Real-time Classification of Electric Guitars using Machine and Deep Learning Methods

Project Report

Group 841

Aalborg University
Electronics and IT

Electronics and IT
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Real-time Classification of Electrical Guitars using Machine- and Deep Learning Methods

Theme:
Acoustics, Sound Processing and Sound Perception

Project Period:
Spring Semester 2023

Project Group:
841

Participant(s):
Benjamin Oliver Musak Hansen
Jacobo Gonzalez de Frutos
Jesper Mortensen
Kata Bujdosó
Vinícius Soares Matthiesen

Supervisor(s):
Christian Sejer Pedersen

Copies: 1

Page Numbers: 101

Date of Completion:
June 1, 2023

Abstract:

Currently, when electric guitar players have to change guitars between different songs on stage, they have to spend time more or less manually reconfiguring the signal processing chain. This project proposes systems to classify relevant parameters in electrical guitars using various machine learning methods such as SVM with and without LDA, KNN with and without LDA, independent LDA, and a deep learning method called TabCNN. These methods are trained to classify the guitar type, pickup position, pickup type, strumming method, and the player who strummed the guitar in a real-time system. The dataset has been collected manually from a total amount of 11 electric guitars with a total of 2078 samples. The samples are converted to Mel spectrograms and MFCCs to train and test the models. It was found that generally all models performed well. Though, SVM without LDA showed the highest average F1-score between the classes.

agreement with the author.

Contents

Preface

Here is the preface. You should put your signatures at the end of the preface.

Benjamin Oliver Musak Hansen
<bomh19@student.aau.dk>

Jacobo Gonzalez de Frutos
<jgonza22@student.aau.dk>

Jesper Mortensen
<jkjar18@student.aau.dk>

Kata Bujdosó
<kbujdo22@student.aau.dk>

Vinícius Soares Matthiesen
<vmatth19@student.aau.dk>

Chapter 1

Introduction

When playing or creating music, electric guitar players often want a specific sound or expression for a given song. This is achieved by switching between various guitar types, selection of amplifiers, and settings of guitar pedals, to alter the signal processing chain to their desire. Often guitarists have different basic configurations of their signal processing chain depending on the type of guitar they use, including a selection of amplifiers. This means that when the guitarists change guitars between two different songs during a live performance, they also have to spend time more or less manually re-configuring the signal processing chain. Currently, switching efficiently between guitar pedals can be achieved by placing them strategically or implementing a path switcher into the signal processing chain, which controls predefined configurations of the signal processing chain.

However, switching between different guitar types will alter the output level based on factors such as type of pickup, construction of the pickup, proximity between pickup and strings, and the overall construction of the guitar [1]. Thus, to acquire similar output levels from different guitars, these factors must be compensated for in the signal processing chain by manually adjusting gain and volume settings on the pedals and knobs on the guitar by the guitarist.

Thus, the goal of this project is to design and train an artificial intelligence (AI) system to optimize the process of switching between guitars and altering the signal processing chain to the desired output level. Hence, the system must be able to distinguish between different guitar types such as Stratocaster, Telecaster, Les Paul, etc. It is a key success factor that the recognition is both precise and fast as the classification must be performed in real-time. Furthermore, this project will investigate if an AI system can distinguish between basic pickup types like single coil and humbucker, pickup positions, strumming patterns, and the player who is strumming the guitar. To conduct a classification of the aforementioned criteria, relevant features from the input signal of a guitar will be examined to determine, which ones are most suited.

Chapter 2

Problem Analysis

The purpose of this problem analysis is to explore the underlying principles of how an electric guitar produces sound when the strings are being strummed by a guitar player. First, this chapter contains a section on the fundamental physics of guitar strings. Following this section, some terms that describe the different tones a guitar can produce will be explained, as they are used throughout the project. The next section mentions the different components that an electrical guitar consists of and how these components influence the sound. Moreover, this chapter will look into some of the most popular electrical guitars and will document the characteristics of their sound, and what components contributes to this. Thereafter, a related work section will research other methods that have been used in similar areas to classify guitars based on their sound. Lastly, the problem analysis will be summed up with a final problem statement that will reflect the main task.

2.1 Physics of Guitar Strings

This section will explain the fundamentals of how a guitar produces sound when a guitar player is strumming the strings.

Vibrating strings have been used to make music throughout human history. One of the first people who attempted to describe the physics of strings was Pythagoras, who observed that if the tension in a string is held constant, then pleasing combinations of notes can be produced by changing the length of the string in specific ways [1]. This theory has been improved upon throughout the later centuries and will be explained in this section.

2.1.1 Modes of vibration

When a guitar string bound at both ends is plucked it will vibrate in a standing wave condition with different modes until the energy in the string is depleted.

Sound from the vibrating string is formed from pressure waves propagating through the air. In a standing wave condition, the ends of the string are nodes and the length of the string is some integer number of half wavelengths as seen in Fig. 2.1. These modes visualize what the string would look like if high-speed pictures were taken at the exact time when the string has maximum deflection. Each mode of vibration produces a specific sound that corresponds with a resonant frequency, and since many modes of vibrations are created when plucking a string, there are many different resonant frequencies excited at once. The relationship between a frequency and a note will be explained in the following sections. In Fig. 2.1 the string is plucked at the center, resulting in the antinodes[1] being stimulated at odd modes, where all even modes that have nodes[2] at the center of the string will be weak or absent. Thus, the sound quality of the guitar changes as the strings are plucked at different locations. This characteristic of guitar strings is what produces overtones, which is explained in the next section. [1, 2]

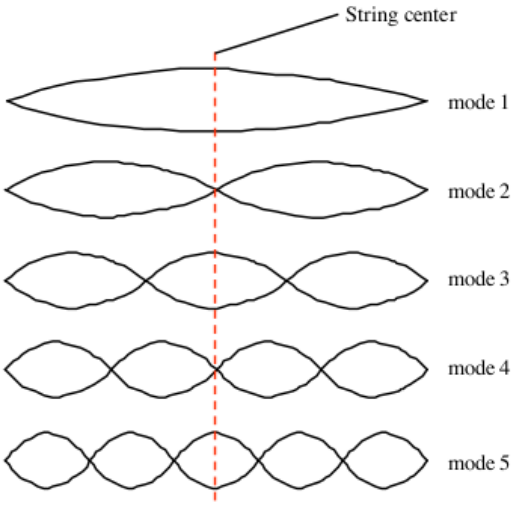


Figure 2.1: The first five modes of a vibrating string that the string goes through when plucked. The dotted red line indicates the center of the string. [2]

2.1.2 Notes and Frequencies

Tonal parts of music can be considered as a sequence of succeeding notes with a variety of different frequencies. These frequencies are not a random collection of frequencies, but rather there is a simple mathematical structure that specifies precise frequencies for the notes. Music notes are defined in terms of intervals

[1] Points in the standing wave that undergo large displacement.
[2] Points in the standing wave that undergo no displacement.

of frequencies. The smallest possible interval between two tones in Western tonal music is called a half-step, and two half-steps equal a whole-step interval. [1]

Every note can be described by the ratio of its frequency to some other note. For example, if a starting note with the frequency f_0 is chosen, the frequency of the next note f_1 will be a half step higher. The ratio of these two frequencies, r, is defined as $r = \frac{f_1}{f_0}$, so $f_1 = r \cdot f_0$. The frequency that is a half step higher than f_1 is f_2 and is described as $f_2 = r \cdot f_1 = r^2 f_0$. The ratio of the first and the twelfth frequency f_{12} is defined as $f_{12} = r^{12} f_0$, where this ratio is known as an octave, also known as a doubling of the frequency, meaning that $r^{12} = 2 = f_{12}/f_0$. Thus $r = \sqrt[12]{2} \approx 1.059$. [1] The relationship of frequencies and the number of half steps can be seen in Table 2.1a.

Number of Half Steps	Relationship
0	f_0
1	$f_1 = rf_0$
2	$f_2 = rf_1 = r^2 f_0$
3	$f_3 = rf_2 = r^3 f_0$
4	$f_4 = r^4 f_0$
5	$f_5 = r^5 f_0$
6	$f_6 = r^6 f_0$
7	$f_7 = r^7 f_0$
8	$f_8 = r^8 f_0$
9	$f_9 = r^9 f_0$
10	$f_{10} = r^{10} f_0$
11	$f_{11} = r^{11} f_0$
12	$f_{12} = r^{12} f_0$

(a) The relationship between a frequency f_0 and the number of half steps.

Number	Note Name
1	A
2	A♯ / B♭
3	B
4	C
5	C♯ / D♭
6	D
7	D♯ / E♭
8	E
9	F
10	F♯ / G♭
11	G
12	G♯ / A♭

(b) The names of the notes from A-G with sharps and flats.

Table 2.1: Tables containing the number of half steps and their relationship to each other, and the name of each note. [1]

All notes have been given a letter from A to G as seen in Table 2.1b. The sharp symbol indicates that a letter note has been raised by a half step and a flat indicates that a letter note has been lowered by a half step. Thus, it can be seen that A sharp and B flat are the same note. However, not all notes have a sharp and flat. [1]

The notes follow the pattern A, B, C, ..., G, since there are more than 12 notes in the human hearing range, these letters repeat themselves after reaching the end of the pattern, but with a higher integer representation. An international standard has defined the A_4 note to be 440 Hz. By using the above-mentioned formulas for describing the ratio of frequencies, Table 2.2 shows a list of all the notes in the

human hearing range 20Hz - 20KHz. [1]

The strings on a guitar are each tuned to a specific frequency (E_2, A_2, D_3, G_3, B_3, E_4) allowing each string to play a different note. When the strings are pressed against a fret, that fret becomes the end of the string, effectively shortening the length of the vibrating part of that string. Most electrical guitars usually have 21, 22, or 24 frets across the fretboard, where every successive fret on the neck raises the fundamental frequency of the string by one-half step, allowing the guitar to produce many different notes. [1, 2]

2.1.3 Harmonics

The frequencies mentioned so far for each note are the fundamental frequencies, also known as the first harmonic, which is a frequency produced by the first mode of vibration. However, there are always additional tones produced simultaneously from higher modes of vibration known as overtones, i.e., the second mode produces the first overtone, the third mode produces the second overtone, and so on. The frequencies of the overtones are related to the fundamental frequency harmonically. Harmonics are overtones that have a simple integer multiple of the fundamental frequency, i.e, if a string produces a fundamental frequency of 110 Hz, then the integer multiples of that frequency will occur at the same time: 220 Hz, 330 Hz, 440 Hz, etc., although the intensities for each will vary. When the overtones are harmonic then the fundamental and the overtones can be classified as an order of harmonic as seen in Table 2.3. A musical instrument's fundamental frequency and all of its overtones can be visualized as a power spectrum. Fig. 2.2 shows the sound spectrum for a guitar playing the D_3 note. [2] The number of relative intensities of the harmonics define the tone characteristics and vary depending on the musical instrument. [2]

2.1.4 Timbre

Different musical instruments sound different even when playing the same note. Every instrument has a characteristic sound quality, also known as timbre, which is the perceived sound quality of a musical note, sound, or tone. Timbre is defined as "that attribute of auditory sensation which enables a listener to judge that two nonidentical sounds, similarly presented and having the same loudness and pitch, are dissimilar" [3]. The existence of the basic tones combined with supplementary tones gives a special flavor to the sound being produced, that characterizes the sound and gives it an identity [1]. Timbre differs from other sound attributes as it is a physical phenomenon and therefore can not be associated with any physical quantity, and is not defined.

E_0	20.602	E_3	164.81	E_6	1318.5	E_9	10548
F_0	21.827	F_3	174.61	F_6	1396.9	F_9	11175
	23.125		185.00		1480.0		11840
G_0	24.500	$\mathbf{G_3}$	**196.00**	G_6	1568.0	G_9	12544
	25.957		207.65		1661.2		13290
A_0	27.500	A_3	220.00	A_6	1760.0	A_9	14080
	29.135		233.08		1864.7		14917
B_0	30.868	$\mathbf{B_3}$	**246.94**	B_6	1975.5	B_9	15804
C_1	32.703	C_4	261.63	C_7	2093.0	C_{10}	16744
	34.648		277.18		2217.5		17740
D_1	36.708	D_4	293.66	D_7	2349.3	D_{10}	18795
	38.891		311.13		2489.0		19912
E_1	41.203	$\mathbf{E_4}$	**329.63**	E_7	2637.0	E_{10}	21096
F_1	43.654	F_4	349.23	F_7	2793.8		
	46.249		369.99		2960.0		
G_1	48.999	G_4	392.00	G_7	3136.0		
	51.913		415.30		3322.4		
A_1	55.000	A_4	440.00	A_7	3520.0		
	58.270		466.16		3729.3		
B_1	61.735	B_4	493.88	B_7	3951.1		
C_2	65.406	C_5	523.25	C_8	4186.0		
	69.296		554.37		4434.9		
D_2	73.416	D_5	587.33	D_8	4698.6		
	77.782		622.25		4978.0		
$\mathbf{E_2}$	**82.407**	E_5	659.26	E_8	5274.0		
F_2	87.307	F_5	698.46	F_8	5587.7		
	92.499		739.99		5919.9		
G_2	97.999	G_5	783.99	G_8	6271.9		
	103.83		830.61		6644.9		
$\mathbf{A_2}$	**110.00**	A_5	880.00	A_8	7040.0		
	116.54		932.33		7458.6		
B_2	123.47	B_5	987.77	B_8	7902.1		
C_3	130.81	C_6	1046.5	C_9	8372.0		
	138.59		1108.7		8869.8		
$\mathbf{D_3}$	**146.83**	D_6	1174.7	D_9	9397.3		
	155.56		1244.5		9956.0		

Table 2.2: The equal tempered notes in the human hearing range (Hz). The bold notes indicate the standard tuning for each of the 6 strings on a guitar. [1]

Mode of vibration	Frequency name (for any type of overtone)	Frequency name (for harmonic overtones)
First	Fundamental	First harmonic
Second	First overtone	Second harmonic
Third	Second overtone	Third harmonic
Fourth	Third overtone	Fourth harmonic

Table 2.3: The mode of vibration and their corresponding names for any type of overtone and harmonic overtones. [1]

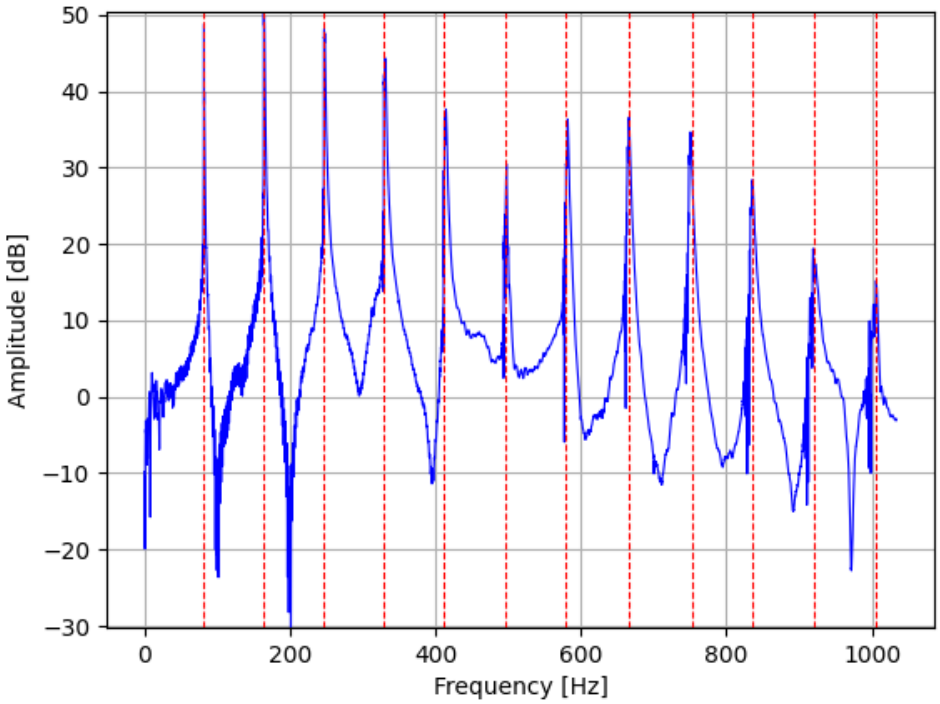


Figure 2.2: Power spectrum of the E_2 string plucked on a guitar. The harmonics can be seen at every peak starting with the first harmonic at 82.41 Hz. [1]

2.2 Tone Terms

In the following sections, terms such as "warm" or "bright" will be used to describe the tones that an electric guitar produces. Therefore, the purpose of this section is to give a fundamental understanding of the physical meaning of these tone terms.

Rosi et al. [4] attempted to describe the physical attributes behind sound. They would describe phrases by interviewing sound experts in a controlled environment and analyzing their answers. In the paper, they published their findings, where 4 different terms were distinguished: warm, bright, round, and rough. Warm and

bright were specified as follows:

- "A **warm** sound seems to be a low-pitched or mid-low-pitched sound. It gives a feeling of spectral richness in the mid-low frequencies. It has a rather soft attack and it is a fairly pleasant sound for the listener, giving a sensation of envelopment."
- "A **bright** sound has most of the spectral energy in the high frequencies. It is often a high-pitched sound. It can be composed with a sharp attack."

One of the important characteristics of the sound produced by a guitar is the resonant frequency which influences how the sound feels for humans [1].

The resonant frequency is specified by an object's natural frequency. The natural frequency is the frequency where objects oscillate the best. In an electrical circuit, the resonant frequency and the amplitude (peak) of the frequency can be modified through circuit elements [5].

If the resonant peak is around 4 kHz - 8 kHz, the sound is described as 'bright'. By raising the resonant peak to 10 kHz, the tone sounds less bright due to that human hearing is less sensitive at higher frequencies. The A-weight curve on Fig. 2.3 approximates the sensitivity of human hearing to different frequencies. Furthermore, frequencies below 4 kHz are described as 'warm' [1].

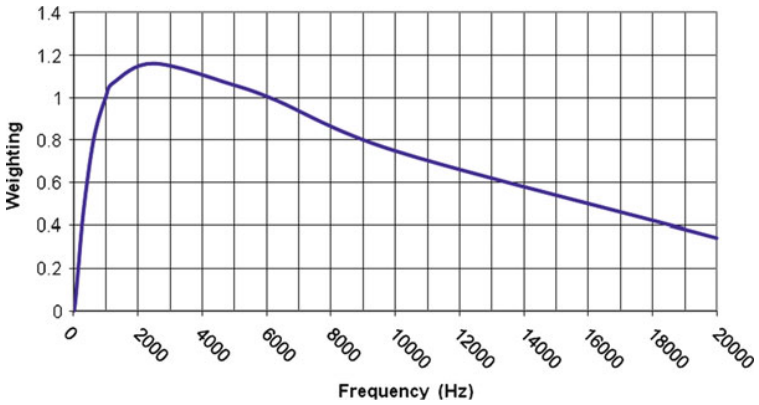


Figure 2.3: Human hearing sensitivity in different frequencies. [1].

2.3 Electric Guitar Parts

This section will present the different components of an electric guitar and how the components influence the characteristics of tone. This includes the build of a guitar and how some of its parts can be altered to achieve different sounds. Additionally, it will be explained how strumming a string on the guitar goes from a vibration

to an electrical signal that can either be analyzed or played back on speakers. The section will cover the following components, see Fig. 2.4.

- Strings
- Body types
- Tonewoods
- Necks
- Volume and tone knobs
- Guitar pickups

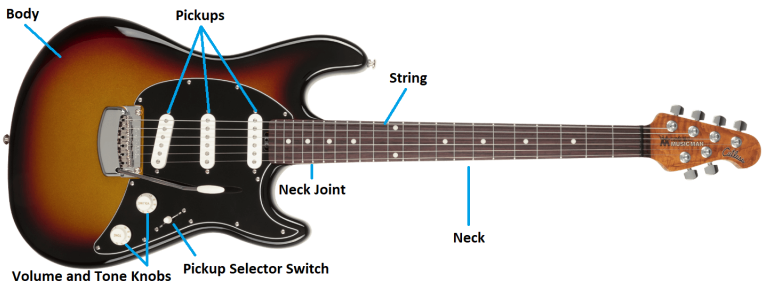


Figure 2.4: Components of a guitar [6].

2.3.1 Strings

An electric guitar usually contains 6 strings that vary in material and thickness to achieve the desired notes of $E_2, A_2, D_3, G_3, B_3, E_4$. The sizes of guitar strings are identified by the diameter of the smallest string, the E_4, which typically are 0.009 or 0.010 inches [1]. Depending on the guitar manufacturer, the strings vary in two key points, which impact their sound: the material and how they are constructed.

Material

- **Steel**: It produces a louder tone and thus a higher frequency sound. Steel strings are popular with metal and hard rock guitarists [7].
- **Nickel**: It creates a softer feel while playing and warm and rich tone [8], often used in genres like blues and classic rock music.
- **Brass and bronze**: These are two variations of the steel strings. The bronze strings usually deliver a greater volume and more brilliance than normal steel, whereas the brass strings offer a more sharp or bright tone compared to bronze. [9].

Construction

- **Gauge**: The gauge dictates how rich and bodied the sound is by the thickness of the strings. Hence, the higher the gauge number of the wire, the thinner its diameter. [10].
- **String core**: This refers to the shape of the strings. Two shapes exist; *round core* and *hex core*. The *round core* strings offer mellower tones that sound used for playing blues or classic rock. Meanwhile, the *hex core* strings are typically louder and brighter, delivering a very modern sound best suited to more recent rock and metal [11]. The two core types are illustrated in Fig. 2.5.
- **Winding type**: For electric guitars, there are mainly two winding types; *roundwound* and *flatwound*. *Roundwound* is the standard type of string that consists of a wire core surrounded by wire wrapping which produces bright sounds and is more likely to cause noise. *Flatwound* strings are surrounded by metal tape and have a smooth surface and a bassy tone [12]. The different winding types are shown in Fig. 2.6.
- **String coating**: The coating is to extend the life of strings and protects them against wear. There are 2 types of string coating; *nanoweb* and *polyweb*. *Nanoweb* strings sound more similar to uncoated ones, however, they are less effective at extending the strings' lives. *Polyweb* is a thicker polymer coating but it makes upper frequencies almost mute. [13]

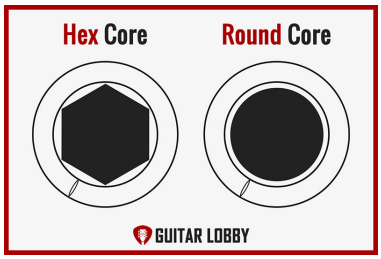


Figure 2.5: String core types [14].

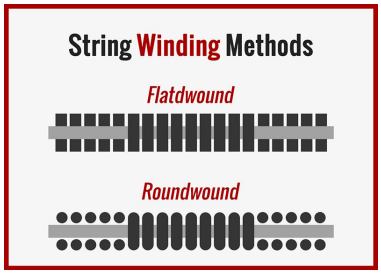


Figure 2.6: Winding types [14].

2.3.2 Body Types

Electric guitars have three body types; solid, hollow, and semi-hollow. Solid bodies are the most common body type for an electric guitar. They consist of a solid piece of wood without any air chamber inside the body. This body type helps to produce cleanly amplified sounds [15].

Hollow bodies have open space between the front and backsides of a guitar which means they can be played as an acoustic instrument, too. If a hollow body guitar is played electronically, it produces warm, jazzy sounds [15].

Semi-hollow bodies have solid bodies with air chambers hollowed out in sections of the body. They produce softer tones than solid bodies. The wood in the body reduces the electronic feedback, which could mute or muddy the sound of the electric guitar [15].

2.3.3 Tonewood

The material of the guitar, also denoted tonewood, has an effect on the tone, generated by the guitar. The tones are spread on a wide spectrum. For instance, a guitar made of mahogany will provide warm and mellow tones, while a maple guitar will have cool and bright tones, as can be seen, on Fig. 2.7. Material requirements for electric guitars are less strict than for acoustic bodies as the tones are converted to electrical signals. Mostly, the wood type will depend if the body is covered with opaque paint. In that case, the grain and appearance of the wood will not matter, or if the body is left clear, therefore appearance will also be important [1].

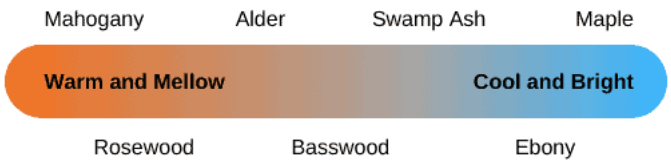


Figure 2.7: Tonewood types and their tones [16].

2.3.4 Neck Types

The neck of a guitar plays an important role in how the guitar will sound, as it shapes the instrument's acoustic voice from how the string vibration is transferred to the body through the neck/body joint (as well as through the bridge). Thus, the differences in wood density and stiffness alter the way the neck translates the sound to the body.
Next, some of the most popular types of necks will be presented:

- **Bolt-On Neck**, see Fig. 2.8: As the name suggests, this is when the neck is bolted onto the body. It is commonly used in guitars where the neck must be fitted tightly to avoid gaps and loss of sound. The obvious advantage of a bolted joint is that the instrument can easily be taken apart when necessary [1].
- **Set-Neck**, see Fig. 2.9: This is different from the previous one, where instead of bolting the neck, glue is used to connect the neck to the body. The neck is at a small angle to the body in order to make the instrument more comfortable to play, and this must be accounted for in the neck joint. This improves the quality of the guitar as it produces a fuller-sounding tone [1].
- **Neck-through-Body**, see Fig. 2.10: This is the most premium neck construction available. In this design, the block from which the neck is carved extends all the way through the body so that the neck and body form one continuous structure [1].

Figure 2.8: Bolt on neck [17].

Figure 2.9: Set-Neck [18].

Figure 2.10: Neck-through-body [19].

The truss rod is another characteristic that most of the guitar's necks have, which is a component located inside the neck of the guitar, as shown in Fig. 2.11. The use of this component is to add stiffness to the neck producing counter-string tension, which can be adjusted at the end of the neck to add or remove tension. Guitars can have two types: nos-adjustable or fixed. [1].

2.3.5 Volume and Tone Knobs

Electrical guitars are equipped with potentiometers, which control the volume and tone of the output signal before it sends it to the amplifier. The volume knob con-

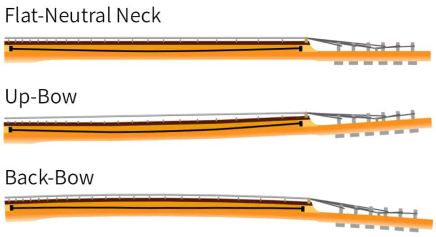


Figure 2.11: Truss rod in a electric guitar [20].

trols the loudness of the output signal, whereas the tone knob act as a filter, which eliminates certain frequencies when turned. Different potentiometers differ in their electrical characteristics, the two most important types are mentioned below [21]:

- **Value**: The resistance between the two outer lugs. For single-coil pickups 250 kΩ, and for humbucking pickups 300-500 kΩ is the most common value.
- **Taper**: The ratio of wiper travel to the resistance between the wiper and the outer lugs.

2.3.6 Guitar Pickups

The pickup is often considered the most prominent element in an electric guitar, as it contributes the most to the characteristics of the produced sound.
The purpose of it is to convert the vibrations of a guitar string into an electrical signal, which can be either recorded or amplified using a guitar amplifier [1].

The pickups of an electric guitar can roughly be divided into two categories:

- **Active pickups**: The active pickup uses magnets and a coil along with a preamplifier circuit to actively boost the output signal.
- **Passive or magnetic pickups**: The passive and magnetic pickups uses magnets and a coil to generate an output signal.

Passive pickups are more commonly used in electric guitars. Thus, the following sections will solely examine passive pickups in terms of; how they function, the components they comprise, and how the components affect the output. [1]

Passive Pickups

A passive pickup comprises three components: magnet(s), pole pieces, and a coil. These operate through the principle of electromagnetic induction. The purpose of the magnet is to magnetize the guitar string, which is hovering above the pickup, through the pole pieces, see Fig. 2.12, making it act like a magnet. This creates a

magnetic field around the string itself, see Fig. 2.13. Since the coils are wrapped around the pole pieces as a convex hull, a current will be induced inside the coil from the frequency of the vibrating string. Since current is proportional to the rate of change of the magnetic flux [22], the positioning of the pickup and strings on the guitars' body is critical to how the electric guitar will sound, as different placements will have a different magnetic flux. This placement is not only dependent on the change in the horizontal axis but also on the vertical axis. If the magnets or pole pieces are placed at different heights inside the pickup, the sound will be different from that of a standard configuration, since some strings are closer to the magnets than others. This enhances the flux of those magnets and therefore the output of those frequencies [22]. The magnets used in the pickups also have an effect on the final input.

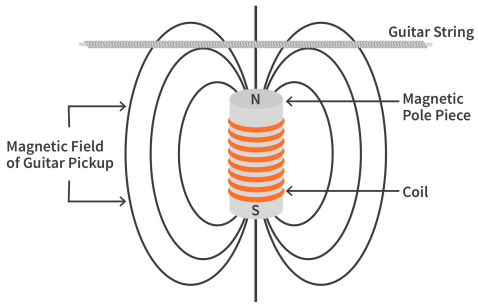


Figure 2.12: Visualization of the pickup magnetizing the string [23].

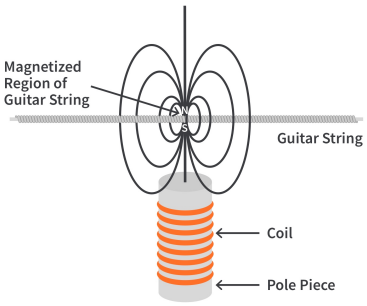


Figure 2.13: Visualization of the string acting like a magnet [23].

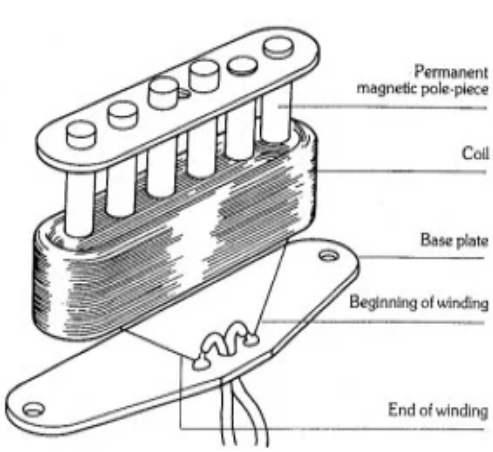


Figure 2.14: Schematic of single coil pickup [24].

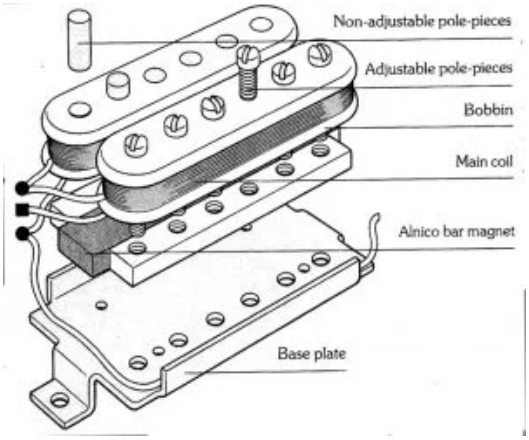


Figure 2.15: Schematic of humbucker pickup [25].

There are considered to be two main types of magnetic pickups: single coils and

humbuckers, see Figs. 2.14 and 2.15. The fundamental principle of how they function is somewhat identical, however, their construction varies, which contributes to different tones, sound characteristics, and output from the guitar. [1]

Single Coil

The single coil consists of a coil of copper wire wrapped around six permanent magnetic pole pieces mounted on a base plate, see Fig. 2.14. The wounding of the coil can be either clockwise or counter-clockwise, as shown in Fig. 2.16 and Fig. 2.17, and the polarity of the magnet facing up towards the string determines the phase of the signal, e.g. south up and clockwise creates a negative phase [26]. However, the construction of the single coil makes it very sensitive to electromagnetic radiation from e.g., radio transmitters and power transformers, which generate an interference around 50-60 Hz also known as a "hum" when the signal is amplified [1].

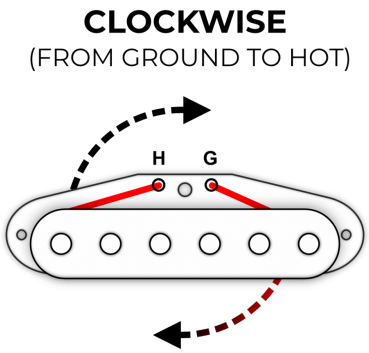


Figure 2.16: Visualization of a pickup with clockwise wounding [27].

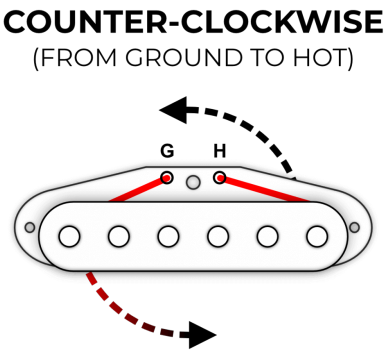


Figure 2.17: Visualization of a pickup with a counter-clockwise wounding [27].

Humbuckers

To eliminate the "humming", the humbucker pickup was designed. The humbucker uses a combination of two single coil pickups wound in opposite directions, which makes them oppositely sensitive to the moving electromagnetic field. Hence, a moving electromagnetic field induces a positive voltage in one coil and induces a negative voltage in the other coil bringing them out of phase, which cancels out the noise signal when connected in series. However, this constellation creates a problem, where the two pickups are oppositely sensitive to string motion. To accommodate for this issue, an opposite magnetic polarity of the pole pieces in

the two coils is introduced, which brings them out of phase as well. The phase difference in the coils in conjunction with the phase difference in the magnets places the string signal in phase, resulting in cancellation of undesired electromagnetic noise and an enhanced signal produced by the vibrating string. [1]

Output of an Electric Guitar

Having discussed how the single coil and the humbucker pickup operate, the physical properties which determine the output of a pickup will now be investigated. The output of a magnetic pickup is primarily determined by three factors [1]:

- **DC resistance:** The DC resistance describes the resistance of the copper wire within the coil from start to end and is affected by two physical parameters: The number of turns around the coil and the gauge of the wire. A magnetic guitar pickup typically uses 6.000-8.500 turns of wire and by increasing the number of turns within the physical limits of the bobbin, more output is generated from the pickup. Concurrently, altering the gauge of the wire will likewise impact the output of the pickup, e.g., a 42 AWG (American Wire Gauge) will have a larger diameter, and thus less resistance compared to a 43 AWG, as thinner wire have more resistance per unit length.
- **Magnetic strength:** The strength of a magnetic field is determined by the material of the magnet. Choosing magnets with different magnetic properties will alter the inductance generated in the pickup by a vibrating string, thus, affecting the output signal. Modern magnetic materials can roughly be divided into three categories: ceramic, alnico, and rare earth[3], which each have distinct magnetic attributes, and thus provides different tonal characteristics when used in a magnetic pickup.
- **Proximity to strings:** Adjusting the proximity from pickup to the string alters the strength of the magnetic field, and, thus induces a stronger signal in the coil. However, moving the pickup closer to the string can influence the motion of the string and affect certain frequency ranges of the sound.

2.4 Characteristics of Different Types of Electric Guitars

The purpose of this section is to establish the different characteristics of guitars that make them sound unique. Some of the most popular electric guitar types and their sound characteristics will be documented.

[3]Neodymium and samarium cobalt

2.4.1 Telecaster

The Fender Telecaster (TC) is the world's first commercially successful electric guitar produced by Fender in 1950. The design of the Telecaster was geared towards mass production to be a simple yet effective electric guitar, which has been a long-lasting success as it is still in production to this day. [28, 29]

The guitars are based on modular construction, meaning that each individual part can be made separately and then bolted together, e.g. the neck and the body are made separately and then bolted together. Controls, pickups, and wiring are fitted to a pickguard[4] and then screwed to a body. Other hardware such as the bridge, machine heads, and jack sockets are also screwed to the body. [28]

The Telecaster is shown in Fig. 2.18a. It features two single coil pickups: one near the bridge, and one near the neck. This is different from its predecessor, the Esquire, which only had a bridge pickup. The single coil pickups also make the Telecaster susceptible to noise. The neck is made out of maple wood, and the body is made out of ash. It features volume and tone knobs for adjusting the raw sound signal before it is transferred to an amplifier. These materials make the Telecaster known for its bright cutting tone and its straightforward no-nonsense operation. Minor changes to the design have been made since the original debut in 1950, however, the essential design has remained the same. [28].

2.4.2 Stratocaster

The Stratocaster (SC) is the most popular guitar from Fender launched back in 1954 as the successor of the Telecaster. This guitar builds upon the Telecaster's design by having an additional pickup between the neck and bridge pickups, a vibrato to distort the sound, and a more comfortable design for guitarists to hold, as seen in Fig. 2.18b [28]. Like the Telecaster, the Stratocaster is also commonly found with a maple neck that is bolted onto the body made of ash [30]. The three single coil pickups on the Stratocaster offer its characteristic bright and clear sound, however, they are susceptible to interference and are sensitive to feedback and background noise [30].

2.4.3 Les Paul

The Les Paul (LP) is Gibson's take on an electric guitar produced in 1952 to challenge the rise of Fender's Telecaster seen in Fig. 2.18c [28]. The body and neck are made of mahogany wood with a maple front to add brightness, where the neck is glued in instead of screwed in like the Fender guitars [28, 31]. This choice in the material makes the Les Paul heavier than the Fender models [28, 31]. The Les Paul

[4]Plastic plate attached to the surface of the guitar to protect it from scratches caused by the pick when strumming [28]

features two humbucker pickups with separate tone and volume controls, and the result of these materials makes the Les Paul sound darker, thicker, and warmer. The tone focuses on mid-range and bass frequencies rather than treble tones that as the Stratocaster [28].

2.4.4 Solid Guitar/Les Paul Standard

Gibson restyled the Les Paul in 1961 by introducing a new shape for the body which resulted in the Les Paul Standard released in 1961, see Fig. 2.18d. The Les Paul Standard also known as the Solid Guitar (SG), is Gibson's other popular guitar, like the Les Paul, this guitar features two humbuckers pickups; one at the bridge and one at the neck. The main difference between the SG and the Les Paul is the body. The body has a drastically different shape than the Les Paul and the body is made out of mahogany which makes the SG much lighter than the Les Paul. [28]

The sound produced by an SG still has a similar dark and warm sound as the Les Paul due to the same double humbucker setup, however, slight differences can still be heard as the materials chosen for the body is different.

(a) Fender Telecaster.[29]

(b) Fender Stratocaster.[30]

(c) Gibson Les Paul Sunburst.[31]

(d) Solid Guitar/Les Paul Standard.[28]

Figure 2.18: Most popular electric guitars

2.5 Effect of Pickup Locations

The previous section stated that different types of guitars have pickups in different locations. The majority of electric guitars have a neck and a bridge pickup, as seen in the TC, LP, or SG. However, some guitars also feature a middle pickup, located in between these two pickups such as the SC. Fig. 2.19 shows an illustration of the three pickups.

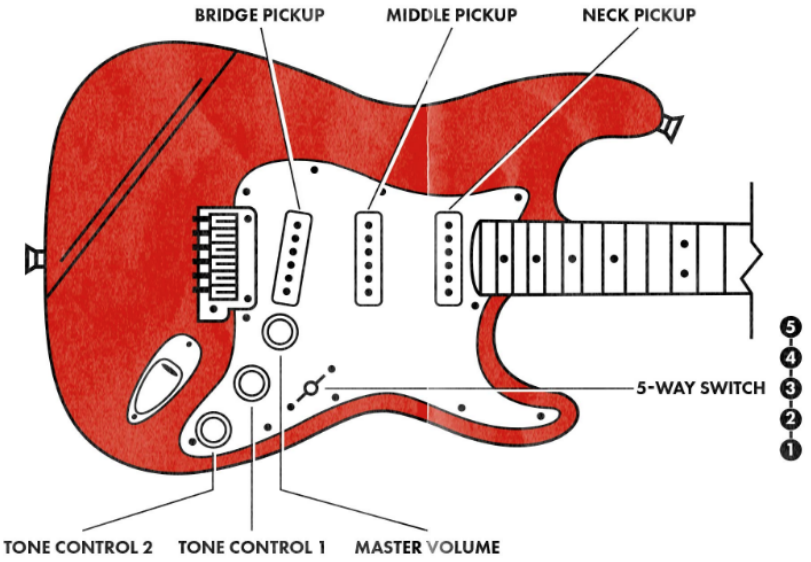


Figure 2.19: The three different pickup locations showcased on a SC. [32]

As presented in Section 2.3.6, guitar pickups detect string vibrations that are close to them. Thus, selecting different pickups will change where string vibrations are recorded and impact on the sound [21]. For example, a guitar with two pickups will have a 3-way selector that can choose between using only the neck pickup, only the bridge pickup, or using both. Whereas, a guitar with three pickups has a 5-way selector, and that can select either the neck, middle, bridge, or a combination of the pickups.

2.5.1 Bridge Pickup

The bridge pickup picks up vibrations that do not contain a strong magnitude of fundamental frequencies, instead, the magnitude of higher frequencies is larger. This produces a brighter tone due to the increased presence of higher harmonics. String vibrations are smaller near the bridge, thus to compensate for this, there are more turns in the coil which makes it more sensitive to small variations in the magnetic field. The increased number of turns in the coil results in the bridge pickup being more mid-range and treble-heavy. [21]

2.5.2 Neck Pickup

The neck pickup senses more of the fundamental frequencies, thus the tone produced from this pickup is a mellowed, clean and pure tone. [21]

2.5.3 Middle Pickup

The tone of the middle pickup is between the two above-mentioned pickups. The resulting sound is clearer than the neck pickup, but not as bright as the bridge pickup. There is a much wider balance of harmonics. [21]

2.6 Related Work

When analyzing audio, several features can be extracted from music tracks and instruments. It can depend on the classification method used and in what context, as to which features would hold the most useful information. Generally, either the harmonics, non-harmonics, Mel Frequency Cepstral Coefficient (MFCCs), Short-Time Fourier Transform (STFT), Constant-Q Transform (CQT), or pitch are used [33, 34, 35]. Often when classifying instruments, some recurring methods are used which include k-Nearest Neighbour (KNN) and Support Vector Machines (SVM). As well, some other recurring methods within the problem of automating guitar tablature use TabCNN or an extended form of it. These solutions will be explored in the upcoming parts.

2.6.1 Guitar Tablature Automation

Guitar tablature classification deals with the specific problem of estimating notes of a solo guitar recording and then identifying which strings were plucked to play them [36]. Making a tablature manually can be a lengthy process and requires expertise [35], and therefore there has been researched on how to automate this process using deep learning methods.

In a paper written by Andrew Wiggins and Youngmoo Kim [35], they talk about how previous methods would divide the problem of tablature automation into two steps; first do a polyphonic[5] pitch estimation, which is then followed by a tablature fingering arrangement. They then propose a method that is able to solve the issues jointly using a Convolutional Neural Network (CNN) model, which is trained to map directly from audio to a tablature. This is done by understanding the physical constraints of a human hand and the difference in string timbres, making it possible to determine the actual fingering being used by the guitarist. They call this method the TabCNN, which is a CNN that estimates the guitar tablature from a solo acoustic guitar performance and is based on AlexNet [38] and VGGNet [39]. It has three convolution layers that have ReLU activations, a max pooling layer, two dense layers, six softmax layers, and six output labels for each string. The input is a constant-Q spectrogram with padding on each side,

[5]Polyphonic is the texture of music which has two or more tones playing at the same time [37], e.g. a guitar is capable of playing more than one tone at a time.

which is used instead of STFT because of its ability to reduce the dimensions of the frequency axis compared to STFT. TabCNN showed great promises with an accuracy (correctly identified which pitch to finger assignment) of 89%.

Two other papers then try to improve upon TabCNN by adding a uni-directional long short-term memory (LSTM) layer [36] (Data-Driven) or a change to the input of six channels instead of one, as well as a change to the feature extraction [40] (FretNet). They both have slightly improved performance compared to TabCNN, but ultimately conclude that the methods they propose are more effectively used for either estimating pairwise likelihoods using curated distributions [36] or for continuous-valued pitch contours within the context of guitar tablature transcription [40].

Looking at other solutions, Kulkarni et. al. [41] studied the difference between two methodologies for instrument classification: harmonic envelopes, and cepstral coefficients. On the other hand, the classification was analyzed with two deep learning approaches: Multilayer Perceptron (MLP) and Temporal Difference Network (TDN). An important fact specified in the paper is the difficulty to process the live instrument data to ensure a good generalization at the time of classification. For this, using FFT, the harmonic envelopes and cepstral coefficients were extracted and later averaged over a number of frames or taken directly for training. The conclusion of this article shows that feeding these neural networks with just single harmonic amplitudes or envelopes is not reflected in a precise classification, and what makes the difference is using cepstral coefficients. The most accurate one was the TDNN with results of 68% using harmonic envelopes and 88% using cepstral coefficients. The MLP achieved an accuracy percentage of 67% for harmonic envelopes and 85% using cepstral coefficients.

2.6.2 Features, KNN, and SVM for Instrument Classification

Within machine learning, SVM and KNN have been proven to be effective methods to classify instruments given a short sample of a song, musical piece, or tone [42, 43, 44]. The efficiency of these methods depends on the quality and quantity of the features selected [45]. Therefore, Foulon et al. [33] used a method to qualify the important features with the most information for detecting guitar modes using the "information gain algorithm of Weka". Here they figured out, that using the harmonic-to-noise ratio, pitch, inharmonicity, and spectral information gave the best results when running tests on neural networks. They also found out, that removing silence or transient sounds gives more robust classifications when looking at audio chunks of about 50 ms.

Another paper written by Livshin et al. [45] studied the importance of looking at both the harmonics and non-harmonics of a sequence, as to see if either of them isolated, or used together, gives the best classification results when used with

Linear Discriminant Analysis (LDA) and KNN. They found out, that using either one can be an effective method to classify instruments. Though, using either only the harmonics, or both, would result in better classification, as the non-harmonics were not as informative about the instruments as the two other features. As well, they end up normalizing the audio sample using the min-max normalization and reached an accuracy of 90.5% with 10 instruments.

Two papers also researched the idea of using Mel Frequency Cepstral Coefficients (MFCCs) for musical instrument classification using SVM and/or KNN [43, 42]. [43] proclaims that using MFCCs for classifying instruments is the most effective. Using this technique on a bigger dataset of three different guitars, they got an accuracy of 80%, whereas the human accuracy on the same dataset resulted in an accuracy of 66%, The second paper, [42] also tested the efficiency of KNN and SVM using MFCCs as features. They found that using SVM has a benefit of about 1-2% accuracy compared to KNN.

2.7 Summary of the Problem Analysis

As presented in Chapter 1, the purpose of this project is to develop a system capable of detecting and classifying different types of guitars to optimize the process of altering the signal processing chain when switching between guitar models. Hence, the various elements a guitar comprises of and how they affect the characteristics of the sound were analyzed in Section 2.3. From this analysis, the pickup was found to be the most notable component when it comes to the sound characteristics of a guitar. The two main pickups used by guitar manufacturers: single coils and humbuckers were found to have vastly different sound characteristics, where single coils are categorized as having a bright and clear sound, whereas humbuckers have warmer and thicker sounds. Fender guitars typically use single coil pickups for their guitars, while Gibson uses humbuckers - this choice along with the materials heavily changes the sound that these guitars produce. The LP, SG, and TC are typically equipped with two pickups at the bridge and neck position, whereas the SC is equipped with three pickups located at the bridge, middle, and neck positions. As the pickups are spread out between the bridge and the neck, the position of the pickup will change how the vibrating string is recorded, and thus alter the characteristics of the sound. Additionally, it was found that the timbre will differ depending on the person playing the guitar. Furthermore, as found in Section 2.1, shortening the length of a vibrating string will change the frequency, and thus the tonal components the sound produced.

To our knowledge, a system that can classify electric guitar models and their pickup placements does not exist. Therefore, the general idea of instrument classification and tablature automation was researched, as to get an understanding of the general problem within the classification of instruments. In the related work

section, it was presented that KNN and SVM are two prominent machine learning methods to detect instruments within a musical piece by using Harmonic peaks and MFCCs features. Furthermore, it was found that LDA in conjunction with KNN can generate good results. As for deep learning methods, the TabCNN was presented to classify the tabs played using a CNN on the spectrogram.

2.8 Delimitation

This section will set the scope for the project based on the desired direction and materials available. This project will be limited to the use of guitars provided by Musictribe, Aalborg University, and guitars personally owned by the group members and their acquaintances. This constitutes to 11 guitars, where our dataset will solely compose of recordings from these guitars.

2.9 Problem Formulation

Based on the problem analysis and the delimitation, a final problem statement is formulated:

How can different parameters of guitars be classified based on the sound they produce using relevant machine/deep learning methods?

Parameters:

- The type of guitar (SC, TC, LP, or SG).
- The pickup used (single coil or humbucker).
- The pickup position (neck, middle, or bridge).
- The person playing the guitar.
- The strumming method (open strings or chords).

Chapter 3

Guitar Dataset

This chapter will present the data set that has been collected for this project. This has been gathered in collaboration with Musictribe, which has provided electric guitars and other materials.

3.1 Data Collection

As presented in the problem formulation, this project intends to create a system capable of classifying the parameters of guitars presented in Section 2.9: guitar type, pickup type, pickup position, strumming method and the player, based on the sound a guitar produces when strummed. Performing classification of guitars requires a large dataset, as it improves the ability of any trained model to better generalize on a bigger scale when encountering unseen data [46]. Hence, the dataset should ideally contain multiple guitars of each type to accommodate for the variance in the components. However, as presented in Section 2.8, only data samples from 11 guitars of various brands within the defined four guitar types have been acquired, where 6 of these guitars have been provided by Music Tribe, 4 guitars by students within the group and acquaintances, and 1 guitar by Aalborg University. Table 3.1 displays the manufacturer, guitar type, pickup type, pickup positions, and the recorded pickup positions of all 11 guitars in the data set.

3.1.1 Setup

To record and digitize the analog signal of the vibrating guitar string, a setup is needed. Thus, the following section will present the setup and its components used. The components of the setup are listed below:

- TC electronics Pedal buffer: prevents signal degradation in the signal processing chain by turning the high input impedance into a tone-saving low output impedance signal (1MΩ input and 100Ω output) [47].

Guitar Dataset

#	Manufacturer	Guitar Type	Pickups	Pickup Positions	Recorded Pickup Positions
1	Supreme	SG	2 Humbuckers	3	Neck and Bridge
2	Hansen	TC	2 Single coils	3	Neck and Bridge
3	Gibson	LP	2 Humbuckers	3	Neck and Bridge
4	Fender	SC	3 Single coils	5	Neck, Middle, and Bridge
5	Squier	SC	3 Single coils	5	Neck, Middle, and Bridge
6	Hansen	SC	3 Single coils	5	Neck, Middle, and Bridge
7	Squier	TC (left)	2 Single coils	3	Neck and Bridge
8	Epiphone	SG	2 Humbuckers	3	Neck and Bridge
9	Gibson	LP	2 Humbuckers	3	Neck and Bridge
10	Axtech	SC	3 Single coils	5	Neck, Middle, and Bridge
11	Fender	SC	3 Single coils	5	Neck, Middle, and Bridge

Table 3.1: General overview of the various guitars used for the dataset.

- Focusrite Scarlett Solo Sound card: is a component that converts analog signals, in our case coming from the guitar to digital audio.
- 2x Fender OFC Noiseless Instrument cables (3m): used for precise guitar-buffer connection and buffer-sound card connection.
- Audacity: is a free software that focuses on editing and recording multitrack audio. It has been used for recording and later on for exporting audio samples.

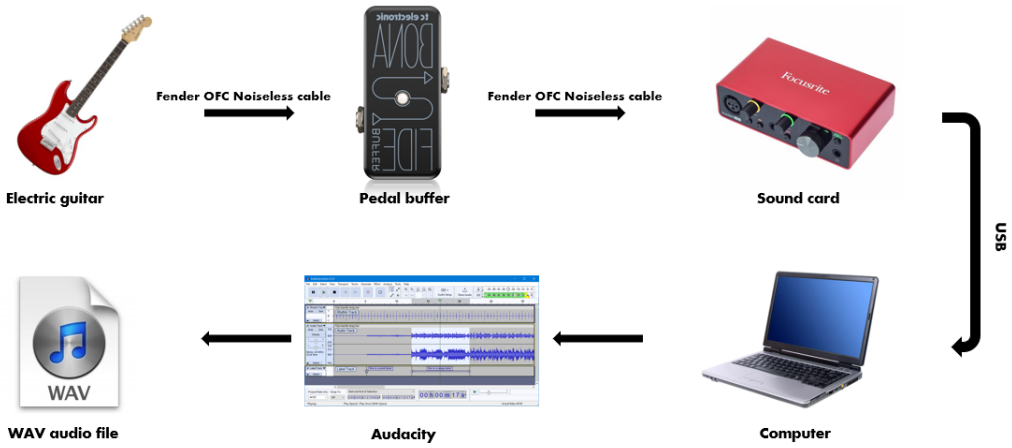


Figure 3.1: Process of data retrieving.

A visual representation of the physical setup is illustrated in Fig. 3.1. To expedite the data collection, two identical setups were used similar to the one shown

in Fig. 3.1, with the exception of the sound card. The first setup utilized a Focusrite Solo gen 3, whereas the second setup utilized a Focusrite Solo gen 1. The specifications of the two sound cards are shown in Table 3.2 [48, 49, 50].

Focusrite Solo sound card comparison

Name	Max Input Level	Gain Range	Max Sample Rate
Focusrite Scarlett Solo gen 1	15dBu	46dB	92kHz
Focusrite Scarlett Solo gen 3	12.5dBu	56dB	192kHz

Table 3.2: Comparison of specifications between the Focusrite Scarlett Solo gen 1 and gen 3

Both sound cards used a sample rate of 48kHz for all recordings. However, as presented in Table 3.2, the Maximum Input Level and Gain Range between the 1st generation and the 3rd generation sound card varies, which produces different gains to the signal when digitized of -31dBfs and -42.5dBfs respectively, see Eq. (3.1). This presented an issue when recording, as some guitars had more output than others. Thus, to compensate for the line-up level difference between the two sound cards, all samples have been normalized before any processing.

$$\text{Max Input Level} - \text{Gain Range} = \text{Line-up Level ("Gain")} \tag{3.1}$$

3.1.2 Recording of Samples

Audacity was used to record all the guitars. The default samples rate of 44.1kHz in Audacity was used for all sound recordings. During the collection of data samples, all guitars in Table 3.1 were tuned to standard E tuning: $E_2, A_2, D_3, G_3, B_3, E_4$, and tone and volume knobs were fixed at maximum. For every guitar, 40 open strums and 40 A-major bar chords were played at the natural strumming position, which lays somewhat between the bridge and neck pickup, for each pickup position. This was done to ensure that two strumming patterns were present in the dataset. For the guitar types: LP, SG, and TC, which have 3 pickup positions, only samples from the neck and bridge pickup were collected. Whereas, only data samples from 3 of the 5 pickup positions on the SC were collected, namely, the neck, middle, and bridge pickup position, as acquiring data from all pickup positions would create a large imbalance in the dataset given the selection of guitars available. The 40 strums of each pickup position and strumming type were recorded in one audio file, where each strum was followed by silence as in Fig. 3.2, where 10 consecutive strums are shown. This was done to simplify the recording procedure by avoiding having to save each strum individually.

For a majority of recordings the audio was too loud as the gain of the sound cards was set too high, which introduced clipping of the signal, as shown in

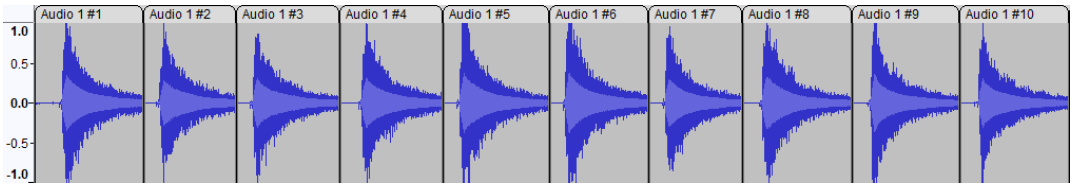


Figure 3.2: Example audio file from the Axtech SC containing 10 consecutive open strums at the bridge pickup on a linear scale.

Fig. 3.3. As clipping contributes to unwanted distortion, loss of audio information, and reduction in sound quality, the inbuilt clip fix function in Audacity was utilized [51]. By applying a threshold of the maximum sample amplitude any sample must be, before being considered clipped in conjunction with a reduction of the amplitude in negative dB, the clip fix function tries to reconstruct the missing peaks by interpolating the lost signal. The threshold and reduction in amplitude to allow for restored peaks were set to 95% and -6.40dB respectively.

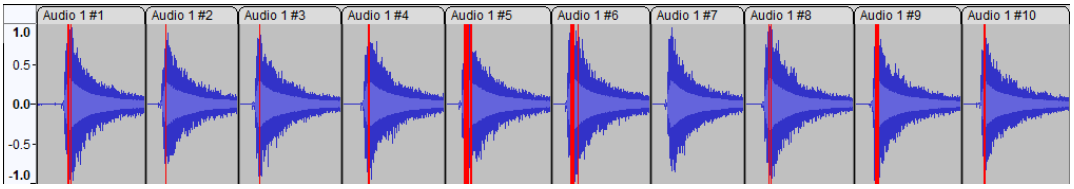


Figure 3.3: Example audio file from the Axtech SC containing 10 consecutive open strums at the bridge pickup on a linear scale. The red lines indicates clipping of the signal, where the recorded audio is too loud to capture.

To split the recordings into individual samples, two methods were developed. In the first method, the vertical scale was changed to decibels relative to full scale (dBfs) as shown in Fig. 3.4, where the recordings were loaded and split based on two parameters: threshold for determining silence in dBfs and minimum silence between each strums. The threshold was set to -42 dB to remove unwanted noise, see Fig. 3.4, and the minimum silence between each strums was set to 150ms. Based on this approach, each sample ended up with a duration between 2-6 seconds depending on the audio recording. The other method to split the dataset is seen in Section 10.1. This method performed worse than the above mentioned, as it struggled with applicability on most the data samples due to their variance. Hence, this method was discarded. The distribution of sample duration is displayed in Fig. 3.6f

A singular sample of a strum is shown in Fig. 3.5, where the vertical scale has been converted back to a linear scale so it displays the amplitude of the waveform. The amplitude scale is linear, with 1.0 being the maximum possible loudness of positive signals and -1.0 being the maximum possible loudness of negative signals without distorting the signal. As seen in Fig. 3.5, the plot has two shades of blue;

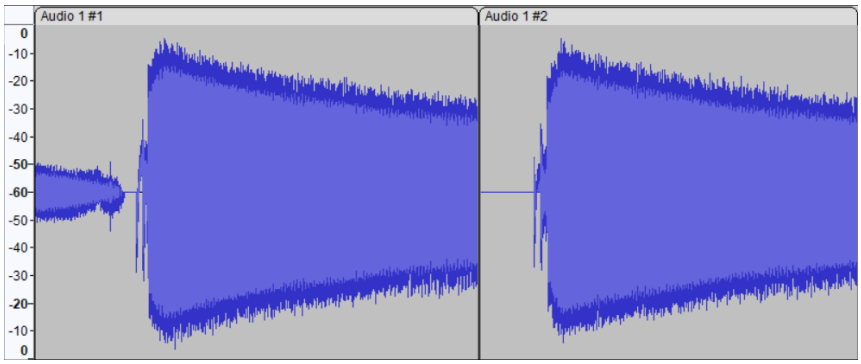


Figure 3.4: Illustration of two consecutive samples displayed on a dB logarithmic scale.

light blue and darker blue. The light blue part of the waveform displays the average Root Mean Square (RMS) value for the entire group of samples and the darker blue display the tallest peak in that area represented by a pixel. Thus, altering the zoom will change the number of samples depicted by the pixel.

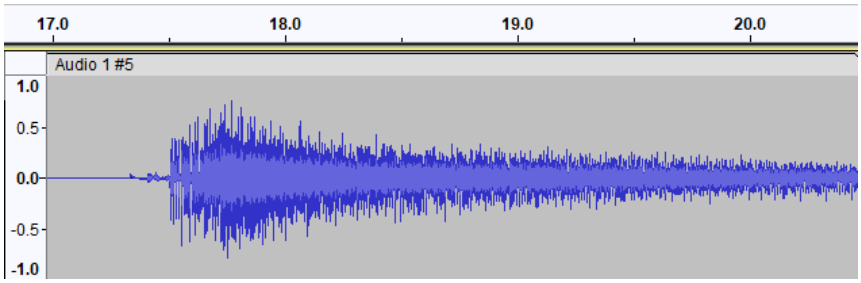
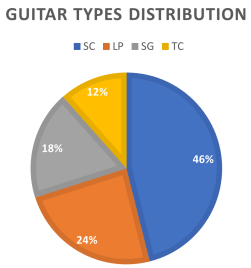


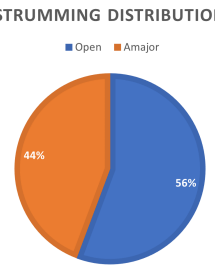
Figure 3.5: Waveform of a SC with an open strum performed at the bridge pickup position corrected for clipping. The vertical scale is linear.

3.2 Class Distribution

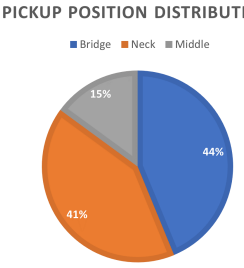
A total of 2,078 guitar samples have been collected. Based on the parameters specified in Section 2.9, the following classes have been created: guitar type, pickup type, pickup position, strumming method, and player. The distribution of the guitar types is fairly skewed as seen in Table 3.1, where the SC is predominant compared to the other guitar types. Additionally, as the SC has three pickups and the residual guitar types only have two pickups, the recording procedure utilized becomes somewhat biased towards the SC, as more samples will be collected. Consequently, the recorded samples between the four guitar types have the following distribution: LP 499, SG 382, SC 957, and 240 TC, where Fig. 3.6a displays the distribution in percentage.



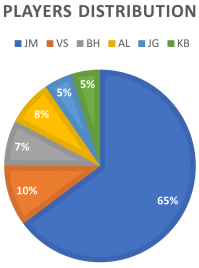
(a) Distribution of guitar types: SC, SG, TC, and LP.



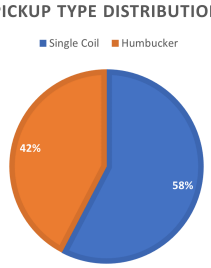
(b) Distribution of open strums and A-major chord strums.



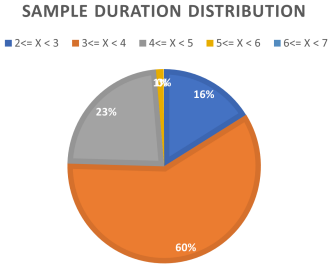
(c) Pickup position distribution.



(d) Players distribution.



(e) Pickup type distribution.



(f) Sample duration distribution

Figure 3.6: Overview of dataset distributions.

The distribution of strums based on pickup position is visualized in Fig. 3.6c, and the distribution of open strums and A-major bar chords are shown in Fig. 3.6b. For the pickup positions, the distribution of recorded samples is skewed towards the neck and bridge pickup, as only the SC have a middle pickup. Moreover, as presented in the Section 2.1, different people playing on the same guitar can cause slight differences in the timbre. Thus, to ensure greater variance and more robustness in the data set, the person strumming the guitar was switched for every 20th strum. The distribution of strums among the group members and acquaintances is shown in Fig. 3.6d, where it can be seen that JM has performed the majority of strums. Both JM and VS had prior experience with playing guitar and were both able to perform the A major chord, while the rest of the group members strug-

gled. Hence, it was decided that solely JM and VS would perform this strumming type. Later on, an acquaintance of one of the group members, AL, was used to increment the amount of data and reduce the disparity of the dataset by recording more data with LP guitar type. Since this player had also experience with playing the guitar, it was possible to equally increment the data in terms of open and A major strumming. The distribution of pickup types in the dataset is fairly balanced with a slight favouring towards the single coil pickup, as shown in Fig. 3.6e.

Chapter 4

Methods

This chapter will present relevant classifiers and features that will be used to create the AI system for classifying the parameters set forth in Section 2.9. The methods that will be presented are derived from Section 2.6 that has analyzed various ways for extracting features of musical pieces and instruments and various models to classify these features. From Section 2.6, it was presented that KNN and SVM are two prominent machine learning methods for detecting instruments in musical pieces using MFCCs. Furthermore, it was found that LDA in conjunction with KNN could generate satisfactory results. Similarly, CNNs were found to be able to classify sound signals by transforming them into spectrograms, which produced results with high accuracy.

4.1 Features

This section will present the theory for extracting spectrograms and MFCCs of a guitar sound signal. All intermediate steps from the raw input signal to the final MFCCs of the signal will be presented and applied to a random guitar sample from the dataset. The sound sample from the dataset is shown in Fig. 4.1.

4.1.1 Windowing

The basic idea of windowing is to extract a finite section of a very long signal $x[n]$ by multiplying it with a window function $w[n]$. The idea comes from looking at a signal through a window and thereby only seeing certain samples of the signal. The window function $w[n]$ is zero outside of a finite-length interval, which allows the function to be sliced to the desired samples. The simplest window function is the L-point rectangular window given in Eq. (4.1). [52]

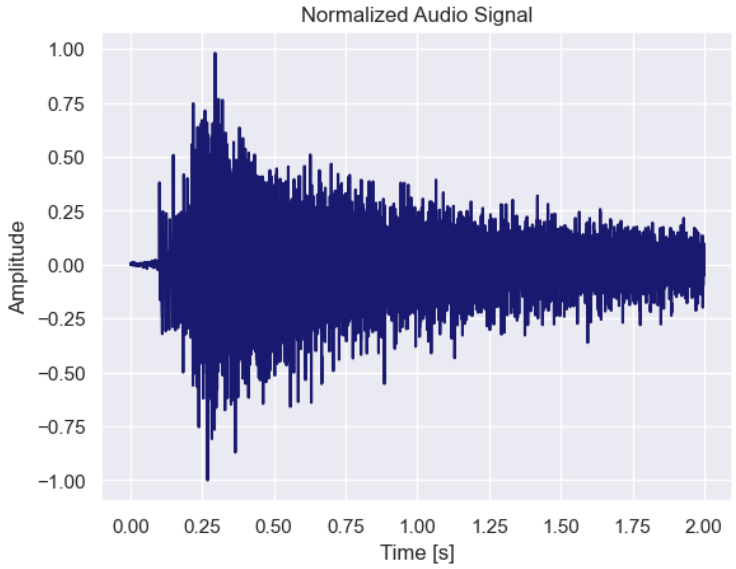


Figure 4.1: Normalized guitar signal from the dataset.

$$w_r[n] = r_L[n] = \begin{cases} 1 & 0 \leq n \leq L-1 \\ 0 & elsewhere \end{cases} \tag{4.1}$$

The rectangular window is 1 at the samples that are desired, and 0 elsewhere. Multiplying the rectangular window $w_r[n]$ on a signal will truncate the signal as seen in Eq. (4.2). [52]

$$w_r[n]x[n+n_0] = \begin{cases} 0 & n < 0 \\ w_r[n]x[n+n_0] & 0 \leq n \leq L-1 \\ 0 & n \geq L \end{cases} \tag{4.2}$$

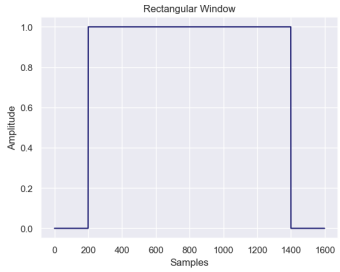
A problem that can occur when using a rectangular window is that the sound signal will have a sudden drop in amplitude at the edges as there is no damping. This causes high spikes in the frequency when analyzing the signal in the frequency domain.

Windowing functions such as the hamming or hanning windows can be used to add a smooth transition to the signal by gradually reducing the amplitude at the boundaries. The hamming window is shown in Eq. (4.3) and the hanning window is shown in Eq. (4.4). [52]

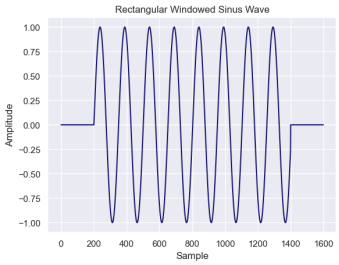
$$w_m[n] = \begin{cases} 0.54 - 0.46cos(\frac{2\pi n}{L-1}) & 0 \leq n \leq L-1 \\ 0 & elsewhere \end{cases} \tag{4.3}$$

$$w_n[n] = \begin{cases} 0.5 - 0.5cos(\frac{2\pi n}{L-1}) & 0 \leq n \leq L-1 \\ 0 & elsewhere \end{cases} \tag{4.4}$$

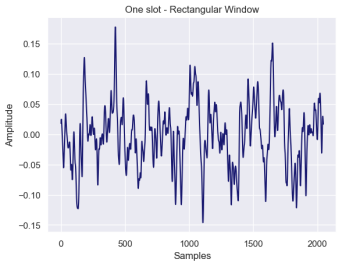
Fig. 4.2 shows the three columns: one showing the different windowing functions, one showing how a random sinus signal is affected when the windowing functions are applied to the signal, and one displaying how the guitar signal from the dataset is affected by the windowing functions. The hamming and hanning windows are symmetric around the center which is where the largest values are also located, and then the values gradually decrease near the edges [52].



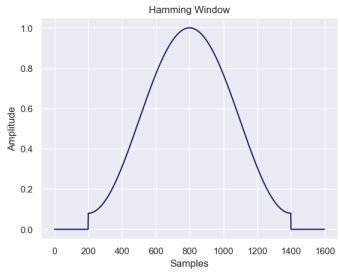
(a) Rectangular window.



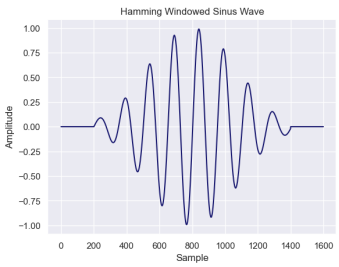
(b) Rectangular window on sinus signal.



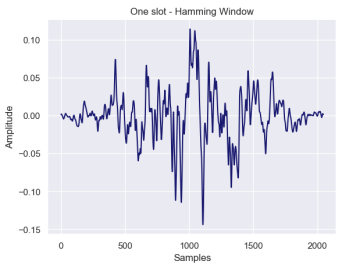
(c) Rectangular window on our data sample.



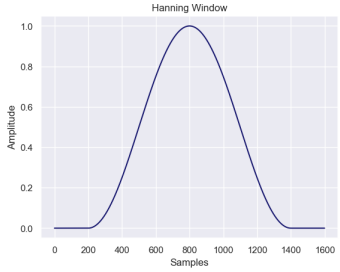
(d) Hamming window.



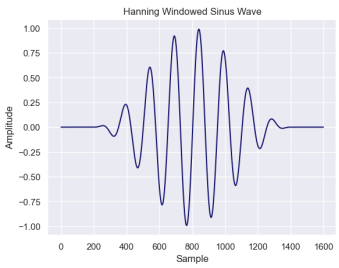
(e) Hamming window on sinus signal.



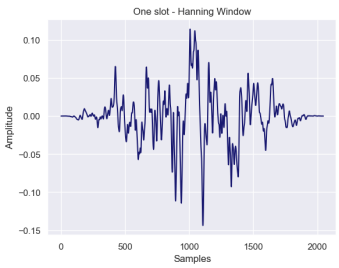
(f) Hamming window on our data sample.



(g) Hanning window.



(h) Hanning window on sinus signal.



(i) Hanning window on our data sample.

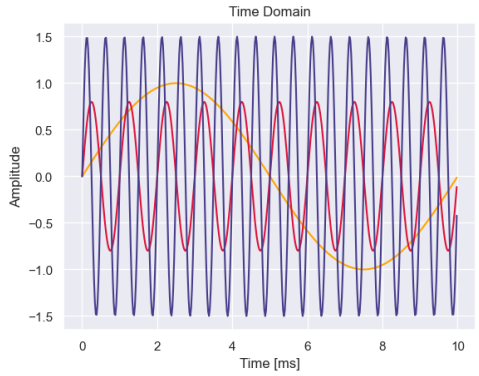
Figure 4.2: Window types and their effect on a sinus signal and our sound sample.

The hanning window will be the primary windowing function used for this project as it provides good frequency resolution and has the advantage of control-

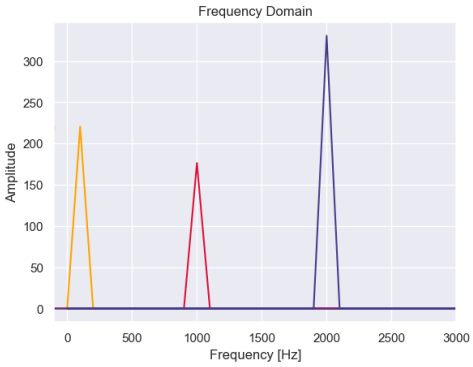
ling the spectral leakage [53]. One frame length is 43 ms and the window is moved by 11 ms (hop length).

4.1.2 Fourier Transform

The Fourier transform is a mathematical operation that converts a signal from the time domain to the frequency domain. Within the frequency domain, signal amplitudes are viewed as a function of frequency, rather than a function of time as seen in Figure 4.3 [52]. This is a very useful operation for processing sound signals from electric guitars, where all of the frequencies that make up a sound signal can be obtained.



(a) Sine waves in time domain



(b) Sine waves in the frequency domain.

Figure 4.3: Illustration of a signal in the time and frequency domain.

The continuous Fourier transform (CFT) is the fundamental method to convert a signal to the frequency domain. The CFT, as the name implies, is used to convert continuous signals from the time domain to the frequency domain by multiplying a signal $x(t)$ with an analyzing function $e^{-j2\pi Ft}$, that are sinusoids.

$$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi Ft}dt \tag{4.5}$$

The CFT calculates the integral of the signal from $-\infty$ to ∞, however in a practical scenario, signals are sampled using an analog-to-digital converter meaning that the signal is represented by a finite number of samples. To conduct the Fourier transform on a discrete set of samples, the discrete Fourier transform (DFT) is used as seen in Eq. (4.6).

$$F_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{j2\pi kn}{N}} \tag{4.6}$$

The DFT is slightly different from the CFT, as the samples are evaluated by a summation of all the samples $n = 0$ to $N - 1$, instead of evaluating at any

frequency. The DFT is restricted by a set of frequency bins F_k, which is determined by the number of samples. The exponent is also different from the CFT, as we cannot look at frequency and time continuously, thus $\frac{k}{N} = F$ and $n = t$.

The DFT equation can be simplified, by setting all the variables after the complex number j to $b_n = \frac{2\pi n}{N}$, as seen in Eq. (4.7).

$$F_k = \sum_{n=0}^{N-1} x_n \cdot e^{-jb_n} \tag{4.7}$$

The complex exponential can be rewritten using Euler's formula in Eq. (4.8).

$$e^{jx} = cosx + jsinx \tag{4.8}$$

Thus, the DFT at the k^{th} frequency-bin can be calculated using Eq. (4.9).

$$F_k = x_0 \cdot (cos(-b_0) + j \cdot sin(-b_0) + x_1 ... F_k = A_k + B_k j \tag{4.9}$$

The x-axis of the DFT is given in frequency bins, where each bin represents the amplitude of an interval of frequencies. The number of frequency bins can maximum be the number of samples in the samples, however, a common number of bins are 256 and 512. The output from the DFT will be a complex number, thus the magnitude of the output is taken, also called the power spectrum, which can be calculated using Eq. (4.10).

The power spectrum of a DFT can be calculated using

$$P = \frac{|DFT(x_i)|^2}{N} \tag{4.10}$$

where x_i is each element in the DFT and N is the number of bins.

The DFT is the fundamental method for converting a signal from the time domain to the frequency domain. This has been built upon to create new methods such as the short Fourier transform (STFT) and the fast Fourier transform (FFT).

However, a sound signal can change over time, and therefore performing a DFT over a long sound signal will in some cases not portray the sound signal accurately. The STFT performs many DFT operations on many small samples of the sound signal using a window function to slice the signal into many different windows.

The DFT has the complexity of N^2, as for each frequency bin F_k, N amount of samples are calculated. This is not a problem at low amounts of samples but can be processing-heavy at larger amounts. The FFT was created to speed up the processing time by re-writing the formulas and splitting up the calculations into different parts. The FFT has the complexity of $Nlog(N)$ and thus has a reduced amount of operations. It is therefore the most commonly used method of converting a signal from the time domain to the frequency domain nowadays. [52]

The discrete cosine transform (DCT) is a Fourier-related transform similar to the DFT, where the distinction lies in the fact that DCTs only use cosine functions while DFT uses both cosines and sines in the form of complex exponentials. The DCT expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. There are eight standard DCT variants, of which four are common. The most common variant of discrete cosine transform is the type-II DCT seen in Eq. (4.11). [54]

$$X_k = \frac{1}{2}(x_0 + (-1)^k x_{N-1}) + \sum_{n=1}^{N-2} x_n cos(\frac{\pi}{N-1} nk) \text{ for } k = 0, ...N-1 \tag{4.11}$$

The guitar sample shown in Fig. 4.1 has been transformed to the frequency domain using the STFT with 2048 frequency bins. The result of one frame can be seen on Fig. 4.4.

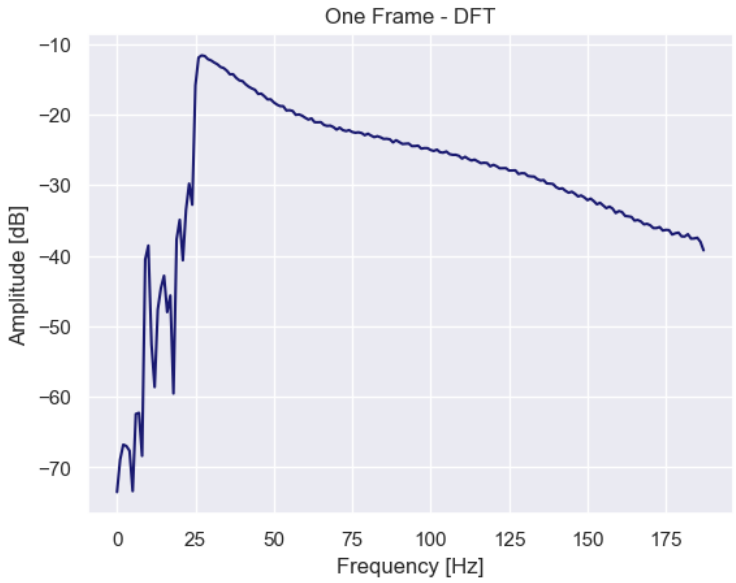


Figure 4.4: DFT of one frame from the guitar sample.

4.1.3 Spectrograms

A spectrogram is a method to visually represent the spectrum of frequencies in a signal through time. A spectrogram is usually portrayed in two dimensions with one axis representing the time in a signal and the other representing the frequency. The intensity of the frequency at that given time is portrayed as a heat map, where the intensity is visualized as color or brightness. Fig. 4.5 shows an example of a spectrogram. [52]

A spectrogram is created using the SFTF on the sound signal. Windowing is used to split the sound signal into many frames, where the magnitude of that frame, portrays the intensity of the frequency at that given frame. Each frame then corresponds to a vertical line in the spectrogram, and by performing multiple STFTs, multiple of these vertical lines can be laid side-by-side to form the spectrogram. The spectrogram of the guitar sample displayed in Fig. 4.1 is visualized in Fig. 4.5. [52]

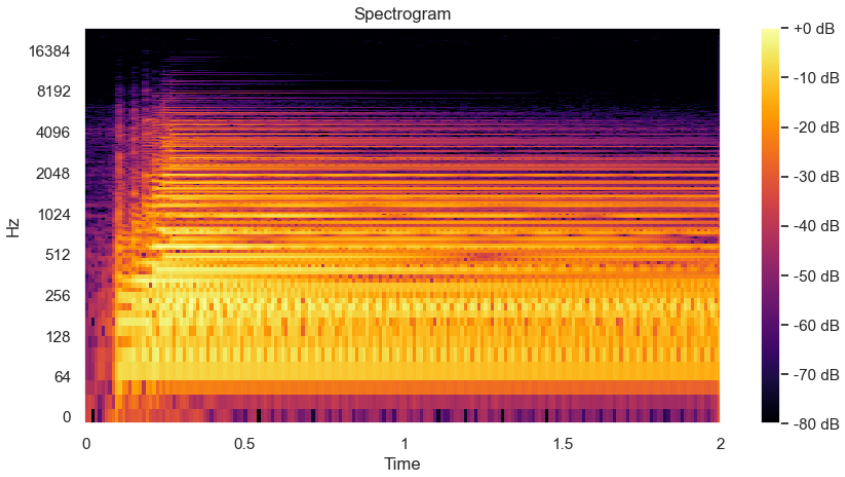


Figure 4.5: Spectrogram of the guitar sample shown in Fig. 4.1.

4.1.4 Mel Scale and Mel Filterbanks

The Mel scale is a scale that mimics the non-linearity in the human auditory system. In human hearing, perceived resolution decreases as frequency increases, e.g., humans are able to distinguish between 500 and 1000 Hz but the difference between 8000 and 8500 Hz is barely noticeable even though the difference is still 500 Hz. The Mel frequency scale is mostly linear below 1 kHz and becomes logarithmic above as shown in Fig. 4.6. [54]

The equation to convert a frequency from a sound signal to the Mel scale is shown in Eq. (4.12), and can likewise be converted back using Eq. (4.13).

$$m(f) = 2595 \cdot log_{10}(1 + \frac{f}{700}) \tag{4.12}$$

$$f(m) = 700(10^{m/2595} - 1) \tag{4.13}$$

Spectrograms can also be converted to a Mel spectrogram by converting the frequencies in the spectrogram to a Mel scale. To create this conversion filter banks are needed.

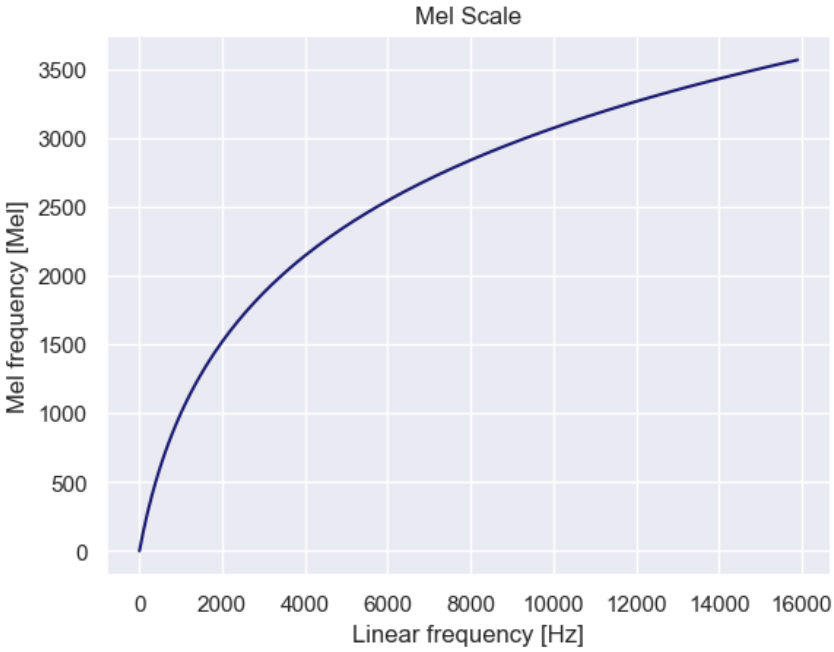


Figure 4.6: Illustration of the Mel scale. [54]

Filterbanks are a system where two or more band-pass filters are used together to analyze and process a given signal. A filter in the filterbank is triangular with the value of 1 at the center frequency and decreases linearly towards 0 at the edges, where it reaches the center frequencies of the two adjacent filters as seen in Fig. 4.7.

To create a Mel filterbank, a lower and upper frequency is needed to define the limit of the filterbank. These limits are usually the lowest and highest frequencies from the DFT. The lower and upper frequencies are then converted to Mels, and n amount of points are spaced linearly between the lower and upper Mel, where n is the desired amount of filterbanks. The Mels are then converted back to frequencies and define the center frequencies for the filterbanks. Lastly, these frequencies are converted to the nearest frequency bin, where the filterbanks can be created.

For example, let the lower and upper frequencies be set to 300 Hz and 8000 Hz, and convert them to Mels: 401.25 Mel and 2834.99 Mel. If the desired amount of filterbanks is set to 10, then 10 linearly spaced points are calculated between the lower and upper (12 points in total): 401, 622, 843, 1065, 1286, 1507, 1728, 194, 2171, 2392, 2613, 2834. The mels are converted back to frequencies: 300, 517, 781, 1103, 1496, 1973, 2554, 3261, 4122, 5170, 6446, 8000, and the nearest frequency bins are found, which for a DFT with 256 bins could look like the following: 9, 16, 25, 35, 47, 63, 81, 104, 132, 165, 206, 256. Now that every frequency bin has been calculated, the Mel filterbanks in the signal can be calculated using Eq. (4.14).

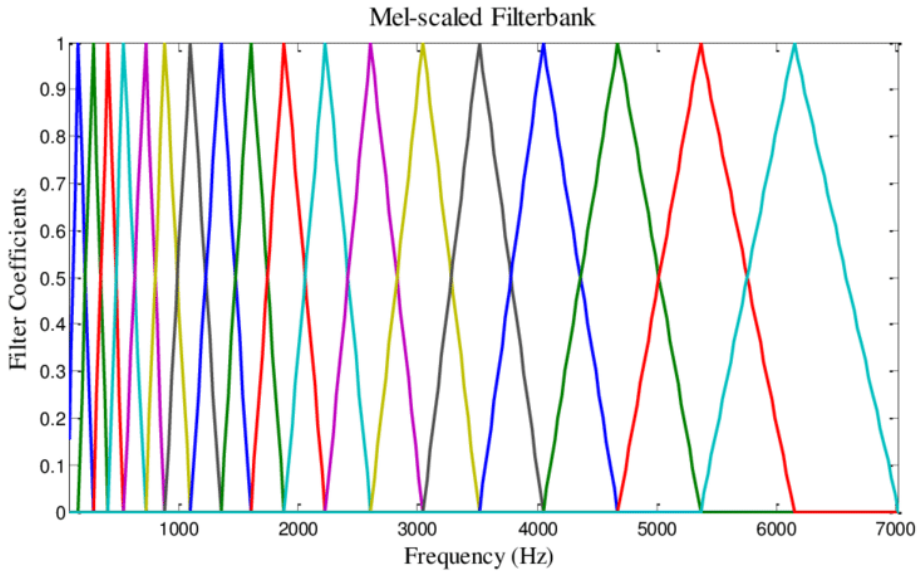


Figure 4.7: Mel filterbanks. [55]

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \leq k \leq f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \leq k \leq f(m+1) \\ 0 & k > f(m+1) \end{cases} \tag{4.14}$$

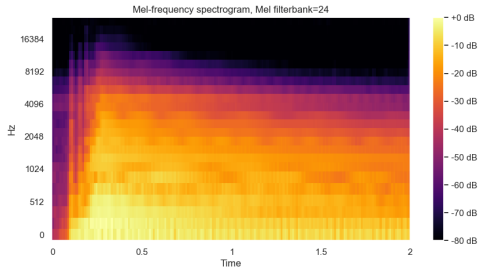
where k is the DFT frequency bin, m is the Mel for that given frequency bin, $m-1$ is the Mel at the previous frequency bin and $m+1$ is the Mel at the next frequency bin.

The amount of filters that can be chosen varies from implementation but is usually around 24-40 [54]. The filterbank can be applied to the power spectrum of the signal to obtain a Mel spectrogram that can be used for classification or further processed to extract MFCCs features.

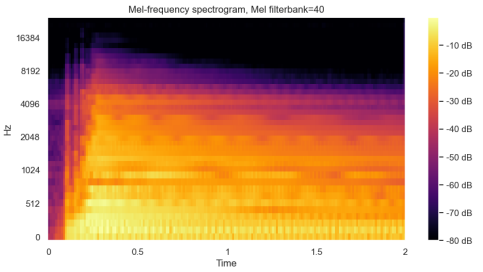
Four Mel spectrograms of the guitar sample are shown in Fig. 4.8, where 24, 40, 64, and 128 Mel filterbanks have been applied.

4.1.5 Mel Frequency Ceptrum Coefficients

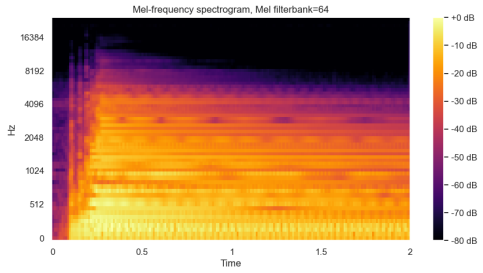
As presented in Section 2.1.4, timbre is defined as "that attribute of auditory sensation which enables a listener to judge that two nonidentical sounds, similarly presented and having the same loudness and pitch, are dissimilar". Thus, when looking at features capable of describing the different guitars and the unique differences between the models, MFCC is considered a good candidate because of



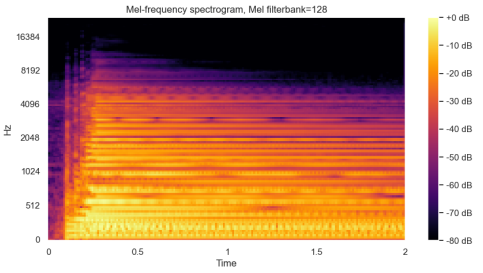
(a) Mel spectrogram with 24 filterbanks.



(b) Mel spectrogram with 40 filterbanks.



(c) Mel spectrogram with 64 filterbanks.



(d) Mel spectrogram with 128 filterbanks.

Figure 4.8: Mel spectrograms with different amount of Mel filterbanks of the guitar sample displayed in Fig. 4.1.

its ability to encode timbral information of a sound signal [56]. MFCCs are coefficients that make up the Mel-frequency cepstrum (MFC). MFCs are a representation of the short-term power spectrum of a sound, based on the DCT of a log power spectrum on the mel scale, which not only removes high-frequency ripples in the spectrum but also serves to decorrelate the frequency content of the coefficients [57]. MFCCs are generally used as features in sound classification systems since not all the frequencies are important when looking at the timbre of guitars, but rather the harmonics scaled to the human hearing in the higher frequencies.

MFCCs are commonly calculated using the following proceedings:

1. Perform the Fourier transform of a windowed signal.
2. Map the powers onto the Mel scale.
3. Calculate the logs of the power at each Mel frequency.
4. Perform the discrete cosine transform
5. Calculate the amplitudes of the resulting spectrum.

Some variations of this process include the addition of dynamic features such as delta and acceleration coefficients. They describe the instantaneous spectral enve-

lope shape of the signal and are used to further accentuate the temporal characters of the signal [58]. The delta coefficient is calculated using Eq. (4.15)

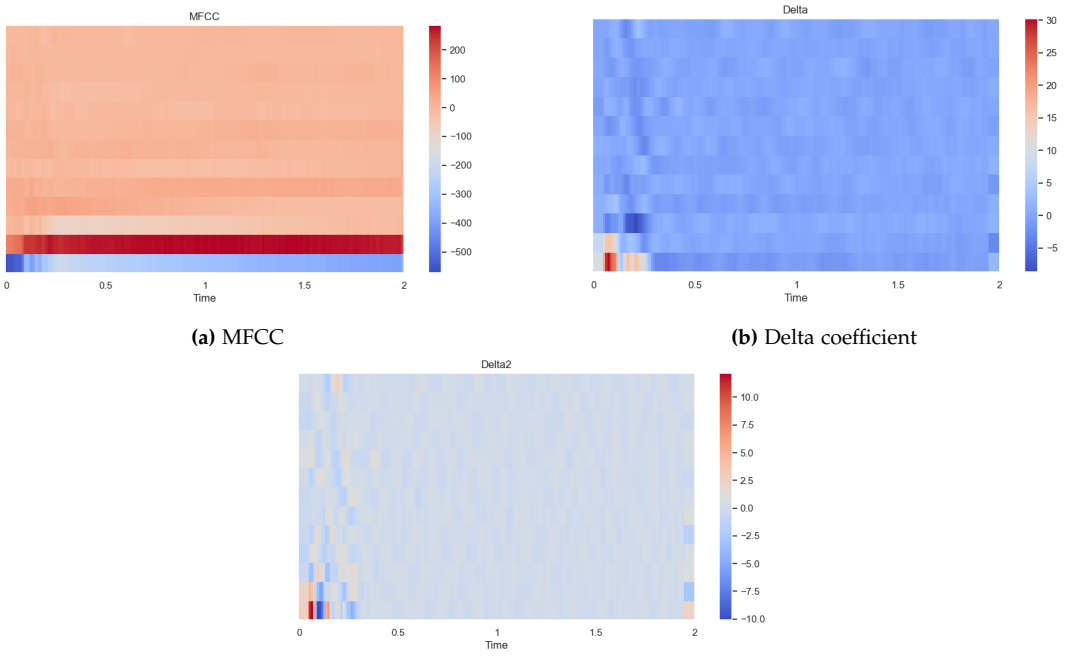
$$\Delta(C_t) = C_t - C_{t-1} \tag{4.15}$$

where C_t is the coefficient from the feature vector at time t. The acceleration coefficient, also known as the delta-delta, is calculated using Eq. (4.16).

$$Acc(C_t) = \Delta(C_t) - \Delta(C_{t-1}) \tag{4.16}$$

The number of MFCC usually ranges between 10-39, where the recommended amount is 13 coefficients where the delta and acceleration is found and added to the total amount of coefficients, which totals to 39 coefficients. [54]

The MFCC, delta and delta-delta coefficients of the input record can be seen on Fig. 4.9a.



(a) MFCC

(b) Delta coefficient

(c) Delta-delta coefficient

Figure 4.9: MFCC, delta, delta-delta coefficient

4.2 Classifiers

This section will present the different classifiers that will be used in this project.

4.2.1 k-Nearest-Neighbour

k-Nearest-Neighbour (kNN) is a supervised learning method for classifying samples by calculating the k-nearest neighbours from the test sample to nearby training samples. The kNN uses a predefined amount of features, where every training sample has been labelled in that feature space. For classification problems, voting[1] is used to predict the testing samples as the most frequent label within its k neighbours. The voting can be weighted by distance meaning that closer samples are assigned higher weighted. Furthermore, different distance formulas can be used for calculating the distance to its k-nearest neighbours. Commonly used distance formulas are the Euclidean and Manhattan distances [59]. Fig. 4.10 shows an example of how a testing sample is classified when different values of k are used.

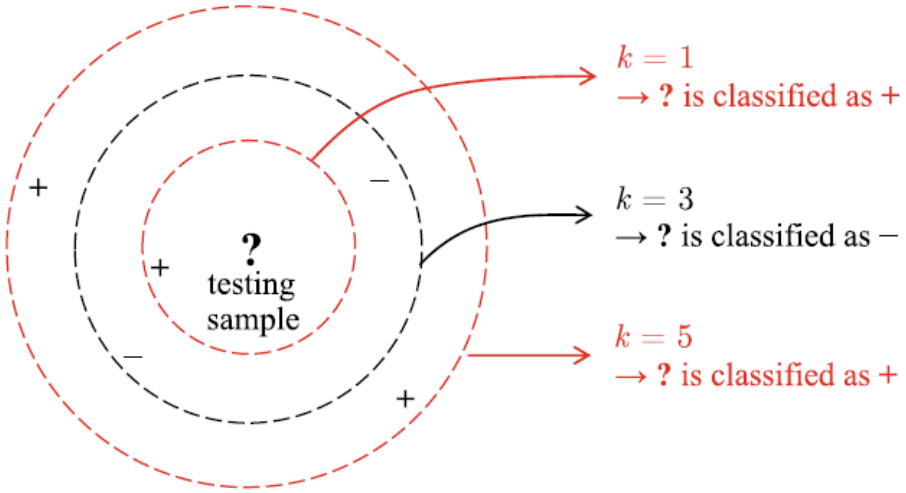


Figure 4.10: kNN predictions in cases when $k = 1$, $k = 3$ and $k = 5$ [59]

4.2.2 Support Vector Machines

Support Vector Machines (SVM) is a supervised learning algorithm that separates the samples by finding a hyperplane that best separates the classes in the sample space. It is mostly used for binary classification of data represented in n dimensional space. Another important aspect of this algorithm is that it is non-parametric, due to the number of support vectors can vary depending on the complexity of the data. The SVM performs the classification by drawing a hyperplane and points on one side are classified as the first class and the rest of the points are

[1] method for classifying the sample

classified as the second class. While there could be multiple hyperplanes dividing the data, SVM tries to find the most optimal one using what is called the support vectors and margin. [59]

- Support vectors: points that are close to the decision boundary in the margin edge and satisfy Eq. (4.17)

$$w^T * x + b = 0 \tag{4.17}$$

- Margin: maximum distance between the supporting vectors. The decision boundary is located in the middle of the support vectors.

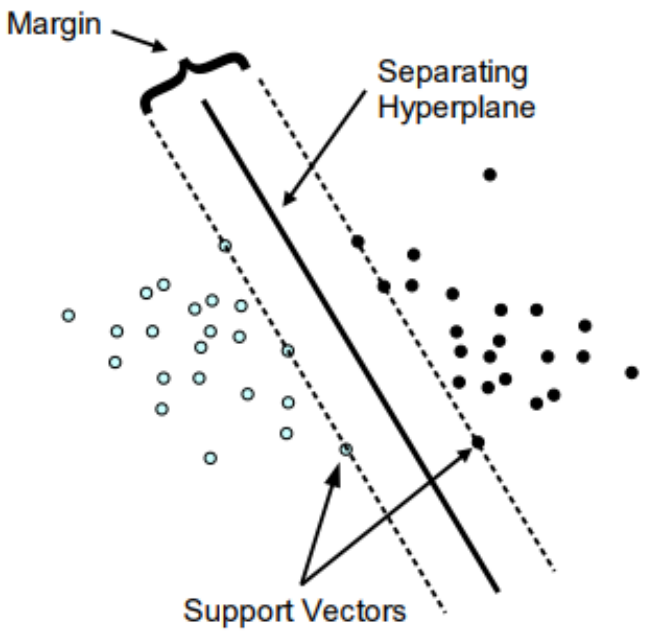


Figure 4.11: SVM in binary classification[60]

Fig. 4.11 shows a binary classification where the margin and the support vectors are represented along with the final decision boundary. The support vector algorithm simply looks for the separating hyperplane with the largest margin and this can be achieved with equation Eq. (4.18), subject to constraints in Eq. (4.19)

$$\min_{w,b} \frac{1}{2}\|w\|^2 \tag{4.18}$$

$$s.t \;\; y_i(w^T \cdot x_i + b) \geq 1 \;\; \forall i \tag{4.19}$$

This can be formulated and solved using the primal Lagrangian formulation in equation Eq. (4.20), where it is minimized with relation to w and by setting the

derivatives of both variables to zero.

$$L_{p(w,b,\alpha)} \equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i y_i (w^T \cdot x_i + b) + \sum_{i=1}^{N} \alpha_i \tag{4.20}$$

The difficulty comes into play when the given data is non-linearly separable as shown in Fig. 4.12. The idea is to turn it into a linear problem by increasing the dimension using the Kernel trick. The Kernel trick augments the data with some nonlinear features that are computed from the existing ones, then find the separating hyperplane in this higher dimensional space and to end, projects back to the original space [61].

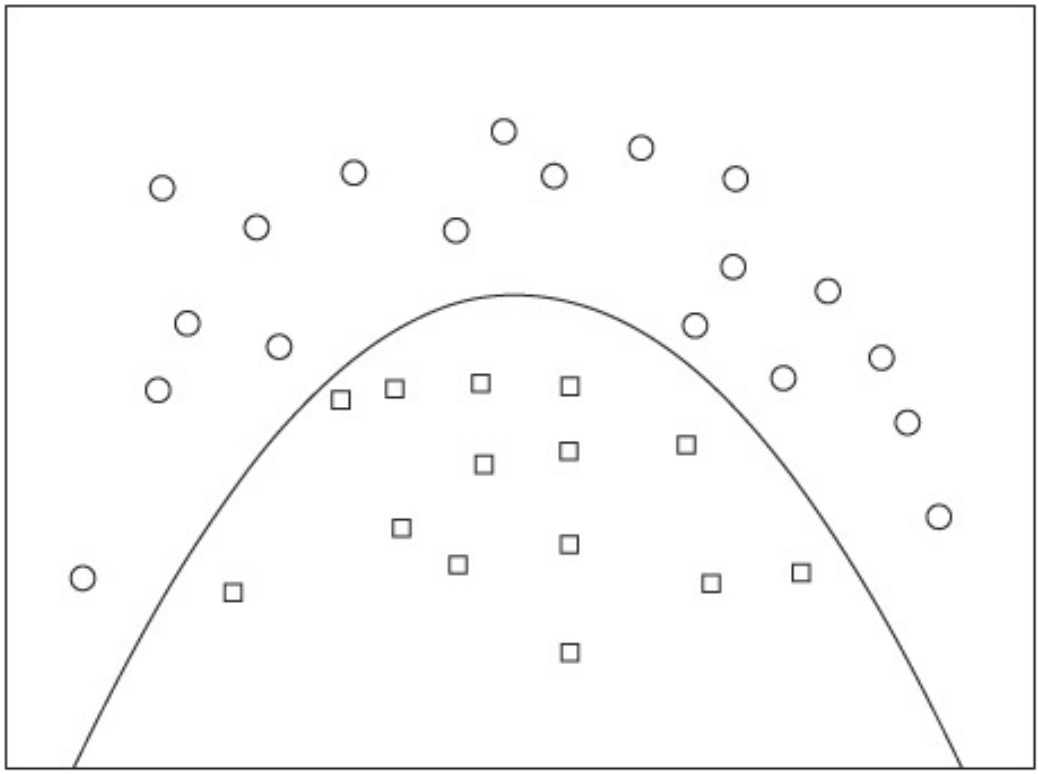


Figure 4.12: SVM nonlinear binary classification [62]

4.2.3 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a linear method, originally used for binary classification problems (Fisher's Linear Discriminant). LDA is used for classification by reducing the dimensionality of samples in the sample space where the mean between classes is maximized and the scatter within classes is minimized.

First, LDA projects training samples onto a line, so that samples from the same class are close to each other and the ones from different classes are as far as they can be from each other. To classify a new sample, it is projected to the same line and the location of the projection will determine the sample's class [59].

Using a mathematical approach, the sample data is considered as a D dimensional input vector x and projected into 1 dimension.

$$y = \mathbf{w^T x} \tag{4.21}$$

The simplest way to calculate $\mathbf{w}$ in a 2 class problem, is where class C_1 has N_1 points and class C_2 has N_2 points. Here, the mean vector of the 2 classes, $\mathbf{m_1}$ and $\mathbf{m_2}$ are calculated.

$$\mathbf{m_k} = \frac{1}{N_k} \sum_{n \in C_k} x_n \tag{4.22}$$

where $k = 1, 2$. Then, the separation of the classes is maximized when projected onto w, which can be calculated as :

$$m_2 - m_1 = \mathbf{w^T}(\mathbf{m_2} - \mathbf{m_1}) \tag{4.23}$$

At this phase, $\mathbf{w}$ could be determined easily, but the only problem is that the classes still could overlap. Fisher's idea to solve this problem was to *maximize a function* that gives a large separation between the projected classes and small variance within each class, as shown on Fig. 4.13. Within the class, variances can be calculated as :

$$s_n{}^2 = \sum_{n \in C_k} (y_n - m_k)^2 \tag{4.24}$$

and that gives the total within-class variance for the whole dataset $s_1{}^2 + s_2{}^2$ in a 2 class case. Fisher criterion is defined by the ratio of the between-class to the within-class variance.

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1{}^2 + s_2{}^2} \tag{4.25}$$

This can also be rewritten as

$$J(\mathbf{w}) = \frac{\mathbf{w^T S_B w}}{\mathbf{w^T S_W w}} \tag{4.26}$$

where $\mathbf{S_B}$ is between-class covariance matrix,

$$\mathbf{S_B} = (\mathbf{m_2} - \mathbf{m_1})(\mathbf{m_2} - \mathbf{m_1})^\mathbf{T} \tag{4.27}$$

and $\mathbf{S_W}$ is total within-class covariance matrix.

$$\mathbf{S_W} = \sum_{n \in C_1} (x_n - \mathbf{m_1})(x_n - \mathbf{m_1})^\mathbf{T} + \sum_{n \in C_2} (x_n - \mathbf{m_2})(x_n - \mathbf{m_2})^\mathbf{T} \tag{4.28}$$

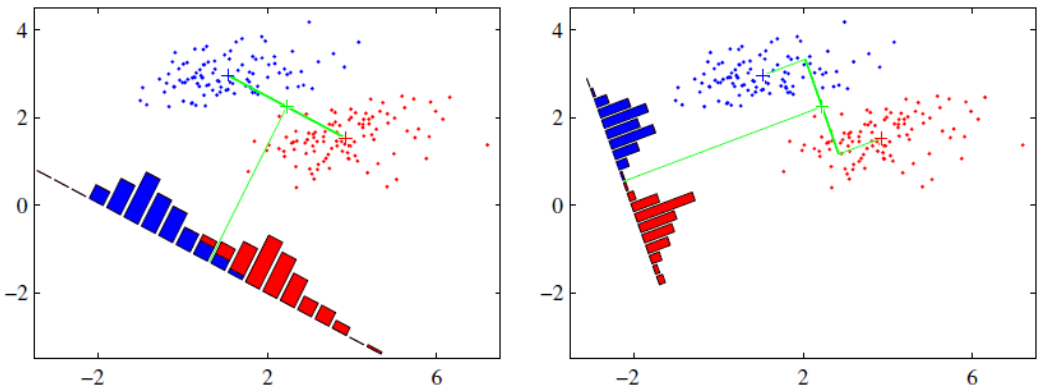


Figure 4.13: Within-class distributions and between-class distances with different projection lines [61].

After maximizing $J(\mathbf{w})$, $\mathbf{w}$ can be determined [61].

$$\mathbf{w} \propto \mathbf{S_W}^{-1}(\mathbf{m_2} - \mathbf{m_1}) \tag{4.29}$$

This example described a 2 class problem, nevertheless, LDA can be extended for multiclass cases. Let N be the number of classes and the iîth class has q_i samples. First, the global scatter matrix is defined as

$$\begin{aligned} S_t &= S_b + S_w \\ &= \sum_{i=1}^{q} (x_i - m)(x_i - m)^T \end{aligned} \tag{4.30}$$

where m is the mean vector of all the samples. The within-class scatter matrix S_W is redefined as the sum of all the scatter matrices for each class,

$$S_w = \sum_{i=1}^{N} S_{w_i} \tag{4.31}$$

where

$$S_{w_i} = \sum_{x \in X_i} (x - m_i)(x - m_i)^T \tag{4.32}$$

Eq. (4.30) can be rewritten as

$$\begin{aligned} S_b &= S_t - S_w \\ &= \sum_{i=1}^{N} q_i (m_i - m)(m_i - m)^T \end{aligned} \tag{4.33}$$

Multiclass LDA can be implemented in many different ways by choosing any two of three scatter matrices: S_b, S_w or S_t. A common implementation is to optimize the object [59]

$$\max_w \frac{tr(w^T S_b w}{tr(w^T S_w w)} \tag{4.34}$$

where $w \in R^{d\times(N-1)}$ and $tr(\cdot)$ is the trace of a matrix.

$$S_b W = \lambda S_w w \tag{4.35}$$

On the other hand, LDA is also a typical supervised dimensionality reduction technique [59].

4.2.4 Convolutional Neural Network

Convolutional Neural Network (CNN) is a method to classify the contents of an image. Unlike the previous methods, this is a deep learning method typically used on images to extract relevant features and thereby classify the image. However, CNNs can also be used on sound signals as the spectrogram can be treated as an image as it has two dimensions and an intensity normally in the form of a heat map. The CNN used for this project is based on the TabCNN mentioned in Section 2.6 and has the architecture shown in Fig. 4.14.

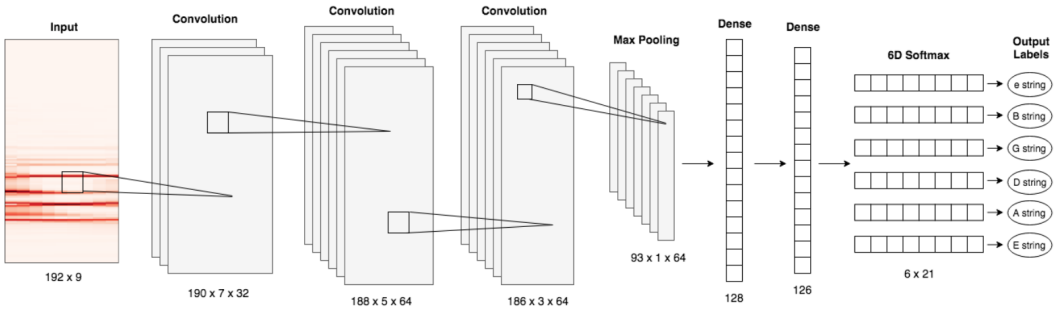


Figure 4.14: The architecture of the TabCNN.[35]

The original architecture is composed of three convolutional layers where the first one takes a grayscale image and applies 32 filters with a kernel size of 3x3. Next, a ReLU activation function is used and afterwards this layer, the output is fed to the next convolutional layer and this process is repeated with the rest of them, where they will be composed of 64 filters. Once all of the convolutional layers are done, a max pooling layer has been used with a kernel size of 2x2 to reduce the dimensionality of the data. After this max pooling process, the output obtained will be an image of size 93x1. The next step will be followed by 2 fully connected

layers where it is attempted to flatten the data in an array of sizes 128 and 126 respectively. Thereafter, six separate linear layers with the appropriate number of output units have been used to predict the probabilities for each class. Softmax is used as the last activation function in the neural network to normalize the output to a probability distribution with a value between 0 and 1 to determine which class has the highest probability of being predicted. Moreover, the cross-entropy loss is then used to calculate a loss after every epoch.

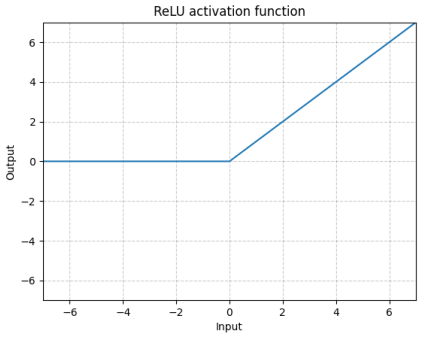
Rectified Linear Activation Function

Rectified linear activation function (ReLU) is an activation function, which returns 0 for any given negative value but for any positive value returns the same exact value, as seen in as seen in Eq. (4.36) and illustrated in Fig. 4.15a. Activation functions are used to prevent linearity, without them the output would be always a linear function [63].

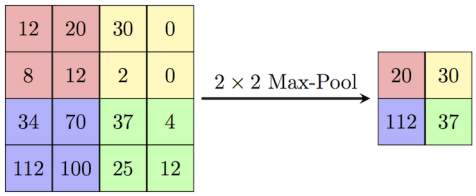
$$f(x) = \max(0, x) \tag{4.36}$$

Max Pooling

Max pooling is used to downsample an image by running a kernel on the image, typically a 2x2 kernel, where the maximum value in the kernel is stored as the output. Max pooling is useful to reduce computational costs and preserve useful information when performing convolutions. Fig. 4.15b illustrates a 2x2 max pooling that halves the input dimensions. [59]



(a) ReLU function that outputs the input value if it is positive, otherwise it passes 0. [64]



(b) A 2x2 max-pooling performed on a 4x4 input image reduced to a 2x2 output image. [65]

Figure 4.15: ReLU and max pooling.

Soft-max Activation

The soft-max activation function is used at the end of the architecture to normalize the input into a probability distribution. Prior to applying soft-max, some vector components could be negative, or larger than one, resulting in the sum not being 1. After applying soft-max, each component will be in the interval $(0, 1)$, and the sum of the components will be 1, allowing it to be interpreted as probabilities. Eq. (4.37) shows the equation for soft-max, where $\vec{x}$ is the input vector to the soft-max function, x_i are the elements in the input vector, e is the standard exponential function applied to each element in the input vector, K is the number of classes, and the bottom term is the normalization term ensuring all output values of the function will sum to 1. [63]

$$\sigma(\vec{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{4.37}$$

Cross Entropy Loss

Cross-entropy is a loss function commonly used in multi-class classification for optimizing the model. Loss functions are used to compute the difference between the current output and the expected output of the model. It evaluates the model on the data, where the ideal model will have a loss as close to 0 as possible. Eq. (4.38) shows the formula to calculate the loss for a given sample in the dataset, where two different distribution functions are used: the predicted probability q(x) and the true probability p(x). When calculating cross-entropy for classification tasks, the base-e or natural logarithm is used. At the end of the training, the given cross-entropy loss will be an average across all of the training data. [66]

$$H(p, q) = -\sum_{x \in classes} p(x) \log q(x) \tag{4.38}$$

Chapter 5

Training of Classifiers

This chapter will present how the four classification methods presented in Section 4.2: KNN, SVM, LDA and CNN have been implemented and trained on the guitar data set.

For the different classifiers implemented, it is necessary that the duration of the data samples used are equally long in order to scale them into the same feature space. However, as presented in Chapter 3, the duration of the sound samples vary between 2-6 seconds. Hence, the duration of the data samples has been shortened to the first 2 seconds to include all the recorded guitar samples. For training, validation, and testing of the various classifiers, only guitars 1-10 presented in Table 3.1 have been used. Guitar 11 (Fender SC) has been omitted, as it is used separately to test how well the classifiers generalize on unseen data. Guitar 11 was selected, as the dataset is limited in terms of the other types of guitars: LP, SG and TC. Thus, choosing one guitar from these three guitar types to be omitted from the training set would remove half of the data samples, which would result in less data for training the classifiers, and impact the performance and ability to generalize.

5.1 The KNN and SVM model

This section will present how the KNN and SVM models have been implemented in this project. These models use MFCCs as the feature space, since it not only compresses the data of a Mel-spectrogram into a smaller number of coefficients, and has strengths in describing the timbre of an audio sample, but the change over time can also be described by using the delta and delta-delta values of the MFCCs. These MFCCs are added to the total feature space for one strum, so the total number of MFCCs increases. When implementing the KNN and SVM models, the feature space, parameters, and efficiency[1] of the model has to be taken

[1]The time it takes to train, validate, and predict with new data points, as well as the accuracy of the model.

into consideration, but will not be the dictating factor for the chosen model.

The input feature vector which has to be fed to the model has to be of two dimensions. Firstly, the number of data points and secondly a feature vector describing that feature point, ergo *(number_samples, number_features)*. To re-scale the two dimensional MFCCs to a one dimensional vector, the product of the two dimensions is calculated and saved into a new vector. This vector is now a one dimensional vector representation of the MFCCs from one strum and can be used for training and testing the vector.

Since each audio strum is cut to be two seconds long, with a sample rate of 44.1kHz, a window size of 43ms, a window step of 11ms, and 2048 FFT bins, 6747 features can be expected when training and testing the model. This can be used to correct or inform the user if the training or test samples are insufficient. Therefore, if a sample strum has less than 6747 features (which is the number of features for the MFCCs for a two seconds clip after converting it to a one dimensional array) it is ignored in the training and testing process to avoid creating any bias or wrong predictions. This applies to both KNN and SVM.

As mentioned in Chapter 4 there are many parameters that are set when training the KNN and SVM models. Thus, the approach to training the best possible models is to train by iteratively testing different combinations of parameters until it finds the best model score. The parameters are given as a grid of parameters on which the model can be trained and tested. The grid parameters can be set to any value as long as it makes sense within the context of the model that is being trained. In the context of the KNN model, the grid parameters are the k-nearest neighbours, the weights, and the distance formula as seen in Table 5.1, Table 5.2 and Table 5.3 respectively. The parameters which are inside the grid parameters for the SVM model are the kernel, regularization and gamma values as seen in Table 5.4, Table 5.5 and Table 5.6 respectively. When using the linear kernel the gamma parameters are not used, as they only apply to the RBF kernel. As mentioned in Chapter 3 there are five classes, each having their own labels. Therefore, the KNN and SVM models have been trained five times each, to create a model for every class in the dataset. Furthermore, this is done for both with and without LDA applied, which totals 20 models.

k-neighbours
3 | 5 | 7 | 9 | 11 | 15

Table 5.1: Table listing the different k-neighbours used for training the KNN model.

Weights
Uniform | Distance

Table 5.2: Table listing the different weights used for training the KNN model.

The dataset has been split into an 80/20 training and testing split for these models. However, since the dataset has a limited size, k-fold cross validation is

Distance Formula	
Euclidean	Manhattan

Table 5.3: Table listing the different distance formulas used for training the KNN model.

Kernels	
RBF	Linear

Table 5.4: Table listing the different kernels used for training the SVM model.

Regularization						
100	10	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}

Table 5.5: Table listing the different regularization values used for training the SVM model.

Gamma						
100	10	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}

Table 5.6: Table listing the different gamma values used for training the SVM model.

used for training to keep as much information as possible while still trying to avoid overfitting. The number of folds can be adjusted when training, but the default value of 5-fold works for this purpose.

5.1.1 LDA Model

The LDA model has been implemented for three purposes in this project. Firstly, to visualize the spread of the dataset, as to get an understanding of how well the different classes would cluster. Secondly, to train a model using solely LDA and lastly, to train a KNN and SVM model in combination with LDA.

The visualization of the spread in the dataset was done by applying dimensionality reduction to the MFCCs of each strum and finding the best linear discriminants that would minimize within-class scatter and maximize between-class scatter, as seen in Fig. 5.1 for the guitar type class. After applying the dimensionality reduction and seeing how effectively LDA separated the classes, the reduced dimensionality dataset was then also used with the predictor that the LDA solution comes with. This predictor applies Bayes's theorem on each training sample, where a k-class is chosen that maximizes the posterior probability for a sample.

To train the LDA model, the parameters chosen were determined by the small dataset, as the LDA model uses Shrinkage [2] which is only effective if either of two solvers is used. Therefore, the solver for the LDA is set to be the eigen solver which is based on an optimization ratio between the between-class scatter and

[2]Shrinkage is a regulator that improves the estimation of the covariance matrices when the amount of training samples is small compared to the number of features. It essentially improves the generalization of the estimator [67].

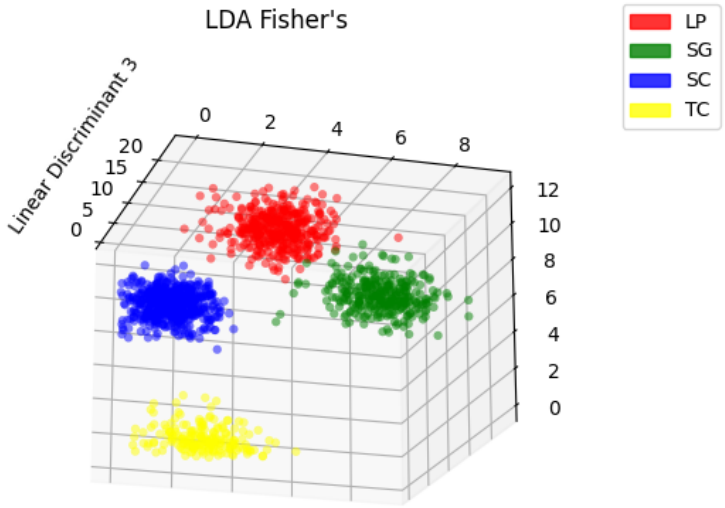


Figure 5.1: LDA dimensionality reduction applied to the dataset. The dimensionality is reduced to three linear discriminants.

within-class scatter. Similar to the KNN and SVM methods, five models have been trained for the LDA, where each one has been trained for a different class inside the dataset.

5.2 CNN

This section will present how a CNN architecture has been implemented and trained.

The CNN architecture used for this project is based on the TabCNN architecture described in Section 4.2.4, where modifications have been made to suit our data samples. An overview of the CNN is shown in Fig. 5.2, where the pipeline starts with a sound sample, which is transformed into a Mel spectrogram. Next, this Mel spectrogram is used as input to the CNN architecture where feature maps are extracted, and lastly, the labels are classified.

As presented earlier in this chapter, the duration of each sound sample is shortened to 2 seconds. This results in the Mel spectrograms having the dimension of 128x173, where 128 is the amount of Mels chosen and 173 is the amount of STFT windows. Hence, the input of the layers will have a different size compared to the original architecture, and the size throughout the neural network will likewise be different. After the Mel spectrogram has been through the three convolutional layers and a max pooling layer, the input will be 61x83 with 64 features, which will be fed into two linear layers. Another modification is applied in the final layers, where this time instead of focusing on having five classes to classify the tabs on

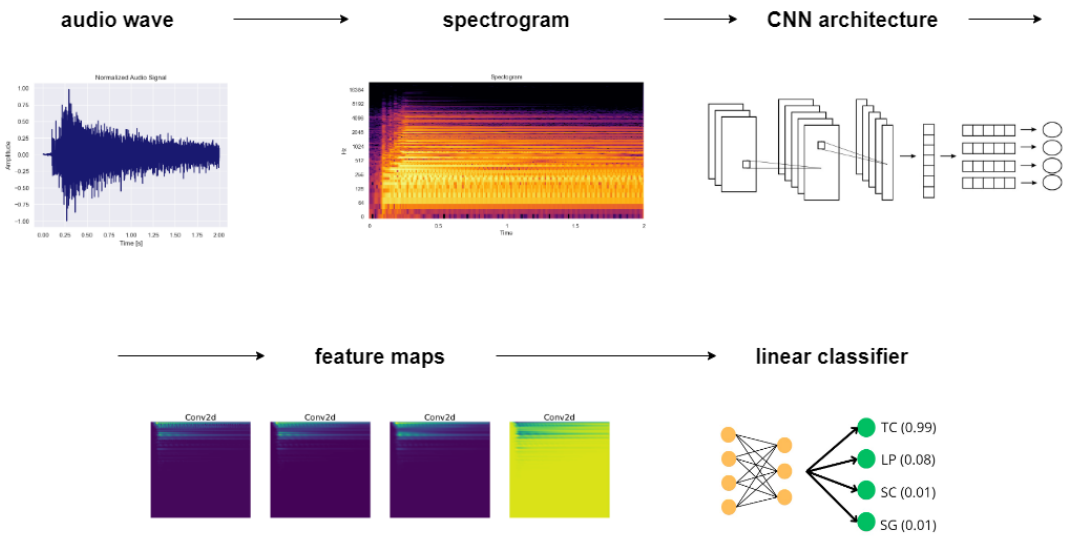


Figure 5.2: Overview of the CNN implementation.

each string, they have been modified to the five classes mentioned in Chapter 3.

5.2.1 CNN Training

The TabCNN model has been trained by inputting the Mel spectrogram of a sound sample along with its respective labels to the model. The dataset has been split to a 70/20/10 split for training, validation, and testing, respectively, using guitars 1-10 presented in Table 3.1. To find optimal parameters for training the model, a combination of different learning rates, batch sizes and optimizers have been used as seen in Table 5.7, Table 5.8, and Table 5.9. There are a total of 108 different parameter combinations. For learning rates 10^{-2}, 10^{-3}, and 10^{-4} the models have been trained for 100 epochs, which was adequate for the models to converge. However, for learning rate 10^{-5}, the models failed to converge within 100 epochs and thus, the training was extended to 200 epochs for these models. The CNN models have been trained 108 times and every model outputs the training, validation and testing loss along with the testing accuracies for each of the five classes. All of the 108 tests are seen in Fig. 5.3, which visualizes the different combinations of the parameters and the testing loss for each model.

Learning Rates			
10^{-2}	10^{-3}	10^{-4}	10^{-5}

Table 5.7: Table listing the different learning rates used for training.

Batch Sizes		
32	64	128

Table 5.8: Table listing the different batch sizes used for training.

Optimizers							
Adam	Adadelta	Adagrad	AdamW	NAdam	RAdam	SGD	RMSprop	Rprop

Table 5.9: Table listing the different optimizers used for training.

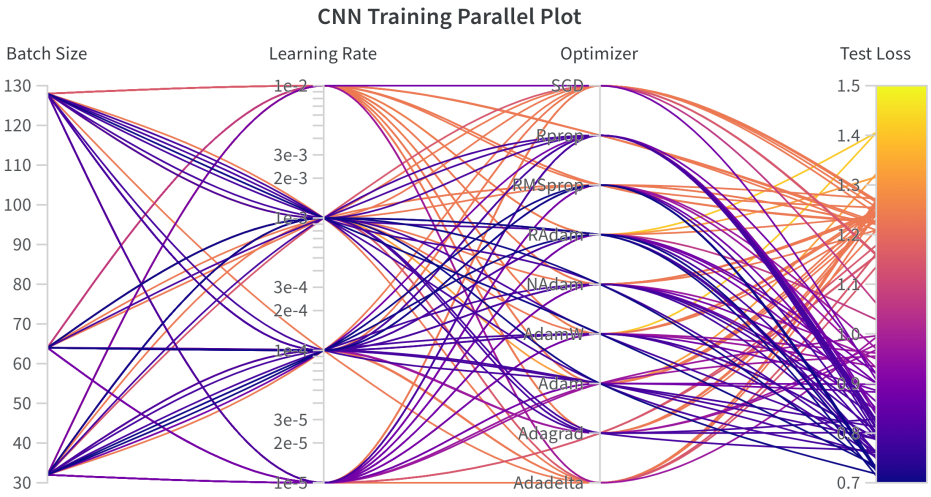


Figure 5.3: All of the 108 trained models are visualized as a parallel plot that shows every parameter combination and the testing loss.

5.3 Real-time System

As mentioned in Chapter 1, it is a key success factor that the AI system is fast and precise to be able to run in real-time. Therefore, this section will present how the methods described earlier in this chapter are integrated into a real-time web-based system and how the system works. The aim of the real-time system is to test the models in a 'real world' scenario, meaning that a user without programming knowledge also could use our models to predict electric guitar classes.

To visually demonstrate the predictions of the classifiers in real time, a web-based system was created using Streamlit[3]. The system was chosen to be web-based and not desktop-based to make it more convenient to run across different environments/OS.

In the real-time system, the audio data is first recorded. Recording audio works almost identically to the setup described in Section 3.1.1, where the sound of the electric guitar goes through a pedal, a sound card, and the sound card is connected

[3]https://streamlit.io/

through a USB to a PC. Instead of processing/saving the sound through Audacity, the system temporarily saves a .wav file locally. The length of the recording is 5 seconds, where silence is removed the same way as mentioned in Chapter 3 and the recording is shortened to 2 seconds to align with the data input for the models. The plot of the recording is displayed in Fig. 5.4.

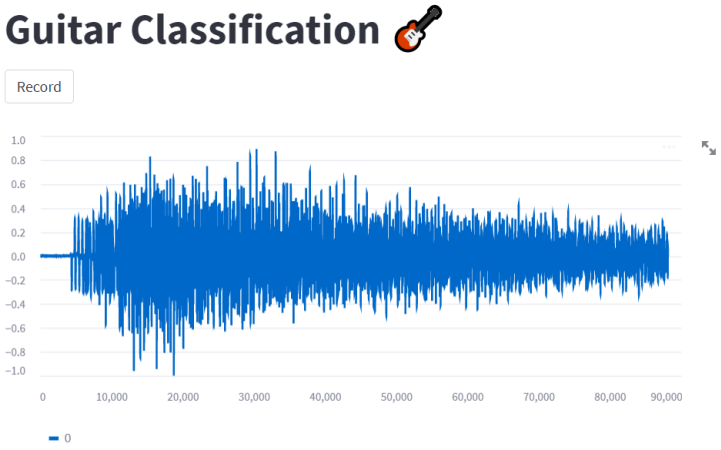


Figure 5.4: A recording of a strum visualized in the real-time system.

From the recording, different features are extracted depending on the demands of each model. The input of the KNN, SVM, and LDA models are MFCCs, whereas the input to the CNN is the Mel spectrogram of the recording, as previously mentioned in this chapter. After the features are extracted from the audio record, the models process the data. When the models finish the prediction, the predictions are displayed on a localhost connection created by Streamlit. Each model is trained to predict the guitar type, the pickup type, the pickup position, the strumming type and the player, as seen on Fig. 5.6. The complete pipeline is shown in Fig. 5.5. The model with the best performance is used in the system. The best result will be introduced in Section 6.5.

For the system to run in real time, multiple variables affect the execution time, such as the duration for recording the sample, the size of the model, and the extraction of features. The size of the different models is displayed in Table 5.10.

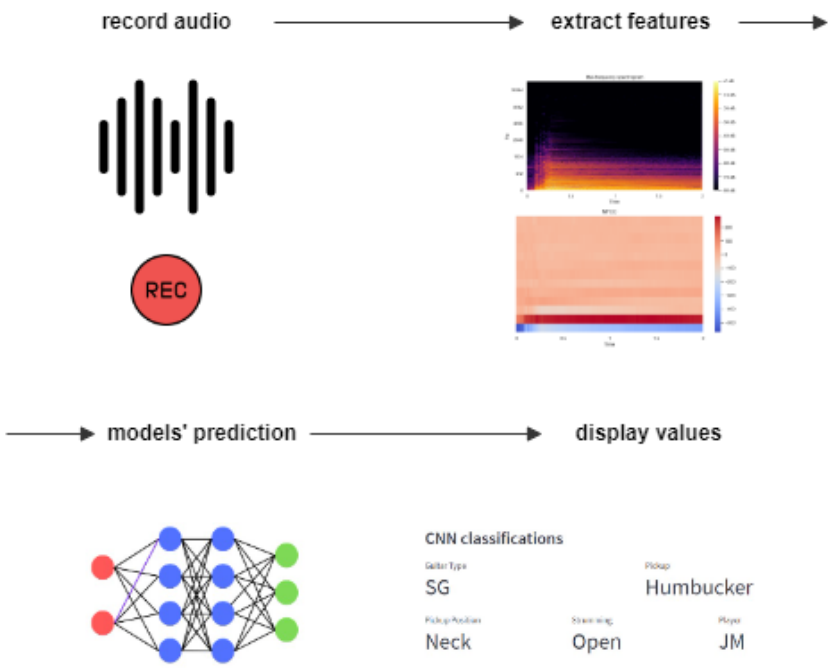
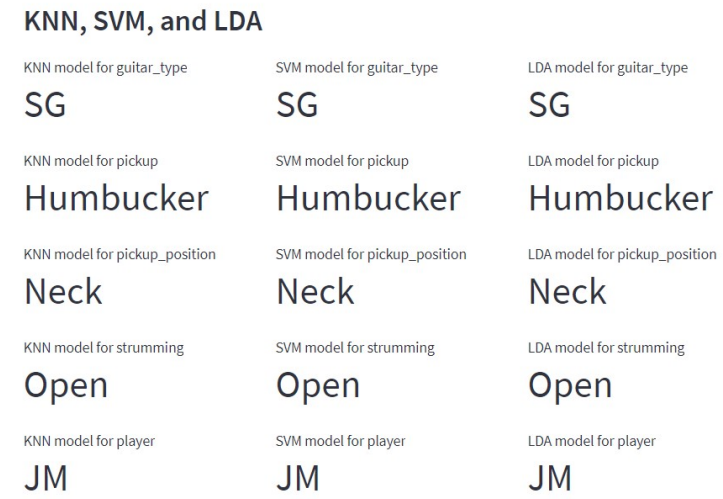


Figure 5.5: The pipeline of the real-time system.

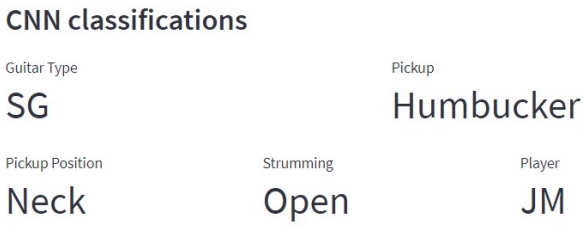
Model Size (kB)

Model	Guitar Type	Pickup Type	Pickup Position	Strumming	Player	Average
KNN (LDA)	97	49	73	49	97	73
KNN	38,754	38,754	38,754	38,754	38,754	38,754
SVM (LDA)	76	23	46	23	35	40.6
SVM	47.229	29.220	42.152	8.339	57.499	36,887
LDA	711.705	711.441	711.599	711.441	711.915	711,620
CNN	533,804	533,804	533,804	533,804	533,804	533,804

Table 5.10: The model size for every class in every model.



(a) Predictions of KNN, SVM and LDA.



(b) Predictions of CNN

Figure 5.6: Predictions of the different models display on Streamlit

Chapter 6

Results

This chapter will present the results for the four models that have been trained. To test the capabilities of the AI system, the models have been tested on two different testing datasets. The first set consists of the testing data from the training, validation and testing split from guitars 1-10 mentioned in Chapter 5. These are all samples that the models have not seen prior to testing, however, they have seen the same guitar when training. The second set consists solely of data samples from guitar 11 presented in Table 3.1. This test set has been created to evaluate how well the model performs and generalizes on unseen guitar data. Furthermore, the confusion matrices for all of the models can be seen in Appendix 6.1.

6.1 Performance Metrics

This section describes the different metrics used for evaluating the performance of the models. Fig. 6.1 shows an example of a confusion matrix with the true positives (TP), true negatives (TN), false negatives (FN) and false positives (FP).

6.1.1 Terms

The metrics are calculated using the confusion matrices for every class that shows the number of correct predictions, wrong predictions and how the wrong predictions were misclassified. For the following examples, the positive class is the LP, and the negative class is the SC.
True Positives (TP) are determined as the amount of correct predictions. I.e a sound sample is predicted as a LP and the groundtruth is also a LP.

True Negatives (TN) are sound samples outside of the given class that have been predicted correctly. I.e a sound sample is not predicted as a LP and the groundtruth

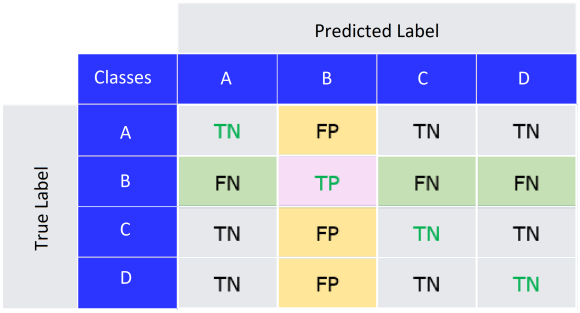


Figure 6.1: Example of a confusion matrix with TP, TN, FN and FP.

is also not a LP.

False Negatives (FN) is when the negative class was wrongly predicted as the positive class. I.e a sound sample has the groundtruth of a LP but is predicted as a SC.

False Positives (FP) is when the positive class has been wrongly predicted as a negative class. I.e sound sample is predicted as an LP, but the groundtruth is an SC.

6.1.2 Metrics

Many metrics exist for evaluating the model. However, for this project only precision, recall, and F1-score will be used. [68]

Precision, also known as positive predicted values (PPV), is a metric that describes how good the model is at not misclassifying the positive class as other classes.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6.1}$$

Recall, also known as Sensitivity (Sens), describes how good the model is at not misclassifying other classes as the positive class.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6.2}$$

F1-Score describes how many times the model correctly predicts a class across the entire dataset by taking their harmonic mean of precision and recall. An F1-score will be 1 if perfect recall and precision is obtained.

$$\text{F1-score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{6.3}$$

6.2 KNN Results

This section will present the results for the KNN classifier with, and without LDA.

6.2.1 KNN Models Best Parameters

The KNN models were trained with various parameters as mentioned in Chapter 5, where the best performing KNN with LDA and without LDA have the parameters as seen in Table 6.1 and Table 6.2 respectively. To determine the best performing model, each configuration of the parameters was evaluated using F1-score. There is one model for each class, where some of the parameters have multiple values that are equally good.

Class	k-neighbours	Distance Formula	Weights
Guitar Type	3/5/7/9/11/15	Euclidean/Manhattan	Uniform/Distance
Pickup	3/5/7/9/11/15	Euclidean/Manhattan	Uniform/Distance
Pickup Position	3/5/7/9/11/15	Euclidean/Manhattan	Uniform/Distance
Strumming	3/5/7/9/11/15	Euclidean/Manhattan	Uniform/Distance
Player	11	Manhattan	Distance

Table 6.1: The parameters for the best performing model for each class using KNN with LDA.

Class	k-neighbours	Distance Formula	Weights
Guitar Type	3	Manhattan	Uniform
Pickup	3	Manhattan	Uniform/Distance
Pickup Position	5	Manhattan	Uniform/Distance
Strumming	3	Manhattan	Uniform/Distance
Player	3	Manhattan	Distance

Table 6.2: The parameters for the best performing model for each class using KNN without LDA.

6.2.2 KNN Test Set 1

The precision, recall and F1-score for the best performing KNN with LDA and KNN without LDA models can be seen in Table 6.3 and Table 6.4 respectively.

KNN with LDA on Test Set 1

Guitar Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
LP	97.1	98.0	97.5
SG	97.4	96.2	96.8
SC	96.6	98.6	97.6
TC	94.9	88.1	91.4
Overall	**96.5**	**95.2**	**95.8**

Pickup Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Single Coil	97.6	98.2	97.9
Humbucker	98.5	98.0	98.2
Overall	**98.1**	**98.1**	**98.1**

Pickup Position			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Bridge	99.4	100.0	99.7
Middle	81.0	82.9	81.9
Neck	96.4	95.3	95.9
Overall	**92.3**	**92.7**	**92.5**

Strumming Method			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Open	99.5	99.5	99.5
A Major	99.3	99.3	99.3
Overall	**99.4**	**99.4**	**99.4**

Player			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
JM	85.0	83.7	84.3
VS	85.1	95.2	89.9
BH	35.0	34.1	34.6
JG	61.9	76.5	68.4
KB	81.3	61.9	70.3
AL	94.7	94.7	94.7
Overall	**73.8**	**74.4**	**73.7**

Table 6.3: The recall (Sens), precision (PPV) and F1-score metrics for every label in each class for the KNN with LDA model on test set 1.

KNN without LDA on Test Set 1

Guitar Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
LP	92.2	94.9	93.5
SG	93.6	91.3	92.4
SC	99.3	100.0	99.7
TC	100.0	95.1	97.5
Average	**96.3**	**95.3**	**95.8**

Pickup Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Single Coil	100.0	98.8	99.4
Humbucker	99.0	100	99.5
Average	**99.5**	**99.4**	**99.5**

Pickup Position			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Bridge	96.8	100.0	98.4
Middle	95.2	85.1	89.9
Neck	96.4	96.4	96.4
Average	**96.2**	**93.9**	**94.9**

Strumming Method			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Open	100.0	99.5	99.8
A Major	99.3	100.0	99.7
Average	**99.7**	**99.8**	**99.7**

Player			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
JM	97.6	94.4	95.9
VS	89.4	97.7	93.3
BH	87.5	92.1	89.7
JG	100.0	87.5	93.3
KB	81.3	100.0	89.7
AL	92.1	94.6	93.3
Average	**91.3**	**94.4**	**92.6**

Table 6.4: The recall (Sens), precision (PPV) and F1-score metrics for every label in each class for the KNN without LDA model on test set 1.

6.2.3 KNN Test Set 2

Likewise, for the second test set, the precision, recall and F1-score for the best performing KNN with LDA and KNN without LDA models can be seen in Table 6.5

and Table 6.6 respectively.

KNN with LDA Results on Test Set 2

Guitar Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
SC	100.0	83.8	91.2
Average	**100.0**	**83.8**	**91.2**

Pickup Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Single Coil	100	99.6	99.8
Average	**100**	**99.6**	**99.8**

Pickup Position			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Bridge	0.0	0.0	0.0
Middle	33.3	100.0	50.0
Neck	0.0	0.0	0.0
Average	**11.1**	**33.3**	**16.7**

Strumming Method			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Open	50.0	100.0	66.7
A Major	0.0	0.0	0.0
Average	**25.0**	**50.0**	**33.4**

Player			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
JM	100.0	30.4	46.6
Average	**100.0**	**30.4**	**46.6**

Table 6.5: The recall (Sens), precision (PPV) and F1-score metrics for every label in each class for the KNN with LDA on test set 2.

KNN without LDA Results on Test Set 2

Guitar Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
SC	100.0	85.0	92.0
Average	**100.0**	**85.0**	**92.0**

Pickup Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Single Coil	100	85.0	92.0
Average	**100**	**85.0**	**92.0**

Pickup Position			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Bridge	91.0	80.0	85.0
Middle	65.0	80.0	72.0
Neck	86.0	86.0	81.0
Average	**80.7**	**82.0**	**79.3**

Strumming Method			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Open	79.0	100.0	89.0
A Major	100.0	74.0	85.0
Average	**89.5**	**87.0**	**87.0**

Player			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
JM	100.0	91.0	95.0
Average	**100.0**	**91.0**	**95.0**

Table 6.6: The recall (Sens), precision (PPV) and F1-score metrics for every label in each class for the KNN without LDA on test set 2.

6.3 SVM Results

The results for the SVM with and without LDA will be presented in this section.

6.3.1 SVM Models Best Parameters

Similar to the KNN models, multiple SVM models were trained with various parameters, where the best performing SVM with LDA and without LDA have the parameters as seen in Table 6.7 and Table 6.8 respectively.

Class	**Regularization**	**Gamma**	**Kernel**
Guitar Type	$1/10/10^2/10^3$	$10^{-4}/10^{-3}/10^{-2}/10^{-1}/1/10/$N.A	RBF/Linear
Pickup	All	All	RBF/Linear
Pickup Position	$1/10/10^2/10^3$	$10^{-4}/10^{-3}/10^{-2}/10^{-1}/1/10/$N.A	RBF/Linear
Strumming	All	All	RBF/Linear
Player	$10^2/10^3$	$10^{-2}/10^{-3}$	RBF

Table 6.7: The parameters for the best performing model for each class using SVM with LDA.

Class	**Regularization**	**Gamma**	**Kernel**
Guitar Type	$10/10^3$	10^{-4}	RBF
Pickup	$10/10^2/10^3$	10^{-4}	RBF
Pickup Position	$10/10^2/10^3$	10^{-4}	RBF
Strumming	$1/10/10^2/10^3$	N.A	Linear
Player	$10/10^2/10^3$	10^{-4}	RBF

Table 6.8: The parameters for the best performing model for each class using SVM without LDA.

6.3.2 SVM Test Set 1

The precision, recall and F1-score for the best performing SVM with LDA and SVM without LDA models can be seen in Table 6.9 and Table 6.10 respectively.

SVM with LDA Results for Test Set 1

Guitar Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
LP	98.0	97.1	97.6
SG	97.4	96.2	96.8
SC	96.6	98.0	97.3
TC	89.7	89.7	89.7
Average	**95.5**	**95.2**	**95.4**

Pickup Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Single Coil	98.5	98.5	98.5
Humbucker	98.2	98.2	98.2
Average	**98.4**	**98.4**	**98.4**

Pickup Position			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Bridge	99.4	99.4	99.4
Middle	76.2	82.1	79.0
Neck	96.4	94.8	95.6
Average	**90.7**	**92.1**	**91.3**

Strumming Method			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Open	100.0	100.0	100.0
A Major	100.0	100.0	100.0
Average	**100.0**	**100.0**	**100.0**

Player			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
JM	83.0	84.2	83.6
VS	85.1	97.6	90.9
BH	37.5	31.3	34.1
JG	61.9	72.2	66.7
KB	75.0	60.0	66.7
AL	94.7	94.7	94.7
Average	**72.9**	**73.3**	**72.8**

Table 6.9: The recall (Sens), precision (PPV) and F1-score metrics for every label in each class for the SVM LDA on test set 1.

SVM without LDA Results for Test Set 1

Guitar Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
LP	100.0	98.1	99.0
SG	98.7	100.0	99.4
SC	99.3	99.3	99.3
TC	97.4	100.0	98.7
Average	**98.8**	**99.4**	**99.1**

Pickup Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Single Coil	99.5	99.5	99.5
Humbucker	99.5	99.5	99.5
Average	**99.5**	**99.5**	**99.5**

Pickup Position			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Bridge	100.0	100.0	100.0
Middle	90.5	90.5	90.5
Neck	97.8	97.8	97.8
Average	**96.1**	**96.1**	**96.1**

Strumming Method			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Open	100.0	100.0	100.0
A Major	100.0	100.0	100.0
Average	**100.0**	**100.0**	**100.0**

Player			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
JM	99.5	94.0	96.7
VS	91.5	97.7	94.5
BH	82.5	100.0	90.4
JG	100.0	95.5	97.7
KB	87.5	100.0	98.7
AL	97.4	100.0	98.7
Average	**93.1**	**97.9**	**96.1**

Table 6.10: The recall (Sens), precision (PPV) and F1-score metrics for every label in each class for the SVM without LDA on test set 1.

6.3.3 SVM Test Set 2

Likewise, for the second test set, the precision, recall and F1-score for the best performing SVM with LDA and SVM without LDA models can be seen in Table 6.11 and Table 6.12 respectively.

SVM with LDA Results on Test Set 2

Guitar Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
SC	100.0	82.9	90.7
Average	**100.0**	**82.9**	**90.7**

Pickup Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Single Coil	100	99.6	99.8
Average	**100**	**99.6**	**99.8**

Pickup Position			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Bridge	0.0	0.0	0.0
Middle	33.3	100.0	50.0
Neck	0.0	0.0	0.0
Average	**11.1**	**33.3**	**16.7**

Strumming Method			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Open	50.0	100.0	66.7
A Major	0.0	0.0	0.0
Average	**25.0**	**50.0**	**33.4**

Player			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
JM	0.0	0.0	0.0
Average	**0.0**	**0.0**	**0.0**

Table 6.11: The recall (Sens), precision (PPV) and F1-score metrics for every label in each class for the SVM with LDA on test set 2.

SVM without LDA results on Test Set 2

Guitar Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
SC	100.0	100.0	100.0
Average	**100.0**	**100.0**	**100.0**

Pickup Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Single Coil	100.0	100.0	100.0
Average	**100.0**	**100.0**	**100.0**

Pickup Position			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Bridge	100.0	100.0	100.0
Middle	86.0	96.0	91.0
Neck	96.0	84.0	89.0
Average	**94.0**	**93.3**	**93.3**

Strumming Method			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Open	100.0	100.0	100.0
A Major	100.0	100.0	100.0
Average	**100.0**	**100.0**	**100.0**

Player			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
JM	100.0	99.0	100.0
Average	**100.0**	**99.0**	**100.0**

Table 6.12: The recall (Sens), precision (PPV) and F1-score metrics for every label in each class for the SVM without LDA on test set 2.

6.4 LDA

The results for the LDA implementation consist of both the dimensionality reduction on each class, as well as the accuracy score on the test set.

6.4.1 LDA Test Set 1

The precision, recall and F1-score for the best performing LDA model can be seen in Table 6.13.

Results for Test Set 1

Guitar Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
LP	97.1	98.0	97.5
SG	97.4	96.2	96.8
SC	98.0	100.0	99.0
TC	100.0	92.9	96.3
Average	**98.1**	**96.8**	**97.4**

Pickup Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Single Coil	98.5	98.5	98.5
Humbucker	98.2	98.2	98.2
Average	**98.4**	**98.4**	**98.4**

Pickup Position			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Bridge	100.0	100.0	100.0
Middle	85.7	85.7	85.7
Neck	96.4	96.4	96.4
Average	**94.1**	**94.1**	**94.1**

Strumming Method			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Open	99.5	99.5	99.5
A Major	99.3	99.3	99.3
Average	**99.4**	**99.4**	**99.4**

Player			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
JM	94.7	92.9	93.8
VS	87.2	97.6	92.1
BH	80.0	84.2	82.1
JG	95.2	80.0	87.0
KB	87.5	93.3	90.3
AL	94.7	94.7	94.7
Average	**89.9**	**90.5**	**90.0**

Table 6.13: The recall (Sens), precision (PPV) and F1-score metrics for every label in each class for the LDA on test set 1.

6.4.2 LDA Test Set 2

Likewise, for the second test set, the precision, recall and F1-score for the best performing LDA model can be seen in Table 6.14.

LDA Results on Test Set 2

Guitar Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
SC	100.0	98.0	99.0
Average	**100**	**98.0**	**99.0**

Pickup Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Single Coil	100.0	100.0	100.0
Average	**100.0**	**100.0**	**100.0**

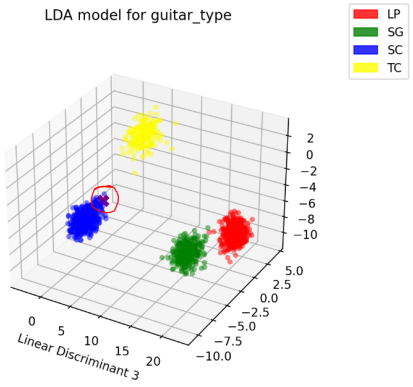
Pickup Position			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Bridge	100.0	100.0	100.0
Middle	100.0	90.0	95.0
Neck	91.0	100.0	95.0
Average	**97.0**	**96.7**	**96.7**

Strumming Method			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Open	98.0	99.0	98.5
A Major	99.0	97.0	98.0
Average	**98.5**	**98.0**	**98.3**

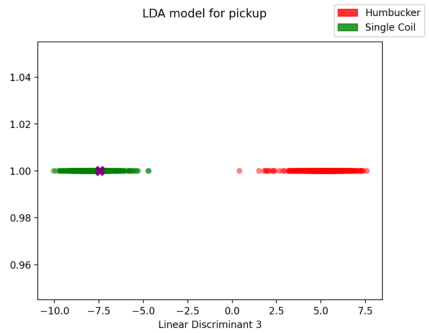
Player			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
JM	100.0	90.0	95.0
Average	**100.0**	**90.0**	**95.0**

Table 6.14: The recall (Sens), precision (PPV) and F1-score metrics for every label in each class for the LDA on test set 2.

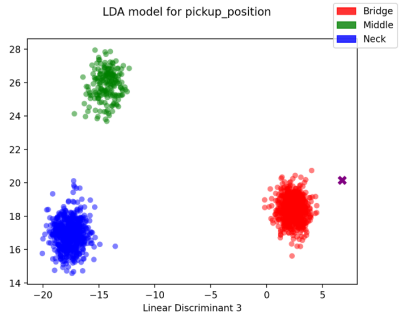
In Fig. 6.2 it is possible to see how the applied LDA dimensionality reduction has split up the labels within every class.



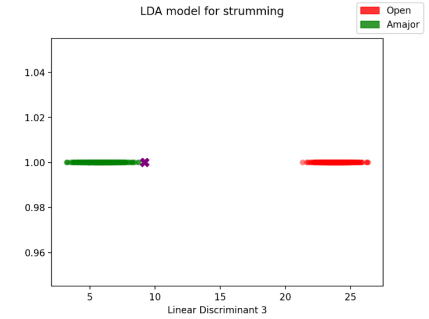
(a) LDA dimensionality reduction applied to the guitar type class.



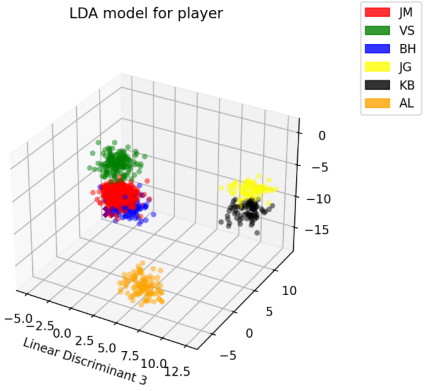
(b) LDA dimensionality reduction applied to the pickup class.



(c) LDA dimensionality reduction applied to the pickup position class.



(d) LDA dimensionality reduction applied to the strumming class.



(e) LDA dimensionality reduction applied to the player class.

Figure 6.2: The LDA dimensionality reduction applied to the classes. The purple X indicates a new test sample.

6.5 CNN Results

The results for the CNN have been obtained using the best performing model with the highest overall metrics. The best performing CNN model has the training parameters shown in Table 6.15, where the training and validation loss per epoch can be seen in Fig. 6.3.

Learning Rate	Batch Size	Optimizer
10^{-3}	64	RAdam

Table 6.15: The parameters for the best performing CNN model.

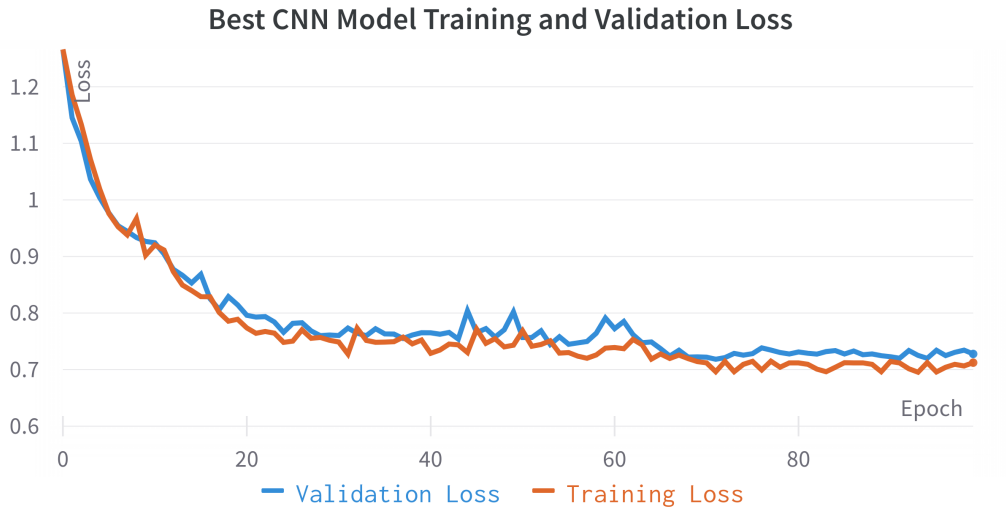


Figure 6.3: The training and validation loss per epoch for the best performing CNN model.

6.5.1 CNN Test set 1

The precision, recall and F1-score for all of the classes on the 10% testing split set can be seen in Table 6.16.

CNN Results on Test Set 1

Guitar Type			
Label	**Sens(%)**	**PPV(%)**	**F1 (%)**
LP	100.0	98.2	99.1
SG	96.9	100.0	98.4
SC	98.7	100.0	99.4
TC	100.0	95.5	97.7
Average	**98.9**	**98.4**	**98.7**

Pickup Type			
Label	**Sens(%)**	**PPV(%)**	**F1 (%)**
Single Coil	100.0	97.8	99.0
Humbucker	97.9	100.0	98.9
Average	**99.0**	**98.9**	**98.9**

Pickup Position			
Label	**Sens(%)**	**PPV(%)**	**F1 (%)**
Bridge	100.0	100.0	100.0
Middle	81.3	100	90.0
Neck	100	92.6	96.2
Average	**93.8**	**97.5**	**95.4**

Strumming Method			
Label	**Sens(%)**	**PPV(%)**	**F1 (%)**
Open	100.0	100.0	100.0
A Major	100.0	100.0	100.0
Average	**100.0**	**100.0**	**100.0**

Player			
Label	**Sens(%)**	**PPV(%)**	**F1 (%)**
JM	100.0	84.1	91.4
VS	0.0	0.0	0.0
BH	0.0	0.0	0.0
JG	100.0	50	66.7
KB	100.0	85.7	92.3
AL	100.0	90.5	95.0
Average	**66.7**	**51.7**	**57.6**

Table 6.16: The recall (Sens), precision (PPV) and F1-score metrics for every label in each class for the CNN on test set 1

6.5.2 CNN Test set 2

The precision, recall and F1-score for guitar 11 can be seen in Table 6.17. This test set is formed only by data retrieved from guitar 11, as was presented at the beginning of the chapter.

CNN Results on Test Set 2

Guitar Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
SC	100	87.5	99.3
Average	**100**	**87.5**	**99.3**

Pickup Type			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Single Coil	100	79.2	88.4
Average	**100**	**79.2**	**88.4**

Pickup Position			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Bridge	100.0	100.0	100.0
Middle	100.0	100.0	100.0
Neck	100.0	100.0	100.0
Average	**100.0**	**100.0**	**100.0**

Strumming Method			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
Open	100.0	100.0	100.0
A Major	100.0	100.0	100.0
Average	**100.0**	**100.0**	**100.0**

Player			
Label	**Sens(%)**	**PPV(%)**	**F1(%)**
JM	100.0	100.0	100.0
Average	**100.0**	**100.0**	**100.0**

Table 6.17: The recall (Sens), precision (PPV) and F1-score metrics for every label in each class for the CNN on the test set 2.

6.6 Model Results

This section will summarize this chapter by comparing the results from all the four different models. Furthermore, the results for both the KNN and SVM include models which are trained on the LDA-dimensionality splits. Table 6.18 and Table 6.19 shows the F1-score for all of the classes in each model for test set 1 and 2 respectively.

Test Set 1

Model	Guitar Type	Pickup Type	Pickup Position	Strumming	Player
KNN (LDA)	95.8	98.1	92.5	99.4	73.7
KNN	95.8	**99.5**	94.9	99.7	92.6
SVM (LDA)	95.4	98.4	91.3	**100.0**	72.8
SVM	**99.1**	**99.5**	**96.1**	**100.0**	**96.1**
LDA	97.4	98.4	94.1	99.4	90.0
CNN	98.7	98.9	95.4	**100.0**	57.6

Table 6.18: Overview of the F1-Score(%) for all the models on test set 1. The best model for each class is highlighted.

Test Set 2

Model	Guitar Type	Pickup Type	Pickup Position	Strumming	Player
KNN (LDA)	91.2	99.8	16.7	33.4	46.6
KNN	92.0	92.0	79.3	87.0	95.0
SVM (LDA)	90.7	99.8	16.7	33.4	0.0
SVM	**100.0**	**100.0**	93.3	**100.0**	**100.0**
LDA	99.0	**100.0**	96.7	98.3	95.0
CNN	99.3	88.4	**100.0**	**100.0**	**100.0**

Table 6.19: Overview of the F1-Score(%) for all the models on test set 2. The best model for each class is highlighted.

6.7 Real-time System Execution Time Test

This section will evaluate the execution time for each of the models and extraction of features, when running the real-time system. The execution time of all models will be tested on each class: Guitar type, Pickup type, Pickup Position, Strumming, and Player, with the exception of the CNN, as the model is able to predict multiple labels at once. An arbitrary guitar from Table 3.1 is strummed 20 times for each class. The mean execution time in milliseconds of each individual class for each model is shown in Table 6.20 and the mean execution time in milliseconds for

extracting the MFCCs and Mel spectrograms is shown in Table 6.21. These results are very hardware dependent and was tested on a single laptop having a AMD Ryzen 7 4700 CPU with 8 cores running windows 10 with 16 gigabytes of RAM.

Execution Time Test (ms)

Model	Guitar Type	Pickup Type	Pickup Position	Strumming	Player	Average
KNN (LDA)	1.40	1.30	1.00	1.30	1.60	1.32
KNN	57.95	54.39	55.61	53.63	53.60	55.04
SVM (LDA)	1.43	0.99	1.00	0.99	0.99	1.08
SVM	7.40	4.70	6.71	1.31	9.20	5.86
LDA	0.99	1.00	0.99	1.00	0.99	0.99
CNN	N.A	N.A	N.A	N.A	N.A	87.07

Table 6.20: The execution time for every model on each class using the real-time application.

Feature Extraction Time (ms)	Average
Mel Spectrograms	54.17
MFCCs	16.60

Table 6.21: The execution time in milliseconds for extracting Mel spectrograms and MFCCs.

Chapter 7

Discussion

This chapter will discuss the results from the four models implemented on the guitar dataset and at the end of it it will be stated which one was the most accurate.

7.1 KNN

Looking at the parameters for the best models found, it is interesting to note how when using LDA with KNN, the different combinations do not matter for four out of the five classes. This could be because of how the data is clustered after LDA is applied to the dataset; it does not matter if the k-neighbours is 3 or 15 since the between-class scatter is high, with a low within-class scatter. As well, even with the different distance formulas and weights, the model scores would be the same. Additionally, when looking at the parameters for KNN without LDA applied, it goes from 1-3 dimensions to 6747 dimensions. This means that there inherently could be a lot more overlap between the labels within a class, making the specific parameters for each of the models that were found the best suited ones.

The KNN with and without LDA results on test set 1 can be seen in Table 6.4 and Table 6.3 where the average F1-scores for all classes are relatively high where the average F1-scores are above 90%, with the only exception being the player class with an average F1-score of 73.7%. As shown in Table 6.3, the KNN model with LDA only managed to achieve sensitivity, precision, and F1-score of 35.0%, 34.1%, 43.6% respectively for the BH label in the player class, which is 52.5%, 58.0%, and 55.1% lower respectively compared to the KNN model without LDA. This deviation is caused by how the different classes are scattered after applying LDA to the data, as shown in Fig. 6.2e, where multiple labels overlap. The mean F1-score difference between all the classes with and without LDA applied is 4.6%, whereas KNN without LDA has the highest overall accuracy.

When looking at the KNN model with and without LDA, the models achieved F1-scores of 91.2% and 92.0% for the pickup type and 99.8% and 92% for the gui-

tar type. Furthermore, as seen in Table 6.5, the KNN model with LDA struggled with classifying the pickup position, strumming method and player classes when tested on test set 2 compared to the KNN model without LDA presented in Table 6.6. Looking at the pickup position and strumming method classes, certain labels within these classes had a sensitivity, precision, and F1-score of 0%, meaning the model failed to correctly predict any of the labels. As the test set 2 dataset only consists of 240 samples, the KNN models become more susceptible to outliers, which might have impacted the overall results for certain classes.

When looking at the sizes of the KNN models with LDA and without LDA, a big difference can be seen. By looking at the average model size with applied LDA as an example, it is 73 KB compared to the same model without applied LDA being 38,754 KB. This means that the model with applied LDA is about 500 times smaller than the model without applied LDA. As seen in Table 6.20, the KNN with LDA average execution time is 1.32 ms, whereas KNN without LDA is much slower, with an average execution time of 55.04 ms.

This could be beneficial in situations where faster computations are needed on a system with low RAM, but as seen in the results, it comes with a cost to the F1-score.

7.2 SVM

For the chosen parameters for the SVM - like the chosen parameters for the KNN- it can be seen that when LDA is applied to the dataset a more diverse set of parameters can be used which yields the same model scores. As well, when the dataset does not have LDA applied more specific parameters are used, presumably because of the higher dimensionality with labels overlapping in the feature space.

Much of the same results can be seen when looking at the results of SVM compared to the KNN, with the biggest difference being the higher overall F1-score between the applied LDA and no LDA. SVM also performed the best on unseen data from test set 2, with four out of five classes having 100% in the F1-score. Though, as well as the KNN, the F1-score falls behind when LDA is applied.

Similar to the KNN models, the sizes of the SVM models are considerably bigger without LDA compared to those with LDA: 36,887 kB and 40.6 kB on average respectively. This also has an impact on the average execution time of 5.86, and 1.08 ms respectively as seen in Table 6.20.

7.3 LDA

Similar to the above mentioned machine learning models, the LDA model performs well with no average F1-score under 90% for any class. However, as seen in the overall model results for test set 1 in Table 6.18, the LDA does not perform as the best model for any of the classes, though it shares the best F1-score with the SVM model on the pickup type for test set 2 with 100%. The LDA model has a high average F1-score for the guitar type, pickup type and strumming method. The LDA plots for the pickup type and the strumming method in Fig. 6.2b and Fig. 6.2d respectively show a clear split between the data points in each class. The results from these two classes also have an average F1-score of 98.4% and 99.4% indicating that the LDA performs really well.
The bridge pickup position has a perfect F1-score of 100% for test 1 and test 2. This can be reflected in the fact that the LDA splits the pickup position into its own corner in the sample space as seen in Fig. 6.2c. The middle and neck positions have F1-scores of 85.7% and 96.4% respectively which can be deduced that these data points have been positioned closer together, thus test samples can be placed in-between and be misclassified. The results for the player class have an average F1-score of 90%. The data points seen in Fig. 6.2e show the spread of the player labels, where some players such as VM and AL, have a higher PPV than others (97.6% and 94.7% respectively), meaning that samples from these labels are not misclassified as other labels, as these players are also spread further out compared to the remaining players who are clustered together. JM and BH are clustered together, and KB and JG are clustered together. However, only one of each player from the clusters has worse results. BH and JG have 84.2% and 80.0% PPV respectively compared to JM and KB's PPV of 92.9% and 93.3%, indicating that they were misclassified as the other person in the cluster.

Overall, for test set 2, the LDA performs very nicely with the lowest average F1-score being the player class with 95% as some JM samples have been misclassified as other labels.

The LDA model, however, has the biggest model size compared to the rest with an average size of 711,620 kB. In spite of the big model size, the execution time is the fastest with an average of 0.99 ms.

Lastly, unlike the other models, the LDA was not trained with various parameters. Thus, better results could potentially be achieved by using other parameters for training.

7.4 CNN

For the behavior of the training and validation loss per epoch visualized in Fig. 6.3, it is noticeable that the model converges around epoch 80 where the validation loss

does not decrease further and is slightly higher than the training. The validation loss also does not increase after converging, meaning that overfitting does not occur. Outside of the training and validation data, there are some solid results in the two test sets. As shown in Table 6.16, most of the class metrics are above 95 % for test set 1 with the exception of labels such as the middle pickup position (which still achieved a very good F1-score of 90 %) and player. The CNN model has difficulty classifying the different players, which could be due to the imbalance in the dataset, as most data has been collected by one individual in the group (JM). It can be seen that the player JM has a recall (Sens) of 100%, which means that the model did not misclassify JM samples as other labels. However, the 84.1% precision (PPV) indicates that other labels were misclassified as JM. The player JG had a similar issue with a 100% recall but 50% precision, though JG did not record as many samples as JM. VS and BH had 0% across all three metrics, as their samples were misclassified as other players, and none of their samples were predicted correctly. Another good reason could be that it is simply difficult for the model to differentiate between players when they are only playing two types of strumming. On the other hand, the rest of the classes shows very good results where the strumming method was the most dominant. The CNN model can predict with high accuracy the difference between an open and A Major strum, which should be expected as the sound produced is completely different on a timbre level.

For the second test set, as it was mentioned in the introduction of Section 6.5, a guitar outside the training/validation/testing split was used to test the model's performance on unseen data. The CNN achieved good results for the Pickup Position, Strumming Method and Player classes, which were predicted with an F1-score of 100%, slightly better than the testing results of test set 1. For the remaining classes, the CNN was also highly accurate, where the Guitar Type and Pickup Type classes achieved an F1-score of 99.3% and 88.4% respectively. These last two classes' metrics decreased a bit compared with test set 1.

Lastly, the model size of the CNN is 533,804 kB which is the second biggest model behind LDA. Unlike the previous models, the CNN model only consists of one model that predicts all of the classes, whereas the rest have one model for each. The CNN has an execution time of 87.07 ms, which is the slowest execution time out of all the models.

7.5 Overall

The best overall model used for classifying the classes is the SVM without LDA model, as this model performs the best across test set 1, and also achieves four out of five best F1-scores for test set 2. In Table 6.21 the time for extracting Mel spectrograms and MFCCs can be seen, where extracting Mel spectrograms take 5 times longer than MFCCs. As specified in Section 5.3, the duration for recording samples

in the real-time system is 5 seconds before the sample is cropped to two seconds, which compared to the execution time of the models presented in Table 6.20 and for feature extraction presented in Table 6.21, is vastly greater. Hence, to optimize the execution time of the real-time system from a software perspective, the recording duration and sample duration could be reduced as long as the accuracy is not compromised.

As it was shown before, two test sets were created to check how well the models generalized. However, since the second test set used the model with the most dominant guitar type and player used during training, the robustness of this test could have been improved by testing across all the types of guitars and not just the SC. Unfortunately, due to the lack of equipment, testing on all guitar types with unseen data was unachievable.

Chapter 8

Conclusion

This chapter will conclude the project. Guitarists who want to switch electric guitars in-between songs have to manually re-configure their signal processing chain based on the type of guitar. The purpose of this project was to assist guitarists in this process by implementing a system to classify parameters in an electric guitar as seen in the problem formulation:

How can different parameters of guitars be classified based on the sound produced using relevant machine/deep learning methods?

- The type of guitar(SC, TC, LP, or SG).
- The pickup used (single coil or humbucker).
- The pickup position (neck, middle, or bridge).
- The person playing the guitar.
- The strumming method (open strings or chords).

To solve this, three machine learning methods and one deep learning method were implemented: KNN, SVM, LDA and CNN, along with a combination of the machine learning methods: KNN with LDA and SVM with LDA. These models were trained on a custom dataset containing 2078 manually gathered sound samples, where MFCCs and Mel spectrograms were extracted to be used as features. Furthermore, a real-time application has been implemented to classify sound samples played by a guitarist in real time.

It can be concluded from the results that the SVM model without LDA is the best suited model for solving the problem, as it has the highest overall F1-score across all classes for test set 1, and four out of five classes in test set 2. These results show that it is possible to differentiate between 4 guitar types with an F1-score of 99.1%, the pickup type with an F1-score of 99.5%, the pickup position with

an F1-score of 96.1%, the strumming method with an F1-score of 100%, and finally the player with an F1-score of 96.1%.

The implemented solution suggests that it can assist guitarists who want to adjust their signal-processing chain by strumming the guitar and having it identify the guitar type. As well, the models also give an insight into how effective MFCCs are at defining timbre, which can be used to define different parameters of a guitar. The model scores show that it is possible to derive and detect the guitar type, pickup type, pickup position, strum, and the person playing the guitar by looking at the MFCCs, delta MFCCs, delta-delta MFCCs, and Mel spectrograms. This also shows how the different builds of the guitars with different setups, such as neck-type, body type, pickup type, string material etc. contribute to creating different tonal characteristics and timbre in the final signal which is analyzed by the developed system.

Chapter 9

Future Work

This chapter will discuss improvements that can be made for future work.

The most important improvement for this project is focused on the data distribution. The lack of guitars resulted in most of the data being collected from the SC guitar type. Hence, an option to solve this problem is by increasing the number of samples collected from other guitar types to achieve a more equal distribution or to either remove or add weights to samples from the dominant labels. Removing samples will drastically reduce the amount of samples for training, but may achieve better results without an over-representation of certain labels.
In addition, the strumming was also limited by the players who collected the data, where A major chords were only performed by experienced players. Using more experienced players in the data collection process could be a big improvement in order to equalize the distribution of A major and Open strumming signals. Additionally, more strumming types could be recorded to ensure that the model is more robust.

Furthermore, the duration of all the samples has been shortened to two seconds to maintain the same dimensions for the Mel spectrogram and the MFCCs when inputting them into the models. However, different signal lengths have not been tested, and thus it can not be concluded if the duration of two seconds is the most optimal length. Therefore, different signal lengths can be tested, where shorter and longer lengths are tested. Though, as seen in Fig. 4.9 that visualizes the MFCCs, delta coefficients and delta-delta coefficients, the delta and delta-deltas only have an impact on the first 0.5 seconds of the signal, which could indicate that longer signals are not necessary. Additionally, using shorter sound signals will have an impact on the execution time of the real-time system, as less information has to be recorded and processed.
Apart from these changes in the data distribution, an improvement in the quality of the testing could be also very helpful. As seen in Section 6.5.2, the second test set is formed by only one unseen guitar. As it was previously mentioned, with a

bigger amount of guitars this test could be extended and have more convincing conclusions about the models' classifying abilities.

In terms of data collection, there is the possibility of increasing the sampling rate in Audacity at the time of collection. This could result in having more information per sample and increasing the model's classification performance. This could also lead to a decrease in the training speed, so it would be important to find a trade-off to get the best outcome possible. Another good option would be testing different window sizes as they also affect the Mel spectrograms and the MFCCs.

Other CNN architectures and available outside the TabCNN. The TabCNN was originally built for a different purpose but was chosen as it had good results in determining the string tabs. Other CNN architectures could have been used and compared to the TaBCNN, similar to comparing the different machine learning methods: KNN, SVM and LDA. Such architectures could have been for example the VGGNet or AlexNet that the TabCNN was based upon. These architectures are very similar to TaBCNN, where VGGNet has four convolutional layers instead of three, and uses max-pooling layers after each convolution and AlexNet has five convolutional layers followed by max pooling and three fully connected layers at the end.

Chapter 10

Appendix

10.1 Cutting the dataset using MFCC

This section in the Appendix presents an alternative method to split the data samples which was unused. Using the MFCCs for cutting the audio files into strum chunks, was done by looking at the delta of the first coefficient. As can be seen in Fig. 10.2, the empty spaces between each strum chunk can be represented using the first and/or second coefficient. Looking at the first coefficient, when there is empty space the coefficient represents that as a dark blue with a value below -700, and when the guitar is strummed, it is represented as white with a value around -300. Knowing this, it is possible to separate the strums into equal sizes by having a certain sampling size for each strum. When the software finds a new strum, a certain amount of samples would be taken, saved, and then labeled. The number of samples can be adjusted using a variable called 'res'. The next strum is then found by looking at the difference between the prior and new chunks. This process can be seen in Fig. 10.1.

While this worked successfully at separating the strums and removing the silence, as can be seen in Fig. 10.3. The variable 'max_difference' would not always be a consistent value, as for each new audio file it would have to be iteratively adjusted until the correct amount of strums were isolated. As well, only the MFCCs were cut but not the audio file itself. This means that saving the individual audio strums as .WAV files was not possible, and the strums would have to be isolated every time the main executable was run. Therefore, a new solution was developed, which took into account the size of the dataset, as well as the time spent having to iteratively processes the data. it also made it possible to save each strum with the correct filename in the folder structure explained in Chapter 3.

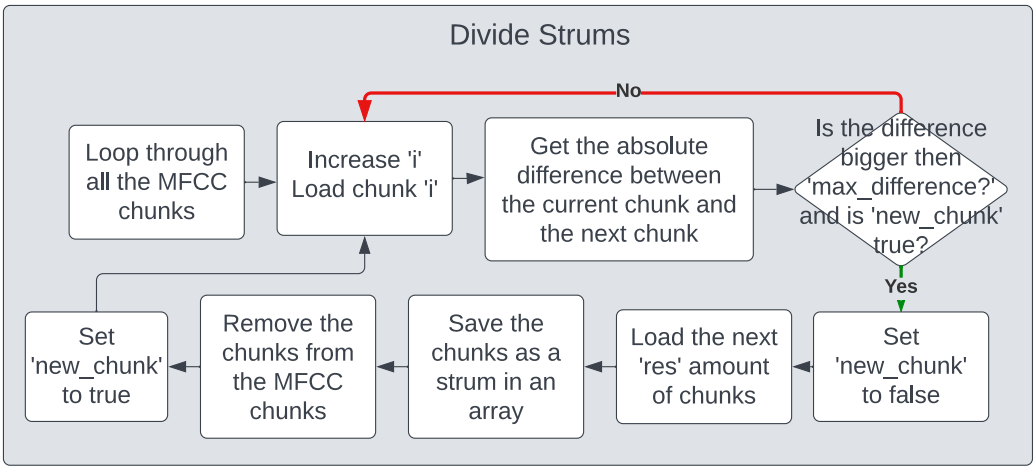


Figure 10.1: MFCC strum chunk separator function using MFCCs.

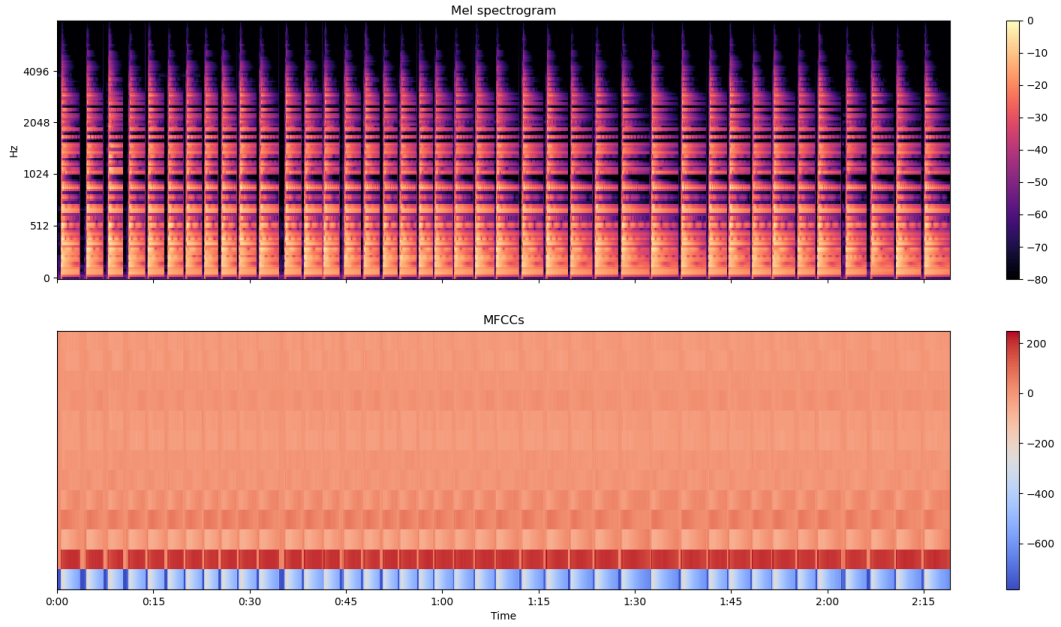


Figure 10.2: A single audio file of the neck pickup for a Stratocaster with the Mel spectrogram and MFFC coefficients displayed. This is with silence between each strum. The strums are also different sizes.

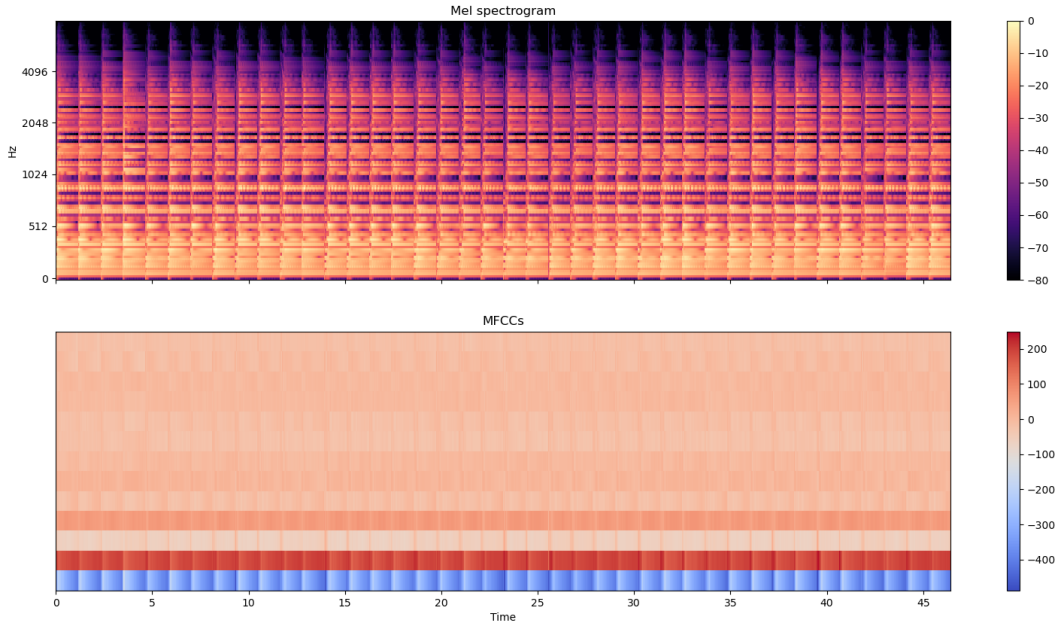
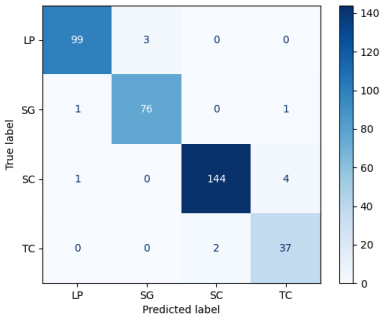
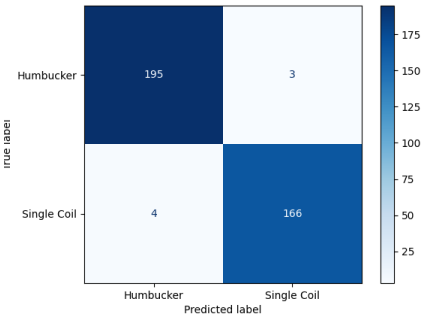


Figure 10.3: A single audio file of the neck pickup for a Stratocaster with the Mel spectrogram and MFFC coefficients displayed. This is with the silence removed and strums made the same size.

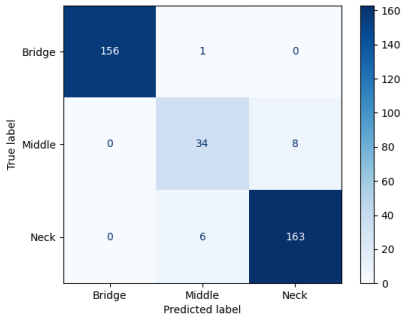
10.2 KNN with LDA Test Set 1



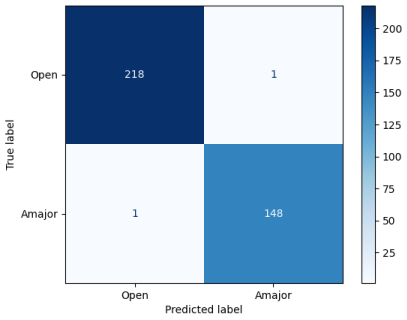
(a) KNN with LDA. Confusion matrix for the guitar type class.



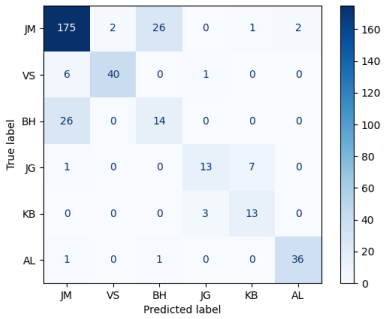
(b) KNN with LDA. Confusion matrix for the pickup class.



(c) KNN with LDA. Confusion matrix for the pickup position class.



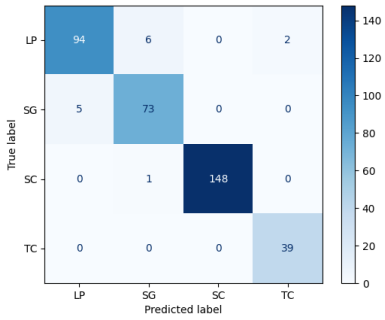
(d) KNN with LDA. Confusion matrix for the strumming method class.



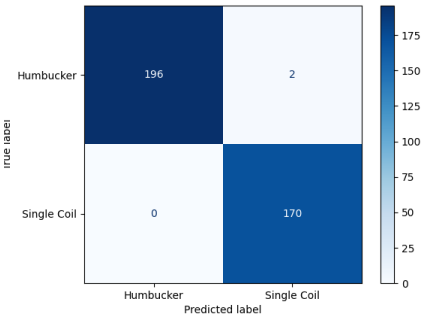
(e) KNN with LDA. Confusion matrix for the player class.

Figure 10.4: KNN with LDA Test Set 1. Confusion matrices for every class.

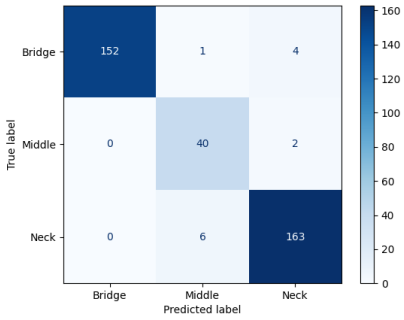
10.3 KNN without LDA Test Set 1



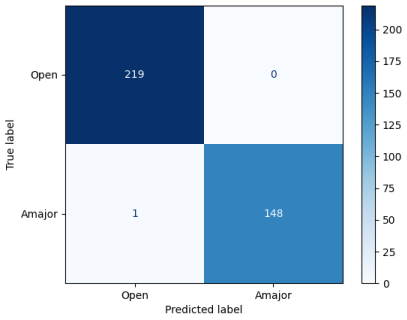
(a) KNN without LDA. Confusion matrix for the guitar type class.



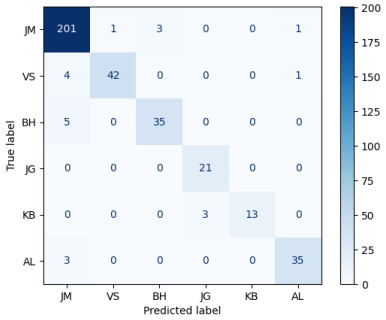
(b) KNN without LDA. Confusion matrix for the pickup class.



(c) KNN without LDA. Confusion matrix for the pickup position class.



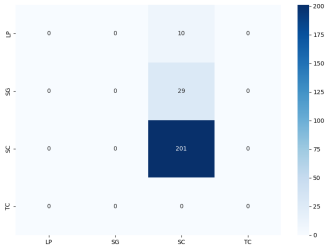
(d) KNN without LDA. Confusion matrix for the strumming method class.



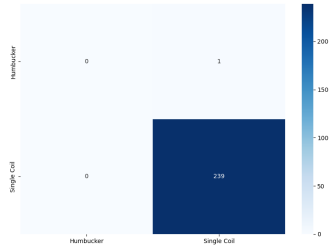
(e) KNN without LDA. Confusion matrix for the player class.

Figure 10.5: KNN without LDA Test Set 1. Confusion matrices for every class.

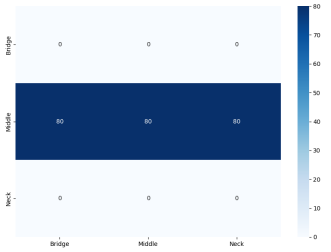
10.4 KNN with LDA Test Set 2



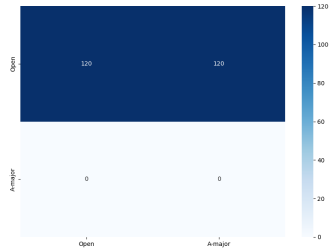
(a) KNN with LDA. Confusion matrix for the guitar type class.



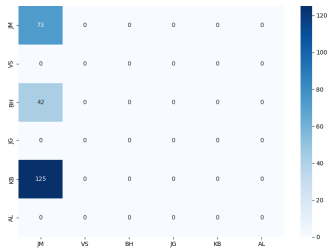
(b) KNN with LDA. Confusion matrix for the pickup class.



(c) KNN with LDA. Confusion matrix for the pickup position class.



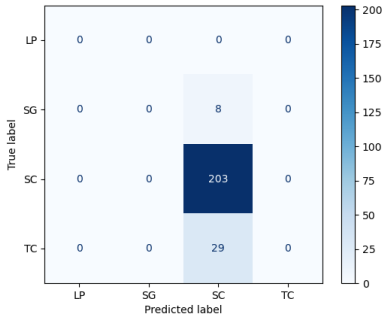
(d) KNN with LDA. Confusion matrix for the strumming method class.



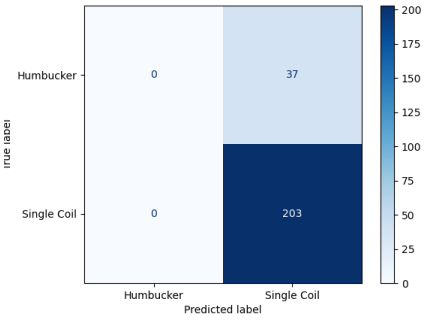
(e) KNN with LDA. Confusion matrix for the player class.

Figure 10.6: KNN Test Set 2. Confusion matrices for every class.

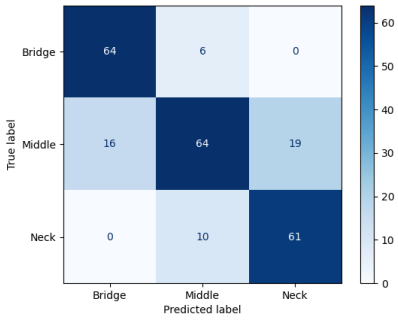
10.5 KNN without LDA Test Set 2



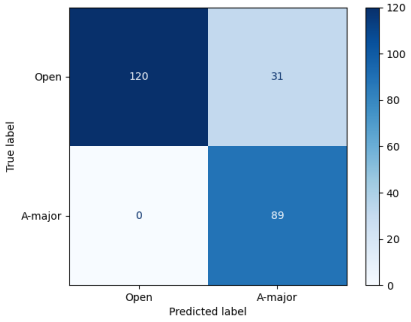
(a) KNN without LDA. Confusion matrix for the guitar type class.(2)



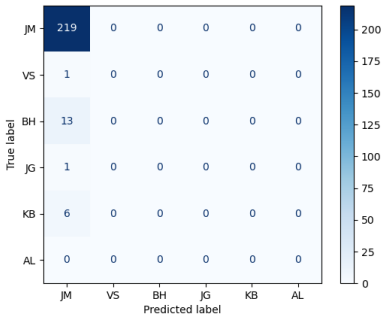
(b) KNN without LDA. Confusion matrix for the pickup class.(test2)



(c) KNN without LDA. Confusion matrix for the pickup position class.



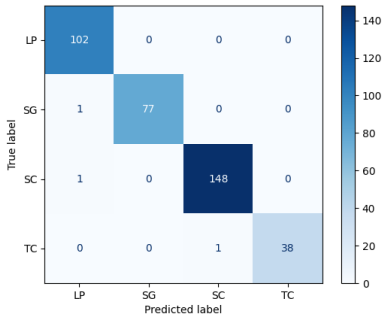
(d) KNN without LDA. Confusion matrix for the strumming method class.



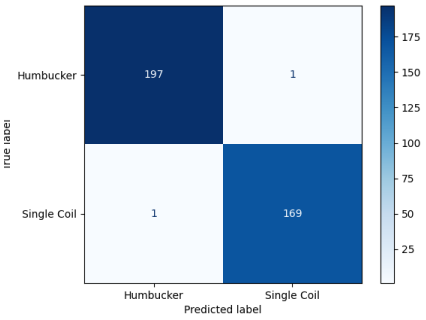
(e) KNN without LDA. Confusion matrix for the player class.

Figure 10.7: KNN without LDA Test Set 2. Confusion matrices for every class.

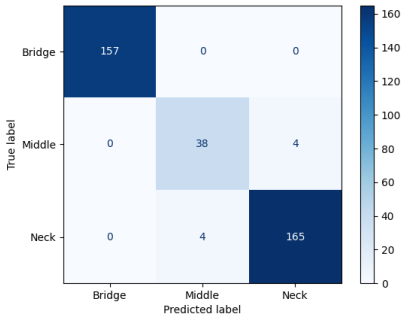
10.6 SVM with LDA Test Set 1



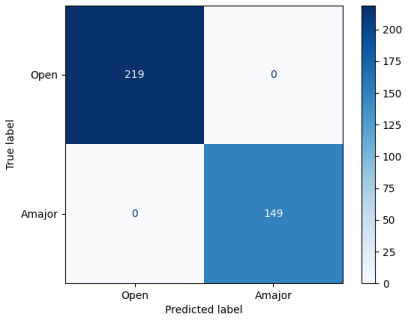
(a) SVM with LDA. Confusion matrix for the guitar type class.



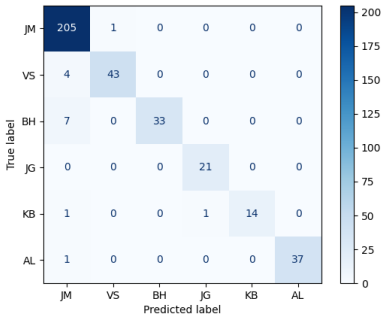
(b) SVM with LDA. Confusion matrix for the pickup class.



(c) SVM with LDA. Confusion matrix for the pickup position class.



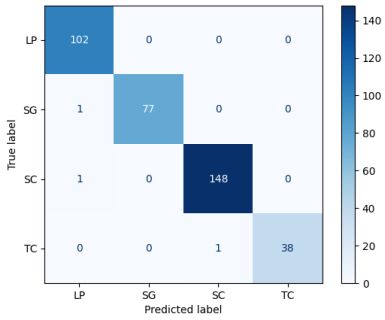
(d) SVM with LDA. Confusion matrix for the strumming method class.



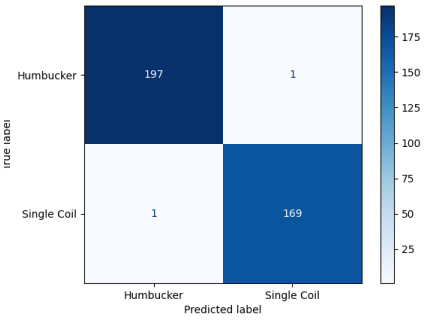
(e) SVM with LDA. Confusion matrix for the player class.

Figure 10.8: SVM with LDA Test Set 1. Confusion matrices for every class.

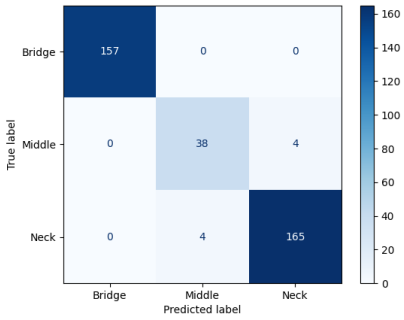
10.7 SVM without LDA Test Set 1



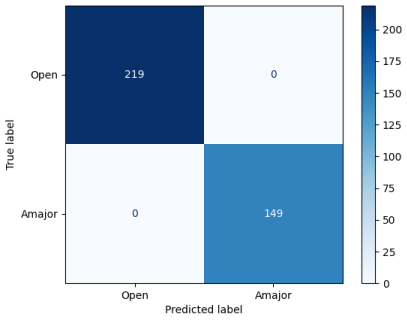
(a) SVM without LDA. Confusion matrix for the guitar type class.



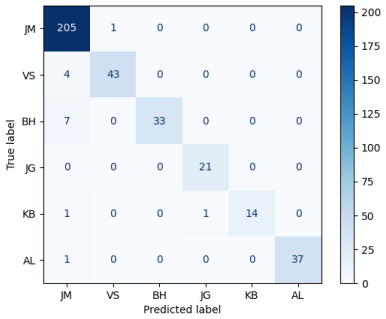
(b) SVM without LDA. Confusion matrix for the pickup class.



(c) SVM without LDA. Confusion matrix for the pickup position class.



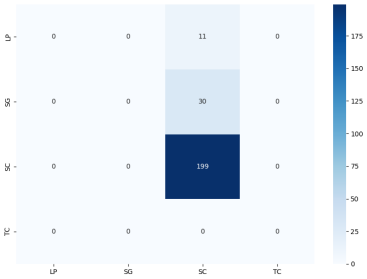
(d) SVM without LDA. Confusion matrix for the strumming method class.



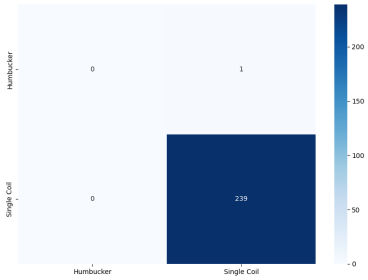
(e) SVM without LDA. Confusion matrix for the player class.

Figure 10.9: SVM without LDA Test Set 1. Confusion matrices for every class.

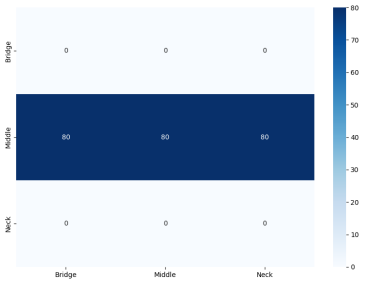
10.8 SVM with LDA Test Set 2



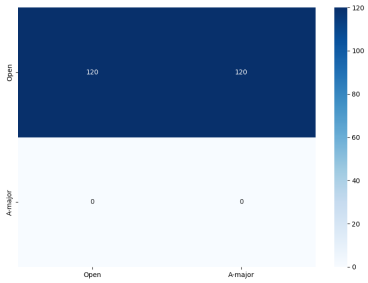
(a) SVM with LDA. Confusion matrix for the guitar type class.



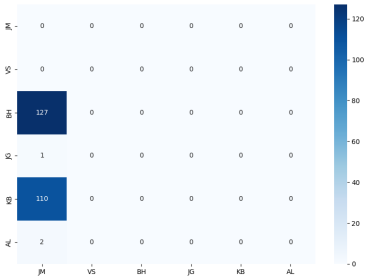
(b) SVM with LDA. Confusion matrix for the pickup class.



(c) SVM with LDA. Confusion matrix for the pickup position class.



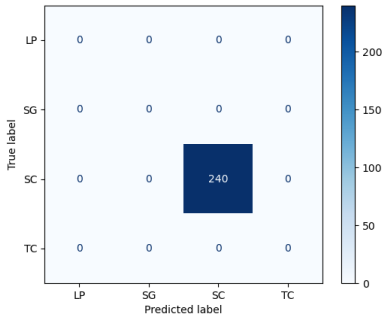
(d) SVM with LDA. Confusion matrix for the strumming method class.



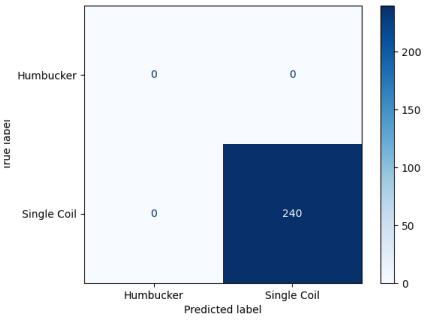
(e) SVM with LDA. Confusion matrix for the player class.

Figure 10.10: SVM with LDA Test Set 2. Confusion matrices for every class.

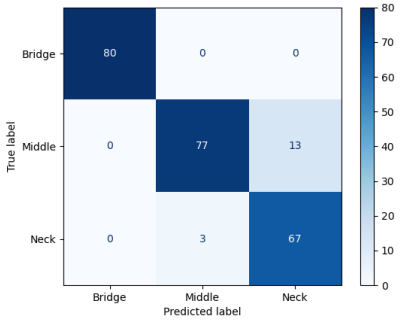
10.9 SVM without LDA Test Set 2



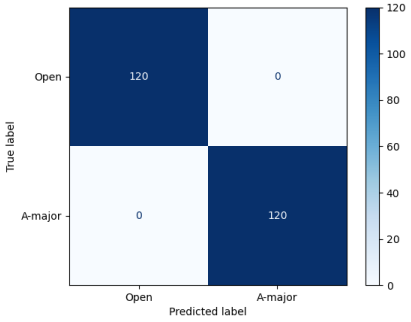
(a) SVM. Confusion matrix for the guitar type class.(2)



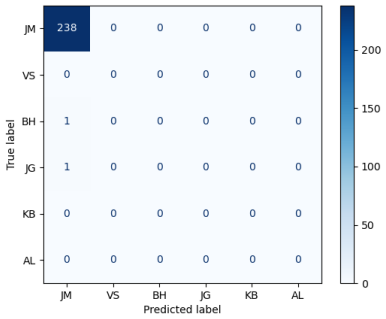
(b) SVM. Confusion matrix for the pickup class.(test2)



(c) SVM. Confusion matrix for the pickup position class.



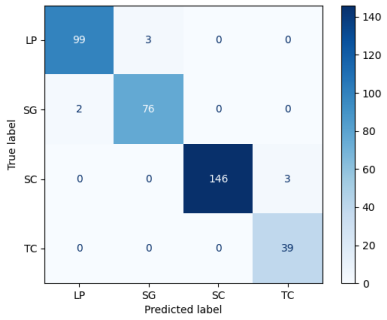
(d) SVM. Confusion matrix for the strumming method class.



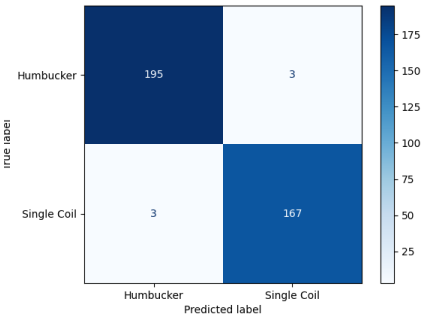
(e) SVM. Confusion matrix for the player class.

Figure 10.11: SVM without LDA Test Set 2. Confusion matrices for every class.

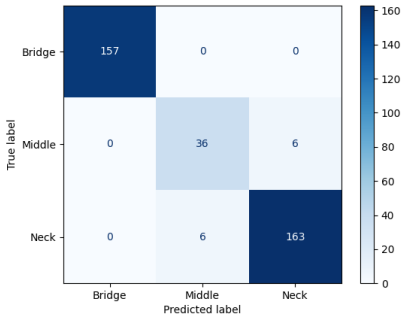
10.10 LDA Test Set 1



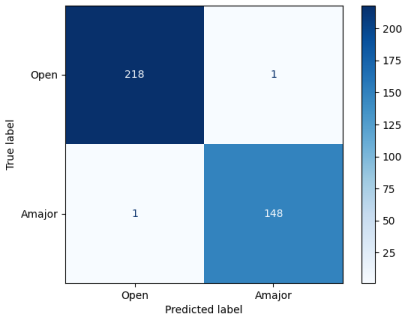
(a) LDA. Confusion matrix for the guitar type class.(2)



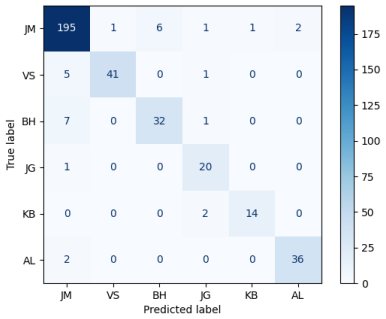
(b) LDA. Confusion matrix for the pickup class.(test2)



(c) LDA. Confusion matrix for the pickup position class.



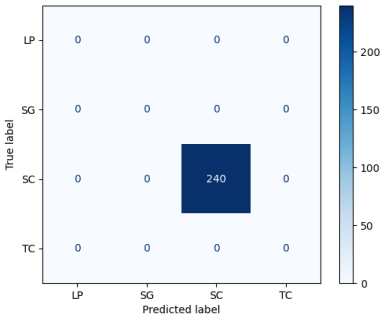
(d) LDA. Confusion matrix for the strumming method class.



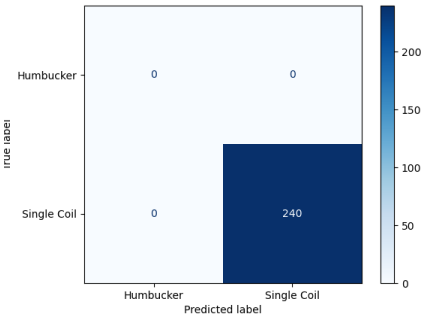
(e) LDA. Confusion matrix for the player class.

Figure 10.12: LDA Test Set 1. Confusion matrices for every class.

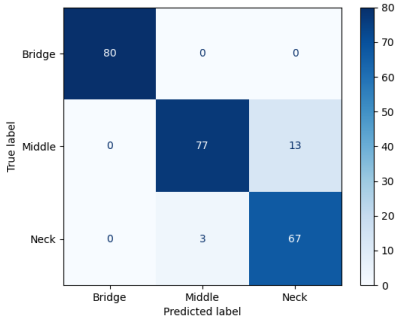
10.11 LDA Test Set 2



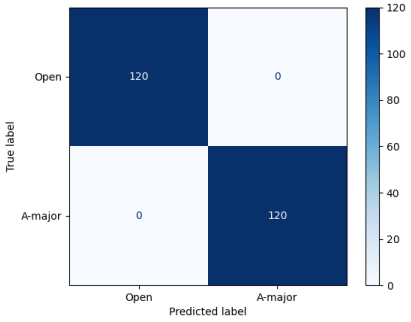
(a) LDA. Confusion matrix for the guitar type class.(2)



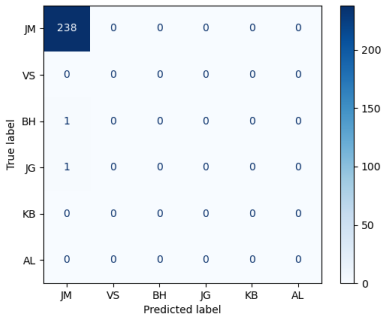
(b) LDA. Confusion matrix for the pickup class.(test2)



(c) LDA. Confusion matrix for the pickup position class.



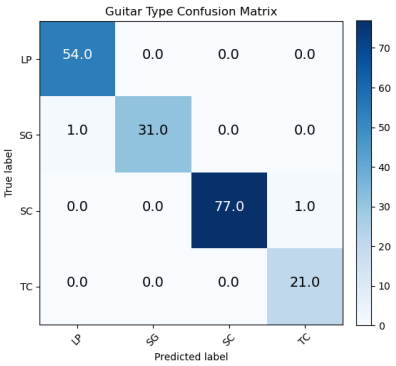
(d) LDA. Confusion matrix for the strumming method class.



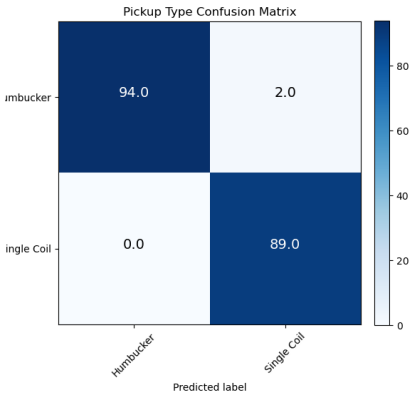
(e) LDA. Confusion matrix for the player class.

Figure 10.13: LDA Test Set 2. Confusion matrices for every class.

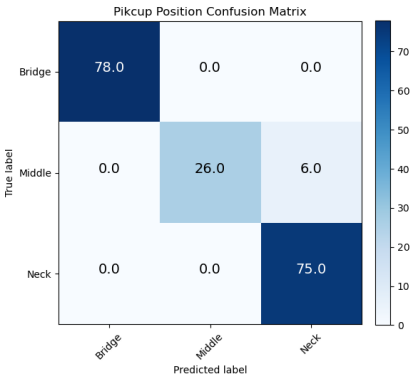
10.12 CNN Test Set 1



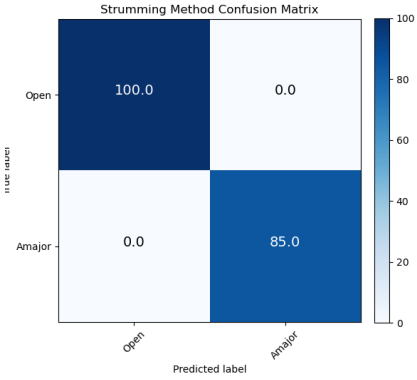
(a) CNN. Confusion matrix for the guitar type class.



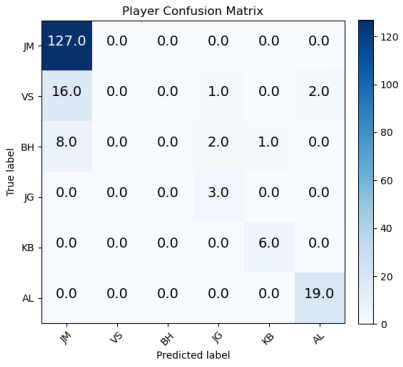
(b) CNN. Confusion matrix for the pickup class.



(c) CNN. Confusion matrix for the pickup position class.



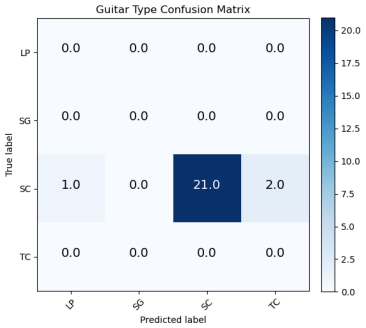
(d) CNN. Confusion matrix for the strumming method class.



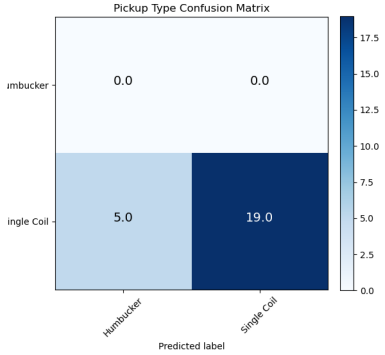
(e) CNN. Confusion matrix for the player class.

Figure 10.14: CNN Test Set 1. Confusion matrices for every class.

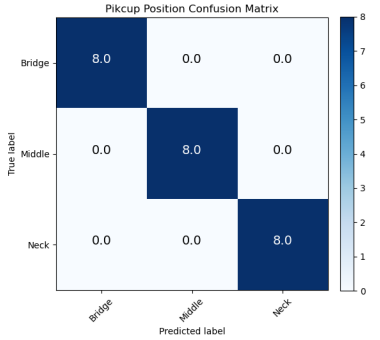
10.13 CNN Test Set 2



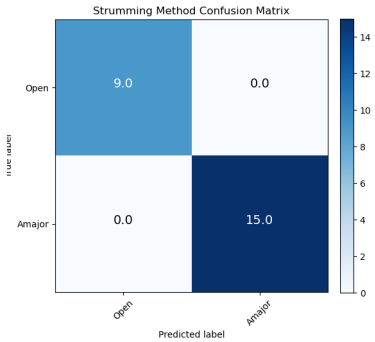
(a) CNN. Confusion matrix for the guitar type class.



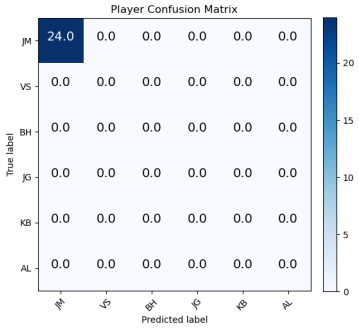
(b) CNN. Confusion matrix for the pickup class.



(c) CNN. Confusion matrix for the pickup position class.



(d) CNN. Confusion matrix for the strumming method class.



(e) CNN. Confusion matrix for the player class.

Figure 10.15: CNN Test Set 2. Confusion matrices for every class.

Bibliography

[1] Richard Mark French. *Technology of the Guitar*. Springer New York, NY, 2012. DOI: 10.1007/978-1-4614-1921-1.

[2] D.R. Lapp and Wright Center for Science Education (Tufts University). *The Physics of Music and Musical Instruments*. Wright Center for Innovative Science Education, Tufts University, 2003. URL: https://books.google.dk/books?id=dAgGkAEACAAJ.

[3] B.C.J. Moore. *An Introduction to the Psychology of Hearing*. Academic Press, 2003. ISBN: 9780125056281. URL: https://books.google.dk/books?id=tkbOivKH2HkC.

[4] Victor Rosi et al. "Uncovering the Meaning of Four Semantic Attributes of Sound : Bright, Rough, Round and Warm". In: *e-Forum Acusticum 2020*. Lyon, France, Dec. 2020. URL: https://hal.science/hal-03016066.

[5] *Collage Physics*.

[6] Ernie Ball Music Man. *Cutlass HT Guitar*. https://www.music-man.com/instruments/guitars/cutlass-ht.

[7] Paul Guy. *A brief history of the Guitar*. 2013.

[8] Dave Hunter. *Guitar Amps & Effects For Dummies*. John Wiley & Sons, 2014.

[9] Tommy Flint. *Anthology of Fingerstyle Guitar*. Mel Bay Publications, 2020.

[10] Tom Hirst. *Electric Guitar Construction*. 2003.

[11] Roger H. Siminoff. *The Luthier's Handbook: A Guide to Building Great Tone in Acoustic Stringed*. 2002.

[12] David Laganella. *Composer's Guide to the Electric Guitar*. 2011.

[13] Andre Roberts. *The Different Types Of Guitar Strings: A Complete Guide*. https://hellomusictheory.com/learn/types-of-guitar-strings/. 2023.

[14] Christopher D. Schiebel. *The 5 Different Types of Guitar Strings Explained (2023)*. https://www.guitarlobby.com/types-of-guitar-strings/. 2022.

[15] Charles R. Fraley. "An econometric estimate of hedonic price functions: The case of vintage electric guitars". English. PhD thesis. 2009, p. 141. ISBN: 978-1-109-07850-3. URL: https://www.proquest.com/dissertations-theses/econometric-estimate-hedonic-price-functions-case/docview/305169809/se-2.

[16] Heather. *7 Factors that Affect an Electric Guitar's Tone*. https://prosoundhq.com/factors-that-affect-electric-guitar-tone/.

[17] *Bolt-on neck*. https://en.wikipedia.org/wiki/Bolt-on_neck. 2022.

[18] *Set in neck*. https://en.wikipedia.org/wiki/Set-in_neck. 2022.

[19] Galeazzo Frudua. *NECK THROUGH VS BOLT ON*. https://www.frudua.com/neck-through-vs-bolt-on.html.

[20] LUTHERIE THEORY. URL: https://theelectricluthier.com/understanding-truss-rods-and-how-to-adjust-them/.

[21] The Music Ambition. *How does an electric guitar work?* 2022.

[22] Nicholas G Horton and Thomas R Moore. "Modeling the magnetic pickup of an electric guitar". In: *American journal of physics* 77.2 (2009), pp. 144–150.

[23] Herald Labial. *How does a Guitar Pickup work*. https://www.circuitbread.com/ee-faq/how-does-a-guitar-pickup-work. 2020.

[24] Kelsey Austin. *Humbuckers vs Single Coil: Ultimate Electric Guitar Pickups Guide*. https://www.axedr.com/post/humbuckers-vs-single-coil-guitar-pickups-guide/.

[25] Sam Ash. *Single coil vs humbucker pickups*. https://www.samash.com/spotlight/single-coil-vs-humbucker-whats-the-difference/.

[26] Lollar Pickups. *Understanding pickup phase*. https://www.lollarguitars.com/understanding-pickup-phase.

[27] Tyler Delsack. *What's the deal with pickup polarity*. https://www.fralinpickups.com/2017/01/23/whats-deal-polarity/. 2017.

[28] Tony Bacon. *The ultimate guitar book*. Alfred A. Knopf, Inc., 1991. ISBN: 0-375-70090-0.

[29] Tony Bacon. *Six decades of the fender telecaster : the story of the world's first solidbody electric guitar.* London: Backbeat UK, 2005. ISBN: 0879308567.

[30] Ray Minhinnett and Bob Young. *The story of the fender stratocaster*. A Carlton book. Carlton, 1999. ISBN: 1858686377.

[31] Tony Bacon. *Sunburst : how the Gibson Les Paul Standard became a legendary guitar*. Milwaukee, Wis: Backbeat, 2014. ISBN: 9781617134661.

[32] Jeff Owens.

[33] Raphael Foulon, Pierre Roy, and François Pachet. "Automatic classification of guitar playing modes". In: *Sound, Music, and Motion: 10th International Symposium, CMMR 2013, Marseille, France, October 15-18, 2013. Revised Selected Papers 10*. Springer. 2014, pp. 58–71.

[34] Daniel Piccoli et al. "Applications of soft computing for musical instrument classification". In: *AI 2003: Advances in Artificial Intelligence: 16th Australian Conference on AI, Perth, Australia, December 3-5, 2003. Proceedings 16*. Springer. 2003, pp. 878–889.

[35] Andrew Wiggins and Youngmoo E Kim. "Guitar Tablature Estimation with a Convolutional Neural Network." In: *ISMIR*. 2019, pp. 284–291.

[36] Frank Cwitkowitz, Jonathan Driedger, and Zhiyao Duan. "A Data-Driven Methodology for Considering Feasibility and Pairwise Likelihood in Deep Learning Based Guitar Tablature Transcription Systems". In: *arXiv preprint arXiv:2204.08094* (2022).

[37] *Four Types of Texture in Music*. `https://www.perennialmusicandarts.com/post/four-types-of-texture-in-music`.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[39] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[40] Frank Cwitkowitz, Toni Hirvonen, and Anssi Klapuri. "FretNet: Continuous-Valued Pitch Contour Streaming for Polyphonic Guitar Tablature Transcription". In: *arXiv preprint arXiv:2212.03023* (2022).

[41] Urja Kulkarni et al. "Comparative Study of Digital Signal Processing Techniques for Tuning an Acoustic Guitar". In: *2020 7th International Conference on Smart Structures and Systems (ICSSS)*. 2020, pp. 1–5. DOI: `10.1109/ICSSS49621.2020.9202368`.

[42] S Prabavathy, V Rathikarani, and P Dhanalakshmi. "Classification of Musical Instruments using SVM and KNN". In: *International Journal of Innovative Technology and Exploring Engineering* 9.7 (2020), pp. 1186–1190.

[43] Kerstin Dosenbach, Wolfgang Fohl, and Andreas Meisel. "Identification of individual guitar sounds by support vector machines". In: *Proceedings of the Conference on Digital Audio Effects (DAFx)*. 2008.

[44] Lewis Guignard and Greg Kehoe. "Learning Instrument Identification". In: *Acoustic Guitar* 146 (2015), pp. 10–0.

[45] Arie Livshin and Xavier Rodet. "The significance of the non-harmonic "noise" versus the harmonic series for musical instrument recognition". In: *International Symposium on Music Information Retrieval (ISMIR'06)*. 2006, pp. 1–1.

[46] Maciej Blaszke, Damian Koszewski, and Szymon Zaporowski. "Real and Virtual Instruments in Machine Learning –Training and Comparison of Classification Results". In: *2019 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*. 2019, pp. 153–157. DOI: 10.23919/SPA.2019.8936792.

[47] *Bonafide Buffer*. https://www.tcelectronic.com/product.html?modelCode=P0DD2.

[48] *Scarlett Solo*. https://focusrite.com/en/audio-interface/scarlett/scarlett-solo.

[49] *Focusrite Scarlett Solo (1st Gen)*. http://diy-fever.com/reviews/studio/focusrite-scarlett-solo-1st-gen/.

[50] *Scarlett Solo 1st Gen vs 2nd Gen vs 3rd Gen*. https://www.whippedcreamsounds.com/scarlett-solo-1st-gen-vs-2nd-gen-vs-3rd-gen/.

[51] *Clip Fix*. https://manual.audacityteam.org/man/clip_fix.html.

[52] James H. McClellan, Ronald W. Schafer, and Mark A. Yoder. *DSP First*. Pearson, 2016. ISBN: 9780136019251.

[53] S. Braun. "WINDOWS". In: *Encyclopedia of Vibration*. Ed. by S. Braun. Oxford: Elsevier, 2001, pp. 1587–1595. ISBN: 978-0-12-227085-7. DOI: https://doi.org/10.1006/rwvb.2001.0052. URL: https://www.sciencedirect.com/science/article/pii/B0122270851000527.

[54] Xuedong Huang et al. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. 1st. Prentice Hall PTR, 2001. ISBN: 0130226165.

[55] Yusnita mohd ali et al. "Analysis of Accent-Sensitive Words in Multi-Resolution Mel-Frequency Cepstral Coefficients for Classification of Accents in Malaysian English". In: *International Journal of Automotive and Mechanical Engineering* 7 (June 2013), pp. 1053–1073. DOI: 10.15282/ijame.7.2012.21.0086.

[56] Tom L. H. Li and Antoni B. Chan. "Genre Classification and the Invariance of MFCC Features to Key and Tempo". In: *Advances in Multimedia Modeling*. Springer Berlin Heidelberg, 2011.

[57] Hiroko Terasawa, Malcolm Slaney, and Jonathan Berger. "The thirteen colors of timbre". In: Nov. 2005, pp. 323 –326. ISBN: 0-7803-9154-3. DOI: 10.1109/ASPAA.2005.1540234.

[58] Min Xu et al. “HMM-based audio keyword generation”. In: *Advances in Multimedia Information Processing-PCM 2004: 5th Pacific Rim Conference on Multimedia, Tokyo, Japan, November 30-December 3, 2004. Proceedings, Part III 5*. Springer. 2005, pp. 566–574.

[59] Zhi-Hua Zhou. *Machine Learning*. Springer Singapore. DOI: 10.1007/978-981-15-1967-3.

[60] Marti A. Hearst et al. “Support vector machines”. In: *IEEE Intelligent Systems and their applications* 13.4 (1998), pp. 18–28.

[61] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. URL: /bib/bishop/Bishop2006/Pattern-Recognition-and-Machine-Learning-Christophe-M-Bishop.pdf, /bib/bishop/Bishop2006/978-0-387-31073-2_sm.pdf, https://www.microsoft.com/en-us/research/people/cmbishop/#!prml-book.

[62] Alessia Mammone, Marco Turchi, and Nello Cristianini. “Support vector machines”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 1.3 (2009), pp. 283–289.

[63] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. “Activation functions in neural networks”. In: *Towards Data Sci* 6.12 (2017), pp. 310–316.

[64] Pytorch. *ReLU function*. [Online; accessed October 25, 2022]. URL: https://pytorch.org/docs/stable/_images/ReLU.png.

[65] Computer Science Wiki. *Max-pooling / Pooling*. [Online; accessed October 24, 2022]. URL: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling.

[66] Anqi Mao, Mehryar Mohri, and Yutao Zhong. “Cross-Entropy Loss Functions: Theoretical Analysis and Applications”. In: *arXiv preprint arXiv:2304.07288* (2023).

[67] Scikit-Learn. *1.2. Linear and Quadratic Discriminant Analysis*. URL: https://scikit-learn.org/stable/modules/lda_qda.html#lda-qda. (accessed: 05.11.2023).

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