Data Augmentation for Cover Song Identification Systems Using Generative Adversarial Networks

Lateef Muizz Kolapo

School of Computing and Mathematics

University of South Wales

A submission presented in

partial fulfilment of the requirements

of the University of South Wales/Prifysgol De Cymru

for the degree of MSc Data Science

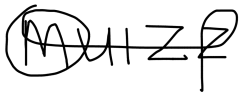
September 21, 2021

UNIVERSITY OF SOUTH WALES
PRIFYSGOL DE CYMRU

FACULTY OF COMPUTING, ENGINEERING & SCIENCE
SCHOOL OF COMPUTING & MATHEMATICS

STATEMENT OF ORIGINALITY

This is to certify that, except where specific reference is made, the work described in this project is the result of the investigation carried out by the student, and that neither this project nor any part of it has been presented, or is currently being submitted in candidature for any award other than in part for the degree of MSc Data Science of the University of South Wales.

Signed  ..... (student)

Date September 21, 2021 .....

Acknowledgments

To my parents, sisters, brother, and my uncle. Thank you for your encouragement and support. I will like to thank everyone at the University especially the Data Science team for their support on this research, especially Ieuan for his guidance and collaboration. I will like to express my appreciation to the team at Breathe Music Limited for sponsoring me on this research.

Abstract

Generative Adversarial Networks (GAN) have seen great improvement in modelling synthetic data with results unparalleled in the field of deep learning. This research explores data augmentation for cover song identification by extending state-of-the-art GAN frameworks used in image processing to cover song identification. Research in this domain is still in the early phases, this work explores the implementation of a GAN model capable of generating new features for cover song identification. The implementation utilises the Deep Convolutional Generative Adversarial Networks (DCGAN) to learn the underlying distribution of the sample dataset provided by Breathe Music, the conditional Generative Adversarial Networks (cGANs) was used for the encoding of the class to give control over which class in the distribution is generated. A control model was also created using an industry-standard dataset, CIFAR-100, to validate the approach. The qualitative and quantitative results were obtained for both two datasets, and a comparative analysis of both models was undertaken. The model results demonstrated a strong performance in capturing the underlying data distribution for the CIFAR100 dataset but performed poorly for the custom music dataset provided by Breathe Music using the statistical probability distribution plot, and Fréchet Inception Score (FID) metric.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction to Breathe Music

In recent years there has been a huge embrace of artificial intelligence (AI) technology by the Music Information Retrieval (MIR) community. AI systems have been routinely implemented for the different retrieval paradigms found in MIR [12]. The increase in high-speed computing power and modern machine learning tools has led to a complete change in approach by the community when solving problems in MIR.

Breathe Music Limited is a Welsh MIR company using AI systems to solve music identification problems, specifically solving the issue of content-based music recognition. There are different tasks that fall under this topic which includes, fingerprinting, cover song identification, genre recognition, music recommendation system, symbolic melodic similarity e.t.c. Breathe Music focuses on cover song identification, they aim to improve the royalties system by providing a more accurate system for determining when a cover is performed. This ensures that money from these renditions is more accurately re-distributed, better supporting local artists. The astonishing growth of music streaming platforms over the last decade has provided independent song cover artists opportunities to make money off cover songs on streaming platforms, now more than ever, it is imperative that a robust cover song identification system in place which helps original song creators receive the right amount of royalties on these cover songs created from their licensed original songs. A look into the work done at Breathe Music Limited showed some positive outcomes in this endeavour as they now have a production-ready implementation of this system in collaboration with the Centre of Excellence in Mobile and Emerging Technologies (CEMET) and the University of South Wales (USW). The majority of completed work focuses on synthesising feature representations for cover songs which covers the different variations of cover songs such as those from the live rendition of the songs, as in any Machine Learning task, proper data representing the different

classes improves the efficiency of any Machine learning system.

A common problem with training supervised classification tasks, such as cover song identification, is the problem of class imbalance due to the varying amount of covers made for a particular song. Finding the right class balance in the set of feature representation used are as important as finding the right model, which is also the same for cover song identification. Data augmentation is a very common practice in the field of machine learning to help solve the problem of class imbalance. This research proposed the use of GANs for the data augmentation procedure of the features used for the cover song identification system at Breathe Music to help generate features representing new cover song samples for specified songs from a random Gaussian noise distribution.

1.1.1 Music Information Retrieval paradigms

Music information retrieval systems at their core are utilised for extracting and inferring information from music. The retrieval of information is implemented in the form of feature extraction from the music (audio signal, symbolic representation, or other sources such as streaming platforms). Information retrieval from music generally follows the same paradigms as any other classical information retrieval system, the features extracted are used for indexing music, developing search and browsing systems(e.g. content-based search) and also used as recommendation systems. Classical Music Information retrieval system paradigms generally can be classified into 3 groups [29]:

- **Retrieval Systems** Retrieval systems generally are designed for cases where the user is in search of music information from a collection or database, they use faceted queries to specify which musical information is retrieved. The queries are executed on these systems initiating a search procedure and the best objects matching the user request are returned.

- **Browsing Systems** Browsing is an exploratory and iterative activity, the user has no specific information requirement but wants to explore the available music in the music Digital library.

- **Recommendation Systems** Recommendation systems are typically content-based systems, in this paradigm the user is searching for music information without any queries being executed. Recommendation systems are interactive, input is required from the user in the form of preferences which might be in form of user activity in the digital library, ad-hoc settings, or browsing history which are then presented to the user through some form of relevance ranking.

1.1.2 Music Representation

Musical information in computing systems is represented in various ways depending on the use case, there is three widely used musical representation which is; sheet music, symbolic, and audio representations [39].

- **Sheet Music Representation** Sheet music uses the visual description of musical information. Sheet music is a graphical and textual encoding of the musical information with parameters such as notes (onsets, pitches, duration), tempo, measure, dynamics and instrumentation. This type of musical representation generally has subjective interpretations.

- **Symbolic Representation** This content-aware form of musical information representation is based on entities with the explicit musical meaning indicated in a digital format and can be parsed by a technical system.

- **Audio Representation** This form of representation is derived either through acoustic sound waves or synthesising audio. This representation has parameters reflecting different aspects of the music such as; waveform (pressure-time plot), Sound, pitch, dynamics, timbre, loudness, e.t.c.

1.1.3 Cover Song Identification

Cover Song Identification (CSI) is a type of Music Information Retrieval task which involves searching large databases of music or music metadata and identifying pieces that share similarities in robust features such as melody. The goal is to identify recordings that are alternative; versions, performances, renditions, or recordings of the query musical piece [6][29][47][57]. When given a query piece of music, an efficient CSI system is able to identify versions of that music from the database which share similar musical information or structure[18]. State of the art techniques for cover song identification extract robust features relating to melody, and harmonic progression to measure the similarity between two songs [51].

Cover song identification tasks are generally categorised by three different stages:

1. Musical information feature generation using robust features(e.g. chroma features).

2. Similarity measurement algorithms (e.g. Dynamic Time Warping).

3. Evaluation of results(e.g. Mean average Precision).

Information retrieval requires a feature representation task, computing systems require specific information representation to be able to perform any form of computation on the data. These computations are generally in the form of implementing algorithms such as dynamic-time warping, edit distances, cosine similarity, string-matching and common similarity measures to evaluate the musical piece. Feature fusion involves the use of composite features which combines multiple robust features into a single feature as seen in the state-of-the-artwork of Christopher J. Tralie [51].

1.1.4 Audio Feature Representation and Pre-processing

Cover Song Identification like any other tasks using intelligent systems require features, these features represent the audio signal which contains the desired musical information. The feature level representation can vary, human-readable features (e.g. note, melody, etc) are high-level features. Machine-readable feature representations are referred to as low-level features (acoustic)Figure 1.2. Feature representation and extraction creates a level of abstraction which generally leads to a loss of some information in the data which is referred to as the semantic gap found in most information retrieval tasks [29].

Several low-level features are employed in the task of cover song identification, the most popular features employed in CSI are chroma based music features which describe the musical information into 12 different classes, and can capture some of the underlying properties of the music relating to harmony and melody through the use of Short-Term Fourier Transform Features (STFT). Other features employed in the field of CSI involves the use of Constant-Q Transform (CQT) features and Mel-Frequency Cepstral Coefficients (MFCC) features.

- **Short-Term Fourier Transform Features (STFT)**: The STFT is used for deriving various musical features from audio signals. It uses a short-time frame of the signal for the time-frequency analysis and is used for representing the timbral texture of the audio [20][35].

- **Mel-Frequency Cepstral Coefficients (MFCC)**: This is a spectral based feature used in the task of CSI, the feature is obtained through the windowing of each frame of the audio signal, applying the discrete Fourier transform (DFT), the logarithm is taken and then transformed on the Mel-frequency scale, finally the inverse discrete cosine transform (DCT) is then applied [35].

- **Constant Q Transform (CQT) Based Features**: These are set of features derived from the transformation of audio signals from the time-domain to the frequency domain using a logarithmic frequency axis. CQT features have a specific range of values of

bins(12–96) per semitone from which the chroma features can be extracted [8][46].

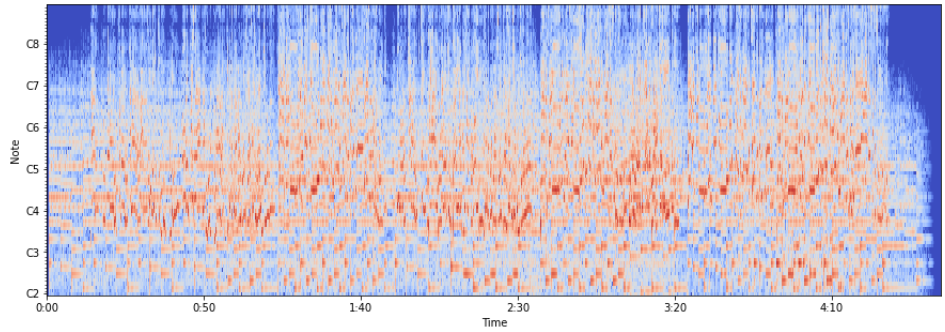


Figure 1.1: Constant-Q feature of Vince Staples - ARE YOU WITH THAT?

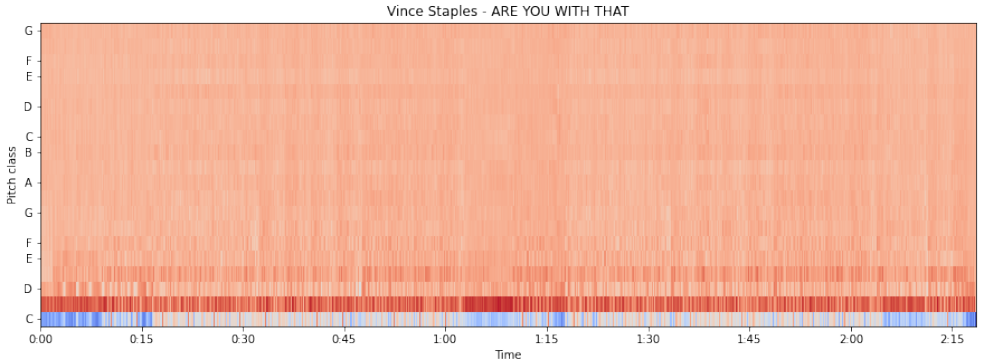


Figure 1.2: MFCC feature of Vince Staples - ARE YOU WITH THAT?

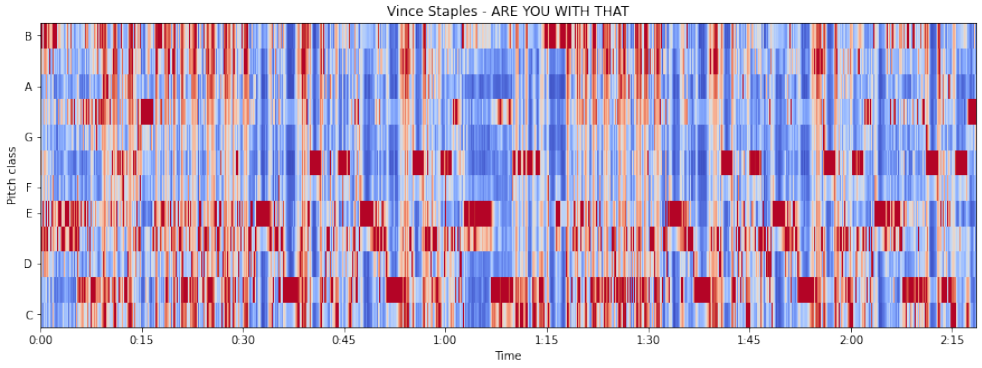


Figure 1.3: Chroma-STFT feature of Vince Staples - ARE YOU WITH THAT?

1.1.5 Data Augmentation

Data Augmentation techniques are very common in machine learning as they are used to improve the general performance of classifiers. There are different approaches to data augmentation, traditional transformations, and using deep learning methods such as generative models [43] are two methods used for this task. The traditional transformations include the duplication of the current samples in the data by; flipping, rotating, up-scaling, cropping, adding Gaussian noise e.t.c. The deep learning approach is an advanced approach to data augmentation involving the use of generative models which learn the set of parameters required to generate a feature representation of the data. There are different types of generative models, such as the Variational Autoencoders (VAEs) [13] and the Generative Adversarial Networks[21] Both methods try to model the probability distribution of the features in the underlying data and find their representation in a latent space.

1.1.6 Deep generative models

Deep generative models are probabilistic models used for modelling the probability of observing a set of features X from a distribution, they capture the joint probability $P(X|Y)$, or $P(X)$ if there are no labels. Deep generative models have seen a surge in interest over the past few years due to the introduction of state-of-the-art deep learning architectures [13][21][44] and computational advancement.

Variational Autoencoders include two neural networks, the encoder and the decoder. The VAEs learn by inserting real samples through the encoder, the encoder then finds a representation of the samples in a latent space and then samples are taken from the latent space in a bid to reconstruct the original samples as close to the input data as possible using the decoder. After training, new samples can be generated by removing the encoder and selecting random points from this latent space and passing them into the decoder network to recreate realistic samples of the data. To ensure different points from the latent space distribution are sampled, noise is injected into the model which helps the encoder to encode the sample data into a probability distribution that ensures generation of diverse samples [13].

Generative Adversarial Networks are a type of generative model which learns the underlying latent structure of a feature in order to generate new samples from the same distribution. This is achieved by implementing a competitive network that involves a Generative network and Discriminative network which are in a min-max contest [21]. In this research, Generative Adversarial Network will be explored in an attempt to generate realistic data that can then be used to augment the original dataset to further develop the CSI system at Breathe Music.

1.2 Research Aim and Objectives

The main aim of this research is to design and develop a parametric cover song synthesis system using GANs, which means generating cover songs given some user determined parametric inputs such as a copy of the original song, resulting in improved data augmentation capabilities that increase the size of the original dataset, and thus improve performance of the cover song identification algorithm.

As stated earlier, a central challenge in cover song identification system is the problem of class imbalance, the system needs to be able to synthesise cover songs for original songs immediately they are added to the identification system. Deep learning generative models offer a control principle for synthesising data, the model generates new data for specified classes using class conditioning. This is an interesting approach to augmenting the data, addressing the class imbalance problem by creating new data samples that span the different variations and use cases when building the CSI system. Due to the varying conditions under which cover songs are recorded, there is a potential for class imbalance in a cover song identification system. Hence, there is a need for a more adaptable system that creates variations of cover songs that covers the different scenarios under which they are created in real time. This new approach to synthesising cover songs utilising deep learning paradigms offers an opportunity to solve this problem.

Deep learning offers a practical solution for building highly efficient cover song systems. There is already a widespread use case of this method in cover song identification as seen in the work from Yu et al. [56]. Deep learning methods are able to extract useful features and also model semantic variations into them which is a useful tool for MIR tasks such as cover song identification which more often than not are impacted by the semantic gap which varies from tempo, different accompaniments and reconditioning in another genre.

The aim and objective of this research is to explore a GAN architecture for augmenting audio data for Breathe Music Cover Song Identification system. The specific aims are:

1. Research the field of Music Information Retrieval to understand the different features used for state of the art CSI system and find a way to integrate these features into the GAN network.

2. Review current GAN and Deep Learning architectures and explore the possibilities and limitations of using GANs for the augmentation of cover song data.

3. Use PyTorch [41] to develop a GAN network that could be integrated into the existing CSI pipeline of the business, a successful implementation will be incorporated into the current CSI pipeline.

4. Compare the generated samples to original samples using appropriate evaluation metrics, to determine the model's success.

1.3 Research Structure

This research will include a systematic review of the literature and state-of-the-art techniques used in the field of music information retrieval. The focus will be more on understanding the feature representation used for the different machine learning and deep learning techniques explored in the various literature reviewed. This research aims to use explore data augmentation of cover song identification using the different state-of-the-art architectural designs of GANs. The report will then explore the methodological paradigms used in generative adversarial networks which could be integrated into the current pipeline at Breathe Music. The research will include a comparative analysis of the GAN used to train the CQT data generated using the pre-processing techniques at Breathe Music and evaluating the performance of the same architecture on a public image dataset CIFAR100 [30]. A critique of the approach and the evaluation method will be summarised to give an insight into the performance of the network. The nature of this approach to augmenting the cover song dataset will be discussed and an outline of the techniques will be provided. The results of the research conducted in this work will be displayed, further work and approaches that can be used to improve performance of the proposed model will be recommended and a conclusion on this body of work will be provided to end the report.

Chapter 2

Literature Review

2.1 Review of Audio Data Augmentation and Generation using Generative Models

A common problem in deep learning tasks is having an insufficient amount of data catering to diversity and variations required to train a high-performance network. In the training of a neural network, it is sometimes observed that the dataset has issues with class distribution, creating a class bias in the network. As stated earlier, generative models are a type of neural network in deep learning that aim to capture the underlying distribution of a dataset and use this distribution to create new samples from the latent space representing the training features. This section of the literature review identifies publications that use GANs for data augmentation and generation in the field of audio analysis. Generative audio models utilise deep learning techniques to synthesis raw audio or create a new representation of audio (i.e. spectrograms) [14]. This type of method generally involves synthesising audio from a probability distribution learnt under parametric control of the generative models [24]. There are different approaches to generative audio models, one such solution bootstraps algorithms from image generating GANs for audio, which operates on the image representation of the audio in terms of spectrograms [57]. There are other techniques that take advantage of autoregressive neural networks to output raw audio [52]. Furthermore, recent deep learning methods are providing new techniques for advancing generative audio models.

2.1.1 Literature review

- *GAN-based Augmentation for Populating Speech Dataset with High Fidelity Synthesized Audio* [5]. This study covers research done for generating high fidelity audio data samples to augment speech datasets for deep learning, specifically this study used Harmonic

Percussive Source Separation (HPSS) to extract spectral features and then applies a progressively-growing GAN [27] to generate high fidelity audio samples. The HPSS was used to decompose a given audio signal into a sum of two component signals, these two components account for the harmonic sounds (pitched instruments) and the percussive sounds (percussion). Median filtering was used to decompose the spectrograms into a harmonic spectrogram and a percussive spectrogram. Binary masking was used to ensure the sum of the two decomposed spectrograms equals the original spectrogram by assigning the time-frequency bins of the original spectrogram to either the harmonic or the percussive spectrogram. The progressive growing GAN was used to learn the distribution of the training set from the harmonic spectrogram and a percussive spectrogram which was then used to generate new data. For evaluation, the authors used the Frechet Inception Distance (FID), a metric for image fidelity. The FID was adapted to measuring the fidelity of audio generated from their method which showed a lower score than existing image-generating GANs, the lower FID indicates better fidelity.

- *Data Augmentation using GANs for Speech Emotion Recognition* [11]. The authors present a GAN-based data augmentation approach to tackle the problem of class imbalance in the Speech Emotion Recognition (SER) dataset. The Mel-scaled spectrograms were extracted from the dataset using; a short-term window of 50 milliseconds with a 50 overlap ratio, a Mel coefficient of 128, and applying a logarithmic scale after the frequency power calculation. A custom GAN architecture based on the Balancing GAN (BAGAN) [37] which was initially proposed to tackle the imbalance problems in image classification tasks was utilised in this study. The study also employs other signal-based data augmentation methods which include; methods such as the sample copying (CP), Signal-based Audio Augmentation (SA), SA with replacement (SAR), SA with replacement of the majority class only (SARM), SAR adding only Background Noise (SARB), SAR using only time stretch (TS) and pitch shift (PS) (SAR_S). The evaluation of the results was conducted through a comparative analysis between the GAN based method and other signal-based augmentation techniques, The authors used the IEMOCAP (interactive emotional dyadic motion capture database) [9] and FEEL-25k dataset. The results showed that when augmenting the minority classes using the GAN method on the IEMOCAP & FEEL-25k dataset, there was a tangible increase in performance. IEMOCAP dataset had a 10% improvement in performance and the FEEL-25k had a 5% improvement in performance.

- *WAVENET: A Generative Model For Raw Audio*[52]. The authors explored the application of neural autoregressive generative models based on the PixelRNN for modelling

an audio generative model. They proposed a model which formed a generic framework for audio generation tasks. This study introduced a generative model that models the joint probability of a waveform $\vec{\mathbf{x}} = \{x_1, ..., x_T\}$ which is factorised as a product of conditional probabilities modelled by a stack of convolutional layers as expressed below:

$$p(\vec{\mathbf{x}}) = \prod_{T}^{t=1} p(x_t | x_1, ..., x_{t-1})$$

Each audio sample x_t is dependent on the samples at all previous timesteps. They used causal convolutions to ensure the model does not violate the ordering of the modelled data. The prediction from the model at a timestep t cannot depend on any future timesteps. The performance of this method was evaluated on three tasks; multi-speaker speech generation (not conditioned on text), text-to-speech (TTS), and music audio modelling. They successfully showed that their model could generate novel and realistic musical fragments and successfully evaluated the model as a discriminative model, returning promising results for phoneme recognition.

- *Adversarial Audio Synthesis WaveGAN*[14]. This study employs GANs for unsupervised generation of audio in the time domain. The WaveGAN architecture is based on the two-dimensional deep convolutional GAN (DCGAN) [44]. The DCGAN generator was modified to widen its receptive field. A longer one-dimensional filter of length 25 was used, and the stride was increased from 2 to 4. Similar changes were made to the discriminator, the WaveGAN had the same number of; parameters, numerical operations and output dimensionality as the DCGAN. An extra layer was then added to increase the sample generated from 4096 audio samples to 16384 samples. Batch normalization was removed from the generator and the discriminator and training were done using the Wasserstein Gradient Penalty Loss (WGAN-GP Loss)[22]. The Phase shuffle operation was implemented to prevent the discriminator from learning a simple solution for discriminating between classes.

- *Adversarial Audio Synthesis SpecGAN*[14]. The authors who discussed WaveGAN [14], also proposed a SpecGAN model based on the WaveGAN architecture. The SpecGAN uses Generative adversarial network strategies for generating spectrogram-based and frequency-domain audio data. To pre-process the audio data the authors used a short-time Fourier transform (STFT) with 16 ms windows and 8ms stride which resulted in a 128 frequency bin linearly spaced between 0 to 8 khz. The frequency bin were normalized to zero mean and unit variance, To generate bounded spectrograms, the spectra was clipped at 3 standard deviations and then rescaled to a range

of [1,1]. The DCGAN was implemented on the resulting spectrograms. The generated spectrograms were then rendered as waveforms by inverting the pre-processing steps described earlier. The Griffin-Lim algorithm [36] parametrised by 16 iterations was then used to estimate phase and produce 16384 audio samples. Both methods were evaluated using different datasets which are; Speech Commands dataset [53], the Speech Commands Zero Through Nine(SC09) dataset, Drum sound effects(0.7hours), Bird vocalizations(12.2hours) by Peter Boseman [2], Piano(0.3hours) and Large vocab speech (TIMIT)(2.4hours) [19].The performance of the SpecGAN was compared to the WaveGAN method with respective accuracy of 58% and 66%.

2.1.2 Literature Summary

A review of published studies in the field of generative models for audio shows that these models can be used as an approach for data augmentation of CSI systems. One of the biggest hindrances to the application of deep learning to CSI systems is the availability of enough varied data for a particular song. This approach offers promising results for audio data synthesis as the different literature shows that generative models can help generate audio samples from random noise in a distribution. This offers the advantage of creating synthetic data which can help train the CNNs better for the song identification tasks. The most popular generative model are GANs which use two neural networks, discriminator and generator, which are in a min-max competition to find the proper distribution from which audio samples can be created, In this approach, the image generation method is leveraged into audio systems by treating audio as image and attempting to generate new audio samples with GANs. This method generally offers a scalable, efficient and practical method of augmenting data for CSI systems using deep learning.

2.2 Review of Cover Song Identification Techniques

In this section of the literature review; the concepts, theories and technical details from published research in the field of Cover Song Identification will be introduced. Based on the different published research explored in the literature review, CSI methods will be categorised into three approaches, which are; cross-similarity and sequence based approaches, index-based methods and Deep Learning methods. In the following below; subsection 2.2.1 identifies the evaluation metrics and datasets used in published research in the field of MIR, subsection 2.2.2 explores Cross Similarity and Sequence based approaches, subsection 2.2.3 is based on literature for the index-based methods, and in subsection 2.2.4 Deep Learning methods in

CSI is reviewed.

2.2.1 A Review of evaluation techniques and Datasets

Music Information Retrieval has a convention called Music Information Retrieval Evaluation Exchange (MIREX), this convention is generally responsible for evaluating the criteria for measuring performance of the different techniques in the MIR paradigm. The large majority of the literature presented in this research evaluates their performances based on criteria from MIREX. The criteria for performance from MIREX is often measured using the following [3]:

- Mean average precision (MAP).
- Total number of covers identified in top 10, precision at 10 (P@10).
- Mean rank of first correctly identified cover(MR1) (smaller values are better)

A review of the literature shows that different datasets are used in the MIR domain, the most popular datasets identified in the literature were reviewed below.

- *DA-TACOS* [55]: This is a dataset in the study by [54], they extracted this dataset from the publicly available API of the secondhandsongs.com. Different features were extracted and the techniques used by the author are publicly available on GitHub.
- *Covers80* [17]: Created by D. P. W. Ellis, it contains 80 pairs of pop music cover versions used for the purpose of Cover song identification systems.
- *SHS dataset* [7]: A subset of the Million Song Dataset (MSD), it has a total of 18,196 tracks from the MSD. They are organised into groups of versions of a single underlying work. The clips are around 30 seconds only.
- *Chopin Mazurkas* [1]: Open-source dataset used as part of the MIREX task. It contains 367 Chopin Mazurkas, represented as full audio tracks (WAV format). The dataset also contains tempo changes for beat tracking algorithms.
- *YoutubeCovers dataset* [48]: This dataset contains 50 cliques with 7 songs for each clique striped from Youtube videos. It contains pre-computed Chroma, CENS and Chroma DCT-Reduced log Pitch (CRP) features.

2.2.2 Cross Similarity Sequential Based Methods

The techniques in this category generally use robust mid-level features [29] to represent the underlying songs which offers some level of abstraction for the details about a particular song. After getting a mid level representation of the songs, the next step is generally to align them for optimal matching probability, a form of similarity measures is then used to measure the difference between these two feature vectors (original and cover), similarity measures such as; the Hamming distance, the Euclidean interval vector distance, the interval difference vector distance, the swap distance, and the chronotonic distance [10]. Some studies also employ sequence alignment techniques, such as Dynamic Time Warping (DTW) or the Smith-Waterman algorithm [18].

- *Identifying 'cover songs' with chroma features and dynamic programming beat tracking* [18]. Ellis and Poliner proposed a method utilising beat-synchronous chroma representation of audio recordings. In this study, beat tracking was used to overcome variation in the tempo of the pieces. A feature vector representing a single beat was created by normalizing the tempo since the cover and original piece have been abstracted to the same number of beats per phrase. The beat is a normalized chroma vector, the chroma feature vector creates a feature matrix that helps overcome the variation in instrumentation. A combination of these two features creates the beat-by-chroma representation of a song. A comparison is conducted by cross-correlating the beat-by-chroma representation of the two pieces and looking for sharp peaks which indicates good local alignment between the pieces. This is because if both songs follow the same beat-by-chroma they will have similar shapes, the shapes may not be the same but their peaks and lows should be similar. The system developed in this study was able to identify 761 cover songs and had a Mean Reciprocal Rank (MRR) of 0.49. The next best performing system identified 365 covers with an MRR of 0.22. Significance testing confirmed that the system was superior to the others [15].

- *SiMPle: Assessing music similarity using subsequences joins*[49] Silva et al. proposed the use of similarity joins, a data processing operation that retrieves data point pairs from two different datasets which are considered similar at a certain threshold [42]. An algorithm based on the Fast Nearest Neighbour Search Algorithm [40] was used to compute the similarity matrix profile, which is as an efficient alternative for Dynamic Time Warping an algorithm. This allows a space efficient representation of the similarity join matrix between sub-sequences. The profile vector representing the distances between a sub-sequence in song A and B is calculated using the Mueen's Algorithm [40]. This research exploits the use of global and local pattern matching through the similarity

matrix profile calculated from sub-sequences of the audio recordings to measure the distance between two audio recordings, the median value was used as the measure for global distance estimation between the two songs.

$$dist(A, B) = median(SiMPle(B, A))$$

This approach was evaluated using the YouTube Covers database and the collection of Chopin's Mazurkas. The performance was measured using mean average precision (MAP), precision at 10 (P@10), and mean rank of the first correctly identified cover (MR1)(smaller values are better). The method was approximately 130 times faster than the Serra et al method described above and gave a better result across the measured statistics. The method in this paper was also compared with Dynamic Time Warping and similar performance was observed although the method proposed was approximately two times faster. This method also provides applications in Streaming Cover Song Recognition, Audio Thumbnailing, Motifs and Discords.

- *Summarizing and comparing music data and its application on cover song identification*[50]. Silva et. al. proposed a cover song identification system based on summarised audio snippets acquired using the SIMPLE method proposed by Silva et al [49] to summarise the audio snippets into thumbnails or repeated section of the song sequence to improve the memory usage and reduce the time necessary to assess the similarity. ThesSummarizing and comparing methods proposed in this study is comprised of two main steps, summarizing the reference recording using the SIMPLE method, and comparing the distance between the queried and reference audio using MASS algorithm [40]. Figure 2.1 shows the pipeline for the implementation of this approach. The evaluation of this method used the YouTubeCovers and Mazurkas dataset, the algorithm was evaluated on both Runtime Performance and Retrieval Performance. The approach was compared with the SIMPLE algorithm and SuCo was observed to have a runtime performance 30 times faster than SIMPLE, The retrieval accuracy was compared using the mean average precision (MAP), precision at 10 (P@10), and mean rank of the first correctly identified cover (MR1). SuCo is able to attain an accuracy very close to the best performing method while providing much higher performance, with a much lower runtime.

Literature Summary

A review of the Cross Similarity Sequential Based Methods shows that they are applied in the CSI domain in a way that requires a solution that transforms the songs into a mid-level

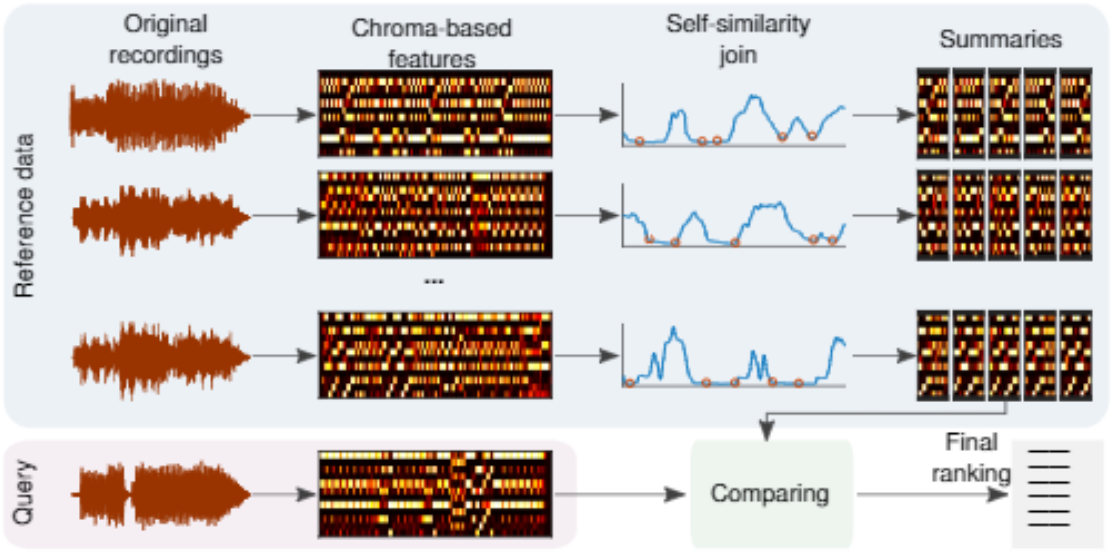


Figure 2.1: Pipeline for the SUCO Method illustrating the entire process [50].

feature representation, mostly Chroma based features. Most of the applications in this section use these Chroma variant features in some way as a representation of the song which offers a robust system with invariance features for CSI. This review showed that the performance of the CSI systems varies based on the set of features used as a representation for the songs. The methods in this section involve the design of features which are chroma variants such as; the Harmonic Pitch Class Profile (HPCP), the beat-by-chroma representation of a song, the cross recurrence plots (CRP) e.t.c. An analysis is then performed using some algorithm measuring similarity, this generally forms a similarity matrix and a distance based metric is used to discriminate between the different similarity matrix which is typically ranked on the similarity likelihood. These methods involving the quantification of cross similarity matrix (CSM) produces a more accurate CSI, although they require a high amount of computing resource due to the feature extraction process required, which generally makes it an inefficient approach for production ready cover song identification systems.

2.2.3 Index-based Methods

Using the cross similarity sequential methods requires the use of the entire feature vector extracted from the raw audio recording of a particular song. This type of approach generally requires a quadratic comparison space for each comparison between the feature vectors which leads to the creation of CSM which is computationally expensive to process [49]. Index methods hope to use further an abstracted representation of the features and high-level summary of songs to offer efficient CSI, this requires a trade-off between loss in semantic

meaning and getting a low-level representation of the audio [29]. Generally, the task is framed by extracting an index or feature that summarises the full song, which can be stored. New queries are then compared to the stored indices of all songs, often in a simple fashion e.g. using Euclidean or cosine distances.

- *Large-scale cover song recognition using the 2D Fourier transform magnitude*[6]. Bertin-Mahieux and Daniel P.W. Ellis in their study propose an efficient and scalable approach to cover song identification by using dimensionality reduction techniques to create low-level features of audio recordings. The feature representation of audio recordings were acquired using the beat-synchronous chroma feature representation which was then transformed by taking the two dimensional Fourier transform (2DFTM) which is a common task in digital image processing used for removing noise and compression. The median as opposed to average was obtained from the 2DFTM representation of the audio recordings. Principal Component Analysis(PCA) was used for the creation of the domain for comparison between the audio representations to improve generalisation and enable summarising to be done by simple summary statistics. When two PCA vectors lie within the same feature space, they are likely song covers. Reducing the dimension of the song to a 50-dimensional vector per song, the proposed method was observed to provide a result approximately 2,500 times faster than the Dynamic Time Warping methods.

- *Large-scale cover song identification using chord profiles*[28]. Khadkevich and Omologo in their article proposed the use of low-level features for CSI, through a high-level musical summary of the chord progressions and chord profiles for indexing and fast retrieval in cover song identification. The chord progression used in this study was extracted from audio or from chroma features provided in the Million Song Dataset (MSD) [7]. The chord progression from the songs is extracted which is further compressed into chord profiles. Chord profiles are 24-dimensional vectors, with each dimension corresponding to the rate of occurrence of a chord in a song that is invariant from one cover to another. The song retrieval method used in this study is a two step process. Locality Sensitive Hashing was used to retrieve the nearest neighbours using the L1 distance between two chord profiles a and b.

$$||a - b||_i = \sum_{i=1}^{24} |a_i - b_i|$$

The second step involves the use of the top k results from the first step which are progressively ranked using edit or Levenshtein distances. The results obtained showed

an improvement on the 2DFT method of Bertin-Mahieux [6] in terms of speed and average runtime per query without any parallelisation.

- *Data driven and discriminative projections for large-scale cover song identification*[23]. Humphrey, Nieto and Bello used an approach similar to that of Bertin-Mahieux and Daniel P.W. Ellis for CSI [6]. They proposed a modified feed-forward architecture using a sparse high-dimensional data-driven component estimation for improving the separability of the audio tracks and using a supervised dimensionality reduction to recover a representation of the similarity space. The data pre-processing stage involved the application of a three step process to the 2D-Fourier Magnitude Coefficient (2D-FMCs) representation of the audio tracks derived from chromagrams with the first two steps represented by:

$$\hat{X} = log\left(\frac{CX}{\|X\|_2} + 1\right)$$

C is a constant hyperparameter, X is a 2D-FMC vector, and $\|X\|_2$ is the application of L2-norm and then log-scaling is applied to scale all coefficients independently. The dimensionality reduction technique PCA was applied to reduce the dimensionality from 900 to 450 coefficients. The 2DFT features of a song are projected onto the dictionary to form an embedding of the song by summing the resulting vectors for each 2DFT feature. The Linear Discriminant Analysis was used for organizing the similarity space of the embeddings to better encode semantic relationships with distance measures. This approach resulted in a state-of-the-art performance with respect to previously used evaluation metrics (MAP), the million song dataset was used for the evaluation which shows a significant improvement in performance and the importance of PCA in alleviating the problem of over-fitting.

Literature Summary

A review of the index-based approach shows that this is also a popular technique for CSI systems, although the review of this approach shows the CSMs systems offer better accuracy for cover song identification, the index-based approach generally offers a faster design for a CSI system which require less computational resources. The methods in this section generally transform the song into a low-level representation which makes it faster to implement the CSI system. Generally, the task is framed by extracting an index or feature that summarises the full song, which can be stored. The methods in this section generally use dimensionality reduction techniques such as PCA, signal processing transformations such as 2DFTM. The

features in this category could be of benefit to training a GAN due to the reduced dimension and summarised information they offer which means the size of the GAN network will be reduced and model training can be done more efficiently. There is a downside to this approach which is that the important features in the latent space required by the GAN might be in the low variance components thereby leading to poor performance of the model. These methods generally offer a faster and scalable CSI system which is one that will be of advantage to a company like Breathe Music with the goal of producing production ready CSI systems, although like most tasks in machine learning a decision on the trade-off for accuracy would need to be deeply considered.

2.2.4 Deep Learning Based Methods

Recently Deep learning has gain popularity mostly due to its success in applications such as the AlexNet [31]. Deep learning techniques and paradigms have been embraced by the MIR community and have seen applications for different MIR tasks including Cover song identification [32][57]. These methods generally require using features such as CQT features as input, and the network returns a classifier output. This method basically converts an audio identification task into an image classification task with the CQT features representing the input image in form of spectrograms which is then converted into vectors of its pixels.

- *Cover Song Identification Using Song-to-Song Cross-Similarity Matrix with Convolutional Neural Network*[32]. The authors proposed a new method for CSI that uses a convolutional neural network along with a cross similarity matrix for the comparison of sub-melody sequences for the cover/non-cover songs. They compared different network sizes and designs and also used different similarity measures, such as melody lines derived from the sections of the two songs in with cover relationship. Three different convolutional neural network architectures were trained to output a vector [1,0] when the two audio recordings are similar and [0,1] when they are not similar. The training set contains 1,175 songs with 2,113 cover pairs inside, and all other combinations are non-cover. The test dataset consists of 1,000 songs. The proposed algorithm performed significantly better than the baseline method which was the SIMPLE method [49].

- *Learning a Representation for Cover Song Identification Using Convolutional Neural Network*[57] Yu et al. proposed the use of a convolutional neural network for overcoming the challenge of property variation between different versions of the same audio recording, differences such as; key transposition, speed change and structural variations. The

CQT of the audio recordings were extracted and then resampled and downsampled in the time direction. This approach employed a specialized kernel size which is a novel technique in the field of music information retrieval. The model was observed to be faster than the control method which was the dynamic programming method, although the time taken for feature extraction was not included in the time summary. This research also introduced dilation convolution and data augmentation to simulate tempo changes on the training data. The network was trained for classification and then used for cover song identification which is robust to tempo changes. The evaluation was done on four public datasets; Second Hand Songs 100K (SHS100K), Youtube, Covers80 and Mazurkas. They observed a performance better than state-of-the-art methods on all public datasets, with improved performance on the large dataset.

Literature Summary

The Deep learning approach generally uses neural networks in the task of CSI, they offer varying performance. A review of this technique shows that Deep learning can be applied for both the CSMs methods and the index-based methods. The deep learning method generally uses a Convolutional Neural network to evaluate similarities from a CSM, the accuracy seems high, while training is expensive. The review shows that this approach often provides a faster and large scale design for CSI. Methods such as temporal pooling techniques can be used to extract scalable and shift-invariant representations of an audio recording which are then used in a CNN for classification. These techniques generally have better promise for production ready CSI systems. A review of Deep learning methods using the index-based features shows that CNN coupled with CQT based inputs and classifier outputs can produce results that outperform the state-of-the-art methods. The CQT based method outperforming the stat-of-the-art methods makes it a good candidate feature for the implementation of a data augmentation process using GANs.

Chapter 3

Model Design and Implementation

The implementation of this research consists of three separate sections. The first section involves the data preparation and feature extraction of the custom music dataset provided by Breathe Music (Top25), the results from this stage were stored in NumPy files. In the second part, the NumPy files were converted to tensors using the PyTorch Dataset and DataLoader interface and then used to train a generative adversarial network, the same network architecture was used to train the CIFAR100 [30] dataset which is available as a preloaded dataset in the PyTorch dataset interface. The third and final step implements the metrics discussed in chapter 4 to evaluate the performance of the models. The entire research was implemented using the PyTorch development toolkit and executed using a CUDA enabled machine. Chapter 3 gives an intuitive explanation of the thought process behind the feature extraction, GAN implementation and evaluation techniques used for this research. The insights from this methodology provide a logical basis for which further modification can be made to enhance the data augmentation procedure, and the subsequent cover song identification system at Breathe Music.

3.1 Motivation

Cover song identification using deep learning, like all tasks in the deep learning paradigm, could see improved performance given a sufficiently large dataset. Cover song identification as defined in chapter 1 involves searching large databases of music to find music pieces with similar features. A common challenge in the implementation of a CSI system is the varying amount of cover song versions created for a song by different cover artists, this may vary based on different reasons such as popularity of original song, song genre (pop or rock) e.t.c. Machine learning researchers in the MIR field need to find a way to solve this problem which can be considered a class imbalance problem, where some songs have very sparse covers

made for them. Only a small subset of the literature appears to focus on the problem of data augmentation to address the class imbalance issue found in CSI. Most methods will employ the typical signal based transformations such as; time stretching, pitch shifting, Gaussian noise addition and re-sampling. This research utilises state-of-the-art GAN architectures to learn the distribution of the cover song identification dataset and attempt to generate artificial samples for the minority classes i.e songs with sparse covers.

3.2 Implementation Hardware

The implementation of this research was performed on machines provided by the University of South Wales. The machine has 4 hyper-threaded CPU cores (Intel(R) Core(TM) i7-6700k CPU @ 2.60GHz 2.81 GHz), the machine is also packed with Nvidia GeForce GTX 1060 (6GB) Graphics Card, with 15.9GB usable RAM. The machine is running the current version of PyTorch along with all the python libraries used for this research. To improve the rate of learning, the CUDA library created by NVIDIA Corporation was installed to enable GPU access in the PyTorch environment.

3.3 Pytorch Framework

PyTorch is an open-source machine learning and deep learning library based on the torch library, it uses tensors for implementing the various machine learning and deep learning tasks. PyTorch offers a simple interface to load data as tensors which aid the efficient implementation of the different Machine and deep learning algorithms. PyTorch provides a Pythonic paradigm that makes code implementation more efficient and provides support via CUDA for GPUs. The fundamental reason PyTorch was used in the implementation of this research was to be consistent with the framework used for the development of Breathe Music's CSI pipeline.

Pytorch Usability centric design

PyTorch ensures usability by introducing the "everything is a just a program" [41] approach to deep learning, it extends this approach to all aspects of deep learning. Using PyTorch means stages in the deep learning workflow such as layer definition, model composition, data loading, model optimizations and model training are all expressed as PyTorch Models. Some of the important features that help with this approach to deep learning are:

- Automatic Differentiation: This is a core building block of PyTorch, this engine called Autograd enables the implementation of automatic differentiation. The implementation of a neural network generally involves two passes, the forward pass to compute the output of the error function and the backward pass to compute the gradients of the learnable parameters. The backward pass is performed generally using a differentiation intensive algorithm called backpropagation [34], which uses chain rule to compute the gradients of the weights w.r.t to the error function. The automatic differentiation feature of PyTorch makes the implementation of this algorithm easy to use.

- Computation Graphs: The PyTorch Computational graph comprises nodes which are mathematical operators, these operators are Tensors and edges are functions that produced the output Tensors from input Tensors. This feature helps PyTorch compute the gradients of the network in a simple design approach.

- Pytorch DataLoaders and Datasets: One of the advantages of using PyTorch for developing Deep Learning algorithm is the design principles, Pytorch is Pythonic [41] which provides an interface for easy integration with other python libraries. The Numpy files generated were loaded into PyTorch by creating a custom Dataset class which inherits from the pre-loaded dataset class in PyTorch, this made working with the raw files intuitive and easy to ingest into the environment. The custom dataset class does the necessary transformations and converts the NumPy files into tensors which are then loaded in batches using the PyTorch DataLoader class.

3.4 Dataset

Two datasets were used in the implementation of this research, the CIFAR100 dataset [30] and Top25. The CIFAR100 dataset was used as a control dataset to validate the approach in this research. CIFAR100 is a standard dataset in machine learning and deep learning, the motivation behind its use was to conduct a comparative analysis using an industry-standard dataset, versus the Top25 dataset. The CIFAR100 dataset is a part of the PyTorch pre-loaded datasets and as such, no feature extraction is required, however, some transformations were done to make the tensors more efficient to work with. The Top25 dataset provided by the team at Breathe Music included excerpts from the top 25 hits per decade for the past 40 years(1980 - 2019). The songs were extracted from UK charts (www.uk-charts.top-source.com), the top 25 songs from each decade were curated. These datasets are considered representative of actual popular music within this time period and also contains music of different genres. The dataset contains 100 different original songs and their official cover songs. In conclusion, the

entire training set for the Breathe Music Top25 dataset adds up to approximately 1117 songs, while the CIFAR100 dataset is 50,000 samples.

3.5 Feature Extraction

This section involves an explanation of the feature extraction steps utilised for the datasets in this research.

3.5.1 CIFAR100 image features

The CIFAR100 dataset was imported using the PyTorch DataLoader and Dataset modules, these modules make it easy to load industry-standard datasets into the PyTorch environment. The hyperbolic transformation was then applied to normalize the data between (-1,1).

3.5.2 Audio Features

The techniques and toolkits to extract the features needed for tasks in MIR have been discussed extensively in chapter 2, exploring the different features discussed in the literature. This section focuses on the steps taken to transfer raw audio provided by Breathe Music into the necessary feature representations needed in this research. The features were extracted using the librosa library to extract the CQT features for each audio recording which aids the easy extraction of the features and help avoid the complex task of manually extracting these features.

Top25 dataset CQT features

The CQT was the feature of choice for the implementation of the GAN, the different features used in MIR tasks have been extensively discussed in the literature review section of this research. The CQT spectrogram was one with good performance as seen in the work of Yu et al. [57]. CQT can be plotted as spectrograms, which is then be treated like an image for the GAN implementation. The audio dataset provided was used along with the feature generator script to produce the set of features used to train the network. The total Top25 as stated in the dataset subsection in chapter 3 was 1171 which is quite small for a deep earning task, this is one of the reasons this task is being embarked on to generate more data for the identification network over at Breathe Music. The evaluation conducted, explores how well the models capture the underlying distribution of the two datasets used in this research. All the underlying statistical and mathematical computation required was conducted using the

librosa library. The output of the transformations were exported as NumPy files which were concatenated and imported to PyTorch.

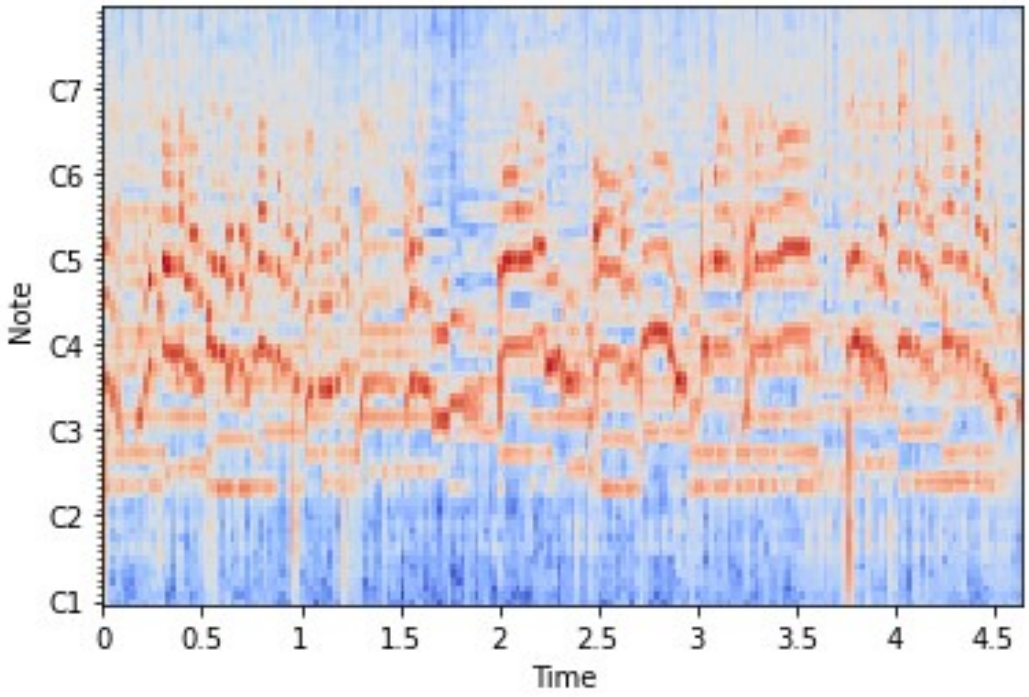


Figure 3.1: Sample CQT spectrogram from the training dataset

3.6 Generative Adversarial Networks

Generative adversarial networks is a type of unsupervised learning algorithm that includes two neural networks aiming to learn the probability distribution of a set of training samples [21]. GANs employ both unsupervised or semi-supervised techniques to learn the underlying probability distribution, they use backpropagation to learn the distribution through a competitive network design. GANs are composed of two competitive networks in a min-max competition, the two networks are called the generator and the discriminator. A common analogy used to form intuition for GANs is that of an art forger and the art expert. The art forger in this case is the generator G which tries to generate realistic fake art samples (features) to fool the art expert, known as the discriminator D. The discriminator receives both samples of fake and real images and aims to distinguish between the two samples. These two networks are trained simultaneously and are in a continuous competition to gain an advantage over the other. The output dimension of the generator is typically the input dimension of the discriminator. The generator G and discriminator D are both trained simultaneously:

The optimizers adjust the parameters for G to minimize $\log(1 - D(G(z)))$ and adjust parameters for D to minimize $\log D(x)$, to form the competitive min-max game with value function $V(G, D)$:

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

Generator

The generator typically is a neural network that learns to map random noise to a feature representation in a latent space, the network takes a fixed length vector drawn from a multivariate normal distribution and generates a sample in from a specific class domain. The GAN is designed in such a manner that the generator has no direct access to the real images used for training, the generator learns to get better by evaluating the loss from the discriminator.

Discriminator

The discriminator is tasked with distinguishing real images from fake images, the network is similar to typical classifiers, with the discriminator classifying samples into real or fake samples from the generator. The discriminator has access to both the real and fake samples, the discriminator loss is derived by taking the average of the discriminator loss for both the real and fake samples and the error from this is used to train the generator to get better samples.

3.7 Models

The proposed GAN-based model was based on the DCGAN [44] and the cGANs [38]. The model was built based on these two architectures as detailed in the subsections below and a summary can be found in Appendix A of this research.

3.7.1 DCGAN

The Deep Convolutional Generative Adversarial Networks (DCGAN) is a GAN framework which is built on the convolutional neural network architecture for generating optimized GAN models. The DCGAN learns a representation of the features by using only a specific part of the entire image in both the generator and discriminator network through convolutions [44]. Figure 3.2 shows the network architecture of the DCGAN.

The main features and implementations from the DCGAN can be found below:

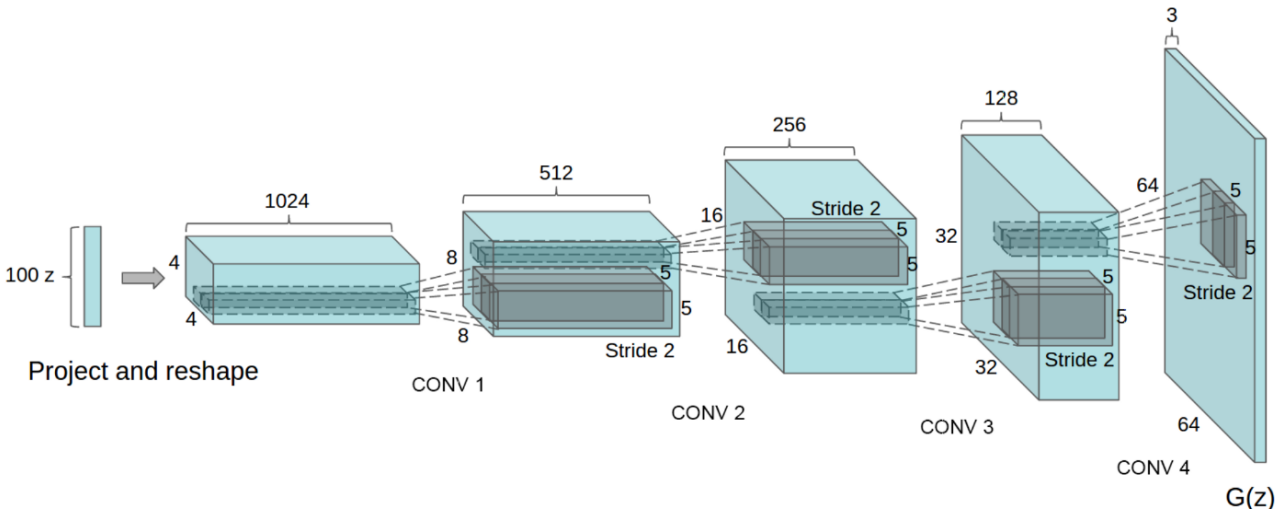


Figure 3.2: Network Design for DCGANs [44]

- **All pooling layers were replaced with strided convolution for the discriminator and a fractional strided convolution for the generator.** Given an input greyscale image with a width and height of 28 pixels i.e 28 x 28 images, each entry in this 28 x 28 square of neurons indicates 28×28 pixel intensities. The convolution operation is an iterative process that takes a kernel (filter) of a certain size e.g 3 x 3 and slides this filter over the entire input image starting from the top left and uses the dot-product to calculate the output which now has a new dimension and holds important details of the input image. The convolutional kernel is usually parameterised by the padding and the stride used. The stride refers to the degree of the jump from one pixel to another during the convolution operation, the padding allows the use of convolutions without shrinking the height and width of the input and also helps ensure more information is captured at the border of the images. An equation for getting the feature map size when Using strided and padded convolutions, for an input image of size nxn, and a kernel of size kxk:

$$\left(\left\lfloor \frac{n+2p-k}{s} \right\rfloor + 1\right) \text{x} \left(\left\lfloor \frac{n+2p-k}{s} \right\rfloor + 1\right)$$

- **Batch normalization was implemented for both the discriminator and generator network.** Batch normalization is an optimisation technique used in deep learning which makes the hyperparameter search tasks in the network easier and also provides a more robust and stable network. This technique normalises the inputs to a layer for each mini-batch of the neural networks, which means every activation going from one

layer to another is normalized. The batch normalization algorithm is summarized in the figure below.

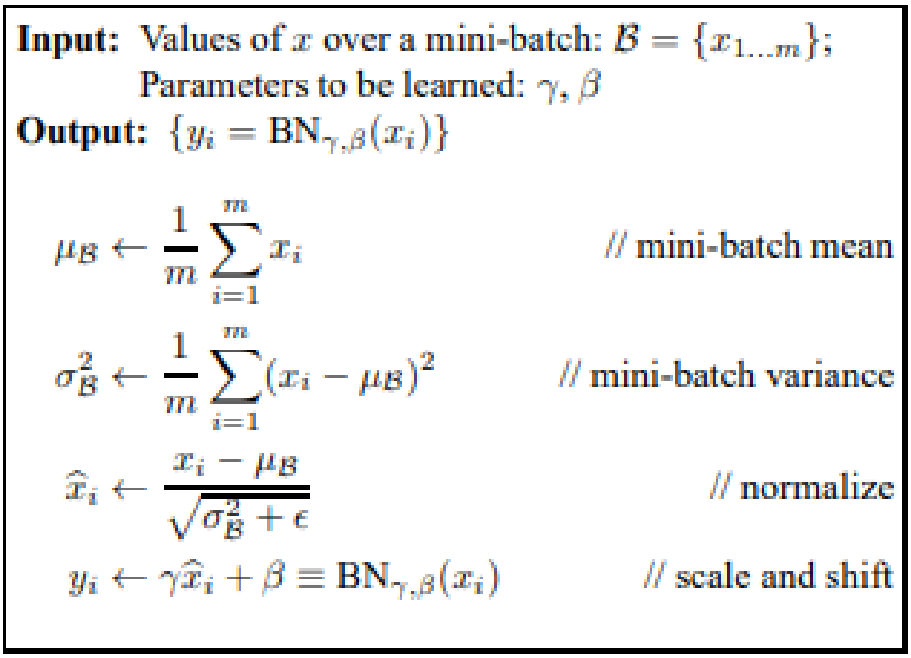


Figure 3.3: Batch Normalization Algorithm applied to activation 'x' over a mini-batch [25]

- **All fully connected hidden layers are removed from the networks.** Fully connected layers in a neural network are layers where every single neuron in each layer connects to every neuron in adjacent layers as seen in figure 3.4.

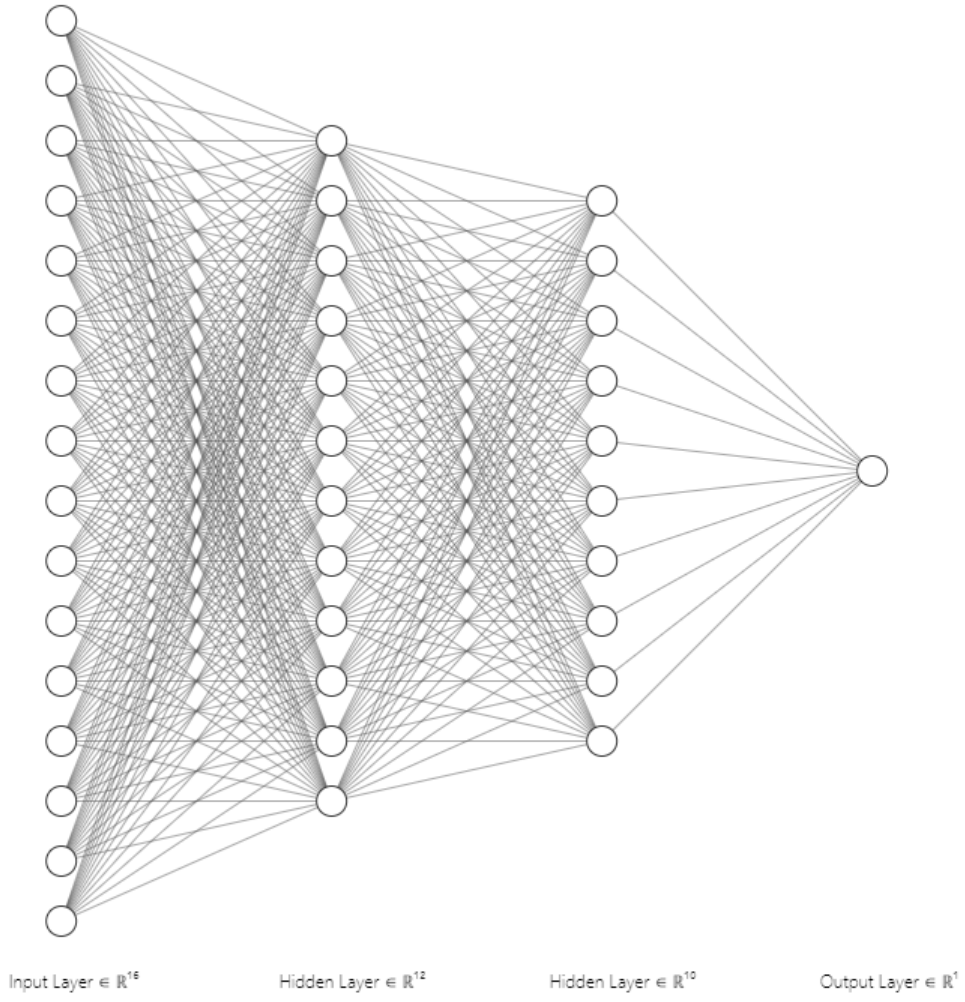


Figure 3.4: Fully connected Network architecture [33]

- **All layers in the generator used the ReLU activation function except the output layer which employed the hyperbolic activation function.** Activation functions is a component of a neural network with a special use case in Machine Learning, the activation of a particular neuron can hold information such as how much of importance is the neuron to the desired outputs. The Rectified Linear Unit (ReLU) activation is one of the most effective activation functions, it takes the maximum be-

tween the input z and $zero$ $f(x) = max(0, x)$, this activation function squashes all negatives in the network by returning zero for negative values. The function can be represented graphically as seen in figure 3.5 below, where negative z values will output zero basically eliminating negative values.

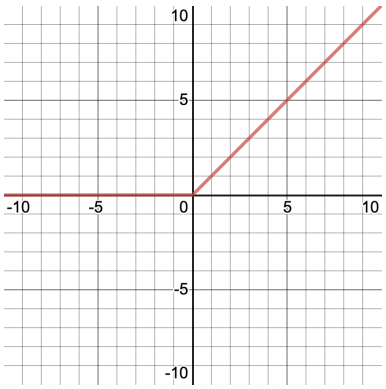


Figure 3.5: graphical representation of the ReLU activation function at given values.

- **The discriminator network used LeakyReLU activation for all layers.** The graphical representation of the ReLU activation function above in 3.5 has a flat part which can be tricky for the learning of the network since learning depends on finding derivatives for updating weights and biases. The zero derivative means some neurons cannot update weights and biases which is called the dying ReLU problem and this is where LeakyReLU 3.6 comes in to solve this problem, LeakyReLU solves the dying ReLU problem and ensure learning is done by ensuring weights and biases are updated.

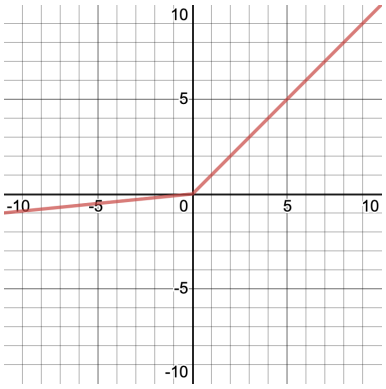


Figure 3.6: graphical representation of the LeakyReLU activation function at given values.

3.7.2 Conditional GAN

Conditional GANs (cGANs) are a type of GAN framework where the discriminator and the generator are both conditioned to generate outputs from a specific domain or class. They are implemented to find the probability distribution of a feature representation X given a class Y $\mathbb{P}(X|Y)$. Conditional GANs are important when dealing with multimodal data generation, it helps the network generate a diverse class of features.

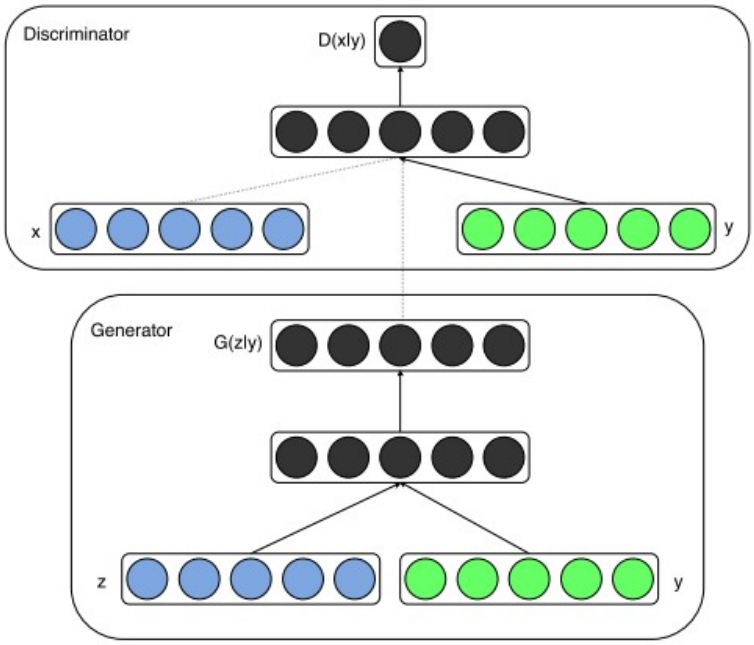


Figure 3.7: Network Design for Conditional GANs [38]

The control over what class of features are produced by the generator is implemented by adding a vector of features holding the class information. The vector of features is derived by performing a one-hot encoding of the class and concatenating it with the random noise vector sent into the generator and this class label is also passed into the discriminator model. The discriminator is then tasked with distinguishing between fake and real samples and also the accuracy of the input labels.

3.8 Training Strategy

The training dataset for both the Top25 dataset and CIFAR100 dataset consists of the entire dataset, no train/ test splitting was done since this was an unsupervised learning task with the aim of learning the underlying distribution of the datasets. The Top25 dataset was categorised into classes of the song, which means for training no discrimination was made between the original and cover version of a song, the classification was based on the song, a cover and the original version of a song belonging to the same class. The Binary Cross Entropy function was the loss function employed in this implementation, while the optimization algorithm used was the Adam Optimizer for both datasets — Top25 dataset and CIFAR100.

3.8.1 Binary Cross Entropy function

The cross entropy function for activation function with $y \in (0, 1)$ is written as;

$$\text{BCE} = -\frac{1}{m}\sum_{i=1}^{m}[y\log(p) + \log(1-p)]$$

The binary cross entropy function is generally used when dealing with binary outputs, this gives a better outcome for a loss function than using the mean squared error since it is a classification task and not regression and the expected output is discrete unlike that of regression which is continuous. The cross entropy function written above is used for the output of the sigmoid function since it takes the pre-activation input and returns output between 0 and 1.

The hypothesis for a binary classification task which has $y \in (0, 1)$ can be represented as:

$$\hat{y} = \sigma(b + W^T x)$$

$$z = (b + W^T x)$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Removing the sigmoid function in the hypothesis leads to outputs that are greater than 1 or lesser than 0 but with the goal of classification which is to classify outcomes into discrete groups this will cause problems for the model, so the sigmoid function helps squish the outcome into a range of 0 to 1, hence the output of the hypothesis function can be interpreted as the probability of getting an output which is 0 or an output which is 1.

$$\hat{y} = P(y = 1|x; W) = 1 - P(y = 0|x; W)$$

$$P(y = 0|x; \theta) + P(y = 1|x; \theta) = 1$$

A threshold above which the model classifies the output as 1 or below the threshold which the model classifies the output as 0 is then decided such as the one in the equation below:

$$\hat{y} \geq 0.5 \rightarrow y = 1$$

$$\hat{y} < 0.5 \rightarrow y = 0$$

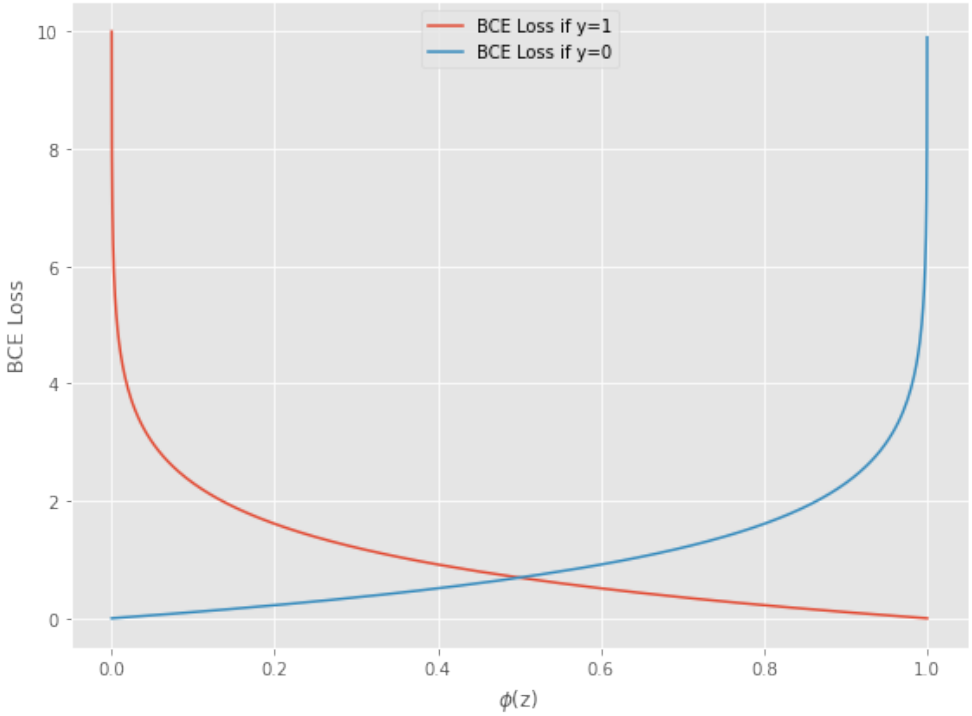


Figure 3.8: Binary Cross Entropy Loss Function Plot

3.8.2 Optimization

Neural networks are trained to optimise the weight and biases of the network, the network tries to find a set of weights and biases for the network that reduces the error loss. There are a wide range of optimization algorithms employed in the field of machine learning. The change in weight and biases of the network to reduce losses is performed by the optimizers and they help the network provided the best results. The ADAM optimizer Adaptive Moment

Estimation, ADAM, this is an adaptive learning rate method that adapts its learning rate for different parameters. The ADAM optimizer combines the adaptive gradients and root means square propagation. This method uses random samples from the data subset to create a stochastic approximation.

3.9 Evaluating GANs

GANs generally are evaluated empirically, unlike other machine learning or deep learning tasks where performance can be estimated using metrics such as likelihood estimation, or mean accuracy. The performance of generative models are generally subjective and there is no universal technique for measuring how well a model performed. When evaluating GANs, observing how well the model maps the underlying feature representation shows how well the model is doing, some important criteria to evaluate in GANs are fidelity and diversity. Fidelity shows how well the generated samples mimic the original samples, diversity shows how well the network is able to generate samples from the different classes in the dataset. These evaluation criteria might require human in the loop to visually observe the samples generated. For large scale manual evaluation of fidelity and diversity, tools such as amazon mechanical turk [26] can be used to get a more diverse judgement of the sample quality using human visual evaluation. An evaluation technique that was designed to solve the problems with evaluating GANs, called the Fréchet Inception Distance [16]. This method for evaluating GANs was identified in the published literature reviewed in chapter 2 of this research. The FID was then implemented in this research to evaluate the performance of the GAN models used.

3.9.1 Fréchet Inception Distance

The Fréchet Inception Distance (FID) is a technique adopted in deep learning for measuring the quality of generated image samples, this evaluation metric was an improvement on the inception score [45]. The FID intuitively finds the shortest walking distance along two lines or curves simultaneously. The FID measures the similarity between two curves or lines by measuring their distance apart. The FID is used to statistically compare samples from two different Gaussians probability distributions, these distributions can either be univariate or multivariate.

Univariate Fréchet Distance

The univariate distance between two Gaussian distributions X and Y can be calculated using their mean μ_X and μ_Y and standard deviation σ_X and σ_Y by implementing the Fréchet Distance:

$$d(X, Y) = (\mu_X - \mu_Y)^2 + (\sigma_X - \sigma_Y)^2$$

Multivariate Fréchet Distance

Using the Fréchet inception distance for two multivariate gaussian distributions, will involve the use of the covarance matrix Σ and mean μ [16], multivariate distributions requires the matrix product, matrix square root and the trace, the sum of the diagonal elements of a matrix:

$$d(X, Y) = \|\mu_X - \mu_Y\|^2 + \mathrm{Tr}\left(\Sigma_X + \Sigma_Y - 2\sqrt{\Sigma_X \Sigma_Y}\right)$$

Chapter 4

Results

In this chapter, the results obtained from the GAN model implemented for the Top25 dataset and the CIFAR100 dataset were presented. An empirical and quantitative approach was taken to evaluate the outcome of this research. The first part involves plotting the distribution of fake and real images, the second part comprises using a quantitative approach by calculating the Fréchet Inception Distance (FID) for both datasets and conducting a comparative analysis. A systematic approach is used where two different datasets are evaluated using the same GAN architecture and their performance reviewed. The two datasets are:

- The CIFAR100 [30] dataset is evaluated using a visual plot and the FID score comparing real images to fake.
- The Top25 dataset is evaluated using a visual plot and the FID score comparing real spectrograms to fake.

Finally, the performance of these two datasets will be compared using the same GAN architecture and hyperparameters where the number of epoch is 1000 and learning rate is 0.002.

4.1 Visual exploration

CIFAR100

The pairplot of the Real CIFAR100 dataset and the fake images generated by the generator shows that, at an epoch of 1000 and learning rate of 0.002, there is an overlap between the probability distribution of the real samples and the fake samples from the generator as seen in the pairplot in figure **??**. This shows the generator was able to capture the underlying distribution of this dataset and is capable of generating samples that are wrongly classified as the real samples by the discriminator.

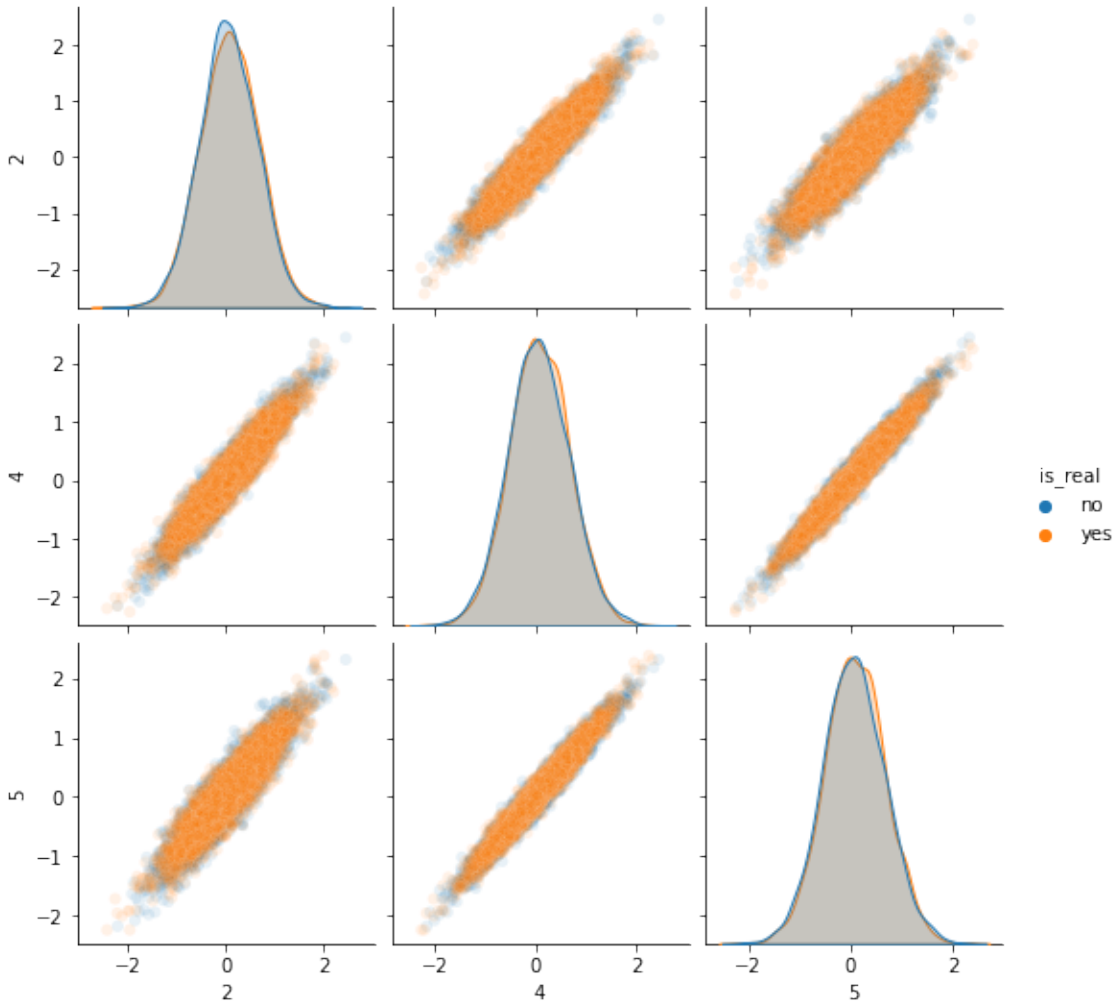


Figure 4.1: Pairplot of Real and Fake images in Cifar100

4.1.1 Breathe Music CQT Data

The Top25 dataset was pre-processed with librosa and the CQT features were extracted and converted to spectrograms. This dataset shows a very unusual distribution as seen in figure 4.2. The plot shows the real and fake data samples generated do not fall within the same distribution, this points in the direction of having a generator that is not producing samples from the original distribution. Based on the output of the pairplot, the model was unable to create a generator capable of generating samples superseding the discriminator. This shows a poor performance from the model.

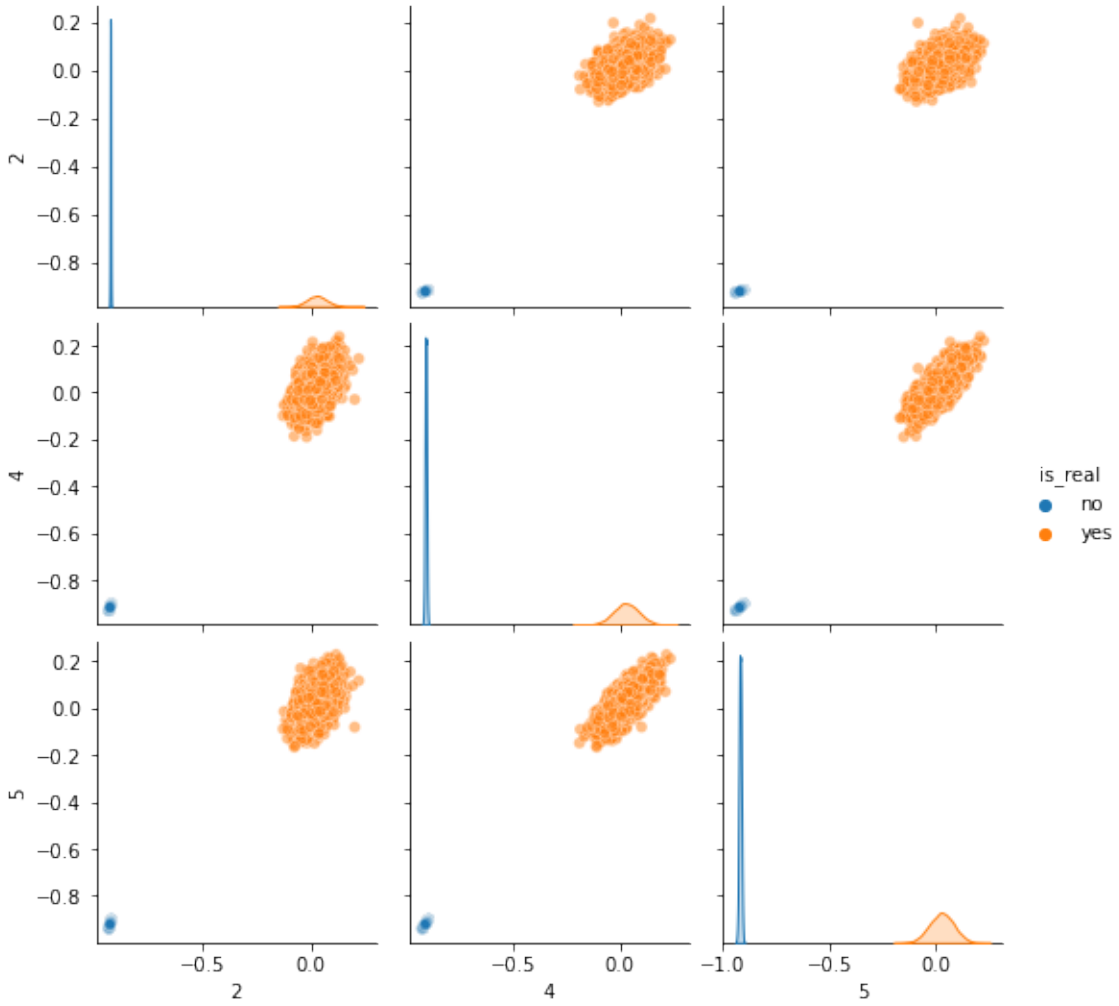


Figure 4.2: Pairplot of Real and Fake spectrograms in Top25 Dataset

4.2 Fréchet Inception Distance scores

The Fréchet Inception Distance score as detailed in the evaluation section of chapter 3 of this research measures similarity between two distributions by measuring the distance between the distributions. The lower this value the better, the table below show that the CIFAR100 dataset at 1000 epoch and learning rate of 0.002 has a lower score than the FID of Top25 dataset which is expected from the visual exploration done in figure 4.2.

Dataset	FID
CIFAR100	82.6212158203125
Top25 Dataset	14762.7197265625

Table 4.1: Fréchet Inception Distance scores for CIFAR100 and Top25 Dataset.

4.3 Result Summary

The results from evaluating the generated samples from the GAN for the CIFAR100 and Top25 datasets show that although the model was capable of generating samples for the CIFAR100 dataset, the model struggled to produce reasonable samples using the Top25 dataset. The FID observed for the CQT features extracted from the Top25 dataset was extremely high, which showed the distance between the real and generated sample is quite high. The observed FID for CIFAR100 dataset was very low, this means a shorter distance between the real and fake samples in the data space. The results show that, despite good performance observed for CIFAR100 using the same model, the performance for the Top25 dataset was poor.

Chapter 5

Discussion

The task of augmenting the dataset for cover song identification using generative adversarial networks has been implemented and evaluated. The entire research was done using PyTorch and the team at Breathe Music now have a basis on which to build further work into using this approach to augment the dataset for the CSI system. This research makes the following contributions to help the team at Breathe Music answer the following important questions about Data augmentation with GANs:

Why Generative Adversarial Networks? As discussed extensively in chapter 2 and 3 of this research. Generative adversarial networks are capable of generating new samples of a dataset from random noise vectors selected from a learned latent space, this makes GANs a more technically advanced approach to data augmentation rather than using simple methods such as rotating, cropping e.t.c. This research has explored state-of-the-art GAN models, although the desired results were not achieved using the CQT features, using the CIFAR100 dataset to validate the approach showed that this can be improved. Generative adversarial networks provide a more efficient means to augment a dataset on a larger scale and is capable of generating high fidelity and diverse samples, which is one of the advantages this method brings to the CSI system at Breathe Music.

What are the different types of GANs to be used? Implementing a generative adversarial network for a highly diverse dataset require a way to specify what class of song is expected, this research showed that conditional GANs can help solve this problem by concatenating the preferred class to the input noise vector passed to the generator and the input to the discriminator. Training a GAN network can lead to an explosion in the number of parameters to train, a type of GAN that can provide a more stable and trainable model was implemented, DCGAN [44]. A combination of the cGANs and DCGAN is the recommended approach to implementing a GAN for a dataset with diverse classes.

How can samples generated from the GAN models be evaluated? A quantitative

and visual approach to Evaluating GANs was proposed which is based on the methods used in deep learning for measuring similarities between data samples from similar distributions. This research explored two methods to evaluate GANs, the Frechet Inception Distance, and a visual exploratory analysis showing the probability distributions on a pair plot.

Can a GAN model used for images and computer vision problems be abstracted for audio tasks? This is one of the main questions this research attempts to answer, the review of the study by Chatziagapi et al. [11] showed great promise in this endeavour. However, based on this research to train a GAN model for audio problems will involve the combination of multiple generative models including variational autoencoders and GANs as seen in the work of Chatziagapi et al. [11]. The research by Back et al. [5] also supports this, the review of this research revealed that implementing the image models for the audio problem will involve further pre-processing of the features.

Finally, the results from this research showed that GANs are capable of creating samples that can be used to augment the original dataset, this was proved by the results from the CIFAR100 dataset which was used as a control mechanism to validate this approach. The CQT features were used as input to the GAN, with a training data size of 1171 which will be considered a small dataset for most deep learning tasks. The GAN implemented in this research using the CIFAR100 dataset showed a relatively low FID compared to that observed from the CQT features. The promising result derived from the CIFAR100 dataset shows that GANs can be used to augment a dataset, while the Breathe Music CQT features may have shown poor performance due to the much smaller datasets for training and the need for hyperparameter tuning in the model.

5.1 Further Work

Based on the work done in this research, there a few suggestions and approaches for future research in using GANs for augmenting data for CSI systems. Exploring the methodology of some published research reviewed in chapter 2, two main suggestions stand out as a way to improve the model:

5.1.1 Increasing Training Dataset

The total count of the Top25 dataset used to train the model was 1171, which is a very small dataset for a deep learning task. Learning the latent space of a data distribution exists in the context of the training dataset used, the fidelity and diversity of the underlying dataset used for the training of the model in this research need to be improved on, more datapoints

and accurately labelled datapoints needs to be collected. Collecting enough diverse and high fidelity data will help improve the performance of the GAN on the data, a poor data representation is likely to lead to poor performance. A good data representation can lead to high performance for this model implementation.

5.1.2 Improving Model Design

The model design used can be further improved by combining the current model with other model design paradigms used in GANs. Model stability during training is an issue, an attempt can be made to solve this by implementing the Wasserstein GAN with Gradient Penalty (WGAN-GP) [4]. The literature for data augmentation of speech dataset conducted by Chatziagapi et al [11] shows that to create a model with faster convergence, the GAN model can be initialized using a pre-trained autoencoder. By using this method, the variational autoencoder can learn the general latent space representation of the underlying dataset without any explicit class knowledge, sample vectors can then be selected from this space and passed into the generator. The use of progressively-growing GAN can also be employed to generate high fidelity samples, all these different improvement [11].

Chapter 6

Conclusion

This research was an attempt to provide an exploratory deep dive into using GANs as a data augmentation technique that can be embedded into the current cover song identification system at Breathe Music. The results from this research showed that although GANs are capable of approximating the distribution of a dataset with highly specialised images such as the CIFAR100 dataset, GANs struggle with creating spectrogram samples using the same model that is used for images. This showed that a more specialised model design for audio data needs to be implemented. Breathe Music currently uses traditional data augmentation methods to augment their data, this research was undertaken to allow further research work to be done for creating high performing GANs capable of generating highly diverse cover song samples to help train the current implementation of CSI at the company. To further improve on the augmentation approach at the company, research into the use of GANs and its different frameworks was conducted, the discovery made was that converting the audio into CQT feature spectrograms and treating these spectrograms as images showed a very poor performance which means a highly specialised model customised for spectrograms need to be researched and experimented with. This research incorporates observations from published work in the field of MIR, using a feature that showed great results for CSI. The GAN models used were critically evaluated and the areas with shortcomings were detailed, areas needing further exploration and improvement were also suggested. Results show that while GANs can capture the underlying distribution of a highly curated image dataset, using CQT features performed woefully, further improvements will be required to create a baseline performance for CQT features.

Bibliography

[1] 2010:Audio Beat Tracking - MIREX Wiki. URL: https://music-ir.org/mirex/wiki/2010:Audio_Beat_Tracking#Data.

[2] Peter Boesman :: xeno-canto. URL: https://www.xeno-canto.org/contributor/OOECIWCSWV.

[3] Anon. 2019:Audio Cover Song Identification - MIREX Wiki. URL: https://www.music-ir.org/mirex/wiki/2019:Audio_Cover_Song_Identification#Evaluation.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. jan 2017. URL: https://arxiv.org/abs/1701.07875http://arxiv.org/abs/1701.07875, arXiv:1701.07875.

[5] Moon Ki Back, Seung Won Yoon, and Kyu Chul Lee. GAN-based Augmentation for Populating Speech Dataset with High Fidelity Synthesized Audio. *International Conference on ICT Convergence*, 2020-Octob:1267–1269, 2020. doi:10.1109/ICTC49870.2020.9289283.

[6] Thierry Bertin-Mahieux and Daniel P.W. Ellis. Large-scale cover song recognition using the 2D Fourier transform magnitude. In *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012*, pages 241–246, 2012. URL: http://www.spotify.com.

[7] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011*, pages 591–596, 2011. URL: http://www.infochimps.com/.

[8] Judith C. Brown. Calculation of a constant Q spectral transform. *Journal of the Acoustical Society of America*, 89(1):425–434, 1991. doi:10.1121/1.400476.

[9] Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N. Chang, Sungbok Lee, and Shrikanth S. Narayanan. IEMOCAP: interactive emotional dyadic motion capture database. *Language Resources and Evaluation 2008 42:4*, 42(4):335–359, nov 2008. URL: https://link.springer.com/article/10.1007/s10579-008-9076-6, doi:10.1007/S10579-008-9076-6.

[10] Michael A. Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008. doi:10.1109/JPROC.2008.916370.

[11] Aggelina Chatziagapi, Georgios Paraskevopoulos, Dimitris Sgouropoulos, Georgios Pantazopoulos, Malvina Nikandrou, Theodoros Giannakopoulos, Athanasios Katsamanis, Alexandros Potamianos, and Shrikanth Narayanan. Data augmentation using GANs for speech emotion recognition. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2019-September:171–175, 2019. doi:10.21437/Interspeech.2019-2561.

[12] Keunwoo Choi, György Fazekas, Kyunghyun Cho, and Mark Sandler. A Tutorial on Deep Learning for Music Information Retrieval. sep 2017. URL: https://arxiv.org/abs/1709.04396, arXiv:1709.04396.

[13] Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak. Hyperspherical variational auto-encoders. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, volume 2, pages 856–865, apr 2018. URL: https://arxiv.org/abs/1804.00891, arXiv:1804.00891.

[14] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. 2 2018. URL: http://arxiv.org/abs/1802.04208.

[15] J. S. Downie, K. West, E. Pampalk, and P. Lamere. Mirex2006 audio cover song evaluation., 2006. URL: https://www.music-ir.org/mirex/wiki/2006:Audio_Cover_Song.

[16] D. C. Dowson and B. V. Landau. The Fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450–455, 1982. doi:10.1016/0047-259X(82)90077-X.

[17] Daniel P.W. Ellis. The "covers80" cover song data set. URL: http://labrosa.ee.columbia.edu/projects/coversongs/covers80/.

[18] Daniel P.W. Ellis and Graham E. Poliner. Identifying 'cover songs' with chroma features and dynamic programming beat tracking. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, volume 4, 2007. URL: http://labrosa.ee., doi:10.1109/ICASSP.2007.367348.

[19] John S. Garofolo, Lori F. Lamel, Wiliam M. Fischer, Jonathan G. Fiscus, David S. Pallett, and Nancy L. Dahlgren. The darpa timit acoustic-phonetic continuous speech corpus cd-rom. *Nist*, 1986.

[20] Michal Genussov and Israel Cohen. Musical genre classification of audio signals using geometric methods. *European Signal Processing Conference*, 10(5):497–501, 2010.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, jun 2014. URL: http://arxiv.org/abs/1406.2661, arXiv:1406.2661, doi:10.1145/3422622.

[22] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. 3 2017. URL: http://arxiv.org/abs/1704.00028.

[23] Eric J Humphrey, Oriol Nieto, and Juan P Bello. Data driven and discriminative projections for large-scale cover song identification. In *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013*, pages 149–154, 2013. URL: http://soundcloud.com.

[24] M. Huzaifah and L. Wyse. Deep generative models for musical audio synthesis. jun 2020. URL: https://arxiv.org/abs/2006.06426, arXiv:2006.06426.

[25] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *32nd International Conference on Machine Learning, ICML 2015*, 1:448–456, 2015. arXiv:1502.03167.

[26] Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. Quality management on Amazon Mechanical Turk. *Workshop Proceedings - Human Computation Workshop 2010, HCOMP2010*, pages 64–67, 2010. doi:10.1145/1837885.1837906.

[27] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, oct 2018. URL: http://arxiv.org/abs/1710.10196, arXiv:1710.10196.

[28] Maksim Khadkevich and Maurizio Omologo. Large-scale cover song identification using chord profiles. In *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013*, pages 233–238, 2013. URL: https://github.com/FBK-SHINE/CoverSongData.

[29] Peter Knees and Markus Schedl. *Music similarity and retrieval*. 2013. doi:10.1145/2484028.2484193.

[30] Alex Krizhevsky and G Hinton. Learning multiple layers of features from tiny images.(2009). Technical report, 2009. URL: https://www.cs.toronto.edu/~kriz/cifar.html.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. URL: http://code.google.com/p/cuda-convnet/.

[32] Juheon Lee, Sungkyun Chang, Sang Keun Choe, and Kyogu Lee. Cover Song Identification Using Song-to-Song Cross-Similarity Matrix with Convolutional Neural Network. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, volume 2018-April, pages 396–400. Institute of Electrical and Electronics Engineers Inc., sep 2018. doi:10.1109/ICASSP.2018.8461395.

[33] Alexander LeNail. NN-SVG: Publication-Ready Neural Network Architecture Schematics. *Journal of Open Source Software*, 4(33):747, jan 2019. URL: https://joss.theoj.org/papers/10.21105/joss.00747, doi:10.21105/joss.00747.

[34] H. Leung and S. Haykin. The complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, 39(9):2101–2104, 1991. doi:10.1109/78.134446.

[35] Tao Li and Mitsunori Ogihara. Toward intelligent music information retrieval. *IEEE Transactions on Multimedia*, 8(3):564–574, 2006. doi:10.1109/TMM.2006.870730.

[36] Daniel W Griffin A N D Jae S Lim and Senior Member. Signal estimation from modified short-time fourier transform, 1984.

[37] Giovanni Mariani, Florian Scheidegger, Roxana Istrate, Costas Bekas, and Cristiano Malossi. BAGAN: Data Augmentation with Balancing GAN. mar 2018. URL: https://arxiv.org/abs/1803.09655http://arxiv.org/abs/1803.09655, arXiv:1803.09655.

[38] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. nov 2014. URL: https://arxiv.org/abs/1411.1784http://arxiv.org/abs/1411.1784, arXiv:1411.1784.

[39] Meinard Mller. Fundamentals of music processing: Audio, analysis, algorithms, applications. 2015.

[40] Abdullah Mueen, Yan Zhu, Michael Yeh, Kaveh Kamgar, Krishnamurthy Viswanathan, Chetan Gupta, and Eamonn Keogh. The fastest similarity search algorithm for time series subsequences under euclidean distance, August 2017. http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html.

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019. arXiv:1912.01703.

[42] Spencer S. Pearson and Yasin N. Silva. Index-based R-S similarity joins. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8821, pages 106–112. Springer, Cham, oct 2014. URL: https://link.springer.com/chapter/10.1007/978-3-319-11988-5_10, doi:10.1007/978-3-319-11988-5_10.

[43] Luis Perez and Jason Wang. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. dec 2017. URL: http://arxiv.org/abs/1712.04621, arXiv:1712.04621.

[44] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, nov 2016. URL: http://arxiv.org/abs/1511.06434, arXiv:1511.06434.

[45] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016. URL: https://github.com/openai/, arXiv:1606.03498.

[46] Christian Schörkhuber and Anssi Klapuri. Constant-Q transform toolbox for music processing. *Proceedings of the 7th Sound and Music Computing Conference, SMC 2010*, page 20, 2010.

[47] Joan Serra, Xavier Serra, and Ralph G. Andrzejak. Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11, sep 2009. doi:10.1088/1367-2630/11/9/093017.

[48] Diego F. Silva, Vinícius M.A. Souza, and Gustavo E.A.P.A. Batista. Music shapelets for fast cover song recognition. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015*, pages 441–447, Malaga, Spain, 2015.

[49] Diego F Silva, Chin Chia M. Yeh, Gustavo E.A.P.A. Batista, and Eamonn Keogh. SiMPle: Assessing music similarity using subsequences joins. In *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016*, pages 23–29, 2016.

[50] Diego Furtado Silva, Felipe Vieira Falcão, and Nazareno Andrade. Summarizing and comparing music data and its application on cover song identification. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, pages 732–739, 2018.

[51] Christopher J. Tralie. Early MFCC and HPCP fusion for robust cover song identification. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017*, pages 294–301, jul 2017. URL: http://arxiv.org/abs/1707.04680, arXiv:1707.04680.

[52] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. 9 2016. URL: http://arxiv.org/abs/1609.03499.

[53] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. 4 2018. URL: http://arxiv.org/abs/1804.03209.

[54] Furkan Yesiler, Joan Serra, and Emilia Gomez. Accurate and Scalable Version Identification Using Musically-Motivated Embeddings. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, volume 2020-May, pages 21–25, 2020. arXiv:1910.12551, doi:10.1109/ICASSP40776.2020.9053793.

[55] Furkan Yesiler, Chris Tralie, Albin Correya, Diego F Silva, Philip Tovstogan, Emilia Gómez, and Xavier Serra. Da-tacos: A dataset for cover song identification and understanding. In *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, pages 327–334, 2019. URL: http://secondhandsongs.com.

[56] Zhesong Yu, Xiaoshuo Xu, Xiaoou Chen, and Deshun Yang. Temporal pyramid pooling convolutional neural network for cover song identification. In *IJCAI International Joint Conference on Artificial Intelligence*, volume 2019-Augus, pages 4846–4852, 2019. URL: https://github.com/yzspku/TPPNet, doi:10.24963/ijcai.2019/673.

[57] Zhesong Yu, Xiaoshuo Xu, Xiaoou Chen, and Deshun Yang. Learning a Representation for Cover Song Identification Using Convolutional Neural Network. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, volume 2020-May, pages 541–545, nov 2020. URL: http://arxiv.org/abs/1911.00334, arXiv:1911.00334, doi:10.1109/ICASSP40776.2020.9053839.

Appendix A

PyTorch Model Summary

```
----------------------------------------------------------------------------
         Layer (type)          Input Shape         Param #     Tr. Param #
============================================================================
    ConvTranspose2d-1       [1, 100, 1, 1]       4,096,256       4,096,256
        BatchNorm2d-2      [1, 256, 8, 20]             512             512
               ReLU-3      [1, 256, 8, 20]               0               0
    ConvTranspose2d-4      [1, 256, 8, 20]       1,638,528       1,638,528
        BatchNorm2d-5     [1, 128, 19, 48]             256             256
               ReLU-6     [1, 128, 19, 48]               0               0
    ConvTranspose2d-7     [1, 128, 19, 48]         294,976         294,976
        BatchNorm2d-8     [1, 64, 42, 100]             128             128
               ReLU-9     [1, 64, 42, 100]               0               0
   ConvTranspose2d-10     [1, 64, 42, 100]             257             257
              Tanh-11      [1, 1, 84, 200]               0               0
============================================================================
Total params: 6,030,913
Trainable params: 6,030,913
Non-trainable params: 0
----------------------------------------------------------------------------
----------------------------------------------------------------------------
         Layer (type)         Output Shape         Param #     Tr. Param #
============================================================================
    ConvTranspose2d-1      [1, 256, 8, 20]       4,096,256       4,096,256
        BatchNorm2d-2      [1, 256, 8, 20]             512             512
               ReLU-3      [1, 256, 8, 20]               0               0
    ConvTranspose2d-4     [1, 128, 19, 48]       1,638,528       1,638,528
        BatchNorm2d-5     [1, 128, 19, 48]             256             256
               ReLU-6     [1, 128, 19, 48]               0               0
    ConvTranspose2d-7     [1, 64, 42, 100]         294,976         294,976
        BatchNorm2d-8     [1, 64, 42, 100]             128             128
               ReLU-9     [1, 64, 42, 100]               0               0
   ConvTranspose2d-10      [1, 1, 84, 200]             257             257
              Tanh-11      [1, 1, 84, 200]               0               0
============================================================================
Total params: 6,030,913
Trainable params: 6,030,913
Non-trainable params: 0
```

Figure A.1: Generator Model summary

```
----------------------------------------------------------------------------
      Layer (type)           Input Shape         Param #     Tr. Param #
============================================================================
          Conv2d-1      [10, 1, 84, 200]          34,624          34,624
     BatchNorm2d-2      [10, 64, 38, 74]             128             128
       LeakyReLU-3      [10, 64, 38, 74]               0               0
          Conv2d-4      [10, 64, 38, 74]       5,292,160       5,292,160
     BatchNorm2d-5     [10, 128, 11, 19]             256             256
       LeakyReLU-6     [10, 128, 11, 19]               0               0
          Conv2d-7     [10, 128, 11, 19]          23,041          23,041
============================================================================
Total params: 5,350,209
Trainable params: 5,350,209
Non-trainable params: 0
----------------------------------------------------------------------------
----------------------------------------------------------------------------
      Layer (type)          Output Shape         Param #     Tr. Param #
============================================================================
          Conv2d-1      [10, 64, 38, 74]          34,624          34,624
     BatchNorm2d-2      [10, 64, 38, 74]             128             128
       LeakyReLU-3      [10, 64, 38, 74]               0               0
          Conv2d-4     [10, 128, 11, 19]       5,292,160       5,292,160
     BatchNorm2d-5     [10, 128, 11, 19]             256             256
       LeakyReLU-6     [10, 128, 11, 19]               0               0
          Conv2d-7         [10, 1, 1, 1]          23,041          23,041
============================================================================
Total params: 5,350,209
Trainable params: 5,350,209
Non-trainable params: 0
----------------------------------------------------------------------------
----------------------------------------------------------------------------
```

Figure A.2: Discriminator Model summary

Appendix B

Outputs From GAN

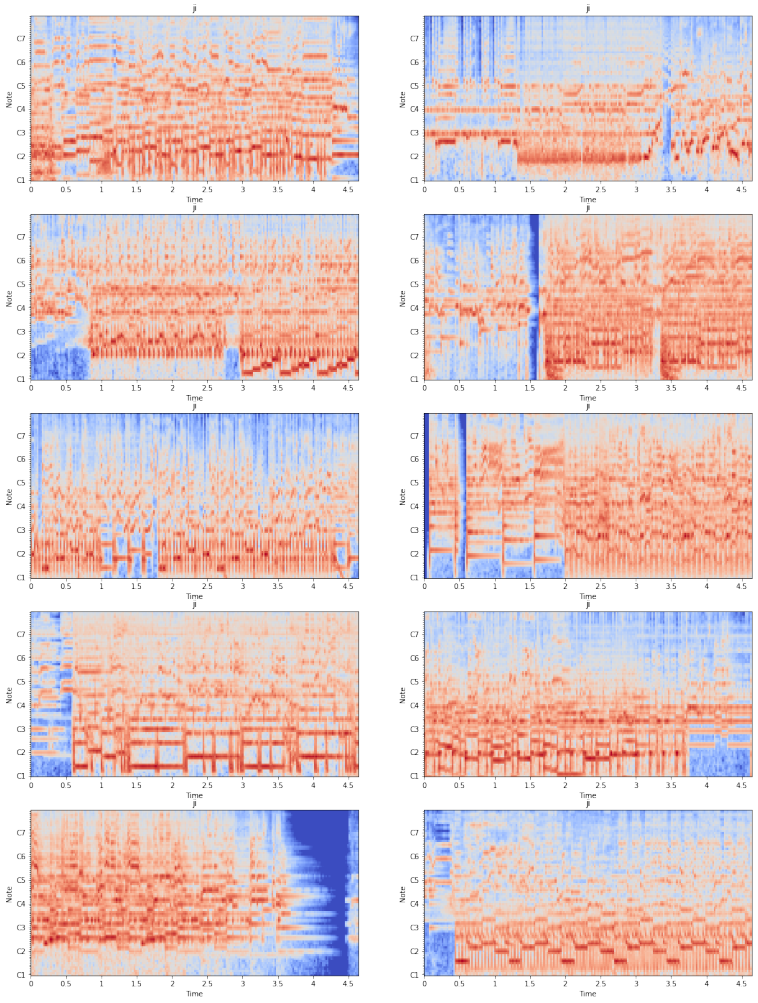


Figure B.1: Samples from the real Top25 Data CQT

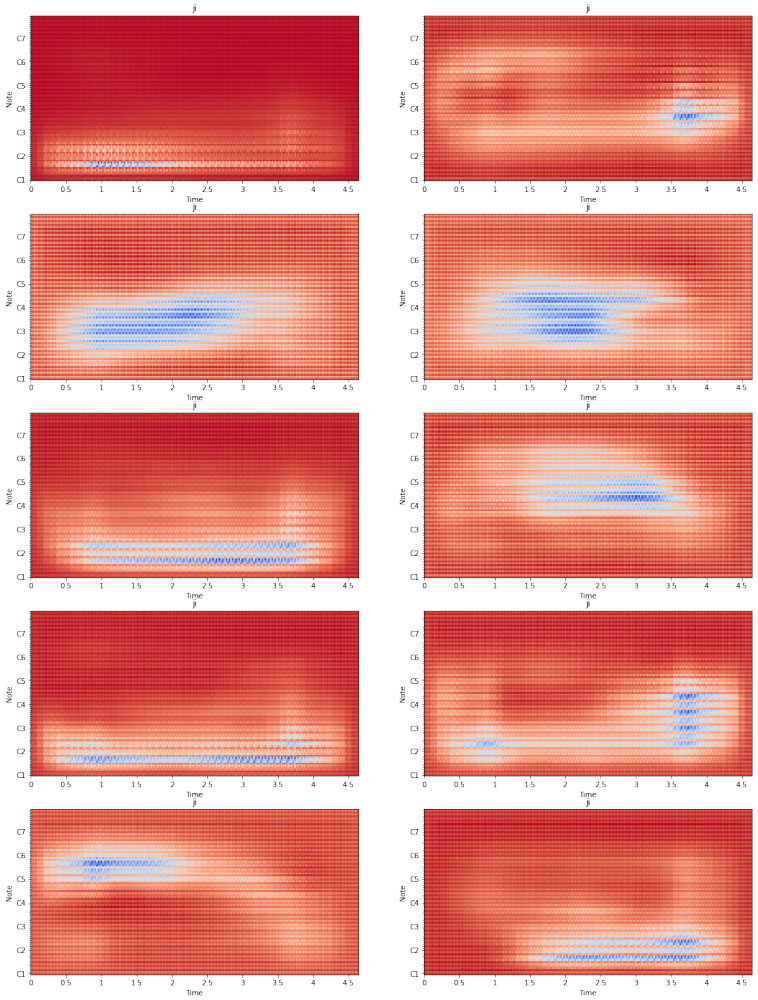


Figure B.2: Samples from the fake Top25 Data CQT from Generator

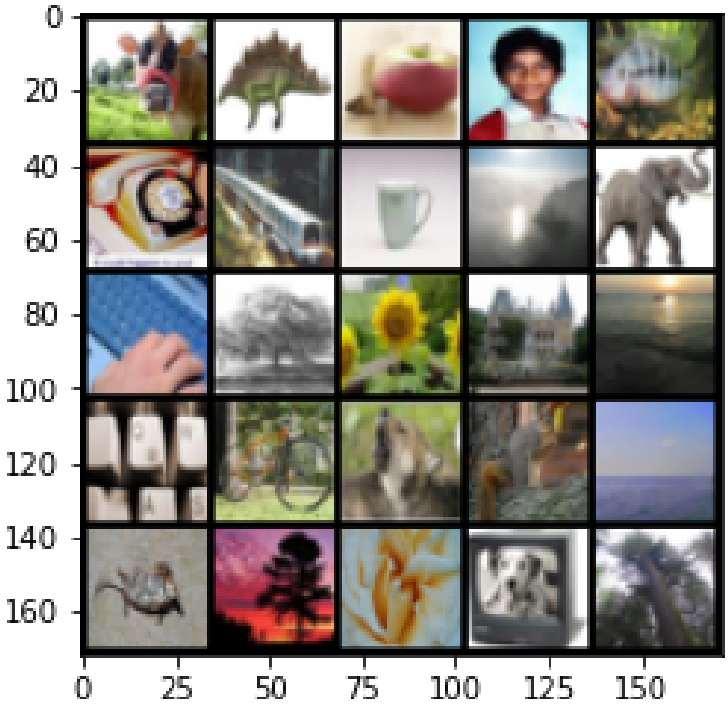


Figure B.3: Samples from the real CIFAR100 data

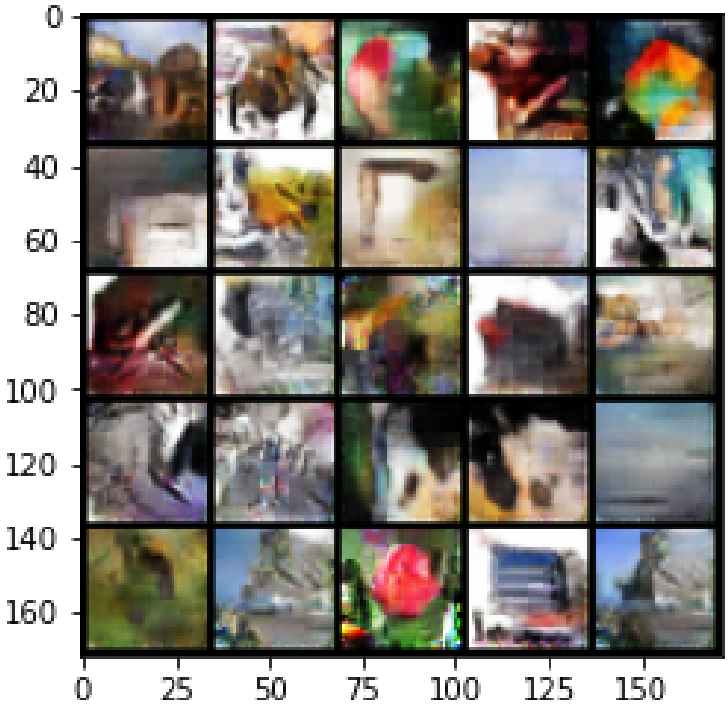


Figure B.4: Samples from the fake CIFAR100 data from Generator