Music Genre Classification

Bollengier Maxime, Di Bella Leandro, Bussios Maxime

University of Brussels, Belgium

1 Introduction

Music classification has attracted a lot of attention in the past years thanks to the advent of audio streaming platforms[1], gathering in large databases millions of songs coming from popular artists as well as from regular people. The lack of supervision in the labelling of songs in these ever-growing databases is detrimental for their owners and users, as, for example, it complexifies music recommendation and research of new songs relatives to user's tastes, diminishing user retention on these platforms.
For this purpose, automatic content-based music classification has been thoroughly studied, aiming at finding characteristics of the songs (such as genre, author, mood, era) based on their content, such as rhythmic structure or instrumentation[2][3].

This paper will focus on genre classification, firstly by reviewing features and machine learning-based models relevant for this purpose (section 2), and then by implementing (section 3 and 4) and discussing (section 5) the most promising ones. Finally, a conclusion will be drawn in section 6.

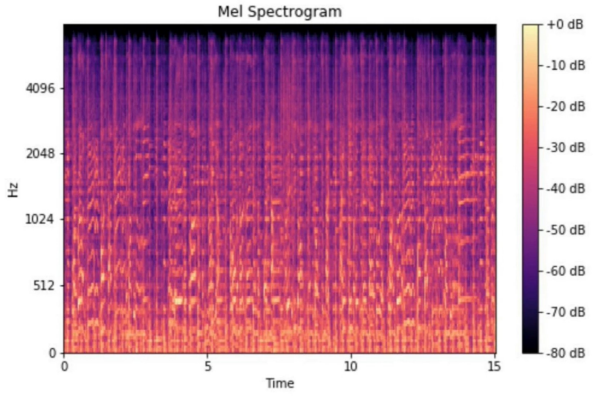

Figure 1. MFCC representation [5]

2 State of the art

2.1 Features

In order for machine learning algorithms to infer good predictions, it is critical to provide to it meaningful information from which it will be able to extract relevant correlation for the classification task.
This subsection presents some of the most used and/or useful features in music genre classification.

2.1.1 Mel-Frequency Cesptral Coefficients (MFCCs)

The MFFCs are based on the Short-Time Fourier Transform (STFT), aim to capture short-term spectral-based features in a way closer to the human ear perception than usual Fast Fourier Transform (FFT)[1], as human perception of audio signals is non-linear, humans tend to perceive better low-frequency signals than high-frequency ones, in a more or less logarithmic way[4][5]. An example of MFCC can be seen in figure1.

2.1.2 Chroma

Based on the observation that a human perceives similarities between musical pitches on different octaves[6], it is possible to decompose the pitch in two : the pitch height, describing the octave the pitch lays in and the chroma, describing the energy content of a pitch based on a 12-elements vector {C,C#,...,B}[6]. The later one enable to capture harmonic and melodic characteristics without taking into account changes in timbre and instrumentation [6]. A representation of a chroma feature is shown in figure 2.

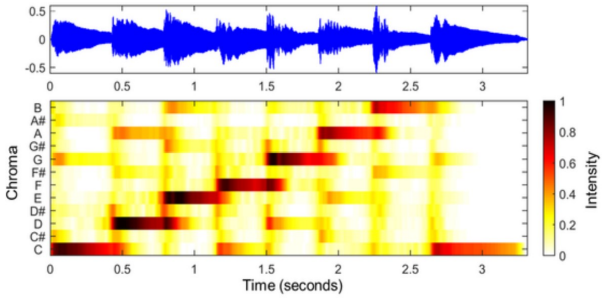

Figure 2. Audio recording of C-major key played on a piano and its corresponding chromagram [6]

2.1.3 Daubechies Wavelet Coefficients Histograms (DWCH)

By use of wavelet transform, DWCH represents the audio signal at different resolutions, enabling to acquire information on a local as well as a global scale. The coefficients histogram of each subband created is used as an indication of the waveform variations at the scale of the subband[1]. DWCHs for various genre are shown in figure3.

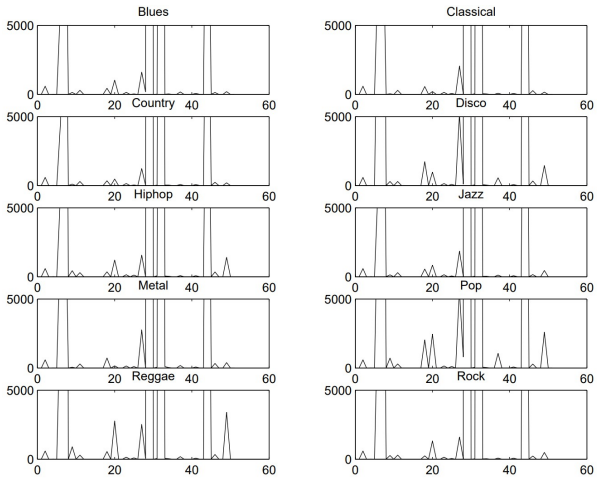

Figure 3. DWCHs of 10 different musics from different genres [1]

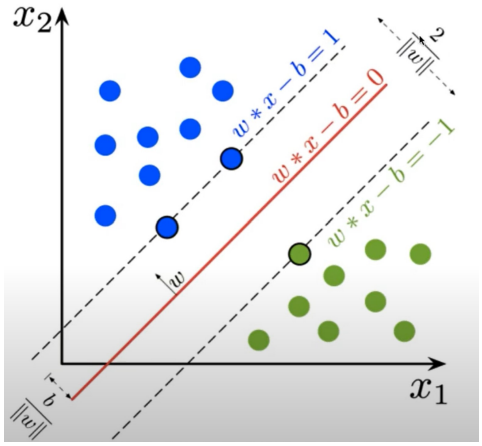

Figure 4. representation of the data, the hyperplane and the margin for a linear SVM [8]

2.2 Models

The model of a machine learning algorithm is the second cornerstone for a good classification algorithm, as it will define the strategy used.
In the following subsection, 3 models are presented, one coming from traditional machine learning, a multi-class classification model and the second and third one coming from deep machine learning.

2.2.1 Support Vector Machines (SVM)

SVM are the traditional machine learning model providing the best accuracy for music genre classification task [1][7].
SVM is a binary classification technic that aim to find a hyperplane maximizing the margin between it and the data of the 2 classes through convex optimization.
The most simple case is linear SVM, where the decision boundary (the hyperplane) is considered to be a linear function of the form

$$w \cdot x - b = 0, \tag{1}$$

where w and b are the parameters to optimize.
As it is a binary classification method, the label for the two set of data are defined as $y_i = \pm 1$, leading to the linear model

$$y_i(w \cdot x - b) \geq 1 \tag{2}$$

such that data with label "+1" are on the left side of the margin and the data with label "-1" are on its right side. A representation of the data and the hyperplane is shown in figure4.

For the convex optimization, gradient descend on the following cost function can, for example, be used.

$$J = \lambda||w||^2 + \frac{1}{n} \cdot \sum_{i=1}^{n} max(0, 1 - y_i(w \cdot x - b)), \tag{3}$$

Where the first term tries to maximize the margin $\frac{2}{||w||}$ while the second term is equal to 0 when the data is on the good side of the boundary or growth the further away the data is on the wrong side of it.
If the data are separable but not in a linear way, it is possible to use non-linear SVM by transforming the input into a higher dimension feature space using a non-linear transformation, by means of a kernel function, and perform a linear separation in the feature space[7][9].

It is possible to generalize SVM algorithms to multi class classification problems by using multiple SVM, either by using one-vs-one, one-vs-all or "successive division" methods[10][7].
One-vs-one method consists on finding a hyperplane between each pair of class, without taking the other classes into account. Leading to $\frac{L(L-1)}{2}$ SVM used, where L is the number of class[10].
One-vs-all method seeks to find the hyperplane separating the class from all the other class and repeat it for each class. Leading to L SVM used[10].
"Successive division" method separates first the data in 2 groups, containing each $\frac{L}{2}$ classes. Each subgroup is then divided in 2 new subgroups, containing each $\frac{L}{4}$ classes, until the last subgroups contain only one class[7]. Leading to ($\frac{L}{2}$-1)·2+1 SVM used. This method is valid only for a power of 2 number of classes.

2.2.2 Convolutional Neural Network (CNN)

Although traditional machine learning techniques are efficient, deep learning technique has showed that they can outperformed the ML techniques such GNN or CNN. In that sense, Music Classification has been performed for this study using a feed foward convolutional neural network (CNN). Convolutional neural network is a class of articial neural network. To be adapted to music classification, scientist needed to get features like MFCC or spectrogram which would serve as inputs for the CNN as done

by Yandre M.G. Costa Luiz [11] where the deep convolutional neural network was using spectrogram divided into patches as input with convolutional layers with 5x5 kernels and 64 filters followed by an multi layer perceptrons and fully connected layers. In [12], Safaa Allamy and Alessandro L. Koerich has showed that a 2D CNN can be problematic at some point and use a 1D CNN architecture instead where the best accuracy they could obtained were with small convolutional filters at the first layers, namely 1D ResNet CNN. Also, Tao Feng [13] has performed multiclass classification task of labelling music genres using Restricted Botlzmann machine algorithm to build a deep belief neural network. The features used in his experiments were the MFC coefficient, Mel frequency Cepstral. Finally, Hareesh Bahuleyan [14] has used deep learning approach wherein a CNN model is trained end-to-end, to predict the genre label of an audio signal and was using MEL spectrogram, time domain features and frequency domain features. The performance of the model has been assessed with the accuracy score and the F1-score. This paper as well as [11] combined deep learning techniques with traditional learning techniques such as SVM. Lastly, Sander Dieleman [15] has performed genre classification with a CNN and then use the learnt parameters to initialize a convolutinal multilayer perceptron.

2.2.3 Graph Neural Network (GNN)

Graph Neural Networks can be good model candidates for the case of music genre classification as, thanks to their strong inductive bias, they can handle better non euclidean data than more traditional deep learning models[16].

In GNN, graphs **G** are represented by their nodes $x_i \in \mathbf{V}$ (which initially contain as information the features provided to the GNN) and their edges $e_{i,j} \in \mathbf{E}$ (the links between the nodes).

GNNs update the information associated with a node by aggregating its own information and the ones from its neighbors [17]. This operation is done through a message passing layer. The outputs of the GNN are node level embeddings containing information on the other nodes features as well as the structure of the graph[18].

The number of message passing layers as well as the size of node embeddings are hyperparameters of GNNs[18].

For the specific case of classification, it is whether possible to use the embedding of a node in addition to the knowledge of the labels of (some) of the nodes in its neighborhood to predict its label (node level prediction) or to combine the embeddings of all the nodes to get a representation of the whole graph (graph level prediction)[19].

A message passing through the graph **G(V,E)** has the form[16] :

$$x_i^{(k)} = \gamma^{(k)}(x_i^{(k-1)}, F_{j\in\mathcal{N}(i)}(x_i^{(k-1)}, x_j^{k-1}, e_{i,j}), \qquad (4)$$

where F is the aggregation operation (that has to be permutation-invariant), applied on all the neighbors of i, $\mathcal{N}(i)$. It is provided to a non-linear transformation function γ for a given layer k[16].

A visual interpretation of GNNs, node embeddings and message passing layers is shown in figure5

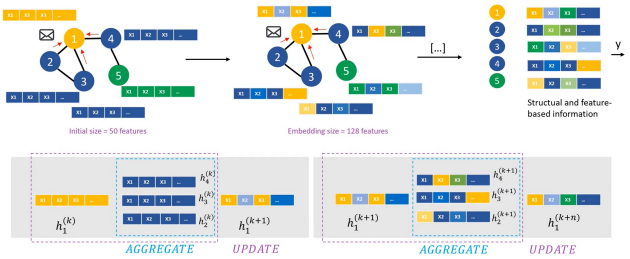

Figure 5. representation of a GNN, node embeddings and message passing layer (horizontal black arrows)[18]

3 Description of the algorithm

3.1 Dataset

The dataset used in the following experimentations is a subset of the Million Song Dataset which contains 10.000 songs. The core of the dataset is the feature analysis and metadata for 10.000 songs, provided by The Echo Nest. The dataset does not include any audio, only the derived features since the given data are uploaded as H5 files. More over, we can extract from those HDF5 file decscriptors like the track ID of the songs which will help us tho identify and keep track of the song we are working on. We can also find some MFCC-like features for each segments or normalized chroma features.

Since this dataset doesn't provide genre for any song, two other dataset has been analyzed here. The first one is the Lastfm dataset which contains informations about each song as well as a genre or multiple genres for each song. This Lastfm dataset contains over 1.000 unique genres and could be very good if we were classifying the one million song dataset. Since we are only using 10.000 song, we needed to find a dataset which contains a very low number of unique genre. The tagtraum dataset satisfies this condition. This dataset contains more or less 300.000 track IDs and their associated genre. Fortunately, this dataset only contains 15 label/genre, which fits very well our project. When making the correlation between the 10.000 song of the subset of the MSD and the 300.000 songs of the tragtraum dataset, only 3000 song could be used, either not present on one of the dataset or no genre given.

3.2 Feature extraction

As previously said, the implemented methods perform classification based on the extraction of two different sets of features.

1. The first one is called the Mel-Frequency Cepstral Coefficients (MFCC). Mel Frequency Cepstral Coefficents

(MFCCs) are a feature widely used in automatic speech and speaker recognition. They were introduced by Davis and Mermelstein in the 1980's, and have been state-of-the-art ever since. The mel frequency cepstral coefficients (MFCCs) of a signal are a small set of features (usually about 10-20) which concisely describe the overall shape of a spectral envelope. In our dataset, it is often used to describe timbre and are array of shape (300x12) (See Figure 6)

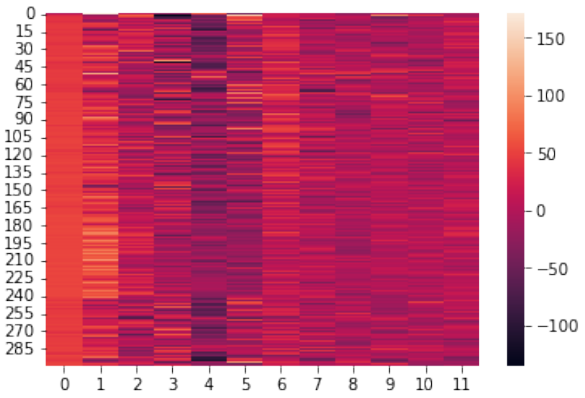

Figure 6. Mel Frequency Cepstral Coefficents of a song from the Million Song Dataset

2. The second one is called Chroma features which represents the pitches of our songs. The chromagram feature is also represented as an array of size (300,12) (See Figure 7).

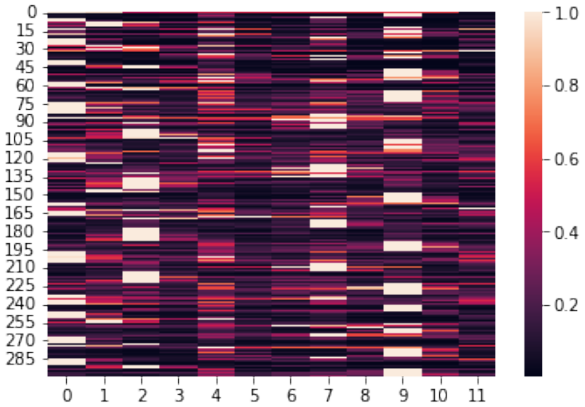

Figure 7. Chromagram of a song from the Million Song Dataset

The Daubechies Wavelet Coefficients Histograms (DWCH) feature haven't been used in this work because they weren't available on the MSD. For future improvements, one could take the three features as input. As a note, to extract the feature from the MSD dataset, the tutorial for the Million Song Dataset by Thierry Bertin-Mahieux has been followed. ((2011) Columbia University, tb2332@columbia.edu. Copyright 2011 T. Bertin-Mahieux, All Rights Reserved)

3.3 Convolutional Neural Network

Starting with the main deep learning techniques used on this project, the convolutional neural network (CNN). The deep neural network contains repeated use of convolutional layers with 8 to 32 filters followed by max-poolings layers. At the end of the convolutional layer, a batch normalization is done. The activation function used is a ReLU function. Lastly, the kernel size is a 3x3 with stride 1 and padding 1. The pool layers were achieved with window size 2x2 and a stride 2. Those convolutional layer is followed by a multilayer perceptron to achieved the classification. The MLP is build with a fully connected layer followed by a softmax activation function. The architecture is illustrated in Figure 8.

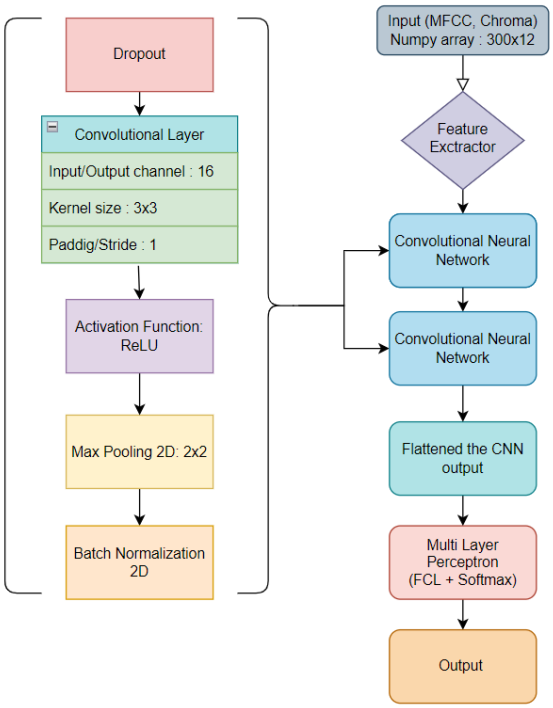

Figure 8. Architecture of the Convolutional Neural Network

Now that the architecture is done, the model needs to handle the input and the feature given. As explained in the previous section, the input used are the two features called Chroma and MFCC which are numpy arrays. Those two numpy arrays can be send in the CNN as 2D arrays or 1D arrays. In order to find the best architecture for this study, we are going to send the input as 1D array using a 1D CNN and as 2D array using a 2D CNN.
Multiple experiments has been done on those CNN. The first one consists of testing which feature delivers the best performances: MFCC, CHROMA or a combination of

both. The second experiments consisted of varying the number of convolutional layer. Finally, some other hyper-parameters were studied to find the best architecture such as the number of hidden layer or the dropout ratio. All the experiments except the first one has been done only on the MFCC features.

3.4 Graph Convolutional Neural Network

Graph Convolutional Neural Networks are well known for processing non-Euclian data. They have proven their efficiency in many domains for their ability to capture long range contextual information [19]. For this reason, we were interested in how it was possible to apply this type of network to music classification. To do so, we experimented with two ways of representing features.

3.4.1 Graph representation

First representation The first representation consists in building a graph per music where each node contains the mfcc features for a given time. The graph has 12 nodes where each node contains a 128-dimentional features vector coming from the transformation of the mfcc by a fully connected layer at a particular time (eq5). Each node is connected to all other nodes of the graph. The purpose of this representation is to allow each node to have access to the information of the other nodes of the graph and thus capture all the information unlike a classical convolution network which has a smaller receptive field accros the layers [17].

$$h_i = U^i x^i + u^i \tag{5}$$

were $h_i \in \mathbb{R}^{128}$ represent the input node features.$U^i \in \mathbb{R}^{128\times300}$ and $u^i \in \mathbb{R}^{128}$ are the parameters for the linear transformation applied to the mfcc feature. $x^i \in \mathbb{R}^{300}$ represent the original i^{th} mfcc feature for $i = 0, ..11$.The edge feature is simply $e_{i,j} = 1$.

Second representation For the second representation, the idea was to try to exploit the contextual information between samples. For this, the graph is built with 100 nodes where each node represents a 1024-dimentional vector coming from the transformation of the mfcc features for a sample (eq 6). In other words, each node corresponds to a music. To build the edges, the idea is to exploit the similarity between the features. The graph will be fully connected with edges features representing on the cosine similarity between the mfcc features of the song.

$$h_i = U^i x^i + u^i \tag{6}$$

In this case the dimension of x^i is $\mathbb{R}^{(300\cdot12)\times1}$ because all the information from the song is assigned to one node. Then, $h_i \in \mathbb{R}^{1024}$,$U^i \in \mathbb{R}^{1024\times(300\cdot12)}$ and $u^i \in \mathbb{R}^{1024}$. The edges features are represented by :

$$e_{i,j} = similarity(h_i, hj) \tag{7}$$

where similarity is the cosine similarity defined by :

$$similarity(h_i, hj) = \frac{\langle h_i\ ,\ hj\rangle}{\|h_i\| \times \|hj\|} \tag{8}$$

3.5 Graph convolution

To build the network, we chose to use graphSage [20] for its ability to aggregate features from neighboring nodes. In addition, given the best empirical results, GrapheSage also allows a great flexibility in the face of changing topology of the graph and will be very useful to set up an inference method. As described in the equation 9, the graphSage layer consists in using an aggregation function to aggregate the features coming from the neighboring nodes $\mathcal{N}(i)$. Once aggregated, the neighboring features will be concatenated and passed into a non-linear activation function σ and then normalized [20].

$$h_{\mathcal{N}(i)}^{(l+1)} = \text{aggregate}\left(\{h_j^l, \forall j \in \mathcal{N}(i)\}\right) \tag{9}$$

$$h_i^{(l+1)} = \sigma\left(W \cdot \text{concat}(h_i^l, h_{\mathcal{N}(i)}^{l+1})\right)$$

$$h_i^{(l+1)} = \text{norm}(h_i^{(l+1)})$$

Empirically, the aggregation function giving the best results is the "max pooling" and we use the activation function ReLu. We tested the "mean" aggregation giving results very close to "maxpool" and the "lstm" aggregator which adds a lot of parameters and creates overfitting very quickly during training.

3.5.1 Graph Network

First approach : Graph prediction The first approach is related to the first way of representing features. For this approach, as explained in the section3.4.1 , each song is represented by a graph of 12 nodes with 128-dimensional vectors coming from the mfcc features. The dimension of the nodes being small, we decided to maintain it through layers. So we pass the graph in 4 GraphSage layers with residual connection and batch normalization. After, all the node features are summed and passed in a multilayer perceptron of 2 layers and ending with a softmax activation to get the probability of belonging to a class. A scheme of this approach can be seen in figure9

Second approach : Node prediction In the second approach, each graph has 100 fully connected nodes representing the 1024-features vector of the mfcc of each song. The edge features between each node represent the similarity between the mfcc features.At the beginning, we apply an edge sampling on the neighboring nodes based on the highest similarity (the procedure is detailed in section 5). As the size of each node is large, we have chosen to decrease the size of the nodes features by two at each layer. So we have 4 GraphSage layers with residual connection and batch normalization. In output we have the graph

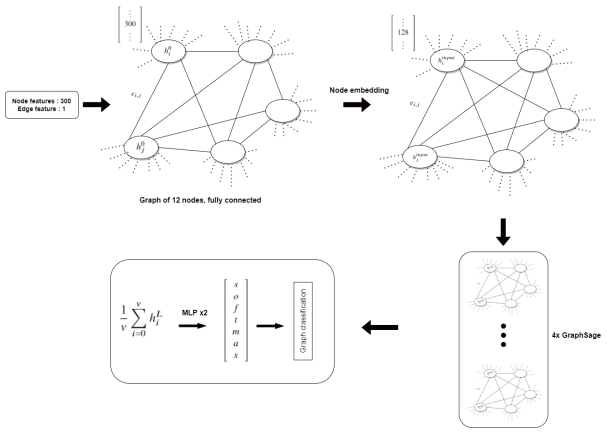

Figure 9. Scheme graph prediction approach

with node features dimensions equal to 64. After each node features is passed in 2 multilayers perceptrons with ReLu activation and a softmax layer at the end. For inference, the idea is to use samples used for training to create a "context graph". At each step, the validation sample will connect to the context graph thanks to the similarity it has with each of the nodes as edge features. Then we pass the graph in the networks and we take the classification result only for the node corresponding to the evaluation. This inference technique is possible thanks to GraphSage because the convolution is done in an isotropic way and therefore independent from the graph node reparametrization [17]. A scheme of the approach is shown in figure10.

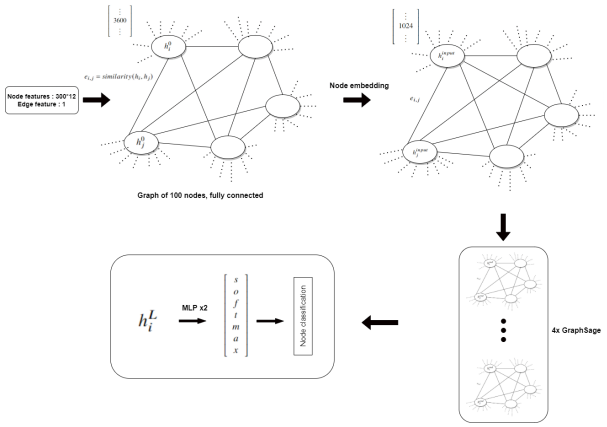

Figure 10. Scheme node prediction approach

4 Training procedure

The dataset is split in 3 for training, test and validation. 60% of the dataset is used for training (1806 samples), 20% for testing (602 samples) and 20% for validation (602 samples). Cross-fold validation is used by dividing the dataset in 5 subsets.
The algorithms have been trained with a unlimited number of epochs (max = 1000) and early stop, when the validation loss doesn't change during 10 epochs. The initial learning rate is set to 5e-4, and it is divided by 2 each time no improvement is observed for 10 epochs. The batch size is set to 10 for most of the tests, but some have been performed with batch sizes of 100 or 1000. Tests with dropout of 0, 0.2 and 0.5 have been done.
The Adam optimizer is used.
Cross entropy has been used as the loss function, the dataset being unbalanced (the number of samples per class is shown in figure11, a weighted version is actually implemented :

$$WCE(p,\hat{p}) = -(\beta \cdot p \cdot log(\hat{p}) + (1-p) log(1-\hat{p})), \quad (10)$$

where p is the ground truth, p$\in$ {0,1} and $\hat{p}$ is the sigmoid function $\frac{1}{1+e^{-x}}$ used to give the predictions. [21].
The metrics used to asses the performance of the different models is the balanced accuracy as the number of class was judged big and they are imbalanced :

$$BalancedAccuracy = \sum_{i=1}^{N} \frac{Recall(i)}{N}, \quad (11)$$

$$Recall(i) = \frac{TP}{TP + FN}, \quad (12)$$

where N is the number of class, TP is the number of true positive and FN the number of false negative for the class i.

Figure 11. Number of samples per class in the dataset

All the models were trained on Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz 2.11 GHz.

5 Experiments and Results

5.1 Conv 2D

1. The first experiment was focus on finding which one of the three features (MFCC, Chroma and a combination of both) was the best one to classify songs. In order to do it, we used the CNN architecture in Figure 8 with 3 convolutional layers, each of them having 16 input/output channel. We set the number of epoch at 20 and a batch

Feature Input	test accuracy
Chroma	65.8899 ± 0.8915
MFCC	66.7155 ± 0.9008
Chroma + MFCC	66.2939 ± 0.7542

Table 1. Input feature accuracy with 3 ConvLayer

size of 10 : As the experiment suggest, all three of the feature can be used as the three balanced accuracy for the same model are close to each other but a light advantage for the MFCC. The same experiment in table 2 has been done with 2 convolutional layers only and the same behaviour can be observed. For all further experiments,

Feature Input	test accuracy
Chroma	65.2289 ± 0.2144
MFCC	65.4333 ± 0.7906
Chroma + MFCC	65.2407 ± 0.2356

Table 2. Input feature accuracy with 2 ConvLayer

we decided to go with the MFCC feature which has still the best accuracy score.

2. The second experiment consisted of choosing how many convolutional layer was needed for our CNN. As the previous table suggested it, we can see on the following table 3 that the CNN having three convolutional layer has the best performances.

# of Conv Layer	test accuracy
1 layer	64.9289 ± 0.127
2 layer	65.4333 ± 0.7906
3 layer	66.7155 ± 0.9008

Table 3. Accuracy score for different number of convolutional layers

3. The next experiment focused on the number of input/output channel in the 3 convolutional layer. The table 4 shows that 8 input/output channel gives the best performances.

# of input/output channel	test accuracy
8	66.9087 ± 0.7293
16	66.7155 ± 0.9008
32	66.4169 ± 1.0750

Table 4. Accuracy score for different number of input/output channel channels

4. The last experiment focused on the size of the batch size. The table 5 shows a batch size bigger than 10 and smaller than 1000 suits better our application.

As a conclusion, the best architecture we could find for this application is a CNN with 3 convolutional layer, each of them having 8 input/output channel with MFCC feature as inputs. The training is done on 20 epochs with a batch size of 100.

Size of the batch size	test accuracy
10	66.7155 ± 0.9008
100	67.3827 ± 0.6886
1000	65.4010 ± 0.7282

Table 5. Accuracy score for different batch sizes

5.2 Conv 1D

Similarly to the previous experiment, a first set of tests has been done with the same hyperparameters for the 3 kind of features, the tests with only MFCC shown the best results.
The number of layers and the dropout has then been changed to see its influence. The results being very similar to the previous case, no further study have been done. The results are shown in Table6

model	Layers	Dropout	test accuracy
chroma	3	0	65.6440 ± 0.8357
MFCC	2	0	65.486 ± 0.703
MFCC	3	0	65.6792 ± 0.637
MFCC	3	0.2	65.4859 ± 0.703
Chroma + MFCC	3	0	65.4687 ± 0.712

Table 6. Accuracy score for different feature, number of layer and dropout for the Conv1D model (number of channel = 16, batch size = 10)

5.3 Graph models

5.3.1 First approach

As shown in the Table 7, the mfcc features were chosen because they seem to provide better performances, so we chose to use only these features for the experiments. We

model	test accuracy
graph_ chroma	63.2 ± 0.83
graph_ chroma+mfcc	64.6 ± 0.56
graph_ mfcc	65.2 ± 1.01

Table 7. graph model for various input features

also wanted to study the impact of the number of GraphSage layers in the network (Table 8). We conclude that 3 layers are sufficient. Above, the results are slightly less good and above, we have overfitting. This first approach gives results in the same order of magnitude as the other models of the project but with much less parameters.

5.3.2 Second approach

As explained above. The objective of this approach is to use the similarity between samples. It has been observed that the sensitivity of the model is based on the way the edges are represented. As shown in the table 9, a first experiment consists in simply connecting all the nodes together. The second experiment consists in selecting the topK (K=30) neighboring node based on the highest similarity. This step takes place only at the first layer. For the third experiment, we studied the histogram of similarity as shown on the figure 12. Based on the histogram

model	test accuracy	#params
2 layers	65.04 ± 0.32	150k
3 layers	65.2 ± 1.01	200k
4 layers	65.1 ± 0.97	250k

Table 8. model performance for various number of GraphSage layers

and a study of the similarity across classes, we defined a threshold (=0.11) that allows sampling the edges. In other words, if the similarity between two nodes is smaller than the threshold, the connection between the two nodes is cut. This last experiment is currently giving us the best results. In order to see the maximum performance of the

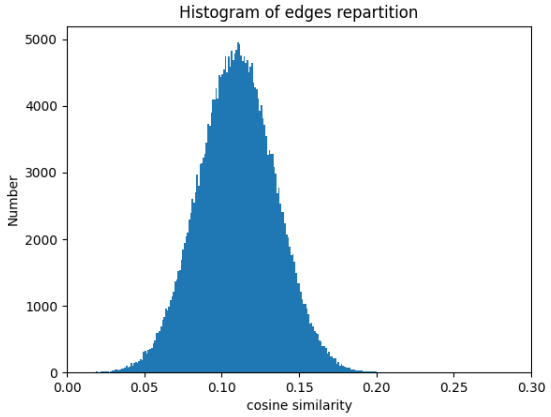

Figure 12. Edges similarity histogram between all classes

edge representation, we tried to connect the nodes of the graph according to their classes. The similarity is thus guaranteed but this method is not feasible in real scenario but the experience encourages us to improve the edges representation.

model	test accuracy	#params
fully connected	65.09 ± 0.55	5.8m
Topk (k=30)	66.28 ± 0.64	5.8m
tresh = 0.11	67.33 ± 0.58	5.8m
class connection	72.33 ± 0.87	5.8m

Table 9. graph model for edges connection

5.3.3 Further work

The results, encourage us to continue searching in the graph representation. Specifically, it could be observed that the representation of edges impacts the model. Moreover, edges are only used to define connections but edge features are not explicitly used. By including edges features in the convolution, we expect to weight the convolution and thus include anisotropy [17]. Another idea could be to update the edge features across layers to allow the network to progressively determine the similarity between nodes by itself. Another direction of research could be to use other types of convolutional layer graphs and also to study the impact of the density of classes in the context graph.

6 Conclusion

Our models, in terms of accuracy, show similar results than the literature for the same dataset. It could be interesting to use other dataset (such as GTZAN) containing .wav files of the songs in order to dig more into the variety of features extractable from audio files, enabling for example to use DWCHs.
The code can be found here https://github.com/mbolleng2303/Music$_{C}lassification.git$

7 Contribution

- **Maxime Bollengier** : CNN theory and code, GNN theory and code, main code
- **Leandro Di Bella** : States of the art theory, Feature extraction code, CNN2D code and theory
- **Maxime Bussios** : States of the art theory, CNN1D code and theory, GNN theory

References

[1] Tao Li, Mitsunori Ogihara, and Qi Li. A comparative study on content-based music genre classification. (January):282, 2003. doi:10.1145/860435.860487.

[2] Michal Genussov and Israel Cohen. Musical genre classification of audio signals using geometric methods. *European Signal Processing Conference*, 10(5): 497–501, 2010. ISSN 22195491.

[3] Peter Ahrendt. Music genre classification systems -a computational approach. *Ph.D. dissertation, Technical University, Denmark*, 01 2006.

[4] Lav R. Varshney and John Z. Sun. Why do we perceive logarithmically? *Significance*, 10(1):28–31, feb 2013. ISSN 17409705. doi:10.1111/J.1740-9713.2013.00636.X.

[5] Valerio Velardo. Mel spectrograms explained easily. URL `https://www.youtube.com/watch?v=9GHCiiDLHQ4`.

[6] Manasi Kattel, Araju Nepal, Ayush Kumar Shah, and Dev Chandra Shrestha. Chroma Feature Extraction. *Encyclopedia of GIS*, (January):1–9, 2019.

[7] Changsheng Xu, Namunu C. Maddage, Xi Shao, Fang Cao, and Qi Tian. Musical genre classification using support vector machines. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 5(March):429–432, 2003. ISSN 15206149. doi:10.1109/icassp.2003.1199998.

[8] Patrick "Python Engineer". Svm (support vector machine) in python - machine learning from scratch 07 - python tutorial. URL https://www.youtube.com/watch?v=UX0f9BNBcsY.

[9] R Thiruvengatanadhan. Music Genre Classification using SVM. *International Research Journal of Engineering and Technology*, pages 2008–2010, 2018. ISSN 2395-0072. URL www.irjet.net.

[10] Multiclass Classification Using SVM. URL https://www.analyticsvidhya.com/blog/2021/05/multiclass-classification-using-svm/.

[11] Yandre M.G. Costa, Luiz S. Oliveira, and Carlos N. Silla. An evaluation of Convolutional Neural Networks for music classification using spectrograms. *Applied Soft Computing*, 52:28–38, mar 2017. ISSN 1568-4946. doi:10.1016/J.ASOC.2016.12.024.

[12] Safaa Allamy and Alessandro Lameiras Koerich. 1D CNN Architectures for Music Genre Classification. *2021 IEEE Symposium Series on Computational Intelligence, SSCI 2021 - Proceedings*, may 2021. doi:10.48550/arxiv.2105.07302. URL https://arxiv.org/abs/2105.07302v1.

[13] Tao Feng. Deep learning for music genre classification. 2014.

[14] Ali Karatana and Oktay Yildiz. Music Genre Classification using Machine Learning Techniques. pages 1–4, apr 2018. doi:10.48550/arxiv.1804.01149. URL https://arxiv.org/abs/1804.01149v1.

[15] Sander Dieleman, Philemon Brakel, and Benjamin Schrauwen. Audio-based music classification with a pretrained convolutional network. pages 669–674, 01 2011.

[16] Shubham Dokania and Vasudev Singh. Graph Representation learning for Audio Music genre Classification.

[17] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks, 2022.

[18] "DeepFindr. Understanding graph neural networks | part 2/3 - gnns and it's variants. URL https://www.youtube.com/watch?v=ABCGCf8cJOE.

[19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2018. URL https://arxiv.org/abs/1810.00826.

[20] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2017. URL https://arxiv.org/abs/1706.02216.

[21] Loss Functions For Segmentation. URL https://lars76.github.io/2018/09/27/loss-functions-for-segmentation.html#:~:text=Weighted$%$20cross$%$20entropy$%$20$%$28WCE$%$29$%$20is$%$20a$%$20variant$%$20of,classification$%$2C$%$20it$%$20is$%$20mostly$%$20used$%$20for$%$20multiple$%$20classes.