Received 20-06-21; Accepted 23-06-21; Published 30-06-2021

Scraping Relative Chord Progressions Data for Genre Classification

Noelia Rico and Irene Díaz
Department of Computer Science,
University of Oviedo, Spain.
Susana Montes {NOELIARICO,SIRENE,MONTES}@UNIOVI.ES
Department of Statistics and Operation Research and Maths teaching,
University of Oviedo, Spain.

Corresponding Author: Irene Díaz.

Abstract

Genre classification has been a hot topic for years now in the field of music information retrieval. Most of the current works study music using song waves as input data. In this work, we present a different approach to genre classification taking into account chord progressions. A full data set has been created for this work: first gathering songs for each genre (pop, indie, rock and reggae) from Spotify and then scraping chord progressions data of that songs from the website Ultimate Guitar. Different models which aim to classify the genre of the songs have been trained using convolutional neural networks for pair comparison between genres classification. Some of those models are used for discerning between two concrete genres given, getting up to a value of 91% for AUC metric classifying songs between pop and rock. Music Information Retrieval Chord Progressions Convolutional Neural Networks Spotify API Genre Classification.

Keywords: Music Information Retrieval, Chord Progressions, ConvolutionalNeural Networks, Spotify API, Genre Classification.

1. INTRODUCTION

The field of music classification has attracted considerable interest in the last decade due to its potential for generating recommendation systems in the widespread market of streaming music among other applications [1,2].

The research based on the analysis of music waves as input data has received most of the attention in the field. It is a common practice to use either a song wave or spectrogram as input data or to perform feature extraction [3]. Nevertheless, these techniques are usually computationally demanding and tend to create black box models when used in machine learning or deep learning for predictive tasks. Furthermore, some the authors perform genre classification using features obtained from a MIDI (Musical Instrument Digital Interface) file such as in [4].

The term chord progression (sequence of chords) defines a series of two or more triads of notes used in a song as base harmony of the piece. Few researchers have addressed music related problems using sequences of chords and, the ones who did focus mostly on music generation rather than classification [5]. Approaches using chord progressions for genre classification have been presented although these sequences require again multimedia files to get the data [6].

The Million Song Dataset [7], is very popular in this field but it does not contain information about the chord progressions. Researchers have explored different kinds of sources for gathering data such as feature extraction from images of music sheets [8]. In the approach presented by [9], multimedia files are used in order to extract the chord progressions before classifying the songs into genres. The authors comment that the most difficult task is classifying the songs when the data set is formed by broadly used genres, and they explicitly talk about genres such as rock and pop. In [10], feature extraction is done over multimedia files for gathering song-related attributes of the songs, obtaining an accuracy around 60% at its best.

In a previous work [11], the authors used the API of the website Hooktheory in order to retrieve some data[12]. Hooktheory is a community website where users analyze pieces of songs and determine if a song has a progression of the type defined within a set of preferred common progressions. Nevertheless, because the songs are analyzed by pieces and not completely, some information is missing. The progressions of chords can be combined in a song in order to get a pattern, and how they are combined may be relevant to get the genre of the songs. For this reason, it is worth studying how these chord progressions behave in the task of classifying songs by genre.

This paper presents a new approach to classify songs in genres that avoids the need for the sound wave or any other multimedia input. On the contrary, the input data are sequences of integer numbers in the interval [1,7] that represent the relative chord progressions which constitute the harmony of the musical pieces.

For this work, a set of songs is chosen for a set of artists which belong to a concrete genre. The songs are retrieved from the Spotify API. Then, the chord progressions are gathered scraping the website Ultimate Guitar [1]. These progressions must be clean and translated before using them as input of neural networks in order to get a suitable format for the training process.

This paper is organised as follows: The concepts about music theory that are necessary to understand the work are presented in Section 2. For the supervised learning task Convolutional Neural Networks are introduced in Section 2. In Section 3 is explained in detail. Experiments for genre classification and their results are commented in Section 4. Finally, the conclusions drawn from this work are expressed Section 6.

2. ABOUT MUSIC

The *pitch* is the quality of sounds defined by the rate of vibrations that produces the sound. Hence, it determines the degree of highness or lowness of a *tone* [13]. More concretely, the pitch may be quantified as frequency measured in *hertzs* (Hz). This property makes possible sorting the notes in a frequency-related scale.

Each tone is associated with a concrete frequency. To simplify the nomenclature, instead of referring the tones by their measure in hertz, they are sorted from low to high frequencies and named. Different names have been proposed since the primeras muestras de mÃºsicas back in siglo y ref. Along this work we will use the English notation for naming the notes *{A, B, C, D, E, F, G}*, which is equivalent to the Latin notation *{La, Si, Do, Re, Mi, Fa, Sol}*. ref a Harvard dictionary.

The tones that sound good together to the human ear are somehow related in their frequencies. This is what make some combinations of tones much more common than others.

When writing music, tones are represented as *notes* in a *staff*, this is, figures over a set of 5 horizontal lines and their corresponding four spaces. Notes can be written on the lines and spaces and the higher the note is written, the more high-pitched it is.

The distance between two notes is called ***interval***. Intervals measure how far away one note is from another. Generally, two consecutive natural notes are separated by one unit. One this happens, it said that they are separated by ***one tone***. In the ordered set of notes $\{A, B, C, D, E, F, G\}$ most pairs of consecutive notes is separated by one tone. There are two exceptions though: The distance between *E* and F as well as the distance between *B* and *C* is separated only by half of a tone, which is known as a ***semitone*** (also called *halftone* in more informal environments).

To the original set of tones, additional tones can be added by altering each tone by a semitone using the two following mechanisms:

- ***sharp*** ($\#$): Add a semitone to a tone.
- ***flat*** ($\flat$): Subtract a semitone from a tone.

Note that, using these symbols, each altered tone has two different names, depending on how it is thought. For example, given the tones *A* and *B* with distance equal to two semitones (1 tone), the note in the middle that is one semitone from both can be named as $A\#$ and $B\flat$, this is, adding a semitone to A or subtracting a semitone from B.

The distances between the notes are clearly illustrated in an image of a keyboard where the representation of the keys show these separations. Between one key and the next one (black or white), there is always a semitone. The white keys represent natural notes and the black keys on the keyboard represent altered notes. FIGURE1 shows the names of all the notes placed on a keyboard.

2.1 Intervals

The unit measure of distance between two notes is a tone. Nevertheless, different intervals can be defined using a number and a label. The number depends always on the number of notes from one natural note to the next counting themselves, being thus the minimum 2. For example, the interval between C and E is 3 (a third) and between C and G is 5 (a fifth). Not all the intervals of the same number imply the same amount of tones and semitones, and thus a label is required.

Notation for major and minor intervals is usually written as uppercase M and lowercase m. Nevertheless, during this work for the sake of clarity the names maj for major and min for minor will be

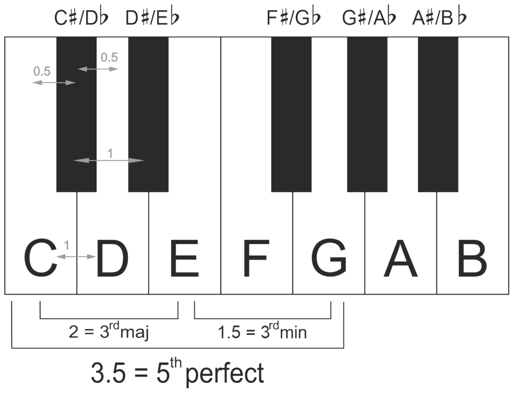

Figure 1: The keyboard illustrates the distance between the notes. The distance from a note to the next one (taking both black and white keys into account) is one by semitone. Thus, the distance between two notes with a black key in between adds up to one tone, a semitone from the white to the black plus one semitone from the black to the next white key. Names of the notes represented on a keyboard. The pitch of the notes increments a semitone key by key from left to right.

Table 1: Names of the intervals in relation to their distance and number of tones of separation between the notes.

Interval	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6
	2min	2maj	2-Aug	-	-	-	-	-	-	-	-	-
	-	3dim	3min	3maj	3-Aug	-	-	-	-	-	-	-
	-	-	-	4dim	4per	4-Aug	-	-	-	-	-	-
	-	-	-	-	-	5dim	5per	5-Aug	-	-	-	-
	-	-	-	-	-	-	6dim	6min	6maj	6-Aug	-	-
	-	-	-	-	-	-	-	-	7dim	7min	7maj	7-Aug
		-	-	-	-	-	-	-	-	-	8dis	8per

used. Common intervals and the ones that must be known for our study are. Intervals with 2, 3, 6, 7 can be *major* or *minor* but intervals with 4 and 5 are commonly *perfect*. Furthermore, all of them can be *augmented* (aug) or *diminished* (dim). TABLE 1 shows all the possible intervals.

Intervals can be melodic or harmonic, this is, the notes can sound successively or at the same time. The former include chords, where three or more notes sound simultaneously.

2.2 Chords

The simultaneous sounding of three or more tones is called a chord. Chords can be divided into two main classes, consonant and dissonant. To the former belong the *major* and *minor* triads. Triads are the most common kind of chord and include three notes: the root, the third and the fifth. Depending on the intervals used for building the triad, the chord will be classified differently. There most common triads used in western music are:

- *major*: major third (3maj) + perfect fifth (5per) = major third (3maj) + minor third (3min)
- *minor*: minor third (3min) + perfect fifth (5per) = minor third (3min) + major third (3maj)
- *augmented*: major third (3maj) + augmented fifth (5aug) = major third (3maj) + major third (3min)
- *diminished*: minor third (3min) + diminished fifth (5dis) = minor third + minor third

For example, the *major chord* with C as root is build with an interval of 3M (2 tones) and over it a 3m (1.5 tones = 1 tone + 1 semitone), so, looking at the keyboard we can confirm that it is $C+E+G$. This is the main idea to construct the scales and determine the key of the song.

The study of the chords and their relationships and functions is an important field of music theory called harmonic analysis.

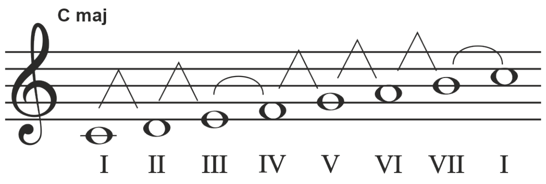

Figure 2: C maj scale. V edges represent one tone between the notes and U edges a semitone between the notes.

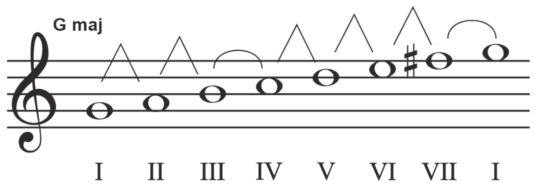

Figure 3: G maj scale. V edges represent one tone between the notes and U edges a semitone between the notes.

A total of seven notes are part of a scale. The intervals between those notes and consequently the chords that are build them over them determine the mode of the scale (FIGURE 2 and FIGURE 3).

2.3 Scales, Scale Degrees, Tonality, And Mode

All the tones represented in the keyboard can be combined in one scale which consists of twelve semitones, the chromatic scale. Nevertheless, the basic scale of western music is the diatonic scale. It consists of five whole tones (t) and two semitones (s) in the following arrangement t t s t t t s. This scale is usually referred as major scale as distinguished from the minor scale in which the

Table 2: Function of each chord according to its relative position to the tonic of the scale.

I	Tonic
II	Supertonic
III	Mediant
IV	Subdominant
V	Dominant
VI	Submediant
VII	Leading note

arrangement of intervals is t s t t t t s. The center tone of the scale is called tonic and the other tones of the scale are not considered equally important but are related and subordinate to them. FIGURE 2 represents the most basic major scale, C maj.

Both major and minor scales may be transposed, starting on any one of the twelve notes (the basic set and their alterations). Thus, there are twelve major scales and twelve minor scales, one in each key. When this scales are transported, they must conserve the distance between the notes altering the necessary notes.

The main note of tonal center to which all its notes are related is the key. This is important as each degree of the scale has a concrete function. That means that a named chord has a different function depending on the degree they have in relation to the tonic. The possible degrees of a chord in a scale are shown in TABLE 2.

The type of chords that are used to build a scale varies depending on the **mode** of the scale. These chords have harmony because of their structure and, in the same way, a scale sound equals to other of its mode and different from those that have different modes. This is because of the structure of the scale. For each scale, a triad of notes built over each of the notes of the scale has a different structure. This is motivated by the distance between each note.

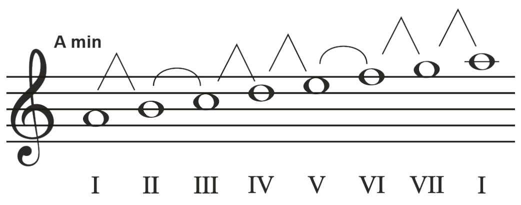

Figure 4: A min scale. V edges represent one tone between the notes and U edges a semitone between the notes.

The mode of the scale denotes the selection of tones arranged in a scale, that form the basic tonal substance of a composition (FIGURE 4). Each major scale has a relative scale in minor mode. Two scales are relative if they have the same number of alterations.

Scales contain only one type of alterations, this is, they cannot mix flats and sharps. Relative scales with their alterations are shown in TABLE 3.

Table 3: Major and minor scales and their number and type of alterations.

Alteration	Number	Major	Minor
♭	7	C♭ maj	A♭ min
♭	6	G♭ maj	E♭ min
♭	5	D♭ maj	B♭ min
♭	4	A♭ maj	F min
♭	3	E♭ maj	C min
♭	2	B♭ maj	G min
♭	1	F maj	D min
	0	C maj	A min
#	1	G maj	E min
#	2	D maj	B min
#	3	A maj	F# min
#	4	E maj	C# min
#	5	B maj	G# min
#	6	F# maj	D# min
#	7	C# maj	A# min

The chord progression represents the harmony base of a song. Once created, our data set contains sequences of chord progressions.

We consider in this work sequences of chord progressions to define the songs that are going to be classified. Then, given a chord progression, we will try to classify the song in different genres. Commonly, chord progressions are represented by the name of their base. In order to be able to compare all the songs, the chord progressions must be written relatively to their grade in the scale. All the concepts explained in this section are applied in our work to estimate the key of each song given a chord progressions for later translating the chords to their relative positions in the scale.

3. CREATION OF THE DATA SET

In order to perform a supervised task for classifying songs, both their chord progressions and their respective genres are necessary. As there are no available datasets with this information, the first step is to construct one.

In order to compare the songs, the sequence of chords must be expressed in in a common scale. As it was explained in Section 2, the function of each chord in a scale varies. This has a heavy impact on the impressions that the listener has of that chord. Thus, the data must express their chords relatively to its tonality. Although in the classical music notation roman numerals are usually employed, here we will use integer values within the interval $[1, 7]$. To create the dataset, the Spotify API and the Ultimate Guitar website have been used. The steps followed to create these dataset are:

1. Define a list of genres that will be use for the study.
2. Elaborate a list of songs' titles for each genre.

3. Get the chords of each song progression.
4. Determine the key of each song if not provided.
5. Assign to each song the relative position for each chord.

In this work four genres have been evaluated: *indie, pop, reggae, rock*. For each of those genres a list of top artists is retrieved using the Spotify API [14]. These artists were carefully examined by the authors who completed these lists with custom election until getting a list of approximately twenty authors per genre. The 10 most popular songs of these were retrieved using the Spotify API. Note that the list of top songs would be different if it were gathered now since Spotify does not use the global number of reproductions but some other factors such as streaming in short periods of time or abruptly increase of streams. This means that the songs collected for this set of artists can be different depending on the moment that they are retrieved.

At this point, the three first steps of the list have been completed and the number of songs collected for each genre goes up to 200.

The next step is gathering the chord progressions of each song. The website chosen to retrieve this data is Ultimate Guitar. Each song is characterised by its genre, author, name, chord progressions and tonality if it is available.

3.1 Cleaning the Chord Progressions

Once the chord progressions of each song is obtained, the data must be cleaned in order to obtain the desired format. FIGURE 5 shows two examples of songs gathered from UltimateGuitar represented by their chord progressions. The format of the name of each chord varies depending on their nature. It is necessary to use a standard nomenclature in order to process the data. This requires a careful cleaning process that searches all the patterns that appear in the list of chords.

```
# Example song 1
"Am" "Em" "G"  "D"  "Am" "Em" "G"  "D"  "Am" "Em" "G"  "D"
"Am" "Em" "G"  "D"  "Am" "Em" "G"  "D"  "Am" "Em" "G"  "D"
"Am" "Em" "G"  "D"  "Am" "Em" "G"  "D"  "Am" "Em" "G"  "Em"
"Am" "Em" "G"  "D"  "Am" "Em" "G"  "D"  "Am" "Em" "G"  "D"  ..

# Example song 2
"C" "G" "F" "F" "F" "G" "G" "Gadd9" "Fmaj7" "G6" "F" "G" "C"
"F" "G" "C" "F" "G" "Gadd9" "G" "C" "F" "G" "Gadd9" "G" "C"
"F" "G" "Gadd9" "G" "C" "F" "G" "G6" "C" ...
```

Figure 5: Example of two different songs represented by their chord progressions gathered from UltimateGuitar.

FIGURE 5 shows all the different chords' names obtained from the songs. In this case the only interesting chords are those *major*, *minor*, *augmented* and *diminished*. This means that if a chord of any type adds (add) or suspends (sus) any extra note the program will not take that note into account because it is not relevant for our purpose, and it will keep only the basic type of the chord. As we are interested in triads, the problem with this representation is that a chord identifying the triad may have different nomenclatures. The main triad may suffer modifications such as added notes. At the moment, we are only interested in the three notes that constitute the main triad so that will be omitted and such chords will be translated to the name of the main triad (FIGURE 6).

```
"A" "A#" "A#dim" "A#m" "A2" "A4" "A5" "A6" "A7" "A7sus" "A7sus4" "A9"
"A9sus4" "Aadd4" "Aadd9" "Ab" "Ab(9)" "Ab5" "Ab7" "Abm" "Abm7" "Absus4"
"Am" "Am11" "Am6" "Am7" "Amaj7" "Asus" "Asus2" "Asus4"

"B" "B#5" "B5" "B7" "B7sus2" "B9" "B9sus4" "Bb" "BB" "Bb5" "Bb7" "Bbadd9"
"Bbm" "Bbmaj7" "Bbsus4" "Bm" "Bm11" "Bm6" "Bm7" "Bm7add11" "Bm9" "Bmadd11"
"Bmaj7" "Bsus2" "Bsus4"

"C" "C#" "C#5" "C#6" "C#7" "C#dim7" "C#m" "C#m7" "C#m7b5" "C#maj7" "C#sus4"
"C2" "C3" "C5" "C6sus2" "C7" "C7sus4" "Cadd2" "Cadd9" "Cdim" "Cm" "Cm6"
"Cm7" "Cmaj7" "Cmaj9" "Csus2"

"D" "D#" "D#5" "D#7" "D#dim" "D#m" "D#maj7" "D#sus4" "D2" "D4" "D5" "D6"
"D6add9" "D7" "D7sus4" "D9" "Dadd4add9" "Dadd9" "Db" "Db(b5)" "Dbm" "Dbmaj7"
"Dm" "Dm13" "Dm7" "Dm9" "Dmaj11" "Dmaj7" "Dmaj9" "Dsus" "Dsus2" "Dsus4"

"E" "E2" "E5" "E7" "E7sus4" "Eadd2" "Eaug" "Eb" "Eb(b5)" "Eb5" "Eb6" "Ebm"
"Ebmaj7" "Ebsus2" "Edim" "Edim7" "Em" "Em11" "Em6" "Em7" "Em9" "Emadd4"
"Emadd9" "Emaj7" "Esus2" "Esus4"

"F" "F#" "F#5" "F#7" "F#m" "F#m7" "F#sus4" "F2" "F5" "F6" "F7" "F7sus4" "Fm"
"Fm13" "Fm7" "Fmaj" "Fmaj7"

"G" "G#" "G#5" "G#7" "G#dim" "G#m" "G#m7" "G#m9" "G13" "G2" "G3" "G4" "G5"
"G6" "G6add11" "G6add9" "G6sus2" "G7" "Gadd11" "Gadd4" "Gadd9" "Gb" "Gm"
"Gm(maj7)" "Gm6" "Gm7" "Gmaj7" "Gsus" "Gsus2" "Gsus4"
```

Figure 6: All the possible chord names that can be used in the chord progressions got from the UltimateGuitar website.

Also, chords that only contain uppercase are major chords, but those who have text such as 'maj' are also major. Hence, a common nomenclature must be selected in order to standardized the input data. On the other hand, minor chords are those which contain a lower case 'm', (obviously not followed by 'aj' in which case will be major) but it can be followed by text like 'add' and it would be still minor. These names must by unified prior to training the models.

All the chords are translated to a common nomenclature where each of the chords is named with a name following the format $x[y]z_1z_2z_3$ where:

- x exists in the set $\{A, B, C, D, E, F, G\}$ and represents the name of the note.
- y is optional and represents the alteration of the note if any. It can be # (representing sharp) or b (representing flat). If y does not appear the note x is natural, without alterations.

- $z_1z_2z_3$ is a three-letter code that represents the type of the chord. It can take any value in the set {maj, min, aug, dim}.

The duration of each chord is not considered in this work. Thus, some authors can write the same chord consecutively more than once when it has a long duration while others just write a new chord when a change must be made. This is a problem that can introduce noise in the data, because it should be irrelevant. For example a *C - F - G - C* progression is equivalent to a *C - F - G - G - C* progression as we are not considering the extension of each chord in time. Thus, in order to avoid inconsistencies, repeated consecutive chords have been removed.

3.2 Estimate the Key Of The Songs

If the key is provided with the progression, it is straightforward to get the relative position of each chord in the song. Unfortunately, most of the songs (up to the 80% of the songs gathered) do not have this predefined value. Hence, it is necessary to define the scale in which the song is written. The idea is to estimate the scale based on the chords. In order to do this, a set of scales will be generated and then calculated. Although there are many modes of scales, only the most widely used in contemporary commercial music are considered. Those are major an minor scales. These scales are composed of chords of types already mentioned: *major*, *minor*, *diminished*, *augented*. To produce them, an auto-generator of scales has been developed. This auto-generator needs the modes (defined in the previous section) to generate all the possible scales in that modes based on of fifths for determining the key signature. Once the key is established it takes a progression with all the notes. All the scales used for estimating the key of a song are shown in TABLE 3. The steps followed to estimate the key of a song are:

1. Define a set of chords C.
2. Define a set of candidate scales $\mathcal{S}$. Each scale $s \in \mathcal{S}$ is defined by a subset of seven chords $C_s \subset C$.
3. For each scale s, consider each chord $c \in C_s$ and count the number of appearances in the song.
4. Select the scales s with highest count as key of the song.
5. Determine the tonality of the song untying between the relative major and minor scales.

To estimate the key of a given progression the first step is to calculate the number of chords of the progression that appear in each scale and then return the key with highest percentage of appearance as key of the progression. Then it calculates the number of notes of the progression that appears in each scale. When this percentage is available the one with highest appearance would be assigned as the key.

In classical music theory it is thought that in tonal modes, such as major and minor, the rest of the harmony lies around the key of the scale, so the first and last notes of the scale are usually the tonic (they key, the first note of the scale). This is taken into account in order un-tie the relative scales. If the scale starts with one of the candidates key, we choose that. If it does not, we choose the last

Table 4: Number of songs obtained for each genre that will be used for the supervised learning task of classifying songs by genre using chord progressions.

Genre	indie	pop	reggae	rock
Number of songs	164	176	82	151

one. For the current data set with these two simple steps the ties are always solve, but in case of tie when checking the last note one of the relatives would be chosen based in the most repeated note of the song and otherwise randomly.

In order to test this estimation, the algorithm is evaluated estimating the key for the 20% of the songs that have a defined key concluding that, using this technique all, but 2 songs of the gathered with value for tonality match the key estimated.

3.3 Final Data Set

The songs are defined as sequences of characters. TABLE4 shows the number of songs, considering a song as a progression of chords, gathered for each genre:

The final output of the data set consist of a unique json file. FIGURE 7 gives a glimpse of its structure:

```
[
  {
    "genre": "indie",
    "chords": [1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7,
     1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7,
     1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7,
     1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7,
     1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7, 1, 6, 3, 7]
  },
  {
    "genre": "indie",
    "chords": [1, 5, 2, 3, 1, 5, 1, 5, 2, 3, 1, 5, 1, 5, 2, 3, 1, 5, 1, 5, 2, 3, 1, 5,
    1, 5, 2, 3, 1, 5, 1, 5, 2, 3, 1, 5, 1, 5, 2, 3, 1, 5, 1, 5, 2, 3, 1, 5, 1, 5, 2, 3,
    1, 5, 1, 5, 2, 3, 1, 5, 1, 3, 2, 1, 3, 2, 1, 3, 2, 1, 5, 2, 3, 1, 5, 1, 5, 2, 3,
    1, 5, 1, 3, 2, 1, 3, 2, 1, 3, 2, 1, 3, 2, 1, 3, 2, 1, 3, 2, 3, 1]
  },
  ... # all the indie songs, followed by all the pop songs and then reggae and rock songs
  {
    "genre": "rock",
    "chords": "[1, 6, 5, 6, 4, 5, 1, 6, 1, 5, 6, 4, 5, 1, 6, 1, 5, 6, 4, 5, 1, 6, 1, 5,
    6, 4, 5, 1, 6, 1, 5, 6, 4, 5, 1, 6, 1, 5, 6, 4, 5, 1, 6, 1, 6, 1, 5, 6,  4, 5, 1,
    6, 1, 5, 6, 4, 5, 1]
  }
]
```

Figure 7: A piece of the final dataset used for classify chord progressions in genres.

At this point a full data set has been built. Some statistics about the length of the chord progressions are shown in TABLE 5.

Table 5: Information about the length of the chord progressions that represent the songs.

genre	number of songs	mean	median	max	min
indie	163 (28.62%)	81.56707	77	184	6
pop	176 (30.72%)	93.96000	90	276	19
reggae	82 (14.32%)	100.95062	97	330	9
rock	151 (26.36%)	88.59333	86	282	7
total	570 (100%)	90	87	330	6

The distribution of chord progressions' length is presented in FIGURE 8 below.

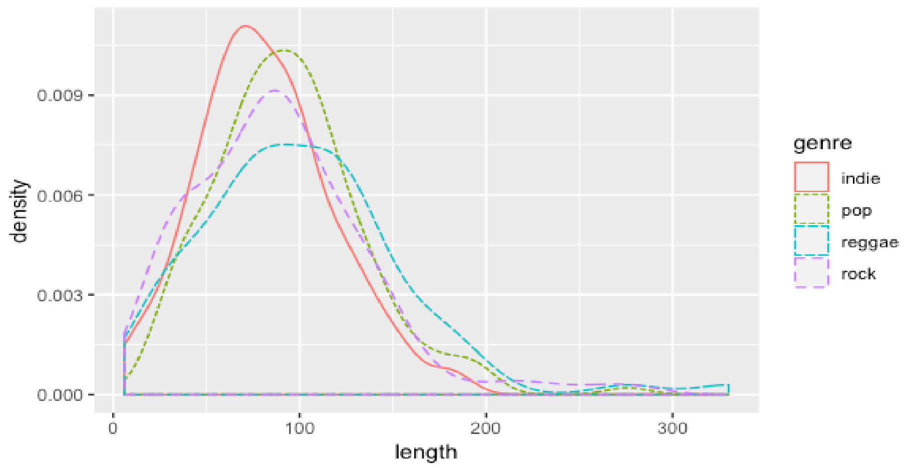

Figure 8: Plot of the distribution of the length of the chord progressions representing each song grouped by genre.

4. SONG CLASSIFICATION

The aim of our work is to perform a supervised task to classify songs of four different genres (*indie*, *pop*, *reggae* and *rock*). In order to achieve this purpose, we are going to use chord progressions of songs whose genre is part of the dataset developed in Section 3. This is a supervised classification task as the real genre of each chord progressions is known.

A one vs one classification process is studied with the aim of studying how easy is to differentiate between the different genres.

4.1 Classification Model and Evaluation

To perform such classification, we propose the use of neural networks (NN). NN learns how to map a set of inputs to a set of outputs given a training data. NN are trained iterating multiple times over the same data using stochastic gradient descent in order to find the parameters for the network that optimizes a loss function. This loss function is used to evaluate a candidate solution, which is in practice a set of weights. Because it is the value that the network tries to optimize, it is usually referred to as *the objective function*. The most used loss function by far for classification models is *cross-entropy*, which measures the error between two probability distributions. The choice of loss function is directly related to the activation function used in the output layer [15].

A special kind of neural networks are Convolutional Neural Networks (henceforth CNNs) that have been proved to be a successful tool in fields such as image or text [16,17] classification. CNNs are fully connected networks where each neuron in one layer is connected to every neuron in the next layer. More concretely, 1-dimensional CNNs (CNNs-1D) are able to extract features from local one dimensional input patches. These are the networks employed in this work.To configure a CNN is necessary to consider the following parameters:

- An optimization algorithm to obtain a; configuration of the internal parameters that performs well against some evaluation measure such as logarithmic loss or mean squared error.
- A batch size that establishes the number of examples presented to the network before updating the weights. A training set can be divided in one or more batches.
- The number of epochs, that defines the number times that each example is presented to the network. An epoch is comprised of one or more batches.

To prevent overfitting, the models are built using a validation data, so evaluation measures are computed at the end of every epoch. In particular NN is trained using a cross validation scheme. The full dataset is divided in k stratified folds and maximum number of epochs e is established. For each fold i of the k folds, a training set tr and test set te is defined. The training set tr contains the data from all the folds except the i fold, that is stored in the test set te. For each fold, a new model m_i is built using the training tr_i and test te_i datasets. For each fold i the neural network is iterated e times to fit the model m_i. For each echo $j \in 1:e$, the $accuracy_{m_{ij}}$ is stored after being computed the accuracy of the model trained with tr_i in the test set te_i. Once this process is finished for all the k folds, the average accuracy of for each echo j is computed $accuracy_j = \frac{1}{k}\sum_i^k accuracy_{m_{ij}}$.

The number of epochs for the final model is chosen based of the best average accuracy. The process followed for training the network is outlined in Algorithm 4.1. The number of epochs tested for the neural networks of this study is 30.

4.2 One Vs. One Classification Problem: Discerning Between a Pair of Genres

For the pairwise comparison of the genres, different subsets have been selected from the data for training each model. More concretely, 6 different subsets have been used, building each of them

Algorithm 1: Algorithm describing the process followed for training the models to classify the chord progressions by genre

```
Divide the dataset into $k = 10$ different folds;
Set the number of echos $e$;
for $i \in 1 : k$ do
    $tr_i$ = data from all folds but $i$;
    $te_i$ = data from $i$ fold;
    for $j \in 1 : e$ do
        fit model $m_i$ using $tr_i$ as training data;
        validate $m_i$ the model using $te_i$ as test data;
        store accuracy of $m_{ij}$;
    end
end
for $j \in 1 : e$ do
    avg_accuracy$_j$ = 0;
    for $j \in 1 : e$ do
        avg_accuracy$_j$ += accuracy$_{m_{ij}}$;
    end
    avg_accuracy$_j$ = avg_accuracy$_j$ / $k$;
end
```

by filtering the chord progression that belongs to two genres of the original dataset each time. The final structure for the binary neural network is shown in FIGURE 9.

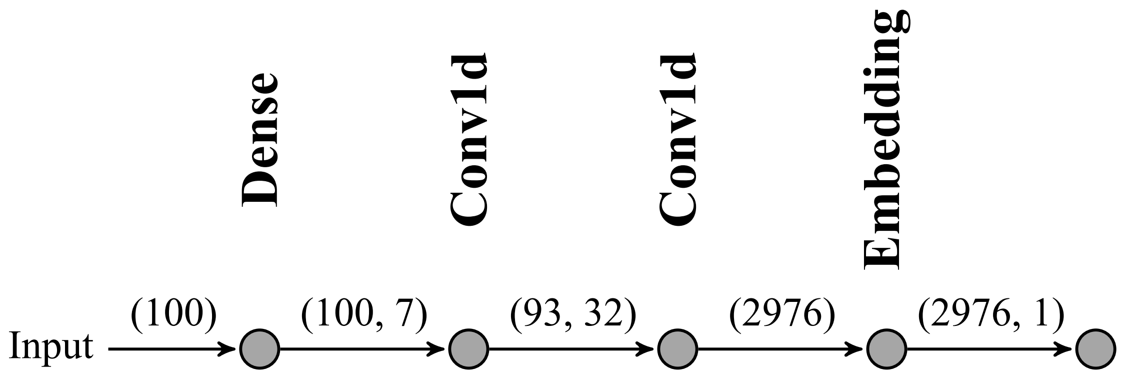

Figure 9: Architecture of the neural network for the binary classification problem.

The last layer of the neural network requires a *loss function* that fits the problem that the network aims to resolve. For these binary problems, the ***sigmoid activation****function* has been selected:

$$Sigmoid(x) = \frac{e^x}{e^x + 1}$$

Using this activation function, the output of this last layer is a number in the interval $[0, 1]$ which represents the probability of the song to belong to a given genre.

Table 6: Confusion matrix.

	Predicted		
		Preterm	Term
***Real**	Preterm	True Positive (TP)	False Negative (FN)
	Term	False Positive (FP)	True Negative (TN)

To measure how well the neural network works, the ***binary crossentropy*** loss function has been used. At this point, all the labels will be encoded as a vector of 0s and 1s that can be compared for each instance with the numeric value that the network outputs.

The neural networks has been trained with chord progressions of length 60 for the binary problem with the sequences of pairs.

5. EXPERIMENTS

The experiments discussed in this section have been obtained running the models in a Jupyter Notebook hosted in Google Colab with a NVIDIA Tesla T4 GPU. Once the optimal number of *epochs* is found, training a model for that optimal number of epochs has an execution time around 5 seconds for the binary problems. The dataset as well as the code used for developing this project is available in a (hitherto) private Github repository and the obtained results are easily reproducible. The authors will be glad to grant access to the repository to anyone who wants to check the code. The networks were trained in Python using the library keras [18].

In order to evaluate the methods, different settings have been tried to fit the optimal number of layers (and their parameters) as well as the length of the sequences. It has been observed that, passing the input as *n-grams* with *n*= 2 or *n*= 3 of consecutive numbers (chords), the network behaves better than when the input is given as a sequence of one-digit numbers. Also, different architectures of neural networks have been tried before choosing the two described in the following Paragraphs. Models of one and two one-dimensional convolutional layers with different parameters and with and without pooling and embedded layers have been tried. Note also that, as presented in FIGURE 8, the length of the chord progressions varies for each song. In order to make the data suitable for the input layer, a fixed length for the chord progressions must be established and the sequences shorter than this length must be padded with zeros or truncated in order to equalize its length before they are given to the network. The network parameters have been optimized. Thus the results shown here are those obtained with the best configuration.

To evaluate the performance of the presented method several metrics are selected from the confusion matrix shown in TABLE 6. The confusion matrix is a contingency table that shows how the objects of a binary classification problem are classified according to their real value and the one predicted by the classification method.

Table 7: F1 results obtained with the model trained using the binary neural network.

vs.	**indie**	**pop**	**reggae**	**rock**
indie	-	0.6826	0.4000	0.7742
pop	-	-	0.7401	0.8424
reggae	-	-	-	0.6245
rock	-	-	-	-

Table 8: AUC results obtained with the model trained using the binary neural network.

vs.	**indie**	**pop**	**reggae**	**rock**
indie	-	0.7647	0.5313	0.7334
pop	-	-	0.7794	0.9176
reggae	-	-	-	0.5916
rock	-	-	-	-

From the above defined confusion matrix the value $F_1 = \frac{2*TP}{2*TP+FP+FN}$, is computed [19]. In addition, AUC [20], and Accuracy are also considered to evaluate the different learning processes. Results are shown in TABLE 7, TABLE 8 and TABLE 9 respectively.

The embedding representation of the songs will be learnt together with the main task of the neural network, this is, classifying songs into different genres. To learn this representation, an embedding layer that takes a 2D tensor (matrix) of integers as input with the shape (number of samples, sequence lengths) is used.

The embedding of sequences of numbers created by the *Embedding* layer of keras are learned from the data, and their dimension is up to the developer criteria.

The results obtained are better than the ones obtained with the data set described previously that was gathered from Hook Theory [12]. This can be interpreted as a symptom that the fact of how the progressions are combined in a song influences somehow the genre to what the song belongs. However, the results in terms of performance are not goodenough, especially to differentiate between reggae and indie.

Table 9: Accuracy results obtained with the model trained using the binary neural network.

vs.	**indie**	**pop**	**reggae**	**rock**
indie	-	0.6969	0.6666	0.7741
pop	-	-	0.8000	0.84375
reggae	-	-	-	0.6956
rock	-	-	-	-

6. CONCLUSIONS AND FUTURE WORK

In this work, we classify songs in different genres using chord progressions as input data and performing a supervised task training models with CNNs.

Furthermore, a new approach for generating datasets using Spotify and UltimateGuitar has been presented. The songs retrieves songs represented as chord progressions and is able to compute the relative position of those chords to the tonality after estimating their key. This process allows to retrieve a set of 500 chord progressions to be gathered in no more than two minutes and to perform a cleaning task in seconds. Then, the data is ready to be fetched to a Convolutional Neural Network. Also, this data set is easily reusable and not only in order to classify songs into genre but also for validating the results in tasks where the aim is to extract chord progressions of music waves such as [21, 22].

Although the performance classification is higher than the obtained by previous approaches, in future works we plan to improve both song representation and classification models.

7. ACKNOWLEDGEMENTS

This research was funded by Spanish Government through the MINECO grant number TIN2017-87600-P.

8. CONFLICT OF INTEREST

The authors declare that they have no conflict of interest.

References

[1] https://www.ultimate-guitar.com/

[2] Song Y, Dixon S, Pearce M. A Survey of Music Recommendation Systems and Future Perspectives. The 9th International Symposium on Computer Music Modeling and Retrieval (CMMR) At: London, UK. 2012.

[3] Su Y, Zhang K, Wang J, Madani K. Environment Sound Classification Using a Two-Stream CNN Based on Decision-Level Fusion. Sensors. 2019;19:1733.

[4] http://www.haralick.org/ML/CLASSIFICATION_OF_MUSICAL_GENRE_A_MACHINE_LEARNING_APPROACH.pdf

[5] http://cs229.stanford.edu/proj2015/136_poster.pdf

[6] Perez-Sancho C, Rizo D, Inesta JM., Ponce de Leon PJ, Kersten S. Genre Classification of Music By Tonal Harmony. Intell Data Anal. 2010;14:533-545.

[7] Bertin-Mahieux T, Ellis DPW, Whitman B, Lamere P. The Million-Song Dataset. In Proceedings of the 12th International Conference on Music Information Retrieval, ISMIR, 2011.

[8] Pacha A, Hajic J, Calvo-Zaragoza J. A Baseline for General Music Object Detection with Deep Learning. Applied Sciences. 2018;8:1488.

[9] Perez-Sancho C, Rizo D, Inesta J. Genre Classification Using Chords and Stochastic Language Models. Connect. Sci. 2009;21:145-159.

[10] Tzanetakis G, Cook P. Musical Genre Classification of Audio Signals. IEEE Transactions on Speech and Audio Processing. 2002;10:293-302.

[11] Rico N, Dıaz I. Chord Progressions Selection Based On Song Audio Features. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), LNAI. 2018;10870: 490-501.

[12] https://www.hooktheory.com/theorytab/common-chord-progressions

[13] Milne A. A Psychoacoustic Model of Harmonic Cadences. University of Jyväskylä. 2009.

[14] Diaz F. Spotify: Music Access at Scale. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR. New York, NY, USA. 2017;1349.

[15] Charu C. Aggarwal. Neural Networks and Deep Learning - A Textbook. Springer;2018.

[16] Krizhevsky A, Sutskever I, Hinton GE. Imagenet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012; pages 1097-1105.

[17] Zhang X, Zhao J, LeCun Y. Character-Level Convolutional Networks for Text Classification. In Proceedings of the 28th International Conference on Neural Information Processing Systems, Cambridge, MA, USA, MIT Press. 2015;1:pages 649–657.

[18] Chollet F. Deep Learning with Python. Manning Publications Co., Greenwich, CT, USA, 1st edition; 2017.

[19] Chinchor N. MUC-4 Evaluation Metrics. In Proceedings of the Fourth Message Understanding Conference. 1992:22–29.

[20] Bradley AP. The Use of The Area Under The Roc Curve In The Evaluation Of Machine Learning Algorithms. Pattern Recogn. 1997;30:1145-1159.

[21] https://arxiv.org/pdf/1706.02921.pdf

[22] https://arxiv.org/pdf/1808.05335.pdf