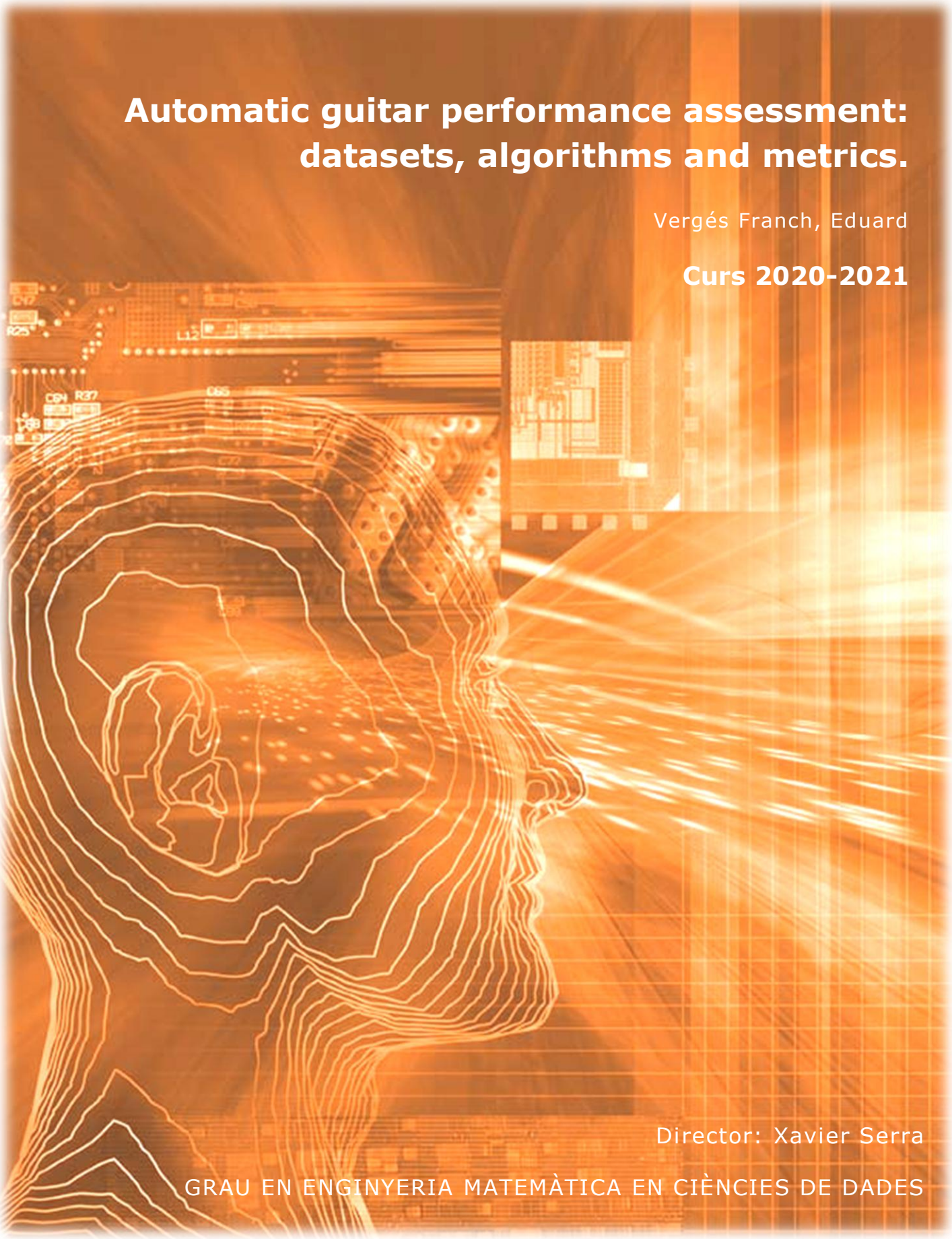
Automatic guitar performance assessment: datasets, algorithms and metrics.

Vergés Franch, Eduard

Curs 2020-2021

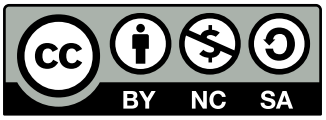


Director: Xavier Serra

GRAU EN ENGINYERIA MATEMÀTICA EN CIÈNCIES DE DADES

upf. Universitat Pompeu Fabra Barcelona
Escola d'Enginyeria

Treball de Fi de Grau



Acknowledgement

First, I want to thank Xavier Serra for introducing me to the Music Information Retrieval research field, giving me the opportunity of working inside Music Technology Group with an international team of researchers, helping to do this project by dedicating so much of his time to me and contributing with his rigor.

Also, I wanted to thank Vsevolod Eremenko which has dedicated so many hours to me and is an amazing researcher from which I learned a lot about MIR and good development and experimental practices.

Thanks to all researchers, Ph.D. students, SMC Master students and collaborators of MTG Asplab2 for helping and advising me when doing the internship and this project and motivating me in the context of the COVID-19 pandemic. Especially thanks to Alia Morsi, Jyoti Narang, Marius Miron, Genís Plaja, Pedro Ramoneda, Nazif Cant Tamer, Thomas Nuttall, Rafael Caro, Ramon Romeu and Hyon Kim. Its been a pleasure to had shared with you these months, it would have been nice to meet in person.

Thanks to my friend Nil Sallés for checking the spelling.

Finally, special thanks to my family for supporting me all these years and financing my studies.

Abstract

The aim of this project is to contribute to the development of a music transcription system of guitar performances in the context of music education. Using the Music Critic system as a baseline we identified aspects to improve and then we implemented and tested our contributions. We show how the system reacts to different contextual variations and discuss possible repercussions on a real context application. Also, we studied how the model used behaves using different training conditions. Furthermore, the *Five Guitar dataset*, with 90 guitar recordings, is designed especially for this project and publicly available. We use a data augmentation strategy to obtain a higher number of recordings simulating different rooms, mics, effects and recording setups. We observe that room acoustics and recording setup could generate biases on the final performance of the model and that the system is consistent according to timber. Also, we discover the need of representing all the pitch-class sets into the training set, which could be a limitation in a real situation, plus a high bias in the model.

Keywords: Automatic music performance assessment; Guitar; Pitch; Chroma; Datasets

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Context

This project is part of an internship made by the author of this project inside Music Technology Group in Universitat Pompeu Fabra (MTG) [1] and that tries to contribute to Music Critic [11], a system currently being developed by the same group.

While Music Critic is a system that aims to perform an automatic assessment of musical performances, this project focus on guitar performances and the capacity of the system to transcribe them. A dataset of guitar performances is constructed and used to **test the performance of the system regarding the automatic transcription of musical events**.

This project could be considered part of Audio Chord Estimation (ACE) inside Music Information Retrieval (MIR) field of study [2] and it tries to illustrate the difficulties when evaluating musical data [23, Chapter 5.1].

1.2 Motivation

First of all, music online classes still have a lot to desire. During the COVID-19 pandemic, many students have continued their studies online without any substantial difference despite being at home. But this is not the case for music students, because instrument lessons still rely on face-to-face learning and do not have any

[1] https://www.upf.edu/web/mtg

[2] https://www.music-ir.org/mirex/wiki/2017:Audio_Chord_Estimation

good alternative. Hence, in a digitized society and with advanced technology, researchers must provide solutions for musical online education. Now, some companies started to develop some solutions, such as Yousician [3], but they are focused on beginner and amateur guitarists and are not based on educational meaningful rubrics, comparable to the ones that a teacher can give to a student.

Secondly, having open-source online musical educational tools could help to break some economical and social barriers and music would benefit from it as many interesting artists would appear. The same applies to open-source databases which facilitates scientific research.

Finally, there are some personal motivations for doing this work. For me, as a jazz guitarist and engineering student, its is pretty important to merge both worlds and try to get the best of both implementing solutions to actual shortcomings.

1.3 Music concepts and nomenclature used

To understand some of the observations I make in this project it is important to know some basic music theory. In music, we have notes and chords. A note represents a single musical sound and has a pitch and a duration assigned to it and a chord are various notes played at the same time. The chromatic scale (Table 1.1), is a sequence of twelve notes each a semitone above or below its adjacent notes. Each of the notes would have a degree assigned to it according to the position it is found in the scale.

Pitch	C	Db	D	Eb	E	F	Gb	G	Ab	A	Bb	B
Degree	1	b2	2	b3	3	4	b5	5	b6	6	b7	7

Table 1.1: C chromatic scale

Chords could have different harmonic properties depending on the notes composing them and this would imply different kinds: major, minor and power chords. At least there must be three notes to consider a sound as a chord, if there are two notes it is considered as an interval. In this project, we focus on third intervals, which also can be major or minor depending on the tone distance between the two notes.

Then, here are the note degrees which form each chord and that are considered as relevant when studying chroma:

[3] https://yousician.com/

- Major chord: 1 , 3 and 5 (I, III and V).
- Minor chord: 1, b3 and 5 (I, IIIb and V).
- Power chord: 1, 5 and 1 (I, V, I). I degree repeated two times. the second time and octave above.
- Major third interval: 1 and 3 (I and III).
- Minor third interval: 1 and b3 (I and IIIb).
- Single note (note):1 (I).

Finally, we talk about musical events referring to a set of audio frames representing a piece of musical information that could be a chord, a note or an interval and that have some pitch, duration and kind defining it. A combination of a pitch plus a kind forms a pitch-class set (e.g. C major or D5). Hence, each musical event would correspond to a specific pitch-class set. In this project I would assign a label to represent each musical event kind:

- Major chord : maj
- Minor chord : min
- Single note : 1
- Power chord: 5
- Major third interval: +3
- Minor third interval: -3

Pitches would be named according to the American encryption: *C, Db, D,Eb, E, F, Gb, G, Ab, A, Bb, B*. Considering 12 pitches plus 6 kinds, we would have **72 pitch-class sets** in this project.

1.4 Structure of the report

Chapter I introduces the context and motivations of the project plus basic music theory needed to understand some of the project observations and analysis.

Chapter II gives background concepts related to my project. It talks about compositional data properties, the machine learning models used in pitch assessment, symbolic music representation for annotating the audio performances, signal processing software used and overview of Music Critic.

Chapter III illustrates the datasets used plus the *Five guitar dataset* designed and created specifically for this project. It gives statistics and explains the creation process for each of them.

Chapter IV explains the objectives of this project and how the experiments are designed. It also explains how the model predicts the pitch to understand how Music Critic works.

Chapter V contains all the experiments done and their results. You can find tests about how recording setup, guitar effects and filters and room acoustic affect the model performance. Also, you could see how training data could influence the model and what happens if the model is expanded to more pitch-class sets.

Chapter VI summarizes all the conclusions extracted from the experiments part.

Chapter 2

BACKGROUND

2.1 Introduction

In this chapter, some of the concepts that have a close relationship with this project are explained. We would take a look at some theoretical concepts about compositional data, talk about python libraries and software used for signal processing tasks, machine learning algorithms used for modeling pitch and an introduction to Music Critic.

2.2 Compositional data theory

In this project, we will work with NNLS chroma vectors (See section 2.5.1), which represent sound in a 12-dimensional space. Each dimension represents a portion of the total sound intensity and we need to know some concepts about compositional data for being able to work with this kind of vectors.

2.2.1 Definition

Compositional data provides information about portions of a total using vectors with all the components strictly positive and carrying out relative information [10, Chapter 2.3.1]. In our case, NNLS Chroma vectors represent a chord divided into different portions which are the 12 semi-tones in the chromatic scale [10, Chapter 4.1], each one with its intensity. These individual parts (semi-tones) are called the *components* and each one has its amount, which represents its importance within the total. They must sum up to a constant, in our case 1.

The amounts can be measured as absolute values (e.g. taking the intensity of each semi-tone as a measured value and sum up) or by ratios (e.g. ratio of rele-

vant degrees intensities inside a chord). Compositional Data Analysis studies the balance between different amounts and not the overall composition. It is typical to use only the amounts of interest for the specific application (e.g. to recognize the chord type, extract dimensions corresponding to relevant degrees) and force them to sum up to the constant. This is known as *closure operation* and it could be done as our evaluation does not depend on all the amounts. Also, as amounts sum up to a constant, they are *compositionally equivalent* [29, Chapter 2.1.4].

Compositional data must satisfy:

- Scaling invariance: all vectors must be normalized.
- Perturbation invariance: all vectors must be evaluated with the same units.
- Subcompositional Coherence:
  - Distance between two data points with D components must be higher when considering all components than only considering a sub-composition of these components.
  - Dispersion on a D-part composition (composition with D dimensions) must be higher than on a lower part composition.
  - If we fit a model in a D-part, adding a new random component would not affect the results.
- Permutation Invariance: sequence in which the components are given does not matter.

It's important knowing that methods from multivariate statistics analysis do not work on this type of data. Compositional data lies in a *simplex space* and must be transformed before using classical statistical tools. I would not go further in deep on this as it is out of my final degree work scope, but you could take a look at [29, Chapter 1.1.1, Chapter 2.4.5]. Transformations used in this project are explained in the following sections.

2.2.2 Visualization

Compositional data could not be plotted simply with a scatter plot considering only a subcomposition of the data, as these diagrams are not scaling and permutation invariant nor subcompositionally coherent. To visualize compositional data we can use:

- *Ternary plots*, which are a projection of a 3-dimensional scatter plot into a planar triangle spanned by three points (1,0,0), (0,1,0) and (0,0,1). Each corner of the triangle would represent a component. As a point is closer to a component, the component has more contribution on the data point and vice versa. The contributions for the three components must sum up to a constant.

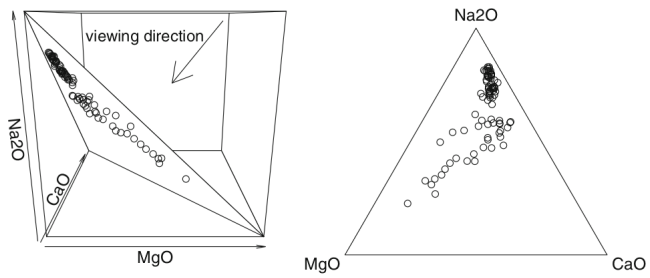


Figure 2.1: K Gerald van den Boogaart and Raimon Tolosana-Delgado.Analyzing com-positional data with R. [electronic resource] (2013, pg 25)

  In this project, we use these plots to visualize the Multivariate Gaussian models learned by the system (See 5.2).

- *Bar plots* which illustrate the amount for each component in the data. In this project, we use bar plots to see the intensities of each degree for each different kind of musical event (See 5.3).

- *Violin plots*, a variation of box plots with local densities represented [16] and used in this project to represent the NNLS chroma vectors learned (See 3.3.

- Chromagram, which is the representation of NNLS chroma vectors. For each semi-tone, we represent its intensity (See 5.1).

2.2.3 Subcomposition

This operation consists of only considering a subset of the dimensions in the data to obtain a particular result. In this work context, it is used when training the Gaussian Mixture model to learn the balance of the musical events according to its relevant degrees (See section 2.3.2). For each chroma vector, only the dimensions of the relevant degrees for a particular chord or note are considered to learn the Gaussian Mixture distribution. For example, in the case of a major chord, dimensions 1 (the root note), 5 (representing third degree) and 8 (representing fifth degree), would only be considered.

2.2.4 Additive Log-Ratio transformation

This transformation maps a composition of D parts lying on a *simplex space* into a D-1 euclidean vector using one part as the common denominator, allowing the use of classical statistical analysis tools. In this project, this transformation is applied before training a Gaussian Mixture model.

As an example, consider an NNLS Chroma vector representing a major chord. The relevant degrees of a major chord would be the first, the third and the fifth corresponding to the first, fifth and eighth dimensions of the vector. Doing a sub-composition of relevant degrees into the original vector produces a vector lying on a 3-dimensional *simplex*. Before training the model, we would apply an additive log-ratio transformation considering the first dimension (root) the common denominator, obtaining a 2-dimensional vector lying on an euclidean space, that can bused to train our Gaussian Mixture model (See 2.3.2).

2.2.5 Amalgamation

Amalgamation is the process of unifying various components within the compositional data. In our case, it is used to sum up all the relevant degrees intensities for an NNLS chroma vector and later learn a Beta distribution or evaluate its probability with a CDF (See 2.3.1).

When amalgamating, much information is lost. This process should only be done in the definition of the problem (e.g. before training the beta distribution model) and should be meaningful (e.g. only considering relevant degrees to predict the chord kind for a chroma vector, as if their intensities sum up to 1, it is highly probable that the vector is representing the desired chord and vice-versa).

2.3 Probabilistic and Machine Learning concepts

Here the models used to predict pitch in pysimmusic (See section 2.5.4) are explained. A pitch-class set is a combination of pitch plus a kind which could be major/minor/power chord, single note or major/minor third. Hence the objective is to estimate the pitch-class set from an input NNLS Chroma vector distribution. As explained in [11] two models have been used to estimate pitch accuracy:

- Beta distribution to model the *correctness* of a musical event.
- Gaussian distribution to model the *balance* of a musical event.

2.3.1 Beta distribution

Ideally, a major triad chord would be correct if the intensities of the first, third and fifth degrees represented in an NNLS chroma vector sum up to 1 and non-important degrees intensities equal to 0. Hence, the *correctness* of a musical event is understood as the ratio between relevant degrees and non-relevant degrees in the chroma vector. In Music Critic, from the input NNLS chroma vectors a Beta distribution [14] is learned and later its cumulative distribution function (*cdf*) is used to estimate how *correct* a new musical event is. This beta distribution is learned for each kind recognized by the model.

The beta distribution have the following density function:

$$f_X(x : \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1 - x)^{\beta-1} \tag{2.1}$$

defined for for *0 <= x <= 1, a>0, b>0*. In our case *x* would be the sum of relevant degrees or *in chord/note chromas* and α and β would be the learned parameters.

The use of this distribution allows obtaining a continuous probability which is used to color the notes in the final score (being red low probability/bad pitch accuracy and green high probability/good pitch accuracy).

2.3.2 Multivariate Gaussian

Music Critic is able to recognize various types of music events such as major, minor and power chords and single notes. For each of them, the system learns a Multivariate Gaussian model which describe the balance between the relevant degrees for the particular musical event (e.g. For major chords, the model learns the intensities relations that first, third and fifth degrees have). When predicting a new music event, the system would compare the probabilities for each Gaussian and return the music event type with higher probability.

It is important to remark that before training the model, some operations have to be applied to the input data such as subcompositon and additive log-ratio transformation (See section 4.3.2).

2.4 Symbolic Music Representation

In this section, a set of machine-readable music formats are explained. Notice that I would only explain the ones that have some relation with my TFG and that there are many more. This is one of the challenges in the MIR community, not having a standardized/generalized way of representing music. Furthermore, the idea of a

standard machine-readable music format could be misleading as there are music such as Indian ragas which are not symbolically represented in the same way as European Jazz or Pop music. Hence, all of this adds a little bit more complexity to MIR research.

2.4.1 Symbolic representation of musical chords

For being able to process automatically a music score, the music notation needs to be represented in a machine-readable way. Music Critic (MC) has decided to follow a standard defined in [15] which encodes different musical events. The notation represents notes/chords in the following way

$$theoretical\ root : (degrees) : (duration)\ /\ inversion\ root$$

Theoretical root represents the root of the note/chord while *inversion root* refers to the real physical root of the sounding chord, allowing the representation of inverted chords. For example, a C major chord in the first inversion, C would be the *theoretical root* while E the *physical root*. Rests are represented with an *N* and to represent linked notes a *X* is used.

Degrees represent the frequencies sounding together with the root note. The value would be:

- (1) for a single note.
- (3,5) or *maj*for a major chord and (b3 , 5) or *min* for a minor chord.
- (5) for power chords.
- (1,3) and (1, b3) for major and minor thirds respectively.

Duration represents the time length of the note/chord in terms of division per quarter note. Quarter notes would be represented with a 4, half notes with a 2, eight notes with an 8, sixteenth notes with a 16 and so on.

All of this note/chord information plus beats time position, time signature, duration of the score and other metadata such as the title of the song, it is later encoded in a *json* string. This information would serve as the Ground Truth for the assessment of a musical performance. Below an example of an annotation:

```
{
"parts": [{
"name": "Guitar",
"beats":[5.00, 5.54, 6.10, 6.66, 7.20, 7.76, 8.33, 8.87],
"chords": ["| N::4 B:(1):8 G:(1):8 X::4 A:(1):8 E:(1):8 |A:min:1"]
"duration": 9.40,
"metre": "4/4",
"title": "Simple example",
"tuning": 440
}
```

2.4.2 Music XML

Music XML [1] is a popular standard open format for exchanging digital sheet music. Musical data is represented in a hierarchy. There are two hierarchies:

- *score part-wise* where we have score information divided between different parts (number of different staves in the score) and inside each part we have all the measures information. Imagine a score which have a Guitar and Trumpet staves and that the length of the song is N bars. The XML file would be similar to:

```
<part id="Guitar" >
    <measure number="1">
    ...
    <measure number="N">

<part id="Trumpet">
    <measure number="1">
      ...
    <measure number="N">
```

- *score time-wise* where we would have the previous hierarchy inverted. This could be interesting when having different voices in the same stave.

At the beginning of each file there is a header containing metadata about the song represented and also a list of all the parts present. After specifying which hierarchy is used, it only remains filling up the measures/part information with the note/chord representation. Music XML uses the following nomenclature:

[1] https://www.musicxml.com/

- Single Note (e.g: quarter note G4)

```
<note default-x="27">
    <pitch> (Here the pitch is encoded)
      <step>G</step>
      <octave>4</octave>
    </pitch>
    <duration>24</duration>
    (Duration of the note)
    <type>quarter</type> (Type of note)
    <stem default-y="6">up</stem>
    (Direction of the stem)
  </note>
```

Notice: The number of divisions for quarter note is specified in each case. In this particular one, there are 24 divisions per quarter note. Hence, the duration of a quarter note would be 24 and for an eight-note would be 12.

- Chord (e.g: eight note Eb major chord)

```
<note>
    <chord/> (To identify chord)
        <pitch>
          <step>E</step> (root)
          <alter>-1</alter>
          (flat = -1 / sharp = +1)
          <octave>4</octave>
        </pitch>
    <duration>12</duration>
  </note>
```

Music XML standard can be understood by many important applications such as *Finale, Cubase or Band-in-Box* among many others. Furthermore there are python libraries such as *music21*[2] which make easy to parse *music xml* scores and develop code. Exactly this last library could be used for generating *lilypond* files, and could be interesting in a real application of Music Critic because for the moment is tedious to introduce new songs in the system.

[2] https://web.mit.edu/music21/doc/

2.4.3 LilyPond

Music Critic creates a visualization of the assessment results using *lilypond*. *Lilypond* [3] is a music typography software that focus on the aesthetics of the score and it is part of the GNU Project making it accessible to everyone.

In terms of usability, *lilypond* manages automatically spacing and distribution of the score, it could be used together with LaTeX, LibreOffice or HTML between others and there are some editors which make easier the creation of new scores such as *Frescobaldi* [4] or *Denemo* [5]. Below a simple example of code and its output to illustrate how this software works:

```
\clef "treble"
\numericTimeSignature\time 4/4 \key g \major \break | % 1 (line 1)
\tempo 4=108  <e b' e' g' b e''>1  -\mf | % 2 (line 2)
d''2 e'4 g'8 g'8 \bar "|." %3 (line 3)
```

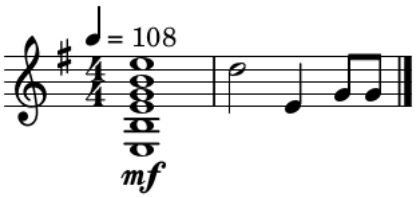


Figure 2.2: Lilypond example

We can see how chords and notes are written in *line 2* and *line 3* respectively. We indicate note duration in terms of division per quarter note (e.g. g8 would be a G single note eight-note). The symbols *'* and *,* indicate the octave of each note being *c'* the center *Do (C4)* and *g,* the *Sol (G3)* represented on top of the first line in bass clef . In *line 1* we indicate the time signature and the key of this little piece. Finally, in *line 2* the tempo is assigned and the *mezzo forte* dynamic is assigned to that piece.

In the Music Critic context, lilypond is used together with a templating language for Python called *jijna*[6] to assign a color to each note, which would illustrate how good a note/chord was played, and to plot a waveform aligned with the bars illustrating the tempo deviations of the student concerning expected onsets.

[3] http://lilypond.org/
[4] http://www.frescobaldi.org
[5] http://denemo.org
[6] https://jinja.palletsprojects.com/en/2.11.x/

2.5 Signal processing

Nowadays, there are many signal processing libraries, such as *MIRtoolbox*[18], *librosa*[20] or *Marsyas*[27], allowing to compute low-level and high-level music features that could be used for MIR applications and that are important in the MIR field when developing new algorithms. In this chapter, I give an overview on some of the signal processing tools and libraries related to my project.

2.5.1 NNLS Chroma Vectors

The Vamp [8] is an audio processing plugin system, developed at the Centre for Digital Music at Queen Mary University of London, used to extract symbolic information from audio data. The plugins have an open-source SDK and were published alongside Sonic Visualizer [9], a friendly end-user application used for analysis, visualization and annotation of music audio files. These plugins could perform tasks such as beat (*QM Tempo and Beat Tracker*)and onset positions detection(*QM Note Onset Detector*), key estimation (*QM Key Detector*), chords transcription (*Chordino*), estimation of the fundamental frequency (*Silvet*) or chromagram estimation (*NNLS chroma*) between others.

This last one, the NNLS chroma [19] [7], is important in our project as it is used when assessing pitch. What this plugin does is to convert an input audio signal to a chromagram divided in 12 bins, each bin representing a semi-tone. Hence, we obtain a vector lying on a 12-simplex where each dimension would correspond to a degree of the scale 2.1 and that would represent a musical event (e.g. major chord would have dimension 1, 5 and 8 with high intensity and a single note would only have dimension 1 with high intensity).

Dimension:	1	2	3	4	5	6	7	8	9	10	11	12
Degree:	I	IIb	II	IIIb	III	IV	Vb	V	VIb	VI	VIIb	VII

Table 2.1: NNLS Chroma vector dimensions and degrees

Our model uses NNLS chroma vectors to represent musical events, which are used to train the model and to assess incoming guitar performances. During this project, I plot some visualizations of chroma vectors representing for each dimension, the intensity of that particular degree (See 5.1).

[7] http://www.isophonics.net/nnls-chroma

2.5.2 Madmom

Madmom [4], is an open-source audio processing and music information retrieval python library, which puts emphasis on musically meaningful high-level features and that includes machine learning components for MIR tasks (in comparison with other libraries that only provide signal processing algorithms).

This library also comes with standalone programs covering many areas of MIR that can be used with the command line. Some of the most important are the *DBNBeatTracker*[5], which outputs the beat positions for incoming audio and that in Music Critic, can be used to get the ground truth beat positions for the *json* annotations (See 2.4.1). It is based on a neural network that has multiple modules, each one trained with different music styles, and a switcher which chooses the module that performs better for a specific recording. This allows the algorithm to be consistent through different music styles.

There are many interesting programs inside Madmom which use deep learning and machine learning techniques and that propose modern implementation of some classic signal processing algorithms. As an example, *CNNOnsetDetector*[24] uses a convolutional neural network to detect onset positions or *DeepChromaChordRecognitionProcessor* recognise major and minor chords using Conditional Random fields and deep chromagrams [], chromas robust to irrelevant interferences. Could be interesting to use deep chromas in Music Critic and try its performance (student recordings could contain many residual noises and sometimes NNLS chroma could cause confusions when predicting pitch-class set), despite having a limited application only on minor and major chords.

2.5.3 Essentia

Essentia [6][8], is a C++ library ready to use with python which provides a broad set of tools for audio analysis and audio-based music information retrieval and it was created by MTG. It is realized under *Affero GPLv3 license* and it is open-source.

Inside the pysimmusic library (See 2.5.4), Essentia software is much present. Apart from basic functionalities like loading or writing audio files with MonoLoader and MonoWritter classes, it is also used in more complex situations. For example to extract pitch melody information using PredominantPitchMelodia method based on the MELODIA algorithm [22], to calculate spectral picks using the SpectralPeaks method based on the algorithm explained in [1] or to estimate the tuning frequency of an audio using the *TuningFrequency* algorithm in Essentia.

[8] https://essentia.upf.edu/documentation.html

It has other interesting methods regarding onset detection such as OnsetDetection method, which can compute onsets using four different approaches based on high-frequency content, complex-domain spectral differences, spectral flux detection or energy flux [2].

2.5.4 Pysimmusic

Pysimmusic is a private MTG python library providing signal processing algorithms used to analyze musical performances. It is the core of Music Critic (See section 2.6.1) and it is also under development. It can be found in a private GitHub repository where only members of MTG have access to it.

Between others, this library is in charge to read the annotations, calculate the NNLS chroma vectors (See section 2.5.1), predict the position of the onsets, calculate tuning deviations and generate the visualizations and results for the assessment.

At this moment, the library uses two models to assess musical performances:

- *picking workflow* which evaluates picking (single note) exercises.
- *strumming workflow* which evaluates chord exercises.

The MTG has created a hybrid version of the two models which is under development (See 4.3.1), able to assess a music score combining chords and notes. The model is able to recognize:

1. 12 pitches: *C, Db, D, Eb, E, F, Gb, G, Ab, A, Bb and B*
2. 4 kind of events (*single note, power chord, major chord and minor chord*.

During this project, I make use of methods and algorithms implemented inside this python library.

2.6 Music Education

To learn how to play an instrument, it is important having a good music teacher. Due to economical or social situations, this is not accessible for everybody. Using online tools for music education would improve the accessibility of people in music education and the use of MIR algorithms would help on this objective [17]. Furthermore, another challenge is to provide tools that give proper feedback that help an individual improving their instrument skills. Nowadays, there are some

companies which started to build some applications aimed to learn an instrument and to help doing online musical classes such as *Yousician*[9], *Aloha*[10] and more recently *Fender play* [11] but none of them give concrete feedback nor offer free service.

In this section, I present Music Critic, software under development that aims to provide meaningful feedback to musical students, working as a support of a music teacher.

2.6.1 Music Critic

Music Critic (MC)[12] is an online platform tool that aims to produce meaningful feedback of musical performances for educational purposes. It was designed with the idea to scale music education on MOOC (Massive Open Online Courses) [7], first tested with voice recordings and more recently used in the MOOC "Guitar for Beginners" by Berklee College of Music.

It is built using signal processing and machine learning techniques which, as explained in [11], produce a pipeline that estimates pitches, onsets and tunning. These measures later are used to produce a final assessment of notes/chords accuracy, rhythm (closeness to the metronome) and tuning/intonation. A grade from 1 (being bad execution) to 4 (being excellent execution) is assigned to each of them plus an overall score which reflects the general quality of the performance and that depends on the three above mentioned features (also from 1 to 4).

The idea is that students play an exercise, previously uploaded by the teachers/organizers of the MOOC course and correctly displayed by the web API of the system, from their home recording themselves with headphone/computer mic and a backing track/metronome as a reference. MC automatically provides feedback to them.

The system, despite being so promising, is still under development. There are still many limitations in terms of the recording setup, the room where the student is located, the internet connection, the instrument used, the difficulty of the musical piece and giving proper feedback which sometimes is subjective. In section 4.3.1, the pipeline followed to assess pitch is explained.

[9] https://yousician.com/
[10] https://elk.audio/aloha/
[11] https://www.fender.com/play
[12] https://musiccritic.upf.edu/

2.7 Conclusions

In this chapter, we have gone through various concepts useful for the understanding of my project. We revisited theory about compositional data and explained how it applies to this project, which probabilistic models are used to assess pitch, different formats of music annotations for being able to transcribe audio content in a machine-readable format and signal processing and software related to MIR and used in this work context.

Chapter 3

DATASETS

3.1 Introduction

In this chapter, I am going to talk about the different data collections that have been used in this project. I will be presenting some of the main characteristics for each of them and the building process. In tune with data transparency and fairness [13], this chapter must serve as a datasheet for the *Five Guitar dataset*. My principal objective is trying to underline all the assumptions and decisions I made and the purpose for which is created for other people using it being well informed about its characteristics. Also, considering other datasets for MIR research and good practices when building a dataset[25], I will build an extended dataset that can be used for other researchers.

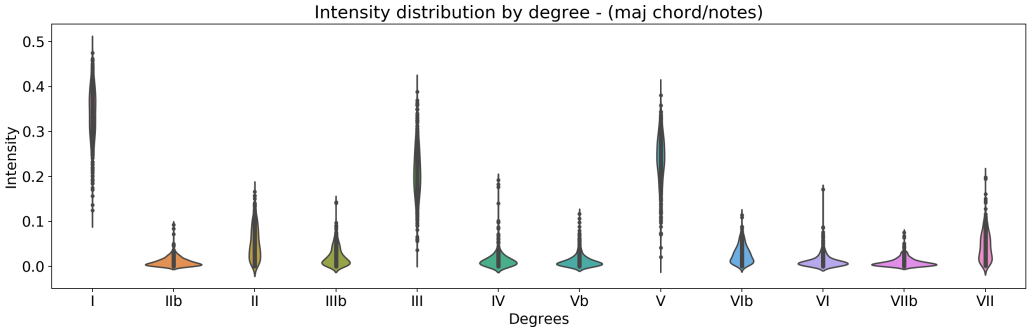
3.2 Basic Guitar dataset

Music Critic is an online educational tool (See section 2.6.1). On the Music Critic website, there are five different exercises, students can submit their guitar performances for the system to assess them. As students feed audio to the web, they are stored in a server storage system. MTG researchers rated the 233 available submissions in the server with a score going from 1-4 regarding to pitch, rhythm and tuning. Furthermore, some MTG researchers also provided 6 extra recordings, which belonged to 6 different exercises and had their own annotations associated (lilypond and json files (see sections 2.4.1 and 2.4.3). These achieved the maximum grade in the above-mentioned fields. The student submissions with higher grades in all the fields plus the recordings of the MTG researchers were used to create a dataset which I called *Basic Guitar dataset*, used to train the Music Critic pitch assessment model (See 4.3).

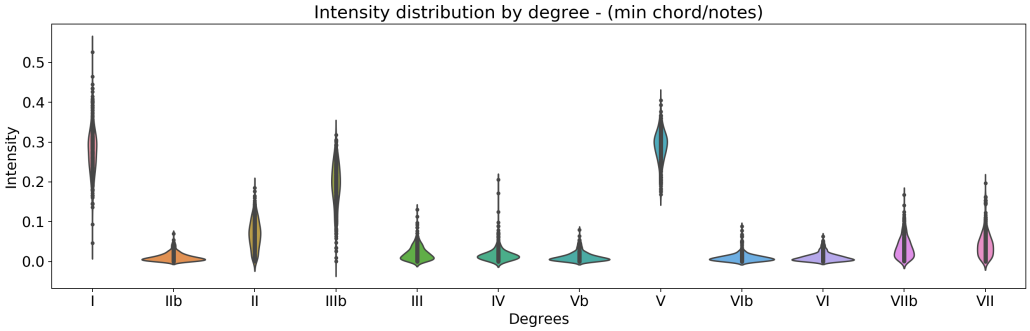
The dataset contains in total 86 guitar performances, with varying durations. In total it has 1810 NNLS chroma vectors from which 411 are major chords, 307 are minor chords, 838 are single notes and 192 are power chords. In section 5.2, you can see the appearance of all the chromas.

Figure 3.2 shows the distribution per degree for each chroma vector kind. As you can see, chromas have in general higher intensity for the relevant degrees of the particular musical event which are representing. An interesting thing we can observe is the high variance in the III and V degrees of major chords, which sometimes have intensity near to 0 despite being present in the actual performance of the student. In the experiments part, we will study what can produce this phenomenon. Minor chords have a similar effect with the IIIb degree, and single notes with the I degree. With single notes, we observe a large variation of all the other degrees, representing noises or harmonics (degrees III and V are the most prominent). Power chords seem to be the ones with better characterization as degrees I and V, the relevant ones, are in general more intense than the others.

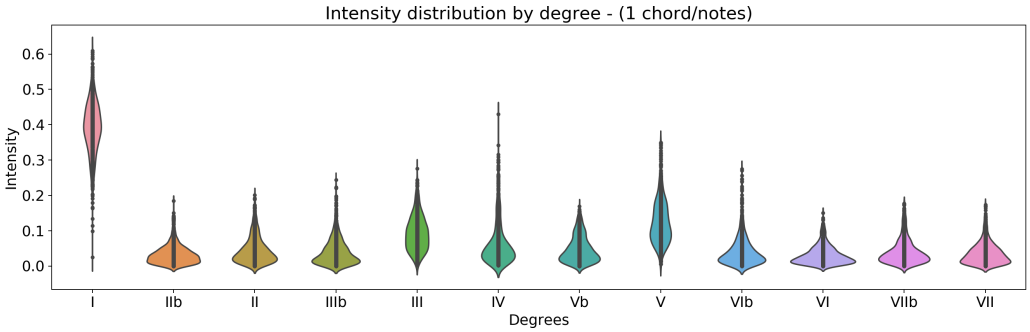
The dataset is property of MTG. Hence, it is not open-source and there is no possibility of working with it without approval of MTG. It could be used for different MIR tasks such as the construction of Automatic Transcription algorithms, score following algorithms and onset detection methods.



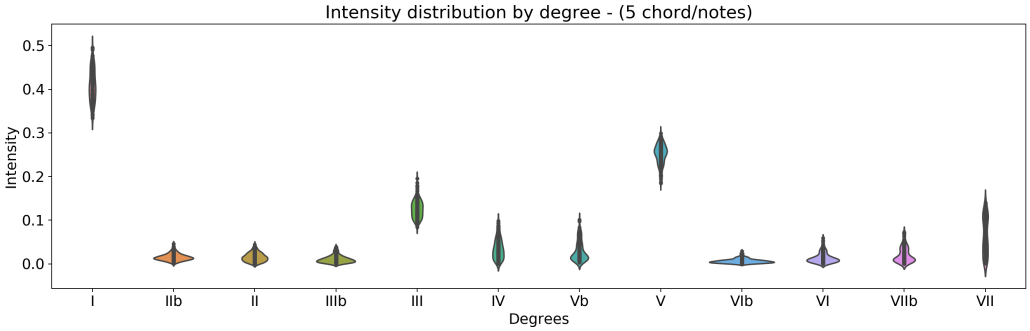
(a) Major chromas



(b) Minor chromas



(c) Single notes chromas



(d) Power chords chromas

Figure 3.1: Basic Guitar dataset chroma intensities by degree.

3.3 Impulse Responses datasets

The impulse response dataset contains all the impulse responses I used to generate the augmented dataset part in 3.4. It could be separated into four batches of data:

- Melda Production Box, with 18 simulations of guitar cabinets.
- Melda Production Microphones, with 10 simulations of microphones.
- Melda Production Rooms, with 34 simulations of rooms reverbs.
- MIT Acoustical Reverberation or MIT Reverbs, with 270 simulations of room reverbs.

The *Melda Production* impulse responses have been obtained from the publicly available *MAudioPlugins* pack [1]. Specifically, they come from *MConvolutionEZ* plugin, which has various impulse responses simulating guitar cabinets, rooms, microphones, plates and halls. The MIT Reverbs dataset has been obtained from [2], and it was created for studying purposes [26]. The aim was to measure impulse responses from day to day places. The team responsible for obtaining the dataset, tracked 7 volunteers over 3 weeks with text messages sent in randomized times, asking them for the place where they were at the time the message was sent. Using this information, they recorded 271 impulse responses of 301 locations. The recordings were made with 1.5 meter spacing between speaker and microphone simulating a conversation and each impulse response filename has a prefix representing the number of text messages that they received from that particular location. In the study they performed, they give more detailed statistics of impulse response properties and the creation process.

These impulse response sets are pretty useful. Apart from being publicly available, they allow creating larger datasets, that with music and sound are difficult to find as the collecting process could be tedious and there are many variations difficult to represent. On the other hand, they could be counterproductive because synthetic data may not reflect the reality with total security. The impulse responses can be downloaded in the link below [3].

[1] https://www.meldaproduction.com/downloads

[2] https://mcdermottlab.mit.edu/Reverb/IR_Survey.html

[3] https://drive.google.com/file/d/1F37HKCLSSQfanLZT-jxTR8xCCTjqtPM2/view?usp=sharing

3.4 Five guitar dataset

The *Five guitar dataset* contains various guitar performances and its annotations in json (2.4.1) and lilypond (2.4.3) formats created specially for this project. The performances have been recorded with 5 different guitars: a *Fender Player Telecaster*[4], an *Ibanez ART 120 with EMG pickups*[5], an *Epiphone Joe Pass Emperor-II PRO*[6], a *Larrivée OM-40*[7] and finally an *Eastman E1OM* [8].

Remark that the dataset do not contain errors in terms of pitch and rhythm, as we use it to test if the guitar performance is well transcribed by the system. The dataset is divided into two big parts: the Raw and the Augmented. Following an explanation of both.

3.4.1 Raw dataset

This part of the dataset contains 30 guitar performances of 6 different guitar songs (5 recordings/song, each of the 5 recordings with a different guitar) recorded simultaneously from 3 different setups (DI, mobile mic and computer mic) and annotations for the 6 songs. In total there are **90 recordings**. The audios can be found in [12] and the annotations can be found inside *test_data* folder in the GitHub repository of this project (See section 4.4.3).

The aim of recording the performance from different sources was to represent various contexts in which students can record themselves when using Music Critic and then being able to make comparisons between them.

The songs were chosen using as reference external company guitar books dedicated to music education[9]. From there, I transcribed manually the annotations for the following songs:

- Lily Was Here (Lily)
- Mountain At My Gates (Mountain)
- Where Did You Sleep Last Night (Where)
- 20th Century Boy (Century)

[4] https://shop.fender.com/en-ES/electric-guitars/telecaster/player-telecaster/0145212515.html?rl=en_US
[5] https://ibanez.fandom.com/wiki/ART_series
[6] https://www.youtube.com/watch?v=BsWgVx9ioaM&ab_channel=OfficialEpiphone
[7] https://www.larrivee.com/products/om-40-legacy-series
[8] https://www.eastmanguitars.com/e1om
[9] https://www.trinitycollege.com/

- Runaway Train (Train)
- Hole In My Shoe (Hole)

The audios performances are named following the pattern:

SongName_GuitarUsed_Tempo_RecordingSetUp.wav.

The setup used a *Yamaha Audio Gram 6* audio interface and *Reaper*[10] to record guitar from direct input (DI), a *Huawei P30 lite model MAR-LX1A* to record guitar from a mobile and a *DELL Inspiron 13 5000* to record guitar from a computer. To record acoustic guitar in DI, I used a *Fishman Neo D Single Coin* on Larrivée and *Seymour Duncan Seth love* on the Eastman connected to a *NAP-5 Stageman Floor* acoustic preamp going then to the audio interface. The computer and mobile were positioned approximately 3.8 cm from the player position (measured from the twelve fret of the guitars). The electric guitars were plugged into a *DV Mark Little Jazz* guitar amp which has a DI output, that was connected to the audio interface. The amp was located at a distance of 114.5 cm from the mobile and the computer.

In total, this Raw dataset contains 11355 NNLS chroma vectors from which 1290 are major chords, 1080 are minor chords, 5835 are single notes, 780 are power chords, 540 are major thirds and 660 are minor thirds. Figure 3.2 shows the degrees intensities distributions for each chroma type.

We can observe chroma degrees having more variance as now there are more chroma examples. Same characteristics that in 3.2 are found for major, minor, power chords and single notes but with more variation added. Notice that in this new dataset, there are major and minor third intervals that were not present before. Figure 3.3 shows how third intervals chromas are distributed. A major chord, when having V degree with low intensity, is similar to a major third and the same with minor chords and minor thirds intervals. This could lead to confusions studied in the experiments part.

This dataset is useful for any type of algorithm that needs audio and annotations, such as automatic score following, automatic pitch transcription and automatic chord estimation algorithms.

3.4.2 Augmented dataset

This part of the dataset is obtained from the Raw part. It contains the direct input line recordings convolved with the impulse responses from 3.3. This process allows us to do more experiments and comparisons.

[10] https://www.reaper.fm/

There are six different augmented datasets:

- Melda Production Box, containing 540 recordings obtained with Melda Production Box impulse responses.
- Melda Production Microphones, containing 300 recordings obtained with the Melda Production Microphones impulse responses.
- Melda Production Rooms, containing 1020 recordings obtained with Melda Production Rooms impulse responses.
- MIT Acoustical Reverberation or MIT Reverbs, containing 1111 recordings obtained with MIT Reverbs impulse responses.
- SOX Effects, containing 120 recordings obtained with *PySox* library [3][11] and methods *flanger, chorus, overdrive and phaser* from this library.
- SOX Filters, containing 570 recordings obtained with *PySox* library and methods *highpass and lowpass* from this library.

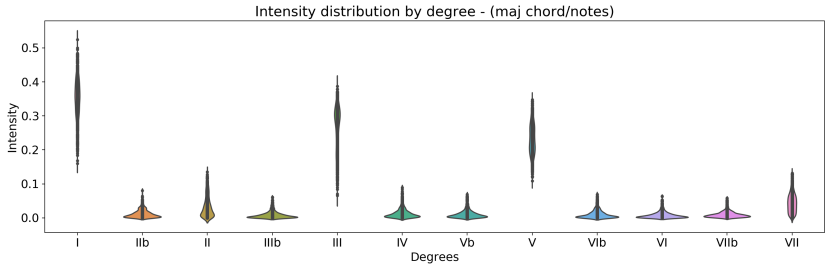
Augmented set	Major	Minor	Single note	Power	Maj. third	Min. third	Total
Melda Production Box	7740	6480	35010	4680	3240	3960	61110
Melda Production Microphones	4300	3600	19450	2600	1800	2200	33950
Melda Production Rooms	14620	12240	66130	8840	6120	7480	115430
MIT Acoustical Reverberation	15910	13320	71965	9620	6660	8140	125615
SOX Effects	1720	1440	7780	1040	720	880	13580
SOX Filters	5593	4707	25417	3380	2340	2860	44297
Total	49883	41787	225752	30160	20880	25520	393982

Table 3.1: Number of chromas for each augmented dataset

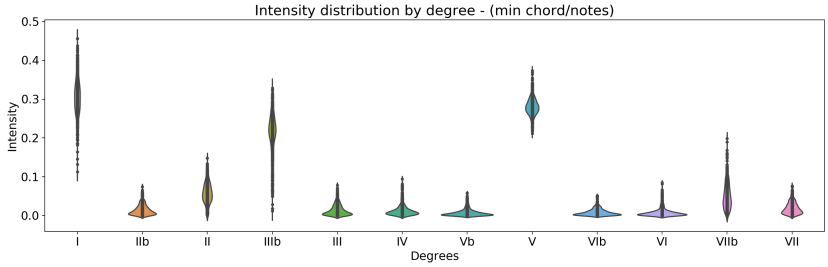
Table 3.1 shows the number of chromas for each pack. Observe that single notes predominate in the dataset and that major and minor thirds are less represented in comparison to other types of music events. In total, we are obtaining a large number of chromas that we now can use for experimental purposes.

A limitation of this dataset part is that despite having many audios, there are only 30 original performances. Hence, it could not be appropriate depending on the tasks performed. In my case, I want to compare how different contextual conditions affect the performance of the system. Hence, it is good to have the same guitar performance with variations to study the effect of each of them. Also, remark that all the augmented datasets can be reconstructed using the jupyter notebook *Build Augmented database.ipynb* stored in GitHub [12] inside *Code/* repository.

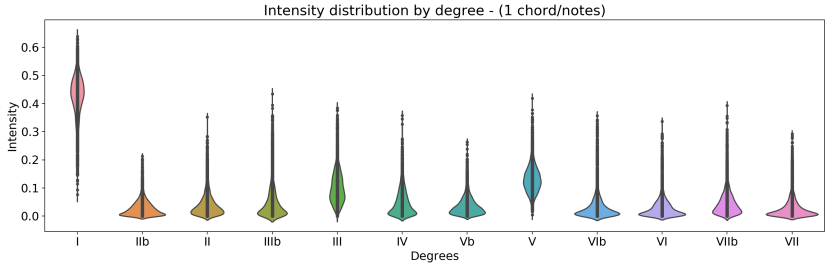
[11] https://pypi.org/project/sox/
[12] https://github.com/EduardVergesFranch/TFG_EduardVergesFranch



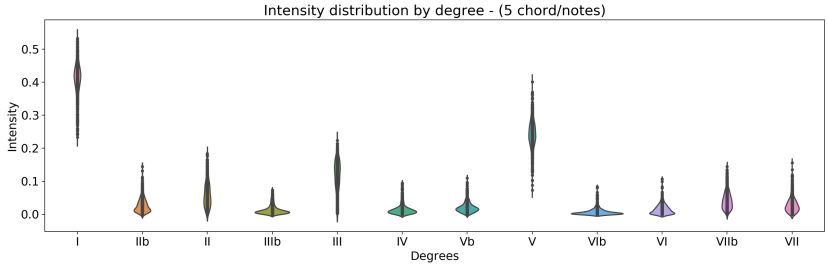
(a) Major chords



(b) Minor chords

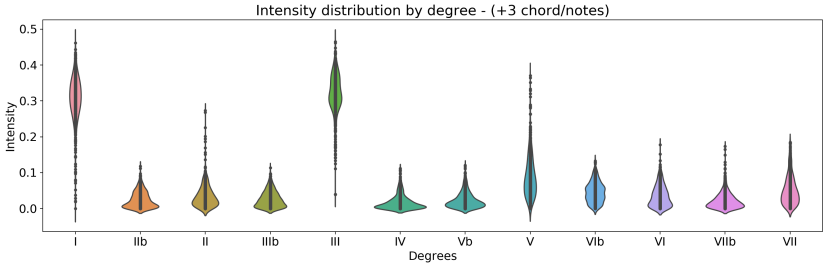


(c) Single notes

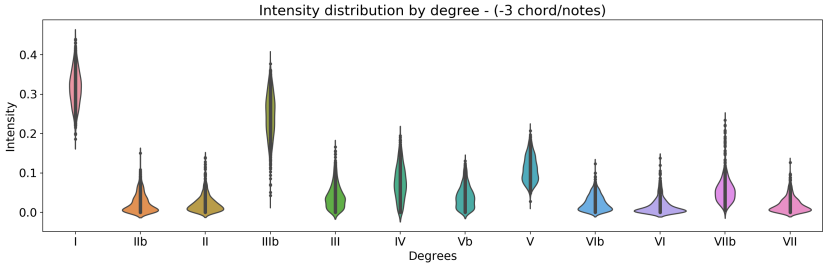


(d) Power chords

Figure 3.2: Raw part of the Five guitar dataset, chroma vectors intensities per degree.



(a) Major third chromas



(b) Minor third chromas

Figure 3.3: Raw part of the Five guitar dataset, third interval chroma vectors intensities per degree

3.5 Conclusions

In this chapter I presented the datasets used in this project and their main characteristics. Also, I provide the links to download them. Remark that the *Basic Guitar dataset* is not publicly accessible, as it is property of MTG.

Chapter 4

OBJECTIVES AND METHODOLOGY

4.1 Introduction

This chapter introduces the main objectives of this project and also states some of the basic principles used when assessing pitch in Music Critic, to understand what are the possible causes of the problems and the results obtained in the experiments and results part.

4.2 Objectives

As explained before, Music Critic has an educational purpose. Hence, it would have to provide reliable evaluations and advice for the student to be able to improve her/his musical skills. This means the system must be consistent and its algorithms must have a good performance and accuracy in many of the possible scenarios.

Taking these things into account, my final degree work will focus on evaluating part of the Music Critic system. We could divide the system workflow in three parts: pitch assessment, rhythm assessment and tuning assessment. I will focus on the pitch assessment part where I will be doing some tests and experiments to evaluate its performance and understand the errors experimented by the system.

I use the premise that if for a specific guitar performance, all the notes and chords (musical events) are played correctly, the system must be able to recognize and transcribe them perfectly, no matter the contextual conditions of the particular individual playing the guitar (room properties, recording setup, microphone

distance, the guitar used, pickup used...). Hence, I will be testing the automatic musical transcription performance of the model used in Music Critic.

To do so, I created dataset called *Five Guitar dataset* (See section 3.4) which allow me to test the performance of the model and try to improve it, as we provide more data. Also, using data augmentation I simulate different contextual variations such as rooms, effects, filters and recording setups. These are used to see how the model behaves in different situations.

In summary, my main objectives are:

- Construct a data set of guitar performances.
- Use data augmentation to simulate different audio variations (room acoustics, recording setup, effects and filters).
- Evaluate and improve the automatic musical transcription for the Music Critic system with previous data.

4.3 Pitch assessment in Music Critic

Here I explain how pitch assessment is modeled in Music Critic system. Also, the training process and evaluation of a new recording pipeline is stated.

4.3.1 Probabilistic Model

Pitch assessment in pysimmusic is a multi-class classification problem. The logic behind involves taking as an input a NNLS chroma vector representing a musical event and predicting a specific **pitch-class set**. A pitch-class set is a combination between a pitch (*C, Db, D,Eb, E, F, Gb, G, Ab, A, Bb, B*) and a **kind** (*major chord, minor chord, power chord and single note*. Hence, in total there are 12 pitches and 4 kinds which will create 48 different pitch-class sets. A **musical event** is a group of audio frames belonging to the same musical information (e.g. A set of frames representing C major chord is a musical event with pitch-class set C major and its kind would be major chord).

The modeling of pitch-class sets is divided into two independent distributions:

- Beta distribution, where the input is an amalgamation of the musical event relevant degrees for a particular NNLS Chroma vector. This is referred to as modeling the *correctness* of a musical event.

- Gaussian Mixture, where the input is a subcomposition of the musical event relevant degrees for a particular NNLS Chroma vector followed by an additive log-ratio transformation that produces a vector in an euclidean space. This is referred to as modeling the *balance* of a musical event. The Gaussian Mixture models have only one component, as the system learns a Gaussian Mixture model for each music event kind independently.

To make a prediction, the probabilities of each distribution are calculated and summed up. Finally, the pitch-class set with more probability is the one that would be returned.

During this work, I would study accuracies regarding the prediction of a **music event pitch-class set** and also **music event pitch kind**.

4.3.2 Training the model

To train the model, the system needs a set of NNLS chroma vectors with their corresponding kinds annotated. The chroma vectors usually are obtained from recordings together with annotations (2.4.1) from where they can be calculated and where the ground truth musical event kind is stated.

First of all, the chroma vectors are transposed to C root note. This means that the vectors would have to rearrange their dimensions matching the distribution that they would have had if the root pitch of the musical event had been a C note. Now, all the vectors are comparable and equivalent in terms of the note represented in each dimension.

After normalizing the chromas, the distributions explained in 4.3.1 are learned for each musical event kind that the systems is trained on. At least two vectors for each kind must be present in the training set. Before learning the beta distribution, amalgamation is applied to the vectors and before learning the Gaussian Mixture model the system applies a subcomposition followed by an additive log-ratio transform.

In conclusion, this produces a model that has in total two distributions for each music event kind learned.

4.3.3 Pitch assessment

When assessing a performance the input must be a recording made by a student playing the guitar. The system relies on that the recording is isolated from other instruments. The specific song that the student is playing must be accompanied by two annotations, one for visualization purposes in a lilypond file format 2.4.3 and

the other with the score in a machine-readable format explained in section 2.4.1 which is in *json* format.

Using the *MonoLoader* Essentia class, the audio is uploaded and the latency, previously calculated, is removed. Then, for each frame, the NNLS chroma vectors are calculated. Using the *json* annotation, the system groups all the vectors belonging to the same musical event and they are averaged using a *Hanning window*[21, p.2]. Furthermore, the musical events that corresponding to rests or not recognizable musical events are removed.

Then, all chromas are transposed to C and the probabilities for the two models in each kind are predicted and summed up. This process is then repeated for the same chromas transposed one semitone below. This would allow checking all 48 pitch-class sets probabilities.

Finally, for each chroma the pitch-class sets with more probability is returned as the prediction. For visualization purposes, the probability value of the ground truth pitch-class set (not the one with higher probability) would be used to color the notes in the score (using *lilypond* file). If the probability predicted is low, the color would be red, otherwise would be green. As more green the color, the higher the probability the model has given to the ground truth pitch-class set and this would mean students playing better that part.

Figure 4.1, summarizes the process of pitch assessment in Music Critic.

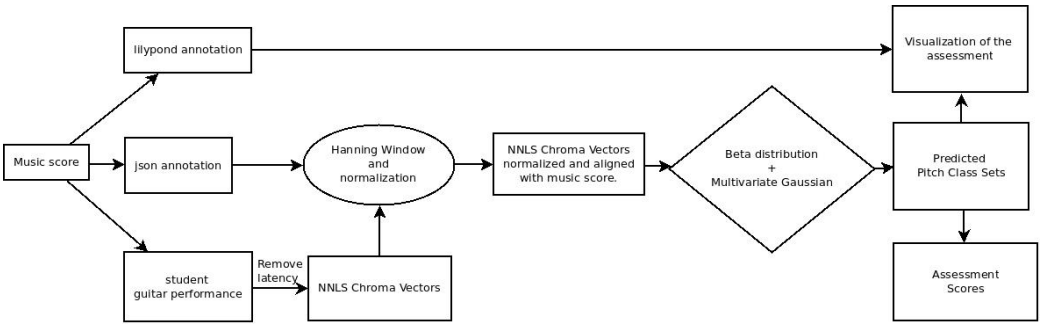


Figure 4.1: Pitch assessment pipeline

4.4 Experimental design

Music Critic is a system aimed to evaluate music students online. This implies users being students located in remote places playing a music piece in front of a computer, mobile or using a more professional setup. According to [28], it is difficult to evaluate the whole system, as there are so many variations to take into

account and I have to be aware in terms of validity and reliability when doing the experiments.

I will reduce the user target set to be more reliable, only considering guitar students. This would imply that my results would only apply to guitar performances. When interpreting the results, the songs chosen as train and test sets when doing the experiments, have to be considered also.

One of the most important objectives in a MIR application is that the user experience is the best. In this case, the best experience would be if the system can evaluate students properly. To assure this, the system must be able to transcribe perfectly a good guitar performance no matter the guitar used, the room where the student is located, the recording setup or the use of effects. To validate this part, I will evaluate the model accuracy for the same guitar performance in different contextual situations. This assures that all the accuracies ideally have to be the same for all the variations.

Also, I will test what type of biases we can find when training the model and its scalability. Remember that in all experiments I made references to repositories and the python scripts used that can be found inside the GitHub repository of this project (See section 4.4.3)

4.4.1 Experiments choices

To do the experiments I first defined a Baseline, which is the model trained with the same data that in Music Critic. This was possible because I had access to the data and I was able to replicate it. In many of the experiments I used the *Five guitar dataset* 3.4).

The idea is to evaluate the baseline performance on the above-mentioned dataset and then see how it reacts to different contextual situations:

- setup used by the student at the moment of submitting a recording on Music Critic.
- Effects, distortions and filters that could be applied to the guitar sound when performing a specific music piece.
- Different room acoustics, to see if the model has biases towards any specific kind of room design.

Also, I test the model design by observing what is the model performance according to the training set, what happens if I train the model with large amounts of data and what if I add more pitch-class sets to the model to make it more complete. In total I made 7 experiments.

4.4.2 Evaluation and metrics

As the main metric to evaluate the model performance I use classification accuracy according to pitch-class set prediction and also to musical event kind prediction. Studying musical event kind prediction allows having a simpler interpretation of the errors.

Furthermore, for studying and visualizing the errors produced by the system I make use of different evaluation tools such as:

- Confusion Matrix according to pitch-class set prediction and also to musical event kind prediction.
- Box plots to study how the model behaves in each contextual variation.
- Violin plots to study the intensity distribution of degrees in chroma vectors.
- Chromagrams to represent NNLS chroma vectors.
- Ternary plots 2.1 plus histograms to represent the model distributions learned and to detect possible biases.

The criteria I used is a mix between objective results according to accuracy, confusion matrix and box plots distributions, plus observational and subjective reasoning relating the experimental results with real applications of the model.

4.4.3 Reproducibility

To reproduce exactly my results, you would need access to *Pysimmusic* library which is property of MTG. I tried to track all the steps I have taken and store the exact sets and results in each of the experiments to be as much transparent as possible.

I created a GitHub repository [1] where all the code and data is available and can be reused. Inside *Code* repository you can find the following software components:

- Experiments.ipynb, which is the jupyter notebook where all the experiments have been done.
- Build_Augmented_database.ipynb, the code used to build the augmented dataset from the Raw dataset.

[1] https://github.com/EduardVergesFranch/TFG_EduardVergesFranch

- Renaming_Files.ipynb, the code used to rename the audio files for being able to use the loaders for the experiment part.
- MusicCritic_Template.py, which reproduces the Music Critic web interface. Used to get a standard assessment for a particular performance.
- test_utils/test_functionality.py, which contains various functions that format the input for the system to process it.
- test_utils/training_individual_chord_model.py, which contains the definition of the model used.
- test_utils/Training_Model.py, which contain various methods for loading the data to the model.

Also, in *test_pipeline_tools* there are python codes with methods used for various functionalities when doing the experiments. Inside *Databases* repository you can find:

- Augmented and Raw parts of the *Five guitar dataset* 3.4. Inside the augmented part, there are subdirectories for each batch of impulse response recordings. Augmented data is aimed to be generated from the Raw part with the Build_Augmented_database.ipynb code.
- IR folder with impulse responses used.
- Loaders with *json* files, with recording paths used in each experiment and that makes it possible to reproduce the chroma vectors I obtained.

Inside *Folds* repository you will find a json file with the folds created for the 5.7 experiment. Similarly, inside *MODELS/Cross-validation* you will find all the models generated and used in the project. Finally, inside *test_data* there are the *json* and *lilypond* annotations for the *Five guitar dataset*.

Lastly, I kept track of the loading times for each set, found in the appendix section of this project. In each of the experiments, I illustrate the characteristics of each batch of data and also show which python methods I used.

4.5 Conclusions

This chapter has stated the principal objectives of this project. It explains the pitch assessment in Music Critic and has discussed the design and reproducibility of the experiments.

Chapter 5

EXPERIMENTS AND RESULTS

5.1 Introduction

In this section, the baseline case is stated and the results for the experiments realized in the Music Critic pitch assessment model are exposed. All the experiments are done with *Experiments.ipynb* found inside *Code* repository in the GitHub of this project.

5.2 Baseline case

To test the automatic musical transcription performance of Music Critic, I have to define a baseline case from which I will build different tests and obtain some results and conclusions. As a starting point, I will use the model trained with the *Basic Guitar dataset* (See section 3.2). From now on, I will call *Baseline model* the model obtained when training the system with this particular dataset. Now, let's study some characteristics of this model.

The *Baseline model* has been trained with a total of **1748** NNLS chroma vectors, from which **411** are major chords, **307** are minor chords, **838** are single notes and **192** are power chords. It is only capable of recognizing four types music event (major, minor, power chord and single note). Below (5.1) some images representing the chromas learned where the y-axis represents the degrees of the chromatic scale and the x-axis represents the chroma vectors contained in the dataset.

We see how the different chromas have more intensity on the relevant degrees for each type. For example, in major chords the degrees *I , III and V* or *I, IIIb and V* for minor chords. We can observe the presence of **residual noises**, which correspond to degrees that despite not being relevant for a particular music event,

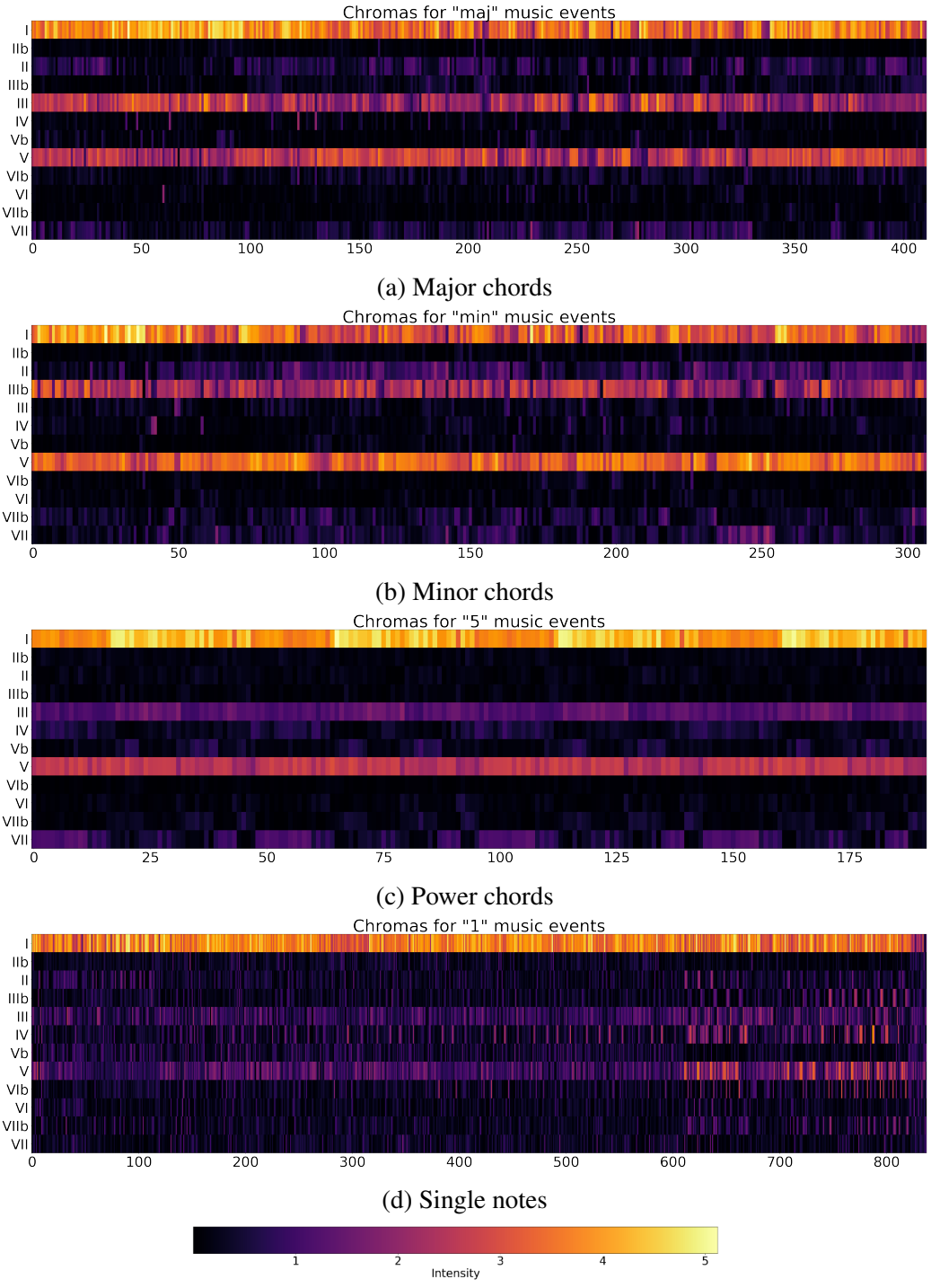
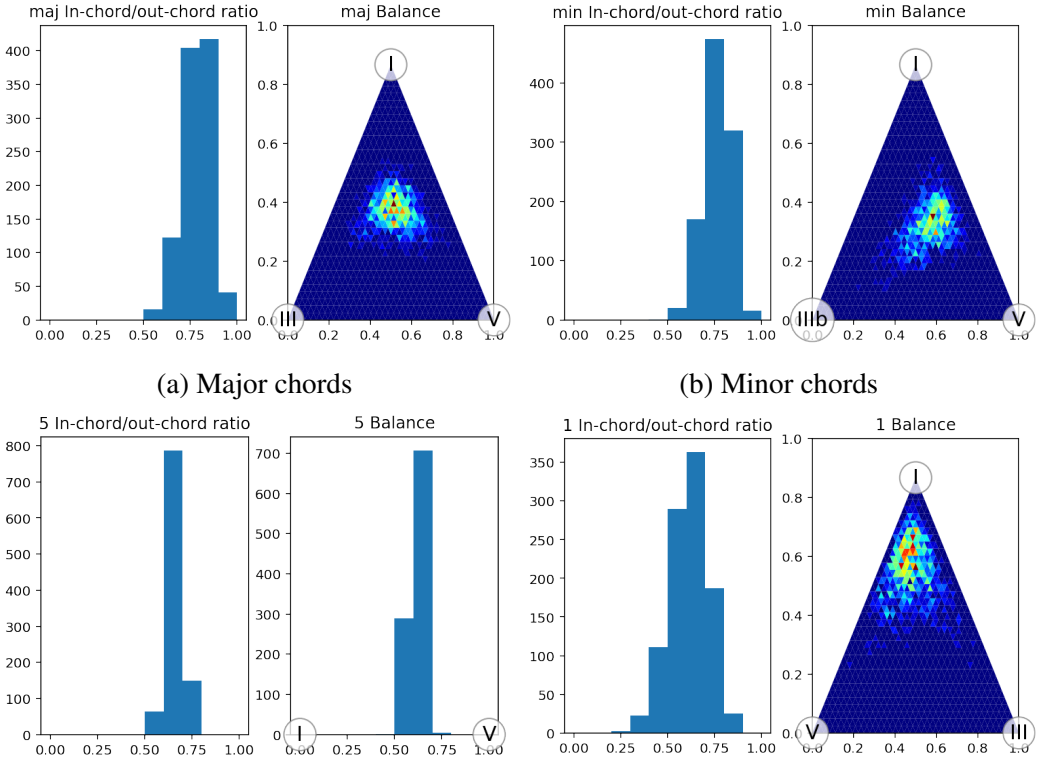


Figure 5.1: Baseline NNLS Chroma vectors

are present in the resulting chroma vector with smaller intensities (e.g. the *II* degree in major chords). These residual noises cause higher variance when learning distributions and also could lead to miss-classifications if their intensities are high.

Furthermore, it is interesting to observe that for single notes, the degrees *III and V* are considered relevant as they are most of the time present. This is a consequence of the harmonic series where the third and fifth degrees appear on the first harmonics. This characteristic could interfere in the differentiation between *major* and *single notes (1)* and even *power chords (5)* as they have similar degrees. With this set of chromas, the models in 5.2 have been learned.



(a) Major chords

(b) Minor chords

(c) Power chords

(d) Single notes

Figure 5.2: Baseline Models

For each sub-figure, the *beta distribution* is displayed on the left and the *Gaussian Mixture* on the right. We can observe that minor chords have a bias towards the *I and V* degrees which could lead to classify minor chords as power chords. When assessing single notes, most of the times the intensities of the relevant degrees only contribute to half of the chroma vector total intensity. This is due to a

higher number of residual noises that could come from harmonics or song-related characteristics (e.g. if the song has open string notes, these could superimpose one with another causing residual noises). We may find errors when classifying single notes as they could easily be confused with any other type. We see the high presence of residual noises and the variance of the chroma intensities in single notes in 5.3 .

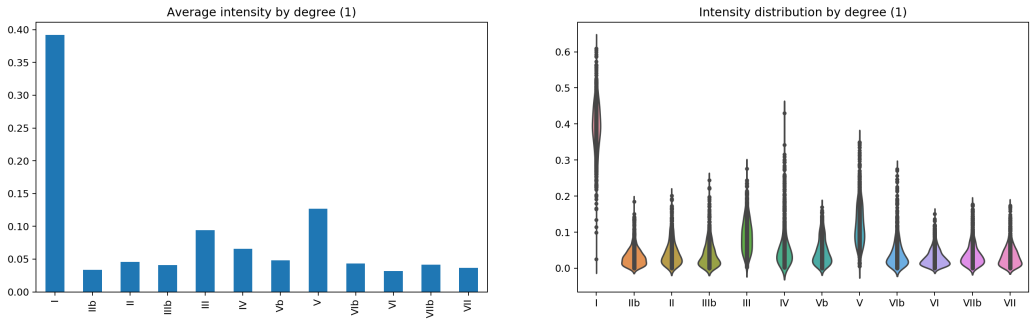


Figure 5.3: Degrees intensities for single notes in baseline case.

Major chords seem to be more balanced but have a high variance. Hence, it could be the case that a major chord is miss-classified as a power chord (e.g. due to particular fingering of a chord, third degree has smaller intensity and model predicts it as power chord). Finally, power chords seem the be the more balanced ones, but the presence of the third degree may lead to confusion.

In summary, we observe that all the chords have similar degrees and not a clear differentiation between them. Notice that it would be much easier to classify only between major or minor chords, as the presence of third or flat third degree would automatically characterize the musical event kind, but this would have a reduced application in real world. To prove these hypotheses, I will perform some initial tests. I will use the *Raw data* part of the *Five guitar dataset* (See section 3.4) as the test set for the *Baseline model*.

5.2.1 Not considering third intervals in the test set

To begin with, I will check the overall accuracy of the model with the *Five guitar* dataset. The recordings in the test set do not contain any errors regarding pitch and tempo and the chords and notes would have to be transcribed perfectly by the system. Hence, we expect a high accuracy since if the performance is correct the transcription of the model must be perfect. Also, it is important to say that the **test set contains major/minor third intervals**, which the model is not able

to recognize. **In this first test, we will note be considering these two types of events** when computing accuracy. We will add them late on.

Loading the test set using *Raw database* loader with *load chromas for dataset* method inside *test pipeline tools.SEGMENTS LOADER* class, containing 10185 NNLS chroma vectors from which 1290 are major chords, 1080 are minor chords, 780 are power chords and 5835 are single notes, lasts *196.97* seconds (the remaining chromas are major/minor third intervals which we are not going to consider for the moment). Predicting the pitch-class set, *Baseline model* obtains **0.867 accuracy**. It is a lower accuracy than we expected as it means that the transcription fails with a large numbers of chromas.

From confusion matrix in 1, different things can be observed. We see that single notes sometimes are confused with major or power chords because the presence of third and fifth degrees due to harmonics, creates uncertainty in the distinction of between these three kinds of events. Also, due to open string notes, mostly in Lily Was Here recordings in which two notes ringing superimposed, there can be errors in the prediction of the root (e.g. In 3 see Lily Was Here performance where there is the sequence G:1, played open string, followed by F#1 and this last one being classified as G:1) or chroma vectors superimposing with the next musical event (e.g. In 3 see F#1 classified as E1 which is the next musical event).

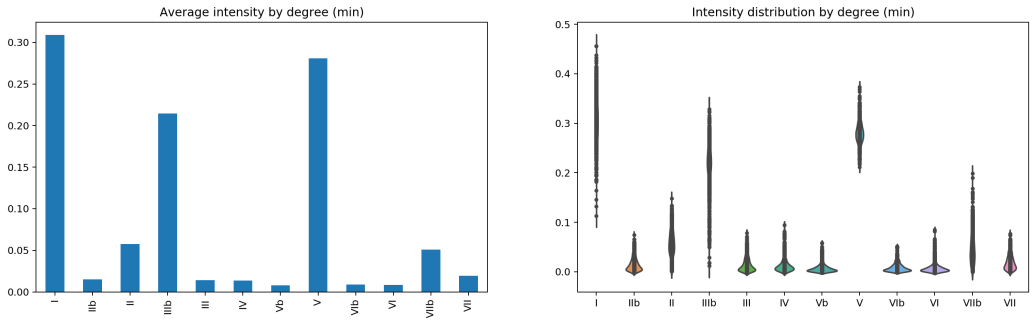


Figure 5.4: Minor chord degrees distribution in Baseline case test set

With minor chords happens that sometimes they are classified as power chords as a consequence of the bias observed in the model and due to minor chord degrees being badly balanced in the test set. In figure 5.4 we observe the high variance in the *IIIb* degree of minor chords. This explains minor chords being classified as power chords when the intensity of this flat third degree is low for various reasons, such as the particular fingering used when performing a musical piece for a minor chord (e.g. E minor chord played in open string position in a guitar, contains three

times the first degree, two times the fifth degree and one time the flat third degree, causing this last one not being prominent as the others).

A similar case could happen with major chords. For example, the E major chord played in open string position in a guitar only differentiates in one finger position to the E minor chord, and the occurrence of the degrees is the same despite the flat third being now a major third. Then the third degree is less prominent and the chord is classified as a power chord as we could see in the confusion matrix. This would imply having a lower recall in major and minor chords.

Furthermore, power chords have a small precision as the system would predict major, minor and even single notes as power chords. The power chord would not be classified as a minor chord because the harmonic series would make more prominent the third, fifth and first degrees causing confusions with major and single notes but not with minor chords as the flat third harmonic appears in the last harmonics and is not audible usually. In 2 is plotted the precision and recall for each music event kind and figure 5.5 illustrates the resulting confusion matrix.

The confusion matrix clearly shows that the model still is not able to classify perfectly all the chroma vectors, despite having a **0.885 accuracy** when predicting the music event kind. This tells us that the model is not able to differentiate well between music event pitch-class sets, as most of them share similar degrees and have similar distributions, and musical events could be superimposed causing bad predictions. In consequence, Music Critic is still not able to transcribe well-played guitar performances with high accuracy.

Finally remark that the system is using NNLS chroma vectors, which are not able to represent the octave the note is played on. This implies not being possible to determine if a note or chord is played with the correct fingering proposed in the music score. In an educational context, this is an important drawback as the main objective is to provide appropriate rubrics for the student to improve.

5.2.2 Considering third intervals in the test set

Let's see what happens if now I do not omit the major and minor third intervals present in the test set and more specifically *Where did you sleep last night* recordings. The reason for doing this test is to calculate the true accuracy that the *Baseline model* has used all the information in the test set. I expect less accuracy as these intervals would be random classified.

To the previous section test set chromas (See section 5.2.1), now are added 540 major third interval chromas and 660 minor third interval chromas which are represented in 5.6. The model has now a **0.765 accuracy**. I observe that third

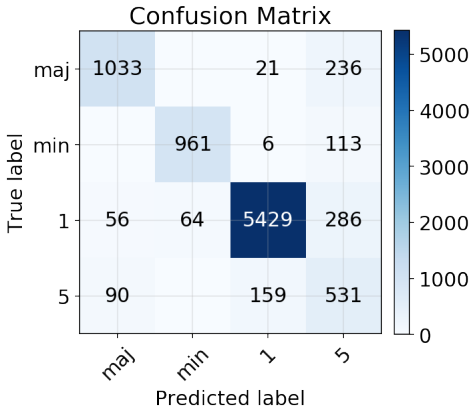


Figure 5.5: Baseline Music Event Kind confusion matrix

intervals have high degree variance regarding first and third or flat third degrees. In the major third case, we found that for some chromas the first degree has not even been detected. This could be a consequence of the guitar type or acoustic characteristics such as the recording setup or the acoustics. All this variance in the degrees could add more confusion when classifying. In later test, we will investigate what happens when training the model with major/minor third intervals.

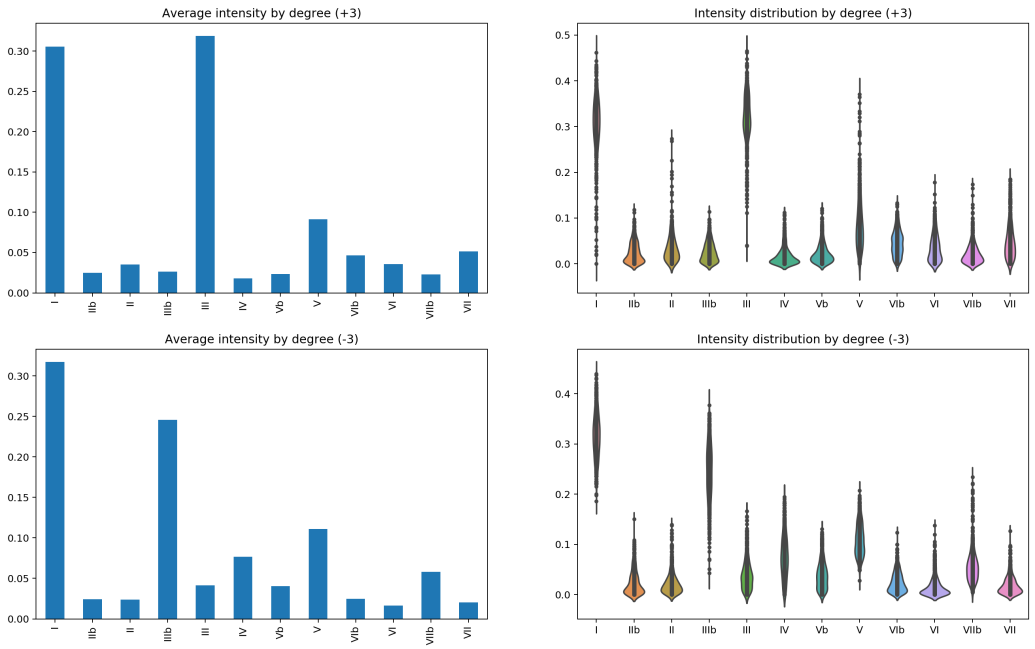


Figure 5.6: Chroma degrees intensities for major/minor third intervals

5.3 Recording setup

In this experiment, I will continue working on *Baseline model* and the same test set as section 5.2.1 with the same conditions but with more recordings, as I will use some of the augmented database recordings found in *MeldaProduction Microphones and MeldaProduction Box* repositories with different mics and recording sources simulations. The test aims to see how the recording source affects the model performance. Ideally, as the performances are equal through all the recording setups, the model has to perform equally for each of them

From the augmented datasets, I would only use a subset of impulse responses and name them in a particular way for better understanding of my results. In the *MeldaProduction Box* IR database, I will use:

- Laptop Speaker (named as SimComp).
- Mobile Phone 01 (named as SimMobile1).
- Mobile Phone 03 (named as SimMobile2).
- Radio 01 (named as SimRadio1).
- Radio 02 (named as SimRadio2).
- Voice Recorder (SimVoiceRecorder).

Radio and Voice impulse responses, are chosen as a simulation for old or bad quality mobiles and computer mics. And in the *MeldaProduction Microphones*:

- AKG C3000 25cm (named as AKG).
- Sennheiser e609 Direct (named as Snnheiser).
- Shure Beta52 Direct (named as Shure).
- TakStar Direct A (named as TakStar).

Loading the test set, using *Recording source test* loader with *load chromas for dataset* method in *test pipeline tools.SEGMENTS LOADER* python class, with 690 recordings lasts 1675.41 seconds. Contains 68885 chroma vectors, from which 9890 are major chords, 8280 are minor chords, 44735 are power chords and 5980 are single notes. The *Baseline model* has a **0.859 total accuracy** on this test set. Now let's see how the recording source affects the performance of the model (See table 1 to check results obtained).

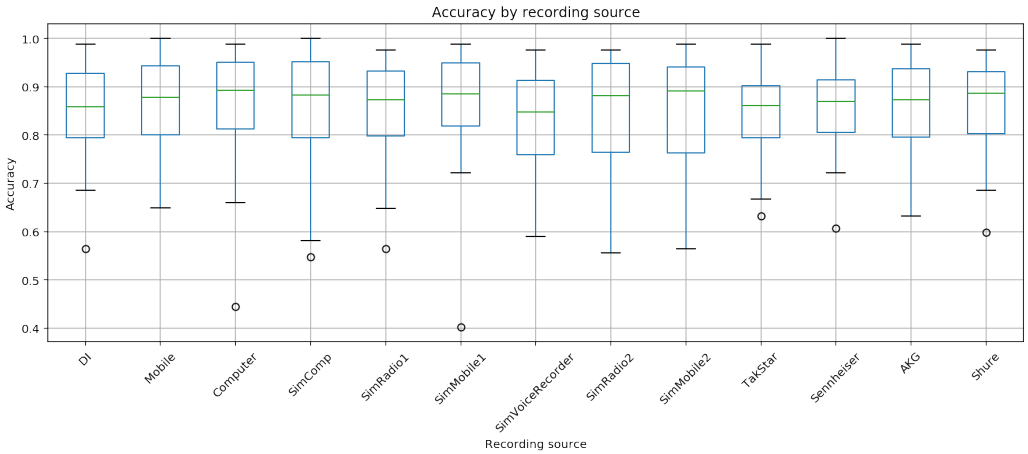


Figure 5.7: Accuracy box plot for different recording sources.

In figure 5.7, is observed that each recording source influences the model performance. Despite not being clear which is the best or the worst, as all have obtained pretty similar distributions, the *Computer* recordings have been the ones with higher average accuracy (**0.867**) and the *SimVoiceRecorder* the lower ones (**0.835**).

The interesting part comes when observing the outliers. Doing an IQR test, the outliers correspond to the *20th Century Boy song played with Larrivée guitar* performance. It seems that the model has a very bad performance for this specific recording and listening to it, we could hear that some strings are ringing in the background causing residual noises (See section 5.2.1). Observing each source recording accuracy for this performance, something interesting happens (See figure 5.8):

The *Mobile* recording source obtains much higher accuracy than the other sources. This represents the implication that the recording source has in the final result. The reason for this is that each recording has a different frequency response, boosting or attenuating some of the frequencies, causing NNLS chroma vectors to be slightly different. Since *Baseline model* is not good at differentiating some music event kinds and that in the recording there are residual noises, the predictions for each source diverge. As an example, you could look at figure 4 in the appendix, where the first five chroma vectors for Mobile, Computer and DI recording sources obtained with the Century Boy Larrivée guitar performance, are showed with the true note and the predicted one, to see how the model has behaved in each case illustrating what I just said.

In a real situation, this could cause a bias towards students having specific recording setups and guitars deriving to incorrect qualifications in extreme cases.

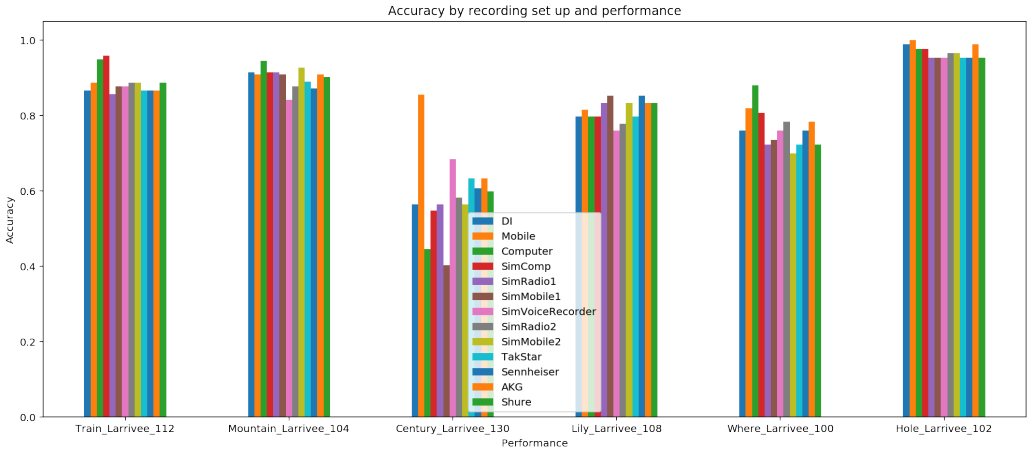


Figure 5.8: *Accuracy by setup and performance*

It is important to assure consistency on recording setups as the system is intended to work online, and each user would have its own material. Another option would be making students record themselves in a specific manner but this would make the system more invasive.

We see that different recording setups affect the model performance. Hence, we could test how different typical guitar effects, different filters (to simulate pickup position) or different room acoustics influence the final result.

5.4 Effects, distortion and filters

In this test, I will check how different guitar effects affect the *Baseline model* performance. After testing different recording sources, I expect effects to have a noticeable impact as timber changes. To develop this test, I will use data from *SOX effects* augmented database, where we have augmented versions for DI recordings corresponding to the same performance but with different effects applied.

Loading the test set, using *Guitar effects test* loader with *load chromas for dataset* method in *test pipeline tools.SEGMENTS LOADER* python class, with 150 recordings lasts 407.20 seconds. Contains 14975 chroma vectors, from which 2150 are major chords, 1800 are minor chords, 1300 are power chords and 9725 are single notes. The *Baseline model* has a **0.8627 total accuracy** on this test set. Surprisingly, the accuracy does not go down.

Looking at figure 5.9, we observe that the different effects behave similarly. It seems that DI recordings have a lower average accuracy. We can say that NNLS chroma vectors are more consistent with effects. It seems that phaser with **0.863**

mean accuracy is the one with a better performance followed by the chorus with **0.862 mean accuracy** (You could see all the results in table 2 on appendix section).

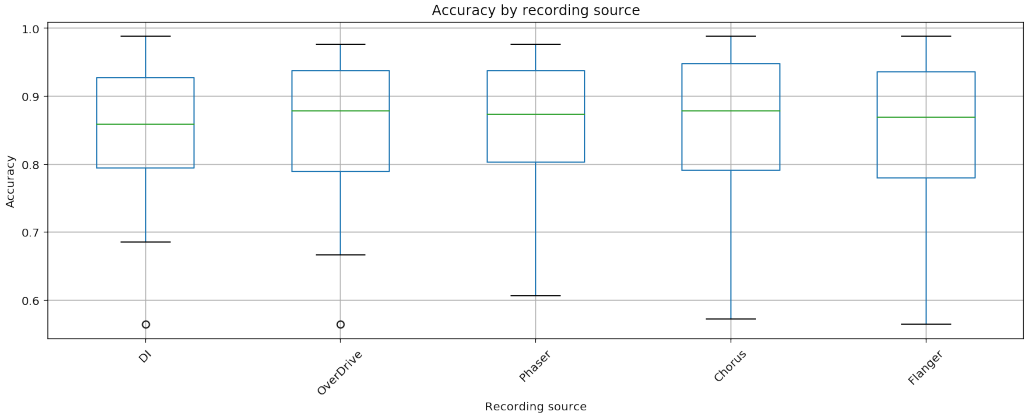


Figure 5.9: Accuracy box plot for different guitar effects.

Looking each individual performance, the accuracy for different effects vary with a **standard deviation of 0.099**. We can see how the system is much more consistent through effects in figure 5.10.

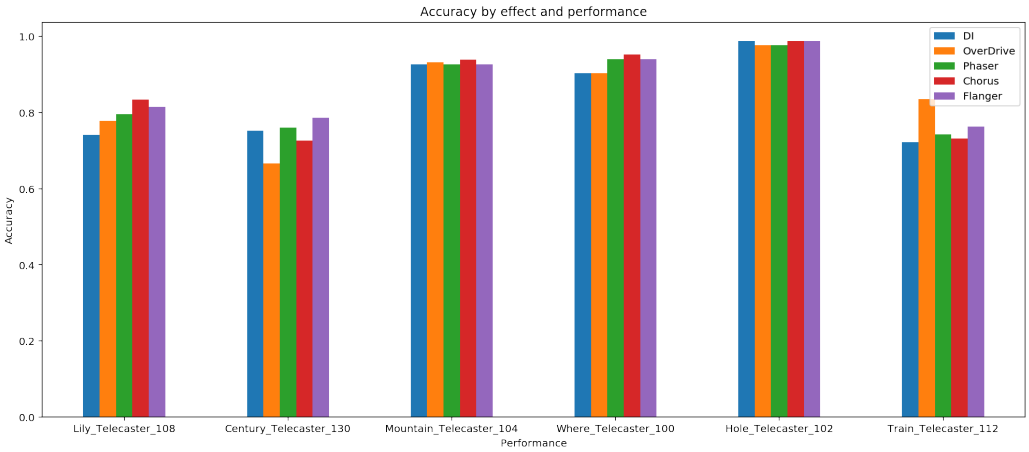


Figure 5.10: Accuracy by effect and performance

In general guitar effects could have a positive or negative impact to the prediction. It could be the case where frequencies that in DI recording had low intensity, after applying the effect get more present. If the particular frequency boosted is relevant for the musical event kind, the system may make a good prediction. Otherwise, it is only adding more noise to the vector.

The second test would be applying some EQ filters in the performances. This would simulate different pickups or pickup positions. For doing so, I will use the *SOX_Filters* augmented database as the test set, which has various recordings with different filters applied:

- *Hpass* applies a high pass frequency filter from the indicated frequency (e.g. Hpass500 applies a high pass from 500Hz).
- *Lpass* applies a low pass frequency filter from the indicated frequency (e.g. Lpass 1000 applies a low pass from 1000Hz).

Loading the test set, using *Guitar filters test* loader with *load chromas for dataset* method in *test pipeline tools.SEGMENTS LOADER* python class, with 210 recordings lasts 453.71 seconds. Contains 20965 chroma vectors, from which 3010 are major chords, 2520 are minor chords, 13615 are power chords and 13615 are single notes. The *Baseline model* has a **0.864 total accuracy** on this test set.

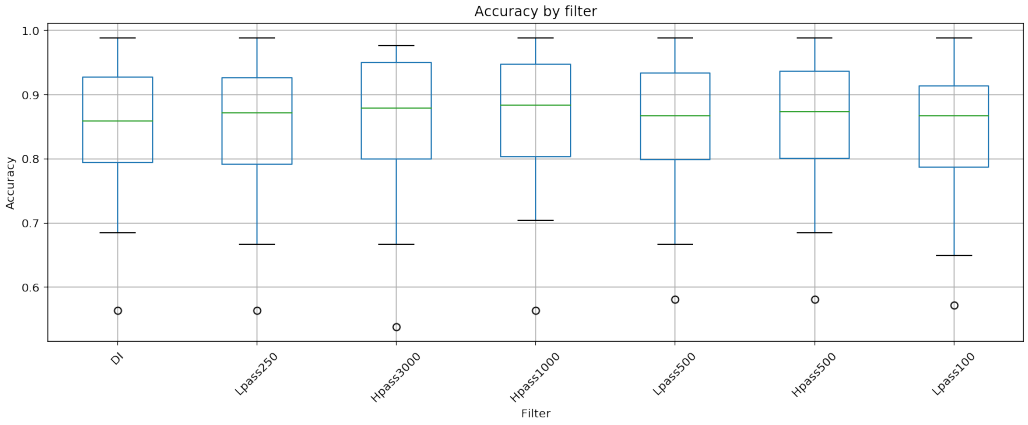


Figure 5.11: Box plot accuracy for different filters

Figure 5.11 shows that the model is also consistent through the filters. Despite some little variations in the predictions, there is not a filter that clearly works better than all. But, in general, it seems that high pass works in average better (See 3 for the results).

Looking for particular performances, we observe that now the variation is present but not as much as with the recording setups (See figure 5.12).

In a real context, using guitar effects or different pickups/pickup positions would influence the model performance. Users of Music Critic would use a large number of guitars with different specifications and effects and the system must assure consistency in the assessment and transcription. This would be a difficulty added to the model.

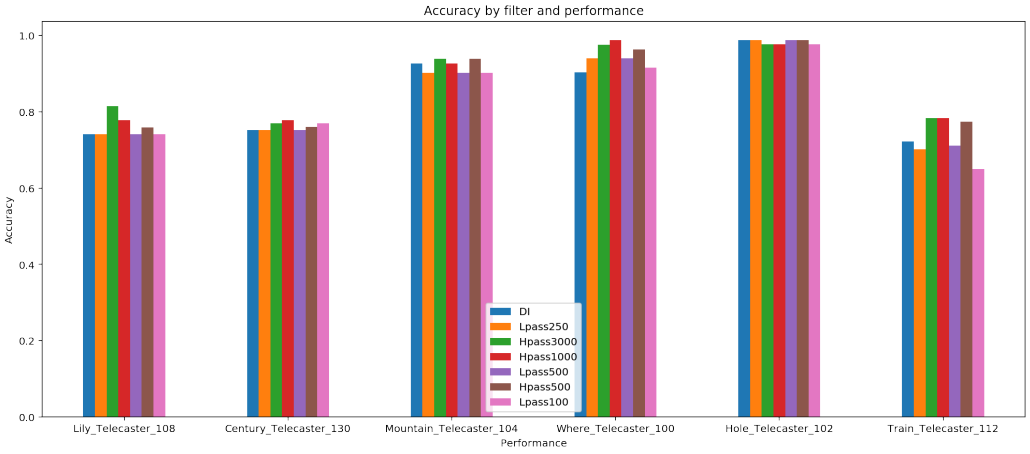


Figure 5.12: Accuracy by filter variations

5.5 Room acoustics

Pysimmusic is intended to be used for online education tools (such as Music Critic). Hence, the audios would be student performances, which they would record themselves from their houses. Then, it is interesting to see how the model performs depending on the room characteristics where the student would be recording. The ideal would be the model being consistent through different rooms but as seen with previous tests, guitar effects or recordings source affects the performance. I expect those room characteristics also would influence the result.

To do this test I will use *Melda Production Rooms test* and *MIT reverbs* augmented dataset as the test results. I would only use a subset of impulse responses. Following, a list of the impulse responses used and their correspondence in the results.

- Small 04 (Small 1), Small 05 (Small 2), Small tiled room (Small 3) and h008_Bedroom_35txts (Small 4).
- Living_Room (Medium 1), Medium_02 (Medium 2), Medium_05 (nMedium 3) and h027_Classroom_8txts Medium 4).
- Large 05 (Large 1), Large tiled room (Large 2), Large wooden room (Large 3) and h094_Campground_CabinLivingroom_2txts (Large 4).

Loading the test set, using *Guitar rooms test* loader with *load chromas for dataset* method in *test pipeline tools.SEGMENTS LOADER* python class, with 660 recordings lasts 1510.36 s seconds. Contains 14975 chroma vectors, from which 9460

are major chords, 7920 are minor chords, 5720 are power chords and 42790 are single notes. The *Baseline model* has a **0.8652 total accuracy** on this test set.

Figure 5.13 shows a box plot with the accuracies obtained for each room. According to different room properties, the system obtains different performances. The room with better average accuracy has been the Large 2 with **0.895 accuracy** and the worst the Small 1 with **0.823 accuracy**. The model does not seem consistent through different rooms (See table 4 for results).

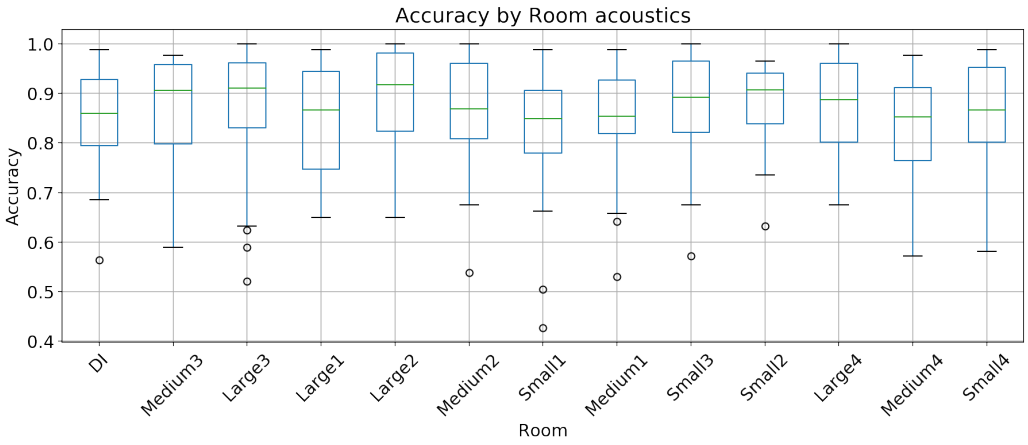


Figure 5.13: Accuracy box plot for different rooms.

Rooms performance have approximately a **0.106 standard deviation** each. It is interesting to see that the performance of the model for a particular room depends also on the musical performance as seen in 5.14, where there is not a room that works better or worse for all the cases. Instead, each room has a different accuracy for each performance.

In the appendix, there is a comparison between NNLS chroma vectors obtained for the same performance in different rooms (See 5). We see that each room has an effect on the frequency content and hence, to the final result. See for example the F#1 appearing at the last four chroma vectors, which is classified only correct in *Large 3* room. In *Small 1* room is classified as a B1, as it could be the case that, as the following note is an E1, the room makes the harmonics of E (G and B) more intense. Then, the musical event of F#1 obtains some residual noises from the next musical event. The same happens with room *Medium 3*, which is classified as an E1 for the same reason. Also, there is a comparison between NNLS Chroma Vectors obtained from different performances in each room 6 illustrating errors committed are different (e.g. the same F#1 as before, classified as Bmin or B1 depending on the performance).

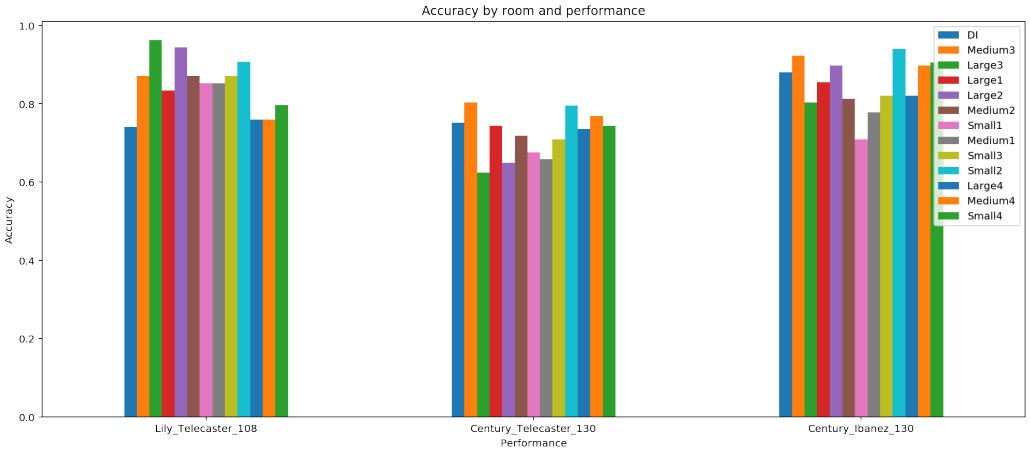


Figure 5.14: Room accuracy by performance and guitar

In conclusion, the room acoustics seem to influence the model performance. In a real world application, this could be an important limitation and an important thing to take into account when designing the software, as it could lead to different assessments for students that both have played well. Also, it could generate a model biased towards people able to practice and record themselves into specific spots where the model is better.

5.6 Model performance depending on training data

In this section, I will test how the model performance depends on the training data. I want to discover which kind of biases we could find when training a model. I would use the Raw database (*Raw database* loader) and see what happens when training a model with a specific song (set of recordings belonging to the same song) and testing with all the other recordings (belonging to other songs). In order to do this, I will use the *train test split filtered field* method in *test pipeline tools.SEGMENTS LOADER* class to filter train and test set by song name. Below are the loading times for each train/test split:

Each particular train set will be referenced with the name of the song used, and the same will apply to each model learned. The models can be found inside *RESULTS/SongWise_test* repository. Below, in table 5.2, the number of chroma vectors by musical event kind for each train set is showed. Observe that not all the train sets have all music event kinds, and this would have an effect on the model performance. Also a table with the accuracies obtained for each model(See table 5.3).

Song in the train set	Train set	Test set
Lily was Here	31.45 s	175.07 s
Mountain in My Gates	33.67 s	173.87 s
Where did you sleep last night	38.74 s	170.44 s
Runaway Train	34.02 s	183.00 s
20th Century Boy	30.95 s	184.87 s
Hole in my shoe	32.32 s	185.99 s

Table 5.1: Loading times for each data split.

Train set	Major	Minor	Single note	Power chord	Total
Lily was Here	15	135	660	0	810
Mountain in My Gates	30	45	2370	0	2445
Where did you sleep last night	240	240	585	180	1245
Runaway Train	540	510	405	0	1455
20th Century Boy	135	0	1020	600	1755
Hole in my shoe	330	150	795	0	1275
Total:	1290	1080	5835	780	

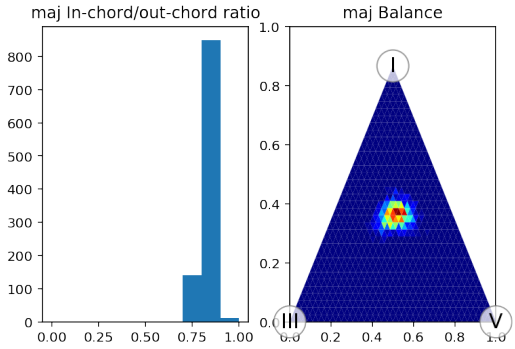
Table 5.2: Number of chroma vectors for each train set.

Model	pitch-class set	Music event kind
Lily was Here	0.8489	0.8683
Mountain in My Gates	0.7976	0.8197
Where did you sleep last night	0.8632	0.8884
Runaway Train	0.8517	0.8884
20th Century Boy	0.583	0.6062
Hole in my shoe	0.8158	0.8414

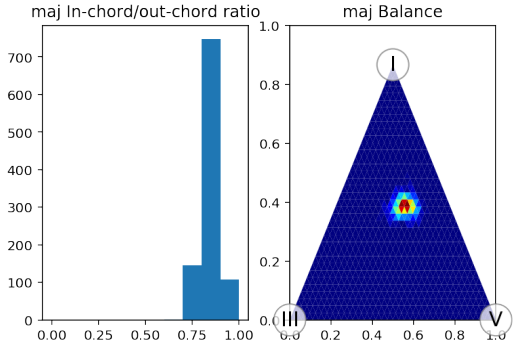
Table 5.3: Accuracies for each model by pitch-class set and music event kind.

First of all, the training data clearly influences on the models learned. For example, if we compare the *Lily Was Here* major model with the *20th Century Boy*, can be observed that the distributions learned are different (See figure 5.15). In the first one, the model seems centered according to balance but the second one is biased towards first and fifth degrees. Also, the first one seems to tolerate more variance than the second one which is more narrow. This would imply different probabilities for the same chroma vector depending on the model used and also, in an extreme case where the bias is greater a consequence of the recording conditions, the room properties or the recording source could imply less differentiation between pitch-class sets.

Another observation is the importance of representing all the pitch-class sets



(a) Lily Was Here model



(b) 20th Century Boy model

Figure 5.15: Major model comparison

in the test set when training the model. We observe that *20th Century Boy* does not contain minor chords, which are present in large number on the test set, resulting in bad model performance. For a real application, this is an important limitation, because the model would have to be trained on all the possible pitch-class sets found in the test set. This is a difficult task because as the number of songs in the test set increase, more pitch-class sets exist. Also, it seems more important to train the model with the most common musical events in the dataset despite not being trained in all the music event kinds, because the accuracy would be higher (e.g. Runaway Train model is not trained in power chords, but trained with all the other music event kinds more represented in the test set, and obtaining better performance than 20th Century Boy model).

Finally, looking at *Where did you sleep last night* model makes me think that training the model with a little amount of data does not seem to decrease a lot the total accuracy for much of the cases (comparing to *Baseline model*). This may means that the model performance would not depend on the amount of data used for training.

In conclusion, the model is sensitive to training data as observed. Then, the training set must be constructed carefully thinking about which songs have to be assessed. This is a difficulty added as the model is intended to be used as an online education tool and student recordings would vary depending on the recording conditions, causing different performances for the model.

5.7 Training model with large amount of data

In this section, I will test if training the model with a large amount of data improves the model performance or introduces more variance to the model. The hypothesis is that, if the system is trained with more chroma vectors, it would be able to catch more variations of a music event but the high variance would imply less differentiation between music event kinds. Hence, the accuracy would not improve and maybe decrease.

To do this test, I would use the *Melda Production Rooms Folds Test* augmented dataset simulating many recording contexts. The dataset would be divided into five-folds, and then I would perform a **5 fold cross validation** to obtain an estimation of the performance. Also, I would take a look at the models learned at each step and compare them with the *Baseline model* (See figure 5.2).

Table 5.4, represents the number of chromas for each split in the validation process. You could see all the models learned in figures 7,8,9,10 and11. Can be observed that the learned distributions are much narrow due to a high number of samples that define better the Gaussian distribution for each case.

Split	Major	Minor	Single note	Power chord	Total
1 Train	13741	11265	51295	5960	82261
1 Test	879	975	14835	2880	19569
2 Train	9422	8220	57111	7940	82693
2 Test	5198	4020	9019	900	19137
3 Train	12083	9741	50895	7580	80299
3 Train	2537	2499	15235	1260	21531
4 Train	12351	10888	51462	7160	81861
4 Test	2269	1352	14668	1680	19969
5 Train	10883	8846	53757	6720	80206
5 Test	3737	3394	12373	2120	21624

Table 5.4: Number of chroma vectors for each split.

After doing cross-validation we find that the model accuracy does not improve when adding more data to it. To obtain the reason why this happens, we are going

Model	pitch-class set
Baseline	0.87202 ±0.0279
MeldaProduction Rooms	0.8026 ±0.0306

Table 5.5: Mean accuracies and standard deviations for each train/test split compared to *Baseline model*

Model	Training error
Baseline	0.7820
Augmented Baseline	0.8064

Table 5.6: Training error

to compare the training error for the *Baseline model* and the *Augmented Baseline* which would be a model trained with the same data as the *baseline* but adding the *MeldaProduction_Rooms* augmented dataset to the training set. We obtain the training errors in 5.6.

We observe that adding more data does not seem to affect the training error. This means that the model expressiveness is limited and that has high bias as it is not able to differentiate better the pitch-class sets.

This lead to suggest some changes in the model. At this point, the model does not consider onset positions when calculating chroma and this information could be used to extract chroma from the onset position and not from the ground truth beat position annotation. Also, it is not considering previous events, and it could be useful to consider previous event spectral peaks to avoid residual noises.

5.8 Adding more pitch-class sets

Each song has its characteristics and this would imply that as the number of songs increases, the pitch-class sets present would also increase. At this point, the model is only able to recognize major/minor/power chords and single notes. I will test what happens if we add to the model the possibility of recognizing third intervals (as they are present in our datasets, but have not been considered during all the experiments) to simulate realistic situations with more pitch-class sets.

To do this test, I would create a model called *Thirds model*, with which I would use some recordings from *Raw database* when training, specifically *Mountain at my gates* ones that contain third intervals (*Mountain_Larrivee_104_Computer.wav, Mountain_Epiphone_104_Computer.wav, Mountain_Eastman_104_DI.wav, Mountain_Telecaster_104_Computer.wav, Mountain_Eastman_104_Mobile.wav, Moun-*

tain_Ibanez_104_Mobile.wav, Mountain_Larrivee_104_Mobile.wav) plus the data used for training the *Baseline model* . The rest of the *Raw database* recordings would serve as the test set. This would create a model that can recognize also major and minor third intervals. Then I would compare the *Thirds model* and *Baseline model* to see which one performs better on the same test set.

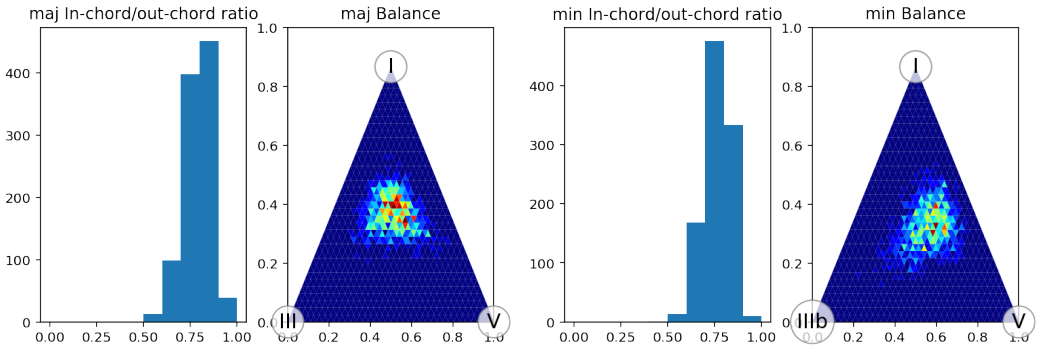
The *Thirds model* has learned the patterns in figure 5.16. A comparison between the models is shown in table 5.7. We could observe that the *Baseline model* performs better despite not being able to recognize the third intervals in the test set. This means that the model would perform badly as we add more and more pitch-class sets to it.

Model	pitch-class set	Music event kind
Baseline	0.7909	0.8081
Baseline with third intervals	0.7373	0.7446

Table 5.7: Comparison between model trained with/without third intervals.

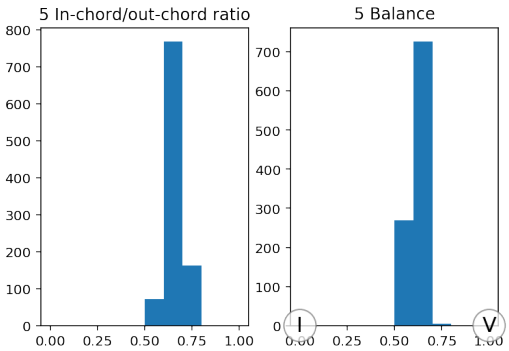
We can observe that in *thirds model*, major and minor thirds seem balanced but this would not imply a good classification. Major and minor thirds chroma have degree intensity distribution as shown in figure 5.17. It could be seen that the fifth degree also has an influence to the chroma vector. Then this could generate confusions between major thirds interval and major chord, or minor third interval and minor chord. If we take a look at the confusion matrix for each model (see 5.18) we observe this kind of errors are frequent in the *Thirds model*, and that as the *Baseline model* has to choose between fewer classes, it would perform better.

In conclusion, adding new pitch-class sets to the model would decrease its performance. The reason for that is the quantity of common relevant degrees that different pitch-class sets share. This causes the model having to decide between distributions having similar probabilities distributions (overlapping) and resulting in bad interpretability of some musical events.

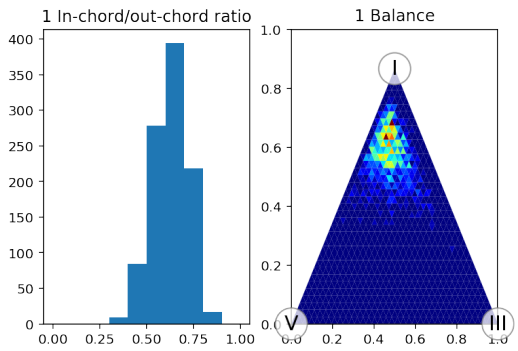


(a) Major chords

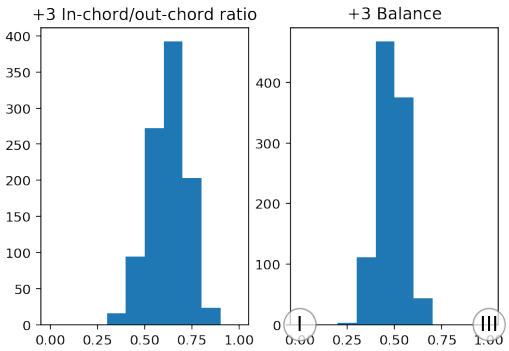
(b) Minor chords



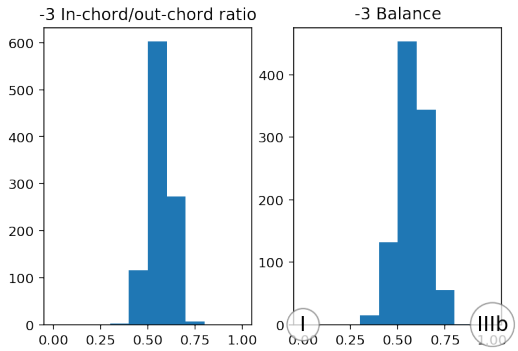
(c) Power chords



(d) Single notes

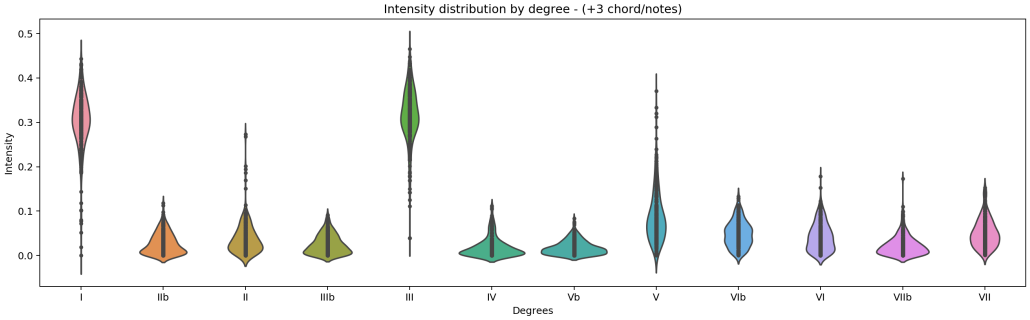


(e) Major thirds

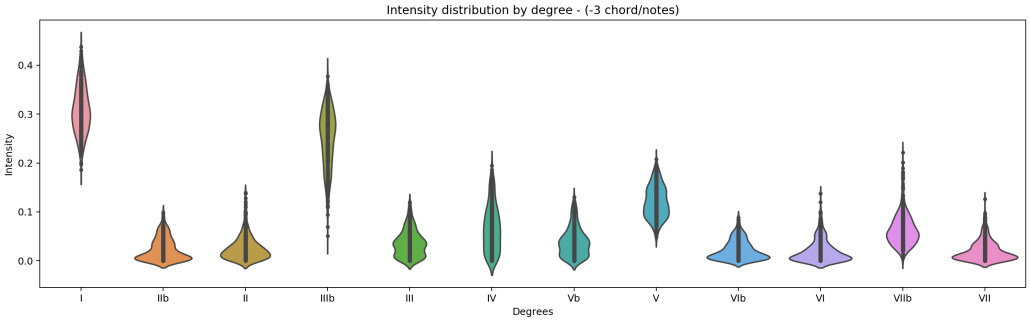


(f) Minor thirds

Figure 5.16: Model trained with thirds

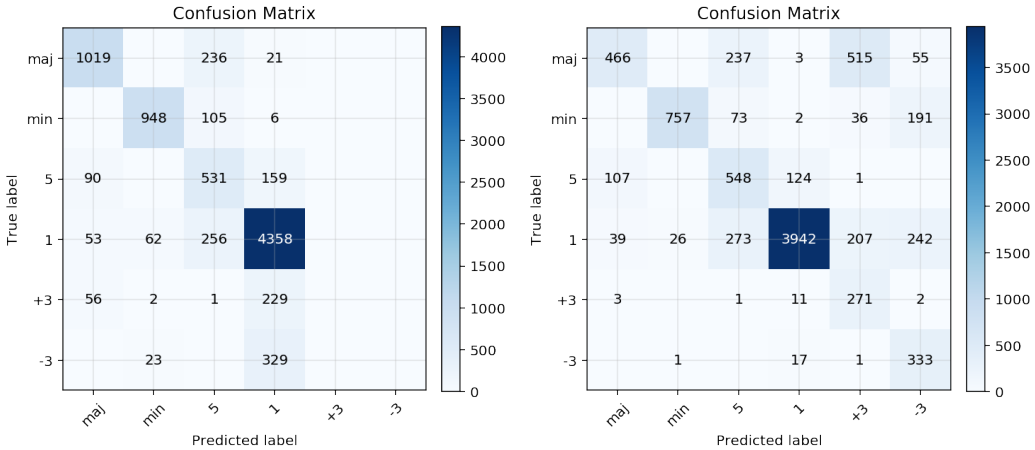


(a) Major third intervals



(b) Minor third intervals

Figure 5.17: Intensity distribution by degree in third intervals



(a) *Baseline model* (b) *Thirds model*

Figure 5.18: Music event kind errors caused by the two models.

5.9 Conclusions

In this chapter, we have observed how the model reacts to different situations and some problems that could be found in a real life situation have been stated. I have evaluated the model through different setups, effects, filters and training sets and conditions using metrics such as accuracy of pitch-class set and music event kind, plus confusion matrix plots and also subjectively predicting future errors and problems. Detailed conclusions about the experiments part are found in 6.

Chapter 6

CONCLUSIONS

In this project we have constructed a dataset called *Five Guitar dataset*, containing 30 guitar performances simultaneously recorded from 3 different setups. Then we applied data augmentation to generate variations of the audios to simulate different rooms, effects, filters and recording setups. With this batch of data, we tried to improve the model, but we saw that it is not possible due to high bias.

We realized that the current model still needs to improve. It is capable of differentiating between pitch-class sets, due to the overlap between degrees in the different pitch classes. Also, the model still recognizes a small number of pitch-class sets, and the limitation of having to represent all the pitch-class sets in the train set could not be applicable in a real context. In addition, another issue is the inability of identifying the octave of a note, and this makes it impossible to evaluate correct fingerings with the guitar. Lastly, we suggested considering onset positions and previous event spectral peaks for the calculation of the chroma vectors as a way of improving the model performance.

Recording setups or room acoustics influence the model performance. Regarding recording setups, we have observed cases where the performance varies a lot, causing completely different results. In terms of room acoustics, we have seen that different rooms behave disparately. Both factors could lead to biases in favor of students having specific conditions.

In terms of effects and filters, we observe timbre consistency. We were expecting a higher variance, mainly with the overdrive, but it seems that the model performance suffers less than expected. There is still variation implying that the final assessment could be affected by these two conditions.

Moreover, regarding system scalability, as we add more pitch classes to the model the performance decreases. This is due to shared degrees between pitch-class sets, making differentiation difficult. Also, we observed that the model per-

formance is highly dependent on the training set, as there are some biases generated by it. These biases alter the predictions made by the model.

Finally, despite having things to improve, Music Critic is still under development. Its main objective, providing proper feedback to students, makes it different from all the other products in the market, and this could be a strength. I expect that in the future it will be a powerful tool used by many musicians.

Bibliography

[1] Peak detection. https://ccrma.stanford.edu/~jos/parshl/Peak__Steps_3.html. Accessed: 2021-05-09.

[2] Juan Pablo Bello, Chris Duxbury, Michael Evan Davies, and Mark Sandler. On the use of phase and energy for musical onset detection in the complex domain. *IEEE Signal Processing Letters*, 11(6):553–556, 2004.

[3] Rachel M. Bittner, Eric Humphrey, and Juan Pablo Bello. Pysox: Leveraging The Audio Signal Processing Power of Sox in Python. *ISMIR Demo*, pages 4–6, 2016.

[4] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. Madmom: A new python audio and music signal processing library. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 1174–1178, 2016.

[5] Sebastian Böck, Florian Krebs, and Gerhard Widmer. A multi-model approach to beat tracking considering heterogeneous music styles. In *ISMIR*, pages 603–608. Citeseer, 2014.

[6] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, José Zapata, and Xavier Serra. Essentia: An audio analysis library for music information retrieval. In *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013*, pages 493–498. International Society for Music Information Retrieval (ISMIR), 2013.

[7] Baris Bozkurt, Sankalp Gulati, Oriol Romani, and Xavier Serra. MusicCritic : A technological framework to support online music teaching for large audiences. *33rd World Conference of International Society for Music Education (ISME)*, pages 13–20, 2018.

[8] Chris Cannam, Emmanouil Benetos, Matthias Mauch, Matthew EP Davies, Simon Dixon, Christian Landone, Katy Noland, and Dan Stowell. Mirex

2015: Vamp plugins from the centre for digital music. *Proceedings of the Music Information Retrieval Evaluation eXchange (MIREX)*, 2015.

[9] Chris Cannam, Christian Landone, and Mark Sandler. Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, page 1467–1468, New York, NY, USA, 2010. Association for Computing Machinery.

[10] Vsevolod Eremenko. *Automatic Harmony Analysis of Jazz Audio Recordings*. PhD thesis, Universitat Pompeu Fabra, October 2018.

[11] Vsevolod Eremenko, Alia Morsi, Jyoti Narang, and Xavier Serra. Performance assessment technologies for the support of musical instrument learning. In *CSEDU 2020 - Proceedings of the 12th International Conference on Computer Supported Education*, volume 1, pages 629–640, 2020.

[12] Eduard Vergés Franch. Five guitar dataset (version v.1.0) [data set]. In *Zenodo. http://doi.org/10.5281/zenodo.4988354*, 2021.

[13] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, and Hanna Wallach. Datasheets for datasets. *arXiv:1803.09010*, 372, 2018.

[14] Arjun K. Gupta and Saralees Nadarajah. *Handbook of Beta Distribution and Its Applications*. Statistics: A Series of Textbooks and Monographs. Taylor & Francis, 2004.

[15] Christopher Harte, Mark Sandler, Samer Abdallah, and Emilia Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *ISMIR 2005 - 6th International Conference on Music Information Retrieval*, pages 66–71, 2005.

[16] Jerry L Hintze and Ray D Nelson. Violin Plots: A Box Plot-Density Trace Synergism. *The American Statistician*, 52(2):181–184, 1998.

[17] P. Kasák, Roman Jarina, and Michal Chmulik. Music information retrieval for educational purposes - an overview. In *2020 18th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pages 296–304, 2020.

[18] Olivier Lartillot and Petri Toiviainen. A matlab toolbox for musical feature extraction from audio. In *International conference on digital audio effects*, volume 237, page 244. Bordeaux, 2007.

[19] Matthias Mauch and Simon Dixon. Approximate note transcription for the improved identification of difficult chords. In *ISMIR*, pages 135–140, 2010.

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

[21] Prajoy Podder, Tanvir Zaman Khan, Mamdudul Haque Khan, and M Muktadir Rahman. Comparative Performance Analysis of Hamming, Hanning and Blackman Window. *International Journal of Computer Applications*, 96(18):1–7, 2014.

[22] Justin Salamon and Emilia Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech and Language Processing*, 20:1759–1770, 08/2012 2012.

[23] Markus Schedl, Emilia Gómez, and Julián Urbano. Music Information Retrieval: Recent Developments and Applications. *Foundations and Trends® in Information Retrieval*, 8(2-3):127–261, 2014.

[24] Jan Schlüter and Sebastian Böck. Musical onset detection with convolutional neural networks. In *6th international workshop on machine learning and music (MML), Prague, Czech Republic*, 2013.

[25] Xavier Serra. Creating research corpora for the computational study of music: the case of the compmusic project. In *Audio engineering society conference: 53rd international conference: Semantic audio*. Audio Engineering Society, 2014.

[26] James Traer and Josh H. McDermott. Statistics of natural reverberation enable perceptual separation of sound and space. *Proceedings of the National Academy of Sciences*, 113(48):E7856–E7865, nov 2016.

[27] George Tzanetakis and Perry Cook. Marsyas: A framework for audio analysis. *Organised sound*, 4(3):169–175, 2000.

[28] Julián Urbano, Markus Schedl, and Xavier Serra. Evaluation in music information retrieval. *Journal of Intelligent Information Systems*, 41(3):345–369, 2013.

[29] K Gerald van den Boogaart and Raimon Tolosana-Delgado. *Analyzing compositional data with R. [electronic resource]*. 2013.

Appendices

In this section can be found all the figures and tables which are considered extra material useful for the understanding of some seen concepts.

.1 Figures

Figure 1, illustrates the confusion matrix obtained when evaluating the pitch class set classification accuracy with the *Baseline model* in section 5.2.1.

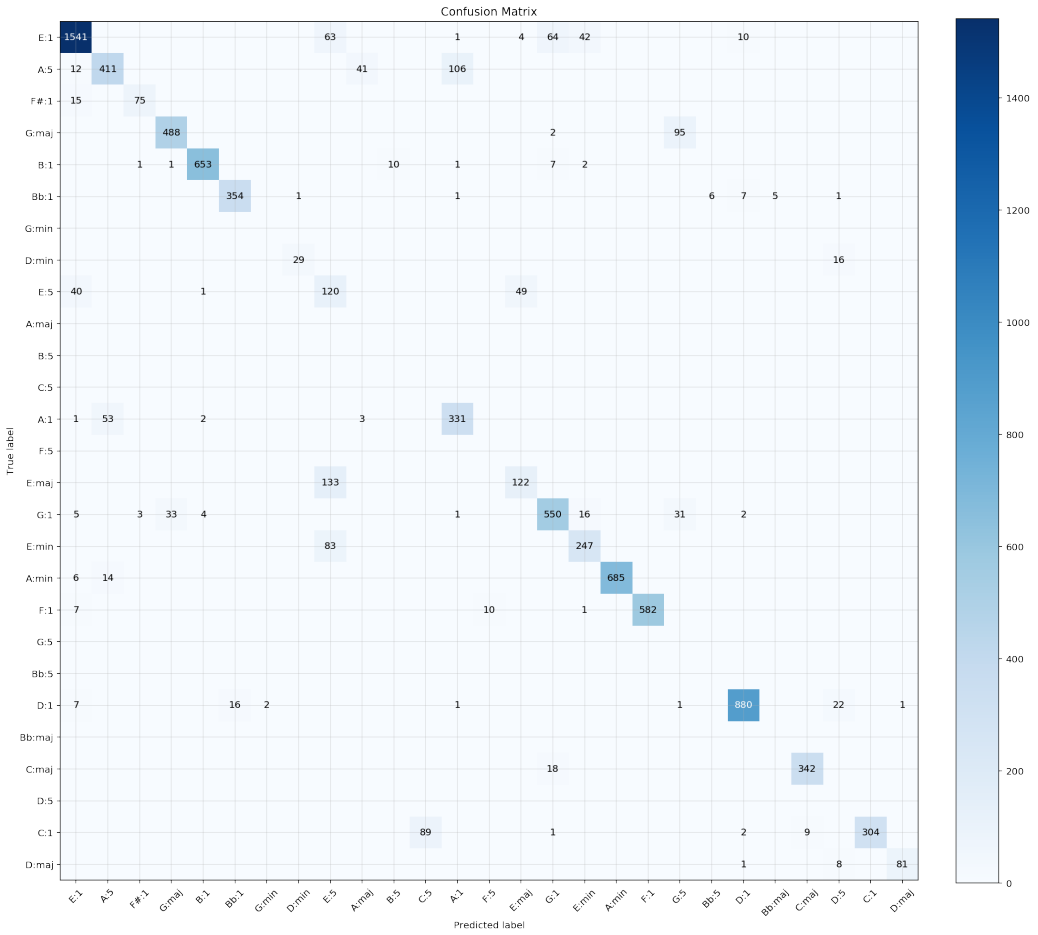


Figure 1: Baseline Case (no third intervals) Confusion Matrix

Figure 2, illustrates precision and recall obtained for each musical event type with *Baseline model* in section 5.2.1. It aims to illustrate how the system performs for each of the represented classes in the plot.

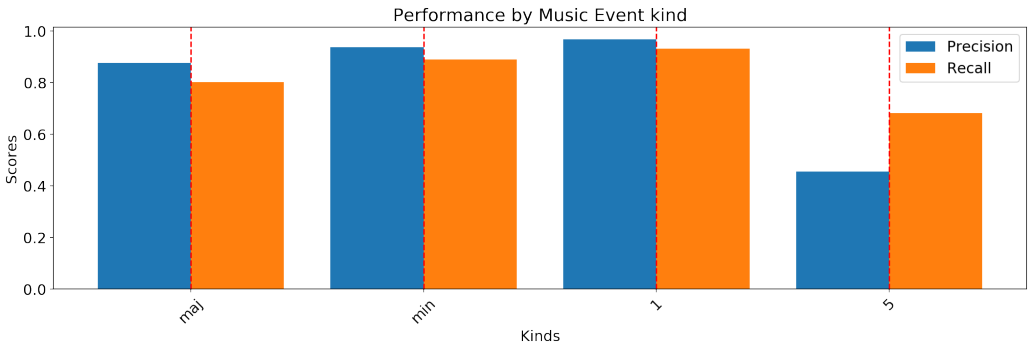


Figure 2: Precision/Recall for each music event kind in baseline case.

Figure 3, illustrates the confusion matrix obtained for a Lily Was Here guitar performance in section 5.2.1. The most important part is to see how single notes are miss-classified.

Figure 4, represents chromas obtained for different recording set ups in section 5.3. It tries to illustrate what type of errors we could find when predicting pitch class set, that are consequence of the recording set up used.

Figure 5, represent the different NNLS chroma vectors obtained for the same performance played in different rooms in section 5.5. It tries to illustrate how the prediction depends on the room acoustics.

Figure 6, represent the predictions obtained for two different performances of the same song in the same room. It tries to illustrate how the model performances is also dependent on the performance.

Figures 7, 8, 9, 10 and 11 illustrate the models learned when doing cross validation in section 5.7.

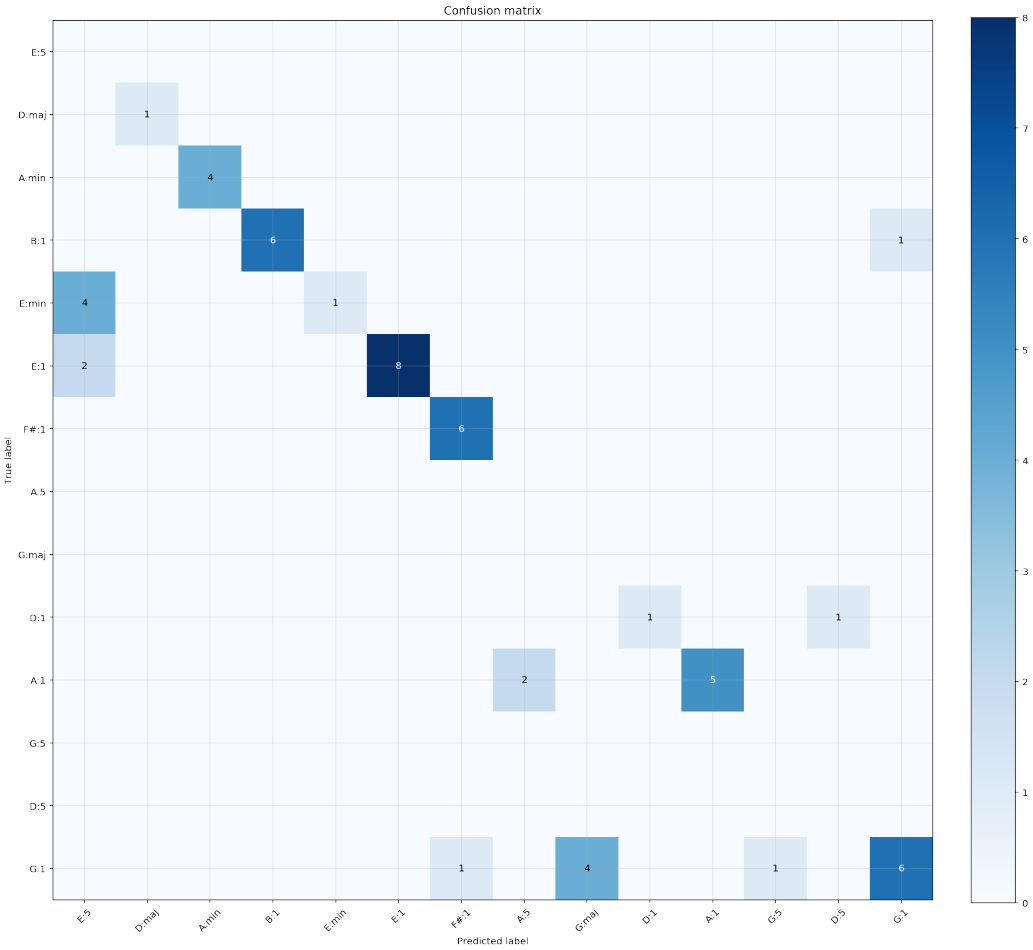
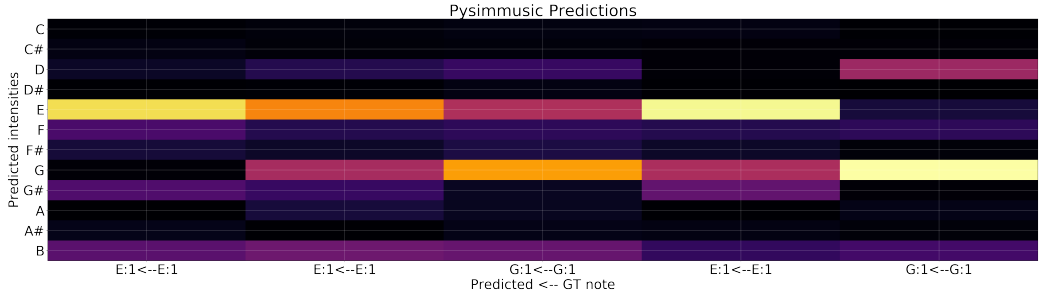
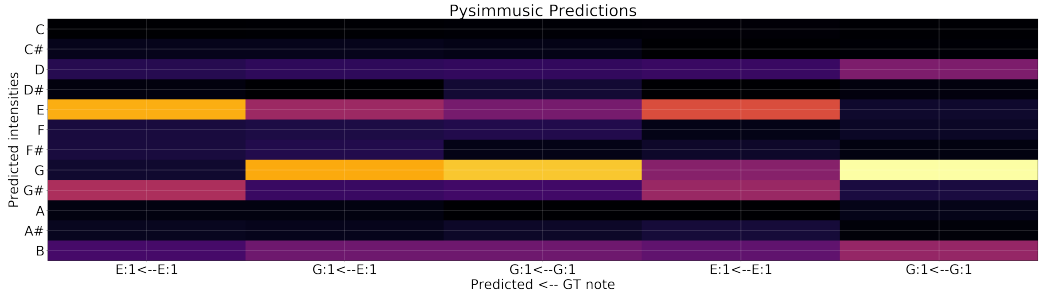


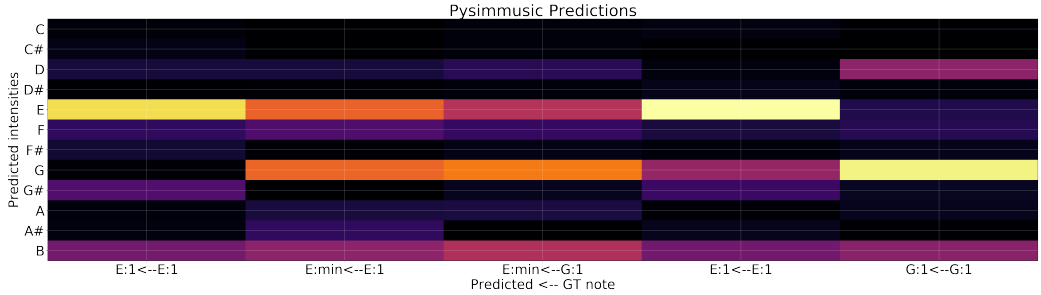
Figure 3: Lily Was Here confusion matrix for *Lily_Telecaster_108_DI* performance.



(a) Mobile

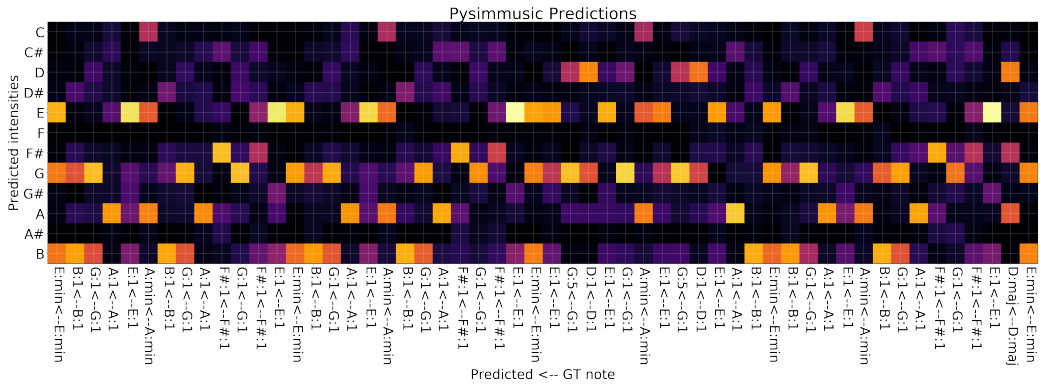


(b) DI

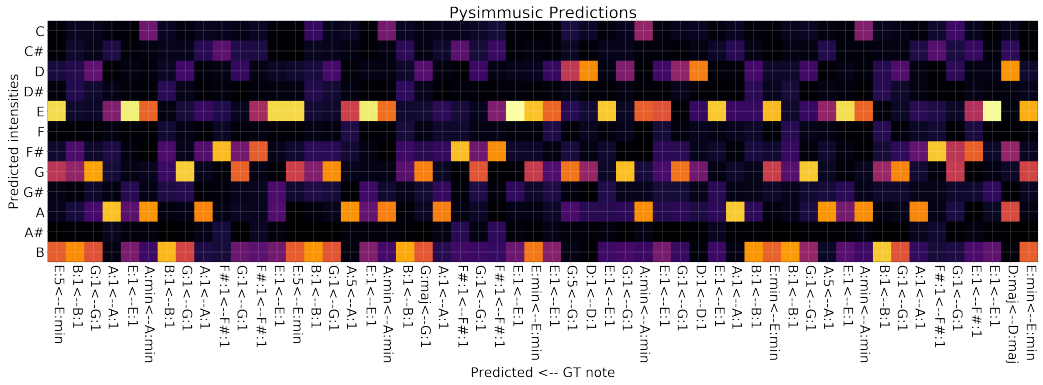


(c) Computer

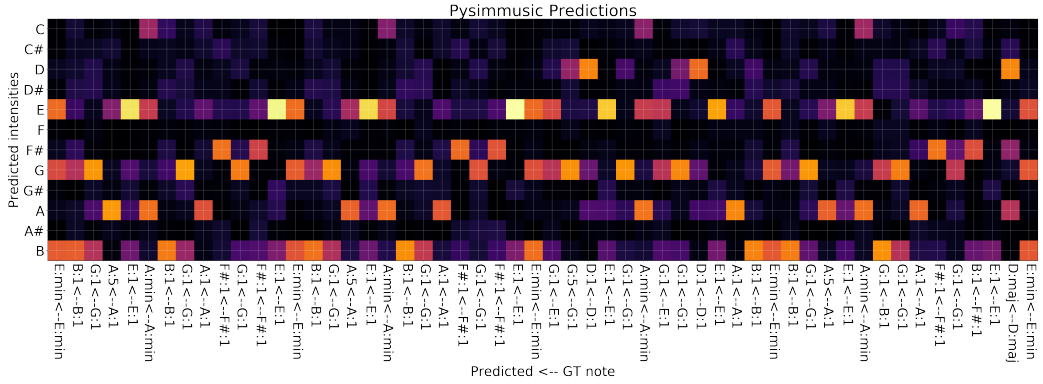
Figure 4: Recording Source chroma comparison



(a) Room Large 3. Accuracy 0.962

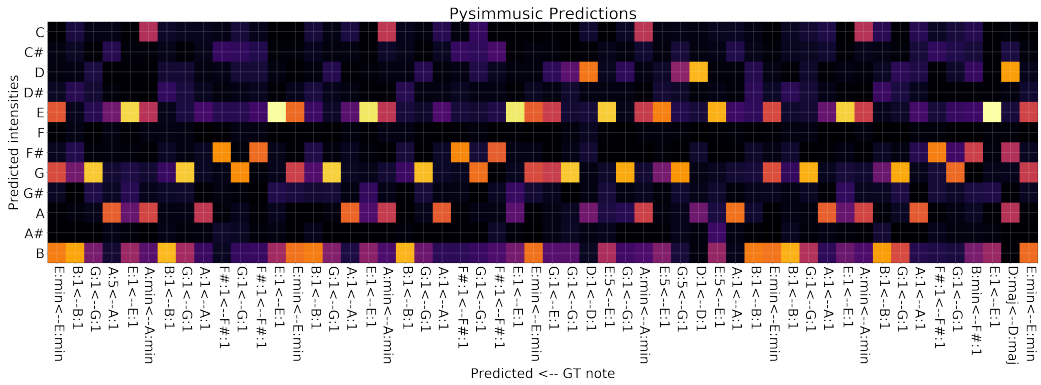


(b) Room Medium 3. Accuracy 0.870

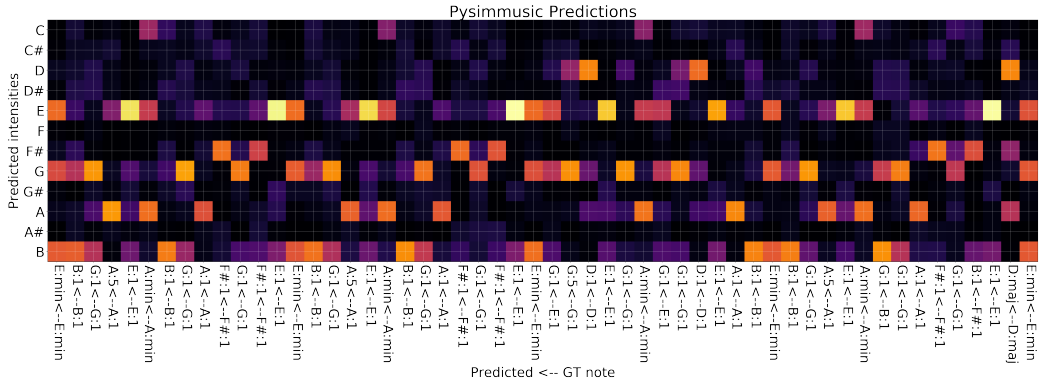


(c) Room Small 1. Accuracy 0.851

Figure 5: NNLS chroma vectors for same performance in different rooms

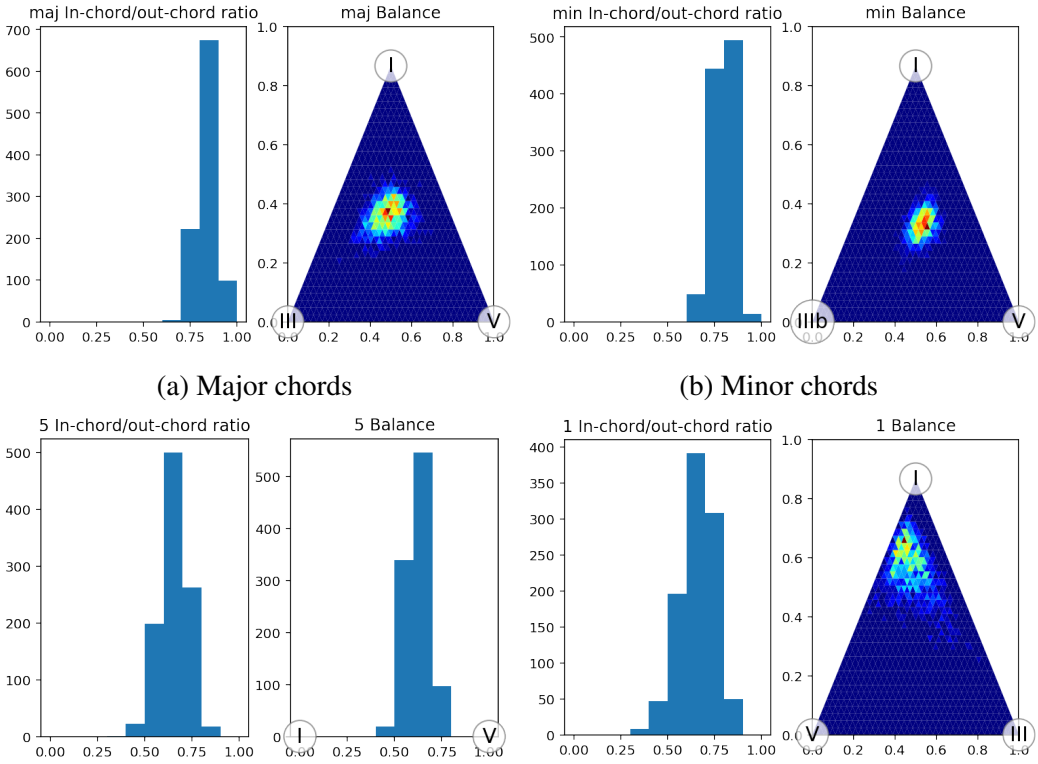


(a) Room Small 1. Accuracy 0.870



(b) Room Small 1. Accuracy 0.851

Figure 6: NNLS chroma vectors for different performances in the same room.



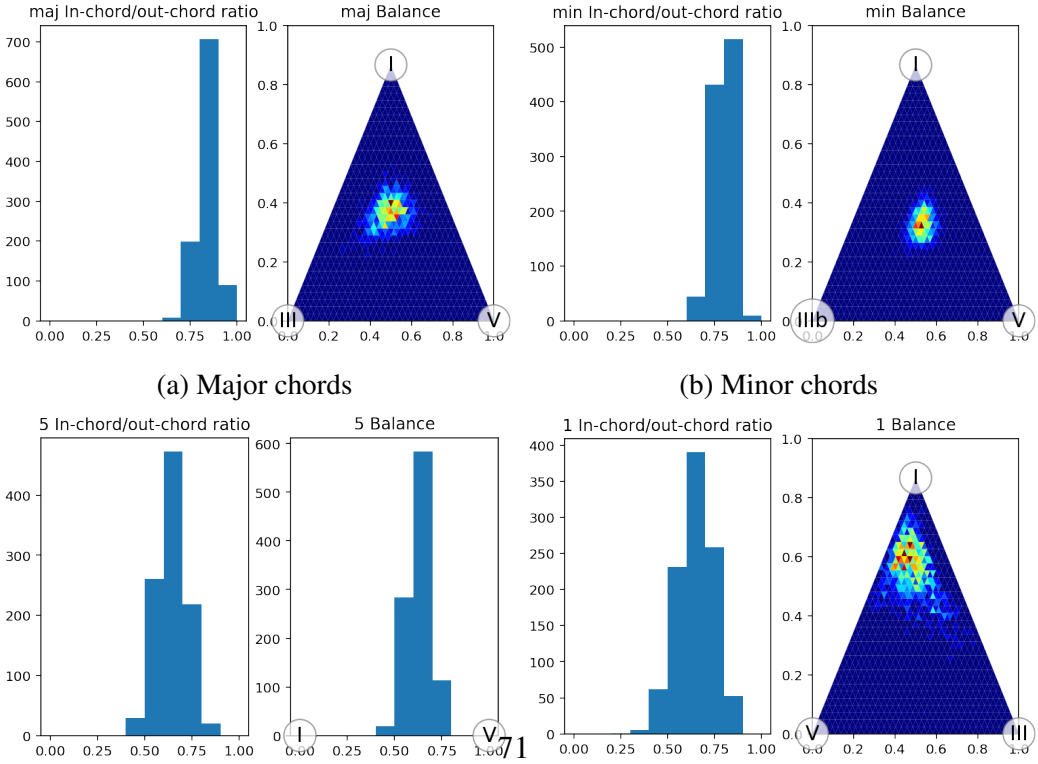
(a) Major chords

(b) Minor chords

(c) Power chords

(d) Single notes

Figure 7: Split 1



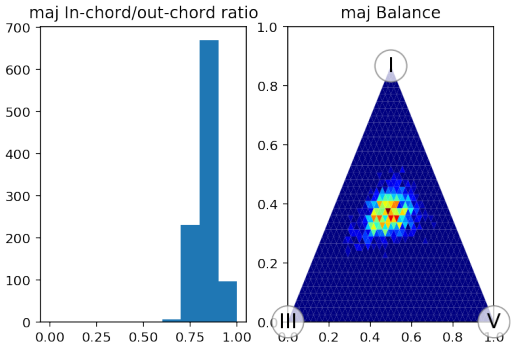
(a) Major chords

(b) Minor chords

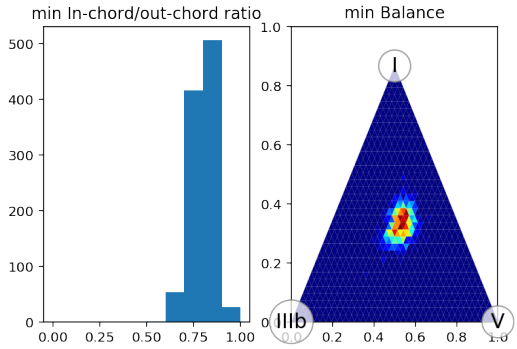
(c) Power chords

(d) Single notes

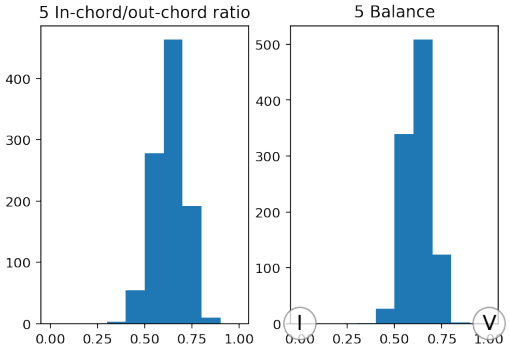
Figure 8: Split 2



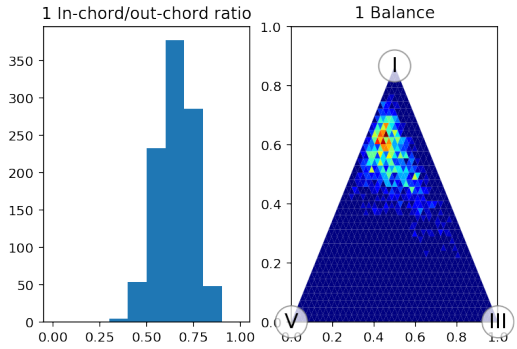
(a) Major chords



(b) Minor chords

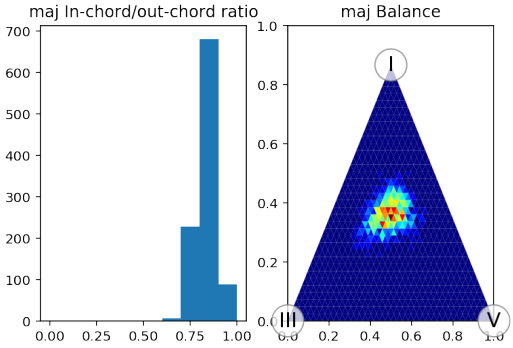


(c) Power chords

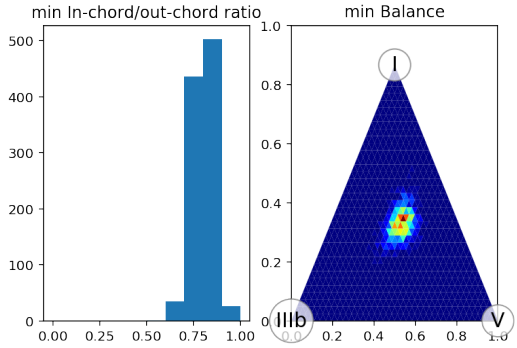


(d) Single notes

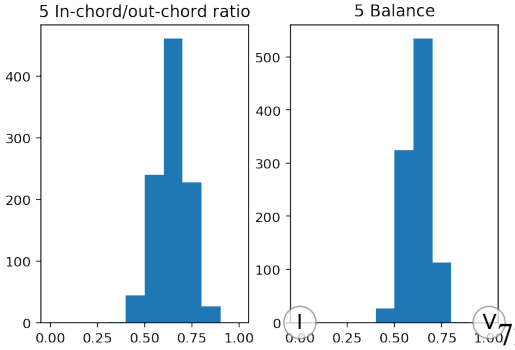
Figure 9: Split 3



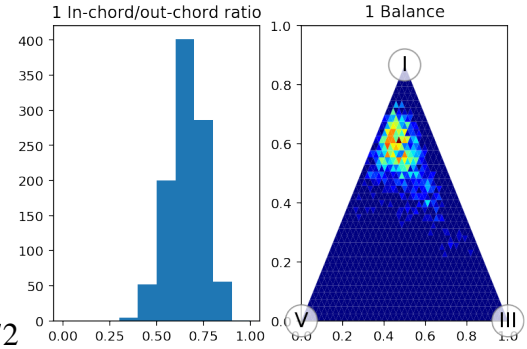
(a) Major chords



(b) Minor chords

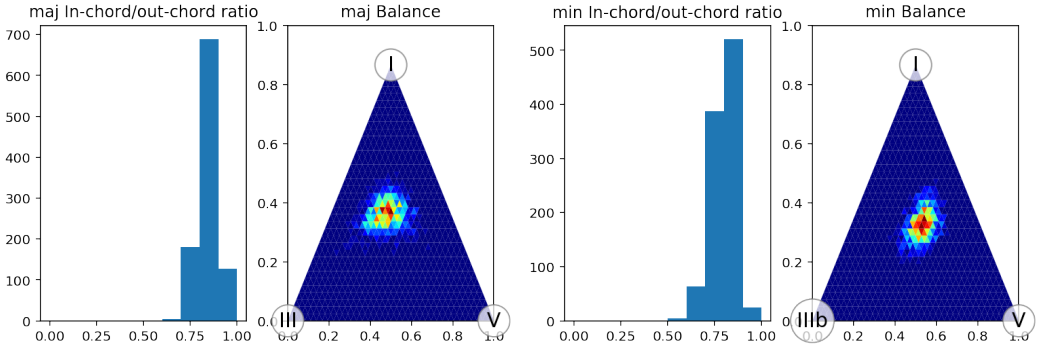


(c) Power chords



(d) Single notes

Figure 10: Split 4



(a) Major chords

(b) Minor chords

(c) Power chords

(d) Single notes

Figure 11: Split 5

.2 Tables

Table 1 contains all the classification accuracies obtained for the experiment done in 5.3.

Table 2 contains all the classification accuracies obtained for the experiment done in 5.4 with guitar effects.

Table 3 contains all the classification accuracies obtained for the experiment done in 5.4 with filters.

Table 4 contains all the classification accuracies obtained for the experiment done in 5.5.

Performance	DI	Mobile	Computer	SimComp	SimRadio1	SimMobile1	SimVoiceRecorder	SimRadio2	SimMobile2	TakStar	Sennheiser	AKG	Shure
Lily_Telecaster_108	0.7407	0.8333	0.8519	0.8519	0.7963	0.8704	0.7593	0.7593	0.7222	0.7407	0.7963	0.7778	0.7963
Hole_Epiphone_102	0.9765	0.9882	0.9882	1.0000	0.9765	0.9882	0.9765	0.9765	0.9882	0.9765	0.9765	0.9765	0.9765
Train_Larrivee_112	0.8660	0.8866	0.9485	0.9588	0.8557	0.8763	0.8763	0.8866	0.8866	0.8660	0.8660	0.8660	0.8866
Lily_Eastman_108	0.6852	0.7037	0.7037	0.7037	0.6481	0.7222	0.8148	0.6481	0.5741	0.6667	0.7222	0.6667	0.6852
Mountain_Ibanez_104	0.9202	0.9816	0.9509	0.9571	0.9325	0.8834	0.9448	0.9080	0.9448	0.8896	0.8957	0.9141	0.9325
Mountain_Larrivee_104	0.9141	0.9080	0.9448	0.9141	0.9141	0.9080	0.8405	0.8773	0.9264	0.8896	0.8712	0.9080	0.9018
Mountain_Epiphone_104	0.9448	0.9509	0.9387	0.9693	0.9264	0.9202	0.9509	0.9509	0.9448	0.8896	0.9202	0.9325	0.9448
Century_Larrivee_130	0.5641	0.8547	0.4444	0.5470	0.5641	0.4017	0.6838	0.5812	0.5641	0.6325	0.6068	0.6325	0.5983
Century_Telecaster_130	0.7521	0.6496	0.8120	0.7436	0.7350	0.7607	0.7179	0.7265	0.7521	0.7521	0.7607	0.7692	0.7521
Lily_Larrivee_108	0.7963	0.8148	0.7963	0.7963	0.8333	0.8519	0.7593	0.7778	0.8333	0.7963	0.8519	0.8333	0.8333
Train_Epiphone_112	0.7938	0.8763	0.8041	0.8866	0.8763	0.8866	0.7216	0.8763	0.8763	0.8041	0.8351	0.8454	0.8866
Century_Epiphone_130	0.8205	0.7350	0.8034	0.7009	0.7350	0.7436	0.7179	0.7436	0.8034	0.7949	0.7949	0.7949	0.8205
Mountain_Eastman_104	0.9509	0.9448	0.9509	0.9387	0.9387	0.9509	0.9571	0.9264	0.8957	0.9509	0.9387	0.9509	0.9448
Where_Ibanez_100	0.9036	0.9398	0.8675	0.9398	0.9036	0.9639	0.9157	0.9518	0.9398	0.8675	0.8795	0.9157	0.9036
Where_Telecaster_100	0.9036	0.7952	0.8554	0.8675	0.9277	0.9518	0.8193	0.8916	0.9157	0.8072	0.8795	0.9398	0.8675
Century_Ibanez_130	0.8803	0.7179	0.8889	0.7863	0.9402	0.8803	0.8718	0.7009	0.8974	0.9060	0.8803	0.9231	0.8974
Century_Eastman_130	0.8120	0.7692	0.7009	0.5812	0.8034	0.7350	0.5897	0.5556	0.7265	0.7949	0.8547	0.8376	0.8462
Hole_Telecaster_102	0.9882	0.9647	0.9294	0.9765	0.9765	0.9882	0.9412	0.9765	0.9765	0.9882	0.9765	0.9765	0.9765
Hole_Eastman_102	0.9412	0.9412	0.9765	0.9529	0.9294	0.9412	0.8588	0.9529	0.9412	0.9294	0.9294	0.9294	0.9412
Lily_Ibanez_108	0.8148	0.7963	0.8148	0.8889	0.8333	0.8519	0.9074	0.7963	0.7963	0.8333	0.8333	0.7963	0.7963
Train_Telecaster_112	0.7216	0.7320	0.6598	0.7938	0.7732	0.8144	0.6804	0.8144	0.7526	0.7113	0.7526	0.7835	0.7835
Lily_Epiphone_108	0.8519	0.9444	0.9630	0.9074	0.8704	0.9444	0.8333	0.7407	0.7963	0.8519	0.8519	0.8148	0.8889
Hole_Ibanez_102	0.9765	1.0000	0.9647	0.9765	0.9647	0.9765	0.9529	0.9765	0.9882	0.9882	1.0000	0.9882	0.9765
Where_Larrivee_100	0.7590	0.8193	0.8795	0.8072	0.7229	0.7349	0.7590	0.7831	0.6988	0.7229	0.7590	0.7831	0.7229
Hole_Larrivee_102	0.9882	1.0000	0.9765	0.9765	0.9529	0.9529	0.9529	0.9647	0.9647	0.9529	0.9529	0.9882	0.9529
Train_Eastman_112	0.7216	0.9072	0.9278	0.7526	0.7423	0.8041	0.7732	0.7938	0.7113	0.7423	0.7526	0.7320	0.7320
Train_Ibanez_112	0.9278	0.8144	0.8969	0.9485	0.9381	0.9381	0.8969	0.9278	0.9485	0.8763	0.8866	0.9381	0.9072
Mountain_Telecaster_104	0.9264	0.9387	0.9877	0.8528	0.9325	0.9509	0.8957	0.9387	0.9264	0.9387	0.9202	0.9387	0.9264
Where_Epiphone_100	0.8434	0.8795	0.9157	0.8434	0.8675	0.9157	0.8554	0.9639	0.9398	0.8554	0.8795	0.8675	0.8554
Where_Eastman_100	0.8434	0.8193	0.8795	0.8795	0.8434	0.8313	0.8313	0.9157	0.8434	0.8795	0.8675	0.8795	0.8434

Table 1: Recording source test results

Performance	DI	OverDrive	Phaser	Chorus	Flanger
Lily_Telecaster_108	0.7407	0.7778	0.7963	0.8333	0.8148
Century_Larrivee_130	0.5641	0.5641	0.6068	0.5726	0.5641
Lily_Larrivee_108	0.7963	0.7037	0.8519	0.8148	0.7778
Mountain_Eastman_104	0.9509	0.9509	0.9571	0.9755	0.9387
Lily_Ibanez_108	0.8148	0.8333	0.8519	0.8519	0.8333
Century_Telecaster_130	0.7521	0.6667	0.7607	0.7265	0.7863
Where_Larrivee_100	0.7590	0.8916	0.7952	0.7831	0.7349
Hole_Larrivee_102	0.9882	0.9765	0.9647	0.9765	0.9529
Train_Epiphone_112	0.7938	0.8557	0.8351	0.8247	0.8351
Train_Eastman_112	0.7216	0.8247	0.7216	0.7423	0.7423
Train_Ibanez_112	0.9278	0.9278	0.9278	0.8969	0.9278
Hole_Eastman_102	0.9412	0.9529	0.9412	0.9176	0.9529
Century_Eastman_130	0.8120	0.7607	0.8034	0.7778	0.7607
Lily_Epiphone_108	0.8519	0.8519	0.8704	0.8333	0.8519
Mountain_Telecaster_104	0.9264	0.9325	0.9264	0.9387	0.9264
Where_Eastman_100	0.8434	0.9157	0.8554	0.8554	0.8675
Mountain_Ibanez_104	0.9202	0.8528	0.9202	0.9509	0.9141
Where_Telecaster_100	0.9036	0.9036	0.9398	0.9518	0.9398
Hole_Ibanez_102	0.9765	0.9647	0.9647	0.9882	0.9765
Mountain_Epiphone_104	0.9448	0.8896	0.9325	0.9509	0.9264
Century_Epiphone_130	0.8205	0.6838	0.8034	0.7521	0.7692
Lily_Eastman_108	0.6852	0.7593	0.7037	0.7407	0.7222
Hole_Epiphone_102	0.9765	0.9412	0.9647	0.9765	0.9647
Hole_Telecaster_102	0.9882	0.9765	0.9765	0.9882	0.9882
Mountain_Larrivee_104	0.9141	0.9018	0.9018	0.9141	0.8712
Train_Larrivee_112	0.8660	0.9691	0.8763	0.8969	0.8557
Train_Telecaster_112	0.7216	0.8351	0.7423	0.7320	0.7629
Where_Epiphone_100	0.8434	0.8675	0.8675	0.8675	0.9036
Century_Ibanez_130	0.8803	0.7692	0.9145	0.8889	0.8718
Where_Ibanez_100	0.9036	0.9398	0.9398	0.9398	0.9398

Table 2: Guitar effects test results

Performance	DI	Lpass250	Hpass3000	Hpass1000	Lpass500	Hpass500	Lpass100
Lily_Telecaster_108	0.7407	0.7407	0.8148	0.7778	0.7407	0.7593	0.7407
Century_Larrivee_130	0.5641	0.5641	0.5385	0.5641	0.5812	0.5812	0.5726
Lily_Larrivee_108	0.7963	0.8333	0.7778	0.8333	0.8333	0.8704	0.7963
Mountain_Eastman_104	0.9509	0.9325	0.9571	0.9509	0.9387	0.9571	0.9325
Lily_Ibanez_108	0.8148	0.8148	0.8519	0.8333	0.8148	0.7963	0.8148
Century_Telecaster_130	0.7521	0.7521	0.7692	0.7778	0.7521	0.7607	0.7692
Where_Larrivee_100	0.7590	0.7711	0.7952	0.7952	0.7711	0.8193	0.7590
Hole_Larrivee_102	0.9882	0.9765	0.8471	0.9412	0.9882	0.9882	0.9765
Train_Epiphone_112	0.7938	0.7835	0.8557	0.8454	0.7938	0.8144	0.7835
Train_Eastman_112	0.7216	0.7010	0.7732	0.7423	0.7010	0.7216	0.6804
Train_Ibanez_112	0.9278	0.9072	0.9588	0.9485	0.9175	0.9278	0.9072
Hole_Eastman_102	0.9412	0.9412	0.9529	0.9529	0.9412	0.9529	0.9176
Century_Eastman_130	0.8120	0.8205	0.8205	0.8291	0.8205	0.8376	0.8205
Lily_Epiphone_108	0.8519	0.8519	0.8519	0.8333	0.8519	0.8333	0.8519
Mountain_Telecaster_104	0.9264	0.9018	0.9387	0.9264	0.9018	0.9387	0.9018
Where_Eastman_100	0.8434	0.8675	0.8554	0.8554	0.8675	0.8554	0.8675
Mountain_Ibanez_104	0.9202	0.8896	0.9387	0.9264	0.9080	0.9202	0.8834
Where_Telecaster_100	0.9036	0.9398	0.9759	0.9880	0.9398	0.9639	0.9157
Hole_Ibanez_102	0.9765	0.9882	0.9647	0.9882	0.9882	0.9765	0.9882
Mountain_Epiphone_104	0.9448	0.9325	0.9018	0.9325	0.9448	0.9264	0.9202
Century_Epiphone_130	0.8205	0.8205	0.7607	0.7778	0.8120	0.7949	0.8120
Lily_Eastman_108	0.6852	0.6667	0.6667	0.7037	0.6667	0.6852	0.6667
Hole_Epiphone_102	0.9765	0.9765	0.9412	0.9647	0.9765	0.9765	0.9647
Hole_Telecaster_102	0.9882	0.9882	0.9765	0.9765	0.9882	0.9882	0.9765
Mountain_Larrivee_104	0.9141	0.9018	0.9202	0.9202	0.9018	0.9141	0.8957
Train_Larrivee_112	0.8660	0.8763	0.9691	0.8866	0.8660	0.8763	0.8660
Train_Telecaster_112	0.7216	0.7010	0.7835	0.7835	0.7113	0.7732	0.6495
Where_Epiphone_100	0.8434	0.8554	0.9036	0.8795	0.8554	0.8434	0.8554
Century_Ibanez_130	0.8803	0.8889	0.9145	0.9402	0.8803	0.9145	0.8974
Where_Ibanez_100	0.9036	0.8795	0.9759	0.9639	0.9157	0.9277	0.8675

Table 3: Filter test results

Performance	DI	Medium3	Large3	Large1	Large2	Medium2	Small1	Medium1	Small3	Small2	Large4	Medium4	Small4
Lily_Telecaster_108	0.7407	0.8704	0.9630	0.8333	0.9444	0.8704	0.8519	0.8519	0.8704	0.9074	0.7593	0.7593	0.7963
Century_Larrivee_130	0.5641	0.5897	0.5214	0.6667	0.6752	0.5385	0.4274	0.5299	0.5726	0.6325	0.6752	0.5726	0.5812
Lily_Larrivee_108	0.7963	0.7963	0.8519	0.7593	0.8889	0.8519	0.7778	0.8519	0.8704	0.8333	0.8704	0.7593	0.8333
Mountain_Eastman_104	0.9509	0.9755	1.0000	0.9755	1.0000	1.0000	0.8834	0.8834	1.0000	0.9509	0.9877	0.9141	0.9632
Lily_Ibanez_108	0.8148	0.8889	0.9444	0.9074	0.9259	0.9259	0.8333	0.8704	0.9074	0.9259	0.8704	0.8889	0.8148
Century_Telecaster_130	0.7521	0.8034	0.6239	0.7436	0.6496	0.7179	0.6752	0.6581	0.7094	0.7949	0.7350	0.7692	0.7436
Where_Larrivee_100	0.7590	0.7831	0.7831	0.7229	0.8072	0.6747	0.6627	0.7711	0.7711	0.7711	0.8193	0.6627	0.7831
Hole_Larrivee_102	0.9882	0.9647	0.9294	0.9059	0.9882	1.0000	0.9765	0.9882	0.9882	0.9529	0.9529	0.9294	0.9647
Train_Epiphone_112	0.7938	0.8454	0.9175	0.7423	0.8969	0.8557	0.9072	0.9175	0.8763	0.9072	0.9175	0.8660	0.8247
Train_Eastman_112	0.7216	0.7835	0.8247	0.7216	0.8144	0.7423	0.8454	0.8454	0.7835	0.8557	0.7938	0.6598	0.7526
Train_Ibanez_112	0.9278	0.9381	0.9691	0.8660	0.9278	0.9381	0.9588	0.8454	0.9175	0.9381	0.9485	0.9381	0.9381
Hole_Eastman_102	0.9412	0.9529	0.9412	0.9412	0.9882	0.9529	0.9294	0.9176	0.9176	0.9529	0.9882	0.9059	0.9647
Century_Eastman_130	0.8120	0.7607	0.6325	0.8632	0.8120	0.7607	0.5043	0.7009	0.6752	0.7949	0.7949	0.7778	0.8291
Lily_Epiphone_108	0.8519	0.9259	0.8704	0.8333	0.9630	0.9630	0.8704	0.9815	0.9074	0.9444	0.8704	0.7963	0.8333
Mountain_Telecaster_104	0.9264	0.9693	1.0000	0.9571	0.9816	0.9939	0.8528	0.9264	0.9509	0.9325	0.9816	0.8650	0.9509
Where_Eastman_100	0.8434	0.7831	0.8675	0.8675	0.8916	0.8072	0.7831	0.8554	0.8554	0.9036	0.8434	0.8193	0.8554
Mountain_Ibanez_104	0.9202	0.9202	0.9693	0.9816	0.9816	0.9693	0.8037	0.9448	0.9877	0.9018	0.9632	0.7607	0.9448
Where_Telecaster_100	0.9036	0.8072	0.9036	0.9518	0.9518	0.8193	0.9036	0.8434	0.9157	0.9398	0.9036	0.9036	0.9518
Hole_Ibanez_102	0.9765	0.9765	0.9882	0.9882	1.0000	0.9765	0.9529	0.9765	0.9882	0.9529	0.9647	0.9765	0.9882
Mountain_Epiphone_104	0.9448	0.9755	0.9571	0.9448	0.9080	0.9816	0.7975	0.9264	0.9755	0.9264	0.9693	0.7914	0.9325
Century_Epiphone_130	0.8205	0.8034	0.5897	0.7436	0.7094	0.7350	0.7607	0.6410	0.7179	0.7350	0.7350	0.7949	0.7778
Lily_Eastman_108	0.6852	0.6296	0.8519	0.6852	0.8519	0.8333	0.7778	0.8148	0.7778	0.7963	0.7407	0.6667	0.7037
Hole_Epiphone_102	0.9765	0.9765	0.9765	0.8824	1.0000	0.9176	0.9647	0.9647	0.9765	0.9412	0.9529	0.9529	0.9765
Hole_Telecaster_102	0.9882	0.9765	0.9765	0.9294	0.9882	0.9882	0.9882	0.9647	0.9882	0.9647	0.9882	0.9176	0.9882
Mountain_Larrivee_104	0.9141	0.9202	0.9571	0.9509	0.9816	0.9387	0.7975	0.9325	0.9693	0.8834	0.9448	0.8405	0.9080
Train_Larrivee_112	0.8660	0.9588	0.8454	0.9691	0.9381	0.9175	0.8969	0.8969	0.9278	0.8660	0.9381	0.8969	0.8660
Train_Telecaster_112	0.7216	0.7010	0.8041	0.6495	0.8144	0.6804	0.8144	0.7938	0.8247	0.8041	0.7113	0.7629	0.7835
Where_Epiphone_100	0.8434	0.8916	0.8916	0.8795	0.8795	0.8313	0.9157	0.8434	0.8554	0.9157	0.8554	0.9398	0.8675
Century_Ibanez_130	0.8803	0.9231	0.8034	0.8547	0.8974	0.8120	0.7094	0.7778	0.8205	0.9402	0.8205	0.8974	0.9060
Where_Ibanez_100	0.9036	0.9277	0.9518	0.8313	0.8072	0.8675	0.8675	0.8313	0.8554	0.8795	1.0000	0.9518	0.9518

Table 4: Room Acoustics results