Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática
Maestría en Diseño Electrónico

Smart Audio Equalizer

TESIS que para obtener el **GRADO** de
MAESTRO EN DISEÑO ELECTRÓNICO

Presenta: **LUIS ALBERTO TEYSSIER RAMÍREZ**

Director **OMAR HUMBERTO LONGORIA GÁNDARA**

Tlaquepaque, Jalisco. 05 de febrero de 2024.

Acknowledgements.

Thanks to my beloved wife Paola who has always supported me all these years and for encouraging me to obtain this master's degree and continue pursuing my dreams.

Thanks to my parents Abelardo and Genoveva, my sister Berenice, my parents-in-law Pedro and Patricia, and my brother-in-law Jose Maria for their support and for believing in me.

I want to express my sincere gratitude to my advisor Dr. Omar Humberto Longoria Gandara for his guidance and support.

Finally, thanks to ITESO for allowing me to study for this master's degree and for providing financial support.

Abstract

Nowadays, people are used to listening to music on different devices such as smartphones, portable speakers, or car infotainments. Still, some are unaware of the integrated equalizers that those devices already possess, and others do not know how to tune them to get the best audio quality to improve the listening experience. In this thesis project, a smart way of audio equalization is proposed by using an artificial intelligence approach, so equalization presets can be applied automatically according to the music genre. Two types of equalizers were analyzed and compared. Then Matlab application was created as a proof of concept to show the advantages of these equalizers and to implement an automatic tagging process that uses Music Information Retrieval (MIR) techniques and a Convolutional Neural Network (CNN) model included in the Essentia library. The tagging results were analyzed and discussed to review the performance of Essentia's deep learning model which was included in the application. In the future, this model could be trained with different datasets that include more music genres which could improve its functionality and then it could be implemented on an embedded device using open-source solutions like Android Auto, for emulating an embedded system such as a car infotainment.

Contents

1. Introduction

1.1. Problem Statement.

Nowadays, there are different audio devices such as smartphones or car infotainments, which include an embedded audio graphic equalizer that has preset configurations for setting the best audio parameters according to a music genre such as rock, pop, electronic, hip-hop, or classical, among others. However, many of the users are not familiar with these parameters (or do not even know these configuration presets exist) and therefore, they listen to the music as it was recorded without improving or adapting its quality and features.

It is well known that most of the songs were recorded and mixed by a professional audio engineer. Nevertheless, the quality of the digital audio for many smartphone applications is not the same as a compact disc or a vinyl record, since some of the commonly used digital audio formats are lossy and compress the audio information. For example, the default quality for the Spotify app is equivalent to 96 kbit/s and most of the users do not change this setting to a higher quality.

Another point to consider is that most cars do not include a premium sound system by default, and for that reason, the quality of the car speakers could not be the best to appreciate the way the music was meant to be heard. Also, usually, the sound system is tuned by an acoustic engineer who decides the optimal settings according to his/her knowledge and experience, but this may imply that certain music genres do not sound good if the engineer did not consider them during the tuning process.

The current related work of intelligent or automatic audio equalizers is very narrow since there is not so much information about these devices nowadays. The current car infotainments or aftermarket car solutions seem to miss this kind of feature. Most of these devices have a basic graphic equalizer with three or more frequency bands which allows to manually emphasize or attenuate the audio signal to achieve good listening quality.

One of the modern ways to apply audio equalization automatically is to extract and analyze the sound characteristics by using Music Information Retrieval (MIR) techniques. These characteristics provide important information parameters like the zero-crossing rate, spectral centroid, melodic scale bands, Mel Frequency Cepstral Coefficients (MFCC), Beats Per Minute (BPM), etc., which will be discussed later, in this document. These parameters can be obtained by using an artificial intelligence method such as Deep Learning (DL) algorithms, that allow to train a model capable of identifying music genres.

1.2. Proposal and Contributions

This project proposes a Smart Audio Equalizer (SAE) based on DL algorithms for identifying the music genre of any song loaded through an application created in Matlab App Designer. Henceforth, every time a song is loaded into the application, it is analyzed and categorized. Consequently, the corresponding audio equalization is applied before it is played.

This application could be implemented on a car infotainment system, which would allow the users to not worry about manually changing the presets of the audio equalizer. Therefore, it would improve the audio experience when they listen to different kinds of music genres and if their cars are not equipped with a premium sound system.

1.3. Thesis Outline

Chapter 2 describes the fundamentals of audio equalization and describes two types of audio equalizers: graphic and parametric. Some related works to smart audio equalizers are mentioned in this chapter as well. On the other hand, the concept of MIR, its applications, and how it is used for solving the automatic music tagging problem are introduced.

1. INTRODUCTION

Chapter 3 mentions the tools that were chosen for executing the automatic music tagging algorithm: Essentia library and its algorithm MusiCNN, which makes use of a convolutional neural network for determining the music genre tags of music. Then the methodology used to design the smart audio equalizer Matlab application is described.

The application implementation and how MusiCNN functionality was integrated is presented in Chapter 4. The experimentation results are shown, analyzed, and discussed in Chapter 5. Finally, the conclusions and future work suggestions are presented in Chapter 6.

2. Background.

2.1. Audio equalization.

To design the smart audio equalizer, the concept of audio equalization must be understood first: It can be defined as the process of adjusting the gain of an audio signal in a certain frequency range (usually divided into different frequency bands) for improving its tone or timbre quality. This is achieved by using an audio equalizer, which is a device composed of a group of audio-signal filters that are designed according to different parameters such as gain, center frequency, or Q factor (which allows to adjust the bandwidth of each filter, since its value is equal to the ratio of the center frequency and the bandwidth), depending on its type.

Over the years, audio equalizers have been evolving from the ones used for telephony to correct the spectrum of the sound at the receiver side to the most modern ones used in the current audio receivers to enhance the quality of the audio to satisfy the more demanding audiophile people. All the history of audio equalizers can be read in the article by Välimäki and D. Reiss [1], where they describe the different approaches that some engineers have created to innovate in this field of audio processing. For example, Daniel Flickinger created the device known as sweepable audio equalizer that allowed the selection of different frequencies and gain values using only three overlapping bands.

Another important concept related to audio equalizers is the audible spectrum, which is located in a frequency range from about 20 Hz to 20 kHz for humans. However, the average upper limit for adults is closer to 15 or 17 kHz. This spectrum can be divided into sub-bands which are described below according to [2]:

1. **Sub-bass (from 20 Hz to 60 Hz):** This is the lowest range that humans can hear and the sound waves at these frequencies are more felt rather than heard, which gives the sensation of a more powerful sound. The bass guitar is located at this

sub-band since its lowest pitch is normally at 41 Hz. However, if this sub-band is boosted too much, the resulting sound is distorted.

2. **Bass (from 60 Hz to 250 Hz):** This sub-band is very popular since most modern music contains bass sounds located between 90 and 200 Hz and gives the impression of a "fat" sound. Nevertheless, it can result in an irritating sound if it is emphasized excessively.

3. **Low Midrange (from 250 to 500 Hz):** The sound located in this sub-band can add depth to the instruments when it is properly equalized. Also, it may sound unnatural if it is decreased too much. But, if it is boosted too much, the sound can be perceived as "muddy". Deep masculine vocals are located within this range.

4. **Midrange (from 500 Hz to 2 kHz):** This range is important since it includes the fundamental notes of vocals, and it can improve the sound of the musical instruments, making them stand out among other tones.

5. **Upper Midrange (from 2 kHz to 4 kHz):** Humans are more sensitive to this range and if it is not adjusted properly, it will be noticed immediately. For this reason, if the audio signal is boosted above 3 kHz it may result in listening fatigue. On the contrary, if the signal is decreased in this range, it could impact the quality perception and timbre.

6. **Treble (from 4 kHz to 6 kHz):** This range is also known as the presence range because it can improve the sound quality and a "live performance" perception. If it is correctly boosted, it gives the impression of a warmer sound to orchestral music. The excess of boosting may result in an irritating harsh sound and the excess of cutting can make the sound appear distant.

7. **Upper Treble (from 6 kHz to 20 kHz):** This is also known as the brilliance range since it is responsible for adding detail and "sparkle" to the sound because it is entirely composed of harmonics. When it is properly adjusted, the sound can be

perceived as High-Fidelity quality, due to the addition of clarity. However, if it is over-boosted, it can result in a hissing sound that can be irritating and cause fatigue.

To have a clear idea of the audio frequency spectrum, Figure 2-1. Instrument frequency chart [3]. shows the frequencies where some instruments are located, hence this can be used as a guide to improve the sound using an audio equalizer:

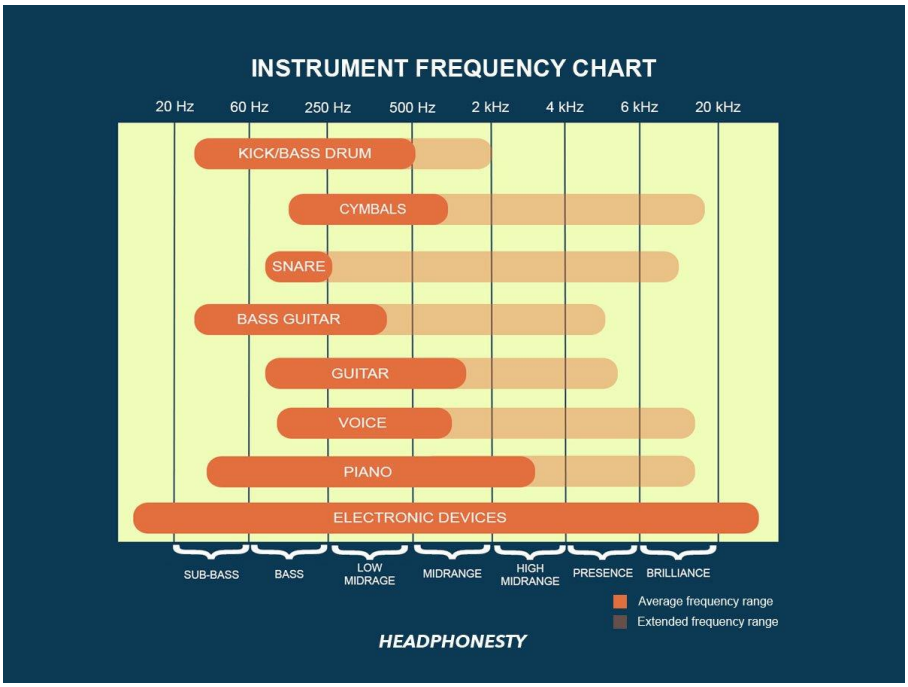

Figure 2-1. Instrument frequency chart [3].

Nowadays and according to [4], the function of audio equalizers is to correct and enhance the performance of the audio system by correcting the loudspeaker response and its interaction with the room, the headphones improvement to ensure natural music listening, and the enhancement of recorded music.

The most common equalizer types are the graphic equalizer and the parametric equalizer. Both were implemented in this thesis project; hence they are described below.

2.1.1 Graphic equalizer.

The graphic equalizer is a very popular device for sound enhancement since it allows the adjustment of the gain of multiple frequency regions of an audio signal. In this case, the term "graphic" refers to the fact that the position of the slider knobs that represent each band can be seen as a graph of the magnitude response versus frequency. This type of equalizer can be found on modern digital music players, such as those available on home audio receivers, mobile phones, or car infotainments, as shown in Figure 2-2. Car infotainment graphic equalizer [5].and Figure 2-3. Graphic equalizer settings included on Samsung phone with Android operating system.. These equalizers usually have several preset settings for different music genres, such as pop, rock, jazz, classical, and electronic.

Figure 2-2. Car infotainment graphic equalizer [5].

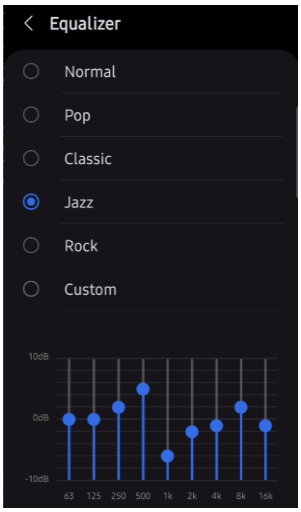

Figure 2-3. Graphic equalizer settings included on Samsung phone with Android operating system.

2. BACKGROUND

As explained by [6], the bands that conform to a graphic equalizer are distributed logarithmically in frequency to match human hearing perception, and its gain is the only parameter that the user can boost or decrease (usually in a range of ±12 dB) since the center frequency and bandwidth values are fixed. According to [1], this logarithmic distribution is usually specified using a fixed ratio between each band and it determines if the equalizer is of one-octave type or one-third octave.

An octave is a musical interval defined by a doubling in frequency, for that reason the octave implementation has a ratio of $R=2$ between each band. On the other hand, the one-third octave implementation has three bands per octave and therefore a ratio of $R^3=2$ [1]. Therefore, taking these ratio values into account, an octave equalizer is composed of 10 frequency bands, which are in the range of 31 Hz to 16 kHz, whilst a one-third octave equalizer has 30 bands ranging from 25 Hz to 20 kHz. These ranges were chosen to cover human hearing perception. However, not all people can hear up to 20 kHz since this depends on the age or the health of the ear.

Table 2-1. Octave frequency bands according to the ISO standard [1]. shows the standardized ISO values for an octave and one-third octave implementations:

Lower frequency f_l (Hz)	**Geometric mean frequency f_c (Hz)**	**Upper frequency f_u (Hz)**
22	31.5	44
44	63	88
88	125	177
177	250	355
355	500	710
710	1000	1420
1420	2000	2840
2840	4000	5680
5680	8000	11360
11360	16000	22720

Table 2-1. Octave frequency bands according to the ISO standard [1].

The graphic equalizer implemented in this project is a bank of bandpass filters arranged in parallel as shown in Figure 2-4. Parallel implementation of graphic equalizer [1]

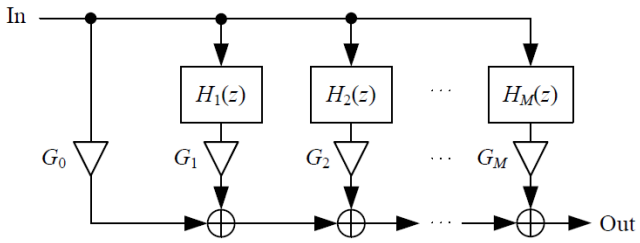

Figure 2-4. Parallel implementation of graphic equalizer [1]

In this setup, the audio signal passes through every passband filter which allows only a narrow frequency to pass, and then a gain is applied according to the value of the controls of every band in the equalizer. Finally, the output of every band is summed together with the original signal which can be escalated by applying a gain G_0. This type of implementation is the best to get rid of quantization noise and the accumulating phase errors according to Välimäki and Reiss [1].

Therefore, the transfer function of a parallel graphic equalizer in the z-plane domain can be expressed as the sum of all the bandpass filters and the gain applied to the input signal as expressed in the Equation 1-1.

$$H_{PGEQ}(z) = G_0 + \sum_{k=1}^{K} G_k H_k(z) \tag{1-1}$$

where $H_k(z)$ corresponds to each filter of the kth band and G_k is the gain of the kth band.

An example of spectrum magnitude response of this implementation is shown in Figure 2-5. Magnitude response of a parallel implementation where the red circles represent the gains at octave bands [1].

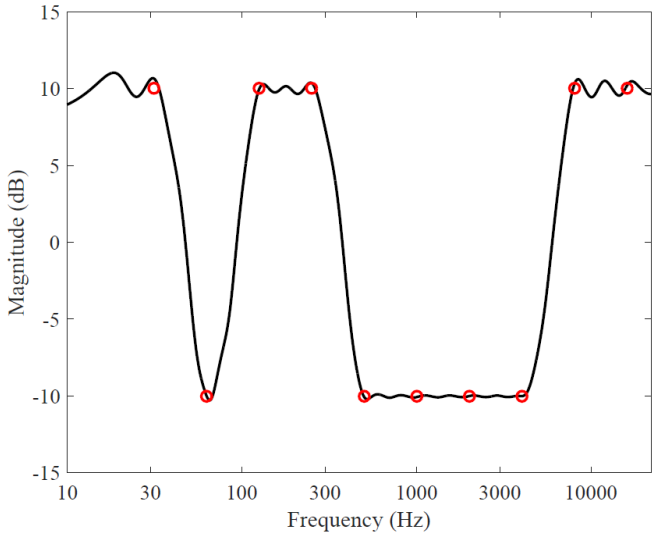

Figure 2-5. Magnitude response of a parallel implementation where the red circles represent the gains at octave bands [1].

2.1.2 Graphic equalizer presets.

Typically, graphic equalizers that have a user interface (i.e., the graphic equalizer of a car infotainment, or a smartphone application), provide a group of presets that can be chosen according to the music genre. The following presets were taken from iTunes [7] for Windows:

1. **Rock music:** For this music genre, it is important to enhance the low and high frequencies due to the presence of electric guitars and drums. In this case, the midrange frequencies can be left as they are or decrease a little.

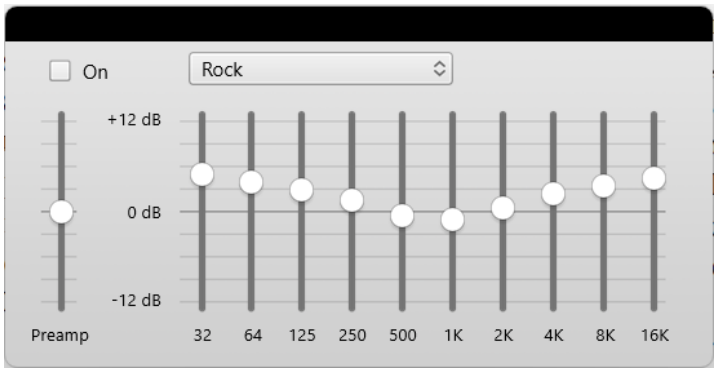

Figure 2-6. Equalizer preset for rock music.

2. **Pop music:** Since pop music is more focused on vocals and midrange instruments, the mid frequencies are boosted in this preset.

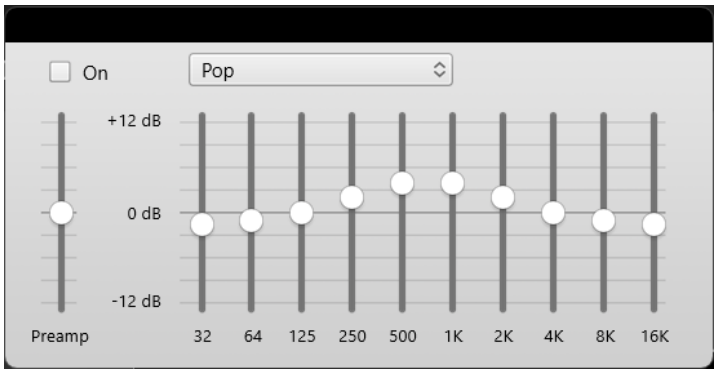

Figure 2-7. Equalizer preset for pop music.

3. **Electronic music:** In this genre, the bass is boosted due to its typical low frequency beats. Vocals do not need to be improved in this case, so the mid frequencies are just slightly adjusted.

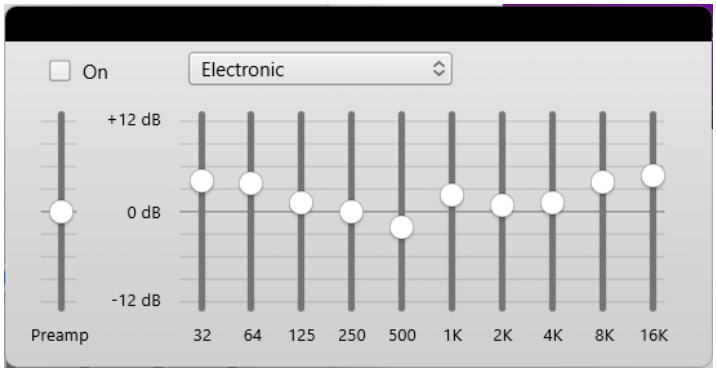

Figure 2-8. Equalizer preset for electronic music.

4. **Classical music:** This music genre contains various instruments such as violin, cello, piano, flute, and other orchestral ones. For this reason, this type of genre only needs to be boosted a little.

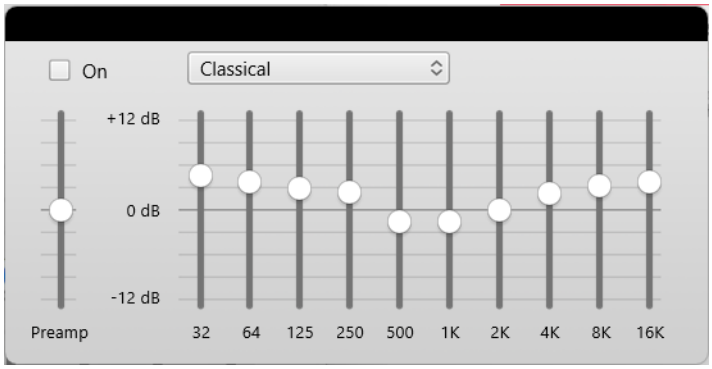

Figure 2-9. Equalizer preset for classical music.

5. **Jazz music:** This music genre has dominant sounds in the brilliance (high frequencies), sub-bass, and midrange since the vocals and the used instruments are located on these ranges.

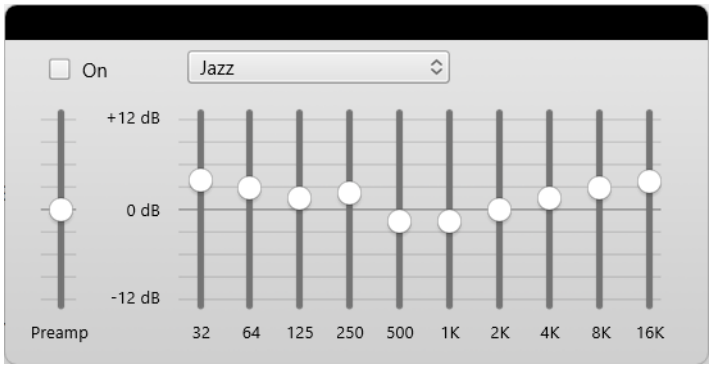

Figure 2-10. Equalizer preset for jazz music.

6. **Metal music**: For this genre, it is needed to boost the sub-band ranges where the electric guitars, vocals, and drums are located (it is pretty similar to rock music preset). In this case, the bass and brilliance are emphasized. Low midrange and midrange subbands are slightly cut to avoid the muddy effect in the sound.

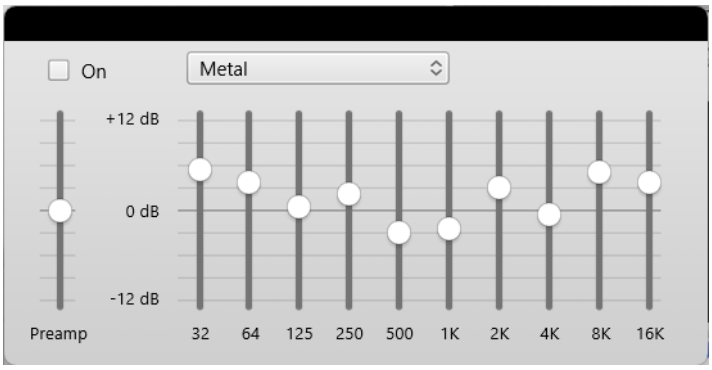

Figure 2-11. Equalizer preset for metal music.

These presets were used as a guide for the Matlab graphic equalizer implementation that is described in the following chapters. There is no standardization for these presets, because usually they are created by a specialized audio engineer, and the settings are recorded according to his/her criteria.

Table 2-2 summarizes the graphic equalizer presets and its values.

Preset	Band (Hz)	Value (dB)	Preset	Band (Hz)	Value (dB)	Preset	Band (Hz)	Value (dB)
Rock	32	4.8	Pop	32	-1.5	Instrumental	32	3
	64	4.5		64	-1.2		64	2.8
	125	3		125	0		125	0
	250	1.5		250	3		250	2.8
	500	0		500	3.5		500	3
	1K	-0.5		1K	3.5		1K	0.5
	2K	0.5		2K	1.5		2K	3.2
	4K	2.8		4K	0		4K	5
	8K	3.2		8K	-1.2		8K	3
	16K	4.5		16K	-1.5		16K	3.5
Preset	**Band (Hz)**	**Value (dB)**	**Preset**	**Band (Hz)**	**Value (dB)**	**Preset**	**Band (Hz)**	**Value (dB)**
Jazz	32	4.5	Electronic	32	3.5	Metal	32	4.5
	64	3		64	3.2		64	4
	125	0		125	1.5		125	3
	250	0		250	0		250	0
	500	-1.5		500	-1.5		500	-3.5
	1K	-1.5		1K	3		1K	-1.5
	2K	-1.5		2K	1		2K	0
	4K	0		4K	1.5		4K	2.8
	8K	3		8K	3.5		8K	3.2
	16K	4.5		16K	5.5		16K	4.5

Table 2-2. Summary of Graphic Equalizer presets.

2.1.3 Parametric equalizer.

Unlike the graphic equalizer, the parametric equalizer can tune the audio quality more flexibly, since it allows the user to add a peak or notch at a certain point in the audio spectrum by modifying the values of the center frequency, gain, and the Q factor. This allows the spectral content to remain unmodified at the points located far away from the peaks and notches.

The advantage of this equalizer is that certain elements of the music such as a specific instrument can be improved. On the contrary, unwanted sounds can be excluded. For example, if the power line hum located at 50 or 60 Hz can be heard in a recording, it is possible to remove it by adjusting a notch at that frequency range. The Q factor value can be increased or decreased as well to modify the bandwidth and therefore not affect other elements of the music.

To better understand a parametric equalizer, Figure 2-12 depicts a graphical user interface (GUI) of a parametric equalizer plugin. It has some knobs that can be adjusted by the user to modify the gain, center frequency, and the Q factor. The spectrum magnitude response is shown in the upper part of the GUI.

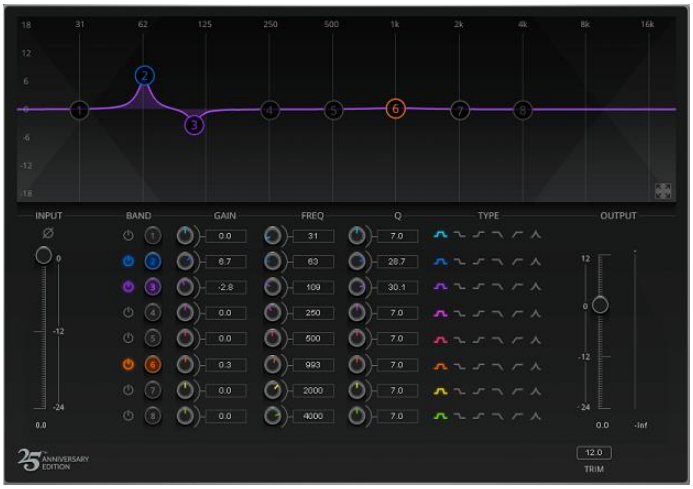

Figure 2-12. Waves Q10 parametric equalizer plugin.

2.1.4 Types of filters used on parametric equalization.

Normally, the parametric equalizers are built using first or second-order shelving filters for boosting or attenuating the lowest and highest frequencies without affecting other frequencies in the spectrum. This means that this kind of filter will not entirely cut off the other frequencies like a low-pass or high-pass filter. Figure 2-13 and Figure 2-14 show the magnitude response of low-shelf and high-shelf filters with a crossover frequency set to 1000 Hz.

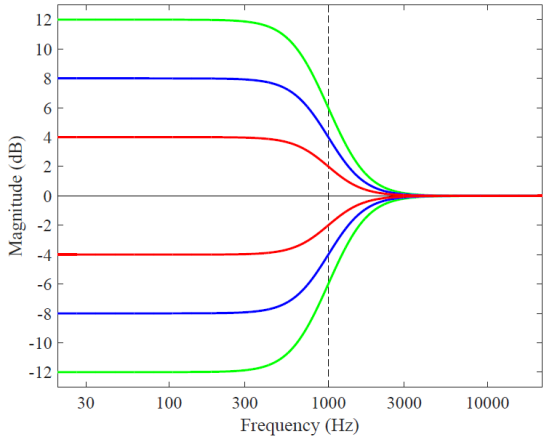

Figure 2-13. Magnitude response of a second-order low-shelf filter with a crossover frequency of 1000 Hz and different gain values [1].

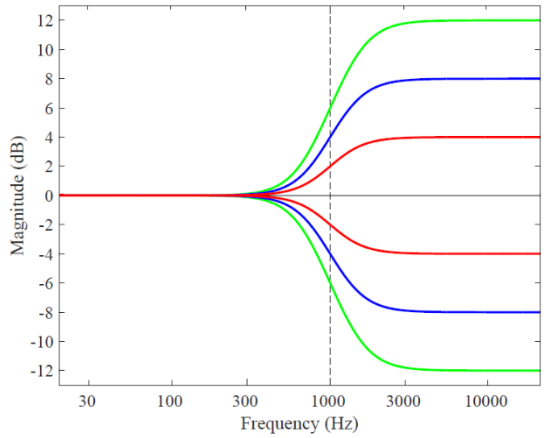

Figure 2-14. Magnitude response of high-shelf filter with a crossover frequency of 1000 Hz and different gain values [1].

Low-pass and high-pass filters can be used as well, depending on the desired behavior of the parametric equalizer. The bands located between the low/high shelf filters are equalized using peak/notch filters connected in cascade, which allow to boost or attenuate the audio signal at a certain point of the frequency spectrum (depending on the gain's value of course) and its bandwidth can be adjusted as well by modifying the value of the Q factor. The highest the value of the Q factor, the narrower the peak/notch will be. This allows us to get an idea of the sharpness of the transitions between the reference value of 0 dB and the value of the gain [8].

To visualize better this concept, Figure 2-15 shows an example of the magnitude response of a parametric equalizer conformed by peak/notch filters with frequencies centered at 50 Hz, 200 Hz, 1000 Hz, and 4000 Hz respectively. In this case, the Q factor is set to 1.25. If the value of the Q factor is increased, the width of the peaks or notches will be narrower, which allows filtering of certain frequencies with more accuracy.

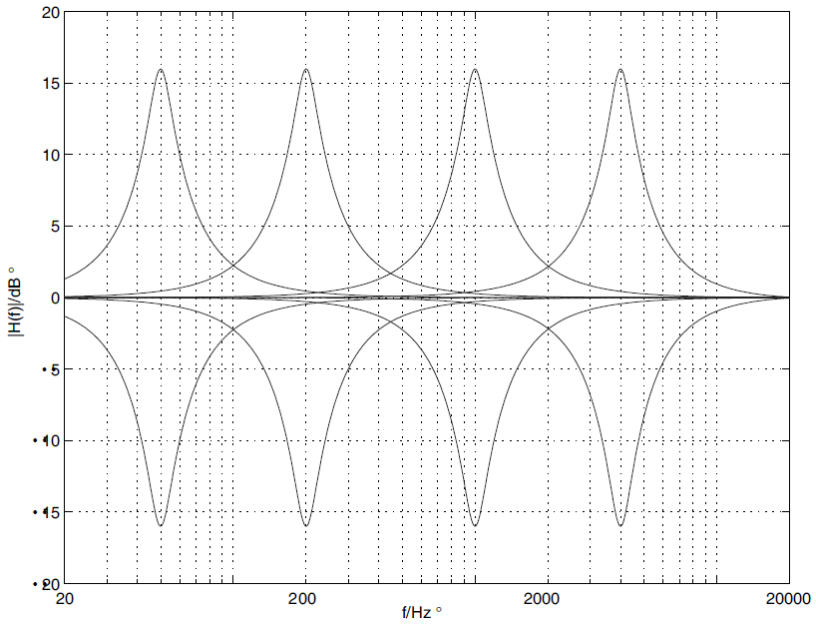

Figure 2-15. Magnitude response of peak/notch filters at different center frequencies when Q=1.25 and boost/cut ± 16 dB [8].

Figure 2-16 shows the construction of a parametric equalizer using shelving and peak/notch filters connected in cascade:

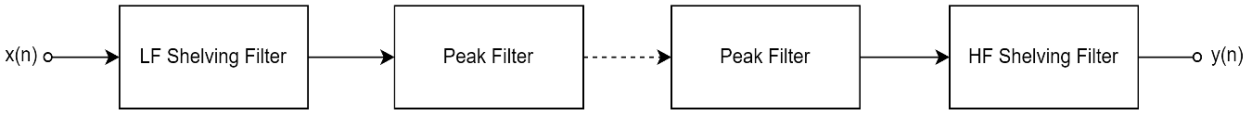

Figure 2-16. Cascade connection of shelving and peak/notch filters [8].

2.1.5 Parametric equalizer presets.

Although it is not so common to have presets on a parametric equalizer, some audio plugins allow you to save the settings for applying them later. Tyler Connaghan from Mastered Blog [9] created a guide for the best parametric equalizer settings:

1. **Rock music**: In this case, it is recommended to emphasize the bass and brilliance sub-bands to give more presence to drums and electric guitars.

Figure 2-17. Rock music preset for FabFilter Pro-Q 3 parametric equalizer plugin [9].

2. **Pop music:** Since the pop genre is more focused on the vocals, the midrange from 1 kHz to 3 kHz should be boosted. Also, normally the songs contain low beats in the range of 60 to 200 Hz and high frequency elements between 8 kHz and 16 kHz that need to be emphasized as well.

Figure 2-18. Pop music preset for FabFilter Pro-Q 3 parametric equalizer plugin [9].

3. **Electronic music:** The sounds on the bass region are one of the elements that define this music genre. Therefore, it is important to boost the range between 32 Hz and 125 Hz. Some audio enthusiasts recommend cutting the range between 250 Hz to 500 Hz to attenuate the sound "muddiness" that can be located on these frequencies.

Figure 2-19. Electronic music preset for FabFilter Pro-Q 3 parametric equalizer plugin [9].

4. **Classical music:** In the case of this genre, normally the sound is not boosted so much because most of these tracks are live recordings. Hence it is recommended to boost low frequencies from 32 Hz to 250 Hz and the high frequencies in the range of 4 kHz to 16 kHz.

Figure 2-20. Classical music preset for FabFilter Pro-Q 3 parametric equalizer plugin [9].

2.1.6 Intelligent audio equalization.

Through the years, intelligent audio equalization has been evolving: In the 90's there were some home audio high-fidelity systems such as Kenwood UD-901 [10] that had an "AI equalizer" integrated. This equalizer was activated by the user by pressing a button after introducing a compact disc (CD). Then, the system performed a sampling of a track for thirty seconds to determine the best equalization preset according to the music genre of the disc.

Later, some methods for applying audio equalization automatically to multichannel audio were created: Hyoung-Gook Kim et al [11], proposed in 2006 an automatic equalizer method for enhancing audio in real-time in an automobile audio system. This method extracts timbre and rhythm features which are fed to a Gaussian Mixture Model (GMM) to achieve music genre classification (the concept of music features is explained in the next chapter).

Once the music genre is determined, one of the five genre presets is applied, and therefore audio equalization is achieved for each song instead of having a generic preset for all the music.

There is another approach designed by Ma et al in 2013 [12], which uses an algorithm based on the Yule-Walker method and some IIR digital filters to fit the signal equalization to a target spectral distribution in real-time (on a VST plugin). However, this kind of equalizer aimed to enhance audio during mastering, which is the final quality check stage in music recording process where the sound is 'polished' to improve the audio quality and to elevate the listening experience [13].

In the year 2015, Gonzalez et al [14] created a smart equalizer that extracts eighteen music features by using the MIR (Music Information Retrieval) Toolbox from Matlab, such as zero-cross, centroid, low energy, roll-off, flux, MFCC (Cepstral Coefficients of Mel Frequencies), amongst others. Then those features were introduced in a fuzzy logic inference engine to determine the equalization levels for each frequency band, according to the music genre.

The company Gracenote proposed a dynamic EQ in the same year [15], which can apply an equalization preset by searching for it within its huge song database. Since many car audio units already use the Gracenote service in the U.S. to get song metadata, in this case, the equalization present is previously assigned to every song and then applied in the background without the user's noticing. However, if the song is not in their database, an equalization preset cannot be determined. This feature was planned to go live in early 2019.

One of the most recent works related to smart equalization is the one created by Iriz et al in 2020 [16], where they propose a system that extracts 13 MFCC of a 30-second song's segment (which is equivalent to 600 window shots per song). Then, the resulting 600*13 parameters are introduced into a convolutional neural network with 2 hidden layers for determining up to 10 music genres from the GTzan dataset: Blues, Classical, Country, Disco, Hip-Hop, Jazz, Metal, Pop, Reggae and Rock.

In this way, the neural model is not too complex to embed on the platform with limited resources such as a smart speaker or a smartphone. Once it determines the corresponding music genre, it applies the corresponding equalizing preset to each segment. Nevertheless, if the system determines more than one genre, a mixture of equalization profiles weighed by the probability of belonging to a certain genre is applied.

Figure 2-21 depicts the architecture of the Iriz smart equalizer:

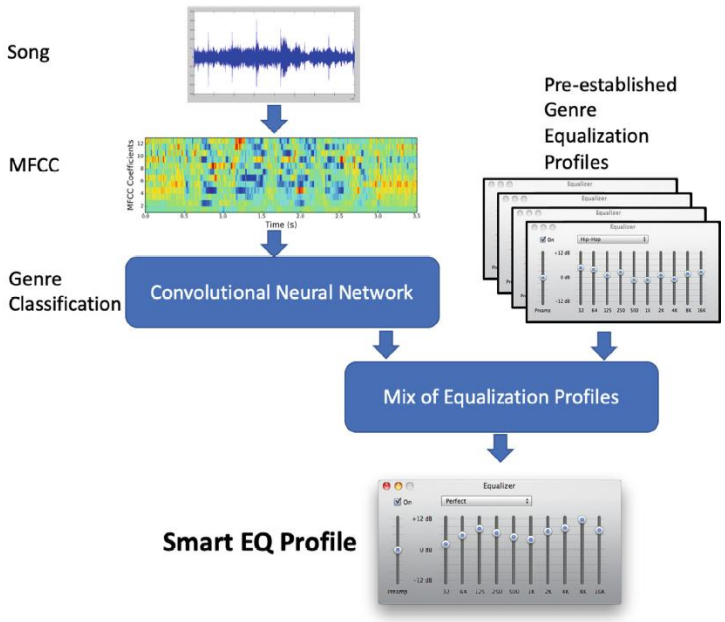

Figure 2-21. The architecture of Iriz smart equalizer [16].

The smart equalization method implemented in this thesis project is similar to the work of Iriz et al, but in this case, a pre-trained CNN (Convolutional Neural Network) and a different dataset were used, since it was decided to use an algorithm that is already included in a C++ library called Essentia [17].

More details about the architecture and the integration process are explained in the methodology chapter.

2.2. Music Information Retrieval.

To be able to understand how the algorithm of the smart equalizer works, the concept of Music Information Retrieval (MIR) must be defined first: Knees and Schedl [18] define it as the research and development of intelligent methods for extracting and analyzing music descriptors from the audio signals or their contextual sources. Downie [19] defines MIR as a multidisciplinary research area that includes computer science, information retrieval, music theory, musicology, audio engineering, and Digital Signal Processing (DSP) among others, which is aimed to develop innovative content analysis to preserve, access and research collections of musical material.

2.2.1 MIR Applications

One of the most common applications for MIR is audio identification, which consists of finding a query (a fragment of audio recorded with a smartphone) within a large database, that allows retrieval of song metadata such as song title, artist name, and album. Popular smartphone applications like Shazam, Sound Hound, Gracenote MusicID, or MusicBrainz use this technology for music fingerprinting.

Another application is “cover song” identification, where state-of-the-art algorithms are used to find similarities between two songs by extracting descriptors related to melody, bass line, and harmonic progression. There is also an application for automatically generating playlists (also known as personalized radio stations) according to the user’s music preferences which are used by popular music services such as Pandora, Spotify, Last.fm, iTunes Radio, and YouTube Music.

On the other hand, MIR can be used for researching areas like instrument detection, which consists of identifying the different instruments present in a recording. Also, the sound source separation, where the elements like tracks, voices, or sounds are extracted.

Nevertheless, the related application to this project is classification, which relies on categorical data like genre or tags. Knees and Schedl [20] define the genre as a class of similar music that describes a musical style, such as rock, pop, classical, jazz, blues, metal, and electronic, among others. However, there are some genres based on a specific era like 90's rock, a location like Chicago blues, or even genres related to marketing such as "love songs". Even though a song can be associated with two or more genres, usually it is related to only one.

2.2.2 Music features extraction.

The first step for performing MIR analysis and processing is the extraction of content descriptors depending on the application. This process consists of converting audio raw data into a music feature representation like pitch, rhythm, melody, timbre, or loudness. In the case of genre classification and auto-tagging applications, the features of interest that need to be extracted are the ones related to rhythm and timbre.

Usually, a preprocessing step is performed before extracting the desired features such as downsampling/upsampling the audio signal data to a different sample rate, downmixing to only one channel, normalization, or removing the DC offset. Then, the next step is to extract the desired features by using DSP techniques.

The MPEG-7 standard classifies the audio descriptors into low-level and high-level categories. The low-level descriptors are usually related to loudness and timbre (which are associated with the time variation of the spectrum, the time variation of energy, and the spectral envelope shape) [20].

Low-level descriptors can be divided into the following groups:

Group name	Descriptors
Temporal descriptors	• Zero Crossing Rate • Energy envelope • Log attack time • Temporal centroid
Cepstral descriptors	• Mel-Frequency Cepstrum Coefficients
Spectral descriptors	• Spectral centroid • Spread • Skewness • Kurtosis • Spectral slope • Spectral roll-off • Spectral flatness • Spectral flux • Spectrogram
Perceptual descriptors	• Loudness • Sharpness

Table 2-3. Low-Level descriptors [20].

On the other hand, the high-level features are the ones that are more related to the symbolic representation of music like melody, chords, harmony, key, and instruments used, amongst others.

2.3. Melodic scale and Mel Spectrogram.

As mentioned before in Section 2.1, humans can perceive a range of sound frequencies between 20Hz and 20kHz approximately and it varies according to age because this perception decreases over time. However, when humans try to perceive the pitch feature in sound, it is not a linear scale. For instance, a person can differentiate between a 500 Hz and a 1000 Hz sound but cannot tell the difference between a 10kHz and a 10.5kHz sound, even though it is the same difference range of 500 Hz. Therefore, this perception is nonlinear.

For this reason, a scale based on the measure of tone height was introduced in 1937 by Stevens et al [21] and it was called the Mel scale. They used empirical data to build analytical models of this scale based on some psychological experiments. Figure 2-22 shows the Mel scale plot:

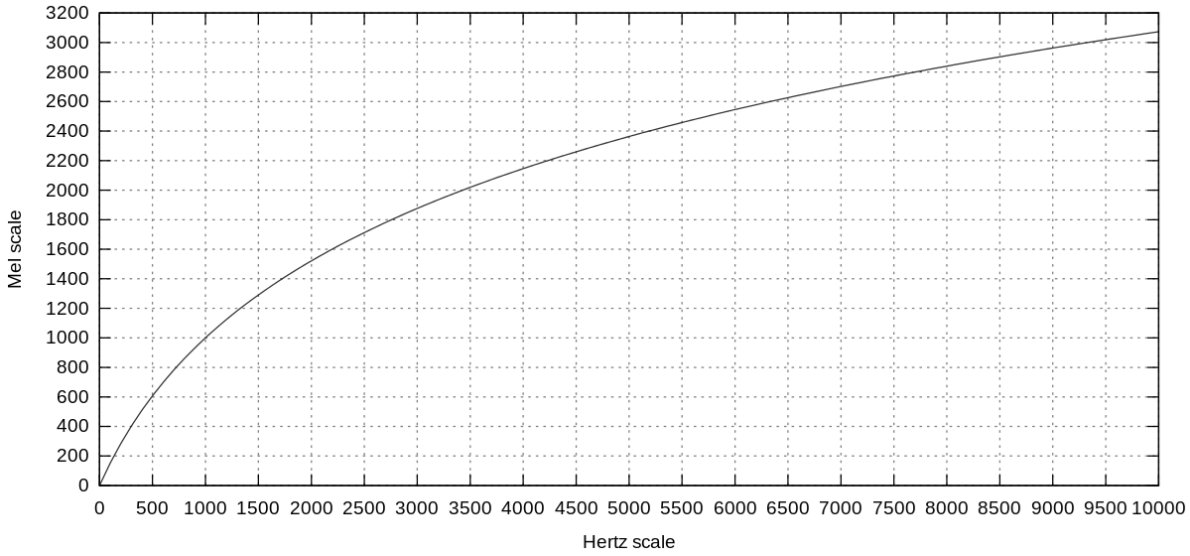

Figure 2-22. Mel scale [21].

The Hertz values can be converted to the Mel scale with the O'Shaughnessy formula [22]:

$$m_s(f) = 2595 \cdot log_{10}\left(1 + \frac{f}{700\ Hz}\right)$$

To know how the audio signal frequency content varies over time on the Mel scale, a Mel spectrogram is needed. A spectrogram is a way of representing the signal's loudness as it changes over time at different frequencies. It is calculated by obtaining the Fast Fourier Transform (FFT) of overlapping window segments of the signal. This process is known as the Short-Time Fourier Transform (STFT). More details about the calculation can be found at [23] and [24]. An example of a Mel Spectrogram is shown in Figure 2-23.

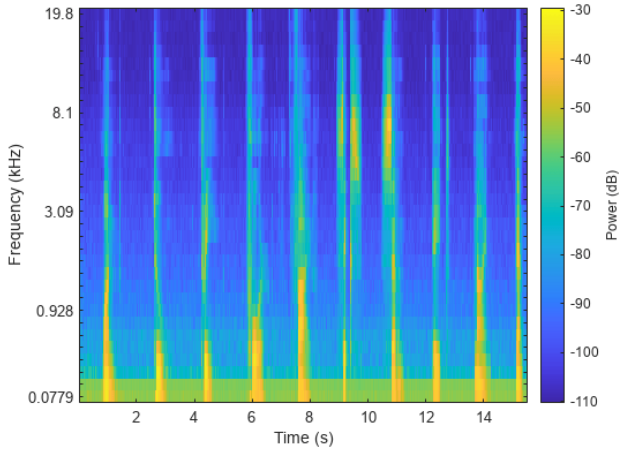

Figure 2-23. Mel Spectrogram [24].

The spectrogram data is used as input for some deep learning models that use CNNs since it is considered a type of image. Therefore, it is useful for music genre classification/tagging applications, as it is explained in the next section.

2.4. Music automatic tagging.

One of the most interesting and important applications of MIR is automatic tagging because it allows us to identify artists, music genres, moods, and instruments, among other high-level information. In the past, tagging was done manually by embedding the metadata to the music tracks, which was a big effort and time-consuming task for musicians or fans. Nowadays, this application is used by some music services such as auto playlist generators and music recommendation systems.

Music automatic tagging is considered a classification task whose objective is to predict music tags of an audio signal. However, it differs from music genre classification because it is possible to tag a song with multiple tags. For this reason, it requires more data and a larger capacity model with optimized methods to solve the problem.

Nowadays, the most common way of executing this task is by using ML approaches such as random forests, decision trees, Support Vector Machines (SVM), or DL such as neural networks [25]. This thesis project focuses on the DL way since the used audio processing library contains a CNN model that is explained in the next chapter.

Choi et al [26], affirm that CNNs are widely used for this purpose because they are designed to learn hierarchical features over multilayer structures such as music tags, which are one of the topmost high-level features that can represent information at the song level. Also, the properties of CNNs like distortion, translation, and local invariances are useful to learn music features since the events that are relevant for tagging can occur at any frequency or time range.

Normally, three steps are required for this application. The first step is the preprocessing of the audio which should be moderated to retain as much information as possible (i.e., downsampling). The second step is to extract the required features that will help to estimate the type of tags of interest. Commonly, MFCCs or spectral features like Mel spectrograms are used for this purpose. However, MFCCs do not perform well with large time scales which are more suitable for music tagging. Therefore, its time scale has to be enlarged. On the other hand, Mel spectrograms are more stable to time-warping deformation on larger time scales, but they can lose some significant information that can affect the third step, which is the deep learning process. Choi et al concluded that using Mel spectrograms resulted in better performance in comparison to STFT and MFCC.

The deep learning process is divided into two stages: the training stage and the testing stage. The training is an iterative process where the network weights are adjusted to reduce the loss, which is a function that indicates the difference between the predicted output and the truth ground output with respect to the current weight values. Figure 2-24 shows the training stage where the classifier system learns the tag models based on the relationship that exists between the tags and the extracted feature vectors.

Since the scope of this thesis project was to use a pre-trained model, the training process is not explained in detail. Nevertheless, there are many resources like Google Developers [27] or OpenAI [28] to read more about this topic.

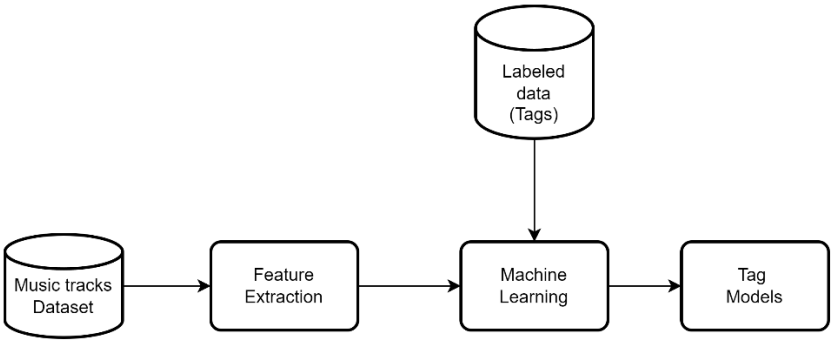

Figure 2-24. Deep learning training process.

Once the system is trained, the obtained model can be used to infer the music genre, mood, or instrument tags when it is fed with new music data, as shown in Figure 2-25:

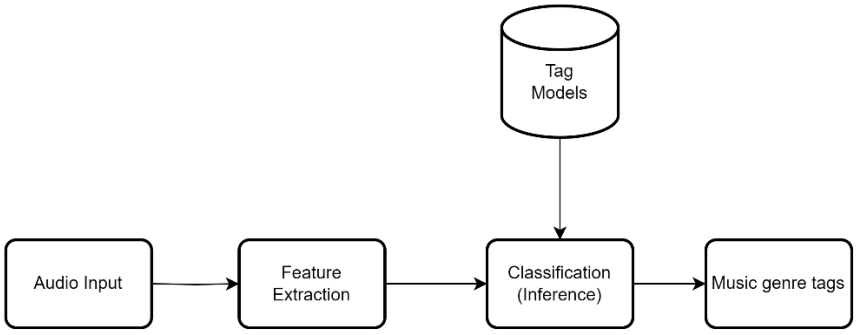

Figure 2-25. Deep learning testing process.

Its accuracy will depend on some factors like the number of epochs during the training process, the used dataset, and the extracted music features, among others.

2.4.1 Related work.

Most of the current work related to music automatic tagging uses deep learning approaches such as CNNs or DNNs. In 2016, Nam et al [29] proposed a Bag-of-features model that is composed of two stages for predicting music labels. The first stage uses a Restricted Boltzmann Machine to learn local features in an unsupervised way by extracting the feature activations for each audio clip in a convolutional way, then it summarizes this as a bag-of-features by applying a max-pooling layer, which is an operation that reduces the dimensions of a feature map by using a two-dimensional filter that selects the maximum element of that region, thus reducing the computation in the neural network and the number of parameters to learn [30]. After that, it averages the result. The second stage predicts the tags from the Bag-of-features in an unsupervised way by using a DNN. They used the MagnaTagATune dataset to train this model which contains a set of 160 tags.

Choi et al [31] proposed a Convolutional Recurrent Network (CRNN) in the same year, which was compared with some CNNs with different sizes of filter kernels. In this case, the experiments were performed using the Million Song Dataset (MSD). The results showed that the CRNN outperformed the other architectures using all the parameters, which makes this type of network a suitable option for auto-tagging tasks.

Ferraro et al [32] compared two system architectures that make use of Mel spectrograms and CNNs. The first one is composed of four convolutional layers of small 3x3 2D filters that originally were used for computer vision purposes. It is commonly known as VGG (Visual Geometry Group). The second architecture is a musically motivated CNN that contains more filters with different shapes designed to capture timbre and temporal patterns in the first layer. Then the result is fed to three more convolutional layers for determining the tags.

Both architectures use Mel spectrogram as inputs with 96 Mel bands (29 seconds length for VGG and 15 seconds length for CNN), a frame size of 512 samples, and a hop size of 256 samples for overlapping. VGG uses a sample rate of 12 kHz while CNN uses 16 kHz. They made experiments with different spectrogram resolutions and different data sets as well and they concluded that it is possible to reduce the input size and preserve the performance when executing the auto-tagging process. This allows researchers to decide whether the training and inference time, the accuracy of models, and the data storage size as well.

3. Materials and Methods.

3.1. Essentia Library.

Since this project was intended to use deep learning algorithms to determine the musical genre of a song, there was the need to use an already developed tool that allowed to perform this task.

Essentia 2.0 was the library selected because it is open source and written in C++ language, which is good for performance when executing audio algorithms. Also, it is cross-platform and can be used on Windows, Mac OS X, Linux, iOS, Android, and the web. Therefore, it can be used for analysis and processing of digital audio, and music information retrieval applications as well, since it contains a collection of algorithms that are organized in different blocks such as digital signal processing, audio input and output functionalities, statistics tools, high-level music descriptors and low-level descriptors like spectral, tonal, and rhythm amongst others. These blocks can be combined to create a more complex one like the *'Key Extractor'* algorithm shown in Figure 3-1:

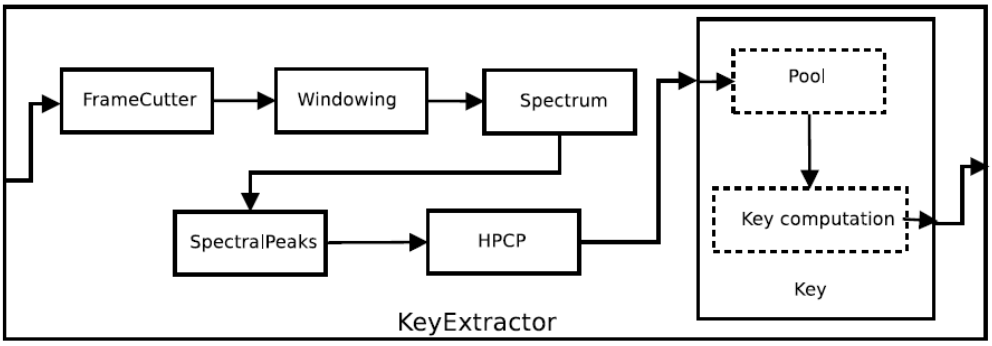

Figure 3-1. Example of a composite algorithm using Essentia blocks [17].

Another interesting feature of this library is that the algorithms can be used in standard mode and streaming mode. Whereas the standard mode allows the connection of different blocks to create a network and the user needs to manually execute each block during the processing. The streaming mode on the other hand has an internal scheduler that is in charge of passing data between the algorithms in the corresponding order. It is simpler to implement since the user only needs to connect the desired blocks of algorithms. It can be used to create real-time applications as well.

Another aspect of considering the Essentia library is the fact that it has been used in many research activities conducted at Music Technology Group (MTG) [33] and the Music Information Retrieval Evaluation Exchange (MIREX)[34][35] such as music classification, semantic auto-tagging, music similarity and recommendation, sound indexing, detection of musical instruments, cover detection, etc. Nevertheless, the most important aspect of choosing this library is that it contains algorithms that make use of TensorFlow, which is an open machine learning platform created by Google that allows one to easily create and deploy machine learning models, and it is widely used in many applications such as image generation and classification, audio recognition, music recommendations, video processing, data analysis, etc. Many examples of applications and tutorials can be found on its website at [36].

Since the objective of this thesis project was to use automatic tagging for determining the music genre, Essentia's composite algorithm called MusiCNN was used, and it is described in the next section.

This algorithm has been used in some interesting works such as [37] where the principal objective was to use Google's Magenta [38] and MuseGAN [39] generative AI to compose music. Then, the results of both AI tools were analyzed with MusiCNN to obtain the related musical tags and corroborate in this way that the generated music matched the expectations.

3.2. MusiCNN.

The MusiCNN algorithm was built using a set of pre-trained CNN which receive as input 187 Log-Mel spectrogram bands extracted from the music to deliver a 50-genre tag-gram at the output.

Figure 3-2 depicts the stages that conform the MusiCNN algorithm:

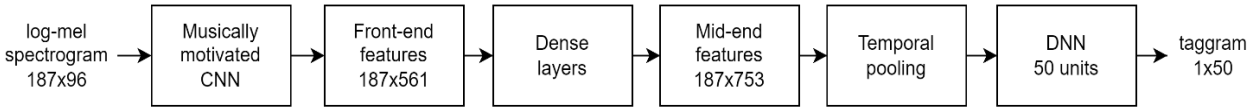

Figure 3-2. MusiCNN stages [40].

The first stage is a musically motivated CNN which is the convolutional layer that processes a log-Mel spectrogram divided into fragments of 187 frames by 96 Mel bands, which are fed at the input. The term *"musically motivated"* was defined by Pons et al [41] since this type of CNN uses filters that are focused on learning temporal and timbral features in audio. The temporal filters are the size of 1-by-n, hence the frequency dimension is set to one for finding temporal dependencies such as relevant rhythm/tempo patterns. On the other hand, the frequency filters are the size of m-by-1, which means the time dimension is set to one to learn pitch, timbre, or equalization setups.

The musically motivated CNN is conformed to two timbral CNNs and three temporal CNNs: The first timbral CNN has an input filter of size 7-by-38, and its output is a 3-dimensional matrix of 187 frames by 59 Mel bands by 204 features. The second one has an input filter of size 7-by-67, and its output is a 3-dimensional matrix of 187 frames by 30 Mel bands by 204 features.

The first temporal CNN has an input filter of size 32-by-1, the second one has a 64-by-1 input filter, and the third one a 128-by-1 input filter. Each of them delivers at the output a 3-dimensional matrix of 187 frames by 96 Mel bands by 51 features. All of the extracted features are concatenated together in a matrix of 187 frames by 561 features at the output of the musically motivated CNN. Figure 3-3 shows the musically motivated CNN in detail on the left side and the timbral/temporal filters on the right side.

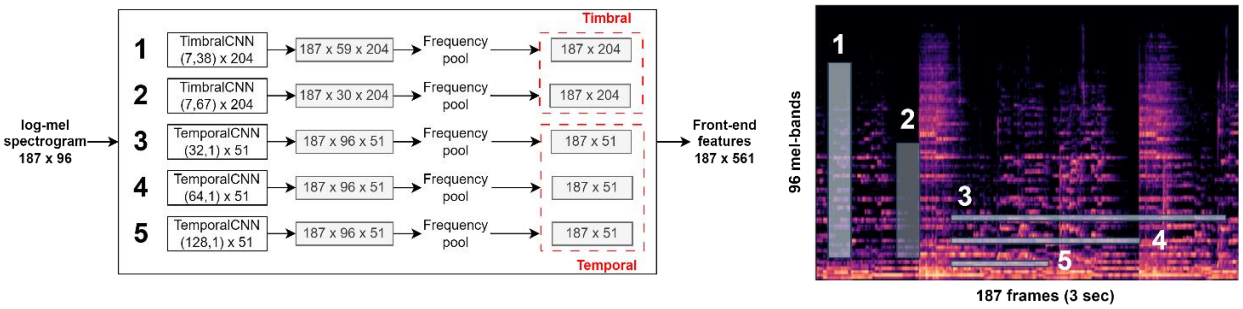

Figure 3-3. Musically motivated CNN [40].

The next stage consists of a "dense layers" CNN that extracts higher-level representations from the resultant low-level front-end features obtained in the previous stage. This type of network makes the training easier due to its residual connections. Huang et al introduced this type of CNN and a detailed description of the architecture and operation can be found at [42].

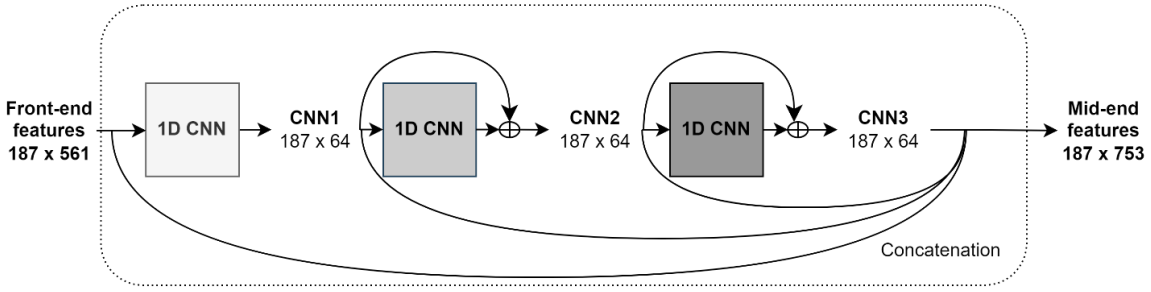

Figure 3-4. Mid-end dense layers stage [40].

Finally, the dense temporal-pooling stage is in charge of predicting the fifty-music genre tags from the mid-end features previously extracted. It adapts any feature-map length to a fixed feature-map size, which allows for variable-length inputs. Figure 3-5 shows how this stage is formed:

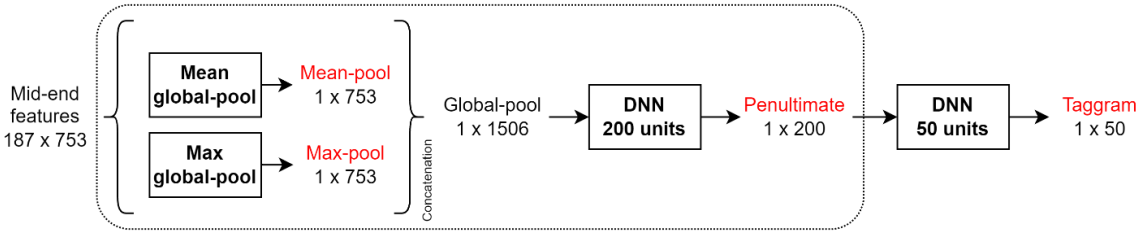

Figure 3-5. Temporal pooling stage [40].

An example of the resulting tag-gram with the fifty different genres is depicted in Figure 3-6. The lighter colors mean the song is more likely to belong to a genre, whereas darker colors mean the opposite.

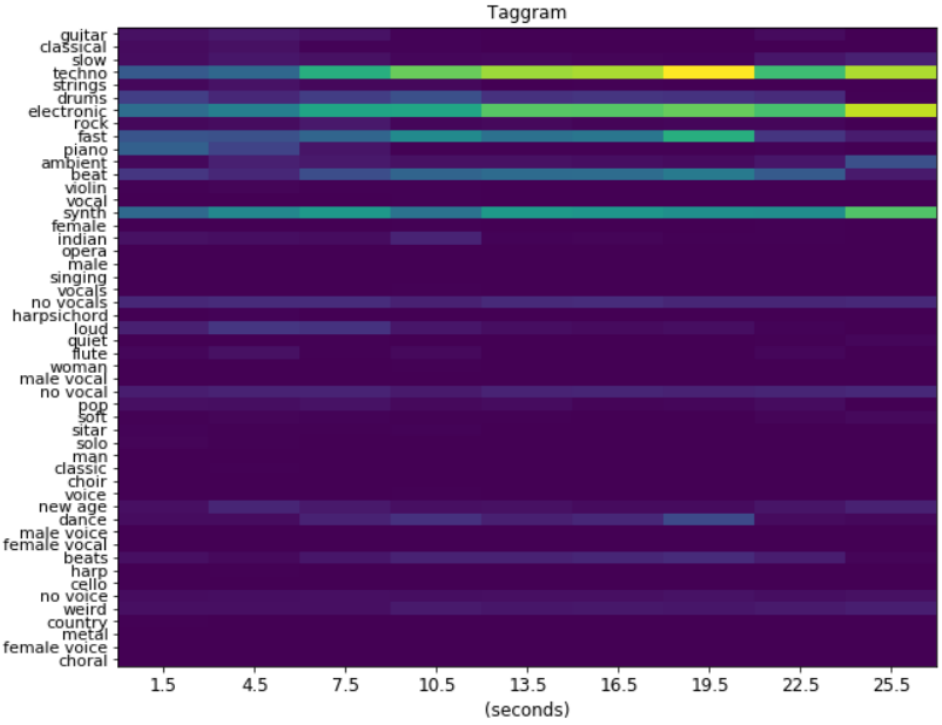

Figure 3-6. Resulting tag-gram created with Jupyter Notebook.

3.3. Music dataset.

A dataset can be defined as a collection of data that is used for certain purposes, such as data analysis or metadata. In the case of the deep learning models, these datasets are used to train the neural networks, so they can deliver the desired inference results at the output.

MusiCNN offers two different models: one that was trained using the MSD dataset and another one trained with the MagnaTagATune dataset. The MSD model was chosen for this project because it contains one million Western popular music tracks that can be used for research in the MIR field because it comes with some additional features like tempo, loudness, fade-in/fade-out timings, and MFCC data [43].

In this case, only six tags out of fifty were of interest for this project: *Pop, Rock, Electronic, Instrumental, Metal, and Jazz.* The rest of them were mapped to a similar genre. For example, Indie, Alternative, Punk, Blues, Hard Rock, and Progressive Rock were mapped to Rock genre.

3.4. Smart Audio Equalizer Design.

The next step was to integrate all the elements into one application which is depicted in Figure 3-7. This application was intended to run on Linux OS because this kind of implementation is more likely to be deployed into an embedded system such as car infotainment. The chosen OS was Ubuntu 20.2.

It was necessary to create an application with a GUI to be able to handle the gain values and change it with some knobs or sliders. At first, it was considered to implement a graphic equalizer only. However, it was created a parametric equalizer as well to compare the quality of the sound between them.

Matlab App Designer was chosen for this purpose because it allows us to easily create an interface by adding different elements like a container, knobs, drop-down boxes, buttons, and sliders amongst others. Also, Matlab has very useful tools for designing the necessary filters that handle the different frequency bands and a tool for executing Essentia's library C++ code that is in charge of creating the tag gram and determining the music genre of each song.

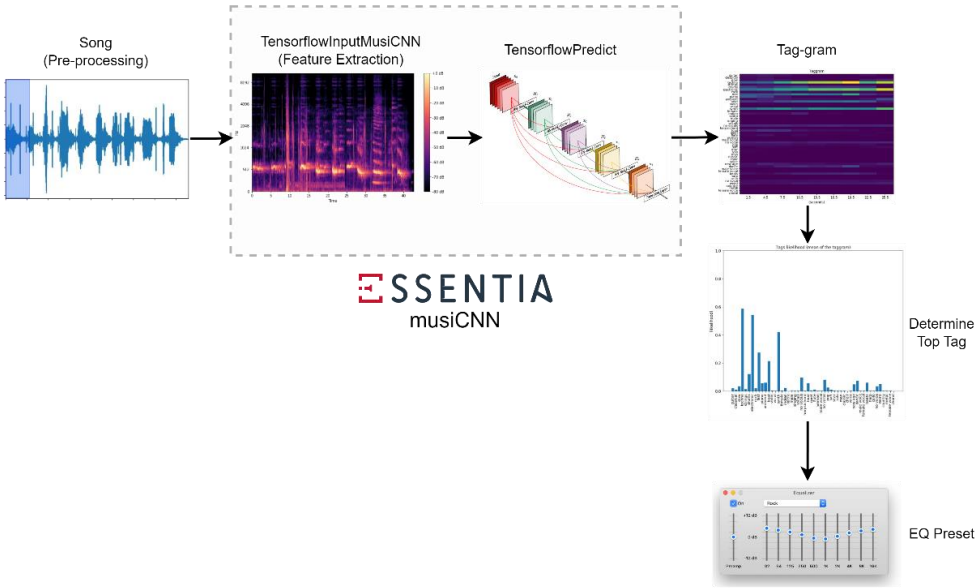

Figure 3-7. Smart Audio Equalizer application architecture.

3.4.1 Audio Pre-processing.

Before starting with the music feature extraction, the song has to be pre-processed by downsampling the signal data to 16 kHz. This is done because the used model was trained using song fragments that were sampled at this frequency for faster processing.

Then, a thirty-second segment of the song is extracted to reduce processing time and resource consumption. Also, the desired behavior is to determine the music genre when the song is loaded and not for every processed audio frame during the playback. This time was enough to determine the music genre according to the results shown in Chapter 4.

3.4.2 Music Genre Tag Determination.

The next step is to feed the audio input into the MusiCNN algorithm, which internally divides the audio into frames with a length of 512 samples, then each frame is processed by the 'TensorflorInputMusiCNN' algorithm, that is in charge of extracting the music features (Mel bands) that are fed into the algorithm 'TensorflowPredict' later on for obtaining the resulting tag-gram as it was explained in detail on section 3.2.

Then, the tag with the maximum likelihood value of the resulting predictions is determined by using the max element standard C++ function to get the corresponding index value of the music genre amongst the 50 different tags, which are defined by the Million Song dataset.

3.4.3 Audio Equalization Preset Mapping.

Finally, the obtained tag is compared against the existent equalization presets to apply the corresponding equalization values to the song before it starts to play.

Table 3-1 shows the mapping of the dataset tags to the six available audio equalizer presets from the Matlab application. In case the determined tag does not correspond to a music genre, the equalizer preset that will be applied is "flat".

Preset	Mapped music genre
rock	blues
	alternative rock
	indie rock
	classic rock
	progressive rock
	punk
	indie
pop	indie pop
	hip hop
	rnb
	soul
	alternative
electronic	dance
	chillout
	electronica
	party
	electro
	house
	chill
	chillout
instrumental	ambient
	acoustic
	guitar
jazz	experimental
	funk
metal	heavy metal
	hard rock

Table 3-1. Equalizer presets mapping.

3.4.4 Filter Design.

The filters used in the first version of the graphic equalizer were designed with Matlab's Filter Designer tool, which allows the configuration of the response type, design method, filter order, sampling frequency, and stop and pass frequencies, among other design parameters. Figure 3-8 shows the parameters used for designing the low-pass filter used in the lowest band of the equalizer. In this case, the IIR filter type was chosen since this type is more suitable for audio applications.

Figure 3-9 depicts the window with the filter coefficients where they can be exported in text format for implementing the filter in the Matlab application code.

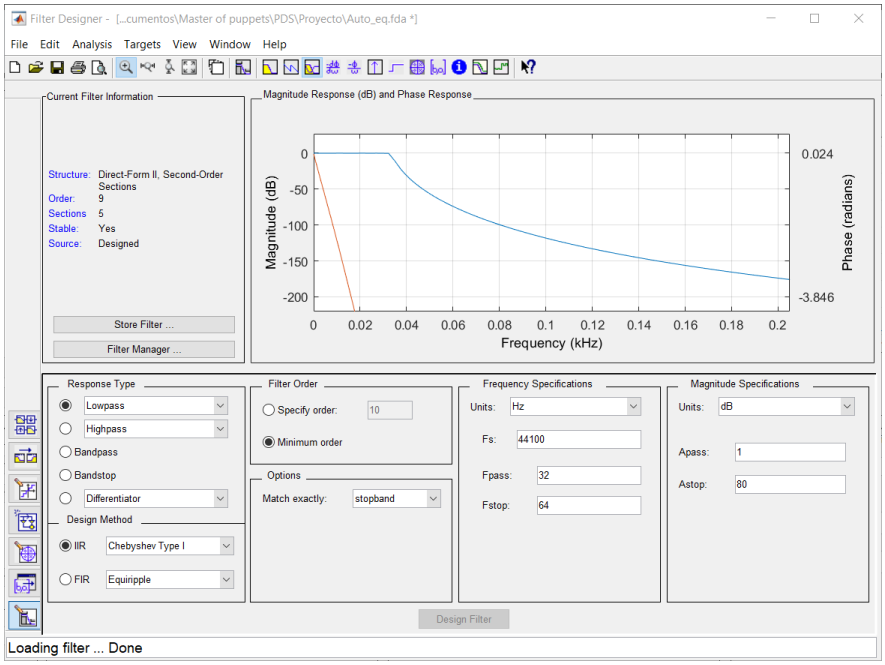

Figure 3-8. Low-pass filter created with Filter Designer application.

3. MATERIALS AND METHODS

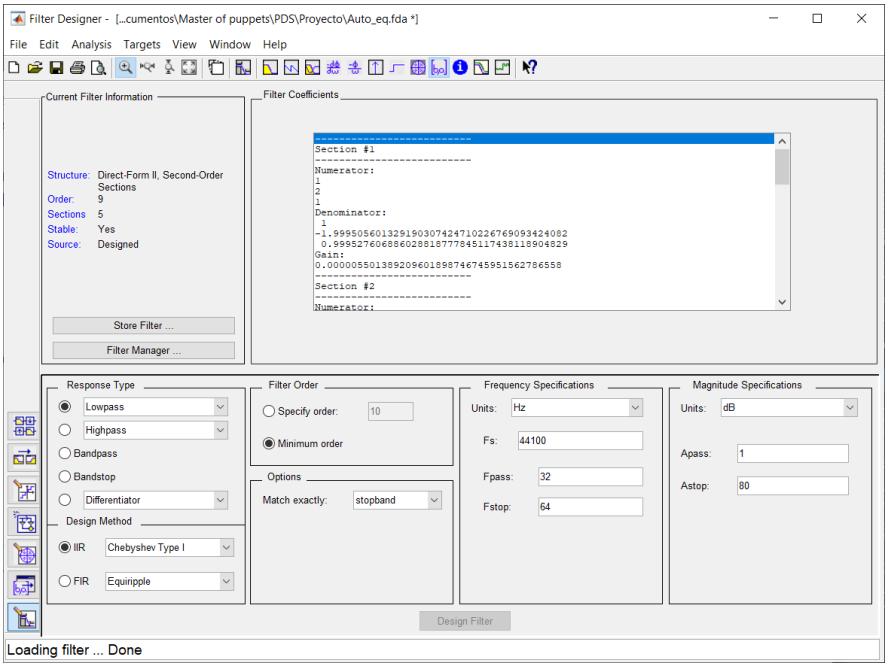

Figure 3-9. Low-pass filter coefficients.

Figure 3-10 and Figure 3-11 show the design of one of the bandpass filters that are used for each band that is part of the graphic equalizer.

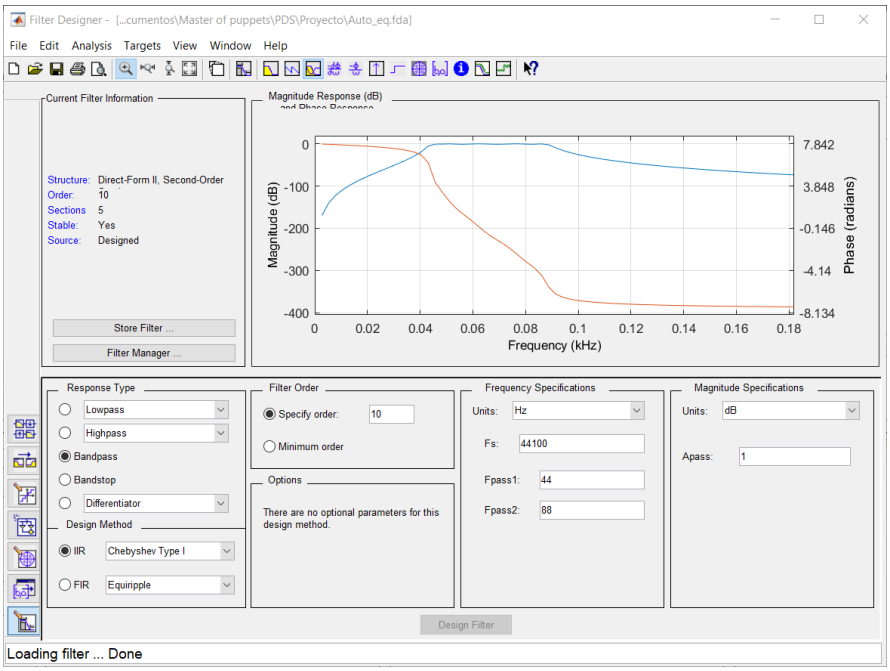

Figure 3-10. Bandpass filter created with Filter Designer application.

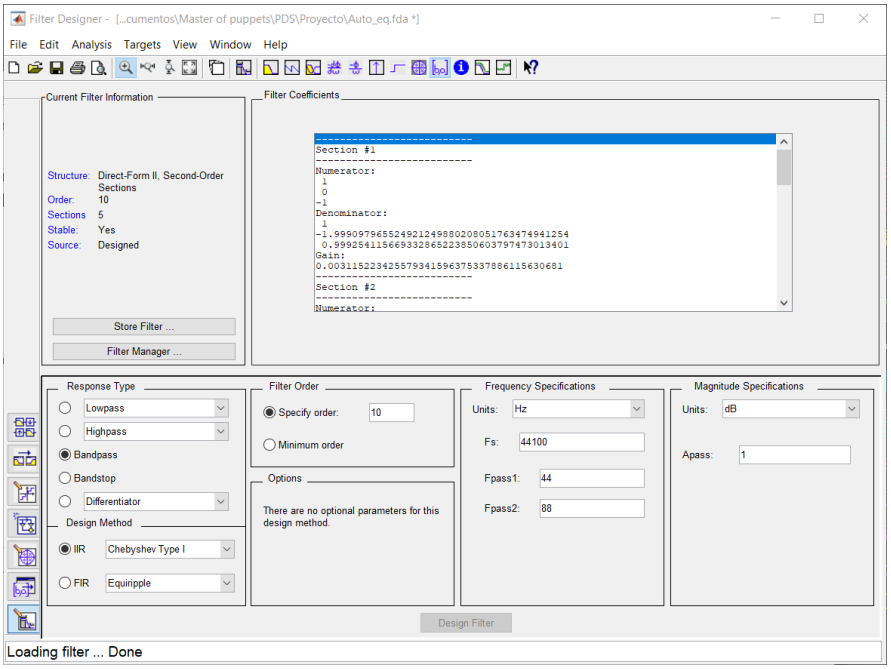

Figure 3-11. Bandpass filter coefficients.

Finally, Figure 3-12 and Figure 3-13 show the design of the high pass filter used in the highest band of the graphic equalizer.

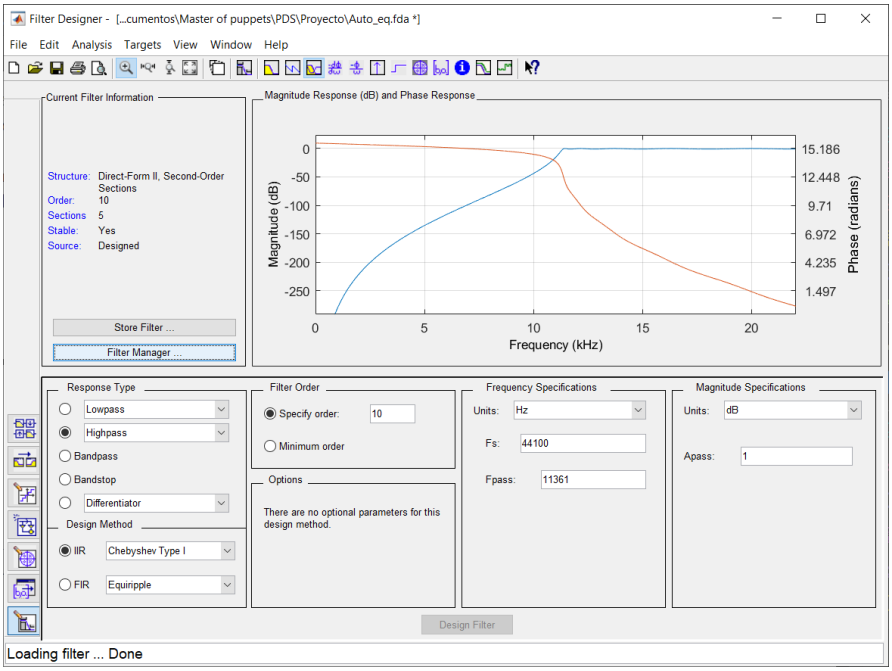

Figure 3-12. High-pass filter created with Filter Designer application.

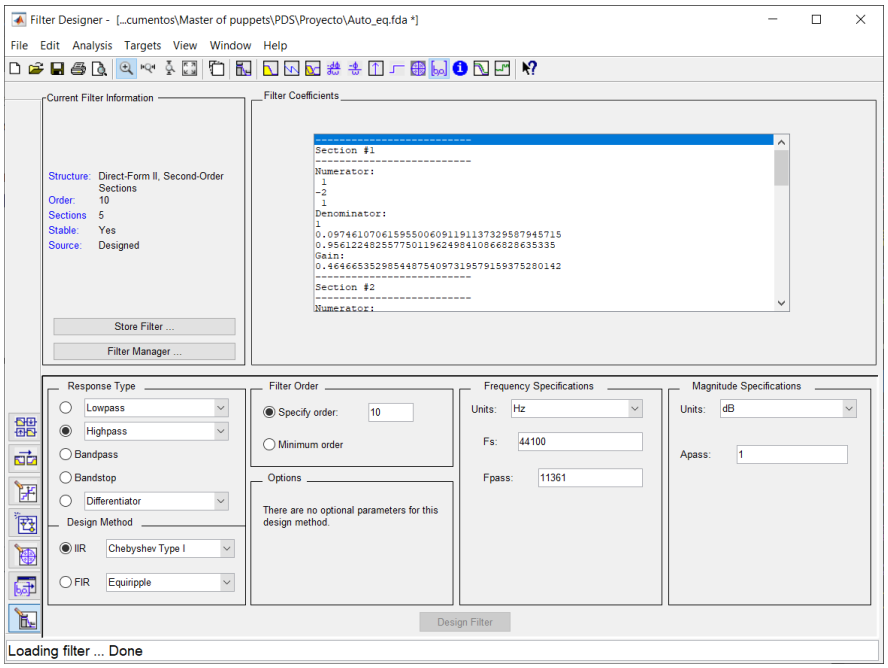

Figure 3-13. High-pass filter coefficients.

However, the quality of these filters was not good enough, because the audio sounded choppy, distorted, and of low quality. For this reason, it was decided to change the way of designing the filters to the use of *'designShelvingEQ'* and '*designParamEQ'* functions in Matlab.

The function 'designShelvingEQ' allows the design of shelving filters easily by specifying parameters like gain, slope, cutoff frequency, and the type of filter [44].

Figure 3-14 and Figure 3-15 show the resulting low-shelf and high-shelf filters used for the lowest and highest equalizer bands.

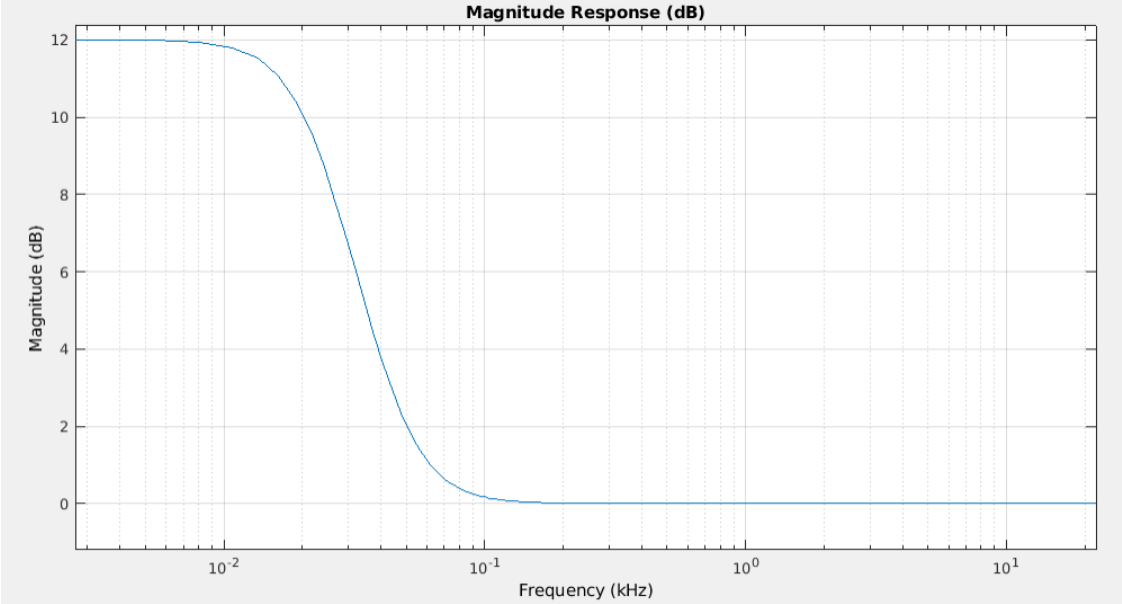

Figure 3-14. Low-shelf filter with a cutoff frequency of 32 Hz, slope of 1, and 12dB gain.

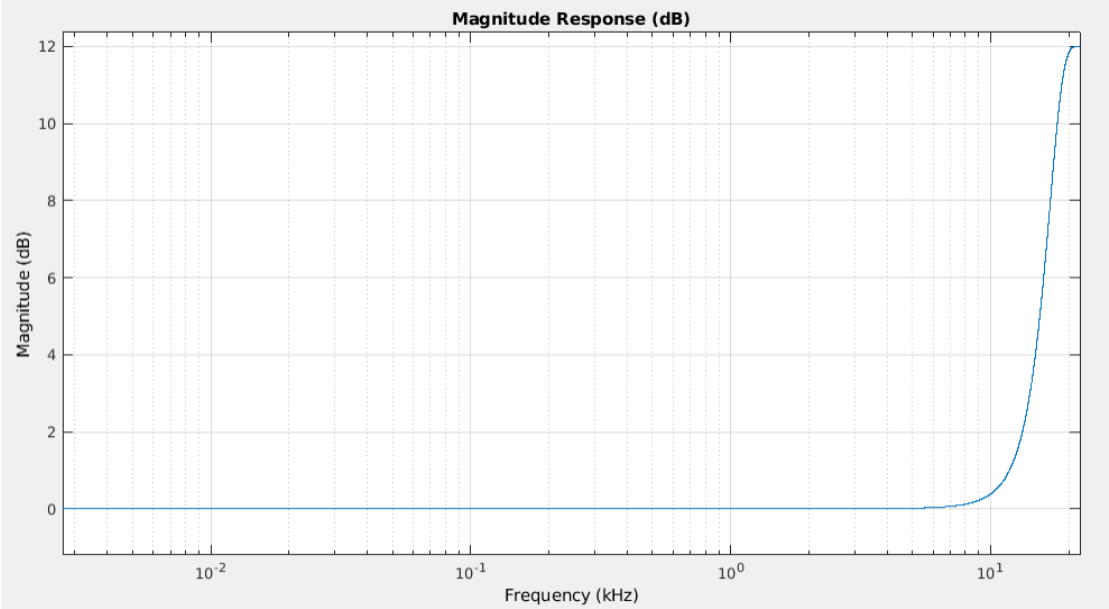

Figure 3-15. High-shelf filter with a cutoff frequency of 16 kHz, slope of 1, and 12dB gain.

In the case of the bandpass filters, the function *'designParamEQ'* [45] allowed to design a parametric filter for using them in the graphic audio equalizer. The Q factor, order, center frequency, and bandwidth were fixed values because a graphic equalizer only modifies the value of the gain at a specific center frequency. This time, the audio quality was improved significantly, and the distortion was no longer heard.

The chosen fixed parameters for the filters were the following:

- Q factor = 1
- Filter order = 2
- Bandwidth = Center frequency / Q factor

Figure 3-16 shows the plot of the designed filter bank used in the graphic equalizer using the '*fvtool'* function and a gain value of 12 dB:

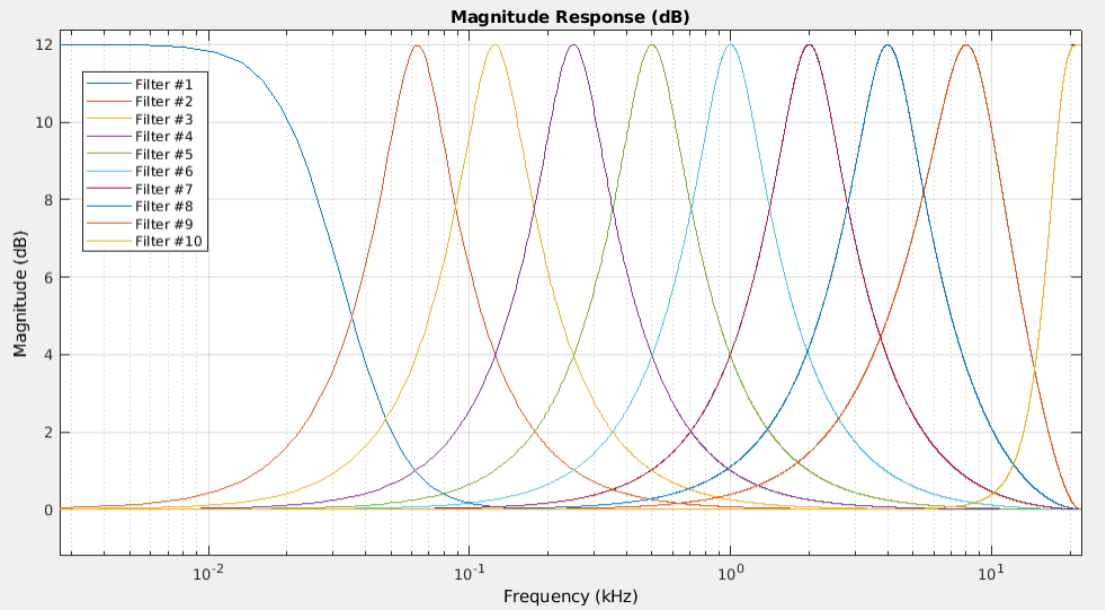

Figure 3-16. Bandpass filter bank designed in Matlab.

Figure 3-17 shows the overall magnitude response of the graphic equalizer.

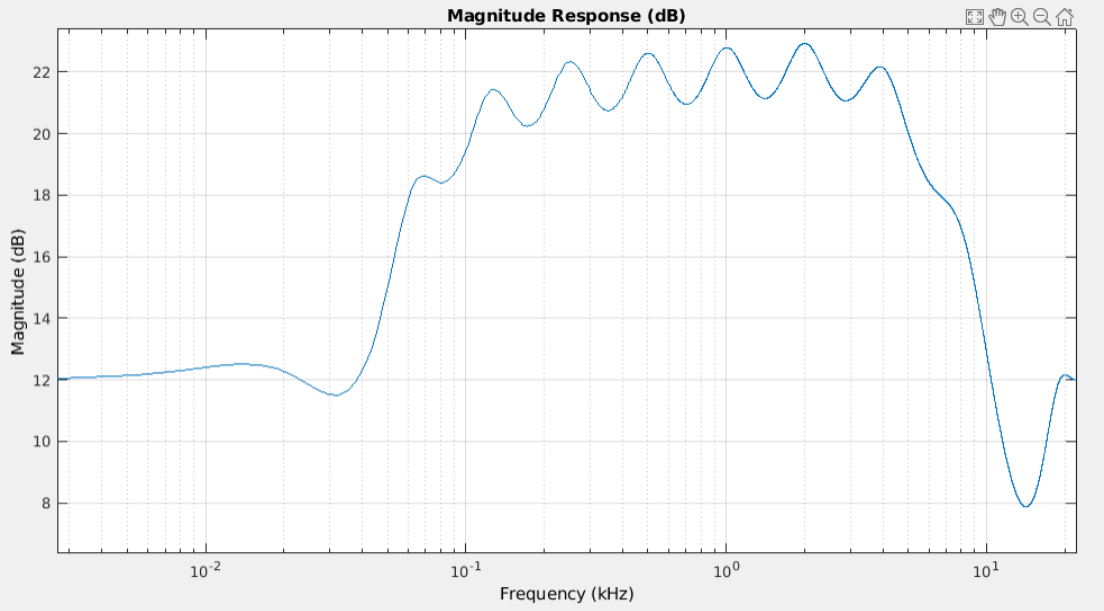

Figure 3-17. Magnitude response of the cascaded graphic equalizer.

On the other hand, the design of the parametric equalizer was easier thanks to the use of the functions mentioned before. However, this time the parameters to consider were the Q factor, cutoff frequency, order, and gain, which value is variable in the implementation. Figure 3-18 shows the magnitude response of every filter using different parameter values.

Figure 3-19 depicts the magnitude response of the cascaded filter with the same values.

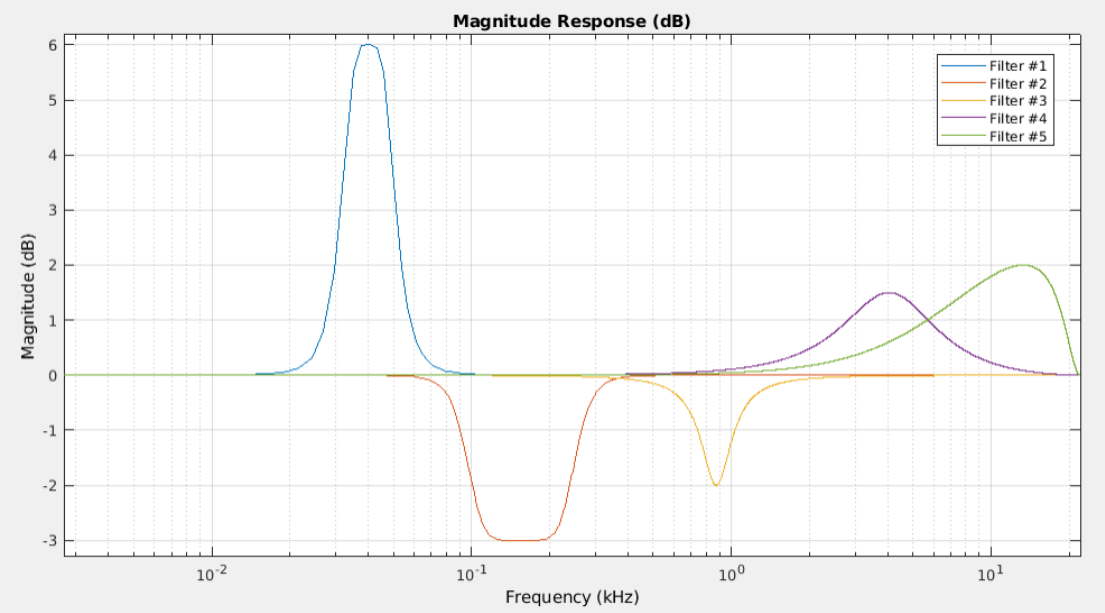

Figure 3-18. Magnitude response of the parametric filters.

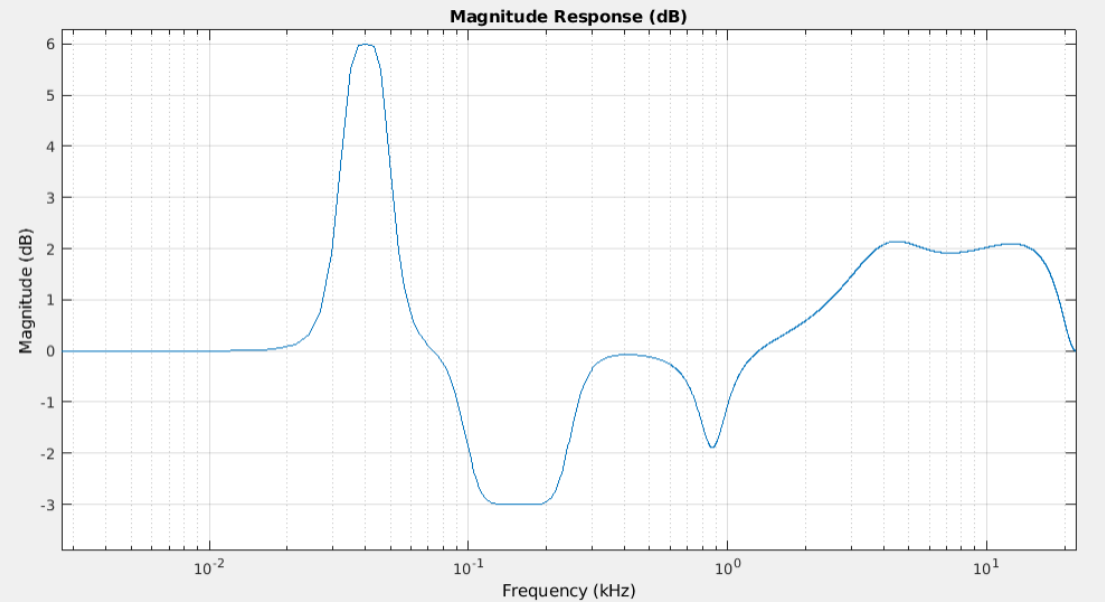

Figure 3-19. Magnitude response of the parametric equalizer.

The next chapter describes the implementation process for both equalizers and the integration of automatic music genre determination.

4. Implementation and Results.

4.1. Graphic equalizer implementation.

The first implementation of the Matlab application was the graphic equalizer since the design is simpler than the parametric one because it only has 10 sliders to control the amplitude in decibels of each frequency band. It has basic control buttons for playing, stopping, and pausing the loaded song which can be opened by specifying its location.

It was decided to add extra features like a plot with the current equalization curve, so the user has a better idea of the improvements made to the audio. A volume knob, an Auto Mode switch, and an equalizer on/off switch were implemented as well. Also, a drop-down list was added for manually selecting the desired equalizer preset in case the user wants to test the other available presets and hear the changes in the audio. Figure 4-1 shows the "Graphic EQ" tab from the Smart Equalizer Matlab application:

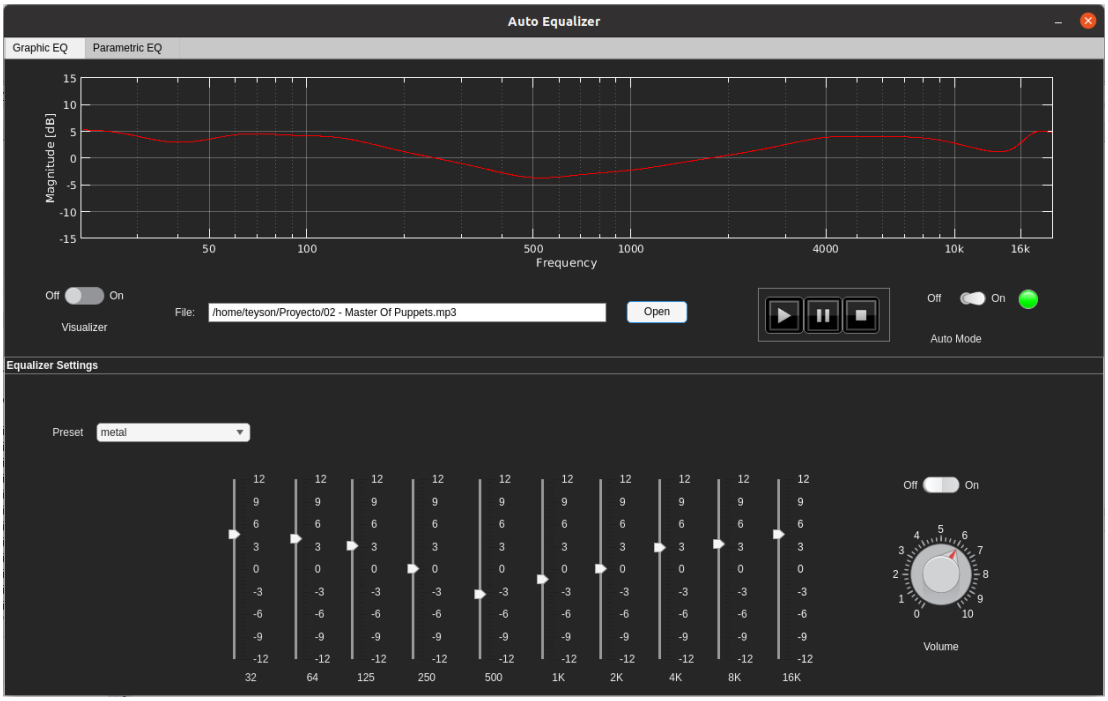

Figure 4-1. Graphic equalizer GUI.

4.2. Parametric equalizer implementation.

The parametric equalizer GUI is similar to the graphic one and it was implemented with 5 groups of knobs that control the center frequency, order, Q factor and gain of the following frequency range of the parametric filters: Low, low-mid, mids, high-mids and high.

In this case, the user can freely move the value of each parameter like the center frequency that is identified with a point in the plot located at the top. Every filter is plotted individually and has a unique color, so the user can identify them easily. If another parameter such as order or gain is changed, the plot is automatically updated. This allows the user to adjust the audio quality more precisely.

There is a drop-down menu to select manually the desired preset in case the “Auto Mode” is deactivated. Figure 4-2 shows the “Parametric EQ” tab from the Smart Equalizer Matlab application:

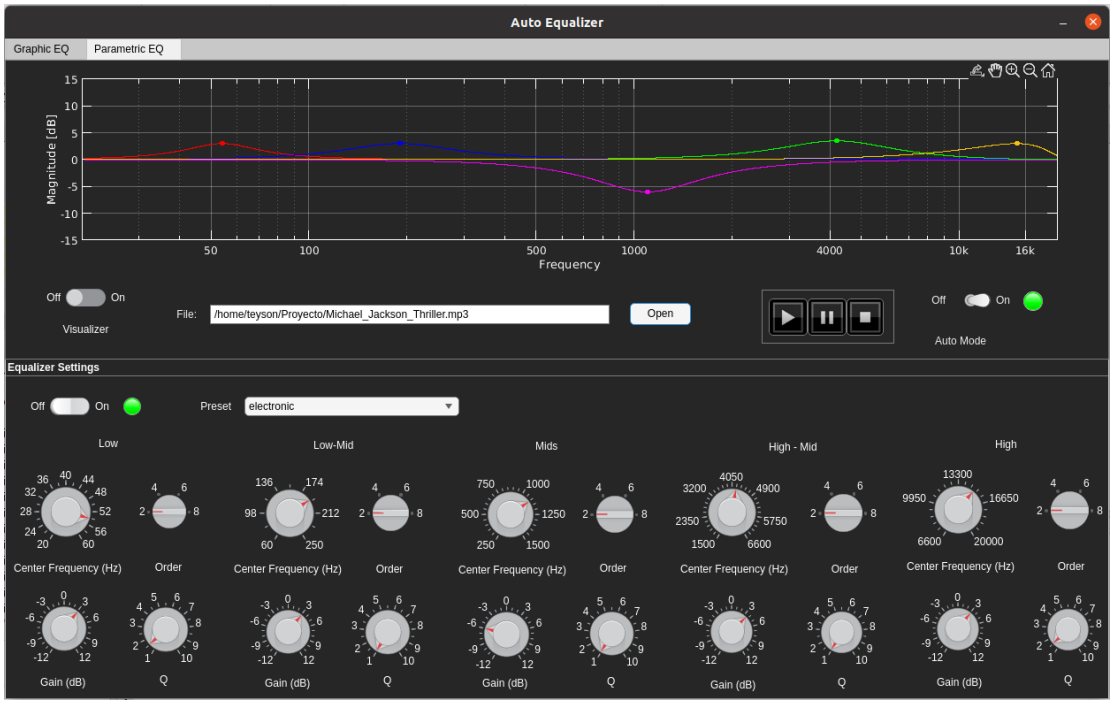

Figure 4-2. Parametric equalizer GUI.

4.3. Execution of MusiCNN in Matlab.

Before the Essentia library could be executed, it was necessary to compile the code by executing a Python script included in the root directory. This script can be configured according to the Essentia features that will be used. In this case, the TensorFlow option had to be included, so the artificial intelligence algorithms could be compiled.

After that, the library was installed in the Linux environment, and another Python script which is located in the "third party" folder was executed for installing TensorFlow support as well. The details of the installation process can be found on Essentia's website [45]. Once the library was set up, a C++ program was created for executing the deep learning algorithm that is in charge of determining the music genre tags and calling it from the Matlab application as well.

To compile and create the MEX function from the C++ mentioned before, the command shown in Figure 4-3 has to be entered in Matlab's command window:

```
>> mex mexMusiCNN_v3.cpp -lessentia -lfftw3f -lsamplerate -lavcodec -lavformat -lavutil  -lyaml
Building with 'g++'.
MEX completed successfully.
```

Figure 4-3. Command to generate MEX function.

Where the values with the prefix -l are the required libraries to build the function.

Once it was created, it was possible to test the Essentia algorithm by specifying the path of an mp3 or WAV file as an input parameter. Figure 4-4 shows the command window with the resulting output messages:

4. IMPLEMENTATION AND RESULTS

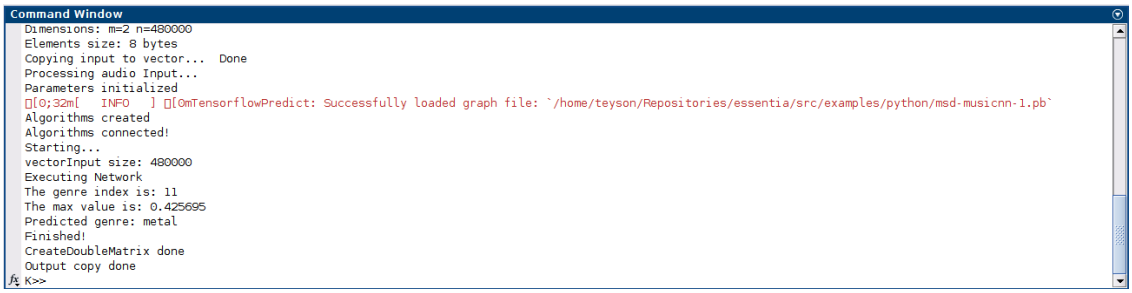

Figure 4-4. Execution results of the MusiCNN MEX function in Matlab.

Figure 4-5 shows the resulting elements of the workspace after executing a test script for processing three different songs. The determined music genres were stored in the "tag" variables:

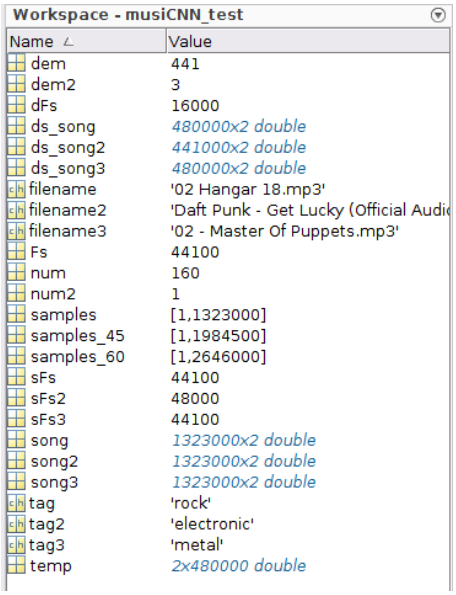

Figure 4-5. Tag results displayed on Matlab workspace.

Figure 4-6 shows the internal stages of the MEX function which allows us to have a high-level idea of how it works:

Figure 4-6. Internal stages of MEX function.

The first stage of the MEX function consists of getting the audio vector frame from a 30-second fragment of music. This is done by copying the audio input samples into an Essentia's input vector. Once it is done, the algorithm parameters are initialized and then the inference is computed.

Then, the output of the deep learning algorithm is a pool with the prediction results which are aggregated in the next step. After that, the top value of the aggregated results is determined for getting the corresponding music genre tag.

4.4. Integration of automatic genre classification into Matlab application.

When the load audio file button is pushed in the smart equalizer application, the pre-processing mentioned in section 3.4.1 is done to match the parameters of the data that was used to train the CNN. The decision to use a 30-second audio input fragment was made based on the results shown in section 4.5.2. This time was enough to determine the music genre without sacrificing performance and accuracy in the inference results.

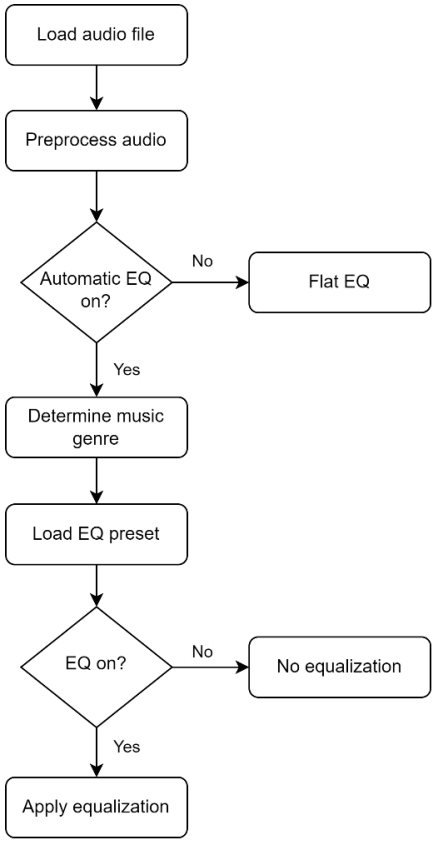

Figure 4-7. Smart equalization flow diagram.

Once the pre-processing is done, the smart equalizer application executes the generated MEX function *'mexMusiCNN_v3'* only if the automatic EQ switch is turned on. Otherwise, it will not apply any preset to the equalizer.

After the MEX function finishes the execution, the resulting tag is compared against the list of the available presets of the equalizer. Then it applies the corresponding one automatically. However, if the "EQ On" button is active, then the equalization process will be executed. If not, the audio will be played as it is. Figure 4-7 depicts the flow diagram of the process.

4.5. Genre classification results.

4.5.1 Jupyter notebook tests.

To test MusiCNN and understand how it works, there is an included tutorial in Essentia repository that allows running a Jupyter Notebook where an example Python code is executed to visualize the auto-tagging process in real-time. This helped to create the C++ code that was included in the MEX function.

Figure 4-8 shows the plot of the Mel spectrogram and the tag-gram of the first seconds of the song "Beat It" by Michael Jackson, which is considered to belong to the Dance Rock genre. However, this part of the song was mostly tagged as electronic because the Rock genre elements such as electric guitars are not heard until later and the rhythm is similar to an Electronic genre song.

Figure 4-9 depicts the Mel spectrogram and tag-gram plots of the middle of the song. In this case, it clearly shows that the song was tagged as a Rock genre. Nevertheless, this is an example of how a song sometimes can be tagged with a different genre because it contains parts that can belong to other genres.

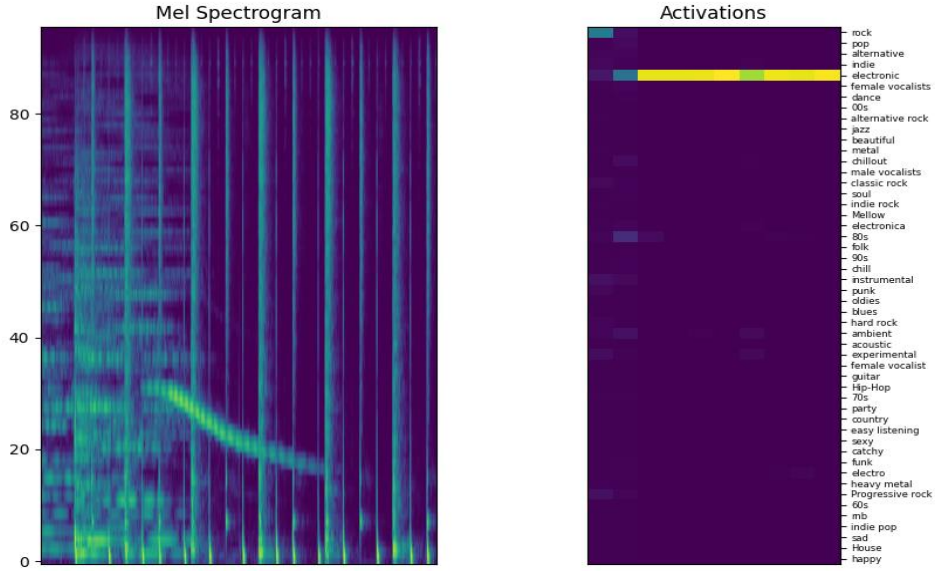

Figure 4-8. Mel spectrogram and tag-gram of the introduction part of the song "Beat It" by Michael Jackson.

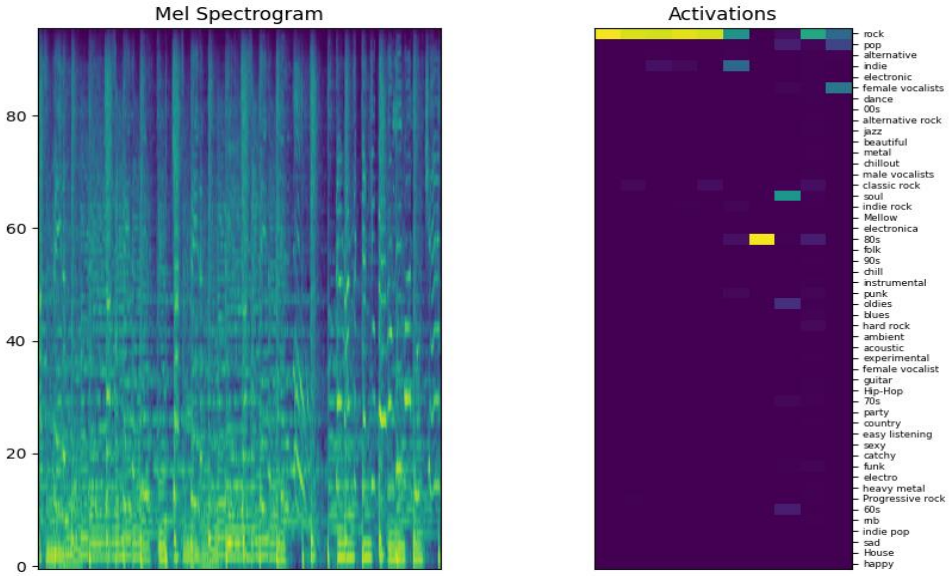

Figure 4-9. Mel spectrogram and tag-gram of the middle part of the song "Beat It" by Michael Jackson.

4.5.2 Full length songs.

Once the MEX function was compiled, some experiments were performed using full-length songs to check the MusiCNN algorithm performance and tagging precision. Since these songs contain more sample frames to process, the performance was not very good because it took about one minute to calculate the prediction tags. Some examples are shown in the following figures.

Figure 4-10 shows the resulting tag prediction of a Rock song that seems to have some elements from alternative Rock and Indie:

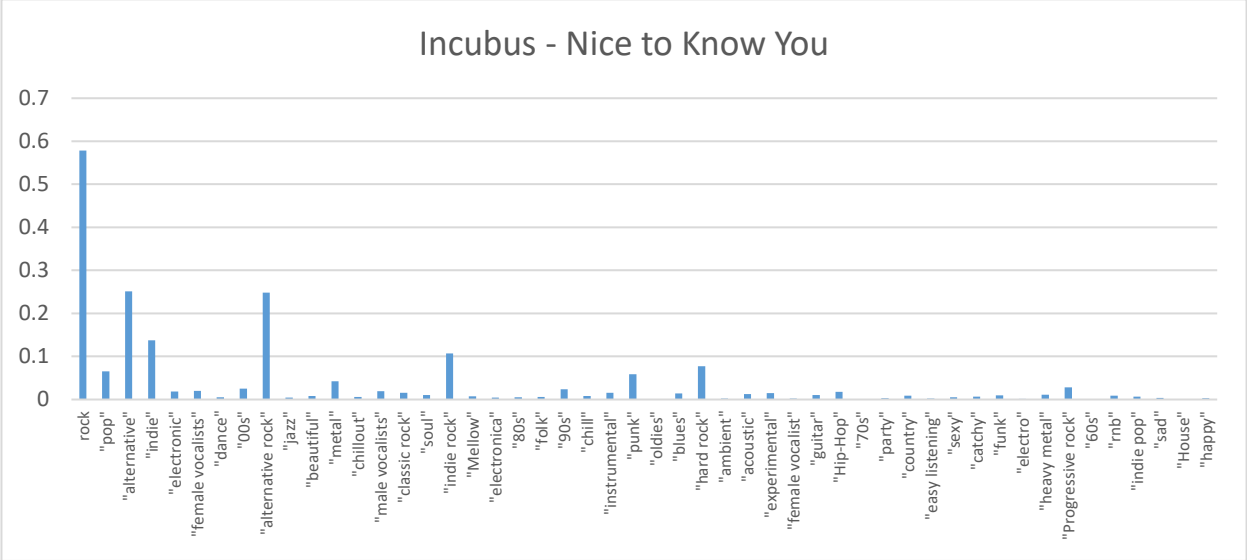

Figure 4-10. Rock genre prediction result using a full-length song.

Figure 4-31 shows the result of an instrumental full-length song, and it seems that MusiCNN identified some similarities with the Jazz and Folk genres. However, the probability values are low. The correct tag was determined despite this.

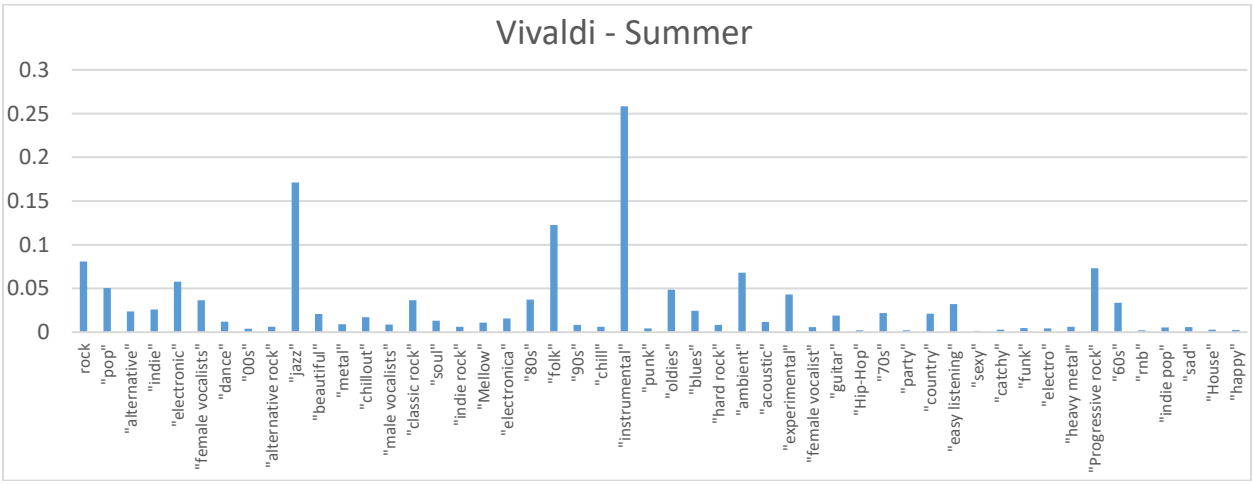

Figure 4-31. Instrumental genre prediction results using a full-length song.

Figure 4-12 depicts an Electronic music genre song that seems to have similarities with Ambient and Experimental genres. This could be because different types of Ambient music such

as Ambient House, Ambient Techno, and Ambient Trance use electronic instruments such as synthesizers to emulate acoustic instruments like piano or strings [47].

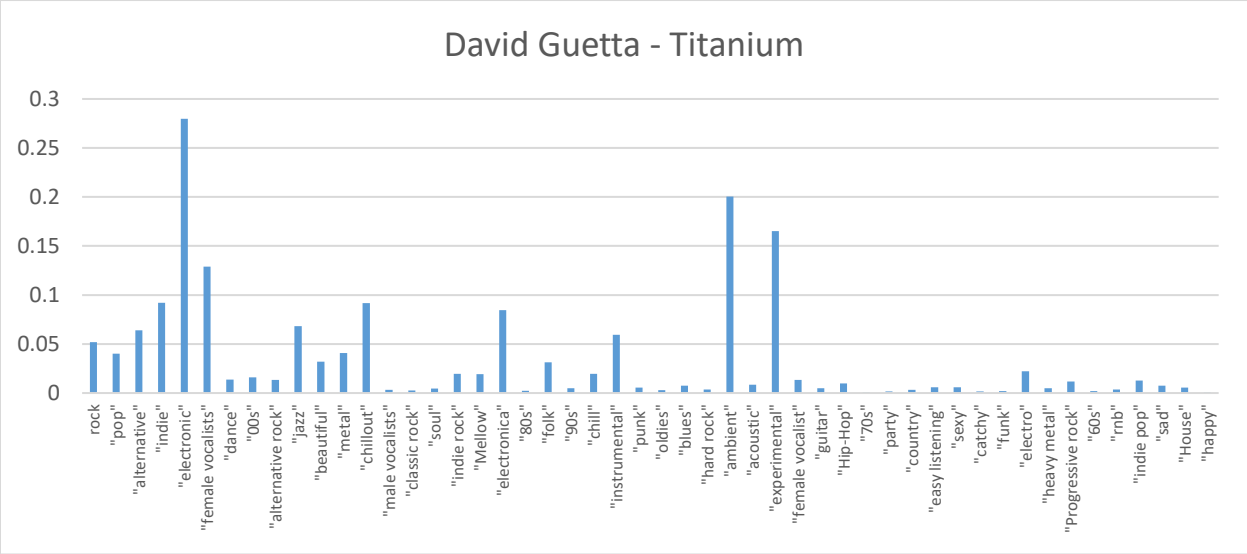

Figure 4-42. Electronic genre prediction results using a full-length song.

A special case was tested with a Progressive Rock song which length is 20 minutes long (it took several minutes to process). Despite the fact there is a tag for Progressive Rock, this song was tagged as Rock, followed by Classic Rock and Hard Rock. This type of song can be difficult to identify its music genre tag since they tend to include elements from other genres and several changes in rhythm or tempo as well. Figure 4-53 shows the results:

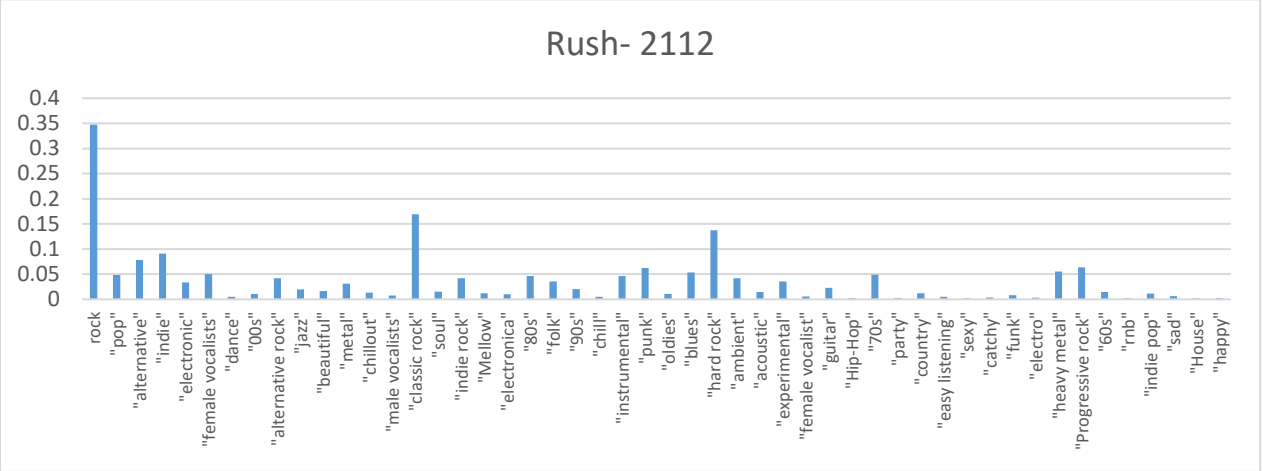

Figure 4-53. Progressive Rock genre prediction results using a 20-minute length song.

4.5.3 Fixed length songs.

On the other hand, some tests were performed with a small fragment of the song: The first experiments were done with a 30-second fragment, because the song clips of MSD are the same length, and the neural network was trained with those audio clips. It was observed that the probability of the song belonging to a certain music genre decreased in comparison with the full-length song. Nevertheless, the top tag result tended to be accurate. Figure 4-64 shows the tagging results of a 30-second length metal song:

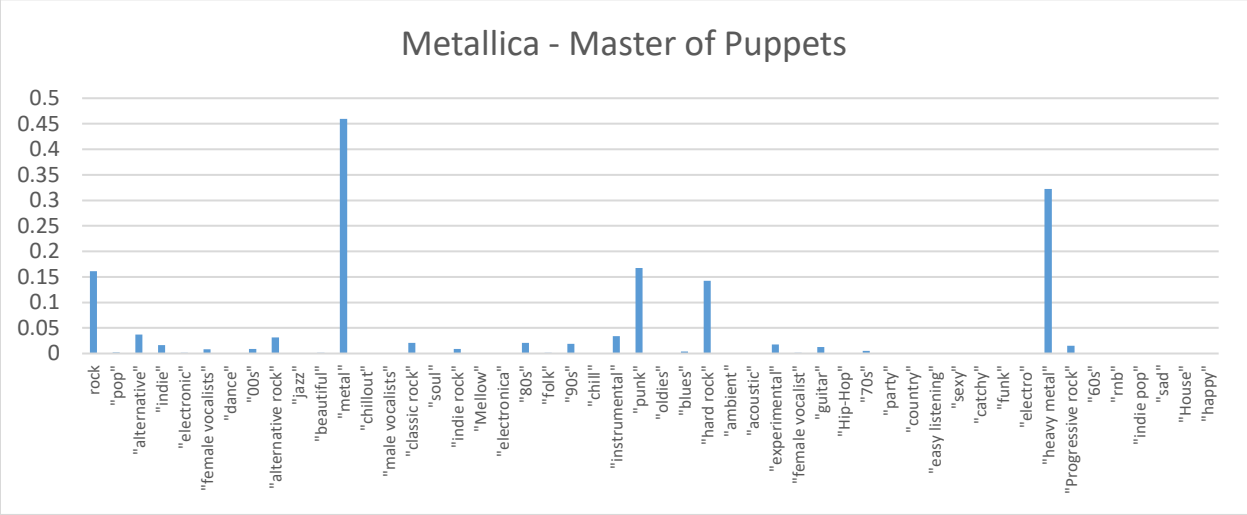

Figure 4-64. Metal genre prediction results using a 30-second fragment of the song.

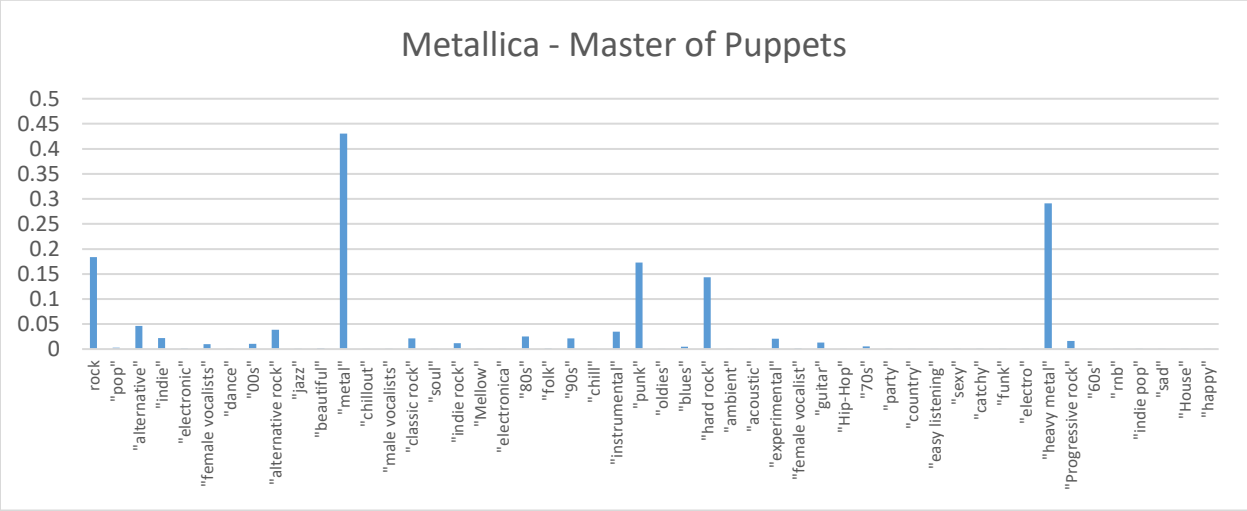

Figure 4-75. Metal genre prediction results using a 20-second fragment of the song.

Figure 4-7 and Figure 4-86 show a decrease in the probability, but the tagging results still look accurate. The top determined tag was Metal followed by Punk, Heavy Metal, and Rock, which are similar genres that use distorted guitars and heavy riffs.

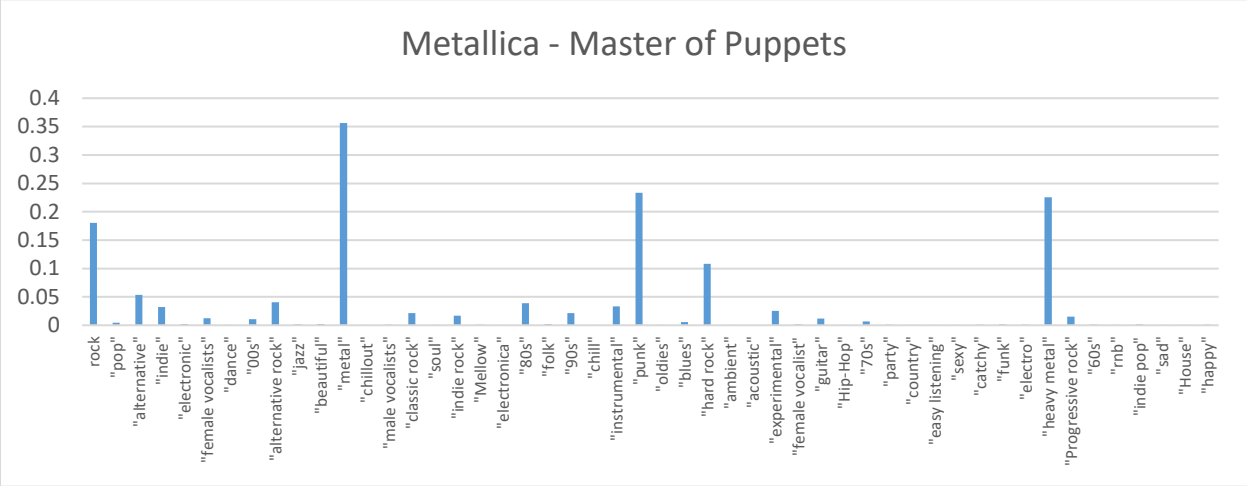

Figure 4-86. Metal genre prediction results using a 10-second fragment of the song.

Figure 4-97 depicts the result of a Pop song that seems to have similarities with Electronic, RnB, and Soul. The auto-tagging neural network also was able to identify that the artist is a female vocalist, and it has some elements like the 80s era.

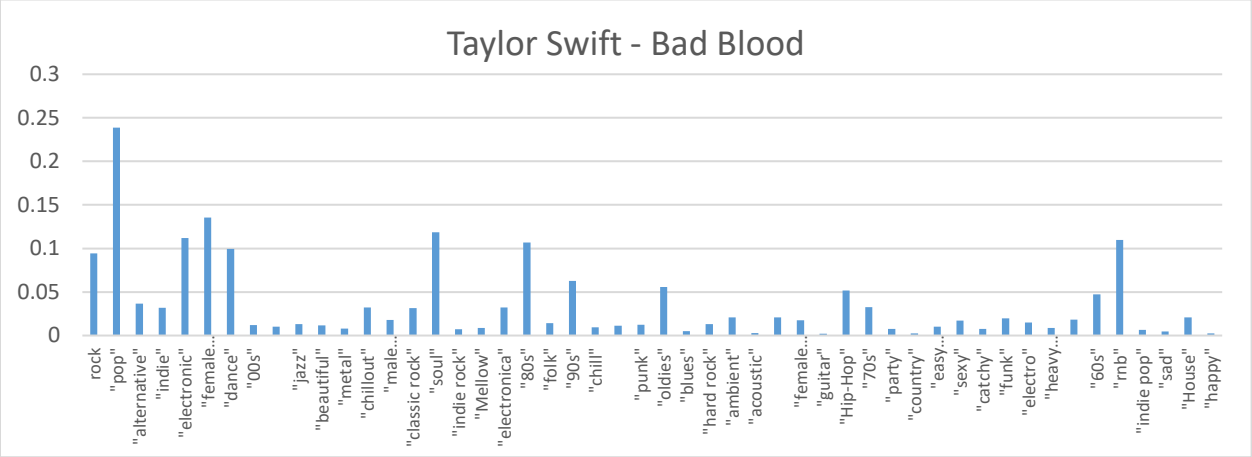

Figure 4-97. Pop genre prediction results using a 30-second fragment of the song.

Figure 4-108 depicts the result of an electronic song that has nuances of Indie, Pop, and even Rock genres because there is the presence of electric guitars during the whole song and the vocals are like the ones found in a Pop song:

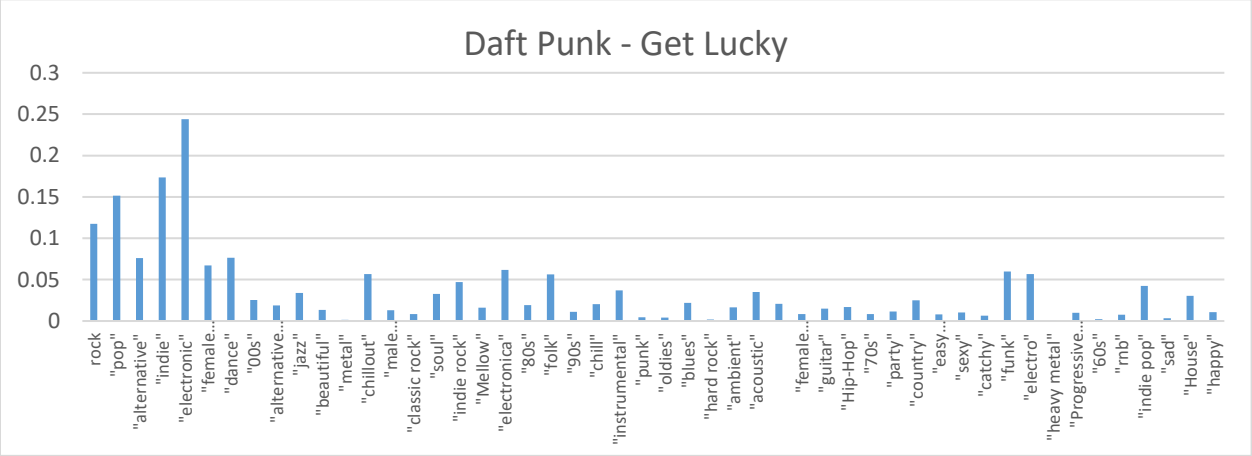

Figure 4-108. Electronic genre prediction results using a 30-second fragment of the song.

In the case of the Rock genre, the song seemed to have some nuances of Alternative, Indie, and Punk genres due to the rhythm and the distorted electric guitars. Figure 4-119 shows the results of this genre:

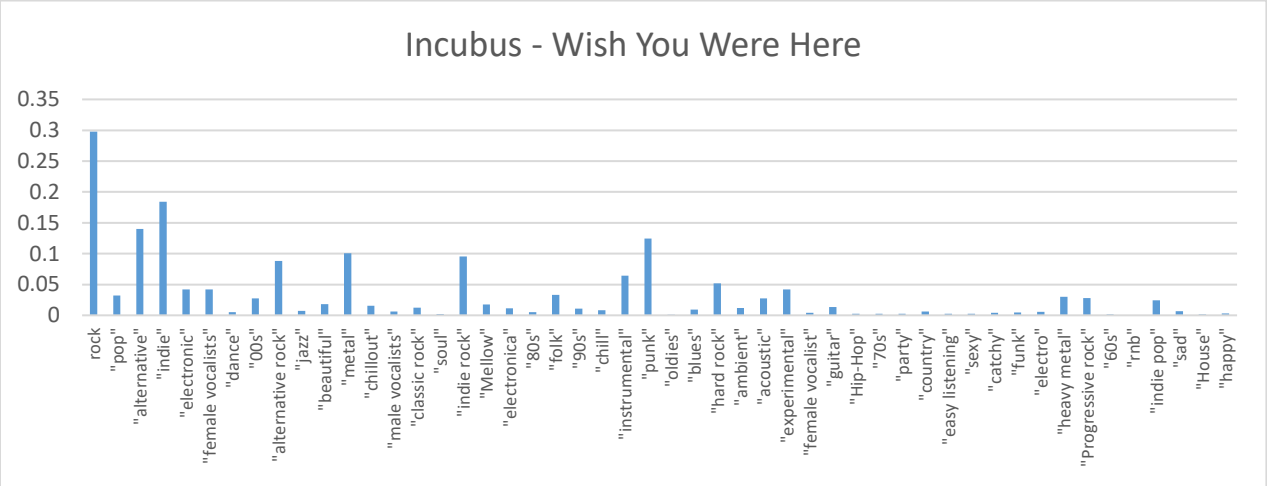

Figure 4-119. Rock genre prediction results using a 30-second fragment of the song.

The experiment with the instrumental genre was interesting because the values from the top 3 determined tags for this song were too close. MusiCNN identified high similarities of blues

and folk music in this song. Perhaps this result was determined due to some elements of the song such as classical guitar. Figure 4-20 shows this result:

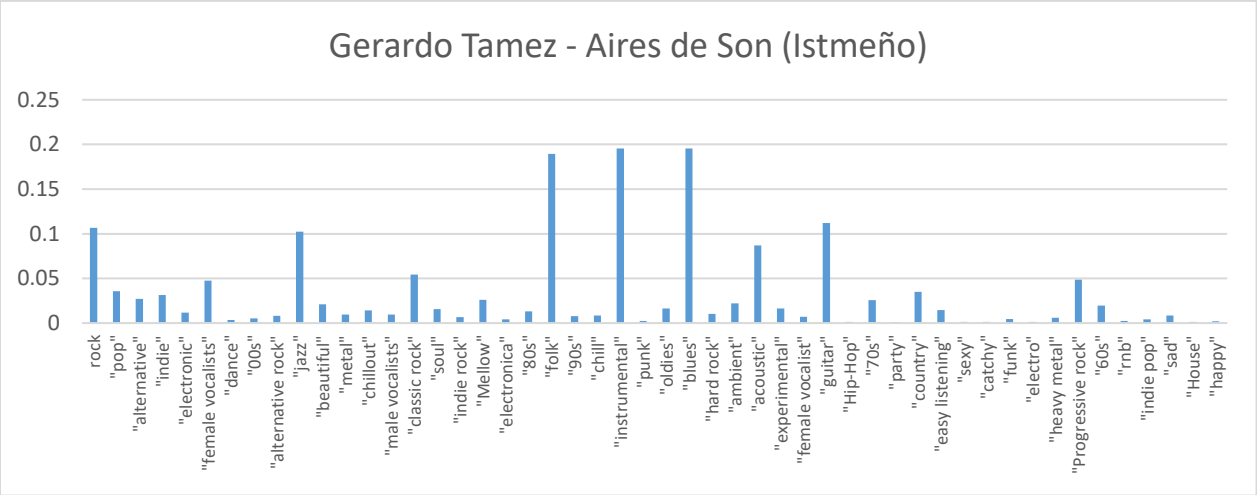

Figure 4-20. Instrumental genre prediction results using a 30-second fragment of the song.

In the case of the jazz song, MusiCNN precisely determined the tag with a high probability of belonging to that genre, and the rest of the genres were below 0.1. This was the most accurate of the automatic tagging experiments. Figure 4-121 shows the results:

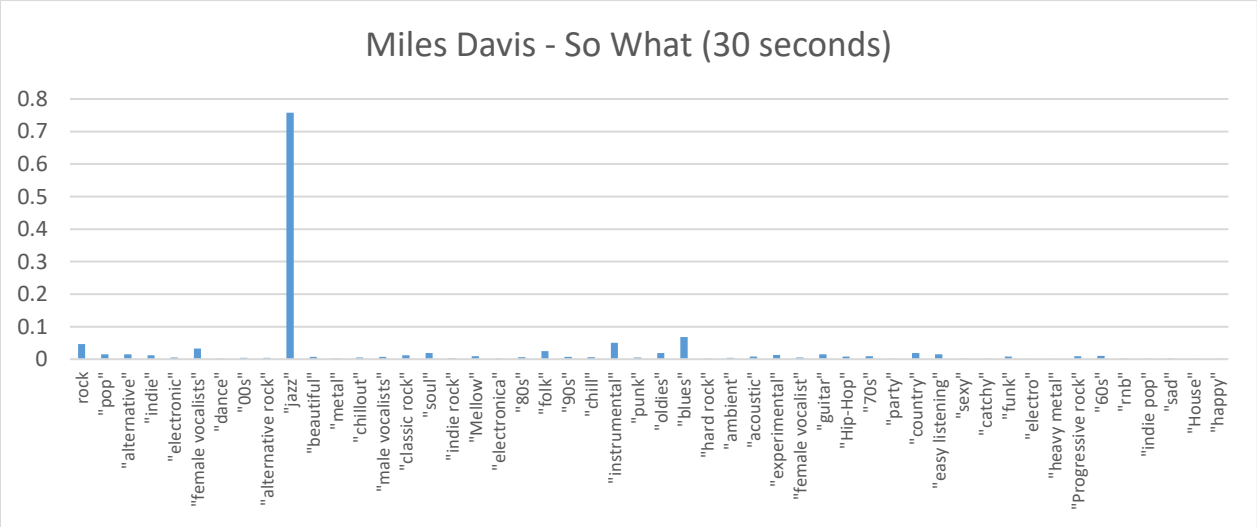

Figure 4-121. Jazz genre prediction results using a 30-second fragment of the song.

On the other hand, a 30-second fragment of Progressive Rock was also tested. However, the results were not accurate as expected, since this genre changes a lot throughout the song. This time, MusiCNN determined the tags Ambient, Electronic, and Experimental, which is not correct at all. Figure 4-132 shows this result:

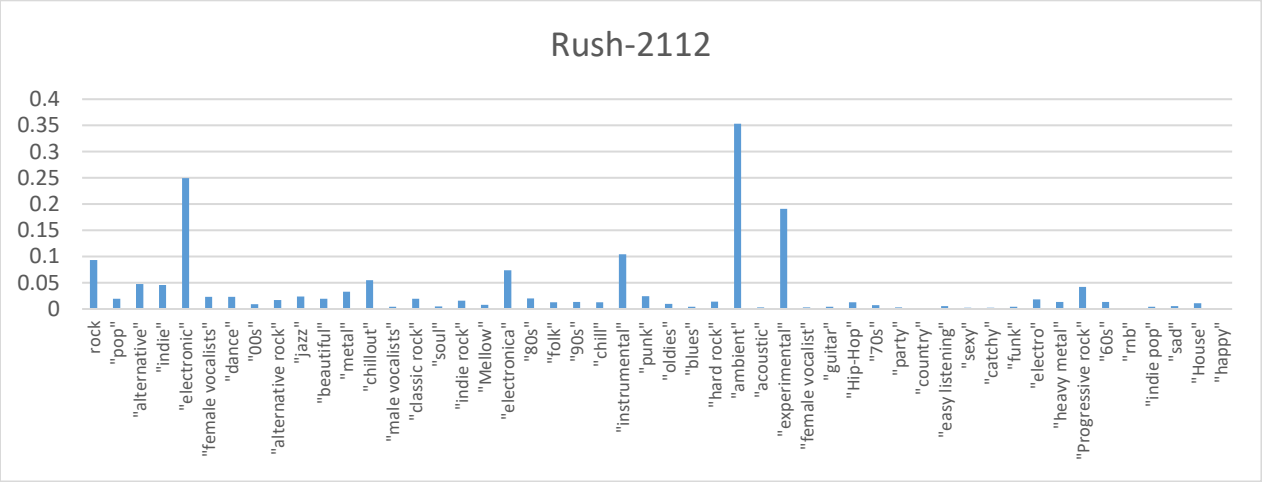

Figure 4-132. Progressive Rock genre prediction results using a 30-second fragment of the song.

4.5.4 Tests with genres not included in the dataset.

Some experiments were done with songs from other music genres that are not included in the dataset used to train the CNN. In this case, the tested genres were Salsa, Banda, and Reggaeton. MusiCNN determined that the Salsa genre is very similar to the Jazz genre, since this genre was born from the mixture of son Cubano with Jazz or Funk, and the used musical instruments are piano, trumpets, bass, and hand percussion, among others. Figure 4-143 shows the results of the Salsa song:

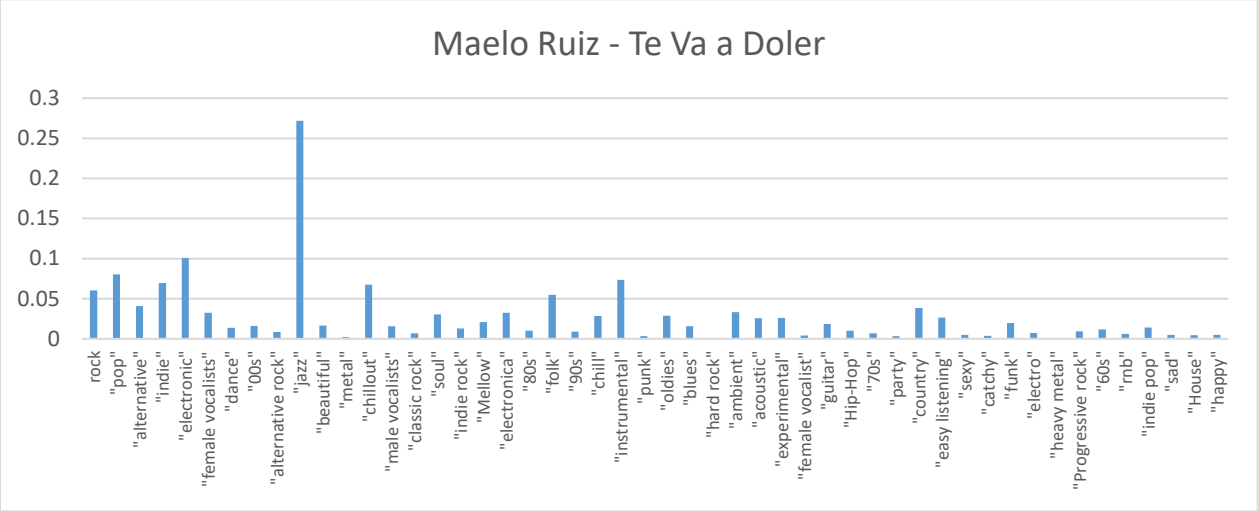

Figure 4-143. Salsa genre prediction results using a 30-second fragment of the song.

Surprisingly, the result of the experiment using a Banda Sinaloense song with MusiCNN determined that Jazz is the closer music genre tag since Banda Sinaloense music uses several instruments like trumpets, trombones, clarinets, snare drums, and congas among others. Also, it has some influences from German and French band music according to the ISIC (Instituto Sinaloense de Cultura) [48]. Figure 4-154 shows the automatic tagging results:

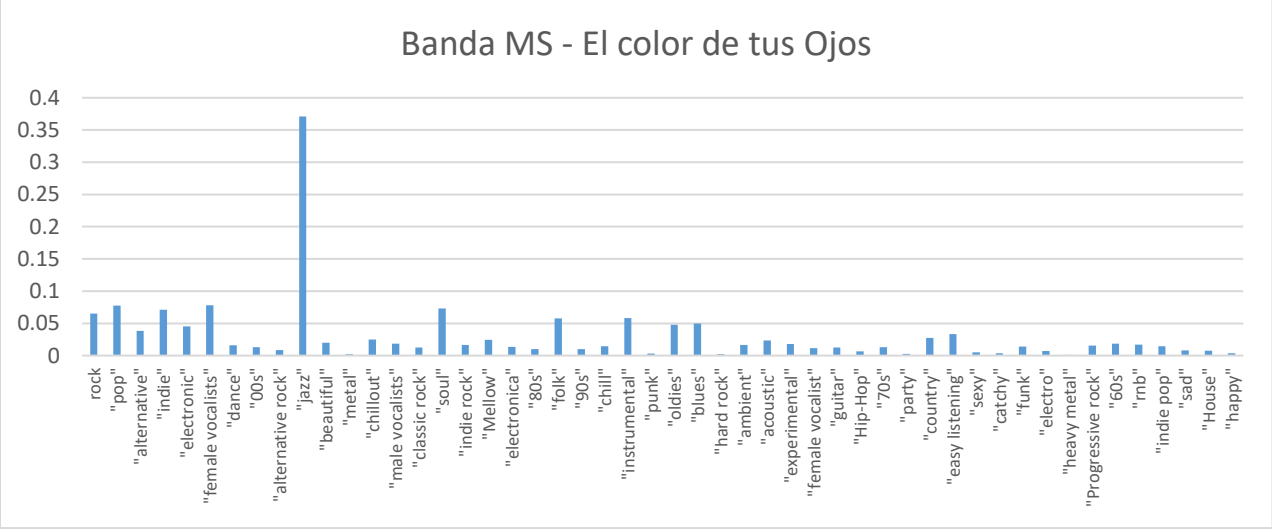

Figure 4-154. Banda Sinaloense genre prediction results using a 30-second fragment of the song.

Finally, a Reggaeton song was tested with MusiCNN, and the results shown in Figure 4-165 showed that the closer genres are Pop, Electronic, and Hip-Hop. This was expected because this type of music has elements used in Pop and Electronic music like rhythm or electronic sounds. The vocals are pretty similar to Hip-Hop music as well.

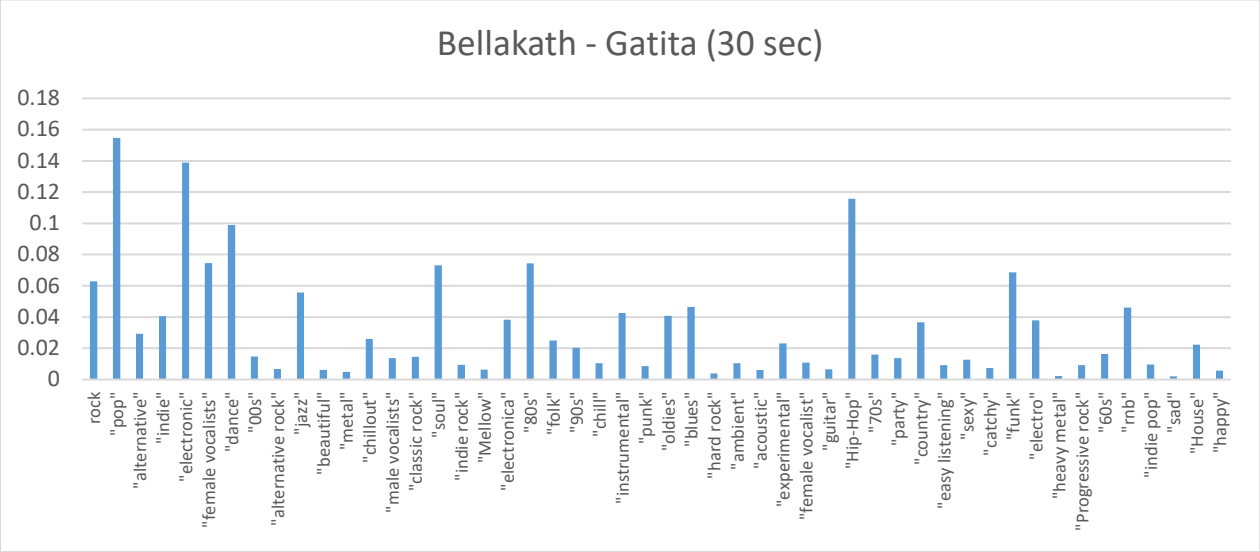

Figure 4-165. Reggaeton genre prediction result using a 30-second fragment of the song.

4.5.5 Accuracy Tests.

The following tables show the results of the automatic music genre tag determination for twenty songs grouped by genre. Table 4-1 depicts the result of the Pop genre:

Song	Determined tag	Likelihood value	EQ Preset
Michael Jackson - The way you make me feel	Electronic	0.436995	Electronic
Taylor Swift - Bad Blood	Pop	0.313525	Pop
Billie Eilish - Bad Guy	Electronic	0.372491	Electronic
Dua Lipa - Illusion	Female Vocalists	0.268005	Flat
Katy Perry - Roar	Electronic	0.200170	Electronic
Lady Gaga - Poker Face	Electronic	0.495953	Electronic
Mark Ronson - Uptown Funk	Electronic	0.281816	Electronic
Pharrell Williams - Happy	Jazz	0.219518	Jazz
Adele - Set Fire to the Rain	Female Vocalists	0.274769	Flat
The Weekend - Blinding Lights	Electronic	0.498184	Electronic
Madonna- Material Girl	Electronic	0.326386	Electronic
Britney Spears - Baby One More Time	Pop	0.274062	Pop
Spice Girls - Wannabe	Pop	0.394489	Pop
Backstreet Boys - Everybody	Electronic	0.357478	Electronic
Roxette - Joyride	Rock	0.230184	Rock
Black Eyed Peas - Where is The Love	Pop (Hip-Hop)	0.343471	Pop
Rihanna - Umbrella	Pop (Hip-Hop)	0.643334	Pop
Shakira - Whenever, Wherever	Pop	0.291562	Pop
Gwen Stefani - Hollaback Girl	Pop	0.186380	Pop
Pink - Get the Party Started	Dance	0.307324	Electronic

Table 4-1. Tag determination results of twenty Pop songs.

Pop was one of the less accurate music genres since only seven songs were tagged correctly. The cause of the low accuracy can be the presence of Electronic music elements in most of the Pop songs. There were a couple of songs classified as Hip-Hop, but this music genre was mapped to the Pop equalization preset. For that reason, those cases were considered correct.

On the other hand, the Electronic genre was the second most accurate due to the results shown in Table 4-2. In this case, only two songs were tagged incorrectly as Pop and Female Vocalist genres. As discussed before, Pop music may have rhythms and instruments similar to Electronic music, thus the model can determine this genre tag instead.

Song	Determined Tag	Likelihood Value	EQ Preset
Daft Punk - Get Lucky	Electronic	0.244034	Electronic
Avicii - Levels	Electronic	0.508414	Electronic
David Guetta - Titanium	Electronic	0.354341	Electronic
DJ Tiesto - Insomnia	Electronic	0.485662	Electronic
Calvin Harris - Outside	Pop	0.247093	Pop
Armin Van Buuren - In and Out of Love	Electronic	0.551007	Electronic
Stromae - Alors on Danse	Electronic	0.247312	Electronic
Steve Aoki & Laidback ft Lil Jon - Turbulence	Electronic	0.413554	Electronic
The Prodigy - Omen	Electronic	0.482420	Electronic
Fatboy Slim - Weapon of Choice	Electronic	0.306754	Electronic
The Crystal Method - Name of the Game	Electronic	0.242282	Electronic
Deadmau5 ft. Rob Swire - Ghosts 'n' Stuff	Electronic	0.461054	Electronic
Moby - Extreme Ways	Electronic	0.185865	Electronic
Loui Martz - Febrero	Electronic	0.548248	Electronic
Infected Mushroom - I Wish	Electronic	0.436318	Electronic
Poolside - Harvest Moon	Electronic	0.498684	Electronic
Skrillex ft. Sirah - Bangarang	Electronic	0.264052	Electronic
Astrix - Artcore	Electronic	0.621440	Electronic
The Prodigy - Breathe	Electronic	0.412883	Electronic
Martin Garrix - In the Name of Love	Female Vocalists	0.302904	Flat

Table 4-2. Tag determination results of twenty Electronic songs.

The rock genre was one of the most accurate as well since sixteen songs were tagged correctly. In some cases, the determined tag was Indie, Blues, or Punk and those genres were mapped to the Rock equalization preset because they can be considered correct. On the contrary, there were a couple of songs that the model determined as Electronic and Metal respectively. The song tagged as Electronic has a slow rhythm introduction, hence it was classified into that genre. In contrast, the song tagged as Metal has a heavy riff at the beginning of it, therefore it could be classified as that.

Table 4-3 shows the results of the analyzed Rock songs:

Song	Determined Tag	Likelihood Value	EQ Preset
Green Day - Basket Case	Rock	0.327540	Rock
Placebo - The Bitter End	Punk	0.343028	Rock
Nirvana - About a Girl	Indie	0.438068	Rock
Arctic Monkeys - Do I Wann Know?	Electronic	0.237886	Electronic
Frank Zappa - Peaches en Regalia	Jazz	0.149445	Jazz
Alice In Chains - Man in the Box	Metal	0.497095	Metal
The Beatles - Day Tripper	Rock	0.247930	Rock
Pink Floyd -Money	Blues	0.202551	Rock
Pearl Jam - Evenflow	Rock	0.362702	Rock
Rush - 2112	Ambient	0.355053	Instrumental
Incubus - Wish You Were Here	Rock	0.297585	Rock
Weezer - Hash Pipe	Rock	0.424622	Rock
Porcupine Tree - Time Flies	Rock	0.272847	Rock
The Jimi Hendrix Experience - Purple Haze	Rock	0.365887	Rock
Foo Fighters - The Best of You	Rock	0.395599	Rock
Guns N' Roses - Nightrain	Rock	0.364328	Rock
Oasis - Live Forever	Rock	0.249943	Rock
Muse - Hysteria	Rock	0.290559	Rock
Audioslave - Like a Stone	Rock	0.281083	Rock
The Offspring - Come Out and Play	Rock	0.244873	Rock

Table 4-3. Tag determination results of twenty Rock songs.

In the case of the Jazz genre, the accuracy of the model was 100% since all of the songs used in the test were predicted correctly. Also, some of the likelihood values were above 80% as shown in Table 4-4.

On the contrary, the Instrumental genre was one of the least accurate because many of the songs were predicted as Jazz genre. This could be due to the presence of instruments similar to Jazz such as piano or saxophone. Another reason could be the fact that the introduction of many classical songs is slow and calm, hence a thirty-second fragment is not enough to predict the correct music genre tag.

Table 4-5 describes the results of the Instrumental genre test.

Song	Determined Tag	Likelihood Value	EQ Preset
Jamie Cullum - But For Now	Jazz	0.537693	Jazz
Chet Baker - I'm Old Fashioned	Jazz	0.621336	Jazz
Diana Krall - Autum in New York	Jazz	0.268979	Jazz
Canonball Aderley - Dancing in the Dark	Jazz	0.826594	Jazz
Bill Charlap Trio - The Duke	Jazz	0.666668	Jazz
Houston Person - Willow Weep for Me	Jazz	0.378593	Jazz
Immanuel Wilkins	Jazz	0.885300	Jazz
Miles Davis - So What	Jazz	0.757720	Jazz
Breezing Easy	Jazz	0.595159	Jazz
Julian Lage - Serenade	Jazz	0.524415	Jazz
ARTEMIS - Nocturno	Jazz	0.580628	Jazz
Giovanni Guidi - Avec Le Temps	Jazz	0.432158	Jazz
Rymdem - Ro	Jazz	0.453604	Jazz
Mary Lou Williams - It Ain't Necessarily So	Jazz	0.800368	Jazz
Connie Han - Captain's Song	Jazz	0.872530	Jazz
Bobby Hutcherson - Stardust	Jazz	0.556211	Jazz
Dave Holland - Another Land	Jazz	0.802280	Jazz
Bill Frisell - Lookout for Hope	Jazz	0.513493	Jazz
Wynton Marsalis - Where or When	Jazz	0.683802	Jazz
Roy Hargrove - Trust	Jazz	0.730861	Jazz

Table 4-4. Tag determination results of twenty Jazz songs.

Song	Determined Tag	Likelihood Value	EQ Preset
Gerardo Tamez - Aire Istmeño	Instrumental	0.195596	Instrumental
John Williams - Fool on the Hill	Instrumental	0.237493	Instrumental
Ottmar Liebert - In the Morning (Hope, Coffee + Poetry)	Instrumental	0.200223	Instrumental
Julia Lange - Asturias (Isaac Albéniz)	Instrumental	0.228445	Instrumental
Andrea González Caballero - Sevilla (Isaac Albéniz)	Blues	0.257602	Rock
Gerardo Tamez - Danza de la Paloma Enamorada	Instrumental	0.260094	Instrumental
Amsterdam Guitar Trio - Sonate K 159 Allegro	Instrumental	0.263474	Instrumental
Ramin Djawadi - Light of the Seven	Instrumental	0.219790	Instrumental
Chopin - Nocturne Op. No. 2	Jazz	0.522365	Jazz
Niccolo Paganini - Caprice No. 24	Jazz	0.337616	Jazz
Andre Rieu - And the Waltz Goes On	Jazz	0.524746	Jazz
Agustin Barrios - La Catedral	Jazz	0.341150	Jazz
Michiru Yamane - Wood Carving Partita	Instrumental	0.222861	Instrumental
Phillip Glass - Opening	Ambient	0.331662	Instrumental
John Williams - The Imperial March	Jazz	0.278026	Jazz
Beethoven - Moonlight Sonata No.14	Ambient	0.247160	Instrumental
Handel - Harp Concerto	Jazz	0.366887	Instrumental
John Williams -Guitar Concerto No. 1 in A Major Op. 30	Instrumental	0.260707	Instrumental
Max Ablitzer - Grave Song	Instrumental	0.199598	Instrumental
Ludovico Einaudi - Fly	Instrumental	0.251421	Instrumental

Table 4-5. Tag determination results of twenty Instrumental songs.

Last but not least, the accuracy of the predictions of the Metal genre was good, since only two songs were tagged as Rock and some of the likelihood values were above seventy percent. This could be because the introduction part of some Metal songs does not have heavy riffs, which depends on the subgenre. Another reason could be the date when the song was recorded since some of the early Metal songs have big influences of mid-seventies Heavy Rock. Table 4-6 shows the results of this test:

Song	Determined Tag	Likelihood Value	EQ Preset
Metallica - Master of Puppets	Metal	0.425695	Metal
Megadeth - Symphony of Destruction	Metal	0.306767	Metal
Gojira - Stranded	Rock	0.175188	Rock
Amon Amarth - Guardians of Asgaard	Metal	0.556476	Metal
Hypocrisy - Warpath	Metal	0.837848	Metal
Cradle of Filth - Cruelty Brought Thee Orchids	Metal	0.574322	Metal
Slayer - Repentless	Metal	0.243292	Metal
Arch Enemy - Nemesis	Metal	0.719406	Metal
The Black Dahlia Murder - Miasma	Metal	0.708850	Metal
Pantera - Domination	Metal	0.640246	Metal
Bloodbath - Process of Disillumination	Metal	0.751553	Metal
Dimmu Borgir - Gateways	Metal	0.274767	Metal
Iron Maiden - The Trooper	Rock	0.309714	Rock
Children of Bodom - Needled 24/7	Metal	0.317787	Metal
Dark Tranquillity - Punish My Heaven	Metal	0.471217	Metal
Trivium - Pull Harder the Strings of Your Martir	Metal	0.701427	Metal
The Haunted - All Against All	Metal	0.718858	Metal
Opeth - Ghost of Perdition	Metal	0.572815	Metal
Korn - Divine	Metal	0.703683	Metal
Slipknot - My Plague	Metal	0.792367	Metal

Table 4-6. Tag determination results of twenty Metal songs.

5. Discussion.

The first part of the project was the implementation of the audio equalizers and the comparison between them. In this case, the parametric equalizer seems to be the one that possesses a better audio quality, because it allows the user to adjust more parameters. On the other hand, the graphic equalizer seems to be a friendlier option to the user, since only the gain values need to be adjusted and it does not require so much technical knowledge. However, the audio quality was not as good as the parametric one.

Regarding the auto-tagging problem, the results of the experiments showed good accuracy of the MusiCNN neural network for determining music genre tags. However, some tracks had some difficulties when determining the correct genre, such as executing the auto-tagging process at the beginning of the song. Sometimes this part does not represent the actual genre since the track has a long introduction, or its progression is not quick, for example, when it is a live or music video version. This can cause the neural network to confuse the real genre with other ones, like in the case of the example shown in section 4.5.1.

Another difficulty that can be present is when the track contains elements of other genres. For example, Progressive Rock songs sometimes have an introduction or bridge with a calmer or faster rhythm or use other musical instruments such as flute or sax, which can cause the determined genre not to be accurate, as shown in sections 4.5.2 and 4.5.3. Some modern Pop songs include Electronic sounds which can cause the CNN to determine a different tag. In some cases, the algorithm determined the Electronic genre when it was a Rock song. For this reason, it may be a difficult task to tag a song with only one music genre and the result also depends on the music genres included in the dataset that was used to train the neural network.

To have a better idea of the results, the model accuracy can be calculated using the Formula 5-1 [49] and the values from Table 5-1 which summarizes the results from Tables 4-1 to 4-6:

		True Values						
	Classes	Pop	Electronic	Rock	Jazz	Instrumental	Metal	Total
Predictions	Pop	7	9	1	1	0	0	18
	Electronic	1	18	0	0	0	0	19
	Rock	0	1	16	1	1	1	20
	Jazz	0	0	0	20	0	0	20
	Instrumental	0	0	0	6	13	0	19
	Metal	0	0	2	0	0	18	20
	Total	8	28	19	28	14	19	**116**

Table 5-1. Confusion Matrix of Multiclass Accuracy

$$Multiclass\ Accuracy = \frac{7+18+16+20+13+18}{116} = 0.7931 \quad (5\text{-}1)$$

These results indicate that the MusiCNN model had an accuracy of 79.31% for the multiclass problem. Even though the value is not a hundred percent, it seems the overall accuracy was good. Pop and Instrumental genres were the less accurate classifications which could be caused by the data that was used to train the model and the fact that thirty-second length fragments were used for performing the tests. Perhaps using a larger song fragment could improve the accuracy of the results.

Because of these difficulties, the option to execute the auto-tagging process during the playing of the whole song and therefore have a more precise audio equalization was considered, but it would imply more processing effort that can have an impact on the audio output performance. Perhaps this feature could be implemented in a future version of the application.

Even though the use of deep learning algorithms that employ CNNs is more precise and recommended for the auto-tagging process, the understanding and setup are not so straightforward. To use this kind of tool, some previous basic knowledge of low-level programming (when using Essentia library in a C or C++ program) and how to compile the libraries is needed. Some basic knowledge of Ubuntu OS was needed as well to install all the prerequisites and compile TensorFlow, which is necessary to execute Essentia's artificial intelligence algorithms. Also, some difficulties may arise when using a Virtual Machine

environment: In this case, some updates from the host OS Windows 11 caused crashing errors when trying to execute TensorFlow in Ubuntu OS. These updates deactivated the AVX functionality (Advanced Vector Extensions) that is needed to run TensorFlow functions. To fix this issue, Hyper-V and Windows Memory Integrity features had to be disabled.

6. Conclusions.

In this thesis, a smart equalizer application was presented to suggest an innovative idea of audio equalization that can be used in different devices such as a smartphone or car infotainment, where the objective was to determine the music genre of each song for applying the most suitable equalization preset. Therefore, the quality of audio can be improved. A Matlab application was created as a proof of concept to demonstrate the functionality of this equalizer using a deep learning approach to achieve the music genre tag determination.

Different ways of performing automatic tagging were explored and it was decided in the end to use the Essentia library due to its capacity to extract music features with MIR techniques and the fact that it includes the MusiCNN deep learning algorithm that excels in performing the automatic tagging task needed to determine the music genre of each song tested.

The application has good performance since the DL algorithm was programmed in C++ language and Matlab allows to easily call its functionality, which makes it worth considering for the implementation on an embedded system. Nevertheless, this kind of implementation needs to be analyzed first to choose the best hardware option that is capable of executing TensorFlow functionality, due to the current limitations of its lighter version.

The implementation of the proposed smart audio equalization solution on an embedded system such as Raspberry Pi 4 was considered at the beginning of the project. However, the use of TensorFlow on embedded devices is currently limited to a few platforms. An analysis has to be done first to determine if the operators used in the MusiCNN neural network are supported by the platform and the models have to be converted into a lighter version known as TensorFlow Lite for Microcontrollers, so they do not use so much storage, since this type of devices possess limited resources such as low memory quantity. The use of Open Auto was also considered because it is an open-source platform based on Android Auto that can be used to emulate a car infotainment system on some devices like a Raspberry Pi 4. This implementation ended up out of the scope for this thesis project.

6. CONCLUSIONS

In the future, the implementation mentioned before could be tried by adapting the current application made in Matlab to a solution that can be executed on an embedded environment. TensorFlow Lite for Microcontrollers currently supports devices like Raspberry Pi, DSPs, and some Cortex-M microcontrollers since it includes optimized kernels for these specific devices. The possibility of adapting the MusiCNN model should be analyzed and in case it is not feasible, a simpler model should be considered instead.

Another interesting aspect worth considering would be to train the neural network using a dataset that includes a wider range of music genres, such as AcousticBrainz or MTG-Jamendo, so it can identify modern genres and apply a more suitable audio equalization preset to the music.

Although this is an innovative idea of audio equalization that could be helpful to drivers in avoiding distractions on the road and improving the audio quality without complex configurations, some people like to listen to the music the way it was originally recorded. Also, it has to be considered that the equalization presets are usually created by acoustic engineers who adjust the parameters according to their own experience and criteria, but some users do not like them and prefer to use custom ones. The quality of the audio depends on the speaker setup as well; hence it should not be considered that audio equalization will improve it like a high-end system.

References.

[1] Välimäki V, Reiss JD. All About Audio Equalization: Solutions and Frontiers. Applied Sciences. 2016; 6(5):129. https://doi.org/10.3390/app6050129.

[2] Audio frequency spectrum explained.
Available at: https://audiosolace.com/audio-frequency-spectrum-explained

[3] Best equalizer settings.
Available at: https://www.headphonesty.com/2021/07/best-equalizer-settings/

[4] J. Rämö, V. Välimäki and B. Bank, "High-Precision Parallel Graphic Equalizer," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 22, no. 12, pp. 1894-1904, Dec. 2014.

[5] Seat Leon audio review: Better than the Audi A3?
Available at: https://totallyev.net/seat-leon-audio-review-better-than-the-audi-a3/

[6] Understanding Sound Levels.
Available at: https://www.kissyourears.com/pages/understanding-sound-levels

[7] Adjust the sound quality in iTunes on PC.
Available at: https://support.apple.com/guide/itunes/adjust-the-sound-quality-itns2991/windows

[8] U. Zölzer, Digital Audio Signal Processing (2nd ed)., J. Wiley & Sons, 2008.

[9] Best Equalizer Settings: The Definitive Guide.
Available at: https://emastered.com/blog/best-equalizer-settings

[10] Kenwood UD-901, UD-951M, Instruction Manual.
Available at: https://archive.org/details/manual_UD901_OM_KENWOOD_EN/

[11] Hyoung-Gook Kim, Sang-Soon Nam. Automatic Equalizer Control Method Using Music Genre Classification in Automobile Audio System. The Journal of The Korea Institute of Intelligent Transport Systems. 2009.

[12] Ma, Z.; Reiss, J.D.; Black, D. Implementation of an intelligent equalization tool using Yule-Walker for music mixing and mastering. In Proceedings of the 134th Convention of the Audio Engineering Society, Rome, Italy, 4–7 May 2013

[13] What Is Mastering & Why Is It Important?
Available at: https://www.abbeyroad.com/news/what-is-mastering-why-is-it-important-productionhub-2957

[14] González, M.R., Cuadra, P.D., & Cádiz, R.F. (2015). Fuzzy Equalization of Musical Genres. International Conference on Mathematics and Computing

[15] Gracenote's Dynamic EQ Automatically Tunes Car Stereo Systems One Song At A Time. Newcomb, D.
Available at: https://www.forbes.com/sites/dougnewcomb/2015/12/30/gracenotes-dynamic-eq-automatically-tunes-car-stereo-systems-one-song-at-a-time/

[16] Iriz, J., Patricio, M.A., Molina, J.M., Berlanga, A. (2021). Smart Song Equalization Based on the Classification of Musical Genres. In: Herrero, Á., Cambra, C., Urda, D., Sedano, J., Quintián, H., Corchado, E. (eds) 15th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2020). SOCO 2020. Advances in Intelligent Systems and Computing, vol 1268. Springer, Cham.

[17] Bogdanov, D., Wack N., Gómez E., Gulati S., Herrera P., Mayor O., et al. (2013). ESSENTIA: an Audio Analysis Library for Music Information Retrieval. International Society for Music Information Retrieval Conference (ISMIR'13).

[18] Knees, P., Schedl, M. Contextual music meta-data: comparison and sources. In: Music Similarity and Retrieval. TIRS, vol. 36, pp. 107–132. Springer, Heidelberg, 2016.

[19] Knees, P., Schedl, M. Music Similarity and Retrieval: An Introduction to Audio- and Web-based Strategies. Springer Berlin Heidelberg, 2016.

[20] Schedl, M., Gómez, E., Urbano, J., Music Information Retrieval: Recent Developments and Applications. Now, 2014.

[21] J. Volkmann, S. S. Stevens, E. B. Newman; A Scale for the Measurement of the Psychological Magnitude Pitch. J. Acoust. Soc. Am. 1 January 1937; 8 (3_Supplement): 208.

[22] O'Shaughnessy, D. Speech communication: human and machine. Addison-Wesley. p. 150. 1987.

[23] Short-Time Fourier Transform. Mathworks.
Available at: https://la.mathworks.com/help/dsp/ref/dsp.stft.html

[24] Mel Spectrogram. Mathworks.
Available at: https://la.mathworks.com/help/audio/ref/melspectrogram.html

[25] Song, G., Wang, Z., Han F., Ding, S., Gu, X., Music auto-tagging using scattering transform and convolutional neural network with self-attention, Applied Soft Computing, Volume 96, 2020.

[26] Choi, K., Fazekas, G., Sandler, M., Automatic Tagging Using Deep Convolutional Neural Networks. International Society of Music Information Retrieval Conference, 2016.

[27] Google Developers.
Available at: https://developers.google.com/focus/ai-development

[28] OpenAI.
Available at: https://openai.com/news/research/

[29] Nam, J., Herrera, J., Lee, K., A Deep Bag-of-Features Model for Music Auto-Tagging, 2016.

[30] Max Pooling.
Available at: https://medium.com/@danushidk507/max-pooling-ef545993b6e4

[31] Choi, K., Fazekas, G., Sandler, M.B., & Cho, K., Convolutional recurrent neural networks for music classification. 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[32] Ferraro, A., Bogdanov, D., Jeon, J.H., Yoon, J., & Serra, X., Music Auto-tagging Using CNNs and Mel-spectrograms With Reduced Frequency and Time Resolution, 2019.

[33] Music Technology Group (MTG). ESSENTIA.
https://www.upf.edu/web/mtg/technology-portfolio/-/asset_publisher/eFDkHCFq97rR/content/technologies-essentia/maximized

[34] The Music Information Retrieval Evaluation Exchange (MIREX)
Available at: https://www.music-ir.org/mirex/wiki/MIREX_HOME

[35] K. C. B. Liang, Q. M. BU, and T. M. Entertainment, "Do user preference data benefit music genre classification tasks?" MIREX, 2020.

[36] TensorFlow Tutorials.
Available at: https://www.tensorflow.org/tutorials

[37] Prudencio de Dueñas, A. Audio generado frente a audio original: Análisis y comparación mediante técnicas de recuperación de información musical. Master's thesis, 2020.

[38] Magenta. Make Music and Art Using Machine Learning.
Available at: https://magenta.tensorflow.org/

[39] MuseGAN.
Available at: https://hermandong.com/musegan/

[40] Pons, J., & Serra, X. MusiCNN: Pre-trained convolutional neural networks for music audio tagging, 2019.

[41] Pons J., Lidy T., Serra X., Experimenting with musically motivated convolutional neural networks, 2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI), Bucharest, Romania, 2016.

[42] Huang, G., Liu, Z., & Weinberger, K.Q. Densely Connected Convolutional Networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[43] Bertin-Mahieux, T., Ellis, D. P. W., Whitman, B. & Lamere, P. The Million Song Dataset, In A. Klapuri & C. Leider (eds.), ISMIR (p./pp. 591-596), University of Miami. 2011.

[44] Design Shelving Filter. Mathworks.
Available at: https://la.mathworks.com/help/audio/ref/designshelvingeq.html

[45] Parametric Equalizer Design. Mathworks
Available at: https://la.mathworks.com/help/audio/ug/parametric-equalizer-design.html

[46] Installing Essentia.
Available at: https://essentia.upf.edu/installing.html

[47] Ambient Music Genre. MusicBrainz.
Available at: https://musicbrainz.org/genre/608b0471-7531-4854-a348-e698c69cb699

[48] La Tradicional Musica de Viento.
Available at: http://culturasinaloa.gob.mx/index.php/patrimonio-cultural/patrimonio-inmaterial-/festejos-y-carnavales/4515-la-tradicional-musica-de-viento

[49] How to check the accuracy of your machine learning model.
Available at: https://deepchecks.com/how-to-check-the-accuracy-of-your-machine-learning-model/

Index