Automatic Music Genre Classification : Comparative Study and Experimentation

Machine Learning final project report

Anthony Pansard
Ecole CentraleSupélec
Gif-Sur-Yvette
anthony.pansard@student.ecp.fr

Marius Hullin
Ecole CentraleSupélec
Gif-Sur-Yvette
marius.hullin@student.ecp.fr

Erwan De Lépinau
Ecole CentraleSupélec
Gif-Sur-Yvette
erwan.de-lepinau@student.ecp.fr

Luca Serra
Ecole CentraleSupélec
Gif-Sur-Yvette
luca.serra@student.ecp.fr

ABSTRACT

With the advent of several web-based music sharing platforms, the volume of data exchanged and uploaded on a daily basis exploded. This explosion, in the beginning of the 20th century, led to a humongous number of uploaded tracks, that exceeds the amount of data a human could possibly process. Hence, a real need for automated ways to process data was expressed in the early 2000's, as the processing was entirely done by hand at this time [1] [2]. Among the processing required was the need for automatic genre recognition. Music genre is a human construction used to classify music into different classes. Each class has its own set of rules and conventions.

The demand attracted a lot of researchers from the field of Machine Learning, which led to many techniques for Automatic Music Genre Classification (abr. AMGC).

This paperwork will summarize our work of studying and comparing some of the techniques developed to carry out AMGC using machine learning. It will also detail our experimentation of reproducing AMGC, using as best as we could the lessons that we learned from the other techniques.

KEYWORDS

Music Genre Classification, Music Information Retrieval, Feature Selection, Machine Learning, Audio Signal Processing

1 INTRODUCTION

1.1 Subject definition and deviation from the project proposal

Originally, our work was also focusing on music, but we were to work not on genre recognition, but on the extraction of another feature : the tempo of a music track, also called the number of Beats per Minute (BPM).

However, when we began to work on the subject, we soon realized that it was hardly a difficult or complicated task. We noticed that it could be achieved without the help of Machine Learning, with a simple "naive" algorithm.

In fact, the BPM was often used as a simple feature that was extracted from music to later perform classification on it [1][2][3].

Final Report, January 2019, Machine Learning, MA2823, 2019.

Thus, after we searched for another subject related to the first one, we came up with the concern of automatic genre retrieval. This subject belongs to the more vast subject of Music Information retrieval (MIR), that focuses on the extraction of information out out a music track. Of course, music genre classification began to be investigated a relatively long time ago, at the beginning of the 2000's. There hence exists lots of research works on this subject, with one of the first serious papers on the topic published in 2002, by Tzanetakis and Cook. [1].

Given the problem's size and the research work done in this field, the current techniques are numerous, and their results aren't always satisfying. Hence, in this project we focused on the **comparison of the different methods** and on the definition of an **optimal methodology to achieve AMGC** according to the current State of the Art.

In order to do so, we will focus on two points :

— The feature selection, that is often the most important factor [2] in AMGC : we will try to present the different features that exist, how they are extracted and their impact on the final accuracy (or other metrics) of the different techniques.
— The classifiers used to give the best results.

But let's formalize the problem :

The point of AMGC is to obtain the best likelihood for a song to belong to a specific genre. This can be expressed as *Find $\tilde{g}$ such as :*

$$\tilde{g} = \underset{g \in G}{\arg\max}\, P(g \mid X)$$

Where $\tilde{g}$ is the predicted genre of the song, G the set of genre that are examined (usually about 5 to 10 genres simultaneously), and $X = (x_1, x_2, ..., x_D), D \in \mathbb{N}$ the feature vector, with x_i the features.

The goal is the estimation of the probability, with a Bayesian formula : $P(g \mid X) = \frac{P(X|g)P(g)}{P(X)}$.

To do so, it is important to try and select the **most relevant features and classifiers.**

1.2 Subject relevance and concrete applications

Tagging and labeling of music content has been attracting interest since the beginning of the so-called "Web 2.0 era", characterized by the emergence of social and interactive platforms, including in the music domain (online radios, online streaming services) [4].

Music genre is not the only kind of musical tag that exists : other relevant tags can concern the mood, style [5] of a song, or even express personal opinions. However, the majority of tags applied on audio files on community platforms are relative to the genre (68% of tags on Last.fm [4]), which shows how much attention people give to this specific feature.

With the exponential growth of online content (including music) that the internet has known in the past decades, the task of labeling music by hand (as it has been done for a long time and still is) [1] [2] has become time- and cost-expensive (since it is usually done by music experts), although it is of primary importance for companies such as online streaming platforms (Spotify, iTunes, Tidal...) [6] who need this information to enhance features such as customized user recommendations and automatic playlist generation, among many others. Moreover, it is important to note that even among human annotators, there are usually different interpretations of music genre, resulting in inconsistent labeling of audio files [7] : automatically performing the task of AMGC could allow for a standardization of the process.

1.3 Plan of the work

This paper is divided in two parts :

— The first one is a review of a lot of the techniques that currently exist to perform AMGC.
— In the second one, we will present our own experimentation on AMGC, based on the previously studied related work.

In the first part, we will start by presenting some of the research fields related to AMGC, most of which had a very strong influence on the development of the techniques that we will present in the following part.

We will then summarize and critically compare what we learned from the other works of research on the subject. The goal is to define theoretically an optimal way to perform music genre classification and the Machine Learning pipeline of such an algorithm. We decomposed the task in several parts :

(1) Choice of the data set

(2) Feature extraction

(3) Choice of the classifiers

(4) Evaluation process

(5) Expected results

Once the ideal methodology is been found, we will focus on how we developed our own algorithm, and how we implemented each of the steps.

The last step will be to compare the results that we obtained and draw conclusions from there.

Finally, we will conclude this paper by summarizing our work and give insights on what remains to be done.

2 RELATED RESEARCH FIELDS

Work on audio signal processing for classification tasks is anterior to the application to AMGC. A great amount of work had already been reported in the early 2000s [2] in the domains of speech recognition, or music from speech discrimination. Although these works are not directly linked to our subject of research, they allowed for significant improvements in audio signal processing, especially when it comes to finding non-trivial features (such as Mel-Frequency Cepstral Coefficients, or MFCCs [8]) to extract from the audio files.

Insights can also be taken from research works in musicology and psychology as described in [5], considering that a better understanding of the – more complex than what meets the eye – human way of classifying music could help algorithms obtain results similar to what is achieved by actual people.

There are a few other very closely related research topics in the machine learning community, some opting for a broader approach such as Auto-Tagging of music (tags can be genres, but also much more) [4], while others are investigating into a narrower aspect of the topic, such as classifying music between *vocal* and *non-vocal* tracks [9]. Nonetheless, researchers from all of these fields have a lot to gain by keeping an eye on the work of others, and communicating on the lessons learned through each new study, since most of the techniques (especially in signal processing and feature extraction) that are successful in one paper could be applied to related works, even if the objective is slightly different.

It is safe to say that the objectives of most of these research fields are convergent. Indeed, 15 years from now, we will not only expect from an artificial intelligence that it classifies an audio extract between 5 or 6 caricatural music genres, but rather that it discriminates it precisely in a subgenre with the granularity [5] required by the user (an electronic music expert will not be satisfied with a track being labeled as "electro", he will want to know if it is rather a "future house" track or even a "glitch hop" one) ; the AI should also be able to identify and write the lyrics to a certain song ; it should understand if a certain speech extract is taken from a sports commentary podcast, or from a stand-up comedy show.

3 CRITICAL COMPARISON OF EXISTING METHODOLOGY

3.1 Data set selection

The first part of the work is to find the adapted data set. It must be big enough to avoid risks of under-fitting, but small enough to avoid a computation time too large. Several parameters will be important in the choice (or constitution) of our work data base. Among them, we can cite :

— The format of the audio files (.mp3, .au, .wav, ...)
— The number of songs and of genres represented
— The sampling of different songs : if they are in the database as full-length tracks, or just a part. Researchers often conduct their tests on extracts from the beginning, middle and/or end of the music tracks.

Data set size : The problem of the data set size is important in terms both of length of the extracts and of number of files. Generally, the number of files is around several hundreds to a

few thousand, here is the graph of the data base sizes used in the research works we studied :

Number of Tracks	Number of genres
60	4
100	4
300	6
756	5
1000	10
1000	10
1515	10
1614	6
2000	4
3160	10

Table 1: Distribution of the sizes of the 10 data sets used in the studied work.

The differences in size of sets used are striking, with some being ridiculously small [10], and others being of substantial size, as in [11] or in [3].

This concern is studied in a comparative work on AMGC [4]. The problem is that the constitution of the data sets is usually done by hand. This is because music genre is a human abstraction and hence is not a ground truth : it can sometimes change, depending on the person, time or condition of listening. Thus, we are facing either small and well-constructed and labelled data sets, or bigger ones that are less precisely populated. Small ones are generally done by hand by a few people (usually music experts), as opposed to bigger ones, often based on collaborative tools to label the sets (e.g. *Google's Audio Set*, *Last.FM*, ...).

Data format : Then comes the concern of the format of sampling. The most used types among the databases are .mp3, .au and .wav. All three are formats that contain directly the graph of the wave (after an analogical to numerical sampling). There is no problem for the feature selection, since all the features extracted can be computed from a discrete signal.

The only exception to this was one paper ([7]) where **MIDI files** were used. Of course, the results of the algorithm were stunning, since the trick is that MIDI files contain **only** the notes that have to be played, when they have to be played, and by what instrument. Such information is obviously very useful to determine the genre of an audio file, but can not be found in widely used formats such as the ones mentioned above. Such a way of proceeding is pretty unrealistic, as no data base of online platform contains music under MIDI format.

3.2 Feature extraction

The two following sections cover what is probably the most important part of AMGC, namely Feature Extraction and Feature Selection. This topic is where most of the major leaps in the domain of AMGC have been done, and it probably still is the step where the most room for improvement lies [2].

An audio signal is represented (in digital formats) as a sequence of points, each one assigned with a value coded over a certain number of bits (usually 16), and encoded at a certain sample rate (usually around 45 kHz). From this simple description, we can understand that a tremendous amount of work has to be carried out in order to extract higher-level features (closer to human abstraction and concepts for describing music) from this extremely low-level encoding.

This feature extraction process aims at describing in a numerical manner certain characteristics of music that will be used by the machine learning algorithms to predict the genre of an audio sample, in a similar fashion as what humans naturally do, usually without even noticing it. However, as mentioned in [5], humans do not simply use "necessary and sufficient conditions" on basic features of songs, such as rhythm or pitch, which explains why researchers tend to fall back on advanced mathematical processes and extremely abstract features.

A first step in feature extraction is to determine the length and the position in the audio file of the sample that we are going to use to extract features : using full 3- or 4-minute long songs would be computationally expensive (the feature extraction part would take too long, and the data obtained would be too large), and [3] reminds us that studies have been conducted showing that humans could identify correctly the genre of a music 70% percent of the time by only listening to it for 3 seconds. From what we have reviewed in the literature, researchers rarely use samples exceeding 30 seconds. Some would rather use shorter samples of about 3s [12], even going down to 1s [10], while others recommend to extract samples at different moments of a song (for instance, 30 seconds at the very beginning, 30 seconds in the middle, 30 seconds at the end) [11] because they allow for a more comprehensive depiction of the audio file.

What seems to come out of these studies is that longer samples are not necessarily associated with dramatic increases in performance, and the most representative samples of a music are usually the ones extracted from the middle part of the extract [11]. Choosing a 3-second long sample from the chorus of a music piece seems to be a good middle ground.

As far as signal pre-processing is concerned, it has been suggested [6] that audio signals should be filtered in order to improve the Signal to Noise Ratio (SNR), however this step may be very situational and dependant on the dataset used, since AMGC is mainly to be used on commercial platforms (Spotify, iTunes) where the audio quality is in general excellent, even for older albums/songs that are usually remixed/remastered before upload on the platforms.

We will now present and describe some of the most commonly used and effective features for AMGC. The first group of features, sometimes referred to as "timbral features", are based on the Fourier Transform, which allows for representation of the signal in the frequency domain. They have been first applied to AMGC in 2002 by Tzanetakis et al [1], and are used in almost every paper dealing with AMGC since then. These features are usually computed over short overlapping frames (20ms, usually obtained through Hann or Hamming windowing, see Figure 1), and are then generalized to the whole several-seconds long sample, usually by computing the mean and variance of the individual values for each short frame [2].

The most commonly used timbral features are the following (in the formulas, $M[f_i]$ is the magnitude of the FFT at

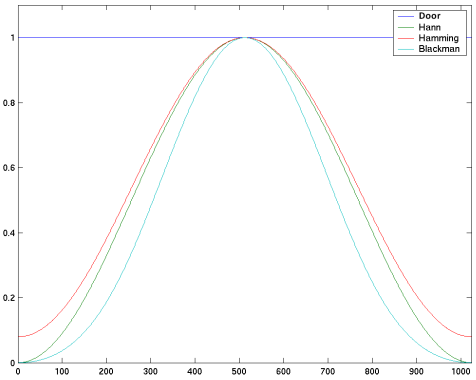

Figure 1: Representation of the most common window functions

frequency bin f_i and N the number of frequency bins) :

— **Spectral Centroid :**

$$C = \frac{\sum_{i=1}^{N} f_i M[f_i]}{\sum_{i=1}^{N} M[f_i]}$$

The Centroid indicates where the "center of mass" of the spectrum is located. It is linked with the perceived "brightness" of the music.

— **Spectral Bandwidth :**

$$[\sum_{i=1}^{N} (f_i M[f_i] - C)^p]^{1/p}$$

It is the weighted $p-th$ order moment around the Spectral Centroid. For example, the $2nd$ order moment is called **Spectral Spread** and is commonly used [12].

— **Spectral Roll-off :** It is the value R such that

$$\sum_{i=1}^{R} M[f_i] = 0.85 * \sum_{i=1}^{N} M[f_i]$$

It indicates the point under which 85% of the total energy of the spectrum lies. It is linked with the "shape" of the spectrum.

— **Spectral Flux :**

$$F = ||\vec{M}[f] - \vec{M}_p[f]||$$

where $\vec{M}[f]$ and $\vec{M}_p[f]$ are respectively the normalized magnitude vectors of the spectrum at the current and previous frame in time. This measures the amount of change in the spectrum from one frame to the other.

Other very commonly used features in time domain are :

— **Zero Crossing Rate :** It is the rate at which the time-domain signal crosses the zero amplitude line. It helps measure the noisiness but also the percussive nature of an audio signal.

— **Spectral Flatness :** The ratio between the geometric mean and the arithmetic mean of the amplitudes in the frequency spectrum :

$$F = \frac{\sqrt[N]{\sum_{n=1}^{N} M[f_i]}}{\frac{\sum_{n=1}^{N} M[f_i]}{N}}$$

Flatness indicates if a signal is rather *noise-like* ($F \lessapprox 1$) which means that there is a similar amount of power in all spectral bands (similar to white noise), or if it is rather *tone-like* which is characteristic of spectrums with peaks and resonant bands.

— **Short-Time Energy :**

$$E_m = \sum_{n=1}^{N} [x(n)w(n-m)]^2$$

where m is the central point of the considered frame, N the total number of points in the frame, $x(n)$ the value of the signal at point n and w is the chosen window function. This measures the energy of the signal in the considered frame : it is sometimes used as part of the pre-processing [9] in order to remove silences (e.g. frames whose energy is below a certain threshold).

— **Low Energy Rate :** The percentage of frames whose energy is lower than the average energy of the sample.

Another very commonly used family of features is the **Mel-frequency cepstral coefficients**, or **MFCCs**. Widely used and very successful in the field of speech recognition, these features have also shown effectiveness in the AMGC domain (see [1] [2] [6] [9] [11] [12] [10]).

The **Mel-frequency cepstrum** of a frame is obtained by :

(1) taking the Fourier Transform of the signal on the frame
(2) mapping the obtained spectrum from the Hertz scale to the Mel scale (which is a frequency scale that is more coherent with the human perception of pitch intervals)
(3) taking the *logs* of the amplitudes of the mel-frequency bins
(4) applying a *discrete cosine transform* to the list of *log*-mel-amplitudes
(5) the MFCCs are the amplitudes of the peaks of the obtained spectrum (called a *cepstrum*, see Figure 2)

Autoregression coefficients or **Linear Predictive Coefficients (LPC)** are also frequently used [12] [10]. They are obtained by autocorrelation of the signal, and allow to express the time-domain signal at a given moment as a linear combination of the past values of the signal. They are defined as the coefficients a_i such that

$$x(n) = \sum_{i=1}^{N} a_i x(n-i)$$

where N is the model order and $x(n)$ is the signal at point n in

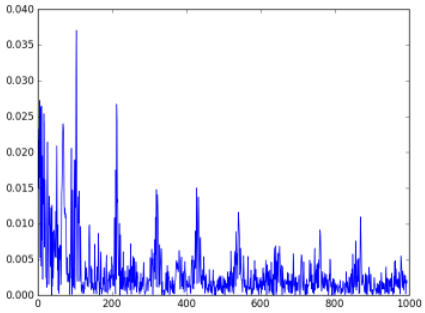

Figure 2: Cepstrum of an electric guitar sample

time-domain.

Wavelet Transform has also been documented in AMGC [2] with very good performance, and is also widely used in image processing and image classification. Wavelet transform is similar to Fourier transform, but essentially differs in that Fourier Transform only retains frequency information (with perfect resolution) while losing all time-domain information, whereas Wavelet Transform retains both frequency and time-domain information, in return for a loss in the resolution in both domains. This process is interesting for AMGC because although most of the relevant information in music is present in the frequency domain, time-based features can also prove useful [5].

The Wavelet Transform used in [2] is a *Daubechies 8 Wavelet filter* or *Db8*, which is then used to compute **Daubechies Wavelet Coefficient Histograms (DWCHs)** over 3-seconds long windows (see [2] for more detail on the technical implementation), resembling Figure 3.

Although feature extraction by Wavelet Transform of the signal seems to be a good path to follow, the vast majority of research works after Li et al. (2003) [2] did not use this feature as part of their dataset. Considering the considerable amount of work that has been done since then using and perfecting Wavelet Transform, especially in the automatic image classification domain, giving this family of features a second chance could prove useful in future work.

Finally, we deem important to be careful with the usage of "high-level features". Such features are meant to be closer to human abstraction and concepts, and are promoted by McKay and Fujinaga (2006) [5] because they have supposedly shown promising results. However, the issue with such features is that they are not always well-defined from a technical standpoint : a good example is the "danceability" feature that was used in Clark et al. (2012) [3]. This feature was generated by the platform used by these researchers to retrieve features from their dataset (Echonest), but no clear definition of it is given, besides describing it as a "synthesis of beat-strength, tempo stability, and overall temp". The researchers themselves agreed on the fact that their inclusion in their feature set was controversial, because such features can not easily be recreated by other teams in further works. High-level features should not necessarily be avoided at all cost, but it is necessary to give a precise definition of what they represent, and if possible a procedure to extract them from any dataset.

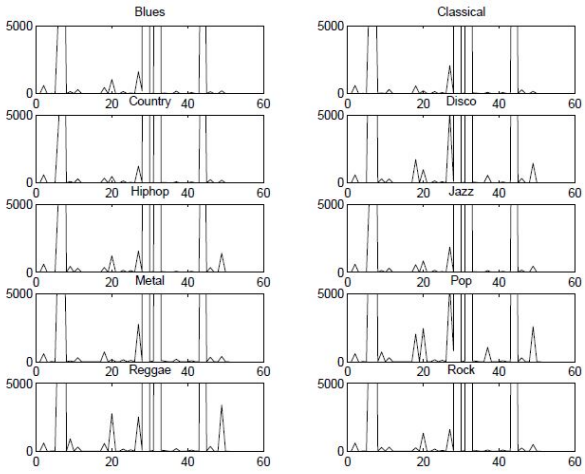

Figure 3: DWCHs of 10 music signals in 10 different genres, obtained by Li et al. in [2]

3.3 Feature Selection

After the feature extraction part, it is often needed to reduce the amount of features used to train the model, since at that point the feature vector frequently contains hundreds of values : training a model with so many features can be too computationally extensive, and it is important to note that adding features does not necessarily dramatically improve the performance of a model [2] [6], since different features sometimes contain redundant information.

There are two main approaches to select features [11] :

- **Wrapping :** The goal of wrapper methods is to find an optimal subset of features that maximizes the performance of the algorithm while reducing the dimensionality of the dataset. There are several ways to implement wrapping, such as **Forward Feature Selection (FFS)**, **Backward Feature Elimination (BFE)**, **Recursive Feature Elimination (RFE)** or **Generatic Algorithm (GA)** which is used in [11].
  The major drawback regarding wrapper methods is that they are usually very **computationally expensive**, since they repeat the training phase a great number of times until an optimal subset of features is found. This can prove problematic when dealing with large datasets or large feature vectors. Another issue is that such methods can lead to overfitting, because they attempt to chose the subset of features that is optimal **on the training dataset**.
- **Filtering :** This approach uses *Dimensionality Reduction* techniques, such as **Principal Component Analysis (PCA)** or *Singular Value Decomposition (SVD)*. Features are selected or discarded based on their statistical correlation with the outcome variable of the dataset (the class label). The biggest disadvantage of such methods is that they usually do not preserve the original identity of the features [5] (this is the case with PCA or SVD), which prevents the researchers from analyzing the role

that the different features play in the performance of the algorithm.

As we can see, both approaches have distinct advantages and drawbacks, and the choice of the feature selection method appears to depend a lot from the situation.

3.4 Choice of the classifiers

This part is, with feature extraction, the most important part of our work, as it is the stage that gives us the answers to the initial question. The main topic here is to discuss the different ways to perform the regression or the classification of the data gathered in the features extracted.

First lies the problem of the use of classification rather than regression. In fact, classification is almost always used in AMGC problems. The reason is that we seek the genre of a song, which is a human construction used to label music, meaning that we are looking for a single response. However, this can be questioned, as music genre classification can sometimes be very blurry, even for human beings.

The problem when we are using classification algorithms is that we use mostly binary classifiers, even though hardly any data base contains only two genres. Some algorithms transition very easily into mutliclass classification (such as LDA or k-Nearest-Neighbors), however others are inherently only capable of binary classification (SVMs are a good example of that) [2].
To solve this problem, several techniques are used, among them :

- The **One Against One approach** : A classifier is constructed for each pair of classes, and the result genre is voted according to the majority of 'victories' (c.f. fig. 4).
- The **One Against All approach** : A classifier is constructed for each class, and all the other classes are opposed to the class, which gives the probability of belonging to that class. The result genre is the one with the higher probability (c.f. fig. 5).
- The **Multi-layered approach** : A decision tree is made, with two choices determined by a binary classifier at each node (c.f. fig. 6).

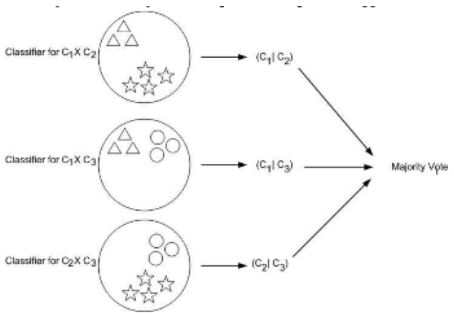

Figure 4: Principle of the One Against One approach [11]

Support Vector Machines : SVMs have proven to be very effective in a lot of classification problems in the machine learning field, and AMGC is no exception to this rule : SVMs

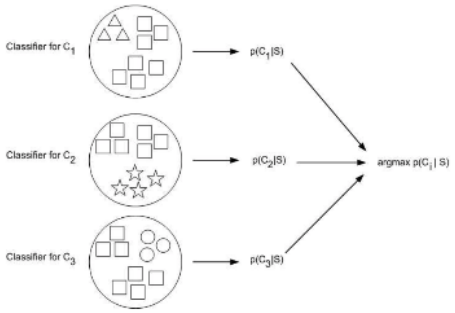

Figure 5: Principle of the One Against All approach [11]

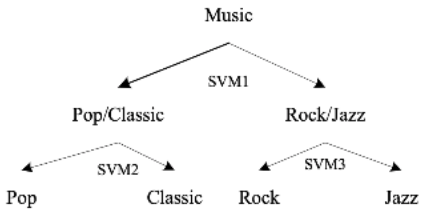

Figure 6: Principle of the Multi-layered Approach [10]

are indeed the most widely used machine learning algorithms used in AMGC. More generally speaking, their performance has been demonstrated in this field, as one of the classifier with the best results. However, the only concern with SVMs is the computation speed, of order n^2, where n is the number of data points that needs to be tested : SVMs scale poorly with the size of the dataset.

Basically, SVMs are binary classifiers, whose goal is to draw the best hyperplane in a feature space. The feature space is first populated with the feature vectors $V_i = (x_i, f_1, f_2, ..., f_d)$, where $i \in [\![1; n]\!]$ is the index of the vector, x_i refers to the data, and $f_1, f_2, ..., f_d$ are the features extracted from it in the previous stage of the algorithm (d the number of features). The fact that SVMs are binary classifiers means that the hyperplane drawn separates only to classes (which are often referred as -1/fail and $+1$/success).

Another problem is that SVMs only intend to draw a hyperplane (linear SVM), whereas the feature vectors are not always linearly separable. In these cases, we use a nonlinear transformation Φ, and then to perform a linear SVM on the result space. The most common transformation includes[9][10] :

- Polynomial kernel of degree k :
$$K(x, y) = (\langle x, y\rangle + 1)^d$$
- Radial basis function with Gaussian kernel
$$K(x, y) = \exp\left(\frac{-|x - y|^2}{c}\right)$$
Where $C > 0$ is the width of the Gaussian Kernel.
- Neural networks with $tanh$ activation function :
$$K(x, y) = \tanh\left(k\langle x, y\rangle + \mu\right)$$
Where k and μ are the gain and shift parameters of the neural network.

In the end, the decision rule of the SVM can be expressed as

$$\operatorname{sgn}\left(\sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b\right)$$

Where the α_i are the weights of each of the features, x the input (=feature) vector, and K the chosen kernel function.

The last task is to organize the SVMs into a multiclass classifier, which is done exactly as explained in the beginning of this part.

AdaBoost : AdaBoost has also proven to be a very good classifier [12] in terms of performance in AMGC. We will review here the AdaBoost method as it is used in Bergstra et al. (2006) [12], known as *AdaBoost.MH Schapire and Singer.*

Method :

It is an iterative classifier (which explains the Boost in AdaBoost), meaning that it incrementally corrects itself.

- In each iteration, ($t \in \mathbb{N}$), the algorithm calls a simple learning classifier (called the *weak learner*), that returns a classifier $f^{(t)}$, and computes an associated coefficient $\alpha^{(t)}$.
- The input of $f^{(t)}$ is the feature vector previously extracted, which is represented as $x \in \mathbb{R}^d$. Its output is a binary vector $y \in -1, 1^k$, where k is the number of music genres. ($f_l^{(t)} = 1$ stands for a vote for the chosen genre l, and -1 for a vote against this genre.).
- After T iterations, the algorithm outputs a vector-valued discriminant function :

$$g(x) = \sum_{t=1}^{T} \alpha^{(t)} f^{(t)}(x)$$

- Now we got a certain number of votes for each genres for the music track x, but **not necessarily equals to 1**. To reduce the classifier to a simple multiclass choice, we can now use the global function F :

$$F(x) = \arg\max_{l \in L} g_l(x)$$

L being the ensemble of genre studied.

It should be noted that each weak classifier only takes into account one feature. However, we can have $T > d$ (d the number of features), so the coefficients $\alpha^{(t)}$ can be used to order the features by order of relevance.

During lab-experiments, the AdaBoost classifier achieves results slightly poorer than SVM. However, when it has to be deployed, it is more commonly used than SVM, as it scales up linearly with the size of the dataset. This is an enormous advantage considering the amount of data that have to be processed.

k-Nearest Neighbors : This classifier is also sometimes used (in [11] and [2]), due to the extreme simplicity of its functioning and the relatively decent performance it reaches (although not as good as the two other classifiers detailed above).

The K-Nearest Neighbors classifier is a non-parametric classifier. It classifies the current data point according to the class of the nearest other points in the feature space. In order to do so, we need to compute the distance with the other points in an appropriate space, with the help of a distance function (also called a metric). The final classification will be computed regarding the k data points from the training set that are closest to the currently evaluated point, and the predicted label will be designated by majority vote or averaging of the labels of the k closest points.

3.5 Evaluation process

There is currently a serious issue about the absence of a common methodology to evaluate and interpret the results between the different papers on the topic of AMGC. There are many different metrics that can be used to evaluate the performance of an algorithm on a classification task, some of which are : precision, recall, accuracy, F-score, specificity, area under the ROC (Receiver Operating Characteristic) curve... The first 4 ones are explicited in Figure 7. At first sight, it is hard to establish one of these measures as a clear "best indicator" of the performance of the algorithm ([11], [2], [12] and the MIREX (www.music-ir.org/mirex) annual AMGC contest use Accuracy, [6] uses Area under ROC, [9] and [10] use "error rates" which are not clearly defined), and researchers are sometimes taking advantage of the absence of a coherent evaluation process by using the metric that highlights their work the best.

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Figure 7: Definitions of the most commonly used metrics (*tp* refers to *true positive rate, fp* refers to *false positive rate, tn* refers to *true negative rate, fn* refers to *false negative rate.*

[4] suggests that the metric to look at should be Precision, but this recommendation applies to the more general context of tagging of music (not only genre), in which the problem is easily subdivided in binary classification tasks : "should tag T be attributed to audio sample X ?", iterated over all the tags to choose from. In traditional AMGC problems, the question is rather "what is genre G that fits audio sample X the most ?", which probably explains the predominance of the use of Accuracy as the go-to metric. However, if several genres are allowed to be used to describe a certain sample (which would be more realistic [5]), or even if percentages are used (for instance, a song can be *80% jazz* and *20% rock*), the most suited metric could be very different.

On top of that, [5] and [1] remind us that music genres are organized hierarchically, and that certain misclassifications should therefore be penalized less harshly than others (Blues is closer to Rock than Classical is, for example) : yet, current evaluation metrics do not take the hierarchical structure of music genres into account.

Another issue lies in the inconsistency between the genres used in the different AMGC works, in number as well as in nature. Some studies classify samples between only 4 genres (such as in [10]), while others classify them among 10 classes [2] : as it is reminded in McKay and Fujinaga (2006) [5], the classification algorithms' performance goes down dramatically as the number of classes to choose from increases. This makes it really hard to compare the performance of models from different papers, since they usually work with different datasets and a different number of classes.

The nature of the test dataset on which the results are computed is sometimes also problematic : some studies do not specify how they separated their dataset between training and test, whether or not they used cross-validation, etc. The best methodology would obviously be to provide results with different training-test splitting strategies, providing the evolution of results with different values of k for cross-validation, etc., as done in Basili et al. (2005) [7]. However, we realize that such a procedure can be time-costly, and we can consider that performing cross-validation ($k = 10$ folds is a commonly used value) can be sufficient, as longs as researchers **clearly indicate in their paper the evaluation strategy that they used**.

A last suggestion that is taken from [1] and [6] is to provide confusion matrices with the results of a certain model : such a matrix represents the cases where an audio sample of *actual genre i* was classified into *genre j*, as shown in Figure 8.

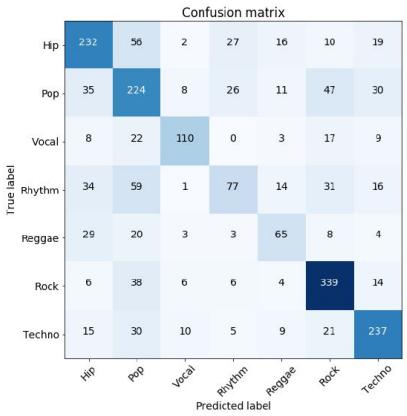

Figure 8: Confusion matrix of the results obtained by a Convolutional Neural Network model used in [6]

Such matrices help the researchers understand what genres are most problematic (usually because they overlap with one another), and can give insights on improvements to make in the feature extraction or in the classification part to better discriminate between certain genres.

3.6 Expected results

After studying the corpus of research reports, we can see a certain trend in the results obtained. Indeed, the results tend to get higher with time, as research work strongly always strongly relies on previous work. The results are usually computed using the accuracy on the testing dataset. Usually, this accuracy was as high as 60-65% in the beginning of the 2000's for 4 to 6 genres. We can note a few peaks around 80 %, in some cases (see [12]), usually in contests as the MIREX contest, a contest that challenges its competitors of Music Information Retrieval topics.

We have to note that the accuracy is also highly dependent on the number of genre studied. For instance, an algorithm of 2018 achieved an accuracy of 65% on 10 genres[6], where in 2003, we could get 87% with... only 4 genres [10].

Anyway, results are never over 90% with small data sets and quite different genres (for instance techno, rock, jazz and classical), and 70% with bigger data set and geater amounts of genres, as music genre recognition is a task that is defined by human high-level concepts. The criteria of genre belonging being quite subjective, thus introduces a human bias in the machine learning protocol.

4 EXPERIMENTATION

4.1 Motivation

After this comparative work on the existing methodologies and techniques used in AMGC, we deemed interesting to try and conduct our own experiment, in order to try and implement effective techniques used in some of the papers that we reviewed, and so as to put into perspective some of the recommendations we expressed in the first part of this project.

Yet, we must be realistic about what we can expect regarding the results of our experiment : we obviously do not have the same expertise, computational power and time resources as the professional research teams who have spent months or years mastering the topic. As a result, we must be aware that the conclusions that we will be able to draw will mostly be qualitative and focused on the general methodology of the process, rather that on the mere percentage values of our results.

4.2 Data set used

To conduct our experiments, we chose to use the **GTZAN genre collection dataset**, created by George Tzanetakis, available for download at *marsyas.info/downloads/datasets.html*. This dataset was previously used in other major papers on AMGC, notably in Tzanetakis et al. (2002) [1] and Li et al. (2003) [2].

The datasets consists of 1000 audio tracks of 30 seconds each, covering 10 different genres (Blues, Classical, Country, Disco, Hiphop, Jazz, Metal, Pop, Reggae, and Rock), with 100 tracks for each genre. The tracks are 22050Hz Mono 16-bit audio files in .wav format, recorded in various conditions, so as to provide a rather heterogeneous collection in termes of audio quality, which is meant to increase the realism of the dataset.

Although this dataset presents certain drawbacks (the low sample rate – nowadays 40 kHz is a minimum – and the varying quality of the recordings are not anymore representative of what we find on streaming platforms), its relative large size and the number/variety of genres it represents are major advantages.

4.3 Features extracted

Feature extraction was the biggest challenge that we were faced with : with very limited computational power and time, we could not afford to extract features from our dataset over

days and on dedicated servers, as it is often available to professional research team.

The first pre-processing step that we took was to segment the 30s long audio files into 3 3.5s long sub-samples located at the beginning/middle/end of the audio file, on each of which we would individually extract the desired features. The result for the whole audio file can they be obtained by a majority vote, or by averaging the features from the 3 samples composing the file for example. This sub-segmentation is used to ensure that the selected parts of the file are representative enough of the variety of the extract, while reducing the size of the data to analyze by approximately a third.

Once this segmentation was done, we were to perform the actual feature extraction process. We knew that certain frameworks had already been developed as part of previous research works the the signal processing community, but the one we initially found (the **MARSYAS** framework, available at *marsyas.info/downloads/binaries.html*, used in [1] and [2] amongst many others) was getting outdated, and most importantly was not supported by Python, which is the technology that we decided to use for this work. Luckily, we soon found the **LibROSA** Python library (*librosa.github.io/librosa*), a package designed specifically for music and audio analysis in the context of Music Information Retrieval.

LibROSA allowed us to extract most of the features that we mentioned in the 3.2 section, as well as a few others that we did not detail in this paper for conciseness reasons : Tempo, MFCCs (Mel-frequency cepstral coefficients) average and variance, Spectral Centroid average and variance, Spectral Spread average and variance, Spectral Roll-off average and variance, Spectral contrast (a feature indicative of the dynamic range of a signal, see [13] for precisions) average and variance, Spectral flatness average and variance, Zero Crossing Rate average and variance.

A hard choice to make was to discard the Wavelet Transform features mentioned in 3.2 : unlike with the previous ones, LibROSA did not provide us with an easy way to extract and compute the Wavelet Transform features. Although a **PyWavelets** (*pywavelets.readthedocs.io*) Python library exists, it is not designed specifically for Music Information Retrieval, but rather used for image processing ; on top of that, we were afraid the feature extraction process would take too long if we added these on top of all the others that we already had to compute. Still, wavelet-related features could be worth dedicating more time on in future work.

4.4 Classifier used

Regarding the experimental results of the algorithms given in the corpus, we naturally chose to implement the SVM classifier. The next choice was deciding which of the multi-class transformation approaches we should use in our own algorithm : since **scikit-learn** already comes with an implementation of SVM with the **One Against All approach** (which provides the best results according to [2]), we decided to use it.

We also tried to use dimensionality reduction techniques (feature selection by wrapping, as explained in section 3.3), but we noticed that the accuracy was greatly decreased when using the reduced feature vectors. Indeed, by removing 10% of the energy of the feature vectors (which is a general rule of thumb for dimensionality reduction) using SVD, which in our case reduced the dimensions of the feature vector from 15 to only 6, the accuracy was decreased by about 8%. As a consequence, and given the reasonable size our feature vectors even without dimensionality reduction, we chose to rule out the SVD.

We implemented the SVM with scikit-learn's **sklearn.svm.SVC**. After trying the different kernel functions that are available to us (linear, polynomial or RBF, radial basis function), we found that the RBF setting was working best for us (which is coherent with the literature, [9][14]). This indicates, as can be expected given the complexity of the task, that the problem is linearly non-separable, but non-linearly separable.

4.5 Evaluation process

The metric that we used to evaluate the performance of our classifiers was the **accuracy**, since it appears to be the criteria most suited to such a multi-class classification problem, as discussed in section 3.5.

The method used to split our data between training and testing was a 10-fold cross-validation process. When choosing the number k of folds, we must ensure that both the training set and the test set remain large enough, in order to allow sufficient training while averaging statistic fluctuations that would come with a small test set.

We also computed **confusion matrices** over our whole cross-validation results, in order to identify what genres were the hardest to discriminate for our algorithm, which can prove useful when trying to improve the feature extraction and selection parts.

4.6 Results and comparison between theory and practice

We configured the SVM to use a penalty parameter $C = 10$ and a radius influence parameter γ set on $'auto'$, which, according to scikit-learn's documentation, sets it to $\gamma = 1/n\_features$. These values were the ones maximizing the performance of our algorithm on cross-validation, while avoiding to set a C value too high or γ too low to prevent too much overfitting.

Although the performance results were varying by 1 or 2% from one run of our code to another, the average accuracy that we were reaching is **62-63%** using 10-fold cross validation.

These results were pretty coherent with the values that we could find in the literature. We knew that we would probably not end up challenging the best results that we found in papers (around 75 to 80% [2]), but given the relatively limited resources in both computation power and time that we had available, as well as our lack of previous experience of the topic, we were satisfied to see that we could implement from scratch an algorithm to carry out such a difficult task in appearance, and with a decent performance level.

Table 2 is the confusion matrix that we computed from a run of our program (rows represent actual labels, with columns being the predicted ones) : we can draw some interesting conclusions from it. For instance, our algorithm is really accurate when it comes to distinguishing Classical music from the rest, but typically has a lot of trouble with the Rock genre,

which could very well indicate that the human definition of Rock music is pretty vague.

	B.	Cl.	Co.	D.	H.	J.	M.	P.	Re.	Ro.
Blues	198	3	13	9	11	21	26	0	13	6
Classical	3	265	5	2	0	19	0	2	0	4
Country	28	3	152	18	3	30	8	12	20	26
Disco	13	2	21	172	14	8	13	22	12	23
Hiphop	13	2	12	23	169	3	14	26	30	8
Jazz	18	23	33	9	5	195	1	3	4	9
Metal	21	1	1	9	11	0	240	0	3	14
Pop	1	2	17	24	24	8	0	192	16	16
Reggae	20	2	23	17	28	6	1	24	174	5
Rock	32	4	37	43	10	17	25	19	16	97

Table 2: Confusion matrix of our own algorithm

5 CONCLUSION

5.1 Summary

In this paper, the goal was to learn a bit about the use of machine learning techniques in the field of Music information retrieval. We did so through the example of Automatic Music Genre Classification. After examining a corpus of scientific publications, we found that the main factors that influence the results of AMGC are the choice of the relevant feature (as well as their extraction) and classifiers.

The relevant features can be split in two categories : "timbral" features, calculated over short periods of time (often related with spectral characteristics), and "high-level" features, closest to the human abstraction, such as the pitch, the tempo, etc. In fact, timbral features tends to be more precise than high-level features.

The classifiers commonly used are binary classifiers, which are organized to achieve multiclass classification. The best results comes from the SVMs based classifiers (usually with a Gaussian kernel function) and from the AdaBoost classifier. However, the AdaBoost is way more useful on larger scales, as it scales up in linear complexity.

5.2 Possible future work

As emphasized in [1], AMGC should expand both in width (more general families of audio signals should be recognized by the algorithms) and in depth (within a certain family of audio signals, or within a broad genre, the algorithm should be able to classify the extracts in subfamilies and subgenres to attain the desired granularity).

Another aspect that cannot be overlooked is the recent emergence [3] [6] of deep learning algorithms : such algorithms differ from "classical" machine learning in that they use deep neural networks to process (almost) **raw data**, without the need to hand-craft and extract any feature, which appears to be the biggest difficulty in AMGC using "classical" machine learning.

The reasons that lead AMGC researchers to look into deep learning are both the tireless increase in computational power (deep learning algorithms are very computationally expensive) and the high level of abstraction of the features that deep learning algorithms manage to extract by themselves, which are *a priori* well suited to describe such a complex and artistic phenomenon as music. The impressive performance of deep learning in the field of image classification (amongst others) leads us to think that it could also perform well to classify music genres.

RÉFÉRENCES

[1] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. **IEEE Transactions on speech and audio processing**, 10(5) :293–302, 2002.

[2] Tao Li, Mitsunori Ogihara, and Qi Li. A comparative study on content-based music genre classification. In **Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval**, pages 282–289. ACM, 2003.

[3] Sam Clark, Danny Park, and Adrien Guerard. Music genre classification using machine learning techniques, 2012.

[4] Thierry Bertin-Mahieux, Douglas Eck, and Michael Mandel. Automatic tagging of audio : The state-of-the-art. In **Machine audition : Principles, algorithms and systems**, pages 334–352. IGI Global, 2011.

[5] Cory McKay and Ichiro Fujinaga. Musical genre classification : Is it worth pursuing and how can it be improved ? In **ISMIR**, pages 101–106, 2006.

[6] Hareesh Bahuleyan. Music genre classification using machine learning techniques. **arXiv preprint arXiv :1804.01149**, 2018.

[7] Roberto Basili, Alfredo Serafini, and Armando Stellato. Classification of musical genre : a machine learning approach. In **ISMIR**, 2004.

[8] Lawrence R Rabiner and Biing-Hwang Juang. **Fundamentals of speech recognition**, volume 14. PTR Prentice Hall Englewood Cliffs, 1993.

[9] Changsheng Xu, Namunu Chinthaka Maddage, and Xi Shao. Automatic music classification and summarization. **IEEE transactions on speech and audio processing**, 13(3) :441–450, 2005.

[10] Changsheng Xu, Namunu C Maddage, Xi Shao, Fang Cao, and Qi Tian. Musical genre classification using support vector machines. In **Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on**, volume 5, pages V–429. IEEE, 2003.

[11] Carlos N Silla, Alessandro L Koerich, and Celso AA Kaestner. A machine learning approach to automatic music genre classification. **Journal of the Brazilian Computer Society**, 14(3) :7–18, 2008.

[12] James Bergstra, Norman Casagrande, Dumitru Erhan, Douglas Eck, and Balázs Kégl. Aggregate features and adaboost for music classification. **Machine learning**, 65(2-3) :473–484, 2006.

[13] Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. Music type classification by spectral contrast feature. In **Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on**, volume 1, pages 113–116. IEEE, 2002.

[14] Changsheng Xu, Namunu Chinthaka Maddage, and Qi Tian. Support vector machine learning for music discrimination. In **Pacific-Rim Conference on Multimedia**, pages 928–935. Springer, 2002.