Music Composition Using Deep Learning

Neil Bugeja

Supervisor: Mr Tony Spiteri Staines

May 2023

Submitted in partial fulfilment of the requirements for the degree of B.Sc. in IT (Hons)(Melit.).

L-Università ta' Malta
Faculty of Information & Communication Technology

Abstract

Computer-based music creation has witnessed remarkable advancements, with recent developments in deep learning and artificial intelligence. This study focuses on evaluating the performance of Music Composition (MC) software by analyzing the quality of the composed samples, specifically addressing the challenge of genre alignment. The objective is to establish an objective and reliable method for assessing the quality of the generated samples by incorporating a Music Genre Recognition (MGR) model.

To achieve this objective, a Convolutional Neural Network (CNN) model is utilized for genre classification, using 40 Mel Frequency Cepstral Coefficients (MFCCs) as input features. The CNN model achieves a test accuracy of 70%, enabling the objective determination of genre alignment in the composed samples.

The findings reveal when evaluating Jukebox, an open-source music composer developed by OpenAI, the adherence to the requested genre is influenced by the specified parameters. When the artist and genre align from the same grouping (ex: Artist: Mozart, Genre: Classical), approximately 2/3 of the compositions demonstrate a positive genre match. However, when the artist and genre oppose each other (ex: Artist: Mozart, Genre: Pop), only 42% of the samples match the requested genre, with 58% aligning with the artist's genre instead. Moreover, when lyrics from a different genre are provided, none of the test cases yields a positive genre match.

This study highlights the significance of genre alignment in assessing the quality of computer-generated music and emphasizes the importance of careful parameter selection. By objectively evaluating genre adherence, the study provides valuable insights into the strengths and limitations of MC software.

Acknowledgements

I would like to express my heartfelt appreciation to my mentor, Mr. Tony Spiteri Staines, whose unwavering guidance has been instrumental in shaping my thesis journey. Engaging in discussions with him about music and his continuous encouragement played a pivotal role in steering my focus towards this field.

Next, my gratitude extends to Mr Joseph Bonello, whose generosity and continual guidance greatly contributed to this project. He gave inspiration to the title of this project and his vast expertise and insights in this domain, particularly regarding MIR techniques, fundamentally shaped my approach and were critical to the project's artefact.

To my family, comprising of my mother, father, and sister Kim, along with my supportive friends, I express my deepest appreciation. Your unwavering belief in me and the reassurance you provided were a constant source of comfort during the difficult process of this project. Your trust in my abilities has always motivated me to persevere, regardless of the challenges I faced.

Particularly, a special note of thanks goes to my dear mum, whose constant encouragement and motivation were my driving force. Her willingness to lend an ear to my troubles, including the various challenges I encountered during the artefact development, offered a source of great solace and determination. Big shout out for proofreading my thesis with me till 5 in the morning.

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
FFT	Fast Fourier Transform
FMA	Free Music Archive
GAN	Generative Adversarial Networks
MC	Music Composition
MFCC	Mel-Frequency Cepstral Coefficients
MGR	Music Genre Recognition
MIDI	Musical Instrument Digital Interface
MIR	Music Information Retrieval
ML	Machine Learning
NLP	Natural Language Processing
NN	Neural Network
RNN	Recurrent Neural Networks
VAE	Variational autoencoders

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

On October 9th, 2021, the debut of Beethoven's Tenth Symphony, which had been previously incomplete, was held in Bonn to mark the 250th birthday of the composer. The media reported that artificial intelligence (AI) had finished the composition [1]. However, this oversimplifies the complex process that led to the completion of the work; the final piece was founded on fragments sketched by Beethoven, with the initial two movements assembled from these pieces by British musicologist and composer Barry Cooper in 1988. The concluding two movements were crafted with some assistance from AI tools, yet they necessitated significant input from human composers [2].

Applying computer-based technologies in music creation or aiding its creation is not a novel concept. Experiments to model music with neural networks (NN) started being developed in the 1980s and continued to the early 2000s [3] [4]. Eventually, with the growing development of technology and Deep Learning (DL), several studies attempted to model music using NNs [5]. Certain NN architectures that perform well in fields, such as Natural Language Processing (NLP) and Computer Vision (CV), were proven to also be successful in music composition.

1.2 Problem Statement

The subjective nature of music quality and genre classification makes it challenging to objectively evaluate the performance of Music Composition (MC) models. This study seeks to develop an objective evaluation methodology using the Music Genre Recognition (MGR) model to determine the genre alignment of the generated samples. By incorporating this approach, the study aims to establish a reliable framework for assessing the quality of computer-generated music compositions.

1.3 Structure

The paper is structured as follows: In the subsequent chapter, a brief background on the leading technologies relevant to this project will be explained. Chapter 3 will explain how these technologies are used, their popularity and their performance to determine the best approach to develop this project's artefact. Chapter 4 describes the methodology and the artefact's structure, while Chapter 5 explains how this was implemented. Finally, Chapter 6 presents the results from the artefact and a detailed report on the findings.

1.4 Research Questions

This study aims to answer the following research questions:

1. Is it possible to objectively analyse the performance of MC models by evaluating the genre alignment of the generated samples?
2. How do specific parameters, such as artist selection and genre combinations, influence the quality and genre alignment of the composed samples?

1.5 Aims and Objectives

The main goal of this study is to evaluate the performance of MC models by analysing the quality of the generated samples. The author's aim is to establish an objective and reliable method for assessing the quality of samples produced by MC software. As this concept is relatively new, the study aims to determine the usefulness of the generated results. Furthermore, during the analysis of the MC model's outcomes, the study will investigate whether specific parameters have a substantial impact on the quality of the generated songs.

1.6 Project Resources

Below please find resources related to the program. All MC samples generated during this project as available on google drive. Additionally, the full code is available on GitHub and a docker image is also available on DockerHub.

Github:
https://github.com/neilBugeja00/MusicGenreClassification

Jukebox composed songs:
drive.google.com/drive/folders/1FyqQlfWciNhiveshoayFRjlVEtEAh_Fl?usp=share_link

Docker:
https://hub.docker.com/repository/docker/neilbugeja00/genre_classifier/general

2 Background

2.1 Introduction

Deep Learning (DL), a specialized Machine Learning (ML) branch, has been increasingly evolving into a powerful technique. It has shown remarkable performance in numerous applications, including but not limited to Computer Vision (CV), Natural Language Processing (NLP), and Bio-informatics [6].

DL techniques have seen extensive use in Audio Signal Processing (ASP) and Music Signal Processing (MSP), resulting in numerous successful commercial utilities, like music recommendation systems [7]. Furthermore, an increase in research about Music DL has resulted in significant strides in two key areas: **Music Information Retrieval (MIR)** and **Music Composition (MC)** [8].

2.2 Music Information Retrieval (MIR)

2.2.1 Music Genre

Music Information Retrieval (MIR), also called Multimedia Information Retrieval, encompasses the process of extracting valuable insights from music data. MIR finds application in various fields, including classification, recognising genres, separating music sources, and identifying instruments [9].

Songs are characterized by various distinguishing features, which can be utilised to categorize them into specific genres. These genres, assigned by the music industry, are typically derived from attributes in the songs or the time period they became popular. However, given the inherent subjectivity in this classification process and the lack of universally accepted standards, most academic studies [10] [11] agree that these definitions are somewhat ambiguous [12]. This ambiguity gives rise to alternative and non-traditional classification systems such as automatic classification of music into genres using Deep Learning (DL).

2.3 Music Representation

Music analysis often involves extracting different features to understand the various aspects of the song. The choice of representation is critical because it directly impacts the quality and reliability of the analysis. It is debated whether it is better to represent music as an **audio** or **symbolic** format [13].

Symbolic representation is a format that represents music using symbols or notations, such as sheet music or MIDI (Musical Instrument Digital Interface) files. In symbolic representation, music is depicted as a sequence of discrete events, such as notes, chords, and rhythms. This format allows precise control over musical elements and can be easily edited and manipulated using software or hardware tools. Symbolic representation is commonly used in music composition, analysis, and notation [11].

On the other hand, audio representation captures the actual sound wave produced by musical instruments or voices. It is a continuous representation of music, capturing the nuances and subtleties of timbre, dynamics, and performance. Audio representation is commonly found in formats like WAV or MP3 files and is primarily used for playback and listening. It provides a more immersive and expressive experience of music but is less amenable to direct editing or manipulation of individual musical elements [14].

Both symbolic and audio representations have their distinct advantages and applications. Symbolic representation is well-suited for tasks like composition, arrangement, and analysis, where precise control over musical elements is essential. Audio representation, on the other hand, excels in capturing the emotional and aesthetic aspects of music and is primarily used for listening and performance purposes [13].

The primary focus of this literature is audio representation, hence, only this form of representation will be elaborated on in the following section.

2.3.1 Waveform

The most direct representation of raw audio is the waveform. This representation has an advantage as, considering it is untransformed, the audio is in its full initial resolution. Architectures that process this type of data are sometimes called end-to-end architectures. However, processing waveforms is very resource heavy, in terms of both processing and memory[12].

2.3.2 Time-Domain Representations

This type of representation involves direct analysis of the raw audio waveform. Converting audio signals into transformed representations results in data compression and higher-level information. However, this comes with the downside of potential information loss and the introduction of certain biases [13].

Frequency-Domain Representations

A Fast Fourier Transform (FFT) is frequently applied to the digital signal. This process generates a spectrum that maps various frequencies within the signal to their corresponding magnitudes. This transformation allows us to access the frequency domain, although it does so at the cost of losing specific time domain information. This is because the resulting spectrum provides a static snapshot of the signal, devoid of time-based details [12].

Mel Scale

Research has demonstrated that human perception of sound frequencies does not occur linearly. Our ability to detect differences in lower frequencies is more refined compared to higher frequencies. For instance, humans can easily discern a distinction between 500 and 1000 Hz, but may struggle to perceive a difference between 10,500 and 11,000 Hz [15].

In 1937, Stevens, Volkmann, and Newmann introduced a concept known as the Mel scale. This scale aimed to establish a unit of pitch wherein equal intervals in pitch sound equally perceptually distant to listeners. To convert frequencies to the Mel scale, a mathematical operation is applied [15].

Mel Spectrogram

A Mel spectrogram is a 2D representation of a signal where the x-axis is time, and the y-axis is the Mel Scale frequency. It can reflect human auditory perception, which makes it useful for tasks like genre classification or mood detection, which require understanding the song in a way that is similar to how a human listener would perceive it [16].

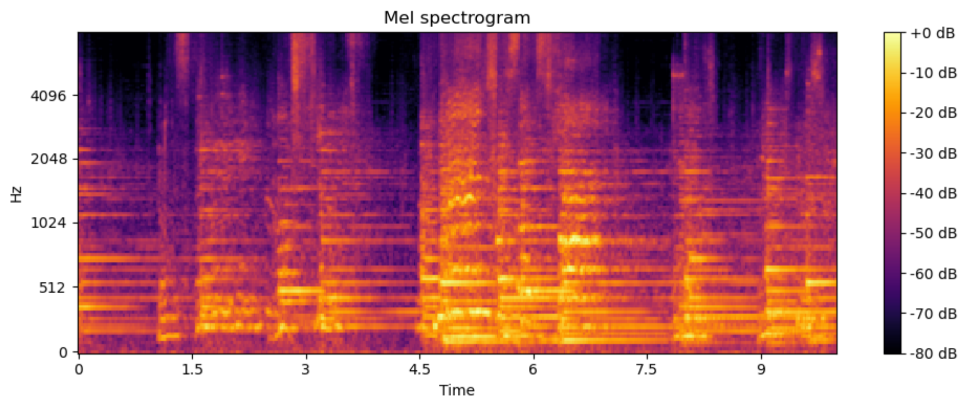


Figure 2.1 Mel Spectrogram generated using Librosa

Mel-frequency cepstral coefficient (MFCC)

To obtain MFCCs from an audio signal, several steps are followed. Firstly, like the Mel spectrogram, the audio signal's frequency values are converted from Hertz to the Mel scale, which better reflects the human perception of pitch. Next, the Mel representation of the audio signal is transformed using a logarithm, which helps approximate the perception of loudness. This logarithmic Mel representation is then subjected to a logarithmic magnitude calculation.

Finally, a Discrete Cosine Transform is applied to these logarithmic magnitude values. This transformation captures the spectral characteristics of the signal and results in the creation of MFCCs. These coefficients form a spectrum over Mel frequencies rather than time, providing valuable information about the audio signal's important features [17].

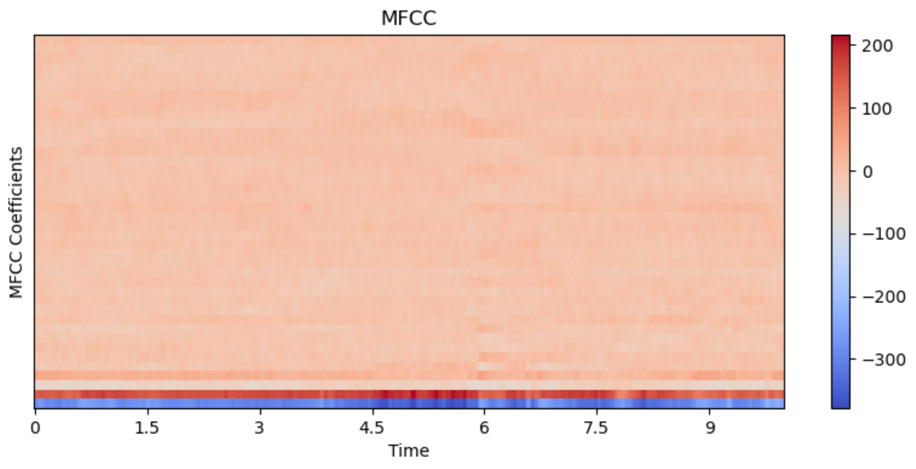


Figure 2.2 MFCC generated using Librosa

MFCC vs Mel Spectrogram

Both MFCC and Mel spectrograms are commonly used in ML due to their ability to capture information about how the power spectrum of a sound changes over time. MFCCs are beneficial for reducing the dimensionality of the data and de-correlating the features, which can be advantageous for many machine learning tasks. On the other hand, Mel spectrograms maintain a more detailed representation of the power spectrum, which may be beneficial for tasks requiring detailed frequency information.

2.4 Deep Learning Methods For MIR

The DL architectures that are most frequently employed for MIR tasks are: Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) [8].

2.4.1 Recurrent Neural Network (RNN)

An RNN excels in handling sequential data with varying lengths, a characteristic shared by audio data. One positive aspect of RNNs is their ability to process data while considering the context, leveraging the inherent sequential nature of the data [6]. A subset of RNNs, the Long Short Term Memory (LSTM) networks, has frequently been applied to MIR [18].

2.4.2 Convolutional Neural Network (CNN)

A CNN is a type of DL model that processes data with a grid-like topology [6]. It captures patterns from images and recognised spacial dependencies [12]. In the MIR field, where time-frequency data is often dealt with, CNNs are commonly employed to extract information from music data [8].

2.5 Music Composition Based On Deep Learning

DL-based MC makes use of the results that are produced by MIR methods [8]. The most used NN architectures that are providing positive results in the approach to MC are; Recurrent Neural Networks (RNNs), Generative Models, such as Variational Auto-Encoders (VAEs) or Generative Adversarial Networks (GANs), and NLP-based models such as Long Short-Term Memory (LSTM) or Transformers [5].

2.5.1 Neural Network (NN) Architecture Description

The simplest type of network is a feedforward network. This is an artificial neural network (ANN) where the connections between the nodes do not form a cycle. These networks are most commonly used in image-processing applications. CNNs are widely used as parts of more complex systems [2].

Recurrent Neural Network (RNN)

RNNs were once the most popular option when composing music. This was due to their ability to process variable-length sequences, which makes them ideal for dealing with time series and sequences [19].

2.5.2 Generative Models

Generative Adversarial Network (GAN)

Generative Adversarial Networks (GANs) are a class of generative models designed to produce new data instances that mimic the distribution of the training data. The architecture of a GAN consists of two interconnected networks: the generator, whose role is to fabricate outputs that mirror the training set, and the discriminator, tasked with discerning between authentic data and the creations of the generator. The generator's goal is to craft data that is so realistic that the discriminator cannot distinguish it from the real data, while the discriminator's mission is to improve its ability to accurately differentiate real from generated data as shown in 2.3 [2].

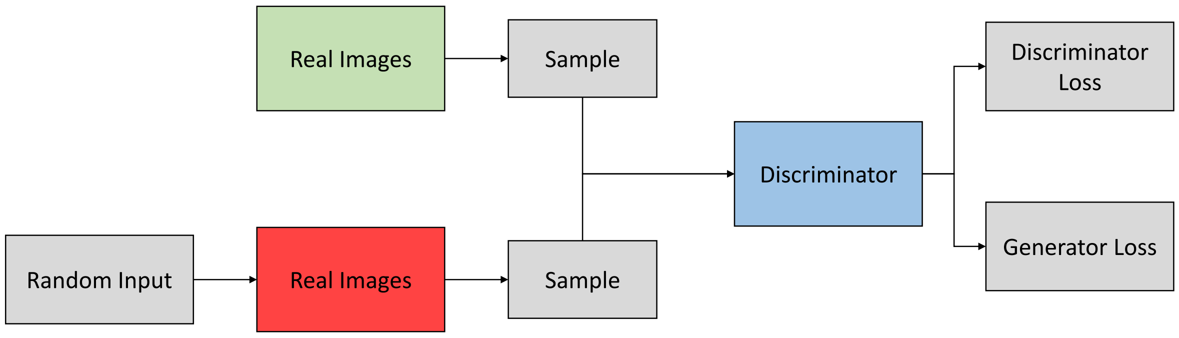


Figure 2.3 GAN model, reproduced from [20]

GANs are extensively utilised in automatic music composition tasks since they aim to produce melodies that are indistinguishable from those present in the training dataset [2].

Variational Autoencoders (VAE)

A VAE is a type of generative model. Its main goal is to learn a compressed representation of a given input data distribution, known as the latent space. A VAE compresses the input information and maps it to a lower-dimensional latent space (encoding) to reconstruct it as accurately as possible (decoding). As a result, VAEs excel at compressing information about a piece of music into a compact representation within the latent space, preserving the crucial characteristics while reducing the overall amount of information [2]. VAEs can also be part of Generative Adversarial Network (GAN) based generators [21].

These models have found extensive application in automatic music composition. For instance, Google's MusicVAE can learn long-term structures utilising a hierarchical VAE [22]. In addition, this model allows for custom adjustments in the latent space.

2.5.3 NLP Models

Transformers

Transformers are built upon the fundamental concept of self-attention, which involves selectively assigning importance to different input data components. While transformers can handle sequential input data, they do not necessarily process it in a strict order [8].

2.6 Conclusion

This section explained some effective techniques and technology related to MGR and MC. The following chapter will examine results from academic studies that applied these technologies and discuss their findings to determine the best technologies for this project's artefact.

3 Literature Review

3.1 Introduction

In this section, results determined from academic literature on the technologies discussed in the previous section will be presented. This literature is split in two. The first section discusses music composition (MC) models, particularly how they work and their performance. The second covers Music Genre Recognition (MGR) models.

3.2 Music Composition Models

In their paper, authors Civit, Civit-Masot, Cuadrado and Escalona (2022) [2], delved deep into the field of MC models, partly to determine which AI-based techniques are most prevalent in this field. The following table and bar chart is adapted from their work.

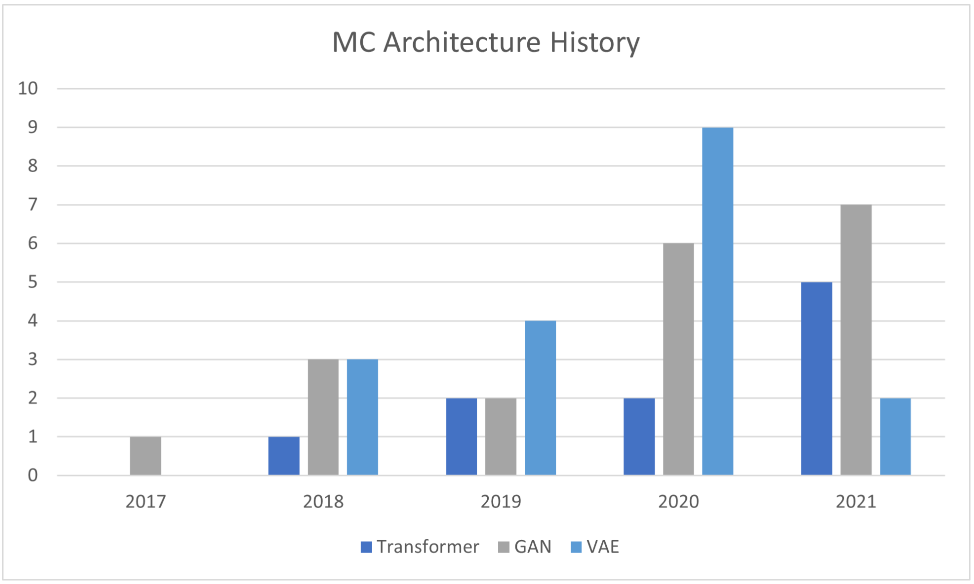


Figure 3.1 Music Composition Architecture History, adapted from [2]

It can be seen that as of 2021, there has been a steady increase in Transformer and GAN-based architectures. There is evidence that this trend is ongoing, with Google releasing their new MusicLM in 2023, which makes use of the Transformer architecture [23].

3.3 Comparing Different System Architecture Performance

Everyone has a different opinion on a single piece of music, which makes it difficult to grade music samples objectively. In an effort to accomplish this in the best way possible, during the development of Jukebox, samples composed by the model were manually evaluated based on their coherence, musicality, diversity and novelty [24].

Authors Civit, Civit-Masot, Cuadrado and Escalona (2022) [2], analysed models by comparing the consistency of the musical structure and the motivic development that occurred during the pieces. The authors evaluated the top-performing models as of 2022 and determined that works based on transformer architectures, such as Magenta Transformer [25], MuseNet [26] or PIA [27] initially appeared to offer the best performance but this performance is not consistent. However, the launch of Google's MusicLN in 2023 may serve to improve the consistency of Transformer based architectures. MusicLN composes a piece of music from a text description and is also based on a transformer architecture, meaning continuous development is done using this technique [23].

Among these Transformation models, Jukebox was also graded by the authors. They noted that it is quite unique in its ability to work directly with audio music (instead of symbolic music) and to compose a song that includes music, rhythm and lyrics. However, it was criticised that the output was in .wav files, which contain artefacts that are recognisable and are therefore not usable in a professional setting [2]. The authors [2] also noted that systems based on a GAN or LSTM offer "good long-term structure but may not develop such complex or surprising musical content as those of the transformer-based"[2].

Researchers Hernandez-Olivan and Beltran (2021) [5] found that there is no specific NN architecture that can be considered superior to its peers. This is because different NNs perform better in different situations and some are better than others at generating certain types of output. That being said, Transformers and Generative models seem to have an advantage over others and some of the best open-source music composers use these NN architectures [5]. Some of these music composers include MuseGAN and Jukebox.

3.3.1 MuseGAN

MuseGAN, an application of Generative Adversarial Networks (GANs) specifically designed to generate multi-track music, is a product of OpenAI. What follows is a summary of their paper titled 'MuseGAN: Multi-Track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment' (2019) [28].

MuseGAN takes the concept of a GAN and applies it to music. The generator creates new pieces of music, and the discriminator evaluates them. But unlike many other GANs, MuseGAN is capable of generating music with multiple tracks, like a song with separate tracks for drums, bass, and melody [28].

The multi-track capability of MuseGAN is a significant advancement over other music-generating GANs. It is designed to capture the interdependencies between different tracks in a piece of music, which can lead to more coherent and musically interesting compositions [28].

As with any GAN, the quality of the output from MuseGAN depends heavily on the training data it is given. If it is trained on a wide variety of music, it can potentially generate a wide variety of new compositions. However, it may struggle to reproduce specific musical styles or structures if they are not well-represented in the training data [28].

3.3.2 Jukebox

Jukebox is a generative model for MC developed by OpenAI. What follows is a concise review of the academic publication, 'Jukebox: A Generative Model for Music' (2022) [24].

Early versions of generative models utilised symbolic data, thereby simplifying the issue at hand. However, using symbolic data constrains the music that can be composed to being a specific sequence of notes and a fixed set of instruments to render with. Jukebox pushes research into models using audio data to provide better results [24].

Jukebox uses a VQ-VAE architecture to compress audio in a way that retains the maximum amount of musical information. An autoregressive Sparse Transformer, a variant of the Transformer model, is also employed. The model is trained over the compressed space using maximum likelihood estimation, a common statistical method used for model training [24].

The aim here is to create a model that can generate sequences based on previous data in the sequence - which is a characteristic of autoregressive models. The sequences in this context would be musical notes or pieces of audio data [24].

When creating music with Jukebox, users are presented with a plethora of choices. They can utilise the pre-generated models to craft their compositions, selecting from a diverse collection of artists and genres, and freely combining them to their liking.

Additionally, users have the option to input a piece of music as a sample for the model, which will subsequently compose music in a comparable style [29].

3.4 Music Genre Recognition (MGR)

3.4.1 Data Preperation

Buttigieg Vella's paper [12] presents a variety of MGR projects and their results. Information from his paper is presented in a table, created by the author:

Table 3.1 Music Genre Recognition Studies

Author	Sample Length	File Type	Dataset	Num. Tracks
Boxler (2020) [30]	30 secs	mp3	FMA Medium	106,574
Murauer & Specht (2018) [31]	30 secs	mp3	FMA Medium	60,000
Bisharad & Laskar (2019) [32]	3 secs	mp3	GTZAN	9,991
Bisharad & Laskar (2019) [32]	29 secs	mp3	MagnaTa-gATune	25,863
Buttigieg Vella (2022) [12]	30 secs	mp3	FMA Medium	5,000

As shown in table 3.1, the majority of authors made use of 30 seconds of song length. However, Bishard & Laskar [32] split the 30-second samples found in the GTZAN dataset into ten seconds in an effort to inflate the dataset size. All authors used a file type of MP3, with the FMA Medium dataset being the most popular source of song samples.

Bishard and Laskar [32], and Boxler [30] both performed a 70%, 15% and 15% split for training, validation and testing, respectively. Buttigieg Vella [12] performed a split of 60%, 15% and 25% for his data.

3.5 MGR Machine Learning Models

3.5.1 CNN

CNN is a prevalent architecture in MGR applications. For instance, Boxler [30] applied CNNs among other models, incorporating batch normalisation and the Exponential Learning Unit function in each convolutional layer, achieving a loss of 1.42. Similarly, Murauer and Specht [31] implemented a CNN along with other models on an image dataset, registering the highest loss of 1.65. However, they attribute this outcome to hardware constraints.

Buttigieg Vella [12] ran tests to determine the best NN for MGR. The author tested ANN, CNN, RNN and GBM. His results determined that the CNN model performed the best, achieving a loss of 0.942 and an accuracy and F1 score of 0.69. Additionally, several other papers [33] [34] have found that the CNN model performs well in other MIR tasks such as music tagging or instrument recognition .

Lastly, it has been shown by Senac, Pellegrini, Mouret and Pinquier [35], that the CNN performs better if a spectrogram is given as an input, as opposed to enhanced autocorrelation.

3.5.2 Gradient Boosting Machines

In their respective studies, Boxler [30], Murauer & Specht [31], and Buttigieg Vella [12] all identified Gradient Boosting Machines (GBMs), particularly XGBoost, as one of their top-performing models. A GBM is a powerful machine learning algorithm primarily employed for tackling regression and classification problems. It utilises the strategy of ensemble learning, where additional models are incrementally introduced to amend the inaccuracies committed by the preceding models. The integration of these models continues sequentially until no substantial enhancements can be achieved. The underlying principle involves constructing new base learners, or weak models, which are most significantly correlated with the negative gradient of the loss function linked to the complete ensemble [36].

Boxler [30] made use of 3 GBMs, LightGBN, CatBoost and XGBoost, with the latter performing the best with a log loss of 0.92. The XGboost of Murauer & Specht [31] reached a log loss of 0.82, and Buttigieg Vella's [12] had 0.987.

3.6 Conclusion

As evident from the examination of relevant literature, evaluating the performance of an Music Composition (MC) model proves challenging due to the inherent subjectivity surrounding the definition of a ”good” song. To address this issue, a novel approach was adopted to assess the quality of the composed samples, using the diverse Music Information Retrieval (MIR) techniques discussed in the background and literature review sections. Specifically, the decision was made to evaluate the samples based on their genre classification. Given that Jukebox empowers users to specify explicitly the genre of the musical segment they intend to compose, it emerged as the perfect representative for conducting this evaluation.

A CNN will be used to create a Music Genre Recognition (MGR) model. This decision was taken because, as shown in the current literature, the CNN model is one of the best options to tackle this problem. In determining the type of music data inputted into the model, it was decided that two model versions should be made, one using MFCCs and the other using Mel spectrograms. This is due to the similarities between the two and to determine whether the Mel spectrogram’s more detailed representation influences that model’s accuracy in any way.

4 Methodology

4.1 Introduction

This project consists of three distinct sections. First, a model is created that is capable of identifying the genre of an audio file. Following that, music is composed using OpenAI's Jukebox, with a request of the genre. Finally, the model predicts if the piece of music belongs to the requested genre.

4.2 Evaluation Method

Generated Music Evaluation

Two methods were established to assess the quality of the samples produced by Jukebox. An objective evaluation was carried out using the genre classification model to verify whether the generated samples align with the requested genre, thereby assessing the capability of Jukebox to accurately generate songs as specified. Additionally, a subjective evaluation was also carried out where the author assigned a rating ranging from zero to seven to indicate the quality of the given sample.

Genre Classification Model Evaluation

It is equally crucial to accurately assess the music genre recognition (MGR) algorithm that has been developed. Inaccurate evaluation could lead to erroneous conclusions in the study findings. Accuracy, loss and f1 score were used to evaluate the performance of the genre classification model.

4.3 Classification Model

Upon the generation of music, it became necessary to develop a method for evaluating its characteristics. Initially, the assessment was aimed at determining the quality of the music produced. However, further analysis revealed that this was not a feasible approach, due to the subjectivity and time restraints involved. As discussed in 3.3 Jukebox was evaluated manually by its creators.

The evaluation process was adapted to focus on the genre of the generated music, ensuring its alignment with the specific requirements set forth by the user. For instance, if a piece of classical music was requested, the evaluation would confirm that the program indeed delivered a composition that adhered to the classical genre.

4.3.1 Dataset

There are a number of datasets available for MIR purposes. These datasets' properties were evaluated to choose which would best suit this project. These requirements included the fact that the dataset's content had to be of appropriate audio quality, that it contained regularly used music genres, and that the dataset was mainly balanced.

It was decided that the best option was to use a modified version of the FMA dataset. The full dataset contains audio from 106,575 tracks performed by 16,341 artists from 161 genres, with the majority having a sampling rate of 44.1kHz and a bit rate of 320 kbit/s [37]. When compared to other datasets, its high quantity and quality of audio, as well as the range of genres it contains, made it an appealing choice for this study.

From the full dataset, the medium FMA subset was used. This subset contains 25,000 tracks, all of which are 30 seconds in length, and contains 16 unbalanced genres. Only the audio files and genre were used in this project, ignoring all metadata, such as track, artist, and album. Necessary cleaning had to be done as certain tracks were less than 30 seconds in length. These tracks can be found on the 'Wiki' section of the FMA datasets GitHub repository. The below figure 4.1 is a graphical representation of the full dataset.

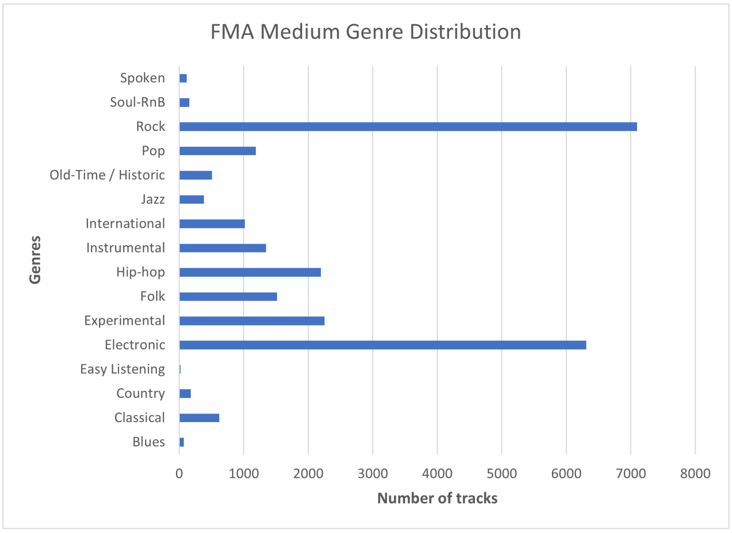


Figure 4.1 FMA Medium Data Distribution

After the cleaning process, the audio files were stored in folders relating to their genre. Certain genres, such as historic or international, were ignored as well as genres with a low amount of songs. Additionally, in order to maintain balance in the dataset, many songs had to be eliminated as genres, such as classical had 619 samples whilst rock had 7000. The genres that remain in the final version are classical, folk, hip-hop, jazz, pop and rock.

Each genre in the dataset is represented by 619 samples, except for Jazz, which has only 384 samples. This results in a total dataset size of 3,479 samples, each comprising 30-second songs across six different genres. The accompanying Figure 4.2 visually illustrates the distribution of the modified data.

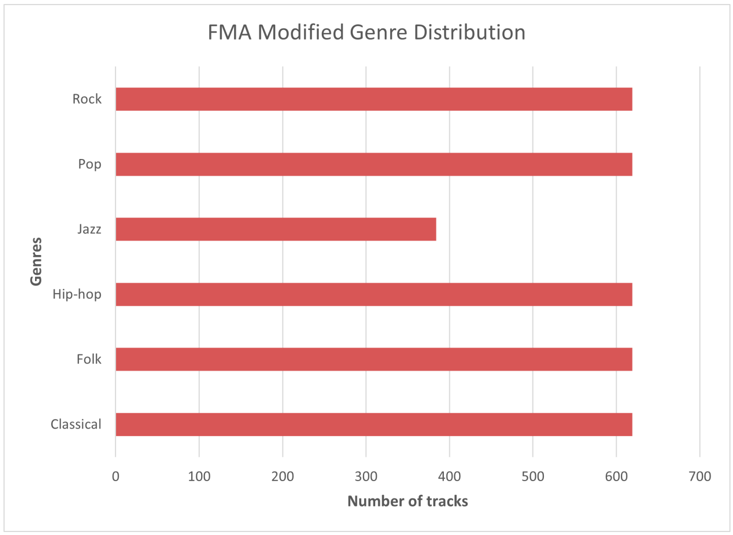


Figure 4.2 FMA Modified Data Distribution

4.3.2 Data Manipulation

It was decided that the songs in the dataset will be broken into various snippets to increase the number of training data. While the data could be snipped every three seconds, determining the genre of a song in three seconds is difficult. As a result, it was decided that each sample should be ten seconds long as this had no negative effect on performance and would fulfil the goal of increasing the dataset. This means that the final dataset consists of 10,437 ten second song snippets.

4.3.3 Feature Extraction

Pre-processing was executed on the given dataset by extracting 40 Mel Frequency Cepstral Coefficients (MFCCs) every 2048 samples, while also downsampling the snippets to 22.05Khz. The chosen acoustic features for this analysis were MFCCs and mel-spectrograms, as they are widely regarded as the most effective acoustic features. Two similar models were developed, each utilising one of these Music Information Retrieval (MIR) techniques, and their respective performances were evaluated. Additionally, the dataset was divided into training, validation, and testing sets, with a distribution of 70%, 15%, and 15%, respectively.

4.3.4 Machine Learning Model

The ML learning model chosen for this project is a CNN. This was chosen due to the numerous positive results generated by models of this type as explained in the previous chapter.

In adherence to the experimental protocol, the dataset will be partitioned into training, validation, and testing sets, and each model will be subjected to a standardized training procedure. The training set, which will undergo shuffling, will be utilised to fit the model for a specified number of epochs, following which the model's performance will be tested against the validation set after every epoch. Once the model has been adequately trained, its architecture and parameters will be stored, and the model will be tested on the previously unseen testing set. The objective of the testing phase is to classify song tracks into the pre-defined genres on which the model has been trained. The model's predictions will be contrasted with the actual genre labels, and the results will be subjected to rigorous analysis.

4.4 Jukebox

Upon installation of Jukebox, users gain access to three distinct models: the 1B Lyrics, 5B, and 5B Lyrics. These models primarily differ in the resources required for their operation and the volume of data utilised during their training. The 1B Lyrics model underwent training using a dataset comprised of one billion words from various sources. Conversely, the 5B models were trained with five billion words. The 5B and 5B Lyrics models differ in their function; the latter generates songs with lyrics, while the former solely produces the accompanying music [24].

Several critical decisions were made during the utilisation of Jukebox. The initial step involved determining whether any of the aforementioned models would be employed or if a new model would be trained using an alternative dataset. Subsequently, it was necessary to decide whether lyrics should be incorporated in the samples.

The MAESTRO dataset (encompassing 200 hours of piano performances) and the Classical Music MIDI dataset were both considered as potential sources for training data. However, it was discovered that the 1B model had already learned from the MAESTRO dataset, and utilising the Classical Music MIDI dataset would hinder the generation of songs from other genres. The GTZAN dataset proved unsuitable for learning multiple genres due to an insufficient number of samples in each genre for the comprehensive composition of original music pieces. Therefore, the decision was made to employ the Jukebox models. With the availability of multiple GPUs for rent through Vast.ai, the larger 5B Lyrics models were chosen as the main evaluation models.

Moreover, it was necessary to determine whether lyrics should be integrated into the samples. The inclusion of lyrics was ultimately decided upon, as they are a fundamental aspect of Jukebox and their performance warranted evaluation.

4.5 Genre Detection

This section synthesises the work conducted in the previous two chapters. The highest-performing CNN model, responsible for identifying the genre of a music snippet, is loaded alongside the Jukebox samples. Subsequently, the model predicts the genre of the song. Users can then assess the accuracy of the genre predictions through the calculation of an accuracy percentage.

4.6 Development Methodology

During the artefact's development, the prototyping model was utilised. This method entails creating an initial prototype, and then examining and refining it until a satisfactory result was reached. This method was ideal for this project as multiple modifications were made to the model depending on the results generated, and on how certain hyperparameters were optimised.

4.7 Conclusion

Figure 4.3 has been designed to explain the interconnected operations of the three programs described above.

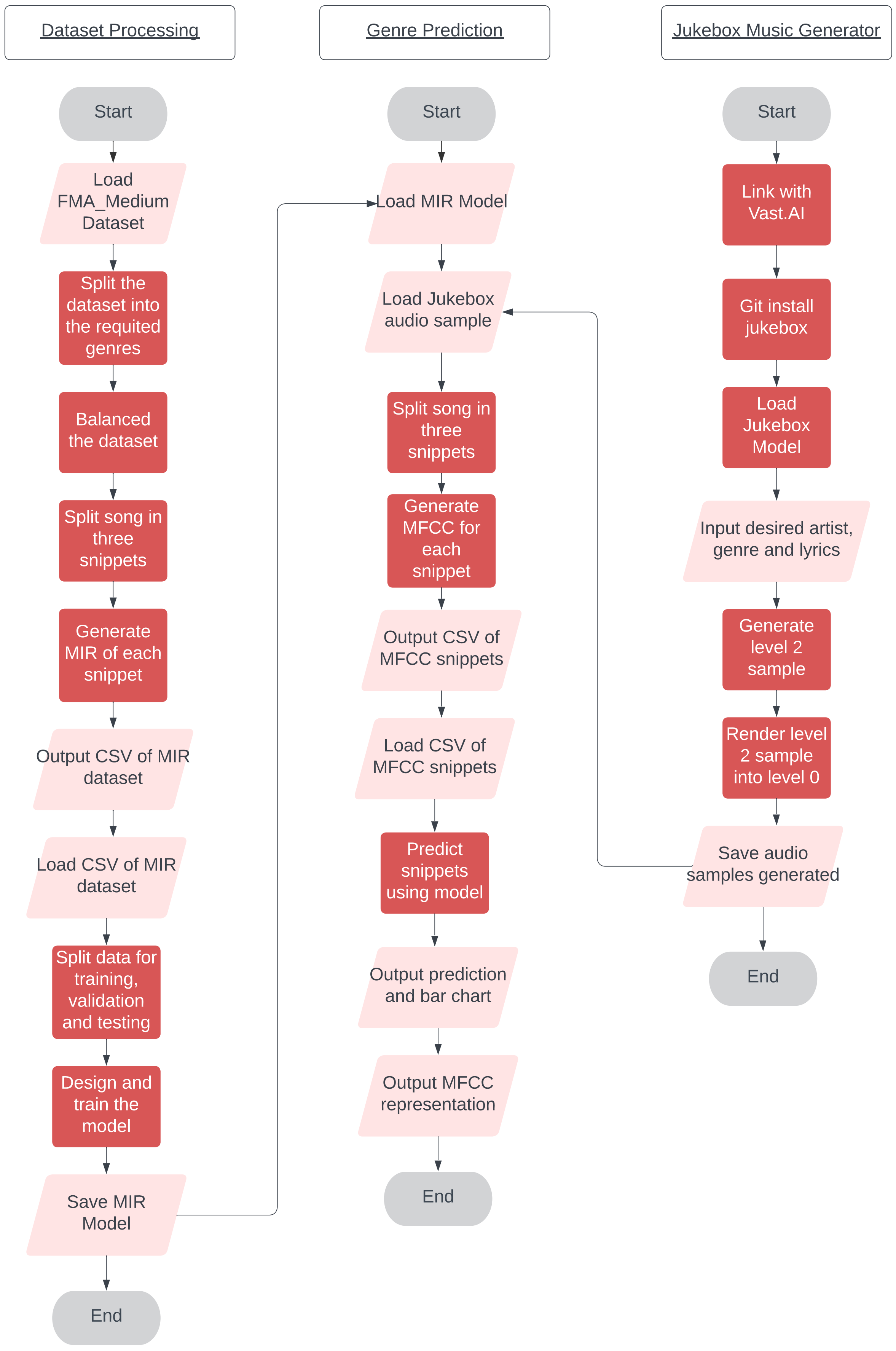

Dataset Processing
Start
Load FMA_Medium Dataset
Split the dataset into the requited genres
Balanced the dataset
Split song in three snippets
Generate MIR of each snippet
Output CSV of MIR dataset
Load CSV of MIR dataset
Split data for training, validation and testing
Design and train the model
Save MIR Model
End
Genre Prediction
Start
Load MIR Model
Load Jukebox audio sample
Split song in three snippets
Generate MFCC for each snippet
Output CSV of MFCC snippets
Load CSV of MFCC snippets
Predict snippets using model
Output prediction and bar chart
Output MFCC representation
End
Jukebox Music Generator
Start
Link with Vast.AI
Git install jukebox
Load Jukebox Model
Input desired artist, genre and lyrics
Generate level 2 sample
Render level 2 sample into level 0
Save audio samples generated
End

Figure 4.3 Flowchart

5 Implementation

5.1 Introduction

This chapter will describe the implementation of the various programs described in the previous chapter. Details such as the system architecture, the data flow through the system, optimization, testing and evaluation will be discussed.

5.2 System Architecture

The composition of the artefact is characterised by the integration of five distinct modules. The initial module is focused on data preparation, which includes cleaning and categorising the dataset. The second module handles the extraction of required characteristics (MFCCs and Mel Spectrograms). The following module describes the process of developing, optimising, testing and evaluating the Convolutional Neural Network (CNN) model.

The fourth module acts independently from the others and includes the use of OpenAI's Jukebox to generate original musical compositions in the form of three 30-second WAV files every iteration. Finally, the concluding module involves the integration of the CNN model and the Jukebox music samples to determine the genre corresponding to each respective sample.

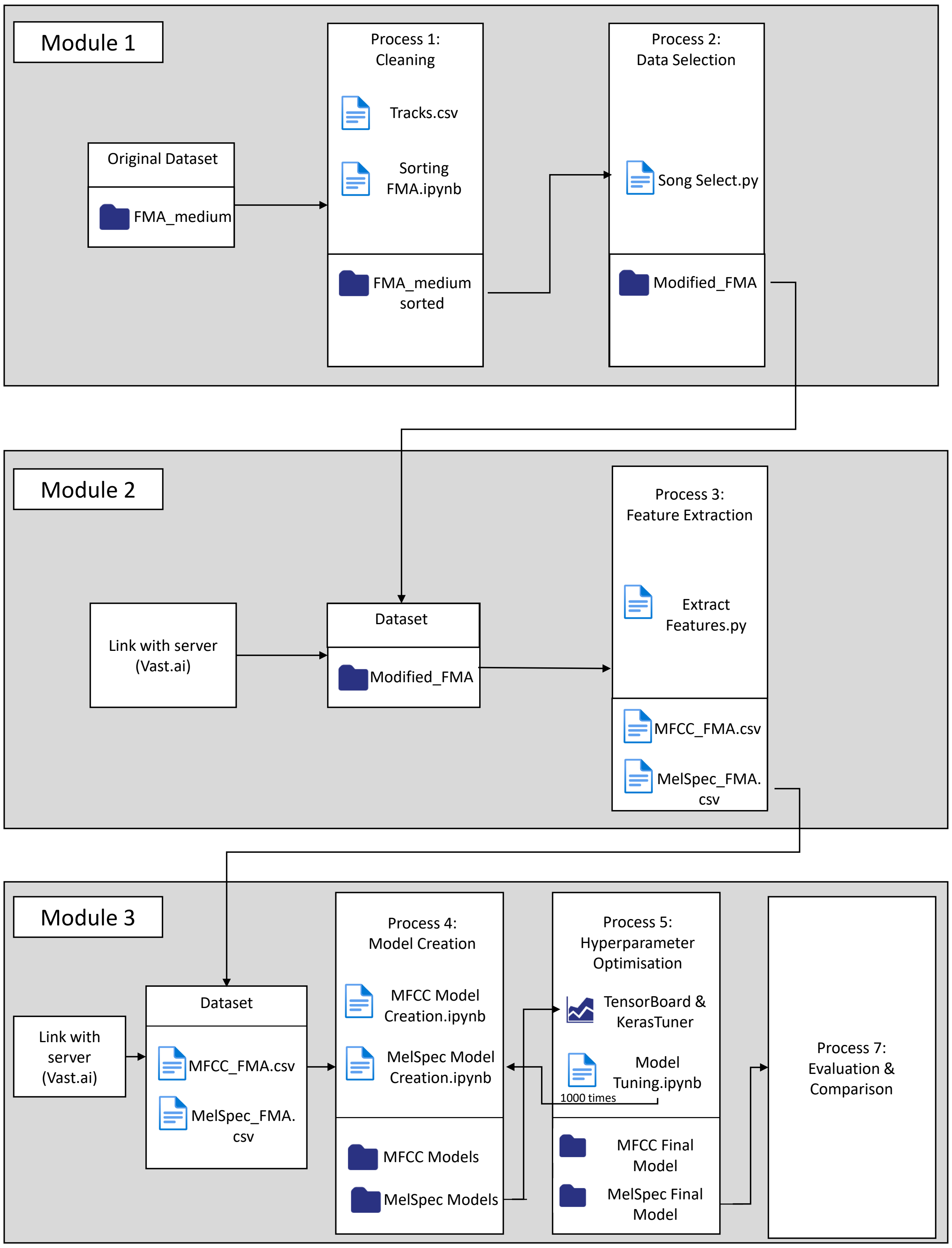


Figure 5.1 System Architecture Diagram 1, adapted from [12]

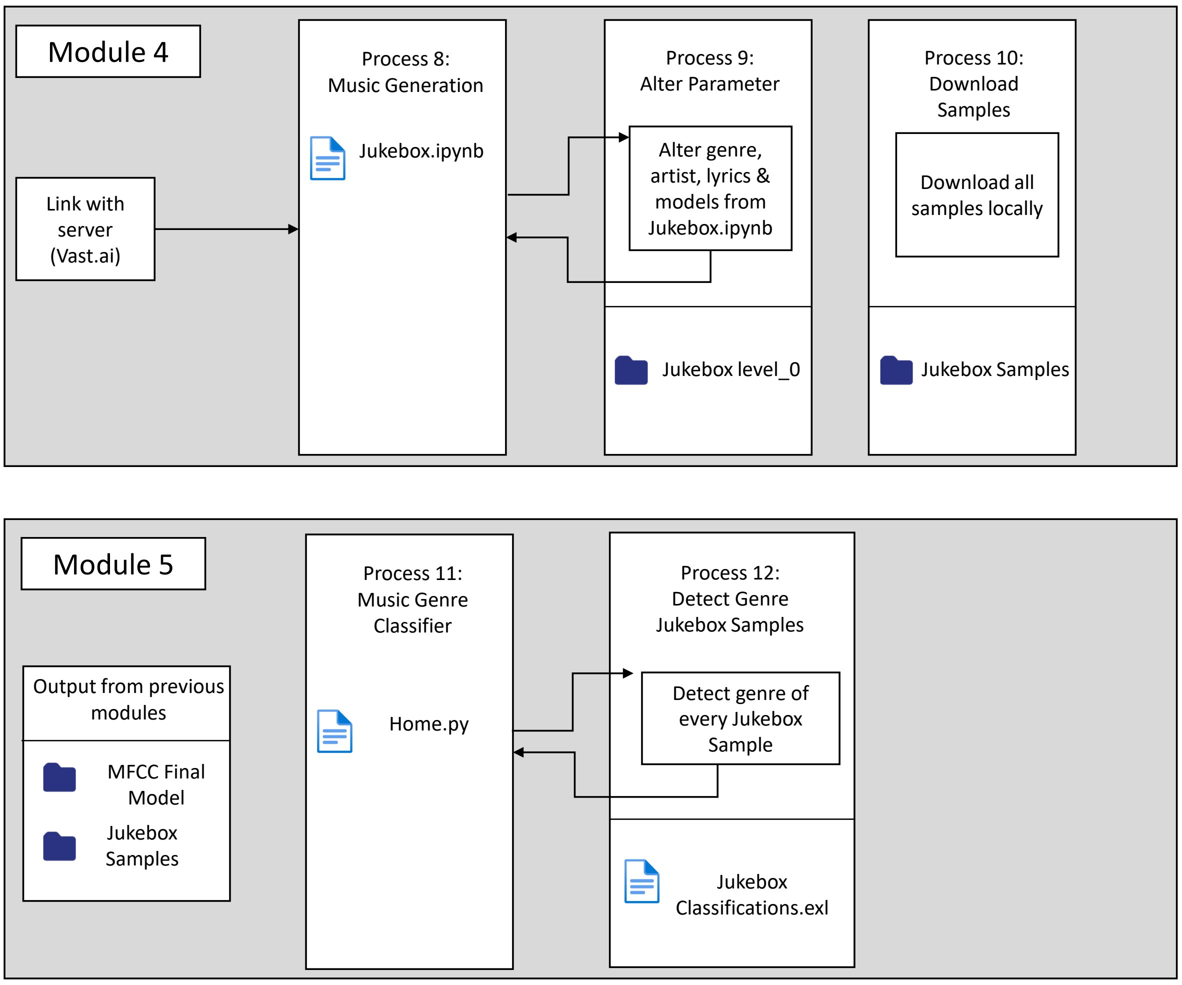

Module 4
Link with server (Vast.ai)
Process 8: Music Generation
Jukebox.ipynb
Process 9: Alter Parameter
Alter genre, artist, lyrics & models from Jukebox.ipynb
Jukebox level_0
Process 10: Download Samples
Download all samples locally
Jukebox Samples
Module 5
Output from previous modules
MFCC Final Model
Jukebox Samples
Process 11: Music Genre Classifier
Home.py
Process 12: Detect Genre Jukebox Samples
Detect genre of every Jukebox Sample
Jukebox Classifications.exl

Figure 5.2 System Architecture Diagram 2

5.2.1 Module 1 - Dataset

The medium subset of the FMA dataset, along with its associated metadata folder, were downloaded locally from the Github repository. Following that, data was sorted in folders corresponding to their genre using a python script and the 'tracks.csv' file provided in the metadata. Finally, songs of insufficient length (as determined from the wiki page) were removed.

After reviewing the sorted dataset, a decision had to be taken on which genres to preserve. This conclusion was reached based on the number of songs in the genre and its popularity (as explained in section: 4.3.1).

The new dataset, henceforth referred to as "modified_FMA_dataset", includes all of the samples from which MIR features will be retrieved.

5.2.2 Module 2 - Feature Extraction

Since certain modules are quite compute-intensive, Vast.ai was used. Vast.ai is a cloud computing service that focuses on computing resources. Physical resources can be rented out to perform computing-intensive tasks on them [38].

An instance is set up using the TensorFlow docker on Vast.ai and the *modified_FMA_dataset* is uploaded to it. The goal of this module is to extract MIR features from the dataset and save them as a CSV file.

A python program was developed to accomplish this by using the **Librosa** and **Pandas** libraries, to extract any MIR features and save them in a CSV file. Another important feature of this program is that each song sample was split into snippets of 10 seconds each and the features were extracted from that snippet. This was done to further increase the amount of data present.

Using Librosa, the following code snippets were used to extract MFCCs and Mel Spectrograms respectively from the audio samples:

```
mfcc = librosa.feature.mfcc(y=song[start_sample:end_sample],
                            sr=sr,
                            n_mfcc=40)

mel_spec = librosa.feature.melspectrogram(
                y=song[start_sample:end_sample],
                sr=sr)
```

•y: Requires the song as an input. In this case, the code is in a for loop that iterates through the entire dataset. The *song* variable is the current song, and the *start* and *end* samples are to determine if the feature of the first, second or third snippet will be generated.

•sr : A *sample rate* variable has already been declared as 22050

•n_mfcc: The number of MFCCs that are generated per snippet, in this case, 40.

The outputs *mfcc* and *mel_spec* are saved in a CSV file along with the genre of the song. The first column contains the genre of the song, and the second column contains its corresponding feature, represented as a 3-dimensional ndarray list. The final output is two *mfcc_fma.csv* and *melSpec_fma.csv* files.

5.2.3 Module 3 - Model Creation

An ipynb program was developed using Jupyter Notebook. The program reads two CSV files containing MFCCs and Mel Spectrograms of the dataset. These files will be used as data inputs for the model, albeit only once at a time. The model is compatible with both file types which eliminates the necessity for modification.

The chosen CSV file is passed through a stratified split of 70/15/15 for training, validation and testing data respectively. The model is run iteratively 1000 times using **KerasTuner** to optimise its hyper-parameters as well as the architecture such as the layers and learning rate used. The CNN model was built using the **Keras** library found within **Tensorflow**. Since this model is being run on Vast.ai, the built-in *TensorFlow* docker was used which correctly configures the environment.

After each iteration, a graph of the training, validation and loss accuracy was plotted on **Tensorboard**, to determine the hyper-parameters of the best performing neural networks. Additional manual tweaking was done and the testing accuracy of the best-performing model was determined.

5.2.4 Module 4 - Music Composition

This module acts independently of the previous modules. Significant computational resources are required, thus necessitating the connection to a new instance on Vast.ai.

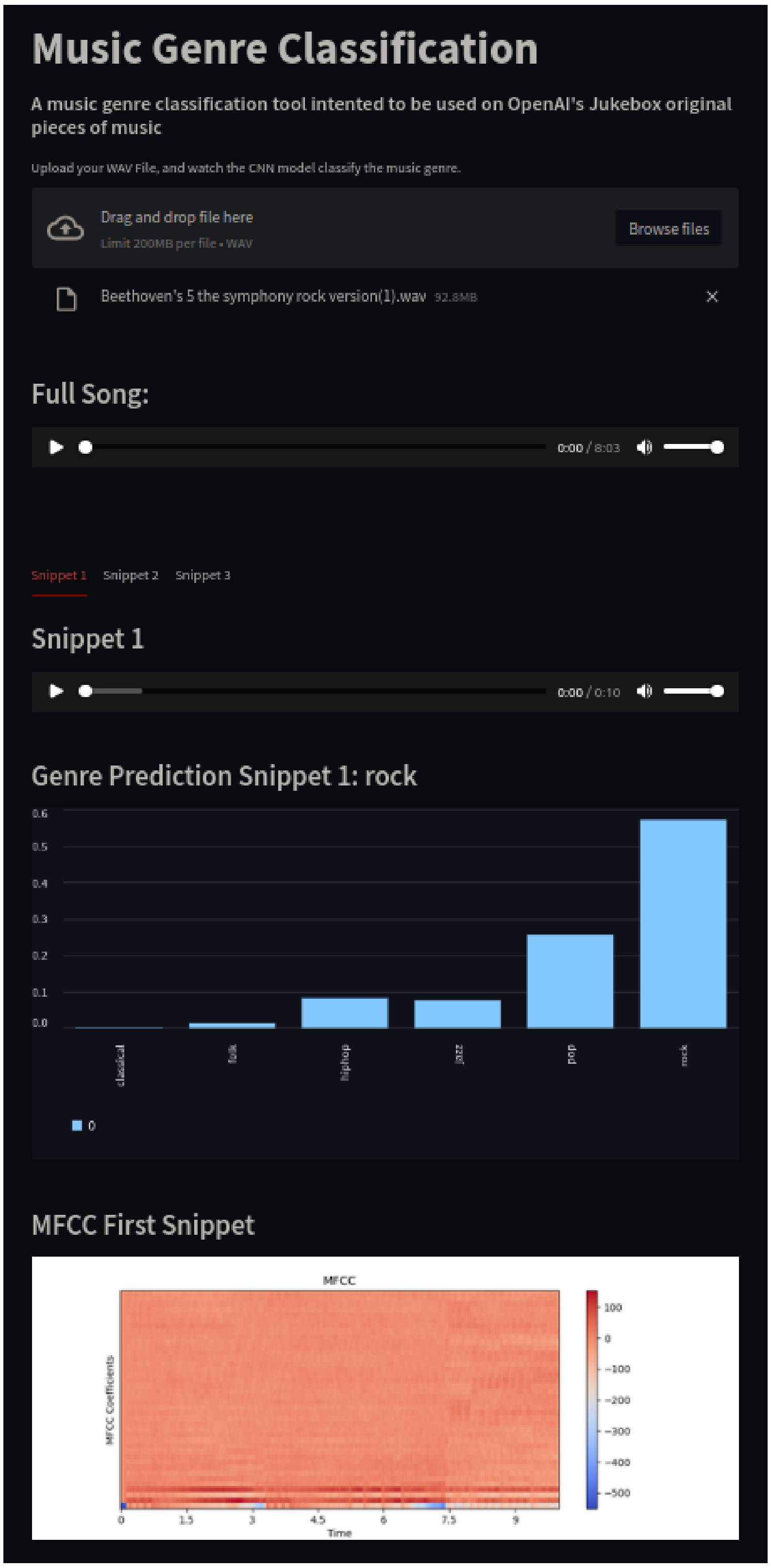
This module's music-generating utility is based on OpenAI's Jukebox, which is installed via their git repository. The operational backbone of this system is a pre-downloaded Jupyter notebook, which serves as the required code interface for Jukebox.

As a setup and hyperparameter optimisation, the collab notebook [39] was used. The temperature is set to 0.99 and the model, genre, artist, and lyrics are all changed during every iteration depending on the test case. Furthermore, the configuration allows for the generation of three samples every iteration, with each sample lasting 30 seconds.

Jukebox initially provides a level 2 sample, which is grainy and lacking in clarity. A significant amount of processing power is required to render the sample into its final form, level 0. This method is repeated several times, resulting in songs inspired by various artists, genres and employing different models. After each iteration, the Jukebox samples are saved locally to be used in the next module.

5.2.5 Module 5 - Music Genre Classifier

This module represents the program's final solution, leveraging the solutions derived from modules 3 and 4 to determine the genre of music samples generated via Jukebox. Streamlit is used to create a user-friendly web experience. The user is prompted to input a 30-second WAV song sample, which the application segments into three distinct 10-second snippets. For each snippet, the model predicts its genre and a visual representation of its confidence is provided via a bar chart. The program also provides the MFCCs of the snippets and allows the user to listen to the snippet.



Music Genre Classification

A music genre classification tool intented to be used on OpenAI's Jukebox original pieces of music

Upload your WAV File, and watch the CNN model classify the music genre.

Beethoven's 5 the symphony rock version(1).wav 92.8MB

Full Song:

Snippet 1

Genre Prediction Snippet 1: rock

MFCC First Snippet

Figure 5.3 Final product in action

5.3 Environment Settings

Connecting to Vast.ai, to rent out the hardware resources for this project was a necessity. Vast.ai provides its users with various dockers which allow them to easily set up their environment. When composing music using Jukebox, the author used a Pytorch docker, and only a Conda installation of Mpi4py was required to complete the environment.

For the MGR model, the TensorFlow docker was required. In this environment, multiple libraries were also installed, such as Keras, Matplotlib and Librosa. These were necessary for optimisation, plotting, and generating the MIR features required.

On the local machine, a Conda environment, python version 3.9.16, was set up. The Streamlit library was installed to run the final application (module 5), along with Pandas, Pydub, TensorFlow, Librosa and Matplotlib.

5.4 Hyperparameter Optimisation

KerasTuner was used to optimise the model's hyperparameters. The hyperparameters include the number of units in each layer, the activation function, whether to use dropout, the number of layers, and the learning rate.

After defining the function, a RandomSearch tuner from the Keras Tuner library is created. This tuner will automatically search for the best hyperparameters for the model by training and evaluating the model multiple times, each time with different hyperparameters. It aims to optimise the validation accuracy. The tuner will run up to 1000 trials and will run each trial twice.

```
def build_model(hp):
    units = hp.Int("units", min_value=32, max_value=128, step=32)
    activation = hp.Choice("activation", ["relu", "tanh"])
    dropout = hp.Boolean("dropout")
    layers = hp.Int("n_layers", min_value=1, max_value=3, step=1)
    lr = hp.Float("lr", min_value=1e-4, max_value=1e-2, sampling="log")
    model = call_existing_code(
        units=units, activation=activation, lr=lr, dropout=dropout,
        layers = layers
    )
    return model
```

TensorBoard graphs of the top performing trials can be found in appendix F. Following this, manual adjustments were made on the hyperparameters using the best resulting optimisation.

6 Results and Discussion

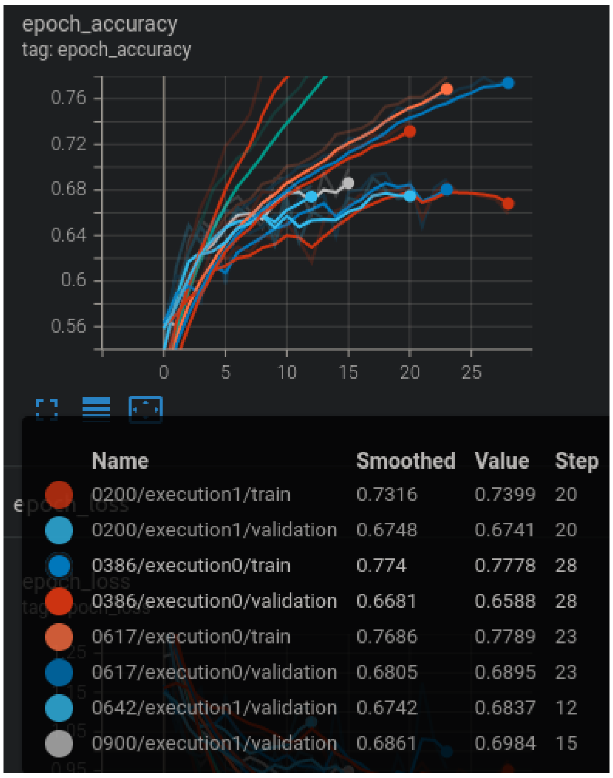
6.1 Introduction

The execution of the aforementioned system produced a wide range of results. These cover a variety of topics, including model optimisation, a comparison of the MFCC model with the Mel Spectrogram Model, and an evaluation of the overall model performance. In addition, an evaluation was carried out to identify the fraction of Jukebox samples that adhered to the chosen genre.

6.2 Genre Classification Model

6.2.1 Model Optimisation

The optimisation tool, KerasTuner, was used to conduct an extensive investigation of the model, carrying out a total of 1000 trials, with each trial consisting of two distinct executions. In order to effectively track the progression and performance of the model, every iteration was documented and the highest-performing models were visually represented using TensorBoard, with a specific focus on validation accuracy.



	Name	Smoothed	Value	Step
●	0200/execution1/train	0.7316	0.7399	20
●	0200/execution1/validation	0.6748	0.6741	20
●	0386/execution0/train	0.774	0.7778	28
●	0386/execution0/validation	0.6681	0.6588	28
●	0617/execution0/train	0.7686	0.7789	23
●	0617/execution0/validation	0.6805	0.6895	23
●	0642/execution1/validation	0.6742	0.6837	12
●	0900/execution1/validation	0.6861	0.6984	15

Figure 6.1 KerasTuner Top Trials Validation Accuracy

The primary objective assigned to KerasTuner was to identify the model demonstrating the highest level of validation accuracy. However, it is paramount to stress the significance of monitoring loss accuracy concurrently, as this serves as a preventative measure against the potential overfitting of the model. Consequently, the hyperparameters corresponding to the top-performing models were systematically retained. This data was then used as the foundation for additional manual tests, designed to further enhance the accuracy of the model, reinforcing the thoroughness of the experimental approach.

From the results generated, the final model was determined with the following specifications:

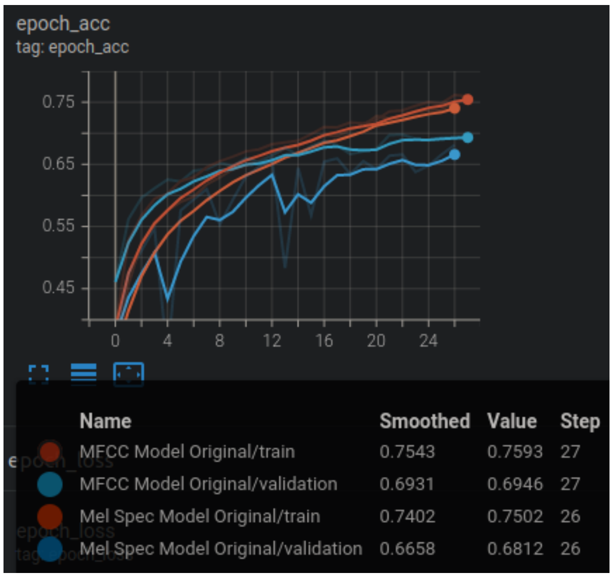
- Input Layer: number of neurons is dependent on input size
- Convolutional layers: The model has four convolutional layers, each with 64 filters of size (3,3) and ReLU (Rectified Linear Unit) activation.
- MaxPooling Layers: After each convolutional layer, there is a max pooling layer with pool size (3,3) and stride (2,2). The padding is set to 'same'.
- BatchNormalization Layers: The model has three batch normalization layers.
- Dropout Layers: The model has two dropout layers each with a dropout rate of 0.3.
- Dense Layers: The model has two dense layers. The first has 64 neurons and uses the ReLU activation function. The second is the output layer and has as many neurons as there are unique target classes. It uses softmax activation.

The full model summary can be viewed in the appendix B and the TensorBoard graphs (validation accuracy and loss) of the various trial runs can be viewed in the appendix F.

6.2.2 MFCC vs Mel Spectrogram Model

The aforementioned model was run twice during the course of experimentation. MFCC was used as the primary data in the first trial, followed by Mel Spectrogram data. To ensure a thorough and precise study of the model's performance, the metrics of training, validation accuracy, and loss were tracked and plotted. TensorBoard was once again used for this, allowing for a clear comparison and evaluation of the model's efficacy when trained on the two different types of data.

In the evaluation of the two MFCC and Mel Spectrogram models, their testing accuracies were recorded as 69% and 67% respectively. The MFCC model also had a better validation loss of 0.84 when compared to the Mel Spectrogram's model of 0.96. It is noteworthy to observe that the difference in performance between these two models was quite marginal, with the MFCC model exhibiting slightly superior performance. These results align with current research as MFCC is a derivative of Mel Spectrograms. Figure 6.2 is a representation of the validation accuracy difference between the MFCC model and the Mel Spectrogram model. The figure demonstrating the difference in loss can be found in appendix D



	Name	Smoothed	Value	Step
●	MFCC Model Original/train	0.7543	0.7593	27
●	MFCC Model Original/validation	0.6931	0.6946	27
●	Mel Spec Model Original/train	0.7402	0.7502	26
●	Mel Spec Model Original/validation	0.6658	0.6812	26

Figure 6.2 MFCC vs Mel Spectrogram Model Validation Accuracy

Consequently, considering the slight edge in accuracy and the significantly smaller data file size, the decision was made to proceed with the MFCC model for further applications and analyses.

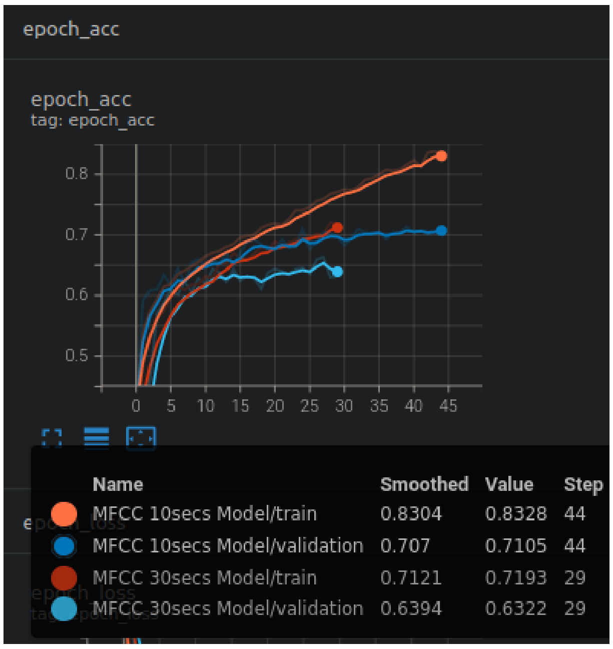
6.2.3 MFCC Model Time Snippets

As previously mentioned, the model was partitioned into snippets of 10 seconds each. However, tests were conducted using varying snippet durations to determine any fluctuations in precision or loss.

The model was re-run two times, initially with samples of 10 seconds, and then with the original sample length of 30 seconds. Table 6.1 is drawn up to display the performance of the models. A TensorBoard graphical representation of the validation accuracy can be found in figure 6.3 with the validation loss found in appendix E.

Table 6.1 MFCC Model 10 vs 30 sec performance

	10 seconds	**30 seconds**
Epochs	44	29
Training Accuracy	0.832	0.719
Validation Accuracy	0.71	0.632
Training Loss	0.437	0.774
Validation Loss	0.84	1.005
Testing Accuracy	0.702	0.649
Precision	0.702	0.649
F1 Score	0.702	0.649



	Name	Smoothed	Value	Step
●	MFCC 10secs Model/train	0.8304	0.8328	44
●	MFCC 10secs Model/validation	0.707	0.7105	44
●	MFCC 30secs Model/train	0.7121	0.7193	29
●	MFCC 30secs Model/validation	0.6394	0.6322	29

Figure 6.3 MFCC 10 seconds vs 30 seconds models

F1 Score and precision were all set to *average='micro'*. Sklearn explains this as 'calculates metrics globally by counting the total true positives, false negatives and false positives.' [40]. This decision was taken because there is a slight imbalance in the dataset. It is important to note that the model was optimised with 10-second song snippets in mind, and further optimisation of the 30-second sample may result in better performance. The current version of the 30-second model suffers from overfitting as can be seen in graph 6.3.

6.2.4 Conclusion

The final iteration of the genre classification model is a Convolutional Neural Network (CNN) that has been trained using MFCC data and 10,437 ten-second snippets of songs ranging across 6 genres. The data was divided using a stratified split of 70% for training, 15% for validation, and 15% for testing. The model achieved a validation accuracy of 0.71, and for the testing phase, it recorded an accuracy, precision, and f1 score of 0.7.

This performance puts it in line with the CNN models generated by other researchers in this field. As stated in section 3.5.1, Buttigieg Vella's model had a loss of 0.94 and an accuracy 0.69[12]. Boxler's [30] CNN resulted in a loss of 1.42, and Murauer and Specht had 1.65[31].

Given these positive results, the author is confident in the model's ability to accurately identify the genres of the Jukebox sample tracks in the subsequent analysis.

6.3 Jukebox Samples

Tables A.1, A.2 and A.3, which can be found in the Appendix, display songs composed using Jukebox model 1B lyrics, 5B lyrics and 5B respectively. The test case number references specific songs stored in Google Drive. Jukebox compositions are based on lyrics, artist, and genre inputs. The MGR model predicts the genre and a personal score rating the quality of the song from a scale of 0-7 is given. See appendix A for rating details.

The 5B and 5B lyrics models each produced 18 samples, while only 3 were generated by the 1B lyrics model, making a total of 39 samples generated. This is due to the 1B lyrics model being established as less effective than the others, hence, it is more advantageous to evaluate the 5B models.

All samples composed by Jukebox can be viewed by clicking 'here' or by accessing to the provided link:
drive.google.com/drive/folders/1FyqQlfWciNhiveshoayFRjlVEtEAh_Fl?usp=share_link

6.3.1 Genre Correctness

Based on the data presented in tables A.1, A.2 and A.3, the bar chart in figure 6.4 was constructed to provide a visual summary of the generated results. The term "Genre Match" signifies that the genre of two ten-second music snippets or more aligns with the pre-identified genre in question. "Artist Match" denotes instances where the detected genre coincides with the designated artist and "Genres/Artist Match" is the sum of both. This classification arises due to most songs having been assigned contrasting genres and artists.

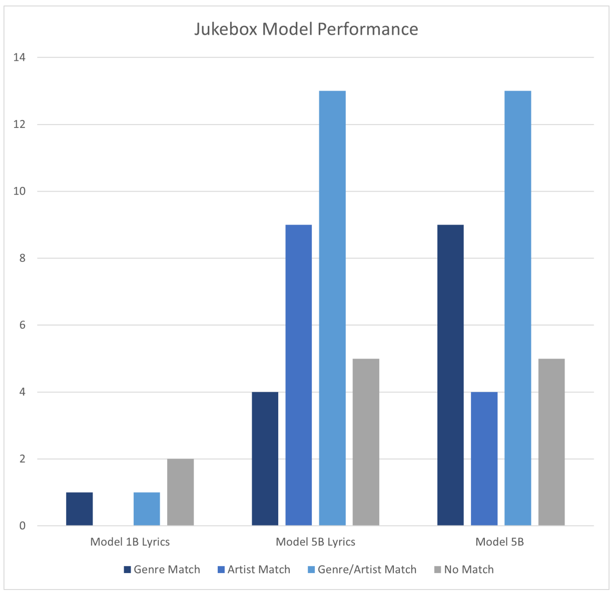


Figure 6.4 Jukebox Model Genre Correctness

From the 39 total test cases, several striking patterns emerged. As anticipated, Model 1B underperformed in comparison to the others. The results also revealed that half of the samples produced by Model 5B corresponded with the requested genre. On the other hand, only 22% of the samples from Model 5B Lyrics matched the requested genre, indicating a significant discrepancy between the two models that warranted further examination. To ensure consistency, it was confirmed whether, at the very least, the genre matched the artist in cases where test cases involved mismatched genres and artists (e.g., Genre: Classical, Artist: Lady Gaga). For Model 5B, 22% of the samples showed a genre correlation with the artist, and for Model 5B Lyrics, half of the samples exhibited the same result. This concluded that, across all 5B models, 72% of the produced samples had a genre aligning with either the requested genre or the artist's genre.

Several different scenarios were investigated to determine the performance of the different Jukebox models, notably models 5B and 5B lyrics.

6.3.2 Scenario 1: Artist is from the same genre

Given that Jukebox encompasses various parameters, the initial focus is on examining scenarios where the artist belongs to the same genre and no lyrics are specified. There are a total of fifteen test cases available for analysis, comprising three from Model 5B Lyrics and twelve from Model 5B, which satisfy this criterion. Both models exhibit a 67% success rate in achieving a genre match for these test cases. However, for the cases where a match was not achieved, no discernible pattern or reason could be identified to explain the discrepancies.

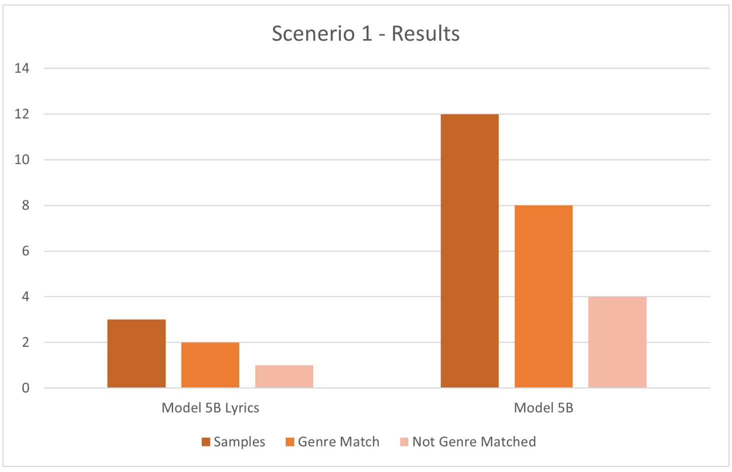


Figure 6.5 Jukebox Scenario 1 Results

6.3.3 Scenario 2: Artist is not from the same genre

The next set of scenarios focuses on cases where the artist does not belong to the same genre, and no lyrics are specified. Model 1B Lyrics consists of three test cases falling under this category, while both 5B models encompass six test cases each, resulting in a total of 15 cases for investigation. Model 1B and 5B Lyrics achieved a genre match rate of only 33%, with Model 5B performing slightly better at a 50

Interestingly, across all 5B models, it was observed that in test cases where the predicted genre did not align with the requested genre, it instead corresponded to the genre associated with the artist. For example, when the parameters were set to Mozart as the artist and the rock genre, the generated sample would unexpectedly yield a classical song instead of a rock song. However, this pattern was not observed in Model 1B Lyrics, where the generated samples did not even match the artist, highlighting its significantly poorer performance compared to the other models.

Consequently, it can be inferred that Model 5B Lyrics had a 66% higher likelihood of composing a song belonging to the artist's genre, indicating that this model places greater emphasis on the artist parameter compared to its counterpart.

6.3.4 Scenario 3: Impact of the lyrics parameter

Significant findings emerged from test cases 13 to 18 of Model 5B Lyrics, comprising six test scenarios with the classical genre and Mozart as the designated artist. The distinction lies in the fact that the first three test cases had their lyrics set to the chorus of 'It's My Life' by Bon Jovi, while the remaining three had no established lyrics.

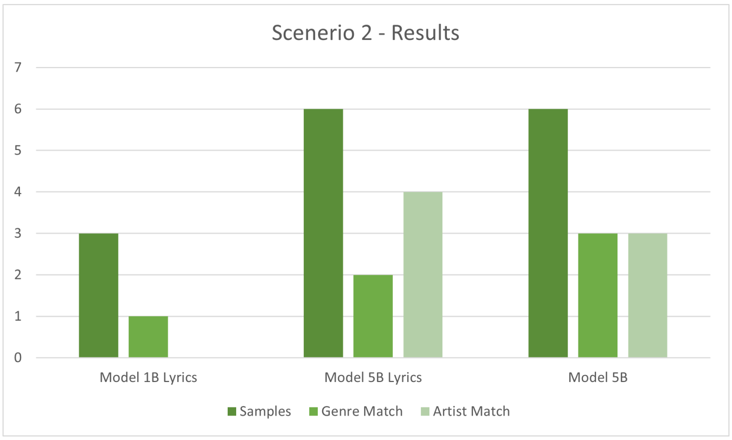


Figure 6.6 Jukebox Scenario 2 Results

Interestingly, the model did not predict a classical genre for the samples with lyrics, except for a brief 10-second segment in case 15. In contrast, two-thirds of the samples without specified lyrics aligned with the expected classical genre.

A similar pattern emerged in test cases 4 to 9, where all test cases were assigned the classical genre and Lady Gaga as the artist. In this instance, three samples (test cases 7 to 9) had their lyrics set to the chorus of Lady Gaga's 'Poker Face'. These samples yielded consistent results, with two-thirds of the samples without lyrics aligning with the desired genre, while samples with lyrics produced a mismatch.

These findings suggest that the lyrics parameter significantly influences the predicted genre of the song, likely because the 5B Lyrics Model draws inspiration from the original music itself. This observation is further supported by test cases 19 to 21, where the genre was set to 'pop' and the lyrics were extracted from a song belonging to this genre. In all three test cases, the predicted genre aligned positively with the specified genre in this scenario.

6.3.5 Sample Quality

Revisiting tables A.1, A.2 and A.3, the following figure 6.7 was drawn up to visually represent my personal assessment of the sample quality.

An intriguing observation while listening to the samples composed by the various models was that Model 5B consistently produced songs with lyrics (Test Case 37, 38, 39 for example). In evaluating the following section, the primary emphasis will be on the two polar ends of the song ratings: the 0-1 and 6-7 ranges. This is based on the assumption that most listeners would likely agree upon the reasons behind these categorizations.

44% of all the generated samples fall into the lower classification tier. Songs that belong to this group, especially those given a rating of 0, are of very poor quality.

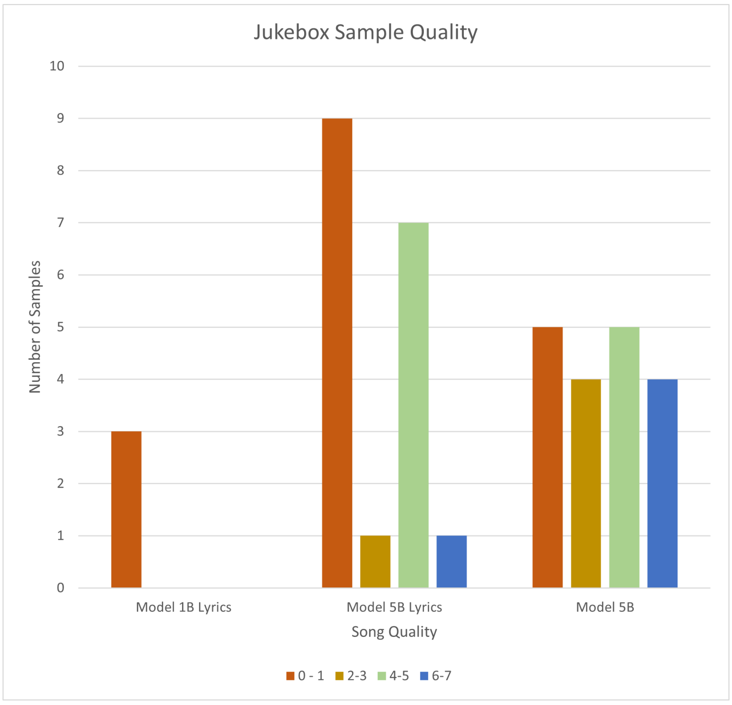


Figure 6.7 Jukebox Model Sample Quality

This includes instances of complete silence (Test Case 17), songs with only a person speaking (Test Case 4) and unintelligible noises (Test Case 10).

Only 5 samples (13%) were of sufficient quality for the author to feel confident enough to place award them the six or seven rankings.

6.3.6 Jukebox Composition of Classical Music

Due to the resource-intensive nature of composing songs, the decision was made to predominantly focus on creating classical compositions. Specifically, when examining Models 5B and 5B Lyrics, it was found that classical songs accounted for 70% of the total samples generated. Surprisingly, out of the 14 songs that received low ratings, half of them had their genre set to classical by the author. However, only two of these songs were correctly identified as belonging to the classical genre. Additionally, three out of the top five highest-rated samples were also detected as classical songs, even when a different genre was requested (as seen in Test Cases 37 and 39).

Considering the superior performance of classical song samples, particularly when taking ratings between 4 to 5 into consideration, and the occasional composition of classical pieces by Jukebox, it can be concluded that there is a subtle inclination towards generating high-quality classical samples.

6.3.7 Correlation Between Genre Correctness and Quality

A notable observation was the frequent occurrence of low-rated samples that still managed to match the predicted genre. One such example is Test Case 12, which consists of applause and unintelligible noises. It could be that Jukebox falsely believed it had produced a correct outcome, which was not the case. Alternatively, it could be attributed to certain genres and artists not harmonizing well together. For instance, all test cases involving classical music from Kanye West yielded consistently poor results.

6.4 Conclusion

Numerous samples within the dataset exhibit different detected genres for their snippets. While part of this inconsistency may be attributed to the MGR model, some can be attributed to the quality of the song snippets themselves. For instance, consider test case 14, where the sample begins with silence and then transitions into a brief melody. Although the genre classification model meets certain standards, it is not 100% accurate, and running these test cases on a different model could yield vastly different outcomes.

Based on the aforementioned results, in the simplest test cases where the genre aligns with the artist and no lyrics are specified, Jukebox achieved a 67% genre match rate. Since there is no comparable research conducted on other music composition models, the author considers a two-thirds success rate in test cases to be acceptable.

However, when the requested genre contradicts the artist, the predicted genre tends to favor the artist rather than adhering to the desired genre. This discrepancy may arise from Jukebox's tendency to compose songs that closely resemble the artist's style rather than aligning with the specified genre. While it is commendable that all samples generated by the superior 5B models exhibit alignment with an artist or genre, it is crucial to acknowledge that the genre accuracy has decreased to 42%.

Furthermore, the inclusion of lyrics as a parameter has shown a significant influence on the sample's genre classification. Songs with matching genres and lyrics achieve a 100% genre match rate, while songs with opposing genres and lyrics have no matches. This observation may once again stem from the fact that songs are composed with substantial influence from the original song containing those specific lyrics.

Jukebox's performance is significantly impacted by a multitude of parameters, and different hyper-optimisations can alter these results. Additionally, the quality of the samples produced by Jukebox varies considerably even in the same batch of song composition.

7 Conclusion

In conclusion, this study aimed to evaluate the performance of Music Composition (MC) software by analysing the quality of the composed samples. The main objective was to establish an objective and reliable method for assessing the quality of the samples produced by MC software. To achieve this, a Music Genre Recognition (MGR) model was developed, enabling the evaluation of genre alignment.

Throughout the study, it became evident that assessing the quality of a song remains subjective, to the point where even the genre of certain songs is subjective. However, the MGR model provided a valuable tool for objectively determining whether the composed samples aligned with the requested genre. This approach allowed for a more comprehensive and unbiased assessment of the MC software's performance.

The findings of this study demonstrate the effectiveness and importance of incorporating the MGR model into the evaluation process. By comparing the predicted genre with the requested genre, it was possible to assess the MC software's ability to compose samples that align with the desired genre. This innovative approach addresses the need for a more objective evaluation methodology in the field of MC software, providing a viable approach.

Furthermore, the study focused on analysing the impact of specific parameters on the genre of the composed songs. When evaluating Jukebox, it was observed that contradictory genre and artist combinations often resulted in the MC models favouring the genre of the artist. Additionally, when the lyrics of an existing song were included in the parameters, the composed music always aligned with the genre of the original piece.

These findings highlight the complexities involved in genre prediction and the influence of different parameters on the output of MC models. The study provides valuable insights into the strengths and limitations of these models in composing songs of the desired genre.

Overall, this research contributes to the understanding of MC model performance and offers insights for future improvements in composing high-quality music samples that align with specific genre requirements.

7.1 Limitations and Future Work

The project faced significant limitations in terms of time and resources, which had a substantial impact. Despite this, considerable effort was dedicated to developing the Music Genre Recognition (MGR) model. Although it was not initially the primary focus, having a robust and accurate MGR model was deemed crucial to ensure a reliable evaluation of Jukebox. An alternative approach considered by the author was to design a model capable of detecting the genre of full 30-second songs. Although attempted, further optimisation was necessary to achieve performance comparable to the 10-second model. Utilising longer song lengths would provide the model with more contextual information, and it would also streamline the results generation process by yielding a single outcome instead of three.

In terms of the research approach, the grading of music composition models through genre validation appears to be a novel approach, as no prior research matching this methodology was found. This opens up avenues for further exploration and additional studies in this field. For instance, conducting more test cases with Jukebox, exploring its ability to take a song sample and continue the composition, or even conducting comparative tests using samples from different models altogether could yield valuable insights. There is substantial potential for future research to delve deeper into these areas and expand the understanding of music composition models and their capabilities.

References

[1] *How a team of musicologists and computer scientists completed Beethoven's unfinished 10th Symphony*. [Online]. Available: https://theconversation.com/how-a-team-of-musicologists-and-computer-scientists-completed-beethovens-unfinished-10th-symphony-168160.

[2] M. Civit, J. Civit-Masot, F. Cuadrado, and M. J. Escalona, "A systematic review of artificial intelligence-based music generation: Scope, applications, and future trends," *Expert Systems with Applications*, vol. 209, Dec. 2022, ISSN: 09574174. DOI: 10.1016/J.ESWA.2022.118190.

[3] D. Eck, "A network of relaxation oscillators that finds downbeats in rhythms," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2130, pp. 1239–1247, 2001, ISSN: 16113349. DOI: 10.1007/3-540-44668-0{_}173/COVER. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-44668-0_173.

[4] M. C. Mozer, "Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multi-scale Processing," *http://dx.doi.org/10.1080/09540099408915726*, vol. 6, no. 2-3, pp. 247–280, 2007, ISSN: 13600494. DOI: 10.1080/09540099408915726. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/09540099408915726.

[5] C. Hernandez-Olivan and J. R. Beltrán, "Music Composition with Deep Learning: A Review," *Signals and Communication Technology*, pp. 25–50, Aug. 2021, ISSN: 18604870. DOI: 10.48550/arxiv.2108.12290. [Online]. Available: https://arxiv.org/abs/2108.12290v2.

[6] Y. Bengio, I. Goodfellow, and A. Courville, "Deep Learning," 2015.

[7] J. Pons and X. Serra Casals, "Deep neural networks for music and audio tagging,"

[8] L. A. Iliadis, S. P. Sotiroudis, K. Kokkinidis, P. Sarigiannidis, S. Nikolaidis, and S. K. Goudos, "Music Deep Learning: A Survey on Deep Learning Methods for Music Processing," *2022 11th International Conference on Modern Circuits and Systems Technologies, MOCAST 2022*, 2022. DOI: 10.1109/MOCAST54814.2022.9837541.

[9] K. Choi, G. Fazekas, K. Cho, and M. Sandler, "A Tutorial on Deep Learning for Music Information Retrieval," [Online]. Available: http://www.ismir.net.

[10] T. L. Li and A. B. Chan, "Genre classification and the invariance of MFCC features to key and tempo," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6523 LNCS, no. PART 1, pp. 317–327, 2011, ISSN: 03029743. DOI: 10.1007/978-3-642-17832-0{_}30/COVER. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-17832-0_30.

[11] T. Lidy, A. Rauber, A. Pertusa, and J. Manuel, "IMPROVING GENRE CLASSIFICATION BY COMBINATION OF AUDIO AND SYMBOLIC DESCRIPTORS USING A TRANSCRIPTION SYSTEM," 2007.

[12] J. Buttigieg, V. Supervisor, and J. Bonello, "Analysing Diverse Algorithms performing Music Genre Recognition," 2022.

[13] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, "Deep Learning Techniques for Music Generation," Computational Synthesis and Creative Systems, 2020. DOI: 10.1007/978-3-319-70163-9. [Online]. Available: http://link.springer.com/10.1007/978-3-319-70163-9.

[14] A. Natsiou and S. O'Leary, "Audio representations for deep learning in sound synthesis: A review," *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, vol. 2021-December, Jan. 2022, ISSN: 21615330. DOI: 10.48550/arxiv.2201.02490. [Online]. Available: https://arxiv.org/abs/2201.02490v1.

[15] S. S. Stevens, ; J. Volkmann, and ; E. B. Newman, "A Scale for the Measurement of the Psychological Magnitude Pitch Related Content Architecture psychology," *J Acoust Soc Am*, vol. 8, pp. 185–190, 1937. DOI: 10.1121/1.1915893. [Online]. Available: https://doi.org/10.1121/1.1915893.

[16] T. Zhang, G. Feng, J. Liang, and T. An, "Acoustic scene classification based on Mel spectrogram decomposition and model merging," DOI: 10.1016/j.apacoust.2021.108258. [Online]. Available: https://doi.org/10.1016/j.apacoust.2021.108258.

[17] *Learning from Audio: The Mel Scale, Mel Spectrograms, and Mel Frequency Cepstral Coefficients | by mlearnere | Towards Data Science*. [Online]. Available: https://towardsdatascience.com/learning-from-audio-the-mel-scale-mel-spectrograms-and-mel-frequency-cepstral-coefficients-f5752b6324a8.

[18] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/NECO.1997.9.8.1735. [Online]. Available: https://direct.mit.edu/neco/article/9/8/1735/6109/Long-Short-Term-Memory.

[19] H. Chu, R. Urtasun, and S. Fidler, "SONG FROM PI: A MUSICALLY PLAUSIBLE NETWORK FOR POP MUSIC GENERATION," [Online]. Available: https://youtu.be/OMq9he-5HUU.

[20] *Overview of GAN Structure | Machine Learning | Google for Developers*. [Online]. Available: https://developers.google.com/machine-learning/gan/gan_structure.

[21] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer," *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, pp. 747–754, Sep. 2018. DOI: 10.48550/arxiv.1809.07600. [Online]. Available: https://arxiv.org/abs/1809.07600v1.

[22] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music," Mar. 2018. [Online]. Available: http://arxiv.org/abs/1803.05428.

[23] A. Agostinelli *et al.*, "MusicLM: Generating Music From Text,"

[24] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A Generative Model for Music," Apr. 2020.

[25] C. Hawthorne, A. Huang, D. Ippolito, and D. Eck, "Transformer-NADE for Piano Performances," [Online]. Available: https://goo.gl/magenta/notetuple-examples..

[26] *MuseNet*. [Online]. Available: https://openai.com/research/musenet.

[27] G. Hadjeres and L. Crestel, "THE PIANO INPAINTING APPLICATION,"

[28] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "MuseGAN: Multi-Track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment," [Online]. Available: https://salu133445.github.io/musegan/.

[29] *openai/jukebox: Code for the paper "Jukebox: A Generative Model for Music"*. [Online]. Available: https://github.com/openai/jukebox.

[30] D. Boxler, "MACHINE LEARNING TECHNIQUES APPLIED TO MUSICAL GENRE RECOGNITION by," 2020.

[31] B. Murauer, "Detecting Music Genre Using Extreme Gradient Boosting," 2018. DOI: 10.1145/3184558.3191822. [Online]. Available: https://doi.org/10.1145/3184558.3191822.

[32] D. Bisharad and R. H. Laskar, "Music genre recognition using convolutional recurrent neural network architecture," *Expert Systems*, vol. 36, no. 4, e12429, Aug. 2019, ISSN: 1468-0394. DOI: 10.1111/EXSY.12429. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1111/exsy.12429%20https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.12429%20https://onlinelibrary.wiley.com/doi/10.1111/exsy.12429.

[33] V. Lostanlen and C.-E. CelláCellá, "DEEP CONVOLUTIONAL NETWORKS ON THE PITCH SPIRAL FOR MUSICAL INSTRUMENT RECOGNITION," [Online]. Available: www.github.com/lostanlen/ismir2016..

[34] P. Li, J. Qian, and T. Wang, "AUTOMATIC INSTRUMENT RECOGNITION IN POLYPHONIC MUSIC USING CONVOLUTIONAL NEURAL NETWORKS,"

[35] C. Senac, T. Pellegrini, F. Mouret, and J. Pinquier, "Music feature maps with convolutional neural networks for music genre classification," *ACM International Conference Proceeding Series*, vol. Part F130150, Jun. 2017. DOI: 10.1145/3095713.3095733. [Online]. Available: https://dl.acm.org/doi/10.1145/3095713.3095733.

[36] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in Neurorobotics*, vol. 7, no. DEC, p. 21, Dec. 2013, ISSN: 16625218. DOI: 10.3389/FNBOT.2013.00021/BIBTEX.

[37] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A Dataset For Music Analysis," *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017*, pp. 316–323, Dec. 2016. [Online]. Available: https://arxiv.org/abs/1612.01840v3.

[38] Vast.ai, *FAQ | Vast.ai*, 2023. [Online]. Available: https://vast.ai/faq.

[39] *Interacting with Jukebox - Colaboratory*. [Online]. Available: https://colab.research.google.com/github/openai/jukebox/blob/master/jukebox/Interacting_with_Jukebox.ipynb.

[40] *sklearn.metrics.f1_score — scikit-learn 1.2.2 documentation*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.

Appendix A Jukebox Sample Results

A.1 Jukebox Test Cases Genre

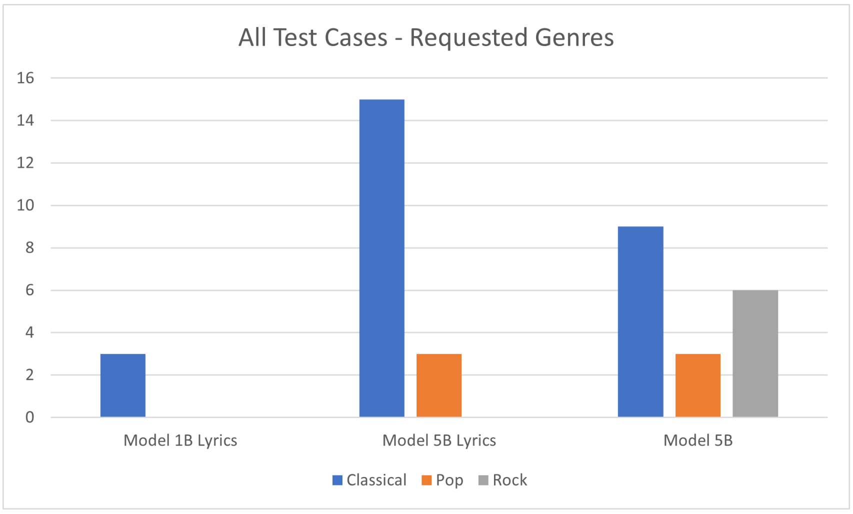


Figure A.1 Jukebox All Samples Requested Genres

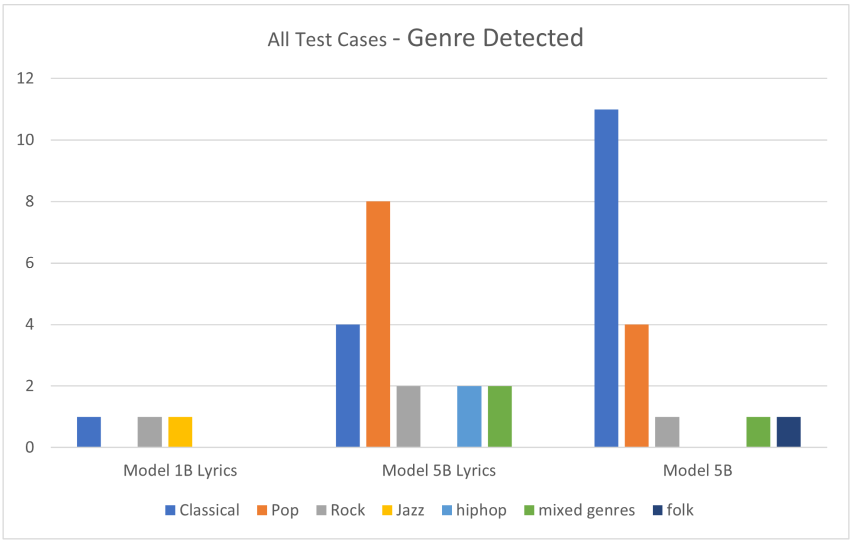


Figure A.2 Jukebox All Samples Detected Genres

A.2 Jukebox Test Cases Details

Predicted Genre:

0-10 / 10-20 / 20-30 (seconds)

Rating Explanation:

- 0 - Samples at this level are plagued by excessive static or completely lacking in melody
- 1 - Samples at this level have hints of instrumental music, but are plagued with static or other noises.
- 2 - There is a distinct jump in quality between this rating and the other. The sample is very clearly a song but is plagued with minor issues such as a simple melody or some background noises
- 3 - Samples at this level are more complex and involve multiple instruments
- 4 - Songs at this level start displaying a sort of progression. They may begin slowly and end with a faster tempo in a progression that makes sense. However, sound quality is still lacking.
- 5 - Songs at this rating are similar to the previous but stand out in some way.
- 6 - Songs in this rating feature multiple instruments and a clear quality. These are songs that the author enjoyed listening to.
- 7 - The very best of the samples available. They are quite complex compared to the rest and contain multiple instruments and in some cases even lyrics, to create a feeling of listening to an orchestra. Additionally, songs at this rating are also of a high sound quality.

Table A.1 1B Lyrics Model Test Cases

Test Case	Lyrics	Artist	Genre	Predicted Genre	Rating
1	None	Kanye West	Classical	pop/jazz/jazz	0
2	None	Kanye West	Classical	hiphop/rock/rock	0
3	None	Kanye West	Classical	classical/classical/-classical	0

Table A.2 5B Lyrics Model Test Cases

Test Case	Lyrics	Artist	Genre	Predicted Genre	Rating
4	None	Lady Gaga	Classical	pop/pop/pop	0
5	None	Lady Gaga	Classical	classical/classical/-classical	4
6	None	Lady Gaga	Classical	classical/classical/-classical	5
7	Poker Face	Lady Gaga	Classical	pop/rock/rock	5
8	Poker Face	Lady Gaga	Classical	pop/pop/pop	5
9	Poker Face	Lady Gaga	Classical	pop/classical/pop	0
10	None	Kanye	Classical	pop/hiphop/hiphop	0
11	None	Kanye	Classical	pop/hiphop/pop	0
12	None	Kanye	Classical	pop/pop/hiphop	0
13	It's My Life	Mozart	Classical	rock/rock/folk	6
14	It's My Life	Mozart	Classical	jazz/folk/rock	3
15	It's My Life	Mozart	Classical	pop/classical/pop	4
16	None	Mozart	Classical	classical/jazz/rock	5
17	None	Mozart	Classical	pop/classical/clas-sical	0
18	None	Mozart	Classical	classical/classical/-classical	5
19	Poker Face	Mozart	Pop	pop/folk/pop	1
20	Poker Face	Mozart	Pop	pop/pop/rock	1

Continued on next page

Table A.2 – continued from previous page

Test Case	Lyrics	Artist	Genre	Predicted Genre	Rating
21	Poker Face	Mozart	Pop	pop/pop/hiphop	1

Table A.3 5B Model Test Cases

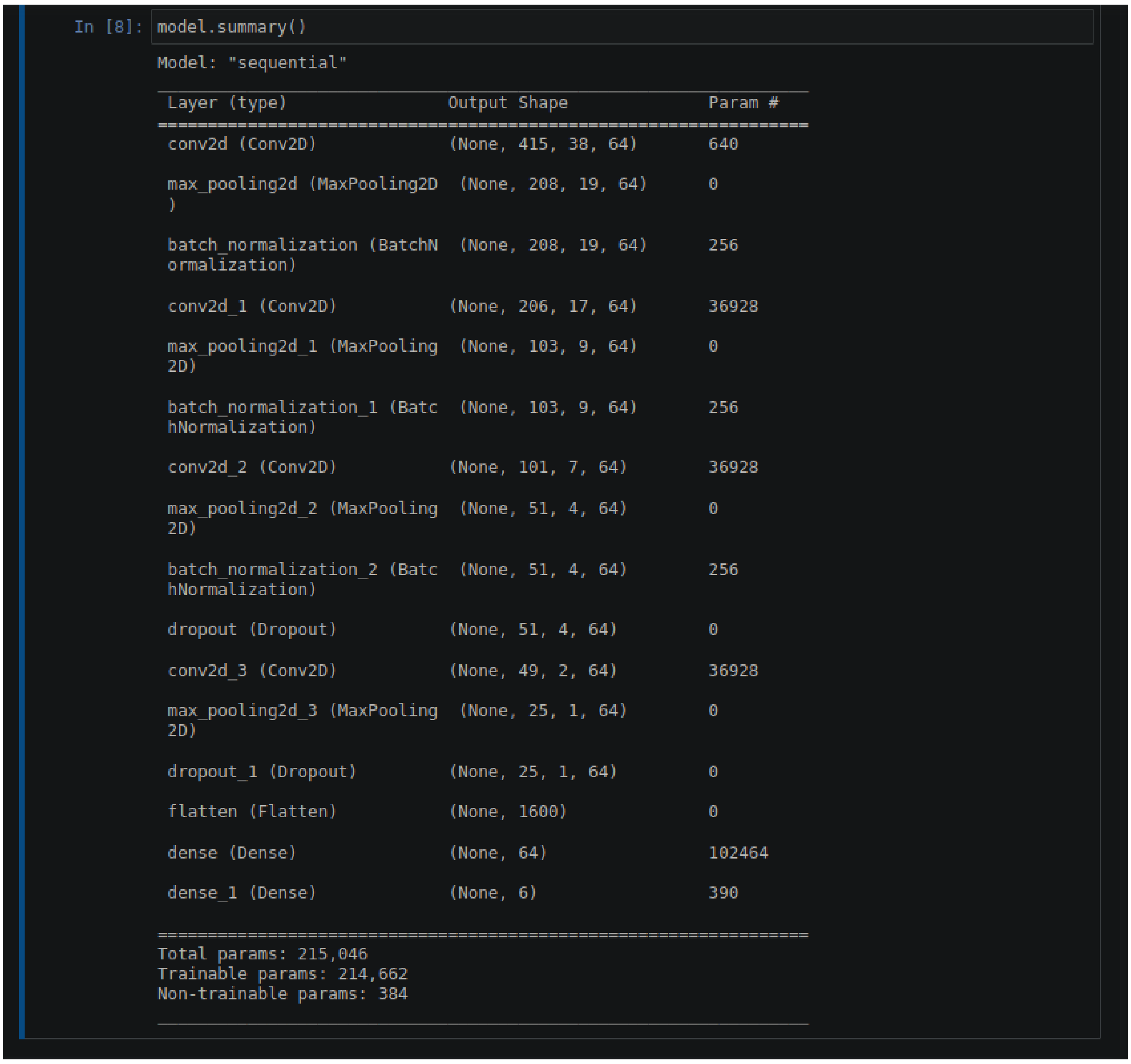
Test Case	Lyrics	Artist	Genre	Predicted Genre	Rating
22	None	Beethoven	Classical	classical/pop/folk	6
23	None	Beethoven	Classical	classical/classical/-classical	7
24	None	Beethoven	Classical	classical/folk/clas-sical	4
25	None	Lady Gaga	Classical	classical/folk/folk	4
26	None	Lady Gaga	Classical	classical/classical/-folk	4
27	None	Lady Gaga	Classical	pop/pop/pop	4
28	None	Mozart	Classical	classical/classical/-classical	3
29	None	Mozart	Classical	classical/folk/clas-sical	1
30	None	Mozart	Classical	classical/classical/-classical	2
31	None	Lady Gaga	Pop	pop/pop/pop	1
32	None	Lady Gaga	Pop	hiphop/classical/-classical	1
33	None	Lady Gaga	Pop	hiphop/pop/pop	1
34	None	Bon Jovi	Rock	rock/rock/rock	1
35	None	Bon Jovi	Rock	classical/classical/-classical	2

Continued on next page

Table A.3 – continued from previous page

Test Case	Lyrics	Artist	Genre	Predicted Genre	Rating
36	None	Bon Jovi	Rock	pop/pop/pop	3
37	None	Mozart	Rock	rock/classical/classical	7
38	None	Mozart	Rock	classical/classical/hiphop	4
39	None	Mozart	Rock	classical/classical/folk	7

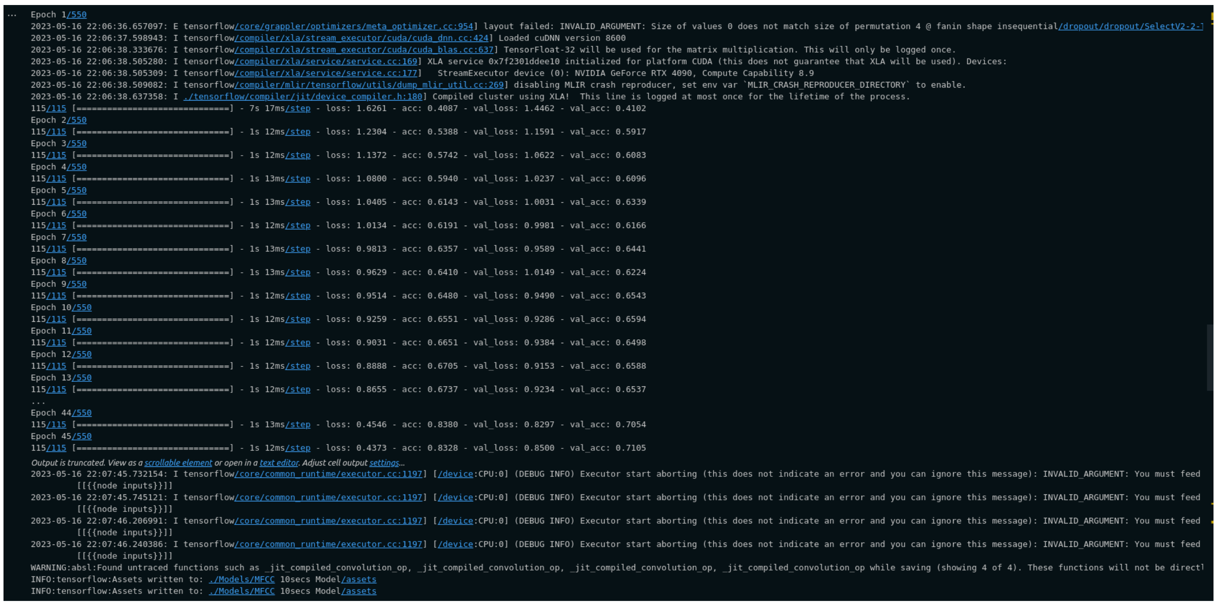
Appendix B Music Genre Recognition (MGR) Model Summary



```
In [8]: model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 415, 38, 64)       640

 max_pooling2d (MaxPooling2D  (None, 208, 19, 64)      0
 )

 batch_normalization (BatchN  (None, 208, 19, 64)      256
 ormalization)

 conv2d_1 (Conv2D)           (None, 206, 17, 64)       36928

 max_pooling2d_1 (MaxPooling  (None, 103, 9, 64)       0
 2D)

 batch_normalization_1 (Batc  (None, 103, 9, 64)       256
 hNormalization)

 conv2d_2 (Conv2D)           (None, 101, 7, 64)        36928

 max_pooling2d_2 (MaxPooling  (None, 51, 4, 64)        0
 2D)

 batch_normalization_2 (Batc  (None, 51, 4, 64)        256
 hNormalization)

 dropout (Dropout)           (None, 51, 4, 64)         0

 conv2d_3 (Conv2D)           (None, 49, 2, 64)         36928

 max_pooling2d_3 (MaxPooling  (None, 25, 1, 64)        0
 2D)

 dropout_1 (Dropout)         (None, 25, 1, 64)         0

 flatten (Flatten)           (None, 1600)              0

 dense (Dense)               (None, 64)                102464

 dense_1 (Dense)             (None, 6)                 390

=================================================================
Total params: 215,046
Trainable params: 214,662
Non-trainable params: 384
_________________________________________________________________
```

Figure B.1 MGR Model Summary

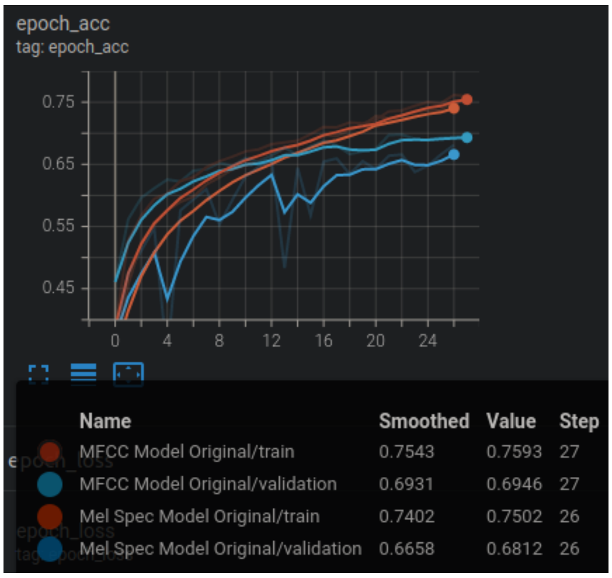
Appendix C Music Genre Recognition (MGR) Model Fit



```
...  Epoch 1/550
     2023-05-16 22:06:36.657097: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:954] layout failed: INVALID_ARGUMENT: Size of values 0 does not match size of permutation 4 @ fanin shape insequential/dropout/dropout/SelectV2-2-]
     2023-05-16 22:06:37.598943: I tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:424] Loaded cuDNN version 8600
     2023-05-16 22:06:38.333676: I tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:637] TensorFloat-32 will be used for the matrix multiplication. This will only be logged once.
     2023-05-16 22:06:38.505280: I tensorflow/compiler/xla/service/service.cc:169] XLA service 0x7f2301ddee10 initialized for platform CUDA (this does not guarantee that XLA will be used). Devices:
     2023-05-16 22:06:38.505309: I tensorflow/compiler/xla/service/service.cc:177]   StreamExecutor device (0): NVIDIA GeForce RTX 4090, Compute Capability 8.9
     2023-05-16 22:06:38.509082: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:269] disabling MLIR crash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
     2023-05-16 22:06:38.637358: I ./tensorflow/compiler/jit/device_compiler.h:180] Compiled cluster using XLA!  This line is logged at most once for the lifetime of the process.
     115/115 [==============================] - 7s 17ms/step - loss: 1.6261 - acc: 0.4087 - val_loss: 1.4462 - val_acc: 0.4102
     Epoch 2/550
     115/115 [==============================] - 1s 12ms/step - loss: 1.2304 - acc: 0.5388 - val_loss: 1.1591 - val_acc: 0.5917
     Epoch 3/550
     115/115 [==============================] - 1s 12ms/step - loss: 1.1372 - acc: 0.5742 - val_loss: 1.0622 - val_acc: 0.6083
     Epoch 4/550
     115/115 [==============================] - 1s 13ms/step - loss: 1.0800 - acc: 0.5940 - val_loss: 1.0237 - val_acc: 0.6096
     Epoch 5/550
     115/115 [==============================] - 1s 13ms/step - loss: 1.0405 - acc: 0.6143 - val_loss: 1.0031 - val_acc: 0.6339
     Epoch 6/550
     115/115 [==============================] - 1s 12ms/step - loss: 1.0134 - acc: 0.6191 - val_loss: 0.9981 - val_acc: 0.6166
     Epoch 7/550
     115/115 [==============================] - 1s 13ms/step - loss: 0.9813 - acc: 0.6357 - val_loss: 0.9589 - val_acc: 0.6441
     Epoch 8/550
     115/115 [==============================] - 1s 13ms/step - loss: 0.9629 - acc: 0.6410 - val_loss: 1.0149 - val_acc: 0.6224
     Epoch 9/550
     115/115 [==============================] - 1s 12ms/step - loss: 0.9514 - acc: 0.6480 - val_loss: 0.9490 - val_acc: 0.6543
     Epoch 10/550
     115/115 [==============================] - 1s 12ms/step - loss: 0.9259 - acc: 0.6551 - val_loss: 0.9286 - val_acc: 0.6594
     Epoch 11/550
     115/115 [==============================] - 1s 12ms/step - loss: 0.9031 - acc: 0.6651 - val_loss: 0.9384 - val_acc: 0.6498
     Epoch 12/550
     115/115 [==============================] - 1s 12ms/step - loss: 0.8888 - acc: 0.6705 - val_loss: 0.9153 - val_acc: 0.6588
     Epoch 13/550
     115/115 [==============================] - 1s 12ms/step - loss: 0.8655 - acc: 0.6737 - val_loss: 0.9234 - val_acc: 0.6537
     ...
     Epoch 44/550
     115/115 [==============================] - 1s 13ms/step - loss: 0.4546 - acc: 0.8380 - val_loss: 0.8297 - val_acc: 0.7054
     Epoch 45/550
     115/115 [==============================] - 1s 12ms/step - loss: 0.4373 - acc: 0.8328 - val_loss: 0.8500 - val_acc: 0.7105
     Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
     2023-05-16 22:07:45.732154: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must feed
              [[{{node inputs}}]]
     2023-05-16 22:07:45.745121: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must feed
              [[{{node inputs}}]]
     2023-05-16 22:07:46.206991: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must feed
              [[{{node inputs}}]]
     2023-05-16 22:07:46.240386: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must feed
              [[{{node inputs}}]]
     WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing 4 of 4). These functions will not be directl
     INFO:tensorflow:Assets written to: ./Models/MFCC 10secs Model/assets
     INFO:tensorflow:Assets written to: ./Models/MFCC 10secs Model/assets
```

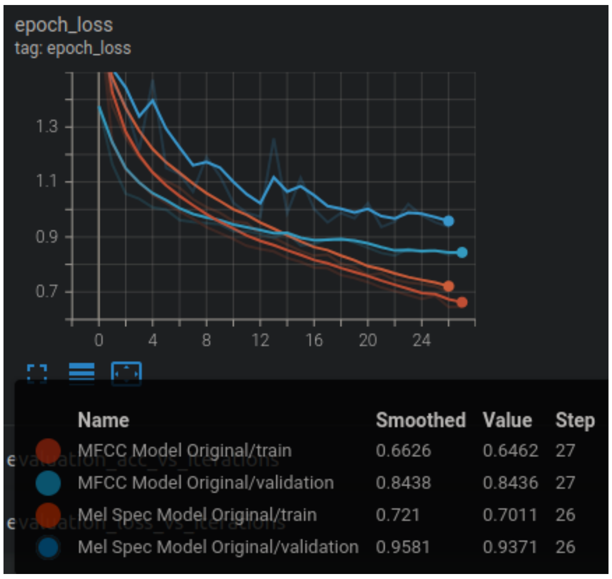
Figure C.1 MGR Model fit

Appendix D MGR: MFCC vs Mel Spectrogram



	Name	Smoothed	Value	Step
●	MFCC Model Original/train	0.7543	0.7593	27
●	MFCC Model Original/validation	0.6931	0.6946	27
●	Mel Spec Model Original/train	0.7402	0.7502	26
●	Mel Spec Model Original/validation	0.6658	0.6812	26

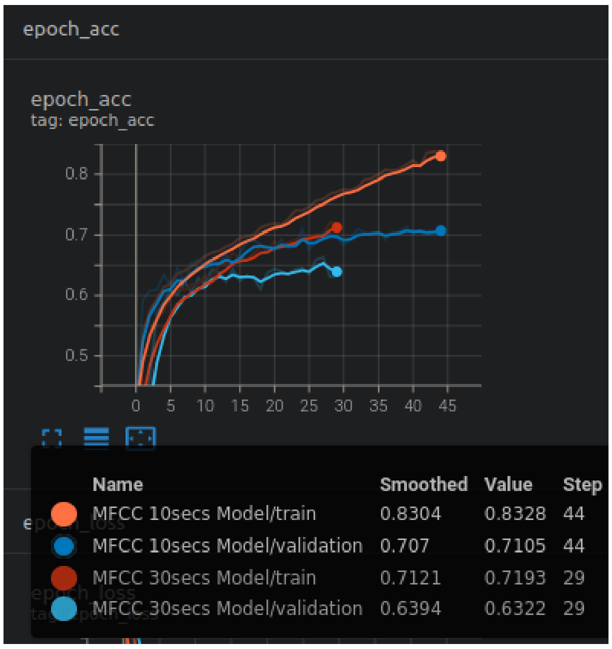
Figure D.1 MGR: MFCC vs Mel Spectrogram Testing & Validation Accuracy



	Name	Smoothed	Value	Step
●	MFCC Model Original/train	0.6626	0.6462	27
●	MFCC Model Original/validation	0.8438	0.8436	27
●	Mel Spec Model Original/train	0.721	0.7011	26
●	Mel Spec Model Original/validation	0.9581	0.9371	26

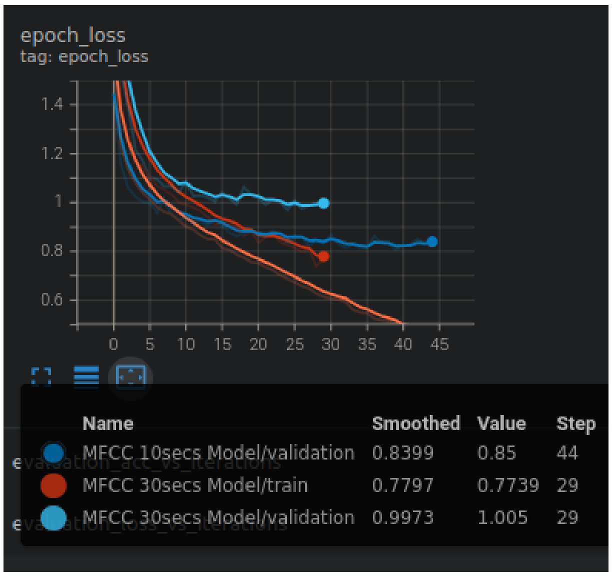
Figure D.2 MGR: MFCC vs Mel Spectrogram Loss Accuracy

Appendix E MGR 10 seconds vs 30 seconds



	Name	Smoothed	Value	Step
●	MFCC 10secs Model/train	0.8304	0.8328	44
●	MFCC 10secs Model/validation	0.707	0.7105	44
●	MFCC 30secs Model/train	0.7121	0.7193	29
●	MFCC 30secs Model/validation	0.6394	0.6322	29

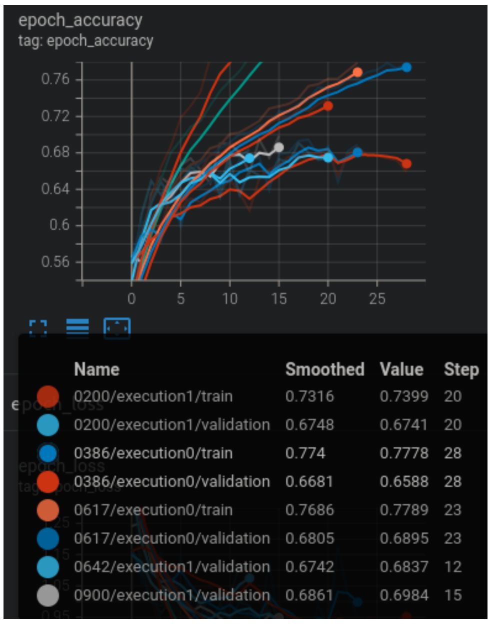
Figure E.1 MGR 10vs30 seconds Validation Accuracy



	Name	Smoothed	Value	Step
●	MFCC 10secs Model/validation	0.8399	0.85	44
●	MFCC 30secs Model/train	0.7797	0.7739	29
●	MFCC 30secs Model/validation	0.9973	1.005	29

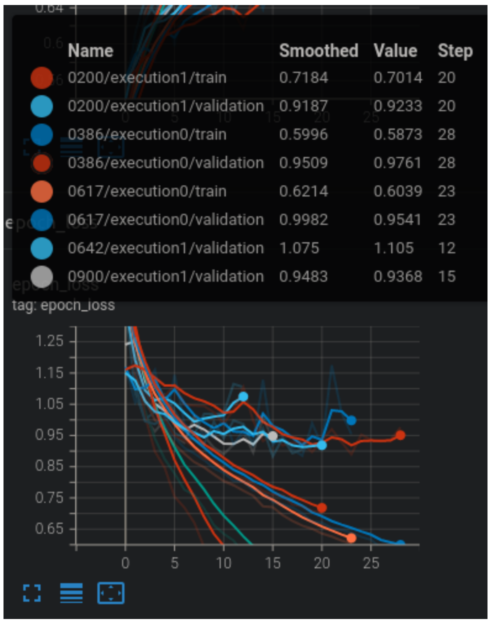
Figure E.2 MGR 10vs30 seconds loss

Appendix F MGR Model Optimisation



	Name	Smoothed	Value	Step
●	0200/execution1/train	0.7316	0.7399	20
●	0200/execution1/validation	0.6748	0.6741	20
●	0386/execution0/train	0.774	0.7778	28
●	0386/execution0/validation	0.6681	0.6588	28
●	0617/execution0/train	0.7686	0.7789	23
●	0617/execution0/validation	0.6805	0.6895	23
●	0642/execution1/validation	0.6742	0.6837	12
●	0900/execution1/validation	0.6861	0.6984	15

Figure F.1 MGR Model Trials Training & Validation Accuracy



	Name	Smoothed	Value	Step
●	0200/execution1/train	0.7184	0.7014	20
●	0200/execution1/validation	0.9187	0.9233	20
●	0386/execution0/train	0.5996	0.5873	28
●	0386/execution0/validation	0.9509	0.9761	28
●	0617/execution0/train	0.6214	0.6039	23
●	0617/execution0/validation	0.9982	0.9541	23
●	0642/execution1/validation	1.075	1.105	12
●	0900/execution1/validation	0.9483	0.9368	15

Figure F.2 MGR Model Trials Loss Accuracy