The impact of temporal features in Music Genre Recognition

Master thesis on Data Analytics

VÍCTOR SANTIAGO GONZÁLEZ

Student no: **D16128055**, Programme: **DT228A**
Source of data: **GTZAN**

Dublin, January 2, 2019

DECLARATION

I declare that this work is entirely my own and all sources have been acknowledged.

ACKNOWLEDGEMENTS

Dr. John McAuley.

He taugth me how data science can be a creative activity.

Dr. Sean O'Leary.

He asked me all the questions I didn't want to answer.

Mamá, Papá...

All of them provided valuable guidance and advice.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Background

This project addresses the problem of classifying audio tracks according to their musical genre.

Therefore, the project is grounded in the research area of Music Information Retrieval (MIR) a specialisation of the broader discipline of Digital Signal Processing (DSP) that encompasses particular problems such as instrument detection, beat detection, musical transcription, song recognition by example (Parascandolo et al., 2016) or Music Genre Recognition (MGR) (Tzanetakis and Cook, 2002) which is the problem covered in this work.

Since the early 2000s, due to the emergence and success of online music streaming platforms, we find significant research activity on MIR, being the Music Information eXchange (MIREX, 2005) the main source in the topic.

MIR's progress has always been closely tied to its industrial applications like Shazam (2002), the most popular application developed using MIR findings or the different strategies adopted by services like Spotify or Deezer.

Within the collection of problems included in MIR, the problem of identifying musical genre have had a deep impact on the modern music industry.

Homsi et al. (2012) found evidence that users are more likely to browse and search by genre than similarity or artist. Venrooija and Schmutz (2018) also pointed out that music classification (machine-based or human-based) has significant effects in sales and the economic success of musical products.

A good example is Epidemic (2009), a record company based in "open source" principles that have found recently its target audience among the YouTube community. A close look on their website reveals that music genre or music similarity is the main criteria to distribute their catalogue.

Therefore accurate classifiers play a crucial role in the music industry. To the point that MGR can be seen as *"the fundamental problem in* MIR*"*

However, MGR is an ill-posed problem: the criteria used to classify a song as *rock* or *blues* can vary across many factors, including:

- Marketing purposes
- The granularity of the analysis

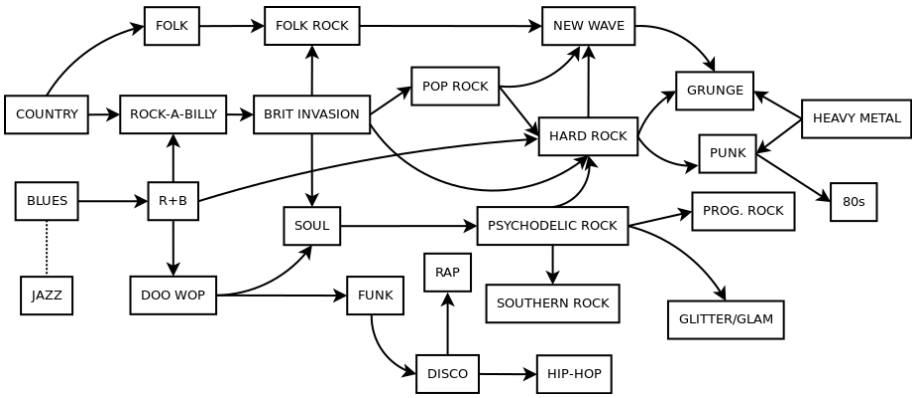

Figure 1.1: *Simplified transcription of the "Genealogy of Pop Rock music", Tufte (1997). The graph presents a fine-grained categorisation of rock genres, most of these categories are not present in common* MGR*'s datasets*

- Discrepancies between authors or listeners. An experiment conducted by Sturm (2012b), for instance, found that "*humans do not recognize the genres supposedly represented [in the dataset]*"[1]

This fuzzy definition of what determines a musical genre produces (inevitably) contentious results in any MGR model.

[1] We will see during the analysis of our dataset some examples of this problem

Although this project is focused on MGR and the design of music classifiers, the strength in our development is to study the effects of sound representations and algorithmic design. Thus, our approach mitigates that complexity adopting as "grounded truth" the labels of a well-known dataset.

Even though, this dimension of complexity in MGR cannot be ignored completely, as we will see later in our literature review.

1.2 Research project

The goal of this project consists of answering the following question:

" Do musical temporal features have a positive and significant impact to classify audio tracks according to their musical genre? "

In this context, *temporal features* have a double meaning. These features can be expressed in two ways: the method that encodes the raw audio files and the algorithm in use to process it.

Usually, there are two approaches in MIR: use of physical metrics (McKinney and Breebaart, 2003) like the zero crossing rate, spectral centroids, band energy ratios... Or using spectrograms (Costa et al., 2011) where the audio signal is analysed in the temporal domain to produce an image based representation.

We will use the second alternative, implementing spectrograms based on different aspects of sound: perceptual or musical features.

On the other hand, our choice for a classifier can also be inspired in the temporal aspects of sound.

In recent years, the best results in MIR have used deep learning techniques over datasets in the form of spectrograms. Taking advantage of successful advances in artificial

vision, i.e., the use of convolutional neural networks.

But convolutional neural networks have more than one application: they also have shown good performance addressing natural language processing problems, where the combination of unidimensional convolutions and LSTM layers have met the sequential nature of text strings.

Music, in its very nature, is a stream of data: a sequence of pulses over time. Thus, our algorithmic approach grasping temporal features in audio clips will include a contrast between unidimensional and bidimensional convolutions.

Since the temporal features are two folded (via sound representation and the algorithm in use) our statistical study of the results will take the form of a multifactorial experimental design.

In addition, we are not interested in outstanding performance figures, but to compare the compounding effects between algorithm and sound representations.

Despite we are not actively chasing high-performance numbers, we will only take into account those steps whose performance is close to the state of the art in MGR. Otherwise, our conclusions would not be truly valid.

Therefore, we decided to use the GTZAN dataset (Tzanetakis, 2018); because it has been and studied in depth which let us define precise what is the state of the art in MGR.

1.3 Scope and limitations

As we have mentioned in the *Background* (Section 1.1) this project is grounded in MGR (a specialised branch of the broader research area known as MIR) but it is constraint by the use of GTZAN dataset.

GTZAN is a popular dataset for MGR research, we will justify this assert in the literature review, (Chapter: 2) which helps to contextualize our findings. It is also a relatively small

dataset where the songs are labelled using only 10 categories. Hence, the power of GTZAN based classifiers are limited to this relatively small number of musical genres.

While it is true that GTZAN would not be the best choice to train an ambitious MGR classifier[2], it was a reasonable choice for a project like ours. Our aim is to compare the impact of several factors addressing temporal features in music, rather than creating a "*perfect*" classifier.

In addition, it worth to mention that this project has been implemented and completely developed using a regular commercial laptop: No university's or cloud computing services have been used.

This does not undermine the validity of our conclusions, but it constraints the complexity of the neural networks in use, which turns into a reduction of the performance that we can achieve.

It is a reasonable assumption that an improvement in hardware resources would lead to better accuracies due to more complex designs or longer training phases.[3]

Again, since the aim of this work is not getting outstanding classifiers but researching the factors that help to create them, we did not try to improve our project in this way.

1.4 Organisation of this document

This document contains detailed explanations about the ideas that support our research and those processes required to reproduce the experiments.

---

[2] A system that can reach the production stage of an online music platform like Spotify, because such system's requirements deal often with a much greater number of categories: rock, southern rock, hard rock...

[3] This point is crucial and does not refer to "time" considerations but "number of epochs". For instance, our experimentation (running 80 epochs on average) took an entire month 24 hours a day of computing.

The report is organised as follows:

Chapter 1: Introduction.

Research background. This chapter introduces the project´s guidelines: problem description, research question, limitations...

Chapter 2: Literature Review.

Since our research is quite specific, we have focused our literature review to assess what exactly the *state of the art* is for the specific circumstances of our problem.

Therefore, let alone with a generic background, this chapter analyses historical trends and the performance evolution in MGR research. It also introduces the GTZAN dataset: the results obtained using it and the criticism about GTZAN found in the literature.

Chapter 3: Design and methodology.

This chapter covers all the underlying ideas on our project: different types of sound representation and the reasons to justify our experimental design and methodology.

Chapter 4: Implementation.

This chapter describes the tools and implementations nuances of our design. It also includes a detailed description of a set of command line utilities we have implemented to let the reviewers of this work reproduce the experiments from the scratch.

Chapter 5: Discussion.

We present here the results of our statistical analysis conducted over our experimental data and we comment briefly (and form an *agnostic* perspective) our findings.

Chapter 6: Conclusions.

In this chapter, we examine our findings in the context of MGR research: how do they confirm other research projects, what is the novelty of our work, future research directions...

2 LITERATURE REVIEW

A typical approach in MGR is two fold: First, we need a proper representation of sound. Second, we will use a particular algorithm or technique to process the data in our representation.

It is the combination of representation and a classifier which determines good or bad performance.

Most of these representations come from DSP practice, being common choices the Mel's Frequency Cepstral Coefficients (Zhouyu et al., 2010) combined with spectrograms (Mark and Handy, 2007)

Meanwhile, spectrograms use the Short-Time Fourier transform to represent the sound in a graphical manner, MFCC consists of a numerical representation of sound features.

Usually, the representation choice is grounded in DSP background, i.e., domain knowledge. However, it has to be noted that, due to copyright restrictions, the common datasets of choice in MGR do not provide access to the actual songs. GTZAN, for instance, includes 30*s* excerpts.

2.1 Generic approaches

2.1.1 Research on feature engineering

Yandra et al. (2013), used a combination of spectrotemporal features (timbre, MFCC...) and SVM classifiers to report accuracy varying from 80% to 90% in the GTZAN[1] dataset.

Valero and Alias (2012) presented an alternative to MFCCs: the gammatone cepstral coefficients. However, they found that these coefficients do not represent a significant

[1]We will see later on that some of these excellent results are controversial

improvement over the commonly used MFCC.

This work, thus, seems to support the idea of MFCCs as "*unarguably the single most important feature*" (Zhouyu et al., 2010)

Granma and Rusu (2017), faced the sub-problem of representation defining an optimal set of cepstral coefficients. In the first case[2] the focus was to select an *accurate* number of Mel's coefficients.

2.1.2 Research on experimental design

Spectrograms provide a pictorial representation of the temporal evolution of frequency domain sound features (Mark and Handy, 2007; Costa et al., 2011) Several studies have taken advantage of this image-based representation.

Shim-Cheol et al. (2012) built an SVM classifier using spectrotemporal features, reporting accuracies that varies from 80% to 90% (the exact figures depend on feature selection in their experiments) in the GTZAN dataset.

However, with no doubt, it is the use of neural networks what get the most out of the spectrogram representation in recent years.

A paper on the topic (Rajanna et al., 2015) reported the use of Convolutional Neural Networks (CNN), Deep Belief Networks (DBNN) and Deep Neural Networks (CNN) to address the problem of MGR.

The common approach seems to be a direct translation of findings in the area of image

[2]This paper reported the best classification accuracy; peaking at 99%. However, Granma and Rusu, 2017 addressed a slightly different problem: audio classification where the audio clips under study do not represent music tracks.

classification using the spectrogram representation.

Costa et al. (2017) compared the use of CNN over LMBD and ISMIR (2000) datasets with the results obtained using SVM classifiers. They reported accuracies in the range of 80% that peak to 90% when the experiment involved an ensemble setup mixing SVM and CNN's.

Costa et al. (2004) describes an experiment using CNN reporting a classification rate up to 90%. Unfortunately, they do not provide access to the dataset under study.

Gwardys and Grzywczak (2014) used an original approach known as *"learning transfer"* where a CNN was trained against a purely visual classification dataset (LSVRC, 2012) and then the resulting model was used to classify instances of in the GTZAN dataset. They obtained accuracies up to 75%

Using a different approach Hamel and Douglas (2010) combine DBNN as a feature selectors to feed an SVM classifier obtaining accuracies up to 85% using the GTZAN dataset.

Even though deep learning techniques have become popular, different authors conducted research using pure machine learning systems: Bergstra (2006) presented AdaBFFs an AdaBoost algorithm that reached accuracy up to 85% in GTZAN using SVM weak learners; and performed the best in the 2005 international MGR conference MIREX (2005)

Panagakis et al. (2009) described a system using a sparse representation classification based on auditory features dictionaries, reporting a classification accuracy up to 90% in GTZAN. However, Sturm (2013) pointed flaws in their experimental process that reduced their performance to a more common 80% in subsequent replications.

Mallat (2012) designed a method making use of the features proposed by "Multiscale Scattering for audio classification" and Bayesian classifiers. They reported an 82% classification accuracy in GTZAN.

2.2 MGR research

2.2.1 Music Information Retrieval eXchange

The Music Information Retrieval eXchange is the reference conference in Music Information Retrieval. Regularly since 2007, MIREX has run contests covering the main topics in the area: tempo estimation, musical transcriptions, onset detection, and, music genre recognition.

Regarding MGR, MIREX has different tracks that depend on the type of music under analysis: Kpop, Latin, and general popular music.

Since the competition based in "general popular music" has a long recorded history in MIREX and it is quite similar to the contents present in our dataset, it is a good source to define MGR's *state of the art.*

Each submission to MIREX is evaluated against a randomized experimental ANOVA design and they include a brief paper summarising the contents of the submitted works.

We comment here our conclusions after an exhaustive[3] literature review of MIREX's papers since 2007. However, Appendix A contains a paper breakdown and the bibliography includes their references.

Having said that, our interest in MIREX data is to answer some of the questions we pointed out in chapter 1: *Which are popular algorithms in* MGR *research? What is a good performance in* MIREX *submissions? Are there any relevant trends to notice?*

[3]Sadly, some papers in MIREX history are not available from its website. Their links throw 404 HTTP errors. We have excluded these papers in our analysis. This is the reason why we don't have data for 2010 and 2011

ALGORITHMS We see (Fig: 2.1) that the research community consider Support Vector Machines as the main technique to develop classifiers in MGR, despite some other algorithms such as Neural Networks (NN) or K-nearest neighbours (KNN) are also present.

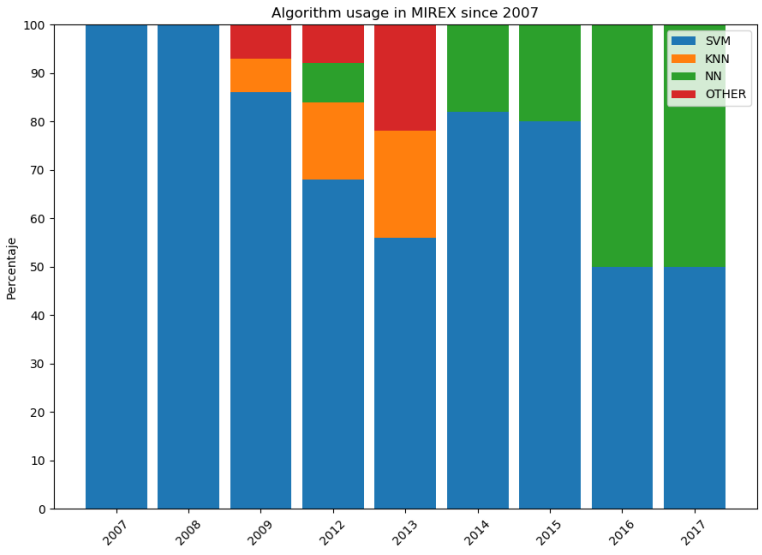

Figure 2.1: *Algorithm usage in* MIREX *since 2007*

However, the most noticeable trend is the **use of Neural Networks** which peaked at 50% in the last editions, constantly growing since its debut in 2007.

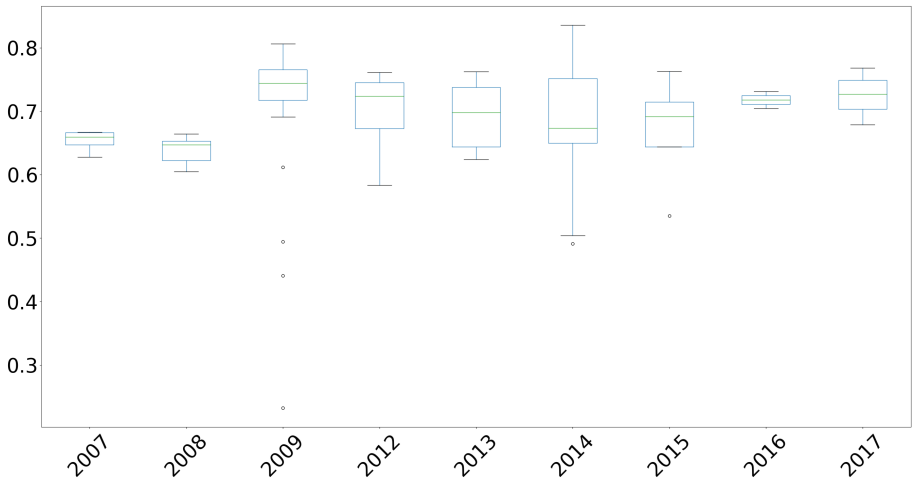

Figure 2.2: *Average accuracy evolution in* MIREX *since 2007*

PERFORMANCE We see in the chart (Fig: 2.2) MIREX accuracy for music genre recognition has progressed slowly from a 60% **to the band of** $70-80\%$.

There are also some outliers (Fig: 2.2). A closer read of MIREX papers shows that those data points belong to papers which use *ad-hoc* algorithms. Therefore, MIREX data seems to confirm MGR as a playground for machine-learning fundamental research.

2.2.2 The GTZAN dataset

The music collection under study in this work is the GTZAN dataset, publicly available from:

`http://opihi.cs.uvic.ca/sound/genres.tar.gz`

Its availability and the fact that the dataset provides real audio files[4] turn this dataset into *"the most-used public dataset"* (Sturm, 2012a)

GTZAN includes 1000 audio excerpts (30s length each) from popular music, i.e., GTZAN does not represent a certain musical niche (like the Latin Music Database, Silla (2018)) but an example of the mainstream musical background. (see Table 2.1)

[4]It is a common practice in MGR datasets, like the Million songs dataset, to provide references and metadata (including pre-calculated features) instead of the real audio files

GTZAN DESCRIPTION

AUDIO FORMAT	`.au`
SAMPLE RATE	$22050Hz$
EXCERPT LENGTH	30 secs
TOTAL EXCERPTS	1000
N^{O} CLASSES	10
CLASSES	`blues, classical, country, disco, hip-hop, jazz, reagge, rock, pop`
EXCERPTS / CLASS	10
AUTHORS (SUMMARY)	Mozart, Morphine, John Lee Hooker, Britney Spears...

Table 2.1: GTZAN *description (Sturm, 2012a)*

GTZAN FLAWS DESCRIPTION

EXACT REPLICAS	50
MISLABELLINGS	22
VERSIONS	13
CONSPICUOUS CLASSIFICATIONS[5]	43
CONTENTIOUS MISLLABELINGS[6]	63
ARTIST UNBALANCE	$\frac{35}{100}$ reggae songs are Bob Marley's songs, $\frac{24}{100}$ pop songs are Britney Spears, ...
QUALITY	1 sound is heavily distorted

Table 2.2: GTZAN *problems description (Sturm, 2012a)*

GTZAN is so common in MGR research that we also have a good understanding of its contents and flaws. We know, for instance, that GTZAN present problems such as exact

[5]Decisions made by GTZAN authors are controversial

[6]The audio content and its labelling does not fit musicological criteria

replicas or artist unbalance.

Therefore the fact that GTZAN is a widespread dataset in MGR research and the abundance of metadata and information about its contents qualifies it as a good benchmark for our purposes.

In their study of GTZAN, cite, offers a summary (Fig: 2.3) of the perceived performance in MGR using GTZAN. The also calculated a theoretical estimation of the expected performance for a "perfect classifier" that would peak at 94.5% average accuracy.

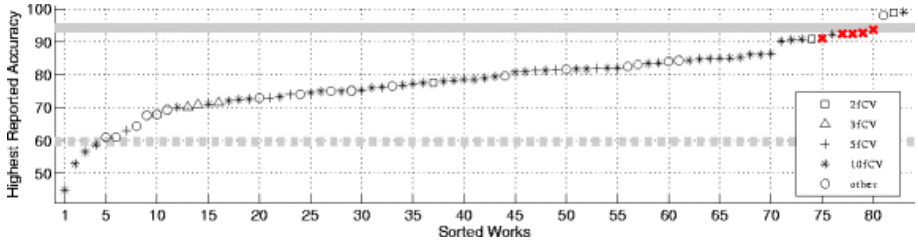

Figure 2.3: GTZAN*'s state of the art according to Sturm (2012a): each mark represents the accuracy declared in a study, shapes denotes the number of folds used to evaluate their results, red marks indicates papers that have been questioned in the literature and the upper grey horizontal line established the theorical maximum accuracy*

As we see (Fig: 2.3) the results follow a *"logarithmic"* shape, where the vast majority of papers reported accuracies between 70% to 80%. The best results (Bagci and Erzin, 2007; Panagakis et al., 2009; Chang et al., 2010) peak around 90% (some of them even exceed the theoric maximum Marques et al. (2011)) but it worth to mention that most of the papers reporting performances beyond 85% have been challenged or invalidated. (Sturm, 2013)

Thus, **the range of** 75% **to** 85% **seems to be a good assumption to describe the state of the art in** GTZAN.

2.2.3 What does *state of the art* really mean?

We have explored so far many examples of research papers in MGR, including generic approaches, research activities in MIREX and GTZAN driven works.

We conclude that classifiers applied to music genre recognition can be considered **good** if they reach an accuracy from 60% to 70% and **state of the art** from 70% to 85%, being 94.5% our theorical maximum performance.

We will use this description to describe (or discard) the performance of models in our experiment.

3 DESIGN AND METHODOLOGY

3.1 Representations of sound

The first factor in our experiments is pairing the algorithms with a time-based representation of GTZAN's songs.

We will use spectrograms to encode the sounds in our dataset, i.e., we have adopted visual representations as for the source of data for all the algorithms in our work.

Essentially, a spectrogram comes from a process which starts windowing a raw audio signal in several chunks and applying some calculations over them. This process ends up creating a set of coefficients able to be interpreted as pixels in an image. Thus, what differentiates two representations (at least for our purposes and the scope of this work) is the type of transformation they apply to the windowed excerpts.

We have studied four different representations of sound:

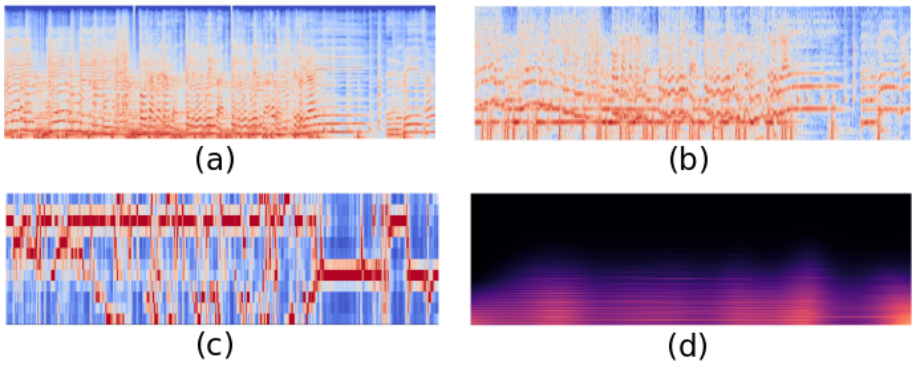

Figure 3.1: *Different spectrograms from the first Rock song in* GTZAN. *The first two examples are based in perceptual features and the other two use musical concepts. All of them describe the evolution of sound accross time (the x axis)*

a) **MELSPECTROGRAMS**

They apply a Short Time Fourier Transform (STFT) to each excerpt and map the results onto a Mel scale.

Perceptual features like the Mel scale its closely related Mel Cepstral Coefficients are

widely used in the literature, and considered by Zhouyu et al. (2010) as *"unarguably the single most important feature in* MGR*"*. The melspectrogram is no exception and it is the most used sound representation in MGR.

b) **CQT-SPECTROGRAMS** (Schörkhuber, 2010)

This representation substitutes the use of STFT for CQT transforms, and a logarithmic scale mapping.

Given a signal $x(n)$ a CQT transform is defined as:

$$X^{CQ}(k,n) = \sum_{j=n-\lfloor \frac{N_k}{2} \rfloor}^{n+\lfloor \frac{N_k}{2} \rfloor} x(j) a_k^*(j-n+\frac{N_k}{2}) \tag{3.1}$$

where:

$$a_k(n) = \frac{1}{N_k} \omega(\frac{n}{N_k}) e^{-i2\pi n \frac{f_k}{f_s}} \tag{3.2}$$

and a^* is the complex conjugate of a.

The f_k members are the center frequencies of each k bin and f_s denotes the sampling rate and ω is the hamming window. Since they use a logaritmic mapping, CQT-SPECTROGRAMS are also a *perceptual* representation.

c) **CHROMATOGRAMS** (Müller and Ewert, 2011)

A **chromatogram** it's a type of sound plot that represents the evaluation of harmony over time.

In a piece of music, we find that at any given moment several notes are being played at the same time with different intensities. The Chromatogram represents the intensity of each of the 12 pitch classes per instant using a colour code.

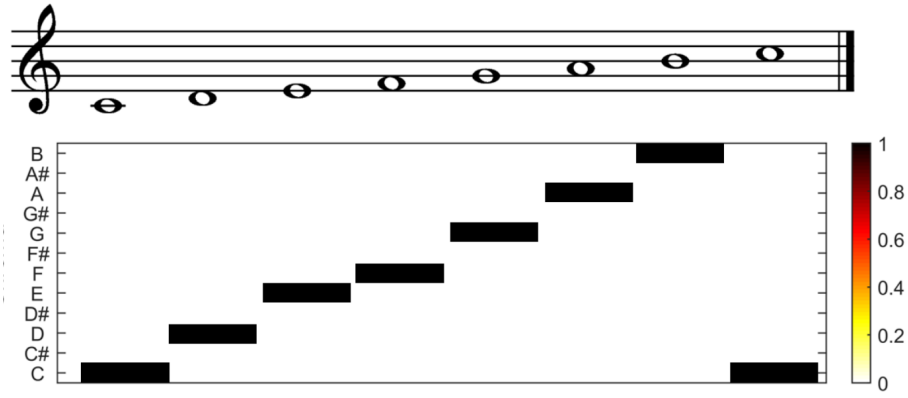

Figure 3.2: *A simplified case: chromatogram of a C-major scale*

d) **TEMPOGRAMS** (Grosche et al., 2010)

The process to calculate a chromatogram relies upon a similar procedure: we first apply a windowing process and then we calculate the STFT of each window.

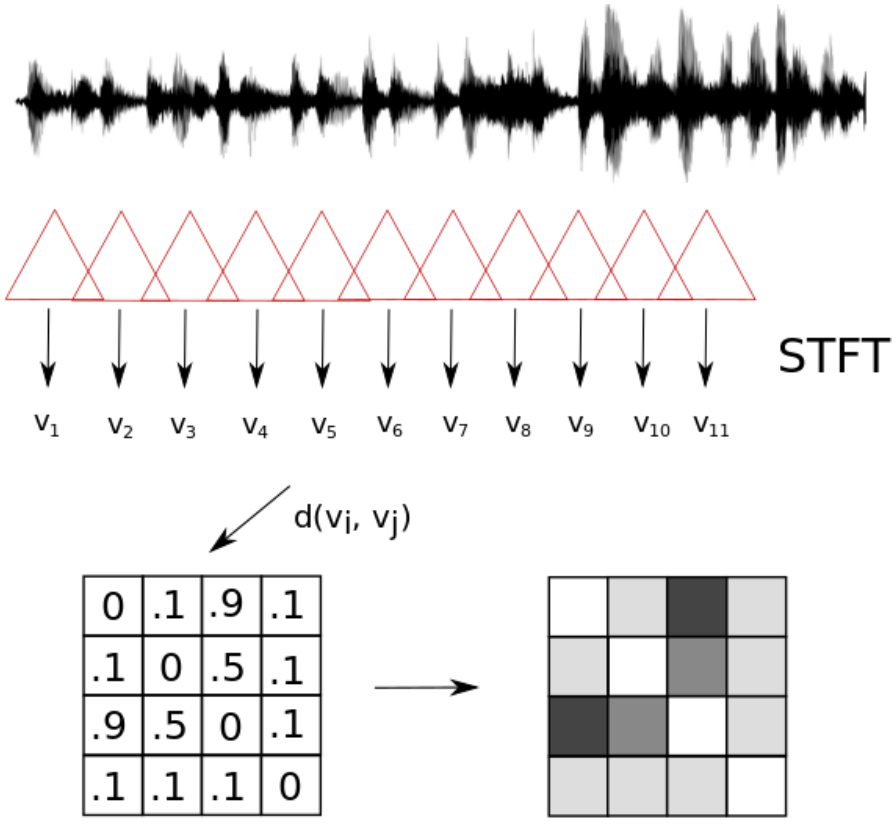

Figure 3.3: *Calculation process of a tempogram and its image interpretation in greyscale*

This produces a set of numbers $v_1, v_2, v_3, \ldots \in \mathbb{R}$ that we will use to create a self-similarity matrix using the cosine distance.

This self-similarity matrix is known as the **beat spectrogram** or **tempogram**.

So far, we are dealing with different time-based sound representations: MELSPECTROGRAM and CQT-SPECTROGRAMS implement logarithmic scales, i.e., they are grounded in perceptual or psychoacoustics aspects of sound. Meanwhile, CHROMATOGRAMS and TEMPOGRAMS try to grasp musicological concepts such as tempo and harmony.

Despite we have described all the algorithms that support these representations, in practice, we have used the four functions provided by LIBROSA to calculate them. Since our aim is not outperform the works found in the literature but to compare them, we have used all LIBROSA defaults in our experiments.

3.2 Temporal features in neural network design

Since the sound representations we have presented so far translate the sound waves into images (spectrograms) our neural networks design relies in convolutional layers.

A closer look on the spectrograms obtained from GTZAN songs reveals that we have two types of patterns to learn (Fig: 3.4): those that are expressed horizontally, and those expressed vertically.

Our intuition is that vertical patterns encode the sequential nature of music. Therefore we have developed a network geometry inspired in previous deep learning success on sequential data as natural language processing. I.e., we use unidimensional convolutions progressing on spectrograms' time axis followed by LSTM layers.

To contrast this setup with the effect of horizontal patterns, we also include a more traditional deep learning design, using bidimensional convolutions and dense layers.

This *dimensionality* of the convolutions is the way in which our algorithms will

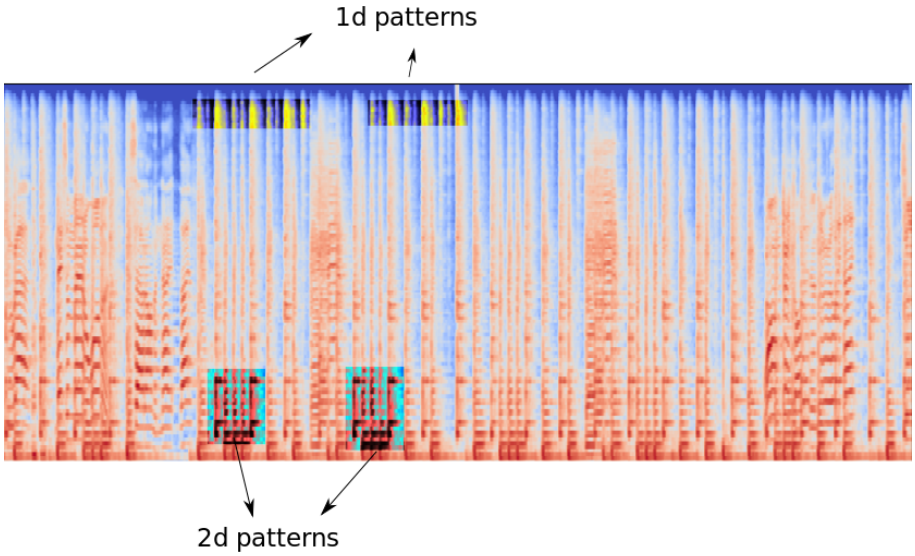

Figure 3.4: *Vertical and horizontal patterns over a* GTZAN *song spectrogram*

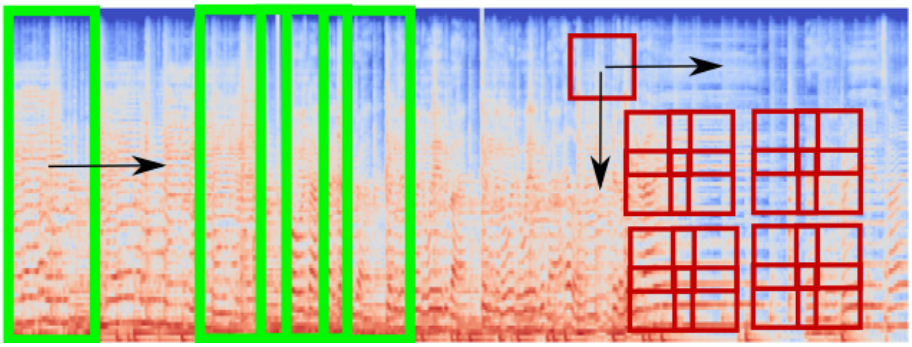

Figure 3.5: *Behaviour of convolutional layers on a spectrogram. 1D convolutions (green) evolve horizontally, meanwile 2D convolutions (red) evolve in both horizontal and vertical directions*

produce models adapted (or not) to the temporal features in GTZAN songs.

3.3 Data augmentation and voting

The design of the algorithms (1D and 2D) with which we have built the models of our experiments have been based on a trial and error process.

Due to the use of graphical sound representations and our intention to adapt the design revealing the temporal aspects in the GTZAN songs (See Fig: 3.4) the use of

convolutional neural networks is natural. On the other hand, we have combined these layers of convolution with dense layers or LSTM as is usual practice.

However, at least at the beginning, no configuration produced acceptable results for any representation of the concessions contained in GTZAN.

The reason why none of these tests proved successful is due to the lack of data in GTZAN. After all, this dataset only contains only 1000 audio excerpts divided equally into 10 musical genres. While other commonly used datasets raise the figure to several thousands or millions.

Following this intuition, we decided to implement a process of *data augmentation* or *feature engeenering* prior to the model creation phase, which significantly increased the performance of the models. From maximum figures around 40%, to exceed 80% in the best case scenarios.

In essence, this process consists of cutting each song of the TRAIN dataset into smaller chuncks overlapping each other and considering these chuncks as complete songs to be classified separately.

As indicated in Fig: 3.6, our criterion for cutting each song of TRAIN is to use excerpts of 3 seconds duration overlapped at 50%. This managed to increase the number of data points in TRAIN from 700 examples to the range of 10000, causing an increase in the effectiveness of the learning process.

This procedure has, however, a problem: training our algorithms on pieces of 3 seconds in length, the resulting models are only able to make predictions using 3 seconds length songs; but our original challenge was to classify the GTZAN tracks (which are the ones that make up the TEST dataset) that have a duration of 30 seconds.

To solve this problem we consider each model produced for a duration of 3 seconds as a *weak learner* in an bigger ensemble model that combines them using a voting system.

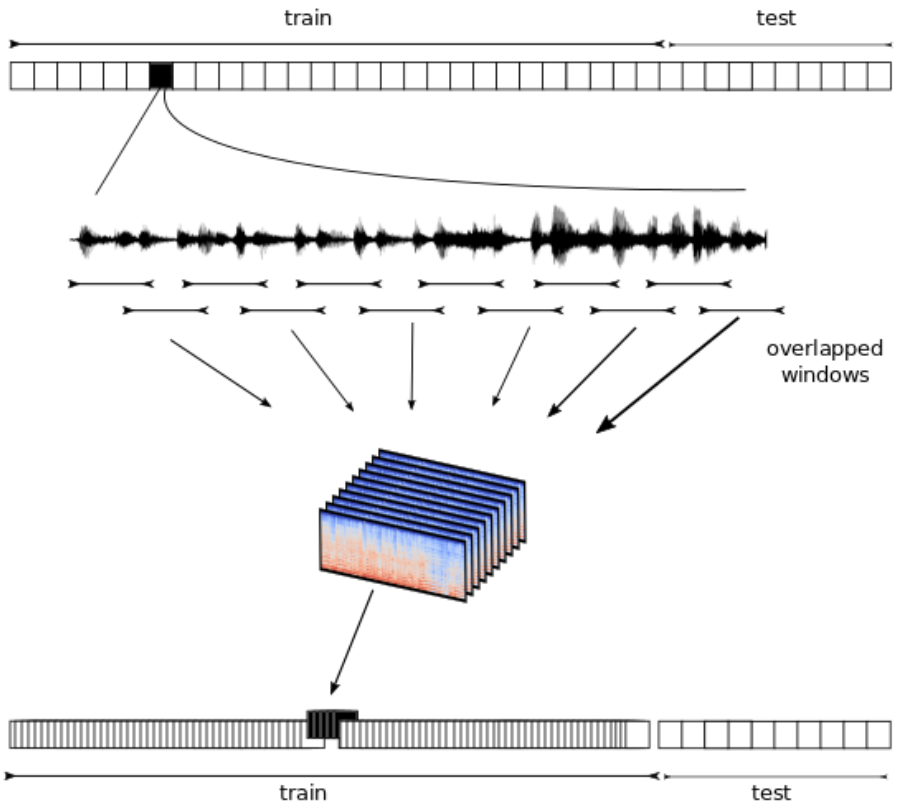

Figure 3.6: *Data agumentation process: each song in GTZAN is splitted in* 3*s chunks overlapped by 50% to create new data points*

Thus, when it comes to classifying a song of 30 seconds, first we reproduce the process of increasing the data for that particular song, obtaining several chuncks. Then we train and apply the *weak learner* to each one, obtaining an array of predictions (not necessarily coincident) from which we take the most common prediction of them all as our final output.

This process produced a significant increase in the results obtained. achieving valid models for the analysis.

3.4 Experimental design

3.4.1 Hypothesis

Once we have built models from our design of neural networks and using several different representations of the songs, we need an experimental method that allows us to draw conclusions.

In our case, we adopted an approach similar to that used in MIREX, i.e., using a *2 way ANOVA with repeated measures* statistical model, being the factors under study *the dimensionality of the networks* and *the sound representations* in use in each repeated measure.

This let us to test the following hypotheses:

- **Dimensionality**

  H_0 : There are no significant differences between the perceived performance using models from one-dimensional or two-dimensional networks.

- **Representation**

  H_0 : There are no significant differences in the perceived performance due to the use of the different representations of the sound used.

- **Interactions**

  H_0 : There are no significant interactions that affect the performance of the models between the use of algorithms of different dimensionality and different representations of sound.

3.4.2 Repeated measures

So far, we mentioned that our experimental design takes the form of a 2 way ANOVA design with repeated measures. Each of these measurements is an experiment in its own, consisting in the following steps:

1. Set a sound representation and an algorithmic design.
2. Create a TRAIN/TEST split using a particular random seed to randomize the original dataset. This step is what it truly differentiates one experiment to the next.
3. Calculate the chosen sound representation from the original audio files in GTZAN
4. Perform the augmentation process over the TRAIN dataset.
5. Train the selected algorithm over the augmented TRAIN dataset.
6. Evaluate the model on the TEST dataset using the voting system.
7. Store the performance metrics obtained for further analysis.

3.4.3 Statistical setup

The use of the ANOVA model for the case of repeated measures is due to the fact that, once an algorithm and a representation are set, what differentiates two models in our experiment is the initial randomization of the songs in GTZAN, used before creating the TRAIN / TEST partition. That is, we try to avoid the influence of a *"lucky split"* by correctly setting the sample size (n), the power of the test (β) and the level of significance (α)

To correctly choose these values, we have used the available scientific literature:

1. $\alpha = 0.05$ and $\beta = 0.8$ as observed in the papers presented in MIREX. (Stephens, 2007)

2. $n = 97$ calculated from the previous values using the *G*Power* (Karada and Aktag, 2012; Faul et al., 2007) software with the restriction of noticing *small effects*

4 IMPLEMENTATION

During our research and literature review we have found several papers describing ideas and works on the topic, from a generic perspective and directly applied to MGR and GTZAN dataset. Regardless of the theories and ideas that may contain, there is a problem common to all of them: *the impossibility to reproduce the results*.

Either because of the lack of access to the original data that supports the study or because the authors do not provide the code used to process them; the truth is that the content of most papers contain only a summary of the methods used and a description of their results.

The lack of reproductability reduces our confidence with respect to the conclusions of these studies, i.e., we must trust the authors of the studiess or its reviewers. Since there is no way to confirm the conclusions independently. It is also not possible, at least it is not easy, to develop new versions based on previous work.

This work does not follow this practice: reproductability has been a main criteria to implement the code and has influenced the design of the software necessary for experimentation.

In this chapter, we will describe the implementation developed with attention to the tools developed to promote the reproduction of the experiments.

4.1 Requirements, tools and libraries

This project has been developed under Linux using a Python 3.6.5 distribution, that uses Anaconda to manage all its dependencies. Hence the necessary environment to replicate this work can be replicated following a these steps:

1. Install Anaconda Python[1]

2. Install Git[2]

3. Clone the Bitbucket repo containing the code.

   ```
   git clone git@bitbucket.org:victor-santiago/thesis.git
   ```

4. Replicate the development environment using conda tools:

   ```
   conda env create -f requirements.yml;
   source base;
   ```

The *"requirements.yml"* includes a llist of all third party libraries in use and their own dependencies. But only a few of them are relevant to describe the project:

- **Tensorflow and Keras**

  Both libraries are in use to implement the neural networks that produce the models in our experiments, being Keras a frontend to use the tensorflow library.

- **Librosa (McFee et al., 2015)**

  A DSP library that we use to encode and calculate the sound representations described in chapter 3.

[1] See https://www.anaconda.com

[2] See https://git-scm.com/

4.2 A data science framework

All the necessary code to implement the experiments in this work follows the principles and ideas presented in Cookiecutter (2018), a data science framework focus on Python development.

This framework, basically, creates a folder structure and promotes the use of `make` scripts to deal with common tasks and reproduce the experiments.

The folder structure proposed by this framework introduce the use of the following folders:

- `/data/`

  All data files will be stored here.

- `/data/external/`

  A folder to store all external datasets. In our case a copy of GTZAN will be stored here after download.

- `/data/raw/`

  Raw versions of data, such as GTZAN audio files.

- `/data/interim/`

  Semi-processed datasets, converted audio files...

- `/data/processed`

  Fully processed data files: features extraced from the audio viles, `.csv` results...

- `/data/temp/`

  Temporal our auxiliary files.

- `/src/`

  Python files.

- `/src/data/`

  Code implementing `make` targets.

- `/src/features/`

  Scripts that implement feature extraction tasks.

- `/src/models/`

  Scripts that implement our models and their training phase.

- `/src/utils/`

  Common functions and scripts.

- `/references/`

  Metadata.

- `/models/`

  Trained models are stored here.

- `/notebooks/`

  We will find here several notebooks where we conducted our statistical analysis on our experimental results.

Let alone with the folder structure, the framework promotes the use of `make` files to deal with common tasks. In our case these tasks include downloading the GTZAN dataset, pre-processing, model training, etc... Hence, the `make` commands are the tools that let us to reproduce the experimental process from scratch.

To reproduce an experiment, we need to run a series os steps: download the GTZAN dataset, extract the audio files and convert them to `.wav` format, calculate features from these audio files (i.e., calculate a particular sound representation), split the dataset into a TRAIN/TEST split, train a model and evaluate it against the songs included in the TEST dataset.

The make scripts help us to deal with all the complexity of these tasks. Despite the `make` command provides its own documentation, it worth to describe each of them in detail:

- **Gathering data**: `make data`

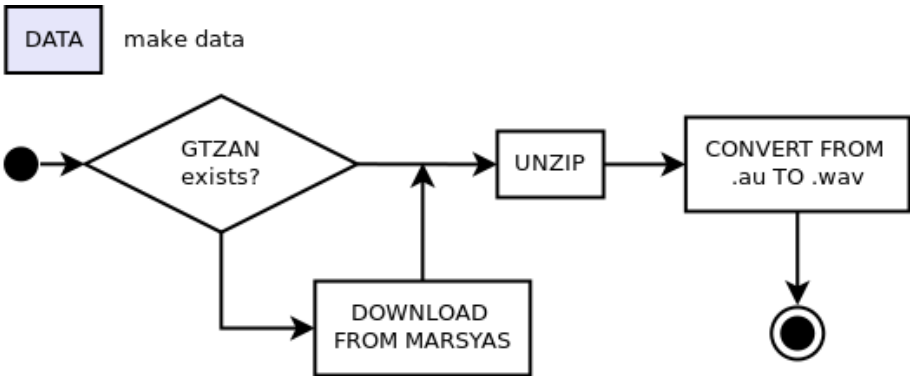

Figure 4.1: *Data gathering process, activities diagram*

  This command downloads the dataset from it source (if it hasn't been downloaded yet), extracts its contents and pre-process the audio files.

- **Train a model**: `make train`

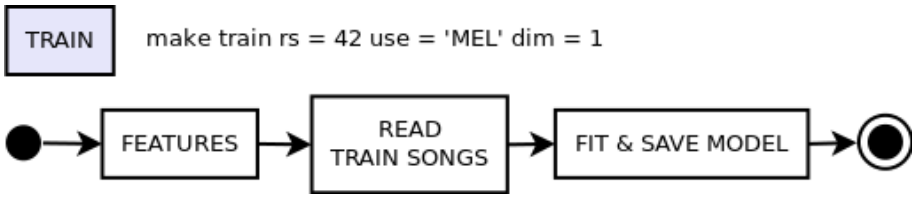

Figure 4.2: *Train process, activities diagram*

  This command superseeds the previous and adds a model training on the TRAIN dataset.

- **Create TRAIN/TEST split and complete sound representation**: `make features`

  The arguments for this command let us define the sound representation (use) the dimensionality (dim) and the random seed (rs) in use to create the TRAIN/TEST split.

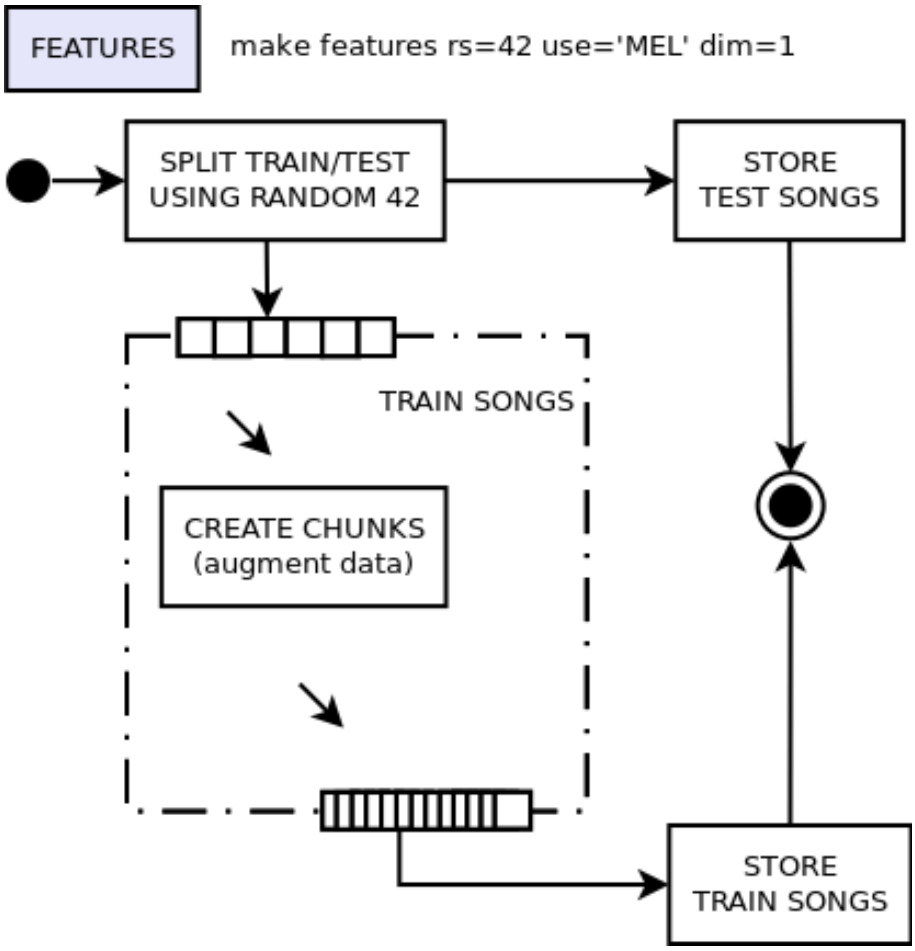

Figure 4.3: *Feature creation process, activities diagram*

- **Evaluation of a model**: `make eval`

  This command implements a full experiment process, i.e., if performs the tasks included in the previous `make` scripts and it also evaluates the trained model and store the results. The argument `batch` let us execute the evaluation process in a loop, hence, this is the routine we have used to complete our analysis.

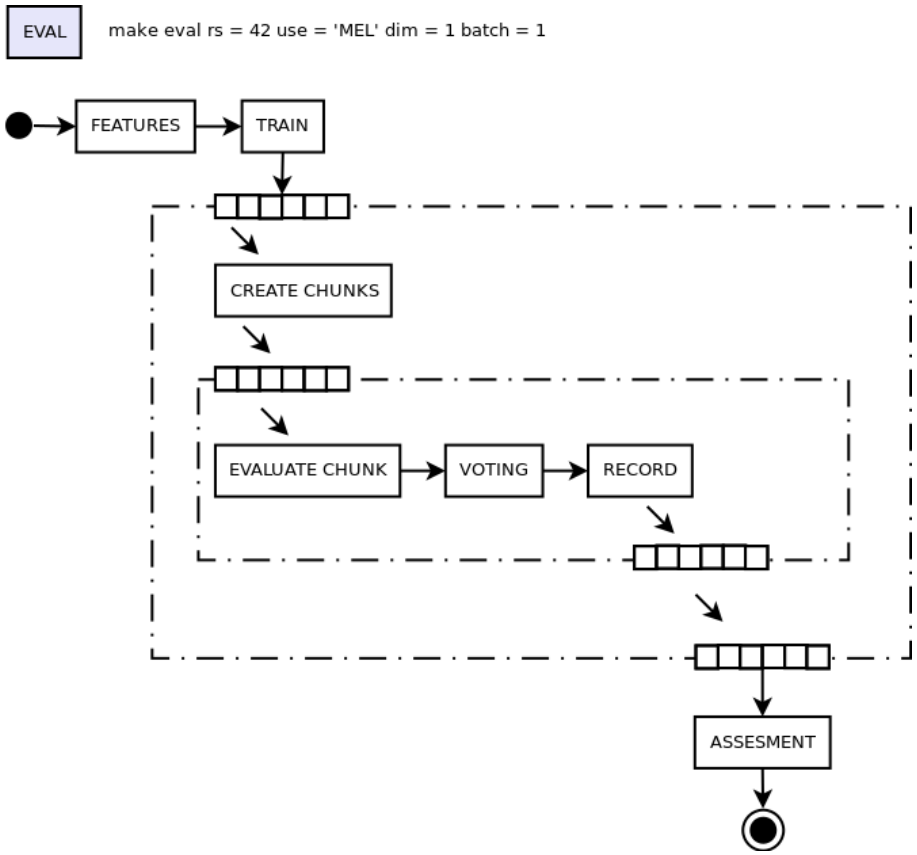

Figure 4.4: *Evaluation process, activities diagram*

All the experiments in this work have been produced making use of these commands. Particularly, running executions of:

```
make data;
make eval rs=0 use='MEL' dim=1 batch=97;
make eval rs=0 use='MEL' dim=2 batch=97;
make eval rs=0 use='QT' dim=1 batch=97;
make eval rs=0 use='QT' dim=2 batch=97;
```

Therefore the results can be reproduce running the same commands.

4.3 Notes on implementation

All processes and functions in the code are built upon two main objects, implemented in `Features.py` and `Model.py` files.

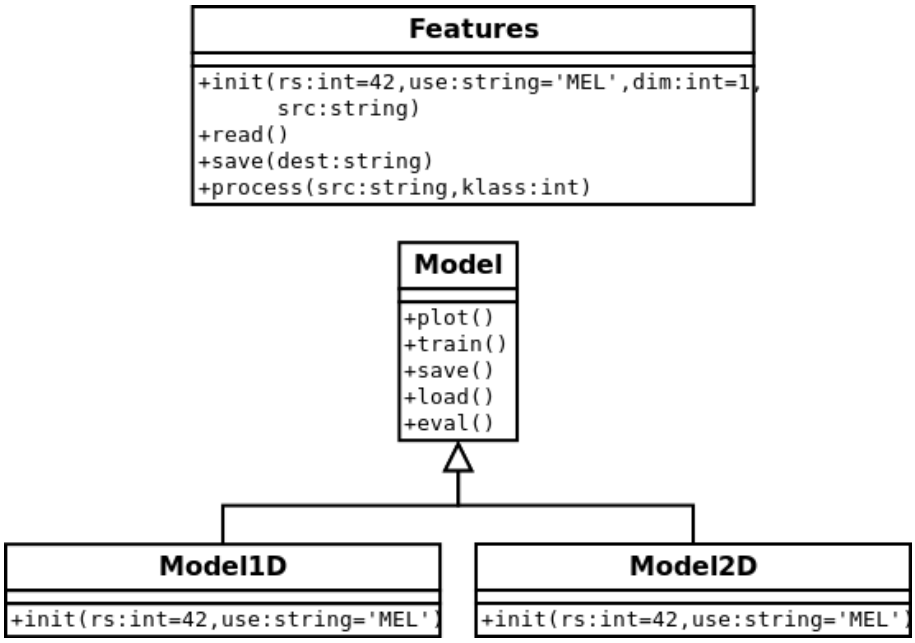

Figure 4.5: *Feature class makes a distiction between 1D and 2D initializations which produces appropiate output shapes for the same representations*

`Features` takes the role of enconding all the sound representations considered in our experiment and save them using the appropiate shape for each dimensionality setup.

`Model` implements an inheritance patter where each children defines a single convolutional neural network (1D or 2D) of their convolutional frames. Table 4.1 and Table 4.2 describe in detail the geometry of both models.

Layer (type)	**Output shape**	**Params**
Conv1D_1	(None, 84, 16)	6208
Max_Pooling1D_1	(MaxPooling1, (None, 42, 16))	0
Dropout_1 (Dropout)	(None, 42, 16)	0
Conv1D_2 (Conv1D)	(None, 42, 32)	1568
Max_Pooling1d_2	(MaxPooling1 (None, 21, 32))	0
Dropout_2 (Dropout)	(None, 21, 32)	0
Conv1d_3 (Conv1D)	(None, 21, 64)	6208
MaxPooling1d_3	(MaxPooling1 (None, 10, 64))	0
Dropout_3 (Dropout)	(None, 10, 64)	0
Lstm_1 (LSTM)	(None, 100)	66000
Dense_1 (Dense)	(None, 10)	1010
Total params: 80,994		
Trainable params: 80,994		
Non-trainable params: 0		

Table 4.1: *Keras summary for 1D models*

Layer (type)	**Output shape**	**Params**
Conv2d_1 (Conv2D)	(None, 84, 129, 8)	80
MaxPooling2d_1	(MaxPooling2 (None, 42, 64, 8))	0
Dropout_4 (Dropout)	(None, 42, 64, 8)	0
Conv2d_2 (Conv2D)	(None, 42, 64, 16)	1168
MaxPooling2d_2	(MaxPooling2 (None, 21, 32, 16))	0
Dropout_5 (Dropout)	(None, 21, 32, 16)	0
Conv2d_3 (Conv2D)	(None, 21, 32, 32)	4640
MaxPooling2d_3	(MaxPooling2 (None, 10, 16, 32))	0
Dropout_6 (Dropout)	(None, 10, 16, 32)	0
Flatten_1 (Flatten)	(None, 5120)	0
Dense_2 (Dense)	(None, 100)	512100
Dropout_7 (Dropout)	(None, 100)	0
Dense_3 (Dense)	(None, 100)	10100
Dropout_8 (Dropout)	(None, 100)	0
Dense_4 (Dense)	(None, 10)	1010
Total params: 529,098		
Trainable params: 529,098		
Non-trainable params: 0		

Table 4.2: *Keras summary for 2D models*

Both models can use each sound representation provided by the `Features` object. They use a shared training phase (`Model.fit`) that contains a few important parameters and techniques:

- A halt guard: to prevent overfitting we stop the training phase after 15 epochs without improvement. Where *improvement* means a perceived reduction in validation loss greater or equal than 0.001

- L1 regularizers and dropout layers to prevent overfitting.
- We use an Adam optimizer (Diederik and Ba, 2014) configured using the author's recommendation, i.e., $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$
- A 25 batch size. This value was selected due to hardware constraints in the particular machine where we run the experiments. Higher values could have produced memory overflow errors.
- A maximum of 400 epochs. Althought the halt condition usually stops the process before this limit.
- The use of a validation set (an extra partiton which uses a 20% of the TRAIN songs) to control the training of each algorithm.

All the configuration options mentioned above have an impact on the training phase.

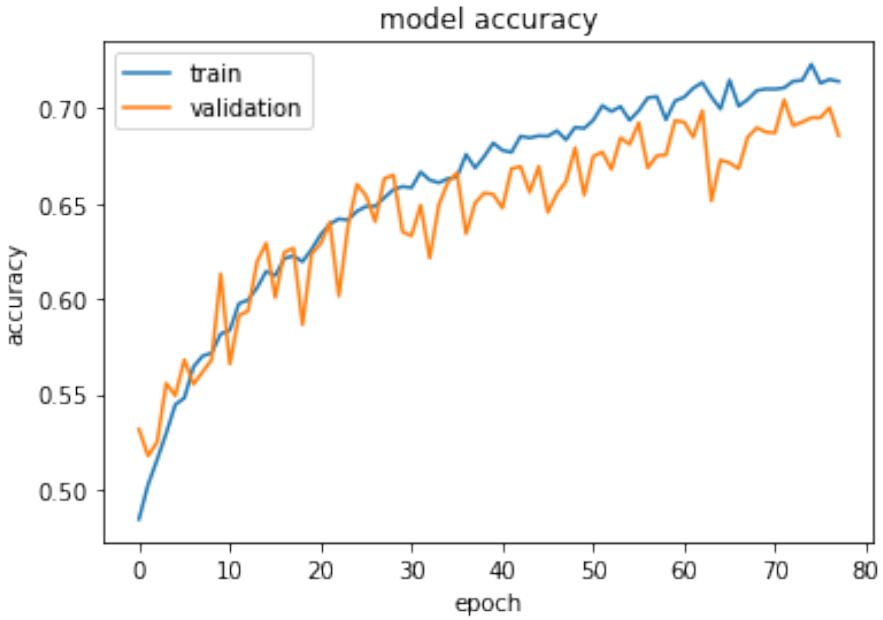

Figure 4.6: *Accuracy evolution against the* TRAIN *and* VALIDATION *datasets using* MELSPECTROGRAMS *and a* 1*D algorithm*

Figure 4.7: *Loss evolution against the* TRAIN *and* VALIDATION *datasets using* MELSPECTROGRAMS *and a* 1*D algorithm*

Particularly, they apply an early stop policy which reduces the number of epochs, preventing the learning process to adopt a "hockey stick" shape, both in terms of validation and training accuracy and loss.

Since the evolution of validation loss figures is stochastic (Fig: 4.6) with high variability across epochs, that also means that our resulting models could achieve greater accuracies if we used a more exhaustive training policy. However, our aim is to compare the factors involved in the experiment, not to produce highly accurate models. Therefore we decided to stop training as soon as the validation accuracy growth reached a plateau.

4.3.1 Mixing languages: why do we use both Python and R?

We have used Python 3.*X* to implement the vast majority of the scripts in this work: neural network design, training and evaluation, the data augmentation tasks, GTZAN downloading and processing...

However, the ANOVA analysis (an important part of the project, commented in Chapter: 5) has been implemented under a R-powered jupyter notebook.

The reason for this is that Python 2-way ANOVA implementations [3] do not run under Python version 3.*X*, and the alternative (Python Statsmodels) does not have the 2-way ANOVA with repeated measures use case implemented. On the best of our knowledge, none of Python's 3.*X* provide functions to perform this experimental design, but R alternatives are widely available.

Since we based our neural network design in Keras (which requires Python 3.*X*) using both Python and R looks like a reasonable workaround.

4.4 Results

After each evaluation process the trained model is used to calculate a set of performance metrics:

- **Multiclass confusion matrix.**

  A detailed accountability of correctly and incorrectly classified data points per class.

- **Per class recall**

  When a song belongs to a certain category, how often the model classifies it correctly.

- **Per class precision.**

  How often the algoritm is correct when it classifies a song in a certain class.

- **Overall accuracy**

  Overall, how often the model is correct.

[3] The common Python library for this tasks

The outcome of an evaluation process is automatically stored in a `.csv` file named after the parameters used during the evaluation. For instance, a `make eval rs=42 use='MEL' dim=1 batch=1` execution will produce the file

`/data/processed/results/MEL/1D/42.csv`

containing the aforementioned metrics.

We will use these results (the `.csv` files) as starting point for our statistical analysis, developed in the jupyter notebooks and commented in chapter 5. In particular, precision and recall are used to explore the inter-class behaviour of the algorithms meanwhile we will use accuracy to compare the performance of different models.

5 DISCUSSION

As we have already mentioned (Section: 3) our experimental design consists of a 2-way ANOVA test with repeated measures, where the algorithm's dimensionality (*dim*) and the sound representation (*repr*) are the two factors studied and we recorded 97 samples of the overall accuracy per each combination of their levels.

During our experimentation we noticed that only MELSPETROGRAMS and CQT-SPECTROGRAMS were able to achieve overall accuracies up to 50%. Therefore, we excluded those levels in the *dim* factor corresponding to CHROMAGRAMS and TEMPOGRAMS, due to their poor performance in our experimental setup.

Thus, we are considering onl the accuracy obtained using MELSPECTROGRAMS and CQT representations as levels for the *repr* factor and $1D$ or $2D$ as the levels for *dim* factor, i.e., our ANOVA repeated measures table is as follows:

DIM / REPR	MEL	CQT
1D	mel_{1D} (x 97)	cqt_{1D} (x 97)
2D	mel_{2D} (x 97)	cqt_{2D} (x 97)

Table 5.1: ANOVA *repeated measures breakdown*

This means that we have created and evaluated 97 models per each group. The results can be found at `/data/processed/results/` folder. And they are easily reproductable using the `make` commands detailed in chapter 4.

Let alone with the overall accuracies we have also recorded partial metrics like per-class precision and recall. We will use these metrics to analyse the behaviour of each combination of the levels in our experiment in detail.

Here, we will only describe the results and conclusions we can draw from this data. However a complete breakdown of our statistical analysis (including Python and R's scripts) are available via two notebooks at:

`notebooks/CHECKING ANOVA ASSUMPTIONS.ipynb`

`notebooks/ANALYSIS.ipynb`

included in this project's bitbucket repository.

5.1 Checking ANOVA assumptions

The ANOVA design has been built on top of a number of statistical assumptions about our experimental practice and the shape of our data.

1. **The samples must be independent.**

   This is a consequence of the process we have followed to obtain each sample: once a sound representation and the algortihm are set, the training and evaluation phases do not depend on previous results. Thus, all samples are independent.

2. **The groups must have the same sample size.**

   All groups in our experiment have 97 examples.

3. **The populations from which the samples were obtained must be normal.**[1]

   We have found evidence of normality for each combination of factor's levels using *QQ-plots* and *histograms* (Fig: 5.1 and Fig: 5.2)

   All cases show symetric histograms and data points lie closely to the diagonal, the present only small departures from normality at the extremes.

[1] We used graphical methods to assess normality due to a relatively small sample size that can undermine analytical normality tests like Shapiro-Wilks. Either way, normality assumption for ANOVA is a weak condition, since the F-test has been reported reliable even with approximately normal data. (Glass. et al., 1972; Harwell et al., 1992)

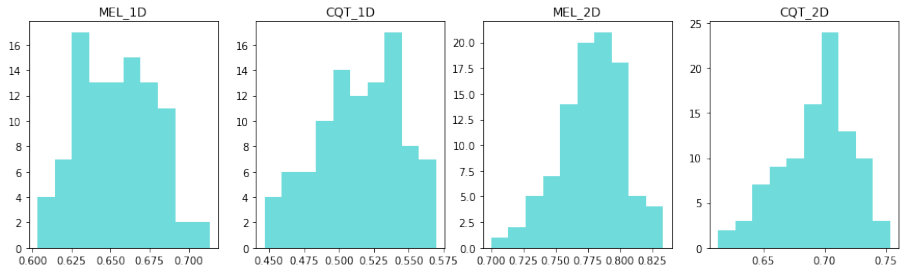

Figure 5.1: *Sample groups histograms*

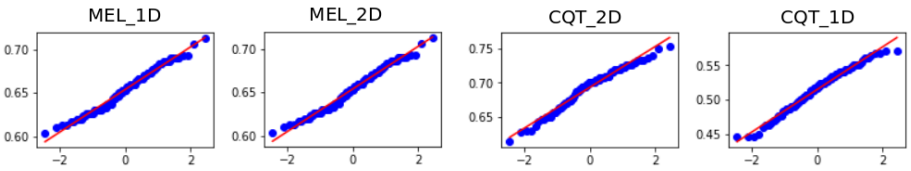

Figure 5.2: *Sample groups QQ-plots*

4. **Homocedasticity: all the populations have the same variance.**

   Bartlett test did not find evidence to reject the hypothesis that all populations have the same variance ($T = 7.451$, $p - value > .05$)

5.2 ANOVA results

The overall accuracies obtained in our experiments were subjected to a two-way analysis of variance (See the results in Tables: 5.2 and 5.3 and Fig: 5.3) having two levels: algorithmic dimensionality ($1D$ or $2D$) and two levels due to the representation of sound in use (MEL and CQT). All effects are statistically significant at the .05 significance level.

	Sum Sq.	Df.	F	Pr(>F)
dim	2.19953310	1	2884.42369	1.188813e-180
repr	1.19728396	1	1570.09423	9.700310e-138
dim:repr	0.07515464	1	98.55629	7.952936e-21
Residuals	0.29282131	384	NA	NA

Table 5.2: ANOVA *test results.*

DIM / REPR	MEL	CQT
1D	$\bar{x} = 65\%, \sigma = 2\%$	$\bar{x} = 51\%, \sigma = 3\%$
2D	$\bar{x} = 78\%, \sigma = 3\%$	$\bar{x} = 69\%, \sigma = 3\%$

Table 5.3: *Experiment results summary*

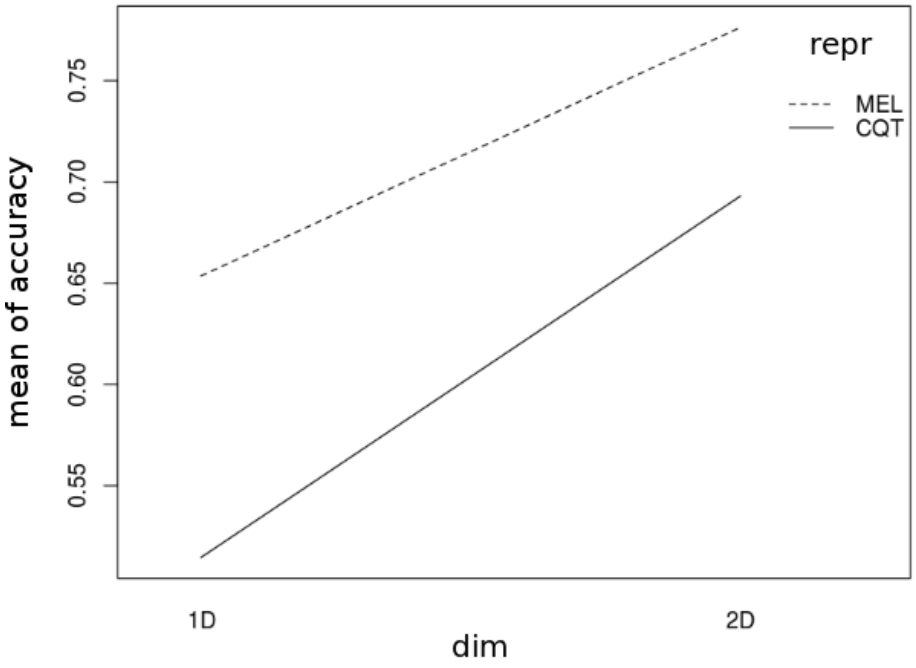

Figure 5.3: *Interaction effects plot*

The main effect of algorithmic dimensionality yielded an F-ratio of $F = 2884.42369$, $p - value < 0.05$ indicating that the mean accuracy of the models produced by $2D$-driven algortihms was significantly greater than the accuracy of $1D$ models.

The main effect of sound representation yielded an F-ratio of $F = 1570.09423$, $p - value < 0.05$ indicating that the mean accuracy obtained by models using MELSPECTROGRAMS is significantly greater than the mean accuracy of those models produced using CQT-SPECTROGRAMS

The interaction effect between dimensionality and sound representation ($F = 98.55629$, $p - value < 0.05$) was found significant.

Hence, we see that dimensionality has a positive impact in the performance: 2*D* algorithms obtain better results than unidimensional setups. There is also a weak interaction effect (the two non-parallel lines at 5.3 have almost the same slope) showing that algorithmic dimensionality affects more when the algorithm uses CQT-SPECTROGRAMS.

Having said that, the sound representation seems to be the key factor: MELSPECTROGRAMS outperforms CQT-SPECTROGRAMS regardless the dimensionality used to built the model.

5.3 Precision and recall per-class

In adition to overall accuracy, the evaluation phase of each model also recorded its precision and recall scores per class. Now, averaging these metrics we can analyse each group behaviour per class (See Figs: 5.4, 5.5, 5.6 and 5.7)

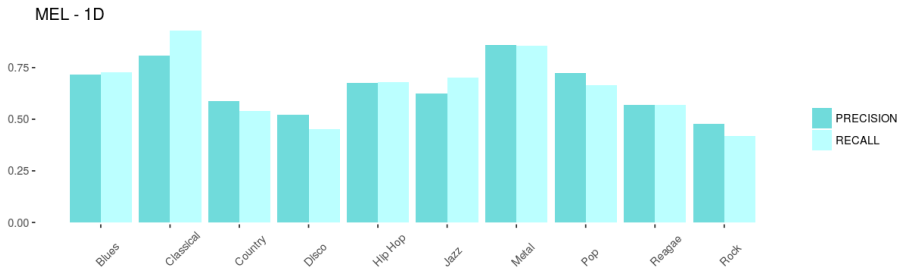

Figure 5.4: *MEL_1D per-class performance*

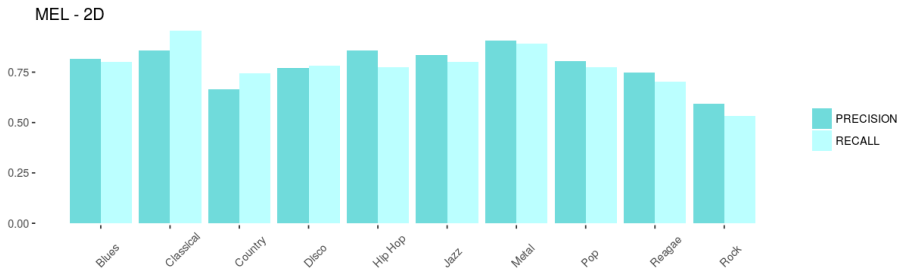

Figure 5.5: *MEL_2D per-class performance*

Classical and *Metal* genres are the two best categories with both precision and recall peaking up to 80% using MELSPECTROGRAMS, meanwhile *Country*, *Disco* and *Rock*

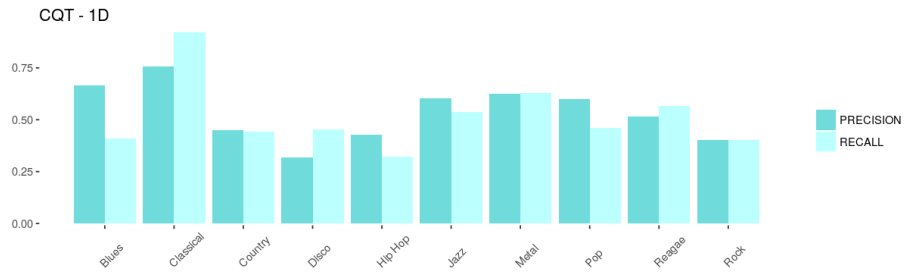

Figure 5.6: *CQT_1D per-class performance*

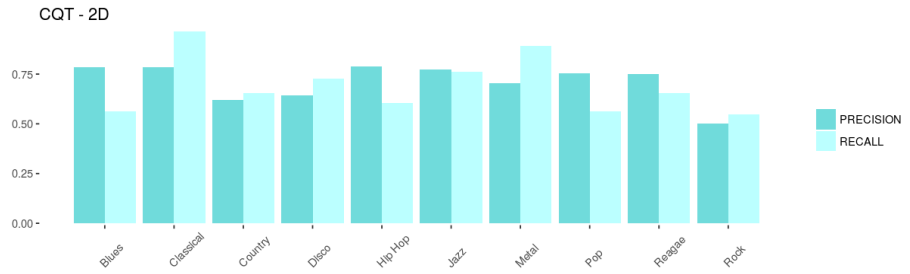

Figure 5.7: *CQT_2D per-class performance*

present lower figures. This behaviour across categories is dimension independent

In terms of per-class performance using MELSPECTROGRAMS, the 2-dimensional convolutional networks perform better. Not only because the figures per each class are higher, but because they are more uniform, i.e., we appreciate less dispersion.

Using CQT-SPECTROGRAMS, we find a similar behaviour in the per-class charts: *Classical* and *Metal* are the top categories, being *Country*, *Disco* and *Rock* the genres were the algorithms peform worst. Again, the performance observed using bidimensional convolutions is better and more uniform between genres than the figures obtained using unidimensional algorithms.

Except for *Classical* songs (they all peak up to 80%) the performance observed in the charts describing CQT experiments turned out to be worst than those using MELSPECTROGRAMS.

This situation is consistent with the results of our ANOVA analysis, where we found the

2*D* based models better than their unidimensional counterparts; and MELSPECTROGRAMS better than CQT-SPECTROGRAMS

Regarding the per-class performance breakdown, the models meet our intuition: songs in *Classical* and *Metal* genres are quite distintive from a *listener* perspective, whilst *Country*, *Rock* and *Disco* are similar genres. Actually, we can think about *Rock* and *Disco* as derived genres from *Country* (See Fig: 1.1)

6 CONCLUSION

6.1 Research overview

Unlike many other studies in MGR our project does not try to create a high-performance classifier. We have focused our efforts instead in exploring which is the impact of temporal factors in MGR classifiers design.

We have used a common modern approach, based in convolutional neural networks and visual sound representations. We have also applied a data augmentation technique which was the key to obtain models close to the state of the art and, on the best of our knowledge, is not in use in any academic research project.

However, in our opinion, the most valuable insight that can be drawn from this study is the incidence of sound representation, algorithmic design and their relationships.

6.2 Design and experimentation

Based on exhaustive research on scientific literature (we have reviewed 20 years of MIREX submissions and GTZAN specific research) we have addressed the problem of classifying musical genre in natural songs.

This is an important and widely studied (but still open) problem in MIR that have received much attention from the scientific community. Thus, a wide range of ideas and techniques have been tested in the last decade.

While most of this studies tried to borrow data science findings and applied them to the MGR problem, our work focused on figuring out what are the causes that make a MGR classifier successful, rather than trying to just create another brand new model.

6.3 Contributions and impact

Our implementation design uses a common ensemble approach: the voting system commented in section 3.3. However, this design (due to the need to augment GTZAN data)

ended up being a more simple setup than many other GTZAN-based state of the art designs, which are often based on a combination of cutting-edge ensemble techniques (AdaBoost, Bagging) and deep DSP knowledge.

Despite we have used a relatively straight configuration, we could create models that meet general MGR standards and, in the best case scenario (MELSPECTROGRAMS and $2D$ networks), are close to GTZAN state of the art, which ranges from 70 to 85%.

We think about this performance achievement as a solid basis to fundament our conclusions, i.e., the models are good enough to be considered. Therefore, the conclusions about what factors influence the performance might be valid, if they reach statistical significance.

Regarding our results, we have found that certain sound representations are more likely to produce good models. **Mel-based representations perform best in all our experiments**, which confirm previous research in the field.

The presence of $2D$ and $1D$ patterns in song spectrograms (See section: 3.2) play a crucial role: those algorithms using $2D$ **convolutions always produce better models**, regardless of the sound encoding used during training.

On the other hand, we found an interaction effect between dimensionality and the sound representation: a $2D$ design has a more positive impact when the model is trained using CQT-SPECTROGRAMS. The improvement due to dimensionality is also present for MELSPECTROGRAMS but the effect is smaller.

On the best of our knowledge, the inspiration of the research question and the experimental design are the main novelty of this work.

Both of them could be easily adapted to incorporate new sound representations or algorithmic design variability.

6.4 Future work

We have used a conservative approach in our algorithmic desing: convolutional neural networks ($1D$ or $2D$) have become a *de-facto* standard in recent years. That was appropiate to our research goals, since our aim was to contrast the effects of temporal features in MGR classification via sound representation and model design, i.e., we needed to keep the experimental design simple using algorithms that could produce models from each sound representation.

Hence, we rejected those sound representations (CHROMAGRAMS and TEMPOGRAMS) that showed poor performance. But even if they don not carry information enough to produce good classifiers, they are both music-grounded representations (they encode *tempo* and *harmony*) that could play a benefitial role for an experiment focused in industry interests, i.e., to create accurate classifiers.

That will lead to more complicated (but interesting) algorithms, mixing several sound representations as the input of our neural networks.

On the other hand, we have conducted a *supervised* learning experiment, and we have found that genre similarity (from a musichology perspective) leads to poor performance in self-related categories such as *Country, Rock* or *Disco.*

It is not a bold assumption, thinking that an *unsupervised learning* process would have classified many of GTZAN's *Rock, Disco* or *Country* songs into a single cluster.

Since the habits of online music consumers rely in MGR to browse the catalogues of streaming publishers and the emergence of new platforms (Epidemic Sound, for instance) where the business model is focused on open-source music to play the role of background music for third-party creators, in our opinion an *unsupervised* version of this project could meet industry needs and a comparision between these to approaches could constitute an interesting research.

A MIREX MGR BREAKDOWN

MIREX ID	Paper	Algorithm	Avg. Accuracy
GH	Guaus and Herrera (2007)	SVM	62.8%
TL	Lidy et al. (2007)	SVM	66.71%
ME	Mandel and Ellis (2007)	SVM	66.6%
GT	Tzanetakis (2007)	SVM	65.34%

Table A.1: *MIREX 2007 submissions*

MIREX ID	Paper	Algorithm	Avg. Accuracy
CL1	Cao and Li (2008)	SVM	62.04%
CL2	Cao and Li (2008)	SVM	63.39%
GP1	Peeters (2008)	SVM	63.9%
GT1	Tzanetakis (2008)	SVM	64.71%
GT2	Tzanetakis (2008)	SVM	66.41%
GT3	Tzanetakis (2008)	SVM	65.62%
LRPP1	Lidy et al. (2008)	SVM	65.06%
LRPP2	Lidy et al. (2008)	SVM	62.26%
LRPP3	Lidy et al. (2008)	SVM	60.84%
LRPP4	Lidy et al. (2008)	SVM	60.46%
ME1	Mandel and Ellis (2008)	SVM	65.41%
ME2	Mandel and Ellis (2008)	SVM	65.3%
ME3	Mandel and Ellis (2008)	SVM	65.2%

Table A.2: *MIREX 2008 submissions*

MIREX ID	Paper	Algorithm	Avg. Accuracy
BP1	Burred and Peeters (2009)	SVM	77.61%
BP2	Burred and Peeters (2009)	SVM	76.21%
CL1	Cao and Li (2009)	SVM	80.48%
CL2	Cao and Li (2009)	SVM	80.61%
GLR1	Grecu et al. (2009)	SVM	78.51%
GLR2	Grecu et al. (2009)	SVM	69.11%
HNOS1	Hasegawa et al. (2009)	SVM	72.93%
HNOS2	Hasegawa et al. (2009)	LINEAR	23.22%
HNOS2	Hasegawa et al. (2009)	SVM	72.97%
HNOS3	Hasegawa et al. (2009)	SVM	72.97%
HW1	Wang (2009)	SVM	74.33%
VA1	Lidy et al. (2009)	SVM	76.53%
VA1	Lidy et al. (2009)	SVM	75.56%
LZG	Liu et al. (2009)	SVM	76.29%
RCJ1	Ren et al. (2009)	KNN	44.08%
RCJ2	Ren et al. (2009)	KNN	49.46%
RCJ3	Ren et al. (2009)	SVM	61.2%
RK1	Rao and Kini (2009)	GMM	70.2%
SS	Rao and Kini (2009)	SVM	74.54%
TTOS	Tsunoo et al. (2009)	SVM	76.47%
MTG1	Wack et al. (2009)	SVM	73.05%
MTG3	Wack et al. (2009)	SVM	71.95%
MTG4	Wack et al. (2009)	SVM	71.69%
MTG5	Wack et al. (2009)	SVM	77.69%
XLZZG	Xu et al. (2009b)	SVM	76.54%
XZZ	Xu et al. (2009a)	SVM	77.25%

Table A.3: *MIREX 2009 submissions*

There are no data available for 2010 and 2011 editions: the links are broken.

MIREX ID	Paper	Algorithm	Avg. Accuracy
DM3	de León and Martínez (2012a)	KNN	60.74%
DM4	de León and Martínez (2012b)	KNN	58.34%
GT1	Ren and Jang (2012)	SVM	67.97%
LBLJK1	cheon Lin et al. (2012)	SVM	73.43%
LBLJK2	cheon Lin et al. (2012)	SVM	65.96%
PH1	Hamel (2012)	NN	71.95%
PP1	Panda and Pavía (2012)	SVM	74.43%
PP2	Panda and Pavía (2012)	SVM	67.73%
SSKS1	Seryerlehner et al. (2012)	OTHER	74.9%
WJ1	Wu and Jang (2012)	SVM	76.12%
WJ2	Wu and Jang (2012)	SVM	76.11%
YCP2	Yang et al. (2012)	SVM	72.81%

Table A.4: *MIREX 2012 submissions*

MIREX ID	Paper	Algorithm	Avg. Accuracy
BBLK1	Byum et al. (2013a)	SVM	73.76%
BBLK2	Byum et al. (2013b)	SVM	72.9%
BBLK3	Byum et al. (2013b)	SVM	63.2%
BBLK4	Byum et al. (2013b)	SVM	69.83%
BBLK5	Byum et al. (2013b)	SVM	65.87%
DM3	de Leon and Martínez (2013)	KNN	62.4%
DM4	de Leon and Martínez (2013)	KNN	64.38%
JJ1	Wu (2013)	OTHER	76.22%
JJ2	Wu (2013)	OTHER	76.05%

Table A.5: *MIREX 2013 submissions*

MIREX ID	Paper	Algorithm	Avg. Accuracy
BK1	Back and Kim (2014)	SVM	74.77%
BK2	Back and Kim (2014)	SVM	70.2%
BK3	Back and Kim (2014)	SVM	63.72%
BK4	Back and Kim (2014)	NN	66,27%
QK2	Kong and Feng (2014b)	NN	67.34%
SS6	Kong and Feng (2014a)	SVM	75.51%
SSKS1	Seyerlehner and Schedl (2014)	SVM	75.55%
WJ2	Wu and Jang (2014)	SVM	83.56%
ZL2	Zou et al. (2014)	SVM	50.4%
ZL3	Zou et al. (2014)	SVM	49.09%
ZL4	Zou et al. (2014)	SVM	67.34%

Table A.6: *MIREX 2014 submissions*

MIREX ID	Paper	Algorithm	Avg. Accuracy
CYC1	Cai et al. (2015)	SVM	71.47%
CYC2	Cai et al. (2015)	SVM	53.52%
CYC3	Cai et al. (2015)	SVM	69.18%
TL2	Lidy (2015)	NN	64.41%
WFJ1	Wu et al. (2015)	SVM	76.27%

Table A.7: *MIREX 2015 submissions*

MIREX ID	Paper	Algorithm	Avg. Accuracy
FFF1	Foleis and Tabares (2016)	SVM	70.44%
LS	Lidy and Schindler (2016)	CNN	73.14%

Table A.8: *MIREX 2016 submissions*

MIREX ID	Paper	Algorithm	Avg. Accuracy
LNPKK1	Lee et al. (2017)	NN	76.84%
LNPKK1	Lee et al. (2017)	SVM	71.13%
PLNPH1	Park et al. (2017)	SVM	74.27%
XLJ1	Xu et al. (2017)	NN	67.9%

Table A.9: *MIREX 2017 submissions*

BIBLIOGRAPHY

Back, S.-R. and Kim, M. Y. (2014). Music genre/mood/composer classification: Mirex 2014 submissions. https://www.music-ir.org/nema_out/mirex2014/results/act/mixed_report/.

Bagci, U. and Erzin, E. (2007). Automatic classification of musical gneres using inter-genre similarity. *IEEE Signal Proc. Letters*, 14(8):521 – 524.

Bergstra, J. a. (2006). Aggregate features and adaboost for music classification. *Machine Learning*, 65(2 - 3):473 – 484.

Burred, J. J. and Peeters, G. (2009). An adaptative system for music classification and tagging (mirex 2009 submissions). https://www.music-ir.org/mirex/wiki/2009:Audio_Genre_Classification_(Mixed_Set)_Results.

Byum, K., Back, S.-R., Lee, J.-S., Jang, S.-J., and Kim, M.-Y. (2013a). Music genre/mood/composer classification: Mirex 2013 submissions. https://www.music-ir.org/nema_out/mirex2013/results/act/mixed_report/.

Byum, K., Back, S.-R., Lee, J.-S., Jang, S.-J., and Kim, M.-Y. (2013b). Music genre/mood/composer classification: Mirex 2013 submissions. https://www.music-ir.org/nema_out/mirex2013/results/act/mixed_report/.

Cai, K., Yang, D., and Chen, X. (2015). Audio classifiction tasks: Mirex 2015 submissions. https://www.music-ir.org/nema_out/mirex2015/results/act/mixed_report/.

Cao, C. and Li, M. (2008). Thinkit audio genre classification system. https://www.music-ir.org/mirex/wiki/2008:MIREX2008_Results.

Cao, C. and Li, M. (2009). Thinkit submissions for mirex 2009 audio music classification and similarity tasks. https://www.music-ir.org/mirex/wiki/2009:Audio_Genre_Classification_(Mixed_Set)_Results.

Chang, K., Jang, J.-S. R., and Iliopoulos, C. S. (2010). Music genre classification via compressive sampling. *ISMIR*, pages 387 – 392.

cheon Lin, S., Byum, K., Lee, J.-S., Jan, S.-J., and Young-Kim, M. (2012). Music genre/mood classification: Mirex 2012. https://www.music-ir.org/nema_out/mirex2012/results/act/mixed_report/.

Cookiecutter (2018). Cookiecutter data science. http://drivendata.github.io/cookiecutter-data-science/.

Costa, C., Valle, J. D., and Koerich, A. L. (2004). Automatic classification of audio data. *IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583), The Hague*, 1:562 – 567.

Costa, Y., Oliveira, L., and Silla, C. (2017). An evaluation of convolutional neural networks for music classification using spectrograms. *Applied soft computing*, 52.

Costa, Y. M. G., Oliveira, L. S., Koerich, A. L., and Gouyon, F. (2011). Music genre recognition using spectrograms. *18th International Conference on Systems, Signals and Image Processing*, pages 1 – 4.

de Leon, F. and Martínez, K. (2013). Using timbre models for audio classification. https://www.music-ir.org/nema_out/mirex2013/results/act/mixed_report/.

de León, F. and Martínez, K. (2012a). An efficient approach for genre classification. https://www.music-ir.org/nema_out/mirex2012/results/act/mixed_report/.

de León, F. and Martínez, K. (2012b). Using rhythym timbre and tempo models for music genre classification. https://www.music-ir.org/nema_out/mirex2012/results/act/mixed_report/.

Diederik, K. and Ba, J. (2014). Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 3(11).

Epidemic (2009). Epidemic sound. https://www.epidemicsound.com/.

Faul, F., Erdfelder, E., Lang, A.-G., and Buchner, A. (2007). G*power 3: a flexible statistical power analysis program for the social, behaioural and biomedical sciences. *Behaviour research Methods*, 39(2):175 – 191.

Foleis, J. E. and Tabares, T. F. (2016). A spectral bandwise feature-based system for the mirex 2016 train/test task. https://www.music-ir.org/nema_out/mirex2016/results/act/mixed_report/.

Glass., G. V., Peckman, P. D., and Sanders, J. R. (1972). Consequences of failure to meet assumptions underlying fixed effects analysis of variance and covariance. *Review of Educational Research*, 42:237 – 288.

Granma, L. and Rusu, C. (2017). Choosing an accurate number of mel frequency cepstral coefficients for audio classification purpose. *10th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 225 – 230.

Grecu, A., Lidy, T., and Rauber, A. (2009). Mirex 2009 enhancing audio classification with template features and postprocessing existing audio descriptors. https://www.music-ir.org/mirex/wiki/2009:Audio_Genre_Classification_(Mixed_Set)_Results.

Grosche, P., Müller, M., and Kurth, F. (2010). Cyclic tempogram - a mid-level tempo representation for music signals. *ICASSP*, pages 5522 – 5525.

Guaus, E. and Herrera, P. (2007). A basic system for music genre classification. https://www.music-ir.org/mirex/wiki/2007:Audio_Genre_Classification_Results.

Gwardys, G. and Grzywczak, D. M. (2014). Deep image features in music information retrieval. *International journal of electronics and communications*, 60(4):59 – 66.

Hamel, P. (2012). Multi-timescale pmscs for music audio classification. https://www.music-ir.org/nema_out/mirex2012/results/act/mixed_report/.

Hamel, P. and Douglas, E. (2010). Learning features from music audio with deep belief networks. *Proceedings of the 11th International Society for Music Information Retrieval conference*, pages 657 – 662.

Harwell, M. R., Rubinstein, E. N., Hayes, W. S., and Olds, C. C. (1992). Summarizing montecarlo results in methodological research: the one and two-factor fixed effects anova cases. *Journal of Educational Statistics*, 17:315 – 339.

Hasegawa, T., Nishimoto, T., Ono, N., and Shagayama, S. (2009). Music classification using features based on musical knowledge. https://www.music-ir.org/mirex/wiki/2009:Audio_Genre_Classification_(Mixed_Set)_Results.

Homsi, A., Capobianco, R., and Dias, C. (2012). Exploring different approaches for music genre classification. *Egyptian Informatics Journal*, 2(13):59–63.

ISMIR (2000). International society for music information retrieval. http://www.ismir.net/.

Karada, O. and Aktag, S. (2012). Optimal sample size for the anova designs. *International Journal of Applied Mathematics and Statistics*, 25(1).

Kong, Q. and Feng, X. (2014a). Mirex 2014: Optimizing the fluctuation pattern extraction process. https://www.music-ir.org/nema_out/mirex2014/results/act/mixed_report/.

Kong, Q. and Feng, X. (2014b). Music genre/mood/composer classification: Mirex 2014 submissions. https://www.music-ir.org/nema_out/mirex2014/results/act/mixed_report/.

Lee, J., Park, J., Nam, J., Kim, C., Kim, A., Park, J., and Ha, J.-W. (2017). Cross-cultural transfer learning using sample level deep convolutional networks. https://www.music-ir.org/nema_out/mirex2017/results/act/mixed_report/.

Lidy, T. (2015). Spectral convolutional neural networks for music classification. https://www.music-ir.org/nema_out/mirex2015/results/act/mixed_report/.

Lidy, T., Grecu, A., Rauber, A., Pertusa, A., de León, P. J. P., and Iniesta, J. (2009). A multifeature set multi-classifier ensemble approach for au-

dio classfication. https://www.music-ir.org/mirex/wiki/2009:Audio_Genre_Classification_(Mixed_Set)_Results.

Lidy, T., Rauber, A., Pertusa, A., de León, P. P., and Iniesta, J. (2008). Audio music classification using a combination of spectral timbral rhythmic, temporal and symbolic features. https://www.music-ir.org/mirex/wiki/2008:MIREX2008_Results.

Lidy, T., Rauber, A., Petrusa, A., and Iniesta, J. M. (2007). Combining audio and symbolic descriptors for music classification from audio. https://www.music-ir.org/mirex/wiki/2007:Audio_Genre_Classification_Results.

Lidy, T. and Schindler, A. (2016). Parallel convolutional networks for music genre and mood classification. https://www.music-ir.org/nema_out/mirex2016/results/act/mixed_report/.

Liu, Y., Zheng, T., and Gao, Y. (2009). Mirex audio genre classification. https://www.music-ir.org/mirex/wiki/2009:Audio_Genre_Classification_(Mixed_Set)_Results.

LSVRC (2012). Imagenet. large scale visual recognition challenge. http://image-net.org/challenges/LSVRC/2012/.

Mallat, S. (2012). Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331 – 1398.

Mandel, M. I. and Ellis, D. P. W. (2007). Labrosa's audio music similarity and classification submissions. https://www.music-ir.org/mirex/wiki/2007:Audio_Genre_Classification_Results.

Mandel, M. I. and Ellis, D. P. W. (2008). Labrosa's audio classification submissions. https://www.music-ir.org/mirex/wiki/2008:MIREX2008_Results.

Mark, F. and Handy, R. (2007). Spectograms: Turning signals into pictures. *Journal of Engineering Technology*, 1(24):32 – 35.

Marques, C., Guiherme, I. R., Nakamura, R. Y. M., and Papa, J. P. (2011). New trends in musical genre classification using optimum-path forest. ISMIR, 2011.

McFee, B., Raffel, C., Liang, D., Ellis, D. P. W., McVicar, M., Battenberg, E., and Nieto, O. (2015). librosa : Audio and music signal analysis in python.

McKinney, F. and Breebaart, J. (2003). Features for audio music classification. *ISMIR*.

MIREX (2005). International society for information retrieval. `http://www.music-ir.org/mirex/wiki/MIREX_HOME`.

Müller, M. and Ewert, S. (2011). Chroma toolbox: Matlab implementations for extracting variants of chroma-based audio features. *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*.

Panagakis, Y., Kotropoulos, C., and Arce, G. R. (2009). Music genre classification via sparse representations of auditory temporal modulations. *17th European Signal Processing Conference*, pages 1 – 5.

Panda, R. and Pavía, R. P. (2012). Mirex 2012: mood classification tasks submission. `https://www.music-ir.org/nema_out/mirex2012/results/act/mixed_report/`.

Parascandolo, G., Huttunen, H., and Virtannen, T. (2016). Recurrent neural networks for polyphonic sound event detection in real life recordings. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6440–6444.

Park, J., Lee, J., Park, J., and Ha, J.-W. (2017). Representation learning using artist labels for audio classification tasks. `https://www.music-ir.org/nema_out/mirex2017/results/act/mixed_report/`.

Peeters, G. (2008). A generic training and classification system for mirex 2008 classification tasks: audio music, audio mood, audio genre and audio tag. `https://www.music-ir.org/mirex/wiki/2008:MIREX2008_Results`.

Rajanna, A., Kamelya, A., Shoukoufandeh, A., and Ptucha, R. (2015). Deep neural networks: A case study for music genre classification. *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 582 – 587.

Rao, P. and Kini, S. (2009). Musical genre classification. https://www.music-ir.org/mirex/wiki/2009:Audio_Genre_Classification_(Mixed_Set)_Results.

Ren, J.-M., Chen, Z.-S., and Jang, J.-S. R. (2009). Using sequential patterns as short-term and long-term features for adio genre classification. https://www.music-ir.org/mirex/wiki/2009:Audio_Genre_Classification_(Mixed_Set)_Results.

Ren, J.-M. and Jang, J.-S. R. (2012). Marsyas submission for mirex 2012. https://www.music-ir.org/nema_out/mirex2012/results/act/mixed_report/.

Schörkhuber, C. (2010). Constant-q transform toolbox for music processing.

Seryerlehner, K., Schedl, M., Knees, P., and Sonnleitner, R. (2012). Draft: A refined block-level feature set for classification, similarity and tag prediction. https://www.music-ir.org/nema_out/mirex2012/results/act/mixed_report/.

Seyerlehner, K. and Schedl, M. (2014). Mirex 2014: Optimizing the fluctuation pattern extraction process. https://www.music-ir.org/nema_out/mirex2014/results/act/mixed_report/.

Shazam (2002). Shazam entertainment ltd. https://www.shazam.com/.

Shim-Cheol, L., Jong-Seol, L., Sei-Jin, J., Soek-Pil, L., and Kim, M. Y. (2012). Music-genre classification system based on spectro-temporal features and feature selection. *IEEE Transactions on Consumer Electronics*, 58(4):1262 – 1268.

Silla, C. N. (2018). The latin music database. https://sites.google.com/site/carlossillajr/.

Stephens, D. J. (2007). The music information retrieval exchange: a window into music information retrieval exchange. *Journal of Acoustical Science and Technology*, 29(4):247 – 255.

Sturm, B. L. (2012a). An analysis of the gtzan music genre dataset. In *Proceedings of the Second International ACM Workshop on Music Information Retrieval with User-centered and Multimodal Strategies*, MIRUM '12, pages 7–12, New York, NY, USA. ACM.

Sturm, B. L. (2012b). Two systems for automatic music genre recognition: What are they really recognizing? In *Proceedings of the Second International ACM Workshop on Music Information Retrieval with User-centered and Multimodal Strategies*, MIRUM '12, pages 69–74. ACM.

Sturm, B. L. (2013). Classification accuracy is not enough. *Journal of Intelligent Information Systems*, 41:371 – 406.

Tsunoo, E., Tzanetakis, G., Ono, N., and Sagayama, S. (2009). Audio genre classification using rhythym and bass-line pattern information. `https://www.music-ir.org/mirex/wiki/2009:Audio_Genre_Classification_(Mixed_Set)_Results`.

Tufte, E. (1997). *Visual explanations: Images and Quantities, Evidence and Narrative*. Graphic Press, Cheshire, Connecticut.

Tzanetakis, G. (2007). Marsyas submissions to mirex 2007. `https://www.music-ir.org/mirex/wiki/2007:Audio_Genre_Classification_Results`.

Tzanetakis, G. (2008). Marsyas submissions to mirex 2008. `https://www.music-ir.org/mirex/wiki/2008:MIREX2008_Results`.

Tzanetakis, G. (2018). Marsyas. http://marsyas.info/index.html.

Tzanetakis, G. and Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 2(10):283–302.

Valero, X. and Alias, F. (2012). Gammatone cepstral coefficients: Biologically inspired features for nonspeech audio classification. *IEEE Traansactions on Multimedia*, (14):1684 – 1689.

Venrooija, A. and Schmutz, V. (2018). Categorical ambiguity in cultural fields: The effects of genre fuzziness in popular music. *Poetics*, 66.

Wack, N., Guaus, E., Laurier, C., Meyers, O., Marxer, R., Bogdanov, D., Serrá, J., and Herrera, P. (2009). Music type groupers (mtg) generic music classification algorithms. https://www.music-ir.org/mirex/wiki/2009:Audio_Genre_Classification_(Mixed_Set)_Results.

Wang, H. (2009). Mirex audio genre classification. https://www.music-ir.org/mirex/wiki/2009:Audio_Genre_Classification_(Mixed_Set)_Results.

Wu, M.-J. (2013). Mirex 2013 submissions for train/test tasks. https://www.music-ir.org/nema_out/mirex2013/results/act/mixed_report/.

Wu, M.-J., Fan, Z.-C., and Jang, J.-S. G. (2015). Mirex 2015 submission for audio classification tasks. https://www.music-ir.org/nema_out/mirex2015/results/act/mixed_report/.

Wu, M.-J. and Jang, J.-S. R. (2012). Mirex submissions - combining acoustic and multi-level visual features for music genre classification. https://www.music-ir.org/nema_out/mirex2012/results/act/mixed_report/.

Wu, M.-J. and Jang, J.-S. R. (2014). Confidence-based late fusion for music genre classification. https://www.music-ir.org/nema_out/mirex2014/results/act/mixed_report/.

Xu, H., Lv, J., and Ja, B. (2017). Convolutional neural networks system for the mirex 2017 audio classification (train/test) tasks. https://www.music-ir.org/nema_out/mirex2017/results/act/mixed_report/.

Xu, J., Liu, Y., Zeng, T., and Zeng, C. (2009a). Audio music classification using support vector machine. https://www.music-ir.org/mirex/wiki/2009:Audio_Genre_Classification_(Mixed_Set)_Results.

Xu, J., Liu, Y., Zeng, T., Zeng, C., and Gao, Y. (2009b). Music mood classification based on voting. https://www.music-ir.org/mirex/wiki/2009:Audio_Genre_Classification_(Mixed_Set)_Results.

Yandra, C., Oliveira, L., Koerich, A., and Gouyon, F. (2013). Music genre recognition based on visual features with dynamic ensemble of classifiers selection. *2013 20th International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 55 – 58.

Yang, D., Chen, X., and Peng, L. (2012). Submission to mirex 2012 audio music mood classification and audio pop music genre classification. https://www.music-ir.org/nema_out/mirex2012/results/act/mixed_report/.

Zhouyu, F., L., G., Ting, K. M., and Zhang, D. (2010). A survey of audio-based music classification and annotation. *IEEE Transactions on Multimedia*, 2(13).

Zou, X., Li, G., and Su, S. (2014). Combining spectral and rhythym features for music genre classification. https://www.music-ir.org/nema_out/mirex2014/results/act/mixed_report/.