importing, exporting and preprocessing of the mp3 files. For data management of the different datasets and filepaths, Pandas was used. For visualization, matplotlib was used. Hugging Face and Hugging Face Hub was used for the HuBERT and the 'BYOL' singer representation model.

For testing on the entire Suno and Udio datasets a server running 6 GPUs was used. On this server which had 40 CPU cores, a Docker instance was used for the project and was connected to one NVIDIA 3090 GPU. Local testing on the 'other' dataset and a subset of 500 songs from Suno and Udio was done on a NVIDIA 1060 6GB and Ryzen 1600 with 16 GB of RAM. Additionally Python's http server package was used to generate and interact with the large Suno and Udio graphs drawn using plotly.

The RAPIDS cuML library [39] was used for running UMAP on the GPU to significantly speedup the process, and the kmedioids library's [40] FasterPAM implementation was used for clustering. For running the MFCC approach on the GPU servers, torchaudio's MFCC implementation was used instead of librosa.

# Chapter 4

## Demonstration and Analysis

The demonstration chapter is split into two parts. The first presents the intermediary results from the different approaches tested, and the second shows the results when testing the ‘BYOL’ singer representation model on the Suno and Udio datasets.

The ‘BYOL’ singer representation model seemed to be the most promising model on the initial tests, especially with the use of UMAP as dimensional scaler. The correctly implemented version of the MFCC approach also provided somewhat distinct clusters, while the other tests — whether they used bandpass filters or not — showed very limited success. When testing the singer representation model on the Suno and Udio datasets, the model provided much less distinct clusters compared to the previous tests. Similarly, the K-medoids clustering had difficulty accurately determining the distinct number of voices in the datasets. However, manual testing did provide some success, with certain clusters of distinct voice characteristics being discovered. A distinct separation between male and female voices was discovered when analyzing the graphs, as well as certain clusters containing specific styles of vocals such as rap vocals or heavy metal growls.

### 4.1 Results on the dataset of real singers

Initially the results of the MFCC approach as seen in fig. 4.1 seemed quite bad, with a lack of separation between individual data points and no distinct clusters. This indicates that the dimensional scaler is not being able to reduce the dimensions correctly or that the underlying data has very weird structural properties. This result was however produced with the incorrect implementation of the dimensional scalers. After fixing the implementation

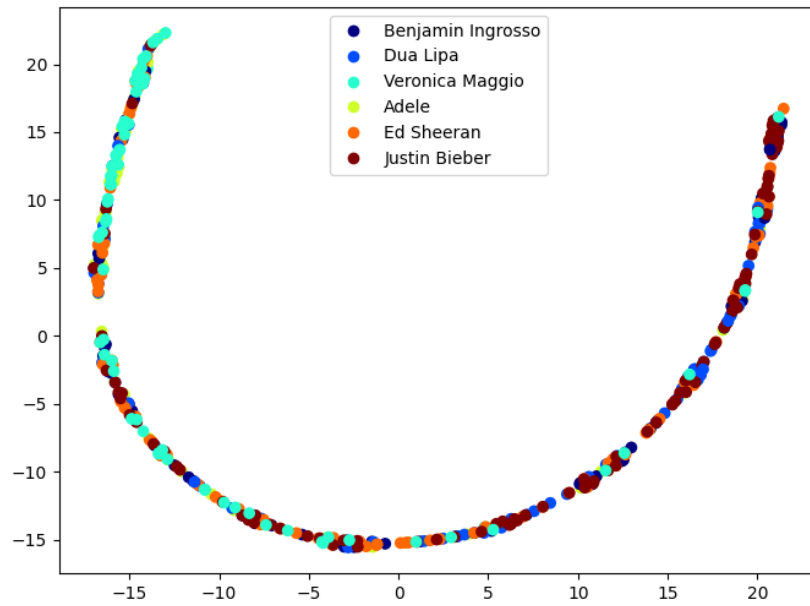
the result instead shows that the non band-pass filtered approach creates much more distinct clusters with clear separation between them compared to both the filtered implementation and the incorrectly configured scalars, as can be seen in figure 4.2.

When this is compared to the HuBERT and SUPERB pretrained models as seen in fig. 4.3, their results are much less clearly separated than the correctly implemented MFCC approach, while still being better than the incorrect implementations that were created during the initial experiments. The fact that the models were created for speech and speaker recognition and not singer identification likely negatively impacts their performance in regards to singer representation modeling. It is however interesting to note that both models seem to have a separate somewhat distinct cluster of ‘Veronica Maggio’ songs.

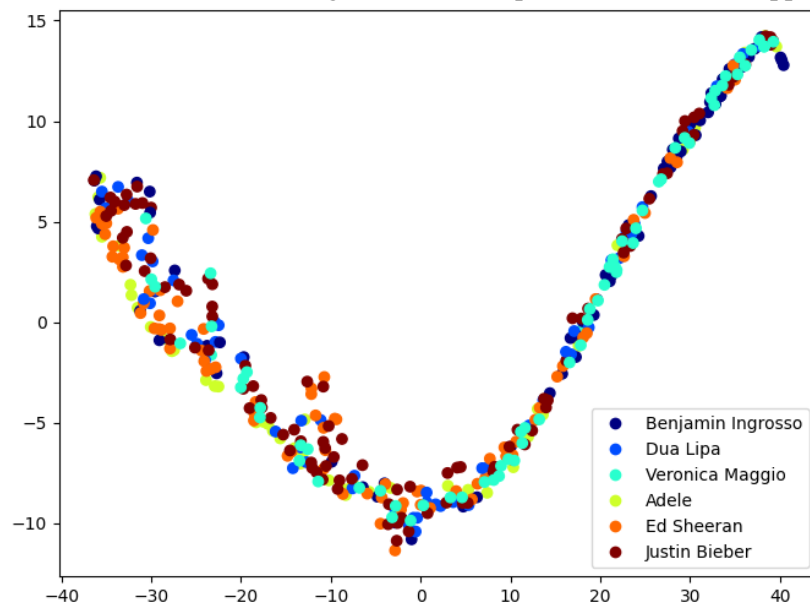
With the ‘BYOL’ singer representation model the results as seen in figure 4.4 are similar to that as seen in the working MFCC implementation in fig. 4.2. Both the t-SNE and UMAP projections show clear clusters, with them being more clearly separated when using UMAP. Between some clusters the separation is less clear, with multiple Justin Bieber songs blending into the surrounding clusters that contains mostly female singers. The same is true for some Dua Lipa songs, landing between the cluster of Ed Sheeran and Justin Bieber. One such Dua Lipa song is the song ‘Sean Paul - No Lie (ft. Dua Lipa)’. The song features mostly Sean Paul’s vocals for the first 40 seconds, which could explain its position in the plot. The incorrectly clustered songs in the cluster of Veronica Maggio is five Justin Bieber songs, all released in 2010 and 2011, as well as one song released by Benjamin Ingrosso that has a female voice as the only vocal content. Benjamin Ingrosso also has a separate distinct sub cluster featuring songs from the album ‘Live at Konserthuset Stockholm (with the Royal Stockholm Philharmonic Orchestra)’ as well as the track ‘IKNOW IKNOW’ which features quite a lot of reverb. In general the plot seems to be roughly divided in two parts, with male singers on one side and female voices on the other.

## 4.2 Udio and Suno

The final processed dataset consists of 51693 Suno songs and 46043 Udio songs, resulting in a total of 97736 points in the final graph, as can be seen in fig. 4.5. The overall distribution is split quite evenly in four quadrants, with minor clusters in the middle and along the edge of the distributions. As can be seen in the zoomed out subfigure, there is a large cluster of outliers significantly above the rest of the distribution. When performing a listening

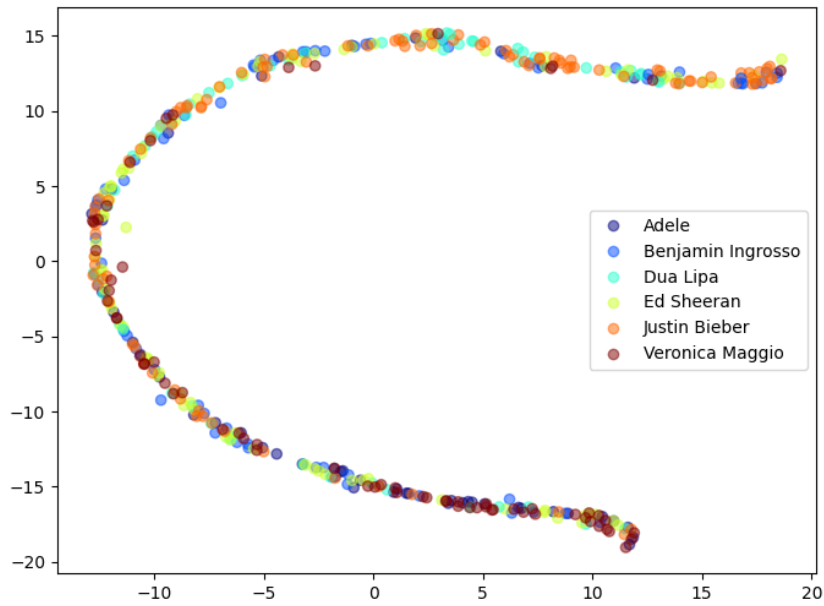


(a) The visualization of songs where a bandpass filter have been applied.

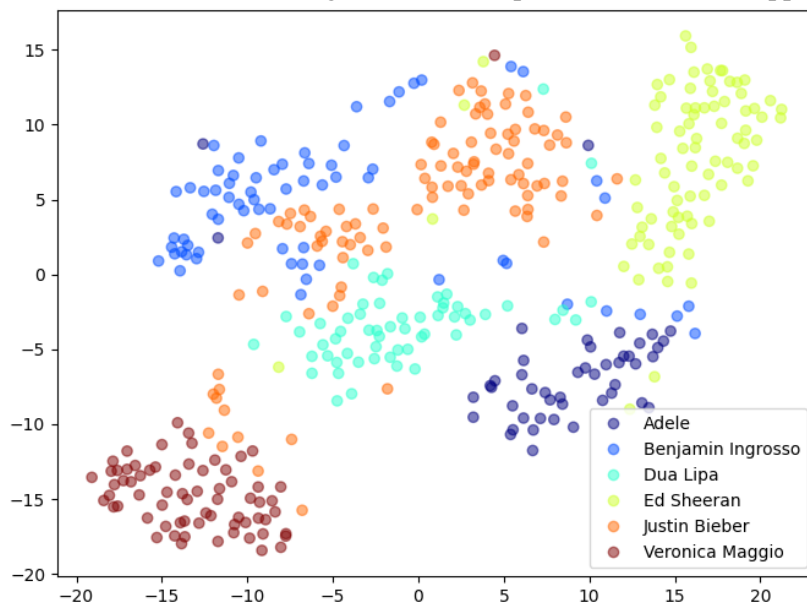


(b) The visualization of songs without a bandpass filter.

Figure 4.1: Scatter plots visualizing the MFCC algorithm clustering of the ‘other’ dataset with the incorrect implementation of KL divergence calculations projected with t-SNE. More separated distinct clusters of colors indicates better separation in the higher dimensional space.

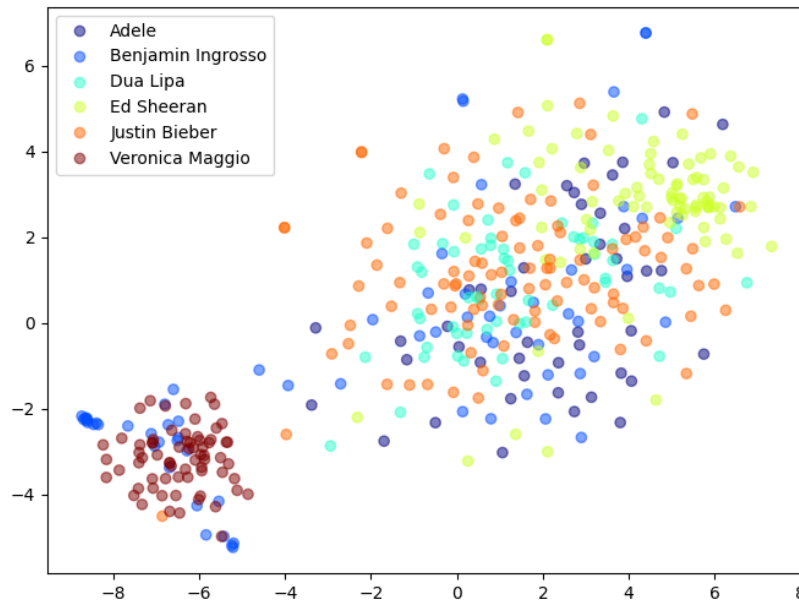


(a) The visualization of songs where a bandpass filter have been applied.

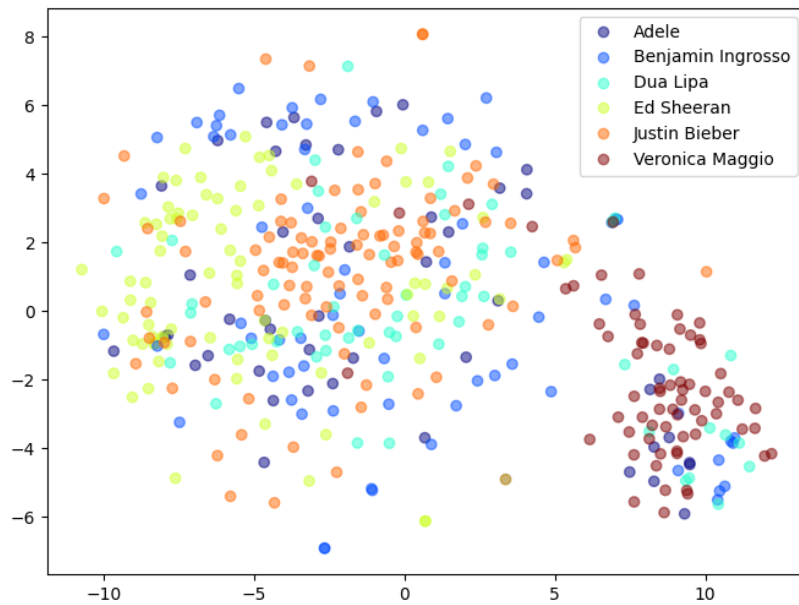


(b) The visualization of songs without a bandpass filter.

Figure 4.2: Scatter plots visualizing the MFCC algorithm clustering of the ‘other’ dataset where a symmetrization of the KL divergence matrix has been applied. More separated distinct clusters of colors indicates better separation in the higher dimensional space.

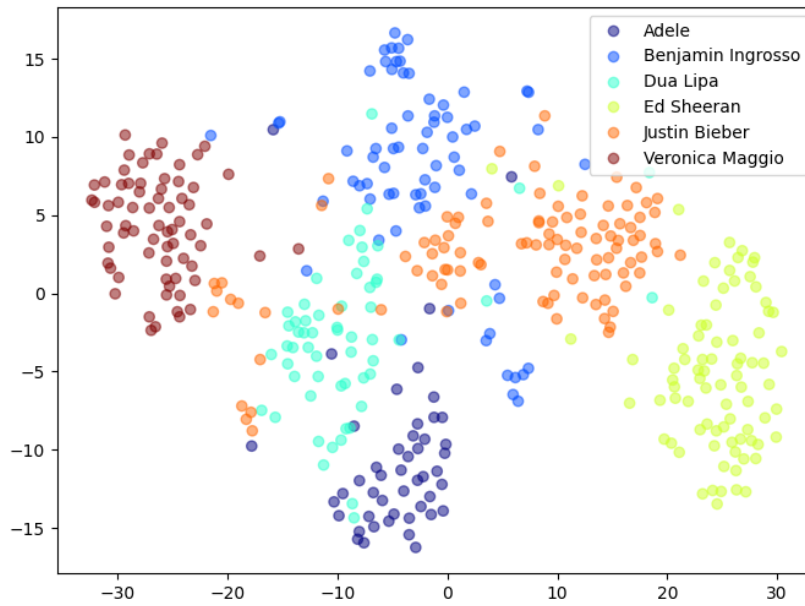


(a) The visualization of songs using the HuBERT model.

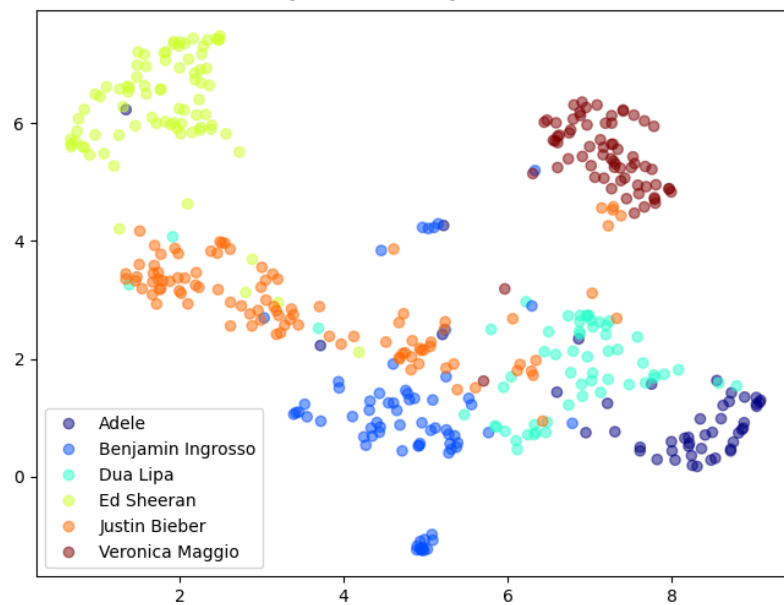


(b) The visualization of songs using the SUPERB model.

Figure 4.3: Scatter plots visualizing the HuBERT and SUPERB algorithm clustering of the ‘other’ dataset where a symmetrization of the KL divergence matrix has been applied and then scaled using t-SNE. More separated distinct clusters of colors indicates better separation in the higher dimensional space.



(a) The visualization of songs where using t-SNE as the dimensional scaler.



(b) The visualization of songs where using UMAP as the dimensional scaler.

Figure 4.4: Scatter plots visualizing the ‘BYOL’ singer representation model clustering of the ‘other’ dataset scaled using t-SNE and UMAP. More separated distinct clusters of colors indicates better separation in the higher dimensional space.

test on this cluster, the songs feature large amounts of audio artifacts such as clicking and popping noises. In the large distribution across the four quadrants there is a distinct separation between songs generated by Suno on the left and Udio songs on the right, as well as a mostly vertical separation between male and female voices. The smaller clusters along the edges feature more unique characteristics. The bottom right of the Udio cluster feature a lot of male rap vocals, while the bottom left instead have a lot of heavy metal characteristics with growl like vocals. The cluster in the very center of the graph of both Udio and Suno songs consist of audio that contain mostly stem separation artifacts and reverb, with very little correctly separated vocal content. Compared to the large distorted cluster far above the rest of the graph, this cluster contains a lot less of the clicking noises and instead contain source separation misclassifications such as trumpet sounds or vaguely voice adjacent noises. To the very right of this cluster at origo, there is a cluster of choral music generated by Udio. In contrast, the clusters in the Suno dataset seem much less distinct, with the voices not being as specifically tied to certain genres. The green cluster of Suno songs above the rest of the graph in the zoomed in plot seem to be female ballads. There is also a cluster of heavy metal voices present in the bottom right of the Suno dataset, adjacent to the Udio metal cluster, but otherwise it seems much less varied in terms of voice characteristics.

When processing the Suno and Udio datasets separately as seen in fig. 4.6 and 4.7, the 4 clusters disappear and the overall spread seemingly increases. The distorted data is still present in its own cluster for both two datasets and can be seen in the zoomed out visualizations.

For the Suno plot in fig. 4.6, the green cluster consists of audio clips that are either distorted or consist mostly of source separation errors. The blue cluster contains mostly female voices while male voices are in the red clusters. The separate red cluster with some blue songs in the top right appear to not have as much high end content as the most other Suno songs have, making them appearing slightly less breathy. Otherwise it is quite difficult to find any distinct patterns among the songs in the UMAP projection.

In the Udio plot in fig. 4.7, the blue clusters contain songs that that have little to no vocal content as well as songs that mostly contain heavy metal growls or distorted data. The green cluster generally consist of male vocals, while the red is mostly female. In the top right there is a mix of blue and green points that form a small cluster that consist of heavy metal songs. In the top middle there is a grouping of green points that mostly contain rap vocals.

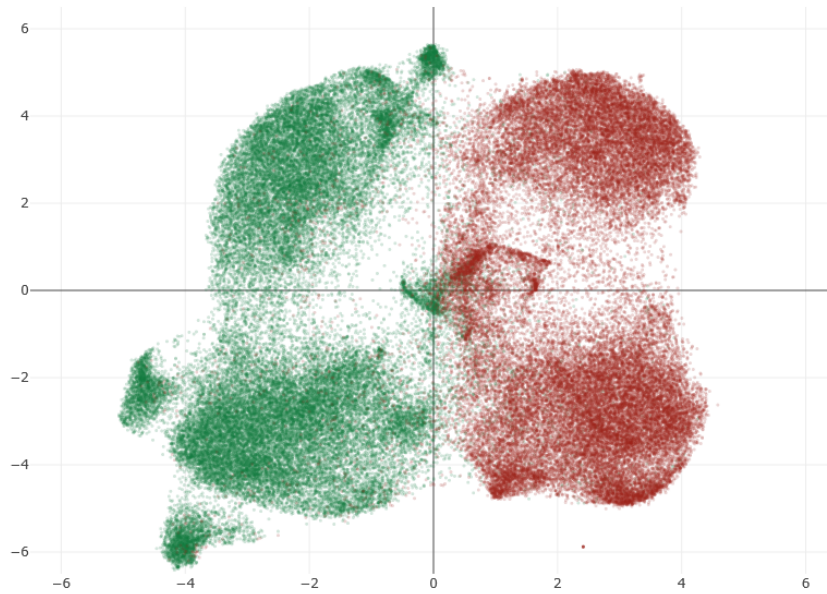
In the bottom of the red cluster there are heavily processed vocals that sound similar to those present in nightcore music.

The **UMAP** visualizations of the datasets appear to not closely align with the euclidean distance of the higher dimensional output space of the ‘BYOL’ model. When traversing the graph by highlighting the euclidean neighbors of the songs, they very rarely appear as the closest points in the **UMAP** projection. This makes the information gained from analyzing the **UMAP** plots quite limited, since the projections do not closely resemble the underlying data.

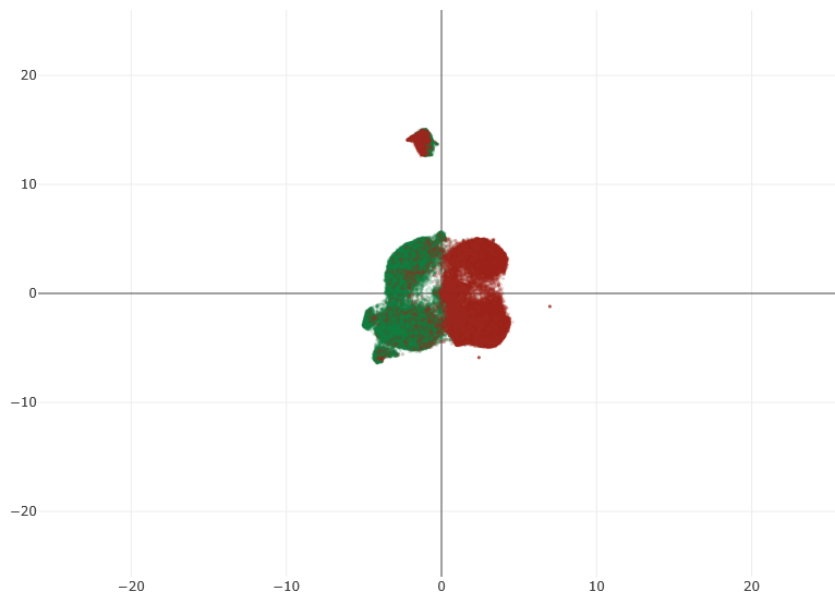
### 4.2.1 Clustering performance testing

The result of the running the K-medoids clustering algorithm on the Suno dataset indicates that the clustering is not great. A higher Dunn index indicates better clustering. As can be seen in 4.8, the Dunn index is consistently below 1, with the best clustering being 3 groups of singers. This is likely not tied to the voices of the individual artificial singers in the dataset, but rather if the voice is male or female, or in the cluster of distorted data. Similarly you can see a decrease in the Davies-Bouldin (DB) index for three groups. Interestingly however, you can see a rapid increase in the DB index for when the number of groups are above 25, going from 2.51 to 1136 in one step. Similar patterns of mediocre performance and rapid increases around 26 clusters are visible in the Udio plots as well, as can be seen in fig. 4.9.

As seen in the zoomed out plots in fig. 4.6 and fig. 4.7, the clusters are far from ideal. For the Suno plots, the green separate cluster to the right is a cluster that contains songs that have been distorted by errors in the audio pipeline. The green cluster below are songs with artifacts from the vocal separation, where they seem to not contain any real lead vocals. The red cluster contains mostly male voices, while the blue cluster is mostly female. For the Udio plots the green corresponds to male voices, the red to female ones and blue to mostly data issues. However a large part of the blue area to the bottom right seems to contain a lot of choral voices, or songs with a lot of reverb effects. There are also two small but distinct separate clusters, one green and one red to the bottom and bottom left of the large clusters. They seem to be iterations of two different songs, where there were enough versions for them to create their own cluster in the UMAP dimensionality reduction.

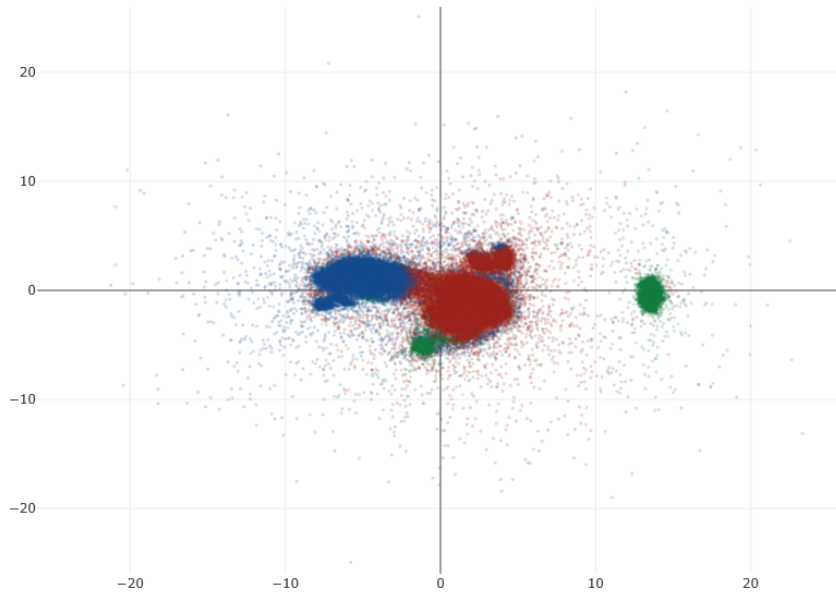


(a) Visualization of AI generated songs. Green is Suno and red is Udio.

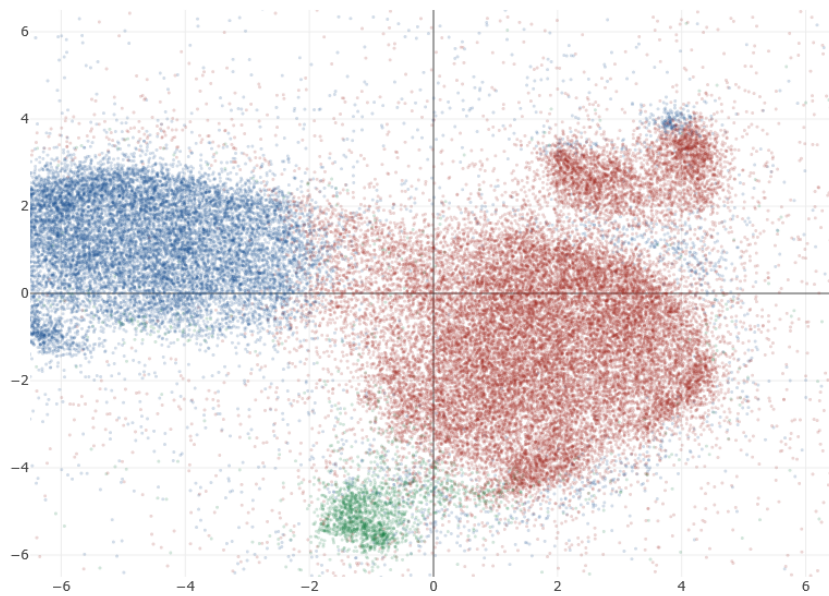


(b) Zoomed out visualization of AI generated songs. Green is Suno and red is Udio.

Figure 4.5: Scatter plots visualizing the ‘BYOL’ singer representation model results of the Suno and Udio datasets dimensionally scaled using UMAP.

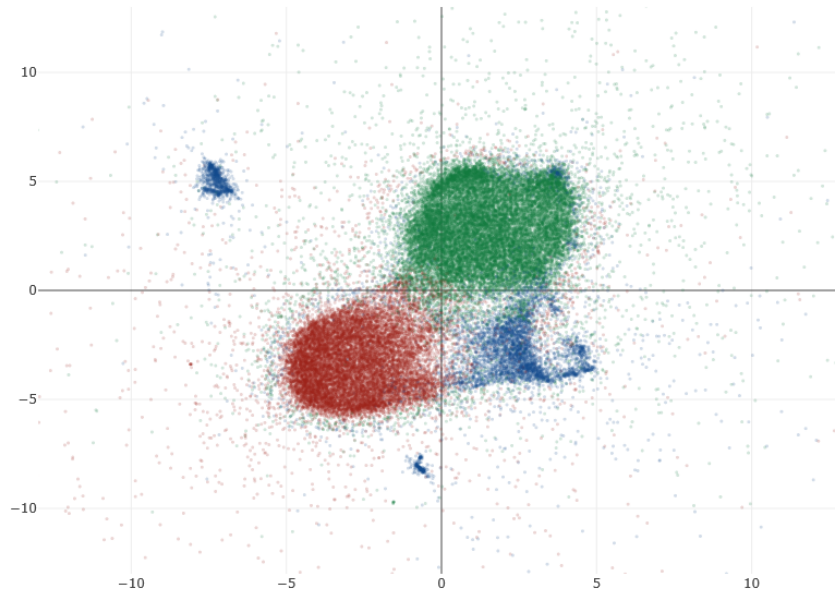


(a) Visualisation of Suno songs using 3 clusters with k-medoids

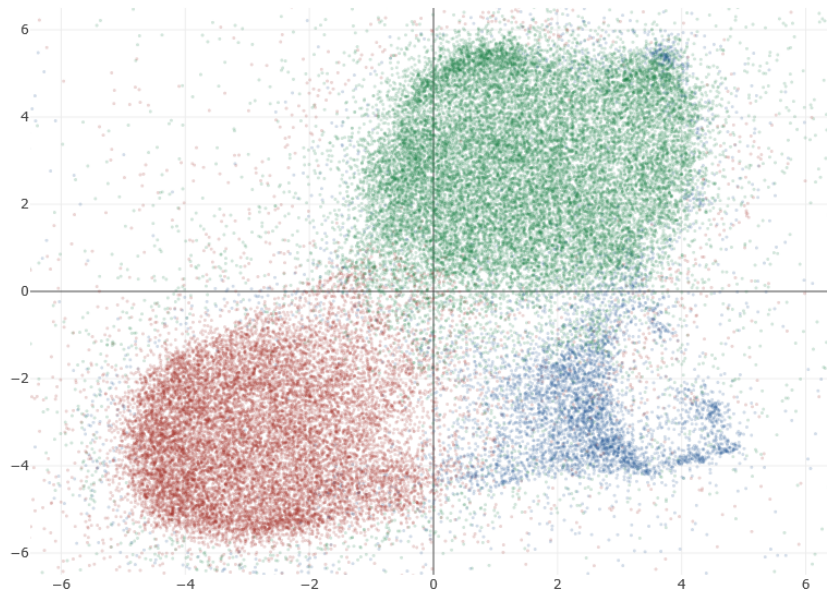


(b) Zoomed in visualisation of Suno songs using 3 clusters with k-medoids

Figure 4.6: Scatter plots visualizing the ‘BYOL’ singer representation model k-medoids clustering of the Suno dataset scaled using UMAP.

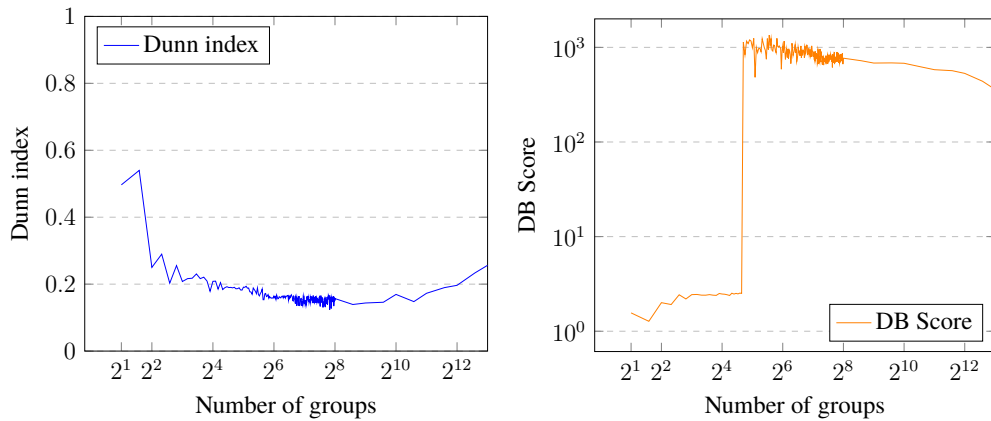


(a) Visualization of Udio songs, using 3 clusters with k-medoids



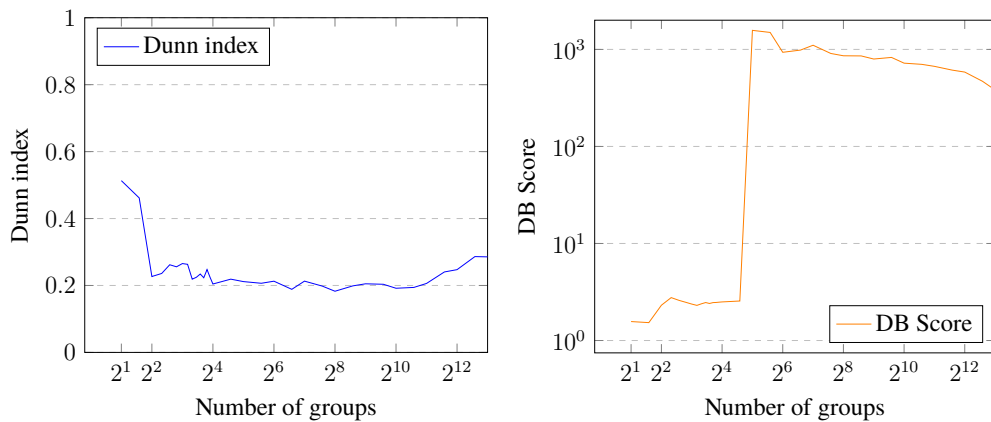
(b) Zoomed in visualization of Udio songs, using 3 clusters with k-medoids

Figure 4.7: Scatter plots visualizing the ‘BYOL’ singer representation model k-medoids clustering of the Udio dataset scaled using UMAP.



(a) Graph showing how the number of groups in the K-medoids clustering affects the Dunn index. Higher values are better. (b) Graph showing how the number of groups in the K-medoids clustering affects the Davies-Bouldin index. Lower values are better.

Figure 4.8: Graph showing the impact of changing the number of clusters in the ‘BYOL’ singer representation model K-medoids clustering of the Suno dataset. Every number of groups from 2 up to  $2^8$  was tested, then with roughly half exponent increases.



(a) Graph showing how the number of groups in the K-medoids clustering affects the Dunn index. Higher values are better. (b) Graph showing how the number of groups in the K-medoids clustering affects the Davies-Bouldin index. Lower values are better.

Figure 4.9: Graph showing the impact of changing the number of clusters in the ‘BYOL’ singer representation model K-medoids clustering of the Udio dataset. Every number of groups from 2 up to  $2^4$  was tested, then with roughly half exponent increases.

# Chapter 5

## Discussion

From the results of the clustering, we found no distinct voices using the current implementation of the method. The Dunn and Davies-Bouldin indices indicate that three groups are optimal for both the Suno and Udio datasets. Through listening tests there is however a much larger population of voices within the datasets. The output of the ‘BYOL’ singer representation model seems to reflect this. When processing a real vocal from the ‘other’ dataset and qualitatively comparing it to the closest songs in the Suno and Udio datasets, the vocals often share some characteristics. As the euclidean distance from the reference song increases, the perceived difference in the vocals also increase. The same is true when comparing AI generated songs to each other. Interestingly though, the closest songs are rarely the closest dots in the **UMAP** projection, instead spanning over a much larger area in the 2D graph. This indicates that while the clustering and the **UMAP** projections might not be great, the ‘BYOL’ singer representation model seems to be working.

### 5.1 Clustering performance

Qualitatively when doing sample tests the model performs quite well, but according to the quantitative testing it is not possible to discern distinct voices through objective measurements. It is possible that other clustering methods — such as HDBSCAN — could provide significantly better quantitative results, since it seems to exist minor clusters of voice types in the data that K-medoids do not cluster distinctly. Another possibility is that the heuristics used do not handle such large datasets well. The rapid increase of the Davies–Bouldin index at 26 groups for both datasets is mysterious and could indicate such. It is also possible that using the euclidean distance as the distance

metric for the K-medoids clustering in the 1000-dimensional space does not relate well to human hearing and therefore does not map accurately to the clusters. If there are certain dimensions that more closely match the human perception or the overall distribution of the voices, they should perhaps be given more weight in the clustering. The problem space could be too complex for such a relatively simple algorithm as K-medoids. Running **t-SNE** or **UMAP** for dimensionality reduction to some lower dimensional space before the clustering could perhaps emphasize some inherent clusters in the data, but using other clustering methods likely has a more significant impact on the final performance.

## 5.2 Suno and Udio split

The split between the Suno and Udio songs in the graph containing both datasets is quite significant. While there are acoustic differences between the two models that become apparent for avid listeners of AI music, the differences are much less clear than that between a male and female vocalist. As such, the **UMAP** projection where both datasets are shown likely does not reflect the overall vocal space of the voices involved. Including a much larger dataset of real voices and comparing the Suno and Udio **UMAP** projections to that would better visually indicate the vocal space and how the two models relate to each other. However, the projection is then likely to simply reflect some other acoustic property that varies throughout the dataset, that might not be reflecting our perception of it.

## 5.3 Issues with the source separation

As can be seen in the zoomed out plots with Suno and Udio data, there is a cluster of points that are songs that have been distorted in the audio pipeline. The artifacts appearing in those files indicate there has been a sample rate conversion error. When looking at the intermediary files of the audio pipeline, it appears the distortions were created in the source separation step. Additionally, songs without these distortions that contained no vocals were still present in the dataset due to other source separation artifacts. Trumpets, flutes, certain synths and parts of drums appear in a lot of the processed vocal stems. Overall these artifacts seem more common in the artificially generated music compared to the dataset of real songs, although that perception could be mostly from the large difference in quantity between the datasets. Because

the distorted files are not removed in the pipeline and are included in the clustering and the **UMAP** step, they still affect how the **UMAP** algorithm distributes the points in the final plots. They also affect how the K-medoids clustering is done. However, they do not affect the steps where the songs are processed independently of each other, which are all the steps leading up to the clustering. The underlying coordinates created by the 'BYOL' representation model should still be correct, for all the samples that have not been distorted. The pairwise distance between the songs should therefore also be accurate.

## 5.4 UMAP

From manual testing it is quite clear that the **UMAP** projection is too much of a simplification for the data. When traversing the data by repeatedly selecting the closest or the second closest song, the jumps in the projected space are often quite large. It is likely that the complexity of a voice encoded into 1000 parameters is too complex to accurately map down to two variables and that a greater projection space would provide better insight into the actual distribution of the data. It could therefore be interesting to change the number of dimensions **UMAP** maps down to, to perhaps 3 or 4, in order to see if the higher dimensional clusters then appear more visible.

## 5.5 Improvements to the MFCC method

It is unfortunate that the correct implementation of the **MFCC** method was discovered quite late in the thesis project timeline, as a more comprehensive study comparing the methods could prove interesting. Using more than 10 **MFCC** coefficients during the Suno and Udio tests could also have improved the performance of the model, or at the very least, made it more comparable to the 20 used in the 'other' tests. The result of these minor tests can be seen in the appendix in figure **A.1**. Additionally, it would be interesting to see if increasing the number of components of the **GMMs** provides better results than what this study achieved. A more advanced algorithm for finding the **KL** divergence would need to be implemented since the time complexity of the Monte Carlo method otherwise makes it infeasible on a large dataset. It is also possible that the fix of using a symmetrized Jeffrey's divergence version of the **KL** divergence with the diagonal set to zero is not the most suitable for this purpose. The changes were made because the deviations from zero seemed like rounding errors and it seemed reasonable that the divergence

metric from a distribution to itself should be zero. Other divergence metrics could perhaps yield better results, but the current implementation is at the very least an improvement to the previous incorrect implementation.

## 5.6 Ethical considerations regarding Dataset usage

The dataset of real singers was acquired without explicit permission for research purposes. Due to the copyright on the songs, the dataset will not be shared in the study and is solely used for testing and verification purposes of the method. No training of the models used have been done on the data, since the deep learning models used in this study are pre-trained local models.

The Suno and Udio datasets were created by scraping data from the Suno and Udio discover platforms. This data has explicitly been published by the users who wrote the prompts that created the content. While express permission for usage in the study or dataset has not been given by the users, choosing to publish the songs and prompts have made them publicly available at the time of the data gathering. Not asking the users for permission to include this content in the study could be ethically problematic, but is likely permitted given EU regulations [9].

The legal aspect regarding the copyright of the generated songs is not determined yet. A court in Czechia ruled that due to the limited creative input and control in the creation of an image created through a prompt to an AI, the user should not be given copyright over the image created [41]. In the EU, the copyright law has been largely harmonized [42]. As such, it could be argued that a similar case in Sweden would result in a similar outcome. If — as Suno and Udio have previously stated — the AI models have been trained on copyrighted works without the artists' explicit permission [6, 7], sharing these AI generated songs that have been indirectly created through usage of their songs could be ethically questionable. Parts of this study could also be used for finding what prompts created similar voices to real artists which in turn could be detrimental to the original artists. However, the same process could be used to determine if the Suno and Udio models have recreated the singers real voices without permission and indicate if the models have been trained on the artists' vocals. This dataset and study could therefore help inform the artists potentially present in the dataset.

## Chapter 6

# Conclusions and Future work

### 6.1 Conclusions

In conclusion, no distinct number of unique voices in the AI-generated music could be found. This is likely partially due to the chosen clustering method, K-medoids, not being able to handle the large number of dimensions of the output space well. The result was also highly impacted by audio artifacts created in the audio pipeline, most likely from a sample rate conversion error in the stem separation process. However, through qualitative sampling tests as well as testing the pipeline using real songs the overall method does provide promising results. The songs closest to each other in the output space often share similarities that become less prevalent when comparing songs further apart. Interestingly the method accurately separates male and female vocals, as well as the songs generated by Udio from the Suno songs. This is likely due to sonic characteristics of the audio that not necessarily relate to how a listener would perceive the singing voice, but instead are due to biases in the models. Another noticeable pattern in the final graph was that effects such as reverb seemed to have a significant impact on the positioning of the audio files, grouping songs with large amounts of reverb together with choral music.

### 6.2 Future Work

There are multiple interesting extensions that could be made to this study. Besides removing the vocal files with audio artifacts, other data processing steps to further emphasize the vocals could impact the performance of the model. This is mainly due to the difference in mixing between different songs. Running dereverberation algorithms could decrease this difference in mixing,

which in turn could decrease its impact on the ‘BYOL’ model’s representation of the audio.

Testing other embedding models such as LAION CLAP [43] could prove interesting, especially since it allows for comparing the artificially generated songs’ audio representations to their textual prompts. Similarly, a study comparing the distance from a generated song mentioning a specific artist in its prompt to songs of the original artist could show how closely the AI models recreate an existing artist’s vocals. Additionally, using a larger dataset of real vocals for comparison in the final plots could better visualize potential limitations of the range of generated vocals, illustrating voice characteristics that the models have not yet generated. Testing other types of voice representation models could also prove as an interesting minor extension to this project.

Expanding the MFCC method tests to include more AI generated songs than 4000 as well as more GMM components would likely be one of the best extensions to this study, since those test runs were quite limited. This would more properly test the initial approach proposed in this study. Due to the time complexity of finding KL divergences, it is not likely for this method to in practice be suitable for voice comparisons of large datasets. If it was done however, it would allow for better comparisons between the more traditional audio analysis method and the deep learning model based approach tested in this study.

Finally, testing other clustering methods besides K-medoids could significantly improve the final results and provide a concrete answer to the question of how many voices there are in the Suno and Udio datasets. From the lack of patterns in the data illustrated in this study, it is however unlikely there are any distinct voices in the two datasets.

# References

- [1] R. He, J. Cao, and T. Tan, “Generative artificial intelligence: a historical perspective,” *National Science Review*, vol. 12, no. 5, p. nwaf050, 02 2025. doi: 10.1093/nsr/nwaf050. [Online]. Available: <https://doi.org/10.1093/nsr/nwaf050> [Page 1.]
- [2] A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzetti, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. Tagliasacchi, M. Sharifi, N. Zeghidour, and C. Frank, “MusicLM: Generating music from text,” 2023. [Online]. Available: <https://arxiv.org/abs/2301.11325> [Page 1.]
- [3] J. Copet, F. Kreuk, I. Gat, T. Remez, D. Kant, G. Synnaeve, Y. Adi, and A. Défossez, “Simple and controllable music generation.” [Online]. Available: <http://arxiv.org/abs/2306.05284> [Page 1.]
- [4] Udio | AI music generator - official website. [Online]. Available: <https://www.udio.com> [Page 1.]
- [5] Suno | AI music. [Online]. Available: <https://suno.com/home> [Page 1.]
- [6] D. Tencer. As suno and udio admit training AI with unlicensed music, record industry says: ‘there’s nothing fair about stealing an artist’s life’s work.’. [Online]. Available: <https://www.musicbusinessworldwide.com/as-suno-and-udio-admit-training-ai-with-unlicensed-music-record-industry-says-theres-nothing-fair-about-stealing-an-artists-lifes-work/> [Pages 1 and 36.]
- [7] J. Koebler . AI music generator suno admits it was trained on ‘essentially all music files on the internet’. [Online]. Available: <https://www.404media.co/ai-music-generator-suno-admits-it-was-trained-on-essentially-all-music-files-on-the-internet/> [Pages 1 and 36.]
- [8] M. A. Rahman, Z. I. A. Hakim, N. H. Sarker, B. Paul, and S. A. Fattah, “SONICS: Synthetic or not - identifying counterfeit songs,” in

- The Thirteenth International Conference on Learning Representations*, 2025. [Page 1.]
- [9] L. Cros Vila, B. Sturm, L. Casini, and D. Dalmazzo, “The AI music arms race : On the detection of AI-generated music,” vol. 8, no. 1, pp. 179–194, publisher: Ubiquity Press, Ltd. [Online]. Available: <https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-365668> [Pages 1, 14, and 36.]
- [10] D. Afchar, G. Meseguer-Brocal, and R. Hennequin, “Detecting music deepfakes is easy but actually hard,” 2024. [Online]. Available: <https://arxiv.org/abs/2405.04181> [Page 1.]
- [11] “suno-ai/bark,” original-date: 2023-04-07T19:34:48Z. [Online]. Available: <https://github.com/suno-ai/bark> [Page 1.]
- [12] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, “Spleeter: a fast and efficient music source separation tool with pre-trained models,” *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, 2020. doi: 10.21105/joss.02154 Deez Research. [Online]. Available: <https://doi.org/10.21105/joss.02154> [Page 5.]
- [13] S. Rouard, F. Massa, and A. Défossez, “Hybrid transformers for music source separation,” 2022. [Online]. Available: <https://arxiv.org/abs/2211.08553> [Pages 5 and 15.]
- [14] T. D. Rossing, Ed., *Springer Handbook of Acoustics*, ser. Springer Handbooks. Springer. ISBN 978-1-4939-0754-0 978-1-4939-0755-7. [Online]. Available: <https://link.springer.com/10.1007/978-1-4939-0755-7> [Pages 6 and 7.]
- [15] M. Müller, *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer International Publishing. ISBN 978-3-319-21944-8 978-3-319-21945-5. [Online]. Available: <https://link.springer.com/10.1007/978-3-319-21945-5> [Page 6.]
- [16] M. Sahidullah and G. Saha, “Design, analysis and experimental evaluation of block based transformation in mfcc computation for speaker recognition,” *Speech Communication*, vol. 54, no. 4, pp. 543–565, 2012. doi: <https://doi.org/10.1016/j.specom.2011.11.004>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167639311001622> [Page 6.]

- [17] Vowel perception - will styler. [Online]. Available: [https://wstyler.ucsd.edu/talks/vowelperception\\_advanced.html#/40/0/1](https://wstyler.ucsd.edu/talks/vowelperception_advanced.html#/40/0/1) [Page 7.]
- [18] D. Yu and L. Deng, *Automatic Speech Recognition: A Deep Learning Approach*, ser. Signals and Communication Technology. Springer. ISBN 978-1-4471-5778-6 978-1-4471-5779-3. [Online]. Available: <https://link.springer.com/10.1007/978-1-4471-5779-3> [Page 7.]
- [19] P. Zwan and B. Kostek, “System for automatic singing voice recognition,” *Journal of the Audio Engineering Society*, vol. 56, no. 9, pp. 710–723, 2008. [Page 7.]
- [20] N. Maddage, Changsheng Xu, and Ye Wang, “Singer identification based on vocal and instrumental models,” in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004*. IEEE. doi: 10.1109/ICPR.2004.1334225. ISBN 978-0-7695-2128-2 pp. 375–378 Vol.2. [Online]. Available: <http://ieeexplore.ieee.org/document/1334225/> [Page 7.]
- [21] Z. Shen, B. Yong, G. Zhang, R. Zhou, and Q. Zhou, “A deep learning method for chinese singer identification,” *Tsinghua Science and Technology*, vol. 24, no. 4, pp. 371–378, 2019. doi: 10.26599/TST.2018.9010121 [Pages 7 and 8.]
- [22] S. Ternström, “Hi-fi voice: observations on the distribution of energy in the singing voice spectrum above 5 kHz,” pp. 3171–3176. [Online]. Available: <https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-52030> [Page 7.]
- [23] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, “Hubert: Self-supervised speech representation learning by masked prediction of hidden units,” 2021. [Online]. Available: <https://arxiv.org/abs/2106.07447> [Page 7.]
- [24] S. wen Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin, T.-H. Huang, W.-C. Tseng, K. tik Lee, D.-R. Liu, Z. Huang, S. Dong, S.-W. Li, S. Watanabe, A. Mohamed, and H. yi Lee, “Superb: Speech processing universal performance benchmark,” 2021. [Online]. Available: <https://arxiv.org/abs/2105.01051> [Page 7.]

- [25] T. Jayashankar, J. Wu, L. Sari, D. Kant, V. Manohar, and Q. He, “Self-supervised representations for singing voice conversion,” 2023. [Online]. Available: <https://arxiv.org/abs/2303.12197> [Page 8.]
- [26] S. Cheripally, “A unified model for voice and accent conversion in speech and singing using self-supervised learning and feature extraction,” 2024. [Online]. Available: <https://arxiv.org/abs/2412.08312> [Page 8.]
- [27] B. Torres, S. Lattner, and G. Richard, “Singer identity representation learning using self-supervised techniques,” in *International Society for Music Information Retrieval Conference (ISMIR 2023)*. [Online]. Available: <https://telecom-paris.hal.science/hal-04186048> [Pages 8 and 19.]
- [28] W. Cai, Q. Li, and X. Guan, “Automatic singer identification based on auditory features,” in *2011 Seventh International Conference on Natural Computation*, vol. 3, 2011. doi: 10.1109/ICNC.2011.6022500 pp. 1624–1628. [Page 8.]
- [29] J.-L. Durrieu, J.-P. Thiran, and F. Kelly, “Lower and upper bounds for approximation of the kullback-leibler divergence between gaussian mixture models,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012. doi: 10.1109/ICASSP.2012.6289001 pp. 4833–4836. [Page 9.]
- [30] J. R. Hershey and P. A. Olsen, “Approximating the Kullback Leibler divergence between gaussian mixture models,” in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, vol. 4, 2007. doi: 10.1109/ICASSP.2007.366913 pp. IV–317–IV–320. [Page 9.]
- [31] F. Nielsen, “On the Jensen–Shannon symmetrization of distances relying on abstract means,” *Entropy*, vol. 21, no. 5, 2019. doi: 10.3390/e21050485. [Online]. Available: <https://www.mdpi.com/1099-4300/21/5/485> [Page 9.]
- [32] K. Kiani and A. Baniasadi, “Speaker recognition system based on identity vector using t-sne visualization and mean-shift algorithm,” in *2019 5th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*, 2019. doi: 10.1109/ICSPIS48872.2019.9066007 pp. 1–4. [Page 9.]

- [33] H.-S. Park and C.-H. Jun, “A simple and fast algorithm for k-medoids clustering,” *Expert Systems with Applications*, vol. 36, no. 2, Part 2, pp. 3336–3341, 2009. doi: <https://doi.org/10.1016/j.eswa.2008.01.039>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S095741740800081X> [Page 9.]
- [34] D. L. Davies and D. W. Bouldin, “A cluster separation measure,” vol. PAMI-1, no. 2, pp. 224–227. doi: 10.1109/TPAMI.1979.4766909. [Online]. Available: <https://ieeexplore.ieee.org/document/4766909> [Page 10.]
- [35] B. McFee, M. McVicar, D. Faronbi, I. Roman, M. Gover, S. Balke, S. Seyfarth, A. Malek, C. Raffel, V. Lostanlen, B. van Niekirk, D. Lee, F. Cwitkowitz, F. Zalkow, O. Nieto, D. Ellis, J. Mason, K. Lee, B. Steers, E. Halvachs, C. Thomé, F. Robert-Stöter, R. Bittner, Z. Wei, A. Weiss, E. Battenberg, K. Choi, R. Yamamoto, C. Carr, A. Metsai, S. Sullivan, P. Friesch, A. Krishnakumar, S. Hidaka, S. Kowalik, F. Keller, D. Mazur, A. Chabot-Leclerc, C. Hawthorne, C. Ramaprasad, M. Keum, J. Gomez, W. Monroe, V. A. Morozov, K. Eliasi, nullmightybofo, P. Biberstein, N. D. Sergin, R. Hennequin, R. Naktinis, beantowel, T. Kim, J. P. Åsen, J. Lim, A. Malins, D. Hereñú, S. van der Struijk, L. Nickel, J. Wu, Z. Wang, T. Gates, M. Vollrath, A. Sarroff, Xiao-Ming, A. Porter, S. Kranzler, VoodooHop, M. D. Gangi, H. Jinoz, C. Guerrero, A. Mazhar, toddrme2178, Z. Baratz, A. Kostin, X. Zhuang, C. T. Lo, P. Campr, E. Semeniuc, M. Biswal, S. Moura, P. Brossier, H. Lee, and W. Pimenta, “librosa/librosa: 0.10.2.post1,” May 2024. [Online]. Available: <https://doi.org/10.5281/zenodo.11192913> [Page 16.]
- [36] GitHub - audacity/audacity: Audio editor. [Online]. Available: <https://github.com/audacity/audacity> [Page 17.]
- [37] “facebook/hubert-large-ls960-ft · Hugging Face,” Jul. 2025. [Online]. Available: <https://huggingface.co/facebook/hubert-large-ls960-ft> [Page 18.]
- [38] S.-w. Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin *et al.*, “Superb: Speech processing universal performance benchmark,” *arXiv preprint arXiv:2105.01051*, 2021. [Page 18.]
- [39] RAPIDS | GPU accelerated data science. [Online]. Available: <https://rapids.ai/> [Page 20.]

- [40] E. Schubert and L. Lenssen, “Fast k-medoids clustering in rust and python,” *Journal of Open Source Software*, vol. 7, no. 75, p. 4183, 2022. doi: 10.21105/joss.04183. [Online]. Available: <https://doi.org/10.21105/joss.04183> [Page 20.]
- [41] Out now – update on recent european case-law on infringement and enforcement of IP rights. [Online]. Available: <https://www.euipo.europa.eu/sv/news/observatory/out-now-update-on-recent-european-case-law-on-infringement-and-enforcement-of-ip-rights-08-2024> [Page 36.]
- [42] A next step in the harmonisation of EU copyright law. [Online]. Available: <https://www.aippi.org/news/the-vitra-kwantum-case-a-next-step-in-the-harmonisation-of-eu-copyright-law/> [Page 36.]
- [43] Y. Wu, K. Chen, T. Zhang, Y. Hui, M. Nezhurina, T. Berg-Kirkpatrick, and S. Dubnov, “Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation,” 2024. [Online]. Available: <https://arxiv.org/abs/2211.06687> [Page 38.]

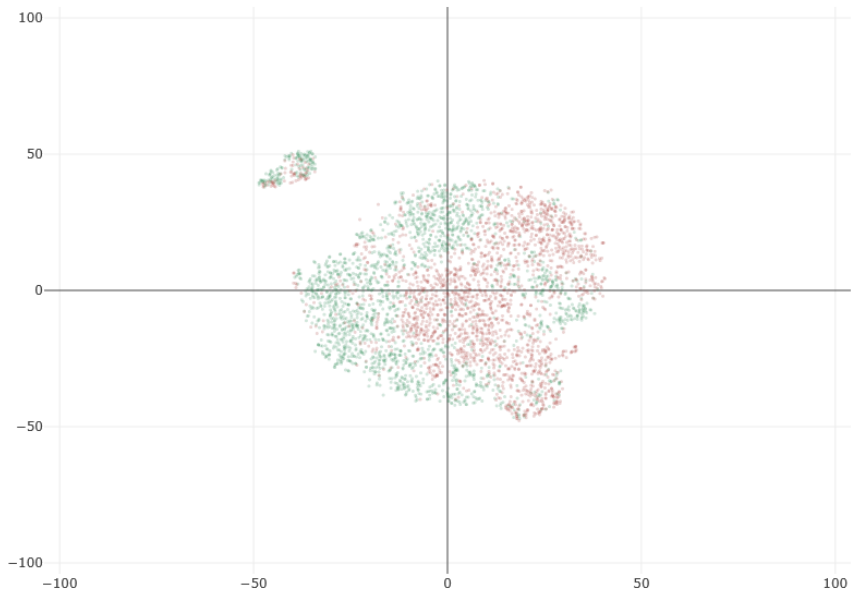
# Appendix A

## Additional Figures

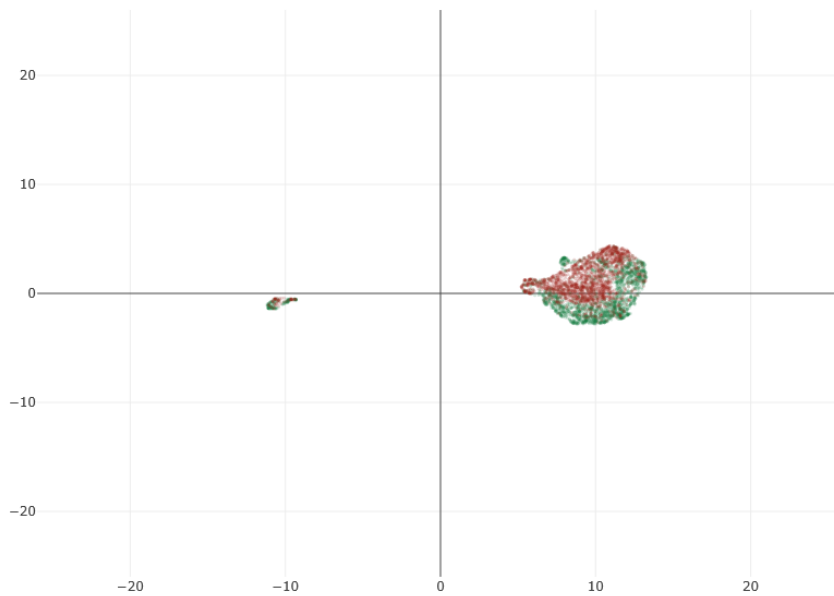
This appendix contains graphs from the MFCC test runs on the Suno and Udio datasets (see fig. A.1), as well as additional intermediary graphs that were used for evaluating the audio pipeline on the subset of 1000 Suno and Udio songs (see fig. A.2).

To interact with some of the visualizations yourself visit: <https://github.com/torenylen/thesis-plots>.

If you want to contact me about this paper I recommend using my personal email: torenylen (zero zero) @ gmail (dot) com.

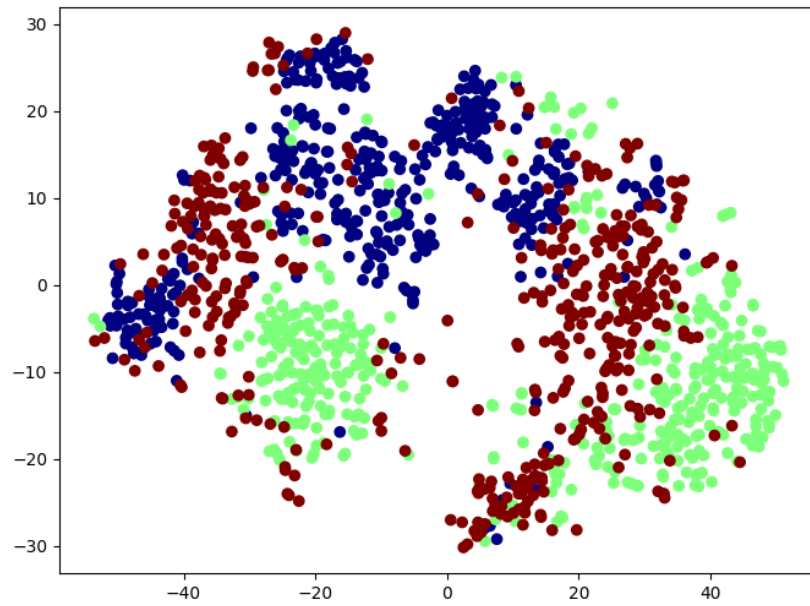


(a) Visualization of Suno and Udio songs, using t-SNE as projection method.

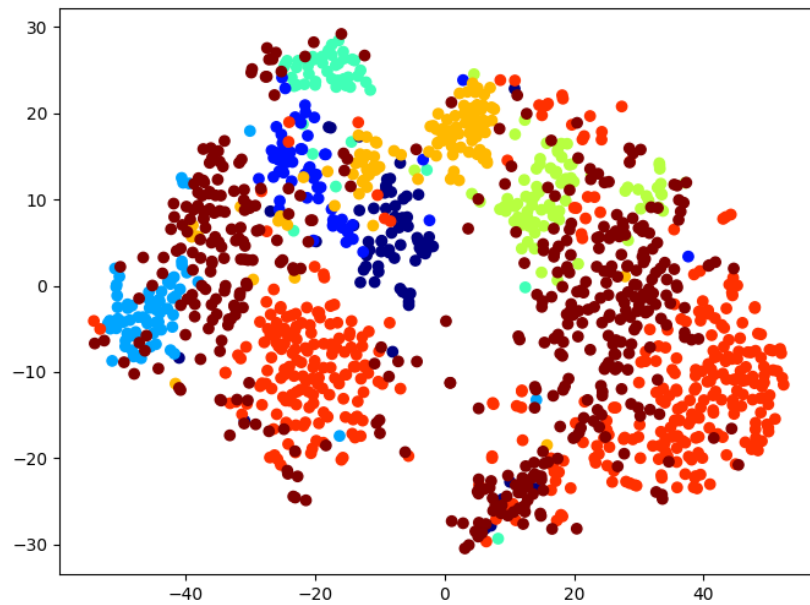


(b) Visualization of Suno and Udio songs, using UMAP as projection method.

Figure A.1: Scatter plots visualizing projections of the MFCC approach on a subset of 4000 songs from the Suno and Udio datasets. Green is Suno and red is Udio.



(a) Visualization of the distribution of the three datasets, with light green being Suno, red being Udio and blue being the real vocals.



(b) Visualization of the distribution of the three datasets as well as the different singers in the 'other' dataset. Dark red is Udio songs while dark orange is Suno. Light blue is Veronica Maggio, clear blue is Dua Lipa, dark blue is Benjamin Ingrosso, cyan is Adele, mustard is Justin Bieber and lime is Ed Sheeran.

Figure A.2: Scatter plots visualizing the real singer dataset and the subset of 1000 Suno and Udio songs, processed using the 'BYOL' singer representation model and t-SNE.



TRITA-EECS-EX-2025:944  
Stockholm, Sweden 2025

[www.kth.se](http://www.kth.se)