Silesian
University
of Technology

DOCTORAL THESIS

Algorithmic methods for detecting the tempo and time signatures of music tracks

Jeremiah Oluwagbemi ABIMBOLA
Student identification number: 4948

Programme: Informatics
Specialisation: Informatics

SUPERVISOR
Pawel Kasprowski, PhD, DSc
DEPARTMENT Applied Informatics
Faculty of Automatic Control, Electronics and Computer Science

CONSULTANT
Daniel Kostrzewa, PhD

Gliwice 2024

DEDICATION

This thesis is dedicated to God, whose guidance, wisdom, and strength have been my constant source of inspiration throughout this journey. In moments of doubt and uncertainty, I found solace in faith, which illuminated my path and provided me with the courage to persevere. I am profoundly grateful for the countless blessings bestowed upon me. Each challenge transformed into an opportunity for growth. To God, I dedicate this work as a testament to His grace and love. May it serve as a reflection of the knowledge and understanding You have instilled in me.

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to all those who have supported me throughout my PhD journey. This work would not have been possible without the contributions and encouragement of many individuals. First and foremost, I would like to thank my supervisor, Pawel Kasprowski (PhD, DSc) and co-supervisor Daniel Kostrzewa (PhD) for their invaluable guidance, support, and mentorship. Your expertise and insights have been instrumental in shaping my research and helping me navigate the complexities of academia. Thank you for discovering and believing in my potential and for pushing me to strive for excellence. I am mostly grateful for the support of my wife Sara Abimbola who stood by me through the entire period of my PhD journey, helping in the areas of translations and providing emotional support when I needed it the most.

I am also deeply grateful to the members of my dissertation committee, for their constructive feedback and encouragement. Your diverse perspectives have enriched my work and challenged me to think critically. A special thanks goes to my colleagues and fellow graduate students in the department of Applied Informatics. Your collaboration, and shared experiences made this journey enjoyable and fulfilling. I am thankful for the countless discussions, brainstorming sessions, and moments of laughter we shared. I would like to acknowledge the support of Silesian University of Technology for providing the resources and funding necessary for my research. The opportunities afforded to me through specific grants and scholarships were crucial in advancing my work. To my Dad and mum, my siblings and the rest of my family and friends, thank you for your unwavering support and love. Your encouragement during the challenging times kept me motivated. I am especially grateful to Pastor Alfred Richter of Arka Church Gliwice for being a great support during my medical crisis and for being my confidant throughout this process. Lastly, I want to extend my appreciation to all the participants in my research study. Your willingness to share your experiences made this work possible, and I am honored by your contributions.

Thesis title

Algorithmic methods for detecting the tempo and time signatures of music tracks

Abstract

Time signature is a fundamental component in the structure of music, controlling the rhythm and the arrangement of beats in a measure. It provides the framework within which music is composed, performed, and interpreted, and it plays a crucial role in defining the genre, mood, and style of a piece. Accurate detection of time signatures is essential for various applications in music analysis, transcription, and information retrieval. With the proliferation of digital audio formats, there is a growing need for robust methods capable of detecting time signatures directly from audio signals, as traditional methods based on Musical Instrument Digital Interface (MIDI) signals are proving inadequate. This study addresses the critical task of time signature detection in music, focusing on the analysis and enhancement of models to handle audio signals effectively. Traditional approaches have predominantly utilized models designed for MIDI signals, which are increasingly insufficient due to the prevalence of audio signals. The research objective was to develop new, sophisticated models capable of accurately processing audio data, thereby improving the robustness and accuracy of time signature detection. A comprehensive review of existing time signature detection methods such as Audio Similarity Matrix (ASM), Beat Similarity Matrix (BSM), identified significant challenges and limitations, particularly with audio signals. To address these, the study introduced a new model, the Mel-Frequency Cepstral Coefficient Similarity Matrix (MFCCSM), which leverages MFCCs for feature extraction, significantly enhancing detection accuracy. Additionally, due to the unavailability of an adequate dataset for rigorous validation, the Meter2800 dataset was created. This dataset consists of 2800 annotated audio samples, each 30 seconds long, and is critical for the comprehensive testing and validation of the models. The dataset was divided into training and test sets and categorized into four distinct meter classes. The relationship between meter and tempo within the dataset was analyzed using scatter plots and statistical summaries to ensure robustness and accuracy in the evaluation process. Furthermore, the research explored machine learning and deep learning models, developing innovative hybrid models that outperformed traditional methods. Machine learning algorithms like K-Nearest Neighbour(KNN), Support Vector Machine (SVM), Naive Bayes (NB), and Random Forest (RF) were explored for this task with SVM achieving the highest accuracy of 86.67% for binary classification and 74.29% for multi-class classification. Further advancements were observed with deep learning models. Convolutional Neural Networks (CNN), Convolutional Recurrent Neural Networks (CRNN), Residual Network-18 (ResNet18), and Residual Network-Long Short Time Memory (ResNet18-LSTM) were employed on MFCC and spectrogram features. For

binary classification, these models exhibited accuracies ranging from 88.00% to 89.00% with MFCC features and up to 99.67% with spectrogram features, with ResNet18 showing the highest performance. In multi-class scenarios, accuracies ranged from 80.29% to 86.00%, with ResNet18-LSTM achieving the highest accuracy. Eventually, optimization techniques, including Bayesian optimization and Genetic algorithms, were employed to fine-tune learning parameters, enhancing model performance and accuracy. By combining traditional models with advanced machine learning and deep learning approaches, the research sets a new benchmark for time signature detection accuracy and robustness in audio signals, laying a strong foundation for future advancements in audio signal processing and music information retrieval.

Key words

Tempo, Time signature, machine learning, algorithms, music tracks

Tytuł pracy

Algorytmiczne metody wykrywania tempa i metrum utworów muzycznych

Streszczenie

Metrum jest podstawowym składnikiem opisującym strukturę muzyki, kontrolującym rytm i aranżację uderzeń (pulsu, beatu) w miarę (takt). Zapewnia ramy, w których muzyka jest komponowana, wykonywana i interpretowana, oraz odgrywa kluczową rolę w definiowaniu gatunku, nastroju i stylu utworu. Dokładne wykrywanie metrum jest niezbędne dla różnych zastosowań analizy muzyki, transkrypcji i uzyskiwaniu informacji. Wraz z rozprzestrzenianiem się cyfrowych formatów audio rośnie zapotrzebowanie na metody zdolne do wykrywania metrum bezpośrednio z sygnałów audio, ponieważ tradycyjne metody oparte na zapisie MIDI (and. Musical Instrument Digital Interface) okazują się niewystarczające. Treść tejże pracy doktorskiej odnosi się do krytycznego zadania wykrywania metrum w muzyce, skupiając się na analizie i ulepszaniu modeli do efektywnej analizy sygnałów audio. Tradycyjne podejścia w przeważającej mierze wykorzystywały modele przeznaczone dla sygnałów MIDI, które nie są wystarczające ze względu na rozpowszechnienie sygnałów audio. Celem badań było opracowanie nowych, wyrafinowanych modeli zdolnych do dokładnego przetwarzania danych audio, a tym samym zwiększenie niezawodności i dokładności wykrywania metrum. Kompleksowy przegląd istniejących metod wykrywania metrum takich, jak Audio Similarity Matrix (ASM), Beat Similarity Matrix (BSM), zidentyfikował istotne wyzwania i ograniczenia, szczególnie w przypadku sygnałów audio. Następnie przedstawiony został nowy model, Mel-Frequency Cepstral Coefficient Similarity Matrix (MFCCSM), który wykorzystując MFCCs dla funkcji ekstrakcji, znacząco poprawia dokładność wykrytego metrum. W dalszych rozważaniach zauważono, że

dostępne zbiory danych są niewystarczające do rygorystycznej walidacji, w związku z tym utworzono zestaw danych Meter2800. Ten zestaw danych składa się z 2800 próbek audio z adnotacjami, każda o długości 30 sekund, mając kluczowe znaczenie dla kompleksowego testowania i walidacji modeli. Zbiór danych podzielono na zestawy treningowe i testowe oraz podzielono na cztery odrębne klasy metrum. Zależność między metrum a tempem w zbiorze danych została przeanalizowana za pomocą wykresów i podsumowań statystycznych w celu zapewnienia solidności i dokładności procesu oceny. Ponadto zbadano modele uczenia maszynowego i głębokiego, opracowując innowacyjne modele hybrydowe, które przewyższały tradycyjne metody. Klasyczne algorytmy uczenia maszynowego, takie jak k-najbliższych sąsiadów (K-Nearest Neighbors, KNN), maszyna wektorów nośnych (ang. Support Vector Machine, SVM), naiwny klasyfikator Bayesa (ang. Naive Bayes, NB) i las losowy (ang. Random Forest, RF) zostały wykorzystane do realizacji tego zadania, gdzie SVM osiągnął najwyższą dokładność na poziomie 86,67% dla klasyfikacji binarnej i 74,29% dla klasyfikacji wieloklasowej. Dalsze postępy zaobserwowano w przypadku modeli uczenia głębokiego. Sieci konwolucyjne (ang. Convolutional Neural Networks, CNN), sieci konwolucyjno-rekurencyjne (ang. Convolutional Recurrent Neural Networks, CRNN), sieć rezydualna (ang. Residual Network, ResNet18) i sieć rezydualno-rekurencyjna z komórkami LSTM (ang. Residual Network-Long Short Time Memory, ResNet18-LSTM) zostały wykorzystane w funkcjach MFCC i spektrogramów. Dla klasyfikacji binarnej modele te wykazywały dokładność w zakresie od 88,00% do 89,00% ze współczynnikami MFCC i do 99,67% z funkcjami spektrogramu, przy czym ResNet18 wykazał najwyższą skuteczność. W scenariuszach wieloklasowych dokładność wahała się od 80,29% do 86,00%, przy czym ResNet18-LSTM osiągnął najwyższą dokładność. Ostatecznie, techniki optymalizacyjne, w tym optymalizacja Bayesowska i algorytmy genetyczne, zostały zaangażowane w dostrajanie parametrów uczenia się, poprawiając dodatkowo skuteczność klasyfikacji. Połączczenie tradycyjnych modeli z zaawansowanymi metodami uczenia maszynowego i głębokiego, badane w niniejszej pracy doktorskiej wyznacza nowy standard dokładności wykrywania metrum i niezawodności na podstawie sygnałów audio, kładąc silny fundament dla przyszłych postępów w przetwarzaniu sygnałów audio i wyszukiwaniu informacji muzycznych.

Słowa kluczowe

Tempo, metrum, uczenie maszynowe, algorytmy, utwory muzyczne

Contents

Chapter 1

Introduction

1.1 Background to the Study

Music classification is a fundamental task in the field of music information retrieval (MIR), which involves automatically categorizing music tracks based on their musical attributes. Two of the key attributes used in music classification are tempo and time signature, which involve identifying the speed and rhythmic structure of a given piece of music. Accurately detecting the tempo and time signature can be challenging due to the complexity and variability of music, and it requires the use of various algorithms and techniques. Tempo and time signature detection can help to classify music into different genres, moods, and styles, and can be useful in a variety of music-related applications such as music recommendation systems and music analysis [42, 138].

Tempo refers to the perceived speed or pace of a musical piece and is often measured in beats per minute (BPM). On the other hand, time signature defines the rhythmic structure of a musical piece and is represented as a fraction, where the top number (which otherwise can be referred to as the numerator) indicates the number of beats per measure, and the bottom number (the denominator) represents the note value that receives one beat. It can be divided into two categories: regular and irregular time signatures. Regular time signatures have a consistent pattern of strong and weak beats, while irregular time signatures do not follow a predictable pattern. Regular time signatures can be further categorized as duple, triple, or quadruple. Duple time signatures have two beats per measure, with the first beat being stronger than the second. Triple time signatures have three beats per measure, with the first beat being the strongest and the third being the weakest. Quadruple time signatures have four beats per measure, with the first and third beats being stronger than the second and fourth. Furthermore, time signatures can also be classified as simple or complex. Simple time signatures have a basic beat pattern that can be divided into two equal parts, while complex time signatures have a basic beat pattern that can be divided into three or more equal parts. Therefore, a time signature notation

could be referred to as simple duple, simple triple, and simple quadruple, or compound duple, compound triple, and compound quadruple.

Regular time signatures, such as 2/4, 3/4, 4/4, and 6/8, have a predictable pattern of beats and are commonly used in different genres of music. Irregular time signatures, like 5/4, 7/8, and 9/8, have an unpredictable pattern of beats and are used to create rhythmic complexity and asymmetry in music. The time signature 5/4, for instance, is common in progressive rock and jazz music; 7/8 is used in Balkan and Middle Eastern music; and 9/8 is used in Irish, Balkan, and classical music. The use of irregular time signatures challenges the listener's expectations and adds interest to the rhythmic structure of a musical piece. In this study, the numerator holds greater significance than the denominator as it indicates the number of beats per measure. Thus, when dealing with a 4/4 time signature, the numerator value of 4 will be the primary focus of our analysis.

The tempo and time signature are closely connected and play important roles in music classification. Different time signatures can provide diverse rhythmic feelings, and the pace might have an impact on the piece's energy level. The combination of tempo and time signature may give a piece of music its distinct rhythm and feel, aiding its classification into a certain genre or style.

The concept of time signature cannot be fully understood without reference to the three levels in the time scale that were discovered by Klapuri et al. [86]; namely the tatum pulse level, the tactus pulse level that corresponds to a piece's tempo, and the harmonic measure level shown in Figure 1.1.

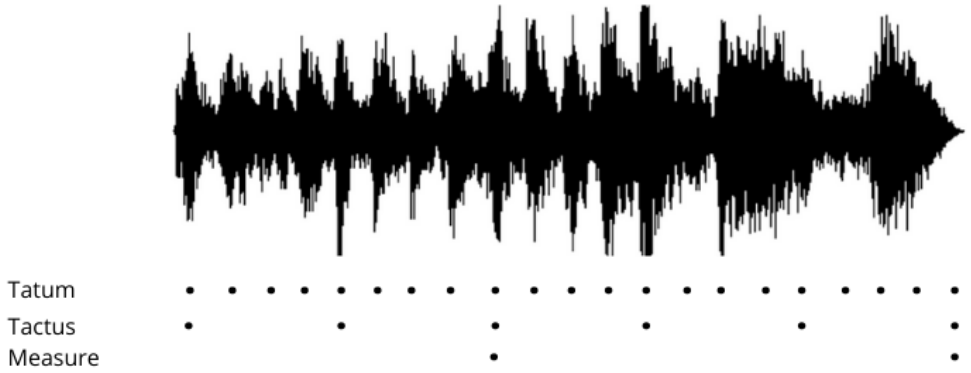

Figure 1.1: An audio signal with three metrical levels illustrated: tatum, tactus, and measure levels.

These three levels are interconnected and are used to determine the time signature of a piece of music. The tatum pulse level provides information about the smallest rhythmic units in the piece, which can be used to estimate the tactus pulse level. The tactus pulse level, in turn, can help to determine the meter of the piece, which is indicated by the time signature. The harmonic measure level is used to group beats into measures or bars, which is a fundamental aspect of music notation.

Understanding these three levels of time scale is important for developing effective algorithms for time signature detection, as it allows for the extraction of useful features at different levels of abstraction [6]. For example, features extracted at the tatum pulse level can be used for onset detection [14, 68], while features extracted at the tactus pulse level can be used for beat detection and tempo estimation. By taking into account all three levels of time scale, it is possible to develop more accurate and robust algorithms for time signature detection.

1.2 Audio and Midi Signals

In detecting time signatures, the type of input signal used is as important as the algorithm itself. There are two types of input signals that could be used: MIDI (Musical Instrument Digital Interface) and digital audio signals. MIDI is a protocol that is used to transmit information about musical notes, such as pitch, duration, and velocity, between electronic musical instruments and computers [131]. On the other hand, audio signals refer to the digital representation of sound waves that are captured by microphones and stored as digital files.

MIDI and audio signals have their own strengths and weaknesses when it comes to detecting time signatures, as shown in Table 1.1. MIDI files contain information about the timing and duration of musical notes, which can be useful for detecting the pulse of the music and identifying the time signature. Since MIDI files do not contain any audio information, they are much smaller in size and can be easily manipulated without losing quality. In addition, the timing and tempo of the music are usually very precise and consistent because they are created using electronic instruments.

On the other hand, audio signals are the direct representation of sound waves and therefore contain more detailed information about the music. They are useful for detecting the nuances of a performance, such as the slight variations in tempo and timing that occur naturally when musicians perform. However, audio signals take up much more storage space than MIDI files and can be more difficult to manipulate without losing quality. In addition, audio signals may contain background noise and other extraneous sounds that can interfere with the detection of time signatures.

In music, time signature and "meter" refer to the same concept and can be used interchangeably. For this study, the focus is on four meters, namely, 3, 4, 5, and 7. The choice of these meters is not random but deliberate and strategic. By concentrating on two regular meters (3/4 and 4/4) and two irregular meters (5/4 and 7/4), the study can gather extensive knowledge on the rhythmic patterns and characteristics of each meter, which can then be scaled up to other meters. Furthermore, the selected meters are prevalent in various genres of music and are likely to yield valuable insights into music classification and analysis.

Criteria	Midi	Digital Audio
Definition	A MIDI file is a piece of computer software that contains music information.	A digital audio refers to the reproduction and transmission of digital sound.
Information Data	Does not contain any audio information.	Contains recorded audio information.
Pros	Files of small size. Fit on a disk easily. The files are perfect at all times.	The exact sound files are reproduced. It replicates superior quality.
Cons	There is variation from the original sound.	They take more disk space with more minutes of sound. Files can get corrupted with a little manipulation.
Format type	Compressed.	Compressed.
Precision	Precise timing and tempo	May contain slight variations in tempo and timing
Quality	Can be easily manipulated without losing quality	Can lose quality when manipulated

Table 1.1: Comparison of MIDI and digital audio.

1.3 Objective of the Thesis

The objective of this study is to develop novel algorithmic methods for detecting the tempo and time signature of music tracks. Specifically, it aims to investigate the effectiveness of different feature extraction techniques, signal processing methods and machine learning algorithms for tempo and time signature detection. The proposed methods will be evaluated using a large dataset of music tracks from various genres and styles.

1.4 Motivation and Hypotheses

Time signature detection is a challenging task due to the high variability and complexity of musical rhythms and the presence of various musical genres and styles in the music industry today. While some of them are popularly known, such as jazz, blues, classical, etc., others are fusions of these popular ones, such as classical crossover, which combines elements of classical music with pop, rock, or other contemporary styles to create a fusion of traditional and modern sounds; electronic jazz fusion, which combines elements of jazz improvisation with electronic music to create a fusion of traditional and contemporary sounds; and Afro-Jazz, which combines elements of Afro-music and jazz, to name a few. The more complex the fusion, the harder it is to detect the time signature. However, time signature detection has significant applications in music analysis, music recommendation systems, and music education.

Various algorithms can be used to detect the time signature of a piece of music, ranging from classic to machine learning methods. Classic methods involve processing raw digital signals to extract acoustic features. Among the set of features extractable from music tracks, Mel-frequency cepstral coefficients (MFCC) serve as a valuable component for analyzing musical content. Furthermore, the features used for time signature detection encompass various elements such as zero-crossing rate, spectral bandwidth, spectral centroid, and energy, each contributing significantly to onset detection. These extracted features collectively aid in the onset detection process, consequently facilitating the accurate determination of the time signature within a piece of music. Based on these findings, the first hypothesis can be defined as follows:

Hypothesis 1 (H1): *Leveraging the Mel-Frequency Cepstral Coefficient Similarity Matrix as a tool for time signature detection can potentially yield significant advancements in accurately identifying musical time signatures.*

Classic methods are often limited by their inability to handle complex rhythms and variability in tempo. Therefore, to overcome the limitations of classic methods, optimization techniques such as dynamic programming and Bayesian models can be applied. Dynamic programming involves finding the optimal path through a sequence of beats that best fits the time signature of the music. Bayesian models use statistical inference to estimate the probability of different time signatures given the observed features. Furthermore, genetic algorithms can be used to improve these methods. GAs are a type of optimization algorithm that use the principles of natural selection and genetics to identify the optimal solution to a problem. They have been applied to various problems in music information retrieval (MIR), including tempo estimation and beat tracking. In the case of time signature detection, GA can help optimize the process of identifying the correct time signature for a given music track. It can be used to search through a large space of possible meter combinations and identify the one that best fits the musical features of the track. By using this optimization technique, the search space can be efficiently explored to identify the optimal set of weights to arrive at an accurate time signature, and the process can be repeated iteratively. This can result in improved performance for music classification systems that rely on the time signature. For this purpose, GA was used to investigate the following hypothesis:

Hypothesis 2 (H2): *Genetic algorithm may be used for optimization of Mel-Frequency Cepstral Coefficient-based model for time signature detection.*

In recent years, the music information retrieval (MIR) domain has undergone a profound transformation owing to machine learning methodologies. These approaches have notably elevated the landscape of time signature detection, with deep learning standing out as an influential technique. Particularly, the prowess of deep learning stems from its capacity to discover intricate features from vast collections of annotated music datasets. Through the training of neural networks on such expansive datasets, these models can identify patterns and relationships that surpass conventional digital signal processing techniques.

Within this paradigm, convolutional layers have emerged as pivotal components in deep learning architectures for musical analysis. The utilization of convolutional networks, specifically built upon architectures like ResNet, presents an intriguing avenue for advancing time signature detection. This particular architecture (ResNet), with their hierarchical representation and ability to capture hierarchical features, hold promise in utilizing music data for nuanced patterns crucial for accurate time signature identification. From this, the third hypothesis was defined:

Hypothesis 3 (H3): *Implementing a convolutional network utilizing the ResNet architecture holds potential for effective application in time signature detection.*

Transfer learning is another promising approach for time signature detection, where knowledge learned from a different domain is incorporated into the time signature detection task. For instance, a model trained to detect rhythms in speech signals can be adapted to detect time signatures in music. Also, hybrid approaches such as ensemble models that combine different techniques, like the classic methods along side machine learning methods, can provide even better accuracy. This potential led to the definition of the fourth hypothesis:

Hypothesis 4 (H4): *The ensemble model based on deep learning model performs better than the base classifier models.*

Overall, machine learning methods offer great potential for accurate and efficient time signature detection in music.

1.5 Project Methodology

The project methodology involves a combination of classic methods and machine learning techniques for meter detection. The classic methods include Audio Similarity Matrix (ASM), Beat Similarity Matrix (BSM), and Mel-Frequency Cepstral Coefficient Similarity Matrix (MFCCSM). These methods are widely used in music analysis and feature extraction. In addition to the classic methods, various machine learning algorithms are employed for classification. They include Support Vector Machines (SVM), Random Forest, Naive Bayes, and k-Nearest Neighbors (KNN). These algorithms are known for their ability to handle large datasets and extract meaningful patterns for classification tasks.

To further improve the detection task, deep learning architectures such as Convolutional Neural Networks (CNN), Recurrent Neural Networks with Long Short-Term Memory (RNN-LSTM), Convolutional Recurrent Neural Networks (CRNN) and Residual Networks (Resnets) are implemented. These architectures are capable of learning complex representations from raw audio data and have shown promising results in music classification tasks [118, 114]. Finally, to enhance the performance and robustness of the classification models, ensemble methods are introduced. They combine the predictions of multiple individual models to obtain a more accurate and reliable classification outcome. This approach helps mitigate the limitations of individual models and leverages the strengths of different algorithms.

With all these techniques, the data is first pre-processed, thereafter the feature extraction from audio signals is performed. The extracted features are used to train and evaluate the machine learning models. The models are hereafter trained on annotated music datasets and validated using appropriate evaluation metrics such as accuracy, precision, recall, and F1 score. The hyper-parameter are also tuned to optimize the performance of the models.

In terms of dataset for the study, a combination of popular datasets as well as a novel dataset (to augment for the deficiencies in the existing ones) are used. The reason for creating the new dataset was to ensure that the annotations were accurate and consistent. A diverse set of music pieces spanning different genres and styles were collected and ensured that they covered the range of time signatures in focus for this study. The datasets used included the GTZAN [153], which contains 1000 popular music pieces with annotations for beat and tempo information. Others include the FMA medium containing 25,000 audio files and MagnaTagATune [92]. Despite the availability of these datasets, it was observed that they were often limited in terms of the range of time signatures covered, and some were biased towards certain genres or styles of music. Hence, the reason for creating a new one by manually annotating a collection of music audio tracks with a wide range of time signatures. This ensures the accuracy and consistency of the annotations and also provided a more diverse range of music pieces for the study.

The research has the potential to make significant contributions to the field of MIR by advancing the state-of-the-art in tempo and time signature detection. The developed methods can be used in various music-related applications, such as music recommendation systems, music transcription, and music education.

Chapter 2

Music Classification

Music classification is a sub-field of music information retrieval (MIR) that deals with the task of automatically categorizing music tracks based on various musical attributes such as genre, mood, tempo, key, and instrumentation. Its main goal is to organize music into meaningful categories, making it easier for users to discover and explore new music based on their preferences.

It can be applied to a variety of contexts, including music recommendation systems, music search engines, and playlist generation. By automatically categorizing music into genres, for example, music recommendation systems can suggest new music to users based on their listening history and preferences. Music search engines can also use music classification to allow users to search for music based on genre, tempo, or other attributes. Playlist generation also benefits from it by automatically creating playlists based on specific attributes such as genre or mood.

Music classification systems typically involve two processing stages: feature extraction and classification [162, 104]. Feature extraction involves the extraction of relevant information from an audio signal that can be used for classification. A variety of low-level signal features such as zero-crossing rate [62], spectral bandwidth [104], spectral centroid [94], and signal energy have been used for this purpose. Additionally, higher-level features such as Mel-frequency cepstral coefficients (MFCCs), chroma features [112], and rhythmic features [90] have also been used for audio classification. MFCCs, for instance, are commonly used for music classification tasks, as they capture the spectral envelope of a sound signal. Chroma features, on the other hand, are used to capture the pitch content of music and are useful for tasks such as music genre classification. Rhythmic features, such as tempo and beat information, are important for tasks such as music mood and danceability classification.

After feature extraction, various classification algorithms can be used to assign a label to the audio signal based on its extracted features. The choice of algorithm depends on the specific task and the size and complexity of the dataset.

2.1 Genre Classification

Genre classification is one of many crucial tasks in music classification, focusing on categorizing music tracks into different genres based on their distinctive stylistic and musical characteristics as illustrated in Figure 2.1. Genres serve as a way to classify and organize music, providing listeners with a means to explore and discover tracks that aligns with their preferences. It can cover a wide range, including rock, pop, jazz, classical, hip-hop, electronic, country, and many more. Due to the elements of various music tracks and the diversity of musical genres, genre classification presents considerable challenges. For instance, instrumentation, rhythmic patterns, melodic structures, harmony, lyrics, and cultural settings are just a few of the elements that frequently come together to establish a genre. As an example, electric guitars are often used in rock music, and brass instruments are often used in jazz. Genres are also greatly influenced by pace and rhythmic patterns; reggae, for example, has a distinctly off-beat rhythm.

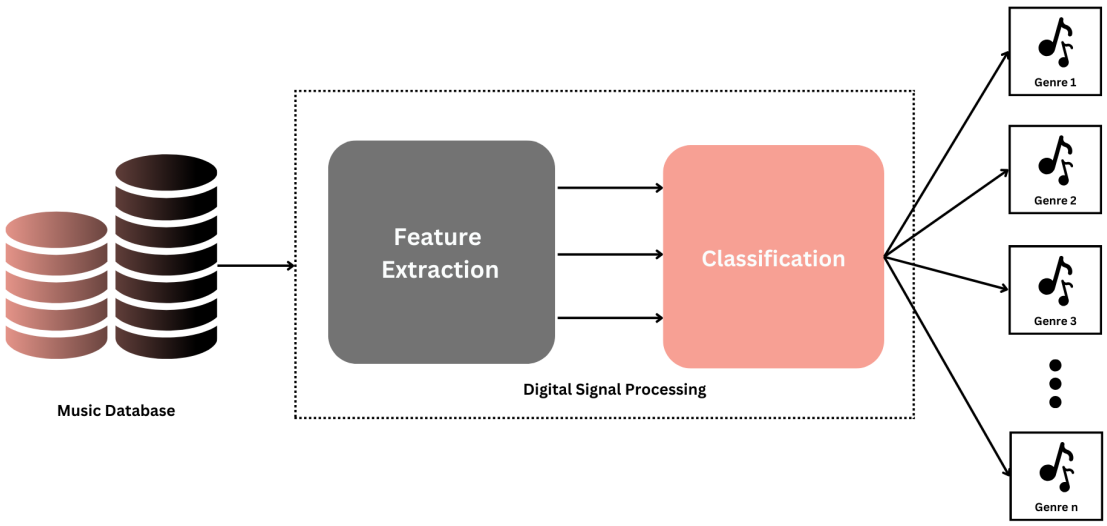

Figure 2.1: A classic music genre classification model

There are several uses for genre classification in the music industry and other. It helps music streaming platforms like Spotify, Apple Music etc to enable their users with the ability to find new music based on their favorite genres and to make customized radio stations and playlists based on the preferred genres. Additionally, categorizing music by genre helps scholars and analysts better understand musical trends, cultural influences, and the development of various genres over time.

Despite its challenges, genre classification has seen significant advancements in recent years, thanks to the development of sophisticated machine learning algorithms and the availability of large-scale annotated music datasets. These advancements have led to im-

proved genre classification accuracy and the ability to handle a wide range of music genres and sub-genres. Ongoing research continues to explore new approaches and techniques to enhance the accuracy and robustness of genre classification systems, further expanding our understanding and appreciation of the diverse world of music genres [132].

Tzanetakis and Cook [153] were the early pioneers in taking up the challenge to classify music tracks by genres. In doing so, they created a dataset which is very popular in the music MIR domain; the GTZAN dataset [147]. They developed an algorithm to automatically classify musical genres, which can be used to structure large collections of music available on the web. In their study, the researchers explored the automatic classification of audio signals into a hierarchy of musical genres. They proposed three feature sets for representing timbral texture, rhythmic content, and pitch content. The performance and relative importance of the proposed features were investigated by training statistical pattern recognition classifiers using real-world audio collections. Both whole file and real-time frame-based classification schemes were analysed and reported. Using the proposed feature sets, accuracy of 61% for ten musical genres was achieved.

Cataltepe et al.[29] conducted a study on music genre classification, focusing on the use of MIDI files and audio features extracted from MIDI data. They utilized the 3-root and 9-leaf genre dataset developed by McKay and Fujinaga [103]. To calculate the distances between MIDI pieces, a method called normalized compression distance (NCD) is employed. It approximates the Kolmogorov complexity of a string using its compressed length. NCD has previously been used in music genre and composer clustering tasks. In their approach, the MIDI pieces were converted to audio, and the audio features were used to train different classifiers. The results showed that the accuracies achieved by the classifiers using MIDI data or audio features alone were lower compared to the accuracies reported by McKay and Fujinaga, who employed domain-based MIDI features for their classification. However, when the MIDI and audio classifiers were combined, the accuracy improved but still fell short of the accuracies achieved by McKay and Fujinaga. The highest accuracies obtained for the root genres using MIDI, audio, and their combination were 0.75, 0.86, and 0.93, respectively, while McKay and Fujinaga achieved an accuracy of 0.98.

Xu et al.[161] in their paper presented an efficient and effective approach for automatic musical genre classification. They extracted a set of features to characterize music content while utilizing a multi-layer classifier based on support vector machines (SVMs) for genre classification. The SVMs are used to determine the optimal class boundaries between different music genres by learning from training data. The experimental results demonstrate that the multi-layer SVM approach exhibits good performance in musical genre classification, outperforming traditional Euclidean distance-based methods and other statistical learning methods. Some of the features used were beat spectrum, zero crossing rates, short-time energy, and mel-frequency cepstral coefficients. They developed three nonlin-

ear SVM classifiers to establish class boundaries between classic/jazz and pop/rock, classic and jazz, and pop and rock genres, respectively. Different music features were used for each SVM learning and classification task. The experiments validated the effectiveness of the multi-layer SVM learning method for accurate musical genre classification, highlighting its advantages over traditional methods and other statistical learning approaches.

Hareesh Bahuleyan also investigated the use of machine learning techniques on automatic music genre classification [10]. The performance of two classes of models was compared. The first approach was a deep learning method that utilized a convolutional neural network (CNN) model trained end-to-end to predict the genre label of an audio signal solely using its spectrogram. The second approach used hand-crafted features from both the time domain and the frequency domain. Four traditional machine learning classifiers were used for the training with these features and their performances were also compared. He also identified the features that contributed the most to this multi-class classification task. The experiments were conducted on the Audio set dataset, and an ensemble classifier that combined the two proposed approaches achieved an AUC value of 0.894. These results suggest that the combination of deep learning and hand-crafted features can improve the accuracy of genre classification in music information retrieval systems.

Pelchat et al.[121] in their study, examined the progress made in music genre classification using neural networks (NNs). They highlight the influence of factors such as song libraries, machine learning techniques, input formats, and types of NNs on the success of genre classification. The study also presented a research work on music genre classification, where they utilize spectrogram images generated from time slices of songs as the input for the NN with the aim of classifying songs into their respective musical genres.

2.2 Tempo Detection

Tempo detection is an important part of music classification as it gives significant information about a piece of music's rhythmic features. Tempo is the speed or pace of the music, which is usually measured in beats per minute (BPM). It has a big impact on a track's overall intensity, mood, and danceability. Music classification systems can assign appropriate genre labels, help in music recommendation, and enable a variety of music-related applications by precisely recognizing the tempo. The diversity and complexity of musical rhythms is one of the most difficult issues in tempo recognition. Different musical genres and styles have different rhythmic patterns, making tempo estimation a difficult process. Furthermore, tempo changes within a single piece of music, changes in consecutive sections, or subtle tempo alterations all contribute to the complexity.

To address these challenges, researchers have explored various techniques for tempo detection. These techniques often involve analyzing the rhythmic patterns of the music [90, 52], identifying periodicities in the audio signal [166], and extracting tempo-related

features [64]. These features can include beat information, onset detection, spectral analysis, and statistical measures. Machine learning algorithms have also been very useful to learn and model the relationships between these features and the corresponding tempos.

By providing a basic rhythmic descriptor, accurate tempo detection may considerably improve music classification algorithms in the identification of genres like as dance, rock, and classical by distinguishing between fast-paced and slow-paced music. Furthermore, tempo recognition may aid in the organization of music collections, the creation of playlists based on tempo preferences, and the implementation of tempo-aware music synchronization for a variety of applications such as fitness or dancing routines.

In a study done by McKinney et al.[105], eight different algorithms for musical tempo detection and beat tracking were extensively analyzed. These algorithms were evaluated as part of the 2006 Music Information Retrieval Evaluation eXchange (MIREX) using a standardized dataset comprising 140 musical excerpts. The dataset was annotated with beats by 40 different listeners, providing a comprehensive and diverse set of ground truth annotations. The primary objective of the evaluation was to assess the algorithms' performance in accurately predicting the perceptually salient musical beats and tempos of the excerpts. To assess the algorithms' ability to capture the most perceptually prominent beats and tempos, a variety of performance criteria were developed. The examination took into account a variety of characteristics, including musical genre, the presence of percussion, musical meter, and the most prominent perceptual tempo for each piece. The study sought to give insights into genre-specific problems and possibilities for tempo extraction and beat tracking by assessing algorithm performance in relation to musical genre; an approach that was also explored by Ajoodha et al. [7]. The existence of percussion in the song was also evaluated, as it may have a considerable impact on the rhythmic structure. Furthermore, the study analyzed how well the algorithms captured the musical meter and the most salient perceptual tempo, which are critical features of tempo extraction.

Eronen et al. in a study [49], approached tempo estimation in musical pieces with near-constant tempo. It involves three main steps: measuring the degree of musical accent over time, conducting periodicity analysis, and estimating the tempo. This approach introduces novel accent features based on the chroma representation to capture musical accent variations. The periodicity of the accent signal is then analyzed using the generalized auto-correlation function, followed by tempo estimation using k-Nearest Neighbor regression. The tempo estimation task was designed as a regression task, with the goal of estimating the continuous valued tempo given the periodicity observation. In classic k-NN regression, an object's property value is computed as the average of its k nearest neighbors' values using the Euclidean distance as a metric to compute distances to the nearest neighbors. To improve the accuracy, a re-sampling step is introduced, which applies to an unknown periodicity vector before finding the nearest neighbors. This additional step enhances the performance of the approach significantly.

A recent study to classify music by tempo using three features from MPEG-7: Audio Spectrum Centroid, Audio Spectrum Flatness, and Audio Spectrum Spread was conducted by Lazaro et al. in [93]. The primary objective was to classify music tempo based on its BPM value. The results reveal that the BPM value plays a critical role in the classification process, with the highest classification rate observed for slow and medium tempos. Despite the positive results, the experiment uncovered some flaws in the system, such as fast music being identified as medium tempo and a single slow music track being identified as medium tempo. The overall classification rate of the experiment was approximately 80%. As a result, the authors suggest that future work should focus on improving the accuracy of the system, specifically in the detection of fast-tempo music. De Souza et al. in [142] also confirmed the importance of the BPM value for tempo classification using a comparative examination of two artificial neural networks with distinct designs. Their study introduced a B-RNN (Bidirectional Recurrent Neural Network) model that was trained and tested to estimate the tempo of musical pieces in beats per minute (bpm) without the usage of external auxiliary modules. For quantitative and qualitative analysis, a large collection of 12,550 pieces, including percussion-only recordings, was compiled. The performance of the B-RNN model was compared to that of state-of-the-art models, and a state-of-the-art CNN was also retrained using the same datasets as the B-RNN. The results showed that tempo estimate was more accurate for the percussion-only dataset, implying that estimation can be more exact for tunes with a heavy percussion component.

Gkiokas et al. [59] conducted a study on tempo estimation and beat tracking algorithms, where they utilized percussive/harmonic separation of the audio signal. Their approach involved extracting filter-bank energies and chroma features from the respective components. To analyze periodicity, they convoluted the feature sequences with a bank of resonators. By incorporating metrical relations knowledge, they estimated the target tempo from the resulting periodicity vector. Following tempo estimation, the researchers employed a local tempo refinement method to improve the beat-tracking algorithm. The beat tracking involved computing beat sequences derived from the responses of the resonators and introducing a distance measure for candidate beat locations. They then employed a dynamic programming algorithm to determine the optimal "path" of beats.

2.3 Mood Classification

Mood classification is an interesting aspect of music classification because it seeks to categorize music based on the emotions and moods that it evokes in listeners. Due to the subjective character of emotions and the diversity of musical expressions, mood-based music classification is a difficult undertaking. Researchers have investigated several techniques to mood categorization that take advantage of both auditory and textual characteristics. These qualities include tonal characteristics, pace, timbre, lyrics, and semantic informa-

tion. One common approach to mood classification is the use of audio features extracted from the music signal. Features like spectral centroid, energy, and rhythm can provide insights into the acoustic properties of the music that contribute to specific moods. These features are then fed into machine learning algorithms, such as support vector machines (SVMs), random forests, or neural networks, to classify music into different mood categories.

The study done by Kim et al.[84] presented a music mood classification model based on arousal-valence (AV) values for a music recommendation system. The researchers collected music mood tags and AV values from 10 subjects and used the k-means clustering algorithm to classify the AV plane into 8 regions. For each region, representative mood tags were proposed based on statistical analysis. The findings revealed that some regions could be identified by representative mood tags, similar to previous models, while some mood tags overlapped across regions. The proposed model successfully expressed regions with multiple representative moods and moods distributed uniformly across most regions. The study also compared the proposed model with previous models and concluded that a well-defined music emotion or mood model is essential for improving the performance of music recommendation systems. The experimental setup involved gathering arousal/-valence values and mood tags from the subjects after listening to music clips. The collected data were analyzed to explore the relationship between mood and AV values, and the results were visualized through histograms and clustering.

In the pursuit of effective music discovery and personalized listening experiences, understanding the emotions conveyed in songs holds significant value. This study done by Dang et al. [33] explored the automatic classification of song moods based on lyrics and metadata, proposing various supervised learning methods for mood classification and the potential integration of automatically identified moods as metadata in a music search engine. Three machine learning algorithms, SVM, Naive Bayes, and Graph-based methods, were used for training classifiers. The experiments revealed that factors such as the artist, sentiment words, and giving more weight to words in the chorus and title parts were effective for mood classification. Additionally, the study indicated that the Graph-based method showed promise for improvement when rich relationship information among songs was available.

The rapid growth of internet usage and online music platforms has led to the need for automatic methods to classify music based on mood. In this research carried out by Patra et al. [119], a system was developed to classify moods of Hindi songs using audio features such as rhythm, timber, and intensity. A manually collected dataset consisting of 230 Hindi music clips, categorized into five mood clusters, was utilized, achieving an average accuracy of 51.56% for music mood classification.

Another method for mood classification is to use semantic analysis techniques. This method entails evaluating the music's lyrics or textual material to derive sentiment or

emotion-related information. Textual features derived from song lyrics or metadata can provide valuable information about the mood conveyed by the lyrics or the genre or artist associated with the music. To detect the emotional tone of the lyrics, natural language processing and sentiment analysis techniques can be used, finding keywords or patterns associated with distinct moods. This data may be integrated with audio characteristics to improve the precision of mood classification models.

Hu et al.[77] showed this in their research aimed to improve audio music mood classification by incorporating lyric text. They constructed a large ground truth dataset of 5,585 songs and 18 mood categories based on social tags, representing a user-centered perspective. Various lyric features and representation models were investigated and compared to an audio-based system. The study challenged prior findings, revealing that audio features did not consistently outperform lyric features. However, combining lyrics and audio features enhanced performance in many mood categories, though not all. The study acknowledged the challenges of creating ground truth data for music mood classification due to the subjective nature of music perception and the absence of widely accepted mood categories. They emphasized the necessity of multi modal approaches, as audio-only techniques had limitations in capturing high-level semantic music features. The methodology employed for constructing the ground truth dataset was explained, encompassing the collection of social tags and song lyrics from various sources, as well as the identification of mood categories using linguistic resources like WordNet-Affect [146]. Pre-processing techniques were applied to lyric text, and different lyric features, such as bag-of-words (BOW) representations, part-of-speech (POS) features, and function words, were explored and compared to audio processing and features extracted using the Marsyas system[152, 151]. Additionally, the performance of hybrid features combining lyric and audio sources was investigated, and feature selection methods were employed to identify the most significant lyric features for classification.

Hu et al. [76] further confirms that lyrics and audio can be combined in a hybrid mode to classify music based on mood. The study evaluated the use of lyrics in music mood classification by examining various lyric text features, including linguistic and text stylistic features. These features were compared and combined with audio features using two fusion methods. The results demonstrated that the combination of lyrics and audio features outperformed systems that solely relied on audio features. Furthermore, the study revealed that the hybrid lyric + audio system achieved comparable or superior classification accuracies with fewer training samples compared to systems using only lyrics or audio. The experiments were conducted on a large-scale dataset of 5,296 songs, each containing both audio and lyrics, representing 18 mood categories derived from social tags. These findings contribute to advancing the field of lyric sentiment analysis and automatic music mood classification, facilitating the practical use of mood as an access point in music digital libraries.

Moreover, advances in deep learning and neural network architectures have shown promise in mood classification. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have been employed to automatically learn hierarchical representations of music, capturing both low-level and high-level features relevant to mood classification. These deep learning models can extract intricate patterns and dependencies within the music signal, enabling more accurate mood predictions.

In the contemporary era, the generation of big data from music production and consumption necessitates automated and efficient management. Deep learning shines in this domain when presented with accurate and adequate data. Pyrovolakis et al. in [126] compared single channel and multi-modal approaches for music mood detection using Deep Learning architectures and found that the multi-modal approach outperformed the single channel models. They utilized the MoodyLyrics dataset [27], consisting of 2000 song titles with labels for four mood classes, and achieved a uniform prediction of mood, which has broad applications in various domains.

According to the study conducted by Hizlisoy et al. [71], a convolutional long short term memory deep neural network (CLDNN) architecture may be used to recognize musical emotions. They created a brand-new database of Turkish traditional music with 124 extracts and used this to test the effectiveness of their methodology. By feeding normal acoustic data into convolutional neural network (CNN) layers along with information derived from log-mel filterbank energies and mel frequency cepstral coefficients (MFCCs), they found that the LSTM + DNN classifier performed best when the novel feature set and standard features were combined, with an overall accuracy of 99.19% when performing 10-fold cross-validation. Additionally, the LSTM performs better than classifiers like the support vector machine (SVM), random forest, and k-nearest neighbor (k-NN).

The evaluation of mood classification models is crucial to assess their effectiveness. Researchers often employ large music databases with annotated mood labels to train and evaluate their models. Additionally, subjective user studies and feedback can provide insights into the perceived mood conveyed by the classified music. The continued research in this field can enhance our understanding of the relationship between music and emotions[140], contributing to applications such as personalized music recommendation, music therapy, and emotional content analysis.

2.4 Instrumentation Classification

The identification and classification of the many musical instruments used in a piece of music is a crucial component of musical classification. Due to the great variety of instruments and their various timbral properties, this endeavor presents a number of difficulties. To classify instruments, researchers have used a variety of methods, including as feature extraction, machine learning algorithms, and deep learning architectures. The

extraction of audio features that capture the individual qualities of various instruments is a popular method of classifying instruments. Three different feature schemes are widely known: perception-based features, MPEG-7-based features, and MFCC features [39]. The perception-based features included characteristics such as zero-crossing rate, root mean square, spectral centroid, bandwidth, and flux. The MPEG-7-based features focused on timbral descriptors, including harmonic centroid, harmonic deviation, harmonic spread, harmonic variation, spectral centroid, log attack time, and temporal centroid. Finally, the MFCC features involved transforming the signal to Mel scale, extracting filter bank outputs, and applying discrete cosine transform to obtain the MFCC coefficients. Feature selection techniques are used to optimize the feature sets used in classification and correlation-based approaches are utilized to assess the quality of features in relation to the class concept and their relevance to other features. The goal is usually to identify relevant features while eliminating redundancy to reduce computational costs and address the "curse of dimensionality." Researchers hope to accurately classify the instruments by capturing the distinctive spectral and temporal patterns displayed by each one through the analysis of these features.

The research carried out by Barbedo and Tzanetakis [12] explained that in musical signals, the spectral and temporal contents of instruments often overlapped, posing a challenge for instrument identification. It was reported that one approach to address this challenge was to search for isolated regions in the time and frequency domains where the content of a specific instrument appeared separated. The presented strategy involved identifying isolated partials and extracting features from them to infer the instrument likely responsible for generating the partial. The method was reported to require the existence of at least one isolated partial for each instrument in the signal, and if multiple isolated partials were available, a more accurate classification was achieved.

Instrument classification tasks have seen widespread usage of machine learning systems. These models help to extract features and instrument labels and how they relate to one another. They can recognize patterns, extract pitch and make predictions based on learnt representations since they are trained on big datasets comprising audio samples from a variety of instruments according to Kostek [87]. As a decision-making tool, feature vectors derived from recorded musical sounds are used to train artificial neural networks (ANNs). The study conducted by Essid et al. [50] also used statistical pattern recognition approaches to concentrate on musical instrument identification within solo musical phrases. The study comprised a sizable sound database with extracts from varied performances and recording circumstances, including eleven distinct instrument classes with new descriptors and more than 150 signal processing characteristics were examined. The most pertinent characteristics for each instrument pair were determined using feature selection approaches including inertia ratio maximization and evolutionary algorithms. Gaussian mixture models (GMMs) and support vector machines (SVMs) were used to accomplish

the classification job, and the findings showed that pairwise optimized feature subsets and SVM classification employing a radial basis function kernel led to greater recognition rates.

Deep learning has completely changed instrument categorization in recent years. This task has been successfully completed using convolutional neural networks (CNNs) and recurrent neural networks (RNNs). From audio spectrograms or waveforms, CNNs may automatically learn hierarchical representations that include both low-level and high-level data important for classifying instruments. RNNs, on the other hand, can accurately recognize instruments by capturing temporal relationships in music and utilizing sequential patterns.

Mahanta et al in their study [99] modelled an artificial neural network which was trained to perform classification on twenty different classes of musical instruments. The model utilized only the mel-frequency cepstral coefficients (MFCCs) of the audio data. The training was conducted on the full London Philharmonic Orchestra dataset, which encompassed twenty instrument classes from the woodwinds, brass, percussion, and strings families. Experimental results demonstrated that the model achieved state-of-the-art accuracy. The deep ANN architecture consisted of multiple layers with varying numbers of neurons. The layers terminated in an output layer with the same number of neurons as the classification task's classes. During training, the feature values were multiplied by weights and added with bias terms. The weights were updated after each epoch through forward and backward propagation, driven by a chosen loss function. The input feature values traversed through subsequent layers' neurons. To introduce non-linearity and selectivity in the network, activation functions were applied. The Rectified Linear Unit (ReLU) activation function was employed for all hidden layers. This function activated neurons with positive values resulting from the aforementioned computations. The ANN model's performance, as validated by the experimental outcomes, showcased its effectiveness in achieving accurate classification of musical instruments.

Blaszke and Kostek [17] also showed that the use of deep learning technique has shown outstanding performance in music instrument classification. They presented a novel approach for automatically identifying instruments in audio excerpts using sets of individual convolutional neural networks (CNNs) per instrument. By preparing a new dataset containing underrepresented instruments and the exploration of on-the-fly instrument, it was shown that the presented approach achieved a precision of 93% and an F1 score of 0.93 for instrument identification using a simple convolutional network based on MFCC. The effectiveness of identification varied depending on the instrument, with drums being more easily identifiable compared to guitar and piano. The proposed solution outperformed other methods in terms of AUC ROC and F1 score, offering higher accuracy in instrument recognition.

Datasets play a crucial role in instrument classification research. Large-scale annot-

ated datasets, such as the RWC [60] Instrument Sound Dataset and the NSynth dataset [48], provide researchers with labeled audio samples of various instruments. These datasets facilitate the training and evaluation of instrument classification models, allowing for comparative analysis and bench-marking of different techniques. Real-world applications of instrument classification include instrument recognition in music transcription, automated music production, and interactive music systems.

2.5 Singing Voice Detection

Singing voice detection is a fundamental task in music classification that focuses on distinguishing the presence of a human singing voice in an audio recording. It plays a crucial role in various applications, including music information retrieval, automatic transcription, karaoke systems, and vocal separation[116]. Singing voice detection algorithms aim to accurately identify and isolate the singing voice from the accompanying instrumental or background music as depicted from the Figure 2.2. It is an active research area with ongoing developments and challenges. The exploration of new audio features, the development of innovative deep learning architectures, and the creation of larger and more diverse datasets are essential for further advancements in this field. The continued progress in singing voice detection will contribute to enhancing music understanding, enabling innovative applications, and improving the overall user experience in various music-related domains. Monir et al [109] offered an overview of singing voice recognition methods, with a focus on cutting-edge algorithms like convolutional LSTM and GRU-RNN. It emphasized the significance of singing voice identification as a preprocessing step for various music analysis applications. Datasets like Jamendo [20] and RWC have been the main set of datasets used by existing approaches.

Challenges in singing voice detection arise due to various factors, such as background noise, overlapping vocal and instrumental sounds, and different singing styles. Researchers have addressed these challenges by developing robust feature representations[57], incorporating contextual information, and utilizing advanced signal processing techniques, such as source separation[78] and harmonic modeling[101]. Real-world applications of singing voice detection extend beyond music classification. Being able to do this, plays a crucial role in the field of automatic music transcription, where accurate separation and identification of the singing voice contribute to the accurate transcription of lyrics and melody. Singing voice detection also forms the basis for vocal enhancement and remixing, enabling the manipulation of vocals in audio production. Another challenge lies in the mismatch between artificially generated datasets and real polyphonic music. Artificially combining clean speech clips with instrumental music clips to simulate polyphonic vocals may not accurately represent the complexities of real singing vocals accompanied by instrumental accompaniments[26, 79].

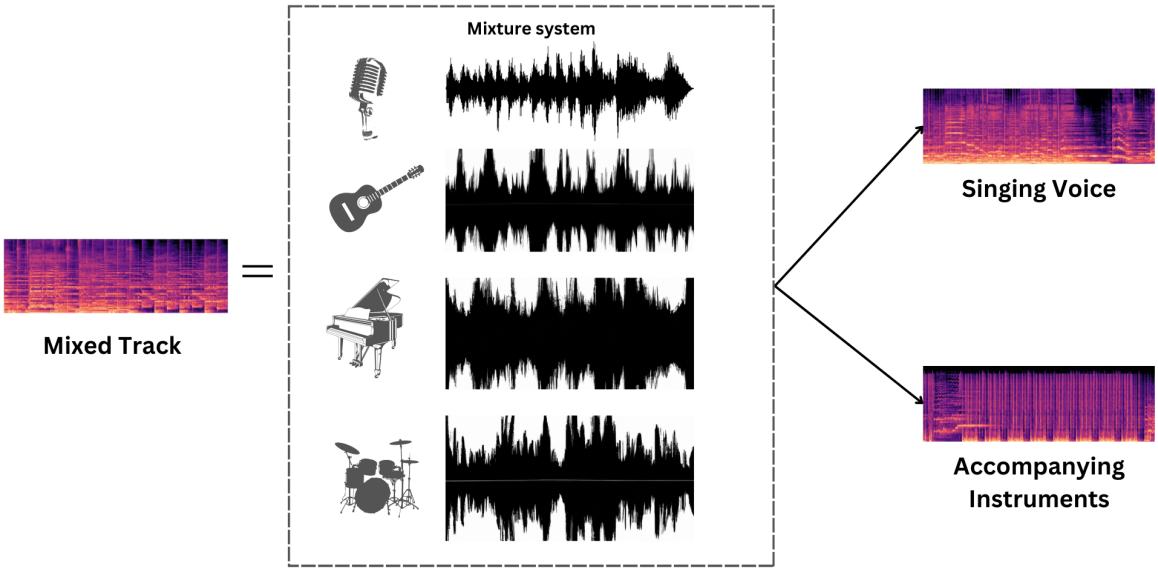

Figure 2.2: A singing voice detection model

Several approaches have been proposed for singing voice detection. One common strategy is to utilize audio features that capture the characteristics of the human voice. These features include pitch, spectral centroid, harmonic content, and mel-frequency cepstral coefficients (MFCCs). By analyzing these features, researchers can differentiate between vocal and non-vocal segments in the audio signal.

In the past, a study was conducted by Regnier et al. [128] to investigate the problem of locating singing voice in music tracks. The approach relied on the extraction of characteristics specific to singing voice, such as harmonicity, formants, vibrato and tremolo. The study focused on vibrato and tremolo characteristics. Sinusoidal partials were extracted from the musical audio signal and the frequency modulation (vibrato) and amplitude modulation (tremolo) of each partial were studied to determine if the partial corresponded to singing voice. The rate and extent of both vibrato and tremolo were estimated for each partial and a partial selection was operated based on these values. A second criteria based on harmonicity was also introduced. Each segment could then be labelled as singing or non-singing. Post-processing of the segmentation was applied to remove short-duration segments. The proposed method was evaluated on a large manually annotated test-set and achieved very close results to a usual machine learning approach.

Machine learning techniques play a vital role in singing voice detection. Supervised learning algorithms are trained on large datasets such as the ikala dataset[30] or jamendo dataset containing labeled examples of singing and non-singing segments. These models learn to classify audio segments based on the extracted features, enabling accurate singing voice detection.

Using neural networks to recognize singing voices, the Schlüter and Grill in a study

[134], examined the efficacy of label-preserving audio modifications for music data augmentation. Several audio changes were implemented and their utility was evaluated, drawing inspiration from current voice recognition research. The researchers found that pitch shifting was the most efficient augmentation technique, according to their data. The classification error was significantly reduced by 10 to 30% when paired with temporal stretching and random frequency filtering. On two open datasets, the researchers' method produced cutting-edge results. In the area of music information retrieval, they predicted that the addition of audio data will significantly enhance a number of sequence labeling and event recognition tasks.

Deep learning has also gained significant attention in recent years for singing voice detection where convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have shown promising results in this area. CNNs can capture local and global dependencies in the audio signal, while RNNs excel in modeling temporal dependencies. Hybrid architectures, such as CNN-RNN models, have also been explored to leverage the strengths of both network types. Recent advancements in deep learning have led to improved performance in singing voice detection. As such, models trained on large-scale datasets have demonstrated the ability to generalize to a large extent, unseen music recordings and handle complex audio scenarios. Furthermore, researchers are also exploring the integration of multi-modal information, such as lyrics and music scores, to enhance singing voice detection accuracy and robustness.

With deep learning architecture, Leglaive et al. [95] developed a novel approach for detecting singing voice in an audio track using a Bidirectional Long Short-Term Memory (BLSTM) Recurrent Neural Network (RNN). Utilizing the sequential structure of short-term feature extraction in music, the classifier was created to take into account both past and future temporal context in order to assess whether or not a singing voice is present. The BLSTM-RNN's several hidden layers made it possible to extract from low-level features a condensed representation that was appropriate for the job. The suggested technique greatly outperformed state-of-the-art methods on a widely used database, according to experimental results.

One interesting aspect of deep learning is the use of transfer learning. Hou et al. [74] addressed the limited availability of well-labeled datasets for singing voice detection (S-VD) by proposing a data augmentation method using transfer learning. They combined clean speech clips with voice activity endpoints and instrumental music clips to create artificial polyphonic vocals for training a vocal/non-vocal detector. Despite the mismatch with real polyphonic music, transfer learning was applied to transfer knowledge from the artificial dataset to a smaller, matched polyphonic dataset. This approach significantly improved S-VD performance, resulting in an F-score increase from 89.5% to 93.2%.

To improve generalization performance, ensemble learning have been reported to be very helpful [58]. It combines a number of separate models. Deep ensemble learning mod-

els combine the benefits of deep learning models and ensemble learning such that the resulting model performs better in terms of generalization. In a previous study, You et al [163], explored different neural network models and stacked ensembles for singing voice detection. Models such as CNN, LSTM, convolutional LSTM, and capsule net were evaluated using input features like MFCC and spectrogram. The CNN model with spectrogram inputs achieved the highest accuracy, reaching 91.8% for the Jamendo dataset. Among the stacked ensemble methods, a voting strategy with five models demonstrated comparable performance with an accuracy of 94.2% for the Jamendo dataset, while being computationally efficient.

In summary, this chapter delves into the domain of music information retrieval, focusing on two key tasks: instrument classification and singing voice detection. It begins by exploring instrument classification, highlighting the effectiveness of deep learning techniques, particularly convolutional neural networks (CNNs) and recurrent neural networks (RNNs), in accurately identifying various musical instruments from audio recordings. The chapter discusses the importance of large-scale annotated datasets, such as RWC and NSynth, in training and evaluating these models. It also touches upon real-world applications of instrument classification, including music transcription, automated music production, and interactive music systems. The chapter then shifts its focus to singing voice detection, a crucial task for distinguishing human singing from instrumental or background music. It emphasizes the challenges posed by factors like background noise, overlapping sounds, and diverse singing styles. The chapter explores various approaches to singing voice detection, including feature-based methods that utilize audio characteristics like pitch and spectral centroid, as well as machine learning techniques, particularly deep learning architectures like CNNs and RNNs. It highlights the advancements made by researchers using deep learning models like BLSTM-RNN and transfer learning to achieve state-of-the-art performance in singing voice detection and eventually highlights the potential of ensemble learning methods for further improving generalization performance in singing voice detection. The summary of the various classification an detection types are shown in Table 2.1.

Table 2.1: Summary of music classification studies

Classification Type	**Year**	**Author**	**Models Used**	**Results**
Mood	2011	Kim et al.	k-means clustering	Representative mood tags
Mood	2009	Dang et al.	SVM, Naive Bayes	Effective for mood classification
Mood	2010	Hu et al.	SVM, RF, KNN,	Not mentioned
Mood	2022	Pyrovolakis et al.	Deep Learning architectures	Multi-modal approach outperformed
Tempo Estimation	2006	Gouyon et al.	Autoregressive models	Tempo estimation accuracy
Tempo Estimation	2012	McLeod et al.	ACFFT, Gaussian mixture model	Accurate tempo estimation
Genre	2003	Tzanetakis and Cook	SVM, KNN, decision trees	High genre classification accuracy
Genre	2010	Li et al.	SVM, ANN, fuzzy KNN	Not Mentioned
Instrument	2008	Deng et al.	Perception-based, MPEG-7, MFCC features	Feature extraction
Instrument	2010	Barbedo and Tzanetakis	Isolated partials	Not Mentioned
Instrument	2006	Essid et al.	GMMs, SVMs	greater identification rate
Instrument	2021	Mahanta et al.	Deep ANN	State-of-the-art accuracy
Instrument	2022	Blaszke and Kostek	CNN	F1 score 0.93
Singing Voice Detection	2009	Regnier et al.	Machine learning	Not Mentioned
Singing Voice Detection	2015	Schlüter and Grill	BLSTM-RNN	State-of-the-art results
Singing Voice Detection	2020	Hou et al.	Transfer learning	Improved performance
Singing Voice Detection	2022	Ganaie et al.	Ensemble learning	High accuracy

Chapter 3

Time Signature Detection

3.1 Introduction

Music time signature detection is a fundamental task in music analysis. It involves determining the underlying rhythmic structure and organization of a musical piece. The time signature, typically represented as a fraction at the beginning of a musical score, indicates the number of beats in each measure and the type of note that receives one beat. In a study conducted by Andrew and Mark [106], the detection of meter was explored by analyzing the metrical structure of individual bars. Excerpts from compositions by Bach were used for this analysis, which resulted in an F-measure of 80.50%. They approached the problem by leveraging the hierarchical tree structure of notes, as depicted in Figure 3.1. The evaluation of the guessed metrical tree was performed at three levels: sub-beat, beat, and bar. Each level was examined to determine if it matched exactly with the corresponding level of the actual metrical tree. If a match occurred, it was considered a true positive. On the other hand, if there was a mismatch, it was counted as a false positive, indicating a clash between the guessed and actual metrical structures.

Building upon their previous work, Andrew and Mark further refined the model in another study [107] to achieve more accurate meter detection. This enhanced model incorporated two musicological theories. The first theory emphasized a reasonably consistent tatum rate without significant discontinuities, where a tatum represents the smallest rhythmic unit. The second theory focused on the similarity between notes and tatums, ensuring that notes align closely with the underlying tatum structure. In this model, each state represents a single bar and consists of a list of tatums belonging to that bar, along with a metrical hierarchy that defines which tatums serve as beats and sub-beats. The tatum list is generated by arranging the tatums in chronological order. Additionally, the downbeat of the next bar is obtained for accurate tracking of the metrical structure. The metrical hierarchy of a state includes specifications such as the number of tatums per sub-beat, the number of sub-beats per beat, the number of beats per bar, and the duration of

the anacrusis (the portion of music preceding the first downbeat). The anacrusis duration is determined by counting the number of tatums that occur before the first downbeat of a given musical piece.

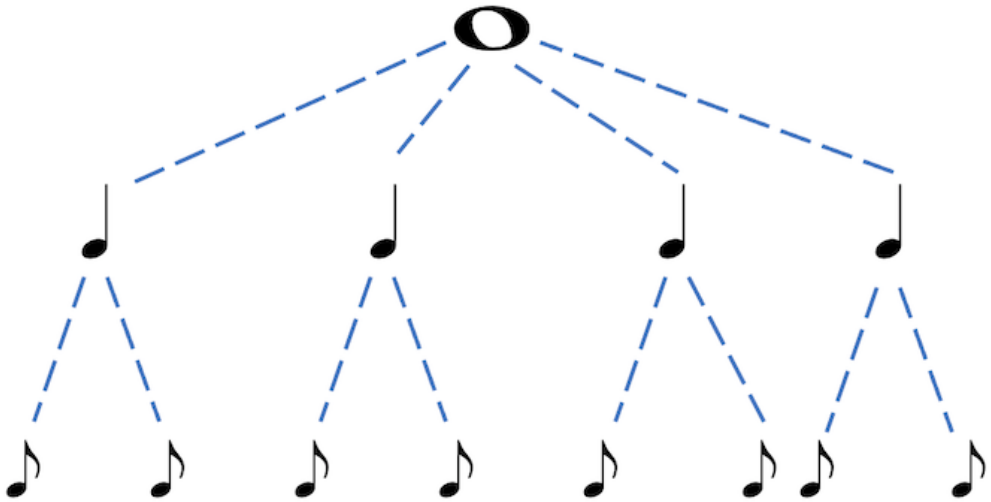

Figure 3.1: The hierarchical tree structure of notes – the metrical structure of a $\frac{4}{4}$ bar (1 whole note = 4 quarter notes = 8 eighth notes).

In the rapidly growing music industry, platforms like Apple Music, Spotify, and Audio Mack play a crucial role in curating playlists for their users. To achieve accurate and personalized recommendations, these platforms rely on genre classification systems that group songs based on similarities in rhythm (time signature and tempo) and harmonic content. Many researchers have made successful attempts to classify different genres of songs using various techniques and models [10, 118, 51, 88]. However, one aspect that has received relatively less attention is the use of time signature as a feature for genre classification. Estimating the time signature of a song is challenging, but incorporating this information could potentially enhance the overall accuracy of genre classification. This unexplored area presents an opportunity for further research and improvement in music genre classification algorithms. Detecting the time signature from an audio recording poses several challenges due to the complexity and variability of musical rhythms.

One of the main challenges in time signature detection is the presence of rhythmic variations and syncopations within a piece of music. Different sections of a song may have varying rhythmic patterns or unexpected accents, making it difficult to discern a consistent time signature. Additionally, certain musical genres, such as jazz or Latin music, often incorporate intricate syncopated rhythms that deviate from traditional metrical structures. These variations require robust algorithms capable of capturing the nuances and irregularities inherent in the music [6].

Another challenge arises from the potential overlap of multiple time signatures within a single composition. In complex musical arrangements or pieces that change time signatures throughout, accurately identifying and tracking these changes can be challenging. Transitions between time signatures may occur abruptly or gradually, and the algorithm needs to handle these shifts smoothly and accurately.

Furthermore, the presence of polyrhythms, where multiple rhythmic patterns coexist simultaneously, can pose difficulties in time signature detection. Polyrhythms are common in various music styles, such as African or Afro-Cuban music, and they involve the simultaneous use of different time signatures. Detecting and disentangling these overlapping rhythmic layers requires advanced algorithms capable of extracting and analyzing multiple rhythmic structures simultaneously.

Also, the use of musical ornaments and expressive timing by performers can further complicate time signature detection. Musicians often employ expressive techniques, such as accelerando [135] (a term that describes a gradual increase in tempo or speed of a musical passage), which includes slight tempo changes that could make the underlying time signature difficult to discern. Handling these expressive nuances and distinguishing them from genuine changes in time signature requires sophisticated analysis techniques and a deep understanding of musical context.

Addressing these challenges in time signature detection requires the development of robust algorithms that combine statistical analysis, signal processing techniques, and vast musical knowledge. By leveraging advanced pattern recognition and feature extraction methods, along with domain-specific insights, we aim to improve the accuracy and reliability of time signature detection algorithms.

Researchers have made significant progress in developing methods for automatically extracting metadata from musical audio signals. Some of the metadata that can be obtained through these algorithms include Pitch detection [75, 113], onset detection [38, 65, 66], key signature estimate [110], and tempo extraction [8, 159]. The ultimate goal is to enable computers to possess the same capabilities as human listeners, allowing them to interpret music and provide insights into specific musical properties. This advancement opens up a wide range of applications, such as automated transcription, playlist generation, and music information retrieval systems. The topic of automated music analysis and retrieval is a recurring theme at the International Symposium on Music Information Retrieval (ISMIR) and continues to drive innovation in the field. Ultimately, advancements in this field contribute to a better understanding of music structure, facilitating tasks such as automatic transcription, music analysis, and creative applications in music composition and performance.

One study by Gulati et al. [67] focused on estimating the meter of music with an irregular time signature, specifically $\frac{7}{8}$, using a set of Indian classical music and multiple comb filters. They employed a filter bank with a specific number of comb filters per filter

bank, which was determined by twice the integer multiple of the tatum duration plus one, accounting for rounding factors. The meter was determined based on the filter with the highest output energy in each filter bank.

Another study by Bas de Haas and Anjan [36] aimed to detect meter and the first downbeat position in symbolic music. They introduced a model called PRIMA that utilized MIDI signals for inner metric analysis instead of the commonly used auto-correlation method. The PRIMA model employed a probability-based approach for classifying different meter classes (duple, triple, and quadruple). The model was trained and tested on two datasets created by music enthusiasts and outperformed the MIDItoolbox meter detection method.

The application of genetic algorithms to signal processing has received significant attention in research generally. In a study by Kaur et al. [83], they utilized a genetic algorithm (GA) in combination with improved Mel-Frequency Cepstral Coefficient (MFCC) speech characteristics for speaker and speech recognition using Deep Neural Network (DNN) for classification purposes. The process involved two main steps. First, MFCC feature extraction was performed, capturing important characteristics of the speech signal. Then, a genetic algorithm was applied to optimize these features, aiming to enhance the performance of the subsequent classification task. The GA was utilized to determine the optimal weights and biases of the DNN, using a fitness function specific to the problem at hand.

These studies demonstrate the efforts made to address meter detection challenges and propose alternative approaches for accurate and efficient analysis. By exploring different techniques, such as comb filters and MIDI-based analysis, researchers are making progress in improving the accuracy and performance of meter detection algorithms.

There are two main approaches used for the time signature detection: the classic/-manual approach and the deep learning approach. In the classical approach, researchers use digital signal processing techniques to analyze music as evident in the study done by Meinard et al [111]. They employ methods like comb filters [85] and Fourier transforms to study aspects such as frequency, tones, duration, and beats. This approach requires a sound understanding of concepts like tempo, timing, and audio spectrum. In contrast, the deep learning approach relies on deep learning models to detect singing voices. Both approaches share some common ideas, such as converting audio signals to log spectrograms for easier analysis. In the past, many researchers used MATLAB and C++ for implementing these approaches [91, 156], but Python gained popularity in recent years due to its simplicity and the availability of powerful machine learning libraries like scikit-learn [120] and librosa [102]

3.2 Detection Methods

The detection methods as it concerns time signature ranges from digital signal processing (DSP) methods to machine learning (ML) methods. Some of these methods have been discussed in our paper published in Sensors [6]. They offer distinct approaches for time signature detection in music. DSP methods rely on mathematical transformations and signal analysis techniques such as Fourier transforms, auto-correlation, and beat tracking to identify rhythmic patterns and extract time signatures directly from audio signals. These methods are typically rule-based, deterministic, and grounded in the physical properties of the signal. On the other hand, ML methods utilize algorithms that learn from large datasets of labeled music to recognize patterns and make predictions. Techniques like neural networks, support vector machines, and hidden Markov models can be trained to detect time signatures by learning complex features and temporal dependencies in the data. While DSP methods emphasize precise, formulaic analysis, ML approaches leverage data-driven learning to adapt and generalize from examples.

3.3 Digital Signal Processing Methods

In the methods discussed in this section, digital signal processing techniques are employed to analyze audio samples. These techniques involve various tasks such as window framing as shown in Figure 3.2, filtering, and Fourier analysis [96]. When working with audio tracks, they are divided into smaller segments known as frames. Each frame typically corresponds to a short duration of time, such as 0.0227 milliseconds for audio sampled at a rate of 44.1 kHz. It's important to note that this duration is much shorter than what the human ear can perceive, which is around 10 milliseconds.

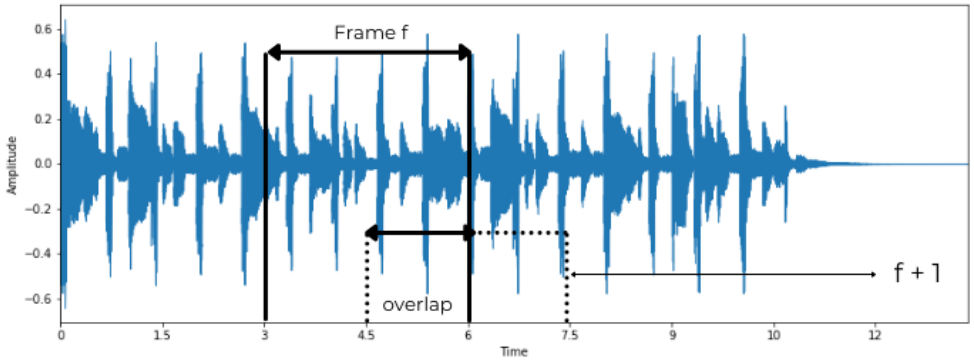

Figure 3.2: One of the most common methods of audio preprocessing – splitting the whole signal into frames with overlapping.

To prevent spectral leakage and ensure accurate analysis, a windowing function is applied to the audio frames. This function eliminates the samples at both ends of the

frame, ensuring a smooth and continuous signal. The choice of windowing function plays a crucial role in the analysis process. Additionally, it's common to use overlapping frames in the analysis. This means that subsequent frames have a certain degree of overlap with the previous frames. The overlap helps in maintaining continuity and capturing transient information in the audio signal. These processes are integral to the analysis of audio signals, and they contribute to the accurate representation of the underlying frequency and time-domain characteristics of the music. A more detailed explanation of these processes can be found in Table 3.1.

Year	Method	Dataset	Data	Accuracy(%)
2003	SVM [61]	Self generated	Audio	83
2003	ACF [154]	Excerpts of percussive music	Audio	73.5
2004	SSM [122]	Greek music samples	Audio	95.5
2007	ASM [32]	Commercial CD recordings	Audio	75
2009	ACF, OSS [72]	Usul	MIDI	77.8
2009	BSSM, ASM [56]	Generated samples	Audio	95
2011	Comb Filter [67]	Indian Music DB	Audio	88.7
2013	SVM [155]	Generated Samples	Audio	90
2014	RSSM [130]	MIDI keyboard scores	MIDI	93
2020	Annotation Workflow [28]	ACMUS-MIR	Audio	75.06

Table 3.1: Summary of classic estimation methods.

Out of all the manual methods used in detecting time signature, two models have been considered in this research.

3.3.1 ASM (Audio Similarity Matrix)

The ASM (Audio Similarity Matrix) model, described by Gainza in [32], aims to identify repeating bars in a song by comparing longer audio segments (bars) with shorter audio fragments (a fraction of a note). The underlying idea is that different parts of a song often have repeating patterns. A clear block diagram is shown in Figure 3.4 To clearly understand this process, a sample audio in a $\frac{6}{8}$ time signature 3.3 (`pop.00009.wav`) from the GTZAN dataset is used.

First, the model creates a spectrogram of the song using a frame length that corresponds to a certain percentage of the beat duration. This spectrogram provides a visual representation of the song's frequencies over time as shown in Figure 3.5. By analyzing the spectrogram, the model identifies the initial note of the song. Next, the model constructs a reference ASM by calculating the Euclidean distance between two frames, denoted as $m = a$ and $m = b$, (where m represents the frame index, a is the index of the first frame,

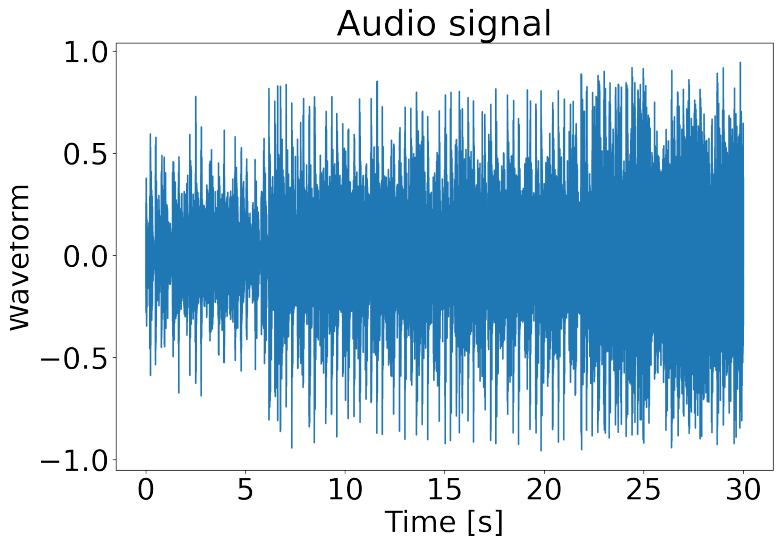

Figure 3.3: An audio signal from the GTZAN dataset

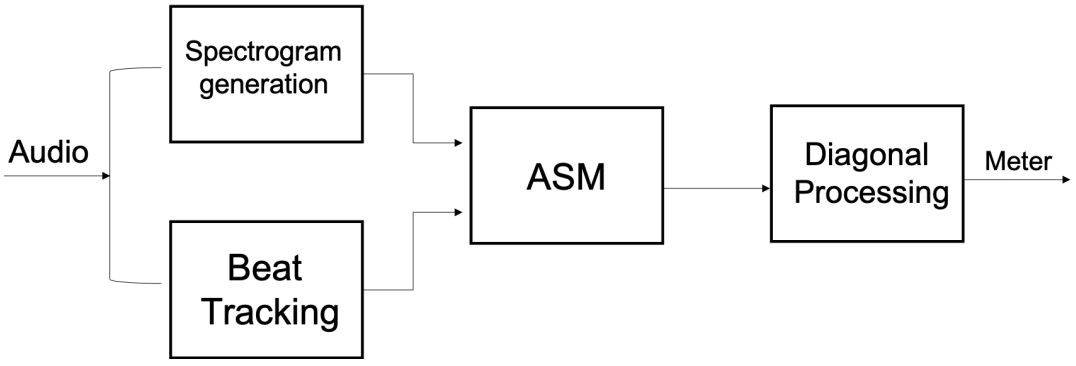

Figure 3.4: The audio similarity matrix architecture

and b is the index of the second frame) starting with the first note obtained earlier given by Equation 3.3. This step allows the model to capture small musical occurrences, such as brief notes. To further explore the song's structure, a multi-resolution ASM approach is employed. This involves creating additional audio similarity matrices that represent various bar lengths. These matrices provide a measure of similarity between different segments of the song as depicted in Fig. 3.6, and focuses on the diagonal of this matrix.

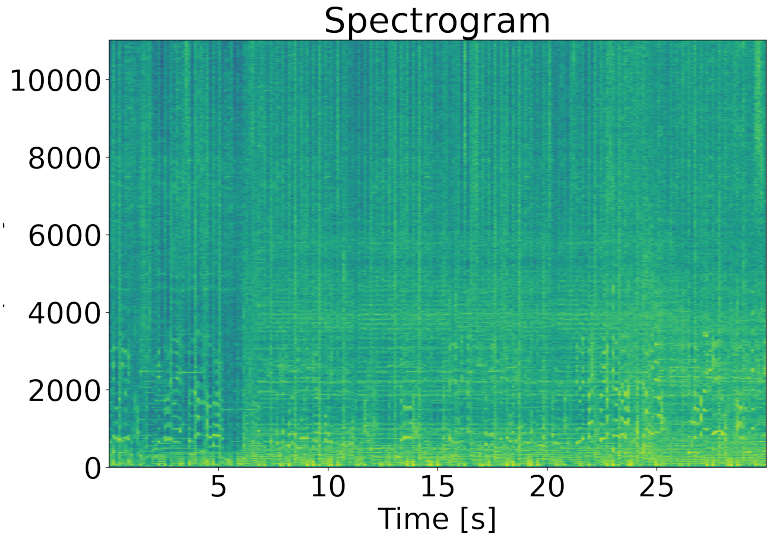

Figure 3.5: An audio signal from the GTZAN dataset

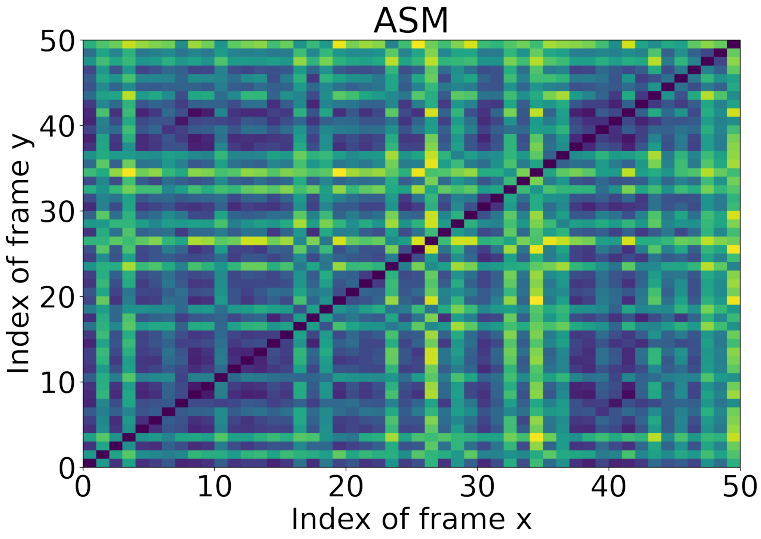

Figure 3.6: Audio similarity matrix of the audio frames

As a result of this, there will be many diagonals to be analysed. The diagonals extracted from one side of the symmetric ASM provide information about the similarities between musical components separated by different numbers of bars. For example, one diagonal $D1$ could represent components separated by one bar, while another diagonal $D2$ could

represent components separated by two bars. The diagonal represents the similarity of frames within the song. In an ideal case, the diagonal values would be all ones (1.0), indicating that the corresponding frames are perfectly similar, resulting in a white color in the matrix. Eventually the average diagonal is calculated in Equation 3.1 and the result from this function can be seen in Figure 3.7.

$$[htbp]d_i = mean(diag(\mathrm{A}SM_i))d = -d + \max(|d|) \tag{3.1}$$

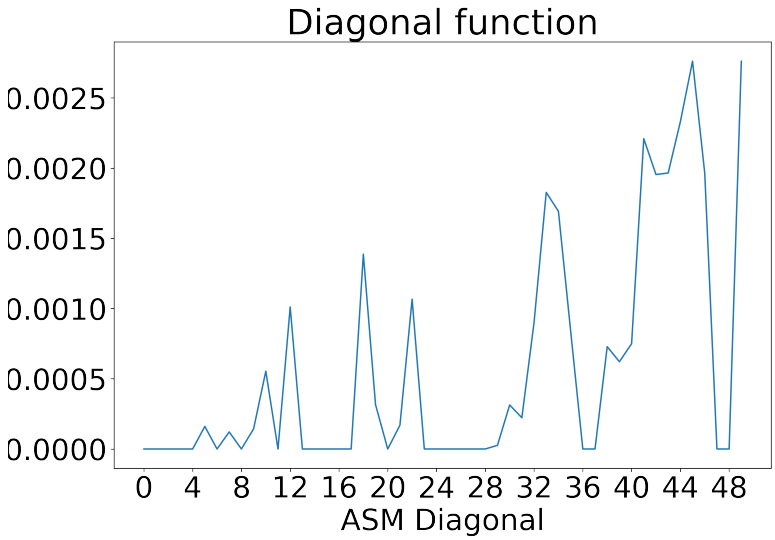

Figure 3.7: Diagonal function of the ASM

The other values in the matrix range between 0.0 and 1.0, reflecting the similarity between different frames. By analyzing the ASM and examining the diagonal values, the ASM model can determine the meter of a song, providing insights into its rhythmic structure and recurring patterns. This meter detection approach considers a wide range of meter candidates. These candidates include duple (c=2) and triple (c=3) meters, as well as common multiples of duple and triple meters such as c=4, 6, and 8. Additionally, complex meters such as c=5 and 7 are also considered. To account for multiples of each meter candidate, which represent similar beat patterns occurring at different musical bars, a weighted comb filter is applied to the function d. The resulting function, denoted as T_c, assigns higher weights to closely spaced musical bars and is given by:

$$T_c = \sum_{p=1}^{lt} \frac{d(p \times c)}{1 - \frac{p-1}{lt}}, \quad c = 2, \ldots, 8 \tag{3.2}$$

where lt corresponds to $\lfloor \frac{nb}{8} \rfloor$ and nb corresponds to the number of beats of the piece of music. The result of this calculation is shown in Figure 3.8.

$$ASM(a, b) = \sum_{k=1}^{N/2} [X(a, k) - X(b, k)]^2 \tag{3.3}$$

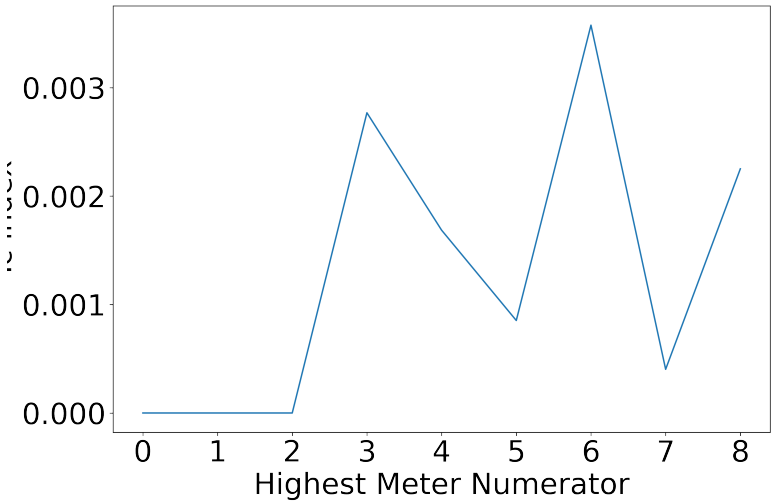

Figure 3.8: Meter is obtained by the highest point

where a and b are spectrogram frames, N is the total number of frames and X is a function of the frames.

Recall that the time signature for this audio track was $\frac{6}{8}$, and from the image in Figure 3.8, the highest point on the graph is 6 which makes the result a good detection.

The Euclidean distance measure was found to be insufficient for the analysis conducted by Gainza due to its reliance on magnitude. To overcome this limitation, the cosine distance measure was adopted as an alternative. The cosine distance measure takes into account the angle between two vectors rather than their magnitudes.

Mathematically, the cosine distance between two vectors x and y is calculated as:

$$\text{cosine\_distance}(x, y) = 1 - \frac{x \cdot y}{\|x\| \cdot \|y\|} \tag{3.4}$$

where $(\cdot)$ represents the dot product of the vectors and $\|x\|$ and $\|y\|$ denote the Euclidean norms of the vectors x and y, respectively. By using the cosine distance measure, the analysis becomes less reliant on the magnitude of the vectors and focuses more on the angular similarity between them. This allows for a more robust comparison and evaluation of musical components, taking into consideration their directional alignment rather than just their magnitudes. Therefore, by using the cosine distance, the ASM is given by (3.5).

$$ASM(a, b) = 1 - \frac{k = 1}{\sqrt{\sum_{k-1}^{s} X(a, k)^2} * \sqrt{\sum_{k-1}^{s} X(b, k)^2}} \tag{3.5}$$

The Kullback-Leibler (K-L) divergence method [23, 18] is another extensively used

approach for computing vector similarities shown in the equation 3.6

$$ASM(a,b) = \sum_{k=1}^{N/2} X(a,k)^* \log_e \left(\frac{X(a,k)}{X(b,k)} \right) \tag{3.6}$$

3.3.2 BSM (Beat Similarity Matrix)

The BSM model, originally proposed by Gainza and further described by Ajay Srinivasamurthy et al. in [144], is another approach for estimating the meter of a song. This model utilizes beat tracking and spectrogram analysis. First, the song's spectrogram (S) is divided into frames that are synchronized with the beats of the music. These beat synchronous frames m are obtained by using the beat locations of the song. Each frame (m) corresponds to a specific beat interval within the song. From these beat synchronous frames, a similarity matrix (B) is generated according to Equation 3.7, where $Si(k,m)$ represents the similarity value between the k-th frame of the beat synchronous frames and the m-th frame.

$$Bi = Si(k,m) \tag{3.7}$$

In this equation, for the i-th beat (Bi), the value of m ranges from the beat location ti to $ti+1$, where ti represents the beat locations and $t0$ is set to 1. Mathematically this can be expressed as, $b_i, t_i \leq m < t_{i+1}$ and $t_0 = 1$

The Detection Function (DF) is a calculation performed on the audio signal s[n] using spectral flux. It serves as a condensed and efficient representation for detecting the start of musical events and tracking beats. To compute the DF, the audio signal is divided into frames with a fixed time resolution of $t_{DF} = 11.6$ ms. Each frame has a length of 22.64 ms and overlaps with adjacent frames by 50%. The DF is obtained by smoothing the values of the spectral flux and applying half wave rectification, resulting in the processed detection function $\tilde{\Gamma}(m)$.

The onset detector finds the peaks of the processed detection function based on specific criteria. To ensure only salient onsets are retained, detected onsets that are less than 5% of the maximum value of $\tilde{\Gamma}(m)$ are ignored. The inter-onset-interval (IOI) histogram is computed as a histogram of the number of pairs of onsets detected over the entire song for a given IOI value.

The tempo of the song is determined by analyzing $\tilde{\Gamma}(m)$ and applying the General State beat period induction algorithm used in [35]. The tempo estimation is carried out in frames of 6 seconds, resulting in a tempo map that spans the entire song. To track the beats, a dynamic programming approach developed by Ellis[47] is employed. The process is given by the equation 3.8. This method utilizes the induced tempo period and the smoothed, normalized detection function to accurately track the beats at different time points within the song. The beat tracking process covers all beats detected in the song. By

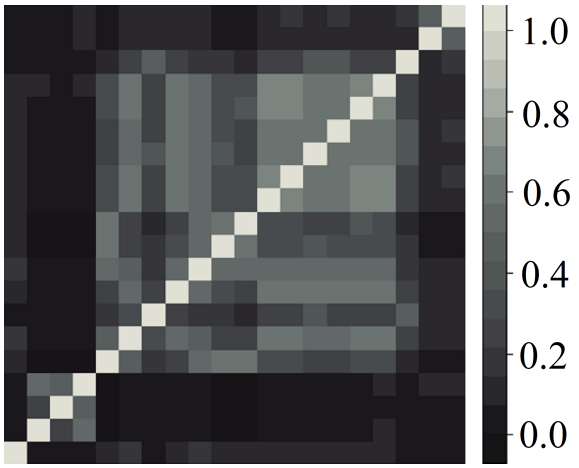

Figure 3.9: Beat similarity matrix showing the diagonal. Each box represents spectrogram frames

combining these techniques, the tempo and beats of the song can be accurately estimated and tracked, providing valuable insights into its rhythmic structure.

$$Tempo(bpm) = \frac{60}{\tau_p t_{DF}} \tag{3.8}$$

The Beat Similarity Matrix (BSM) is then calculated like in Figure 3.9 by measuring the cross-correlation between beats in the song. Since the BSM is symmetrical, we only need to compute half of the matrix to avoid redundancy. To save computational resources, we focus on the first 100 beats of the song. The diagonal elements of the BSM reflect the similarity between different beats, and we compute the average value along these diagonals to capture the overall beat similarity pattern.

By analyzing the similarity matrix and focusing on its diagonal, which represents the similarity of frames within the song, the BSM model determines the meter of the song. This approach was applied to the Indian classical music tracks and achieved a high accuracy rate of 68.8% for one of the datasets used in the study.

3.4 Machine Learning Methods

The approaches shown in this section, as well as those detailed in Table 3.2, employ neural networks to extract the time signature as a feature that may be used as input for additional calculations or classification issues in the MIR domain rather than precisely approximating it. These machine learning models, their mode of operation and architectures are further discussed in chapter 4.

In a study, Varewyck et al. [155] approached the music meter estimation problem as a classification task. They utilized the Support Vector Machine (SVM) algorithm to tackle this task. To perform the analysis, they first employed an external beat tracker to obtain

beat-level information. Additionally, they conducted spectral envelope and pitch analysis. Furthermore, they incorporated the concept of Inter-Beat-Interval (IBI) similarity, which was previously introduced by Gouyon and Herrera (2003) [61]. The IBI similarity was calculated using the cosine similarity measure, as shown in the equation below. This measure captures the similarity between two successive IBIs. By applying these methods and developing a hypothesis, they were able to estimate the meter of the music. The SVM classifier, along with the analysis of beat-level features and IBI similarity, contributed to their meter estimation approach.

$$CS(b) = \frac{\langle \vec{z}(b-1), \vec{z}(b) \rangle}{\|\vec{z}(b-1)\| \|\vec{z}(b)\|} \tag{3.9}$$

where b is the beat, $\vec{z}(b-1)$ and $\vec{z}(b)$ are low dimensional vectors grouped by related features. With a balanced collection of 30 song samples, they eventually developed an automated meter classification approach with the optimal feature combination that caused an error of around 10% in duple/triple meter classification and about 28% in meter 3, 4, and 6.

Year	Method	Dataset	Accuracy(%)
2011	CNN [41]	MSD	Not stated
2016	CNN [43]	Multiple datasets	90
2017	CNN [123]	MSD, MagnaTagATune	88
2019	TCN [19]	Annotated dataset	93
2019	CNN [125]	MSD	89
2019	CRNN [55]	Beatles	72
2019	GMM-DNN[127]	Poetry corpus	86

Table 3.2: Summary of deep learning signature estimation methods

In another study conducted by Sander Dieleman et al. [41], an unsupervised pre-training approach was employed using The Million Song Dataset. The learned parameters from this pre-training phase were then transferred to a convolutional network. The network architecture consisted of 24 input features, representing timbre properties of the audio as shown in Figure 3.10. These features were fed into two separate input layers, one for chroma characteristics and the other for timbre characteristics. Each input layer had its own convolutional layer, and the outputs of these layers were then passed through a max-pooling operation. The max-pooling layer was designed to be invariant to small temporal shifts of less than one bar (up to 3 beats). This ensured that the network could capture rhythmic patterns regardless of their precise temporal alignment. It is worth noting that the study did not explicitly mention the accuracy in terms of time signature classification since it was not the primary focus of their research. The main objective was to investigate unsupervised pre-training and transfer learning using the convolutional

network architecture.

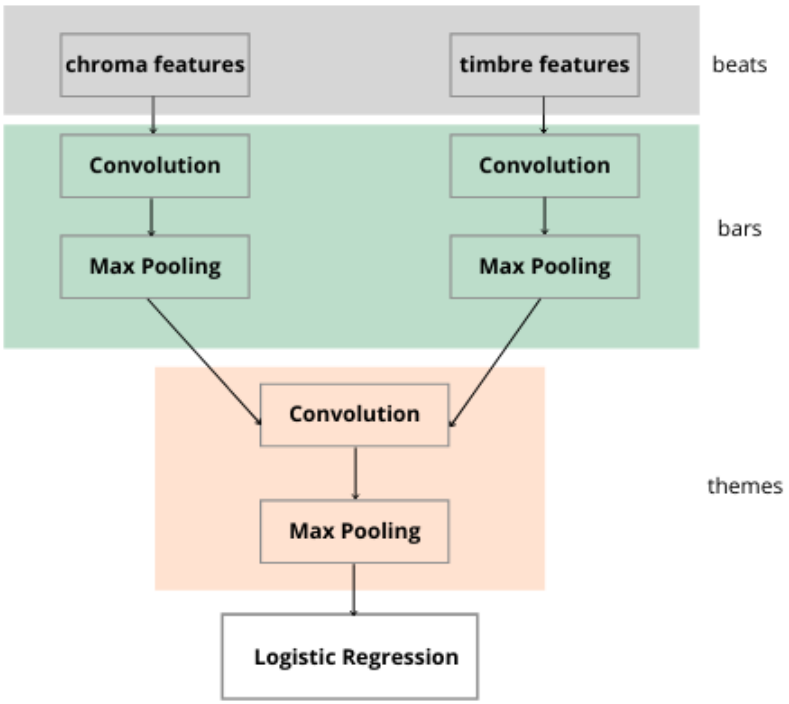

Figure 3.10: The convolutional neural network architecture block diagram with two kinds of input features (chroma and timbre).

Downbeat detection, which involves tracking the meter at a higher metrical level, has been explored in the field of music analysis. In the study by Srinivasamurthy et al. [143], they investigated downbeat detection using deep learning features. Similarly, Durand and Essid [43] proposed a random field method specifically for audio signal downbeat detection. To tackle the task, Durand and Essid incorporated four additional characteristics related to harmony, rhythm, melody, and bass into the audio signal. They also separated the signal into tatum-level segments. Adapted convolutional neural networks (CNN) were then employed to learn features from each characteristic, capturing their specific properties. The networks' final and/or penultimate layers were combined to create a feature representation that described the observation feature functions. This representation was then fed into a Markovian model known as a Conditional Random Field (CRF) to produce the downbeat series. The performance of the model was evaluated using statistical tests such as Friedman's test and Tukey's honestly significant difference (HSD) criterion. The results indicated an improvement in F-measure of approximately 0.9% when using features from the last layer, with a 95% confidence interval.

Using waveform data as input, a deep learning strategy was used in the work by Pons et al. in 2017 [123] to automatically categorize audio samples, including the estimation of meter. They employed a convolutional neural network (CNN) architecture – a common architecture for music genre classification as shown in Figure 3.11, which is frequently

applied to the categorization of musical genres.

Input, front-end, back-end, and output were the four key parts of the CNN architecture employed in the study. The front-end component further examined the input spectrogram after the input component had processed the waveform data. The front-end was a single-layer CNN with several filter shapes and two branches, with the upper branch concentrating on timbral characteristics and the bottom branch concentrating on temporal data. The shared back-end component included three convolutional layers with 512 filters each, along with two residual connections, two pooling layers, and a dense layer.

This back-end component played a crucial role in extracting relevant features from the spectrogram data. Two models were combined in the study: one model utilized classical audio feature extraction methods with minimal assumptions, while the other model focused on analyzing spectrograms. By integrating these models and leveraging musical domain knowledge, the study successfully obtained meter tags. This approach demonstrated the effectiveness of deep learning techniques in automatically categorizing audio samples and estimating meter. By utilizing waveform data and a well-designed CNN architecture, they were able to extract meaningful features and obtain meter tags, showcasing the importance of incorporating music domain knowledge when training data is limited.

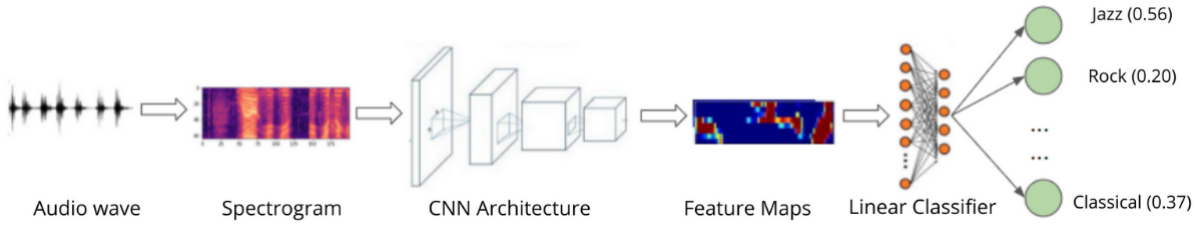

Figure 3.11: A typical convolutional neural network architecture used for time signature detection – audio signal processed into spectrogram which is an input to convolutional layers, and then an outcome is an input to classical artificial neural network.

Sebastian Böck et al. in a similar research [19], demonstrated that tempo estimation can be improved by incorporating a beat tracking process within a multi-task learning algorithm. This approach takes advantage of the strong interconnections between tempo estimation and beat tracking tasks, which has been successfully utilized in optimization tasks in other research areas [167, 137]. The multi-task learning approach extends a beat tracking system that utilizes temporal convolutional networks (TCNs). The output of the beat tracking system is then fed into a tempo classification layer. Instead of using raw audio as input, the authors employ dilated convolutions on a heavily sub-sampled low-dimensional attribute representation. A 16-dimensional function vector is generated by applying several convolution and max pooling operations to the log magnitude spectrogram of the input audio signal. The log magnitude spectrum is used because it aligns with the frequency perception of the human ear [25, 34]. The spectrogram is computed

using a window size and FFT size of 2048 samples, with a hop size of 441 samples. The convolutional layers in the network consist of 16 filters, with kernel sizes of 3×3 for the first two layers and 1×8 for the final layer.

The performance of the proposed method was evaluated on various datasets that contain beat and tempo annotations. The results were compared to reference systems for both tasks. The findings indicate that the multi-task formulation achieves state-of-the-art performance in both tempo estimation and beat tracking. Notably, significant improvements were observed when the network was trained on tempo labels while largely ignoring the beat annotations.

The underlying beat tracking system in this approach draws inspiration from two well-known deep learning methods: the WaveNet model [100] and the latest advancements in musical audio beat tracking, which employ a bi-directional long short-term memory (BLSTM) recurrent architecture. To train the system, annotated beat training data were represented as impulses at the same temporal resolution as the input features (i.e., 100 frames per second). Unlike other approaches that typically use a single dataset divided into training and test sets, this study utilized different datasets for training and evaluation purposes.

In the research conducted by Purwins et al. [125], the application of deep learning techniques in audio signal processing was explored across various tasks, including beat tracking, meter identification, downbeat tracking, key detection, melody extraction, chord estimation, and tempo estimation. The study demonstrated the effectiveness of deep learning models in processing diverse audio sources such as speech, music, and environmental sounds.

Unlike traditional signal processing approaches that often rely on Mel-Frequency Cepstral Coefficients (MFCCs) as the dominant feature, deep learning models in audio processing commonly employ log-mel spectrograms as the primary input feature. The use of log-mel spectrograms allows capturing important frequency and temporal information in the audio signals. The study highlighted the advantages of convolutional neural networks (CNNs) in this domain stating that they have a fixed receptive field, enabling them to consider a specific temporal context for predictions. Moreover, the flexibility of CNNs allows for easy adjustment of the context size, providing control over the level of temporal information considered. Although the study did not explicitly state which specific deep learning method outperformed the others, the choice of method often depends on the nature of the data being analyzed. For example, in the analysis using the Million Song Dataset, a 29-second log-mel spectrogram was reduced to a 1x1 feature map using 3x3 convolutions interleaved with max-pooling. This approach yielded a promising result, achieving an AUC (Area Under the Curve) of 0.894, indicating good performance in the task.

Rajan et al. [127] proposed a meter classification scheme utilizing musical texture

features (MTF) and employing both a deep neural network (DNN) and a hybrid Gaussian mixture model-deep neural network (GMM-DNN) framework. The performance of the proposed system was evaluated using a newly created poetry corpus in Malayalam, one of the widely spoken languages in India. The system's performance was compared to that of a support vector machine (SVM) classifier. To extract the features, the authors employed a combination of 13-dimensional Mel-frequency cepstral coefficients (MFCCs) with a frame size of 40ms and a frame shift of 10ms, along with seven additional features: spectral centroid, spectral roll-off, spectral flux, zero-crossing rate, low energy, root mean square (RMS), and spectrum energy. For the neural network architecture, rectified linear units (ReLUs) were selected as the activation function for the hidden layers, while the softmax function was used for the output layer. The hybrid GMM-DNN framework achieved an accuracy of 86.66% in meter classification. The individual accuracies for the DNN and GMM-DNN models were 85.83% and 86.66%, respectively.

Fuentes et al. [55], showed that Convolutional Recurrent Neural Network (CRNN) can make significant improve in the performance of meter detection. A combination of non-machine learning and deep learning approaches was used to estimate downbeats and extract the time signature of music. The deep learning approach was a combination of convolutional and recurrent networks proposed in their previous work [54]. The Beatles dataset, known for its well-annotated features such as beats and downbeats, was utilized in this study. The researchers considered a set of labels, denoted as Y, which represented the beat positions within a bar. They specifically focused on bar lengths of 3 and 4 beats, corresponding to $\frac{3}{4}$ and $\frac{4}{4}$ meters commonly found in music. The output labels, y, were determined by two variables: $b \in \mathcal{B} = \{1, \ldots, b_{max}(r)\}$, and the number of beats per bar $r \in \mathcal{R} = \{r_1, \ldots, r_n\}$. The beat length relates to the time signature of the musical piece. The model achieved a certain level of success in downbeat tracking, but it faced challenges in identifying rare music variations that deviated from the global time signature consistently. For instance, the model tended to estimate more pieces with a and $\frac{4}{4}$ time signature than $\frac{3}{4}$. Despite this limitation, the model demonstrated an improvement in downbeat tracking performance, as indicated by the mean F-measure increasing from 0.35 to 0.72.

In conclusion, the studies presented in this chapter have shed light on various approaches and techniques used for different aspects of music analysis, including beat tracking, meter identification, tempo estimation, downbeat detection, and meter classification. The use of deep learning methods, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), has proven to be effective in tackling these tasks. These models leverage the power of large-scale data processing and feature extraction from audio signals, allowing for more accurate and efficient analysis. Additionally, the combination of different approaches, such as incorporating non-machine learning techniques alongside deep learning, has shown promising results. Hybrid models, such as the

GMM-DNN framework, have demonstrated improved performance in meter classification. Furthermore, the choice of input features plays a crucial role in the success of these models. While traditional signal processing techniques, like MFCCs, have been widely used, deep learning models often rely on log-mel spectrograms due to their ability to capture relevant audio features. It is important to note that these studies have focused on different musical genres, languages, and datasets, showcasing the versatility and adaptability of these methods across various musical contexts.

Overall, the findings suggest that deep learning techniques, when combined with appropriate feature extraction and modeling strategies, offer great potential for advancing the field of music analysis. These methods have demonstrated improved accuracy and efficiency in tasks related to beat tracking, meter identification, and tempo estimation, contributing to a deeper understanding of music structure and facilitating applications in fields such as music recommendation, genre classification, and music production.

Chapter 4

Machine Learning Methods

4.1 Introduction

Machine learning methods have revolutionized various fields [44, 81, 145] by offering sophisticated techniques [139, 141] for data analysis [108] and pattern recognition [16]. Among the diverse applications of machine learning, one particularly compelling area is music time signature detection. This task involves identifying the time signature of a piece of music, which is a crucial aspect of understanding its rhythmic structure. Time signatures, such as $\frac{4}{4}$, $\frac{3}{4}$, or $\frac{5}{4}$, dictate the underlying beat and are fundamental to how music is composed, performed, and interpreted. Machine learning models can analyze large datasets of musical pieces to detect patterns and regularities, thereby enabling accurate time signature identification. This process not only enhances music transcription and analysis but also opens new avenues in music generation and recommendation systems.

The application of machine learning to music time signature detection exemplifies the intersection of technology and art. By leveraging algorithms that can learn from data, researchers and developers can create systems that mimic human-like understanding of musical rhythm. This involves training models on annotated music datasets where the time signatures are known, allowing the algorithms to learn the intricate relationships between the audio features and the rhythmic patterns. The models considered in this study, including K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Naive Bayes, Random Forest, Convolutional Neural Networks (CNN), Convolutional Recurrent Neural Networks (CRNN), Residual Networks, and Residual Networks-LSTM, represent a range of approaches from traditional machine learning to advanced deep learning techniques.

4.2 Classic Machine Learning Models

Classic machine learning models, such as KNN, SVM, Naive Bayes, and Random Forest, rely on well-established statistical techniques to analyze and classify data. These models excel at making predictions based on structured data and are particularly effective when there is a clear relationship between input features and output labels.

4.2.1 Support Vector Machine

SVMs, as mentioned in this study, use a kernel approach to transform the input data into a higher-dimensional space [31]. This transformation allows for linear separation of the data using a hyperplane. The goal is to find the hyperplane that maximizes the margin between different classes. In this study, an radial basis function (RBF) kernel is specifically used with SVM to address non-linear classification problems. The RBF kernel allows SVM to capture complex relationships between data points by mapping them into a higher-dimensional space. By using the RBF kernel, the SVM can effectively classify data that is not linearly separable.

4.2.2 Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. It incorporates two key concepts: bootstrap aggregation (bagging) and random feature selection [22]. Bagging involves training each decision tree on a randomly selected subset of the training samples, allowing for diversity in the training process. Additionally, during the construction of each tree, only a random subset of features is considered for making decisions. This further promotes diversity among the trees in the ensemble. To make predictions, each tree in the Random Forest independently classifies the input data, and the class with the majority of votes from the individual trees is selected as the final predicted class. By combining the decisions of multiple trees, Random Forest can provide more robust and accurate predictions compared to a single decision tree.

4.2.3 KNN

The k-Nearest Neighbors (KNN) classifier is a simple machine learning technique used for classification tasks. It determines the label of a test instance based on the labels of its k nearest neighbors in the feature space. The concept behind KNN is that similar instances tend to belong to the same class. One advantage of KNN is its ability to capture both direct and indirect relationships in the data, as it does not assume any specific functional form. This makes KNN a non-linear classifier. The notion of "nearness" in KNN is defined by a distance metric, such as the Minkowski distance [117]. The Minkowski distance metric

generalizes different distance measures, including the Euclidean distance when $p = 2$. The Euclidean distance as shown in Equation 4.1 is commonly used in KNN, as it calculates the straight-line distance between two points in the feature space as show in the equation below. By considering the labels of the k nearest neighbors, KNN assigns the class label that occurs most frequently among those neighbors to the test instance. This majority voting process determines the final classification of the test instance.

$$\mathrm{EUC}(P, Q) = \sqrt{\sum_{i=1}^{d} (p_i - q_i)^2}. \tag{4.1}$$

4.2.4 Naive Bayes

The Naive Bayes method is a probabilistic supervised learning technique that classifies by computing the initial probabilities from the data in the dataset and categorizes the new data in accordance with this model. It is an algorithm that can be used to a variety of issues since it is compatible with all types of data and just needs basic statistical computations [80, 45].

The Naive Bayes method is a probabilistic supervised learning technique used for classification tasks. It applies Bayes' theorem to compute the posterior probability of a class given the feature values of an instance. The "Naive" assumption in Naive Bayes refers to the assumption of feature independence, meaning that each feature is assumed to contribute independently to the probability calculation. Naive Bayes classifiers are known for their simplicity and efficiency. They are suitable for a variety of problem domains and can handle different types of data, including categorical and numerical data. The algorithm requires basic statistical computations, such as calculating probabilities based on the training data [80].

To classify a new instance, it computes the initial probabilities of each class based on the frequency of class labels in the training data. Then, it calculates the likelihood of the feature values given each class. The final classification is determined by selecting the class with the highest posterior probability[45]. Due to its simplicity and ability to handle various data types, Naive Bayes is often used as a baseline classifier for comparison with more complex models. While the assumption of feature independence may not hold in all cases, Naive Bayes can still provide reasonable classification results, especially when the independence assumption is approximately met or when there is a large amount of training data.

4.3 Deep Learning Methods

Deep learning models, including CNN, CRNN, RESNET, and RESNET-LSTM, leverage multi-layered neural networks to automatically learn complex patterns in large data-

sets. These models are particularly powerful for tasks involving unstructured data, such as image and audio analysis, where they can extract and represent intricate features with high accuracy. CNNs, characterized by their ability to capture spatial hierarchies in data through convolutional layers, have been instrumental in advancing image understanding tasks. CRNNs extend the capabilities of CNNs by incorporating recurrent layers, enabling them to model sequential data such as audio spectrograms. ResNets introduced skip connections to alleviate the vanishing gradient problem, enabling the training of much deeper neural networks and achieving state-of-the-art performance in image classification tasks. We leverage ResNets combined with LSTM networks to handle sequential data with long-term dependencies, such as audio signals or time series data. In this section, we discuss the theoretical underpinnings of each deep learning method, discuss their architectural design principles, and explore their practical applications in real-world scenarios. By examining and comparing the performance of CNNs, CRNNs, ResNets, and ResNet-LSTM hybrids across different tasks.

For all the models, certain network layers are common such as the convolutional, pooling, activation function, LSTM and the softmax layers

Convolutional Layer

The foundation of a CNN is a convolution layer. It includes a collection of filters that can be learned (also known as kernels). The filter's spatial range is restricted by its physical dimensions. In this situation, a convolution operation is essentially a sum of the filtered features and an element-wise multiplication. By standard, three to four convolutional layers can be used consisting of an even number of filters. The filter size and number are design decisions. A stack of filtered features is produced after the image/raw input data has been convoluted with a number of filters or kernels. This will serve as the input for the following layer. The mathematical equation for the convolution operation can be represented as follows:

$$\text{Convolution}(x, w)_{i,j} = \sum_{m} \sum_{n} x_{i+m,j+n} \cdot w_{m,n} \tag{4.2}$$

where $Convolution(x, w)$ represents the result of the convolution operation, x represents the input matrix, w represents the filter or kernel, and i, j, m, and n are the indices of the elements in the matrices.

MaxPooling

MaxPooling is a down-sampling operation that reduces the spatial dimensions of the feature maps. It selects the maximum value within a window and discards the other values. This helps to capture the most important features while reducing the computational

complexity.

LSTM Layers

The LSTM (Long Short-Term Memory) layers are a type of Recurrent Neural Networks(RNN) that can effectively model sequential data by preserving information over long time steps. The LSTM units are responsible for learning and remembering relevant information from previous time steps and using it to make predictions.

Activation Function

A convolution layer's output activations are linear in nature. To accomplish non-linear transformation, these activations are routed via a non-linear function. Most often, the Rectified Linear Unit (ReLu) layer serves as an activation function for the output of earlier layers. If the input value is negative, this layer returns zero, and if it is positive, it returns the same value. The ReLU function is defined as follows:

$$\text{ReLU}(x) = \max(0, x) \tag{4.3}$$

Softmax Layer

An n-dimensional input vector of real values is transformed into probabilities for each class using a softmax layer given by the equation below. These probabilities are subsequently applied to identify the target class for a particular input. The estimated probabilities for the softmax function are in the range of 0 to 1, and they all add up to 1.

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{4.4}$$

4.3.1 Convolutional Neural Network

CNNs are designed to process and analyze visual data by automatically learning spatial hierarchies of features through a series of convolutional layers. Each convolutional layer applies a set of learnable filters to the input, capturing various patterns such as edges, textures, and shapes. These filtered features are then passed through activation functions, typically ReLU, to introduce non-linearity and enable the network to learn complex representations. MaxPooling layers are used to down-sample the spatial dimensions, reducing the computational load and emphasizing the most salient features. The final layers of the CNN often include fully connected layers and a softmax layer to output class probabilities. In the context of time signature detection, CNNs are particularly effective at identifying rhythmic patterns and structures in audio spectrograms, leveraging

their ability to capture local dependencies and hierarchical information within the input data. The CNN architecture used in this study is illustrated in Figure 4.1.

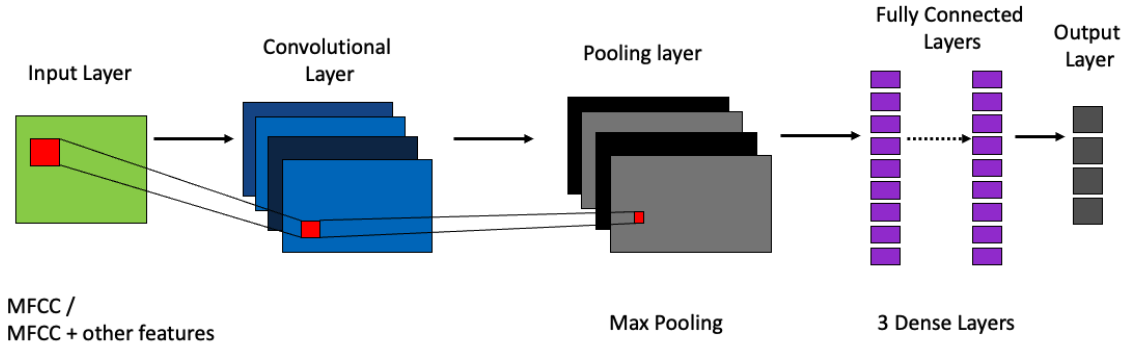

Figure 4.1: A CNN architecture for time signature

4.3.2 Convolutional Recurrent Neural Network (CRNN)

The CRNN architecture combines the strengths of both Convolutional Neural Networks and Recurrent Neural Networks. It leverages the CNN's ability to extract local features and the RNN's capability to capture temporal dependencies. Let's assume we have an input MFCC feature sequence of length T, denoted as $X = (x_1, x_2, ..., x_T)$, where each x_t represents a single MFCC feature vector, This formulation allows us to process sequential data using the defined sequence of feature vectors X.

Model Architecture

The CRNN as shown in the 4.2 contains convolutional layers which have been explained earlier in the CNN architecture. Therefore, the most important layer here is the recurrent layer. To capture temporal dependencies in the MFCC sequence, recurrent layers are employed. Gated Recurrent Units (GRUs) or Long Short-Term Memory (LSTM) units are commonly used recurrent units. In this case, the LSTM is preferred. The output of the recurrent layer at each time step is computed based on the current input and the previous hidden state:

$$\text{Output}_t = \text{RecurrentUnit}(\text{Pooling Layer Output}_t, \text{Recurrent Layer Output}_{t-1}) \quad (4.5)$$

A better explanation can be seen in Figure 4.3. The recurrent state in a recurrent neural network (RNN) is updated at each time step and carries information from previous time steps to the current time step. In the case of the convolutional recurrent neural network

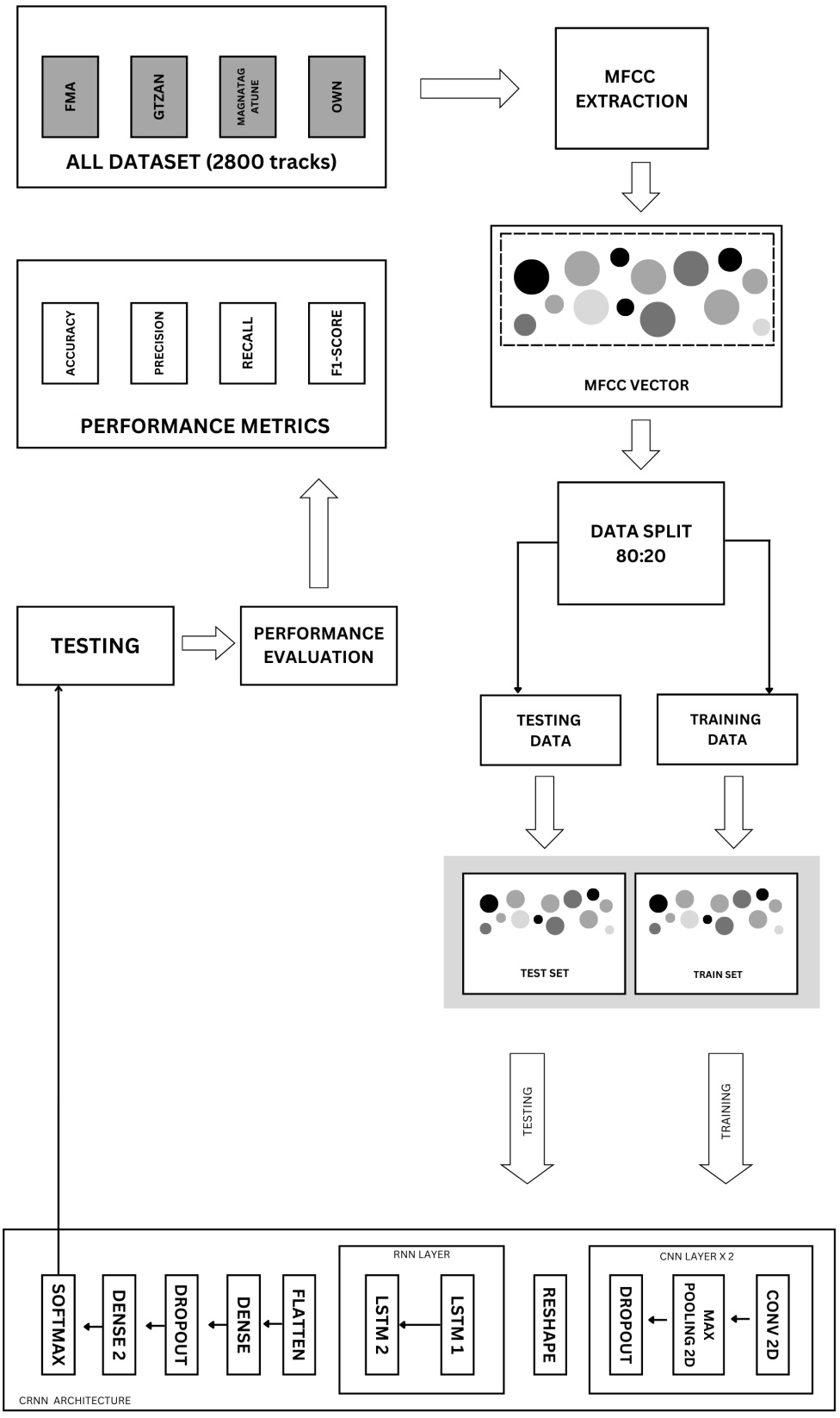

Figure 4.2: Model diagram of the CRNN architecture

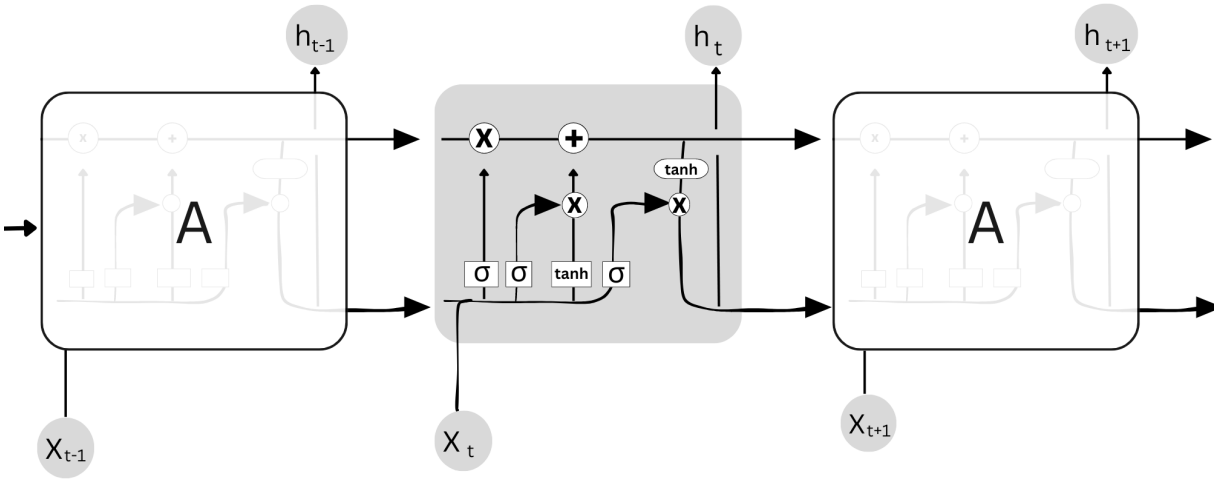

Figure 4.3: Model diagram of the recurrent state

(CRNN), the recurrent state is updated using the tanh and sigmoid activation functions as seen in Equation 4.6

$$h_t = \tanh(W_h \cdot h_{t-1} + W_x \cdot x_t + b_h) \tag{4.6}$$

In this equation, W_h is the weight matrix for the recurrent state, W_x is the weight matrix for the current input x_t and b_h is the bias term. The dot (.) represents the matrix multiplication operation. The `tanh` function is applied element-wise to the sum of the weighted recurrent state from the previous time step and the weighted current input. It squashes the values between -1 and 1, allowing the recurrent state to capture both positive and negative information.

Additionally, the CRNN uses a sigmoid function to control the flow of information from the current input to the recurrent state. This is done through a gating mechanism called the "update gate" or "input gate". The update gate determines how much information from the current input should be incorporated into the recurrent state. The update equation for the update gate can be written as follows:

$$\sigma_u = \text{sigmoid}(W_u \cdot x_t + b_u) \tag{4.7}$$

where σ_u presents the update gate, W_u is the weight matrix for the update gate, and b_u is the bias term. The sigmoid function squashes the values between 0 and 1, representing the amount of information to be updated. The update gate is then used to combine the previous recurrent state with the updated information, resulting in the final recurrent state:

$$h_t = \sigma_u \odot h_{t-1} + (1 - \sigma_u) \odot \tanh(W_h \cdot h_{t-1} + W_x \cdot x_t + b_h) \tag{4.8}$$

The update gate controls the contribution of the previous recurrent state and the updated information, allowing the model to selectively update the recurrent state based on the current input.

LSTM Layers

The LSTM layers are added after the convolutional layers in the CRNN model, allowing the model to capture both local and temporal features in the input data.

4.3.3 Residual Network (ResNet)

Residual Network, also known as ResNet, is a deep learning architecture that revolutionized the field of image classification. It was first introduced by researchers Kaiming He et al. from Microsoft Research in their seminal paper titled "Deep Residual Learning for Image Recognition" [70]. The motivation behind ResNet's development was to address the problem of degradation in very deep neural networks. Conventionally, it was believed that increasing the depth of a neural network would lead to better performance. However, the researchers discovered that as the network gets deeper, the accuracy starts saturating and then degrades rapidly. This degradation phenomenon occurs due to the difficulty of training deeper networks, where the optimization process becomes more challenging.

To overcome this degradation problem, they proposed a novel architecture based on the concept of residual learning. The core idea behind ResNet is the introduction of residual blocks, which allow for the direct propagation of information from earlier layers to later layers. This concept of "shortcut connections" enables the network to learn residual mappings, capturing the difference between the input and the desired output. The residual block in ResNet consists of a skip connection, also known as an identity mapping, that bypasses one or more convolutional layers. By adding the input to the output of the convolutional layers, the network can learn to focus on the residual information, i.e., the difference between the input and output, rather than attempting to directly learn the desired mapping from scratch. This alleviates the optimization difficulties associated with training deep networks by enabling faster and more efficient gradient propagation.

ResNet introduced the concept of "skip connections" and "identity mappings" as a breakthrough in deep learning. These skip connections not only mitigate the vanishing gradient problem but also help in preserving information throughout the network. The skip connections allow gradients to flow easily back through the network, facilitating the training of very deep models as shown in Figure 4.4. Rather than relying on the assumption that consecutive stacked layers will directly learn the desired underlying mapping, the ResNet architecture introduces residual mappings. This is achieved by utilizing shortcut connections in feed-forward neural networks. The formulation of $F(x) + x$ allows the network to learn residual mappings by adding the original input (x) to the transformed

output ($F(x)$). This approach enables the network to focus on learning the residual mapping rather than trying to learn the entire mapping from scratch.

Its effectiveness in image classification has been demonstrated through its outstanding performance on benchmark datasets such as ImageNet[40], where it achieved state-of-the-art accuracy. By leveraging its ability to train very deep networks effectively, ResNet has significantly advanced the field of image classification. It has provided a foundation for subsequent developments in deep learning, particularly in computer vision tasks, and has become a standard architecture used as a benchmark for evaluating new methods. Its usefulness extends beyond image classification. The residual learning concept has been adopted and adapted in various other domains, including object detection, semantic segmentation, and image generation. The key benefit of ResNet is its ability to capture and represent complex features through its deep architecture, allowing for more accurate and robust predictions.

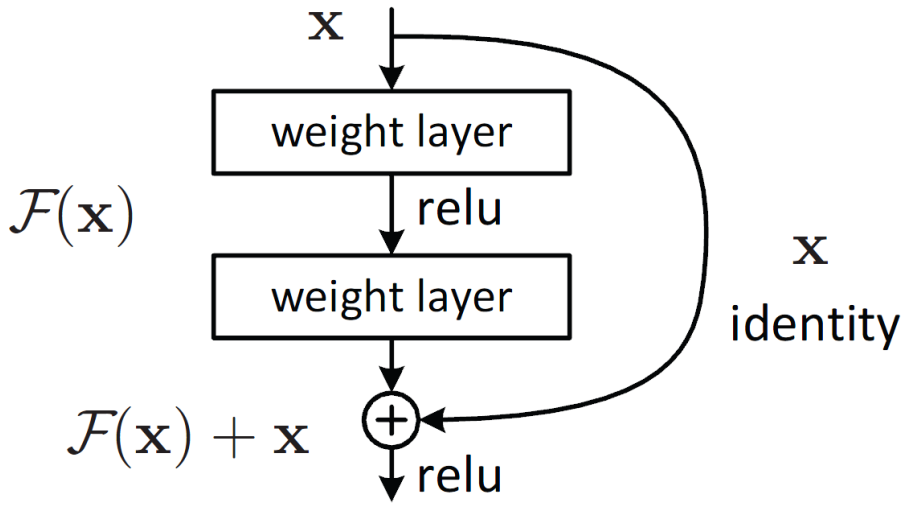

Figure 4.4: Skip connection in the ResNet architecture [70]

In terms of implementation, ResNet can be realized through various network architectures, including ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152, where the numbers represent the depth of the network in terms of layers. Deeper ResNet variants are achieved by stacking more residual blocks. Table 4.1 has the full summary.

4.3.4 Residual Network-LSTM

The RESNET-LSTM architecture builds upon the ResNet model, incorporating LSTM layers to capture temporal dependencies in the data. This modified architecture removes the global average pooling layer and the last softmax layer with 1000 output size, making it suitable for tasks that require sequence modeling and prediction as shown in Figure 4.5

In the original ResNet architecture, the global average pooling layer is typically used to aggregate spatial information from the convolutional layers before the final classification

Layer name	Output Size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112x112	7x7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1x1	average pool, 1000-d fc, softmax				
FLOPS		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 4.1: The architectures for residual networks, (building blocks indicated in brackets), with the numbers of blocks stacked, and down-sampling is achieved by conv3, conv4, and conv5, with a stride of 2

layer. However, in the RESNET-LSTM architecture, this pooling layer is removed and allow for sequential modeling. By retaining the spatial information, the model can capture temporal dependencies and patterns within the data. To introduce temporal modeling, two LSTM layers are added after the last convolutional layer, conv5_x. LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) architecture that is well-suited for sequential data analysis. The LSTM layers enable the model to learn and capture long-range dependencies in the input sequence. After the LSTM layers, the original global average pooling layer is reinstated. The global average pooling layer reduces the spatial dimensions of the features obtained from the LSTM layers to a fixed-size representation. This pooling operation helps to summarize the temporal information learned by the LSTM layers.

Finally, the linear layers similar to the ones used in the original ResNet model are re-used in the RESNET-LSTM architecture. These linear layers serve as the classifier, transforming the fixed-size representation obtained from the global average pooling layer into the desired output size, which corresponds to the number of target classes. Further details of the architecture can be found below;

4.4 Summary

To sum up, in the chapter, various techniques for music time signature detection were explored, utilizing both classic machine learning models and deep learning methods. The study highlighted the application of these techniques in analyzing music datasets to identify the rhythmic structures defined by time signatures like $\frac{4}{4}$, $\frac{3}{4}$, and $\frac{5}{4}$. Classic machine learning models such as SVM, Random Forest, KNN, and Naive Bayes were discussed for their effectiveness in structured data classification tasks. SVMs, leveraging

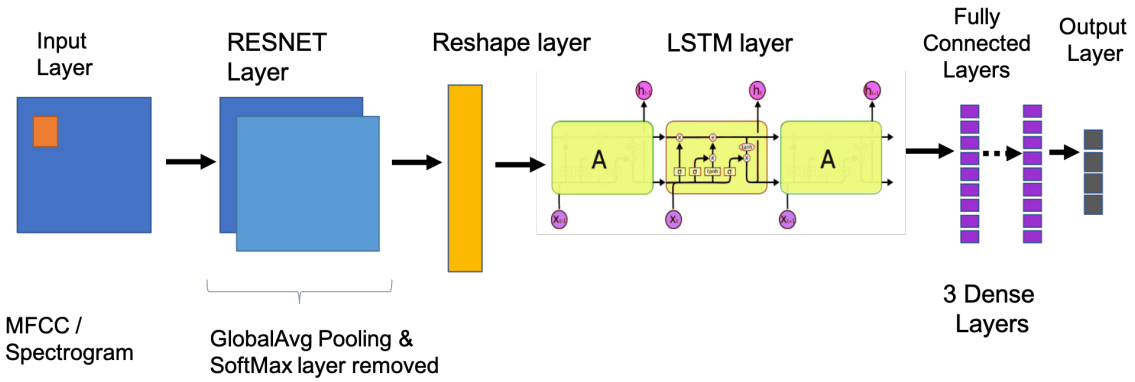

Figure 4.5: The ResNet-LSTM architecture

radial basis function kernels, excel in nonlinear classification, while Random Forests utilize ensemble learning to combine multiple decision trees for robust predictions. KNN, relying on distance metrics like the Euclidean distance, and Naive Bayes, assuming feature independence, serve as baseline classifiers for comparison. Deep learning models like CNN, CRNN, (RESNETs), and RESNET-LSTM were also highlighted. These models were considered to enhance music transcription and analysis by automatically detecting time signatures from audio data.

Chapter 5

Creation of the METER2800 Dataset

5.1 Introduction

Choosing the right dataset is crucial in classification, estimation, or detection projects. It's not always an easy decision as there are various datasets to choose from, each with its own strengths and limitations. The chosen dataset can greatly impact the outcome of the project. The availability of audio datasets plays a significant role in the development and evaluation of algorithms for time signature detection in Music Information Retrieval. These datasets enable researchers to train machine learning models effectively, leading to improved performance. However, creating such audio datasets is a challenging and time-consuming task. It involves various steps such as sorting, annotating, and processing audio files, requiring significant resources and expertise. Before the creation of the Meter2800, there was no dataset that was manually annotated for the purpose of time signature detection. To bridge this gap and facilitate further research, the Meter2800 dataset was developed which is a significant part of this PhD dissertation. We published this dataset to Harvard Dataverse [2] and also in the Data-In-Brief Journal [3]. Additionally, we extracted some low level acoustic features which are stored in separate csv files. More details can be found in the section 5.4.

Over the years, there has been a shift towards creating more comprehensive and diverse datasets, moving away from simpler ones. Researchers have put efforts into providing larger and more robust datasets to support their work. The importance of dataset selection cannot be underestimated, and it plays a significant role in the success of a project. As illustrated in Table 5.1, efforts to provide larger and more diversified datasets have replaced earlier attempts to collect robust and well-balanced datasets.

Dataset Name	Year Created	Number of Samples	Sample Type
RWC [60]	2002	365	Audio
CAL500 [150]	2008	502	MIDI
GZTAN [153]	2002	1,000	Audio
USPOP [149]	2002	8,752	MIDI
Swat10K [148]	2010	10,870	MIDI
MagnaTagATune [92]	2009	25,863	Audio
FMA [37]	2016	106,574	Audio
MusicCLEF [133]	2012	200,000	Audio
MSD [15]	2011	1,000,000	CSV

Table 5.1: Datasets and their statistics.

5.2 Deep Dive into Datasets Created Earlier

The RWC dataset, introduced by Goto et al. [60], played a significant role in the early stages of academic research on music datasets. It was one of the first datasets specifically compiled for academic purposes, following the trend of shared libraries in other scholarly fields. The dataset consists of six distinct collections, each focusing on a different genre or aspect of music. These collections include popular music, royalty-free music, classical music, jazz music, music genres, and musical instrument sounds. In total, the dataset comprises 365 musical pieces with accompanying audio signals, regular MIDI archives, and text files containing lyrics. One notable aspect of the RWC dataset is its inclusion of individual sounds at half-tone intervals, featuring a wide range of playing techniques, dynamics, instrument makers, and musicians. This comprehensive representation allowed researchers to explore various aspects of music analysis and classification. However, it is important to note that the RWC dataset has some limitations. One drawback is its relatively small size, which may restrict the ability to generalize results. Additionally, the dataset is considered unbalanced, meaning that it may not adequately represent the diversity and distribution of real-world music data. Despite these limitations, the RWC dataset served as a valuable starting point for researchers to test and analyze different methodologies and algorithms in the field of music classification and analysis.

In 2008, Ju-Chiang Wang et al. [150] developed the CAL500 dataset as an enhancement of the previous RWC datasets for music auto-tagging purposes. This dataset aimed to improve upon the limitations of the RWC dataset, featuring a larger collection of approximately 502 songs. However, it's worth noting that the audio files themselves are not included in the dataset, which could limit its usability for certain applications. One notable difference in the CAL500 dataset is that the tag labels are annotated at the segment level, rather than the track level. This means that instead of assigning tags to an entire song, tags are provided for specific segments within each song. While this granular annotation approach can provide more detailed information about the audio content, it's

important to consider the implications of segment-level annotations for certain analysis tasks. Despite the efforts to enhance the dataset, the CAL500 dataset still has limitations. The relatively small size of 502 songs may not be sufficient to achieve highly accurate results for music auto-tagging. Having a larger and more diverse dataset is often crucial for training robust machine learning models and achieving better performance in auto-tagging tasks. Researchers in the field continue to explore and develop new datasets that address these limitations and provide more comprehensive resources for music auto-tagging and related applications.

When discussing datasets in the field of music information retrieval, it's impossible not to mention the GTZAN dataset collected by G. Tzanetakis and P. Cook [153]. This dataset has become incredibly popular and influential within the research community. It consists of 1000 song excerpts, each lasting 30 seconds and sampled at a rate of 22050 Hz with 16-bit resolution. The excerpts were sourced from a variety of channels, including personal CDs, radio recordings, and microphone captures. The GTZAN dataset stands out due to its careful curation and organization. The songs are evenly distributed across ten distinct genres, namely Blues, Classical, Country, Disco, Hip Hop, Jazz, Metal, Pop, Reggae, and Rock. This balanced representation allows researchers to explore music genre classification tasks effectively. Consequently, since its release in 2002, the GTZAN dataset has been widely adopted and referenced in numerous studies on music genre analysis. The dataset's popularity and established reputation make it an authoritative reference point for comparing and validating research results. Many researchers have utilized the GTZAN dataset as a benchmark for evaluating the performance of various music genre classification algorithms mention in the previous chapter. Its extensive usage has created a common ground for comparing different methods and techniques. However, despite its widespread usage, the GTZAN dataset does have certain limitations. The most notable drawback is its relatively small size. With only 1000 song excerpts, some researchers argue that the dataset may not capture the full diversity and complexity of music in the real world. As a result, the ability to generalize results from models trained on the GTZAN dataset could be limited. Nevertheless, it continues to serve as an essential resource for music genre classification research, while efforts are made to develop larger and more diverse datasets to address these limitations.

Another notable dataset in the field of music information retrieval is USPOP, created by Mandel and Ellis [149]. USPOP focuses exclusively on popular artists and comprises a collection of over 8752 audio songs. However, it's important to note that the raw audio files are not provided as part of the dataset, limiting its potential applications. While USPOP offers a substantial number of songs for analysis, there are concerns regarding its skewed nature. Skewed datasets, where certain categories or classes are over-represented compared to others, can introduce biases and impact the performance of algorithms trained on them. Various studies [89, 136, 98] have highlighted the importance of addressing skewness in

datasets and its potential effects on the accuracy and reliability of the solutions they are applied to. The skewed distribution of popular artists in this dataset raises questions about how well it represents a larger music landscape. It may not accurately capture the diversity of musical genres and styles, limiting its applicability in tasks that require a comprehensive understanding of music beyond popular artists. Therefore, caution must be exercised when using it as a benchmark or reference for evaluating algorithms and methodologies in music classification and analysis.

The Beatles dataset, introduced by Chris Harte in 2010 [69], is a valuable resource for music analysis and research. It focuses specifically on the music of the iconic band, the Beatles, and consists of 180 meticulously annotated songs by musicologist Alan W. Pollack. The dataset provides detailed insights into the structural elements and characteristics of each song, with section-level annotations for an average of 10 distinct sections representing 5 different section types per recording. The Beatles dataset has had a significant impact on the field of music research and analysis. It played a pivotal role in the creation of the Million Song Dataset, a comprehensive collection of audio and metadata from diverse musical sources. By incorporating the well-annotated Beatles songs into this larger dataset, researchers gained access to a rich resource for studying not only the Beatles' music but also broader patterns and trends in music.

The SWAT10K dataset, also known as the Semantic Web Audio Tagging 10K dataset, is a notable collection of audio songs. It consists of 10,870 songs that were obtained from the Echo Nest API in collaboration with Pandora, a popular music streaming service. The dataset features weakly-labeled annotations using a tag vocabulary of 475 acoustic tags and 153 genre tags. However, the actual audio files are not included in the dataset. The Echo Nest, based in Somerville, MA, is a music intelligence and data platform that was acquired by Spotify in 2014. It originated as a spin-off from the MIT Media Lab, focusing on the analysis of auditory and textual content in recorded music. The Echo Nest's APIs offer various functionalities, including music recognition, recommendation, playlist creation, audio fingerprinting, and analysis, catering to both consumers and developers. Pandora, on the other hand, is a subscription-based music streaming service known for its suggestions based on the "Music Genome Project," a method of categorizing individual songs based on their musical characteristics.

Similarly, MagnaTagATune [92], another dataset based on the Echo Nest API, provides 25,863 audio files in a CSV format. It offers a rich collection of songs that can be used for various research purposes. The MusicCLEF dataset [133] which includes 200,000 audio songs is also worth mentioning because it was specifically designed for research purposes. It provides a substantial resource for studying and exploring popular music.

The Free Music Archive (FMA) dataset, introduced by Defferrard et al. [37], is a vast collection of music tracks with accompanying genre labels. With over 100,000 tracks, the dataset provides a diverse range of musical styles and genres for analysis. Researchers can

access different variations of the dataset, such as the *small* version consisting of 8,000 30-second samples, or the *full* version containing all 106,574 songs in their complete form. One notable advantage of the FMA dataset is its size, which allows for more comprehensive labeling and analysis. With a large number of tracks, the opportunity to explore a wide range of musical genres and study their characteristics is granted. Additionally, the availability of audio files for download enables direct extraction of features from the audio itself, enhancing the accuracy and relevance of the analysis.

The Million Song Dataset (MSD) [15] is an extensive collection of audio features and metadata for one million contemporary songs. As the name suggests, the dataset provides a wealth of information on a wide range of tracks. It encompasses various details such as the release year, artist information, terms associated with the artist, related artists, danceability, energy, song length, beats, tempo, loudness, and time signature. While the dataset includes comprehensive metadata and derived features for all one million songs, it's important to note that full audio files with proper tag annotations are only available for approximately 240,000 previews of 30 seconds [123]. However, the metadata and derived features still offer valuable insights into the songs' characteristics and allow for analysis at a broader level.

After considering the advantages and limitations of various datasets, two datasets stand out as particularly useful for the purpose of time signature extraction: the FMA and the MSD, both of which are derived from the Echo Nest API. Both the FMA and the MSD offer a substantial collection of songs with associated metadata, including information relevant to time signature extraction. However, it's worth noting that the metadata in the MSD has been pre-processed, which means that the specific details of how the processing was carried out are not readily available. Nevertheless, there is a confidence level associated with the extracted data, providing some indication of the reliability of the time signature information. The FMA dataset, on the other hand, offers a comprehensive set of audio features and metadata for a large number of tracks, including genre labels and other relevant information. The availability of audio files in this dataset allows for direct feature extraction from the audio, which can be advantageous for time signature analysis.

Despite their benefits, these datasets have significant shortcomings. The Million Song Dataset, for instance, provides estimated time signatures that were computed using algorithms and techniques applied to audio signals, rather than being directly annotated or labeled by experts. This distinction is crucial because computed time signatures may introduce errors or inaccuracies due to the complexity of the estimation process. Moreover, the MSD does not include the actual audio files used for time signature estimation, further limiting the dataset's usability and transparency as researchers cannot directly access and analyze the audio recordings themselves.

5.3 The Meter2800

To address the limitations already mentioned above, the Meter2800 dataset was created. The Meter2800 dataset was curated by annotating audio files from three well-known existing datasets: GTZAN, FMA-medium, and MagnaTagATune. Additionally, other audio files focusing on irregular time signatures were included. This dataset is specifically designed for time signature detection and is openly accessible to researchers at `https://bit.ly/meter2800`. By combining the three aforementioned datasets, we curated a comprehensive collection of audio recordings with accompanying annotations for time signature information. The Meter2800 dataset offers a standardized and diverse set of audio samples, covering a wide range of musical genres and styles. It aims to support the development of robust and versatile time signature detection algorithms by providing a rich and representative collection of annotated audio data. We are convinced that it will provide researchers with a valuable resource to train and evaluate their time signature detection algorithms and to validate the accuracy and reliability of their models. Before the creation of the this dataset, MIR researchers faced a lack of dedicated resources for their studies, making this dataset a significant contribution to the field.

5.3.1 Method of Creation

The creation process of the Meter2800 dataset involved several meticulous steps. Here is an overview of how the dataset was curated:

- Selection of Audio Files: A diverse set of audio files representing various musical genres and styles were chosen for inclusion in the dataset. Care was taken to ensure that the dataset covers a wide range of musical compositions.
- Expert Annotation: We carefully listened to each audio track, focusing on identifying the time signature, which indicates the meter or rhythmic structure of the music. Multiple listens were conducted to ensure accurate annotation.
- Time Signature Extraction: Once the time signature for each track was determined, we recorded this information along with other relevant metadata, such as the musical tempo. To facilitate the analysis, the tempo and extracted features were processed using the `librosa` [102] library, which is a popular tool for audio and music signal analysis in Python.
- Data Cleaning and Pre-processing: The annotated data underwent a thorough cleaning and pre-processing stage. This step involved removing any duplicate or erroneous entries to ensure data consistency and quality. The data was also converted into a standardized format, making it easily accessible and compatible for further analysis and modeling.

This dataset has been meticulously organized to facilitate easy access and utilization by researchers. Since the dataset is a combination of three well-known datasets, the audio files from each individual dataset are stored in separate folders within the Meter2800 dataset. Specifically, the audio files from the GTZAN dataset are grouped together in a folder named "GTZAN." Similarly, the audio files from the FMA dataset are stored in a folder named "FMA," and the audio files from the MagnaTagATune dataset can be found in the "MAG" folder. Additionally, any additional audio files obtained from personal sources are placed in the "OWN" folder. This folder structure allows researchers to easily locate and extract audio files from the specific datasets they are interested in.

To provide an overview of the dataset, Table 5.2 summarizes the number of files contained within each folder, indicating the distribution of data from each source. Furthermore, the annotated data in the Meter2800 dataset is divided into separate train and test sets. The annotations, along with the relevant metadata, are stored in CSV files. This separation into train and test sets enables researchers to perform accurate evaluations and assessments of their algorithms by training on a designated portion of the dataset and testing on another.

Data Source	Number of files Annotated	Train	Test
FMA	851	598	253
GTZAN	911	632	279
MAG	925	652	273
OWN	113	78	35

Table 5.2: Summary of data source and annotated files for meter2800 dataset

The dataset comprises 2800 annotated audio samples, each with a duration of 30 seconds. It is categorized into four meter classes. However, there is an uneven distribution of data among these classes as summarized in Table 5.3. Classes 3 and 4 contain 1200 audio samples each, while classes 5 and 7 have 200 audio samples each. This imbalance in distribution is a result of the contributing datasets, such as GTZAN, FMA, and MagnaTagATune, which predominantly feature popular music genres like hip-hop, rock, and pop [21]. These popular genres often exhibit simple beat counts, which are easier to estimate. Consequently, the majority of audio tracks in these datasets have meter numerator values of 2, 3, and 4. On the other hand, audio tracks with irregular meters are less common and pose a greater challenge for detection. This scarcity of irregular meter tracks contributes to the dataset's uneven distribution.

It is worth noting that for simple duple and quadruple meters (e.g., $\frac{2}{4}$ and $\frac{4}{4}$), the meter class is recorded as "four." Similarly, for simple triple meters (e.g., $\frac{3}{4}$), the class is recorded as "three." The same principle applies to irregular meters, with the classes recorded as "five" and "seven."

Class	Number of files Annotated
3	1200
4	1200
5	200
7	200

Table 5.3: Summary of annotated files by class for meter2800 dataset

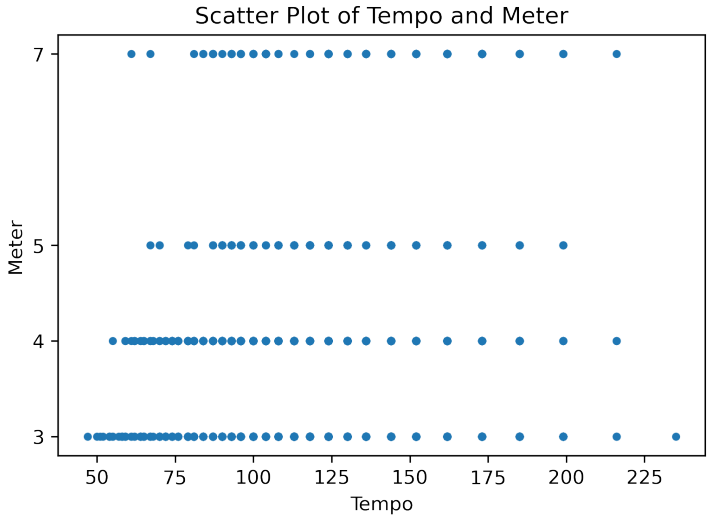

Figure 5.1: Scatter plot of tempo and meter

5.3.2 Data Analysis

The relationship between meter and tempo was investigated using the Meter2800 dataset and the distribution of both variables was examined. A scatter plot was generated to assess their correlation as shown in Figure 5.1. The results indicated that the tempo values followed a normal distribution, while the meter values exhibited an uneven distribution as depicted in Figure 5.2, with a majority of audio samples having a meter of 4 or 3.

The analysis revealed a weak positive correlation between tempo and meter i.e ($r = 0.105, p < 0.05$), with a correlation coefficient of 0.106. This indicates a slight tendency for higher tempos to be associated with higher meter values. However, the correlation was not particularly strong, suggesting that other factors may also influence the observed meter values in the dataset. The p-value, which was less than 0.05, indicates that the correlation is unlikely to be solely due to chance, but other contributing factors are likely present. Further examination of the data in Table 5.4 supports the association between track meter and tempo. On average, music with a higher meter tends to have a faster tempo, and as the meter increases, the diversity in tempo decreases. These findings suggest that tempo and meter are largely independent variables in the Meter2800 dataset, indicating that other factors beyond tempo contribute to the observed meter values.

Overall, the analysis indicates that while there is a weak positive correlation between

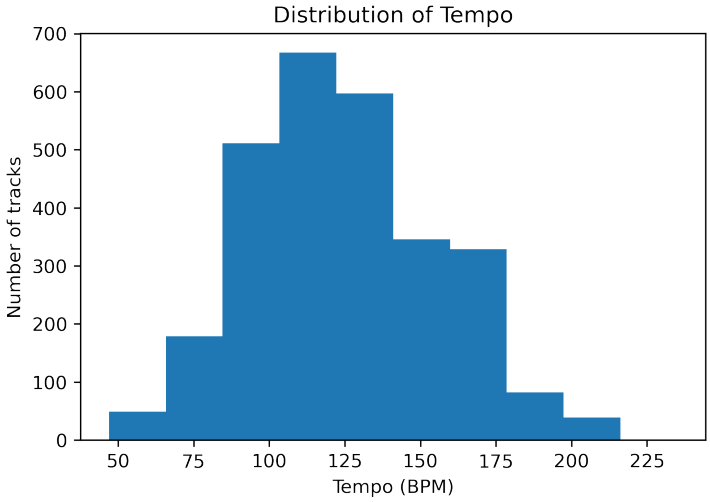

Figure 5.2: Distribution of tempo in the dataset

tempo and meter in the dataset, the relationship is not strong, and other factors play a significant role in determining the meter of the audio samples.

Meter	Mean	Standard Deviation
3	123.53	32.08
4	120.32	27.59
5	127.50	28.46
7	138.04	30.98

Table 5.4: Average and standard deviation of tempo by meter

5.4 Features Extraction

In the domain of machine learning, converting raw data into a usable format for models is essential. One of the key steps in this process is feature extraction. It involves transforming raw data into a set of meaningful features that can be effectively used by machine learning algorithms. This step is crucial because the quality of the features directly affects the performance of machine learning models. Proper feature extraction enhances model performance by providing relevant information and reducing data dimensionality, which in turn improves the accuracy and efficiency of the learning process.

For the Meter2800 dataset, we extracted various low-level audio features, including Chroma Short Time Fast Fourier Transform (STFT) [46], Root Mean Square (RMS), MFCCs [73], Spectral Centroid [94], Spectral Bandwidth [104], Zero Crossing Rate [62], and Spectral Rolloff [164].

Chroma Short Time Fast Fourier Transform (STFT)

This vector represents the overall energy of the signal across all 12 pitch classes. C, C#, D, D#, E, F, F#, G, G#, A, A#, and B. Then, a representative mean and standard deviation are calculated from the chroma vectors aggregated across the frames [46].

Root mean square (RMS)

The signal's energy is calculated as

$$\sum_{n=1}^{N} |x(n)|^2 \tag{5.1}$$

and the root mean square is then calculated as

$$RMS = \sqrt{\frac{1}{N} \sum_{n=1}^{N} |x(n)|^2} \tag{5.2}$$

where x represents the value of the audio signal at a specific time instant. In the formula, x(n) represents the value of the signal at the n^{th} sample. n represents the sample index, which is a discrete-time index used to denote the order of each sample in the audio signal ranging from 1 to N, where N is the total number of samples in the signal.

Mel-Frequency Cepstrum Coefficients (MFCCs)

The MFCCs are a group of decorrelated discrete cosine transform (DCT) parameters [73]. They are produced by a transformation of the logarithmically compressed filter-output energies from a perceptually spaced triangular filter bank that processes the discrete Fourier transform audio signal through which the coefficients are obtained as shown in Equation. (5.3).

$$c_n = \sum_{m=0}^{M-1} \log_{10}(s(m)). \cos\left(\frac{\pi n(m-0.5)}{M}\right) \tag{5.3}$$

where c_n are the cepstral coefficients, $n = 0, 1, 2, ..., C-1$ and C is the number of MFCCs. In this case, 13 was used. Further explanation of this extraction process can be found in chapter 6 as it is used in the MFCCSM.

Spectral Centroid

This is the frequency that most of the energy is focused on for each frame [94]. The magnitude-weighted frequency is determined as follows:

$$fc = \frac{\sum_{k=1}^{K} f_k m_k}{\sum_{k=1}^{K} m_k} \tag{5.4}$$

where K is the total number of frequency bins in the spectrogram of the signal, f_k is the frequency of the k^{th} bin, and m_k is the magnitude of the signal in the k^{th} bin. The formula represents the center of gravity of the spectral content of an audio signal.

Spectral Bandwidth

The p^{th} order moment about the spectral centroid corresponds to the p^{th} order spectral band-width and is calculated as:

$$[\sum_{k}(S(k)f(k) - f(c))^p]^{\frac{1}{p}} \tag{5.5}$$

$p = 2$ for example is an equivalent weighted standard deviation.

Zero crossing rate

The signal's change in sign from positive to negative is referred to as a "zero crossing point" [62]. The number of zero-crossings in each frame of the 10-second signal is calculated as:

$$Z_t = \frac{1}{2}\sum_{n=1}^{N} |sign(x[n]) - sign(x[n-1])| \tag{5.6}$$

where for positive arguments, the sign function is 1, 0 for negative arguments and $x[n]$ is the time domain signal for frame t.

Spectral Rolloff

This characteristic relates to the frequency value below which 85% of the spectrum's total energy is present [164]. However, the user has the privilege to determine this threshold. It is calculated as:

$$\sum_{n=1}^{R_t} M_t[n] = 0.85 * \sum_{n=1}^{N} M_t[n]. \tag{5.7}$$

where $M_t[n]$ is the magnitude of the Fourier transform at frame t and frequency bin n. The choice of these extracted features is based on temporal characteristics of music sound. According to Rocamora [129], MFCCs and their derivatives are the most appropriate features.

Generally, each audio sample in the dataset was divided into 10-second segments. MFCC features were extracted from each segment and saved in a JSON format to aid faster computation for subsequent processing, creating separate training and test sets.

These MFCC features are then used as input to the models for training, evaluation and testing.

All the extracted features are stored in a csv file named `multiple-features`. For each feature, including the MFCC with 13 coefficients, the minimum, maximum, mean, and standard deviation values were calculated. However, it was observed that the maximum value of the chroma STFT feature was consistently equal to 1.0 for all the audio tracks. Based on this observation, it is suggested that researchers can remove the column corresponding to the chroma STFT feature as a pre-processing step. Since the maximum value remains constant for all the audio tracks, this feature does not provide any additional information that would contribute to the training of the model. By removing this column, researchers can potentially streamline their data pre-processing pipeline and focus on the remaining features that are more informative for the task at hand.

Spectrogram Extraction

To extract spectrogram images from audio signals, we utilize the Short-Time Fourier Transform (STFT) method provided by the Librosa library. The spectrogram represents the magnitude of the Fourier Transform of short overlapping segments of the audio signal over time.

The process involves the following steps:

1. **Segmentation of Audio Signal**: Divide the audio signal into short segments of equal length to process iteratively.

   The number of samples per segment is calculated based on the total samples per track and the desired number of segments:

   $$\text{num\_samples\_per\_segment} = \frac{\text{samples\_per\_track}}{\text{num\_segments}}$$

2. **STFT Computation**: Compute the Short-Time Fourier Transform (STFT) of each audio segment using Librosa.

   For a segment of audio signal $x[n]$ with N samples, the STFT $X[m, \omega]$ is computed as:

   $$X[m, \omega] = \sum_{n=0}^{N-1} x[n] \cdot w[n - m] \cdot e^{-j\omega n}$$

   where $w[n]$ is the window function (Hamming window in this case) and ω is the frequency.

3. **Spectrogram Visualization**: Display the spectrogram using Librosa's amplitude-to-decibel conversion to enhance visualization.

The magnitude spectrogram $S[f,t]$ is converted to decibels using:

$$S_{\text{dB}}[f,t] = 10 \cdot \log_{10}\left(\frac{|S[f,t]|}{\max(|S[f,:]|)}\right)$$

4. **Save Spectrogram Images**: Save the generated spectrogram as an image file.

The entire dataset (which includes spectrogram images, mfcc arrays and other features) was split into training, validation, and testing sets with a ratio of 60:15:25, respectively, to facilitate model training, evaluation, and performance assessment.

5.5 Summary

This chapter emphasizes the critical role of dataset selection in music classification, estimation, or detection projects, particularly in the context of time signature detection within Music Information Retrieval. The historical challenges of creating comprehensive audio datasets were discussed and it highlights the lack of dedicated datasets for time signature detection before the development of Meter2800. Meter2800 was specifically created to bridge this gap, providing a valuable resource for researchers by combining and annotating audio files from existing datasets. This dataset, a significant part of this PhD dissertation, addresses the limitations of previous datasets and supports further research in the field. It is meticulously organized and annotated, providing a diverse range of audio samples categorized into four meter classes. Despite an uneven distribution of data across these classes, it offers a standardized resource for time signature detection research. The dataset's creation process, including audio file selection, expert annotation, and data cleaning, is detailed to ensure transparency and ways to reproduce it.

Finally, the relationship between meter and tempo in the Meter2800 dataset was analyzed, finding a weak positive correlation. This analysis suggests that while tempo and meter are related, other factors also influence meter values. The paper concludes with a discussion on feature extraction, highlighting the importance of transforming raw data into meaningful features for machine learning models. Features such as Chroma STFT, RMS, MFCCs, and others are extracted to enhance the accuracy and efficiency of time signature detection algorithms.

In summary, this chapter provides a comprehensive overview of the challenges and advancements in music dataset creation, emphasizing the significance of the Meter2800 dataset for MIR research.

Chapter 6

Mel-Frequency Cepstral Coefficient Similarity Matrix (MFCCSM)

6.1 Introduction

For time signature detection, various digital signal processing methods have been developed to improve accuracy and reliability. Among these, the Audio Similarity Matrix (ASM) and the Beat Similarity Matrix (BSM) which was discussed in chapter 3 have demonstrated promising results by analyzing temporal and rhythmic patterns in audio signals. However, these models still face limitations in capturing the nuanced timbral and accentual information critical for precise meter detection. To address these shortcomings, a new model called MFCCSM was developed. This model leverages Mel-frequency cepstral coefficients (MFCCs), which are widely recognized for their ability to extract detailed acoustic features from audio signals. They've also been applicable for various domains such as speech recognition, speaker recognition, and music genre classification. The result is a model that not only retains the strengths of its predecessors but also achieves slightly better accuracy in time signature detection.

6.2 MFCC Basics

Human perception does not linearly interpret the frequency content of sound. Therefore, the subjective pitch is assessed on a scale known as the 'Mel' scale, which is a non-linear scale. The conversion from frequency in Hz to the subjective pitch in Mels can be calculated using the formula:

$$f_{\text{mel}} = 2595 \log_{10}(1 + \frac{f}{700}) \tag{6.1}$$

MFCC coefficients are derived from a logarithmically compressed filter bank that processes the speech signal using a perceptually spaced triangular filter bank. These coeffi-

cients are obtained by applying a Discrete Cosine Transform (DCT) to the logarithmically compressed filter-output energies. The mathematical expression for computing the MFCC coefficients is given by Equation (6.2), where c_n represents the cepstral coefficients, n is the index of the coefficient, m is the index of the filter bank, $s(m)$ is the output energy of the m-th filter, and M is the total number of filters.

$$c_n = \sum_{m=0}^{M-1} \log_{10}(s(m)) \cdot \cos\left(\frac{\pi n(m-0.5)}{M}\right) \tag{6.2}$$

The MFCC coefficients capture timbral information and represent musical textures of limited duration [97]. In the context of meter detection, they can be used to extract accent and rhythmic properties from the audio signal. Typically, a suitable number of coefficients for audio signal processing is between 10 and 13. To determine the optimal number of coefficients for better accuracy, Bayesian Optimization was employed to search through the frequency bands space. These optimization techniques are discussed in the next section.

6.3 The Detection Process

The MFCCSM model aims to identify repeating bars in a song by comparing longer MFCC segments with shorter audio fragments (a fraction of a note). The underlying idea is that different parts of a song often have repeating patterns. This has been shown in the ASM model. The MFCCSM model builds upon the ASM approach by incorporating MFCCs to capture more detailed timbral and rhythmic properties of the audio signal. The detailed process of MFCCSM is explained below:

First, the audio signal is loaded, and the sample rate *sr* is determined using the `librosa` library mentioned in the dataset creation chapter 5. This step converts the audio file into a numerical format suitable for further analysis. The result of that is that *sr* is 22050. Next, the tempo and beat locations of the audio signal are estimated using `librosa`'s beat tracking algorithm. This step is crucial for segmenting the audio signal into meaningful temporal units (beats), which form the basis for further feature extraction. The duration of each beat (*b_Sec*) is then calculated both in seconds and in samples (*b_Sample*) using the formulas below

$$\text{b\_Sec} = \frac{60}{\text{tempo}} \tag{6.3}$$

$$\text{b\_Sample} = \text{int}(\text{b\_Sec} \times sr) \tag{6.4}$$

This beat duration will define the frame length for subsequent spectrogram and MFCC calculations. A spectrogram of the audio signal is created with a frame length correspond-

ing to the beat duration, and the overlap between frames is set to half the beat duration. The spectrogram provides a visual representation of the audio signal's frequency content over time. MFCCs are extracted from the audio signal with a hop length of half the beat duration, and typically 13 MFCC coefficients are used.

Additionally, the first and second-order deltas (derivatives) of the MFCCs are computed in Equation 6.5 and Equation 6.6. These deltas capture the dynamic changes in the MFCCs over time, providing further insight into the rhythmic and melodic progression of the audio signal. The MFCCs and their deltas are concatenated to form a new feature matrix in Equation 6.7, which is then transformed back into a mel spectrogram representation. This transformation ensures that the features are in a suitable format for similarity computation.

$$\Delta\text{MFCC}[n] = \text{MFCC}[n+1] - \text{MFCC}[n-1] \tag{6.5}$$

$$\Delta^2\text{MFCC}[n] = \text{MFCC}[n+2] - 2 \times \text{MFCC}[n] + \text{MFCC}[n-2] \tag{6.6}$$

$$\text{new\_mfcc} = \text{MFCC} \cdot \Delta\text{MFCC} \cdot \Delta^2\text{MFCC} \tag{6.7}$$

n represents the frame index in the sequence of MFCC coefficients. Each frame corresponds to a short segment of the audio signal over which the MFCCs are computed. The first-order delta $\Delta\text{MFCC}[n]$ is the difference between the MFCC values at frame $n+1$ and frame $n-1$, capturing the change in the MFCC values over time.

An empty similarity matrix (MFCCSM) is initialized, where each element represents the similarity between two time bins. This matrix will eventually capture the repeating patterns within the audio signal. The similarity between each pair of MFCC frames is computed using the Euclidean distance (or alternatively, cosine distance or Kullback-Leibler divergence). This step involves comparing the MFCC feature vectors at different time points to identify similarities and repetitions. The result is a matrix that also looks like that of the ASM in Figure 3.6. The average of the diagonals of the similarity matrix is then calculated to summarize the similarity across different segments of the audio signal. Diagonal values in the matrix represent the similarity of frames within the song at varying temporal offsets.

As a consequence, multiple diagonals emerge for analysis. These diagonals, extracted from one side of the symmetric MFCCSM, reveal insights into the similarities among musical mfcc frame segments separated by varying numbers of bars. For instance, one diagonal $D1$ may depict components spaced apart by one bar, while another diagonal $D2$ could illustrate components separated by two bars. Each diagonal reflects the similarity of frames within the song. Ideally, diagonal values would all be ones (1.0), indicating perfect similarity and resulting in a white coloration in the matrix representation. Subsequently,

the average diagonal is computed using Equation 6.8.

$$[htbp]d_i = mean(diag(\text{MFCC}SM_i))d = -d + \max(|d|) \tag{6.8}$$

The Euclidean distance measure was deemed inadequate for this model due to its dependence on vector magnitudes. To address this limitation, the cosine distance measure was adopted as an alternative. The cosine distance calculates the angle between two vectors rather than their magnitudes, defined mathematically as:

$$\text{cosine\_distance}(x, y) = 1 - \frac{x \cdot y}{\|x\| \cdot \|y\|} \tag{6.9}$$

Here, $\cdot$ denotes the dot product of vectors x and y, while $\|x\|$ and $\|y\|$ represent their Euclidean norms. Using cosine distance reduces reliance on vector magnitudes, focusing instead on their angular similarity. This approach enhances the robustness of comparing and evaluating musical components by considering their directional alignment. Consequently, the MFCCSM (MFCC Similarity Matrix) is defined as:

$$\text{MFCCSM}(a, b) = 1 - \frac{x \cdot y}{\sqrt{\sum_{k=1}^{s} X(a,k)^2} \cdot \sqrt{\sum_{k=1}^{s} X(b,k)^2}} \tag{6.10}$$

The Kullback-Leibler (K-L) divergence method [23, 18] is another widely used approach for computing vector similarities, expressed as:

$$\text{MFCCSM}(a, b) = \sum_{k=1}^{N/2} X(a,k) \log_e \left(\frac{X(a,k)}{X(b,k)} \right) \tag{6.11}$$

In this equation, $X(a, k)$ and $X(b, k)$ denote components of vectors a and b respectively, and N represents the total number of frames. The Kullback-Leibler divergence quantifies the difference between two distributions, providing another perspective on vector similarity.

Finally, the function representing these average diagonal values is adjusted by subtracting its values from the maximum absolute value in the function. This adjustment enhances the contrast between similar and dissimilar regions in the audio signal, making it easier to detect recurring patterns and time signatures.

6.4 Results and Discussion

In evaluating the effectiveness of the MFCCSM, we compared the result with that of the other classic digital signal processing techniques (ASM and BSM) for time signature detection using the Meter2800 dataset. The evaluation was conducted across two categories: binary classification with 2 classes and multi-class classification with 4 classes. The 2 classes aimed to detect $\frac{3}{4}$ and $\frac{4}{4}$meters, while the 4 classes included $\frac{3}{4}$, $\frac{4}{4}$ $\frac{5}{4}$, and $\frac{7}{4}$ meters.

6.4.1 Performance on 2 Classes

The assessment of these techniques in binary classification revealed that ASM achieved an accuracy of 53%, BSM exhibited a lower accuracy of 50%, and MFCCSM showed the highest accuracy at 55%, as summarized in Table 6.1.

Table 6.1: Classic signal processing models classification report for 2 classes

Model	Accuracy (%)
ASM	53.00
BSM	50.00
MFCCSM	**55.00**

6.4.2 Performance on 4 Classes

In the evaluation of these techniques on multi-class classification, ASM achieved an accuracy of 51%, BSM demonstrated a lower accuracy of 49%, and MFCCSM outperformed both with an accuracy of 53%, as shown in Table 6.2.

Table 6.2: Classic signal processing models classification report for 4 classes

Model	Accuracy (%)
ASM	51.00
BSM	49.00
MFCCSM	**53.00**

To further enhance the performance of the MFCCSM model, we conducted an analysis by assigning weights to the coefficients derived from a genetic algorithm (GA) optimization across all four classes. This detailed analysis is discussed further in Chapter 7, where we delve into the significance and contribution of MFCCs in improving time signature detection accuracy.

6.5 Summary

In this chapter, we discussed how MFCCSM was developed and its application for improving time signature detection in audio signals. By integrating MFCCs, known for their ability to capture detailed acoustic features, MFCCSM enhances upon traditional methods like ASM and BSM. The process involves converting audio signals into MFCC representations, computing first and second-order deltas to capture dynamic changes, and constructing a similarity matrix using metrics such as Euclidean, cosine distance, or Kullback-Leibler divergence. Evaluation on the Meter2800 dataset demonstrates that

MFCCSM achieves superior accuracy compared to ASM and BSM in both binary (detecting $\frac{3}{4}$ and $\frac{4}{4}$ meters) and multi-class (detecting $\frac{3}{4}$, $\frac{4}{4}$, $\frac{5}{4}$, and $\frac{7}{4}$ meters) classifications. This highlights MFCCSM's efficacy in capturing intricate musical nuances, thereby advancing the field of audio signal processing and music analysis.

Chapter 7

Optimization Techniques for Time Signature Detection Models

7.1 Introduction

Optimization involves refining algorithms and techniques to achieve the best performance under given constraints. In signal processing, optimization targets enhancing the accuracy, efficiency, and robustness of methods used for analyzing and interpreting signals. This includes optimizing filter design, noise reduction, and feature extraction to improve signal clarity and information retrieval. Generally, in this domain, optimization spans across various domains such as machine learning, where hyperparameter tuning and model selection are critical for maximizing predictive accuracy. Algorithms are often optimized for speed, memory usage, and computational efficiency, ensuring they can handle large datasets and complex computations effectively. Techniques like genetic algorithms, gradient descent, and Bayesian optimization are commonly employed to find optimal solutions in high-dimensional spaces. Overall, optimization is pivotal in advancing technology, enabling more precise, faster, and resource-efficient solutions across diverse applications in signal processing and computer science.

7.1.1 Optimization Techniques for Parameters and Features

Optimization generally helps to choose the best set of parameters while reducing the computational time to go through the search space. In this study, two optimization techniques were used to achieve a better result; the Bayesian optimization and the Genetic Algorithm.

The research conducted by Maider Zamalloa et al. [165] focused on speaker recognition and highlighted the redundancy and dependency of derivatives of MFCC as acoustic features. To address this issue, they employed a genetic algorithm (GA) to find the optimal set of weights for a 38-dimensional feature set. This set consisted of 12 MFCC features,

their first and second derivatives, energy, and its first derivative. The results demonstrated that weighting these acoustic characteristics led to a reduction in inaccuracy by an average of 15% to 25%.

In the domain of multi-modal biometrics, Nancy Bansal et al. [11] investigated the use of GA for feature reduction in a multi-modal biometrics system. They combined physiological biometrics such as face recognition with behavioral biometrics like speech recognition, where MFCC was employed for speech feature extraction. The study revealed that the proposed multi-modal biometric system provided enhanced security compared to existing unimodal biometric identification systems.

Another study by Prafulla Kalapatapu et al. [82] explored the optimization of acoustic feature extraction and selection using a genetic algorithm. The researchers classified Indian songs using four classifiers and observed that not all acoustic features were equally important. Therefore, GA was utilized to optimize the feature selection process, resulting in higher accuracy for the classifiers.

7.2 Optimization of the MFCCSM Model

In our paper published in the Proceedings of the Companion Conference on Genetic and Evolutionary Computation (GECCO) [5], we focused on optimizing the MFCCSM model to significantly enhance its ability to detect time signatures in audio signals. This optimization effort involved fine-tuning the parameters and computational processes to ensure that MFCCs and their derivatives accurately captured the nuanced timbral and rhythmic properties crucial for meter detection. Leveraging advanced optimization techniques, particularly genetic algorithms (GA) among others, we adjusted the model parameters to achieve superior accuracy and reliability. Despite already outperforming the ASM, our study aimed to push the boundaries further by optimizing the MFCCSM through GA, highlighting its effectiveness in refining complex machine learning models for audio analysis tasks.

7.2.1 Bayesian Optimization

Bayesian optimization is a method used to find the best values for parameters in a complex objective function that takes a long time to evaluate. It is particularly useful when the parameter space is continuous and has fewer than 20 dimensions, and when there is random noise in the function evaluation. In the context of improving accuracy, Bayesian optimization starts by initializing the parameters within specified bounds that define the search space. In our case, we have three parameters to optimize:

- Number of MFCCs (mf): This parameter determines the number of Mel-frequency cepstral coefficients (MFCCs) used in the analysis. We restrict the range to be

between 10 and 15, where mf is an integer.

- Number of Mels (nm): This parameter controls the number of mel frequency bins used in the analysis. We set the range to be between 128 and 1024, where nm is an integer.

- Number of Overlap: The amount of overlap is determined as a function of the beat duration. This parameter affects how the audio samples are segmented and processed. The equation is given as:

$$n_{overlap} = \left(\frac{60 \times t_p \times sr}{o}\right) \tag{7.1}$$

where o is the overlap value, sr is the sample rate of the signal, t_p is the tempo in bpm of the song and 60 is beat length in a second. $n_{overlap}$ is denoted as n where $2 \leq n \leq 4, n \in \mathbb{R}$. After optimization, the possible combination of these parameters that yielded the highest accuracy on the model were: mf=11, nm=1024 and n=2.

By exploring different combinations of these parameter values within the defined ranges, Bayesian optimization aims to find the optimal set of parameter values that maximize the accuracy of our objective function. Similar to the ASM, the MFCCSM follows the same approach by generating a similarity matrix from the Mel spectrogram using Euclidean distance, calculates the diagonal of the matrix and uses the diagonal to find the meter of a song as shown in Figure 7.1.

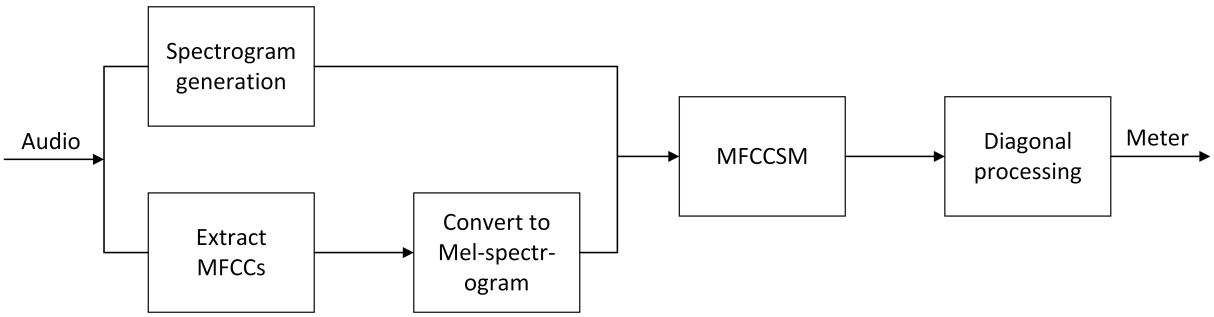

Figure 7.1: Mel-frequency cepstrum coefficients similarity matrix model

7.2.2 Genetic Algorithms Optimization

Genetic algorithms are a type of search technique inspired by population genetics, used to find optimal or near-optimal solutions to optimization and search problems or where traditional optimization techniques may struggle or fail to find satisfactory solutions. GAs employ a population-based approach, simulating the process of natural selection, crossover, and mutation to iteratively search for an optimal or near-optimal solution in a large solution space.

In a Genetic Algorithm, a population of potential solutions, known as individuals or chromosomes, is randomly generated or initialized. Each individual represents a candidate solution to the optimization problem. These individuals are evaluated and assigned a fitness value that quantifies their quality or suitability as a solution. The fitness value is typically based on an objective function that captures the optimization goal. Through a series of iterative generations, GAs apply genetic operators to the population to mimic the processes of selection, reproduction, and variation found in natural evolution. These operators include crossover and mutation. Crossover involves combining genetic information from two parent individuals to produce offspring with traits inherited from both parents. Mutation introduces random changes or alterations to the genetic material of an individual, promoting diversity in the population.

By applying these genetic operators, the population evolves over generations, with individuals that possess favorable traits or better fitness values being more likely to survive and reproduce. This natural selection process favors the propagation of individuals with desirable characteristics, gradually improving the overall fitness of the population. Over time, the GA converges towards a population of individuals that represent good or optimal solutions to the given optimization problem.

In a nut-shell, Genetic algorithms employ three main types of rules at each step: selection rules, crossover rules, and mutation rules.

- **Selection Rules**: These rules determine which individuals, called parents, from the current population will contribute to the next generation. The selection process is usually based on the fitness or objective function value of each individual.

- **Crossover Rules**: Crossover involves combining genetic information from two parents to create new individuals, known as children, for the next generation. The idea is to mimic the process of genetic recombination in natural evolution. There are different crossover strategies, such as one-point crossover or uniform crossover, that determine how genetic information is exchanged between parents.

- **Mutation Rules**: Mutation introduces random changes into individual parents to promote diversity in the population. This randomness helps prevent the algorithm from getting stuck in local optima. Each gene in an individual solution has a small probability of being randomly modified or replaced with a new value.

In the given model, a weight vector w consisting of 11 real numbers between 0 and 1 (i.e., $0 \leq w \leq 1$, where w is a real number) is initialized as the population. The genetic algorithm is executed with a population size of 100, a crossover probability of 0.5 (indicating a 50% chance of inheriting characteristics from existing solutions to offspring in new trials), uniform crossover strategy, a mutation probability of 0.1 (representing a 10% chance of each gene being replaced by a random value), and 100 iterations.

In each run of the genetic algorithm, the newly generated weights are multiplied by the MFCC (Mel-frequency cepstral coefficients) values, and the accuracy is calculated as the objective function. The goal is to find the combination of weights that maximizes the accuracy, indicating the best solution for the problem at hand. One of the key advantages of GAs is their ability to explore a large search space efficiently. The population-based approach allows GAs to simultaneously explore multiple regions of the solution space, increasing the chances of finding promising solutions [5]. The stochastic nature of the genetic operators introduces randomness and diversity, enabling GAs to avoid getting stuck in local optima and promoting the exploration of different solution possibilities [13].

It is worth noting that GAs typically aims to minimize an objective function. However, in the specific task at hand, our goal is to maximize the accuracy. To align with the GA convention, we can modify the objective function by subtracting the accuracy value from 100%. This way, higher accuracy values will correspond to lower objective function values, which the GA can then minimize. As with any optimization problem, the algorithm learns from the available data. In this case, the data was split into a training set and a testing set using a 70:30 ratio. The training set is used to train the model, while the testing set is used to evaluate the performance of the model on unseen data. This splitting allows us to assess how well the optimized GA-based model generalizes to new data and helps avoid over-fitting, where the model performs well on the training data but poorly on unseen data. By training on the training set and evaluating on the testing set, we can assess the effectiveness and generalization capability of the GA-based model in maximizing accuracy for the given task.

Algorithm 1 demonstrates the overall process of the genetic algorithm and the calculation of accuracy as the objective function.

Algorithm 1 Optimization Technique for MFCC Model

```
Load Track list
count ← 0
while Tracklist ≠ 0 do
    Run GA with weights W
    Generate 11 MFCC coefficients
    MFCC ← MFCC × W
    Take derivatives of new MFCC
    Convert MFCC to Mel Spectrogram
    Generate Similarity Matrix
    Calculate meter from diagonal
    if list_meter = calculated_meter then
        count ← count + 1
    else
        count ← 0
Calculate accuracy from count
```

To further enhance the MFCCSM model, we conducted an analysis of the coefficients by assigning them weights based on all four classes. These weights were derived from the optimal solutions generated by the genetic algorithm (GA). Through this analysis, we delved into the significance of the MFCCs. The outcomes of this analysis are depicted in Figure 7.2, where distinct colors are utilized to represent different coefficients.

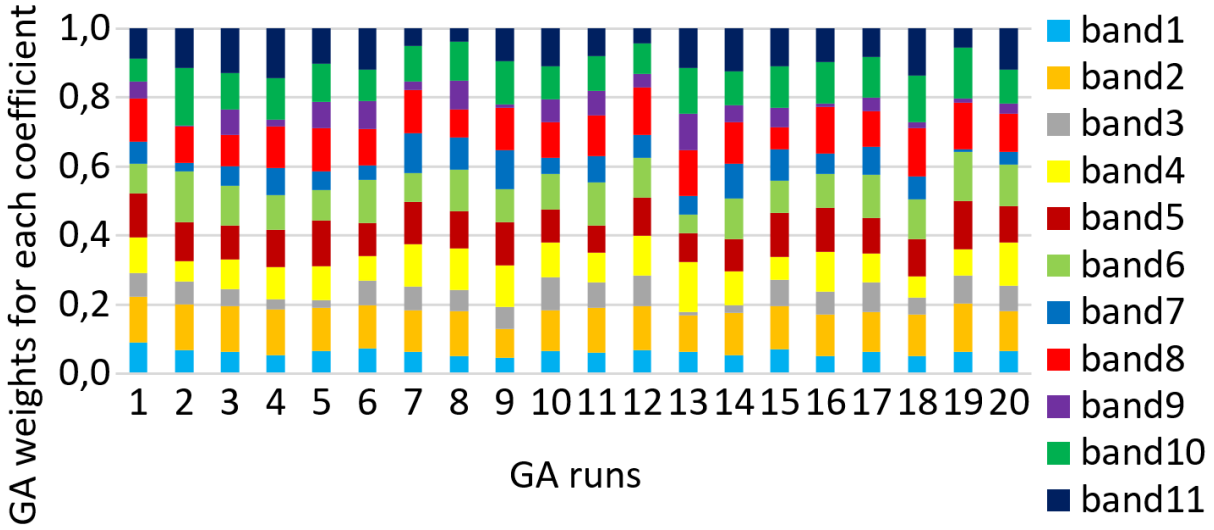

Figure 7.2: MFCC weights obtained for each performed GA run.

Upon analyzing the Figure 7.2, we observed that certain coefficients exhibited higher weights compared to others. Specifically, coefficient 2 (depicted in orange), coefficient 6 (depicted in light green), coefficient 8 (depicted in red), and coefficient 10 (depicted in green) showed higher values relative to the other coefficients. This suggests that these particular coefficients may hold more importance or contribute significantly to the classification task at hand. These findings provide valuable insights into the importance of specific MFCC coefficients for the classification of time signatures. The identified coefficients could potentially serve as key features for distinguishing between different time signature classes. This discovery opens up opportunities for further research in this area.

The results of our experiments are presented in Table 7.1. We observed a significant improvement in the accuracy of the base MFCCSM model that we focused on. The base model refers to the results without any optimization. We achieved more with the optimization techniques with an average accuracy of about 63.5%. The genetic algorithm proved to be efficient at improving the model on various levels. Additionally, Bayesian optimization played a role by providing the best set of parameters for the GA as initial parameters.

Model	BSM	ASM	MFCCSM	MFCCSM After GA			
				Avg.	Max.	Min.	Std.dev.
Accuracy [%]	50.0	53.0	55.0	63.5	65.0	60.0	1.25

Table 7.1: Accuracy of models before and after GA

7.3 Summary

In this chapter, we delved into the optimization of the MFCCSM (Mel-Frequency Cepstrum Coefficients Similarity Matrix) model to enhance its performance in accurately detecting time signatures in audio signals. Optimization techniques, such as Bayesian optimization and genetic algorithms, are employed to fine-tune the model parameters, including the number of MFCCs, mel frequency bins, and overlap. We used Bayesian optimization efficiently to explore the parameter space to find the optimal set of values, while GAs were employed to refine the weights assigned to MFCCs, revealing the most critical coefficients for accurate time signature classification. This chapter also highlighted previous research where GAs successfully optimized MFCC derivatives and acoustic feature sets, demonstrating their efficacy in reducing error rates and improving classification accuracy in various applications, such as speaker recognition and multi-modal biometrics. The analysis of GA-derived weights indicated that specific MFCC coefficients hold more importance, contributing significantly to the classification task. The results demonstrated a notable improvement in the MFCCSM model's accuracy, averaging around 63.5% after optimization, compared to the base model without optimization.

Chapter 8

Using Machine Learning for Time Signature Detection

8.1 Introduction

In this chapter, we present eight models for meter detection: four classic machine learning classifiers and four deep learning models and present results already published in the EURASIP Journal on Audio, Speech, and Music Processing [4]. The classic machine learning classifiers include SVM, Random Forest, Naive Bayes, and KNN. SVM is a popular algorithm that separates data using a hyperplane to maximize the separation between classes. Random Forest is an ensemble method that combines multiple decision trees to make a final classification. Naive Bayes is a probabilistic classifier that calculates the probability of a sample belonging to a class based on its features. KNN assigns a class label to a sample based on the classes of its nearest neighbors. These classifiers were chosen to evaluate their performance and compare the effectiveness of different machine learning approaches for meter detection.

In addition to the classic machine learning models, this chapter explores four deep learning algorithms: CNN, CRNN, ResNet18, and ResNet18-LSTM. Deep learning, a specialized branch of machine learning, focuses on training artificial neural networks with multiple layers to develop hierarchical data representations. These algorithms have demonstrated exceptional performance in various classification tasks and have been successfully applied to audio and music analysis. Their ability to automatically learn complex features and detect intricate patterns makes them particularly advantageous for tasks such as meter detection.

8.2 Feature Engineering and Model Evaluation

Using the Meter2800 dataset, MFCCs were extracted as array values to serve as input features for both classic machine learning and deep learning models during the training, evaluation, and testing processes. The dataset is divided into two subsets: one where all four classes are represented, and another where only two classes are included. This division allows us to evaluate model performance on both balanced and unbalanced data.

For classic machine learning models, a standardized process was followed across different algorithms using default hyper-parameters values. Initially, the audio dataset was pre-processed to extract MFCCs from each audio sample, resulting in a sequence of MFCC feature vectors that capture the spectral characteristics of the audio. The dataset was then split into training, validation, and testing sets as mentioned in Chapter 5. The training set was used to develop the models, the validation set to fine-tune hyper-parameters and monitor model performance, and the testing set to evaluate the final model's ability to generalize to unseen data. During training, the goal was to map the input MFCCs to output class labels, enabling accurate classification of new audio samples. Performance metrics such as accuracy, precision, recall, and F1-score were used to assess model effectiveness.

- **Support Vector Machine (SVM):**
  - *Kernel:* Radial Basis Function (RBF), which is effective in handling non-linear relationships within the data.
  - *Regularization Parameter (C):* Set to 1.0, balancing the trade-off between maximizing the margin and minimizing classification errors.
  - *Gamma:* Set to 'scale,' adjusting the kernel's influence according to the number of features, ensuring proper data fitting.
- **k-Nearest Neighbors (k-NN):**
  - *Number of Neighbors (k):* Set to 3, which is a typical choice in classification tasks, ensuring that the meter class label of a sample is determined by the majority class among its three nearest neighbors.
  - *Distance Metric:* Euclidean distance, used to measure the distance between data points in the feature space.
- **Naive Bayes:**
  - *Assumption:* The model operates under the assumption of feature independence, with a Gaussian distribution used for continuous variables. It calculates the posterior probability of each class by combining the prior probability and likelihood of observing the data given the meter class.

- **Random Forest:**
  - *Number of Trees (Estimators):* Set to 100, providing a robust ensemble by averaging the predictions from multiple decision trees, thereby reducing variance.
  - *Criterion for Splitting:* Gini impurity, which measures the quality of splits by evaluating how well they separate the classes at each node.
  - *Maximum Depth:* Unrestricted, allowing trees to grow until all leaves are pure or contain fewer than the minimum number of samples required to split a node.

These hyperparameter values were chosen to reflect commonly accepted defaults that balance computational efficiency with model accuracy. They provide a strong baseline for evaluating model performance across different datasets and tasks.

In addition to using MFCCs, we also conducted experiments with multiple audio features extracted using the Librosa library. This approach involved leveraging a diverse set of features alongside MFCCs, already discussed in Chapter 5. The rationale behind testing multiple features was to explore whether combining different audio representations could enhance classification performance compared to using MFCCs alone. By incorporating a broader range of features that capture various aspects of audio signals, such as pitch, timbre, and rhythm, we aimed to provide richer information to the machine learning models, potentially improving their ability to differentiate between different audio classes. Our experiments showed that models trained on this extended feature set outperformed those trained solely on MFCCs, as detailed in the results section.

In parallel, deep learning methods were also applied to the Meter2800 dataset, leveraging advanced architectures such as CNNs, CRNNs, ResNet18, and a hybrid ResNet18-LSTM model. These deep learning models, known for their exceptional performance in handling complex data like images, audio, and sequential data, were explored for their ability to automatically learn hierarchical features from the raw audio input. This allows them to capture intricate patterns and relationships within the data, making them particularly effective for tasks such as audio classification and meter detection.

8.2.1 Convolutional Neural Networks

In this study also, we utilized the CNN architecture for meter detection using two distinct input features: MFCC and spectrogram. For the MFCC, we inputted raw array vector values directly into the CNN, allowing the model to learn from the detailed frequency characteristics of the audio signals. For the spectrogram, we treated it as an image and fed it into the CNN to leverage its powerful image processing capabilities. These features were used separately, to evaluate their individual effectiveness in the CNN model for detecting time signatures in audio signals.

Data Pre-processing

Normalization and Scaling: Normalization and scaling of feature representations in the context of spectrogram data for time signature detection involve adjusting the input spectrogram images to a consistent scale or range of values before feeding them into the model Network. This process typically includes calculating the mean and standard deviation of the spectrogram data. For instance, consider a spectrogram dataset where each image has three channels representing different aspects of the audio signal. To normalize these spectrogram images, one might calculate the mean and standard deviation for each channel across the entire dataset. These values, such as a mean of [0.5, 0.5, 0.5] and a standard deviation of [0.5, 0.5, 0.5], are used to transform the input spectrogram tensors during preprocessing. The normalization step involves subtracting the calculated mean and dividing by the standard deviation for each channel of the spectrogram images. This operation ensures that the data is centered around zero with a unit variance, which can enhance the training stability and convergence of the CNN model when processing spectrogram inputs. The choice of specific normalization parameters, like the mean and standard deviation values, can significantly impact the effectiveness and performance of the neural network during training and inference.

Model Architecture

- **Convolutional Layers**: Convolutional layers are designed to extract features from input data using filters. Each filter, or kernel, scans a small window of the input and performs a mathematical operation called convolution. This operation involves element-wise multiplication between the filter and a portion of the input, followed by summing the results to produce a single value in the output feature map. This process is repeated across the entire input, resulting in a series of feature maps that represent various patterns or characteristics of the input data.

  In the case of MFCCs, convolutional layers apply filters to the MFCC features to capture patterns that are indicative of different audio characteristics. Since MFCCs are typically represented as a sequence of coefficient vectors over time, the filters detect local patterns in these sequences, producing feature maps that highlight temporal dependencies and spectral features relevant to the audio signal.

  For spectrograms, which are 2D representations of audio signals with time on one axis and frequency on the other, convolutional layers process the spectrograms by sliding filters across both dimensions. Each filter extracts local patterns or spectral features within a specific receptive field of the spectrogram. The result is a series of feature maps that reveal significant frequency components and temporal patterns within the spectrogram, enhancing the representation of the audio data for further analysis.

In this model, we have three convolutional layers. The filters used in each convolutional layer have a size of 3x3. The first convolutional layer has 32 filters, the second convolutional layer also has 32 filters, and the third convolutional layer has 64 filters. Each filter learns to extract different features from the input data.

- **Activation Function (ReLU)**: Rectified Linear Unit is used as the activation function after each convolutional layer. ReLU sets all negative values in the feature maps to zero and keeps the positive values unchanged.

- **Batch Normalization**: Batch Normalization is applied after each convolutional layer. It helps normalize the activation of the previous layer, making the training process more stable.

- **Pooling Layer**: Max-Pooling is performed with a window size of 4x6 for the MFCC data. The reason for this choice is in the number of MFCC; in this case, 13. With a group of 3's on the y-axis, 13 MFCCs can be max pooled and still leave some room for the RNN to perform further operations in cases of adding LSTM. For the spectrogram images, a window size of 2×2 was used. This choice is motivated by the 2D structure of spectrograms. Each 2×2 pooling window scans over adjacent regions of the spectrogram, selecting the maximum value within each window. This operation reduces the spatial dimensions of the feature maps by selecting the maximum value within each window. The pooling operation is performed with "same" padding, which means the spatial dimensions of the feature maps remain the same after pooling.

- **Flattening**: After the convolutional and pooling layers, the feature maps are flattened into a 1D vector. This is done to prepare the data for the fully connected layers.

- **Dense Layers**: The first Dense layer has 32 neurons, the second Dense layer has 16 neurons, and the third Dense layer has 8 neurons. These layers perform high-level reasoning based on the extracted features. ReLU activation function is used in the Dense layers to introduce non-linearity.

- **Dropout Rate**: Dropout is applied with a rate of 0.3 after the first Dense layer, and with rates of 0.2 after the second and third Dense layers. Dropout randomly sets a fraction of the input units to zero during training, preventing over-fitting.

- **Output Layer**: The output layer has 4 neurons, representing the four classes of time signatures (e.g., 2/4, 3/4, 5/4, 7/4). Softmax activation function is applied to the output layer, which converts the raw predictions into probabilities for each class.

Training and Optimization

- **Training Procedure**: In this study, a **batch size of 64** was employed for training the CNN. The model was trained over **200 epochs on the MFCCs** and **10 epochs on the spectrograms** to iteratively learn from the dataset. The varying number of epochs for training on MFCCs (200 epochs) versus spectrograms (10 epochs) reflects the different complexity and information content of the two feature representations, requiring more iterations to learn from the MFCCs compared to the spectrograms. During training, the **Adam optimizer** with a **learning rate of 0.001** was utilized to update the CNN's parameters based on the computed gradients. This training procedure allowed the model to optimize its weights and biases by minimizing the loss function, ultimately enhancing its ability to make accurate predictions on the training dataset.

- **Hyper-parameter Tuning**: was conducted to optimize the performance of the model. The hyper-parameters considered for tuning included the learning rate, batch size, number of epochs. A grid search approach was employed, where different combinations of hyper-parameter values were systematically evaluated to identify the optimal settings. For instance, the learning rate was tested across a range of values (e.g., 0.001, 0.01, 0.1) to determine its impact on model convergence and performance. Similarly, different batch sizes (e.g., 32, 64, 128) were explored to assess their influence on training stability and generalization. The number of epochs was varied (e.g., 50, 100, 150, 200) to find a balance between model convergence and over-fitting.

8.2.2 Convolutional Recurrent Neural Network (CRNN)

The CRNN architecture builds upon the CNN by incorporating recurrent layers to capture temporal dependencies in the audio signals. Similar to the CNN, we used two distinct input features: MFCC and spectrogram. The MFCC was inputted as raw array vector values, and the spectrogram was treated as an image. These features were used separately to evaluate their effectiveness in the CRNN model, allowing the recurrent layers to exploit temporal patterns while the convolutional layers handled spatial feature extraction for improved meter detection.

LSTM Layers

The LSTM (Long Short-Term Memory) layers are a type of recurrent neural network layer that can effectively model sequential data by preserving information over long time steps. The LSTM units are responsible for learning and remembering relevant information from previous time steps and using it to make predictions. The first LSTM layer consists of 64 LSTM units, which are responsible for capturing temporal dependencies in the input

data. The dropout parameter is set to 20%, which helps to regularize the layer and prevent over-fitting. The layer returns the full sequence of hidden states as output. The second LSTM layer is similar to the first, with 64 LSTM units and a 20% dropout. However, the return sequences parameter is not specified, so the layer defaults to returning only the final hidden state. These LSTM layers are added after the convolutional layers in the CRNN model, allowing the model to capture both local and temporal features in the input data. The LSTM layers contribute to the model's ability to understand the sequential nature of the input data and extract meaningful representations for time signature classification.

Training and Optimization

- **Training Procedure**: A **batch size of 64** was utilized for training the CRNN. The model was trained over **200 epochs on MFCCs** and **30 epochs on spectrograms** to progressively learn from the dataset, reflecting the varying informational content of these two feature representations. Throughout training, the **Adam optimizer** with a **learning rate of 0.001** was employed to adjust the CRNN's parameters based on calculated gradients. This method optimized the model's weights and biases by minimizing the loss function, improving its predictive accuracy on the training dataset.

- **Parameter Tuning**: The model's performance was refined through hyper-parameter optimization. This process involved tuning essential parameters such as the learning rate, batch size, and epochs. A grid search methodology was applied, systematically examining various combinations of hyper-parameter values to identify the optimal configurations. For example, the impact of different learning rates (e.g., 0.001, 0.01, 0.1) on model convergence and performance was thoroughly investigated. Similarly, diverse batch sizes (e.g., 32, 64, 128) were explored to evaluate their effects on training stability and generalizability. Additionally, the number of training epochs was adjusted (e.g., 50, 100, 150, 200) to strike a balance between model convergence and mitigating over-fitting.

8.2.3 ResNet18

A significant aspect of our research, which was published [4], involved utilizing the ResNet18 architecture, as detailed in Figure 8.1. The choice to use ResNet18 was motivated by the nature of the dataset used, specifically the Meter2800 dataset. Being a relatively lightweight variant of the ResNet architecture, it was considered suitable for this dataset due to its ability to handle smaller datasets while still capturing important features and avoiding over-fitting. By leveraging ResNet18, the model can effectively learn from the transformed audio data, enabling accurate classification and analysis of the meter in the given dataset.

It consists of 18 layers, including convolutional layers, batch normalization layers, ReLU activation functions, and a fully connected layer. It incorporates residual connections, which are skip connections that allow the network to learn residual mappings rather than explicit feature mappings. To fully understand its workings, the details of the layers are as follows:

Model Architecture

- Convolutional Layer: The input data is initially passed through a convolutional layer with a 7x7 kernel and a stride of 2. This layer performs a set of convolutions to extract low-level features from the input data.

  Using the MFCC array values, the model is slightly adjusted. It consists of an initial Conv2d layer followed by the ResNet-18 base model with modified fully connected layers. The first Conv2d layer takes the input MFCC, which is a 2D representation of audio, with 1 channel (since they are not images) and outputs a tensor with 3 channels. The MFCCs were pre-processed and stored in array files `(.npy)` and this step helped with faster data processing, so that the model can handle batches of such files efficiently during training and inference. The purpose of this layer is to transform the input to a suitable dimension for the subsequent layers. The output from the first layer is passed to the ResNet-18 Base Model for further processing. After the base model, we replace the original fully connected layer (the last layer of ResNet-18) with our custom sequential layers. These fully connected layers are responsible for mapping the learned features from the convolutional layers to the desired number of output classes. In this case, the architecture includes four linear layers with ReLU activation functions in between. The first linear layer takes 512 input features (output size of the last layer of ResNet-18) and outputs 128 features. The subsequent linear layers reduce the dimensionality further, resulting in 32, 8, and finally the number of output classes. The architecture can be found below;

```
        self.conv1 = torch.nn.Conv2d(1, 3, kernel_size=1)
        self.base_model = models.resnet18(weights=None)
        self.base_model.fc = torch.nn.Sequential(
            torch.nn.Linear(512, 128),
            torch.nn.ReLU(),
            torch.nn.Linear(128, 32),
            torch.nn.ReLU(),
            torch.nn.Linear(32, 8),
            torch.nn.ReLU(),
            torch.nn.Linear(8, num_classes)
        )
```

- Max Pooling: A 3x3 max pooling operation with a stride of 2 is applied after the first convolutional layer. It reduces the spatial dimensions of the feature maps, aiding in down-sampling.

- Basic Blocks: The core building blocks of ResNet, known as Basic Blocks, are repeated throughout the network. Each Basic Block consists of two 3x3 convolutional layers with a fixed number of filters followed by batch normalization and ReLU activation. These convolutional layers learn to capture and enhance higher-level features.

- Residual Connections: Residual connections, or skip connections, are introduced between certain Basic Blocks. These connections allow the network to directly propagate information from earlier layers to later layers. By preserving the original information, they facilitate the training of deeper networks by mitigating the degradation problem.

- Down-sampling: Down-sampling is performed using convolutional layers with a stride of 2 at specific stages in the network (conv3_1, conv4_1, and conv5_1). This down-sampling reduces the spatial dimensions of the feature maps while increasing their depth.

- Fully Connected Layer: At the end of the network, a fully connected layer is applied to the extracted features. This layer aggregates the features and maps them to the corresponding output classes. For the classification using the ImageNet dataset, the output layer consists of 1000 neurons representing the 1000 classes.

In summary, these 18 layers include the initial convolutional layer which is counted as one layer, four sets of Basic Blocks, each set containing two Basic Blocks. Therefore, there are a total of 4 sets $\times$ 2 Basic Blocks which is 8 Basic Blocks. The max pooling layer is not counted as a separate layer since it is a pooling operation applied after the convolutional layer. The final fully connected layer, which maps the extracted features to the output classes, is counted as one layer. However, the ResNet architecture also includes down-sampling layers, which reduce the spatial dimensions of the feature maps. In ResNet18, down-sampling occurs at specific stages in the network, namely conv3_1, conv4_1, and conv5_1. These down-sampling layers are considered as separate layers, bringing the total count to 18.

By utilizing both classic machine learning models and state-of-the-art deep learning architectures, our goal was to comprehensively assess the performance of various approaches on the Meter2800 dataset. The insights gained from these experiments highlight the strengths and limitations of each model and contribute to advancing the field of time signature detection in music.

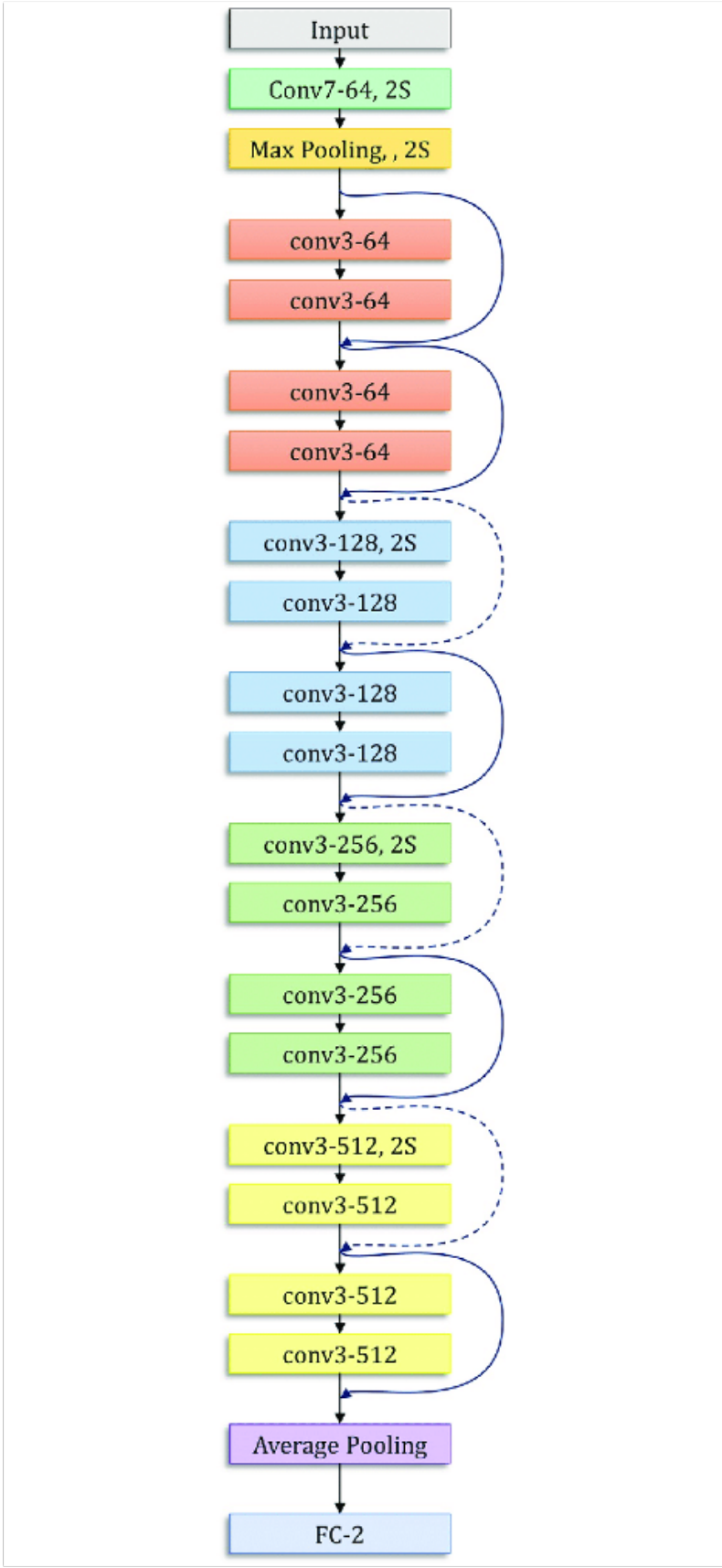

Figure 8.1: The ResNet18 architecture

Training and Optimization

- Training Procedure: Using the Meter2800 dataset, two sets of data were prepared; MFCC as array values and spectrogram as images. The MFCC representation of audio signals can be used as raw values and could also be converted to "images" as shown in 8.2 and used in a manner similar to image classification tasks. However, in this case, it is used as raw values. A **batch size of 32** was utilized for training the RESNET on MFCC and **batch size of 64** on spectrograms. The model was trained over **50 epochs on MFCCs** and **30 epochs on spectrograms** to progressively learn from the datasets.

- Parameter Tuning: The model's performance was enhanced through hyper-parameter optimization, which involved tuning critical parameters like learning rate, batch size, and epochs. Using a systematic grid search approach, various combinations of hyper-parameter values were evaluated to determine optimal configurations, considering factors such as model convergence, training stability, and generalizability.

- Optimizer and Loss Function: Throughout training, the **Adam optimizer** with a **learning rate of 0.001** was used on both input features. The optimizer adjusts the weights during back-propagation to minimize the defined loss function. In this case, the **CrossEntropyLoss function** was used, which is suitable for multi-class classification tasks. For binary classification (where the number of classes M equals 2 as in the case of $\frac{3}{4}$ and $\frac{4}{4}$ classes), cross-entropy is calculated as:

$$-(y \log(p) + (1 - y) \log(1 - p))$$

and in the case of $M > 2$ (i.e., multiclass classification), we calculate a separate loss for each class label per observation and sum the result:

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

where, y represents the true label, p is the predicted probability for the positive class in binary classification, and $y_{o,c}$ and $p_{o,c}$ are the true and predicted probabilities, respectively, for each class c in multi-class classification for a specific observation o.

In summary, all the parameters used for both set of inputs are:

- Number of MFCC coefficients: 13
- Mean for normalization: [0.5, 0.5, 0.5]
- Standard deviation for normalization: [0.5, 0.5, 0.5]

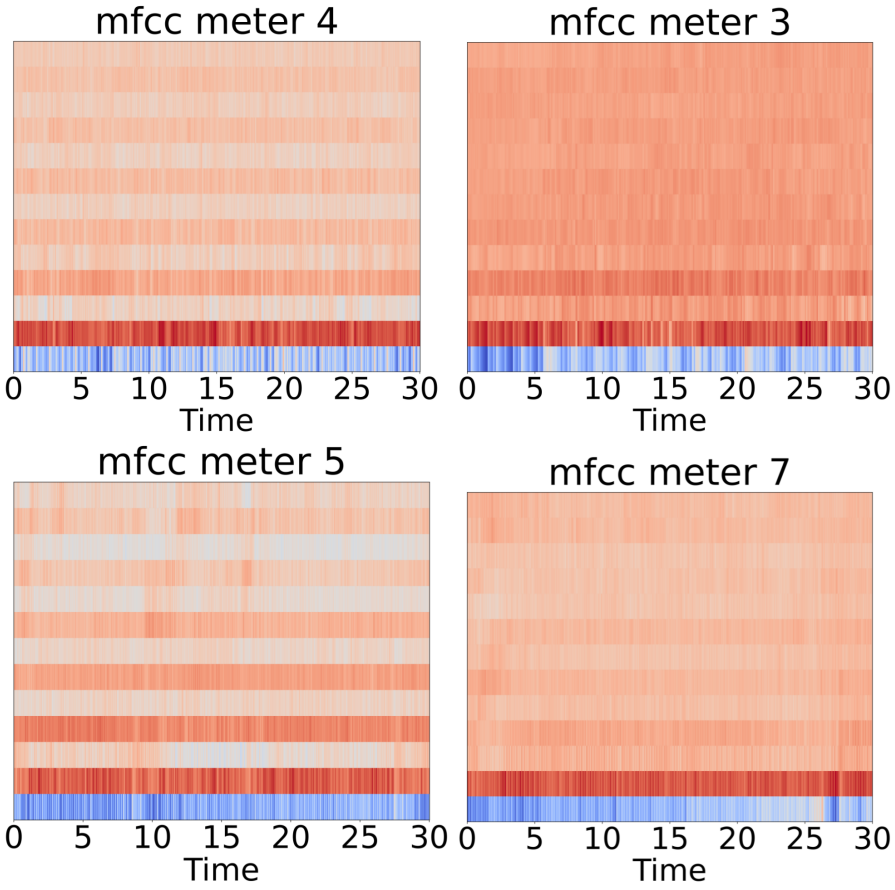

Figure 8.2: Sample MFCCs for one audio signal from each music meter class

- Batch size: 64 for spectrogram and 32 for MFCC
- Number of epochs: 30 for spectrogram and 50 for MFCC
- Optimizer: Adam
- Loss function: Cross Entropy Loss

8.2.4 ResNet18-LSTM

In our exploration of advanced architectures, we also integrated ResNet with LSTM networks to leverage both spatial and temporal features of the audio data. The ResNet-LSTM architecture combines the strengths of residual learning with the sequential modeling capabilities of LSTM networks. The architecture is given below:

```
    self.base_model = models.resnet18()
    self.base_model.avgpool = nn.Identity()
    self.lstm1 = nn.LSTM(512, hidden_size=128, num_layers=2)
    self.lstm2 = nn.LSTM(128, hidden_size=128, num_layers=2)
    self.avgpool = nn.AdaptiveAvgPool1d(1)
    self.fc = nn.Sequential(
        nn.Linear(128, 32),
        nn.ReLU(),
        nn.Linear(32, 8),
        nn.ReLU(),
        nn.Linear(8, num_classes)
    )
```

Training and Optimization

- Training Procedure: Two sets of data were also produced for this model: spectrogram pictures and MFCC array values. The RESNET18-LSTM was trained on MFCC with a **batch size of 128** and spectrograms with a **batch size of 64**. To gradually learn from the datasets, the model was trained over **150 epochs on MFCCs** and **40 epochs on spectrograms**.
- Parameter Tuning: Hyper-parameter optimization, which entailed adjusting crucial parameters including learning rate, batch size, and epochs, improved the model's performance. To find the best configurations, a methodical grid search technique was used to examine different hyper-parameter value combinations while taking into account the convergence of the model and training stability.

- Optimizer and Loss Function: Both input features were trained using the **Adam optimizer** at a **learning rate of 0.001**. The **CrossEntropyLoss function** was applied in this instance, and it works well for situations involving many classes of categorization.

8.3 Results and Discussion

8.3.1 Model Evaluation Metrices

Each method was rigorously evaluated using several performance metrics to assess their efficacy in detecting time signatures accurately. Accuracy, precision, recall, and F1-score were calculated for comparative analysis across the algorithms. The accuracy metric measures the overall correctness of the model predictions, while precision quantifies the correctness of the positive predictions. Recall assesses the model's ability to identify all positive instances, and the F1-score balances precision and recall.

Accuracy

Accuracy measures the overall correctness of predictions made by a model. It is calculated as the ratio of correctly predicted samples to the total number of samples:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Samples}}$$

Recall

Recall, also known as sensitivity or true positive rate, assesses a model's ability to identify all positive instances correctly. It is calculated as the ratio of true positives to the sum of true positives and false negatives:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Precision

Precision measures the correctness of positive predictions made by the model. It is calculated as the ratio of true positives to the sum of true positives and false positives:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

F1-Score

F1-Score is the harmonic mean of precision and recall, providing a balance between these two metrics. It is calculated as follows:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Where: True Positives (TP) are the correctly predicted positive instances. True Negatives (TN) are the correctly predicted negative instances. False Positives (FP) are the instances incorrectly predicted as positive. False Negatives (FN) are the instances incorrectly predicted as negative.

8.3.2 Machine Learning Algorithms

Among the machine learning models, four methods were evaluated for their effectiveness in detecting time signatures in audio signals using the Meter2800 dataset: SVM, KNN, Naive Bayes, and Random Forest. The results obtained from these methods are shown below.

Performance on 2 Classes

From Table 8.1, it can be observed that SVM exhibits the highest accuracy (86.67%) and F-score among the machine learning models, indicating its robust performance in classifying time signatures. Naive Bayes demonstrates comparable performance (accuracy of 84.50%) with slightly lower precision compared to SVM. KNN and Random Forest also display competitive accuracy (84.83% and 86.00%, respectively) but show slightly lower precision and recall (0.8484 and 0.8600, respectively) compared to SVM and Naive Bayes.

Table 8.1: Machine learning models classification report for 2 classes

Model	Accuracy	F-Score	Recall	Precision
SVM	**0.8667**	**0.8661**	**0.8667**	**0.8733**
KNN	0.8483	0.8483	0.8483	0.8484
Naive Bayes	0.8450	0.8440	0.8450	0.8537
Random Forest	0.8600	0.8592	0.8600	0.8679

Performance on 4 Classes

The results from Table 8.2 highlight that SVM achieves the highest accuracy (74.29%) and F-score among the classical machine learning models. However, it's noticeable that Naïve Bayes falls slightly behind in accuracy (66.00%) but demonstrates competitive

performance in precision. KNN and Random Forest show moderate accuracy (70.43% and 72.57%, respectively), with Random Forest exhibiting the highest precision (77.42%) among the models.

Table 8.2: Machine learning models classification report for 4 classes

Model	Accuracy	F-Score	Recall	Precision
SVM	**0.7429**	0.6957	**0.7429**	0.7684
KNN	0.7043	**0.6972**	0.7043	0.6932
Naïve Bayes	0.6600	0.6733	0.6600	0.6982
Random Forest	0.7257	0.6702	0.7257	**0.7742**

8.3.3 Deep Learning Models

The evaluation of deep learning models for time signature detection revealed significant promise in their capability demonstrating high accuracy and robustness for this task.

Performance on 2 Classes

In the context of binary classification, the performance of these deep learning models is captured in Table 8.3. It presents the classification report for deep learning models trained on two different types of feature representations: MFCC and Spectrogram. For both feature types, four models were evaluated: CNN, CRNN, ResNet18, and ResNet18-LSTM. In the case of MFCC features, all models achieved high accuracy ranging from 0.8800 to 0.8900. The F-scores, which balance precision and recall, were also consistently high, ranging from 0.8795 to 0.8896. Also, ResNet18 and ResNet18-LSTM exhibited slightly lower precision compared to CNN and CRNN. For Spectrogram features, the performance of the models was even more impressive, with accuracy ranging from 0.9800 to 0.9967. ResNet18 achieved the highest accuracy of 0.9967, indicating its superior ability to classify the data. Overall, the results demonstrate the effectiveness of deep learning models in classifying data represented by both MFCC and Spectrogram features, with ResNet18 showing particularly promising performance across both feature types.

Performance on 4 Classes

Expanding the evaluation to a scenario involving multiple classes (4 classes) allowed for a more comprehensive assessment of the deep learning models. Across the diverse classification landscape, the performance of these models exhibited varying degrees of efficiency. Detailed insights into the performance of each deep learning model in the context of the 4 classes can be found in Table 8.4

Table 8.3: Deep learning models classification report for 2 classes

Model	Accuracy	F-Score	Recall	Precision
Deep Learning Models with MFCC				
CNN	**0.8900**	**0.8896**	**0.8900**	**0.8964**
CRNN	0.8800	0.8795	0.8800	0.8862
RESNET18	0.8850	0.8835	0.8837	0.8860
RESNET18-LSTM	**0.8900**	**0.8896**	**0.8900**	0.8957
Deep Learning Models with Spectrogram				
CNN	0.9800	0.9800	0.9800	0.9808
CRNN	0.9883	0.9883	0.9886	0.9883
RESNET18	**0.9967**	**0.9967**	**0.9967**	**0.9967**
RESNET18-LSTM	0.9850	0.9850	0.9854	0.9850

Table 8.4: Deep learning models classification report for 4 classes

Model	Accuracy	F-Score	Recall	Precision
Deep Learning Models with MFCC				
CNN	0.8129	0.8004	0.8129	0.8104
CRNN	0.8029	0.7914	0.8029	0.7973
RESNET18	0.8514	0.8502	0.8514	0.8537
RESNET18-LSTM	**0.8600**	**0.8588**	**0.8600**	**0.8608**
Deep Learning Models with Spectrogram				
CNN	0.8286	0.7767	0.8286	0.8734
CRNN	0.8557	0.8076	0.8557	0.8460
RESNET18	**0.9071**	**0.9003**	**0.9071**	**0.9023**
RESNET18-LSTM	0.8557	0.8944	0.8557	0.8128

The comparative analysis revealed a significant performance disparity between classical machine learning algorithms and deep learning models. While traditional methods exhibited reasonable results, they notably lagged behind deep learning architectures, underscoring their limitations in capturing intricate patterns essential for precise time signature detection. Particularly, ResNet18 consistently outperformed CNN and CRNN in this context, highlighting the efficacy of its residual connections and skip connections.

ResNet18's ability to robustly capture and preserve critical features within the Meter2800 dataset underscores its adaptability and reliability in discerning subtle patterns inherent to time signature classification tasks. The obtained results affirm ResNet18's proficiency in tackling the challenging task of time signature detection, owing to its enhanced capacity for learning and representation. Table 8.5 succinctly illustrates these findings, emphasizing ResNet18's superior performance compared to other architectures in terms of model complexity and computational efficiency.

Table 8.5: Trainable and non-trainable parameters for different deep learning models

Model	**Trainable params**	**Non-trainable params**
Using MFCC Input features		
ResNet18	11,263,504	11,008
ResNet18-LSTM	13,679,888	9,728
CNN	35,196	256
CRNN	76,612	96
Using Spectrogram Input features		
ResNet18	25,600,000	22,000
ResNet18-LSTM	28,000,000	20,000
CNN	75,000	500
CRNN	150,000	200

8.4 Summary

In this chapter, we explored the application of both classic machine learning algorithms and deep learning models for time signature detection using the Meter2800 dataset. We evaluated a range of models including SVM, Random Forest, Naive Bayes, KNN, CNN, CRNN, ResNet18, and ResNet18-LSTM, analyzing their performance across different feature representations such as MFCC and spectrograms. The classic machine learning classifiers demonstrated reasonable performance but were outperformed by deep learning architectures, particularly ResNet18, which consistently showed superior accuracy and F1-scores in both binary and multi-class classification scenarios. The robustness of ResNet18 in capturing nuanced patterns within audio data highlights its efficacy for time

signature classification tasks, supported by its residual connections and skip connections which enhance learning and representation capabilities.

Overall, the results showed the impact of deep learning in time signature detection, showcasing ResNet18 as a leading model for accurate and reliable time signature detection [4]. This chapter also provided comprehensive evaluation metrics to validate the effectiveness of each approach, thus contributing valuable insights into advancing machine learning applications in music analysis.

Chapter 9

Ensemble Techniques

9.1 Introduction

In order to improve predictive performance and generate results that are more reliable and accurate than those produced by a single model, ensemble learning techniques make use of the idea of merging many models. The "wisdom of the crowd" theory[158], which holds that the combined decisions of various models often exceed those of a single model, is the foundation for the reasoning behind ensemble approaches. These methods are widely used in many different domains, including data mining[9], statistics, and machine learning[160]. Their adaptability and efficacy in enhancing prediction performance are well-known. Compared to single models, they can handle complicated relationships within data more efficiently, are less prone to over-fitting, and provide superior generalisation. Although ensemble approaches have advantages, training numerous models may make them computationally expensive and lower their interpretability when compared to individual models.

The underlying principle of this method is building a collective model from several base models, which may or may not be of the same kind. Typically, ensembles are made up of a variety of models with varying algorithmic approaches, feature representations, and training approaches. The goal of model diversity is to capture many facets of the data and increase overall forecast accuracy.

In this chapter, we employed various ensemble techniques on various models using MFCC and spectrograms. For each technique, we evaluated its effectiveness in improving the accuracy and robustness of time signature detection systems which are found in result section of this chapter 9.3. For MFCCs, 8 models (SVM, KNN, Random Forest, Naive Bayes, CNN, CRNN, RESNET18, RESNET18-LSTM) were considered and 4 models (CNN, CRNN, RESNET18, RESNET18-LSTM) for spectrogram. The reason for the smaller number of models for spectrograms is that the data is image-based and cannot be used with the other machine learning models. Some of the ensemble techniques are:

9.1.1 Soft Voting

The ensemble employs soft voting to get a more accurate final prediction by averaging the probability ratings of each class projected by several models. Taking into account the levels of confidence associated with each conclusion results in a more probabilistic decision-making process.

Let M be the ensemble of individual models, where m_i represents an individual model in the ensemble, and C be the set of possible classes for a given prediction task. The output of each model m_i for a particular input can be denoted as o_i, where o_i is in C. Additionally, let $P(o_i = c)$ represent the probability score assigned by the i-th model to the class c.

The soft voting mechanism computes the final probability distribution for each class c as given in the equation below:

$$P(V_s = c) = \frac{1}{N} \sum_{i=1}^{N} P(o_i = c) \tag{9.1}$$

Here, $P(V_s = c)$ represents the final probability score for class c in the ensemble, and N is the total number of individual models. The class with the highest final probability is selected as the ensemble's prediction.

9.1.2 Weighted Voting

Weighted voting is an approach that improves accuracy by using the collaborative capabilities of different models. It accomplishes this by giving distinct weights to each model prediction. These weights indicate each model's confidence or credibility. For example, a base model with 95% accuracy will be given more weight than another with 90% accuracy. The class or outcome is determined by the weighted sum of predictions. This strategy enables a more detailed examination of the predictive powers of individual models.

Let M be the ensemble of individual models, where m_i represents an individual model in the ensemble, and C be the set of possible classes for a given prediction task. The output of each model m_i for a particular input can be denoted as o_i, where $o_i \in C$.

Let W_i represent the weight assigned to the prediction of model m_i, and N be the total number of models in the ensemble.

The weighted vote, V, is determined by computing the weighted sum of individual predictions:

$$V = \text{argmax}_{c \in C} \sum_{i=1}^{N} W_i \cdot I(o_i = c) \tag{9.2}$$

Here, $I(o_i = c)$ is the indicator function that takes the value 1 if $o_i = c$ (i.e., the i-th

model predicts class c), and 0 otherwise. The weighted vote V corresponds to the class with the highest weighted cumulative sum across all individual models in the ensemble.

9.1.3 K-Ranked Voting

This technique uses the idea of rank assignment to predict classes while taking into consideration the probability predicted by each model. Each predicted class is given a weight based on its rank, which is determined by the value of K. In this example, $K = 3$. As a result, the class with the highest probability is assigned a weight of three, followed by the second-highest class with a weight of two, and the third-highest with a weight of one. These weighted predictions are then added across all individual models in the ensemble, with an argmax operation used to choose the class with the largest cumulative sum, and this eventually decides the result.

Let M be the ensemble of individual models, where m_i represents an individual model in the ensemble, and C be the set of possible classes for a given prediction task. The output of each model m_i for a particular input can be denoted as o_i, where o_i is in C.

The K-ranked vote, V_k, with $k = 3$, is determined by assigning weights to the predicted classes based on their rank, considering the probabilities of each class predicted by the individual models:

$$V_k = \text{argmax}_{c \in C} \sum_{i=1}^{N} \text{Rank}\left(o_i, \text{P}(o_i = c)\right) \tag{9.3}$$

Here, $\text{Rank}\left(o_i, \text{P}(o_i = c)\right)$ represents the rank assigned to the predicted class c by the i-th model, considering the probability of class c predicted by that model. The K-Ranked Vote V_k corresponds to the class with the highest cumulative sum across all individual models in the ensemble, considering the top k-ranked classes.

9.1.4 Stacked Voting

Stacked voting, also known as stacking, entails developing a meta-model that combines the predictions of numerous base models. Stacking, as opposed to standard weighted voting, entails employing a higher-level model to learn how to optimally combine various models' predictions.

Let M be the ensemble of individual models, where m_i represents an individual base model in the ensemble, and C be the set of possible classes for a given prediction task. The output of each base model m_i for a particular input can be denoted as o_i, where $o_i \in C$.

The stacked vote, V, is determined by training a meta-model (e.g., a classifier) on the predictions of the individual models. Let S be the stacked model, and w_i be the weight assigned to the output of base model m_i.

$$V = \text{argmax}_{c \in C} S \left(\sum_{i=1}^{N} w_i \cdot o_i \right) \tag{9.4}$$

Here, S is the function representing the meta-model, and $\sum_{i=1}^{N} w_i \cdot o_i$ is the aggregated input to the meta-model. The stacked vote V corresponds to the class with the highest predicted probability or score from the meta-model.

Particularly, we implementing stacking with various base models, including:

- Stacking with Gradient Boosting (Stacking-GB): A meta-model that combines the predictions of individual models using a Gradient Boosting classifier [53, 115] as the base learner.
- Stacking with Extreme Gradient Boosting (Stacking-XGB): A stacking method where the meta-model employs Extreme Gradient Boosting (XGBoost) [53] to learn the optimal combination of base models' predictions.
- Stacking with CatBoost (Stacking-CatB): A stacking technique that utilizes the CatBoost algorithm [124], known for its efficiency and performance with categorical features, as the meta-model.
- Stacking with Random Forest (Stacking-RF): This method involves using a Random Forest classifier as the meta-model to combine the outputs of the base models.

Each stacking model (Stacking-GB, Stacking-XGB, Stacking-CatB, and Stacking-RF) was trained to optimize the final prediction by learning the best way to combine the base models' outputs.

9.1.5 Bayesian Model Averaging

Bayesian Model Averaging (BMA) combines predictions from multiple models by taking into account their respective uncertainties. In BMA, each model contributes to the final prediction with a weight proportional to its posterior model probability [157].

Mathematically, let M be the ensemble of individual models, where m_i represents an individual model in the ensemble, and C be the set of possible classes for a given prediction task. The output of each model m_i for a particular input can be denoted as o_i, where $o_i \in C$.

Let $P(m_i|D)$ be the posterior probability of model m_i given the observed data D. The Bayesian Model Averaging vote, V, is determined by computing the weighted sum of individual predictions based on their posterior probabilities:

$$V = \text{argmax}_{c \in C} \sum_{i=1}^{N} P(m_i|D) \cdot I(o_i = c) \tag{9.5}$$

Here, $I(o_i = c)$ is the indicator function that takes the value 1 if $o_i = c$ (i.e., the i-th model predicts class c), and 0 otherwise. The BMA vote V corresponds to the class with the highest weighted cumulative sum across all individual models in the ensemble.

In practice, the posterior probabilities $P(m_i|D)$ are typically estimated using Bayesian inference methods such as Markov Chain Monte Carlo (MCMC[24] or Variational Inference [63]).

9.2 Data Preparation and Aggregation for Accurate Predictions

The audio samples used for each of these models were segmented into 10 parts, resulting in a total of 28000 samples as seen from the 5. This strategy provided more samples for the models to train and increase their ability to capture temporal dependencies in the audio data. However, it is essential to acknowledge that obtaining accurate time signatures within a 3-second interval may not be realistic. Consequently, after processing, it becomes essential to aggregate predictions on the test data.

This aggregation process consolidates predictions made by the model across all 10 segments of MFCCs for each audio sample. It entails obtaining the mode of all predictions as the final prediction and computing the mean of all prediction probabilities as the final probabilities as can be seen in Table 9.1.

The choice of aggregation method was determined based on the observations presented in Table 9.2. This table highlights the variations in accuracy on the test set for four-class meters, both before and after applying the aggregation process. Each individual model underwent evaluation, contributing to a comprehensive understanding of its performance before and after the aggregation.

9.2.1 Optimization of Weighted Voting Technique Using Genetic Algorithm

In our previous investigation, we employed a weighted voting technique to combine the predictions of multiple models. However, the assignment of weights was based solely on our intuition, without any systematic optimization or empirical justification. Specifically, we arbitrarily assigned weights to each model without exploring alternative weighting schemes or evaluating the impact of weight selection on overall performance. This approach, while yielding promising results, highlights the need for a more rigorous and data-driven approach to weight optimization in future studies.

The superior performance of RESNET18 and RESNET18-LSTM suggests that they should be assigned the highest weights in the weighted voting technique, enabling them to

Table 9.1: Aggregation of MFCCs in the test set

Filename	Prediction Probabilities	Prediction Class	True Class
Before Aggregation			
mfcc12-1	[0.0412 0.0408 0.3120 0.6059]	3	2
mfcc12-2	[0.0412 0.0407 0.3120 0.6059]	3	2
mfcc12-3	[0.0121 0.7090 0.2747 0.0040]	1	2
mfcc12-4	[0.0452 0.1336 0.4379 0.3831]	2	2
mfcc12-5	[0.0000 0.9999 0.0000 0.0000]	1	2
mfcc12-6	[0.0062 0.4020 0.5782 0.0135]	2	2
mfcc12-7	[0.0720 0.5561 0.3032 0.0687]	1	2
mfcc12-8	[0.0571 0.1101 0.3464 0.4863]	3	2
mfcc12-9	[0.3885 0.5660 0.0361 0.0092]	1	2
mfcc12-10	[0.0000 0.9999 0.0000 0.0000]	1	2
After Aggregation			
mfcc12	[0.0074 0.3469 0.3410 0.3046]	1	2

Table 9.2: Comparison of model accuracy and aggregated accuracy on the test set using MFCC

Model	Accuracy	Aggregated Accuracy
SVM	0.7374	0.7429
KNN	0.6527	0.7043
NAIVE BAYES	0.6317	0.6600
RANDOM-FOREST	0.7281	0.7257
CNN	0.7484	0.8129
CRNN	0.7464	0.8029
RESNET	**0.7920**	0.8514
RESNET-LSTM	0.7866	**0.8600**

Table 9.3: Comparison of model accuracy and aggregated accuracy on the test set using spectrogram

Model	Accuracy	Aggregated Accuracy
CNN	0.7477	0.8286
CRNN	0.7434	0.8557
RESNET	**0.7844**	**0.9071**
RESNET-LSTM	0.7754	0.8557

exert the most significant influence on the ensemble's decision-making process. However, due to their closely comparable accuracies and F-score values, determining the optimal weight assignments for these models becomes a complex optimization problem. This challenge necessitates the incorporation of a robust optimization technique, such as Genetic Algorithm (GA), to systematically determine the ideal weight distributions and maximize the ensemble's overall performance.

In this optimization process, which we also published in the Proceedings of the Companion Conference on Genetic and Evolutionary Computation (GECCO) [1], we initialized a weight vector w consisting of 8 real numbers between 0 and 1 i.e. $0 \leq w \leq 1, w \in \mathbb{R}$ for MFCCs and 4 real numbers between 0 and 1 i.e. $0 \leq w \leq 1, w \in \mathbb{R}$ for spectrograms. The GA was executed with a population size of 100 individuals. The GA parameters were set as follows: crossover was performed with a probability of 0.2 using a uniform crossover technique, a mutation probability of 0.6 to introduce diversity into the population. The algorithm iterated for a total of 100 generations and 50 multiple runs. In each iteration, the newly generated weight vectors were multiplied by the probability values corresponding to each model. Subsequently, predictions were obtained using the ensemble model based on the weighted probabilities, and the accuracy of the predictions was calculated.

9.3 Results and Discussion

9.3.1 Performance on 2 Classes

The table 9.4 below presents a comprehensive comparison of various ensemble voting techniques applied to two-class classification problems using two distinct feature sets: MFCC and spectrograms. The ensemble methods evaluated include all the learning techniques already mentioned in the previous chapter under section 9

The evaluation of ensemble techniques using MFCC features indicates that Weighted Voting achieved the highest performance, with an F-Score of 0.8543, Accuracy of 0.8571, Recall of 0.8610, and Precision of 0.8571. BMA also performed well, with consistent scores across all metrics (F-Score: 0.8256, Accuracy, Recall, Precision: 0.8343). Summed Voting and Soft Voting showed comparable performance, while Majority Voting, although effective, had slightly lower scores (F-Score and Accuracy: 0.7793, Recall and Precision: 0.8000).

In contrast, the use of spectrogram features significantly enhanced the performance of all ensemble techniques. Summed Voting, Soft Voting, and BMA achieved perfect scores across all metrics (0.9983). Weighted Voting and Stacking-XGB also demonstrated excellent performance, with scores of 0.9967. Stacking-GB, Stacking-CatB, and Stacking-RF performed well but slightly below the top methods, with F-Scores, Accuracy, and Recall at 0.9917 and Precision slightly higher at 0.9918.

Table 9.4: Comparison of ensemble voting techniques for 2 classes

Model	**F-Score**	**Accuracy**	**Recall**	**Precision**
Ensemble Techniques with MFCC				
Majority Voting	0.7793	0.8000	0.8239	0.8000
K-Ranked Voting	0.7973	0.8100	0.8151	0.8100
Summed Voting	0.8225	0.8329	0.8378	0.8329
Soft Voting	0.8225	0.8329	0.8378	0.8329
Stacking-GB	0.7793	0.7729	0.7902	0.7729
Stacking-XGB	0.8025	0.7943	0.8171	0.7943
Stacking-CatB	0.8045	0.7971	0.8175	0.7971
Stacking-RF	0.8118	0.8086	0.8179	0.8086
BMA	0.8256	0.8343	0.8348	0.8343
Weighted Voting	**0.8543**	**0.8571**	**0.8610**	**0.8571**
Ensemble Techniques with Spectrogram				
Majority Voting	0.9883	0.9883	0.9883	0.9886
K-Ranked Voting	0.9917	0.9917	0.9917	0.9918
Summed Voting	**0.9983**	**0.9983**	**0.9983**	**0.9983**
Soft Voting	**0.9983**	**0.9983**	**0.9983**	**0.9983**
Stacking-GB	0.9917	0.9917	0.9917	0.9918
Stacking-XGB	0.9967	0.9967	0.9967	0.9967
Stacking-CatB	0.9917	0.9917	0.9917	0.9918
Stacking-RF	0.9917	0.9917	0.9917	0.9918
BMA	**0.9983**	**0.9983**	**0.9983**	**0.9983**
Weighted Voting	0.9967	0.9967	0.9967	0.9967

9.3.2 Performance on 4 Classes

The various ensemble voting techniques for time signature detection were evaluated, comparing their performance across two different feature extraction methods: MFCC and spectrograms. The table 9.5 below presents a comprehensive comparison of the F-Score, Accuracy, Recall, and Precision metrics for each ensemble technique, highlighting the effectiveness of each approach.

Table 9.5: Comparison of ensemble voting techniques for 4-classes

Model	F-Score	Accuracy	Recall	Precision
Ensemble Techniques with MFCC				
Majority Voting	0.7793	0.8000	0.8239	0.8000
K-Ranked Voting	0.7973	0.8100	0.8151	0.8100
Summed Voting	0.8225	0.8329	0.8378	0.8329
Soft Voting	0.8225	0.8329	0.8378	0.8329
Stacking-GB	0.7793	0.7729	0.7902	0.7729
Stacking-XGB	0.8025	0.7943	0.8171	0.7943
Stacking-CatB	0.8045	0.7971	0.8175	0.7971
Stacking-RF	0.8118	0.8086	0.8179	0.8086
BMA	0.8256	0.8343	0.8348	0.8343
Weighted Voting	**0.8543**	**0.8571**	**0.8610**	**0.8571**
Ensemble Techniques with Spectrogram				
Majority Voting	0.8586	0.8586	0.8233	0.8782
K-Ranked Voting	0.8729	0.8729	0.8311	0.8219
Summed Voting	0.8829	0.8829	0.8429	0.8281
Soft Voting	0.8829	0.8829	0.8429	0.8281
Stacking-GB	0.9114	0.9114	0.9055	0.9051
Stacking-XGB	0.9171	0.9171	0.9120	0.9109
Stacking-CatB	0.9071	0.9071	0.9011	0.8987
Stacking-RF	**0.9200**	**0.9200**	**0.9147**	**0.9130**
BMA	0.8871	0.8871	0.8486	0.8313
Weighted Voting	0.8857	0.8857	0.8474	0.8304

In the analysis of ensemble techniques with MFCC for 4 classes, Weighted Voting emerges as the standout performer, achieving the highest F-Score, Accuracy, Recall, and Precision. BMA also demonstrates strong reliability, with consistent scores across all metrics, showcasing its effectiveness in aggregating model predictions. Meanwhile, Summed Voting and Soft Voting provide a good balance between simplicity and effectiveness, albeit with slightly lower performance compared to Weighted Voting and BMA. Majority Voting, although straightforward, exhibits comparatively lower performance, suggesting potential limitations in fully leveraging individual model strengths.

On the other hand, in the analysis of ensemble techniques with spectrogram for 4 classes, Stacking-RF emerges as the top-performing technique, showcasing superior per-

Table 9.6: Comparison of ensemble accuracy with and without GA

	Ensemble w/o GA	Ensemble with GA			
		Avg.	Max.	Min.	Std. dev.
Accuracy [%]	83.29	86.98	87.29	86.57	0.0014

formance across all metrics. This shows the effectiveness of stacking random forest models in ensemble learning for spectrogram features. Stacking-XGB, Stacking-GB, and Stacking-CatB also demonstrate strong performance, indicating the effectiveness of stacking models based on gradient boosting and categorical boosting algorithms. However, traditional ensemble methods like Majority Voting, K-Ranked Voting, Summed Voting, and Soft Voting exhibit relatively lower performance compared to stacking techniques, suggesting potential limitations in fully leveraging the predictive power of individual models when applied to spectrogram features.

It is important to highlight the impact of optimization techniques on ensemble models, particularly in enhancing predictive accuracy through refined parameter tuning and weight optimization methods such as genetic algorithms. Table 9.6 presents a comparison of ensemble models with 4 classes meters using summed voting (without GA) versus weighted voting (with GA) using MFCC features. In summed voting, probabilities are aggregated directly, whereas weighted voting involves multiplying probabilities by predetermined weights before aggregation. The results demonstrate a significant improvement in accuracy following GA optimization. Post-optimization, the ensemble achieved an average accuracy of 86.98%, with a peak accuracy of 87.29% and a minimum of 86.57%. This performance surpassed that of individual models like RESNET18 and RESNET18-LSTM, as well as the summed voting approach alone. These findings underscore the efficacy of GA in determining optimal weights for enhancing ensemble prediction accuracy.

9.4 Ablation Study of the Ensemble Models

In response to the varying performance of ensemble techniques, an exploratory analysis was conducted to investigate the impact of different model combinations. This ablation study aims to understand how the inclusion or exclusion of individual models affects the overall effectiveness of ensemble methods. The goal is to assess the contribution of each model within the ensemble framework and evaluate their influence on the ensemble's performance.

9.4.1 Model Removal in Ensemble Techniques

To assess the influence of each model on ensemble performance, we implemented a structured model removal approach. This involved iteratively excluding one model at a time, beginning with the weakest performer, to evaluate its impact on predictive accuracy. Table 9.7 presents the outcomes of this iterative process, detailing the performance metrics of different ensemble techniques as models are progressively removed. By systematically evaluating the effect of model exclusion on ensemble performance, we gained insights into the relative importance of individual models within the ensemble framework.

Models Removed:	KNN	KNN, Nayes	KNN, Nayes, RF,	KNN, Nayes, RF SVM	KNN, Nayes, RF, SVM, CNN	KNN, Nayes, RF, SVM, CNN, CRNN
Majority	0.8029	0.8114	0.8286	0.8343	0.8557	0.8500
Ranked	0.8157	0.8286	0.8386	0.8414	**0.8643**	0.8571
Summed	0.8357	0.8414	0.8514	0.8543	0.8614	**0.8700**
BMA	0.8329	0.8500	0.8529	0.8557	**0.8643**	0.8743
Soft	0.8357	0.8414	0.8514	0.8543	0.8614	**0.8700**
GBoost	0.8057	0.8171	0.8400	0.8486	0.8543	0.8586
XGBoost	0.8271	0.8357	0.8486	0.8557	0.8586	0.8514
CatBoost	0.8143	0.8114	0.8400	0.8514	0.8414	0.8529
RF	0.8171	0.8186	0.8243	0.8514	**0.8643**	0.8671
Weighted	**0.8657**	**0.8557**	**0.8571**	**0.8629**	**0.8643**	**0.8700**

Table 9.7: Performance scores of various voting strategies for different models removed

Initially, with only KNN removed as the base model, all ensemble techniques exhibit moderate performance, with Majority Voting yielding an F1 score of 0.8029. As additional models are removed, the overall performance trend shows consistent improvement, indicating the robustness of the ensemble framework in accommodating diverse model combinations. Additionally, the consistent superior performance of weighted voting across various model combinations highlights its effectiveness in combining predictions from multiple models. This shows its robustness and efficacy in ensemble voting strategies.

9.4.2 Ensemble Learning Combinations

The motivation behind combining different models lies in the pursuit of enhancing predictive performance and robustness. Individual models, whether based on Machine Learning (ML), Deep Learning (DL), or Classical (CL) techniques, each have their unique strengths and weaknesses. By integrating these diverse approaches, we aim to leverage the complementary strengths of each model type, thereby mitigating their individual limitations. This ensemble strategy is particularly beneficial in complex classification tasks, as it can lead to more accurate and reliable predictions through the aggregation of varied perspectives and methodologies. The method of combining these models are shown in Table 9.8.

The tables 9.9 and 9.10 compare the accuracies of various ensemble voting techniques using combinations of machine learning, deep learning, and classical model for 4-class and

Table 9.8: Ensemble learning combinations

Combination	Models Included
Only 2	Only Machine Learning Methods
Only 3	Only Deep Learning Methods
1-2	Classic Methods + Machine Learning Methods
1-3	Classic Methods + Deep Learning Methods
2-3	Machine Learning Methods + Deep Learning Methods
1-2-3	Classic Methods + Machine Learning Methods + Deep Learning Methods

2-class classifications. DL models include CNN, CRNN, ResNet18, and ResNet18-LSTM, while ML models comprise KNN, SVM, Naive Bayes, and Random Forest. CL model here is the MFCCSM only as it is a slight improvement from the others (ASM and BSM).

Table 9.9: Comparison of ensemble voting techniques showing their accuracies using combination of models for 2 classes

Model Type	Majority	Ranked	Summed	BMA	Soft	Weighted
ML	0.8567	0.8583	0.8617	0.8583	0.8617	0.8667
DL	**0.8933**	0.8883	**0.9067**	**0.9083**	**0.9066**	**0.9100**
CL_ML	0.8583	0.8650	0.8583	0.8600	0.8583	0.8533
CL_DL	0.8917	0.8900	0.8950	0.8917	0.8950	0.8700
ML_DL	0.8817	**0.8917**	0.8983	0.9017	0.8983	0.8850
ALL	0.8833	0.8883	0.8900	0.8883	0.8900	0.8783

Table 9.10: Comparison of ensemble voting techniques showing their accuracies using combination of models for 4 classes

Model Type	Majority	Ranked	Summed	BMA	Soft	Weighted
ML	0.7271	0.7514	0.7571	0.74571	0.7571	0.7557
DL	**0.8343**	**0.8414**	0.8543	**0.8557**	0.8543	**0.8686**
CL_ML	0.7300	0.7429	0.7629	0.7412	0.7629	0.6914
CL_DL	0.8329	0.8371	**0.8614**	0.8443	**0.8614**	0.8214
ML_DL	0.8000	0.8100	0.8329	0.8343	0.8329	0.7943
ALL	0.8014	0.8114	0.8229	0.8274	0.8229	0.7943

The tables present the performance of different ensemble voting techniques when applied to various combinations of ML, DL, and CL models for 2-class and 4-class classification tasks respectively. Among all the model types, deep learning models (DL) consistently show high performance across all ensemble techniques, indicating their robustness and ability to handle complex classification tasks effectively while the machine learning models (ML) demonstrate a moderate performance, performing well with certain ensemble techniques but generally not matching the performance levels of deep learning models.

They however perform better in 2-class classification than in 4-class classification, indicating that they are well-suited for simpler classification problems.

The classical models (CL) combined with machine learning or deep learning models show mixed results. When combined with deep learning models, classical models exhibit improved performance, while combinations with machine learning models show poorer results. Combining all model types provides a balanced approach, but does not always yield the highest performance, suggesting that adding more diverse models does not necessarily improve the overall accuracy.

9.5 Summary

To summarize this chapter, we explored various ensemble learning techniques aimed at enhancing the accuracy and reliability of time signature detection systems using audio features such as MFCCs and spectrograms. Ensemble methods capitalize on the diversity of individual models to achieve superior predictive performance compared to single models. Techniques like Soft Voting, Weighted Voting, K-Ranked Voting, Stacked Voting, and Bayesian Model Averaging were examined in detail, each leveraging the strengths of multiple base models to make collective predictions. We evaluated these methods across different classification tasks—2-class and 4-class—using a combination of machine learning (ML), deep learning (DL), and classical (CL) models, highlighting their effectiveness and relative performance in each scenario.

The results demonstrated that ensemble techniques significantly improved prediction accuracy, especially when integrating deep learning models like CNNs, CRNNs, ResNet18, and ResNet18-LSTM. These DL models consistently outperformed ML and CL models, particularly evident in spectrogram-based feature sets where complex image-like data required sophisticated modeling approaches. Conversely, ML models like SVMs, KNN, Naive Bayes, and Random Forests showed competitive performance in simpler classification tasks or when combined effectively with DL models. Furthermore, our study showed the importance of a good ensemble design, where model diversity and combination methods played crucial roles in achieving optimal performance across various classification scenarios.

Chapter 10

Summary

In this study, we began by addressing the crucial task of time signature detection in music, an area that had traditionally been tackled using models adept at handling MIDI signals. Initial investigations into past models highlighted their strengths and limitations, particularly in the context of audio signals versus MIDI signals. However, with the increasing prevalence of audio signals, the need for more sophisticated models capable of accurately processing audio data became evident. Our research objectives were centered around developing new models to bridge the gap left by traditional approaches, aiming to enhance the accuracy and robustness of time signature detection in audio signals.

We provided an extensive overview of music classification, covering aspects such as genre, tempo, mood, instrumentation, and singing voice detection. We reviewed existing methods and their performance, highlighting the challenges and limitations inherent in current techniques. This comprehensive review identified specific areas where traditional methods fell short, particularly in handling the complexity and variability of audio signals, thereby setting the stage for the subsequent development of improved models.

We also delved into the specifics of time signature detection, reviewing digital signal processing methods and introducing various models used in this domain. Key concepts such as the Audio Similarity Matrix (ASM) and Beat Similarity Matrix (BSM) were explored, and the potential of machine learning models to enhance detection accuracy was discussed. This theoretical foundation emphasized the need to address the shortcomings of existing approaches to improve detection performance. This research has led to the development of a new model, MFCCSM, which represents a significant enhancement over the traditional models examined. By leveraging Mel Frequency Cepstral Coefficients (MFCCs) for feature extraction, MFCCSM enhanced the detection accuracy of time signatures in audio signals. This improvement in accuracy supports and confirms **H1** 1.4, demonstrating that utilizing MFCCs as a tool for time signature detection can indeed lead to significant advancements in accurately identifying musical time signatures.

To validate our models, we created the Meter2800 dataset, consisting of 2800 annotated audio samples. Each sample was 30 seconds long, with annotations stored in CSV files to

facilitate accurate algorithm evaluations. The dataset was divided into train and test sets, categorized into four meter classes, and analyzed for its relationship between meter and tempo using scatter plots and statistical summaries. This comprehensive dataset was a critical contribution, enabling rigorous testing and validation of time signature detection models.

The use of optimization techniques, including Bayesian optimization and Genetic Algorithms, further enhanced model performance by fine-tuning learning parameters and achieving higher accuracy. This improvement in model performance confirms **H2** 1.4, demonstrating that Genetic Algorithms can indeed be effectively employed to optimize Mel-Frequency Cepstral Coefficient-based models for time signature detection.

We took our analysis further by exploring machine learning and deep learning models as alternatives to traditional approaches. This has led to the development of an array of innovative models that significantly outperform their predecessors in terms of accuracy and robustness. By combining the strengths of both machine learning (ML) and deep learning (DL) techniques, we've created robust hybrid models that leverage ensemble learning methods. Specifically, by integrating powerful algorithms like KNN, SVM, Naive Bayes, and Random Forest with cutting-edge DL architectures such as CNN, CRNN, ResNet18, and ResNet18-LSTM, we have confirmed **H3** 1.4. The application of the ResNet architecture within these models has proven effective for time signature detection, validating its potential in this context.

Finally, we employed several deep learning architectures as components of our ensemble approach. By aggregating the predictions from these advanced models, we created a robust ensemble that integrates the strengths of deep learning techniques. This significant improvement in accuracy and robustness confirms **H4** 1.4, demonstrating that the ensemble model leveraging deep learning techniques indeed provides superior performance compared to the base classifier models.

In summary, our research has made substantial contributions to the field of time signature detection in audio signals. We have demonstrated that combining traditional models with machine learning and deep learning approaches can significantly enhance performance. The development and validation of our new model, MFCCSM, using the Meter2800 dataset, alongside the application of advanced optimization techniques, have collectively resulted in a suite of models that set a new benchmark for accuracy and robustness. This work lays a strong foundation for future research and development in audio signal processing and music information retrieval, offering promising avenues for further advancements and applications.

Bibliography

[1] Jeremiah Abimbola, Daniel Kostrzewa and Pawel Kasprowski. 'Genetic Algorithm-Based Optimization ofWeighted Voting in Ensemble Models for Time Signature Detection'. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM. Melbourne, VIC, Australia: ACM, 2024, pp. 73–74. DOI: 10.1145/3638530.3664083.

[2] Jeremiah Abimbola, Daniel Kostrzewa and Pawel Kasprowski. *Meter2800*. Version V1. UNF:6:VaIntn9/q2FHhMcPvRTXYA==. 2022. DOI: 10.7910/DVN/0CLXBQ. URL: https://doi.org/10.7910/DVN/0CLXBQ.

[3] Jeremiah Abimbola, Daniel Kostrzewa and Paweł Kasprowski. 'METER2800: A novel dataset for music time signature detection'. In: *Data in Brief* 51 (2023), p. 109736.

[4] Jeremiah Abimbola, Daniel Kostrzewa and Pawel Kasprowski. 'Music time signature detection using ResNet18'. In: *EURASIP Journal on Audio, Speech, and Music Processing* 2024.1 (2024), p. 30.

[5] Jeremiah Abimbola, Daniel Kostrzewa and Pawel Kasprowski. 'Optimization of MFCCs for Time Signature Detection Using Genetic Algorithm'. In: *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. 2023, pp. 459–462.

[6] Jeremiah Abimbola, Daniel Kostrzewa and Pawel Kasprowski. 'Time signature detection: a survey'. In: *Sensors* 21.19 (2021), p. 6494.

[7] Ritesh Ajoodha, Richard Klein and Benjamin Rosman. 'Single-labelled music genre classification using content-based features'. In: *2015 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech)*. IEEE. 2015, pp. 66–71.

[8] M. Alonso, G. Richard and B. David. 'Extracting note onsets from musical recordings'. In: *2005 IEEE International Conference on Multimedia and Expo*. 2005, 4 pp. ISBN: 1945–788X. DOI: 10.1109/ICME.2005.1521568.

[9] Yael Amsterdamer, Yael Grossman, Tova Milo and Pierre Senellart. 'Crowd mining'. In: *Proceedings of the 2013 ACM SIGMOD international conference on Management of Data*. 2013, pp. 241–252.

[10] Hareesh Bahuleyan. 'Music genre classification using machine learning techniques'. In: *arXiv preprint arXiv:1804.01149* (2018).

[11] Nancy Bansal, Amit Verma, Iqbaldeep Kaur and Dolly Sharma. 'Multimodal biometrics by fusion for security using genetic algorithm'. In: *2017 4th International Conference on Signal Processing, Computing and Control (ISPCC)*. IEEE. 2017, pp. 159–162.

[12] Jayme Garcia Arnal Barbedo and George Tzanetakis. 'Musical instrument classification using individual partials'. In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.1 (2010), pp. 111–122.

[13] Abul Hashem Beg and Md Zahidul Islam. 'Advantages and limitations of genetic algorithms for clustering records'. In: *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*. IEEE. 2016, pp. 2478–2483.

[14] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies and Mark B Sandler. 'A tutorial on onset detection in music signals'. In: *IEEE Transactions on speech and audio processing* 13.5 (2005), pp. 1035–1047.

[15] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman and Paul Lamere. 'The million song dataset'. In: (2011). `http://millionsongdataset.com/`.

[16] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[17] Maciej Blaszke and Bożena Kostek. 'Musical instrument identification using deep learning approach'. In: *Sensors* 22.8 (2022), p. 3033.

[18] Holger Boche and Sawomir Stanczak. 'The Kullback–Leibler divergence and nonnegative matrices'. In: *IEEE transactions on information theory* 52.12 (2006), pp. 5539–5545.

[19] Sebastian Böck, Matthew EP Davies and Peter Knees. 'Multi-Task Learning of Tempo and Beat: Learning One to Improve the Other.' In: *ISMIR*. 2019, pp. 486–493.

[20] Dmitry Bogdanov, Minz Won, Philip Tovstogan, Alastair Porter and Xavier Serra. 'The MTG-Jamendo dataset for automatic music tagging'. In: ICML. 2019.

[21] Stuart Borthwick and Ron Moy. *Popular music genres: An introduction*. Routledge, 2020.

[22] Leo Breiman. 'Bagging predictors'. In: *Machine learning* 24.2 (1996), pp. 123–140.

[23] M Broniatowski. 'Estimation of the Kullback-Leibler divergence'. In: *Mathematical Methods of Statistics* 12.4 (2003), pp. 391–409.

[24] Stephen Brooks. 'Markov chain Monte Carlo method and its application'. In: *Journal of the royal statistical society: series D (the Statistician)* 47.1 (1998), pp. 69–100.

[25] Christopher JC Burges, John C Platt and Soumya Jana. 'Extracting noise-robust features from audio data'. In: *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. IEEE. 2002, pp. I–1021.

[26] Harshada Burute and PB Mane. 'Separation of singing voice from music background'. In: *International Journal of Computer Applications* 129.4 (2015), pp. 22–26.

[27] Erion Çano and Maurizio Morisio. 'Moodylyrics: A sentiment annotated lyrics dataset'. In: *Proceedings of the 2017 international conference on intelligent systems, metaheuristics & swarm intelligence*. 2017, pp. 118–124.

[28] Estefanía Cano, Fernando Mora-Ángel, Gustavo A López Gil, José R Zapata, Antonio Escamilla, Juan F Alzate and Moisés Betancur. 'Sesquialtera in the colombian bambuco: Perception and estimation of beat and meter'. In: *Proc. Int. Soc. Music Inf. Retrieval Conf.* 2020, pp. 409–415.

[29] Zehra Cataltepe, Yusuf Yaslan and Abdullah Sonmez. 'Music genre classification using MIDI and audio features'. In: *EURASIP Journal on Advances in Signal Processing* 2007 (2007), pp. 1–8.

[30] Tak-Shing Chan, Tzu-Chun Yeh, Zhe-Cheng Fan, Hung-Wei Chen, Li Su, Yi-Hsuan Yang and Roger Jang. 'Vocal activity informed singing voice separation with the iKala dataset'. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2015, pp. 718–722.

[31] Corinna Cortes and Vladimir Vapnik. 'Support-vector networks'. In: *Machine learning* 20.3 (1995), pp. 273–297.

[32] Eugene Coyle and Mikel Gainza. 'Time signature detection by using a multi-resolution audio similarity matrix'. In: *Audio Engineering Society Convention 122*. Audio Engineering Society. 2007.

[33] Trung-Thanh Dang and Kiyoaki Shirai. 'Machine learning approaches for mood classification of songs toward music search engine'. In: *2009 International Conference on Knowledge and Systems Engineering*. IEEE. 2009, pp. 144–149.

[34] Sneha Das, Tom Bäckström et al. 'Postfiltering Using Log-Magnitude Spectrum for Speech and Audio Coding.' In: *Interspeech*. 2018, pp. 3543–3547.

[35] Matthew EP Davies and Mark D Plumbley. 'Context-dependent beat tracking of musical audio'. In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.3 (2007), pp. 1009–1020.

[36] W Bas De Haas and Anja Volk. 'Meter detection in symbolic music using inner metric analysis'. In: *International Society for Music Information Retrieval Conference*. 2016, p. 441.

[37] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst and Xavier Bresson. 'Fma: A dataset for music analysis'. In: *arXiv preprint arXiv:1612.01840* (2016). `https://arxiv.org/abs/1612.01840`.

[38] N. Degara, A. Pena, M. E. P. Davies and M. D. Plumbley. 'Note onset detection using rhythmic structure'. In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2010, pp. 5526–5529. ISBN: 2379–190X. DOI: `10.1109/ICASSP.2010.5495220`.

[39] Jeremiah D Deng, Christian Simmermacher and Stephen Cranefield. 'A study on feature analysis for musical instrument classification'. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 38.2 (2008), pp. 429–438.

[40] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei. 'Imagenet: A large-scale hierarchical image database'. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[41] Sander Dieleman, Philémon Brakel and Benjamin Schrauwen. 'Audio-based music classification with a pretrained convolutional network'. In: *12th International Society for Music Information Retrieval Conference (ISMIR-2011)*. University of Miami. 2011, pp. 669–674.

[42] Simon Dixon, Elias Pampalk and Gerhard Widmer. 'Classification of dance music by periodicity patterns'. In: (2003).

[43] Simon Durand and Slim Essid. 'Downbeat Detection with Conditional Random Fields and Deep Learned Features.' In: *ISMIR*. 2016, pp. 386–392.

[44] Issam El Naqa and Martin J Murphy. *What is machine learning?* Springer, 2015.

[45] Ahmet Elbir, Hilmi Bilal Çam, Mehmet Emre Iyican, Berkay Öztürk and Nizamettin Aydin. 'Music genre classification and recommendation by using machine learning techniques'. In: *2018 Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE. 2018, pp. 1–5.

[46] Dan Ellis. 'Chroma feature analysis and synthesis'. In: *Resources of laboratory for the recognition and organization of speech and Audio-LabROSA* 5 (2007).

[47] Daniel PW Ellis and Graham E Poliner. 'Identifyingcover songs' with chroma features and dynamic programming beat tracking'. In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*. Vol. 4. IEEE. 2007, pp. IV–1429.

[48] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck and Karen Simonyan. 'Neural audio synthesis of musical notes with wavenet autoencoders'. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1068–1077.

[49] Antti J Eronen and Anssi P Klapuri. 'Music Tempo Estimation With k-NN Regression'. In: *IEEE Transactions on Audio, Speech, and Language Processing* 18.1 (2009), pp. 50–57.

[50] Slim Essid, Gaël Richard and Bertrand David. 'Musical instrument recognition by pairwise classification strategies'. In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.4 (2006), pp. 1401–1412.

[51] Tao Feng. 'Deep learning for music genre classification'. In: *private document* (2014).

[52] Robert Fink. 'Goal-directed soul? Analyzing rhythmic teleology in African American popular music'. In: *Journal of the American Musicological Society* 64.1 (2011), pp. 179–238.

[53] Jerome H Friedman. 'Greedy function approximation: a gradient boosting machine'. In: *Annals of statistics* (2001), pp. 1189–1232.

[54] Magdalena Fuentes, Brian McFee, Hélène Crayencour, Slim Essid and Juan Bello. 'Analysis of common design choices in deep learning systems for downbeat tracking'. In: *The 19th International Society for Music Information Retrieval Conference*. 2018.

[55] Magdalena Fuentes, Brian Mcfee, Hélène C Crayencour, Slim Essid and Juan Pablo Bello. 'A music structure informed downbeat tracking system using skip-chain conditional random fields and deep learning'. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 481–485.

[56] Mikel Gainza. 'Automatic musical meter detection'. In: *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2009, pp. 329–332.

[57] Patrick Gampp. 'Evaluation of robust features for singing voice detection'. In: *M. Sc., Institute of Electronic Music and Acoustics, University of Music and Dramatic Arts Graz, Graz* (2010).

[58] Mudasir A Ganaie, Minghui Hu, AK Malik, M Tanveer and PN Suganthan. 'Ensemble deep learning: A review'. In: *Engineering Applications of Artificial Intelligence* 115 (2022), p. 105151.

[59] Aggelos Gkiokas, Vassilis Katsouros, George Carayannis and Themos Stajylakis. 'Music tempo estimation and beat tracking by applying source separation and metrical relations'. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2012, pp. 421–424.

[60] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura and Ryuichi Oka. 'RWC Music Database: Popular, Classical and Jazz Music Databases.' In: *Ismir*. Vol. 2. 2002, pp. 287–288.

[61] Fabien Gouyon and Perfecto Herrera. 'Determination of the meter of musical audio signals: Seeking recurrences in beat segment descriptors'. In: *Audio Engineering Society Convention 114*. Audio Engineering Society. 2003.

[62] Fabien Gouyon, François Pachet, Olivier Delerue et al. 'On the use of zero-crossing rate for an application of classification of percussive sounds'. In: *Proceedings of the COST G-6 conference on Digital Audio Effects (DAFX-00), Verona, Italy*. Vol. 5. 2000, p. 16.

[63] Justin Grimmer. 'An introduction to Bayesian inference via variational approximations'. In: *Political Analysis* 19.1 (2011), pp. 32–47.

[64] Daniel Grzywczak and Grzegorz Gwardys. 'Audio features in music information retrieval'. In: *Active Media Technology: 10th International Conference, AMT 2014, Warsaw, Poland, August 11-14, 2014. Proceedings 10*. Springer. 2014, pp. 187–199.

[65] W. Gui and Shao Xi. 'Onset detection using leared dictionary by K-SVD'. In: *2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA)*. 2014, pp. 406–409. DOI: 10.1109/WARTIA.2014.6976281.

[66] W. Gui and Shao Xi. 'Onset detection using leared dictionary by K-SVD'. In: *2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA)*. 2014, pp. 406–409. DOI: 10.1109/WARTIA.2014.6976281.

[67] Sankalp Gulati, Vishweshwara Rao and Preeti Rao. 'Meter detection from audio for Indian music'. In: *Speech, Sound and Music Processing: Embracing Research in India*. Springer, 2011, pp. 34–43.

[68] Stephen Hainsworth, Malcolm D Macleod et al. 'Onset detection in musical audio signals'. In: *ICMC*. Citeseer. 2003.

[69] Christopher Harte. 'Towards automatic extraction of harmony information from music signals'. PhD thesis. Queen Mary, University of London, 2010.

[70] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. 'Deep residual learning for image recognition'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[71] Serhat Hizlisoy, Serdar Yildirim and Zekeriya Tufekci. 'Music emotion recognition using convolutional long short term memory deep neural networks'. In: *Engineering Science and Technology, an International Journal* 24.3 (2021), pp. 760–767.

[72] André Holzapfel and Yannis Stylianou. 'Rhythmic Similarity in Traditional Turkish Music'. In: *ISMIR - International Conference on Music Information Retrieval* (2009), pp. 99–104. URL: `http://ismir2009.ismir.net/proceedings/PS1-8.pdfhttp://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-193761`.

[73] Md Afzal Hossan, Sheeraz Memon and Mark A Gregory. 'A novel approach for MFCC feature extraction'. In: *2010 4th International Conference on Signal Processing and Communication Systems*. IEEE. 2010, pp. 1–5.

[74] Yuanbo Hou, Frank K Soong, Jian Luan and Shengchen Li. 'Transfer learning for improving singing-voice detection in polyphonic instrumental music'. In: *arXiv preprint arXiv:2008.04658* (2020).

[75] C. Hsu, D. Wang and J. R. Jang. 'A trend estimation algorithm for singing pitch detection in musical recordings'. In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2011, pp. 393–396. ISBN: 2379–190X. DOI: `10.1109/ICASSP.2011.5946423`.

[76] Xiao Hu and J Stephen Downie. 'Improving mood classification in music digital libraries by combining lyrics and audio'. In: *Proceedings of the 10th annual joint conference on Digital libraries*. 2010, pp. 159–168.

[77] Xiao Hu, J Stephen Downie and Andreas F Ehmann. 'Lyric text mining in music mood classification'. In: *American music* 183.5,049 (2009), pp. 2–209.

[78] Yukara Ikemiya, Kazuyoshi Yoshii and Katsutoshi Itoyama. 'Singing voice analysis and editing based on mutually dependent F0 estimation and source separation'. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2015, pp. 574–578.

[79] Jaekwon Im, Eunjin Choi and Wootaek Lim. 'DATA AUGMENTATION FOR SINGING VOICE SEPARATION USING MUSICAL INSTRUMENT TRANSFER AND RESYNTHESIS'. In: ().

[80] MK Jiawei Han and Jian Pei. *Data mining: concepts and techniques: concepts and techniques*. 2011.

[81] Michael I Jordan and Tom M Mitchell. 'Machine learning: Trends, perspectives, and prospects'. In: *Science* 349.6245 (2015), pp. 255–260.

[82] Prafulla Kalapatapu, Srihita Goli, Prasanna Arthum and Aruna Malapati. 'A study on feature selection and classification techniques of Indian music'. In: *Procedia Computer Science* 98 (2016), pp. 125–131.

[83] Gurpreet Kaur, Mohit Srivastava and Amod Kumar. 'Genetic algorithm for combined speaker and speech recognition using deep neural networks'. In: *Journal of Telecommunications and Information Technology* (2018).

[84] JungHyun Kim, Seungjae Lee, SungMin Kim and Won Young Yoo. 'Music mood classification model based on arousal-valence values'. In: *13th International Conference on Advanced Communication Technology (ICACT2011)*. IEEE. 2011, pp. 292–295.

[85] Anssi Klapuri et al. 'Musical meter estimation and music transcription'. In: *Cambridge Music Processing Colloquium*. Citeseer. 2003, pp. 40–45.

[86] Anssi P Klapuri, Antti J Eronen and Jaakko T Astola. 'Analysis of the meter of acoustic musical signals'. In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.1 (2005), pp. 342–355.

[87] Bozena Kostek. 'Musical instrument classification and duet analysis employing music information retrieval techniques'. In: *Proceedings of the IEEE* 92.4 (2004), pp. 712–729.

[88] Daniel Kostrzewa, Piotr Kaminski and Robert Brzeski. 'Music Genre Classification: Looking for the Perfect Network'. In: *International Conference on Computational Science*. Springer. 2021, pp. 55–67.

[89] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas et al. 'Handling imbalanced datasets: A review'. In: *GESTS International Transactions on Computer Science and Engineering* 30.1 (2006), pp. 25–36.

[90] Florian Krebs, Sebastian Böck and Gerhard Widmer. 'Rhythmic Pattern Modeling for Beat and Downbeat Tracking in Musical Audio.' In: *Ismir*. Citeseer. 2013, pp. 227–232.

[91] Olivier Lartillot and Petri Toiviainen. 'A Matlab toolbox for musical feature extraction from audio'. In: *International conference on digital audio effects*. Vol. 237. Bordeaux. 2007, p. 244.

[92] Edith Law, Kris West, Michael I Mandel, Mert Bay and J Stephen Downie. 'Evaluation of algorithms using games: The case of music tagging.' In: *ISMIR*. `https://mirg.city.ac.uk/codeapps/the-magnatagatune-dataset`. 2009, pp. 387–392.

[93] Alvin Lazaro, Riyanarto Sarno, R Johanes Andre and Muhammad Nezar Mahardika. 'Music tempo classification using audio spectrum centroid, audio spectrum flatness, and audio spectrum spread based on MPEG-7 audio features'. In: *2017 3rd international conference on science in information technology (ICSITech)*. IEEE. 2017, pp. 41–46.

[94] Phu Ngoc Le, Eliathamby Ambikairajah, Julien Epps, Vidhyasaharan Sethu and Eric HC Choi. 'Investigation of spectral centroid features for cognitive load classification'. In: *Speech Communication* 53.4 (2011), pp. 540–551.

[95] Simon Leglaive, Romain Hennequin and Roland Badeau. 'Singing voice detection with deep recurrent neural networks'. In: *2015 IEEE International conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2015, pp. 121–125.

[96] Nathan Lenssen. 'Applications of fourier analysis to audio signal processing: An investigation of chord detection algorithms'. In: (2013).

[97] Tom LH Li and Antoni B Chan. 'Genre classification and the invariance of MFCC features to key and tempo'. In: *International Conference on MultiMedia Modeling*. Springer. 2011, pp. 317–327.

[98] Victoria López, Alberto Fernández and Francisco Herrera. 'On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed'. In: *Information Sciences* 257 (2014), pp. 1–13.

[99] Saranga Kingkor Mahanta, Abdullah Faiz Ur Rahman Khilji and Partha Pakray. 'Deep neural network for musical instrument recognition using MFCCs'. In: *Computación y Sistemas* 25.2 (2021), pp. 351–360.

[100] Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

[101] Ricard Marxer and Jordi Janer. 'Modelling and separation of singing voice breathiness in polyphonic mixtures'. In: *Proc. 16th Int. Conf. Digital Audio Effects*. 2013, pp. 2–5.

[102] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg and Oriol Nieto. 'librosa: Audio and music signal analysis in python'. In: *Proceedings of the 14th python in science conference*. Vol. 8. Citeseer. 2015, pp. 18–25.

[103] Cory McKay and Ichiro Fujinaga. 'Automatic Genre Classification Using Large High-Level Musical Feature Sets.' In: *ISMIR*. Vol. 2004. 2004. Citeseer. 2004, pp. 525–530.

[104] Martin McKinney and Jeroen Breebaart. 'Features for audio and music classification'. In: (2003).

[105] Martin F McKinney, Dirk Moelants, Matthew EP Davies and Anssi Klapuri. 'Evaluation of audio beat tracking and music tempo extraction algorithms'. In: *Journal of New Music Research* 36.1 (2007), pp. 1–16.

[106] Andrew McLeod and Mark Steedman. 'Meter Detection and Alignment of MIDI Performance.' In: *ISMIR*. 2018, pp. 113–119.

[107] Andrew McLeod and Mark Steedman. 'Meter Detection From Music Data'. In: *DMRN+ 11: Digital Music Research Network One-day Workshop 2016*. 2016.

[108] Tom M Mitchell. 'Machine learning and data mining'. In: *Communications of the ACM* 42.11 (1999), pp. 30–36.

[109] Ramy Monir, Daniel Kostrzewa and Dariusz Mrozek. 'Singing voice detection: a survey'. In: *Entropy* 24.1 (2022), p. 114.

[110] M. Mounir, P. Karsmakers and T. v. Waterschoot. 'Annotations Time Shift: A Key Parameter in Evaluating Musical Note Onset Detection Algorithms'. In: *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. 2019, pp. 21–25. ISBN: 1947–1629. DOI: 10.1109/WASPAA.2019.8937251.

[111] Meinard Muller, Daniel PW Ellis, Anssi Klapuri and Gaël Richard. 'Signal processing for music analysis'. In: *IEEE Journal of selected topics in signal processing* 5.6 (2011), pp. 1088–1110.

[112] Meinard Muller, Frank Kurth and Michael Clausen. 'Chroma-based statistical audio features for audio matching'. In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.* IEEE. 2005, pp. 275–278.

[113] E. Nakamura, E. Benetos, K. Yoshii and S. Dixon. 'Towards Complete Polyphonic Music Transcription: Integrating Multi-Pitch Detection and Rhythm Quantization'. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, pp. 101–105. ISBN: 2379–190X. DOI: 10.1109/ICASSP.2018.8461914.

[114] Juhan Nam, Keunwoo Choi, Jongpil Lee, Szu-Yu Chou and Yi-Hsuan Yang. 'Deep learning for audio-based music classification and tagging: Teaching computers to distinguish rock from bach'. In: *IEEE signal processing magazine* 36.1 (2018), pp. 41–51.

[115] Alexey Natekin and Alois Knoll. 'Gradient boosting machines, a tutorial'. In: *Frontiers in neurorobotics* 7 (2013), p. 21.

[116] Tin Lay Nwe, Arun Shenoy and Ye Wang. 'Singing voice detection in popular music'. In: *Proceedings of the 12th annual ACM international conference on Multimedia*. 2004, pp. 324–327.

[117] Hui-Lee Ooi, Siew-Cheok Ng and Einly Lim. 'Ano detection with k-nearest neighbor using minkowski distance'. In: *International Journal of Signal Processing Systems* 1.2 (2013), pp. 208–211.

[118] Sergio Oramas, Francesco Barbieri, Oriol Nieto Caballero and Xavier Serra. 'Multimodal deep learning for music genre classification'. In: *Transactions of the International Society for Music Information Retrieval. 2018; 1 (1): 4-21.* (2018).

[119] Braja Gopal Patra, Dipankar Das and Sivaji Bandyopadhyay. 'Automatic music mood classification of Hindi songs'. In: *Proceedings of the 3rd Workshop on Sentiment Analysis where AI meets Psychology*. 2013, pp. 24–28.

[120] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg et al. 'Scikit-learn: Machine learning in Python'. In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[121] Nikki Pelchat and Craig M Gelowitz. 'Neural network music genre classification'. In: *Canadian Journal of Electrical and Computer Engineering* 43.3 (2020), pp. 170–173.

[122] Aggelos Pikrakis, Iasonas Antonopoulos and Sergios Theodoridis. 'Music meter and tempo tracking from raw polyphonic audio.' In: *ISMIR*. 2004.

[123] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik Schmidt, Andreas Ehmann and Xavier Serra. 'End-to-end learning for music audio tagging at scale'. In: *arXiv preprint arXiv:1711.02520* (2017).

[124] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush and Andrey Gulin. 'CatBoost: unbiased boosting with categorical features'. In: *Advances in neural information processing systems* 31 (2018).

[125] Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-Yiin Chang and Tara Sainath. 'Deep learning for audio signal processing'. In: *IEEE Journal of Selected Topics in Signal Processing* 13.2 (2019), pp. 206–219.

[126] Konstantinos Pyrovolakis, Paraskevi Tzouveli and Giorgos Stamou. 'Multi-modal song mood detection with deep learning'. In: *Sensors* 22.3 (2022), p. 1065.

[127] Rajeev Rajan and Anu Alphonsa Raju. 'Deep neural network based poetic meter classification using musical texture feature fusion'. In: *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE. 2019, pp. 1–5.

[128] Lise Regnier and Geoffroy Peeters. 'Singing voice detection in music tracks using direct voice vibrato detection'. In: *2009 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2009, pp. 1685–1688.

[129] Martın Rocamora and Perfecto Herrera. 'Comparing audio descriptors for singing voice detection in music audio files'. In: *Brazilian symposium on computer music, 11th. san pablo, brazil*. Vol. 26. 2007, p. 27.

[130] Carles Roig, Lorenzo J Tardón, Isabel Barbancho and Ana M Barbancho. 'Automatic melody composition based on a probabilistic model of music style and harmonic rules'. In: *Knowledge-Based Systems* 71 (2014), pp. 419–434.

[131] Joseph Rothstein. *MIDI: A comprehensive introduction*. Vol. 7. AR Editions, Inc., 1992.

[132] Nicolas Scaringella, Giorgio Zoia and Daniel Mlynek. 'Automatic genre classification of music content: a survey'. In: *IEEE Signal Processing Magazine* 23.2 (2006), pp. 133–141.

[133] Markus Schedl, Nicola Orio, Cynthia CS Liem and Geoffroy Peeters. 'A professionally annotated and enriched multimodal data set on popular music'. In: *Proceedings of the 4th ACM Multimedia Systems Conference*. `http://www.cp.jku.at/datasets/musiclef/index.html`. 2013, pp. 78–83.

[134] Jan Schlüter and Thomas Grill. 'Exploring data augmentation for improved singing voice detection with neural networks.' In: *ISMIR*. 2015, pp. 121–126.

[135] Hans-Henning Schulze, Andreas Cordes and Dirk Vorberg. 'Keeping synchrony while tempo changes: Accelerando and ritardando'. In: *Music Perception* 22.3 (2005), pp. 461–477.

[136] Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse and Amri Napolitano. 'RUSBoost: Improving classification performance when training data is skewed'. In: *2008 19th International Conference on Pattern Recognition*. IEEE. 2008, pp. 1–4.

[137] Ozan Sener and Vladlen Koltun. 'Multi-task learning as multi-objective optimization'. In: *arXiv preprint arXiv:1810.04650* (2018).

[138] Ayush Shakya, Bijay Gurung, Mahendra Singh Thapa, Mehang Rai and Basanta Joshi. 'Music classification based on genre and mood'. In: *Computational Intelligence, Communications, and Business Analytics: First International Conference, CICBA 2017, Kolkata, India, March 24–25, 2017, Revised Selected Papers, Part II*. Springer. 2017, pp. 168–183.

[139] Yogesh Singh, Pradeep Kumar Bhatia and Omprakash Sangwan. 'A review of studies on machine learning techniques'. In: *International Journal of Computer Science and Security* 1.1 (2007), pp. 70–84.

[140] Janto Skowronek, Martin F McKinney and Steven Van De Par. 'Ground truth for automatic music mood classification.' In: *ISMIR*. 2006, pp. 395–396.

[141] Aized Amin Soofi and Arshad Awan. 'Classification techniques in machine learning: applications and issues'. In: *Journal of Basic & Applied Sciences* 13 (2017), pp. 459–465.

[142] Mila Soares de Oliveira de Souza, Pedro Nuno de Souza Moura and Jean-Pierre Briot. 'Music Tempo Estimation via Neural Networks–A Comparative Analysis'. In: *arXiv preprint arXiv:2107.09208* (2021).

[143] Ajay Srinivasamurthy, Andre Holzapfel, Ali Taylan Cemgil and Xavier Serra. 'Particle filters for efficient meter tracking with dynamic bayesian networks'. In: *Müller M, Wiering F, editors. ISMIR 2015. 16th International Society for Music Information Retrieval Conference; 2015 Oct 26-30; Málaga, Spain. Canada: ISMIR; 2015.* International Society for Music Information Retrieval (ISMIR). 2015.

[144] Ajay Srinivasamurthy, Gregoire Tronel, Sidharth Subramanian and Parag Chordia. 'A BEAT TRACKING APPROACH TO COMPLETE DESCRIPTION OF RHYTHM IN INDIAN CLASSICAL MUSIC'. In: *CompMusic Workshop*. Georgia Tech Center for Music Technology, Atlanta, USA, 2012, pp. 72–78. URL: https://www.academia.edu/download/32402663/Proceedings-2nd-CompMusic-Workshop-2012_0.pdf#page=76..

[145] Aditya Srivastava, Sonia Saini and Deepa Gupta. 'Comparison of various machine learning techniques and its uses in different fields'. In: *2019 3rd International conference on electronics, communication and aerospace technology (ICECA)*. IEEE. 2019, pp. 81–86.

[146] Carlo Strapparava, Alessandro Valitutti et al. 'Wordnet affect: an affective extension of wordnet.' In: *Lrec*. Vol. 4. 1083-1086. Lisbon, Portugal. 2004, p. 40.

[147] Bob L Sturm. 'An analysis of the GTZAN music genre dataset'. In: *Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies*. 2012, pp. 7–12.

[148] Derek Tingle, Youngmoo E Kim and Douglas Turnbull. 'Exploring automatic music annotation with" acoustically-objective" tags'. In: *Proceedings of the international conference on Multimedia information retrieval*. http://calab1.ucsd.edu/~datasets/. 2010, pp. 55–62.

[149] Douglas Turnbull, Luke Barrington, David Torres and Gert Lanckriet. 'Exploring the semantic annotation and retrieval of sound'. In: *CAL Technical Report CAL-2007-01, San Diego* (2007). https://www.ee.columbia.edu/~dpwe/research/musicsim/uspop2002.html.

[150] Douglas Turnbull, Luke Barrington, David Torres and Gert Lanckriet. 'Semantic annotation and retrieval of music and sound effects'. In: *IEEE Transactions on Audio, Speech, and Language Processing* 16.2 (2008). `http://slam.iis.sinica.edu.tw/demo/CAL500exp`, pp. 467–476.

[151] George Tzanetakis. 'Marsyas-0.2: a case study in implementing music information retrieval systems'. In: *Intelligent Music Information Systems: Tools and Methodologies*. IGI Global, 2008, pp. 31–49.

[152] George Tzanetakis and Perry Cook. 'Marsyas: A framework for audio analysis'. In: *Organised sound* 4.3 (2000), pp. 169–175.

[153] George Tzanetakis and Perry Cook. 'Musical genre classification of audio signals'. In: *IEEE Transactions on speech and audio processing* 10.5 (2002), pp. 293–302.

[154] Christian Uhle and Juergen Herre. 'Estimation of tempo, micro time and time signature from percussive music'. In: *Proc. Int. Conference on Digital Audio Effects (DAFx)*. Citeseer. 2003.

[155] Matthias Varewyck, Jean-Pierre Martens and Marc Leman. 'Musical meter classification with beat synchronous acoustic features, DFT-based metrical features and support vector machines'. In: *Journal of New Music Research* 42.3 (2013), pp. 267–282.

[156] Adrian E Villanueva-Luna, Alberto Jaramillo-Nuñez, Daniel Sanchez-Lucero, Carlos M Ortiz-Lima, J Gabriel Aguilar-Soto, Aaron Flores-Gil and Manuel May-Alarcon. *De-noising audio signals using MATLAB wavelets toolbox*. IntechOpen, 2011.

[157] Larry Wasserman. 'Bayesian model selection and model averaging'. In: *Journal of mathematical psychology* 44.1 (2000), pp. 92–107.

[158] Nic M Weststrate, Susan Bluck and Judith Glück. 'Wisdom of the Crowd'. In: *The Cambridge handbook of wisdom* (2019), pp. 97–121.

[159] Fu-Hai Frank Wu and Jyh-Shing Roger Jang. 'A supervised learning method for tempo estimation of musical audio'. In: *22nd Mediterranean Conference on Control and Automation*. IEEE. 2014, pp. 599–604.

[160] Xianjiao Wu, Qiang Ye, Hong Hong and Yijun Li. 'Stock selection model based on machine learning with wisdom of experts and crowds'. In: *IEEE Intelligent Systems* 35.2 (2020), pp. 54–64.

[161] Changsheng Xu, Namunu C Maddage, Xi Shao, Fang Cao and Qi Tian. 'Musical genre classification using support vector machines'. In: *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03)*. Vol. 5. IEEE. 2003, pp. V–429.

[162] Changsheng Xu, Namunu Chinthaka Maddage and Xi Shao. 'Automatic music classification and summarization'. In: *IEEE transactions on speech and audio processing* 13.3 (2005), pp. 441–450.

[163] Shingchern D You, Chien-Hung Liu and Woei-Kae Chen. 'Comparative study of singing voice detection based on deep neural networks and ensemble learning'. In: *Human-centric Computing and Information Sciences* 8 (2018), pp. 1–18.

[164] Yunkai Yu, Yuyang You, Zhihong Yang, Guozheng Liu, Peiyao Li, Zhicheng Yang and Wenjing Shan. 'Spectral Roll-off Points Variations: Exploring Useful Information in Feature Maps by Its Variations'. In: *arXiv preprint arXiv:2102.00369* (2021).

[165] Maider Zamalloa, Germán Bordel, Luis Javier Rodríguez and Mikel Peñagarikano. 'Feature selection based on genetic algorithms for speaker recognition'. In: *2006 IEEE Odyssey-The Speaker and Language Recognition Workshop*. IEEE. 2006, pp. 1–8.

[166] Henry Zelenak and Shahin Mehdipour Ataee. 'Identifying Periodic Signal Patterns in Audio Streams'. In: *2022 IEEE Western New York Image and Signal Processing Workshop (WNYISPW)*. IEEE. 2022, pp. 1–4.

[167] Yu Zhang and Qiang Yang. 'A survey on multi-task learning'. In: *arXiv preprint arXiv:1707.08114* (2017).

Appendices

List of Figures

List of Tables

Technical Documentation

Algorithm 2 build_cnn_model(input_shape)

```
model ← Sequential()

model.add(Conv2D(32, (3, 3), activation = 'relu', input_shape = input_shape, padding = 'same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((4, 6), padding = 'same'))

model.add(Conv2D(32, (3, 3), activation = 'relu', padding = 'same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((4, 6), padding = 'same'))

model.add(Conv2D(64, (3, 3), activation = 'relu', padding = 'same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((4, 6), padding = 'same'))

model.add(Flatten())
model.add(Dense(32, activation = 'relu'))
model.add(Dropout(0.3))

model.add(Dense(16, activation = 'relu'))
model.add(Dropout(0.2))

model.add(Dense(8, activation = 'relu'))
model.add(Dropout(0.2))

model.add(Dense(4, activation = 'softmax'))

return model
```

All Python codes used for data extraction and analysis were written in Jupyter Notebook and published through the GitHub repository. There is no restriction to accessing this public repository of the source code.

Algorithm 3 build_crnn_model(input_shape)

```
model ← Sequential()
model.add(Conv2D(32, (3, 3), activation = 'relu', input_shape = input_shape, padding = 'same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((4, 6), padding = 'same'))

model.add(Conv2D(32, (3, 3), activation = 'relu', padding = 'same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((4, 6), padding = 'same'))

model.add(Conv2D(64, (3, 3), activation = 'relu', padding = 'same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((4, 6), padding = 'same'))

model.add(Reshape((−1, 32)))
model.add(Dense(64, activation = 'relu'))
model.add(LSTM(64, dropout = 0.2, return_sequences = True))
model.add(LSTM(64, dropout = 0.2))

model.add(Flatten())
model.add(Dense(16, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(4, activation = 'relu'))
model.add(Dense(4, activation = 'softmax'))
return model
```

Algorithm 4 build_resnet18_model(num_classes)

```
model ← Resnet18Model()

model.conv1 ← Conv2D(1, 3, kernel_size = 1)
model ← resnet18(weights=None)
model.fc ← Sequential()
model.fc.add(Linear(512, 128))
model.fc.add(ReLU())
model.fc.add(Linear(128, 32))
model.fc.add(ReLU())
model.fc.add(Linear(32, 8))
model.fc.add(ReLU())
model.fc.add(Linear(8, num_classes))

return model
```

Algorithm 5 build_resnet18_rnn_model(num_classes)

```
model ← Resnet18RnnModel()

model.conv1 ← Conv2D(1, 3, kernel_size = 1)
base_model ← Resnet18Model()
base_model.fc ← Identity()
model.mfcc_fc ← Linear(512, 512)
model.lstm ← LSTM(512, hidden_size = 128, num_layers = 2, batch_first = True)

model.base_model.fc ← Sequential()
model.base_model.fc.add(Linear(128, 32))
model.base_model.fc.add(ReLU())
model.base_model.fc.add(Linear(32, 8))
model.base_model.fc.add(ReLU())
model.base_model.fc.add(Linear(8, num_classes))

return model
```

Algorithm 6 optimize_model_weights_with_ga

```
varbound ← np.array([[0, 1]] ∗ 4)

algorithm_param ← {}
algorithm_param['max_num_iteration'] ← 100
algorithm_param['population_size'] ← 100
algorithm_param['mutation_probability'] ← 0.6
algorithm_param['elit_ratio'] ← 0.01
algorithm_param['crossover_probability'] ← 0.2
algorithm_param['parents_portion'] ← 0.5
algorithm_param['crossover_type'] ← 'uniform'
algorithm_param['max_iteration_without_improv'] ← None

model ← ga(function = evaluate_weights,
           dimension = 4,
           variable_type = 'real',
           variable_boundaries = varbound,
           algorithm_parameters = algorithm_param,
           function_timeout = 1000200)

model.run()

model_names ← ['cnn','crnn','resnet','resnet_lstm']

optimized_weights ← model.output_dict['variable']
```

Algorithm 7 majorityVoting

```
function majorityVoting(df, predictions_columns)
    df['majority'] ← df[predictions_columns].mode(axis=1)[0]
    df['majority'] ← df['majority'].astype(int)
    return df
```

Algorithm 8 rankedVoting

```
function rankedVoting(df, softmax_columns)
    highest_ranks ← []
    for i ← 0 to len(df) do
        ranks ← []
        for col in softmax_columns do
            softmax_output ← ast.literal_eval(columns[i])
            ranked_softmax ← rankdata(softmax_output, method='max')
            rank ← ranked_softmax - 1
            ranks.append(rank)
        rank_sums ← np.sum(ranks, axis=0)
        highest_rank ← np.argmax(rank_sums)
        highest_ranks.append(highest_rank)
    df['ranked'] ← highest_ranks
    return df
```

Algorithm 9 summedVoting

```
function summedVoting(df, softmax_columns)
    if not softmax_columns then
        raise ValueError("No softmax columns found in the DataFrame.")
    ensemble_predictions ← []
    for i ← 0 to len(df) do
        sums ← []
        for col in softmax_columns do
            softmax_values ← eval(columns[i])
            sums.append(softmax_values)
        summed_probabilities ← sum(sums, axis=0)
        max_probability ← argmax(summed_probabilities)
        ensemble_predictions.append(max_probability)
    return ensemble_predictions
```

Algorithm 10 weightedVoting

```
function WEIGHTEDVOTING(df, softmax_columns, weights)
    if not softmax_columns then
        raise ValueError("No softmax columns found in the DataFrame.")
    ensemble_predictions ← []
    for i ← 0 to len(df) do
        probabilities ← []
        for col in softmax_columns do
            softmax_values ← eval(columns[i])
            probabilities.append(softmax_values)
        weighted_sum ← zeros_like(probabilities[0])
        for prob, weight in zip(probabilities, weights) do
            weighted_sum+ = array(prob) * weight
        ensemble_prediction ← argmax(weighted_sum)
        ensemble_predictions.append(ensemble_prediction)
    return ensemble_predictions
```

Algorithm 11 stacking

```
function STACKING(df, test_df, val_prediction_columns, test_prediction_columns,
meta_model)
    predictions ← df[val_prediction_columns]
    meta_model.fit(predictions, df['true_label'])
    test_predictions ← test_df[test_prediction_columns]
    final_predictions ← meta_model.predict(test_predictions)
    return final_predictions
```

Algorithm 12 bayesian_model_averaging

```
function BAYESIAN_MODEL_AVERAGING(df, softmax_columns)
    ensemble_predictions ← []
    for i ← 0 to len(df) do
        get the softmax_arrays
        posterior_probabilities ← max(softmax_arrays, axis=1)
        weights ← posterior_probabilities / sum(posterior_probabilities)
        weighted_sum ← zeros_like(softmax_arrays[0])
        for prob, weight in zip(softmax_arrays, weights) do
            weighted_sum+ = array(prob) * weight
        ensemble_prediction ← argmax(weighted_sum)
        ensemble_predictions.append(ensemble_prediction)
    return ensemble_predictions
```

Algorithm 13 softVoting

```
1: function SOFTVOTING(df, softmax_columns)
2:     if not softmax_columns then
3:         raise ValueError("No softmax columns found in the DataFrame.")
4:     soft_votes ← []
5:     for i ← 0 to len(df) do
6:         softmax_outputs ← []
7:         for col in softmax_columns do
8:             softmax_output ← eval(columns[i])
9:             softmax_outputs.append(softmax_output)
10:        avg_probabilities ← mean(softmax_outputs, axis=0)
11:        ensemble_prediction ← argmax(avg_probabilities)
12:        soft_votes.append(ensemble_prediction)
13:    return ensemble_prediction
```