NATIONAL TECHNICAL UNIVERSITY OF ATHENS
School of Electrical Engineering and Computer Science

EVALUATION OF MACHINE LEARNING ALGORITHMS FOR MUSICAL KEY DETECTION

By
Loukas Pastras

A thesis submitted to the National Technical University of Athens in partial fulfillment of the requirements for the degree of electrical engineering and computer science

Athens, Greece
June 2024

EVALUATION OF MACHINE LEARNING ALGORITHMS FOR MUSICAL KEY DETECTION

APPROVED BY:

_____,
Georgios Stamou

_____
Athanasios Voulodimos

_____
Spyridon Kantarelis

Artificial Intelligence and Learning Systems Laboratory

Acknowledgments

I want to thank...

Contents

List of Tables

List of Figures

Abstract

This thesis presents a comprehensive evaluation of various machine learning algorithms for the task of musical key detection, focusing on the use of Pitch Class Profiles (PCPs) to represent the note distribution of the input musical pieces. Our research bridges the gap between traditional rule-based approaches and contemporary machine learning techniques, utilizing three diverse datasets—Music21 Corpus, Lakh MIDI Dataset, and GiantSteps Dataset—to ensure robust and generalizable findings.

Our results demonstrate that certain machine learning algorithms outperform rule-based methods when using the same input representation. Notably, our novel architecture shows significant improvement in the performance of the Krumhansl algorithm while maintaining its operating principle. For instance, our custom model achieved an accuracy of 80.28% on the Music21 dataset, compared to the Krumhansl algorithm's 70.25%, highlighting the potential of machine learning in enhancing key detection accuracy.

However, we also found that PCPs lack essential rhythmic information necessary for confident key detection in some cases. To address this, we suggest that future research should incorporate temporal aspects into the representations used for key detection. By including features that capture rhythmic dynamics, the performance and robustness of key detection algorithms could be further improved, making them more effective across a wider range of musical styles and contexts.

This work contributes to the field of Music Information Retrieval (MIR) by providing valuable insights into the application of machine learning for musical analysis and proposing directions for future research. The findings set the stage for further exploration and refinement of key detection techniques, ultimately aiming to advance the capabilities of MIR systems.

Chapter 1

Introduction

1.1 Motivation

1.1.1 The Intriguing Nature of the Problem

Our species ability to use a structure of symbols, including words, gestures and signs to convey information and socially interact has constituted something more than an evolutionary advantage. Language is an integral part of our identity as individuals as well as as a whole. Maybe it is from the same biological mechanisms that allow us to possess that skill, that our love for music arises. But music has a much more ambiguous connection to tangible meaning than words have, and so, in our effort to intentionally arrange the various sounds we can produce in order to fully utilize the power of this medium, we are forced to carefully examine, categorize and understand our response to those sounds. The fruits of this study constitute what we call "music theory". Even though current music theory is a well structured system of knowledge, the universe of sound it tries to describe is sometimes too vast and diverse to subject to the former. This is exactly the reason why the task of automating the retrieval of information, relevant to our liking of music is a complicated, yet exciting undertaking.

1.1.2 The Importance of Key Detection as an MIR task

Key detection is a fundamental task in Music Information Retrieval (MIR) because it serves as the basis for numerous other music analysis and processing tasks. Understanding the key of a piece of music allows for more accurate music transcription, chord recognition, and harmony analysis, which are crucial for music theory studies and educational tools. Additionally, key detection facilitates music recommendation systems, enabling more personalized and contextually relevant suggestions by understanding the tonal characteristics of a user's preferences. In music production and remixing, key detection ensures harmonic compatibility between different tracks, enhancing the creative process. Furthermore, it aids in the development of sophisticated music search engines and databases, where users can search and categorize music based on key. Overall, accurate key detection enriches the analysis, understanding, and enjoyment of music, making it an indispensable component of modern MIR systems.

1.1.3 The Potential of a Machine Learning Solution

Utilizing machine learning algorithms for key detection in music is an approach brimming with potential for several compelling reasons. First and foremost, machine learning excels at identifying complex patterns and relationships within data, even when those patterns are

intricate or exist in high-dimensional spaces, such as the nuanced connections between notes, chords, and keys in music. Unlike traditional rule-based methods, machine learning models can be trained on extensive datasets encompassing a wide variety of musical styles and genres, allowing them to learn and generalize effectively. This means that machine learning models can often surpass hand-crafted heuristics and rules that may be biased towards specific musical conventions.

Moreover, advanced machine learning techniques, particularly deep learning models like Convolutional Neural Networks (CNNs) and Transformers, are exceptionally skilled at processing and interpreting various forms of musical data, such as spectrograms, chromograms, and Pitch Class Profiles (PCPs). These models can automatically extract and highlight relevant features, minimizing the need for extensive manual feature engineering. Additionally, machine learning models can be continually refined and improved as more data becomes available, ensuring they remain accurate and effective as musical trends and styles evolve.

Another significant advantage is the flexibility and scalability of machine learning approaches. These models can be adapted to incorporate different types of inputs and are capable of handling real-time key detection in dynamic environments, making them ideal for applications in live music performance, music production, and interactive music systems. Overall, the combination of sophisticated pattern recognition, adaptability, and scalability makes machine learning algorithms a powerful tool for achieving high accuracy and robustness in musical key detection tasks.

1.2 Contribution

We believe our thesis can be seen as contributing to the field of key detection in three noteworthy ways:

Comprehensive Evaluation and Comparison: We conducted a thorough evaluation and comparison of traditional rule-based approaches and various machine learning models for key detection. By rigorously testing these methods across diverse datasets, we identified the strengths and limitations of each approach, providing valuable insights into their performance in different musical contexts. This analysis is intended to guide future research and application development in Music Information Retrieval (MIR) by highlighting the most effective techniques for key detection.

Novel Machine Learning Architecture: We designed and implemented a novel machine learning architecture specifically for key detection. This new model, tailored to address the complexities of musical data, has demonstrated improved performance compared to traditional methods and existing machine learning models. We hope our architecture can serve as a foundation for further enhancements and to inspire new approaches within the field.

Future Research Directions: We have outlined the limitations of our current methodology and proposed areas for future research. By identifying existing challenges and suggesting potential solutions, we aim to encourage ongoing exploration and refinement of key detection techniques. This forward-looking perspective is intended to drive continuous progress and innovation in the field of MIR.
Through these contributions, we aim to advance the understanding and capabilities of key detection, providing a solid foundation for future research and development.

1.3 Thesis Outline

The structure of this thesis is designed to systematically address the research objectives and provide a clear path through the various stages of our work. The chapters are organized as follows:

- **Introduction**: This chapter introduces the motivation behind the research, the significance of key detection in Music Information Retrieval (MIR), and the potential of machine learning solutions. It also outlines the contributions of the thesis and provides an overview of the structure.
- **Theoretical and Technical Background**: Here, we provide the foundational knowledge and technical details necessary to understand the subsequent chapters. This includes a review of music theory, an explanation of the key concepts in key detection, and a discussion of the various file formats and pre-processing techniques used.
- **Literature Review**: This chapter presents a comprehensive review of the existing literature on key detection methods. We cover both traditional rule-based approaches and modern machine learning techniques, highlighting their strengths and limitations.
- **Methodology**: In this chapter, we detail our research design, including the main objectives and the comprehensive roadmap we followed. We describe the data collection process, the selection of evaluation metrics, and the implementation of both traditional and machine learning approaches. We also introduce our novel machine learning architecture.
- **Results**: This chapter presents the results of our experiments, including the performance of various key detection methods on different datasets. We provide detailed analyses and comparisons to demonstrate the effectiveness of our approaches.
- **Discussion**: Here, we interpret the results, answering the key research questions posed at the outset. We discuss the implications of our findings, the symmetrical properties of the features, and the versatility of the traditional and novel approaches. We also address the limitations of PCP-based approaches and propose directions for future work.
- **Conclusion**: The concluding chapter summarizes the main findings of the research, reiterates the contributions, and reflects on the overall significance of the work. We also suggest potential areas for further investigation.
- **References**: This section lists all the scholarly works cited throughout the thesis, providing a comprehensive bibliography for readers interested in exploring the topic further.

Chapter 2

Theoretical and Technical Background

In this chapter, we provide the foundational knowledge and technical details necessary to understand both the literature review and our methodology. The goal is to ensure that readers are equipped with the essential definitions, concepts, and technical descriptions that are crucial for comprehending the subsequent sections of this thesis.

2.1 Music Theory

2.1.1 Music as an Organized Sequence of Sounds

Music is defined as an organized sequence of sounds with quantifiable and analyzable characteristics. By focusing on the objective aspects of music, we can systematically study and analyze its elements. [1]

2.1.2 Elements of Music

The fundamental elements of music include:

- **Melody**: A sequence of notes arranged rhythmically to form a recognizable tune.
- **Harmony**: Combination of different notes played simultaneously to produce chords.
- **Rhythm**: Pattern of sounds and silences, determining the timing and flow of music.
- **Texture**: How melodic, rhythmic, and harmonic elements are combined.
- **Form**: Structure and organization of a musical composition.
- **Dynamics**: Volume of sound, providing emotional context through variations in intensity.

Among these elements, **melody** and **harmony** are most closely related to the key of a piece of music, as they directly depend on the scale and the tonal center. **Form** can also be influenced by the key, especially in terms of modulations and key changes throughout a composition. On the other hand, **rhythm**, **texture**, and **dynamics** are less directly related to the key. While they contribute significantly to the overall feel and structure of the music, they do not depend on the specific pitches and intervals that define a key. [2]

2.1.3 Scales as Organizational Tools in Music

Scales are subsets of the 12 notes in the chromatic scale, organized around a tonic (reference tone). Heptatonic scales (scales with seven notes) are particularly important, especially those following a diatonic pattern of whole steps (W) and half steps (H). [3]

Scales serve as useful frameworks for musicians, functioning as ”creative restrictions.” These frameworks make it easier to build intended melodies and harmonies. Without scales, it would be challenging to guide a composition in a way that maintains unity and coherence. By working within the confines of a scale, musicians can more easily create music that sounds cohesive and structured.

Major and Minor Quality

Scales are classified as major or minor based on the arrangement of intervals between their notes. The third interval in relation to the tonic is particularly important, as it ultimately distinguishes whether a scale is considered major or minor:

- **The Major Scale** Follows the interval pattern W-W-H-W-W-W-H, creating a bright and happy sound. All intervals in the major scale are either perfect or major, making it the ”most major of the heptatonic scales”.

- **The Natural Minor Scale** Follows the interval pattern W-H-W-W-H-W-W, creating a somber and melancholic sound.

Every scale exhibits qualities that make it sound more major or more minor, depending on the relationship of its notes to the tonic, but it is the third interval that primarily determines the scale’s classification as major or minor.

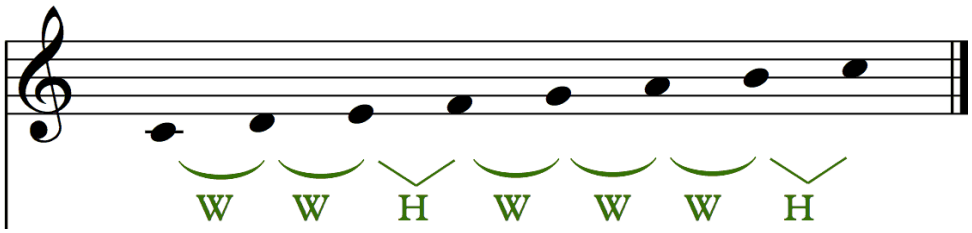

Figure 2.1: The C Major Scale

2.1.4 Definition of a Musical Key

A **musical key** consists of:

1. **Tonic or Tonal Center**: The main note around which a piece of music is organized.

2. **Minor or Major Quality**: This quality tells us whether the harmony of the piece is primarily built upon a scale that is minor or major.

Given the 12 distinct notes in Western music and the two modalities (major and minor), there are a total of 24 possible keys. These keys are listed and numbered as follows:

Number	Major Key	Number	Minor Key
0	C Major	12	A Minor
1	C# Major	13	A# Minor
2	D Major	14	B Minor
3	D# Major	15	C Minor
4	E Major	16	C# Minor
5	F Major	17	D Minor
6	F# Major	18	D# Minor
7	G Major	19	E Minor
8	G# Major	20	F Minor
9	A Major	21	F# Minor
10	A# Major	22	G Minor
11	B Major	23	G# Minor

Table 2.1: List of the 24 possible keys

Understanding the Key in Music

Finding the key of a piece of music is closely related to identifying the scale that is predominantly used. However, the concept of a key is somewhat broader. While a piece of music might incorporate both major and minor scales and the tonal center may shift during chord progressions, the overall key generally remains constant. This constancy captures the prevailing tonal center and the overarching major or minor quality of the piece.

For example, in blues music, songs often use the blues scale, which includes both a minor third and a major third. Despite this, blues songs are categorized as being either in a minor key or a major key based on the overall tonal center and quality. For instance, a blues piece might predominantly feature a C major chord progression, but also incorporate E-flat (minor third) and E (major third) notes. Even with these variations, if the piece primarily revolves around C major chords and scales, it is considered to be in the key of C major. This demonstrates how the key provides a sense of unity and coherence, despite the inclusion of notes from both major and minor scales.

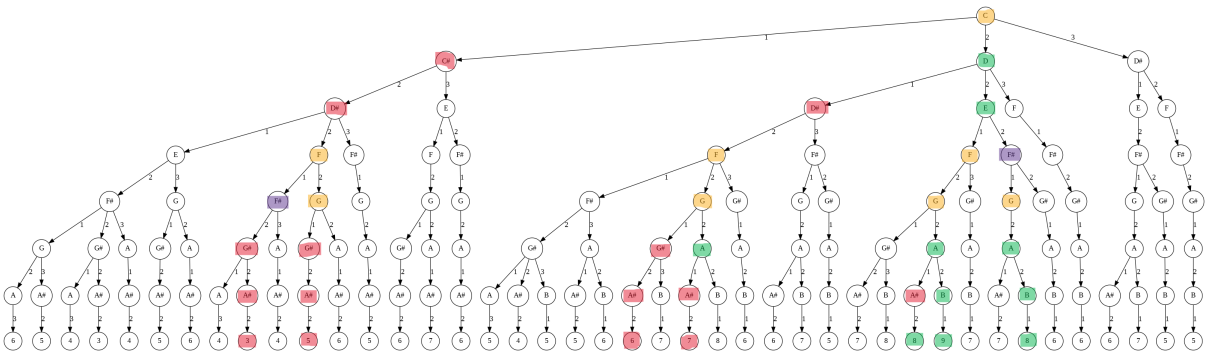

Figure 2.2: Diatonic Paths Colored by Major or Minor Quality in relation to the Tonic

2.2 File Formats and Pre-processing

2.2.1 Storing Music in a Computer

Key Detection algorithms operate on data that should in one way or another encode information about the notes played by non-percussive instruments. A note can be broken down into two main components: **Pitch** and **Duration**. The ability to extract this information for each note in a musical piece largely depends on the file format in which the music is stored.

Music can be stored in a computer in two primary ways: as an audio file (e.g., WAV, MP3) or in a symbolic form (e.g., MIDI, XML). This is directly analogous to how speech can be stored either as an audio file or as a text file. The text file is what we would call a symbolic form of storing speech. In the case of speech, the audio file captures the raw sound, while the text file captures the symbolic representation of the spoken words. Similarly, in music, audio files capture the sound of the performance, whereas symbolic formats capture the notation and instructions for performing the music, allowing for easy manipulation, analysis, and reproduction by musicians and software.

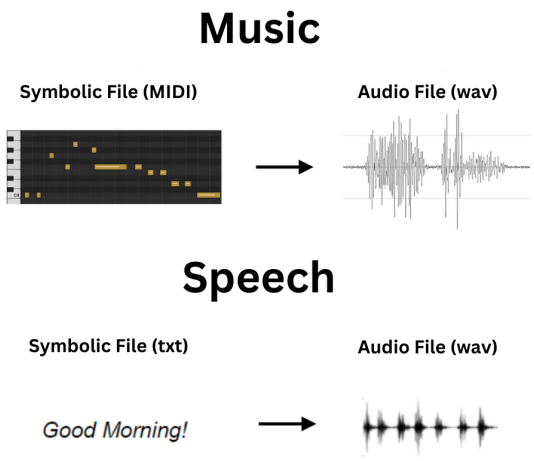

Figure 2.3: Symbolic File Formats vs Audio Files

Audio files are the standard format for distributing music to the public. Symbolic formats, on the other hand, are often used among musicians and music producers for various applications. Symbolic formats directly encode the pitch and duration of each note, which significantly simplifies the preprocessing required for key detection algorithms. This direct encoding eliminates the need for complex audio analysis to determine the pitch and duration, making the symbolic format much more convenient for this purpose.

2.2.2 Common pre-processing steps

Both unprocessed audio files and unprocessed symbolic files contain a lot of irrelevant information regarding the task of key detection. This extraneous data can include aspects such as the texture of the music, dynamics, and other characteristics that do not directly contribute to identifying the musical key. Therefore, preprocessing is essential to obtain representations of the music that primarily include information about note pitch and duration. This allows our key detection algorithms to operate on data that is more likely to be indicative of the key.

Here, we discuss the preprocessing steps needed to obtain the representations commonly used in the literature: Spectrograms, Chromograms, and Pitch Class Profiles (PCPs).

Spectrograms are visual representations of the spectrum of frequencies in a signal as they vary with time. When music is stored in an audio format (e.g., WAV, MP3), a spectrogram can be obtained by applying a Short-Time Fourier Transform (STFT) to the audio signal. This process involves dividing the audio signal into overlapping segments, applying the Fourier Transform to each segment, and combining the results to produce a two-dimensional matrix of time versus frequency with intensity representing amplitude. When music is stored in symbolic form (e.g., MIDI, XML), spectrograms are typically not used directly because symbolic formats already provide precise note information without the need for frequency analysis. Instead, the note information can be used to generate synthetic audio, from which a spectrogram can be created if needed.

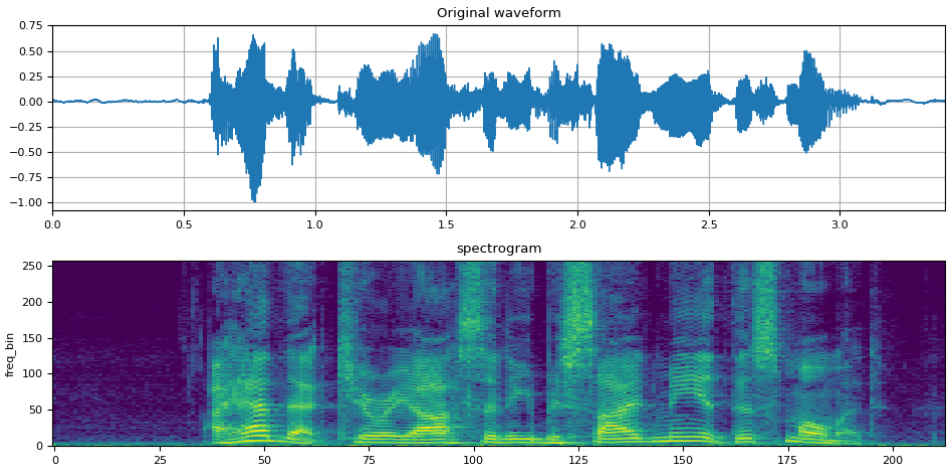

Figure 2.4: Example of a Spectrogram

Chromograms (or Chroma features) represent the 12 different pitch classes of the equal-tempered scale, capturing harmonic and melodic characteristics. In audio format, chromograms are obtained by first computing the spectrogram of the audio signal and then mapping the spectral content into 12 pitch classes (or chroma bins) corresponding to the 12 notes of the musical octave. This process can also involve the Constant-Q transform, which provides a logarithmic frequency resolution that aligns with musical perception. By grouping notes together based on octave equivalence, we lose information about the exact pitch of the note played but simplify our features while keeping the ”musical essence” of the sound. When music is stored in symbolic form, such as MIDI, the chroma features can be directly extracted from the note events without needing a spectrogram. Each note event is mapped to its corresponding pitch class, and the chroma feature vector is constructed by counting or summing the duration of occurrences of each pitch class.

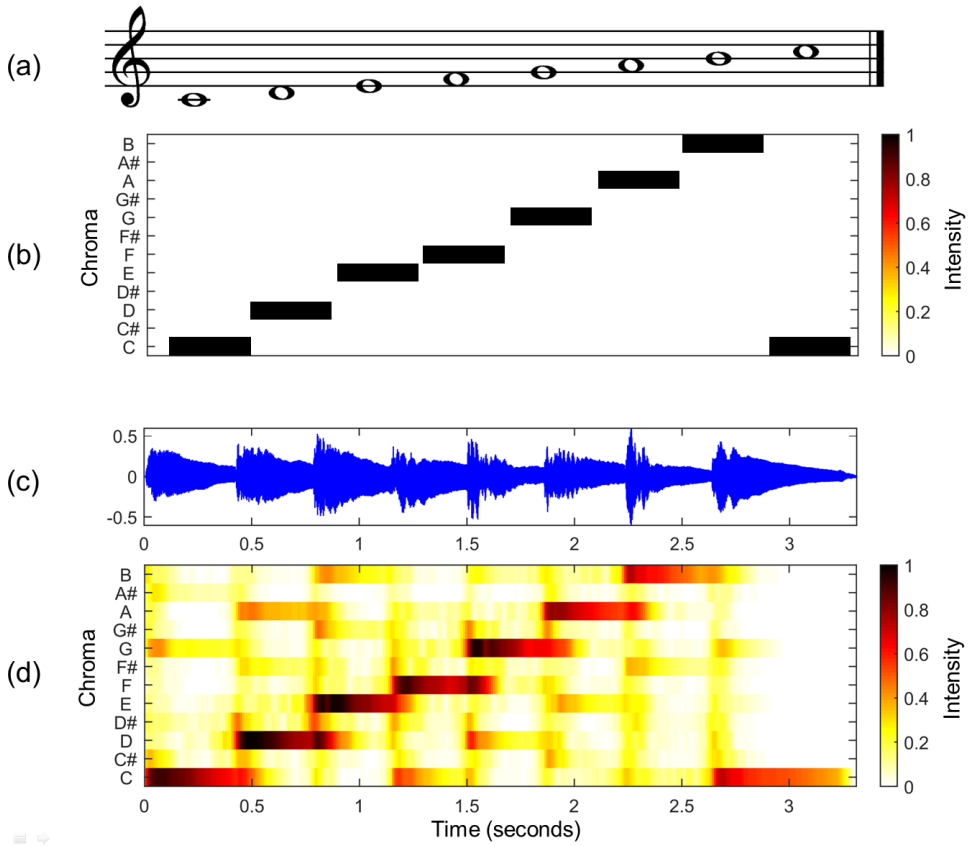

Figure 2.5: Example of a Chromogram

Pitch Class Profiles (PCPs) are the simplest representation used in musical key detection, as they discard information about the timing of each note and focus solely on the relative intensity of each pitch class (note) in a piece of music. This simplification is a tradeoff: while PCPs do not capture the temporal dynamics of the music, they encapsulate essential information about the notes and the scales used in the musical piece in just 12 numbers, corresponding to the 12 pitch classes in an octave.

PCPs essentially represent the distribution of notes played throughout the piece. This captures important details about the harmonic content of the music, making PCPs a valuable tool for key detection. By averaging the occurrences of each pitch class over the entire piece, PCPs can highlight the predominant notes and, by extension, the likely scales and keys used.

For audio files, PCPs are typically derived by first computing a chromogram and then normalizing the energy values within each pitch class. This process creates a profile that reflects the prominence of each pitch class over time. In symbolic formats like MIDI, the PCP can be directly calculated by mapping each note to its pitch class and summing the note durations or counts for each class. The result is a vector representing the distribution of pitch classes throughout the musical piece.

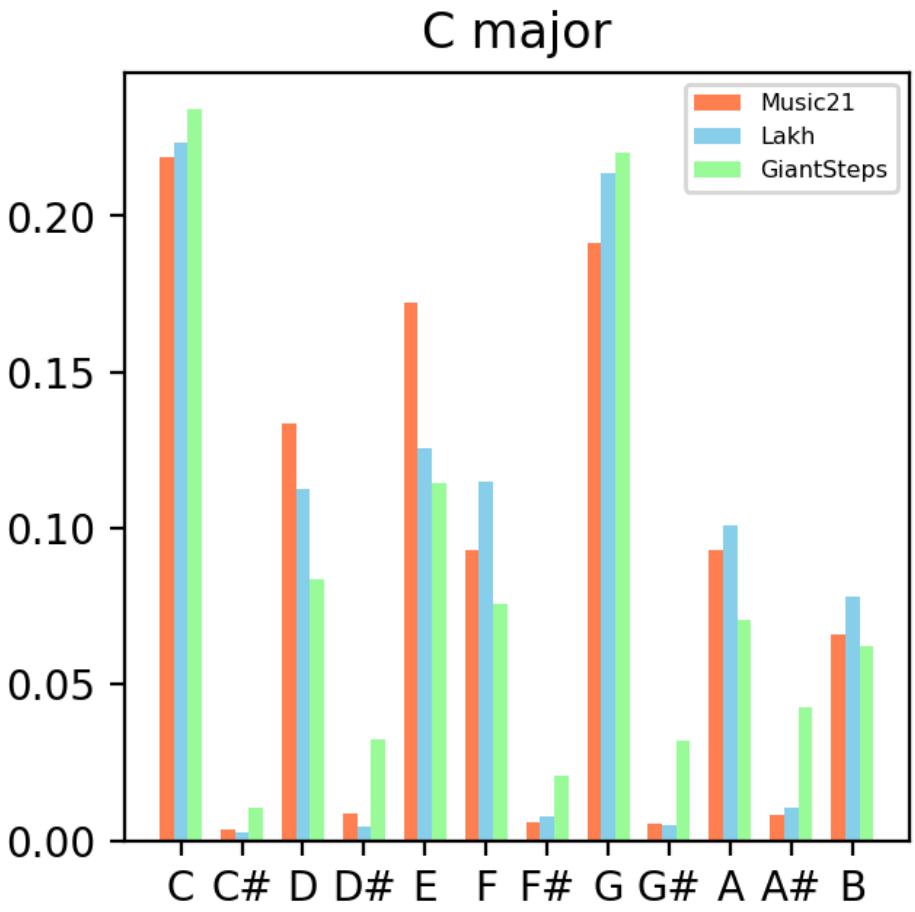

Figure 2.6: Example of a Pitch Class Profile

In our experiments, we chose to use PCPs as they provide a compact and effective representation of the musical content, capturing the overall distribution of notes without the complexity of temporal information. This makes them particularly suitable for our key detection algorithms, which can efficiently process these 12-dimensional vectors to identify the key of the musical piece. Essentially, PCPs are average chromograms, simplifying the data while retaining the core harmonic information needed for accurate key detection.

2.3 Machine Learning Algorithms for Key Detection

The task of key detection is a multiclass classification problem with 24 classes. This section explains how various machine learning techniques [4], capable of performing classification tasks, are applied specifically to key detection. We provide a brief, intuitive explanation of how these algorithms make their decisions.

2.3.1 Naive Bayes Algorithm

Imagine you have no knowledge of music theory, but you are given a large dataset of Pitch Class Profiles (PCPs) labeled according to the musical key they represent. If you were trying to determine the key of a new piece of music based on these PCPs, you might adopt an approach similar to the Naive Bayes algorithm.

You observe that PCPs labeled as C major typically have a high prominence of 'C' notes. Therefore, when you are given a new PCP and asked to determine its key, you would look at how large the 'C' value is in this PCP to make your decision. This intuitive approach is essentially what the Naive Bayes classifier does.

The Naive Bayes classifier [5] is a probabilistic model that applies Bayes' theorem with strong independence assumptions between features. For key detection, this algorithm uses PCPs as features. Each pitch class (note) is treated as an independent feature, and the classifier calculates the likelihood of a piece of music belonging to each key based on the prominence of different pitch classes.

Bayes' theorem is given by:

$$P(C_k|X) = \frac{P(X|C_k) \cdot P(C_k)}{P(X)}$$

where:

- $P(C_k|X)$ is the posterior probability of the piece of music belonging to key C_k given the PCP X.
- $P(X|C_k)$ is the likelihood of observing PCP X given that the key is C_k.
- $P(C_k)$ is the prior probability of the key C_k.
- $P(X)$ is the evidence, or the total probability of observing PCP X.

In the context of key detection:

- $P(C_k)$ can be thought of as the prior probability of each key, which might be uniform if we have no reason to prefer one key over another.
- $P(X|C_k)$ is calculated based on the frequencies of pitch classes in the training data by making two assumptions:
    1. The frequencies of pitch classes are conditionally independent given the key.
    2. The frequency of each pitch class follows a normal distribution.

Despite the independence assumption often being violated (e.g., C and G frequently co-occurring), Naive Bayes provides a solid baseline for key detection by leveraging the statistical patterns observed in the labeled dataset.

This approach allows us to make informed predictions about the key of a new piece of music by considering the relative prominence of each pitch class in the PCP, thus providing a straightforward and computationally efficient method for key detection.

2.3.2 K-Nearest Neighbors (KNN)

Another intuitive approach would be to try and find a Pitch Class Profile (PCP) from the ones you know their key that closely matches the one you were asked to classify. The K-Nearest Neighbors (KNN) algorithm offers an effective method to achieve this.

KNN [6] is a simple, instance-based learning algorithm that classifies a musical piece based on the key labels of its nearest neighbors in the feature space. When you receive a new PCP, you

calculate the distance between this PCP and all the PCPs in your training set. You then identify the k closest (or nearest) neighbors based on these distances.

The decision for the key of the new piece is made by taking a majority vote among the keys of the k nearest neighbors. For example, if you set $k = 5$ and three out of the five nearest neighbors are labeled as C major, the new piece will be classified as C major.

The KNN algorithm is defined as follows:

$$\hat{y} = \arg\max_{k \in K} \sum_{i \in N_k} I(y_i = k)$$

where:

- $\hat{y}$ is the predicted key.
- K is the set of all possible keys.
- N_k is the set of the k nearest neighbors.
- $I(y_i = k)$ is an indicator function that returns 1 if the key of the i-th neighbor is k and 0 otherwise.

KNN relies on the choice of the distance metric, with common choices including Euclidean distance, Manhattan distance, and Minkowski distance. Additionally, the value of k can significantly affect the performance of the algorithm; a small k makes the model sensitive to noise, while a large k may smooth out the decision boundary too much.

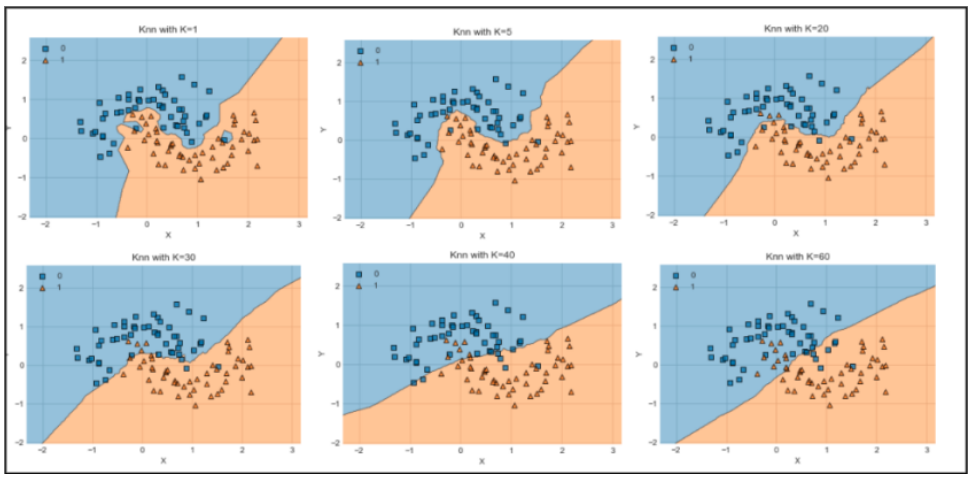

Figure 2.7: Example of K-Nearest Neighbors Classification

For key detection, PCPs are used as features. KNN calculates the distance between the new PCP and all the PCPs in the training set, identifies the k closest ones, and assigns the most common key among them. Transforming the features (e.g., squaring their values) can enhance the performance of KNN by improving the structure and separability of the data.

This approach leverages the similarity between PCPs to make informed predictions about the key of a new piece of music, providing a straightforward and intuitive method for key detection.

2.3.3 Support Vector Machines (SVM)

SVM's [7] can be thought of as "drawing the line" that separates the PCP's that belong to a particular key from all those that don't. In two-dimensional space, this line is called a hyperplane. In our case, instead of drawing a line, we find a hyperplane in 12-dimensional space that best separates each class from the others.

Support Vector Machines (SVM) are powerful supervised learning models that aim to find the optimal hyperplane that best separates the data into different classes. The key idea is to maximize the margin between the hyperplane and the nearest data points from each class, which are known as support vectors.

For binary classification, SVM solves an optimization problem to find the hyperplane:

$$\mathbf{w} \cdot \mathbf{x} - b = 0$$

where $\mathbf{w}$ is the weight vector and b is the bias term.

In the context of key detection, SVM uses PCPs as features. The goal is to find a hyperplane in the 12-dimensional space of PCPs that separates the PCPs of different keys.

For multiclass classification, SVM can be extended using strategies such as:

- **One-vs-One (OvO)**: This approach involves creating a binary classifier for every pair of classes. For k classes, this results in $k(k-1)/2$ classifiers. During prediction, each classifier votes for a class, and the class with the most votes is assigned to the data point.

- **One-vs-Rest (OvR)**: In this approach, a single binary classifier is trained for each class to distinguish that class from all other classes. During prediction, the class with the highest confidence score from the respective classifiers is assigned to the data point.

The SVM algorithm is defined as follows:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

subject to:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1 \quad \forall i$$

This optimization problem ensures that the hyperplane maximizes the margin between classes, making the classification robust.

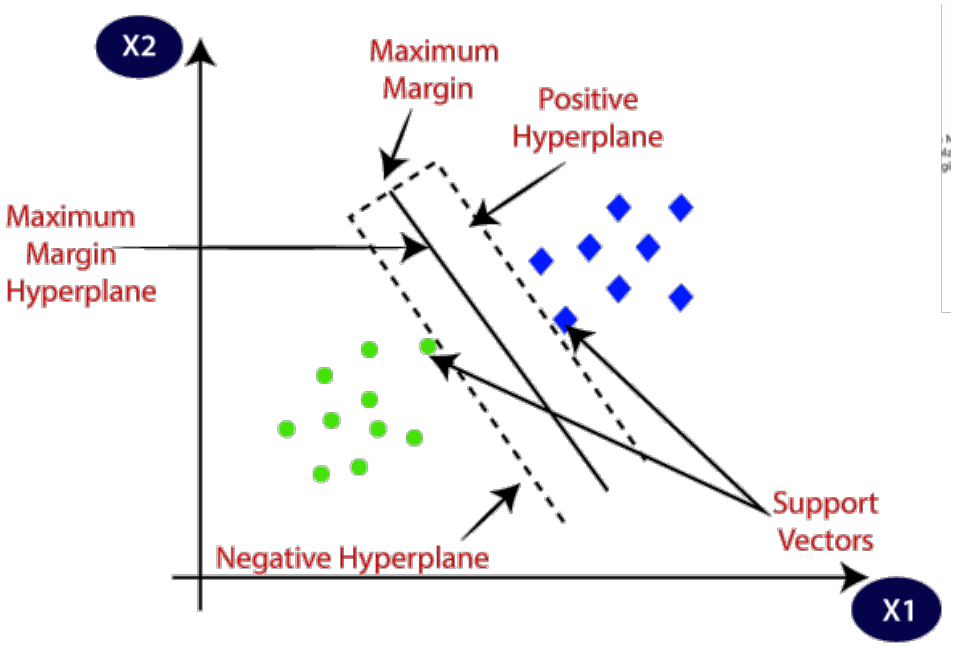

Figure 2.8: Example of Support Vector Machine Classification

This approach leverages the geometric properties of the data to make informed predictions about the key of a new piece of music, providing a powerful method for key detection.

2.3.4 Convolutional Neural Networks (CNNs)

The previous algorithms we examined all used PCPs as the representation of the harmonic and melodic context of the piece. CNNs [8] are capable of working with more complex representations such as spectrograms, chromograms, or even directly with the waveform of the piece.

CNNs do not require us to specifically choose what constitutes a useful feature for the task of key detection. Instead, they use their convolutional layers to extract features that they determine are useful during their training. For example, a CNN can learn to identify patterns in a spectrogram that correspond to specific harmonic or melodic characteristics relevant to identifying the musical key.

Convolutional layers work by applying filters (or kernels) across the input data to detect these patterns. Each filter moves (or convolves) over the input matrix, performing a dot product between the filter and a small region of the input. This operation produces a feature map that highlights the presence of the pattern learned by the filter at different locations in the input.

As training progresses, CNNs learn multiple filters that can detect increasingly complex patterns. Early layers might detect simple features like edges or specific frequencies, while deeper layers can capture more complex structures like chords or rhythmic patterns.

The subsequent layers of the CNN use these extracted features to classify the input. These layers can include:

- **Pooling Layers**: These layers reduce the spatial dimensions of the feature maps, making the computation more efficient and helping to avoid overfitting.
- **Dense (Fully Connected) Layers**: These layers perform the final classification by combining the features extracted by the convolutional layers. Each neuron in a dense layer

is connected to all neurons in the previous layer, allowing the network to make decisions based on the combination of features.

In the literature, CNNs are commonly used with spectrograms as their input. Spectrograms provide a visual representation of the frequency spectrum of the audio over time, capturing both temporal and spectral features. This makes them particularly suitable for CNNs, which excel at processing grid-like data structures.

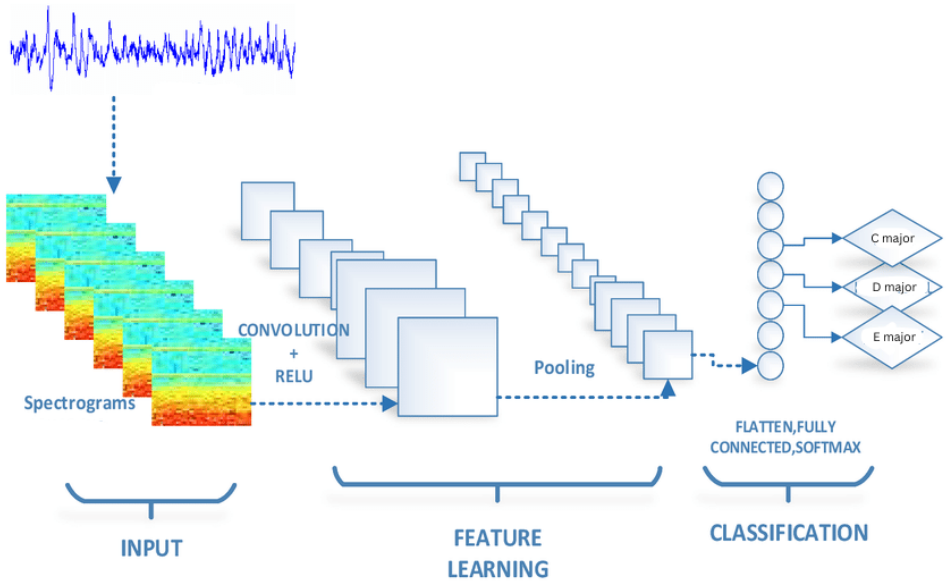

Figure 2.9: Example of Convolutional Neural Network Architecture

By leveraging the powerful feature extraction capabilities of convolutional layers, CNNs can effectively analyze complex musical representations and perform accurate key detection.

2.3.5 Hidden Markov Models (HMMs)

To understand how Hidden Markov Models (HMMs) [9] can effectively model a musical piece and be used for key detection, let us consider the following illustrative scenario.

Imagine a composer who has a very particular system by which she composes music. In her room, she has 24 pianos (P0, P1, P2, ...) corresponding to 24 different musical keys (12 major and 12 minor). Each of these pianos is tuned to primarily play notes from one particular triad. For example, the P1 piano is tuned such that most of the keys are either 'C', 'E', or 'G' (C major triad). However, each piano also contains other notes, though it is less likely for the composer to press them.

For each bar of her composition, the composer randomly chooses one of the pianos and plays some of its keys. The choice of the piano for each bar depends only on the piano chosen for the previous bar, following a Markov process. The sequence of keys she presses creates a Pitch Class Profile (PCP) that can be heard by the listener.

Since the listener cannot see the composer's choice of piano, they only hear the sequence of PCPs. This arrangement is called a "hidden Markov process."

Building a Key Detection Algorithm Using HMMs

Building a key detection algorithm based on HMMs requires us to think of compositions as being created in such a manner. We only observe a sequence of PCPs for each bar of the composition and try to infer the hidden states (chords) and, subsequently, the key.

Components of the HMM:

- **States (Chords)**: The hidden states represent different chords.
- **Observations (PCPs)**: The observable states are the PCPs that result from playing chords.
- **Initial State Probabilities (π)**: The probabilities of starting with each chord.
- **Transition Probabilities (A)**: The probabilities of transitioning from one chord to another.
- **Emission Probabilities (B)**: The probabilities of a chord producing a specific PCP.

Each key corresponds to a different HMM. Therefore, to determine the key of a piece of music, we evaluate which HMM (key model) best explains the observed sequence of PCPs.

Steps to Build and Use the HMM for Key Detection

1. **Training**: Train separate HMMs for each key using labeled datasets. Each HMM learns the initial state probabilities, transition probabilities, and emission probabilities for its respective key.
2. **Feature Extraction**: Extract sequences of PCPs from the musical piece. Each PCP represents the harmonic content at a specific time.
3. **Evaluation**: For a given sequence of PCPs, compute the likelihood that each trained HMM (key model) explains the sequence. The HMM with the highest likelihood indicates the most probable key for the piece of music.

Example Workflow

1. **Initialization**: Define the initial state probabilities (π), transition probabilities (A), and emission probabilities (B) for each HMM corresponding to each key.
2. **Training**: Using a training dataset with known chord progressions and keys, estimate the probabilities for each HMM. For instance, in C major, the HMM learns the probabilities for transitions between chords and the emission probabilities of PCPs given those chords.
3. **Decoding**: Given a new sequence of PCPs, use the Viterbi algorithm to compute the most probable sequence of chords for each HMM (key model). Calculate the likelihood of the observed sequence under each key model.
4. **Classification**: Compare the likelihoods produced by each HMM. The key corresponding to the HMM with the highest likelihood is selected as the key for the piece.

Practical Example
Consider a sequence of PCPs: $[y_1, y_2, y_3, ...]$. The HMMs for key detection work as follows:

- **States**: Different chords (e.g., C major chord, G major chord).
- **Observations**: PCPs corresponding to each bar.
- **Initial State Probabilities**: The likelihood of starting with each chord.
- **Transition Probabilities**: Probabilities of transitioning from one chord to another.
- **Emission Probabilities**: Likelihood of a PCP given a chord.

For each key (e.g., C major, A minor), a separate HMM is trained. Given the sequence of PCPs, the algorithm evaluates which HMM best explains the sequence. This helps in accurately detecting the key throughout the musical piece.

2.3.6 Transformers for Key Detection

Transformers [10] are a powerful type of model originally designed for natural language processing tasks. They excel at modeling sequential data and have been successfully applied to a wide range of language-related tasks, such as audio-to-text transcription, text classification, machine translation, and sentiment analysis. Transformers are particularly effective because they can capture long-range dependencies and contextual information through mechanisms like self-attention, which allows them to weigh the importance of different parts of the input sequence when making predictions.

Musical key detection is similar to language detection in that it involves analyzing sequences of data (notes or chords) to determine an underlying structure (the key). Just as in language processing, where understanding the context and relationships between words is crucial, understanding the relationships between notes and chords is essential for accurate key detection. The challenge in using transformers for musical key detection lies in training an effective tokenizer that can capture meaningful tokens from the music. These tokens need to represent the harmonic and melodic content in a way that the transformer model can process effectively.

An effective tokenizer for musical data must be able to transform the raw input (such as a sequence of notes or chords) into a series of tokens that encapsulate the essential information needed for key detection. This process is akin to tokenizing text in language processing, where the goal is to break down the text into manageable pieces while preserving the meaning and context. By training a transformer model on these musical tokens, it becomes possible to leverage its powerful sequence modeling capabilities to perform key detection with high accuracy.

Chapter 3

Literature Review

3.1 A Guide to the Literature Review

We conducted a comprehensive literature review to chart the diverse approaches used in addressing the challenge of musical key detection. Our initial focus was on exploring rule-based methodologies aimed at accurately determining the key of musical compositions. This investigation aimed to assess the presence of deterministic algorithms capable of reliably identifying musical keys, while also evaluating the efficacy of heuristic-based approaches in achieving similar outcomes. Next, we examined how modern machine learning approaches could be utilized to either refine the aforementioned heuristics or perform direct key predictions given the necessary features of a musical piece.

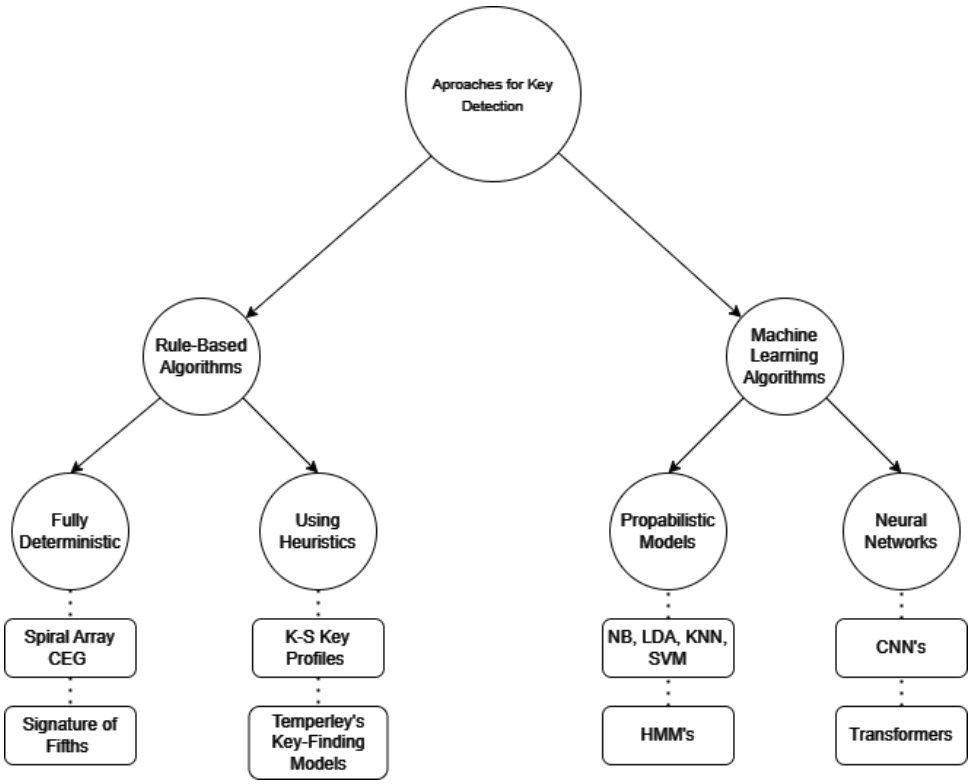

Figure 3.1: Map of the Various Approaches for The Task of Key Detection

The diagram in Figure 3.1 illustrates a taxonomy of the various approaches, which we found to be particularly useful. The following sections of this chapter are structured based on this

taxonomy. We will present eight selected works that we believe provide a comprehensive overview of the landscape of key detection approaches.

3.2 Rule Based Approaches

3.2.1 Heuristic Algorithms

Krumhansl and Shepard's Key Finding Algorithm

Carol L. Krumhansl and Roger N. Shepard published a Journal Article named "Quantification of the Hierarchy of Tonal Functions within a Diatonic Context" [11]. In this publication they present the methodology and results of their "Probe Tone Experiments" which serve as the basis for the key estimation algorithm Krumhansl describes in her book named "Cognitive Foundations of Musical Pitch"[12].

The Probe Tone experiments by Carol L. Krumhansl and Roger N. Shepard investigated how listeners perceive the stability and fit of individual tones within a given musical context. Participants were presented with a musical context, such as a scale or chord progression, which established a particular key. Following the context, a single tone, known as the probe tone, was played. Participants then rated how well the probe tone fit with the preceding context on a numerical scale. This process was repeated for all 12 chromatic pitches within the same context.

The average ratings for each probe tone within a specific key were calculated to create a key profile. These profiles represent the perceived stability of each chromatic pitch within the key. For a major key profile, higher values were assigned to the tonic (1st scale degree), dominant (5th scale degree), and mediant (3rd scale degree) tones, reflecting their perceived stability. The minor key profile was similar but adjusted for the tonal hierarchy in minor keys.

Key profiles are used to identify the key of a musical piece by analyzing the pitch class distribution of the piece and comparing it with stored key profiles. This comparison is done using correlation coefficients to measure the fit between the distribution and each key profile. The key with the highest correlation to the pitch class distribution of the piece is identified as the key of the music.

Temperley's Key Profiles and Dynamic Programming Approach

In his publication named "A Bayesian Approach to Key-Finding" [13], David Temperley introduced significant advancements to the aforementioned key detection algorithm. He enhanced the traditional key profiles by integrating Bayesian statistical models, which provided a more probabilistic representation of pitch distributions within different keys. This approach allowed for a more nuanced assessment of the fit between observed pitch patterns and the expected patterns for each key.

Temperley addressed the challenge of modulation by developing a dynamic key profile adaptation mechanism. Rather than assuming a static key throughout a musical piece, his algorithm dynamically adjusted key probabilities based on evolving pitch distributions. This adaptation enabled the algorithm to accurately detect key changes and transitions, which are common in complex musical compositions.

Furthermore, Temperley employed dynamic programming techniques to optimize the computational efficiency and accuracy of key sequence detection. By considering both local and global coherence in key changes, his algorithm effectively identified the most probable sequence of keys across the entire piece of music. This holistic approach ensured robustness against local pitch anomalies and provided a coherent interpretation of key structures in diverse musical contexts.

Overall, David Temperley's Bayesian approach, coupled with dynamic key profile adaptation and dynamic programming techniques, significantly enhanced the reliability and flexibility of key detection algorithms, making them more suitable for analyzing intricate and modulating musical compositions.

3.2.2 Fully Deterministic Algorithms

A Method Based on the Signature of Fifths

The paper titled "A Comparison of the Music Key Detection Approaches Utilizing Key-Profiles with a New Method Based on the Signature of Fifths" [14], a new music key detection algorithm is proposed. The signature of fifths is a geometrical descriptor of the harmonic content of music. It represents the harmonic content of a piece in terms of polar vectors referring to pitch classes that are arrayed with reference to the circle of fifths. However, this approach differs from the key-profile methods that employ predetermined profiles in order to find correlation coefficients of the profiles with the pitch-class distributions in musical fragments.

The algorithm starts with the representation of the musical piece using only twelve objects, chromatic pitch-classes of the scale. It sums up the number of times of these tones or the total amount of durations of these tones, thus generating a vector X. The elements of X are then normalized to form a vector K they are ordered by the circle of fifths.

The next step is to form the signature of fifths, which is done by creating polar vectors for each pitch-class based on the values obtained and the fixed angles (multiples of 30 degrees). The algorithm also outlines directed axes of the circle of fifths and determines their values, identified by sums of the vector measurements on the sides of each axis. The greatest characteristic value is identified to be the main directed axis of the signature of fifths.

The directed axis aids in the identification of the two relative keys from the numerous available keys. The last key is computed through further calculations, which take lesser efforts as compared to the standard mathematical procedures.

The authors used as materials preludes No. 1, 2 by J. S. Bach, No. 4,12 by F. Chopin, and songs by The Beatles. They analyzed two scenarios: one based on the total amounts of time each pitch-class has been strongly played and the other base on the number of times a pitch-class has repeated itself. Therefore, the given method of signature of fifths revealed higher efficiency in key detection based on the shortest initial musical segments with reference to the aggregate durations, while the methods based on key-profiles are more efficient while taking the multiplicities of occurrences into account.

The result of the study showed that the method identified as the signature of fifths is approximately on par with the key-profile methods, though is better for short music fragments. It provides computational ease, reliability in the decisions taken and works fine for key detection. Research in the future will seek to refine the method to a stage where correlation coefficient

determination for final key selection will not be necessary anymore. This new approach can be viewed as an efficient solution when it comes to real-time music key identification in electronic musical instruments.

Polyphonic Audio Key Finding Using the Spiral Array CEG Algorithm

The paper titled "Polyphonic Audio Key Finding Using the Spiral Array CEG Algorithm" [15] presents a new real-time algorithm for determining the key of polyphonic audio. The method is based on the Spiral Array model [16] and utilizes Chew's Center of Effect Generator (CEG) algorithm. This approach is distinct from traditional key-profile methods, such as the Krumhansl-Schmuckler (K-S) probe tone profile method, which rely on pre-determined profiles to correlate with pitch-class distributions in musical fragments.

The algorithm begins by extracting pitches and pitch strengths from polyphonic audio using the Fast Fourier Transform (FFT) along with a local maximum detection scheme. This extraction results in pitch class and strength information. These pitches are then mapped onto the Spiral Array model, which is a hierarchical representation of pitches, intervals, chords, and keys within a three-dimensional spatial framework. This model allows for efficient pitch spelling and comparison of tonal structures.

The CEG algorithm performs a nearest-neighbor search within the Spiral Array space to identify the key. The key identification process involves generating a center of effect (CE) based on the normalized pitch strengths and determining the key through this nearest-neighbor search.

The study tested the proposed system on audio samples generated from MIDI sources of Mozart's symphonies. The results showed that the CEG algorithm achieved a maximum correct key recognition rate of 96% within the first fifteen seconds and over 90% accuracy within the first three seconds. These results significantly outperformed the K-S and modified K-S methods, which showed correct key recognition rates under 50% within the first three seconds and reached maximum values of 80% and 87% respectively within fifteen seconds.

Two scenarios were analyzed: one based on the total durations of each pitch class and the other on the frequency of pitch class occurrences. The CEG algorithm consistently provided higher accuracy and efficiency in key detection compared to traditional methods, particularly in shorter musical segments.

The paper concludes that the CEG algorithm, with its integration of the Spiral Array model, offers a robust and efficient solution for real-time key detection from polyphonic audio. The authors suggest future research directions, including testing the algorithm on a larger and more varied corpus of music, examining the effect of harmonic reinforcement on pitch class information, and exploring ways to bias the pitch detection process to further enhance key finding.

This method not only improves computational ease but also provides reliable key detection, making it a promising tool for content-based music indexing and retrieval. Future research aims to refine the method and further validate its effectiveness in various musical contexts.

3.3 Machine Learning Approaches

3.3.1 Approaches Utilizing Probabilistic Models

An Intelligent Model Utilizing Probabilistic Classifiers

The paper "Development of an Intelligent Model for Musical Key Estimation Using Machine Learning Techniques" [17] presents a novel approach to determine the musical key of a given song. The authors, Abraham George, X. Anitha Mary, and S. Thomas George, focus on using machine learning algorithms to estimate musical keys, comparing four distinct algorithms: K-nearest neighbor (KNN), Naïve Bayes (NB), Discriminant Analysis (DA), and Support Vector Machine (SVM).

The paper begins by highlighting the importance of musical key data in various musical information retrieval tasks such as transcription, chord estimation, and harmony analysis. Traditional methods for key estimation have primarily focused on developing key profiles, but the possibilities of using machine learning techniques have been underexplored. This study aims to fill that gap by leveraging machine learning for key detection.

The authors use MIDI music data for their experiments due to its precise note alignment and ease of data collection compared to audio data. They compiled a dataset from eight publicly accessible sources, including ORCHSET, Nottingham, MuseData, JSB chorales, Piano-midi.de, Good-sounds, Finnish Folk Tunes, and ClassicalArchives, along with a self-generated dataset. The dataset comprises over 10,000 MIDI songs, but for experimental purposes, only 3243 files were used, annotated with key data by applying widely used algorithms for pitch class profile (PCP) extraction.

The system architecture proposed in the paper consists of two subsystems: one for learning and the other for classification. The learning subsystem trains the machine learning model using extracted features from the MIDI songs, while the classification subsystem uses the learned model to predict the keys of new MIDI songs. The pitch class profile, a 12-dimensional vector representing the relative energy of the twelve pitch classes, is the input for the machine learning algorithms.

Four machine learning algorithms were chosen for the study. The Naïve Bayes algorithm, known for its speed and minimal training data requirements, was optimized by applying different distributions to enhance its objective function. The K-Nearest Neighbor (KNN) method, despite its drawbacks such as decreased efficiency with large datasets, was chosen for its simplicity in multiclass problems. Linear Discriminant Analysis (LDA) was utilized for its effectiveness in dimensionality reduction, and the Support Vector Machine (SVM) approach was selected for its efficiency in high-dimensional spaces and memory utilization. The SVM method, in particular, showed superior performance in separating classes with high accuracy.

Experiments were conducted using 5-fold cross-validation to prevent high bias and variance. The best classifiers from each machine learning algorithm were selected based on their optimized performance metrics. The SVM algorithm achieved the highest accuracy of 91.49%, followed by KNN with 89.76%, Naïve Bayes with 87.11%, and Discriminant Analysis with 86.77%. The SVM model also demonstrated the highest precision, recall, and F1 score values, indicating its effectiveness in key estimation.

The study concludes that the proposed model, particularly with the SVM algorithm, performs

significantly well in musical key estimation. The results show that integrating musical theory with supervised learning techniques can lead to efficient key detection. The authors suggest that future research could focus on applying their findings to other musical traditions, such as Indian music, and exploring the use of deep learning methods for further improvement in key estimation accuracy.

An approach utilizing Hidden Markov Models

The paper "Local Key Estimation From an Audio Signal Relying on Harmonic and Metrical Structures" by Papadopoulos and Peeters [18] presents a method for estimating the progression of musical key from an audio signal. The method addresses the problem of local key finding by combining and extending approaches previously proposed for global key estimation.

The proposed approach relies on the relationship between chords and keys, using a Hidden Markov Model (HMM) where hidden states represent keys observed through chords. This method integrates musical information about key changes and the metrical structure of the audio file to make local key estimation robust.

The audio signal is first down-sampled and converted to mono. A constant-Q transform (CQT) is applied to convert the signal to the frequency domain. Chroma vectors are computed from the CQT, representing the spectral energy of the pitch classes.

Chords are estimated by computing the correlation between chroma vectors and chord templates. Chord templates are constructed considering the presence of higher harmonics of the notes. A chordgram is created, representing the probability of each chord over time.

Two methods for deriving key observation vectors from chords are compared:

- Method 1: Instantaneous chord probabilities are used.
- Method 2: Estimated chord progression is used.

Two HMMs are trained to learn the characteristics of major and minor modes. Twenty-four HMMs are derived, each representing one of the 24 major and minor keys. The model estimates the most likely key sequence over time based on the chord progression.

The model incorporates beat positions and downbeats to integrate the metrical structure. This integration helps to segment the piece into overlapping segments related to measures.

The model is evaluated on two sets of music pieces: classical and popular music. Various parameters, such as analysis window length and key templates, are analyzed to determine their impact on key estimation accuracy.

The proposed model provides a robust method for local key estimation by leveraging the relationship between chords and keys and incorporating metrical information, improving upon previous approaches that often relied on fixed window lengths for key estimation.

3.3.2 Approaches Utilizing Neural Networks

An approach Utilizing Convolutional Neural Networks

In the paper "Musical Tempo and Key Estimation using Convolutional Neural Networks with Directional Filters" [19], the authors explore the task of key estimation from musical audio sig-

nals using convolutional neural networks (CNNs). Their primary goal is to predict the correct key for a given piece of music, focusing on major and minor modes and ignoring other possible modes like Dorian or Lydian. They frame the task as a classification problem with 24 different classes corresponding to 12 tonics in both major and minor modes.

For key estimation, the authors use constant-Q magnitude spectrograms with dimensions 168×60 as input to the network. The spectrogram covers a frequency range of 7 octaves with a resolution of two bins per semitone. Each time frame represents a duration of 0.19 seconds, resulting in 60 frames corresponding to 11.1 seconds of audio. During training, each sample is cropped to 60 frames with a random offset to ensure variability. To account for class imbalances, each spectrogram is randomly shifted along the frequency axis by {-4, -3, ..., 6, 7} semitones, and the ground truth labels are adjusted accordingly. The spectrograms are then normalized to have zero mean and unit variance.

The authors employ two different CNN architectures for the task: a shallow architecture inspired by classic signal processing approaches and a deeper VGG-style architecture commonly used in computer vision. The shallow architecture, referred to as ShallowMod, consists of short directional filters followed by one-dimensional average pooling layers orthogonal to the short filters. This is followed by a layer with long directional filters that stretch in the same direction as the short filters. The classification module, ClassMod, uses global average pooling and softmax activation to output the final class probabilities. ReLU activation functions are used for the convolutional layers, and dropout layers are added to prevent overfitting.

The deep architecture, referred to as DeepMod, uses multiple layers of convolutional filters with batch normalization and max-pooling layers. This architecture is more complex and designed to capture high-level features from the spectrograms. Similar to the shallow architecture, dropout layers are used to prevent overfitting, and ReLU activations are employed.

The models are trained on a dataset of musical pieces labeled with their corresponding keys. The dataset is augmented by scaling the spectrograms along the time axis and adjusting the ground truth labels accordingly. Each model variant is trained five times, and the mean validation accuracy along with its standard deviation is recorded.

The authors evaluate the models on multiple test sets, including GTzanKey, GiantStepsKey, and LMDKey. The best performing model variant for each architecture is selected based on validation performance and tested on these datasets to measure accuracy.

The results indicate that both shallow and deep architectures perform well on the key estimation task, with the deep architecture generally achieving higher accuracy. The shallow architecture with directional filters shows promising results, particularly in scenarios with limited model capacity.

In conclusion, the study demonstrates that CNNs with directional filters can effectively perform key estimation from musical audio signals. The shallow architecture, inspired by signal processing techniques, offers a computationally efficient alternative to deeper architectures while maintaining competitive accuracy. Future work will explore further enhancements to the models and their application to other MIR tasks.

An approach Utilizing Transformers

The paper "MAP-Music2Vec: A Simple and Effective Baseline for Self-Supervised Music Audio Representation Learning" [20] , introduces Music2Vec, a self-supervised learning (SSL) framework designed for music audio representation learning. While the primary focus of the study is not solely on key detection, this task is included as part of a broader suite of music information retrieval tasks.

In the key detection process, data preprocessing involves using audio length cropping to handle music excerpts. This step is essential as longer sequences pose significant challenges for modeling. By cropping audio clips, the authors maintain a consistent batch size while increasing the number of music samples per training batch.

For key detection, the authors utilize a model architecture based on the Data2Vec framework. This model encodes audio recordings using a multi-layer 1-D CNN feature extractor, which maps a 16 kHz waveform to 50 Hz representations. These encoded tokens are then input into a 12-layer Transformer with a hidden dimension of 768, and a feed-forward inner-dimension four times the hidden dimension. Unlike models pre-trained on speech, which show limited benefits for music representation learning, this model is trained from scratch to verify its effectiveness on music audio recordings.

The dataset for training consists of approximately 130,000 hours of music audio files collected from the Internet, with a 1,000-hour subset containing 30-second long wave files used for model training. The Music2Vec models are trained for 400,000 steps using 8 NVIDIA A100-40GB GPUs, with training taking approximately six days.

Evaluation of the model's performance on key detection is conducted using multiple test sets, including the GTzanKey and GiantStepsKey datasets. The results indicate that while the Music2Vec model is not exclusively optimized for key detection, it achieves competitive accuracy. Notably, the CNN representations sometimes outperform the Transformer layers, particularly for key detection. This suggests that local features, which are similar to a bag-of-words approach, play a significant role in the model's performance on this task.

The study concludes that the Music2Vec framework, despite being designed for general music audio representation learning, demonstrates effective performance in key detection. The findings highlight the potential of using SSL frameworks for music information retrieval tasks, although there remains room for improvement in capturing long-range contextual information. Future work may explore further optimization and fine-tuning to enhance performance on specific tasks like key detection.

Chapter 4

Methodology

4.1 Research Design

The initial goal of our research consisted of two clear objectives. First, we aimed to identify a state-of-the-art Machine Learning approach for solving the problem of Key Detection. Second, we sought to improve upon this approach. During our literature review, we made several key observations that highlighted the complexities of identifying a state-of-the-art system based solely on reported results. Specifically, we found that:

i. The accuracy of a Key Detection system varies significantly depending on the type of data used for training and evaluation.

ii. There was insufficient overlap between datasets used in the studies we examined, preventing direct comparison of their results.

iii. The standard metrics used for evaluating these systems do not fully capture what constitutes a ”good” Key Detection system.

4.1.1 Main Objectives

Based on these observations, we re-framed our initial objectives to better align with what was feasible. Instead of focusing solely on identifying a state-of-the-art system, our new approach aimed to;

i. Find a versatile machine learning solution capable of robust performance across various datasets.

ii. Compare this selected solution with a standard non-ML algorithm to assess the added complexity’s impact on performance improvements.

iii. Design and evaluate our own machine learning architecture, comparing its effectiveness with the other two systems mentioned.

4.1.2 A Comprehensive Roadmap

We structured our revised objectives into a concrete plan consisting of the following steps:

1. **Data Collection**: Gather diverse datasets to adequately represent the variability in input formats.

2. **Choice of Evaluation Metrics**: Select metrics that effectively capture the predictive performance.

3. **Traditional Algorithm Selection**: Implement a non-ML algorithm for key detection to serve as a baseline for comparison.

4. **Machine Learning Approach Selection**: Identify and implement a promising machine learning method capable of handling varied datasets.

5. **Design of Novel Architecture**: Develop a new architecture aimed at enhancing performance.

6. **Results Comparison**: Conduct a comprehensive comparison of outcomes generated by the aforementioned systems.

4.2 Data Collection

Three distinct datasets, each representing different input formats and musical genres, were utilized to ensure a comprehensive evaluation of our machine learning models across various types of music data. Utilizing these datasets allowed us to expose our models to a wide range of musical styles and data representations, testing their ability to generalize across different musical contexts. Each dataset was processed to extract relevant features necessary for the key detection task, with feature extraction techniques adapted to the specific characteristics of XML, MIDI, and MP3 formats. This multi-format and multi-genre approach not only tested the robustness of our machine learning models but also provided valuable insights into the performance variations across different types of musical data. In the following chapters, we will provide a detailed summary of each dataset used, including a description of the feature extraction process and any additional data processing steps taken to prepare the data for our algorithms. .

4.2.1 Music21 Corpus

- **Format**: XML

- **Genre**: Classical

Music21 is a Python toolkit designed for computer-aided musical analysis and computational musicology. It includes the music21 core corpus, a diverse collection of musical scores in various formats. This toolkit enables users to access and analyze musical data conveniently for both musicological and computational research purposes.

Labelling and Feature Extraction

Using the functionalities provided by Music21, we automated the segmentation of each musical score into parts where the key remained constant. We then extracted the key labels embedded within the scores by their creators and also computed the Pitch Class Profile (PCP) for each segment. This process allowed us to create a dataset comprising of 54,445 PCPs labeled by their corresponding keys. The class distribution of this dataset is illustrated in figure 4.1.

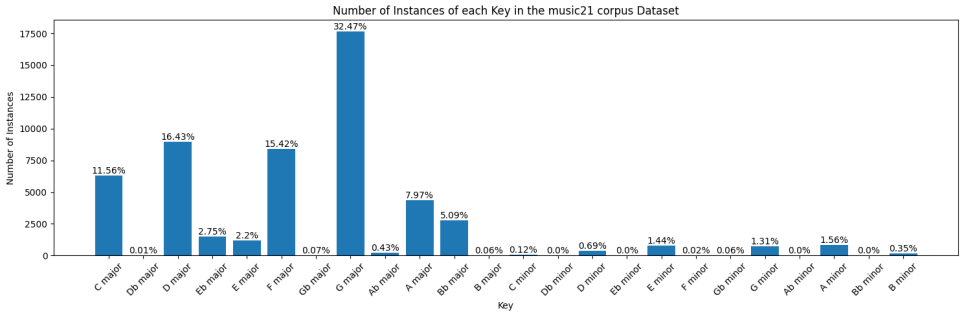

Figure 4.1: Class distribution of the Original Dataset

We subsequently refined our dataset by removing entries where the key label was absent. Additionally, we excluded entries with Pitch Class Profiles that exhibited uniformity, suggesting segments composed solely of rests or brief chromatic passages. This resulted in a reduction of the dataset to a size of 21,720 entries. The class distribution of the updated version of dataset is illustrated in figure 4.2.

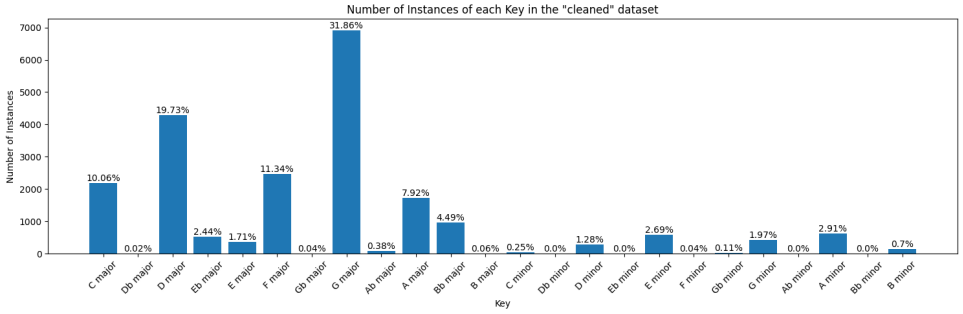

Figure 4.2: Class distribution of the Updated Dataset

Data Augmentation

Despite obtaining a total of 21,720 entries, the highly unbalanced class distribution highlighted the need to incorporate data augmentation techniques to improve class representation. Fortunately, generating new valid entries from existing ones is straightforward because we can simply transpose musical pieces from one key to another key within the same mode. Additionally, we can perform this transposition at the PCP level by simply circularly shifting the PCP vector by the appropriate number of semitones.

In practice, we leveraged this feature property by transposing all 19,560 Major Key entries to C major and all 2,160 Minor Key entries to C minor. Afterwards, we systematically transposed each Minor Key entry to every minor key. To rectify the imbalance between major and minor entries, we randomly selected 2,160 entries from the total Major Key entries for transposition

into each Major Key class. This approach allowed us to achieve a perfectly balanced dataset of 51,840 entries without any repetition within the same class.

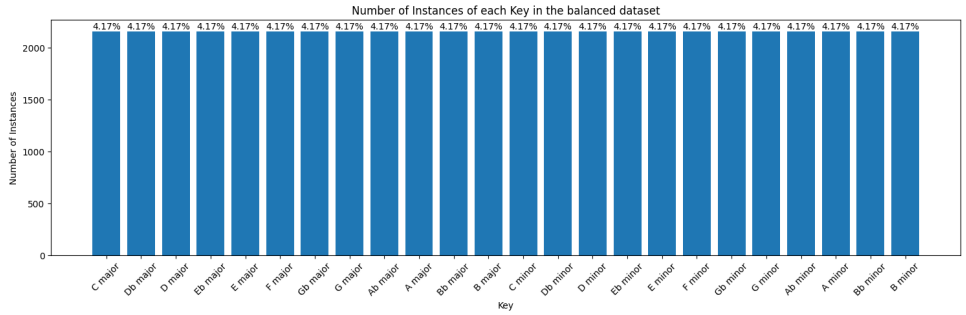

Figure 4.3: Class distribution of the Balanced Dataset

In conclusion, we successfully created a perfectly balanced dataset for our 24-class classification task. We have complete confidence in the accuracy of the labels, as they were directly extracted from transcribed music and not inferred in any manner.

4.2.2 Lakh MIDI Dataset

- **Format**: MIDI
- **Genres**: POP, ROCK, FOLK

The Lakh MIDI dataset is a collection of 176,581 unique MIDI files, 45,129 of which have been matched and aligned to entries in the Million Song Dataset.

Labelling and Feature Extraction

Once again for each data entry we computed a Pitch Class Profile to later use as input for our algorithms. We achieved this with the use of the ”pretty midi” python package. The metadata containing the key in which each entry corresponded to was found through the Million Song Dataset. We discarded any entries whose key label had a confidence score under 0.95, leaving as with a total number of 992 entries. Since this dataset was used only for evaluation purposes we did not proceed into balancing the dataset. The class distribution is shown in figure 4.4

4.2.3 GiantSteps Dataset

- **Format**: mp3
- **Genres**: EDM

The Giant Steps dataset contains 604 entries of 120 second EDM audio files. The key annotation was extracted from forum entries of people correcting the key annotations.

Labelling and Feature Extraction

The feature extraction process for this dataset included some digital signal pre-processing. For each audio file, we calculated its chromogram using the ”librosa” Python package. Next, we applied a simple denoising technique to eliminate artifacts produced by percussive sounds,

distortion and overtones. Finally, we computed the mean value of each pitch class in the chromogram, resulting in a PCP vector. This procedure is detailed in Section 2.2.2. Each entry was clearly labeled according to its key, so no additional steps were required. Since this dataset was used solely for evaluation purposes, we did not balance it. The class distribution is shown in Figure 4.4.

As previously discussed, we chose to extract only one type of feature from every dataset used, namely, the Pitch Class Profile (PCP). This decision is directly related to the category of machine learning models we preferred to examine in this study. The rationale behind this choice will be further explained in Section 4.4. Additionally, it is worth noting that the Music21 dataset is substantially larger than the other two. Therefore, we decided to use it as our primary dataset for training and testing our algorithms. The Lakh and GiantSteps datasets were used to evaluate how well our algorithms could generalize their key detection capabilities to slightly different domains. This division of functions among the datasets also explains why we did not balance the Lakh and GiantSteps datasets, as we wished to test our algorithms in a more realistic scenario in terms of class imbalances.

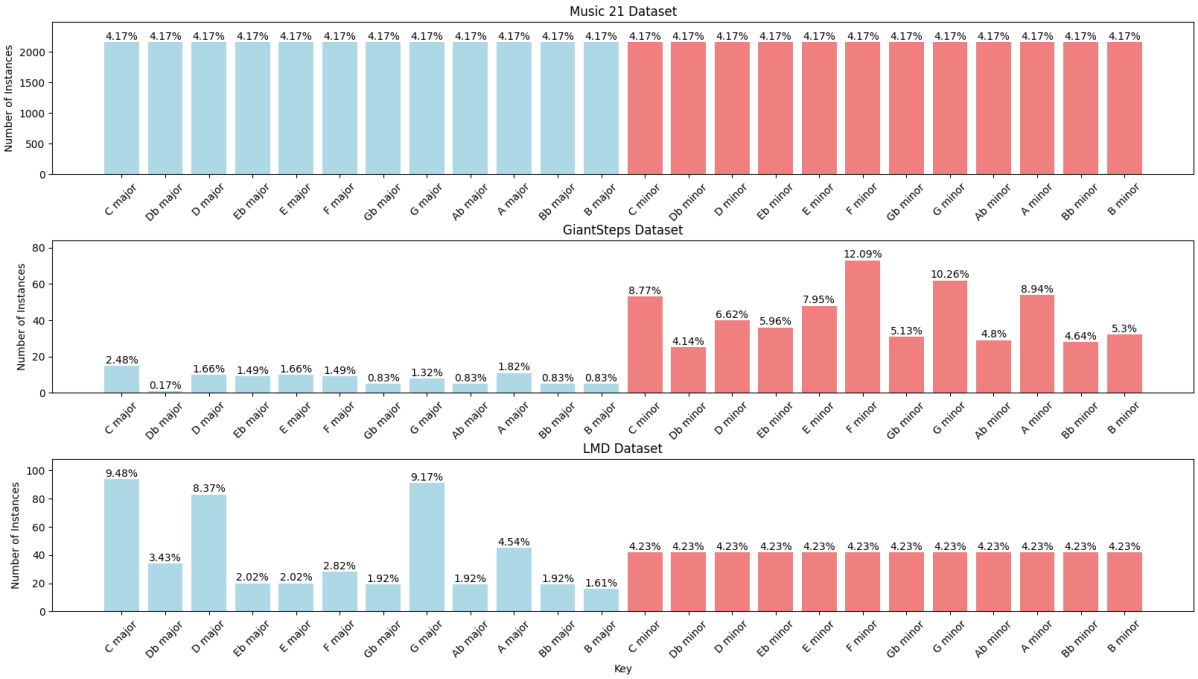

Figure 4.4: Class distributions of all Three Datasets

4.3 Evaluation Framework

The diversity of our datasets allowed us to draw conclusions beyond the general performance of each algorithm in a restricted scenario. Specifically, we designed our evaluation methods to quantify the answers to the following questions:

i. How accurate are the predictions produced by our algorithms when trained and evaluated on the same dataset?

ii. In what ways do our algorithms fail to produce accurate predictions?

iii. Why and in which scenarios do some algorithms perform better than others?

iv. How reliable are the performance scores obtained from one dataset when the algorithms are tested on different datasets?

With these questions in mind, we selected metrics and devised tests capable of providing satisfactory answers.

4.3.1 Traditional Metrics

Since we are dealing with a multiclass classification problem, we calculated the accuracy, precision, recall, and F1 score in each round of evaluation. Each of these metrics provided valuable insights into the behavior of our algorithms, and our evaluation would be incomplete without them.

Accuracy

Accuracy is the most intuitive of the aforementioned metrics and serves as a great indicator of the overall competence of each algorithm. It is defined as the ratio of correctly predicted instances to the total instances in the dataset. In cases where the dataset used for evaluation is balanced, accuracy is sufficient to draw a conclusion about how "good" our key detector is. However, in scenarios where the class distribution is imbalanced, accuracy alone might be misleading.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positives. It is defined as:

$$\text{Precision} = \frac{TP}{TP+FP}$$

where TP is the number of true positives and FP is the number of false positives. Precision is crucial in scenarios where the cost of false positives is high, as it tells us how many of the predicted positive instances were actually correct. This metric becomes particularly important when evaluating imbalanced datasets, as it helps ensure that the algorithm is not biased towards the majority class and is accurately identifying the minority class.

Recall

Recall, also known as sensitivity or true positive rate, is the ratio of correctly predicted positive observations to all the observations in the actual class. It is defined as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

where FN is the number of false negatives. Recall is important in situations where the cost of false negatives is high, as it indicates how well the algorithm identifies all relevant instances within a class.

F1 Score

The F1 score is the harmonic mean of precision and recall, providing a single metric that balances both concerns. It is defined as:

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

The F1 score is particularly useful when we need to balance precision and recall and is a more reliable metric than accuracy in cases of imbalanced class distributions.

Each of these traditional metrics provided a different perspective on the performance of our algorithms, enabling us to conduct a comprehensive evaluation of their effectiveness across various scenarios.

4.3.2 Custom Metrics

In order to further examine the behavior of our algorithms, we also included some custom metrics that provide deeper musical insights into the predictions we observed. Specifically, we wanted to understand how often our predictions were close to a correct answer in a musical sense. For that reason we devised the following four metrics:

1. **Off by Relative Key** Measuring how often the prediction corresponds to either the relative minor or the relative major of the actual key
2. **Off by a Fifth** Measuring how often the prediction corresponds to a key whose key signature has a one-alteration-difference from the actual key
3. **Off by mode** Measuring how often the prediction corresponds to a key with the correct tonic but of the opposite mode
4. **Completely off** Measuring how often the prediction has no musical relevance to the actual key

Essentially these metrics help us make sense of the confusion matrix produced from the evaluation of our algorithms. An example of an annotated confusion matrix is given below.

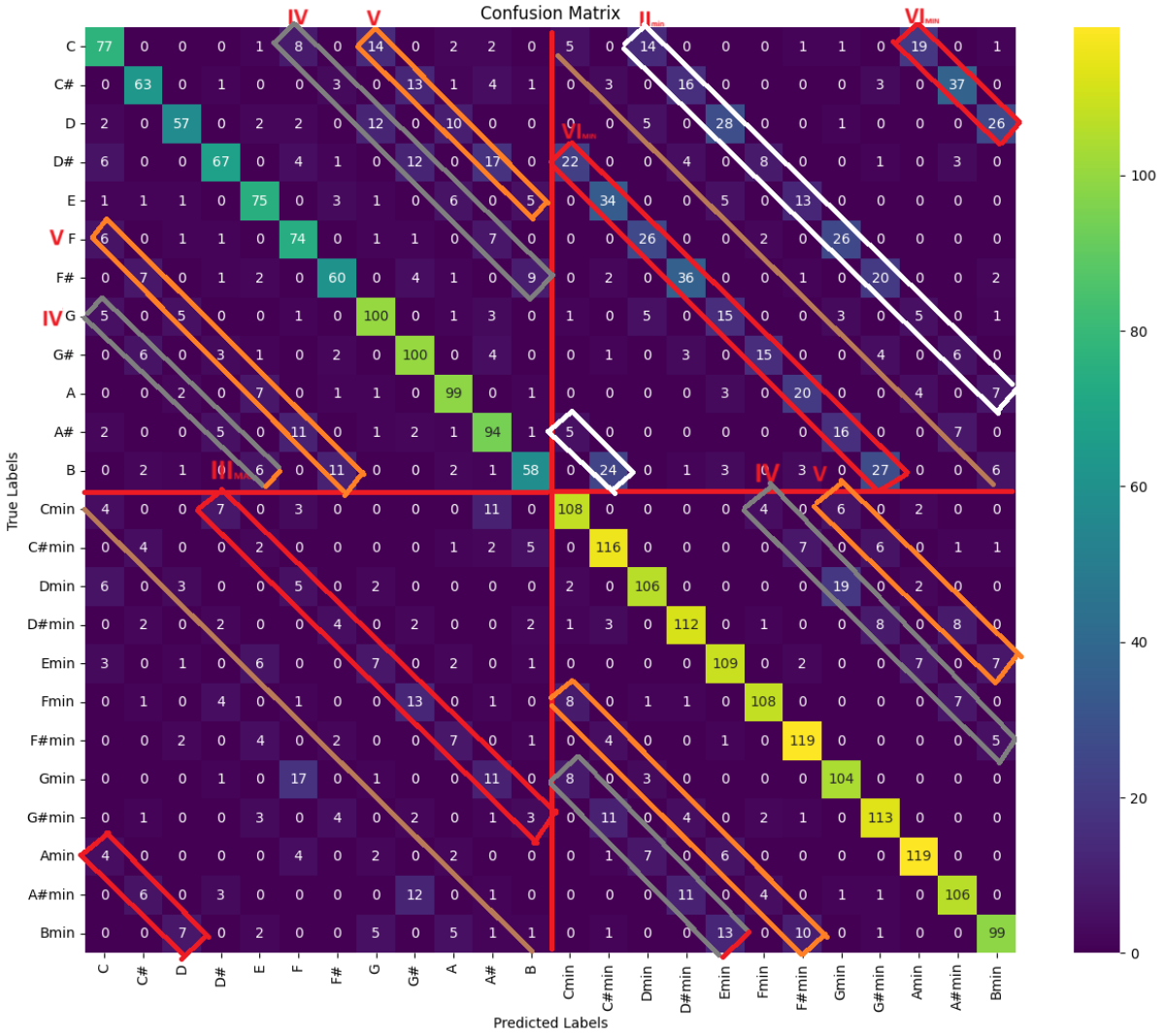

Figure 4.5: Confusion Matrix produced by evaluating Naive Bayes on the Music 21 Dataset

4.3.3 Evaluation Scenarios

To gain a comprehensive understanding of our algorithms' classification capabilities, we evaluated each of them in two distinct scenarios, each designed to offer a different perspective on the efficacy of the classifiers.

Scenario 1 : Training and Evaluating on the same Dataset

This scenario is the classic way of evaluating machine learning models. We first split the dataset into training and testing sets, representing 60% and 40% of the original dataset, respectively. Next, we train our models on the training set and evaluate them on the test set. This type of scenario helps us understand how well our models can learn the desired patterns hidden within the features that indicate the class of each entry. However, it is limited in that it is difficult to judge how well the algorithms can generalize their discriminatory properties. We implemented this type of evaluation using the Music21 dataset, which has a large number of entries.

Scenario 2: Training and Evaluation on Separate Datasets

In practical applications, a Key Detection system is typically deployed for inference on input data that may differ from its training data. To simulate this scenario and assess our algorithms' generalizability, we trained our models using the Music21 dataset and subsequently evaluated them on the GiantSteps and Lakh datasets. This approach allows us to gauge the reliability of our models in real-world applications where data distributions may vary.

4.4 Choice of Algorithms

The accumulation of a diverse set of key-annotated musical pieces, along with our chosen metrics and evaluation scenarios makes for a clear evaluation framework capable of producing confident conclusions about the effectiveness of various key detection algorithms. Through the literature review process, we came across numerous approaches for determining the key of a musical piece, and it fell upon us to decide which of them we would put to the test.

4.4.1 Establishing A Traditional Baseline

Our initial objective was to evaluate a non-machine learning-based method for key detection, which we referred to as the 'traditional approach.' This served as our baseline for comparing the performance of subsequent machine learning algorithms. The term 'traditional' is used here with a touch of whimsy, acknowledging that Krumhansl and Shepard's method, based on Pitch Class Profiles, is a seminal work in music theory. Their pioneering approach predates modern machine learning techniques, reflecting early efforts to quantify musical harmony objectively. By evaluating their method alongside ML-algorithms, we sought to assess its enduring relevance and effectiveness in computational music analysis.

A detailed description of the experiments performed by Krumhansl and Shepard is given in section 3.2. The results of these experiments where two Key Profiles, one for minor keys and one for major keys.

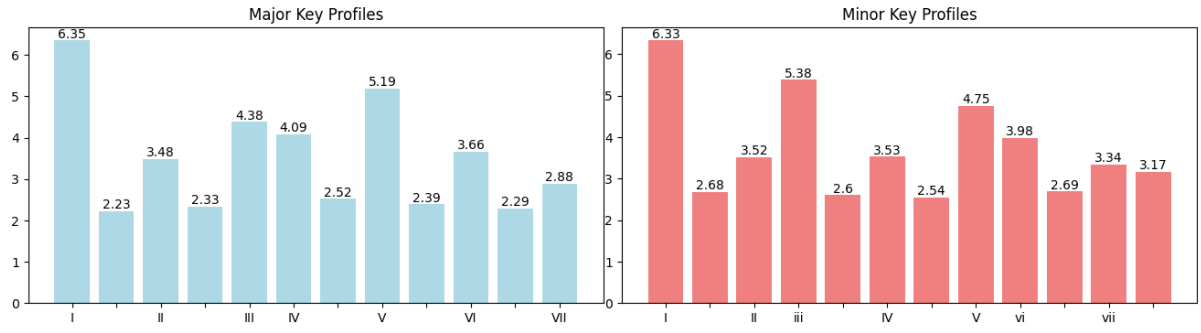

Figure 4.6: The Key Profiles Generated from the Krumhansl and Shepard Experiments

The key detection algorithm proposed by Krumhansl and Shepard operates on the principle that the Pitch Class Profile (PCP) of a musical piece composed in a particular key should exhibit the highest positive correlation with the Key Profile corresponding to that key. The Pearson's correlation coefficient r_{xy} measures the correlation between two vectors x and y and is computed as follows:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where:

- x_i and y_i are the individual data points,
- $\bar{x}$ and $\bar{y}$ are the means of x and y, respectively,
- n is the number of data points.

In our case x represents the Key Profile while y is the PCP of the musical piece. x_i, y_i correspond to the value of the i_{th} Pitch Class of the Key Profile and the PCP, respectively. We implemented this algorithm and evaluated it according to our evaluation framework. The results of this evaluation are shown in chapter 5 and discussed in chapter 6

4.4.2 Implementing Machine Learning Models

In order to choose which of the machine learning approaches we encountered during our literature review, we would test for, we considered the following three criteria:

1. **Reported Results**: We evaluated the reported performance metrics (such as accuracy, precision, recall) from existing studies to gauge the effectiveness of each approach in key detection tasks.
2. **Feature Extraction Requirements**: We assessed the complexity and feasibility of extracting necessary features (e.g., Pitch Class Profiles) required by each approach from our musical datasets.
3. **Implementation Complexity**: We considered the technical requirements and resource constraints (such as computational power, memory, and training time) associated with implementing and running each machine learning approach.

Table 4.1 provides a simplified view how the various models we examined fit the aforementioned criteria. SSM's refer to simple statistical models (e.g Naive Bayes classifier) while Max Accuracy refers to the maximum accuracy value we could find for the task of key-detection regardless of the dataset used. Detailed descriptions of how the models were parameterized, trained and evaluated are provided in chapter 3.

	SSM's	**HMM's**	**CNN's**	**Transformers**
Max Accuracy	91.47%	80.21%	71.30%	50.8%
Feature Type	PCP's	Chords	Spectrograms	Token sequences
Implementation Complexity	LOW	HIGH	MEDIUM	HIGH

Table 4.1: Comparison of Different Model Types

We decided to focus our research on Simple Statistical Models. Our decision to employ Simple Statistical Models for our experiments was driven by several key factors:

- Promising Results: Previous studies have shown that these models can achieve promising results in key detection tasks.
- Simplicity of Feature Requirements: These models require only the Pitch Class Profile (PCP) of the input data. This makes them versatile and easy to apply to any input format, as PCP extraction is straightforward regardless of the format.
- Resource Efficiency: Simple Statistical Models can be trained and evaluated with minimal computational resources, allowing for extensive experimentation without significant overhead.

Specifically, we implemented and evaluated the following models: Naive Bayes, K-Nearest Neighbors, Support Vector Machines adapted for multiclass classification, Linear Discriminant Analysis, and a Multi-Layer-Perceptron with two hidden layers. Each model's hyperparameters were meticulously chosen through rigorous experimentation to optimize performance. For instance, the Naive Bayes classifier was configured with default parameters as it exhibited robust performance across various datasets without significant sensitivity to parameter variations. The K-Nearest Neighbors model was fine-tuned with a choice of 5 neighbors and using the correlation metric after extensive testing showed it to balance computational efficiency and accuracy. Our Support Vector Machine (SVM) approach utilized a polynomial kernel of degree 2, selected after comparative analysis demonstrated superior classification results for our multiclass problem. Similarly, the Linear Discriminant Analysis (LDA) model was parameterized with 'lsqr' solver and automatic shrinkage, based on its superior handling of the dataset's covariance structure. Lastly, our Multilayer Perceptron (MLP) neural network was optimized with two hidden layers of 24 neurons each, employing the 'relu' activation function and 'adam' solver, with careful adjustment of parameters like learning rate and batch size through iterative testing to achieve optimal convergence and generalization. Each decision reflected our commitment to empirical validation and yielded models primed for accurate and efficient classification tasks.

4.4.3 Designing our Novel Architecture

Core Concept

As a final aspect of our research, we ventured into designing a novel machine learning model for key detection, inspired by the insights we gained throughout the process up to this point. The core concept driving our design was to develop a model that seamlessly integrates the idea of Key Profiles (the "traditional" approach) within a machine learning framework.

A critical observation

Krumhansl and Shepard's method essentially compares the Pitch Class Profile (PCP) of a musical piece with two predefined expected patterns. This pattern matching is conceptually similar to the operations performed by the convolutional layers of a Convolutional Neural Network (CNN). Specifically, the set of Pearson correlation coefficients used by Krumhansl and Shepard's algorithm to determine how closely a key profile resembles the PCP of a piece is mathematically equivalent to the cross-correlation of the standard score normalized Key Profile and the PCP. It is important to note that cross-correlation, unlike convolution, does not invert any of the functions involved but it is essentially the same operation.

Mathematical Proof

The Standard Score of a sample $x[i]$ is given by

$$z_x[i] = \frac{x[i] - \bar{x}}{S}$$

where:

- $\bar{x}$ is the mean of the population,
- S is the standard deviation of the population,

$$S = \sqrt{\frac{\sum_{i=0}^{n-1}(x[i] - \bar{x})^2}{n-1}}$$

Thus, we can rewrite the Pearson coefficient as follows:

$$r_{xy} = \frac{1}{n-1} \cdot \sum_{i=0}^{n-1} z_x[i] \cdot z_y[i]$$

where:

- z_x is the standard score normalization of the Pitch Class Profile,
- z_y is the standard score normalization of the Key Profile.

Krumhansl and Shepard's algorithm circularly shifts the key profile by τ to compute the correlation of the PCP to the profile corresponding to each Tonic. Thus,

$$r_{xy}[\tau] = \frac{1}{n-1} \sum_{i=0}^{n-1} z_x[i] \cdot z_y[(i+\tau) \mod n] = \frac{1}{n-1}(z_x \star z_y)[\tau]$$

where τ is the number corresponding to the pitch class of the Tonic. We can ignore the scaling factor $\frac{1}{n-1}$ for our purposes. With this proof we have shown that it is possible to implement the Krumhansl and Shepard's algorithm with the use of a convolutional layer in our model.

Model Summary

Layer (type)	**Output Shape**	**Param #**
Z-normalization (Lambda)	(None, 23, 1)	0
Correlation (Conv1D)	(None, 12, 2)	26
Permutation (Permute)	(None, 2, 12)	0
Reshaping (Reshape)	(None, 24)	0
Dense (Dense)	(None, 24)	600
Total params		**626 (2.45 KB)**
Trainable params		**626 (2.45 KB)**
Non-trainable params		**0 (0.00 Byte)**

Table 4.2: Model summary for our custom model

As seen in Table 4.2, we included the necessary layers to normalize the input and reshape the output of the convolutional layer. Additionally, we included a dense output layer. This layer

serves as the final decision-maker in the model's decision process, determining the contribution of each correlation coefficient to the final result.

We initialized the Conv1D parameters with the z-normalized values of the Krumhansl and Shepard's Key Profiles, and the weight matrix of the dense layer was initialized as a 24x24 identity matrix. Through this initialization and without further training, the model executed the Krumhansl and Shepard's algorithm with absolute precision, thereby achieving our goal.

We trained and evaluated this model alongside the rest of our chosen algorithms. The results of this evaluation are presented in the next chapter.

Chapter 5

Results

5.1 Training and Evaluation within the Music21 Dataset

Metric	KS	NB	KNN	LDA	**SVM**	MLP	CUSTOM
Accuracy	0.7025	0.7798	0.8088	0.7962	0.8220	0.8045	0.8028
Recall	0.7026	0.7799	0.8088	0.7962	0.8222	0.8045	0.8029
Precision	0.7026	0.7831	0.8093	0.7964	0.8226	0.8062	0.8033
F1 Score	0.7025	0.7804	0.8088	0.7962	0.8222	0.8047	0.8028
Relevant Key	0.0609	0.0890	0.0638	0.0715	0.0616	0.0712	0.0738
Off by a Fifth	0.1130	0.0609	0.0551	0.0561	0.0531	0.0534	0.0511
Wrong Mode	0.0283	0.0223	0.0225	0.0244	0.0228	0.0250	0.0242
Completely Off	0.0953	0.0480	0.0498	0.0518	0.0405	0.0459	0.0481

Table 5.1: Performance on the Music21 Dataset

The above results correspond to the first round of evaluation. We trained our classifiers on 60% of the music21 dataset and evaluated them on the remaining 40%. There were no significant class imbalances in either the training or test sets. All machine learning models significantly outperformed the Krumhansl and Shepard algorithm. The best performance was achieved by the SVM classifier, correctly identifying the key of 82.2% of musical pieces and making a musically relevant prediction 96% of the time, as indicated by the "Completely Off" score. Overall, all models demonstrated similar behavior in differentiating between the actual key and closely related ones, as evidenced by the last four metrics in Table 5.1.

5.2 Evaluation on the Lakh and GiantSteps Datasets

The second round of evaluation produced interesting results. To test our algorithms in a "production-like" scenario, we did not retrain the models on either the Lakh or GiantSteps datasets. Instead, we left the models unchanged after the first round of evaluation and used them solely to produce predictions on the entirety of the newly introduced datasets.

To adequately understand the performance of our algorithms in this evaluation round, we focused primarily on Accuracy, Recall, and the Completely Off score. Additionally, we emphasize the importance of comparing our model results to the traditional, non-ML Krumhansl and Shepard algorithm. This comparison highlights the value and effectiveness (or lack thereof) of using machine learning for our classification task.

Metric	KS	NB	KNN	LDA	SVM	MLP	**CUSTOM**
Accuracy	0.6855	0.6784	0.6129	0.6835	0.6784	0.6925	0.7006
Recall	0.6807	0.6765	0.6128	0.6791	0.6780	0.6924	0.7005
Precision	0.6725	0.6504	0.5845	0.6503	0.6474	0.6676	0.6763
F1 Score	0.6569	0.6503	0.5759	0.6556	0.6490	0.6647	0.6760
Relevant Key	0.1290	0.1552	0.1633	0.1048	0.1230	0.1240	0.1159
Off by a Fifth	0.1391	0.1200	0.1371	0.1472	0.1371	0.1260	0.1230
Wrong Mode	0.0161	0.0161	0.0272	0.0212	0.0181	0.0171	0.0192
Completely Off	0.0302	0.0302	0.0595	0.0433	0.0433	0.0403	0.0413

Table 5.2: Performance on the Lakh MIDI Dataset

The Lakh MIDI dataset contained musical pieces from a different genre than the one our models were trained on. This alone caused the NB, KNN, LDA, and SVM algorithms to perform worse than the non-machine learning Krumhansl and Shepard algorithm by every metric. The two models that "survived the KS-Test" in this scenario were the Multilayer Perceptron and our Custom model, but even these performed worse in terms of the Completely Off score. Evaluating our models on the Lakh Dataset clearly showed that the performance the algorithms achieved when tested with the same type of data they were trained on is misleading when one tries to distinguish which algorithm works best in general.

Metric	KS	NB	KNN	**LDA**	SVM	MLP	CUSTOM
Accuracy	0.3841	0.3030	0.3808	0.5248	0.3990	0.4272	0.4470
Recall	0.3774	0.3442	0.3526	0.4163	0.3265	0.4171	0.4738
Precision	0.3514	0.3226	0.3632	0.4139	0.3407	0.3716	0.4199
F1 Score	0.3163	0.2691	0.3127	0.4035	0.3130	0.3461	0.3807
Relevant Key	0.0679	0.1490	0.1076	0.0762	0.0861	0.0795	0.0977
Off by a Fifth	0.1705	0.1258	0.1623	0.1457	0.1921	0.1772	0.1258
Wrong Mode	0.1225	0.1026	0.0712	0.0331	0.0927	0.0646	0.0960
Completely Off	0.2550	0.3195	0.2781	0.2202	0.2301	0.2517	0.2334

Table 5.3: Performance on the GiantSteps Dataset

The GiantSteps dataset, which contained EDM tracks, presented an even greater genre divergence from classical music compared to the Lakh dataset. Additionally, this dataset included more tracks in a minor key, causing our algorithms to make the "Wrong Mode" error more frequently, revealing a bias towards major modes. The algorithms that outperformed the Krumhansl and Shepard (KS) algorithm in this case were the LDA, CUSTOM, MLP, and SVM, with the LDA achieving significantly better results than the others. Our Custom model also produced competent results, surpassing LDA in the recall metric.

Overall, the results indicate that the algorithms demonstrating a clear advantage over the Krumhansl and Shepard (KS) algorithm, and thus can be considered improvements, were the Linear Discriminant Analysis and our Custom Model. However, none of the systems exhibited impressive performance when tested on data outside of their training set. In the next chapter, we will delve into the reasons behind this and propose approaches that we believe could yield significantly better results, paving the way for more robust and generalizable key detection algorithms.

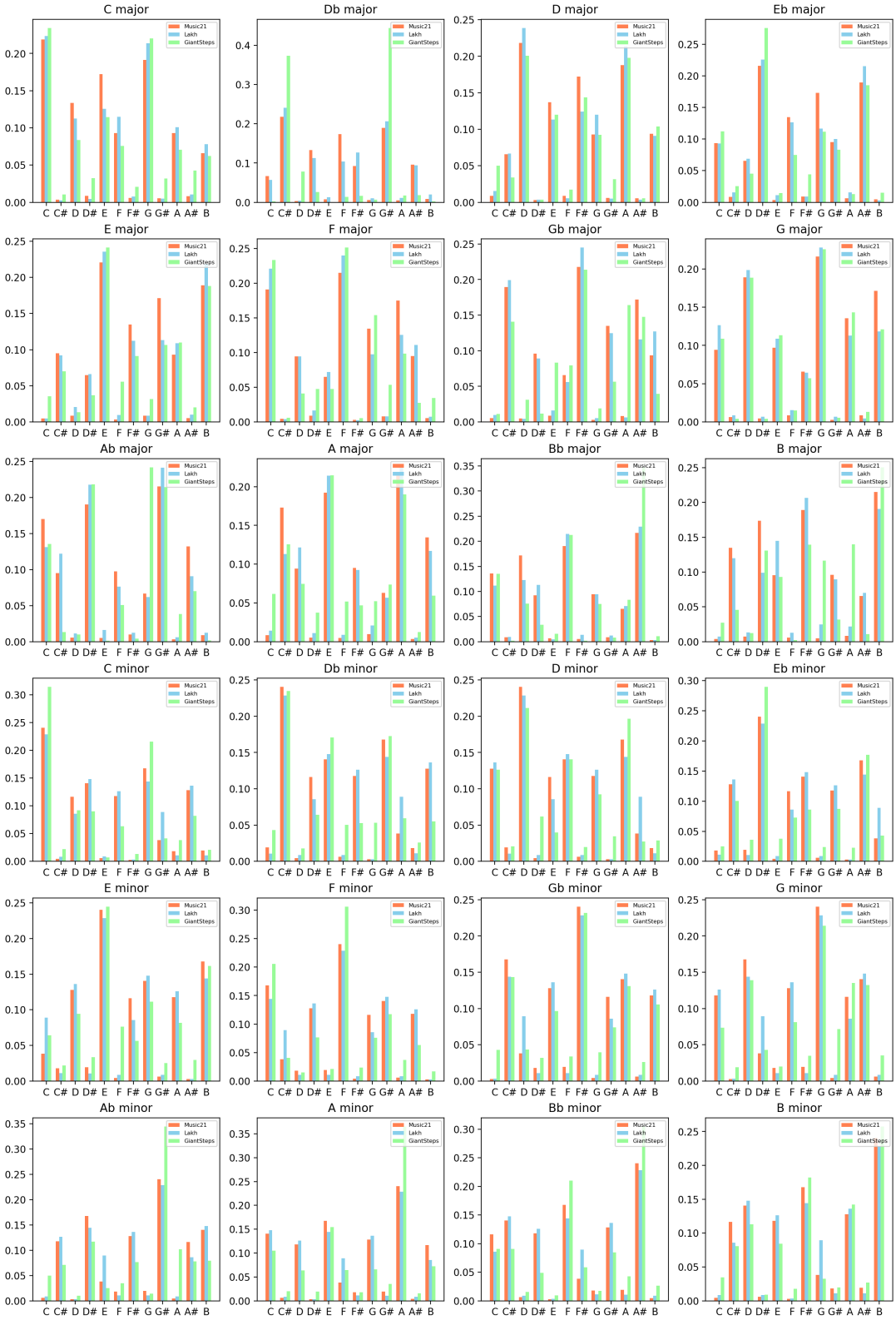

Figure 5.1: Average PCP for each class of all Three Datasets

Chapter 6

Discussion

6.1 Answering the Four Main Questions

Before delving into a detailed discussion of the various topics that emerged from our experiments, we will address the four key questions our evaluation framework was designed to answer.

How accurate are the predictions produced by our algorithms when trained and evaluated on the same dataset?

We can confidently state that all the Machine Learning algorithms we evaluated will correctly classify **8 out of 10** musical pieces, provided they have been trained on similar data. For the misclassifications, most of the predictions will still be musically relevant.

In what ways do our algorithms fail to produce accurate predictions?

It seems that the aspect that most confuses our algorithms is determining the **Tonal Center** (Tonic) of the piece. We arrive at this conclusion by examining the *Off by Relevant Key* the *Off by Fifth* and the *Off by Mode* metrics. We observe that it is common for our algorithms to correctly identify the Key Signature of the piece but fail to determine which note acts as the tonal center. Conversely, correctly identifying the Tonic and predicting the wrong mode is a far less common occurrence.

Why and in which scenarios do some algorithms perform better than others?

In the **restricted scenario**, where the training data closely match the evaluation data, and with the aid of polynomial expansion, our classes became nearly linearly separable. This is why **Support Vector Machines** achieve the best results in this case. **K-Nearest Neighbors** also performs well in this situation, as it classifies musical pieces based on their similarity to those whose class is already known.

On the other hand, in a **production-like scenario**, one cannot be certain whether a piece will closely match enough of the pieces the algorithms have been trained on in terms of harmonic context. In this scenario, algorithms that have managed to encode more fundamental properties of what makes a PCP belong to a particular key clearly outperform the two previously mentioned algorithms. This is why **Linear Discriminant Analysis** and our **Custom Model** achieve better results in this case.

How reliable are the performance scores obtained from one dataset when the algorithms are tested on different datasets?

At best, we can conclude that the models we tested will make an **educated guess** when confronted with musical pieces that significantly differ harmonically from those they were trained on. They exhibit greater precision than Krumhansl and Shepard's method when trained on sufficient data. However, from the perspective of a novice musician trying to determine the appropriate scale for their favorite song, these models will disappoint approximately 3-4 out of 10 times. This level of reliability is insufficient.

6.2 The Symmetrical Property of our Features

The first topic of discussion that merits our attention is how effectively our various algorithms can leverage the inherent symmetry in Pitch Class Profiles to make more accurate predictions.

The symmetrical property a Pitch Class Profile is this:

A Major Pitch Class Profile will remain Major no matter by how match we circularly shift it. The same is true for Minor Pitch Class Profiles repectively. That means that given a PCP that belongs to classes 0 - 11 (Major) our algorithms should classify it to one of these classes even if we circularly shift its values by any amount. As such this is a type of rotational symmetry.

We have already exploited this symmetrical property during our efforts to ensure that the Music21 dataset is completely balanced. Our algorithms have been trained on this balanced dataset and because of that all of them do exploit this symmetrical property in some extend.

Nevertheless, some algorithms are able to inherently exploit this symmetry to make better predictions. The Krumhansl algorithm is built based on this property and that is why it utilized only two key profiles and not twenty four. The same goes for our custom model which, as we have mentioned in our methodology chapter, is designed to perform the Krumhansl algorithm when its CNN kenrels are initialized with the Krumhansl and Shepard key profiles values.

This fact alone likely explains why our custom model achieved noticeably higher recall scores than all other algorithms on both the Lakh and GiantSteps datasets. We speculate that replacing the dense layer at the end of our custom model with another convolutional layer could further exploit this property. This modification would enable our model to train even more effectively on unbalanced datasets.

6.3 A Unifying Working Principle

When examining how our various algorithms predict the key of a given Pitch Class Profile (PCP), we notice that they essentially perform the same set of actions:

For each of the 24 classes, calculate a scalar value v_c where $v_c = f_c(\mathbf{x}) \cdot \mathbf{w}_c$. Output the class whose v_c has the largest positive value:

$$\text{predicted\_class} = \arg\max\{v_0, v_1, \ldots, v_{23}\}.$$

1. $\mathbf{x}$ is the PCP vector.

2. f_c (c belongs to $[0,23]$) is a function from $\mathbb{R}^{12}$ to $\mathbb{R}^{12}$ that scales the values of the PCP vector $\mathbf{x}$ and is learned during training.

3. $\mathbf{w}_c$ is a vector learned during training whose dot product with the scaled PCP produces a score that quantifies how closely it matches the class.

This is an oversimplification in some cases, but the above description serves as a framework that allows us to compare how our algorithms differ. For example, the function f_c in Naive Bayes comprises the probability density functions of the Normal Distribution obtained by statistically analyzing the data during training. In the case of Naive Bayes, all vectors $\mathbf{w}_c$ would be filled with ones: $[1,1,1,1,1,1,1,1,1,1,1,1]$.

On the other hand, the Krumhansl algorithm can be seen as using the function f_c to standard score normalize the vector $\mathbf{x}$, and then it computes its dot product with the circularly shifted key profiles corresponding to $\mathbf{w}_c$.

What ultimately differentiates one algorithm from another is the form of the function f_c and the way its parameters, as well as the values of the vectors $\mathbf{w}_c$, are obtained during training. What we observe through our experiments is that despite the differences f_c and $\mathbf{w}_c$ might have from one algorithm to another, the behavior of the system as a key detection tool does not seem to change drastically.

6.4 The Limitations of PCP-Based Approaches

In our research, we chose to evaluate machine learning algorithms that use the Pitch Class Profile (PCP) of a musical piece to represent its harmonic content due to its simplicity and ease of extraction. However, this simplicity also presents a significant disadvantage. By ignoring the timing information of when notes are played, we limit the sophistication of our decision-making process, which ultimately imposes an upper bound on the accuracy of our algorithms. In other words, we believe that it is impossible to confidently determine the key of a musical piece solely by examining its distribution of notes.

This is not to say that our efforts are in vain. The fact that we can almost determine the key using such a simple representation of harmonic content is both interesting and useful. However, to achieve better results, algorithms like the ones we tested should be incorporated into a more comprehensive key detection system.

The tonal center in a piece of music strongly depends on the rhythmic context in which each note appears. For example, when examining the first beat of a measure, it is reasonable to expect that a root note will appear more often than a note that typically does not belong to the key.

Additionally, besides the loss of temporal information, the PCP also discards information about the specific pitch of the notes. While notes separated by octaves are perceived as similar, they are not identical and function differently. For instance, notes in lower octaves are more likely to serve as root notes of chords and thus are less likely to be outside the key in which the song is composed. Of course, this is speculative, and further data analysis is required to confirm or dispute such claims. Nonetheless, we believe there is valuable information in the specific pitches of each note that we are not capturing by solely computing the PCP of a song.

6.5 Future Work

We believe that future work should focus on approaches that utilize representations of the notes and their durations in a musical piece, retaining more information than Pitch Class Profiles (PCPs). While PCPs offer simplicity and ease of extraction, they discard significant information about the rhythmic context of each note. However, simply applying Convolutional Neural Networks (CNNs) and large Transformer models whose architectures aren't specifically tailored to this task has not proven to yield better results than the simple PCP approach.

Instead, we propose developing representations that maintain the simplicity of PCPs while encoding some rhythmic context. This could be achieved by segmenting the piece into meters, detecting the key for each meter, and then deducing the overall key of the piece by analyzing those sequences. Specifically, an approach similar to the one mentioned by Hélène Papadopoulos and Geoffroy Peeters in their 2012 paper seems promising. We suggest training 24 Hidden Markov Models (HMMs) - one for each key - where the hidden state represents the key and the observed state is the PCP of the segment. This method retains temporal information and could provide a more nuanced understanding of the musical piece. By incorporating rhythmic context into our key detection algorithms, we believe it is possible to achieve higher accuracy and robustness in real-world musical analysis.

Chapter 7

Conclusion

Our research focused on evaluating machine learning algorithms that utilized Pitch Class Profiles (PCPs) representing the note distribution of the total input musical piece. We conclude that compared to rule-based methods operating on the same input representation, certain machine learning algorithms perform better. Specifically, our novel architecture demonstrates a promising approach to improving the performance of the Krumhansl algorithm while maintaining the same operating principle. Our custom model, when evaluated, achieved higher accuracy and recall scores compared to the KS algorithm across multiple datasets. For instance, on the Music21 dataset, our custom model achieved an accuracy of 80.28% compared to the KS algorithm's 70.25%, and a recall of 80.29% compared to the KS algorithm's 70.26% .

We also conclude that PCPs are a representation that lacks essential rhythmic information needed to make confident key detection in some cases. To achieve better results, the representations used should include some temporal aspects. The tonal center in a piece of music strongly depends on the rhythmic context in which each note appears. For example, notes occurring on strong beats or at specific rhythmic positions can significantly influence the perception of the key. Incorporating features that capture these temporal dynamics would likely enhance the performance of key detection algorithms, making them more robust and accurate across a wider variety of musical styles and contexts.

In future work, integrating these temporal aspects and exploring hybrid models that combine the strengths of PCPs and rhythmic information could lead to more sophisticated and reliable key detection systems.

Chapter 8

References

Bibliography

[1] *Definition of MUSIC*, en, Jun. 2024. [Online]. Available: https://www.merriam-webster.com/dictionary/music (visited on 06/30/2024).

[2] H. Shimazu, *What is melody in music: A guide to pitch, scales, and keys*, en-US, Oct. 2022. [Online]. Available: https://splice.com/blog/an-introduction-to-melody/ (visited on 07/03/2024).

[3] *Heptatonic scale*, en, Page Version ID: 1193596250, Jan. 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Heptatonic_scale&oldid=1193596250 (visited on 07/03/2024).

[4] *Machine learning*, en, Page Version ID: 1232414476, Jul. 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Machine_learning&oldid=1232414476 (visited on 07/04/2024).

[5] *Naive Bayes classifier*, en, Page Version ID: 1227129282, Jun. 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Naive_Bayes_classifier&oldid=1227129282 (visited on 07/04/2024).

[6] O. Harrison, *Machine Learning Basics with the K-Nearest Neighbors Algorithm*, en, Jul. 2019. [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 (visited on 07/04/2024).

[7] A. Saini, *Guide on Support Vector Machine (SVM) Algorithm*, en, Oct. 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/ (visited on 07/05/2024).

[8] P. Nandi, *CNNs for Audio Classification*, en, Dec. 2021. [Online]. Available: https://towardsdatascience.com/cnns-for-audio-classification-6244954665ab (visited on 07/05/2024).

[9] *Hidden Markov model*, en, Page Version ID: 1223340736, May 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Hidden_Markov_model&oldid=1223340736 (visited on 07/05/2024).

[10] R. Ghosh, *Complete Introductory Guide to Speech to Text with Transformers*, en, Jul. 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2023/07/speech-to-text-with-transformers/ (visited on 07/05/2024).

[11] C. Krumhansl and R. Shepard, "Quantification of the Hierarchy of Tonal Functions within a Diatonic Context", *Journal of experimental psychology. Human perception and performance*, vol. 5, pp. 579–94, Nov. 1979. DOI: 10.1037/0096-1523.5.4.579.

[12] C. L. Krumhansl and C. L. Krumhansl, *Cognitive Foundations of Musical Pitch* (Oxford Psychology Series). Oxford, New York: Oxford University Press, Nov. 2001, ISBN: 978-0-19-514836-7.

[13] D. Temperley, "A Bayesian Approach to Key-Finding", en, in *Music and Artificial Intelligence*, C. Anagnostopoulou, M. Ferrand, and A. Smaill, Eds., Berlin, Heidelberg: Springer, 2002, pp. 195–206, ISBN: 978-3-540-45722-0. DOI: 10.1007/3-540-45722-4_18.

[14] M. Kania, T. Łukaszewicz, D. Kania, K. Mościńska, and J. Kulisz, "A Comparison of the Music Key Detection Approaches Utilizing Key-Profiles with a New Method Based on the Signature of Fifths", en, *Applied Sciences*, vol. 12, no. 21, p. 11 261, Jan. 2022, Number: 21 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2076-3417. DOI: 10.3390/app122111261. [Online]. Available: https://www.mdpi.com/2076-3417/12/21/11261 (visited on 11/10/2023).

[15] C.-H. Chuan and E. Chew, "Polyphonic Audio Key Finding Using the Spiral Array CEG Algorithm", in *2005 IEEE International Conference on Multimedia and Expo*, ISSN: 1945-788X, Jul. 2005, pp. 21–24. DOI: 10.1109/ICME.2005.1521350. [Online]. Available: https://ieeexplore.ieee.org/document/1521350 (visited on 11/10/2023).

[16] E. Chew and Y.-C. Chen, "Mapping Midi to the Spiral Array: Disambiguating Pitch Spellings", vol. 21, Aug. 2002, ISSN: 978-1-4613-5366-9. DOI: 10.1007/978-1-4615-1043-7_13.

[17] A. George, X. A. Mary, and S. T. George, "Development of an intelligent model for musical key estimation using machine learning techniques", en, *Multimedia Tools and Applications*, vol. 81, no. 14, pp. 19 945–19 964, Jun. 2022, ISSN: 1573-7721. DOI: 10.1007/s11042-022-12432-y. [Online]. Available: https://doi.org/10.1007/s11042-022-12432-y (visited on 11/10/2023).

[18] H. Papadopoulos and G. Peeters, "Local Key Estimation From an Audio Signal Relying on Harmonic and Metrical Structures", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 4, pp. 1297–1312, May 2012, Conference Name: IEEE Transactions on Audio, Speech, and Language Processing, ISSN: 1558-7924. DOI: 10.1109/TASL.2011.2175385. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6074928 (visited on 11/10/2023).

[19] H. Schreiber and M. Müller, *Musical Tempo and Key Estimation using Convolutional Neural Networks with Directional Filters*. Mar. 2019.

[20] Y. Li, R. Yuan, G. Zhang, *et al.*, *MAP-Music2Vec: A Simple and Effective Baseline for Self-Supervised Music Audio Representation Learning*, arXiv:2212.02508 [cs, eess], Dec. 2022. DOI: 10.48550/arXiv.2212.02508. [Online]. Available: http://arxiv.org/abs/2212.02508 (visited on 06/29/2024).