QATAR UNIVERSITY

COLLEGE OF ENGINEERING

USING CONTEXT SPECIFIC GENERATIVE ADVERSARIAL NETWORKS FOR AUDIO DATA COMPLETION: MUSICAL INSTRUMENTS CASE STUDY

BY

MARINA FAWZI FARAH MAAYAH

A Thesis Submitted to

the College of Engineering

in Partial Fulfillment of the Requirements for the Degree of

Masters of Science in Computing

June 2023

COMMITTEE PAGE

The members of the Committee approve the Thesis of

Marina Fawzi Farah Maayah defended on 28/05/2023.

Dr. Abdulaziz Al-Ali
Thesis Supervisor

Dr. Abdelhak Belhi
Thesis Co-Supervisor

Dr. Yin Yang
Committee Member

Dr. Junaid Qadir
Committee Member

Dr. Khaled Shaban
Committee Member

Approved:

Khalid Kamal Naji, Dean, College of Engineering

ABSTRACT

Maayah, Marina, F., Masters : June : 2023, Masters of Science in Computing

Title: Using Context Specific Generative Adversarial Networks for Audio Data Completion: Musical Instruments Case Study

Supervisor of Thesis: Dr. Abdulaziz Al-Ali.

Audio quality plays an essential role in several applications ranging from music to voice conversations. Sound information is subject to quality loss caused by reasons such as intermittent network connections, or storage corruption. Recent approaches resorted to using GANs for audio reconstruction due to their successful deployment in visual applications. However, more often than not audio datasets include sounds from different contexts which increase the complexity of the patterns to be learned, leading to sub-optimal quality reconstruction. We propose a novel audio completion pipeline which clusters audio based on similarity and trains a dedicated specialized GAN for each context separately. The proposed technique is compared with the traditional method of training one general GAN in completing 200ms missing segments of 1 second audio samples. Experimental results on a public benchmark dataset show that using specialized GANs led to a clear improvement in the completion quality while reducing training convergence times.

DEDICATION

To my mother soul, for her endless Love.

ACKNOWLEDGMENTS

First and foremost, praises and thanks to Allah the Almighty, for his unending grace and mercy throughout the writing of this thesis.

My most profound appreciation goes to Doctor Abdulaziz Al-Ali and Doctor Abdelhak Belhi, my advisors and mentors, for their time, effort, and understanding in helping me succeed in my studies. Their vast wisdom and wealth of experience have inspired me throughout my studies. To conclude, I'd like to thank my father, my brothers and sisters for their faith in me, my husband, and my children It would have been impossible to finish my studies without their unwavering support over the past few years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Adam	Adaptative Moment Estimation
CEGAN	Context Encoder Generative Adversarial Network
CNNs	Convolutional Neural Networks
cGAN	Conditional Generative Adversarial Network
DNN	Deep Neural Network
DCGANs	Deep Convolutional Generative Adversarial Networks
FFT	Fast fourier transform
GANs	Generative Adversarial Networks
MAE	Mean Absolute Error
MSE	Mean Squared Error
ODG	Objective Difference Grading
OMP	Orthogonal Matching Pursuit
ReLU	Rectified Linear Unit
TTS	Text-to-Speech
TF	Time Frequency
WGAN	Wasserstein Generative Adversarial Networks

CHAPTER 1: INTRODUCTION

Acoustic information is susceptible to defects for a myriad of reasons, including but not limited to loss of data during transmission, data corruption while being stored, or introduction of noise during recording. While several research groups sought preventative measures for each of these causes, little effort was dedicated towards the reconstruction of the missing and corrupted sounds after-the-fact. Audio reconstruction has the advantage of targeting several of these causes in one solution; to restore damaged or corrupted audio signals regardless of the cause.

Audio reconstruction works by reconstructing a damaged signal, so it is as close to the original signal as possible. This process is done by leveraging the surrounding context around the damaged part to rebuild the signal. If successful, such a solution has a wide range of applications such as reconstructing music or noisy voice conversations, restoring damaged audio from old recordings, and potentially creating new music variations such as introducing virtual instruments and sound effects. Audio reconstruction may also be used to improve speech recognition accuracy when considered as a pre-processing step.

Many different approaches have been studied to accomplish data reconstruction in general. Earlier methods resorted to traditional approaches such as applying the Orthogonal Matching Pursuit algorithm using either a discrete cosine or Gabor dictionary [1] or by using statistical measures from prior records along with data inpainting and smoothing techniques [2]. Machine learning techniques have recently garnered research interest for several tasks such as classification, regression and more recently reconstruction of data. In particular, GANs have become popular for data generation purposes. GANs are a type of deep learning models that can learn the distribution of a

given dataset in order to produce new similar samples. They have been demonstrated to be particularly effective when it comes to the production of visual material [3]. In order for GANs to produce good-quality samples, the presented data patterns need to be consistent. GANs do not work properly when presented with visual or audio data from a variety of contexts [4]. For instance, sounds that represent music are different from those in human conversations. Moreover, produced sounds can greatly differ from one musical instrument to another. Similarly, images can have different visual contexts [5]. Despite being an important factor for successful generation, current audio reconstruction efforts do not target this problem leading to generated outputs that are of sub-optimal quality.

Most studies focused on audio reconstruction using GANs or traditional models without considering the complexity of the data and how it might affect audio generation quality. To this end, we propose a new audio reconstruction pipeline that clusters audio segments based on similarity prior to GAN training, and then proceeds with training a group of specialized GANs; one for each context respectively. We found that using a *divide-and-conquer* strategy to homogenize data led to better reconstruction quality and faster convergence times.

1.1. Problem Statement

This thesis targets the reconstruction problem of damaged audio clips from different audio contexts. We assume that a damaged audio clip, which has a small portion (200 milliseconds) missing, is presented to the system. The goal is to reconstruct the missing part such that it is as close to the original, which could be done by leveraging the surrounding context around the missing part to generate the damaged signal.

One of the popular approaches to target image and audio reconstruction is the use of GANs. However, despite showing promising results, training a GAN on a dataset with a different context can be challenging and time-consuming, and the results may not always be optimal. This thesis aims to mitigate the context variance problem by proposing a divide-and-conquer solution which aims to reduce the complexity of the pattern to be learned to produce higher quality results.

1.2. Research Questions

In this thesis, we tackle the problem of reconstructing audio of various contexts by targeting the following research questions:

- RQ1: Which audio representations are most suitable for effective clustering?
- RQ2: Which clustering techniques are sufficient to generate good-quality audio clusters?
- RQ3: How do the completion results of specialized GANs (one per cluster) compare to a general GAN trained on the entire dataset?
- RQ4: How do specialized GANs perform when the cluster memberships of unseen damaged sound clips are unknown? And can the trained models generalize well even when tested on a different musical instrument dataset?
- RQ5: How does the training computation cost of specialized GANs compare to a general GAN?

1.3. Contribution

This thesis contributes to the field of audio processing in three primary ways. Firstly, a comprehensive framework is presented, aiming to reconstruct audio that has been lost or damaged. This framework tackles the challenge of restoring audio quality and surpasses the traditional methods of training GANs found in the existing literature. By integrating a variety of techniques and approaches, this framework offers an efficient solution for audio reconstruction, yielding superior results.

Secondly, this thesis analyzes the effect of using different data representations and clustering techniques combinations for the purpose of identifying audio contexts. Building upon this analysis, an innovative and effective method for audio data clustering is proposed. This method considers the feature representation methods used to represent audio and examines how they can impact the quality of the resulting clusters. By considering the influence of feature representation on cluster quality, this contribution enhances the accuracy and effectiveness of audio data clustering. Additionally, guidelines are provided to assist in the selection of the most suitable clustering algorithm and data representation for the audio reconstruction problem.

Third, the thesis evaluates the effectiveness of a specialized Generative Adversarial Network (GAN) through specific qualitative and quantitative evaluation methods. This evaluation provides valuable insights into the performance and potential applications of the proposed specialized GAN in audio generation and reconstruction tasks.

Together, these contributions advance the field of audio processing by suggesting a comprehensive framework for audio reconstruction, introducing an effective method for audio data clustering that considers the effect of feature representation, and evaluating the performance of a specialized GAN. The above contributions pave the path

towards developing more accurate and efficient audio reconstruction, clustering, and audio generation techniques, benefiting various domains and applications involving audio data.

1.4. Document Overview

This thesis will be divided into the following chapters:

- Chapter 2: provides an overview of the fundamental concepts along with the background information that the proposed solution is based on.
- Chapter 3: contains a literature review of the current state of the art in data reconstruction (visual and sound), with a focus on the strengths and weaknesses of each approach.
- Chapter 4: illustrates thoroughly how our framework is used, as well as its approaches, methods, algorithm, and implementation.
- Chapter 5: illustrates the physical setup we used in addition to our dataset and explains more about the evaluation methods we utilized in this research.
- Chapter 6: presents all results achieved by the experiments are presented and discussed.
- Chapter 7: draws the conclusion and future possible improvements.

CHAPTER 2: BACKGROUND

This chapter provides an overview of the fundamental concepts used in the development of the proposed solution.

2.1. Sound

Sounds in life are different and can be differentiated because they vary in their characteristics, such as Frequency, Amplitude, Timbre, Duration, Phase, and Directionality; for example, a high-pitched, loud, short-duration sound with a sharp timbre. It may be distinguishable from a low-pitched, soft, long-duration sound with a mellow timbre. Since in this study, we applied our experiments over an instruments dataset, let us focus more on the instruments' sounds.

The sound of an instrument's timbre is what makes it sound different from other instruments. Vibrations are the primary determinants of an instrument's timbre. The essential vibration of a string is the amplitude of its vibration that determines its pitch. The harmonics or overtones are additional vibrations that intersect with the basic vibration. Because fundamental and harmonic vibration patterns are added on top of each other, each instrument has a very unique vibration pattern. People hear different timbres, or sound patterns, from these vibration patterns, even though the pitch is the same. Harmonic numbers and power vary wildly between instruments. Even instruments in the same category (e.g two guitars played by different musicians) can have different timbres due to these different harmonics. Some instruments, such as a guitar or piano, produce relatively simple, harmonic sounds with well-defined frequencies. These instruments may be easier to reconstruct accurately because their sound characteristics can be modeled more efficiently using techniques like Fourier

analysis. Other instruments, such as the saxophone or trumpet, produce more complex, non-harmonic sounds with multiple harmonics and overtones. These instruments may be harder to recreate accurately because their sounds are harder to model with simple math. Furthermore, how an instrument is played can impact the reconstruction of audio signals. Different ways to play a stringed instrument, like plucking or bowing, can make the sound have different qualities that must be taken into account during the reconstruction process. Overall, the type of instrument and how it is played can have a big effect on how well audio signals can be reconstructed.

2.2. Audio representation and analysis

So far, we have many approaches used to analyze sound on a linear scale and nonlinear scale [6]. In this section we will explain the different approaches.

2.2.1. Sound waveform

An audio waveform is a time-domain representation of sound, where the amplitude of the wave at each point in time represents the intensity of the sound at that time. This representation is useful for visualizing the waveform and for some basic processing tasks, such as filtering [7]. Fig. 2.1 shows some samples of audio waveform from the data set we used in this thesis.

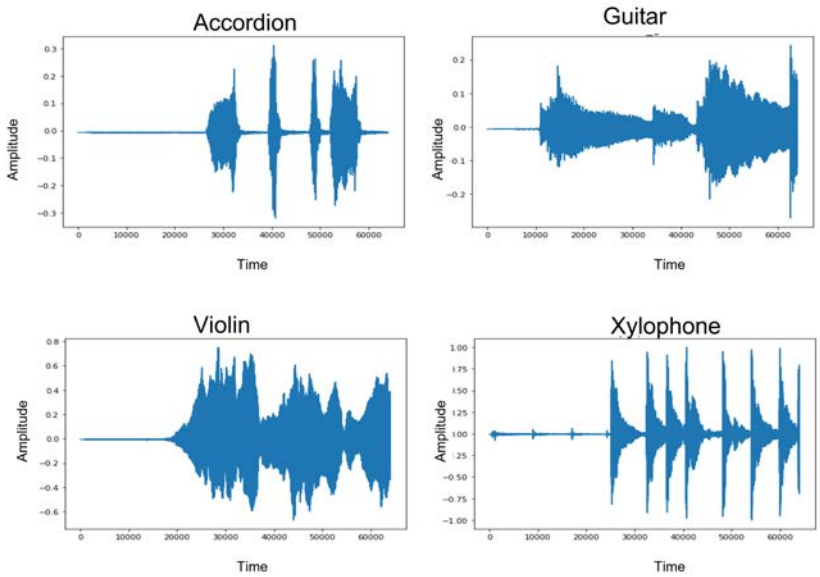


Figure 2.1. Audio waveform samples of different instruments.

2.2.2. Mel spectrogram

A spectrogram is a frequency-domain representation of sound that shows how the frequency content of the signal changes over time. It is a two-dimensional plot that displays the frequency spectrum of the signal at different time intervals. Spectrograms are useful for analyzing the frequency content of a signal and for identifying features such as harmonics, formants, and other spectral characteristics, Fig. 2.2 shows some spectrogram samples.

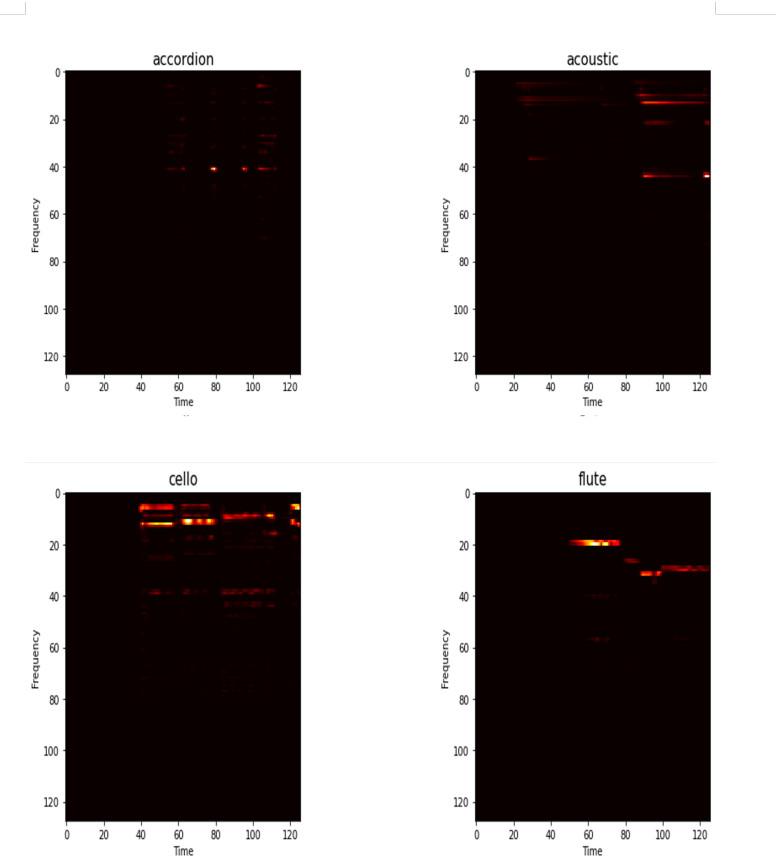


Figure 2.2. Spectrograms samples of different instruments.

A Mel spectrogram is a variant of a spectrogram that uses the Mel scale, which is a perceptually-based scale of frequency that more closely matches human auditory perception. The Mel scale is logarithmic, which means that the difference between two adjacent points on the scale is proportional to the ratio of their frequencies. Mel spectrograms are commonly used in speech and music analysis, as they capture important features of the sound signal that are relevant to human perception. Spectrograms are useful for general analysis of the frequency content of a signal, while Mel spectrograms are particularly useful for speech and music analysis where human perception is relevant. In addition Mel spectrograms are smoother than waveform samples and easier to train with a squared error loss since they are phase invariant throughout each frame [8]. In the GAN training experiments carried out through this thesis, the Mel spectrogram repre-

sentation was used where we transform an audio dataset from waveform representation to Mel spectrogram representation. This representation was used as input to train our GAN models. Fig. 2.3 shows some samples of Mel spectrogram for audio clips from the data set we used in this thesis.

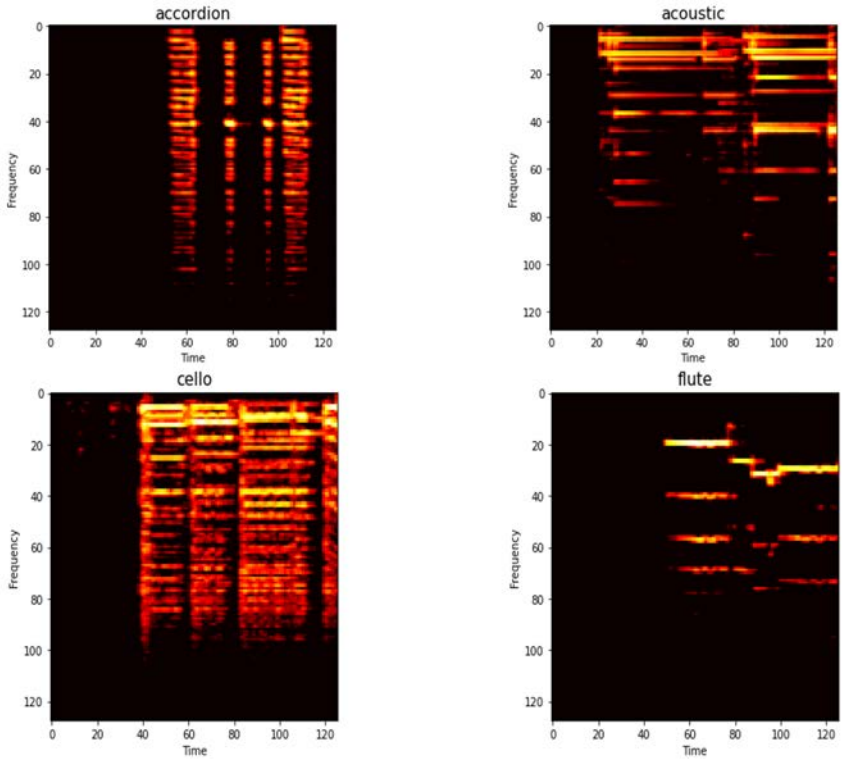


Figure 2.3. Mel spectrograms samples of different instruments.

2.3. Deep Neural Networks

Deep Neural Network (DNN) is a type of Artificial Neural Network (ANN) that has multiple layers between the input and output layers. The deep layers enable DNNs to learn and extract more complex features from the input data than traditional shallow neural networks, which have only one or two hidden layers as shown in Fig. 2.4. DNNs have revolutionized the field of deep learning and have contributed to significant

advances in various areas of technology and science.

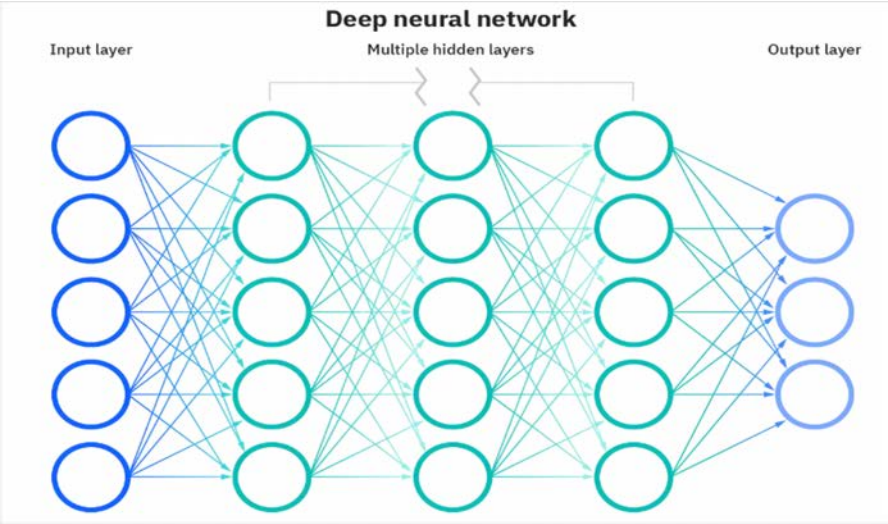


Figure 2.4. Deep Neural Network (DNN) architecture [9].

2.4. Encoder-Decoder

A Deep Neural Network (DNN) encoder-decoder is a type of neural network architecture that is commonly used for tasks such as language translation, image captioning, and speech recognition. It consists of two main parts: an encoder network and a decoder network [10]. The encoder network is responsible for encoding the input data into a fixed-length representation, which is often referred to as a "latent code" or "embedding". The encoder network typically consists of several layers of neurons that transform the input data into a high-dimensional vector epresentation. This vector representation captures the most salient features of the input data and can be used to generate the corresponding output data [10]. The decoder network takes the latent code generated by the encoder network and uses it to generate the output data. The decoder network typically consists of several layers of neurons that transform the latent code into the desired output format. For example, in an image captioning task, the decoder network might transform the latent code into a sequence of words that describe the

contents of the image. During training, the encoder-decoder network is optimized to minimize the difference between the generated output and the ground truth output. This is typically done using a loss function that measures the distance between the generated output and the ground truth output Fig. 2.5.

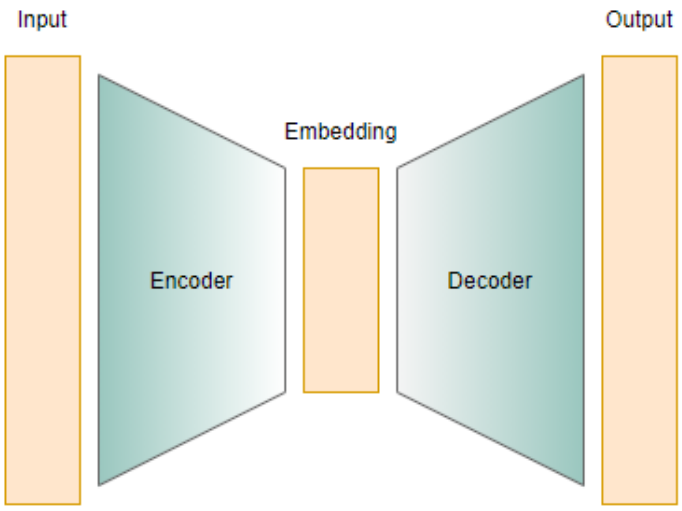


Figure 2.5. A standard autoencoder's internal structure.

2.5. Generative Adversarial Networks

A generative adversarial network is a machine learning model that is used to generate synthetic data. It consists of two components: a generator (G) network and a discriminator (D) network. The generator is trained to generate synthetic data that is similar to the training data, while the discriminator is trained to distinguish between real and synthetic data. The two networks are trained together, with the generator network trying to generate data that is indistinguishable from the real data, and the discriminator network trying to correctly classify the data as real or synthetic [11], [12]. This competition between the two networks drives the generator network to improve its ability to generate realistic data as shown in Fig. 2.6, resulting in synthetic data that is

highly realistic and has a high level of detail. The concept of a min-max loss function is used by all GANs to train the generator and the discriminator at the same time [11]. This means that the generator and the discriminator explicitly learn by maximizing the discriminator loss function L_D and minimizing the generator loss function L_G [13], as per the equation in 2.1.

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[1 - \log D(G(z))] \tag{2.1}$$

Where x $\sim p_{data}$ is the real data and z $\sim p_z$ is the synthetic data that is generated using the generator network. GANs have been used for a variety of tasks, including image generation and audio synthesis. There are many different types of GANs that can be used for audio reconstruction such as Context Encoder GAN (CEGAN) which is the model we based our research on as detailed in section 4.3.

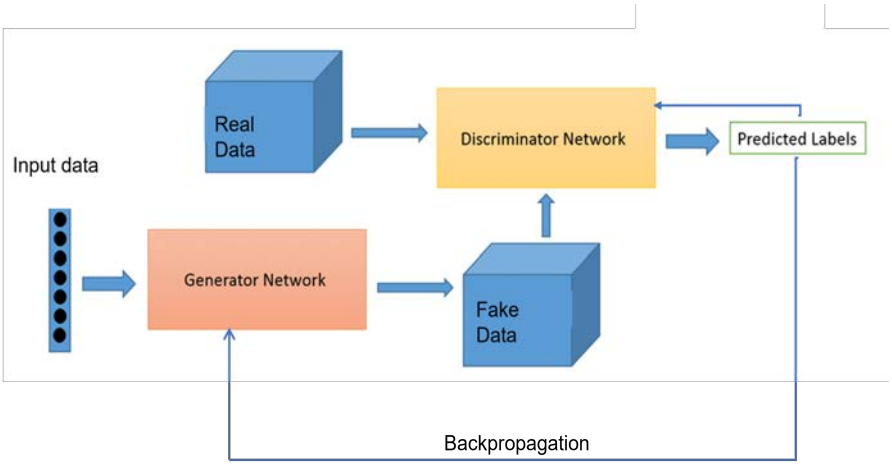


Figure 2.6. General GAN architecture.

CEGAN is a type of GAN architecture *combined* with Encoder-Decoder networks that can generate realistic images by filling in missing or corrupted parts of an input image. CEGAN was first introduced in a research paper in 2016 [14]. While CEGAN

was originally proposed for image inpainting, the concept of context encoding was also applied to audio signals [15] [16], which is known as audio reconstruction. Context Encoder Decoder GANs use an encoder to extract the features of the input data and a decoder to generate the output data. The encoder-decoder architecture of the network allows it to learn the structure of the input data which will be useful to generate the missing parts of the data.

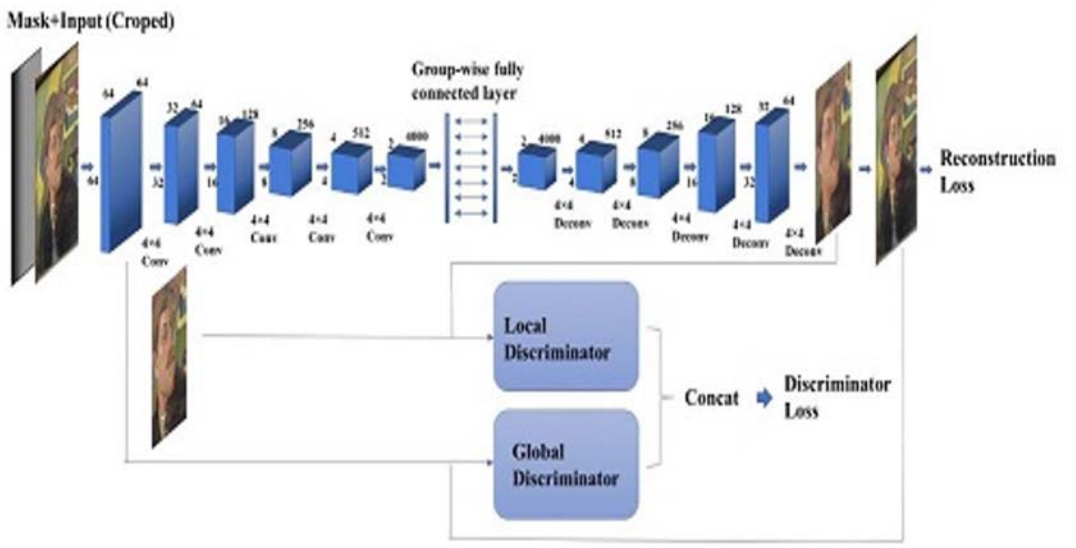


Figure 2.7. Example for encoder-decoder network with a multi-scale GAN used in [17].

2.6. Convolutional Neural Network

Convolutional Neural Network (CNN) is a type of neural network architecture that is commonly used for image and video analysis tasks and lately, was applied to the audio context, In a CNN [18], the input data is typically an image, audio, or a video, which is processed through several layers of convolutional filters. These filters extract features from the input data by performing a convolution operation, which involves sliding a small window over the input data and computing the dot product between the filter weights and the values in the window. The output of the convolutional layer is

then passed through one or more activation functions, such as ReLU (rectified linear unit) or sigmoid, which help to introduce non-linearity into the network and make it more expressive. After the convolutional layers, the output is typically flattened and passed through one or more fully connected layers, which act as a classifier and make a prediction about the input data. The output of the final layer is often passed through a softmax function, which normalizes the output and produces a probability distribution over the different classes. The key advantages of CNNs with audio are their ability to extract local features from audio using 1D convolutional filters [19]. Prior to the use of CNNs, audio data often required extensive feature engineering to extract useful information from the raw signal [20], [21]. CNNs can learn to extract useful features directly from the raw signal, eliminating the need for manual feature engineering. In addition, CNNs can be trained to be robust to noise in the input signal, which can be especially important in real-world scenarios where audio data is often corrupted by environmental noise[22].

2.6.1. CNN layers

A typical CNN architecture consists of several layers, each of which performs a specific function [18]. The mos commonly used layers in a CNN are:

- Input layer: This layer accepts the input data, which is typically an image or a 1D signal such as audio.
- Convolutional layer: This layer performs convolutional filtering on the input data, which helps to extract local features from the data. The output of this layer is typically a set of feature maps.
- Activation layer: This layer applies an activation function to the output of the

convolutional layer, introducing non-linearity into the network and making it more expressive. Common activation functions include ReLU (rectified linear unit) and sigmoid.

- Pooling layer: This layer downsamples the output of the previous layer by taking the maximum or average value within a small region of the feature maps. This helps to reduce the dimensionality of the data and make the network more computationally efficient.

- Dropout layer (optional): This layer randomly drops out a certain percentage of the neurons in the network during training, which helps to prevent overfitting and improve generalization performance.

- Fully connected layer: This layer takes the output of the previous layer and passes it through a set of fully connected neurons, which perform a classification task or produce a regression output.

- Output layer: This layer produces the final output of the network, which could be a class label (in the case of classification) or a numerical value (in the case of regression).

Overall, the combination of these layers in a CNN architecture helps to extract local features from the input data, perform hierarchical feature learning, and produce a final output that is suitable for the given task.

2.6.2. Loss function

In machine learning, a loss function is a measure of how well a model is able to predict the output for a given input [23]. The goal of training a model is to find the set

of parameters that minimize the loss function, thereby improving the accuracy of the model's predictions. Adam (Adaptive Moment Estimation) is a variation of the famous stochastic gradient descent (SGD) used to update the parameters of a neural network during training. Adam uses both the gradient of the loss function and the second moment of the gradient to adaptively adjust the learning rate for each parameter. This allows the algorithm to converge faster and more reliably than other optimization algorithms, such as stochastic gradient descent [23]. The Adam optimization algorithm is typically used in conjunction with a specific loss function, which depends on the problem being solved. In our problem we use a combination between mean absolute error (MAE) and mean squared error (MSE):

The mean absolute error (MAE) loss function, computes the absolute difference between the predicted value and the true value and then takes the mean of all the differences. Mathematically, it can be defined as:

$MAE = \sum_{i=1}^{N} |Y_i - \widehat{Y}_i|$

where N is the number of samples in the dataset, Y true is the true value, and $\widehat{Y}$ is the predicted value.

The mean squared error (MSE) loss function, computes the squared difference between the predicted value and the true value and then takes the mean of all the squared differences. Mathematically, it can be defined as:

$MSE = \sum_{i=1}^{N} (Y_i - \widehat{Y}_i)^2$

where N is the number of samples in the dataset, Y true is the true value, and $\widehat{Y}$ is the predicted value.

2.7. Clustering

A clustering algorithm is a type of unsupervised learning algorithm used in machine learning and data analysis to group a set of data points or objects into clusters based on their similarity. The goal of clustering is to identify meaningful patterns or structures in the data that can be used for further analysis or decision-making. There are many different clustering algorithms, each with their own strengths and weaknesses [24]. In this work, we used agglomerative clustering.

2.7.1. K-means clustering

K-means is a popular unsupervised clustering algorithm used in machine learning to group similar data points into clusters based on their similarities [25], [26]. The algorithm works by iteratively assigning each data point to the closest centroid and updating the centroids based on the mean of the assigned data points until a stopping criterion is reached. The basic steps of the K-means algorithm are as follows:

1. Initialize the centroids: Choose K initial centroids randomly from the data points.

2. Assign each data point to the closest centroid: Calculate the distance between each data point and each centroid, and assign each data point to the closest centroid.

3. Update the centroids: Calculate the mean of the assigned data points for each centroid, and update the centroid to the mean.

4. Repeat steps 2 and 3 until convergence: Iterate steps 2 and 3 until the assignment of data points to clusters no longer changes or a maximum number of iterations is reached.

The K-means clustering algorithm is commonly used for tasks such as image segmentation, customer segmentation, and anomaly detection. One of the advantages of K-means clustering is that it is fast and efficient for clustering large datasets. However, K-means clustering can be sensitive to the initial choice of centroids and can converge to local optima rather than the global optimum [25]. To address this, variants of K-means clustering, such as K-means++, have been developed to improve the initialization of centroids and increase the likelihood of convergence to the global optimum.

2.7.2. Agglomerative clustering

Agglomerative clustering is a hierarchical clustering algorithm used in machine learning to group similar data points into clusters. The algorithm works by iteratively merging the two closest clusters into a single larger cluster until a stopping criterion is reached [27]. The resulting cluster hierarchy can be represented as a dendrogram, which shows the sequence of merges that occurred during the clustering process. In agglomerative clustering, the distance between two clusters is determined by a linkage criterion, which can be one of several types:

- Single linkage: The distance between two clusters is defined as the minimum distance between any two points in the two clusters.
- Complete linkage: The distance between two clusters is defined as the maximum distance between any two points in the two clusters.
- Average linkage: The distance between two clusters is defined as the average distance between all pairs of points in the two clusters.
- Ward linkage: The distance between two clusters is defined as the increase in the

sum of squared distances between the points in the two clusters when they are merged.

Agglomerative clustering can be used for a variety of tasks, such as image segmentation, document clustering, and gene expression analysis. One of the advantages of agglomerative clustering is that it can handle different types of distance metrics and linkage criteria, making it a flexible and versatile clustering algorithm. However, agglomerative clustering can be computationally expensive for large datasets, as the algorithm has to calculate the distances between all pairs of data points during each iteration [27].

CHAPTER 3: LITERATURE REVIEW

Many studies have investigated GAN-based data reconstruction for images and sounds using various methodologies. Our research will attempt to find a general processing pipeline that uses GANs to reconstruct audio data. Although we mainly target acoustic data, previous literature highlights similar GAN-based reconstruction techniques for visual data. Thus, we categorize previous work according to the target data type.

3.1. Visual data

The use of GANs for computer vision applications is very popular. In images, when parts of an image are missing, GANs have been used to reconstruct the missing parts so that they appear intact and believable to human audiences. In order to construct matching patches, attention to contextual data is critical when training GANs. Typically, the surrounding features of the missing regions are used as input during network training to produce coherent fillings. This approach produces good results and was deemed useful for reconstruction [28].

The contributions of [29] and [30] proposed an inpainting and character recognition approach using an improved GoogLeNet and DCGANs. Their method was able to inpaint and recognize occluded characters without knowing the precise locations of corrupted sections. However, when large areas of original characters were obliterated and critical parts were damaged, their approach was unable to recreate them.

In [17] and [31], researchers employed an encoder-decoder network with a multi-scale GAN as shown in Fig. 2.7. Despite achieving a fast model, and some impressive output, there remained other instances with unsatisfactory results. Another research

conducted in 2020 [32] used a similar technique, called PEPSI, that used a parallel decoding approach. Their results show that the technique shortens the overall processing time via a parallel decoding path and an effective joint learning scheme.

Another approach for reconstructing the missing parts in images was implemented in [33]. They presented a Wasserstein GANs with new Discriminator and Generator architectures as an improvement method and succeeded in building an application to generate faces based on reference faces' attributes.

Several others resorted to using special loss functions such as that of [34], [35] which used a modified loss function by combining MSE loss function with MAE during GAN training for having a better picture reconstruction. Similarly, the authors of [36] suggested a semantic image reconstruction technique based on adversarial loss and a self-learning encoder-decoder model. The authors argue that the structural integrity and the clarity must be preserved in an effective picture restoration method. Other known contributions include that of [37] which used a GAN-based approach for facial image reconstruction, and the FiNet approach [38] where a generator network combined with an encoder-decoder was used for filling in the gaps in fashion photographs, both delivering good quality of reconstructed images. Table 3.1 summarizes the contributions of this field.

Despite previous methods delivering good performance in image inpainting, the algorithms require high computational cost resulting in long training times. In addition, all the above methods considered training one GAN for the entire dataset except [5], who presented a picture inpainting model for cultural images based on a divide-and-conquer method. The basic idea is to group cultural images with similar visual contexts and then train a generative model for each group. When the system is presented with

an incomplete image, it first determined the image's category and then used the GAN associated with that category to accomplish inpainting. Our work is inspired by this contribution. However, we target the audio domain instead of the visual one which poses new challenges.

Table 3.1. Summary for the latest studies that used GANs to regenerate visual data.

Reference	Year	Architecture used	Dataset	Evaluation method
[28]	2018	WGAN with contextual attention layer	CelebA, ImageNet, DTD	PSNR Visualization comparison
[5]	2019	Apply a divide-and-conquer strategy prior to training DC GAN	WikiArt, Metropolitan Museum, Rijksmuseum	Visualization comparison, Survey
[32]	2020	Encoder-decoder network with a multi-scale GAN	CelebA-HQ, Place2	PSNR, SSIM, Visualization comparison
[35]	2022	Two adversarial discriminators were used in GAN	CelebA, CelebA-HQ	PSNR, SSIM, Visualization comparison

3.2. Sound data

Deep learning succeeded in generating synthesized sounds using models such as SampleRNN [39] and WaveNet [40] with autoencoders [41], [42] based on the wave features, while [8] synthesized speech. In the latter, Text-to-Speech (TTS) relied on extracted Mel spectrogram features.

Filling the gaps in audio signals was introduced first by Adler et al. [43], they developed a tool to reconstruct gaps in the audio, based on the OMP method to extract

the features from the audio domain and called it “audio inpainting”. The authors of [44], [45] focused on exploiting TF sparsity to reconstruct audio signals. However, these methods are only effective for inpainting short gaps that are less than 10ms. Therefore researchers tried to develop more effective methods as done by [46] where they utilize sparse modeling in the TF domain by developing a dictionary learning technique. This technique learns the dictionary from reliable sections adjacent to the gap. They aimed to achieve a signal representation that exhibits enhanced sparsity in the TF domain, but even here, the maximum gap they could fill was (10 ms – 100 ms).

New techniques for inpainting audio were presented based on deep learning to extract and learn the features. One of the most popular DL techniques that have been used in inpainting is GAN. One of the studies that used GANs to reconstruct the missing part of sounds considered the neighboring borders [47]. The research group used objective difference grading (ODG) and magnitude spectrograms [15] as evaluation measures which are also used in this work and described in section 5.2.

Other researchers were able to fill a larger gap in the audio as GACELA [15] and VIAI [16]. In GACELA the idea is to use a Wasserstein cGAN that can restore missing audio data with intervals ranging from hundreds of milliseconds to a few seconds. The authors of VIAI successfully filled an 8 milliseconds gap in a 4 second audio-video segment by creating synthetic versions of missing audio for videos that do not have it using a semantic image inpainting model based on spectrogram audio features. There are other studies that worked on inpainting audio using other models, such as DNN [48], [49]. Table 3.2 summarizes the studies that targeted audio completion using GANs.

Despite showing some success, none of the previous studies examined the effectiveness of training GANs in a similar “divide and conquer” fashion to Jaboor et al’s

image inpainting in [5] which was described in the previous section. This work inspired us to investigate the feasibility of using a similar strategy on audio reconstruction while using a contextual encoder-decoder GAN architecture such as that in [5].

Table 3.2. Summary for the latest studies that used GANs to regenerate audio data.

Reference	Year	Architecture used	Dataset	duration	Evaluation method
[48]	2019	Context encoder decoder	NSynth	64ms	ODG, SNRs
[47]	2020	WGAN	PIANO, SOLO, MAESTRO	500 ms	ODG, SNRs
[15]	2020	cGAN	Built their own dataset	375 to 1500 ms	ODG
[16]	2019	Generate missing audio segments that correspond to their accompanying videos using WaveNet decoder	MUSIC-Extra-Solo	Not disclosed	SSIM, PSNR, MSE, User study, Visualization

CHAPTER 4: PROPOSED METHOD

After trying to reconstruct audio gaps with GANs trained on whole datasets, we found that training took a long time, and results were not convincing. From this point, we hypothesized that clustering samples into similar groups and training special GANs for each group cluster rather than a single general GAN on the whole dataset may lead to performance improvements. To this end, we propose an audio reconstruction paradigm which applies the *divide-and-conquer* strategy to separate audio contexts prior to training specialized-GANs for each context as summarized by Fig. 4.1. The system is composed of four steps – namely, data representation, clustering, GAN training, and audio completion which are discussed next.

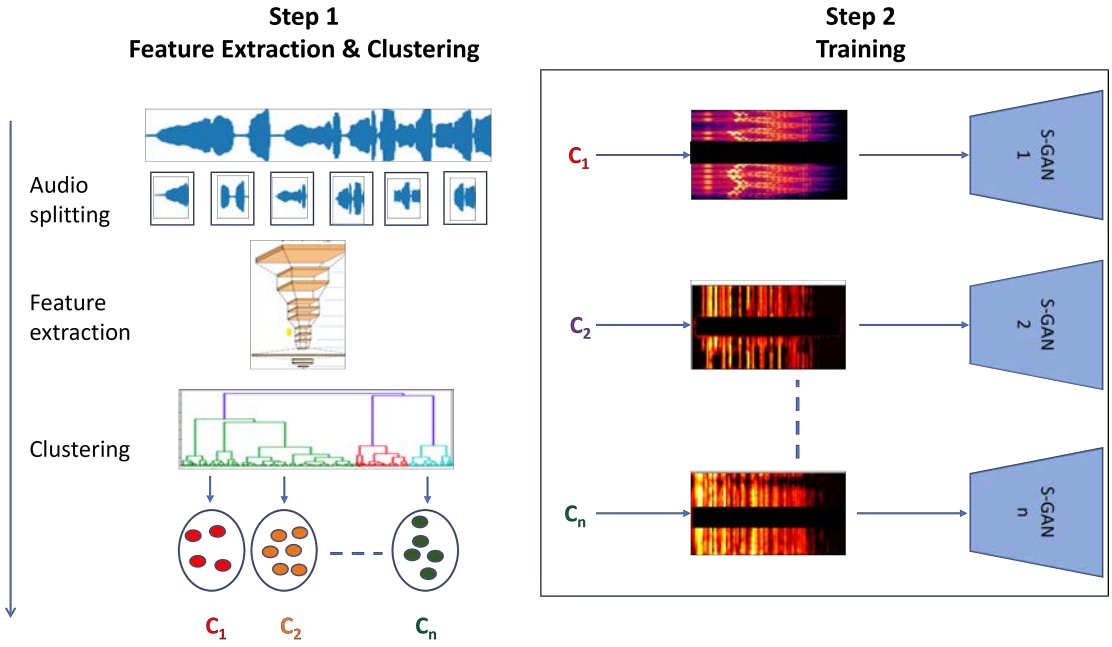


Figure 4.1. The system pipeline. In the first step, the sound is separated into small sound segments and converted into spectrograms that are fed to a CNN model for feature extraction. After that, the sounds are clustered using agglomerative clustering. During training, the center part of each sound sample is removed and a specalized-GAN is trained to reconstruct the missing part. The last step is where we do the cluster lookup and audio completion as shown in Fig. 4.3.

4.1. Audio data representation

Choosing the right data representation is of paramount importance since most clustering approaches employ feature-dependent distance functions. As such, single-instrument music files of a public dataset were first divided into 1-second clips and a pre-trained model is used to extract the pertinent features. For the sake of completeness, two popular audio pre-trained models were used; VGGish model [50] and Yamnet model [51]. VGGish is a deep learning model developed by Google for audio classification and feature extraction. It is based on the VGG architecture and was designed to classify short audio clips of human speech and music. The model is trained on a large dataset of audio recordings and can be used to extract features from audio recordings to use in downstream applications such as audio classification, speech recognition, and audio search. VGGish can be used to identify different types of sounds, such as speech, music, and background noise. It can also be used to extract features such as pitch, timbre, and loudness. Yamnet is an audio classification developed by Google AI's research team. It is designed to recognize a wide range of sounds and classify them into one of 521 audio event categories, including human and animal sounds, musical instruments, natural sounds, and mechanical sounds. The Yamnet model is based on a multi-scale convolutional neural network architecture, which processes audio waveforms at a different time and spectral resolutions. The model is pre-trained on a large-scale dataset of labeled audio clips, and the learned feature representations are used to classify new audio signals.

4.2. Clustering audio

In order to train a specialized-GAN for each audio context separately, sound clips must first be clustered into groups based on their similarity. We applied agglomerative clustering to separate data and chose the ideal number of clusters using a dendrogram [52]. Additionally, the popular K-means algorithm was used as an alternative approach to be compared against.

4.3. Training strategy

In contrast to the traditional method of training one general GAN on the entire dataset, our proposed method trains a specialized GAN for each audio cluster separately. The goal is to apply the divide-and-conquer strategy to minimize variance within each cluster thereby reducing the complexity of the distribution to be learned. To train each specialized-GAN, an encoder-decoder approach is used as shown in Fig. 4.2. Specifically, the architecture includes five convolutional neural network layers in the encoder. The encoder and the decoder are then connected by ReLU activation functions along with batch normalization, whereas the decoder begins with a fully connected layer and is followed by five transposed convolutional layers. Audio signals are divided into 3 parts (i.e., left context with 400ms (LC), the right context (RC) with 400 ms and 200 ms gap (G) in the middle) are used to train the GAN in a similar fashion to previous state-of-the-art techniques; applying Mel spectrogram to the wave files and using normalization and rescaling to the range [-1, 1] to obtain the output of G'. We use LC and RC as input to the GAN.

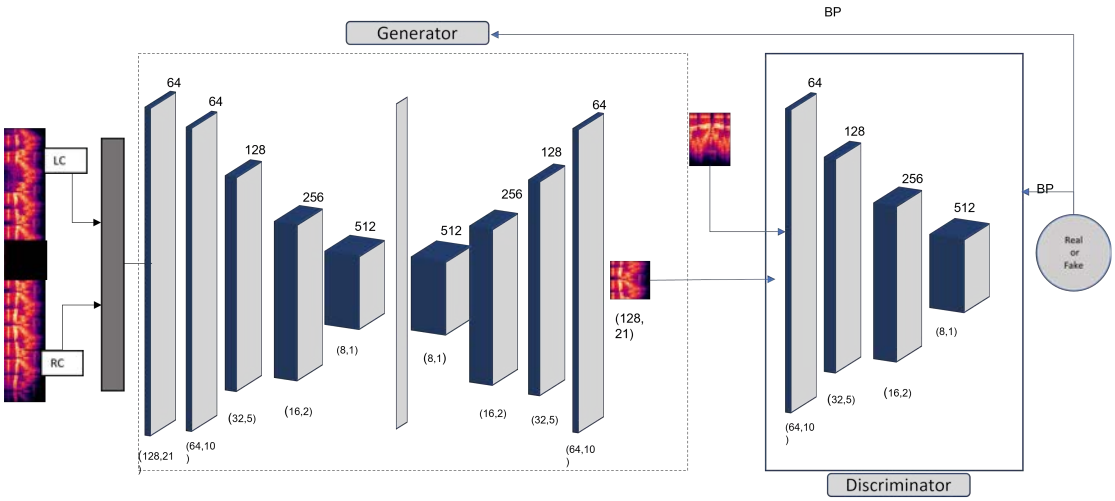


Figure 4.2. Context-Encoder Generative Adversarial Network architecture.

4.4. Cluster lookup and audio completion

This step commenced by identifying the closest signal from the training dataset to the testing signal under consideration. Subsequently, the testing signal was assigned to the specialized GAN associated with the closest signal from the training dataset. The determination of the closest signal was accomplished by calculating the smallest distance using the Euclidean distance measure which is used widely to find the similarity between audio data [53]. Fig. 4.3 illustrates the cluster lookup and audio completion pipeline.

Instead of training a single general GAN on the entire dataset, the proposed method trained specialized GANs for each audio cluster separately. This divide-and-conquer strategy aimed to reduce variance within each cluster, simplifying the distribution to be learned. The GAN architecture employed an encoder-decoder approach, with convolutional neural network layers in the encoder and transposed convolutional layers in the decoder. The GAN was trained using left and right context audio signals, along with a gap in the middle, to generate the missing audio segment.

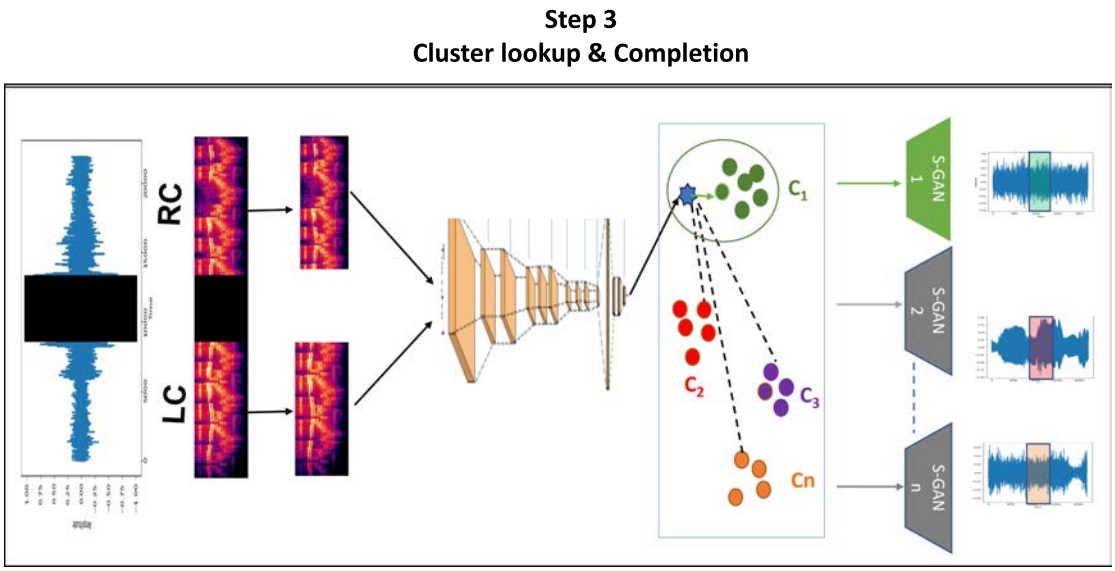


Figure 4.3. Cluster lookup and completion process.

Finally, the audio completion step involved inserting the generated audio segment between the left and right context audio, reversing the preprocessing steps, and applying the inverse Mel spectrogram function which was borrowed from the library Librosa [54] to obtain the reconstructed audio in waveform format.

In summary, this chapter presents a comprehensive methodology for audio reconstruction; the proposed paradigm has the potential to improve the quality and efficiency of audio reconstruction tasks.

CHAPTER 5: EXPERIMENTAL SETUP

The used GAN implementation is close to that followed by [5], [14], [16] with minor modifications such as the input shape for GAN, number of layers in the encoder-decoder and the model's parameters. The model was implemented in Python using PyTorch and trained on an Intel(R) Xeon(R) CPU @ 2.20GHz processor with 12 cores and 84GB of RAM coupled with an NVIDIA A100 GPU with 40GB VRAM. All audio samples were preprocessed using the Mel spectrogram from the Librosa library at a sampling rate of 22050Hz with hop length 0.125 out of the sample rate, windows length 0.05 of the sample rate and 2046 to the length of the Fast Fourier Transform window (FFT). The audio segment length used is 1 second (1000 ms), with a silence of 200 ms in the center. The dataset is divided into a training set which is used to train the specialized GANs and the general GAN, and a testing set which is used only during the testing phase. To ensure fairness, the exact same GAN architecture was used for the generalized approach that is trained on the entire dataset. It is a worthy note that we trained and tested specialized and general GANs with the same training and completion hyper-parameters. The aim of the experiments is to answer the following research questions:

- RQ1: Which audio representations are most suitable for effective clustering?
- RQ2: Which clustering techniques are sufficient to generate good-quality audio clusters?
- RQ3: How do the completion results of specialized GANs (one per cluster) compare to a general GAN trained on the entire dataset?
- RQ4: How do specialized GANs perform when the cluster memberships of unseen

damaged sound clips are unknown? And can the trained models generalize well even when tested on a different musical instruments dataset?

- RQ5: How does the training computation cost of specialized GANs compare to a general GAN?

5.1. Datasets and pre-processing

To evaluate the performance of the proposed method, the SOLO dataset from [55] was used. This dataset consists of recordings of numerous instruments (i.e, accordion, acoustic guitar, cello, flute, saxophone, trumpet, violin, and xylophone) which are each played alone. During pre-processing, audio was segmented into one-second duration segments for each file. Upon segmenting the sound clips, some segments were found to be silent (i.e., their dB is between –20 and –80 dB) and with durations lasting longer than 0.001-second [56]. Such identified segments were eliminated as it made no sense to reconstruct the silence. In the end, the original 252 sound files were segmented into 36201 1-second segments. In order to evaluate the generalizability of the specialized GAN, we sought to find a new instrument that was not present in the SOLO dataset, which was used for training our models. For this purpose, we selected the Maestro dataset [57]. The Maestro dataset focuses specifically on piano music and consists of a large-scale collection of high-quality audio recordings of virtuosic piano performances, totaling over 172 hours. The same data preprocessing steps were applied to the Meastro dataset, including splitting the audio into segments and removing silent portions. By doing so, we selected randomly 256 testing samples from the Maestro dataset for the piano instrument.

5.2. Evaluation measures

We compared the reconstructed audio from the general GAN and the reconstructed audio from the specialized GAN to the real audio, as well as the specialized GAN trained on clusters generated from VGGish features to clusters generated from Yamnet features, using both qualitative and quantitative methods.

- Quantitative: The first evaluation was done using the Mel spectrogram images; after converting the original audio and the inpainted audio to Mel spectrogram images, we calculated the PSNR values [58] where a larger value is associated with better performance. Further more, we calculated the Mean Squared Error (MSE) to evaluate the similarity of the reconstructed raw audio to the original one, where a lower value signifies better reconstruction.

- Qualitative: A visualization of the inpainted sounds in terms of their raw audio signal and spectrogram representation was used to evaluate the reconstruction quality. Randomly selected samples filled by their respective specialized GANs were compared to their equivalent filled by the general GAN and the ground truth to assess their quality. Additionally, a survey was conducted asking 17 participants to listen to sounds generated by specialized GANs and the same samples generated by the general GAN along with the ground truth samples and rate the quality between 1 and 5 (i.e., 1 being the worst quality and 5 the best quality). The survey can be seen in appendix A. This method is considered the gold-standard to evaluate audio since qualitative approaches could give unreliable results [59].

CHAPTER 6: RESULTS AND DISCUSSIONS

In this section, we will discuss the results of our experiments for each particular research question respectively.

6.1. Audio representations and clustering (RQ1 and RQ2)

The first part of our framework is to group audio contexts based on similarity. Our first thought was to group audio clips based on the instrument type and train a different GAN model for each group. However, preliminary results revealed that training specialized GANs based on this grouping strategy did not provide improvements in terms of reconstruction quality. It comes as a result of splitting the audio into 1-second segments that each segment now has different harmonics though it is from the same instrument. As a result, we sought popular unsupervised clustering approaches instead, namely K-Means and Agglomerative clustering. Most distance-based clustering approaches require proper data representation (with relevant features) to work correctly. To this end, we applied two popular pre-trained DNN models used in the audio domain (Yamnet and VGGish described in section 4.1) to extract meaningful representations. To visualize the clustering performance of the audio representation and clustering combinations, we plot the generated clusters using the TSNE visualization technique [60] for each combination accordingly. To determine the best combination between the clustering algorithm and feature extraction model for generating high-quality clusters, we calculated the silhouette score. The silhouette score is a measure of how well each data point fits into its assigned cluster, based on the distances between the point and the points in the neighboring clusters. The silhouette score ranges from [-1,1], with higher values indicating better clustering [61]. The best combination, as represented in table 6.1, is

the agglomerative model with VGGish features; though the k-means clustering model is more popular and faster, this result may be due to specifying the number of clusters at 8; perhaps reducing the number of clusters will yield a different result, which could be a future direction to test.

Table 6.1. Silhouette score for the different combinations between clustering model and feature extraction model.

Clustering model / Feature extraction model	VGGish	Yamnet
Agglomerative clustering	**0.166**	0.154
K-means clustering	0.133	0.125
Clustering based on instruments type	0.132	0.108

Also after using TSNE visualization, we realized that clips from the same instrument were not always close to each other. As depicted in Fig. 6.1 and Fig. 6.2, when we compared the clustering audio based on the category with other feature extraction models, we found clusters created using the agglomerative clustering with VGGish and Yamnet feature extraction models were denser and well separated than using instrument types. It comes as a result of splitting the audio into 1-second segments that each segment now has different harmonics though it is from the same instrument.

To cluster the dataset, the CNN features vector generated by VGGish or Yamnet is passed to the agglomerative clustering algorithm. This step may appear insignificant, but it significantly impacts the clustering methods as illustrated in Fig. 6.3 and Fig. 6.4 the distribution of instruments differs based on the feature extraction model e.g using Yamnet the eighth cluster has only 17 samples while using VGGish almost most of the clusters have a significant number of samples. For each cluster produced we trained a GAN except cluster number 7 in Yamnet because it doesn't have sufficient numbers of

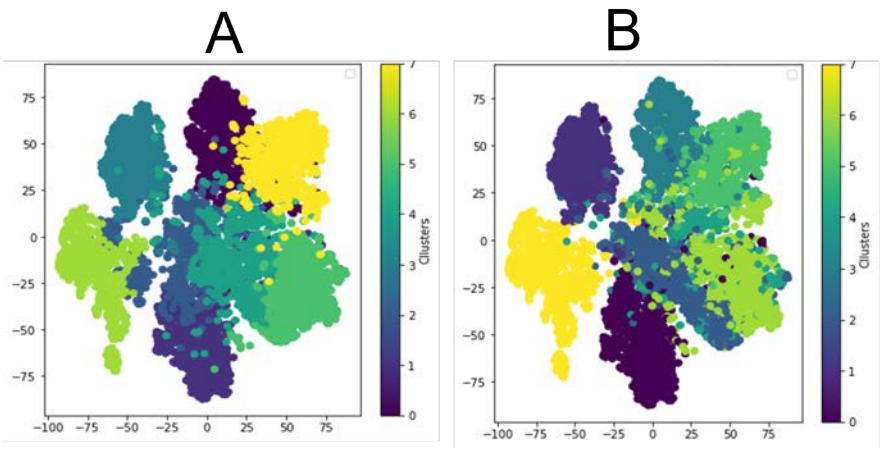


Figure 6.1. 2-dimensional TSNE visualization of clusters generated by an agglomerative clustering method with VGGish features (A), and based on the musical instrument's class (B).

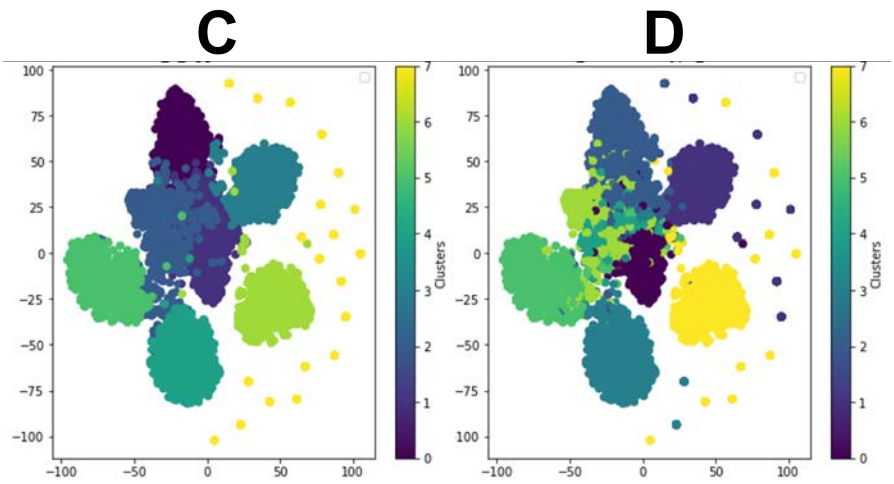


Figure 6.2. 2-dimensional TSNE visualization of clusters generated by an agglomerative clustering method with Yamnet features (A), and based on the musical instrument's class (B).

samples to train. Moreover, from Table 6.4 by comparing the average for the PSNR we found that the Specialized GAN from VGGish features outperforms the specialized GAN from Yamnet features while the MSE shows the opposite, however, independent user evaluations indicate that the sounds generated by Specialized GAN from VGGish features are more clear and comparable to the ground truth than the ones generated by the specialized GAN from Yamnet features as shown in Fig. 6.7. One possible explanation for this is that VGGish extracts feature at a finer time resolution than Yamnet, which makes it better suited for applications where fine-grained temporal information is important.

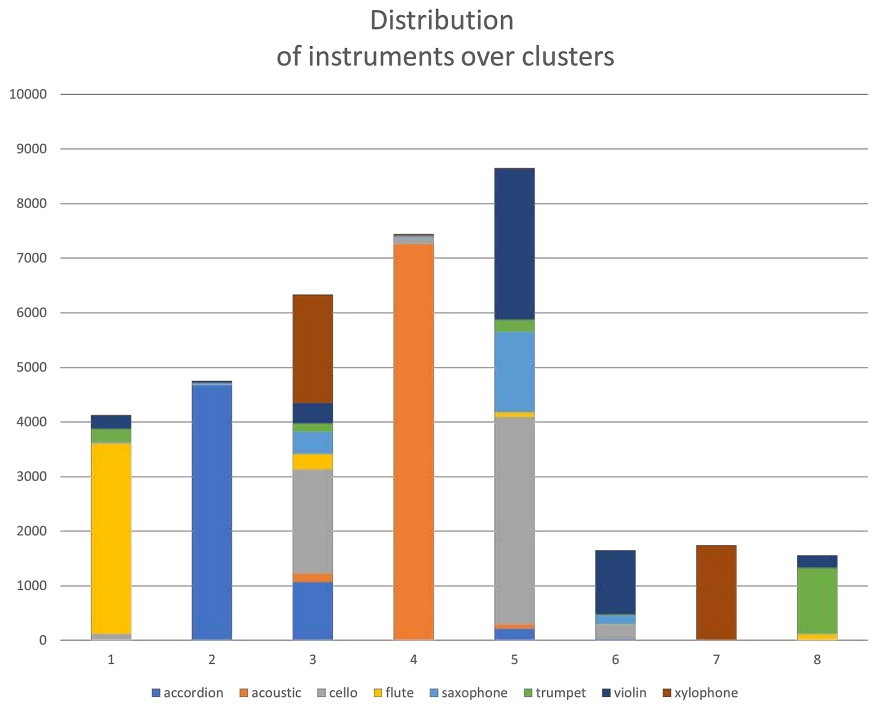


Figure 6.3. The distribution of instruments over clusters when using VGGish features.

6.2. Performance comparison of specialized GANs and general GAN (RQ3)

To answer the third research question we compared the specialized GANs trained on clusters generated by both feature extraction methods (Yamnet and VGGish) with

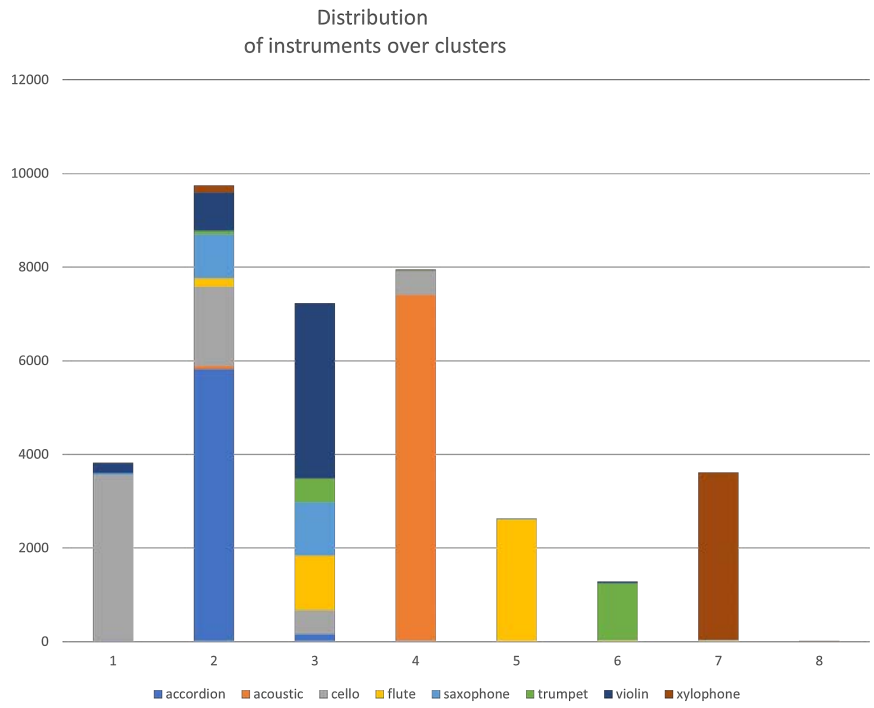


Figure 6.4. The distribution of instruments over clusters when using Yamnet features.

the general GAN, and then they are compared against each others, using quantitative evaluation methods followed by the qualitative measures.

In Fig. 6.5 and Table 6.2, we compare the average PSNR and MSE for 10 samples from each cluster using reconstructions from the general GAN and from the respective specialized GAN of the sample (using the VGGish-based clusters). We found that the PSNR for the specialized GAN was more favorable than the PSNR for the general GAN for all clusters. Similarly, the average MSE value for all samples generated by the specialized GAN is closer to zero than the MSE for the same samples generated by the General GAN. Thus, higher-quality sounds can be generated using specialized GANs. The same outcome is observed in Fig. 6.6 and Table 6.3 where the comparison was between the specialized GAN that is based on clustering using Yamnet features and the general GAN for each cluster. Here also, the specialized GAN beat the general GAN; the results of the experiments show a clear advantage when using the specialized-GAN

setup over the traditional general GAN approach. The difference in performance is more apparent in some clusters than others. For example, Table 6.2 shows that the gap of clusters 2, 4, and 8 are more pronounced than the others. This could be attributed to these clusters having distinct sound samples when compared to others. The same performance difference is also observed on some samples in Fig. 6.8. The first (top) example experienced less degradation with the use of the general GAN than the other two. It is worth mentioning that in addition to the produced quality, the general GANs always required a higher number of epochs to converge due to the complexity of the dataset they are trying to model.

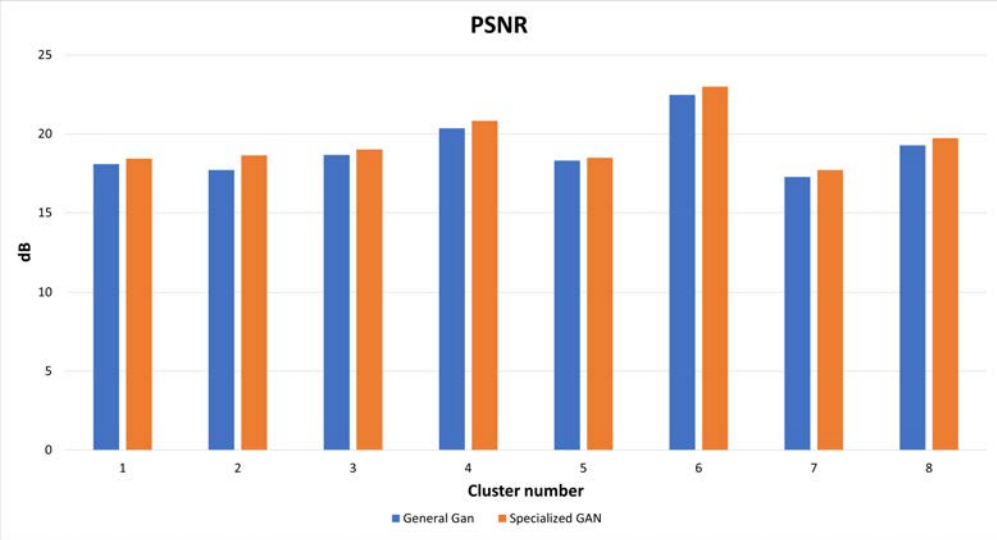


Figure 6.5. The average PSNR for 10 samples generated by the specialized GAN (VGGish clusters) and by the general GAN.

Table 6.2. The average MSE for 10 samples generated by the specialized GAN (VGGish clusters) and by the general GAN.

Cluster number	General GAN	Specalized GAN
Cluster 1	0.090	0.076
Cluster 2	0.126	0.090
Cluster 3	0.102	0.080
Cluster 4	0.100	0.070
Cluster 5	0.075	0.072
Cluster 6	0.145	0.124
Cluster 7	0.128	0.086
Cluster 8	0.205	0.142

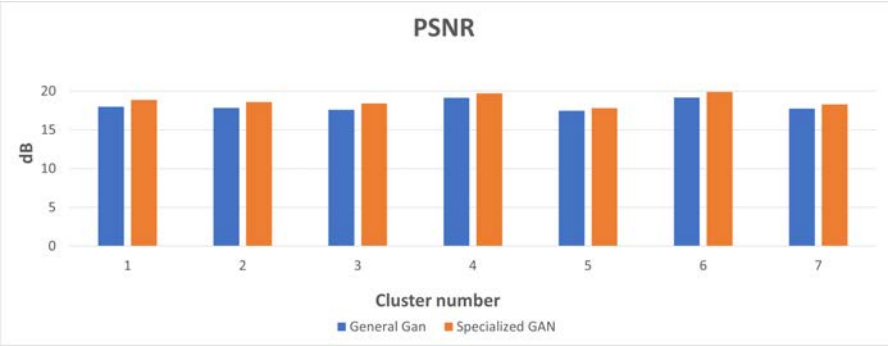


Figure 6.6. The average PSNR for 10 samples generated by the specialized GAN (Yamnet clusters) and by the general GAN.

Table 6.3. The average MSE for 10 samples generated by the specialized GAN (Yamnet clusters) and by the general GAN.

Cluster number	General GAN	Specalized GAN
Cluster 1	0.082	0.047
Cluster 2	0.108	0.062
Cluster 3	0.148	0.100
Cluster 4	0.142	0.089
Cluster 5	0.105	0.086
Cluster 6	0.199	0.126
Cluster 7	0.106	0.064

Table 6.4. The average PSNR and the MSE of the three models.

GAN	PSNR	MSE
General-GAN	19.0197	0.1213
Specialized-GAN (VGGish)	**19.4788**	0.0924
Specialized-GAN (Yamnet)	19.1377	**0.0644**

In terms of qualitative evaluation, according to the evaluations of independent users, as described in the evaluation section 5.2, the sound quality produced by the specialized GANs was significantly better than the sounds produced by the general GAN as shown in Fig. 6.7. Also, Fig. 6.8 compares the samples reconstructed using the specialized GAN and the ones reconstructed using the general GAN. Evidently, the specialized GANs generated sound waves and spectrograms comparable to the original sounds.

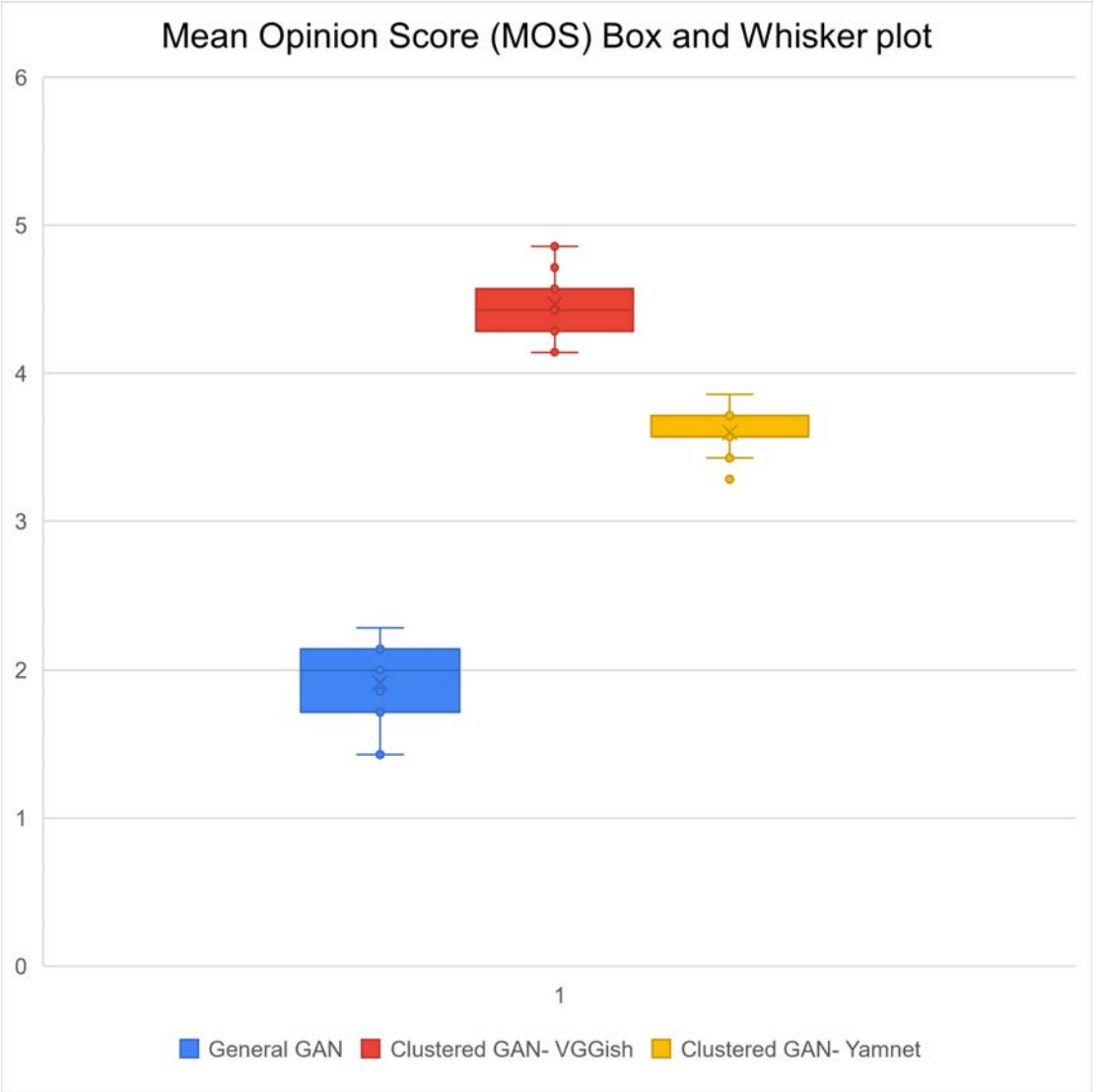
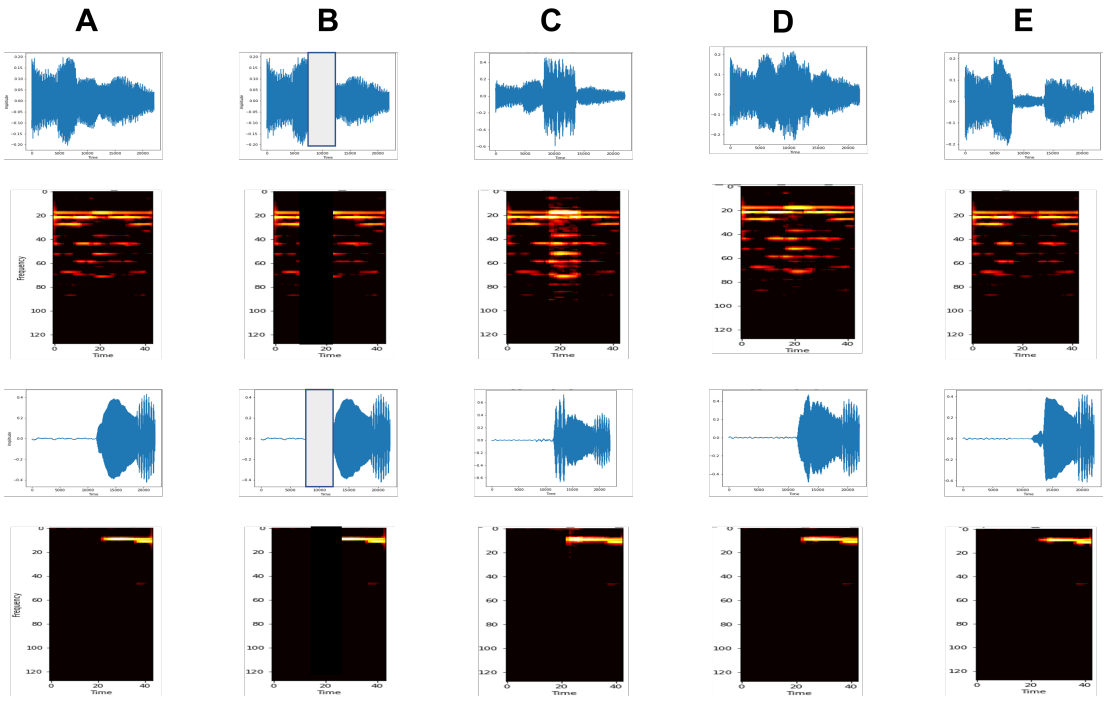
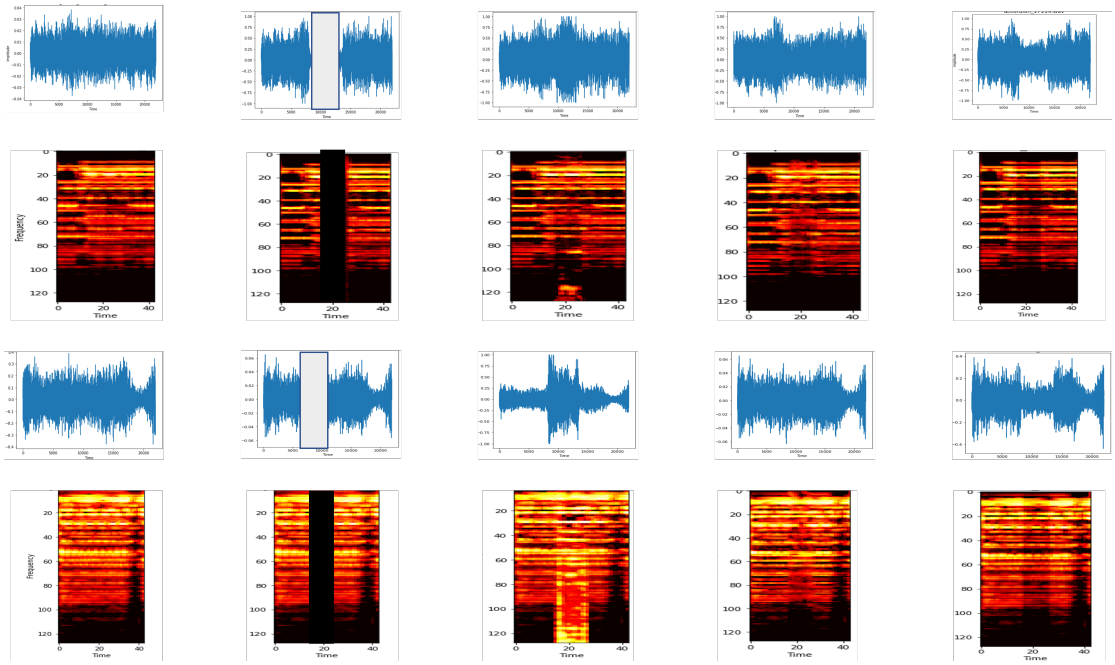


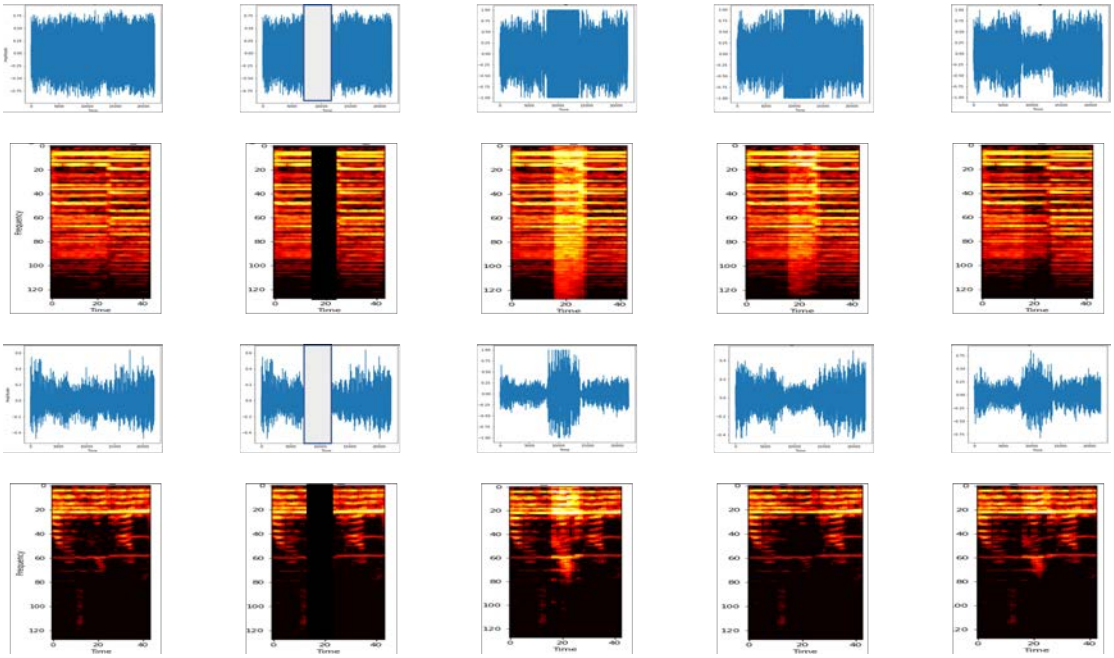
Figure 6.7. User study results of a comparison between the quality of two generated sounds one by general GAN and one by specialized GAN with reference to ground truth.



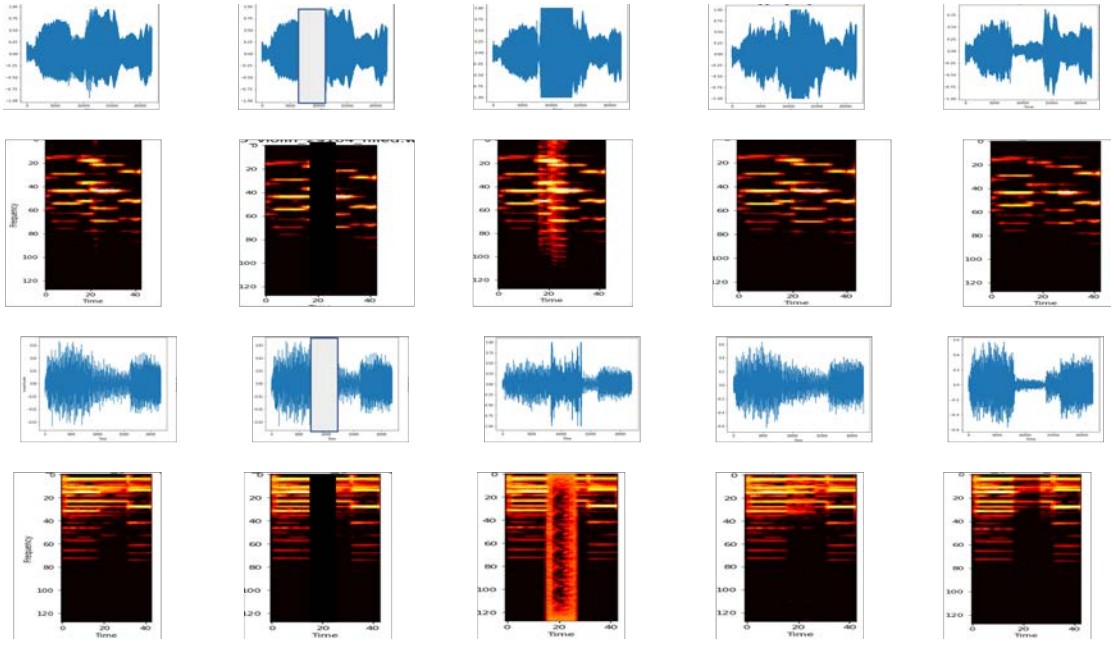
(a) Reconstruction sample results from cluster 1. (A) is the original audio, (B) masked audio, (C) audio generated by general GAN (D) audio generated by specialized GAN (VGGish clusters), and (E) audio generated by specialized GAN (Yamnet clusters).



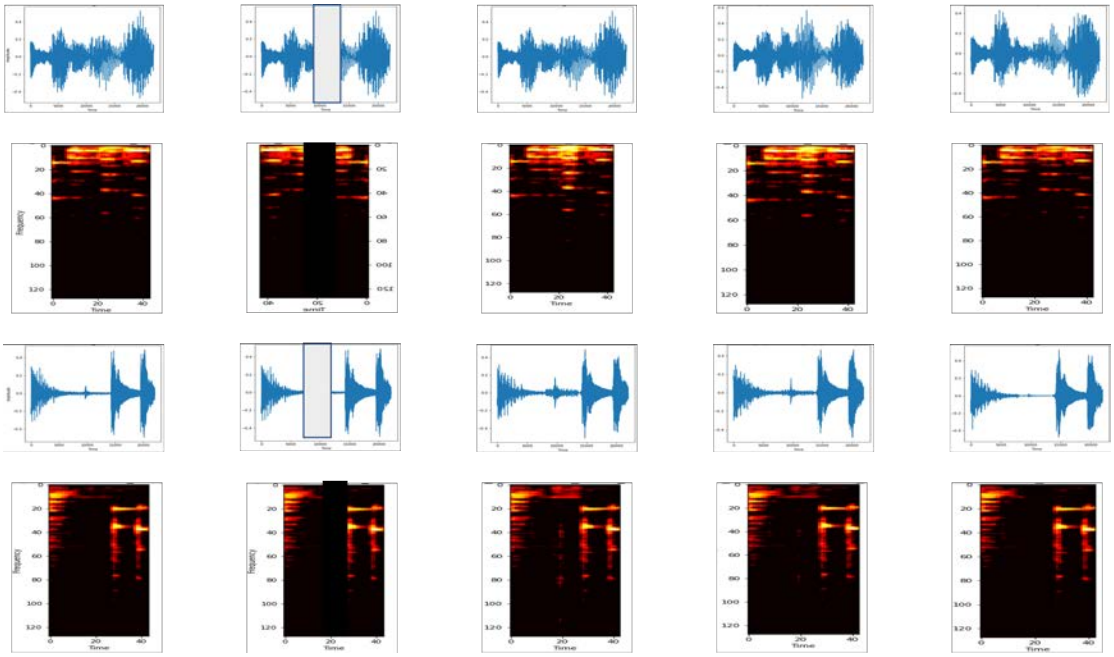
(b) Reconstruction sample results from cluster 2. (A) is the original audio, (B) masked audio, (C) audio generated by general GAN (D) audio generated by specialized GAN (VGGish clusters), and (E) audio generated by specialized GAN (Yamnet clusters).



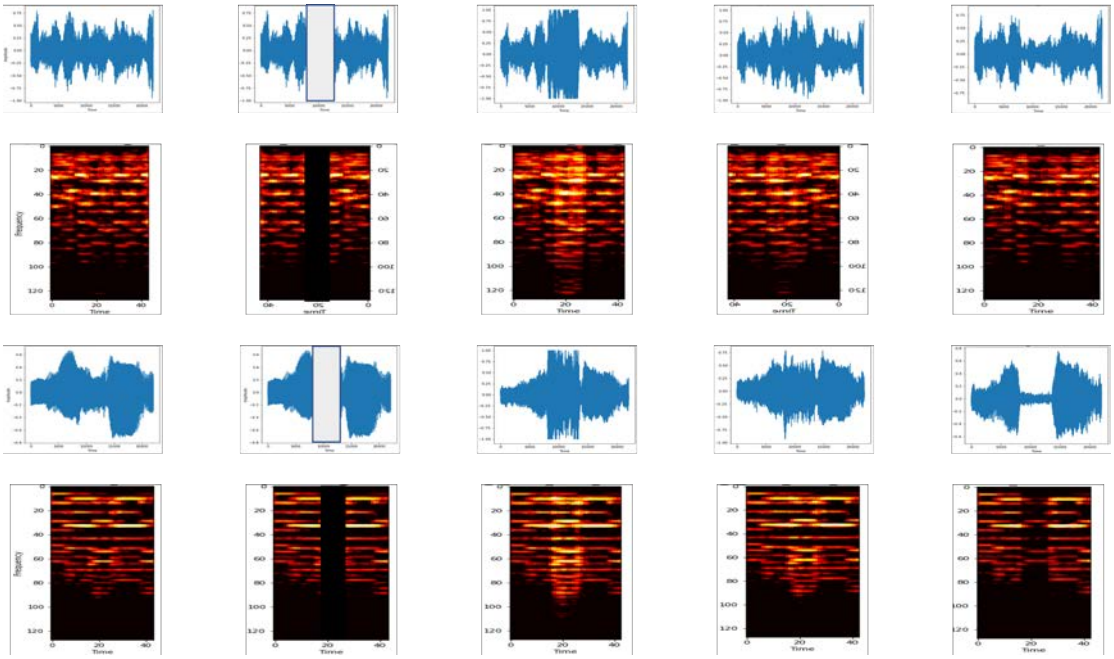
(c) Reconstruction sample results from cluster 3. (A) is the original audio, (B) masked audio, (C) audio generated by general GAN (D) audio generated by specialized GAN (VGGish clusters), and (E) audio generated by specialized GAN (Yamnet clusters).



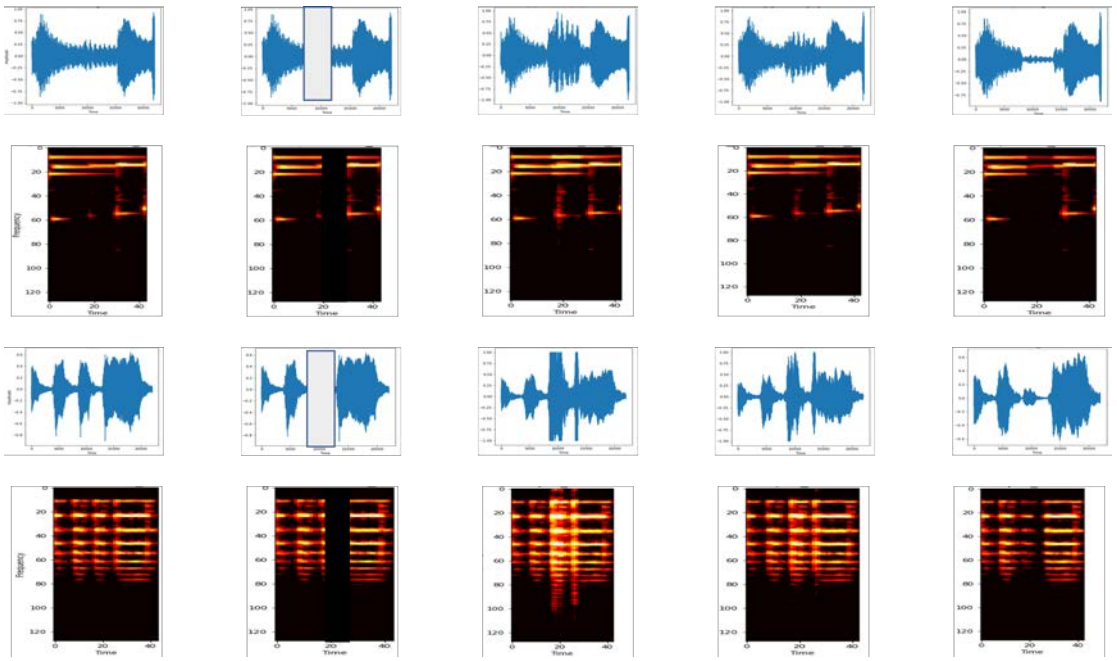
(d) Reconstruction sample results from cluster 4. (A) is the original audio, (B) masked audio, (C) audio generated by general GAN (D) audio generated by specialized GAN (VGGish clusters), and (E) audio generated by specialized GAN (Yamnet clusters).



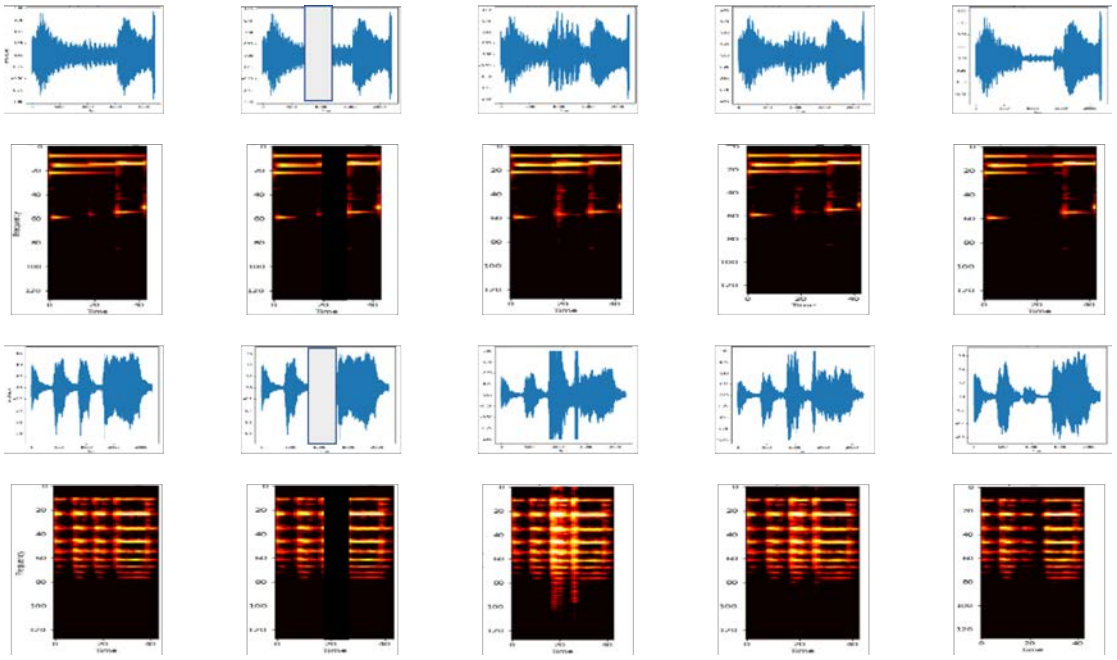
(e) Reconstruction sample results from cluster 5. (A) is the original audio, (B) masked audio, (C) audio generated by general GAN (D) audio generated by specialized GAN (VGGish clusters), and (E) audio generated by specialized GAN (Yamnet clusters).



(f) Reconstruction sample results from cluster 6. (A) is the original audio, (B) masked audio, (C) audio generated by general GAN (D) audio generated by specialized GAN (VGGish clusters), and (E) audio generated by specialized GAN (Yamnet clusters).



(g) Reconstruction sample results from cluster 7. (A) is the original audio, (B) masked audio, (C) audio generated by general GAN (D) audio generated by specialized GAN (VGGish clusters), and (E) audio generated by specialized GAN (Yamnet clusters).



(h) Reconstruction sample results from cluster 8, (A) is the original audio, (B) masked audio, (C) audio generated by general GAN (D) audio generated by specialized GAN (VGGish clusters), and (E) audio generated by specialized GAN (Yamnet clusters).

Figure 6.8. Reconstruction sample results, two samples from each cluster.

6.3. Cluster lookup and validation on a different dataset (RQ4)

To address the fourth research question, we conducted two experiments. The first experiment involved testing samples from the SOLO dataset. We utilized the cluster lookup approach described in section 4.4 to determine the cluster label for each testing sample based on the closest training sample. Subsequently, we generated a new audio signal using the assigned specialized GAN for that cluster. The purpose of this experiment is to find out whether our cluster lookup approach is effective. A comparison was made between the specialized GAN output and the general GAN output, as shown in Table 6.5. We observed that the specialized GAN still outperformed the general GAN in terms of PSNR and MSE values. However, the results are slightly worse compared to those from the previous experiment, where the clusters of test examples were assumed to be known. This may have been attributed to potential errors in assigning samples to the correct cluster during the lookup process.

Table 6.5. The average PSNR and MSE of the General-GAN, and Specialized-GAN (with and without cluster lookup).

GAN	PSNR	MSE
General-GAN	19.0197	0.1213
Specialized-GAN (without lookup)	19.4788	0.0924
Specialized-GAN (with lookup)	19.3451	0.0937

In addition to the quantitative evaluation, we employed visualizations of the samples to demonstrate the effectiveness of the specialized GAN after the cluster lookup in generating spectrograms that closely resemble the original. Fig. 6.9 illustrates that the spectrogram generated by the specialized GAN after the lookup process is closer to

the original and exhibits better quality compared to the spectrogram generated by the general GAN. This indicates that the lookup process successfully identifies the correct cluster for the samples.

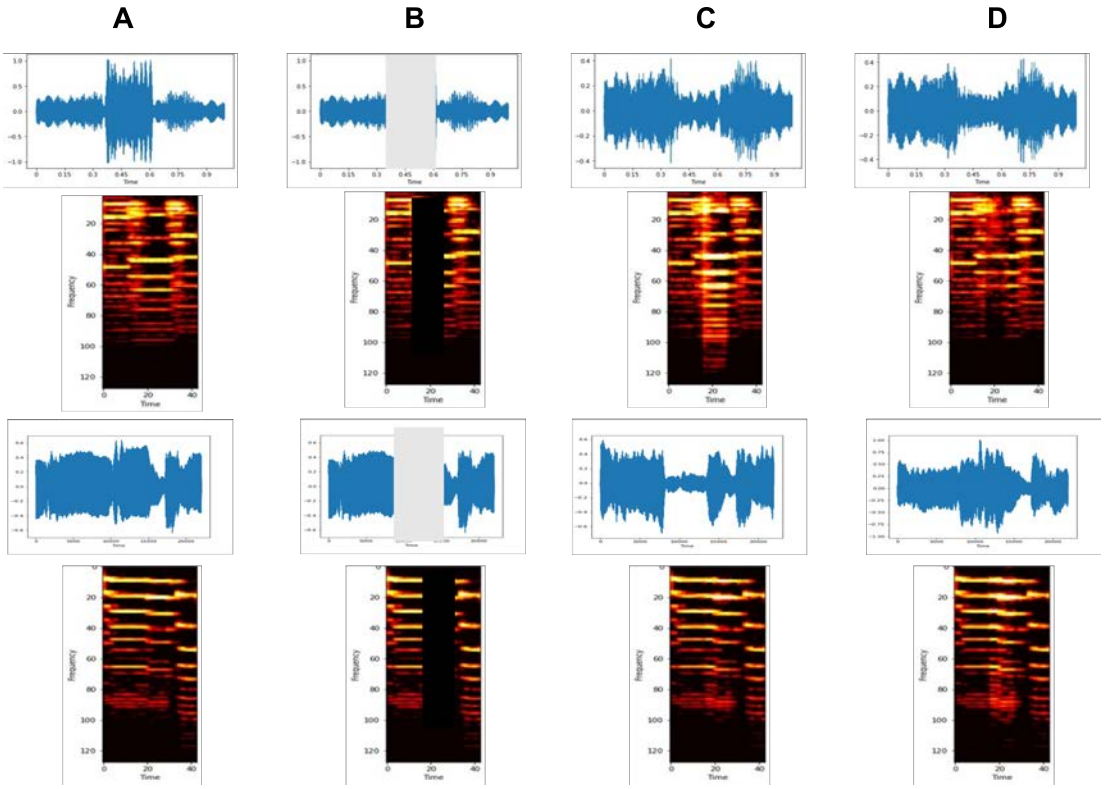


Figure 6.9. Reconstruction sample results after applying cluster lookup. (A) is the original audio, (B) masked audio, (C) audio generated by general GAN (D) audio generated by specialized GAN after cluster lookup.

The second experiment was conducted by calculating the PSNR and MSE metrics for the newly generated piano audio signal derived from the new dataset. As shown in table 6.6, the obtained average PSNR value was 19.357, while the average MSE value was 0.04, which is comparable to the value obtained from the specialized GAN with SOLO testing samples. Moreover, the specialized GAN's value is still better than the general GAN in this experiment. Consequently, this shows that our model exhibits the ability to successfully complete audio signals that closely resemble those in the training dataset, despite their absence from the original training set. This is also confirmed by

the visualization results depicted in Fig. 6.10. That could be referred to as this type of distance measurement fits the agglomerative clustering algorithm as it follows the same determining; Although the cluster lookup technique we used finds the nearest neighboring example in the training set to identify the cluster membership of a testing sample, this would change depending on the clustering algorithm used. For example, in K-Means, the closest centroid would be identified instead.

Table 6.6. The average PSNR and the MSE for Maestro dataset.

GAN	PSNR	MSE
General-GAN (Maestro)	18.9165	0.143
Specialized-GAN (SOLO with lookup)	19.3451	0.0937
Specialized-GAN (Maestro with lookup)	19.357	0.04

6.4. Computation cost (RQ5)

While training the general GAN model, the first 300 epochs generated noise and only after the 800 epochs acceptable sounds were produced. Compared to the other specialized GANs this is more than the double number of epochs used to produce a convincing output. As can be seen in Fig. 6.11 up to 300 epochs and the model still did not produce a sensible output and even when we extended the number of epochs to double as shown in column C in Fig. 6.8 the general GAN model is not able to fill the spectrogram correctly which reflect a noise on the audio form, on the other hand, the specialised GAN filled the spectrogram correctly with 300 epochs as shown in Fig. 6.12. That is because the General GAN was trained on a wide range of sounds, which makes

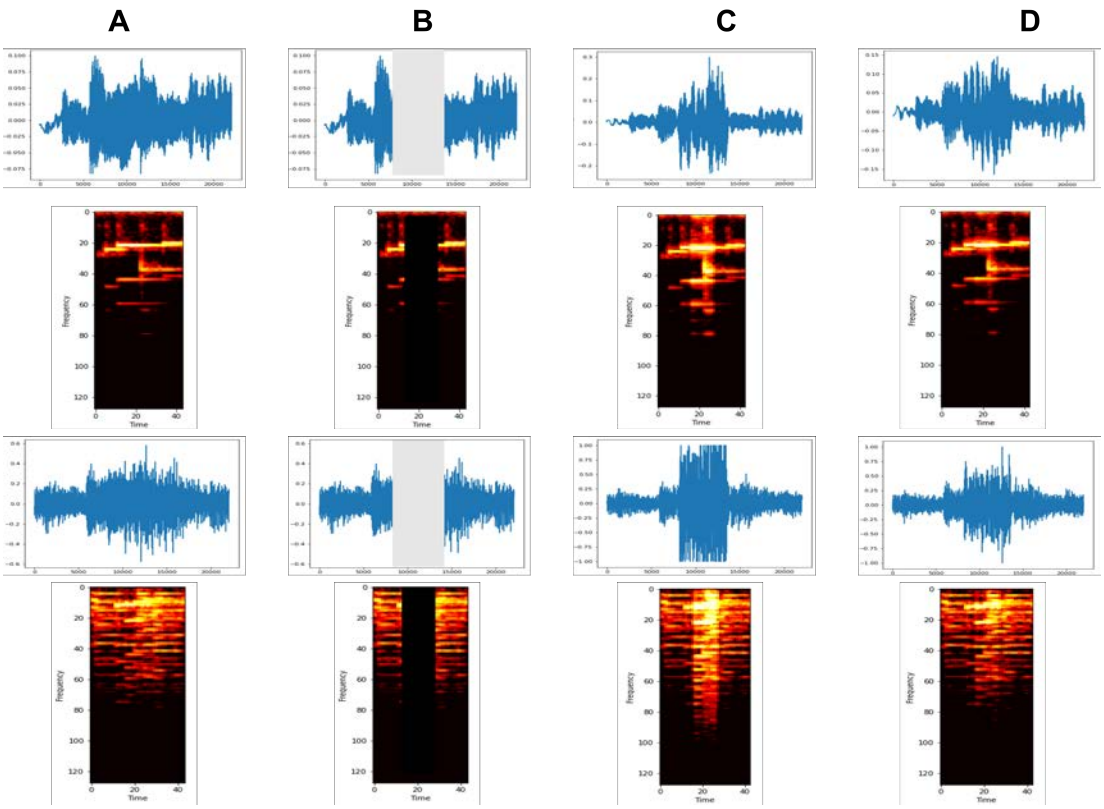


Figure 6.10. Reconstruction sample results for Maestro audio signals after the completion. (A) is the original audio, (B) masked audio, (C) audio generated by general GAN (D) audio generated by specialized GAN after cluster lookup.

the search space very big and makes it hard for the GAN to produce realistic sounds. Furthermore, as shown in Table 6.7, the specialized GAN requires less time to complete the training with acceptable output. It is worth noting that different clusters require various computation to complete the training, and some started providing good output with fewer epochs. To conclude, training the GAN on a mix of audio is an ineffective way for sound reconstruction.

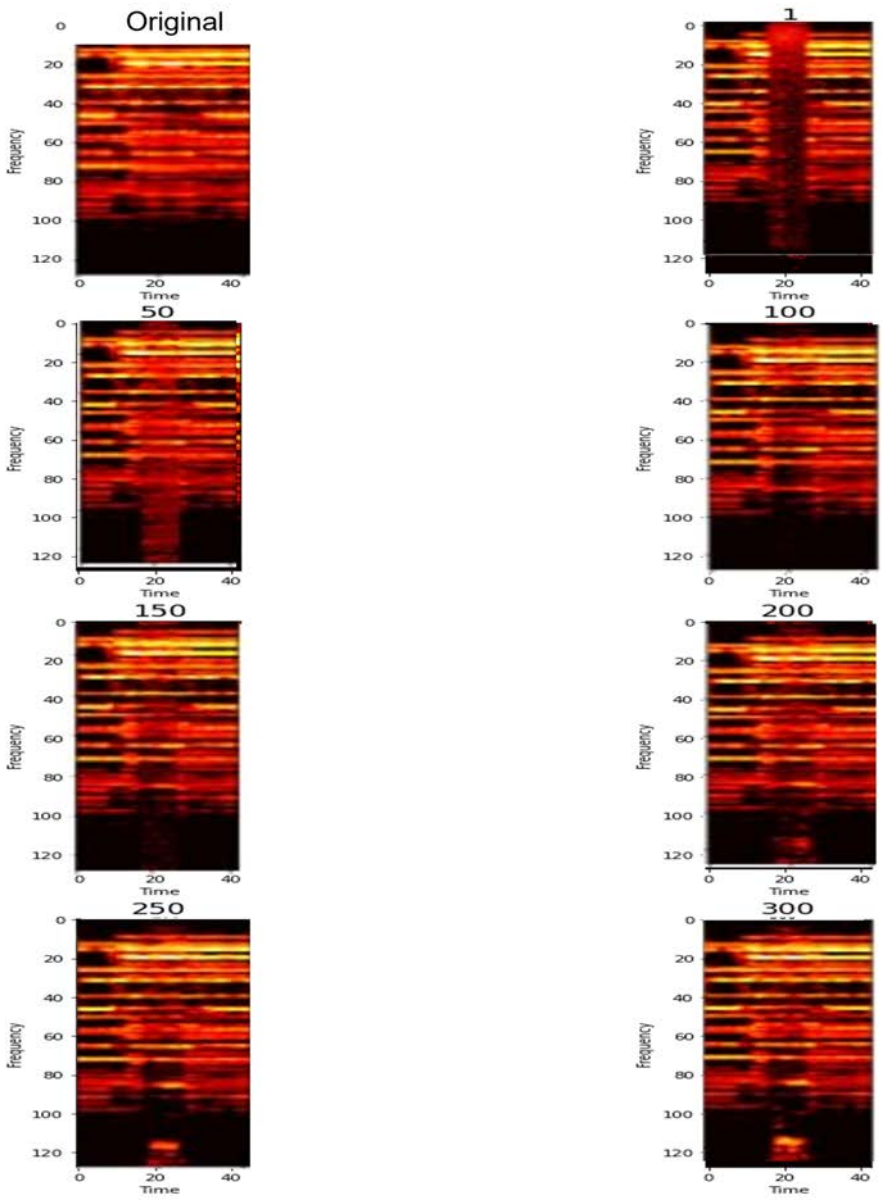


Figure 6.11. The generated spectrogram from the General GAN every 50 epochs up to 300 epochs.

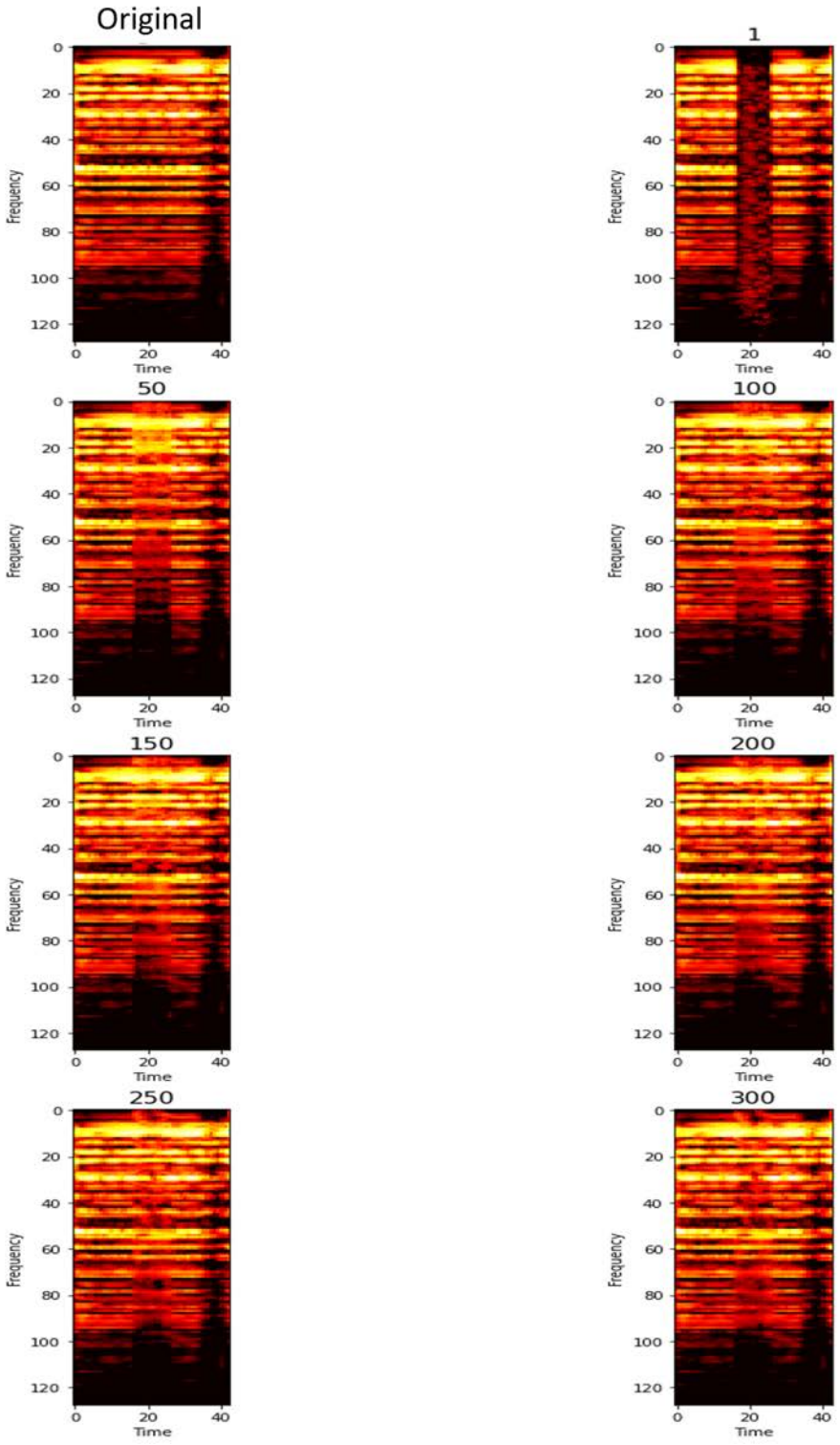


Figure 6.12. The generated spectrogram from the Specialised GAN every 50 epochs up to 300 epochs.

Table 6.7. Computation cost comparison between the general and specialized GANs.

GAN	Average training time	Number of epochs
General-GAN	34 hours	1000
Specialized-GAN (VGGish)	average 4 hours	300
Specialized-GAN (Yamnet)	average 5 hours	300

CHAPTER 7: CONCLUSIONS AND FUTURE WORK

The goal of this study was to develop a model for reconstructing audio that could fill in missing or corrupted audio segments whilst considering different audio contexts. Our model is built around generative adversarial networks, which are used to generate synthetic data. Experiments using a context encoder-decoder GAN show that training a general GAN on the entire dataset to reconstruct missing audio parts could be not persuasive, especially if the training dataset includes data from multiple audio contexts. Using our *divide-and-conquer* strategy to target each context individually leads to great improvements in the completion quality. Rather than train a general GAN with the whole dataset, the agglomerative clustering technique was used to partition the dataset into clusters with similar contexts to reduce the dataset complexity. In addition, the quality of the clusters depends on the features fed to the clustering model which later will affect the GAN performance. In terms of computational cost, specialized GANs were able to produce better results with less time and number of epochs.

Throughout the literature review, it was discovered that the majority of audio reconstruction research is focused on building models that are trained on large datasets or are limited to a single instrument. At the time of this writing, no previous work tested the effect of clustering audio segments, and analyzing the choice of clustering algorithm and feature extraction technique and, therefore, the GAN performance.

The proposed method has several drawbacks linked to the Context-Encoder GAN model training and reconstruction phases. These methods have high computational cost because they necessitate hyperparameters and architectural tweaks to enhance the quality and realism of the final outputs after several training and reconstruction repetitions. Additionally, because the final audio depends on perceptual judgment, GAN training

lack clear stopping conditions and standard evaluation metrics. It is worth noting that using some feature extraction methods with clustering may yield an insufficient number of samples to train the model to learn from and produce a realistic output, such as the case of Yamnet features.

For future work, we intend to test this approach on more complicated datasets, such as voice conversations, and study the effect of varying the number of clusters. Since this divide-and-conquer method is not limited to GANs, alternative approaches can be pursued such as the idea of using Transformers to generate audio [62].

Part of this thesis has been published in IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), IEEE, 2023

REFERENCES

[1] "Audio inpainting toolbox - inria - institut national de recherche en sciences et technologies du numérique." (2020), [Online]. Available: https://inria.hal.science/view/index/identifiant/hal-02963154.

[2] Y. Bahat, Y. Y. Schechner, and M. Elad, "Self-content-based audio inpainting," *Signal Processing*, vol. 111, pp. 61–72, 2015.

[3] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, pp. 1–14, 2017.

[4] Y. Feigin, H. Spitzer, and R. Giryes, "Cluster with gans," *Computer Vision and Image Understanding*, vol. 225, p. 103 571, 2022.

[5] N. H. Jboor, A. Belhi, A. K. Al-Ali, A. Bouras, and A. Jaoua, "Towards an inpainting framework for visual cultural heritage," in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, IEEE, 2019, pp. 602–607.

[6] A. Sable. "Introduction to audio analysis and processing." (2021), [Online]. Available: https://blog.paperspace.com/introduction-to-audio-analysis-and-synthesis.

[7] R. Pethiyagoda, S. W. McCue, and T. J. Moroney, "Spectrograms of ship wakes: Identifying linear and nonlinear wave signals," *Journal of Fluid Mechanics*, vol. 811, pp. 189–209, 2017.

[8] J. Shen, R. Pang, R. J. Weiss, *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE international conference*

on acoustics, speech and signal processing (ICASSP), IEEE, 2018, pp. 4779–4783.

[9] *Ai vs machine learning vs deep learning vs neural networks: Whats the difference?* [Online]. Available: `https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks`.

[10] F. Xiong, Z. Liu, K. Huang, X. Yang, and A. Hussain, "Primitives generation policy learning without catastrophic forgetting for robotic manipulation," in *2019 International Conference on Data Mining Workshops (ICDMW)*, IEEE, 2019, pp. 890–897.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[12] R. Kumar and S. K. Maji, "A novel framework for denoised high resolution generative adversarial network–dhrgan," in *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*, IEEE, 2020, pp. 1033–1038.

[13] J. Zakraoui, M. Saleh, S. Al-Maadeed, and J. M. Jaam, "Improving text-to-image generation with object layout guidance," *Multimedia Tools and Applications*, vol. 80, no. 18, pp. 27 423–27 443, 2021.

[14] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.

[15] A. Marafioti, P. Majdak, N. Holighaus, and N. Perraudin, "Gacela: A generative adversarial context encoder for long audio inpainting of music," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 1, pp. 120–131, 2020.

[16] H. Zhou, Z. Liu, X. Xu, P. Luo, and X. Wang, "Vision-infused deep audio inpainting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 283–292.

[17] H. Wang, L. Jiao, H. Wu, and R. Bie, "New inpainting algorithm based on simplified context encoders and multi-scale adversarial network," *Procedia computer science*, vol. 147, pp. 254–263, 2019.

[18] E. Stevens, L. Antiga, and T. Viehmann, *Deep learning with PyTorch*. Manning Publications, 2020.

[19] M. Maayah, A. Abunada, K. Al-Janahi, M. E. Ahmed, and J. Qadir, "Limitaccess: On-device tinyml based robust speech recognition and age classification," *Discover Artificial Intelligence*, vol. 3, no. 1, p. 8, 2023.

[20] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal processing letters*, vol. 24, no. 3, pp. 279–283, 2017.

[21] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 5484–5488.

[22] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, IEEE, vol. 2, 2004, pp. II–104.

[23] M. A. Nielsen, *Neural networks and deep learning*. Determination press San Francisco, CA, USA, 2015, vol. 25.

[24] I. H. Witten and E. Frank, "Data mining: Practical machine learning tools and techniques with java implementations," *Acm Sigmod Record*, vol. 31, no. 1, pp. 76–77, 2002.

[25] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 1177–1178.

[26] Y. Li, K. Zhao, X. Chu, and J. Liu, "Speeding up k-means algorithm by gpus," *Journal of Computer and System Sciences*, vol. 79, no. 2, pp. 216–229, 2013.

[27] F. Nielsen and F. Nielsen, "Hierarchical clustering," *Introduction to HPC with MPI for Data Science*, pp. 195–211, 2016.

[28] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5505–5514.

[29] J. Li, G. Song, and M. Zhang, "Occluded offline handwritten chinese character recognition using deep convolutional generative adversarial network and improved googlenet," *Neural Computing and Applications*, vol. 32, pp. 4805–4819, 2020.

[30] J. Dong, R. Yin, X. Sun, Q. Li, Y. Yang, and X. Qin, "Inpainting of remote sensing sst images with deep convolutional generative adversarial network," *IEEE geoscience and remote sensing letters*, vol. 16, no. 2, pp. 173–177, 2018.

[31] C. Wang, C. Xu, C. Wang, and D. Tao, "Perceptual adversarial networks for image-to-image transformation," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 4066–4079, 2018.

[32] Y.-G. Shin, M.-C. Sagong, Y.-J. Yeo, S.-W. Kim, and S.-J. Ko, "Pepsi++: Fast and lightweight network for image inpainting," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 252–265, 2020.

[33] P. Vitoria, J. Sintes, and C. Ballester, "Semantic image completion through an adversarial strategy," in *Computer Vision, Imaging and Computer Graphics Theory and Applications: 14th International Joint Conference, VISIGRAPP 2019, Prague, Czech Republic, February 25–27, 2019, Revised Selected Papers 14*, Springer International Publishing, 2020, pp. 520–542.

[34] S. Lou, Q. Fan, F. Chen, C. Wang, and J. Li, "Preliminary investigation on single remote sensing image inpainting through a modified gan," in *2018 10th IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS)*, IEEE, 2018, pp. 1–6.

[35] X. Zhang, X. Wang, C. Shi, *et al.*, "De-gan: Domain embedded gan for high quality face image inpainting," *Pattern Recognition*, vol. 124, p. 108 415, 2022.

[36] N. M. Salem, H. M. Mahdi, and H. Abbas, "Semantic image inpainting using self-learning encoder-decoder and adversarial loss," in *2018 13th International Conference on Computer Engineering and Systems (ICCES)*, IEEE, 2018, pp. 103–108.

[37] H. Liu, G. Lu, X. Bi, J. Yan, and W. Wang, "Image inpainting based on generative adversarial networks," in *2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, IEEE, 2018, pp. 373–378.

[38] X. Han, Z. Wu, W. Huang, M. R. Scott, and L. S. Davis, "Finet: Compatible and diverse fashion image inpainting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4481–4491.

[39] S. Mehri, K. Kumar, I. Gulrajani, *et al.*, "Sample rnn: An unconditional end-to-end neural audio generation model," *arXiv preprint arXiv:1612.07837*, 2016.

[40] A. v. d. Oord, S. Dieleman, H. Zen, *et al.*, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[41] A. Oord, Y. Li, I. Babuschkin, *et al.*, "Parallel wavenet: Fast high-fidelity speech synthesis," in *International conference on machine learning*, PMLR, 2018, pp. 3918–3926.

[42] J. Engel, C. Resnick, A. Roberts, *et al.*, "Neural audio synthesis of musical notes with wavenet autoencoders," in *International Conference on Machine Learning*, PMLR, 2017, pp. 1068–1077.

[43] A. Adler, V. Emiya, M. G. Jafari, M. Elad, R. Gribonval, and M. D. Plumbley, "Audio inpainting," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 3, pp. 922–932, 2012. DOI: 10.1109/TASL.2011.2168211.

[44] O. Mokrỳ, P. Záviška, P. Rajmic, and V. Veselỳ, "Introducing spain (sparse audio inpainter)," in *2019 27th European Signal Processing Conference (EUSIPCO)*, IEEE, 2019, pp. 1–5.

[45] O. Mokrỳ and P. Rajmic, "Approximal operator with application to audio inpainting," *Signal Processing*, vol. 179, p. 107 807, 2021.

[46] G. Tauböck, S. Rajbamshi, and P. Balazs, "Dictionary learning for sparse audio inpainting," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 1, pp. 104–119, 2020.

[47] P. P. Ebner and A. Eltelt, "Audio inpainting with generative adversarial network," *arXiv preprint arXiv:2003.07704*, 2020.

[48] A. Marafioti, N. Perraudin, N. Holighaus, and P. Majdak, "A context encoder for audio inpainting," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 12, pp. 2362–2372, 2019.

[49] Y.-L. Chang, K.-Y. Lee, P.-Y. Wu, H.-y. Lee, and W. Hsu, "Deep long audio inpainting," *arXiv preprint arXiv:1911.06476*, 2019.

[50] V. A. Embedding. "Opensource tools data for music source separation: A pragmatic guide for the mir practitioner." (2021), [Online]. Available: `https://github.com/tensorflow/models/tree/master/research/audioset/vggish`.

[51] E. Tsalera, A. Papadakis, and M. Samarakou, "Comparison of pre-trained cnns for audio classification using transfer learning," *Journal of Sensor and Actuator Networks*, vol. 10, no. 4, p. 72, 2021.

[52] N. Oskolkov. "How to cluster in high dimensions." (2019), [Online]. Available: `https://towardsdatascience.com/how-to-cluster-in-high-dimensions-4ef693bacc6`.

[53] B. Mcfee, L. Barrington, and G. Lanckriet, "Learning content similarity for music recommendation," *IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING*, vol. 20, p. 2207, 8 2012. DOI: `10.1109/TASL.`

2012.2199109. [Online]. Available: http://www.apple.com/pr/library/2008/09/09itunes.html.

[54] "Librosa.feature.inverse.meltoaudio librosa 0.10.1dev documentation." (), [Online]. Available: https://librosa.org/doc/main/generated/librosa.feature.inverse.mel_to_audio.html.

[55] ZHOUSL16. "Solo audio." (2022), [Online]. Available: https://www.kaggle.com/datasets/zhousl16/solo-audio.

[56] audacityteam. "Truncate silence." (2019), [Online]. Available: https://manual.audacityteam.org/man/truncatesilence.html.

[57] "The maestro dataset." (2018), [Online]. Available: https://magenta.tensorflow.org/datasets/maestro.

[58] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[59] E. Manilow. "Opensource tools data for music source separation: A pragmatic guide for the mir practitioner." (2020), [Online]. Available: https://source-separation.github.io/tutorial/basics/evaluation.html.

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

[61] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[62] Z. Borsos, M. Sharifi, D. Vincent, E. Kharitonov, N. Zeghidour, and M. Tagliasacchi, "Soundstorm: Efficient parallel audio generation," [Online]. Available: https://github.com/rishikksh20/SoundStorm-pytorch.

APPENDIX A: AUDIO COMPLETION QUALITY EVALUATION

Audio Completion Quality Evaluation

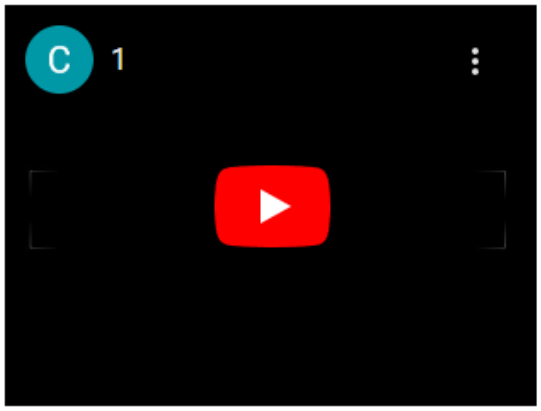
Please rate the quality of the Audio samples you hear from 1 to 5 where 1 represents poor quality and 5 represent good quality completion

mfmaayah@gmail.com (not shared) Switch account

* Required

Please compare Sound 1 and Sound 2 to the Ground Truth and rank them according to their quality for each example.
Please rate the quality of the Audio samples you hear from 1 to 5 where 1 represents poor quality and 5 represent good quality completion

Ground Truth



Sound 1

Quality 1 *

1	2	3	4	5
○	○	○	○	○

Sound 2